

☐ **Gruppe 1** DI (FH) G. Horn-V., MSc **Abgabetermin:** Sonntag, 07.06.2019, 24:00 Uhr

☒ **Gruppe 2** DI (FH) G. Horn-V., MSc **Name:** Angelos Angelis

**Aufwand (h):** 32

## Kommunikationsplattform „Faceblog“ mit PHP und MySQL

Zu entwickeln ist eine Kommunikationsplattform „Faceblog“, die einem registrierten Benutzer das Erstellen eines Blogs in Form von einzelnen Nachrichten erlaubt. Andere angemeldete Benutzer können die Blogeinträge einsehen und bei Gefallen ähnlich zu Facebook „ liken“. Dabei sind die nachstehend beschriebenen funktionalen und technischen Anforderungen umzusetzen.

### Funktionale Anforderungen

- Die Startseite von „Faceblog“ muss ohne Anmeldung zugänglich sein und soll ein paar allgemeine statistische Informationen über „Faceblog“ (z. B. Anzahl der registrierten Benutzer, Gesamtzahl aller erstellten Blogeinträge, Anzahl der neuen Blogeinträge in den letzten vierundzwanzig Stunden, Datum des letzten Beitrags, ...) anzeigen.
- Über eine Login-Seite kann sich ein bereits registrierter Benutzer unter Angabe von Benutzernamen und Passwort am System anmelden können.
- Über eine Registrierungs-Seite können neue Benutzer einen Login für das System erstellen. Dabei müssen mindestens Benutzernamen (zur Anmeldung am System, muss eindeutig sein), Passwort sowie ein Anzeigename (zur Anzeige, muss nicht eindeutig sein) angegeben werden.
- Eine Seite „My Blog“ zeigt einem angemeldeten Benutzer alle seine bisher geschriebenen Blogeinträge (gereiht nach Erstellungsdatum) und bietet ihm auch die Möglichkeit, einen neuen Blogeintrag zu schreiben. Dazu müssen mindestens ein Betreff und ein (beliebig langer) Text eingegeben werden.
- Eine weitere Seite „People“ ermöglicht es einem angemeldeten Benutzer, über einen Anzeigenamen nach anderen registrierten Personen zu suchen. Für alle bei einer solchen Suche gefundenen Personen ist neben Ihrem Anzeigenamen auch anzuzeigen, wie lange sie bereits Mitglied von „Faceblog“ sind.
- Durch Anklicken eines gefundenen Benutzers auf der Seite „People“ wird der entsprechende Blog dieses Benutzers (ähnlich zu „My Blog“) angezeigt.
- Benutzer von „Faceblog“ sollen die Möglichkeit haben, jeden Blogeintrag im System bei Gefallen – ähnlich zu Facebook – über eine Schaltfläche „Gefällt mir“ zu kennzeichnen bzw. diese Kennzeichnung auch wieder zurücknehmen zu können. Dies soll von allen Stellen aus möglich sein, wo ein Blogeintrag angezeigt wird.
- Überall, wo ein Blogeintrag angezeigt wird, sollen neben Erstellungsdatum, Betreff und Text des Eintrags – ebenfalls ähnlich zu Facebook – auch alle Benutzer gelistet werden, denen der Eintrag gefällt.
- Der Ersteller eines Blogeintrags soll jederzeit die Möglichkeit haben, diesen auch wieder zu löschen.

- Das nachträgliche Bearbeiten von einzelnen Blogeinträgen wäre natürlich auch sehr nützlich, muss aber nicht unbedingt implementiert werden.

## Technische Anforderungen

- Die Webseite ist mit PHP und unter Verwendung des Model-View-Controller-Entwurfsmusters zu realisieren.
- Anwendungslogik, Präsentationslogik und Infrastrukturcode müssen sinnvoll und sauber voneinander entkoppelt werden.
- Alle zu speichernden Daten sollen in einer MySQL-Datenbank abgelegt werden. Entwickeln Sie dazu ein entsprechendes Datenbankmodell und dokumentieren Sie es ausführlich (Diagramme etc.).
- Passwörter dürfen nur in ausreichend gesicherter Form in der Datenbank abgelegt werden.
- Die Webseite soll ein einheitliches und ansprechendes Layout bieten und muss auch auf mobilen Endgeräten gut verwendbar sein. Für die Umsetzung dieser Anforderung kann ein entsprechendes UI-Framework wie z. B. Bootstrap verwendet werden.
- Die gesamte Anwendung muss möglichst robust implementiert werden. Diverse Sicherheitsprüfungen dürfen z. B. auch durch das Aufrufen von Skripts mit abgeänderten Parametern oder durch manuell abgesetzte HTTP-Anfragen nicht umgangen werden können.
- Die endgültige Lösung muss auf der in der Übung verwendeten Version von XAMPP betrieben werden können.

## Organisatorische Anforderungen

- Die Projektarbeit ist in Einzelarbeit auszuführen.
- Für die Umsetzung sind nach Möglichkeit nur die in der Lehrveranstaltung vorgestellten Mittel zu verwenden.
- Die Projektarbeit ist spätestens bis zum oben aufgeführten Abgabetermin im Moodle-Kurs der SCR4-Übung hochzuladen und zu dem im Stundenplan vorgesehenen Termin in Form einer Live-Demonstration zu präsentieren.

## Abzugebende Komponenten

- Ausführliche Systemdokumentation als PDF-Datei, mindestens mit folgendem Inhalt:
  - Allgemeine Lösungsidee
  - Datenmodell und Erstellungsskript für Datenbank
  - Architektur und Struktur der Webseite
  - Abgedruckter Code der Webseite (HTML-Seiten, PHP-Skripts, Stylesheets)
  - Testfälle inklusive Screenshots
- Ordner der entwickelten Webseite mit allen relevanten Daten (HTML-Seiten, PHP-Skripts, Stylesheets, andere Ressourcen wie Bilder etc.)
- Datenbankskript zur Erstellung der Datenbank als Textdatei
- Datenbankskript zur Erstellung von Testdaten in der Datenbank, mit denen auch bei der Entwicklung bereits getestet wurde, als Textdatei

## SCR Projekt FaceBlog

### 1.Lösungsidee

Es wird ein wie in der Angabe beschrieben eine Kommunikationplattform erstellt. Hierbei wird die in der LVA verwendete Architektur benutzt.

Es gibt eine Startseite wo man sich ein paar allgemeine Statistiken anschauen kann. Ein nicht registrierter Nutzer kann sich nur diese Statistiken anschauen oder sich einloggen oder sich registrieren mehr aber auch nicht. Ein eingeloggter Nutzer kann dann alle Funktionen der Website benutzen.

Es wurden ebenfalls einige Klassen aus dem Bookshop benutzt.

#### Probleme:

Ich habe ein paar Problemen bei den Redirects wenn man einen Post von einem anderen Nutzer liked. Ansonsten habe ich alles implementiert wonach gefragt war 😊.

### 2.Datenbank

Die Datenbank besteht aus insgesamt 3 Tabellen:

#### 1.Blogs

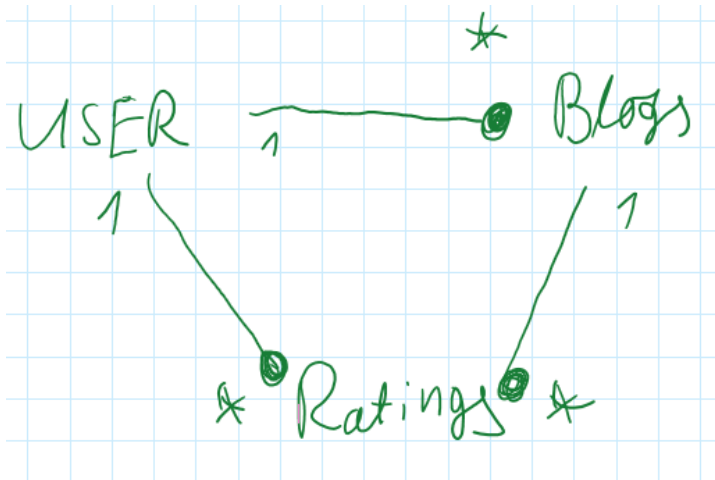
#	Name	Type	Collation	Attributes	Null	Default	Comments	Extra	Action
<input type="checkbox"/> 1	<b>blogsId</b> 🔑	int(11)			No	None		AUTO_INCREMENT	change  Drop  More
<input type="checkbox"/> 2	<b>usersId</b> 🔑	int(11)			No	None			change  Drop  More
<input type="checkbox"/> 3	<b>title</b>	varchar(255)	latin1_swedish_ci		No	None			change  Drop  More
<input type="checkbox"/> 4	<b>text</b>	text	latin1_swedish_ci		No	None			change  Drop  More
<input type="checkbox"/> 5	<b>date</b>	datetime			No	None			change  Drop  More

#### 2.Ratings

#	Name	Type	Collation	Attributes	Null	Default	Comments	Extra	Action
<input type="checkbox"/> 1	<b>ratingsId</b> 🔑	int(11)			No	None		AUTO_INCREMENT	change  Drop  More
<input type="checkbox"/> 2	<b>blogsId</b> 🔑	int(11)			No	None			change  Drop  More
<input type="checkbox"/> 3	<b>usersId</b> 🔑	int(11)			No	None			change  Drop  More
<input type="checkbox"/> 4	<b>rating</b>	tinyint(1)			Yes	NULL			change  Drop  More

#### 3.Users

#	Name	Type	Collation	Attributes	Null	Default	Comments	Extra	Action
<input type="checkbox"/> 1	<b>usersId</b> 🔑	int(11)			No	None		AUTO_INCREMENT	change  Drop  More
<input type="checkbox"/> 2	<b>displayName</b>	varchar(255)	latin1_swedish_ci		No	None			change  Drop  More
<input type="checkbox"/> 3	<b>userName</b> 🔑	varchar(255)	latin1_swedish_ci		No	None			change  Drop  More
<input type="checkbox"/> 4	<b>passwordHash</b>	char(60)	latin1_swedish_ci		No	None			change  Drop  More



### Skript

```

SET SQL_MODE = "NO_AUTO_VALUE_ON_ZERO";
SET AUTOCOMMIT = 0;
START TRANSACTION;
SET time_zone = "+00:00";
CREATE DATABASE IF NOT EXISTS `faceblog` DEFAULT CHARACTER SET latin1 COLLATE latin1_general_ci;
USE `faceblog`;

CREATE TABLE `blogs` (
  `blogsId` int(11) NOT NULL,
  `userId` int(11) NOT NULL,
  `title` varchar(255) NOT NULL,
  `text` TEXT NOT NULL,
  `date` DATETIME NOT NULL
) ENGINE=InnoDB DEFAULT CHARSET=latin1;

CREATE TABLE `users` (
  `userId` int(11) NOT NULL,
  `displayName` varchar(255) NOT NULL,
  `userName` varchar(255) NOT NULL,
  `passwordHash` char(60) NOT NULL
) ENGINE=InnoDB DEFAULT CHARSET=latin1;

CREATE TABLE `ratings` (
  `ratingsId` int(11) NOT NULL,
  `blogsId` int(11) NOT NULL,
  `userId` int(11) NOT NULL,
  `rating` BOOLEAN
) ENGINE=InnoDB DEFAULT CHARSET=latin1;

ALTER TABLE `blogs`
  ADD PRIMARY KEY (`blogsId`),
  ADD KEY `userId` (`userId`);

ALTER TABLE `users`
  ADD PRIMARY KEY (`userId`),
  ADD UNIQUE KEY `userName` (`userName`);

```

```
ALTER TABLE `ratings`  
  ADD PRIMARY KEY (`ratingsId`),  
  ADD KEY `userId` (`usersId`),  
  ADD KEY `blogsId` (`blogsId`);  
  
ALTER TABLE `blogs`  
  MODIFY `blogsId` int(11) NOT NULL AUTO_INCREMENT, AUTO_INCREMENT=30;  
ALTER TABLE `users`  
  MODIFY `usersId` int(11) NOT NULL AUTO_INCREMENT, AUTO_INCREMENT=5;  
ALTER TABLE `ratings`  
  MODIFY `ratingsId` int(11) NOT NULL AUTO_INCREMENT, AUTO_INCREMENT=1;  
  
ALTER TABLE `blogs`  
  ADD CONSTRAINT `blogs_ibfk_1` FOREIGN KEY (`usersId`) REFERENCES `users`  
  (`usersId`) ON DELETE CASCADE ON UPDATE CASCADE;  
  
ALTER TABLE `ratings`  
  ADD CONSTRAINT `ratings_ibfk_1` FOREIGN KEY (`usersId`) REFERENCES `users`  
  (`usersId`) ON DELETE CASCADE ON UPDATE CASCADE,  
  ADD CONSTRAINT `ratings_ibfk_2` FOREIGN KEY (`blogsId`) REFERENCES `blogs`  
  (`blogsId`) ON DELETE CASCADE ON UPDATE CASCADE;  
  
COMMIT;
```

### 3. Architektur

Es wird die zur Verfügung gestellte Dependency Injection genutzt, mithilfe des ServiceProvider.

#### 3.1. Repositories

Es wird das Repository Pattern eingesetzt, allerdings wird wie in der Übung jedes Interface von einer einzelnen Klasse implementiert.

#### 3.2. Queries and Commands

Als Zwischenschicht für Repositories und Controller werden Queries und Commands genutzt. Queries sind für einfache Abfragen und Commands sind Klassen die etwas verändern

#### 3.3. Entities

Bilden die Datenbank in PHP-Klassen ab.

#### 3.4. Views

Html Code dienen zur Darstellung der Website

#### 3.4. Services

Es gibt einen AuthenticationService, welcher die Login-Session des Users verwaltet.

So funktioniert grob das Zusammenspiel der Schichten:

Die Controller holen sich mittels Queries Daten aus dem Repository und geben diese an die Views weiter.

## 4. Quellcode

Css:

*Bootstrap.min.css : Im pdf nicht inkludiert*

*Index.css:*

```
.panel-shadow {
    box-shadow: rgba(0, 0, 0, 0.3) 7px 7px 7px;
}
.panel-white {
    border: 1px solid #dddddd;
}
.panel {
    font-size: 13px;
    color: #454545;
    background: #fafafa;
    overflow-x: hidden;
    font-family: 'Source Sans Pro', 'Oxygen', sans-serif;
}
.panel-white .panel-heading {
    color: #333;
    background-color: #fff;
    border-color: #ddd;
}
.panel-white .panel-footer {
    background-color: #fff;
    border-color: #ddd;
}
.bg2 {
    background-image:
url('https://via.placeholder.com/560x344/FF4500/000000');
}
.profile-widget {
    position: center;
}
.profile-widget .image-container {
    background-size: cover;
    background-position: center;
    padding: 190px 0 10px;
}
.profile-widget .image-container .profile-background {
    width: 100%;
    height: auto;
}
.profile-widget .image-container .avatar {
    width: 120px;
    height: 120px;
    border-radius: 50%;
    margin: 0 auto -60px;
    display: block;
}
.profile-widget .details {
    padding: 50px 15px 15px;
    text-align: center;
}
```

Js:

*Bootstrap.js: Nicht im pdf inkludiert*

Entities:

*Blogs.php:*

```
<?php
namespace Application\Entities;

class Blogs {
    public function __construct(
        private int $blogsId,
        private int $users,
        private string $title,
        private string $text,
        private string $date
    )
    { }

    /**
     * @return int
     */
    public function getUsers(): int
    {
        return $this->users;
    }

    /**
     * @return int
     */
    public function getBlogsId(): int
    {
        return $this->blogsId;
    }

    /**
     * @return string
     */
    public function getTitle(): string
    {
        return $this->title;
    }

    /**
     * @return string
     */
    public function getText(): string
    {
        return $this->text;
    }

    /**
     * @return string
     */
    public function getDate(): string
```

```
{
    return $this->date;
}

}
Ratings.php
<?php

namespace Application\Entities;

class Ratings {
    public function __construct(
        private int $ratingsId,
        private int $blogsId,
        private int $usersId,
        private int $rating
    )
    { }

    /**
     * @return int
     */
    public function getRatingsId(): int
    {
        return $this->ratingsId;
    }

    /**
     * @return int
     */
    public function getRating(): int
    {
        return $this->rating;
    }

    /**
     * @return int
     */
    public function getBlogsId(): int
    {
        return $this->blogsId;
    }

    /**
     * @return int
     */
    public function getUsersId(): int
    {
        return $this->usersId;
    }
}
```



*User.php:*

```
<?php

namespace Application\Entities;

class User {
    public function __construct(
        private int $usersId,
        private string $displayName,
        private string $userName,
        private string $passwordHash
    )
    {

    }

    /**
     * @return int
     */
    public function getUsersId(): int
    {
        return $this->usersId;
    }

    /**
     * @return string
     */
    public function getDisplayName(): string
    {
        return $this->displayName;
    }

    /**
     * @return string
     */
    public function getUserName(): string
    {
        return $this->userName;
    }

    /**
     * @return string
     */
    public function getPasswordHash(): string
    {
        return $this->passwordHash;
    }
}
```

Interfaces:

*BlogsRepository.php:*

```
<?php

namespace Application\Interfaces;
use Application\Entities\Blogs;
```

```
interface BlogsRepository {
    public function getBlogsForUser($usersId): array;
    public function getBlogById($blogsId): Blogs;
    public function getBlogsCount(): ?int;
    public function getBlogs24Count(): ?int;
    public function getLatestBlog(): ?string;
    public function insertBlog(int $usersId, string $title, string $text,
string $date): int;
    public function deleteBlog($blogsId): int;
}
```

*RatingsRepository.php:*

```
<?php
```

```
namespace Application\Interfaces;
```

```
interface RatingsRepository {
    public function getRatings() : array;
    public function getLikedBy(int $bid) : array;
    public function insertRating(int $blogsId, int $usersId, int $rating) :
int;
    public function deleteRating(int $rid): int;
}
```

*Session.php:*

```
<?php
```

```
namespace Application\Interfaces;
```

```
interface Session {
    public function get(string $key): mixed; //mixed = Alles erlaubt
    public function put(string $key, mixed $value): void;
    public function delete(string $key): void;
}
```

*UserRepository:*

```
<?php
```

```
namespace Application\Interfaces;
```

```
interface UserRepository {
    public function getUser(int $id): ?\Application\Entities\User; //? heißt
"oder null"
    public function getUserForUserName(string $userName):
?\Application\Entities\User;
    public function getUserCount(): ?int;
    public function insertUser(string $displayName, string $username, string
$password): int;
    public function getUsersForDisplayname(string $dnm): ?array;
}
```

Services:

*AuthenticationService.php:*

**<?php**

**namespace** Application\Services;

```
class AuthenticationService {    // kein http Token oda so
    public function __construct(
        private \Application\Interfaces\Session $session
    ) {}
```

```
    const KEY = 'userId';
```

```
    public function getUserId(): ?int {
        return $this->session->get(self::KEY);
    }
```

```
    public function signIn(int $userId): void {
        $this->session->put(self::KEY, $userId);
    }
```

```
    public function signOut(): void {
        $this->session->delete(self::KEY);
    }
}
```

Application:

*BlogByIdQuery.php:*

**<?php**

**namespace** Application;

**use** Application\Entities\Blogs;

```
class BlogByIdQuery {
    public function __construct(
        private \Application\Interfaces\BlogsRepository $blogsRepository
    )
    { }
```

```
    public function execute(string $blogsId) : Blogs {
        $blogs = $this->blogsRepository->getBlogById($blogsId);
        $res = null;
        if (isset($blogs)) {
            $res = new Blogs(
                $blogs->getBlogsId(),
                $blogs->getUsers(),
                $blogs->getTitle(),
                $blogs->getText(),
                $blogs->getDate(),
            );
        }
        return $res;
    }
}
```

}

*Blogs24Query.php:*

```
<?php

namespace Application;
class Blogs24Query {
    public function __construct(
        private Interfaces\BlogsRepository $blogsRepository
    ) {

    }

    public function execute(): int {
        return $this->blogsRepository->getBlogs24Count();
    }
}
```

*BlogsCount.php:*

```
<?php

namespace Application;
class BlogsCount {
    public function __construct(
        private Interfaces\BlogsRepository $blogsRepository
    ) {

    }

    public function execute(): int {
        return $this->blogsRepository->getBlogsCount();
    }
}
```

*BlogsData.php:*

```
<?php

namespace Application;

class BlogsData {
    public function __construct(
        private int $blogsId,
        private string $title,
        private string $text,
        private string $date,
        private int $rating,
        private array $likedBy
    )
    {
    }

    /**
     * @return array
     */
    public function getLikedBy(): array
    {
        return $this->likedBy;
    }

    /**
```

```
        * @return int
        */
        public function getRating(): int
        {
            return $this->rating;
        }

        /**
         * @return int
         */
        public function getUsersId(): int
        {
            return $this->usersId;
        }

        /**
         * @return int
         */
        public function getBlogsId(): int
        {
            return $this->blogsId;
        }

        /**
         * @return string
         */
        public function getTitle(): string
        {
            return $this->title;
        }

        /**
         * @return string
         */
        public function getText(): string
        {
            return $this->text;
        }

        /**
         * @return string
         */
        public function getDate(): string
        {
            return $this->date;
        }
    }
}
```

*BlogsQuery.php:*

```
<?php
```

```
namespace Application;
// Entwurfsmuster: CQRS
```

```
class BlogsQuery {
```

```

    public function __construct(
        private \Application\Interfaces\BlogsRepository $blogsRepository,
        private \Application\LikedByQuery $likedByQuery,
        private \Application\Interfaces\RatingsRepository $ratingsRepository
    )
    { }

    public function execute(string $usersId) : array {
        $blogs = $this->blogsRepository->getBlogsForUser($usersId);
        $ratings = $this->ratingsRepository->getRatings();
        $res = [];
        foreach($blogs as $b) {
            $rating = 0;
            foreach($ratings as $r) {
                if ($r->getBlogsId() == $b->getBlogsId()){
                    $rating = $r->getRating();
                }
            }
            $res[] = new BlogsData($b->getBlogsId(), $b->getTitle(), $b->
>getText(), $b->getDate(), $rating, $this->likedByQuery->execute($b->
>getBlogsId()));
        }
        return $res;
    }
}

```

CreatePostComman.php:

<?php

```

namespace Application;

class CreatePostCommand {
    public function __construct(
        private Interfaces\BlogsRepository $blogsRepository,
        private SignedInUserQuery $signedInUserQuery
    ) {
    }

    public function execute(int $usersId, string $title, string $text, string
$date): ?int {
        $user = $this->signedInUserQuery->execute();
        if (!isset($user)) {
            return null;
        }
        return $this->blogsRepository->insertBlog($user->getUsersId(),
$title, $text, $date);
    }
}

```

*CreateRatingCommand.php:*

**<?php**

**namespace** Application;

```
class CreateRatingCommand {
    public function __construct(
        private Interfaces\RatingsRepository $ratingsRepository,
        private SignedInUserQuery $signedInUserQuery
    ) {
    }

    public function execute(int $blogsId, int $usersId, int $rating): ?int {
        return $this->ratingsRepository->insertRating($blogsId,$usersId,$rating);
    }
}
```

*DeleteBlogCommand.php:*

**<?php**

**namespace** Application;

```
class DeleteBlogCommand {
    public function __construct(
        private Interfaces\BlogsRepository $blogsRepository
    ) {
    }

    public function execute(int $bid): bool {
        return $this->blogsRepository->deleteBlog($bid);
    }
}
```

*DeleteRatingCommand.php:*

**<?php**

**namespace** Application;

```
class DeleteRatingCommand {
    public function __construct(
        private Interfaces\RatingsRepository $ratingsRepository
    ) {
    }

    public function execute(int $rid): bool {
        return $this->ratingsRepository->deleteRating($rid);
    }
}
```

*LatestBlogDateQuery.php:*

```
<?php

namespace Application;
class LatestBlogDateQuery {
    public function __construct(
        private Interfaces\BlogsRepository $blogsRepository
    ) {

    }

    public function execute(): ?string {
        $res = $this->blogsRepository->getLatestBlog();
        if (isset($res)) {
            return $res;
        }
        return "No blogs yet";
    }
}
```

*LikedByQuery.php:*

```
<?php

namespace Application;

class LikedByQuery {
    public function __construct(
        private \Application\Interfaces\RatingsRepository $ratingsRepository
    ) {
    }

    public function execute(int $bid) : array {
        $count = $this->ratingsRepository->getLikedBy($bid);
        return $count;
    }
}
```

*PeopleQuery.php:*

```
<?php
namespace Application;

use Application\Interfaces\UserRepository;

class PeopleQuery {
    public function __construct(
        private UserRepository $userRepository
    ) {
    }

    public function execute(?string $dnm): array {
        $res = [];
        if (!isset($dnm)) {
            return $res;
        }
        $people = $this->userRepository->getUsersForDisplayname($dnm);
    }
}
```



```

        foreach ($people as $p) {
            $res[] = new UserData(
                $p->getUsersId(),
                $p->getDisplayname(),
                $p->getUserName()
            );
        }
        return $res;
    }
}

```

*RatingsQuery.php:*

**<?php**

```

namespace Application;
// Entwurfsmuster: CQRS

class RatingsQuery {
    public function __construct(
        private \Application\Interfaces\RatingsRepository $ratingsRepository
    )
    { }

    public function execute() : array {
        $count = $this->ratingsRepository->getRatings();
        return $count;
    }
}

```

*RegisterCommand.php:*

**<?php**

```

namespace Application;

class RegisterCommand {

    public function __construct(
        private Interfaces\UserRepository $userRepository
    )
    {}

    public function execute(string $displayName, string $username, string
$password): ?int {
        if ($this->userRepository->getUserForUserName($username) != null) {
            return null;
        }
        return $this->userRepository->insertUser($displayName,$username,
$password);
    }
}

```

*SignedInUserQuery.php:*

```
<?php

namespace Application;

class SignedInUserQuery
{
    public function __construct(
        private Services\AuthenticationService $authenticationService,
        private Interfaces\UserRepository $userRepository
    ) {
    }

    public function execute(): ?UserData
    {
        $id = $this->authenticationService->getUserid();
        if ($id === null) {
            return null;
        }
        $user = $this->userRepository->getUser($id);
        if ($user === null) {
            return null;
        }
        return new UserData($user->getUsersId(), $user->getDisplayName(),
            $user->getUserName());
    }
}
```

*SignInComman.php:*

```
<?php

namespace Application;

class SignInCommand {
    public function __construct(
        private Services\AuthenticationService $authenticationService,
        private Interfaces\UserRepository $userRepository
    )
    {
    }

    public function execute(string $userName, string $password): bool {
        $this->authenticationService->signOut();
        $user = $this->userRepository->getUserForUserName($userName);
        if ($user != null && password_verify($password, $user->
            getPasswordHash())) { //password_hash erstellt aus einem Passwort einen
            Hash
                $this->authenticationService->signIn($user->getUsersId());
            //password_verify stellt ein vorheriges password_hash voraus
                return true;
            }
        return false;
    }
}
```

*SignInCommand.php:*

```
<?php

namespace Application;

class SignInCommand {
    public function __construct(
        private Services\AuthenticationService $authenticationService
    )
    {}

    public function execute(): void {
        $this->authenticationService->SignIn();
    }
}
```

*UserByDisplaynameQuery.php:*

```
<?php

namespace Application;

use Application\Entities\User;
use Application\Interfaces\UserRepository;

class UserByDisplaynameQuery {
    public function __construct(private UserRepository $userRepository) {

    }

    public function execute(?string $dnm): ?array{
        $user = $this->userRepository->getUsersForDisplayname($dnm);
        $userDTO = [];
        foreach ($user as $p) {
            $userDTO[] = new User($user->getUsersId(), $user->getDisplayname(), $user->getUsername(), $user->getPasswordHash());
        }
        return $userDTO;
    }
}
```

*UserCount.php:*

```
<?php

namespace Application;

class UserCount {
    public function __construct(
        private Interfaces\UserRepository $userRepository
    ) {
    }
}
```

```
    public function execute(): int {  
        return $this->userRepository->getUserCount();  
    }  
}
```

*UserData.php:*

**<?php**

**namespace** Application;

**class** UserData

```
{  
    public function __construct(  
        private int $usersId,  
        private string $displayName,  
        private string $userName  
    ) {  
    }  
  
    /**  
     * @return int  
     */  
    public function getUsersId(): int  
    {  
        return $this->usersId;  
    }  
  
    /**  
     * @return string  
     */  
    public function getDisplayName(): string  
    {  
        return $this->displayName;  
    }  
  
    /**  
     * @return string  
     */  
    public function getUserName(): string  
    {  
        return $this->userName;  
    }  
}
```

Infrastructure:

*Repository.php:*

**<?php**

**namespace** Infrastructure;

```
use Application\Entities\Blogs;  
use Application\Entities\Ratings;  
use Application\Entities\User;
```

```
use Application\UserData;

class Repository
implements
    \Application\Interfaces\BlogsRepository,
    \Application\Interfaces\RatingsRepository,
    \Application\Interfaces\UserRepository
{
    private $server;
    private $displayName;
    private $userName;
    private $password;
    private $database;

    public function __construct(string $server, string $displayName, string
$userName, string $password, string $database)
    {
        $this->server = $server;
        $this->displayName = $displayName;
        $this->userName = $userName;
        $this->password = $password;
        $this->database = $database;
    }

    // === private helper methods ===

    private function getConnection()
    {
        $con = new \mysqli($this->server, $this->userName, $this->password,
$this->database);
        if (!$con) {
            die('Unable to connect to database. Error: ' .
mysqli_connect_error());
        }
        return $con;
    }

    private function executeQuery($connection, $query)
    {
        $result = $connection->query($query);
        if (!$result) {
            die("Error in query '$query': " . $connection->error);
        }
        return $result;
    }

    private function executeStatement($connection, $query, $bindFunc)
    {
        $statement = $connection->prepare($query);
        if (!$statement) {
            die("Error in prepared statement '$query': " . $connection-
>error);
        }
        $bindFunc($statement);
        if (!$statement->execute()) {
            die("Error executing prepared statement '$query': " . $statement-
>error);
        }
    }
}
```

```

    }
    return $statement;
}

// === public methods ===

public function insertUser(string $displayName, string $username, string
$password): int {
    $hash = password_hash($password, PASSWORD_DEFAULT);
    $conn = $this->getConnection();
    $statement = $this->executeStatement(
        $conn,
        'INSERT INTO users (displayName, username, passwordHash) VALUES
(?, ?, ?)',
        function (\mysqli_stmt $stmt) use ($displayName, $username,
$hash) {
            $stmt->bind_param('sss', $displayName, $username, $hash);
        }
    );
    return $statement->insert_id;
}

public function getBlogsForUser($userId): array
{
    $Blogs = [];
    $con = $this->getConnection();
    $stat = $this->executeStatement(
        $con,
        'SELECT blogsId, userId, title, text, date FROM blogs WHERE
userId = ?',
        function ($s) use ($userId) {
            $s->bind_param('i', $userId);
        }
    );
    $stat->bind_result($blogsId, $userId, $title, $text, $date);
    while ($stat->fetch()) {
        $Blogs[] = new \Application\Entities\Blogs($blogsId, $userId,
$title, $text, $date);
    }
    $stat->close();
    $con->close();
    return $Blogs;
}

public function getBlogsCount(): ?int
{
    $count = null;
    $con = $this->getConnection();
    $res = $this->executeQuery(
        $con,
        'SELECT COUNT(*) as cnt FROM blogs'
    );
    if($row = $res->fetch_object()){
        $count = $row->cnt;
    }
    $res->close();
    $con->close();
}

```

```

        return $count;
    }

    public function getUserCount(): ?int
    {
        $count = null;
        $con = $this->getConnection();
        $res = $this->executeQuery(
            $con,
            'SELECT COUNT(*) as cnt FROM users'
        );
        if($row = $res->fetch_object()){
            $count = $row->cnt;
        }
        $res->close();
        $con->close();
        return $count;
    }

    public function getUser(int $id): ?\Application\Entities\User
    {
        $user = null;
        $con = $this->getConnection();
        $stat = $this->executeStatement(
            $con,
            'SELECT usersId, displayName, userName, passwordHash FROM users
WHERE usersId = ?',
            function ($s) use ($id) {
                $s->bind_param('i', $id);
            }
        );
        $stat->bind_result($usersId, $displayName, $userName, $passwordHash);
        if ($stat->fetch()) {
            $user = new \Application\Entities\User($usersId, $displayName,
            $userName, $passwordHash);
        }
        $stat->close();
        $con->close();
        return $user;
    }

    public function getUserForUserName(string $userName):
    ?\Application\Entities\User
    {
        $user = null;
        $con = $this->getConnection();
        $stat = $this->executeStatement(
            $con,
            'SELECT usersId, displayName, userName, passwordHash FROM users
WHERE userName = ?',
            function ($s) use ($userName) {
                $s->bind_param('s', $userName);
            }
        );
        $stat->bind_result($usersId, $displayName, $userName, $passwordHash);
        if ($stat->fetch()) {
            $user = new \Application\Entities\User($usersId, $displayName,

```

```

$username, $passwordHash);
    }
    $stat->close();
    $con->close();
    return $user;
}

public function getRatings(): array
{
    $ratings = [];
    $con = $this->getConnection();
    $res = $this->executeQuery(
        $con,
        'SELECT ratings.ratingsId as rid, ratings.blogsId as
bid, ratings.usersId as uid, COUNT(ratings.ratingsId) AS cnt FROM blogs RIGHT
JOIN ratings ON blogs.blogsId = ratings.blogsId GROUP BY blogs.blogsId'
    );
    while($row = $res->fetch_object()){
        $ratings[] = new Ratings($row->rid, $row->bid, $row->uid, $row-
>cnt);
    }
    $res->close();
    $con->close();
    return $ratings;
}

public function insertBlog(int $usersId, string $title, string $text,
string $date): int
{
    $conn = $this->getConnection();
    $newDate = date('Y-m-d h:i:s', strtotime($date));
    $statement = $this->executeStatement(
        $conn,
        'INSERT INTO blogs (usersId, title, text, date) VALUES (?, ?, ?,
?)',
        function (\mysqli_stmt $stmt) use ($usersId, $title, $text,
$newDate) {
            $stmt->bind_param('iss', $usersId, $title, $text, $newDate);
            $stmt->bind_param('iss', $usersId, $title, $text, $newDate);
        }
    );
    return $statement->insert_id;
}

public function getBlogById($blogsId): Blogs
{
    $blog = null;
    $con = $this->getConnection();
    $stat = $this->executeStatement(
        $con,
        'SELECT blogsId, usersId, title, text, date FROM blogs WHERE
blogsId = ?',
        function ($s) use ($blogsId) {
            $s->bind_param('i', $blogsId);
        }
    );
    $stat->bind_result($blogsId, $usersId, $title, $text, $date);
}

```



```

        if ($stat->fetch()) {
            $blog = new \Application\Entities\Blogs($blogsId, $usersId,
$title, $text, $date);
        }
        $stat->close();
        $con->close();
        return $blog;
    }

    public function deleteBlog($blogsId): int
    {
        $conn = $this->getConnection();
        $statement = $this->executeStatement(
            $conn,
            'DELETE FROM blogs WHERE blogsId = ?',
            function (\mysqli_stmt $stmt) use ($blogsId) {
                $stmt->bind_param("i", $blogsId);
            }
        );
        return $statement->affected_rows > 0;
    }

    public function getUsersForDisplayname(?string $dnm): ?array
    {
        if (!isset($dnm)) {
            $dnm = "%";
        }
        $people = [];
        $conn = $this->getConnection();
        $result = $this->executeStatement(
            $conn,
            'SELECT usersId, displayName, userName FROM users WHERE
displayName LIKE ?',
            function (\mysqli_stmt $stmt) use ($dnm) {
                $stmt->bind_param("s", $dnm);
            }
        );
        $result->bind_result($usersId, $displayName, $userName);
        while ($result->fetch()) {
            $people[] = new UserData(
                $usersId,
                $displayName,
                $userName
            );
        }
        return $people;
    }

    public function insertRating(int $blogsId, int $usersId, int $rating):
int
    {
        $conn = $this->getConnection();
        $statement = $this->executeStatement(
            $conn,
            'INSERT INTO ratings (blogsId, usersId, rating) VALUES (?, ?,
?)',
            function (\mysqli_stmt $stmt) use ($blogsId, $usersId, $rating) {

```

```

        $stmt->bind_param('iii', $blogsId, $usersId, $rating);
    }
    );
    return $statement->insert_id;
}

public function getBlogs24Count(): ?int
{
    $count = null;
    $con = $this->getConnection();
    $res = $this->executeQuery(
        $con,
        'SELECT COUNT(*) as cnt FROM blogs WHERE date >= now() - INTERVAL
1 DAY;';
    );
    if($row = $res->fetch_object()){
        $count = $row->cnt;
    }
    $res->close();
    $con->close();
    return $count;
}

public function getLatestBlog(): ?string
{
    $count = null;
    $con = $this->getConnection();
    $res = $this->executeQuery(
        $con,
        'SELECT MAX(date) as cnt FROM blogs;';
    );
    if($row = $res->fetch_object()){
        $count = $row->cnt;
    }
    $res->close();
    $con->close();
    return $count;
}

public function deleteRating(int $rid): int
{
    $conn = $this->getConnection();
    $statement = $this->executeStatement(
        $conn,
        'DELETE FROM ratings WHERE ratingsId = ?';
        function (\mysqli_stmt $stmt) use ($rid) {
            $stmt->bind_param("i", $rid);
        }
    );
    return $statement->affected_rows > 0;
}

//SELECT ratings.blogsId as bid, users.displayName as dpn FROM ratings
JOIN users ON ratings.usersId = users.usersId WHERE ratings.blogsId = 46;
public function getLikedBy(int $bid): array
{
    $names[] = null;
    $conn = $this->getConnection();

```

```

        $result = $this->executeStatement(
            $conn,
            'SELECT users.displayName as dpn FROM ratings JOIN users ON
ratings.usersId = users.usersId WHERE ratings.blogsId = ?;',
            function (\mysqli_stmt $stmt) use ($bid) {
                $stmt->bind_param("i", $bid);
            }
        );
        $result->bind_result($dnm);
        while ($result->fetch()) {
            $names[] = $dnm;
        }
        return $names;
    }
}

```

Session.php:

```
<?php
```

```
namespace Infrastructure;
```

```

class Session implements \Application\Interfaces\Session {
    public function __construct() {
        session_start(); // Darf nur einmal aufgerufen werden!
        // notwending um $_SESSION Variable zu "aktivieren"
    }

    public function get(string $key): mixed {
        return $_SESSION[$key] ?? null;
    }

    public function put(string $key, mixed $value): void {
        $_SESSION[$key] = $value;
    }

    public function delete(string $key): void {
        unset($_SESSION[$key]);
    }
}

```

Controllers:

Blogs.php:

```
<?php
```

```
namespace Presentation\Controllers;
```

```

class Blogs extends \Presentation\MVC\Controller {
    public function __construct(
        private \Application\BlogsQuery $blogsQuery,
        private \Application\LikedByQuery $likedByQuery,
        private \Application\RatingsQuery $ratingsQuery,
        private \Application\BlogByIdQuery $blogByIdQuery,
        private \Application\CreateRatingCommand $createRatingCommand,
        private \Application\DeleteRatingCommand $deleteRatingCommand,
    ) {
    }
}

```

```

        private \Application\DeleteBlogCommand $deleteBlogCommand,
        private \Application\SignedInUserQuery $signedInUserQuery,
        private \Application\CreatePostCommand $createPostCommand
    )
}

public function GET_Index(): \Presentation\MVC\ActionResult {
    $user = $this->signedInUserQuery->execute();
    $this->tryGetParam('dpn', $dnm);
    if ($this->tryGetParam('uid', $id)) {
        $uid = $id;
        $dpn = $dnm;
    }
    elseif ($this->tryGetParam('bid', $id)){
        $uid = $id;
        $dpn = $user->getDisplayName();
    }
    else{
        $uid = null;
        $dpn = 'null';
    }
    return $this->view('blogsList', [
        'user' => $this->signedInUserQuery->execute(),
        'displaynm' => $dpn,
        'euser' => $uid,
        'blogs' => $this->blogsQuery->execute($uid),
        'ratings' => $this->ratingsQuery->execute(),
        'returnUrl' => $this->getRequestUri()
    ]);
}

public function POST_Search(): \Presentation\MVC\ActionResult {
    $this->tryGetParam('uid', $search);
    $this->tryGetParam('dpn', $search2);
    return $this->redirect("Blogs", "Index", ['uid' => $search, 'dpn' =>
$search2]);
}

public function POST_Like(): \Presentation\MVC\ActionResult {
    $allRatings = $this->ratingsQuery->execute();
    $user = $this->signedInUserQuery->execute();
    $this->tryGetParam('blog', $blog);
    foreach ($allRatings as $rating){
        if ($user->getUsersId() == $rating->getUsersId() and $blog ==
$rating->getBlogsId()){ //IF already Liked
            $this->deleteRatingCommand->execute($rating->getRatingsId());
            return $this->redirectToUri($this->getParam('returnUrl'));
        }
    }
    $this->createRatingCommand->execute($blog, $user->getUsersId(), 1);
    return $this->redirect("Blogs", "Index", ['uid' => $user-
>getUsersId(), 'dpn' => $user->getDisplayName()]);
}

public function GET_NewPost(): \Presentation\MVC\ActionResult {
    $user = $this->signedInUserQuery->execute();
    if ($user == null) {
        return $this->redirect("Home", "Index");
    }
}

```

```

    }
    $data = [
        "user" => $this->signedInUserQuery->execute(),
        "blogs" => $this->blogsQuery->execute(),
    ];
    if ($this->tryGetParam('title', $title)) {
        $data['title'] = $title;
    }
    if ($this->tryGetParam('text', $text)) {
        $data['text'] = $text;
    }
    return $this->view("blogsList", $data);
}

public function POST_NewPost(): \Presentation\MVC\ActionResult {
    date_default_timezone_set('Europe/Vienna');
    $now = date('m/d/Y h:i:s a', time());
    $user = $this->signedInUserQuery->execute();
    if ($user == null) {
        return $this->redirect("Home", "Index");
    }
    $errors = [];
    if (!$this->tryGetParam('title', $title)) {
        $errors[] = "Title required";
    } elseif(strlen($title) > 255) {
        $errors[] = "Title can't be longer than 255 characters";
    }
    if (!$this->tryGetParam('text', $text)) {
        $errors[] = "text required";
    }
    if (count($errors) == 0) {
        $bid = $this->createPostCommand->execute($user->getUsersId(),
        $title, $text, $now);
        if (isset($bid)) {
            return $this->redirectToUri($this->getParam('returnUrl'));
        }
    }
    return $this->redirectToUri($this->getParam('returnUrl'));
}

public function POST_Delete(): \Presentation\MVC\ActionResult {
    $user = $this->signedInUserQuery->execute();

    if ($user == null) {
        return $this->redirectToUri($this->getParam('returnUrl'));
    }
    $bid = null;
    $errors = [];
    $rating = null;
    if ($this->tryGetParam('bid', $bid)) {
        $rating = $this->blogByIdQuery->execute($bid);
    } else {
        $errors[] = 'Blog does not Exist';
    }
    $this->deleteBlogCommand->execute($rating->getBlogsId());
    return $this->redirectToUri($this->getParam('returnUrl'));
}

```

```
    }
}
```

*Home.php:*

```
<?php
namespace Presentation\Controllers;

class Home extends \Presentation\MVC\Controller {
    public function __construct(
        private \Application\SignedInUserQuery    $signedInUserQuery,
        private \Application\BlogsCount           $blogsCount,
        private \Application\Blogs24Query         $blogs24Count,
        private \Application\LatestBlogDateQuery  $latestBlogDateQuery,
        private \Application\UserCount            $userCount
    )
    { }

    public function GET_Index() : \Presentation\MVC\ActionResult {
        // TODO return action result!
        return $this->view('home', [
            'user' => $this->signedInUserQuery->execute(),
            'count' => $this->blogsCount->execute(),
            '24count' => $this->blogs24Count->execute(),
            'latest' => $this->latestBlogDateQuery->execute(),
            'userCount' => $this->userCount->execute()
        ]);
    }
}
```

*People.php:*

```
<?php
namespace Presentation\Controllers;

class Home extends \Presentation\MVC\Controller {
    public function __construct(
        private \Application\SignedInUserQuery    $signedInUserQuery,
        private \Application\BlogsCount           $blogsCount,
        private \Application\Blogs24Query         $blogs24Count,
        private \Application\LatestBlogDateQuery  $latestBlogDateQuery,
        private \Application\UserCount            $userCount
    )
    { }

    public function GET_Index() : \Presentation\MVC\ActionResult {
        // TODO return action result!
        return $this->view('home', [
            'user' => $this->signedInUserQuery->execute(),
            'count' => $this->blogsCount->execute(),
            '24count' => $this->blogs24Count->execute(),
            'latest' => $this->latestBlogDateQuery->execute(),
            'userCount' => $this->userCount->execute()
        ]);
    }
}
```

User.php:

```
<?php
```

```
namespace Presentation\Controllers;
```

```
class User extends \Presentation\MVC\Controller {
    public function __construct(
        private \Application\SignInCommand $signInCommand,
        private \Application\SignOutCommand $signOutCommand,
        private \Application\RegisterCommand $registerCommand,
        private \Application\SignedInUserQuery $signedInUserQuery
    )
    {

    }

    private function signedInUser(): bool {
        return $this->signedInUserQuery->execute() != null;
    }

    public function GET_LogIn(): \Presentation\MVC\ActionResult {
        return $this->view('login', [
            'user' => $this->signedInUserQuery->execute(),
            'userName' => '' // weil erstmaliger Aufruf der Login Page //
        ]);
    }

    public function POST_LogIn(): \Presentation\MVC\ActionResult {
        // Try to authenticate given user
        if(!$this->signInCommand->execute($this->getParam('un'), $this->getParam('pwd'))) {
            // authentication failed - nothing has changed, show view with
            error informaion
            return $this->view('login', [
                'user' => $this->signedInUserQuery->execute(),
                'userName' => $this->getParam('un'),
                'errors' => [ 'Invalid user name or password.' ]
            ]);
        }
        return $this->redirect('Home', 'Index');
    }

    public function POST_LogOut(): \Presentation\MVC\ActionResult {
        // sign out current user
        $this->signOutCommand->execute();
        return $this->redirect('Home', 'Index');
    }

    public function GET_Register(): \Presentation\MVC\ActionResult {
        if ($this->signedInUser()) {
            return $this->redirect("Home", "Index");
        }
        return $this->view("register", ['user' => $this->signedInUserQuery-
```

```

>execute()]);
    }

    public function POST_Register(): \Presentation\MVC\ActionResult {
        if ($this->signedInUser()) {
            return $this->redirect("Home", "Index");
        }
        $displayName = $this->getParam("displayName");
        $username = $this->getParam("username");
        $password = $this->getParam("password");
        $repeatPassword = $this->getParam("repeat-password");
        $errors = array();
        if (strlen($username) < 3) {
            array_push($errors, "Username has to be longer than 2
characters");
        }
        if (strlen($username) > 45) {
            array_push($errors, "Username can only have 45 characters");
        }
        if (strlen($password) < 4) {
            array_push($errors, "The password must be at least 4 characters
long");
        }
        if (strlen($password) > 255) {
            array_push($errors, "Password can only have 255 characters");
        }
        if ($password !== $repeatPassword) {
            array_push($errors, "Passwords do not match");
        }
        if (count($errors) === 0) {
            $id = $this->registerCommand->execute($displayName, $username,
$password);
            if ($id == null) {
                array_push($errors, "Username already in use");
            }
        }
        if (count($errors) > 0) {
            return $this->view("register", [
                'errors' => $errors,
                'username' => $username
            ]);
        }
        $this->signInCommand->execute($username, $password);
        return $this->redirect("Home", "Index");
    }
}

```

MVC: Nicht im pdf enthalten (Gleich wie in der Übung)

Presentation:

*ServiceProvider: Nicht im pdf enthalten (Gleich wie in der Übung)*



Views:

*blogsList.inc:*

```
<?php $render('partial/header', $data); ?>
<?php if ($data['user'] and $data['user']->getUsersId() == $data['euser']):
?>
<h2>New Blog</h2>
<?php $beginForm('Blogs', 'NewPost', ['returnUrl' => $data['returnUrl']],
method: 'post'); ?>
    <div>
        <div class="pb-2">
            <input class="form-control form-control-lg" type="text"
placeholder="Title" name="title">
        </div>
        <div class="pb-3">
            <textarea class="w-100" placeholder="What's on your mind?"
name="text"></textarea>
        </div>
        <div>
            <button class="btn btn-primary">Post</button>
        </div>
    </div>
<?php $endForm(); ?>
    <div class="mb-4">
        <hr class="solid">
    </div>
<?php endif; ?>
<h2><?php $htmlOut($data['displaynm']); ?>'s Blogs</h2>
<?php if ($data['blogs'] != null && sizeof($data['blogs']) > 0) {
    $render('partial/blogs', $data);
} else {?>
    <p>No Blogs Yet.</p>
<?php } ?>
<?php $render('partial/footer', $data); ?>
```

*home.inc:*

```
<?php $render('partial/header', $data); ?>

<h1>Welcome</h1>
<p>Welcome to Faceblog</p>
<div class="row">
    <div class="col-sm-6 pb-2">
        <div class="card">
            <div class="card-body">
                <h5 class="card-title">Anzahl registrierter Nutzer</h5>
                <p class="card-text"><?php $htmlOut($data['userCount']); ?></p>
            </div>
        </div>
    </div>
    <div class="col-sm-6 pb-2">
        <div class="card">
            <div class="card-body">
                <h5 class="card-title">Insgesamt geschriebene Blogs</h5>
                <p class="card-text"><?php $htmlOut($data['count']); ?></p>
            </div>
        </div>
    </div>
</div>
```

```

</div>
<div class="col-sm-6 pb-2">
  <div class="card">
    <div class="card-body">
      <h5 class="card-title">Geschriebene Blogs in den Letzten 24
Stunden</h5>
      <p class="card-text"><?php $htmlOut($data['24count']); ?></p>
    </div>
  </div>
</div>
<div class="col-sm-6 pb-2">
  <div class="card">
    <div class="card-body">
      <h5 class="card-title">Datum des Letzten Blogs</h5>
      <p class="card-text"><?php $htmlOut($data['latest']); ?></p>
    </div>
  </div>
</div>
</div>
<?php $render('partial/footer', $data); ?>

```

*login.inc:*

```

<?php $render('partial/header', $data); ?>

<h1>Login</h1>

<?php $beginForm('User', 'LogIn', method: 'post'); ?>
<div class="mb-3">
  <label for="userName" class="form-label">User name</label>
  <input class="form-control" id="userName" name="un" value="<?php
$htmlOut($data['userName']); ?>">
</div>
<div class="mb-3">
  <label for="password" class="form-label">Password</label>
  <input type="password" class="form-control" id="password" name="pwd">
</div>
<button class="btn btn-primary">Log in</button>
<?php $endForm(); ?>

<?php $render('partial/footer', $data); ?>

```

*people.inc:*

```

<?php $render('partial/header', $data); ?>

<?php $beginForm('People', 'Search', method: 'post'); ?>
<div class="container justify-content-center">
  <div class="input-group mb-3">
    <input type="text" class="form-control input-text"
placeholder="Find your Friends..." id="dnm" name="dnm"/>
    <div class="input-group-append">
      <button class="btn btn-outline-warning btn-lg"><i class="fa
fa-search"></i></button>

```

```

        </div>
    </div>
</div>
<?php $endForm(); ?>
<?php foreach ($data['people'] as $p) : ?>
    <div class="d-flex justify-content-center pb-5">
        <div class="col-md-8">
            <div class="panel panel-white panel-shadow">
                <div class="row">
                    <div class="col-sm-12">
                        <div class="image-container bg2">
                            
                        </div>
                    </div>
                    <div class="col-sm-12">
                        <div class="details">
                            <h4><?php $htmlOut($p->getDisplayname()); ?> <i
class="fa fa-sheild"></i></h4>
                            <div>Mitglied seit:</div>
                            <div>DATUM</div>
                            <div class="mg-top-10">
                                <?php $beginForm('Blogs', 'Search', method:
'post', params: ['uid' => ($p->getUsersId()), 'dpn' => ($p->getDisplayname()),
'returnUrl' => $data['returnUrl']); ?>
                                <button class="btn btn-outline-warning
btn-lg">About <?php $htmlOut($p->getDisplayname()); ?> </button>
                                <?php $endForm(); ?>
                            </div>
                        </div>
                    </div>
                </div>
            </div>
        </div>
    </div>
</div>
<?php endforeach; ?>
<?php $render('partial/footer', $data); ?>
register.inc:
<?php $render('partial/header', $data) ?>

<div class="container mt-3 w-25">
    <h1 class="bd-title">Register</h1>
    <?php $beginForm('User', 'Register', method: 'post'); ?>
    <div class="mt-3 form-group">
        <label for="displayName" class="form-label">DisplayName</label>
        <input id="displayName" name="displayName" placeholder="displayName"
type="text"
class="form-control" required>
    </div>
    <div class="mt-3 form-group">
        <label for="username" class="form-label">Username</label>
        <input id="username" name="username" placeholder="Username"
type="text"
class="form-control" required>
    </div>
</div>

```

```

        <label for="password" class="form-label">Password</label>
        <input id="password" name="password" placeholder="Password"
type="password" class="form-control" required>
    </div>
    <div class="mt-3 form-group">
        <label for="repeat-password" class="form-label">Repeat
Password</label>
        <input id="repeat-password" name="repeat-password"
placeholder="Repeat Password"
type="password" class="form-control" required>
    </div>
    <div class="mt-3 form-group">
        <button type="submit" class="btn btn-primary">Submit</button>
        <?php $link("Login", "User", "Login", cssClass: 'btn btn-secondary');
?>
    </div>
    <?php $endForm(); ?>
</div>
<?php $render('partial/footer', $data) ?>

```

Partial:

*blogs.inc:*

```

<?php foreach ($data['blogs'] as $blog) : ?>
<div class="container">
    <div class="row">
        <div class="col-md-8">
            <div class="media g-mb-30 media-comment">
                <div class="media-body u-shadow-v18 g-bg-secondary g-pa-30">
                    <div class="g-mb-15">
                        <h5 class="h5 g-color-gray-dark-v1 mb-0"><?php
$htmlOut($blog->getTitle()); ?></h5>
                        <span class="g-color-gray-dark-v4 g-font-size-
12"><?php $htmlOut($blog->getDate()); ?></span>
                    </div>
                        <p><?php $htmlOut($blog->getText()); ?></p>
                        <ul class="list-inline d-sm-flex my-0">
                            <li class="list-inline-item g-mr-20">
                                <a class="u-link-v5 g-color-gray-dark-v4 g-color-
primary--hover pe-4">
                                    <?php $beginForm('Blogs', 'Like', ['blog' =>
$blog->getBlogsId(), 'returnUrl' => $data['returnUrl']], method : 'post'); ?>
                                    <button class="btn btn-primary"><i
class="fa fa-thumbs-up"></i> | <?php $htmlOut($blog->getRating()); ?>
                                    Likes</button>
                                </a>
                                <?php $endForm(); ?>
                            </li>
                        </ul>
                    </div>
                </div>
            </div>
            <div class="col-md-4">
                <h5 class="h5 g-color-gray-dark-v1 mb-
0">Liked By:</h5>
                <span class="g-color-gray-dark-v4 g-font-
size-12">
                    <?php foreach ($blog->getLikedBy() as
$lb) : ?>
                        <?php $htmlOut($lb); ?>
                    <?php endforeach; ?>
                </span>
            </div>
        </div>
    </div>
</div>

```

```

        </div>
    </a>
    <a class="u-link-v5 g-color-gray-dark-v4 g-color-
primary--hover">
        <?php if ($data['user']->getUsersId() ==
$data['euser']): ?>
            <?php $beginForm('Blogs', 'Delete', ['bid'
=> $blog->getBlogsId(), 'returnUrl' => $data['returnUrl']], method : 'post');
?>
            <button class="btn btn-danger"><i
class="fa fa-trash"></i> | Delete</button>
            <?php $endForm(); ?>
        <?php endif; ?>
    </a>
</li>
</ul>
</div>
</div>
</div>
</div>
</div>
<div class="mb-4">
    <hr class="solid">
</div>
<?php endforeach; ?>

```

*errors.inc:*

```

<?php foreach ($data['blogs'] as $blog) : ?>
<div class="container">
    <div class="row">
        <div class="col-md-8">
            <div class="media g-mb-30 media-comment">
                <div class="media-body u-shadow-v18 g-bg-secondary g-pa-30">
                    <div class="g-mb-15">
                        <h5 class="h5 g-color-gray-dark-v1 mb-0"><?php
$htmlOut($blog->getTitle()); ?></h5>
                        <span class="g-color-gray-dark-v4 g-font-size-
12"><?php $htmlOut($blog->getDate()); ?></span>
                    </div>
                        <p><?php $htmlOut($blog->getText()); ?></p>
                        <ul class="list-inline d-sm-flex my-0">
                            <li class="list-inline-item g-mr-20">
                                <a class="u-link-v5 g-color-gray-dark-v4 g-color-
primary--hover pe-4">
                                    <?php $beginForm('Blogs', 'Like', ['blog' =>
$blog->getBlogsId(), 'returnUrl' => $data['returnUrl']], method : 'post'); ?>
                                    <button class="btn btn-primary"><i
class="fa fa-thumbs-up"></i> | <?php $htmlOut($blog->getRating()); ?>
                                    Likes</button>
                                    <?php $endForm(); ?>
                                </div>
                                    <h5 class="h5 g-color-gray-dark-v1 mb-
0">Liked By:</h5>
                                    <span class="g-color-gray-dark-v4 g-font-
size-12">

```

```

$1b) : ?>
        <?php foreach ($blog->getLikedBy() as
        <?php $htmlOut($1b); ?>
        <?php endforeach; ?>
    </span>
</div>
</a>
<a class="u-link-v5 g-color-gray-dark-v4 g-color-
primary--hover">
        <?php if ($data['user']->getUsersId() ==
        $data['euser']): ?>
            <?php $beginForm('Blogs', 'Delete', ['bid'
=> $blog->getBlogsId(), 'returnUrl' => $data['returnUrl']], method : 'post');
            ?>
            <button class="btn btn-danger"><i
class="fa fa-trash"></i> | Delete</button>
            <?php $endForm(); ?>
        <?php endif; ?>
    </a>
</li>
</ul>
</div>
</div>
</div>
</div>
</div>
<div class="mb-4">
    <hr class="solid">
</div>
<?php endforeach; ?>

```

footer.inc:

```

</div>
<footer class="bg-light mt-3">
    <div class="container">
        <p class="text-muted p-0"><?php echo htmlentities(date('c')); ?></p>
    </div>
</footer>
<script src="js/bootstrap.bundle.min.js"></script>
</body>

</html>

```

header.inc:

```

<!doctype html>
<html lang="en">

<head>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1">
    <link href="css/bootstrap.min.css" rel="stylesheet">
    <link href="css/index.css" rel="stylesheet">
    <link rel="stylesheet" href="https://cdnjs.cloudflare.com/ajax/libs/font-
awesome/4.7.0/css/font-awesome.min.css">

```

```

<title>Faceblog</title>
</head>

<body>
  <nav class="navbar navbar-expand-lg navbar-dark bg-dark">
    <div class="container">
      <!-- DONE: navigation home -->
      <?php $link('Faceblog', 'Home', 'Index', cssClass: 'navbar-
brand'); ?>
      <button class="navbar-toggler" type="button" data-bs-
toggle="collapse" data-bs-target="#navbar">
        <span class="navbar-toggler-icon"></span>
      </button>
      <div class="collapse navbar-collapse" id="navbar">
        <nav class="navbar-nav me-auto">
          <!-- DONE: navigation -->
          <?php if ($data['user']): ?>
            <?php $link('MyBlog', 'Blogs', 'Index', ['bid' =>
$data['user']->getUsersid()], cssClass: 'nav-link');?>
            <?php $link('People', 'People', 'Index', cssClass:
'nav-link'); ?>
          <?php endif; ?>
        </nav>
        <!-- TODO: navigation user -->
        <?php $render('partial/user', $data['user']); ?>
      </div>
    </div>
  </nav>
  <div class="container mt-3">
    <?php if(isset($data['errors'])) {
      $render('partial/errors', $data['errors']);
    } ?>
  </div>

```

user.inc:

```

<?php if (!isset($data)): ?>
  <nav class="navbar-nav">
    <?php $link('Login', 'User', 'LogIn', cssClass: 'nav-link'); ?>
    <?php $link("Register", "User", "Register", cssClass: 'nav-item nav-
link'); ?>
  </nav>
<?php else: ?>
  <?php $beginForm('User', 'LogOut', method: 'post', cssClass: 'form-
inline'); ?>
  <span class="navbar-text me-2">Welcome, <strong><?php $htmlOut($data-
>getDisplayName()); ?></strong></span>
  <button class="btn btn-secondary">Log out</button>
  <?php $endForm(); ?>
<?php endif; ?>

```

Index.php:

```

<?php
// === register autoloader
spl_autoload_register(function ($class) {
  $file = __DIR__ . '/src/' . str_replace('\\', '/', $class) . '.php';

```

```

        if (file_exists($file)) {
            require_once($file);
        }
    });

    // TODO: handle request
    $sp = new \ServiceProvider();
    // --- TODO: Account Created
    // --- APPLICATION
    $sp->register(\Application\BlogsCount::class);
    $sp->register(\Application\UserCount::class);
    $sp->register(\Application\SignedInUserQuery::class);
    $sp->register(\Application\BlogsQuery::class);
    $sp->register(\Application\PeopleQuery::class);
    $sp->register(\Application\LikedByQuery::class);
    $sp->register(\Application\RatingsQuery::class);
    $sp->register(\Application\Blogs24Query::class);
    $sp->register(\Application\LatestBlogDateQuery::class);
    $sp->register(\Application\DeleteRatingCommand::class);
    $sp->register(\Application\CreateRatingCommand::class);
    $sp->register(\Application\SignInCommand::class);
    $sp->register(\Application\BlogByIdQuery::class);
    $sp->register(\Application\DeleteBlogCommand::class);
    $sp->register(\Application\RegisterCommand::class);
    $sp->register(\Application\SignOutCommand::class);
    $sp->register(\Application\CreatePostCommand::class);
    // ----- SERVICES
    $sp->register(\Application\Services\AuthenticationService::class);
    // --- PRESENTATION
    $sp->register(\Presentation\MVC\MVC::class, function () {
        return new \Presentation\MVC\MVC();
    }, isSingleton: true);
    // controllers
    $sp->register(\Presentation\Controllers\Blogs::class);
    $sp->register(\Presentation\Controllers\Home::class);
    $sp->register(\Presentation\Controllers\User::class);
    $sp->register(\Presentation\Controllers\People::class);

    // --- INFRASTRUCTURE
    $sp->register(\Infrastructure\Session::class, isSingleton: true); // !!
    Important damit start_session nur einmal aufgerufen wird
    $sp->register(\Application\Interfaces\Session::class,
    \Infrastructure\Session::class);
    $sp->register(\Infrastructure\Repository::class, implementation: function() {
        return new \Infrastructure\Repository('localhost', '', 'root', '',
        'faceblog');}, isSingleton: true);
    $sp->register(\Application\Interfaces\BlogsRepository::class,
    \Infrastructure\Repository::class); //statt FakeRepo nur Repository
    hinschreiben
    $sp->register(\Application\Interfaces\RatingsRepository::class,
    \Infrastructure\Repository::class);
    $sp->register(\Application\Interfaces\UserRepository::class,
    \Infrastructure\Repository::class);

    $sp->resolve(\Presentation\MVC\MVC::class)->handleRequest($sp);

```



## 5. Tests

### Registrieren:

*Falsche eingaben:*

Please correct the following and try again:

- Passwords do not match

## Register

DisplayName

Username

Password

Repeat Password

Submit

Login

Please correct the following and try again:

- The password must be at least 4 characters long

## Register

DisplayName

Username

Password

Repeat Password

Submit

Login

2022-06-07T20:23:07+02:00

Please correct the following and try again:

- Username has to be longer than 2 characters
- The password must be at least 4 characters long

## Register

DisplayName

Username

Password

Repeat Password

[Submit](#) [Login](#)

Please correct the following and try again:

- Username already in use

## Register

DisplayName

Username

Password

Repeat Password

[Submit](#) [Login](#)

2022-06-07T10:22:55+02:00

*Richtige eingaben:*

# Register

DisplayName


 

Username

Password

Repeat Password

[Submit](#) [Login](#)

Welcome, **TestUser**. [Log out](#)

Logout:

Welcome, **TestUser**. [Log out](#)

[Login](#) [Register](#)

Login:

*Falsche Eingaben:*

Please correct the following and try again:

- Invalid user name or password.

## Login

User name

a

Password

Log in

*Richtige Eingaben:*

## Login

User name

test

Password

••••

Log in

Welcome, **TestUser.**

Log out

Eigene Blogs anschauen:

*Eigene Blogs finden:*

Faceblog MyBlog People

Welcome, **TestUser.** Log out

## New Blog

Title

What's on your mind?

Post

## TestUser's Blogs

TestBlog

2022-06-07 08:26:36

This is a testblog

👍 | 0 Likes

[Liked By:](#)

🗑 | Delete

*Eigenen Blog Liken:*

## TestUser's Blogs


TestBlog

2022-06-07 08:26:36

This is a testblog

 | 1 Likes

Liked By:  
TestUser

 | Delete

*Eigenen Blog Löschen:*

## New Blog

Post


## TestUser's Blogs


No Blogs Yet.

2022-06-07T20:27:00+02:00

Blogs von Freunden anschauen:

*Freund suchen und finden:*





scr4

Mitglied seit:  
DATUM

About scr4

*Blogs vom Freund anschauen:*

## scr4's Blogs

Hey my best Friend is TestUser

2022-06-07 08:28:09

Pls like my Post!!!!!!!!!!



[Liked By:](#)

*Blogs vom Freund Liken:*

## scr4's Blogs

Hey my best Friend is TestUser

2022-06-07 08:28:09

Pls like my Post!!!!!!!!!!



[Liked By:](#)

[TestUser](#)

Blog schreiben:

### New Blog

This is my second Blog (I deleted the first one :))

Like pleaaasseeeee

Post

## TestUser's Blogs

This is my second Blog (I deleted the first one :))

2022-06-07 08:29:17

Like pleaaasseeeee

👍 | 0 Likes

Liked By:

🗑 | Delete