

☐ **Gruppe 1** DI (FH) G. Horn-V., MSc**Abgabetermin:** Sonntag, 06.06.2021, 24:00 Uhr☐ **Gruppe 2** DI (FH) I. Krammer**Name:** _____**Aufwand (h):** _____

Produktbewertungsportal mit PHP und MySQL

Zu entwickeln ist ein einfaches Produktbewertungsportal, in dem registrierte Benutzer Bewertungen für eine Reihe von Produkten abgeben können, um beliebigen Besuchern des Portals so z. B. bei Kaufentscheidungen zu helfen. Dabei sind die nachstehend beschriebenen Anforderungen zu berücksichtigen bzw. vollständig umzusetzen.

Funktionale Anforderungen

- Das Durchsuchen der Produkte sowie das Einsehen von abgegebenen Bewertungen muss anonym und somit auch ohne Anmeldung uneingeschränkt möglich sein.
- Für das Anlegen eines neuen Produkts oder das Abgeben einer Bewertung muss sich ein Benutzer zuvor registriert und angemeldet haben.
- Neue Benutzer können sich uneingeschränkt direkt über eine eigene Seite im Portal registrieren. Für jeden Benutzer sind dabei mindestens Benutzername (muss systemweit eindeutig sein) und Passwort über eine sinnvolle Benutzeroberfläche zu erfassen.
- Auf einer Übersichtsseite sollen alle im System gespeicherten Produkte aufgelistet werden. Für jedes Produkt muss mindestens sein Name, sein Hersteller, der Benutzer, der das Produkt angelegt hat, die Anzahl der abgegebenen Bewertungen und die durchschnittliche Bewertung (zwischen 1.0 und 5.0) für das Produkt angezeigt werden. Über ein entsprechendes Formular kann ein angemeldeter Benutzer ein neues Produkt anlegen.
- Durch Anklicken eines Produkts auf der Übersichtsseite gelangt man zu einer Detailansicht, in der alle Bewertungen zu einem Produkt chronologisch absteigend geordnet aufgelistet werden. Für jede Bewertung ist zumindest der Ersteller, das Erstellungsdatum, die eigentliche Bewertung (1, 2, 3, 4 oder 5 in Schulnoten) sowie ein möglicher Kommentar in Textform ansprechend darzustellen. Über ein entsprechendes Formular kann auf einfache Art und Weise eine weitere Bewertung für das Produkt erstellt werden.
- Auf einer Suchseite soll über ein Formular durch Eingabe eines Suchbegriffs (Freitext) gezielt nach Produkten im Forum gesucht werden können. Nach Absetzen einer Suchanfrage sollen alle Produkte angezeigt werden, deren Name oder Hersteller den eingegebenen Suchbegriff enthält. Durch das Anklicken eines Suchergebnisses kann direkt zur Detailseite des entsprechenden Produkts gesprungen werden.
- Ein angemeldeter Benutzer soll seine (und nur seine) abgegebenen Bewertungen im Nachhinein auch noch korrigieren bzw. löschen können.
- Ebenso soll der Ersteller eines Produkts (und nur dieser) die Produktdaten selbst auch nachträglich noch bearbeiten können. Dazu zählt aber natürlich nicht das Bearbeiten oder Löschen von Kommentaren anderer Benutzer... ;)

- Eine Gruppierung der Produkte in Kategorien wäre natürlich wünschenswert und auch sehr hilfreich für die Darstellung, ist aber nicht zwingend erforderlich.

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 zum 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

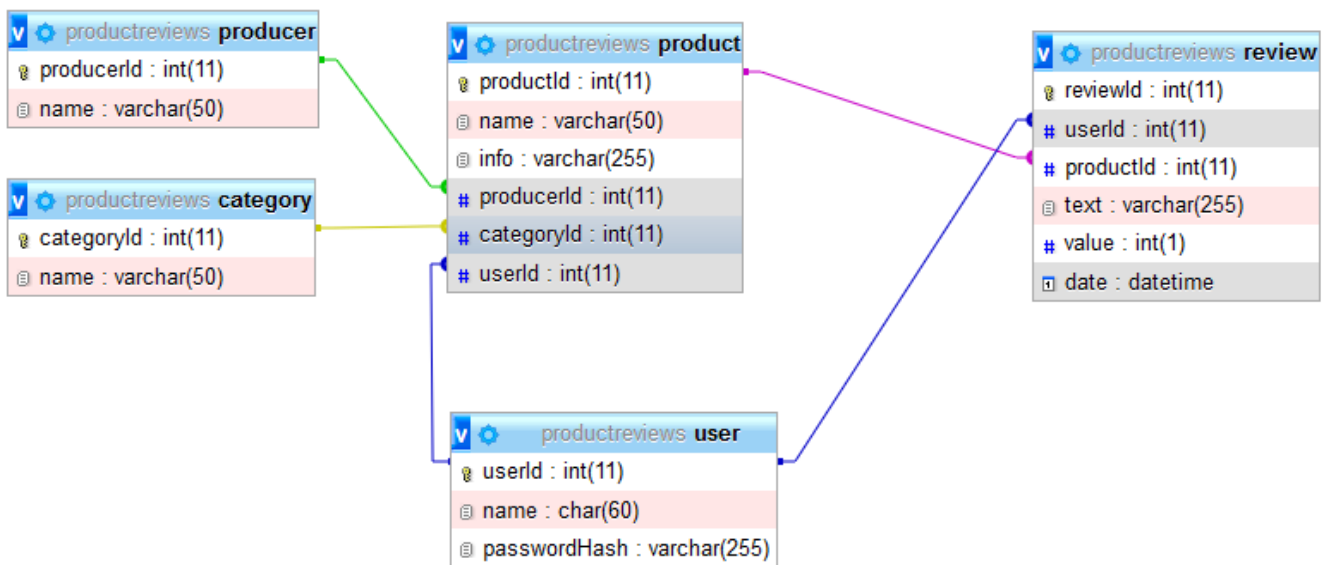
Product Review Page

Lösungsidee

Als Basis für die Umsetzung des Projekts wurde der in der Übung Code verwendet. Dieser wurde entsprechend angepasst und um die gefragte Funktionalität erweitert.

Datendarstellung

Um die Produkte und Reviews zu speichern, wurde ein Datenbankmodell entwickelt.



Für jede Tabelle wird eine Entity-Klasse in php entwickelt. Für eine geeignete Darstellung werden Datenmodell-Klassen entwickelt, die sich von den Entity-Klassen unterscheiden können.

- Category/CategoryData: Kein Unterschied
- Producer/ProducerData: Kein Unterschied
- Product/ProductData:
 - producerId wird durch producerName ersetzt
 - userId wird durch userName ersetzt
 - eine Spalte rating für das kumulative Rating hinzugefügt
- Review/ReviewData: Kein Unterschied
- User/UserData: Der passwordHash wird im Frontend nicht benötigt und entfernt

Controller

Zur Darstellung der Seiten werden folgende Controller verwendet:

- Home
 - Kümmt sich um die Darstellung der Index-Seite

- Products
 - Bietet alle wichtigen Funktionen um Produkte anzuzeigen. Weiters werden Funktionen zum bearbeiten der Daten implementiert.
- Review
 - Dieser Controller bietet Funktionen zum anzeigen, erstellen, bearbeiten und löschen von Reviews.
- User
 - Hier werden Funktionen zum ein- und ausloggen sowie zur Registrierung neuer Benutzer implementiert.

Datenabfrage

Zum Abfragen der Daten werden Interfaces angeboten und in einem Repository implementiert.

- CategoryRepository
 - abfragen von Kategorien
- ProducerRepository
 - abfragen der Produzenten
- ProductRepository
 - abfragen, erstellen, editieren und löschen von Produkten
- ReviewRepository
 - abfragen, erstellen, editieren und löschen von Reviews
- Session
 - speichern und abfragen von Session Variablen (in der Übung entwickelt)
- UserRepository
 - abfragen und erstellen von Benutzern

Views

- home.inc
 - die Index-Seite
- login.inc
 - eine Seite zum Einloggen (entwickelt in der Übung)
- newProduct.inc
 - eine Seite zum erstellen von Produkten. Diese wird auch zum editieren der Produkte verwendet
- productList.inc
 - eine Seite, um Produkte in einer Liste sortiert nach Kategorien anzuzeigen
- productsearch.inc
 - eine Seite, um Produkte gefiltert nach einem Suchbegriff in einer Liste anzuzeigen
- productview.inc
 - rendert die Detailseite eines Produkts
- register.inc
 - eine Seite, um neue Benutzer anzulegen.

Partials

- footer.inc
- header.inc
- product.inc
 - dient der Darstellung der Detailseite
- products.inc
 - liefert Produkte in einer Tabelle
- reviewCard.inc
 - liefert ein Review in einem ansprechenden Card-Format
- user.inc
 - zeigt den eingeloggten User oder zwei Login/Register Buttons an

Tests

Darstellung ohne Login

Index-Seite:

SCR4 Product Reviews [Home](#) [List](#) [Search](#) [Login](#) [Register](#)

Welcome

Welcome to the SCR4 product review page!

Thu Jun 3 12:39:52 2021

Kategorien-Seite:

SCR4 Product Reviews [Home](#) [List](#) [Search](#) [Login](#) [Register](#)

List of products

[books](#) [electronics](#) [clothes](#)

Name	Info	Producer	Rating
G Pro Wireless Mouse	A very light wireless mouse with 4 configurable buttons.	Logitech	3

Thu Jun 3 13:18:29 2021

Such-Seite:

Hier kann nach Produkt-Name oder Hersteller gesucht werden.

SCR4 Product Reviews

[Home](#) [List](#) [Search](#)

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

Search result

Search

Name	Info	Producer	Rating
G Pro Wireless Mouse	A very light wireless mouse with 4 configurable buttons.	Logitech	3

Thu Jun 3 12:45:38 2021

Detail-Seite:

SCR4 Product Reviews

[Home](#) [List](#) [Search](#)

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

[< Back](#)

G Pro Wireless Mouse

Category: electronics | created by: admin

Info:

A very light wireless mouse with 4 configurable buttons.

Reviews:

Review by Testuser

2021-05-30 20:58:27

Test

★★☆☆☆

Review by John

2021-05-29 17:30:24

Not big enough for my hand

★★☆☆☆

Review by admin

2021-05-29 15:57:17

Registrier- und Loginprozess

Registrier-Seite:

[SCR4 Product Reviews](#) [Home](#) [List](#) [Search](#) [Login](#) [Register](#)

Register

User name

Password

Repeat Password

[Register](#)

Thu Jun 3 12:55:05 2021

Eingabe eines bereits vergebenen Usernamen weist darauf hin:

[SCR4 Product Reviews](#) [Home](#) [List](#) [Search](#) [Login](#) [Register](#)

Register

User name

Password

Repeat Password

Username already in use

[Register](#)

Thu Jun 3 12:56:07 2021

Eingabe von unterschiedlichen Passwörtern:

SCR4 Product Reviews

HomeListSearch

LoginRegister

Register

User name

Password

Repeat Password

Passwords have to match

Register

Thu Jun 3 12:56:29 2021

Eine erfolgreiche Registrierung loggt den User ein und leitet auf die Index-Seite weiter. Ein "New Product" Menüeintrag wird jetzt angezeigt.

SCR4 Product Reviews

HomeListSearchNew Product

Welcome, scr4. Log out

Welcome

Welcome to the SCR4 product review page!

Thu Jun 3 12:58:06 2021

Reviews darstellen und bearbeiten

Als eingeloggter User kann auf der Produkt-Detailseite ein neues Review erstellt werden.

SCR4 Product Reviews

HomeListSearchNew Product

Welcome, **scr4.**

Log out

[< Back](#)

G Pro Wireless Mouse

Category: electronics | created by: admin

Info:

A very light wireless mouse with 4 configurable buttons.

Reviews:

Add Review

Review by Testuser

2021-05-30 20:58:27

Test

★ ★ ☆ ☆ ☆

Durch click auf den "Add Review" Button wird ein Textfeld und eine Liste der möglichen Ratings angezeigt.

SCR4 Product Reviews

HomeListSearchNew Product

Welcome, **scr4.**

Log out

[< Back](#)

G Pro Wireless Mouse

Category: electronics | created by: admin

Info:

A very light wireless mouse with 4 configurable buttons.

Reviews:

Add Review

Text

Rating



1 - very bad

Submit

Review by Testuser



Durch click auf den "Submit" Button wird das Review erstellt und als erstes in der Liste angezeigt. Für den Ersteller des Reviews werden zwei Buttons angezeigt, durch die das Review bearbeitet oder gelöscht werden kann.

Reviews:

Review by scr4 2021-06-03 13:03:29	 
I like this mouse very much, but the battery is too small.	
★★★★☆☆	

Durch click auf den "bearbeiten" Button wird ein ähnliches Interface wie bei der Erstellung eingeblendet. Text und rating können verändert werden.

Reviews:

Review by scr4 2021-06-03 13:03:29	 
<div>I like this mouse very much, but the battery is too small.</div>	
3 - medium	<div>Save review</div>

Durch click auf den "löschen" Button wird das Review entfernt.

[< Back](#)

G Pro Wireless Mouse

Category: electronics | created by: admin

Info:

A very light wireless mouse with 4 configurable buttons.

Reviews:

Add Review

Review by Testuser 2021-05-30 20:58:27
Test
★★★★☆☆

Produkt erstellen und bearbeiten

Durch click auf "New Product" in der Navigation wird eine Seite zum erstellen eines Produkts geöffnet.

Beim erstellen muss ein Name, aber keine Zusatzinfo eingegeben werden.

SCR4 Product Reviews

HomeListSearchNew Product

Welcome, **scr4**.

Log out

New Product

Product name:

Product info:

Producer:

Logitech

Category:

books

Submit

Thu Jun 3 13:14:35 2021

Nach Erstellung wird die Detailseite des erstellten Produkts angezeigt.

Als Ersteller werden wie beim Review Buttons zum bearbeiten und löschen angezeigt.

SCR4 Product Reviews

HomeListSearchNew Product

Welcome, **scr4**.

Log out

[< Back](#)

Z333 Speakers

Category: electronics | created by: scr4

Info:

The best value speakers on the market.

Reviews:

Add Review

Thu Jun 3 13:40:52 2021

Durch click auf den bearbeiten button wird das selbe Formular wie zur Erstellung angezeigt, aber bereits ausgefüllt mit den gespeicherten Daten.

SCR4 Product Reviews [Home](#) [List](#) [Search](#) [New Product](#) Welcome, **scr4.** [Log out](#)

Edit Product

Product name:

Product info:

The best value speakers on the market.

Producer:

Category:

[Submit](#)

Thu Jun 3 13:44:09 2021

Wird ein Produkt gelöscht, wird der Nutzer auf die Index-Seite weitergeleitet.

Code

Application

Entity

```
namespace Application\Entities;
class Category {
    public function __construct(
        private int $id,
        private string $name
    ) { }

    public function getId() : int {
        return $this->id;
    }

    public function getName() : string {
        return $this->name;
    }
}
```

```
namespace Application\Entities;
class Producer {
    public function __construct (
        private int $producerId,
        private string $name
    ) { }

    public function getProducerId(): int
    {
        return $this->producerId;
    }

    public function getName(): string
    {
        return $this->name;
    }
}
```

```
namespace Application\Entities;
class Product {
    public function __construct(
        private int $productId,
        private string $name,
        private string $info,
        private int $categoryId,
        private int $producerId,
        private int $userId,
    ) { }

    public function getProductId(): int
    {
        return $this->productId;
    }

    public function getName(): string
    {
        return $this->name;
    }

    public function getInfo(): string
    {
        return $this->info;
    }

    public function getCategoryId(): int
    {
        return $this->categoryId;
    }
}
```

```
public function getProducerId(): int
{
    return $this->producerId;
}

public function getUserId(): int
{
    return $this->userId;
}
}
```

```
namespace Application\Entities;
class Review {
    public function __construct (
        private int $reviewId,
        private int $userId,
        private int $productId,
        private string $text,
        private int $value,
        private string $dateTime
    ) { }

    public function getReviewId(): int {
        return $this->reviewId;
    }

    public function getUserId(): int {
        return $this->userId;
    }

    public function getProductId(): int
    {
        return $this->productId;
    }

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

    public function getValue(): int
    {
        return $this->value;
    }

    public function getDateTime(): string
    {
        return $this->dateTime;
    }
}
```

```
namespace Application\Entities;
class User {
    public function __construct (
        private int $id,
        private string $userName,
        private string $pWHash = ""
    ) { }

    public function getId() : int {
        return $this->id;
    }

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

    public function getPWHash(): string
    {
        return $this->pWHash;
    }
}
```

Interfaces

```
namespace Application\Interfaces;
interface CategoryRepository {
    public function getCategories() : array;
    public function getCategoryById($categoryId) : array;
}
```

```
namespace Application\Interfaces;
interface ProducerRepository {
    public function getAllProducers(): array;
}
```

```
namespace Application\Interfaces;
interface ProductRepository {
    public function getProductsForCategory(int $categoryId) : array;
    public function getProductsForFilter(string $filter) : array;
    public function getProductById(int $productId) : array;
    public function createProduct(\Application\Entities\Product $productData):
int;
    public function editProduct(int $productId, string $nme, string $info, int
$producerId, int $categoryId): int;
}
```

```

        public function deleteProduct(int $productId);
    }

```

```

namespace Application\Interfaces;
interface ReviewRepository {
    public function getReviewsByProductId(int $productId) : array;
    public function getReviewsByUserId(int $userId) : array;
    public function createReview(string $text, int $rating, int $productId, int
$userId) : int;
    public function editReview(int $id, string $text, int $rating): int;
    public function deleteReview(int $reviewId): int;
}

```

```

namespace Application\Interfaces;
interface Session {
    public function get(string $key) : mixed;
    public function put(string $key, mixed $value) : void;
    public function delete(string $key) : void;
}

```

```

namespace Application\Interfaces;
interface UserRepository {
    public function getUserForUserNameAndPassword(string $username, string
$password) : ?\Application\Entities\User;
    public function getUser(int $id) : ?\Application\Entities\User;
    public function createNewUser(string $userName, string $password) : ?
\Application\Entities\User;
}

```

Services

```

namespace Application\Services;
class AuthenticationService {
    const SESSION_USER_ID = "userId";
    const SESSION_USER_NAME = "userName";

    public function __construct(private \Application\Interfaces\Session $session)
    { }

    public function getUserId() :?string {
        return $this->session->get(self::SESSION_USER_ID, null);
    }
}

```



```

    public function signIn(string $userId) : void {
        $this->session->put(self::SESSION_USER_ID, $userId);
    }

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

```

Queries/Commands

```

namespace Application;

class CategoriesQuery {
    public function __construct(private Interfaces\CategoryRepository $catRepo) {
    }

    public function execute(): array {
        $res = [];
        foreach ($this->catRepo->getCategories() as $c) {
            $res[] = new CategoryData($c->getId(), $c->getName());
        }
        return $res;
    }
}

```

```

namespace Application;
class CategoryData {
    public function __construct(
        private int $id,
        private string $name
    ) { }

    public function getId() : int {
        return $this->id;
    }

    public function getName() : string {
        return $this->name;
    }
}

```

```

namespace Application;
class CategoryQuery {
    public function __construct(private Interfaces\CategoryRepository $catRepo) {
    }
}

```

```

    public function execute($categoryId): array {
        $res = [];
        foreach ($this->catRepo->getCategoryById($categoryId) as $c) {
            $res[] = new CategoryData($c->getId(), $c->getName());
        }
        return $res;
    }
}

```

```

namespace Application;
class CreateProductCommand {
    public function __construct(
        private Interfaces\ProductRepository $productRepository
    ) { }

    public function execute(\Application\Entities\Product $product):int {
        return $this->productRepository->createProduct($product);
    }
}

```

```

namespace Application;
class CreateReviewCommand {
    public function __construct(
        private \Application\Interfaces\ReviewRepository $reviewRepository
    ) { }

    public function execute(string $text, int $rating, int $productId, int $userId): bool {
        $reviewId = $this->reviewRepository->createReview($text, $rating, $productId, $userId);
        if ($reviewId != null) {
            return true;
        }
        return false;
    }
}

```

```

namespace Application;
class DeleteProductCommand {
    public function __construct(
        private \Application\Interfaces\ProductRepository $productRepository
    ) { }

    public function execute(int $productId) {
        $this->productRepository->deleteProduct($productId);
    }
}

```

```

    }
}

```

```

namespace Application;
class DeleteReviewCommand {
    public function __construct(
        private \Application\Interfaces\ReviewRepository $reviewRepository
    ) { }

    public function execute($reviewId) : int {
        return $this->reviewRepository->deleteReview($reviewId);
    }
}

```

```

namespace Application;
class EditProductCommand {
    public function __construct (
        private \Application\Interfaces\ProductRepository $productRepository
    ) {}

    public function execute(int $productId, string $name, string $info, int
$productId, int $categoryId) {
        $this->productRepository->editProduct($productId, $name, $info,
$productId, $categoryId);
    }
}

```

```

namespace Application;
class EditReviewCommand {
    public function __construct (
        private \Application\Interfaces\ReviewRepository $reviewRepository
    ) { }

    public function execute(int $id, string $text, int $rating): int {
        return $this->reviewRepository->editReview($id, $text, $rating);
    }
}

```

```

namespace Application;
class ProducerData {
    public function __construct (
        private int $producerId,
        private string $name
    ) { }
}

```

```
public function getProducerId(): int
{
    return $this->producerId;
}

public function getName(): string
{
    return $this->name;
}
}
```

```
namespace Application;
class ProducerQuery {
    public function __construct(
        private Interfaces\ProducerRepository $producerRepository
    ) { }

    public function execute(): array {
        $res = [];
        foreach ($this->producerRepository->getAllProducers() as $pr) {
            $res[] = new \Application\ProducerData($pr->getProducerId(), $pr-
>getName());
        }
        return $res;
    }
}
```

```
namespace Application;
class ProductData {
    public function __construct (
        private int $productId,
        private string $name,
        private string $info,
        private string $categoryName,
        private string $producerName,
        private int $userId,
        private float $rating
    ) { }

    public function getProductId(): int
    {
        return $this->productId;
    }

    public function getName(): string
    {
        return $this->name;
    }
}
```

```
public function getCategoryName(): string
{
    return $this->categoryName;
}

public function getInfo(): string
{
    return $this->info;
}

public function getProducerName(): string
{
    return $this->producerName;
}

public function getUserId(): int
{
    return $this->userId;
}

public function getRating(): float
{
    return $this->rating;
}
}
```

```
namespace Application;
class ProductQuery {
    public function __construct(
        private Interfaces\ProductRepository $productRepository
    ) { }

    public function execute (int $productId) : array {
        $res = [];
        foreach ($this->productRepository->getProductById($productId) as $p) {
            $res[] = new ProductData($p->getProductId(), $p->getName(), $p->
>getInfo(), $p->getCategoryName(), $p->getProducerName(), $p->getUserId(), $p-
>getRating());
        }
        return $res;
    }
}
```

```
namespace Application;
class ProductSearchQuery {
    public function __construct(
        private Interfaces\ProductRepository $productRepository
    ) { }
```

```

    public function execute(string $filter): array
    {
        $res = [];
        foreach($this->productRepository->getProductsForFilter($filter) as $p) {
            $res[] = new ProductData($p->getProductId(), $p->getName(), $p-
>getInfo(), $p->getCategoryName(), $p->getProducerName(), $p->getUserId(), $p-
>getRating());
        }
        return $res;
    }
}

```

```

namespace Application;
class ProductsQuery {
    public function __construct (
        private Interfaces\ProductRepository $productRepository,
    ) { }

    public function execute (string $categoryId) : array {
        $res = [];
        foreach ($this->productRepository->getProductsForCategory($categoryId) as
$p) {
            $res[] = new ProductData($p->getProductId(), $p->getName(), $p-
>getInfo(), $p->getCategoryName(), $p->getProducerName(), $p->getUserId(), $p-
>getRating());
        }
        return $res;
    }
}

```

```

namespace Application;
class RegisterCommand {
    public function __construct(
        private Services\AuthenticationService $authenticationService,
        private Interfaces\UserRepository $userRepository
    ) { }

    public function execute(string $username, string $password): bool
    {
        $this->authenticationService->signOut();
        $user = $this->userRepository->createNewUser($username, $password);
        if ($user != null) {
            $this->authenticationService->signIn($user->getid());
            return true;
        }
        return false;
    }
}

```

```
namespace Application;
class ReviewByProductQuery {
    public function __construct(
        private Interfaces\ReviewRepository $reviewRepository
    ) { }

    public function execute(int $productId) : array {
        $res = [];
        foreach ($this->reviewRepository->getReviewsByProductId($productId) as $r)
        {
            $res[] = new ReviewData($r->getReviewId(), $r->getUserId(), $r->getProductId(), $r->getText(), $r->getValue(), $r->getDateTime());
        }
        return $res;
    }
}
```

```
namespace Application;
class ReviewData {
    public function __construct (
        private int $reviewId,
        private int $userId,
        private int $productId,
        private string $text,
        private int $value,
        private string $dateTime
    ) { }

    public function getReviewId(): int
    {
        return $this->reviewId;
    }

    public function getUserId(): int
    {
        return $this->userId;
    }

    public function getProductId(): int
    {
        return $this->productId;
    }

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

    public function getValue(): int
    {

```

```
        return $this->value;
    }

    public function getDateTime(): string
    {
        return $this->dateTime;
    }
}
```

```
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->getId(), $user->getUserName());
    }
}
```

```
namespace Application;
class SignInCommand {
    public function __construct(
        private Services\AuthenticationService $authService,
        private Interfaces\UserRepository $userRepository
    ) { }

    public function execute(string $username, string $password): bool
    {
        $this->authService->signOut();
        $user = $this->userRepository->getUserForUserNameAndPassword($username,
$password);
        if ($user != null) {
            $this->authService->signIn($user->getId());
            return true;
        }
        return false;
    }
}
```



```
}  
}
```

```
namespace Application;  
class SignOutCommand {  
    public function __construct(  
        private Services\AuthenticationService $authService  
    ) { }  
  
    public function execute() : void {  
        $this->authService->signOut();  
    }  
}
```

```
namespace Application;  
class UserData  
{  
    public function __construct(  
        private int $id,  
        private string $userName  
    ) { }  
  
    public function getId(): int  
    {  
        return $this->id;  
    }  
  
    public function getUserName(): string  
    {  
        return $this->userName;  
    }  
}
```

```
namespace Application;  
class UserQuery  
{  
    public function __construct(  
        private Interfaces\UserRepository $userRepository  
    ) { }  
  
    public function execute(int $userId): UserData  
    {  
        $user = $this->userRepository->getUser($userId);  
        return new UserData($userId, $user->getUserName());  
    }  
}
```

Infrastructure Repository

```
namespace Infrastructure;
class Repository
implements
    \Application\Interfaces\ProductRepository,
    \Application\Interfaces\CategoryRepository,
    \Application\Interfaces\UserRepository,
    \Application\Interfaces\ReviewRepository,
    \Application\Interfaces\ProducerRepository
{
    private $server;
    private $userName;
    private $password;
    private $database;

    public function __construct(string $server, string $userName, string
$password, string $database)
    {
        $this->server = $server;
        $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 getCategories(): array
{
    $categories = [];
    $con = $this->getConnection();

    $res = $this->executeQuery($con, 'SELECT categoryId, name FROM category');

    while($cat = $res->fetch_object()) {
        $categories[] = new \Application\Entities\Category($cat->categoryId,
$cat->name);
    }

    $res->close();
    $con->close();

    return $categories;
}

public function getCategoryById($categoryId): array
{
    $categories = [];
    $con = $this->getConnection();

    $stat = $this->executeStatement(
        $con, 'SELECT categoryId, name FROM category WHERE categoryId = ?',
        function ($s) use ($categoryId) {
            $s->bind_param('i', $categoryId);
        }
    );
    $stat->bind_result($categoryId, $name);

    while($stat->fetch()) {
        $categories[] = new \Application\Entities\Category($cat->categoryId,
$cat->name);
    }

    $stat->close();
    $con->close();

    return $categories;
}
```

```

public function getProductsForCategory(int $categoryId): array
{
    $products = [];

    $con = $this->getConnection();

    $stat = $this->executeStatement(
        $con,
        'SELECT productId, p.name, info, c.name, pr.name as producerName,
p.userId, ROUND(AVG(value), 2) as rating
        FROM product p
        LEFT OUTER JOIN review r USING (productId)
        LEFT OUTER JOIN producer pr USING (producerId)
        LEFT OUTER JOIN category c USING (categoryId)
        WHERE categoryId = ?
        GROUP BY productId',
        function ($s) use ($categoryId) {
            $s->bind_param('i', $categoryId);
        }
    );
    $stat->bind_result($productId, $name, $info, $categoryName, $producerName,
    $userId, $rating);

    while($stat->fetch()) {
        $products[] = new \Application\ProductData($productId, $name, $info,
    $categoryName, $producerName, $userId, ($rating != null) ? $rating : 0.0);
    }
    $stat->close();
    $con->close();

    return $products;
}

public function getProductsForFilter(string $filter): array
{
    $products = [];

    $con = $this->getConnection();

    if ($filter == '') {
        $stat = $this->executeStatement(
            $con,
            'SELECT productId, p.name, info, c.name, pr.name as producerName,
p.userId, ROUND(AVG(value), 2) as rating
            FROM product p
            LEFT OUTER JOIN review r USING (productId)
            LEFT OUTER JOIN producer pr USING (producerId)
            LEFT OUTER JOIN category c USING (categoryId)
            GROUP BY productId',
            function ($s) use ($filter) { }
        );
    } else {
        $filterSql = "%$filter%";
        $stat = $this->executeStatement(

```

```

        $con,
        'SELECT productId, p.name, info, c.name, pr.name as producerName,
p.userId, ROUND(AVG(value), 2) as rating
        FROM product p
        LEFT OUTER JOIN review r USING (productId)
        LEFT OUTER JOIN producer pr USING (producerId)
        LEFT OUTER JOIN category c USING (categoryId)
        WHERE p.name LIKE ? OR pr.name LIKE ?
        GROUP BY productId',
        function ($s) use ($filterSql) {
            $s->bind_param('ss', $filterSql, $filterSql);
        }
    );
}
$stat->bind_result($productId, $name, $info, $categoryName, $producerName,
$userId, $rating);

while($stat->fetch()) {
    $products[] = new \Application\ProductData($productId, $name, $info,
$categoryName, $producerName, $userId, ($rating != null) ? $rating : 0.0);
}
$stat->close();
$con->close();

return $products;
}

public function getProductById(int $productId): array
{
    $products = [];

    $con = $this->getConnection();

    $stat = $this->executeStatement(
        $con,
        'SELECT productId, p.name, info, c.name, pr.name as producerName,
p.userId, AVG(value) as rating
        FROM product p
        LEFT OUTER JOIN review r USING (productId)
        LEFT OUTER JOIN producer pr USING (producerId)
        LEFT OUTER JOIN category c USING (categoryId)
        WHERE productId = ?
        GROUP BY productId',
        function ($s) use ($productId) {
            $s->bind_param('i', $productId);
        }
    );

    $stat->bind_result($productId, $name, $info, $categoryName, $producerName,
$userId, $rating);

    while($stat->fetch()) {
        $products[] = new \Application\ProductData($productId, $name, $info,
$categoryName, $producerName, $userId, ($rating != null) ? $rating : 0.0);
    }
}

```

```

        $stat->close();
        $con->close();

        return $products;
    }

    public function createProduct(\Application\Entities\Product $product): int
    {
        $con = $this->getConnection();
        $con->autocommit(false);

        $name = $product->getName();
        $info = $product->getInfo();
        $producerId = $product->getProducerId();
        $categoryId = $product->getCategoryId();
        $userId = $product->getUserId();

        $stat = $this->executeStatement(
            $con,
            'INSERT INTO product (name, info, producerId, categoryId, userId)
VALUES (?, ?, ?, ?, ?)',
            function ($s) use ($name, $info, $producerId, $categoryId, $userId) {
                $s->bind_param('ssiii', $name, $info, $producerId, $categoryId,
$userId);
            }
        );

        $productId = $stat->insert_id;
        $stat->close();
        $con->commit();
        $con->close();

        return $productId;
    }

    public function editProduct(int $productId, string $name, string $info, int
$producerId, int $categoryId): int
    {
        $con = $this->getConnection();
        $con->autocommit(false);

        $stat = $this->executeStatement(
            $con,
            'UPDATE product
            SET name = ?,
            info = ?,
            producerId = ?,
            categoryId = ?
            WHERE productId = ?',
            function ($s) use ($name, $info, $producerId, $categoryId, $productId)
{
                $s->bind_param('ssiii', $name, $info, $producerId, $categoryId,
$productId);
            }
        );
    }

```

```
);

$stat->close();
$con->commit();
$con->close();

return $productId;
}

public function deleteProduct(int $productId)
{
    $con = $this->getConnection();
    $con->autocommit(false);

    $stat = $this->executeStatement(
        $con,
        'DELETE FROM product WHERE productId = ?',
        function ($s) use ($productId) {
            $s->bind_param('i', $productId);
        }
    );

    $stat->close();
    $con->commit();
    $con->close();
}

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

public function getUserForUserNameAndPassword(string $userName, string
$password): ?\Application\Entities\User
{
    $user = null;
    $pw = password_hash($password, PASSWORD_DEFAULT);
    $con = $this->getConnection();
    $stat = $this->executeStatement(
```

```

        $con,
        'SELECT userId, name, passwordHash FROM user WHERE name = ?',
        function($s) use ($userName) {
            $s->bind_param('s', $userName);
        }
    );
    $stat->bind_result($id, $userName, $pwHash);
    if($stat->fetch()) {
        $user = new \Application\Entities\User($id, $userName, $pwHash);
    }
    $stat->close();
    $con->close();
    if ($user != null) {
        if (!password_verify($password, $user->getPwHash()))
            $user = null;
    }
    return $user;
}

public function createUser(string $userName, string $password): ?
\Application\Entities\User
{
    $con = $this->getConnection();
    $con->autocommit(false);

    $exists = $this->executeStatement(
        $con,
        'SELECT userId, name FROM user WHERE name = ?',
        function($s) use ($userName) {
            $s->bind_param('s', $userName);
        }
    );
    $exists->bind_result($id, $userName);
    if($exists->fetch()) {
        $user = new \Application\Entities\User($id, $userName);
    }

    if ($user != null) {
        return null;
    }

    $passwordHash = password_hash($password, PASSWORD_DEFAULT);

    $stat = $this->executeStatement(
        $con,
        'INSERT INTO user (name, passwordHash) VALUES (?, ?)',
        function ($s) use ($userName, $passwordHash) {
            $s->bind_param('ss', $userName, $passwordHash);
        }
    );

    $userId = $stat->insert_id;
    $stat->close();
    $con->commit();
}

```



```

        $con->close();

        return $this->getUser($userId);
    }

    public function createOrder(int $userId, array $bookIdsWithCount, string
$creditCardName, string $creditCardNumber): ?int
    {
        $con = $this->getConnection();
        $con->autocommit(false);

        $stat = $this->executeStatement(
            $con,
            'INSERT INTO orders (userId, creditCardHolder, creditCardNumber)
VALUES (?, ?, ?)',
            function ($s) use ($userId, $creditCardName, $creditCardNumber) {
                $s->bind_param('iss', $userId, $creditCardName,
$creditCardNumber);
            }
        );

        $orderId = $stat->insert_id;
        $stat->close();

        foreach($bookIdsWithCount as $bookId => $count) {
            for ($i = 0; $i < $count; $i++) {
                $this->executeStatement(
                    $con,
                    'INSERT INTO orderdBooks (orderId, bookId) VALUE (?, ?)',
                    function ($s) use ($orderId, $bookId) {
                        $s->bind_param('ii', $orderId, $bookId);
                    }
                )->close();
            }
        }

        $con->commit();
        $con->close();

        return $orderId;
    }

    public function getReviewsByProductId(int $productId): array
    {
        $reviews = [];

        $con = $this->getConnection();

        $stat = $this->executeStatement(
            $con,
            'SELECT * FROM review WHERE productId = ? ORDER BY date DESC',
            function ($s) use ($productId) {
                $s->bind_param('i', $productId);
            }
        );
    }

```

```

    );
    $stat->bind_result($reviewId, $userId, $productId, $text, $value,
$dateTime);

    while($stat->fetch()) {
        $reviews[] = new \Application\Entities\Review($reviewId, $userId,
$productId, $text, $value, $dateTime);
    }
    $stat->close();
    $con->close();

    return $reviews;
}

public function getReviewsByUserId(int $userId): array
{
    $reviews = [];

    $con = $this->getConnection();

    $stat = $this->executeStatement(
        $con,
        'SELECT * FROM review WHERE userId = ?',
        function ($s) use ($productId) {
            $s->bind_param('i', $productId);
        }
    );
    $stat->bind_result($reviewId, $userId, $productId, $text, $value,
$dateTime);

    while($stat->fetch()) {
        $reviews[] = new \Application\Entities\Review($reviewId, $userId,
$productId, $text, $value, $dateTime);
    }
    $stat->close();
    $con->close();

    return $reviews;
}

public function createReview(string $text, int $rating, int $productId, int
$userId): int
{
    $con = $this->getConnection();
    $con->autocommit(false);

    $stat = $this->executeStatement(
        $con,
        'INSERT INTO review (userId, productId, text, value) VALUES (?, ?, ?,
?)',
        function ($s) use ($userId, $productId, $text, $rating) {
            $s->bind_param('iisi', $userId, $productId, $text, $rating);
        }
    );
}

```

```
$reviewId = $stat->insert_id;
$stat->close();
$con->commit();
$con->close();

return $reviewId;
}

public function editReview(int $id, string $text, int $rating): int
{
    $con = $this->getConnection();
    $con->autocommit(false);

    $stat = $this->executeStatement(
        $con,
        'UPDATE review SET text = ?, value = ? WHERE reviewId = ?',
        function ($s) use ($text, $rating, $id) {
            $s->bind_param('sii', $text, $rating, $id);
        }
    );

    $stat->close();
    $con->commit();

    $getProduct = $this->executeStatement(
        $con,
        'SELECT productId FROM review WHERE reviewId = ?',
        function ($s) use ($id) {
            $s->bind_param('i', $id);
        }
    );

    $productId = null;
    $getProduct->bind_result($pid);
    if($getProduct->fetch()) {
        $productId = $pid;
    }

    $getProduct->close();
    $con->close();

    return $productId;
}

public function deleteReview(int $reviewId): int
{
    $con = $this->getConnection();
    $con->autocommit(false);

    $getProductId = $this->executeStatement(
        $con,
        'SELECT productId FROM review WHERE reviewId = ?',
        function ($s) use ($reviewId) {
```

```

        $s->bind_param('i', $reviewId);
    }
);
$productId->bind_result($product);

if($productId->fetch()) {
    $product = $product;
}

$productId->close();

$stat = $this->executeStatement(
    $con,
    'DELETE FROM review WHERE reviewId = ?',
    function ($s) use ($reviewId) {
        $s->bind_param('i', $reviewId);
    }
);

$stat->close();
$con->commit();
$con->close();
return $product;
}

public function getAllProducers(): array
{
    $producers = [];
    $con = $this->getConnection();

    $res = $this->executeQuery($con, 'SELECT producerId, name FROM producer');

    while($cat = $res->fetch_object()) {
        $producers[] = new \Application\Entities\Producer($cat->producerId,
$cat->name);
    }

    $res->close();
    $con->close();

    return $producers;
}
}

```

Infrastructure Session

```

namespace Infrastructure;
class Session implements \Application\Interfaces\Session {
    public function __construct() {
        session_start();
    }
}

```

```
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]);
}
}
```

Presentation

```
namespace Presentation\Controllers;
class Home extends \Presentation\MVC\Controller {
    public function __construct(
        private \Application\SignedInUserQuery $signedInUserQuery
    ) { }

    public function GET_Index() : \Presentation\MVC\ActionResult {
        return $this->view('home', [
            'user' => $this->signedInUserQuery->execute()
        ]);
    }
}
```

```
namespace Presentation\Controllers;
class Products extends \Presentation\MVC\Controller {
    const PARAM_CATEGORY_ID = 'cid';
    const PARAM_PRODUCT_ID = 'pid';
    const PARAM_FILTER = 'f';

    //Param for creation/editing
    const PARAM_EDIT_ID = 'eId';
    const PARAM_CREATE_NAME = 'cname';
    const PARAM_CREATE_INFO = 'cinfo';
    const PARAM_CREATE_PRODUCER = 'cproducer';
    const PARAM_CREATE_CATEGORY = 'ccategory';
    const PARAM_CREATE_ERROR = 'cerr';

    public function __construct(
        private \Application\CategoriesQuery $categoriesQuery,
        private \Application\ProductsQuery $productsQuery,
        private \Application\ProductQuery $productQuery,
        private \Application\ProductSearchQuery $productSearchQuery,
        private \Application\UserQuery $userQuery,
```

```

        private \Application\ReviewByProductQuery $reviewQuery,
        private \Application\SignedInUserQuery $signedInUserQuery,
        private \Application\ProducerQuery $producerQuery,
        private \Application\CreateProductCommand $createProductQuery,
        private \Application\EditProductCommand $editProductCommand,
        private \Application\DeleteProductCommand $deleteProductCommand
    ) { }

    public function GET_Index(): \Presentation\MVC\ActionResult {
        return $this->view('productlist', [
            'categories' => $this->categoriesQuery->execute(),
            'selectedCategoryId' => $this->tryGetParam(self::PARAM_CATEGORY_ID,
$value) ? $value : null,
            'products' => $this->tryGetParam(self::PARAM_CATEGORY_ID, $value) ?
$this->productsQuery->execute($value) : null,
            'context' => $this->getRequestUri(),
            'user' => $this->signedInUserQuery->execute()
        ]);
    }

    public function GET_Product(): \Presentation\MVC\ActionResult {
        $product = $this->tryGetParam(self::PARAM_PRODUCT_ID, $value) ? $this-
>productQuery->execute($value) : null;
        $user = $this->userQuery->execute($product[0]->getUserId());
        $reviewArray = $this->getReviewsInArray($product[0]->getProductId());
        $currentUser = $this->signedInUserQuery->execute();

        return $this->view('productView', [
            'product' => $product[0],
            'userName' => $user->getUserName(),
            'reviews' => $reviewArray,
            'context' => $this->getRequestUri(),
            'user' => $currentUser,
            'canEdit' => (($currentUser != null) ? $currentUser->getId() : -1) ===
$user->getId(),
            'errors' => [$this->tryGetParam(self::PARAM_CREATE_ERROR, $value) ?
$value : null]
        ]);
    }

    public function GET_Search(): \Presentation\MVC\ActionResult {
        $products = $this->tryGetParam(self::PARAM_FILTER, $value) ? $this-
>productSearchQuery->execute($value) : null;

        return $this->view('productsearch', [
            'products' => $products,
            'filter' => $this->tryGetParam(self::PARAM_FILTER, $value) ? $value :
null,
            'context' => $this->getRequestUri(),
            'user' => $this->signedInUserQuery->execute()
        ]);
    }

    public function GET_NewProduct(): \Presentation\MVC\ActionResult {

```

```

$this->tryGetParam(self::PARAM_EDIT_ID, $value);
if ($value == null) { // create new Product
    $producers = $this->producerQuery->execute();
    $categories = $this->categoriesQuery->execute();

    return $this->view('newProduct', [
        'producers' => $producers,
        'categories' => $categories,
        'user' => $this->signInUserQuery->execute()
    ]);
} else { // edit existing product
    $producers = $this->producerQuery->execute();
    $categories = $this->categoriesQuery->execute();

    return $this->view('newProduct', [
        'isEdit' => true,
        'eId' => $this->getParam(self::PARAM_EDIT_ID),
        'producers' => $producers,
        'categories' => $categories,
        'user' => $this->signInUserQuery->execute(),
        'name' => $this->getParam(self::PARAM_CREATE_NAME),
        'info' => $this->getParam(self::PARAM_CREATE_INFO),
        'cproducer' => $this->getParam(self::PARAM_CREATE_PRODUCER),
        'ccategory' => $this->getParam(self::PARAM_CREATE_CATEGORY)
    ]);
}
}

public function GET_Edit(): \Presentation\MVC\ActionResult {
    $productId = $this->getParam(self::PARAM_EDIT_ID);
    $name = $this->getParam(self::PARAM_CREATE_NAME);
    $info = $this->tryGetParam(self::PARAM_CREATE_INFO, $value) ? $value : "";
    $producerId = $this->getParam(self::PARAM_CREATE_PRODUCER);
    $categoryId = $this->getParam(self::PARAM_CREATE_CATEGORY);
    $user = $this->signInUserQuery->execute();

    $this->editProductCommand->execute($productId, $name, $info, $producerId,
    $categoryId);

    return $this->redirect('Products', 'Product', array('pid' => $productId));
}

public function GET_Delete(): \Presentation\MVC\ActionResult {
    $productId = $this->getParam(self::PARAM_PRODUCT_ID);
    $this->deleteProductCommand->execute($productId);

    return $this->redirect('Home', 'Index');
}

public function GET_Create(): \Presentation\MVC\ActionResult {
    $name = $this->getParam(self::PARAM_CREATE_NAME);
    $info = $this->tryGetParam(self::PARAM_CREATE_INFO, $value) ? $value : "";
    $producerId = $this->getParam(self::PARAM_CREATE_PRODUCER);
    $categoryId = $this->getParam(self::PARAM_CREATE_CATEGORY);

```

```

        $user = $this->signedInUserQuery->execute();

        $newProduct = new \Application\Entities\Product(-1, $name, $info,
$categoryId, $producerId, $user->getId());
        $productId = $this->createProductQuery->execute($newProduct);

        return $this->redirect('Products', 'Product', array('pid' => $productId));
    }

    private function getReviewsInArray(int $productId): array {
        $reviews = $this->reviewQuery->execute($productId);
        $reviewArray = [];

        foreach ($reviews as $review) {
            $reviewUser = $this->userQuery->execute($review->getUserId());
            $currentUser = $this->signedInUserQuery->execute();
            array_push($reviewArray, [
                'reviewId' => $review->getReviewId(),
                'userName' => $reviewUser->getUserName(),
                'dateTime' => $review->getDateTime(),
                'text' => $review->getText(),
                'value' => $review->getValue(),
                'user' => $currentUser,
                'canEdit' => (($currentUser != null) ? $currentUser->getId() : -1)
            ]);
        }
        return $reviewArray;
    }
}

```

```

namespace Presentation\Controllers;
class Review extends \Presentation\MVC\Controller {
    const PARAM_CREATE_TEXT = 'ctext';
    const PARAM_CREATE_RATING = 'crating';
    const PARAM_CREATE_PRODUCTID = 'cproductId';
    const PARAM_CREATE_USERID = 'cuserid';

    //edit params
    const PARAM_EDIT_ID = 'eid';
    const PARAM_EDIT_TEXT = 'etext';
    const PARAM_EDIT_RATING = 'erating';

    public function __construct(
        private \Application\CreateReviewCommand $createReviewCommand,
        private \Application>EditReviewCommand $editReviewCommand,
        private \Application>DeleteReviewCommand $deleteReviewCommand
    ) { }

    public function GET_NewReview() : \Presentation\MVC\ActionResult {
        $text = $this->getParam(self::PARAM_CREATE_TEXT);

```



```

        $rating = $this->getParam(self::PARAM_CREATE_RATING);
        $productId = $this->getParam(self::PARAM_CREATE_PRODUCTID);
        $userId = $this->getParam(self::PARAM_CREATE_USERID);

        if(!$this->createReviewCommand->execute($text, $rating, $productId,
$userId)) {
            return $this->redirect('Products', 'Product', array('pid' =>
$productId, 'cerr' => 'Creating review failed, please try again'));
        } else {
            return $this->redirect('Products', 'Product', array('pid' =>
$productId));
        }
    }

    public function GET_Edit() : \Presentation\MVC\ActionResult {
        $id = $this->getParam(self::PARAM_EDIT_ID);
        $text = $this->getParam(self::PARAM_EDIT_TEXT);
        $rating = $this->getParam(self::PARAM_EDIT_RATING);
        $productId = $this->editReviewCommand->execute($id, $text, $rating);

        return $this->redirect('Products', 'Product', array('pid' => $productId));
    }

    public function GET_Delete(): \Presentation\MVC\ActionResult {
        $rId = $this->getParam(self::PARAM_EDIT_ID);

        $productId = $this->deleteReviewCommand->execute($rId);

        return $this->redirect('Products', 'Product', array('pid' => $productId));
    }
}

```

```

namespace Presentation\Controllers;
class User extends \Presentation\MVC\Controller {

    const PARAM_USER_NAME = 'un';
    const PARAM_PASSWORD = 'pwd';
    const PARAM_PASSWORD2 = 'pwd2';

    public function __construct(
        private \Application\SignedInUserQuery $signedInUserQuery,
        private \Application\SignInCommand $signInCommand,
        private \Application\SignOutCommand $signOutCommand,
        private \Application\RegisterCommand $registerCommand
    ) { }

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

```

```

        return $this->view('login', [
            'user' => $user,
            'userName' => $this->tryGetParam(self::PARAM_USER_NAME, $value) ?
$value : ""
        ]);
    }

    public function POST_LogIn() : \Presentation\MVC\ActionResult {
        if (!$this->signInCommand->execute($this->getParam(self::PARAM_USER_NAME),
$this->getParam(self::PARAM_PASSWORD))) {
            return $this->view('login', [
                'user' => $this->signedInUserQuery->execute(),
                'userName' => $this->getParam(self::PARAM_USER_NAME),
                'errors' => ['Invalid Username or password']
            ]);
        }
        return $this->redirect('Home', 'Index');
    }

    public function GET_Register(): \Presentation\MVC\ActionResult {
        $user = $this->signedInUserQuery->execute();
        if ($user != null) {
            return $this->redirect('Home', 'Index');
        }
        return $this->view('register', [
            'user' => $user,
            'userName' => $this->tryGetParam(self::PARAM_USER_NAME, $value) ?
$value : ""
        ]);
    }

    public function POST_Register(): \Presentation\MVC\ActionResult {
        $pw1 = $this->getParam(self::PARAM_PASSWORD);
        $pw2 = $this->getParam(self::PARAM_PASSWORD2);

        if ($pw1 !== $pw2) {
            return $this->view('register', [
                'user' => null,
                'userName' => $this->tryGetParam(self::PARAM_USER_NAME, $value) ?
$value : "",
                'errors' => ['Passwords have to match']
            ]);
        }

        if (!$this->registerCommand->execute($this->
getParam(self::PARAM_USER_NAME), $this->getParam(self::PARAM_PASSWORD))) {
            return $this->view('register', [
                'user' => null,
                'userName' => $this->tryGetParam(self::PARAM_USER_NAME, $value) ?
$value : "",
                'errors' => ['Username already in use']
            ]);
        } else {
            return $this->redirect('Home', 'Index');
        }
    }

```

```

    }
}

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

```

Die restlichen Dateien in dem Ordner wurden bereits zur Verfügung gestellt.

Views

Die 404 Error Seite wurde von [diesem codepen](#) kopiert.

```

<html>
<head>
<title>Oops</title>
<link href="css/bootstrap.min.css" rel="stylesheet">
<style type="text/css">
/*=====
404 page
=====*/
.page_404{ padding:40px 0; background:#fff; font-family: 'Arvo',
serif;
}
.page_404 img{ width:100%;}
.four_zero_four_bg{
background-image:
url(https://cdn.dribbble.com/users/285475/screenshots/2083086/dribbble_1.gif);
height: 400px;
background-position: center;
background-repeat: no-repeat;
}
.four_zero_four_bg h1{
font-size:80px;
}
.four_zero_four_bg h3{
font-size:80px;
}
.link_404{
color: #fff!important;
padding: 10px 20px;
background: #39ac31;
margin: 20px 0;
display: inline-block;}
.contant_box_404{ margin-top:-50px;}
</style>
</head>
<body>
<div class="container">

```

```

<section class="page_404">
  <div class="container">
    <div class="row">
      <div class="col-sm-12 ">
        <div class="col-sm-10 col-sm-offset-1 text-center">
          <div class="four_zero_four_bg">
            <h1 class="text-center ">404</h1>
          </div>
          <div class="contant_box_404">
            <h3 class="h2">
              Look like you're lost
            </h3>
            <p>the page you are looking for not avaiable!</p>
            <a href="." class="link_404">Go to Home</a>
          </div>
        </div>
      </div>
    </div>
  </div>
</section>
</div>
</body>

</html>

```

```

<?php $render('partial/header', $data); ?>
  <h1>Welcome</h1>
  <p>Welcome to the SCR4 product review page!</p>

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

```

```

<?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">
  <?php if (isset($data['errors'])): ?>
    <p style="color: #bb2d3b"><?php $htmlOut($data['errors'][0]); ?></p>
  <?php endif; ?>
</div>
<button class="btn btn-primary">Log in</button>

```

```
<?php $endForm(); ?>

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

```
<?php $render('partial/header', $data); ?>
    <?php if (isset($data['isEdit'])): ?>
        <h1>Edit Product</h1>
    <?php else: ?>
        <h1>New Product</h1>
    <?php endif; ?>
    <div class="my-3">
        <?php
            if (isset($data['isEdit']))
                $beginForm('Products', 'Edit');
            else
                $beginForm('Products', 'Create');
        ?>
        <div class="mb-3">
            <label for="cname" class="form-label">Product name:</label>
            <input type="text" class="form-control" placeholder="Name"
name="cname" id="cname" value="<?php if (isset($data['name'])) echo $data['name'];
?>" required />
        </div>
        <div class="mb-3">
            <label for="cinfo" class="form-label">Product info:</label>
            <textarea class="form-control" rows="3" id="cinfo" name="cinfo"><?
php if (isset($data['info'])) echo $data['info']; else echo ''; ?></textarea>
        </div>
        <div class="mb-3">
            <div class="row">
                <div class="col">
                    <label for="cproducer" class="form-label">Producer:
</label>
                    <select class="form-select" name="cproducer"
id="cproducer">
                        <?php foreach ($data['producers'] as $pr){
                            echo '<option value="'. $pr->getProducerId().'" '.
((isset($data['cproducer']) && $data['cproducer'] == $pr->getName()) ? 'selected'
: '').>'.
                                $pr->getName(). '</option>';
                        } ?>
                    </select>
                </div>
                <div class="col">
                    <label for="ccategory" class="form-label">Category:
</label>
                    <select class="form-select" name="ccategory"
id="ccategory">
                        <?php foreach ($data['categories'] as $pr){
                            echo '<option value="'. $pr->getId().'" '.
((isset($data['ccategory']) && $data['ccategory'] == $pr->getName()) ? ' selected'
```

```

: ')).'>'.
                                $pr->getName().'</option>';
                                } ?>
                                </select>
                                </div>
                                </div>
                                </div>
                                <div class="mb-3">
                                    <input type="hidden" name="eId" value="<?php if
(isset($data['eId'])) echo $data['eId']; else echo 'help'; ?>">
                                    <button class="btn btn-primary">Submit</button>
                                </div>
                                <?php $endForm(); ?>
                                </div>

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

```

```

<?php $render('partial/header', $data); ?>
<h1>List of products</h1>

<nav class="nav nav-pills my-3">
    <?php
        foreach($data['categories'] as $cat) { ?>
            <?php $link($cat->getName(), 'Products', 'Index',
                array('cid' => $cat->getId()), 'nav-item nav-link' . ($cat->getId() ==
                    $data['selectedCategoryId'] ? ' active' : '')); ?>
            <?php } ?>
        </nav>

<?php if($data['selectedCategoryId'] !== null): ?>
    <?php if (sizeof($data['products']) > 0) { ?>

        <?php $render('partial/products', [
            'products' => $data['products'],
            'context' => $data['context']]); ?>

        <?php } else { ?>
            <p>No products in this category</p>
        <?php } ?>
    <?php else: ?>
        <p>Please select a category</p>
    <?php endif; ?>

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

```

```

<?php $render('partial/header', $data); ?>
<h1>Search result</h1>

```

```

<div class="my-3">
  <?php $beginForm('Products', 'Search'); ?>
  <div class="row g-3">
    <div class="col-auto">
      <input class="form-control" name="f" />
    </div>
    <div class="col-auto">
      <button class="btn btn-primary">Search</button>
    </div>
  </div>
  <?php $endForm(); ?>
</div>

<?php if ($data['products'] !== null): ?>
  <?php if (sizeof($data['products']) > 0) {
    $render('partial/products', [
      'products' => $data['products'],
      'context' => $data['context']
    ]);
  } ?>
<?php endif; ?>

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

```

```

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

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

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

```

```

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

<h1>Register</h1>

<?php $beginForm('User', 'Register', 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']); ?>" required>
</div>
<div class="mb-3">
  <label for="password" class="form-label">Password</label>
  <input type="password" class="form-control" id="password" name="pwd"
required>
  <label for="password2" class="form-label">Repeat Password</label>
  <input type="password" class="form-control" id="password2" name="pwd2"
required>
  <?php if (isset($data['errors'])): ?>
    <p style="color: #bb2d3b"><?php $htmlOut($data['errors'][0]); ?></p>

```

```

        <?php endif; ?>
    </div>
    <button class="btn btn-primary">Register</button>
<?php $endForm(); ?>

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

```

Partials

```

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

```

```

<!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">
    <title>SCR4 Book Shop</title>
</head>

<body>
    <nav class="navbar navbar-expand-lg navbar-dark bg-dark">
        <div class="container">
            <?php $link('SCR4 Product Reviews', '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">
                    <?php $link('Home', 'Home', 'Index', cssClass: 'nav-item nav-
link');?>
                    <?php $link('List', 'Products', 'Index', cssClass: 'nav-item
nav-link');?>
                    <?php $link('Search', 'Products', 'Search', cssClass: 'nav-
item nav-link');?>
                    <?php if (isset($data['user'])): ?>
                        <?php $link('New Product', 'Products', 'NewProduct', cssClass:
'nav-item nav-link'); ?>

```



```

        <?php endif; ?>
    </nav>
    <?php $render('partial/user', $data['user']); ?>
</div>
</div>
</nav>
<div class="container mt-3">

```

```

<a href="javascript:history.go(-1)"> &#60; Back</a>
<div class="row">
    <div class="col-11">
        <h1><?php $htmlOut($data['product']->getName()); ?></h1>
    </div>
    <div class="col-1 d-flex justify-content-end">
        <?php if ($data['canEdit']):
            $beginForm('Products', 'NewProduct', array(
                'eId' => $data['product']->getProductId(),
                'cname' => $data['product']->getName(),
                'cinfo' => $data['product']->getInfo(),
                'cproducer' => $data['product']->getProducerName(),
                'ccategory' => $data['product']->getCategoryName()
            )); ?>
            <button class="btn btn-outline-success m-1" type="submit" value="">
                <svg xmlns="http://www.w3.org/2000/svg" width="16" height="16"
fill="currentColor" class="bi bi-pencil" viewBox="0 0 16 16">
                    <path d="M12.146 14.646a.5.5 0 0 1 .708 0l3 3a.5.5 0 0 1 0 .708l-10 10a.5.5 0 0 1-.168.111l-5 2a.5.5 0 0 1-.65-.65l2-5a.5.5 0 0 1 .11-.168l10-10zm11.207 2.5 13.5 4.793 14.793 3.5 12.5 1.207 11.207 2.5zm1.586 3L10.5 3.207 4
9.707V10h.5a.5.5 0 0 1 .5.5v.5a.5.5 0 0 1 .5.5v.5h.293l6.5-6.5zm-9.761
5.175-.106 106-1.528 3.821 3.821-1.528.106A.5.5 0 0 1 5 12.5V12h-.5a.5.5 0 0
1-.5-.5V11h-.5a.5.5 0 0 1-.468-.325z"/>
                </svg>
            </button>
            <?php $endForm();

            $beginForm('Products', 'Delete', array(
                'pid' => $data['product']->getProductId()
            )); ?>
            <button class="btn btn-outline-danger m-1" type="submit" value="">
                <svg xmlns="http://www.w3.org/2000/svg" width="16" height="16"
fill="currentColor" class="bi bi-trash" viewBox="0 0 16 16">
                    <path d="M5.5 5.5A.5.5 0 0 1 6 6v6a.5.5 0 0 1-1 0V6a.5.5 0 0 1
.5-.5zm2.5 0a.5.5 0 0 1 .5.5v6a.5.5 0 0 1-1 0V6a.5.5 0 0 1 .5-.5zm3 .5a.5.5 0 0 0-
1 0v6a.5.5 0 0 0 1 0V6z"/>
                    <path fill-rule="evenodd" d="M14.5 3a1 1 0 0 1-1 1H13v9a2 2 0
0 1-2 2H5a2 2 0 0 1-2 2V4h-.5a1 1 0 0 1-1 1V2a1 1 0 0 1 1-1H6a1 1 0 0 1 1-1h2a1 1
0 0 1 1 1h3.5a1 1 0 0 1 1 1v1zM4.118 4 4 4.059V13a1 1 0 0 0 1 1h6a1 1 0 0 0 1-
1V4.059L11.882 4H4.118zM2.5 3V2h11v1h-11z"/>
                </svg>
            </button>
            <?php $endForm();

```

```

        endif; ?>
    </div>
</div>

<h5>Category: <?php $htmlOut($data['product']->getCategoryName()); ?> | created
by: <?php $htmlOut($data['userName']); ?></h5>
<hr/>

<h4>Info:</h4>

<p>
    <?php $htmlOut($data['product']->getInfo()); ?>
</p>

<h4>Reviews:</h4>

<?php $hasLeftReview = false;
    if (!isset($data['user'])) {
        $hasLeftReview = true;
    } else {
        foreach ($data['reviews'] as $review) {
            if ($review['userName'] == $data['user']->getUserName()) {
                $hasLeftReview = true;
                break;
            }
        }
    }
    ?>

<?php if (!$hasLeftReview): ?>
<a class="btn btn-primary mb-2" data-bs-toggle="collapse" href="#newReview"
role="button" aria-expanded="<?php echo isset($data['errors']); ?>" aria-
controls="newReview">
    Add Review
</a>

<div class="collapse" id="newReview">
    <div class="card" style="margin-bottom: 10px;">
        <div class="card-body">
            <?php $beginForm('Review', 'NewReview') ?>
            <?php if (isset($data['errors'])) {?>
                <p style="color: #bb2d3b"><?php $htmlOut($data['errors'][0]);
?></p>

                <?php }?>
                <label for="ctext" class="form-label">Text</label>
                <textarea class="form-control" id="ctext" name="ctext" cols="2">

</textarea>

                <label for="crating" class="form-label">Rating</label>
                <select name="crating" id="crating" class="form-select">
                    <option value="1">1 - very bad</option>
                    <option value="2">2 - bad</option>
                    <option value="3">3 - medium</option>
                    <option value="4">4 - good</option>
                    <option value="5">5 - very good</option>

```

```

        </select>
        <input type="hidden" name="cproductId" value="<?php
$htmlOut($data['product']->getProductId()); ?>" />
        <input type="hidden" name="cuserId" value="<?php
$htmlOut($data['user']->getId()); ?>" />
        <button class="btn btn-primary form-control mt-2">Submit</button>
        <?php $endForm(); ?>
    </div>
</div>
</div>

<?php endif; ?>

<?php foreach ($data['reviews'] as $review): ?>

    <?php $render('partial/reviewCard', $review); ?>

<?php endforeach; ?>

```

```

<table class="table">
    <thead>
        <tr>
            <th>Name</th>
            <th>Info</th>
            <th>Producer</th>
            <th style="text-align: center">Rating</th>
            <th></th>
        </tr>
    </thead>

    <tbody>
        <?php foreach ($data['products'] as $b): ?>
            <tr>
                <td><?php $link($b->getName(), 'Products', 'Product', array('pid'
=> $b->getProductId()), 'nav-item nav-link') ?></td>
                <td><?php $htmlOut($b->getInfo()); ?></td>
                <td><?php $htmlOut($b->getProducerName()); ?></td>
                <td align="center"><?php
                    $rating = $b->getRating();
                    if ($rating == 0)
                        $htmlOut("No Ratings yet");
                    else
                        $htmlOut($b->getRating()); ?></td>
            </tr>
        <?php endforeach; ?>
    </tbody>

</table>

```

```

<?php $reviewId = $data['reviewId']; ?>

<div class="card" style="margin-bottom: 10px;">
    <div class="card-header">
        <div class="row">
            <div class="col-10">
                <h5 class="card-title">Review by <?php
$htmlOut($data['userName']); ?></h5>
                <h6 class="card-subtitle"><?php $htmlOut($data['dateTime']); ?>
            </h6>
        </div>
        <div class="col-2 d-flex justify-content-end">
            <?php if ($data['canEdit']): ?>
                <button class="btn btn-outline-success m-1" type="button" data-bs-
toggle="collapse" data-bs-target=".edit-review<?php echo $reviewId; ?>" aria-
controls="edit display">
                    <svg xmlns="http://www.w3.org/2000/svg" width="16" height="16"
fill="currentColor" class="bi bi-pencil" viewBox="0 0 16 16">
                        <path d="M12.146 14.646a.5.5 0 0 1 .708 0l3 3a.5.5 0 0 1 0
.708l-10 10a.5.5 0 0 1 -.168.111l-5 2a.5.5 0 0 1 -.65-.65l2-5a.5.5 0 0 1 .11-.168l10-
10zM11.207 2.5 13.5 4.793 14.793 3.5 12.5 1.207 11.207 2.5zM1.586 3L10.5 3.207 4
9.707V10h.5a.5.5 0 0 1 .5.5v.5h.5a.5.5 0 0 1 .5.5v.5h.293l6.5-6.5zm-9.761
5.175-.106 106-1.528 3.821 3.821-1.528 106-.106A.5.5 0 0 1 5 12.5V12h-.5a.5.5 0 0
1 -.5-.5V11h-.5a.5.5 0 0 1 -.468-.325z"/>
                    </svg>
                </button>
                <?php $beginForm('Review', 'Delete', array(
                    'eid' => $data['reviewId']
                )); ?>
                <button class="btn btn-outline-danger m-1" type="submit">
                    <svg xmlns="http://www.w3.org/2000/svg" width="16" height="16"
fill="currentColor" class="bi bi-trash" viewBox="0 0 16 16">
                        <path d="M5.5 5.5A.5.5 0 0 1 6 6v6a.5.5 0 0 1 1 1 0V6a.5.5 0
0 1 .5-.5zm2.5 0a.5.5 0 0 1 .5.5v6a.5.5 0 0 1 1 1 0V6a.5.5 0 0 1 .5-.5zm3 .5a.5.5 0
0 1 0 0 1 0V6a.5.5 0 0 1 0 0 1 0V6z"/>
                        <path fill-rule="evenodd" d="M14.5 3a1 1 0 0 1 1 1H13v9a2
2 0 0 1-2 2H5a2 2 0 0 1-2 2V4h-.5a1 1 0 0 1-1 1V2a1 1 0 0 1 1-1H6a1 1 0 0 1 1-
1h2a1 1 0 0 1 1 1h3.5a1 1 0 0 1 1 1v1zM4.118 4 4.059V13a1 1 0 0 1 1h6a1 1 0 0
0 1-1V4.059L11.882 4H4.118zM2.5 3V2h11v1h-11z"/>
                    </svg>
                </button>
            <?php $endForm(); ?>
            <?php endif; ?>
        </div>
    </div>
</div>
<div class="card-body">
    <div class="edit-review<?php echo $reviewId; ?> show" id="display">
        <p class="card-text"><?php $htmlOut($data['text']); ?></p>
        <?php $total = 5; for ($i = 1; $i <= $data['value']; $i++): ?>
            <svg xmlns="http://www.w3.org/2000/svg" width="16" height="16"
fill="currentColor" class="bi bi-star-fill" viewBox="0 0 16 16">
                <path d="M3.612 15.443c-.386.198-.824-.149-.746-.746l.03-.291.83-

```

```

4.73L.173
6.765c-.329-.314-.158-.888.283-.9514.898-.696L7.538.792c.197-.39.73-.39.927
012.184 4.327 4.898.696c.441.062.612.636.282.951-3.522 3.356.83
4.73c.078.443-.36.79-.746.592L8 13.1871-4.389 2.256z"/>
    </svg>
    <?php $total--; endfor;
    for ($i = 1; $i <= $total; $i++): ?>
        <svg xmlns="http://www.w3.org/2000/svg" width="16" height="16"
fill="currentColor" class="bi bi-star" viewBox="0 0 16 16">
            <path d="M2.866 14.85c-.078.444.36.791.746.59314.39-2.256 4.389
2.256c.386.198.824-.149.746-.5921-.83-4.73 3.522-3.356c.33-.314.16-.888-.282-.951-
4.898-.696L8.465.792a.513.513 0 0 0-.927 0L5.354 5.121-
4.898.696c-.441.062-.612.636-.283.9513.523 3.356-.83 4.73zm4.905-2.767-3.686
1.894.694-3.957a.565.565 0 0 0-.163-.505L1.71 6.74514.052-.576a.525.525 0 0 0
.393-.288L8 2.22311.847 3.658a.525.525 0 0 0 .393.28814.052.575-2.906
2.77a.565.565 0 0 0-.163.5061.694 3.957-3.686-1.894a.503.503 0 0 0-.461 0z"/>
        </svg>
    <?php endfor; ?>
</div>
<div class="edit-review"<?php echo $reviewId; ?> collapse" id="edit">
    <?php $beginForm('Review', 'Edit'); ?>
    <textarea class="form-control" rows="3" id="etext" name="etext"><?php
$htmlOut($data['text']); ?></textarea>
    <input type="hidden" name="eid" value="<?php
$htmlOut($data['reviewId']); ?>">
    <div class="row mt-3">
        <div class="col-6">
            <select class="form-control form-select" name="erating">
                <option value="1" <?php if ($data['value'] == 1) echo
'selected';?>>1 - very bad</option>
                <option value="2" <?php if ($data['value'] == 2) echo
'selected';?>>2 - bad</option>
                <option value="3" <?php if ($data['value'] == 3) echo
'selected';?>>3 - medium</option>
                <option value="4" <?php if ($data['value'] == 4) echo
'selected';?>>4 - good</option>
                <option value="5" <?php if ($data['value'] == 5) echo
'selected';?>>5 - very good</option>
            </select>
        </div>
        <div class="col-6 d-flex justify-content-end">
            <button class="btn btn-primary" type="submit">Save
review</button>
        </div>
    </div>
    <?php $endForm(); ?>
</div>
</div>
</div>

```

```

<?php if (!isset($data)): ?>
    <nav class="navbar-nav">
        <?php $link('Login', 'User', 'LogIn', cssClass: 'nav-item 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-
>getUserName()); ?></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();

// --- Infrastructure
$sp->register(\Infrastructure\Session::class, isSingleton: true);
$sp->register(\Application\Interfaces\Session::class,
\Infrastructure\Session::class);
$sp->register(\Infrastructure\Repository::class, function() {
    return new \Infrastructure\Repository('localhost', 'root', '',
'productreviews');
}, isSingleton: true);
$sp->register(\Application\Interfaces\CategoryRepository::class,
\Infrastructure\Repository::class);
$sp->register(\Application\Interfaces\ProductRepository::class,
\Infrastructure\Repository::class);
$sp->register(\Application\Interfaces\UserRepository::class,
\Infrastructure\Repository::class);
$sp->register(\Application\Interfaces\ReviewRepository::class,
\Infrastructure\Repository::class);
$sp->register(\Application\Interfaces\ProducerRepository::class,
\Infrastructure\Repository::class);

// --- Application
$sp->register(\Application\CategoriesQuery::class);
$sp->register(\Application\CategoryQuery::class);

```

```
$sp->register(\Application\ProductsQuery::class);
$sp->register(\Application\ProductQuery::class);
$sp->register(\Application\ProducerQuery::class);
$sp->register(\Application\UserQuery::class);
$sp->register(\Application\ReviewByProductQuery::class);
$sp->register(\Application\CreateReviewCommand::class);
$sp->register(\Application\EditReviewCommand::class);
$sp->register(\Application\DeleteReviewCommand::class);
$sp->register(\Application\ProductSearchQuery::class);
$sp->register(\Application\CreateProductCommand::class);
$sp->register(\Application\DeleteProductCommand::class);
$sp->register(\Application\EditProductCommand::class);
$sp->register(\Application\SignedInUserQuery::class);
$sp->register(\Application\SignInCommand::class);
$sp->register(\Application\SignOutCommand::class);
$sp->register(\Application\RegisterCommand::class);

// --- Services
$sp->register(\Application\Services\AuthenticationService::class);

// --- Presentation
// MVC Framework
$sp->register(\Presentation\MVC\MVC::class, function() {
    return new \Presentation\MVC\MVC();
}, isSingleton: true);

// Controllers
$sp->register(\Presentation\Controllers\Home::class);
$sp->register(\Presentation\Controllers\Products::class);
$sp->register(\Presentation\Controllers\User::class);
$sp->register(\Presentation\Controllers\Review::class);

// --- handle Request
$sp->resolve(\Presentation\MVC\MVC::class)->handleRequest($sp);
```

Stylesheets

Es wurde nur das zur Verfügung gestellte bootstrap stylesheet verwendet.

SQL Script zum erstellen und befüllen der DB

```
-- phpMyAdmin SQL Dump
-- version 5.0.4
-- https://www.phpmyadmin.net/
--
-- Host: 127.0.0.1
-- Generation Time: Jun 03, 2021 at 03:49 PM
-- Server version: 10.4.17-MariaDB
-- PHP Version: 8.0.0

SET SQL_MODE = "NO_AUTO_VALUE_ON_ZERO";
```

```
START TRANSACTION;
SET time_zone = "+00:00";

/*!40101 SET @OLD_CHARACTER_SET_CLIENT=@@CHARACTER_SET_CLIENT */;
/*!40101 SET @OLD_CHARACTER_SET_RESULTS=@@CHARACTER_SET_RESULTS */;
/*!40101 SET @OLD_COLLATION_CONNECTION=@@COLLATION_CONNECTION */;
/*!40101 SET NAMES utf8mb4 */;

--
-- Database: `productreviews`
--

--
-- -----
--
-- Table structure for table `category`
--

CREATE TABLE `category` (
  `categoryId` int(11) NOT NULL,
  `name` varchar(50) NOT NULL
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4;

--
-- Dumping data for table `category`
--

INSERT INTO `category` (`categoryId`, `name`) VALUES
(1, 'books'),
(2, 'electronics'),
(3, 'clothes');

--
-- -----
--
-- Table structure for table `producer`
--

CREATE TABLE `producer` (
  `producerId` int(11) NOT NULL,
  `name` varchar(50) NOT NULL
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4;

--
-- Dumping data for table `producer`
--

INSERT INTO `producer` (`producerId`, `name`) VALUES
(1, 'Logitech'),
(2, 'Corsair');
```



```
--
-- Table structure for table `product`
--

CREATE TABLE `product` (
  `productId` int(11) NOT NULL,
  `name` varchar(50) NOT NULL,
  `info` varchar(255) DEFAULT NULL,
  `producerId` int(11) NOT NULL,
  `categoryId` int(11) NOT NULL,
  `userId` int(11) NOT NULL
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4;

--
-- Dumping data for table `product`
--

INSERT INTO `product` (`productId`, `name`, `info`, `producerId`, `categoryId`,
`userId`) VALUES
(2, 'G Pro Wireless Mouse', 'A very light wireless mouse with 4 configurable
buttons.', 1, 2, 1),
(19, 'Test', 'Lorem ipsum dolor sit amet, consectetur adipiscing elit. Nam
vestibulum et mi a laoreet. Sed vel justo quis sem commodo maximus quis vel purus.
Quisque vestibulum pretium ligula, eget pharetra nibh congue vel. Ut finibus
ornare nisl sit amet vivamus. ', 1, 1, 1),
(23, 'Z333 Speakers', 'The best value speakers on the market.', 1, 2, 5);

-----

--
-- Table structure for table `review`
--

CREATE TABLE `review` (
  `reviewId` int(11) NOT NULL,
  `userId` int(11) DEFAULT NULL,
  `productId` int(11) NOT NULL,
  `text` varchar(255) NOT NULL,
  `value` int(1) NOT NULL,
  `date` datetime NOT NULL DEFAULT current_timestamp()
) ;

--
-- Dumping data for table `review`
--

INSERT INTO `review` (`reviewId`, `userId`, `productId`, `text`, `value`, `date`)
VALUES
(12, 1, 2, 'I like it very much', 5, '2021-06-03 13:12:23'),
(14, 4, 2, 'Test', 2, '2021-06-03 13:13:11');

-----

--
```

```
-- Table structure for table `user`
--

CREATE TABLE `user` (
  `userId` int(11) NOT NULL,
  `name` char(60) NOT NULL,
  `passwordHash` varchar(255) NOT NULL
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4;

--
-- Dumping data for table `user`
--

INSERT INTO `user` (`userId`, `name`, `passwordHash`) VALUES
(1, 'admin', '$2y$10$IeJztEK2w0M0tyFXHu0tub3ReU8Lsd0eao1ea43RGZZMa3wlqLpS'),
(4, 'Testuser', '$2y$10$/DHgVlJMaX4pNgtKtdziIOsEiWEznR8qeZUDvpWsYshjeV0kLTL6i'),
(5, 'scr4', '$2y$10$B5fy/vsxFavfN6QW093DzuLfeuVHzV0C770zdOUsul0TNlTx.lw4. ');

--
-- Indexes for dumped tables
--

--
-- Indexes for table `category`
--
ALTER TABLE `category`
  ADD PRIMARY KEY (`categoryId`);

--
-- Indexes for table `producer`
--
ALTER TABLE `producer`
  ADD PRIMARY KEY (`producerId`);

--
-- Indexes for table `product`
--
ALTER TABLE `product`
  ADD PRIMARY KEY (`productId`),
  ADD KEY `producerId` (`producerId`),
  ADD KEY `categoryId` (`categoryId`),
  ADD KEY `userId` (`userId`);

--
-- Indexes for table `review`
--
ALTER TABLE `review`
  ADD PRIMARY KEY (`reviewId`),
  ADD KEY `userId` (`userId`),
  ADD KEY `productId` (`productId`);

--
-- Indexes for table `user`
--
```

```
ALTER TABLE `user`
  ADD PRIMARY KEY (`userId`);

--
-- AUTO_INCREMENT for dumped tables
--

--
-- AUTO_INCREMENT for table `category`
--
ALTER TABLE `category`
  MODIFY `categoryId` int(11) NOT NULL AUTO_INCREMENT, AUTO_INCREMENT=6;

--
-- AUTO_INCREMENT for table `producer`
--
ALTER TABLE `producer`
  MODIFY `producerId` int(11) NOT NULL AUTO_INCREMENT, AUTO_INCREMENT=3;

--
-- AUTO_INCREMENT for table `product`
--
ALTER TABLE `product`
  MODIFY `productId` int(11) NOT NULL AUTO_INCREMENT, AUTO_INCREMENT=24;

--
-- AUTO_INCREMENT for table `review`
--
ALTER TABLE `review`
  MODIFY `reviewId` int(11) NOT NULL AUTO_INCREMENT;

--
-- AUTO_INCREMENT for table `user`
--
ALTER TABLE `user`
  MODIFY `userId` int(11) NOT NULL AUTO_INCREMENT, AUTO_INCREMENT=6;

--
-- Constraints for dumped tables
--

--
-- Constraints for table `product`
--
ALTER TABLE `product`
  ADD CONSTRAINT `product_ibfk_1` FOREIGN KEY (`producerId`) REFERENCES `producer`
(`producerId`) ON UPDATE CASCADE,
  ADD CONSTRAINT `product_ibfk_2` FOREIGN KEY (`userId`) REFERENCES `user`
(`userId`) ON UPDATE CASCADE,
  ADD CONSTRAINT `product_ibfk_3` FOREIGN KEY (`categoryId`) REFERENCES `category`
(`categoryId`) ON UPDATE CASCADE;

--
-- Constraints for table `review`
```

```
--  
ALTER TABLE `review`  
  ADD CONSTRAINT `review_ibfk_1` FOREIGN KEY (`productId`) REFERENCES `product`  
  (`productId`) ON DELETE CASCADE ON UPDATE CASCADE,  
  ADD CONSTRAINT `review_ibfk_2` FOREIGN KEY (`userId`) REFERENCES `user`  
  (`userId`) ON DELETE SET NULL ON UPDATE CASCADE;  
COMMIT;  
  
/*!40101 SET CHARACTER_SET_CLIENT=@OLD_CHARACTER_SET_CLIENT */;  
/*!40101 SET CHARACTER_SET_RESULTS=@OLD_CHARACTER_SET_RESULTS */;  
/*!40101 SET COLLATION_CONNECTION=@OLD_COLLATION_CONNECTION */;
```