Layer Architektur

In Symfony

Rafał Wesołowski

Beruf:

- Developer Evangelist bei NEXUS United a valantic company
- Ausbilder
- PHP Experte
- SOLID Principles and TDD Enthusiast
- Innovations Fan

Privat:

- Kampfsportler
- Mehrfacher Polen-Meister im Ju-Jitsu
- Jugend-Fußball Trainer
- Aktiver Amateur-Boxer

wesolowski

WesolowskiRafal

Rafał Wesołowski

E-Commere

OXID eSales

Shopware

Spryker

Symfony

Was habe ich gelernt und erfahren?

OXID eshop

Positiv

- Alles ist erweiterbar
- Wenig JOINS
- Performance Optimierung
- Table Views

Negativ

- OXID Entwickler, nicht PHP- Entwickler
- Wenig Composer, keine Namespace
- OOP???

Was habe ich gelernt und erfahren?

Positiv

- Stärke von Full-Page-Cache
- Service und Dekoration von Services
- Private Methode als Standard
- Single Point of Truth

Negativ

- Schwäche von Full-Page-Cache
- Events sind schlecht
- JOINS und Table lock

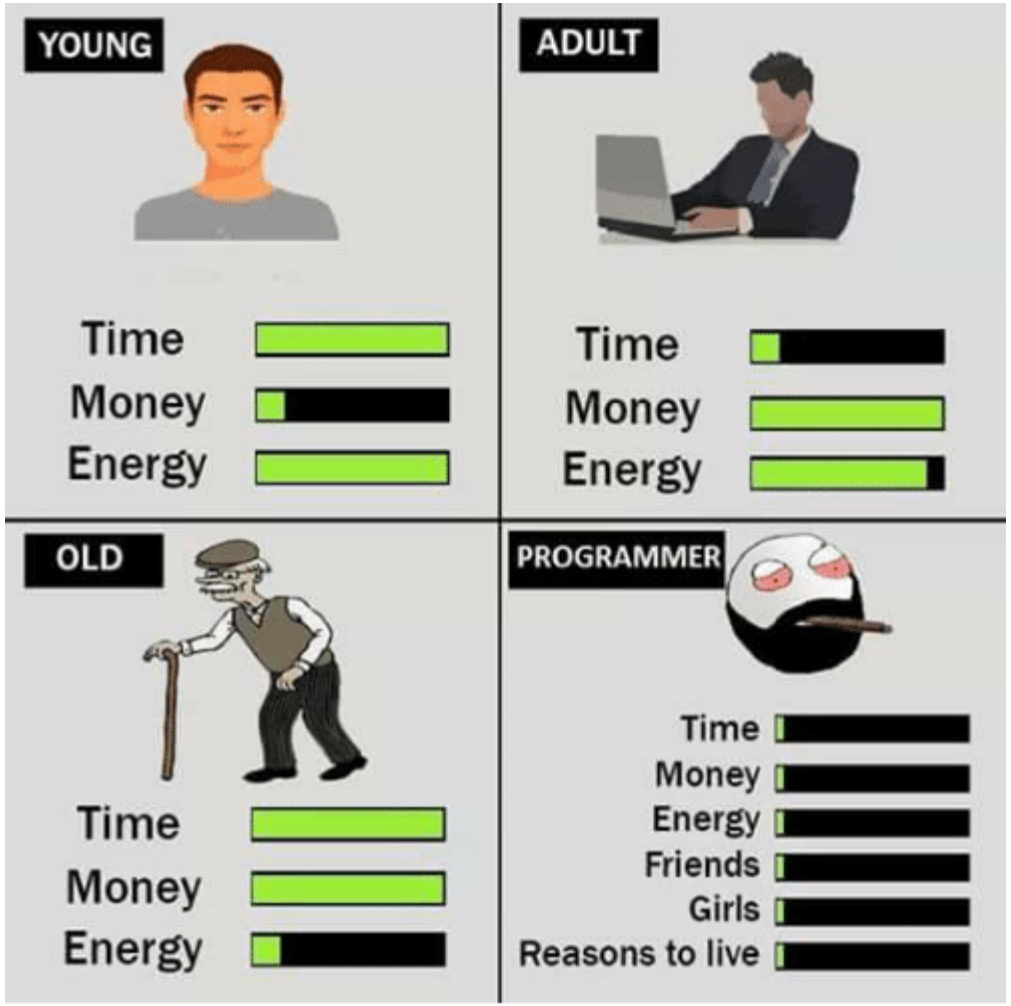
Was habe ich gelernt und erfahren?

Positiv

- Layer-Architektur
- CQRS
- DTO
- SOLID
- PostgreSQL

Negativ

- Komplexität
- Herausfordernd für unerfahrene Entwickler
- Kein Single Point of Truth
- Spryker Entwickler, nicht PHP- Entwickler



I am Programmer, I have no life. : memes

Code-Applikation

Symfony

- Controller
  - Article
    - CreateArticleController.php
    - DeleteArticleController.php
    - FavoriteArticleController.php
    - GetArticlesFeedController.php
    - GetArticlesListController.php
    - GetOneArticleController.php
    - UnfavoriteArticleController.php
    - UpdateArticleController.php
  - Comment
    - CreateCommentController.php
    - DeleteCommentController.php
    - GetCommentsListController.php
  - Profile
    - FollowProfileController.php
    - GetProfileController.php
    - UnfollowProfileController.php
  - Registration
    - RegisterController.php
  - Security
    - LoginController.php
  - Tag
    - GetTagsListController.php
  - User
    - GetUserController.php
    - UpdateUserController.php

- Form
  - DataTransformer
    - TagArrayToStringTransformer.php
  - Type
    - TagsInputType.php
  - ArticleType.php
  - CommentType.php
  - UserType.php

- EventListener
  - JWTAuthenticationSubscriber.php
- Exception
  - NoCurrentUserException.php

- Serializer
  - Normalizer
    - ArticleNormalizer.php
    - CommentNormalizer.php
    - DateTimeNormalizer.php
    - FormErrorNormalizer.php
    - TagNormalizer.php
    - UserNormalizer.php

- Repository
  - ArticleRepository.php
  - CommentRepository.php
  - TagRepository.php
  - UserRepository.php

- DataFixtures
  - Processor
    - UserProcessor.php
  - Provider
    - CollectionProvider.php
  - AppFixtures.php

- Entity
  - Article.php
  - Comment.php
  - Tag.php
  - User.php

README.md

RealWorld example app

> Symfony codebase containing real world examples (CRUD, auth, advanced patterns, etc) that adheres to the RealWorld spec and API.

coverage 91 % | Scrutinizer 10.00 | build passed

This codebase was created to demonstrate a fully fledged fullstack application built with **Symfony** including CRUD operations, authentication, routing, pagination, and more.

We've gone to great lengths to adhere to the **Symfony** community styleguides & best practices.

For more information on how to this works with other frontends/backends, head over to the RealWorld repo.

https://github.com/slashfan/symfony-realworld-example-app

Code-Applikation

Symfony

- Controller
  - Article
    - CreateArticleController.php
    - DeleteArticleController.php
    - FavoriteArticleController.php
    - GetArticlesFeedController.php
    - GetArticlesListController.php
    - GetOneArticleController.php
    - UnfavoriteArticleController.php
    - UpdateArticleController.php
  - Comment
    - CreateCommentController.php
    - DeleteCommentController.php
    - GetCommentsListController.php
  - Profile
    - FollowProfileController.php
    - GetProfileController.php
    - UnfollowProfileController.php
  - Registration
    - RegisterController.php
  - Security
    - LoginController.php
  - Tag
    - GetTagsListController.php
  - User
    - GetUserController.php
    - UpdateUserController.php

- Form
  - DataTransformer
    - TagArrayToStringTransformer.php
  - Type
    - TagsInputType.php
  - ArticleType.php
  - CommentType.php
  - UserType.php

- EventListener
  - JWTAuthenticationSubscriber.php
- Exception
  - NoCurrentUserException.php

- Serializer
  - Normalizer
    - ArticleNormalizer.php
    - CommentNormalizer.php
    - DateTimeNormalizer.php
    - FormErrorNormalizer.php
    - TagNormalizer.php
    - UserNormalizer.php

- Repository
  - ArticleRepository.php
  - CommentRepository.php
  - TagRepository.php
  - UserRepository.php

- DataFixtures
  - Processor
    - UserProcessor.php
  - Provider
    - CollectionProvider.php
  - AppFixtures.php

- Entity
  - Article.php
  - Comment.php
  - Tag.php
  - User.php

- Service
  - Container.php
  - ControllerProvider.php
  - CsvImportLoader.php
  - DatabaseManager.php
  - DependencyProvider.php
  - FileManager.php
  - Importer.php
  - ImportManager.php
  - Mailer.php
  - MappingAssistant.php
  - PasswordManager.php
  - SessionUser.php
  - SymfonyMailerManager.php
  - View.php

- Mapper
  - ProductImportMapper.php
  - ProductMapper.php
  - ProductMapperInterface.php
  - UserMapper.php
  - UserMapperInterface.php

Code-Applikation

Symfony

- Controller
  - Article
    - CreateArticleController.php
    - DeleteArticleController.php
    - FavoriteArticleController.php
    - GetArticlesFeedController.php
    - GetArticlesListController.php
    - GetOneArticleController.php
    - UnfavoriteArticleController.php
    - UpdateArticleController.php
  - Comment
    - CreateCommentController.php
    - DeleteCommentController.php
    - GetCommentsListController.php
  - Profile
    - FollowProfileController.php
    - GetProfileController.php
    - UnfollowProfileController.php
  - Registration
    - RegisterController.php
  - Security
    - LoginController.php
  - Tag
    - GetTagsListController.php
  - User
    - GetUserController.php
    - UpdateUserController.php

- Form
  - DataTransformer
    - TagArrayToStringTransformer.php
  - Type
    - TagsInputType.php
  - ArticleType.php
  - CommentType.php
  - UserType.php

- EventListener
  - JWTAuthentic
- Exception
  - NoCurrentU

- Serializer
  - Normal
    - Artic
    - Comme
    - DateTimeN
    - FormE
    - TagN
    - User

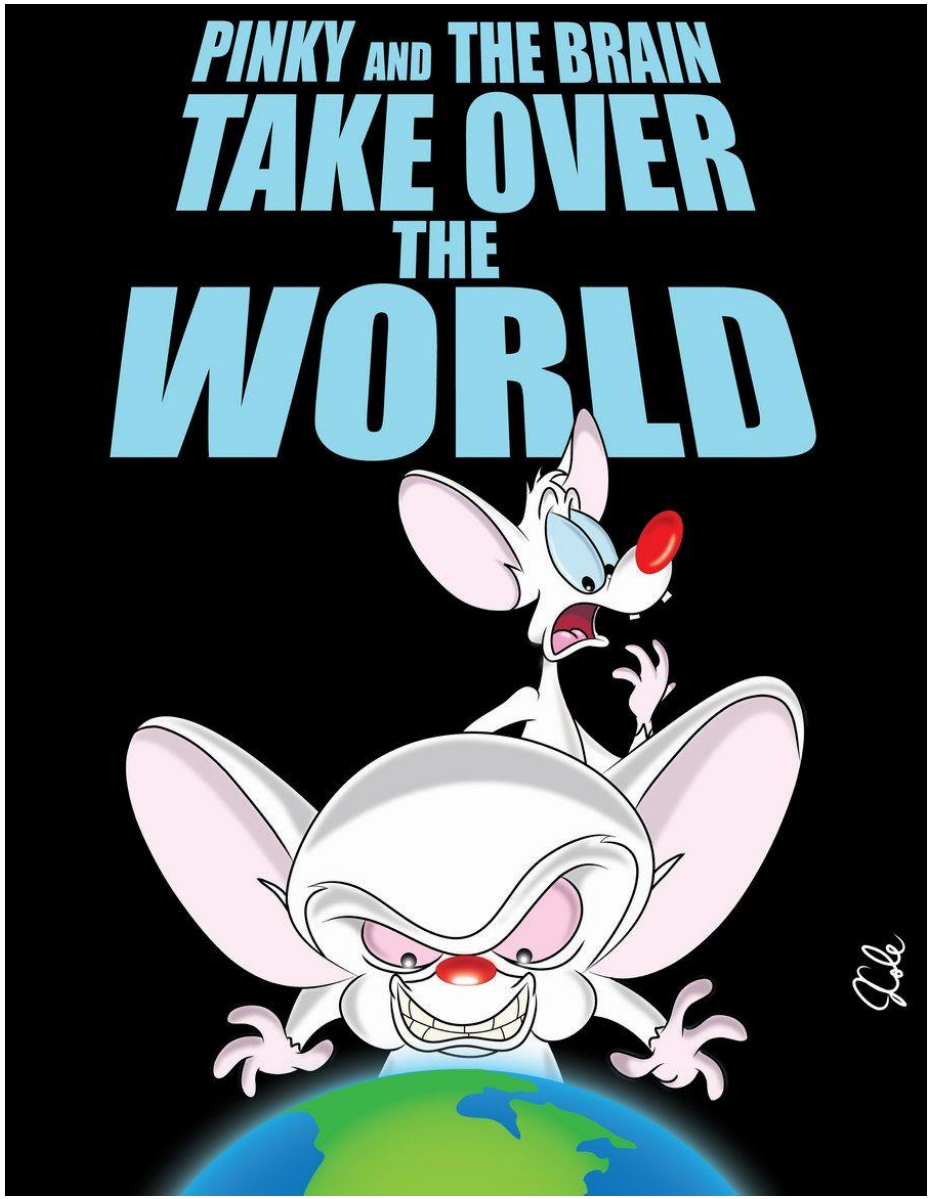
- Repository
  - ArticleRepository.php
  - CommentRepository.php
  - TagRepository.php
  - UserRepository.php

- DataFixtures
  - Processor
    - UserProcessor.php
  - Provider
    - CollectionProvider.php
  - AppFixtures.php

- .php
- rProvider.php
- rtLoader.php
- nager.php
- rovider.php
- hp
- ger.php
- ant.php
- ger.php
- hp
- erManager.php
- View.php

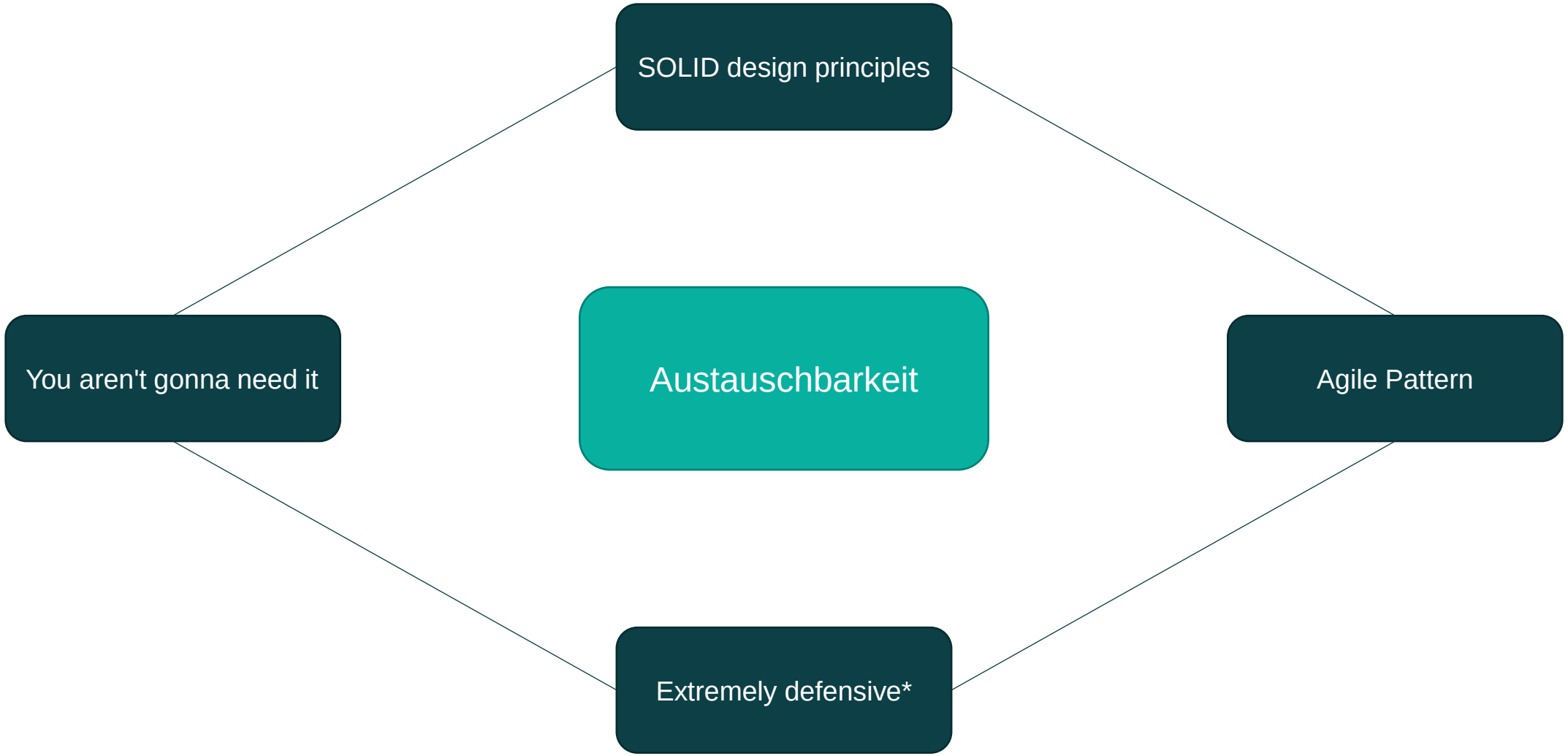
- Entity
  - Article.php
  - Comment.php
  - Tag.php
  - User.php

- Mapper
  - ProductImportMapper.php
  - ProductMapper.php
  - ProductMapperInterface.php
  - UserMapper.php
  - UserMapperInterface.php


PINKY AND THE BRAIN
TAKE OVER
THE
WORLD

Wenn du deinen Code kennst und weißt was der Kunde noch will

Lösung



* Extremely Defensive PHP - Marco Pivetta: https://www.youtube.com/watch?v=8d2AtAGJPno

Eine Applikation erstellen

Brillen Konfigurator

↓

Deutscher Markt

↓

Handler

↓

SAP Import / Export

Fassung

Nach Artikelnummer suchen

Nach Modell suchen

Glasmaterial

Material (*)

Liste durchsuchen

- Kunststoff (inkl. Kratzfestbeschichtung)
- Mineralglas 160
- PC+ (inkl. Kratzfestbeschichtung)
- Polycarbonat (inkl. Kratzfestbeschichtung)
- Trivex (inkl. Kratzfestbeschichtung)

Focustyp (*)

Liste durchsuchen

- Einstärken
- Einstärken HD
- Gleitsicht Optima
- Gleitsicht Optima HD
- Gleitsicht Pro Work
- Nahcomfort Optima

Entspiegelung (*)

Liste durchsuchen

- Super Entspiegelung
- ohne Entspiegelung

Tönungsfarbe (*)

Liste durchsuchen

- Braun
- Grau
- Ohne Tönung
- UV Blue Protect

Tönung in % (*)

Liste durchsuchen

- 12
- 15
- 25
- 65
- 75
- 85
- FIX

Eine Applikation erstellen

Brillen Konfigurator

↓

Deutscher Markt

↓

Handler

↓

SAP Import / Export

Fassung

Nach Artikelnummer suchen

Nach Modell suchen

Glasmaterial

Material (*)

Liste durchsuchen

- [x] Kunststoff (inkl. Kratzfestbeschichtung)
- [ ] Mineralglas 160
- [ ] PC+ (inkl. Kratzfestbeschichtung)
- [ ] Polycarbonat (inkl. Kratzfestbeschichtung)
- [ ] Trivex (inkl. Kratzfestbeschichtung)

Focustyp (*)

Liste durchsuchen

- [ ] Einstärken
- [x] Einstärken HD
- [ ] Gleitsicht Optima
- [ ] Gleitsicht Optima HD
- [ ] Gleitsicht Pro Work
- [ ] Nahcomfort Optima

Entspiegelung (*)

Liste durchsuchen

- [ ] Super Entspiegelung
- [x] ohne Entspiegelung

Tönungsfarbe (*)

Liste durchsuchen

- [ ] Braun
- [x] Grau
- [ ] Ohne Tönung
- [ ] UV Blue Protect

Tönung in % (*)

Liste durchsuchen

- [x] 12
- [ ] 15
- [ ] 25
- [ ] 65
- [ ] 75
- [ ] 85
- [ ] FIX

5500 Skymega

Skymega blau grau

9198299

- Farbe: blau, grau
- Scheibengröße: Millimeter
- Stegweite: Millimeter

KONFIGURIEREN

Softwarearchitektur

Import

- User
- Filter incl. Produkte
- Produkt Bilder

Export

- SAP Bestellung

Frontend

- User Login
- Admin User
- Formulare
- Konfigurator
- Freigabe
- BlackList Konfigurator
- Rolen:
    - Admin
    - User
    - Optiker

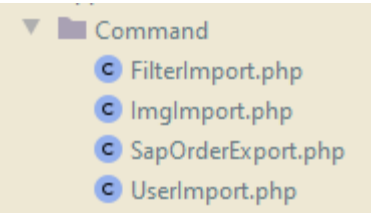
Technische Anforderungen

- Symfony
    - Doctrine
    - Twig
- MySQL
- Psalm
- phpUnit
- Deptrac
- Piplines

Softwarearchitektur - MVC

Import

- User
- Filter incl. Produkte
- Produkt Bilder

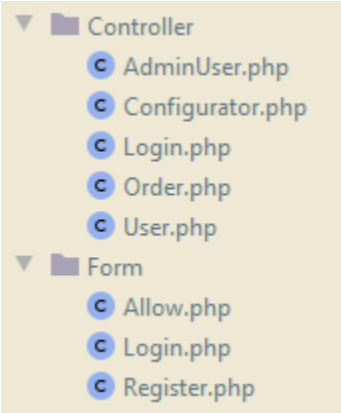


Export

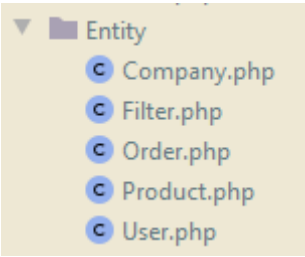
- SAP Bestellung

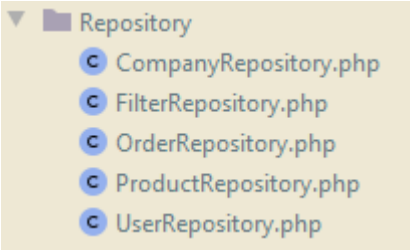
Frontend

- User Login
- Admin User
- Formulare
- Konfigurator
- Freigabe
- BlackList Konfigurator
- Rolen:
    - Admin
    - User
    - Optiker



Doctrine





Wo ist die Business Logik bei MVC

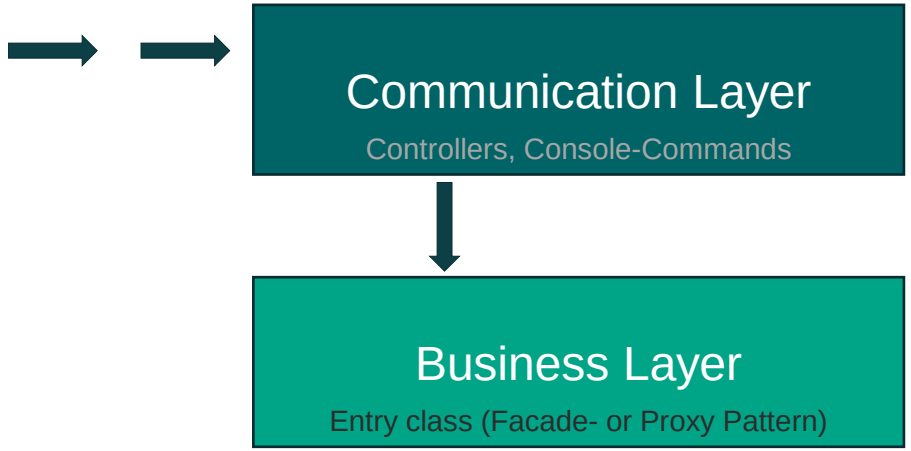
https://softwareengineering.stackexchange.com/questions/176639/why-put-the-business-logic-in-the-model-what-happens-when-i-have-multiple-types

Softwarearchitektur - MVC

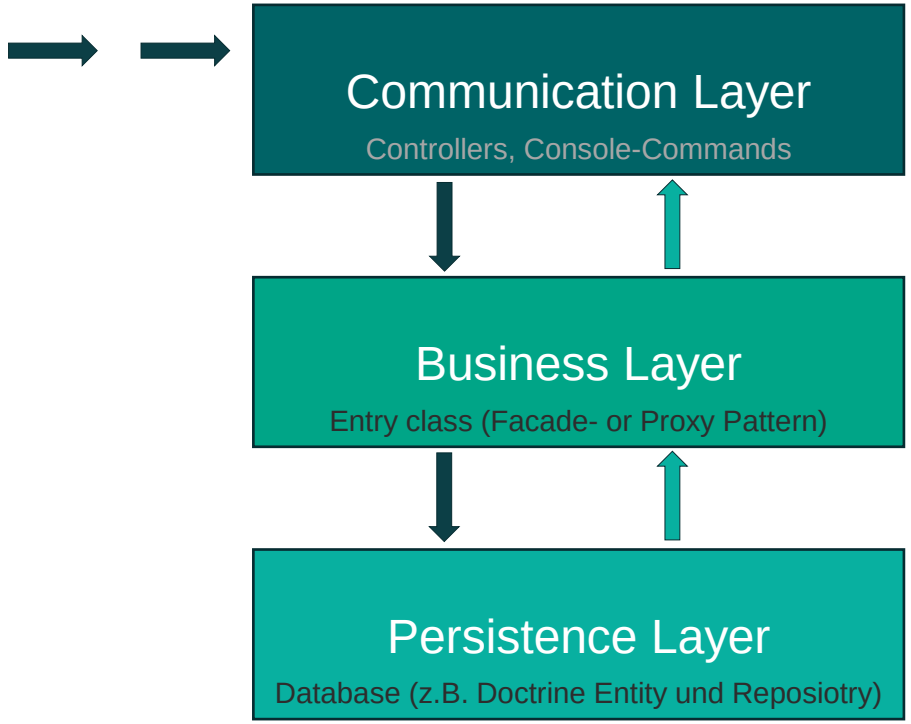
- Command
  - FilterImport.php
  - ImgImport.php
  - SapOrderExport.php
  - UserImport.php
- Controller
  - AdminUser.php
  - Configurator.php
  - Login.php
  - Order.php
  - User.php
- Entity
  - Company.php
  - Filter.php
  - Order.php
  - Product.php
  - User.php
- Form
  - Allow.php
  - Login.php
  - Register.php
- Repository
  - CompanyRepository.php
  - FilterRepository.php
  - OrderRepository.php
  - ProductRepository.php
  - UserRepository.php

Schichtenarchitektur

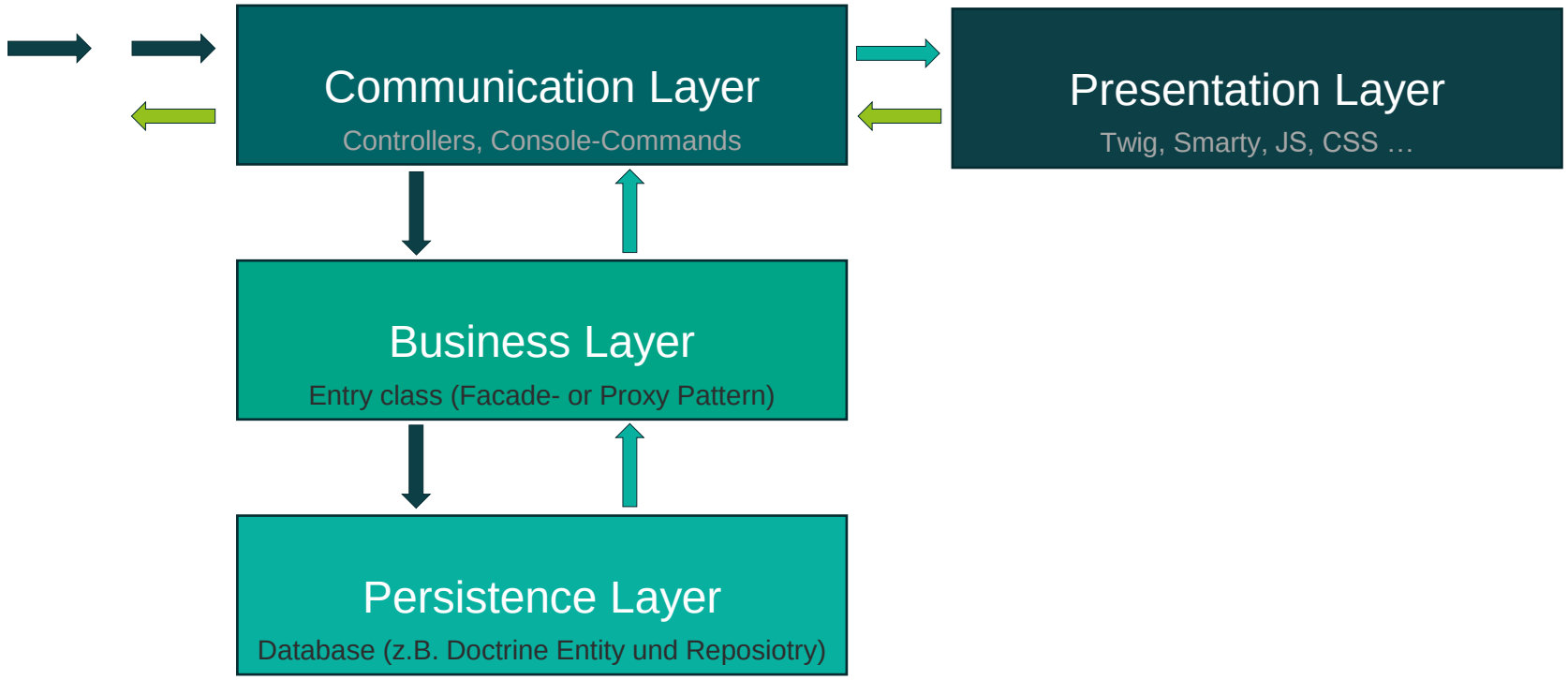
Schichtenarchitektur



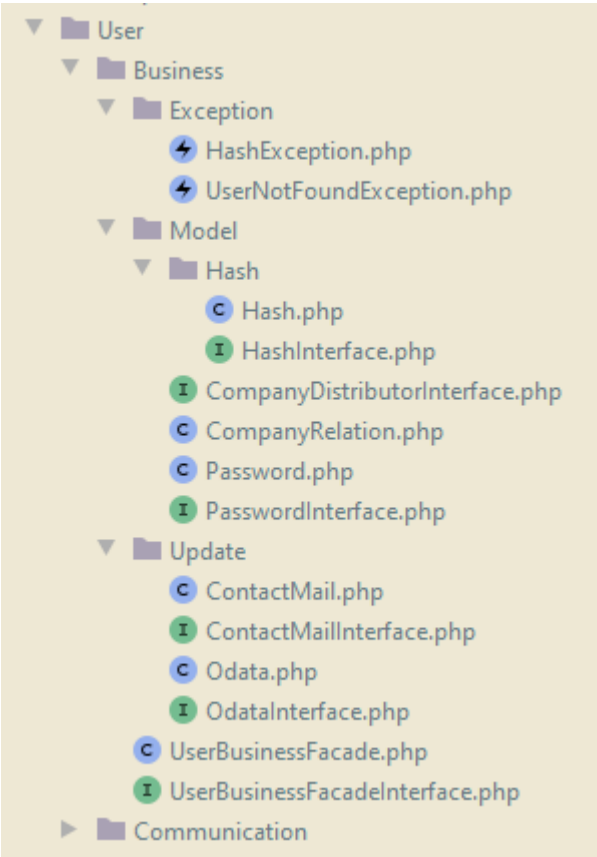
Schichtenarchitektur



Schichtenarchitektur

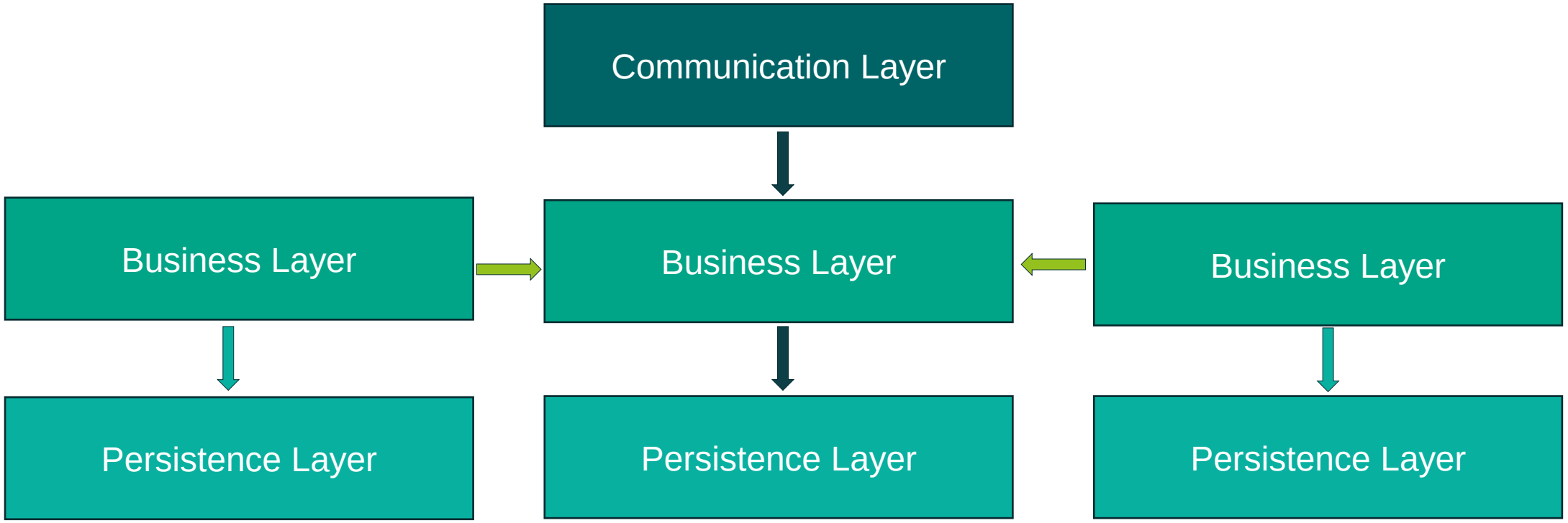


Business Layer

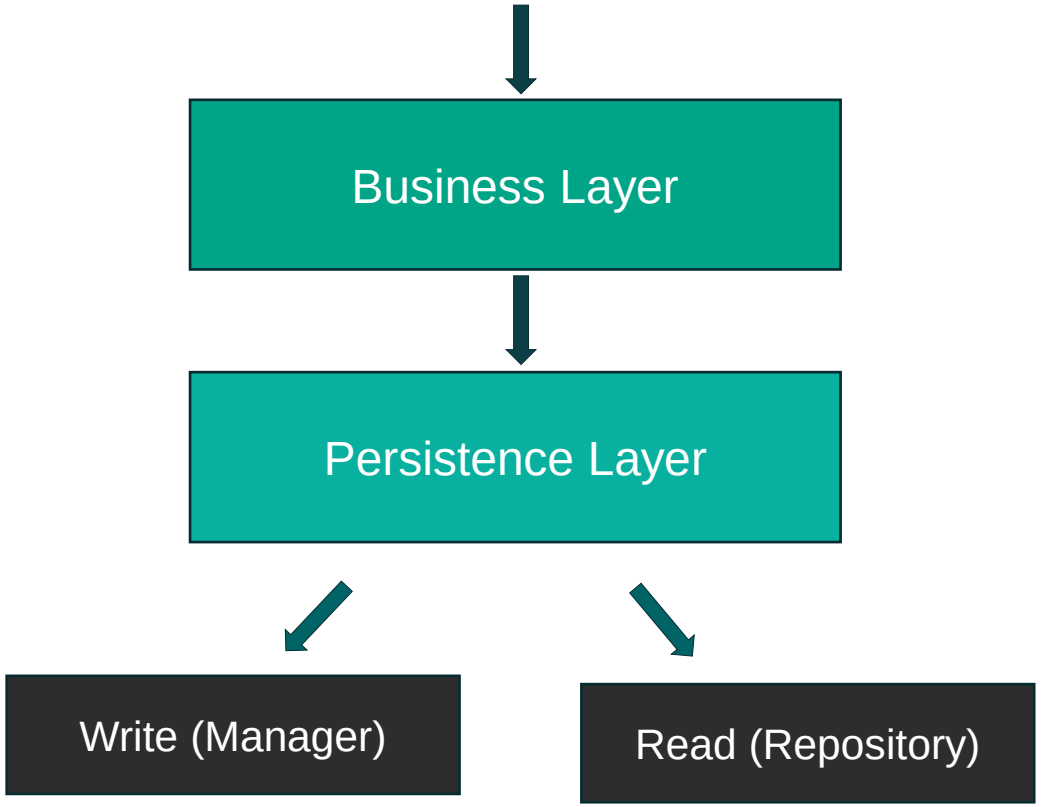



```php
class UserBusinessFacade implements UserBusinessFacadeInterface
{
    public function __construct(
        CompanyDistributorInterface $companyDistributor,
        ContactMailInterface $contactMail,
        PasswordInterface $password,
        CompanyEntityManager $companyWriteManager,
        CompanyRepository $companyRepository,
        UserRepository $userRepository,
        OdataInterface $odata
    )
    {
        $this->companyDistributor = $companyDistributor;
        $this->contactMail = $contactMail;
        $this->password = $password;
        $this->companyWriteManager = $companyWriteManager;
        $this->companyRepository = $companyRepository;
        $this->userRepository = $userRepository;
        $this->odata = $odata;
    }

    public function setEmailWithChangePassword(UserEntityDataProvider $user, string $mainUrl): void
    {
        $this->contactMail->setEmailWithChangePassword($user, $mainUrl);
    }

    public function getOpticianCompany(): array
    {
        return $this->companyRepository->getOpticianCompany();
    }

    public function saveRoleAndInvoiceRecipient(CompanyEntityDataProvider $companyEntityData): void
    {
        $this->companyWriteManager->updateRoleAndInvoiceRecipientById($companyEntityData);
    }

    ...
    ...
    ...
```

Schichtenarchitektur



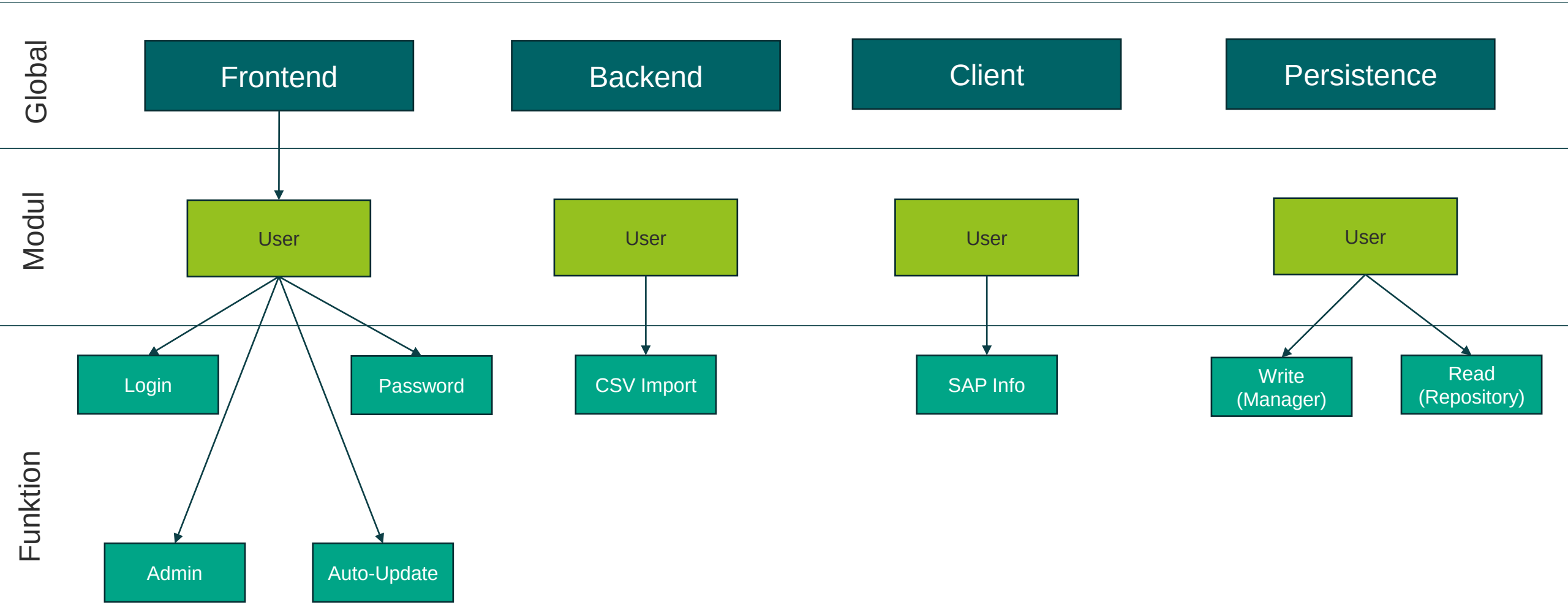
Schichtenarchitektur



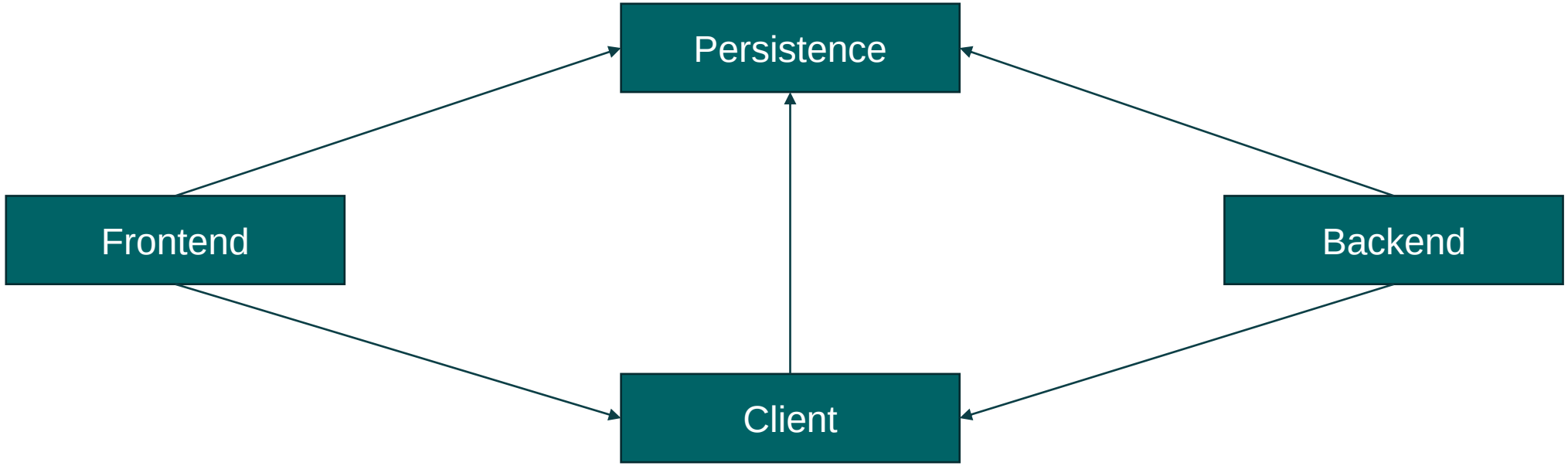

```php
/**
 * @method CompanyEntityDataProvider|null find($id, $lockMode = null, $lockVersion = null)
 * @method CompanyEntityDataProvider|null findOneBy(array $criteria, array $orderBy = null)
 * @method CompanyEntityDataProvider[]    findAll()
 * @method CompanyEntityDataProvider[]    findBy(array $criteria, array $orderBy = null, $limit = null, $offset = null)
 */
class CompanyRepository extends ServiceEntityRepository
{
    public function __construct(ManagerRegistry $registry)
    {
        parent::__construct($registry, Company::class);
    }
}
```

```php
class CompanyEntityManager implements CompanyWriteManagerInterface
{
    /**
     * @var \App\Component\Persistence\CompanyRepository
     */
    private $companyRepository;
    /**
     * @var \Doctrine\ORM\EntityManagerInterface
     */
    private $objectManager;

    public function __construct(
        CompanyRepository $companyRepository,
        EntityManagerInterface $objectManager
    )
    {
    }

    /**
     * @param \App\DataTransferObject\CompanyEntityDataProvider $companyDataProvider
     * @return \App\DataTransferObject\CompanyEntityDataProvider
     */
    public function save(CompanyEntityDataProvider $companyDataProvider) : CompanyEntityDataProvider
    {
    }

    /**
     * @param \App\DataTransferObject\CompanyEntityDataProvider $companyDataProvider
     * @return \App\DataTransferObject\CompanyEntityDataProvider
     */
    public function updateRoleAndInvoiceRecipientById(CompanyEntityDataProvider $companyDataProvider): CompanyEntityDataProvider
    {
    }
}
```

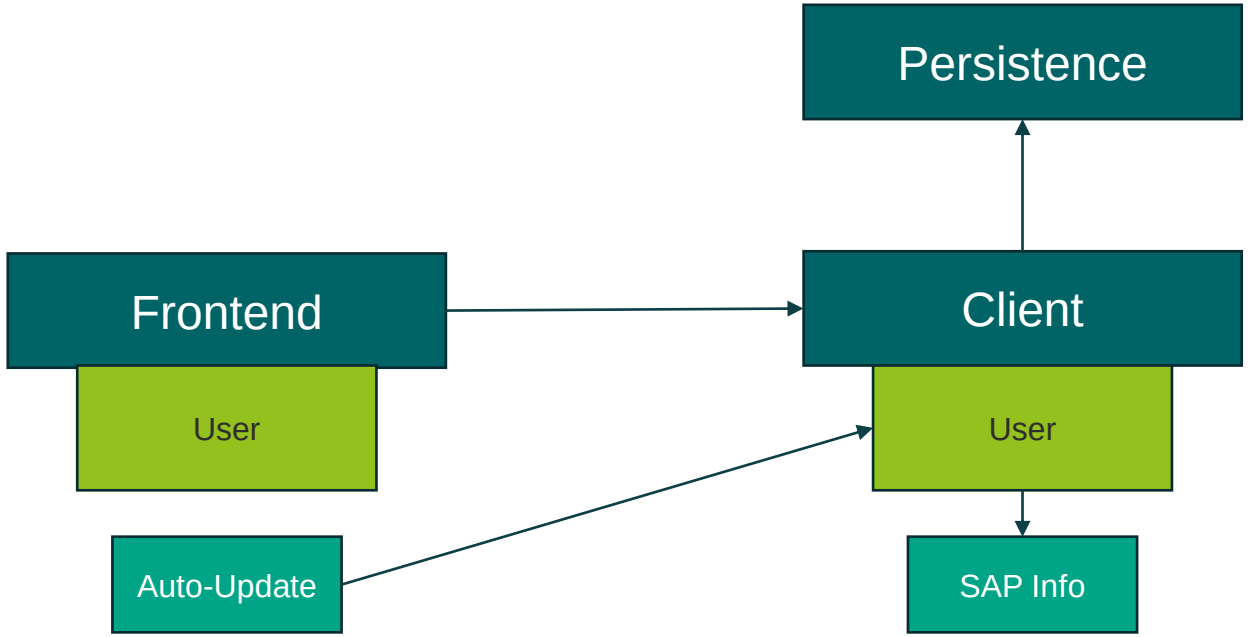
Softwarearchitektur in die Applikation (User Beispiel)



Kommunikation in die Applikation (User Beispiel)



Kommunikation in die Applikation (User Beispiel)



User Aktualisierung Beispiel


```php
class LoginFormAuthenticator extends AbstractFormLoginAuthenticator
{
    ...

    public function getUser($credentials, UserProviderInterface $userProvider)
    {
        ...

        $user = $this->userRepository->findOneByEmail($credentials['email']);

        if (!$user instanceof User) {
            throw new CustomUserMessageAuthenticationException(
                $this->translator->trans('login_exception_email_not_found')
            );
        }

        $user = $this->userClient->updateUserInfo($user);

        if ($user->getStatus() !== User::StatusAllowed) {
            throw new CustomUserMessageAuthenticationException(
                $this->translator->trans('login_exception_user_not_active')
            );
        }

        return $user;
    }

    ...
}
```

```php
namespace App\Client\User;

class UserClient implements UserClientInterface
{
    public function updateUserInfo(UserEntityDataProvider $user) : Us
    {
        return $this->oDataImport->updateUserInfo($user);
    }
}
```

User Aktualisierung Beispiel


```php
namespace App\Client\User;

class UserClient implements UserClientInterface
{
    public function updateUserInfo(UserEntityDataProvider $user)
    {
        return $this->oDataImport->updateUserInfo($user);
    }
}
```

```php
namespace App\Client\User\Odata;

class Import implements ImportInterface
{
    ...

    public function updateUserInfo(UserEntityDataProvider $user): UserEntityDataProvider
    {
        $companyEntityDataProvider = $this->getCompany($user->getCompany()->getNumber());
        $userEntityDataProviderList = $this->getCustomerList($companyEntityDataProvider);

        $userEntityDataProviderListFromDbSortById = $this->getUserListFromDb($companyEntityDataProvider);

        foreach ($userEntityDataProviderList as $userEntityDataProvider) {
            if ($userEntityDataProvider->hasId() && isset($userEntityDataProviderListFromDbSortById[$userEntity
                unset($userEntityDataProviderListFromDbSortById[$userEntityDataProvider->getId()]);
            }
            $this->userWriteManager->save($userEntityDataProvider);
        }

        foreach ($userEntityDataProviderListFromDbSortById as $userEntityDataProvider) {
            $this->userWriteManager->updateStatus(User::StatusNotAllowed, $userEntityDataProvider->getId());
        }

        $companyEntityDataProvider->setUsers(array_values($this->getUserListFromDb($companyEntityDataProvider))

        return $this->findUser($companyEntityDataProvider, $user);
    }

    ...
}
```

Applikation

Fassung

Nach Artikelnummer suchen

Nach Modell suchen

Glasmaterial

Material (*)

Liste durchsuchen

- [x] Kunststoff (inkl. Kratzfestbeschichtung)
- [ ] Mineralglas 160
- [ ] PC+ (inkl. Kratzfestbeschichtung)
- [ ] Polycarbonat (inkl. Kratzfestbeschichtung)
- [ ] Trivex (inkl. Kratzfestbeschichtung)

Focustyp (*)

Liste durchsuchen

- [ ] Einstärken
- [x] Einstärken HD
- [ ] Gleitsicht Optima
- [ ] Gleitsicht Optima HD
- [ ] Gleitsicht Pro Work
- [ ] Nahcomfort Optima

Entspiegelung (*)

Liste durchsuchen

- [ ] Super Entspiegelung
- [x] ohne Entspiegelung

Tönungsfarbe (*)

Liste durchsuchen

- [ ] Braun
- [x] Grau
- [ ] Ohne Tönung
- [ ] UV Blue Protect

Tönung in % (*)

Liste durchsuchen

- [x] 12
- [ ] 15
- [ ] 25
- [ ] 65
- [ ] 75
- [ ] 85
- [ ] FIX

5500 Skymega

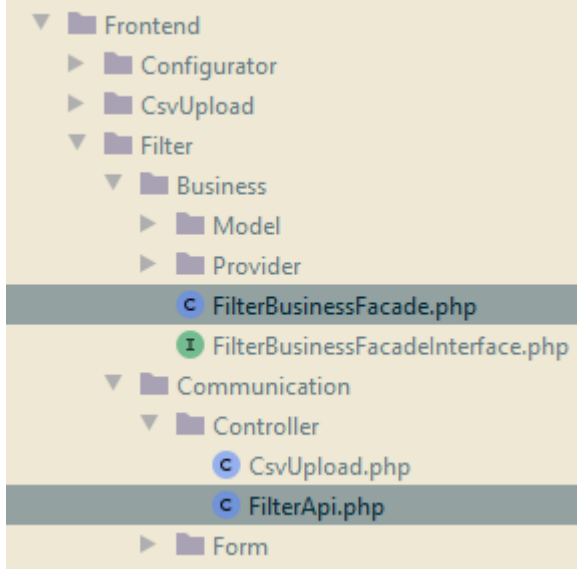
Skymega blau grau

9198299

- Farbe: blau, grau
- Scheibengröße: Millimeter
- Stegweite: Millimeter

KONFIGURIEREN

Filter




```php
namespace App\Frontend\Filter\Communication\Controller;

class FilterApi extends AbstractController
{
    ...

    /**
     * @Route("/api/filter/{id}", name="api_filter_all", methods={"GET"})
     */
    public function getList(Request $request, CompanyEntityDataProvider $comp
    {
        $filterRequest = $this->getFilterRequestDataProvider($request, $compa
        $status = 200;
        $data = [
            'success' => true,
            'data' => $this->filterBusinessFacade->getAll($filterRequest)
        ];

        return $this->json($data, $status)->setEncodingOptions(JSON_UNESCAPED
    }

    ...
}
```

```php
namespace App\Frontend\Filter\Business;

class FilterBusinessFacade implements FilterBusinessFacadeInterface
{
    ...

    /**
     * @param \App\DataTransferObject\FilterRequestDataProvider $filt
     * @return \App\DataTransferObject\ConfiguratorInfoDataProvider
     */
    public function getAll(FilterRequestDataProvider $filterRequest):
    {
        return $this->filter->getAll($filterRequest);
    }

    ...
}
```

Was soll FilterFacade dem Controller zurückgeben?

Array

Data
Transfer Object
(DTO)

Data transfer object (DTO) vs Array

shopware/Detail.php at 5.6 · shop

github.com/shopware/shopware/blob/5.6/engine/Shopware/Controllers/Frontend/Detail.php

```
142
143         // SW-3493 sArticle->getArticleById and sBasket->sGetGetBasket differ in camelcase
144         $product['sReleaseDate'] = $product['sReleasedate'];
145
```

https://github.com/shopware/shopware/blob/5.6/engine/Shopware/Controllers/Frontend/Detail.php

Data transfer object (DTO)

Typen:

- Bool
- Integer
- Float
- String
- Array of DTOs
- DTO

Methoden:

- set
- get
- has
- required (value ist not empty)

Data transfer object (DTO)

README.md

Xervice: DataProvider

Scrutinizer 9.90 | coverage 87 %

Data transfer objects for xervice packages.

Installation

```
composer require xervice/data-provider
```

Define DTO

To define a data provider, you define them in an xml file.

Example:

```
<?xml version="1.0"?>

<DataProviders
  xmlns="xervice:dataprovider-01"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="xervice:dataprovider-01 http://static.xervice.online/schema/dataprovider.schema.xsd"
>

    <DataProvider name="KeyValue">
        <DataElement name="Key" type="string"/>
        <DataElement name="Value" type="string"/>
    </DataProvider>
</DataProviders>
```

Use DTO

```
$dataProvider = new DataProvider\KeyValueDataProvider();

// Set values
$dataProvider->setKey('keyname');
$dataProvider->setValue('value');

// Get values
$dataProvider->getKey();

// Isset
$dataProvider->hasKey();

// you can also work with arrays
$dataProvider->fromArray([
    'Key' => 'keyname',
    'Value' => 'value'
]);

// and back to array
$dataArray = $dataProvider->toArray();
```

Jetzt: https://github.com/xervice/data-provider

In Zukunft: https://github.com/orbit-core/data-transfer

Filter


```php
namespace App\Frontend\Filter\Communication\Controller;

class FilterApi extends AbstractController
{
    ...

    /**
     * @Route("/api/filter/{id}", name="api_filter_all", methods={"GET"})
     */
    public function getList(Request $request, CompanyEntityDataProvider $comp
    {
        $filterRequest = $this->getFilterRequestDataProvider($request, $compa
        $status = 200;
        $data = [
            'success' => true,
            'data' => $this->filterBusinessFacade->getAll($filterRequest)
        ];

        return $this->json($data, $status)->setEncodingOptions(JSON_UNESCAPED
    }

    ...
}
```

```php
namespace App\Frontend\Filter\Business;

class FilterBusinessFacade implements FilterBusinessFacadeInterface
{
    ...

    /**
     * @param \App\DataTransferObject\FilterRequestDataProvider $filt
     * @return \App\DataTransferObject\ConfiguratorInfoDataProvider
     */
    public function getAll(FilterRequestDataProvider $filterRequest):
    {
        return $this->filter->getAll($filterRequest);
    }

    ...
}
```

Filter


```php
namespace App\Frontend\Filter\Business;

class FilterBusinessFacade implements FilterBusi
{
    ...

    /**
     * @param \App\DataTransferObject\FilterRequ
     * @return \App\DataTransferObject\Configura
     */
    public function getAll(FilterRequestDataProv
    {
        return $this->filter->getAll($filterRequ
    }

    ...
}
```

```php
namespace App\Frontend\Filter\Business\Model;

class Filter implements FilterInterface
{
    public function getAll(FilterRequestDataProvider $filterRequest): ConfiguratorInfoDataPro
    {
        $filterRequest->setFilterList(new FilterListDataProvider());
        $filterDataProviderList = $this->fetch($filterRequest);
        ...
        return $configuratorInfoDataProvider;
    }

    public function search(FilterRequestDataProvider $filterRequest): ConfiguratorInfoDataPro
    {
        $filterDataProviderList = $this->fetch($filterRequest);
        ...
        return $configuratorInfoDataProvider;
    }

    private function fetch(FilterRequestDataProvider $filterRequest) : FilterListDataProvider
    {
        $elasticSearchFilterRequestDataProvider = $this->filterForEsRequestInterface->get($fi
        $esFilterListDataProvider = $this->filterClient->search($elasticSearchFilterRequestDa
        ...
    }
}
```

Filter


```php
interface FilterClientInterface
{
    /**
     * @param \App\DataTransferObject\EsFilterDataProvider[] $esFilterRequestDataProvider
     * @return \App\DataTransferObject\EsFilterListDataProvider
     */
    public function search(array $esFilterRequestDataProvider) : EsFilterListDataProvider;
}
```

Filter


```php
interface FilterClientInterface
{
    /**
     * @param \App\DataTransferObject\EsFilterDataProvider[] $esFilterRequestDataProvider
     * @return \App\DataTransferObject\EsFilterListDataProvider
     */
    public function search(array $esFilterRequestDataProvider) : EsFilterListDataProvider;
}
```

```php
interface FilterClientInterface
{
    /**
     * @param \App\DataTransferObject\EsFilterDataProvider[] $esMustFilterRequestDataProvider
     * @param \App\DataTransferObject\EsFilterDataProvider[] $esMustNotFilterRequestDataProvider
     * @return \App\DataTransferObject\EsFilterListDataProvider
     */
    public function search(array $esMustFilterRequestDataProvider, array $esMustNotFilterRequestDataProvider) : EsFilterListDataPro
}
```

Filter


```php
interface FilterClientInterface
{
    /**
     * @param \App\DataTransferObject\EsFilterDataProvider[] $esFilterRequestDataProvider
     * @return \App\DataTransferObject\EsFilterListDataProvider
     */
    public function search(array $esFilterRequestDataProvider) : EsFilterListDataProvider;
}
```

```php
interface FilterClientInterface
{
    /**
     * @param \App\DataTransferObject\EsFilterDataProvider[] $esMustFilterRequestDataProvider
     * @param \App\DataTransferObject\EsFilterDataProvider[] $esMustNotFilterRequestDataProvider
     * @return \App\DataTransferObject\EsFilterListDataProvider
     */
    public function search(array $esMustFilterRequestDataProvider, array $esMustNotFilterRequestDataProvider) : EsFilterListDataPro
}
```

```php
interface FilterClientInterface
{
    /**
     * @param \App\DataTransferObject\EsFilterDataProvider[] $esMustFilterRequestDataProvider
     * @param \App\DataTransferObject\EsFilterDataProvider[] $esMustNotFilterRequestDataProvider
     * @param string $locale de|fr
     * @return \App\DataTransferObject\EsFilterListDataProvider
     */
    public function search(array $esMustFilterRequestDataProvider, array $esMustNotFilterRequestDataProvider, string $locale) : EsF
}
```

DTO und Semantic Versioning

Semantic Versioning 2.0.0 | Sema

semver.org

Semantic Versioning 2.0.0

Summary

Given a version number MAJOR.MINOR.PATCH, increment the:

1. MAJOR version when you make incompatible API changes,
2. MINOR version when you add functionality in a backwards compatible manner, and
3. PATCH version when you make backwards compatible bug fixes.

Additional labels for pre-release and build metadata are available as extensions to the MAJOR.MINOR.PATCH format.

Filter

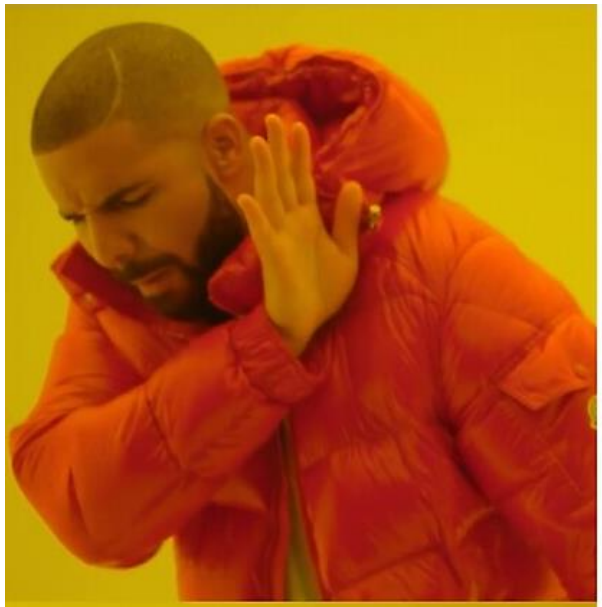

```php
namespace App\Client\Filter;

interface FilterClientInterface
{
    /**
     * @param \App\DataTransferObject\EsFilterDataProvider[] $esMustFilterRequestDataProvider
     * @param \App\DataTransferObject\EsFilterDataProvider[] $esMustNotFilterRequestDataProvider
     * @param string $locale de|fr
     * @return \App\DataTransferObject\EsFilterListDataProvider
     */
    public function search(array $esMustFilterRequestDataProvider, array $esMustNotFilterRequestDataProvider, string $locale) : EsF
}
```

```php
namespace App\Client\Filter;

class FilterClient implements FilterClientInterface
{
    /**
     * @param \App\DataTransferObject\EsFilterDataProvider[] $esMustFilterRequestDataProvider
     * @param \App\DataTransferObject\EsFilterDataProvider[] $esMustNotFilterRequestDataProvider
     * @param string $locale de|fr
     * @return \App\DataTransferObject\EsFilterListDataProvider
     */
    public function search(array $esMustFilterRequestDataProvider, array $esMustNotFilterRequestDataProvider, string $locale) : EsF
    {
        ...
    }
```

```php
namespace App\Client\Filter\Model;

interface FilterInterface
{
    /**
     * @param \App\DataTransferObject\EsFilterDataProvider[] $esMustFilterRequestDataProvider
     * @param \App\DataTransferObject\EsFilterDataProvider[] $esMustNotFilterRequestDataProvider
     * @param string $locale de|fr
     * @return \App\DataTransferObject\EsFilterListDataProvider
     */
    public function search(array $esMustFilterRequestDataProvider, array $esMustNotFilterRequestDataProvider, string $locale) : EsF
```

```php
namespace App\Client\Filter\Model;

class Filter implements FilterInterface
{
    /**
     * @param \App\DataTransferObject\EsFilterDataProvider[] $esMustFilterRequestDataProvider
     * @param \App\DataTransferObject\EsFilterDataProvider[] $esMustNotFilterRequestDataProvider
     * @param string $locale de|fr
     * @return \App\DataTransferObject\EsFilterListDataProvider
     */
    public function search(array $esMustFilterRequestDataProvider, array $esMustNotFilterRequestDataProvider, string $locale)
    {
```

Filter - ElasticSearchFilterRequestDataProvider



```php
namespace App\Client\Filter;

interface FilterClientInterface
{
    /**
     * @param \App\DataTransferObject\EsFilterDataProvider[] $esMustFilterRequestDataProvider
     * @param \App\DataTransferObject\EsFilterDataProvider[] $esMustNotFilterRequestDataProvider
     * @param string $locale de|fr
     * @return \App\DataTransferObject\EsFilterListDataProvider
     */
    public function search(array $esMustFilterRequestDataProvider, array $esMustNotFilterRequestDataProvider, string $locale) : EsF
}
```

```php
namespace App\Client\Filter

interface FilterInterface
{
    /**
     * @param \App\DataTransferObject\ElasticSearchFilterRequestDataProvider $esFilterRequestDataProvider
     * @return \App\DataTransferObject\EsFilterListDataProvider
     */
    public function search(ElasticSearchFilterRequestDataProvider $esFilterRequestDataProvider): EsFilterListDataPro
}
```

Filter - ElasticSearchFilterRequestDataProvider

```xml
<DataProvider name="ElasticSearchFilterRequest">
    <DataElement name="musts" type="EsFilter[]" singleton="Must"/>
    <DataElement name="mustsNot" type="EsFilter[]" singleton="MustNot"/>
    <DataElement name="locale" type="Locale"/>
</DataProvider>

<DataProvider name="EsFilter">
    <DataElement name="ident" type="string"/>
    <DataElement name="options" type="EsFilterOption[]" singleton="Option"/>
</DataProvider>

<DataProvider name="EsFilterOption">
    <DataElement name="value" type="string"/>
</DataProvider>

<DataProvider name="Locale">
    <DataElement name="countryCode" type="string" default="de"/>
</DataProvider>
```

```php
/**
 * Auto generated data provider
 */
final class ElasticSearchFilterRequestDataProvider extends \X
{
    /**
     * @return \App\DataTransferObject\EsFilterDataProvider[]
     */
    public function getMusts(): array
    {
        return $this->musts;
    }

    /**
     * @param \App\DataTransferObject\EsFilterDataProvider[]
     * @return ElasticSearchFilterRequestDataProvider
     */
    public function setMusts(array $musts)
    {
        $this->musts = $musts;

        return $this;
    }

    public function addMust(EsFilterDataProvider $Must)
    {
        $this->musts[] = $Must;

        return $this;
    }

    /**
     * @return bool
     */
    public function hasMusts()
```

Schichtenarchitektur

Communication Layer
Controllers, Console-Commands

↓

Business Layer
Entry class (Facade- or Proxy Pattern)

↓

Persistence Layer
Database (z.B. Doctrine Entity und Reposiotry)

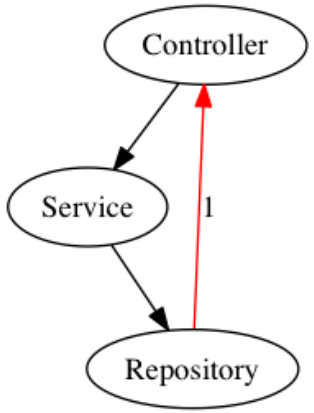
Deptrac

Continuous Integration passing

What is Deptrac

Deptrac is a static code analysis tool that helps to enforce rules for dependencies between software layers in your PHP projects.

For example, you can define a rule like "controllers may not depend on models". To ensure this, deptrac analyzes your code to find any usages of models in your controllers and will show you where this rule was violated.



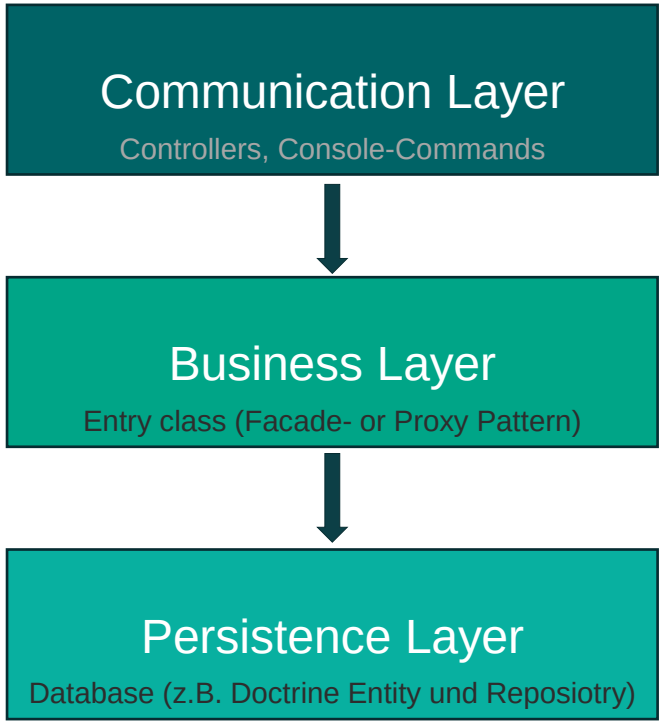
The Depfile

Let's have a look at the generated file:

```yaml
# depfile.yml
paths:
  - ./src
exclude_files:
  - .*test.*
layers:
  - name: Controller
    collectors:
      - type: className
        regex: .*Controller.*
  - name: Repository
    collectors:
      - type: className
        regex: .*Repository.*
  - name: Service
    collectors:
      - type: className
        regex: .*Service.*
ruleset:
  Controller:
    - Service
  Service:
    - Repository
  Repository: ~
```

https://github.com/sensiolabs-de/deptrac

Schichtenarchitektur und Deptrac




```yaml
# depfile.yml
paths:
  - ./src/
layers:
  - name: Communication
    collectors:
      - type: className
        regex: .*Communication.*
  - name: Business
    collectors:
      - type: className
        regex: .*Business.*
  - name: Persistence
    collectors:
      - type: className
        regex: .*Persistence.*
ruleset:
  Communication:
    - Business
  Business:
    - Persistence
  Persistence: ~
```

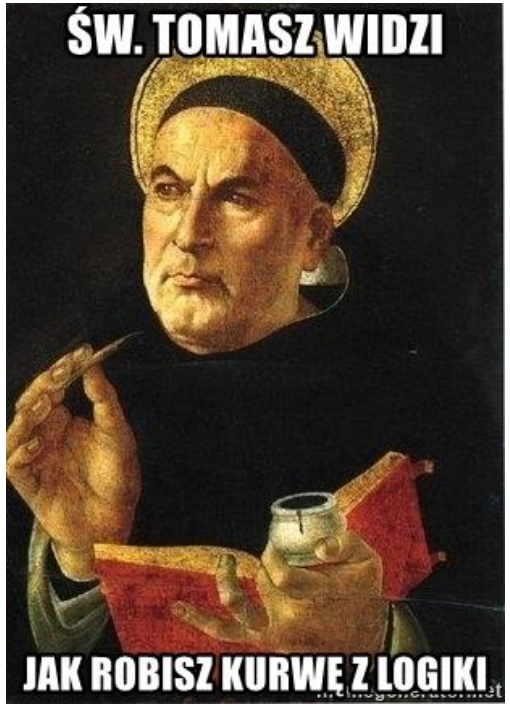
Schichtenarchitektur und Deptrac

Was liefert das Repository?:

Entity

Wo speichere ich das Entity?

Überall wo ich den \Doctrine\ORM\EntityManager habe




```php
// src/Controller/ProductController.php
namespace App\Controller;

// ...
use App\Entity\Product;
use Doctrine\ORM\EntityManagerInterface;
use Symfony\Component\HttpFoundation\Response;

class ProductController extends AbstractController
{
    /**
     * @Route("/product", name="create_product")
     */
    public function createProduct(): Response
    {
        // you can fetch the EntityManager via $this->getDoctrine()
        // or you can add an argument to the action: createProduct(EntityManagerInterface $
        $entityManager = $this->getDoctrine()->getManager();

        $product = new Product();
        $product->setName('Keyboard');
        $product->setPrice(1999);
        $product->setDescription('Ergonomic and stylish!');

        // tell Doctrine you want to (eventually) save the Product (no queries yet)
        $entityManager->persist($product);

        // actually executes the queries (i.e. the INSERT query)
        $entityManager->flush();

        return new Response('Saved new product with id '.$product->getId());
    }
}
```

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

Wo speichere ich das Entity?

Twig?

Wo speichere ich das Entity?

Google

Twig extension entity manager site:stackoverflow.com

Alle | Shopping | Bilder | News | Videos | Mehr | Einstellungen

Ungefähr 3.890 Ergebnisse (0,51 Sekunden) << Add Grepper Answer (a)

stackoverflow.com › questions › u... ▾ Diese Seite übersetzen

Unable to inject EntityManager into Twig extension - Stack ...

08.10.2012 - public function __construct(\Doctrine\ORM**EntityManager** $em, \XYZ\BlubBundl
\Utils\SessionHelper $sessionHelper) {. is wrong — you need ...

1 Antwort

stackoverflow.com › questions › d... ▾ Diese Seite übersetzen

Doctrine connection in Twig extension - Symfony - Stack ...

23.11.2017 - In AppBundle**Twig**\GetProvinceExtention.php: class **GetProvinceExtension** extends **Twig_Extension** { /** * @var **EntityManager** */ protected ...

3 Antworten

stackoverflow.com › questions › a... ▾ Diese Seite übersetzen

Access Doctrine Entity Manager in custom Twig Extension in ...

19.01.2015 - The only problem you have is the following: As mentioned in module.config.php#3 you cann add **Extension** via Factory, but the given ...

stackoverflow.com › questions › s... ▾ Diese Seite übersetzen

Symfony2 Using an Entity Manager within Twig extension ...

26.02.2013 - Making entities aware of any services — including **entity managers** — is a bad practice. There's nothing wrong about injecting an EM into a ...

1 Antwort

stackoverflow.com › questions › s... ▾ Diese Seite übersetzen

Symfony 4.1 twig extension - Stack Overflow

18.07.2018 - I even injected the default **entity manager** using autowire just because. namespace App**Twig**; use Doctrine\ORM ...

Deptrac

```yaml
# depfile.yml
paths:
  - ./src/
layers:
  - name: Communication
    collectors:
      - type: className
        regex: .*Communication.*
  - name: Business
    collectors:
      - type: className
        regex: .*Business.*
  - name: Persistence
    collectors:
      - type: className
        regex: .*Persistence.*
ruleset:
  Communication:
    - Business
  Business:
    - Persistence
  Persistence: ~
```

```yaml
# depfile.yml
paths:
  - ./src/
layers:
  - name: Communication
    collectors:
      - type: className
        regex: .*Communication.*
  - name: Business
    collectors:
      - type: className
        regex: .*Business.*
  - name: Persistence
    collectors:
      - type: className
        regex: .*Persistence.*
  - name: Entity
    collectors:
      - type: className
        regex: .*Entity.*
ruleset:
  Communication:
    - Business
  Business:
    - Persistence
  Persistence:
    - Entity
  Entity: ~
```

Repository

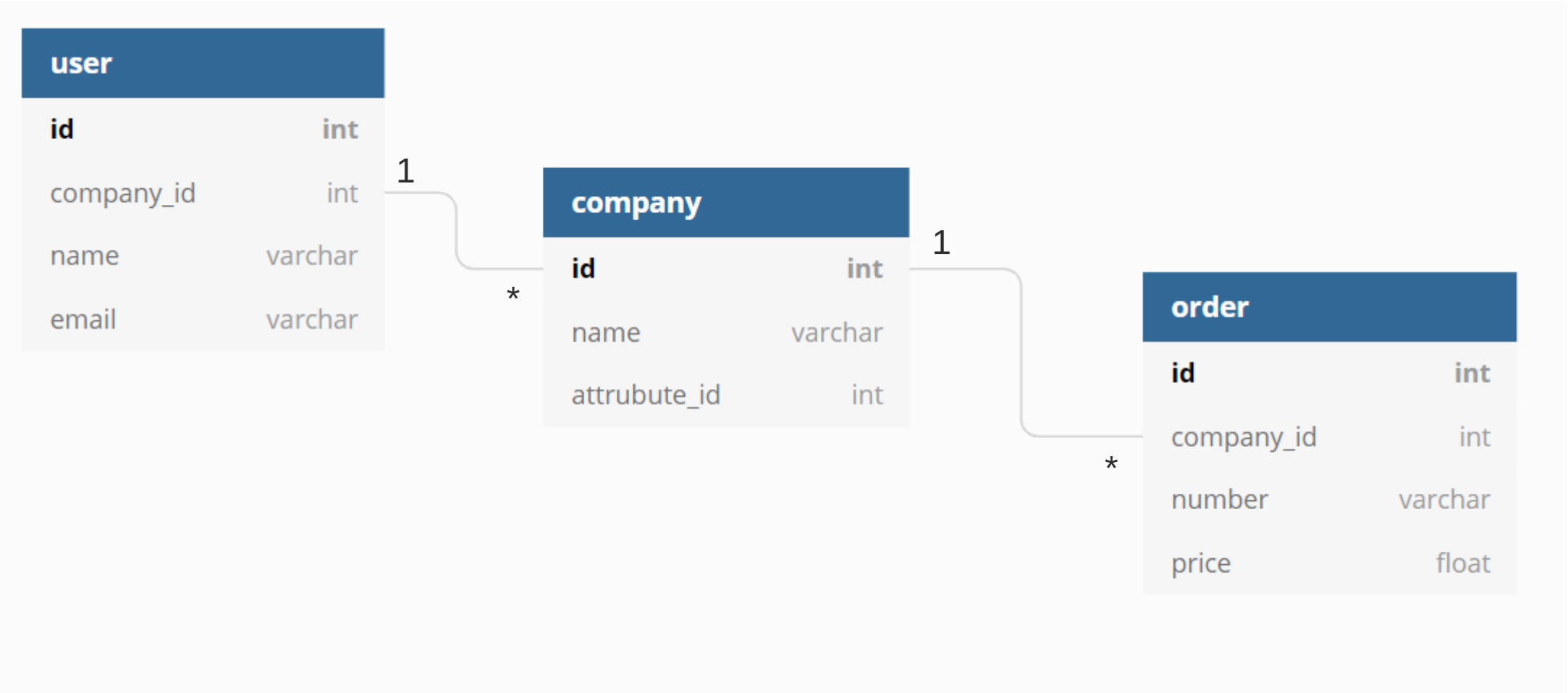
```
/**
 * @method User|null find($id, $lockMode = null, $lockVersion = null)
 * @method User|null findOneBy(array $criteria, array $orderBy = null)
 * @method User[]    findAll()
 * @method User[]    findBy(array $criteria, array $orderBy = null, $limit = null, $offset = nul
 */
class UserRepository extends ServiceEntityRepository
{
    /**
     * @var \App\Component\Symfony\EntityConverter\User
     */
    private $userConverter;

    public function __construct(
        ManagerRegistry $registry,
        \App\Component\Symfony\EntityConverter\User $userConverter
    )
    {
        parent::__construct($registry, User::class);
        $this->userConverter = $userConverter;
    }

    public function findByCustomerNumber(string $customerNumber): ?UserEntityDataProvider
    {
        return $this->fetchOneBy(['number' => $customerNumber]);
    }

    private function fetchOneBy(array $criteria, array $orderBy = null): ?UserEntityDataProvider
    {
        $entity = $this->findOneBy($criteria, $orderBy);
        if ($entity !== null) {
            return $this->userConverter->convert($entity);
        }

        return null;
    }
}
```

Entity Mapper / Converter

```php
namespace App\Symfony\EntityConverter;

class User implements UserInterface
{
    public function convert(Entity\User $user) : UserEntityDataProvider
    {
        $userEntityDataProvider = $this->userMapping->map($user, new UserEntityDataProvider());
        $company = $user->getCompany();

        $companyEntityDataProvider = $this->companyConverter->convert($company);
        $userEntityDataProvider->setCompany($companyEntityDataProvider);

        return $userEntityDataProvider;
    }
}
```

```php
namespace App\Symfony\EntityConverter\Mapping;

class User
{
    public function map(Entity\User $user, UserEntityDataProvider $userEntityDataProvider): UserEntityDataProvider
    {
        $userEntityDataProvider->setId($user->getId());
        $userEntityDataProvider->setEmail($user->getEmail());
        $userEntityDataProvider->setRole($user->getRole());
        $userEntityDataProvider->setPassword($user->getPassword());
        $userEntityDataProvider->setName($user->getName());
        $userEntityDataProvider->setStatus((int)$user->getStatus());
        $userEntityDataProvider->setNumber((string)$user->getNumber());

        return $userEntityDataProvider;
    }
}
```

Datenbank


user
id int
company_id int
name varchar
email varchar
1
*
company
id int
name varchar
attrubute_id int
1
*
order
id int
company_id int
number varchar
price float

Datenbank

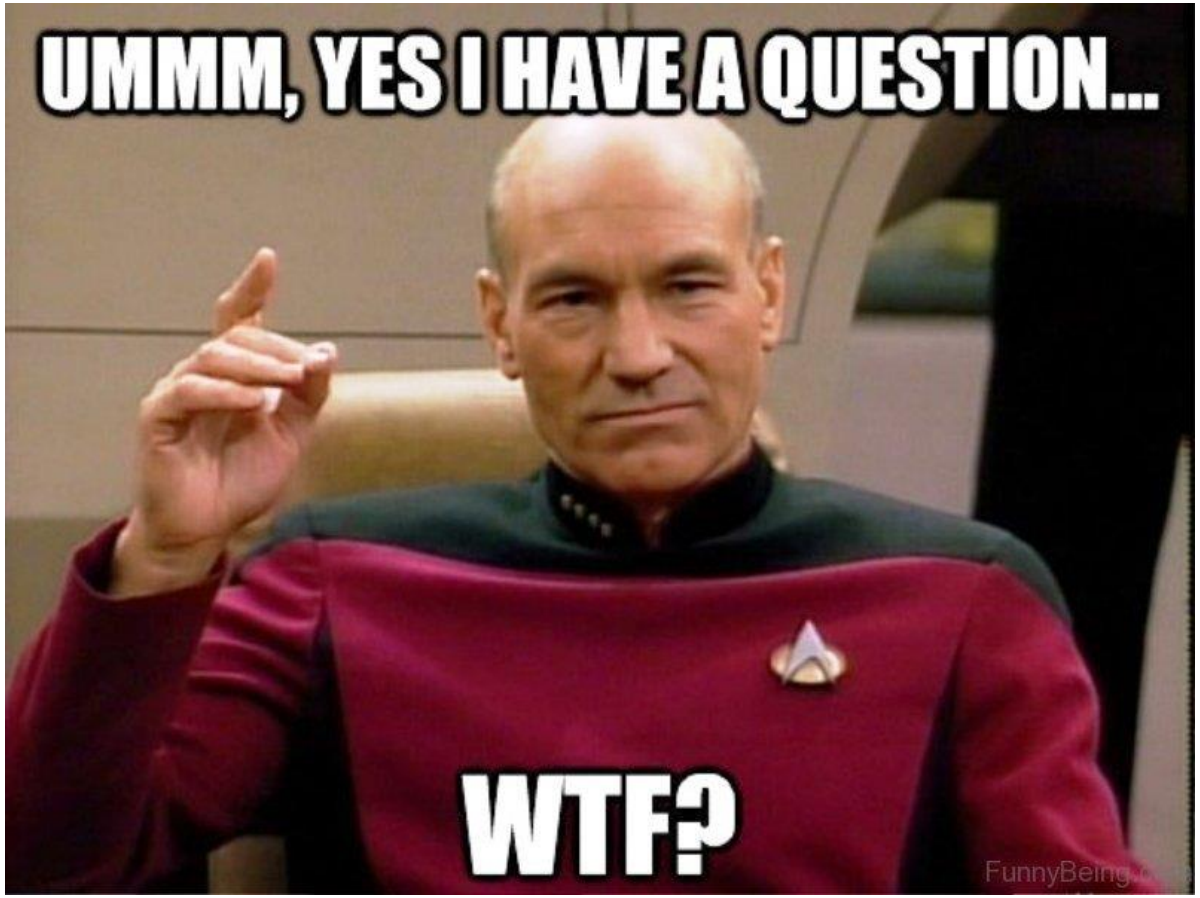
```php
/**
 * @ORM\Entity(repositoryClass="App\Repository\UserRepository")
 */
class User implements UserInterface
{
    ...

    /**
     * @ORM\ManyToOne(targetEntity="App\Entity\Company", inversedBy="company")
     * @ORM\JoinColumn(nullable=false)
     */
    private $company;

    ...

    public function getCompany(): ?Company
    {
        return $this->company;
    }

    public function setCompany(?Company $company): self
    {
        $this->company = $company;

        return $this;
    }

    ...
}
```

```php
/**
 * @ORM\Entity(repositoryClass="App\Repository\CompanyRepository")
 */
class Company
{
    ...

    /**
     * @ORM\OneToMany(targetEntity="App\Entity\User", mappedBy="company")
     */
    private $users;

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

    ...

    /**
     * @return Collection|User[]
     */
    public function getUsers(): Collection
    {
        return $this->users;
    }

    public function addUser(User $user): self
    {
        if (!$this->users->contains($user)) {
            $this->users[] = $user;
            $user->setCompany($this);
        }

        return $this;
    }

    public function removeUser(User $user): self
    {
```

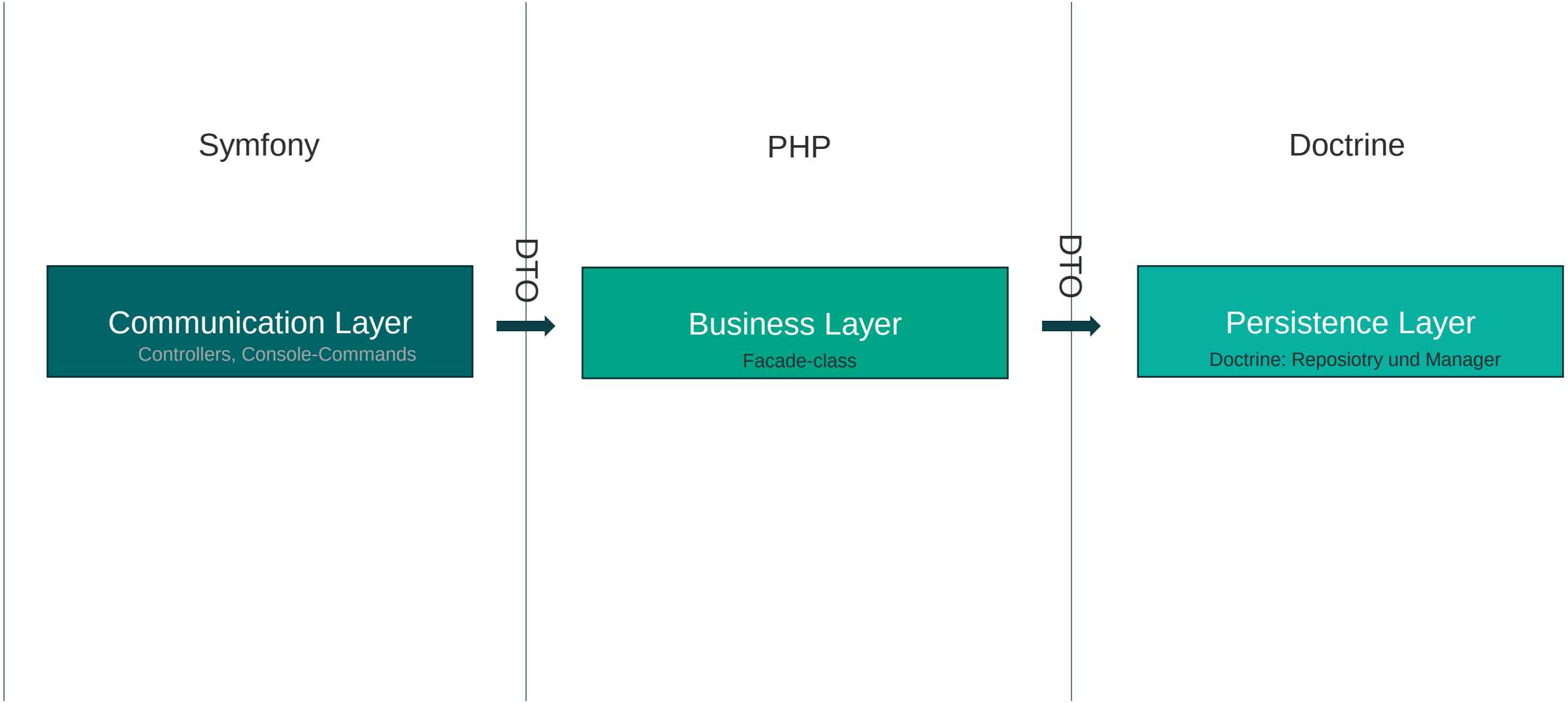
Datenbank

```
$userEntity->getCompany()->getUsers()[0]->getName();
```

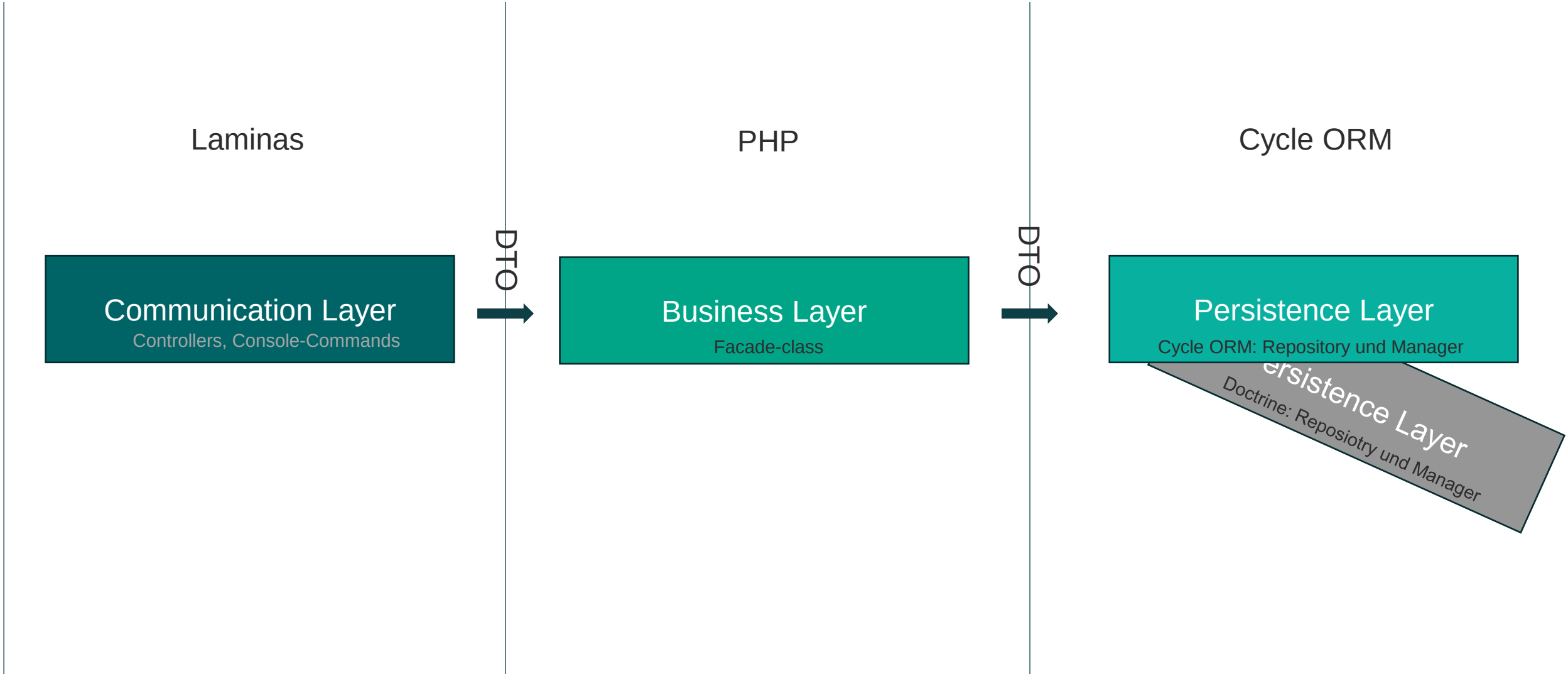
```
$orderEntity->getCompany()->getUsers()[0]->getName();
```



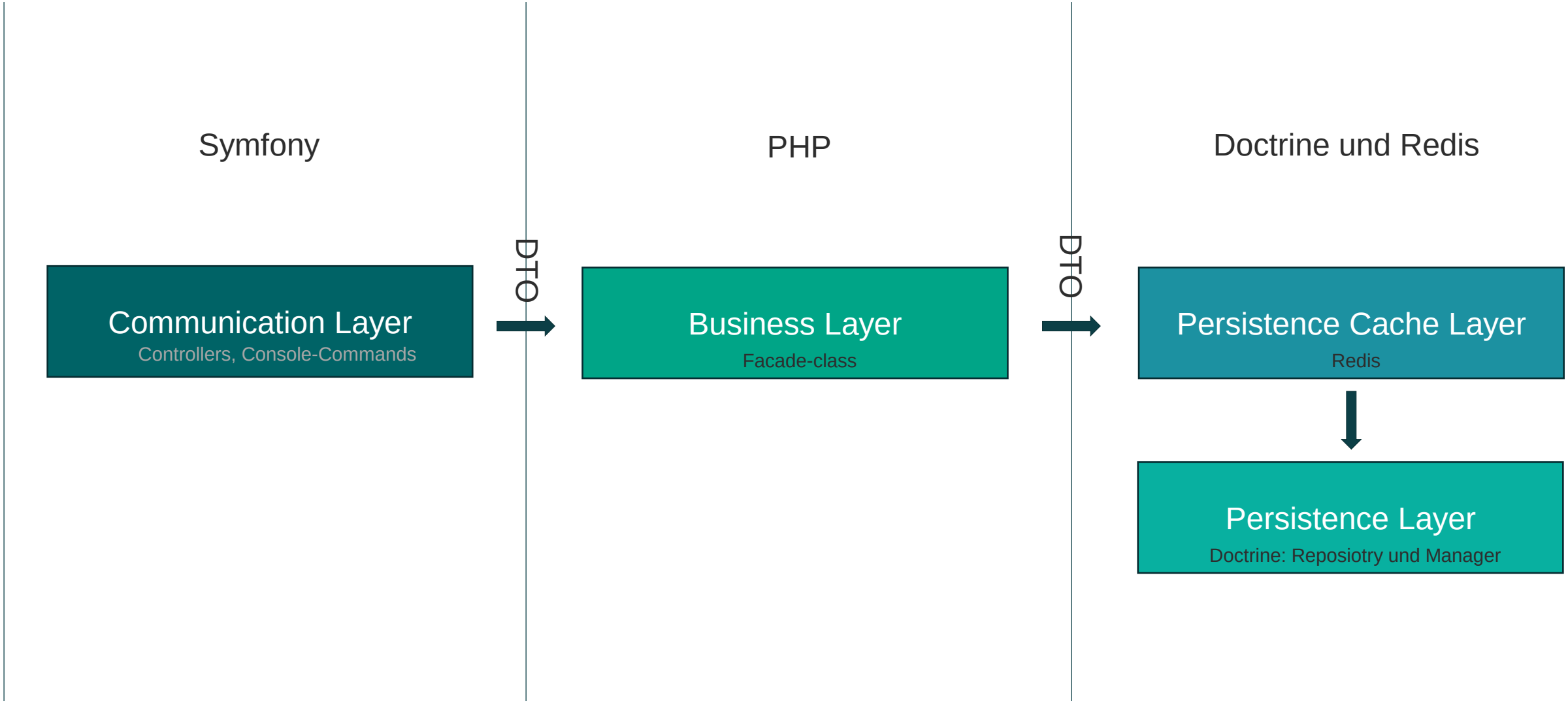
Schichtenarchitektur



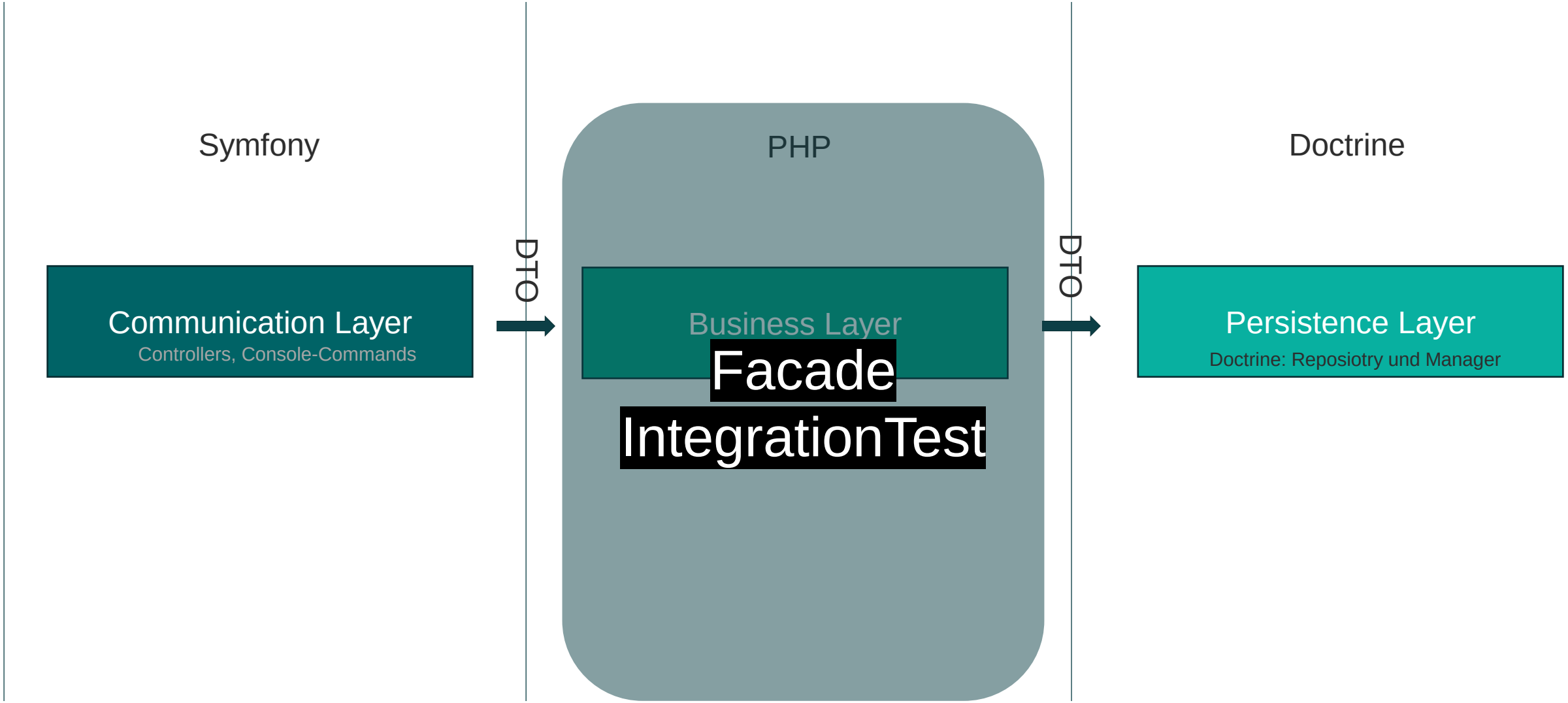
Schichtenarchitektur



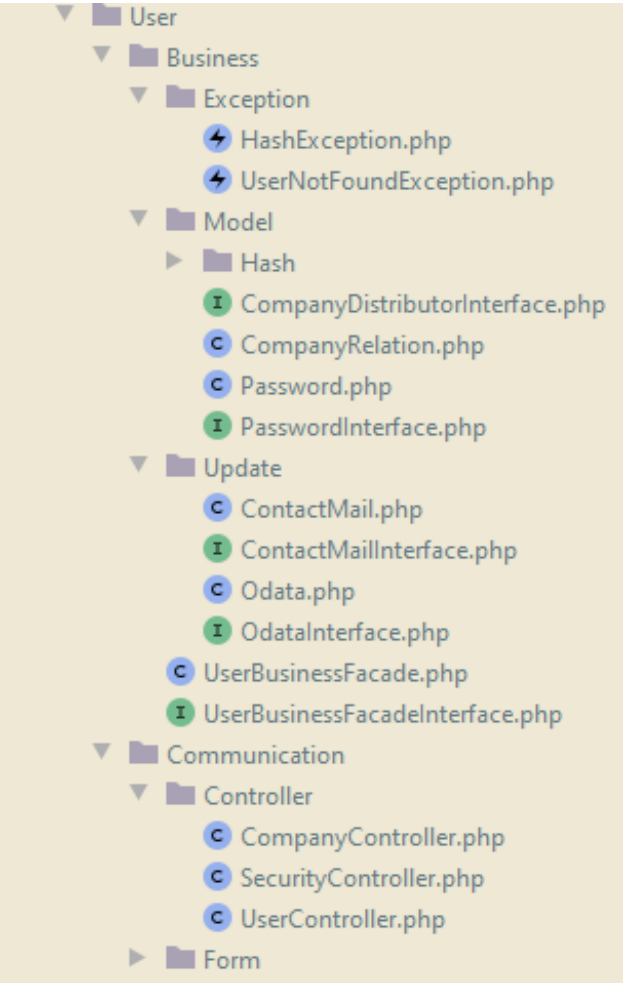
Schichtenarchitektur



Schichtenarchitektur



Schichtenarchitektur Plus

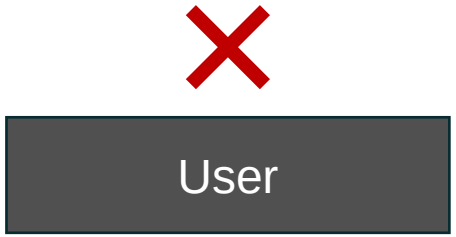


User-Modul Aufgaben:

- Passwort setzen / zurücksetzen
- Mail an User senden
- User in SAP aktualisieren (triggern)
- Zuweisung zwischen Firmen

Ist das SOLID?

Schichtenarchitektur Plus



User

User-Mail

User-Password

User-SAP

User-to-Company

Q&A