.gitignore projeden silindi ve tekrar git hesabı sodyaları güncellendi.

Entityler:

Entity/Book.php içeriği

<?php

namespace App\Entity;

use ApiPlatform\Metadata\ApiResource;

use App\Repository\BookRepository;

use Doctrine\Common\Collections\ArrayCollection;

use Doctrine\Common\Collections\Collection;

use Doctrine\DBAL\Types\Types;

use Doctrine\ORM\Event\LifecycleEventArgs;

use Doctrine\ORM\Mapping as ORM;

use function PHPSTORM\_META\type;

#[ORM\Entity(repositoryClass: BookRepository::class)]

#[ApiResource()]

class Book

{

    #[ORM\Id]

    #[ORM\GeneratedValue]

    #[ORM\Column]

    private ?int $id = null;

    #[ORM\Column(length: 255)]

    private ?string $ad = null;

    #[ORM\Column(length: 255)]

    private ?string $yazar = null;

    #[ORM\Column(type:Types::SMALLINT)]

    private ?int $sayfaSayisi = null;

    #[ORM\Column(type: Types::BIGINT)]

    private ?string $ISBN = null;

    #[ORM\ManyToMany(targetEntity: Borrowedbook::class, mappedBy: 'book')]

    private Collection $borrowedbooks;

    #[ORM\Column]

    private ?int $Sayi = null;

    public function \_\_construct()

    {

        $this->borrowedbooks = new ArrayCollection();

    }

    public function getId(): ?int

    {

        return $this->id;

    }

    public function getAd(): ?string

    {

        return $this->ad;

    }

    public function setAd(string $ad): static

    {

        $this->ad = $ad;

        return $this;

    }

    public function getYazar(): ?string

    {

        return $this->yazar;

    }

    public function setYazar(string $yazar): static

    {

        $this->yazar = $yazar;

        return $this;

    }

    public function getSayfaSayisi(): ?int

    {

        return $this->sayfaSayisi;

    }

    public function setSayfaSayisi(int $sayfaSayisi): static

    {

        $this->sayfaSayisi = $sayfaSayisi;

        return $this;

    }

    public function getISBN(): ?string

    {

        return $this->ISBN;

    }

    public function setISBN(string $ISBN): static

    {

        $this->ISBN = $ISBN;

        return $this;

    }

    /\*\*

     \* @return Collection<int, Borrowedbook>

     \*/

    public function getBorrowedbooks(): Collection

    {

        return $this->borrowedbooks;

    }

    public function addBorrowedbook(Borrowedbook $borrowedbook): static

    {

        if (!$this->borrowedbooks->contains($borrowedbook)) {

            $this->borrowedbooks->add($borrowedbook);

            $borrowedbook->addBook($this);

        }

        return $this;

    }

    public function removeBorrowedbook(Borrowedbook $borrowedbook): static

    {

        if ($this->borrowedbooks->removeElement($borrowedbook)) {

            $borrowedbook->removeBook($this);

        }

        return $this;

    }

    public function getSayi(): ?int

    {

        return $this->Sayi;

    }

    public function setSayi(int $Sayi): static

    {

        $this->Sayi = $Sayi;

        return $this;

    }

    public function decreaseQuantity()

    {

        $this->Sayi -= 1;

    }

}

Borrowedbook.php

<?php

namespace App\Entity;

use ApiPlatform\Metadata\ApiResource;

use App\Repository\BorrowedbookRepository;

use Doctrine\Common\Collections\ArrayCollection;

use Doctrine\Common\Collections\Collection;

use Doctrine\DBAL\Types\Types;

use Doctrine\ORM\Mapping as ORM;

#[ORM\Entity(repositoryClass: BorrowedbookRepository::class)]

#[ApiResource()]

class Borrowedbook

{

    #[ORM\Id]

    #[ORM\GeneratedValue]

    #[ORM\Column]

    private ?int $id = null;

    #[ORM\ManyToMany(targetEntity: Book::class, inversedBy: 'borrowedbooks')]

    private Collection $book;

    #[ORM\ManyToOne(inversedBy: 'borrowedbooks')]

    private ?User $user = null;

    #[ORM\Column(type: Types::DATE\_MUTABLE)]

    private ?\DateTimeInterface $alisTarihi = null;

    #[ORM\Column(type: Types::DATE\_MUTABLE, nullable: true)]

    private ?\DateTimeInterface $iadeTarihi = null;

    #[ORM\Column(type: Types::DECIMAL, precision: 10, scale: '0')]

    private ?string $BarkodNo = null;

    public function \_\_construct()

    {

        $this->book = new ArrayCollection();

    }

    public function getId(): ?int

    {

        return $this->id;

    }

    /\*\*

     \* @return Collection<int, Book>

     \*/

    public function getBook(): Collection

    {

        return $this->book;

    }

    public function addBook(Book $book): static

    {

        if (!$this->book->contains($book)) {

            $this->book->add($book);

        }

        return $this;

    }

    public function removeBook(Book $book): static

    {

        $this->book->removeElement($book);

        return $this;

    }

    public function getUser(): ?User

    {

        return $this->user;

    }

    public function setUser(?User $user): static

    {

        $this->user = $user;

        return $this;

    }

    public function getAlisTarihi(): ?\DateTimeInterface

    {

        return $this->alisTarihi;

    }

    public function setAlisTarihi(\DateTimeInterface $alisTarihi): static

    {

        $this->alisTarihi = $alisTarihi;

        return $this;

    }

    public function getIadeTarihi(): ?\DateTimeInterface

    {

        return $this->iadeTarihi;

    }

    public function setIadeTarihi(?\DateTimeInterface $iadeTarihi): static

    {

        $this->iadeTarihi = $iadeTarihi;

        return $this;

    }

    public function getBarkodNo(): ?string

    {

        return $this->BarkodNo;

    }

    public function setBarkodNo(string $BarkodNo): static

    {

        $this->BarkodNo = $BarkodNo;

        return $this;

    }

}

User.php

<?php

namespace App\Entity;

use ApiPlatform\Metadata\ApiResource;

use App\Repository\UserRepository;

use Doctrine\Common\Collections\ArrayCollection;

use Doctrine\Common\Collections\Collection;

use Doctrine\ORM\Mapping as ORM;

use Symfony\Bridge\Doctrine\Validator\Constraints\UniqueEntity;

use Symfony\Component\PasswordHasher\Hasher\UserPasswordHasherInterface;

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

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

#[ORM\Entity(repositoryClass: UserRepository::class)]

#[UniqueEntity(fields: ['email'], message: 'There is already an account with this email')]

#[ApiResource()]

class User implements UserInterface, PasswordAuthenticatedUserInterface

{

    #[ORM\Id]

    #[ORM\GeneratedValue]

    #[ORM\Column]

    private ?int $id = null;

    #[ORM\Column(length: 180, unique: true)]

    private ?string $email = null;

    #[ORM\Column]

    private array $roles = [];

    /\*\*

     \* @var string The hashed password

     \*/

    #[ORM\Column]

    private ?string $password = null;

    #[ORM\Column(nullable: true)]

    private ?string $ad = null;

    #[ORM\Column(nullable: true)]

    private ?string $soyad = null;

    #[ORM\Column(nullable: true)]

    private ?string $address = null;

    #[ORM\Column(nullable: true)]

    private ?string $telefon = null;

    public function getId(): ?int

    {

        return $this->id;

    }

    public function getEmail(): ?string

    {

        return $this->email;

    }

    public function setEmail(string $email): static

    {

        $this->email = $email;

        return $this;

    }

    /\*\*

     \* A visual identifier that represents this user.

     \*

     \* @see UserInterface

     \*/

    public function getUserIdentifier(): string

    {

        return (string) $this->email;

    }

    /\*\*

     \* @see UserInterface

     \*/

    public function getRoles(): array

    {

        $roles = $this->roles;

        // guarantee every user at least has ROLE\_USER

        $roles[] = 'ROLE\_USER';

        return array\_unique($roles);

    }

    public function setRoles(array $roles): static

    {

        $this->roles = $roles;

        return $this;

    }

    /\*\*

     \* @see PasswordAuthenticatedUserInterface

     \*/

    public function getPassword(): string

    {

        return $this->password;

    }

    public function setPassword(string $password): static

    {

        $this->password = $password;

        return $this;

    }

    private ?string $plainPassword = null;

    #[ORM\OneToMany(mappedBy: 'user', targetEntity: Borrowedbook::class)]

    private Collection $borrowedbooks;

    public function \_\_construct()

    {

        $this->borrowedbooks = new ArrayCollection();

    }

    // ...

    public function getPlainPassword(): ?string

    {

        return $this->plainPassword;

    }

    public function setPlainPassword(?string $plainPassword): self

    {

        $this->plainPassword = $plainPassword;

        return $this;

    }

    public function getAd(): ?string

    {

        return $this->ad;

    }

    public function setAd(?string $ad): static

    {

        $this->ad = $ad;

        return $this;

    }

    public function getSoyad(): ?string

    {

        return $this->soyad;

    }

    public function setSoyad(?string $soyad): static

    {

        $this->soyad = $soyad;

        return $this;

    }

    public function getAddress(): ?string

    {

        return $this->address;

    }

    public function setAddress(?string $address): static

    {

        $this->address = $address;

        return $this;

    }

    public function getTelefon(): ?string

    {

        return $this->telefon;

    }

    public function setTelefon(?string $telefon): static

    {

        $this->telefon = $telefon;

        return $this;

    }

    /\*\*

     \* @see UserInterface

     \*/

    public function eraseCredentials(): void

    {

        // If you store any temporary, sensitive data on the user, clear it here

        // $this->plainPassword = null;

    }

    /\*\*

     \* @return Collection<int, Borrowedbook>

     \*/

    public function getBorrowedbooks(): Collection

    {

        return $this->borrowedbooks;

    }

    public function addBorrowedbook(Borrowedbook $borrowedbook): static

    {

        if (!$this->borrowedbooks->contains($borrowedbook)) {

            $this->borrowedbooks->add($borrowedbook);

            $borrowedbook->setUser($this);

        }

        return $this;

    }

    public function removeBorrowedbook(Borrowedbook $borrowedbook): static

    {

        if ($this->borrowedbooks->removeElement($borrowedbook)) {

            // set the owning side to null (unless already changed)

            if ($borrowedbook->getUser() === $this) {

                $borrowedbook->setUser(null);

            }

        }

        return $this;

    }

}

Form elemanları

Form/BookType.php:

<?php

// src/Form/BookType.php

namespace App\Form;

use App\Entity\Book;

use Symfony\Component\Form\AbstractType;

use Symfony\Component\Form\Extension\Core\Type\IntegerType;

use Symfony\Component\Form\Extension\Core\Type\SubmitType;

use Symfony\Component\Form\Extension\Core\Type\TextType;

use Symfony\Component\Form\FormBuilderInterface;

use Symfony\Component\OptionsResolver\OptionsResolver;

class BookType extends AbstractType

{

    public function buildForm(FormBuilderInterface $builder, array $options)

    {

        $builder

            ->add('ad', TextType::class, [

                'label' => 'Kitap Adı',

            ])

            ->add('yazar', TextType::class, [

                'label' => 'Yazar',

            ])

            ->add('sayfaSayisi', IntegerType::class, [

                'label' => 'Sayfa Sayısı',

            ])

            ->add('ISBN', IntegerType::class, [

                'label' => 'ISBN',

            ])

            ->add('sayi', IntegerType::class, [

                'label' => 'Sayi',

            ])

            ->add('save', SubmitType::class, [

                'label' => 'Kaydet',

            ]);

    }

    public function configureOptions(OptionsResolver $resolver)

    {

        $resolver->setDefaults([

            'data\_class' => Book::class,

        ]);

    }

}

Form/BorrowedbookType.php

<?php

namespace App\Form;

use App\Entity\Book;

use App\Entity\Borrowedbook;

use App\Entity\User;

use Symfony\Bridge\Doctrine\Form\Type\EntityType;

use Symfony\Component\Form\AbstractType;

use Symfony\Component\Form\FormBuilderInterface;

use Symfony\Component\OptionsResolver\OptionsResolver;

class BorrowedbookType extends AbstractType

{

    public function buildForm(FormBuilderInterface $builder, array $options): void

    {

        $builder

            ->add('alisTarihi')

            ->add('iadeTarihi')

            ->add('BarkodNo')

            ->add('book', EntityType::class, [

                'class' => Book::class,

'choice\_label' => 'ad',

'multiple' => true,

            ])

            ->add('user', EntityType::class, [

                'class' => User::class,

'choice\_label' => 'ad',

            ])

        ;

    }

    public function configureOptions(OptionsResolver $resolver): void

    {

        $resolver->setDefaults([

            'data\_class' => Borrowedbook::class,

        ]);

    }

}

Form/ChangePasswordProfileType.php

<?php

// src/Form/ChangePasswordProfileType.php

namespace App\Form;

use Symfony\Component\Form\AbstractType;

use Symfony\Component\Form\Extension\Core\Type\PasswordType;

use Symfony\Component\Form\Extension\Core\Type\RepeatedType;

use Symfony\Component\Form\FormBuilderInterface;

use Symfony\Component\OptionsResolver\OptionsResolver;

use Symfony\Component\Validator\Constraints\NotBlank;

use Symfony\Component\Validator\Constraints\Length;

class ChangePasswordProfileType extends AbstractType

{

    public function buildForm(FormBuilderInterface $builder, array $options)

    {

        $builder

            ->add('oldPassword', PasswordType::class, [

                'label' => 'Current Password',

                'mapped' => false,

                'constraints' => [

                    new NotBlank(['message' => 'Please enter your current password.']),

                ],

                'attr' => [

                    'name' => 'oldPassword', // Bu satırı ekleyin

                ],

            ])

            ->add('newPassword', RepeatedType::class, [

                'type' => PasswordType::class,

                'invalid\_message' => 'The password fields must match.',

                'options' => ['attr' => ['class' => 'password-field']],

                'required' => true,

                'first\_options' => [

                    'label' => 'New Password',

                    'attr' => [

                        'name' => 'newPassword', // Bu satırı ekleyin

                    ],

                ],

                'second\_options' => [

                    'label' => 'Repeat New Password',

                    'attr' => [

                        'name' => 'newPassword\_confirmation', // Bu satırı ekleyin

                    ],

                ],

                'constraints' => [

                    new NotBlank(['message' => 'Please enter a new password.']),

                    new Length([

                        'min' => 6,

                        'minMessage' => 'Your new password should be at least {{ limit }} characters',

                        // Symfony'nin güvenlik nedenleriyle izin verdiği maksimum uzunluk

                        'max' => 4096,

                    ]),

                ],

            ]);

    }

    public function configureOptions(OptionsResolver $resolver)

    {

        $resolver->setDefaults([

            // Configure your form options here

        ]);

    }

}

Form/RegistrationFormType.php

<?php

namespace App\Form;

use App\Entity\User;

use Symfony\Component\Form\AbstractType;

use Symfony\Component\Form\Extension\Core\Type\CheckboxType;

use Symfony\Component\Form\Extension\Core\Type\ChoiceType;

use Symfony\Component\Form\Extension\Core\Type\PasswordType;

use Symfony\Component\Form\Extension\Core\Type\TextType;

use Symfony\Component\Form\FormBuilderInterface;

use Symfony\Component\OptionsResolver\OptionsResolver;

use Symfony\Component\Validator\Constraints\IsTrue;

use Symfony\Component\Validator\Constraints\Length;

use Symfony\Component\Validator\Constraints\NotBlank;

class RegistrationFormType extends AbstractType

{

    public function buildForm(FormBuilderInterface $builder, array $options): void

    {

        $builder

            ->add('email', TextType::class, [

                'label' => false,

                'attr' => [

                    'autocomplete' => 'email',

                    'class' => 'bg-transparent block mt-10 mx-auto border-b-2 w-1/5 h-20 text-2xl outline-none',

                    'placeholder' => 'Email'

                ],

            ])

            ->add('ad', TextType::class, [

                'label' => false,

                'attr' => [

                    'class' => 'bg-transparent block mt-10 mx-auto border-b-2 w-1/5 h-20 text-2xl outline-none',

                    'placeholder' => 'Ad'

                ],

            ])

            ->add('soyad', TextType::class, [

                'label' => false,

                'attr' => [

                    'class' => 'bg-transparent block mt-10 mx-auto border-b-2 w-1/5 h-20 text-2xl outline-none',

                    'placeholder' => 'Soyad'

                ],

            ])

            ->add('address', TextType::class, [

                'label' => false,

                'attr' => [

                    'class' => 'bg-transparent block mt-10 mx-auto border-b-2 w-1/5 h-20 text-2xl outline-none',

                    'placeholder' => 'Address'

                ],

            ])

            ->add('telefon', TextType::class, [

                'label' => false,

                'attr' => [

                    'class' => 'bg-transparent block mt-10 mx-auto border-b-2 w-1/5 h-20 text-2xl outline-none',

                    'placeholder' => 'Telefon'

                ],

            ])

            ->add('roles', ChoiceType::class, [

                'label' => 'Role',

                'choices' => [

                    'Student' => 'ROLE\_STUDENT',

                    'Teacher' => 'ROLE\_TEACHER',

                    // Add more roles as needed

                ],

                'attr' => [

                    'class' => 'bg-transparent block mt-10 mx-auto border-b-2 w-1/5 h-20 text-2xl outline-none',

                ],

                'multiple'=>true,

            ])

            ->add('plainPassword', PasswordType::class, [

                // instead of being set onto the object directly,

                // this is read and encoded in the controller

                'label' => false,

                'mapped' => false,

                'attr' => [

                    'autocomplete' => 'new-password',

                    'class' => 'bg-transparent block mt-10 mx-auto border-b-2 w-1/5 h-20 text-2xl outline-none',

                    'placeholder' => 'Password'

                ],

                'constraints' => [

                    new NotBlank([

                        'message' => 'Please enter a password',

                    ]),

                    new Length([

                        'min' => 6,

                        'minMessage' => 'Your password should be at least {{ limit }} characters',

                        // max length allowed by Symfony for security reasons

                        'max' => 4096,

                    ]),

                ],

            ])

        ;

    }

    public function configureOptions(OptionsResolver $resolver): void

    {

        $resolver->setDefaults([

            'data\_class' => User::class,

        ]);

    }

}

Form/UserProfileType.php

<?php

namespace App\Form;

use App\Entity\User;

use Symfony\Component\Form\AbstractType;

use Symfony\Component\Form\FormBuilderInterface;

use Symfony\Component\OptionsResolver\OptionsResolver;

class UserProfileType extends AbstractType

{

    public function buildForm(FormBuilderInterface $builder, array $options): void

    {

        $builder

            ->add('email')

            ->add('ad')

            ->add('soyad')

            ->add('address')

            ->add('telefon')

        ;

    }

    public function configureOptions(OptionsResolver $resolver): void

    {

        $resolver->setDefaults([

            'data\_class' => User::class,

        ]);

    }

}

Repository

Repository/BookRepository.php

<?php

namespace App\Repository;

use App\Entity\Book;

use Doctrine\Bundle\DoctrineBundle\Repository\ServiceEntityRepository;

use Doctrine\Persistence\ManagerRegistry;

/\*\*

 \* @extends ServiceEntityRepository<Book>

 \*

 \* @method Book|null find($id, $lockMode = null, $lockVersion = null)

 \* @method Book|null findOneBy(array $criteria, array $orderBy = null)

 \* @method Book[]    findAll()

 \* @method Book[]    findBy(array $criteria, array $orderBy = null, $limit = null, $offset = null)

 \*/

class BookRepository extends ServiceEntityRepository

{

    public function \_\_construct(ManagerRegistry $registry)

    {

        parent::\_\_construct($registry, Book::class);

    }

//    /\*\*

//     \* @return Book[] Returns an array of Book objects

//     \*/

//    public function findByExampleField($value): array

//    {

//        return $this->createQueryBuilder('b')

//            ->andWhere('b.exampleField = :val')

//            ->setParameter('val', $value)

//            ->orderBy('b.id', 'ASC')

//            ->setMaxResults(10)

//            ->getQuery()

//            ->getResult()

//        ;

//    }

//    public function findOneBySomeField($value): ?Book

//    {

//        return $this->createQueryBuilder('b')

//            ->andWhere('b.exampleField = :val')

//            ->setParameter('val', $value)

//            ->getQuery()

//            ->getOneOrNullResult()

//        ;

//    }

public function getTotalBookCount(): int

{

    return $this->createQueryBuilder('b')

        ->select('COUNT(b.id)')

        ->getQuery()

        ->getSingleScalarResult();

}

}

Repository/BorrowedbookRepository.php

<?php

namespace App\Repository;

use App\Entity\Borrowedbook;

use App\Entity\User;

use Doctrine\Bundle\DoctrineBundle\Repository\ServiceEntityRepository;

use Doctrine\Persistence\ManagerRegistry;

/\*\*

 \* @extends ServiceEntityRepository<Borrowedbook>

 \*

 \* @method Borrowedbook|null find($id, $lockMode = null, $lockVersion = null)

 \* @method Borrowedbook|null findOneBy(array $criteria, array $orderBy = null)

 \* @method Borrowedbook[]    findAll()

 \* @method Borrowedbook[]    findBy(array $criteria, array $orderBy = null, $limit = null, $offset = null)

 \*/

class BorrowedbookRepository extends ServiceEntityRepository

{

    public function \_\_construct(ManagerRegistry $registry)

    {

        parent::\_\_construct($registry, Borrowedbook::class);

    }

//    /\*\*

//     \* @return Borrowedbook[] Returns an array of Borrowedbook objects

//     \*/

//    public function findByExampleField($value): array

//    {

//        return $this->createQueryBuilder('b')

//            ->andWhere('b.exampleField = :val')

//            ->setParameter('val', $value)

//            ->orderBy('b.id', 'ASC')

//            ->setMaxResults(10)

//            ->getQuery()

//            ->getResult()

//        ;

//    }

//    public function findOneBySomeField($value): ?Borrowedbook

//    {

//        return $this->createQueryBuilder('b')

//            ->andWhere('b.exampleField = :val')

//            ->setParameter('val', $value)

//            ->getQuery()

//            ->getOneOrNullResult()

//        ;

//    }

public function getLateReturnsCount(): int

{

    return $this->createQueryBuilder('bb')

        ->select('COUNT(bb.id)')

        ->where('bb.iadeTarihi < CURRENT\_DATE()')

        ->getQuery()

        ->getSingleScalarResult();

}

public function getBorrowedBookCount(): int

{

    return $this->createQueryBuilder('bb')

        ->select('COUNT(bb.id)')

        ->getQuery()

        ->getSingleScalarResult();

}

public function getUserBorrowedBooksCount(User $user): int

{

    return $this->createQueryBuilder('bb')

        ->select('COUNT(bb.id)')

        ->where('bb.user = :user')

        ->andWhere('bb.iadeTarihi IS NOT NULL') // İade yapılmış kitaplar

        ->getQuery()

        ->setParameter('user', $user)

        ->getSingleScalarResult();

}

}

UserRepository.php

<?php

namespace App\Repository;

use App\Entity\User;

use Doctrine\Bundle\DoctrineBundle\Repository\ServiceEntityRepository;

use Doctrine\Persistence\ManagerRegistry;

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

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

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

/\*\*

 \* @extends ServiceEntityRepository<User>

 \* @implements PasswordUpgraderInterface<User>

 \*

 \* @method User|null find($id, $lockMode = null, $lockVersion = null)

 \* @method User|null findOneBy(array $criteria, array $orderBy = null)

 \* @method User[]    findAll()

 \* @method User[]    findBy(array $criteria, array $orderBy = null, $limit = null, $offset = null)

 \*/

class UserRepository extends ServiceEntityRepository implements PasswordUpgraderInterface

{

    public function \_\_construct(ManagerRegistry $registry)

    {

        parent::\_\_construct($registry, User::class);

    }

    /\*\*

     \* Used to upgrade (rehash) the user's password automatically over time.

     \*/

    public function upgradePassword(PasswordAuthenticatedUserInterface $user, string $newHashedPassword): void

    {

        if (!$user instanceof User) {

            throw new UnsupportedUserException(sprintf('Instances of "%s" are not supported.', $user::class));

        }

        $user->setPassword($newHashedPassword);

        $this->getEntityManager()->persist($user);

        $this->getEntityManager()->flush();

    }

//    /\*\*

//     \* @return User[] Returns an array of User objects

//     \*/

//    public function findByExampleField($value): array

//    {

//        return $this->createQueryBuilder('u')

//            ->andWhere('u.exampleField = :val')

//            ->setParameter('val', $value)

//            ->orderBy('u.id', 'ASC')

//            ->setMaxResults(10)

//            ->getQuery()

//            ->getResult()

//        ;

//    }

//    public function findOneBySomeField($value): ?User

//    {

//        return $this->createQueryBuilder('u')

//            ->andWhere('u.exampleField = :val')

//            ->setParameter('val', $value)

//            ->getQuery()

//            ->getOneOrNullResult()

//        ;

//    }

public function getUsersOrderedByBorrowedBookCount(): array

{

    return $this->createQueryBuilder('u')

        ->leftJoin('u.borrowedbooks', 'bb')

        ->addSelect('COUNT(bb.id) as borrowedBookCount')

        ->groupBy('u.id')

        ->orderBy('borrowedBookCount', 'DESC')

        ->getQuery()

        ->getResult();

}

}

Security/LoginFormAuthenticator.php

<?php

// src/Security/LoginFormAuthenticator.php

namespace App\Security;

use Symfony\Component\HttpFoundation\RedirectResponse;

use Symfony\Component\HttpFoundation\Request;

use Symfony\Component\HttpFoundation\Response;

use Symfony\Component\Routing\Generator\UrlGeneratorInterface;

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

use Symfony\Component\Security\Http\Authenticator\AbstractLoginFormAuthenticator;

use Symfony\Component\Security\Http\Authenticator\Passport\Badge\CsrfTokenBadge;

use Symfony\Component\Security\Http\Authenticator\Passport\Badge\RememberMeBadge;

use Symfony\Component\Security\Http\Authenticator\Passport\Badge\UserBadge;

use Symfony\Component\Security\Http\Authenticator\Passport\Credentials\PasswordCredentials;

use Symfony\Component\Security\Http\Authenticator\Passport\Passport;

use Symfony\Component\Security\Http\SecurityRequestAttributes;

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

class LoginFormAuthenticator extends AbstractLoginFormAuthenticator

{

    use TargetPathTrait;

    public const LOGIN\_ROUTE = 'app\_login';

    public function \_\_construct(private UrlGeneratorInterface $urlGenerator)

    {

    }

    public function authenticate(Request $request): Passport

    {

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

        $request->getSession()->set(SecurityRequestAttributes::LAST\_USERNAME, $email);

        return new Passport(

            new UserBadge($email),

            new PasswordCredentials($request->request->get('password', '')),

            [

                new CsrfTokenBadge('authenticate', $request->request->get('\_csrf\_token')),

                new RememberMeBadge(),

            ]

        );

    }

    public function onAuthenticationSuccess(Request $request, TokenInterface $token, string $firewallName): ?Response

    {

        $userRoles = $token->getRoleNames();

        if (in\_array('ROLE\_ADMIN', $userRoles)) {

            return new RedirectResponse($this->urlGenerator->generate('app\_dashboard'));

        } elseif (in\_array('ROLE\_STUDENT', $userRoles)) {

            return new RedirectResponse($this->urlGenerator->generate('app\_ogrenci'));

        }elseif (in\_array('ROLE\_TEACHER', $userRoles)) {

            return new RedirectResponse($this->urlGenerator->generate('app\_ogretmen'));

        }elseif (in\_array('ROLE\_KUTUPHANECI', $userRoles)) {

            return new RedirectResponse($this->urlGenerator->generate('app\_kutuphaneci'));

        }

        // Default redirect if no specific role is matched

        return new RedirectResponse($this->urlGenerator->generate('app\_dashboard'));

    }

    protected function getLoginUrl(Request $request): string

    {

        return $this->urlGenerator->generate(self::LOGIN\_ROUTE);

    }

}

.env

# In all environments, the following files are loaded if they exist,

# the latter taking precedence over the former:

#

#  \* .env                contains default values for the environment variables needed by the app

#  \* .env.local          uncommitted file with local overrides

#  \* .env.$APP\_ENV       committed environment-specific defaults

#  \* .env.$APP\_ENV.local uncommitted environment-specific overrides

#

# Real environment variables win over .env files.

#

# DO NOT DEFINE PRODUCTION SECRETS IN THIS FILE NOR IN ANY OTHER COMMITTED FILES.

# https://symfony.com/doc/current/configuration/secrets.html

#

# Run "composer dump-env prod" to compile .env files for production use (requires symfony/flex >=1.2).

# https://symfony.com/doc/current/best\_practices.html#use-environment-variables-for-infrastructure-configuration

###> symfony/framework-bundle ###

APP\_ENV=dev

APP\_SECRET=e56b23fdef005e6a79019eaf66a0fba7

###< symfony/framework-bundle ###

###> doctrine/doctrine-bundle ###

# Format described at https://www.doctrine-project.org/projects/doctrine-dbal/en/latest/reference/configuration.html#connecting-using-a-url

# IMPORTANT: You MUST configure your server version, either here or in config/packages/doctrine.yaml

#

# DATABASE\_URL="sqlite:///%kernel.project\_dir%/var/data.db"

DATABASE\_URL="mysql://root@127.0.0.1:3306/MEBOtomasyonDB"

# DATABASE\_URL="mysql://app:!ChangeMe!@127.0.0.1:3306/app?serverVersion=10.11.2-MariaDB&charset=utf8mb4"

#DATABASE\_URL="postgresql://app:!ChangeMe!@127.0.0.1:5432/app?serverVersion=16&charset=utf8"

###< doctrine/doctrine-bundle ###

###> symfony/messenger ###

# Choose one of the transports below

# MESSENGER\_TRANSPORT\_DSN=amqp://guest:guest@localhost:5672/%2f/messages

# MESSENGER\_TRANSPORT\_DSN=redis://localhost:6379/messages

MESSENGER\_TRANSPORT\_DSN=doctrine://default?auto\_setup=0

###< symfony/messenger ###

###> symfony/mailer ###

# MAILER\_DSN=null://null

###< symfony/mailer ###

###> nelmio/cors-bundle ###

CORS\_ALLOW\_ORIGIN='^https?://(localhost|127\.0\.0\.1)(:[0-9]+)?$'

###< nelmio/cors-bundle ###

Composer.js

{

    "type": "project",

    "license": "proprietary",

    "minimum-stability": "stable",

    "prefer-stable": true,

    "require": {

        "php": ">=8.2",

        "ext-ctype": "\*",

        "ext-iconv": "\*",

        "api-platform/core": "^3.2",

        "doctrine/doctrine-bundle": "^2.11",

        "doctrine/doctrine-migrations-bundle": "^3.3",

        "doctrine/orm": "^2.17",

        "endroid/qr-code": "4.8.5",

        "nelmio/cors-bundle": "^2.4",

        "phpdocumentor/reflection-docblock": "^5.3",

        "phpstan/phpdoc-parser": "^1.25",

        "symfony/asset": "7.0.\*",

        "symfony/asset-mapper": "7.0.\*",

        "symfony/console": "7.0.\*",

        "symfony/doctrine-messenger": "7.0.\*",

        "symfony/dotenv": "7.0.\*",

        "symfony/expression-language": "7.0.\*",

        "symfony/flex": "^2",

        "symfony/form": "7.0.\*",

        "symfony/framework-bundle": "7.0.\*",

        "symfony/http-client": "7.0.\*",

        "symfony/intl": "7.0.\*",

        "symfony/mailer": "7.0.\*",

        "symfony/mime": "7.0.\*",

        "symfony/monolog-bundle": "^3.0",

        "symfony/notifier": "7.0.\*",

        "symfony/password-hasher": "7.0.\*",

        "symfony/process": "7.0.\*",

        "symfony/property-access": "7.0.\*",

        "symfony/property-info": "7.0.\*",

        "symfony/runtime": "7.0.\*",

        "symfony/security-bundle": "7.0.\*",

        "symfony/serializer": "7.0.\*",

        "symfony/stimulus-bundle": "^2.14",

        "symfony/string": "7.0.\*",

        "symfony/translation": "7.0.\*",

        "symfony/twig-bundle": "7.0.\*",

        "symfony/ux-turbo": "^2.14",

        "symfony/validator": "7.0.\*",

        "symfony/web-link": "7.0.\*",

        "symfony/yaml": "7.0.\*",

        "twig/extra-bundle": "^2.12|^3.0",

        "twig/twig": "^2.12|^3.0"

    },

    "config": {

        "allow-plugins": {

            "php-http/discovery": true,

            "symfony/flex": true,

            "symfony/runtime": true,

            "endroid/installer": true

        },

        "sort-packages": true

    },

    "autoload": {

        "psr-4": {

            "App\\": "src/"

        }

    },

    "autoload-dev": {

        "psr-4": {

            "App\\Tests\\": "tests/"

        }

    },

    "replace": {

        "symfony/polyfill-ctype": "\*",

        "symfony/polyfill-iconv": "\*",

        "symfony/polyfill-php72": "\*",

        "symfony/polyfill-php73": "\*",

        "symfony/polyfill-php74": "\*",

        "symfony/polyfill-php80": "\*",

        "symfony/polyfill-php81": "\*",

        "symfony/polyfill-php82": "\*"

    },

    "scripts": {

        "auto-scripts": {

            "cache:clear": "symfony-cmd",

            "assets:install %PUBLIC\_DIR%": "symfony-cmd",

            "importmap:install": "symfony-cmd"

        },

        "post-install-cmd": [

            "@auto-scripts"

        ],

        "post-update-cmd": [

            "@auto-scripts"

        ]

    },

    "conflict": {

        "symfony/symfony": "\*"

    },

    "extra": {

        "symfony": {

            "allow-contrib": false,

            "require": "7.0.\*"

        }

    },

    "require-dev": {

        "phpunit/phpunit": "^9.5",

        "symfony/browser-kit": "7.0.\*",

        "symfony/css-selector": "7.0.\*",

        "symfony/debug-bundle": "7.0.\*",

        "symfony/maker-bundle": "^1.53",

        "symfony/phpunit-bridge": "^7.0",

        "symfony/stopwatch": "7.0.\*",

        "symfony/web-profiler-bundle": "7.0.\*"

    }

}