

Inhaltsverzeichnis

Allgemeine Lösungsidee.....	3
Architektur.....	5
Datenmodell.....	7
Erstellungsskript.....	9
Code.....	11
Index.php.....	11
styles.css.....	12
views/home.inc.....	12
views/login.inc.....	13
views/productCreate.inc.....	14
views/productList.inc.....	15
views/productSearch.inc.....	15
views/ratingList.inc.....	16
views/partial/errors.inc.....	16
views/partial/footer.inc.....	16
views/partial/header.inc.....	17
views/partial/products.inc.....	18
views/partial/ratingAdd.inc.....	19
views/partial/ratings.inc.....	20
views/partial/user.inc.....	21
Presentation.....	22
Controllers/Home.php.....	22
Controllers/Product.php.....	23
Controllers/Rating.php.....	26
Controllers/User.php.....	29
Infrastructure.....	31
Repository.php.....	31
Session.php.....	40
Application.....	41
Entities/Product.php.....	41
Entities/Rating.php.....	42
Entities/User.php.....	43
Interfaces/ProductRepository.php.....	44
Interfaces/RatingRepository.php.....	44
Interfaces/UserRepository.php.....	45
Interfaces/Session.php.....	45
Services/AuthenticationService.php.....	46
ProductData.php.....	47
RatingData.php.....	48
UserData.php.....	49
ProductCreationCommand.php.....	50
ProductQuery.php.....	51

ProductSearchQuery.php.....	52
ProductsQuery.php.....	53
RatingCreationCommand.php.....	54
RatingDeleteCommand.php.....	55
RatingsQuery.php.....	56
RegisterCommand.php.....	57
SignedInUserQuery.php.....	58
SignInCommand.php.....	58
SignOutCommand.php.....	59
Testfälle.....	60

Allgemeine Lösungsidee

Bei der Implementierung wird grundsätzlich darauf geachtet, dass die gesamte geforderte Funktionalität auch umgesetzt wird. Dabei habe ich die Funktionalität auf folgende Screens aufgeteilt:

1. Index/Home

Soll eine Willkommensnachricht und Hinweise zu der Navigation auf der Website anzeigen

2. Explore

Hier sollen (anonyme) User alle Produkte inklusive Durchschnittsbewertung und Anzahl der Bewertungen sehen. Diese werden in Form einer Liste angezeigt. Ich habe mich dafür entschieden zusätzlich noch ein Bild anzuzeigen. Diese werden als base64 String in der DB abgespeichert. Ich habe aber bereits beim Testen mit vielen verschiedenen Produkten gemerkt, dass die Get-Requests dadurch merklich langsamer werden. Für den durchschaubaren Datensatz im Übungsprojekt scheint es für mich aber in Ordnung – Mir war auch nicht klar, wie es anders umgesetzt werden kann.

Für eingeloggte User, wird für alle Produkte die er erstellt hat, ein „Edit“ Button in der dazugehörigen Zeile angezeigt. Über diesen kann das Produkt editiert werden. Eine Löschfunktionalität ist in der Angabe nicht gefordert und wurde nicht umgesetzt (Würde ein Produkt gelöscht werden, so müssten auch alle dazugehörigen Ratings gelöscht werden; Steht für mich in Widerspruch mit der Anforderung, dass der Ersteller keine Ratings/Kommentare löschen darf).

Durch Drücken auf den Produktnamen gelangt man zur Detailansicht eines Produkts.

3. Search

Hier sollen (anonyme) User nach Produkten suchen können. Diese werden dann in selbiger Form als bei Explore angezeigt.

4. Login

Hier soll es einem nicht-eingeloggten User möglich sein, sich über ein Formular einzuloggen oder zu registrieren. Bei der Registrierung halte ich es für sinnvoll, den User danach auch direkt einzuloggen

5. New Product

Eingeloggte User sollen über ein Formular neue Produkte erstellen können. Dabei müssen alle Felder auch ausgefüllt werden. Selbige Seite wird verwendet wenn ein Produkt verändert werden will. Sollte dies der Fall sein, so wird die Überschrift durch einen passenden Text ersetzt. Beim Editieren soll es möglich sein, das Bild auszulassen. In diesem Fall bleibt das aktuelle Bild bestehen.

6. Detailansicht der Produkte

Hier wird erneut die Durchschnittsbewertung angezeigt. Darunter folgt eine Tabelle mit allen Ratings fürs aktuelle Produkt. Weiters, wird für eingeloggte User ein Formular angezeigt. Über dieses Formular kann einerseits ein neues Rating abgegeben bzw. das aktuelle Rating bearbeitet werden. In der Angabe wurde nicht spezifiziert ob ein User mehrere Ratings zu einem Produkt abgeben kann → Für mich wäre die Funktionalität unlogisch. Es wurde auch nicht spezifiziert, ob ein User sein eigenes Produkt bewerten kann → Für mich legitim (vgl. Instagram, YouTube: Eigene Posts/Videos können selbst geliked werden)

Wenn der User bereits ein Rating abgegeben hat, so wird zusätzlich ein Delete Button angezeigt.

Um die technischen Anforderungen so gut wie möglich zu erfüllen, wurde das Produktbewertungsportal nach Vorlage des BookShops entwickelt. Die Architektur davon wird später noch detailliert erklärt.

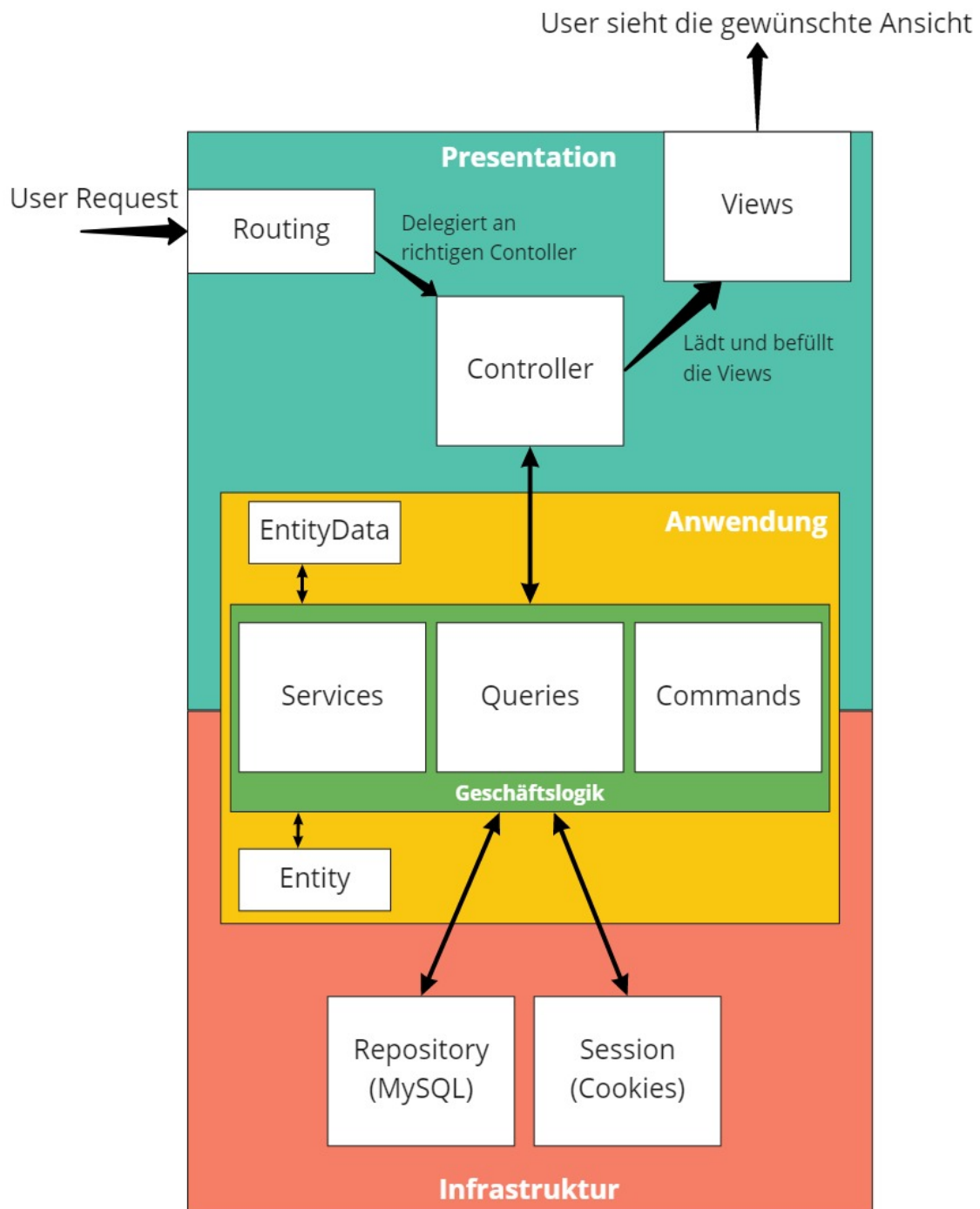
Grundsätzlich wurde darauf geachtet, dass alle Requests in den jeweiligen Commands/Querys korrekt verifiziert werden (auch wenn über die UI solche Requests nicht möglich wären). Zusätzlich wurden in der DB Constraints gesetzt, sodass spätestens auf der DB die Fehler erkannt werden und nicht eingetragen werden (z.B. invalide Ratings, zu lange Comments, nicht-eindeutiger Username).

Zum Sicherstellen, dass keine SQL-Injections möglich sind, wird mysqli verwendet. Dabei halte ich mich an die Struktur bzw. die Hilfsfunktionen aus der Übungseinheit.

Die Aufteilung auf Entity und EntityData hat mich minimal irritiert. Ich habe mich dafür entschieden bei den Entities schon benötigten JOINS einzubauen (bin mir nicht sicher ob die Trennung so sinnvoll ist?) und in EntityData dann noch zusätzliche Felder hinzuzufügen, die im UI benötigt werden.

Dadurch repräsentiert eine Entity aber nicht mehr genau die DB-Relation. Für mich hat das sinnvoll gewirkt, weil ich ansonsten im php Code die Joins manuell machen müsste.

Architektur



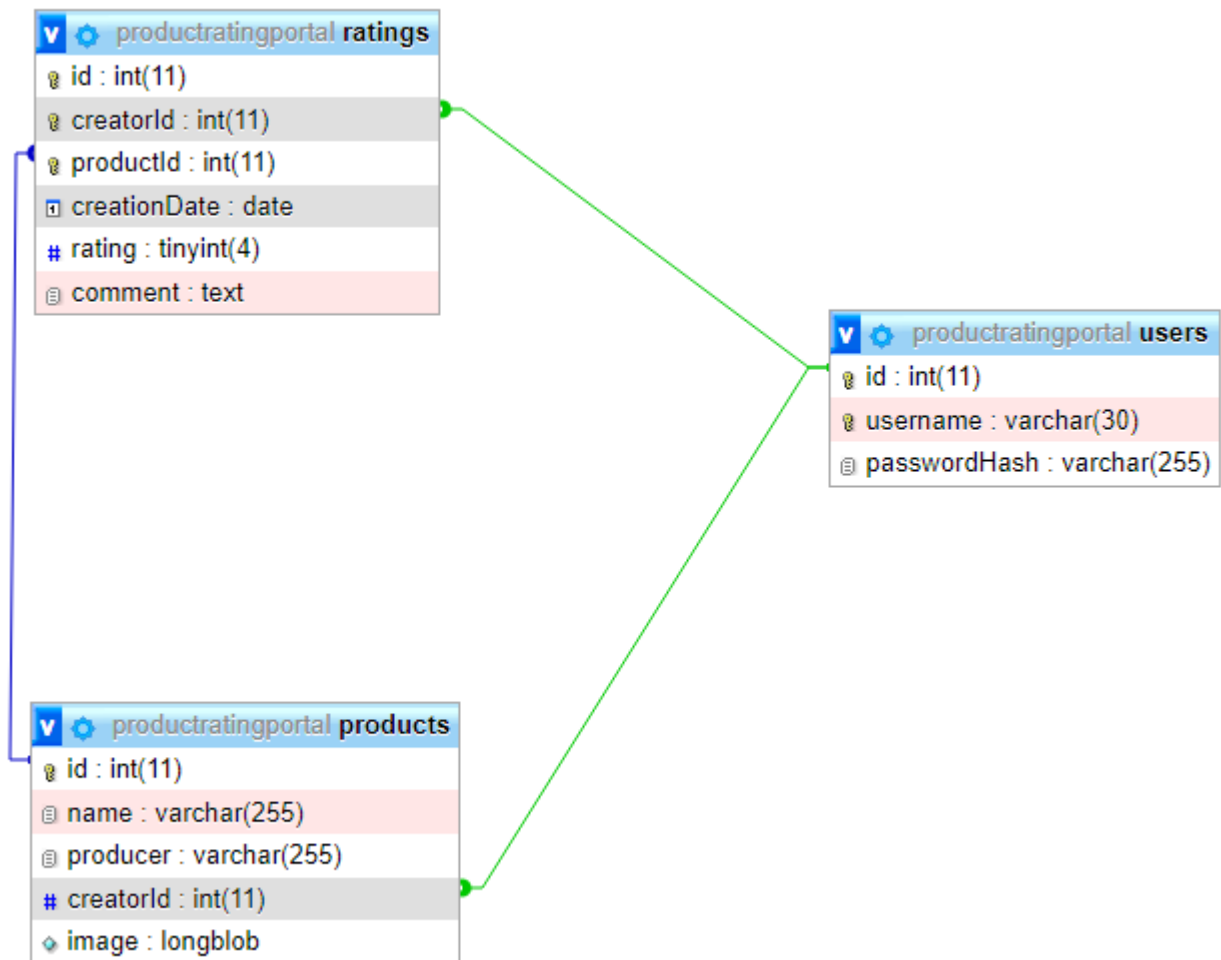
Requests werden über den Controller an die Businesslogik weitergeleitet. Dabei wird unterschieden in Services, Commands und Queries. Wobei bei Queries nur Abfragen getätigt werden und bei Command der Zustand der Applikation geändert wird (z.B. Daten in der DB manipuliert werden).

Die Geschäftslogik kommuniziert mit der Infrastruktur (Session-Cookies und MySQL Datenbank). Als „Schnittstelle“ werden dabei einzelne Repository-Interfaces definiert, die die notwendigen Operationen auf die Datenbanktabellen festlegen. Auf diese Weise, könnte der Persistence-Layer auch recht problemlos ausgetauscht werden. Die Geschäftslogik nimmt die Daten in Form von Entities entgegen, verarbeitet die Daten und liefert dem Presentation-Layer die notwendigen Informationen (Datenbank-Entities werden dabei auf die notwendige Form im Frontend gebracht = EntityData). Zusätzlich validiert die Geschäftslogik sämtliche Inputs sowie Datenbankresultate. Fehler werden den Controllern mit übergeben.

Der Controller verwendet die Resultate aus der Geschäftslogik um die Views mit Daten zu befüllen.

Die Struktur des Codes entspricht grundsätzlich dem obigen Bild und sollte intuitiv verständlich sein. Es werden, wie auch in der Übung gezeigt, Namespaces verwendet, die der Hierarchie in der Folderstruktur wiedergeben.

Datenmodell



Das Datenmodell wird relativ simpel gehalten.

Users:

- Speichert auto-inkrementierende Id, einzigartigen Username und den Passwort-Hash
- Id wäre theoretisch nicht notwendig, da über den Usernamen sowieso ein User eindeutig identifiziert werden kann. Ich halte das zusätzliche Feld trotzdem angebracht, weil dadurch Abfragen auf der DB schneller durchgeführt werden können (speziell bei größeren Datensätzen)
- Username soll maximal 30 Zeichen erlauben

Products:

- Auto-Inkrementierender Primary-Key
- Name mit maximal 255 Characters
- Hersteller wird direkt als String gespeichert. Eine eigene Tabelle wirkt für mich zum Erfüllen der Anforderungen als übertrieben
- CreatorId: Referenziert den User, der das Produkt erstellt hat
- Image: Speichert die base64-Zeichenfolge als longblob ab.

Ratings:

- Auto-Inkrementierender Primary Key
- CreatorId: Referenziert den User, der das Rating erstellt hat
- ProductId: Referenziert das dazugehörige Produkt
- CreatorId und ProductId müssen gemeinsam unique sein. Dadurch wird verhindert, dass ein User für das selbe Produkt zwei Bewertungen abgibt.
- CreationDate: Speichert das dazugehörige Datum
- Rating: Wird in Form eines tinyint gespeichert. Zusätzlich wird über ein Check Constraint geprüft ob der Wert ≥ 1 und ≤ 5 ist.
- Comment: Wird als Text gespeichert. Die genaueren Einschränkungen werden in den dazugehörigen Querys/Commands erledigt.

Erstellungsskript

Ich habe dafür zwei Skripts CREATE_productratingportal.sql und INSERT_productratingportal.sql bei meiner Abgabe angefügt. Hier zeige ich aus Übersichtsgründen nur das Create-Skript.

```
CREATE DATABASE IF NOT EXISTS `productratingportal` DEFAULT CHARACTER
SET utf8mb4 COLLATE utf8mb4_general_ci;
USE `productratingportal`;
```

```
CREATE TABLE `products` (
  `id` int(11) NOT NULL,
  `name` varchar(255) COLLATE latin1_general_ci NOT NULL,
  `producer` varchar(255) COLLATE latin1_general_ci NOT NULL,
  `creatorId` int(11) NOT NULL,
  `image` longblob NOT NULL
) ENGINE=InnoDB DEFAULT CHARSET=latin1 COLLATE=latin1_general_ci;
```

```
CREATE TABLE `ratings` (
  `id` int(11) NOT NULL,
  `creatorId` int(11) NOT NULL,
  `productId` int(11) NOT NULL,
  `creationDate` date NOT NULL,
  `rating` tinyint(4) NOT NULL,
  `comment` text COLLATE latin1_general_ci DEFAULT NULL
) ;
```

```
CREATE TABLE `users` (
  `id` int(11) NOT NULL,
  `username` varchar(30) NOT NULL,
  `passwordHash` varchar(255) NOT NULL
) ENGINE=InnoDB DEFAULT CHARSET=latin1;
```

```
ALTER TABLE `products`
  ADD PRIMARY KEY (`id`),
  ADD KEY `creatorId` (`creatorId`);
```

```
ALTER TABLE `ratings`  
  ADD PRIMARY KEY (`id`),  
  ADD UNIQUE KEY `unique_rating` (`creatorId`,`productId`),  
  ADD KEY `creatorId` (`creatorId`),  
  ADD KEY `productId` (`productId`);
```

```
ALTER TABLE `users`  
  ADD PRIMARY KEY (`id`),  
  ADD UNIQUE KEY `unique_username` (`username`);
```

```
ALTER TABLE `products`  
  MODIFY `id` int(11) NOT NULL AUTO_INCREMENT;
```

```
ALTER TABLE `ratings`  
  MODIFY `id` int(11) NOT NULL AUTO_INCREMENT;
```

```
ALTER TABLE `users`  
  MODIFY `id` int(11) NOT NULL AUTO_INCREMENT;
```

```
ALTER TABLE `products`  
  ADD CONSTRAINT `products_ibfk_1` FOREIGN KEY (`creatorId`)  
REFERENCES `users` (`id`);
```

```
ALTER TABLE `ratings`  
  ADD CONSTRAINT `ratings_ibfk_1` FOREIGN KEY (`creatorId`)  
REFERENCES `users` (`id`),  
  ADD CONSTRAINT `ratings_ibfk_2` FOREIGN KEY (`productId`)  
REFERENCES `products` (`id`);
```

Code

Index.php

```
<?php
// === register autoloader

use Infrastructure\Repository;

spl_autoload_register(function ($class) {
    $file = __DIR__ . '/src/' . str_replace('\\', '/', $class) . '.php';
    if (file_exists($file)) {
        require_once($file);
    }
});

$sp = new \ServiceProvider();
// Application
$sp->register(\Application\RatingCreationCommand::class);
$sp->register(\Application\RatingDeleteCommand::class);
$sp->register(\Application\RatingsQuery::class);
$sp->register(\Application\RegisterCommand::class);
$sp->register(\Application\SignInCommand::class);
$sp->register(\Application\SignedInUserQuery::class);
$sp->register(\Application\SignOutCommand::class);
$sp->register(\Application\ProductCreationCommand::class);
$sp->register(\Application\ProductsQuery::class);
$sp->register(\Application\ProductSearchQuery::class);
$sp->register(\Application\ProductQuery::class);

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

// Infrastructure

$sp->register(\Infrastructure\Repository::class, function() {
    return new Repository('localhost', 'root', '', 'productratingportal');
}, isSingleton: true);
$sp->register(\Application\Interfaces\UserRepository::class, \Infrastructure\
Repository::class);
$sp->register(\Application\Interfaces\ProductRepository::class, \Infrastructure\
Repository::class);
$sp->register(\Application\Interfaces\RatingRepository::class, \Infrastructure\
Repository::class);
```

```
$sp->register(\Infrastructure\Session::class, isSingleton: true);
$sp->register(\Application\Interfaces\Session::class, \Infrastructure\Session::class);

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

$sp->register(\Presentation\Controllers\Home::class);
$sp->register(\Presentation\Controllers\User::class);
$sp->register(\Presentation\Controllers\Rating::class);
$sp->register(\Presentation\Controllers\Product::class);

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

styles.css

```
.button-link {
    border: 0;
    color: blue;
    background-color: transparent;
    text-decoration: underline;
}
```

views/home.inc

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

<h1>Welcome</h1>
<p>Welcome to the product rating portal!</p>
<p>Use the navigation bar to navigate.</p>

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

views/login.inc

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

<container>
  <div class="row align-items-start p-5">
    <div class="col">
      <h1>Login</h1>
      <?php $beginForm('User', 'LogIn', method: 'post'); ?>
      <div class="mb-3">
        <label for="lUserName" class="form-label">User name</label>
        <input class="form-control" id="lUserName" name="un" value="<?php $htmlOut($data['userName']); ?>">
      </div>
      <div class="mb-3">
        <label for="lPassword" class="form-label">Password</label>
        <input type="password" class="form-control" id="lPassword" name="pwd">
      </div>
      <button class="btn btn-primary">Log in</button>
      <?php $endForm(); ?>
    </div>
    <div class="col">
      <h1>No account yet?</h1>
      <h2>Register here:</h2>
      <?php $beginForm('User', 'Register', method: 'post'); ?>
      <div class="mb-3">
        <label for="rUserName" class="form-label">User name</label>
        <input class="form-control" id="rUserName" name="un">
      </div>
      <div class="mb-3">
        <label for="rPassword" class="form-label">Password</label>
        <input type="password" class="form-control" id="rPassword" name="pwd">
      </div>
      <button class="btn btn-primary">Register</button>
      <?php $endForm(); ?>
    </div>
  </div>
</container>

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

views/productCreate.inc

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

<div class="container border p-4">
    <?php if($data['productName'] == '') { ?>
        <h1>Create new Products</h1>
    <?php } else { ?>
        <h1>Edit Products</h1>
    <?php } ?>
    <?php $beginForm('Product', 'Send', ['p' => $data['productId']], method: 'post', enc
type: 'multipart/form-data'); ?>
    <div class="row">
        <div class="col-8">
            <label for="name">Product Name:</label>
            <input class="form-control" id="name" name="pn" value="<?php $htmlOut($data
['productName']); ?>">
        </div>
        <div class="col">
            <label for="producer">Producer:</label>
            <input class="form-control" id="producer" name="pc" value="<?php $htmlOut($data
['producerName']); ?>">
        </div>
    </div>
    <div class="row mt-2">
        <label for="img">Image:</label>
        <input id="img" type="file" name="img"> <br>
    </div>
    <div class="row mt-2">
        <div class="col-1">
            <button class="btn btn-warning">Send</button>
        </div>
    </div>
</div>

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

views/productList.inc

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

<h1>Explore Products</h1>

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

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

views/productSearch.inc

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

<h1>Search</h1>

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

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

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

views/ratingList.inc

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

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

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

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

views/partial/errors.inc

```
<div class="alert alert-danger">
  <p>
    <strong>Please correct the following and try again:</strong>
  </p>
  <ul>
    <?php foreach ($data as $errMsg): ?>
      <li><?php $htmlOut($errMsg); ?></li>
    <?php endforeach; ?>
  </ul>
</div>
```

views/partial/footer.inc

```
</div>
<footer class="bg-dark mt-3 fixed-bottom">
  <div class="container">
    <p class="text-muted p-0"><?php $htmlOut(strftime('%c')); ?></p>
  </div>
</footer>
<script src="js/bootstrap.bundle.min.js"></script>
</body>

</html>
```


views/partial/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/styles.css" rel="stylesheet">
  <title>Product Rating Portal</title>
</head>

<body>
  <nav class="navbar navbar-expand-lg navbar-dark bg-dark">
    <div class="container">
      <?php $link('Product Rating Portal', '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('Explore', 'Product', 'Index', cssClass: 'nav-link'); ?>
          <?php $link('Search', 'Product', 'Search', cssClass: 'nav-link'); ?>
          <?php $render('partial/user', $data['user']); ?>
        </nav>
      </div>
    </div>
  </nav>
  <div class="container mt-3">
    <?php if (isset($data['errors'])) {
      $render('partial/errors', $data['errors']);
    } ?>
  </div>
</body>
</html>
```

views/partial/products.inc

```

<?php if ($data['products'] !== null) { ?>
    <?php if (sizeof($data['products']) > 0) { ?>
        <table class="table table-hover mb-5">
            <thead>
                <tr>
                    <th>Image</th>
                    <th>Name</th>
                    <th>Producer</th>
                    <th>Creator</th>
                    <th>Rating</th>
                    <th></th>
                </tr>
            </thead>
            <tbody>
                <?php foreach ($data['products'] as $product) { ?>
                    <tr>
                        <td>getName(), 0, 10)) ?>..." /></td>
                        <td>
                            <?php $beginForm('Rating', 'Index', ['p' => $product->getId()], 'get',
                                'form-inline'); ?>
                            <button class="button-link"><?php $htmlOut($product->getName()); ?></button>
                            <?php $endForm(); ?>
                        </td>
                        <td><?php $htmlOut($product->getProducer()); ?></td>
                        <td><?php $htmlOut($product->getCreatorName()); ?></td>
                        <td><?php $htmlOut($product->getAvgRating()); ?> <i>( <?php $htmlOut($product->getRatingCount()); ?> Ratings)</i></td>
                        <td>
                            <?php if($data['user'] != null && $data['user']->getUserName() == $product->getCreatorName()) { ?>
                                <?php $beginForm('Product', 'Edit',
                                    ['p' => $product->getId(), 'pn' => $product->getName(), 'pc' => $product->getProducer()],
                                    method: 'get'); ?>
                                <button class="btn btn-warning">Edit</button>
                                <?php $endForm(); ?>
                            <?php } ?>
                        </td>
                    </tr>
                </tbody>
            </table>
        </?>
    </?>
}

```

```

        </td>
    </tr>
    <?php } ?>
</tbody>
</table>
<?php } else { ?>
    <p>No products found.</p>
<?php } ?>
<?php } ?>

```

views/partial/ratingAdd.inc

```

<?php if ($data['product'] !== null && $data['ratings'] !== null && $data['user'] !==
null) { ?>

<div class="container border p-4 mb-5">
    <h2>Your rating: </h2>
    <?php $beginForm('Rating', 'Send', ['p' => $data['product']->getId()], method: 'post
'); ?>
    <div class="row">
        <div class="col-1">
            <label for="rating" class="form-label">Rating</label>
            <select class="form-control" name="rt" id="rating" value="3">
                <option <?php if($data['rating'] && $data['rating']->getRating() == 1) echo "sel
ected" ?>>1</option>
                <option <?php if($data['rating'] && $data['rating']->getRating() == 2) echo "sel
ected" ?>>2</option>
                <option <?php if($data['rating'] && $data['rating']->getRating() == 3) echo "sel
ected" ?>>3</option>
                <option <?php if($data['rating'] && $data['rating']->getRating() == 4) echo "sel
ected" ?>>4</option>
                <option <?php if($data['rating'] && $data['rating']->getRating() == 5) echo "sel
ected" ?>>5</option>
            </select>
        </div>
        <div class="col">
            <label for="comment" class="form-label">Comment</label>
            <textarea class="form-control" id="comment" name="ct" value="" rows="5"><?php i
f($data['rating']) $htmlOut($data['rating']->getComment())?></textarea>
        </div>
    </div>
    <div class="row">
        <div class="col-2">

```

```

        <label for="date">Last update:</label>
        <input type="text" type="date" id="date" value="<?php if($data['rating']) $htmlOut($data['rating']->getDate()) ?>" disabled>
    </div>
</div>
<div class="row mt-2">
    <div class="col-1">
        <button class="btn btn-warning">Send</button>
    </div>
    <div class="col-1">
        <?php $endForm(); ?>
        <?php if($data['rating'] !== null) { ?>
            <?php $beginForm('Rating', 'Delete', ['p' => $data['product']->getId()], method: 'post'); ?>
            <button class="btn btn-danger">Delete</button>
            <?php $endForm(); ?>
        <?php } ?>
    </div>
</div>
</div>

<?php } ?>

```

views/partial/ratings.inc

```

<?php if ($data['ratings'] !== null) { ?>
    <h1><?php $htmlOut($data['product']->getName()); ?></h1>
    <h2>
        Customer ratings: (&Oslash;<?php $htmlOut($data['product']->getAvgRating());?>)
    </h2>

    <?php if (sizeof($data['ratings']) > 0) { ?>
        <table class="table table-hover">
            <thead>
                <tr>
                    <th>Creator</th>
                    <th>Rating</th>
                    <th>Comment</th>
                    <th>Date</th>
                </tr>
            </thead>
            <tbody>

```

```
<?php foreach ($data['ratings'] as $rating) { ?>
    <tr>
        <td class="col-1"><?php $htmlOut($rating->getCreatorName()); ?></td>
        <td class="col-1"><?php $htmlOut($rating->getRating()); ?></td>
        <td><?php $htmlOut($rating->getComment()); ?></td>
        <td class="col-1"><?php $htmlOut($rating->getDate()); ?></td>
    </tr>
<?php } ?>
</tbody>
</table>
<?php } else { ?>
    <p>No ratings found.</p>
<?php } ?>
<?php } ?>
```

views/partial/user.inc

```
<?php if (!isset($data)): ?>
    </nav>
    <nav class="navbar-nav">
        <?php $link('Login', 'User', 'LogIn', cssClass: 'btn btn-warning'); ?>
    </nav>
<?php else: ?>
    <?php $link('New Product', 'Product', 'Edit', cssClass: 'nav-link'); ?>
    </nav>

    <?php $beginForm('User', 'LogOut', method: 'post', cssClass: 'form-inline'); ?>
    <span class="navbar-text me-2">Welcome, <strong class="text-warning"><?php $htmlOut(
$data->getUserName()); ?></strong></span>
    <button class="btn btn-outline-warning">Log out</button>
    <?php $endForm(); ?>
<?php endif; ?>
```

Presentation

MVC Folder wurde komplett aus der Übung übernommen und wird deswegen auch nicht extra in meine Abgabe kopiert. Die einzige Änderung ist folgende:

```
$beginForm = function (string $controller, string $action, array $params = [], string $method = 'get', ?string $cssClass = null, ?string $enctype = 'application/x-www-form-urlencoded') use ($mvc) {
    $cc = $cssClass !== null ? " class=\"$cssClass\" : '';
    echo "<form method=\"$method\" action=\"$?\" enctype=\"$enctype\"$cc>";
    foreach ($params as $name => $value) {
        echo ("<input type=\"hidden\" name=\"$name\" value=\"$value\">");
    }
    echo "<input type=\"hidden\" name=\"{$mvc->getControllerParameterName()}\" value=\"$controller\">";
    echo "<input type=\"hidden\" name=\"{$mvc->getActionParameterName()}\" value=\"$action\">";
};
```

Mit \$beginForm kann nun auch der enctype eingestellt werden.

Controllers/Home.php

```
<?php
```

```
namespace Presentation\Controllers;

use Presentation\MVC\ViewResult;
use Presentation\MVC\Controller;

class Home extends Controller {
    public function __construct(
        private \Application\SignedInUserQuery $signedInUserQuery
    )
    {

    }

    public function GET_index(): ViewResult {
        return $this->view('home', [
            'user' => $this->signedInUserQuery->execute()
        ]);
    }
}
```

```
}  
}
```

Controllers/Product.php

```
<?php
```

```
namespace Presentation\Controllers;
```

```
use Presentation\MVC\ViewResult;
```

```
use Presentation\MVC\Controller;
```

```
use Presentation\MVC\ActionResult;
```

```
class Product extends Controller {
```

```
    public function __construct(  
        private \Application\SignedInUserQuery $signedInUserQuery,  
        private \Application\ProductsQuery $productsQuery,  
        private \Application\ProductQuery $productQuery,  
        private \Application\ProductSearchQuery $productSearchQuery,  
        private \Application\ProductCreationCommand $productCreationCommand  
    )  
{
```

```
    }  
  
    public function GET_index(): ViewResult {  
        return $this->view('productList', [  
            'user' => $this->signedInUserQuery->execute(),  
            'products' => $this->productsQuery->execute(),  
        ]);  
    }  
  
    public function GET_search(): ViewResult {  
        return $this->view('productSearch', [  
            'user' => $this->signedInUserQuery->execute(),  
            'products' => $this->tryGetParam('f', $value) ? $this->productSearchQuery->  
execute($value) : null,  
            'filter' => $this->tryGetParam('f', $value) ? $value : null  
        ]);  
    }  
  
    public function GET_edit(): ViewResult {  
        $user = $this->signedInUserQuery->execute();  
        $productId = $this->tryGetParam('p', $value) ? $value : null;  
  
        //Only user who created the product is allowed to see it  
        if($productId != null || $user == null) {
```

```
$errors[] = 'Access denied';
if($user == null) {
    return $this->view('productList', [
        'user' => $user,
        'products' => $this->productsQuery->execute(),
        'errors' => $errors
    ]);
}
$product = $this->productQuery->execute($productId);
if($product->getCreatorName() != $user->getUserName()) {
    return $this->view('productList', [
        'user' => $user,
        'products' => $this->productsQuery->execute(),
        'errors' => $errors
    ]);
}
}
```

```
$productName = $this->tryGetParam('pn', $value) ? $value : '';
$producerName = $this->tryGetParam('pc', $value) ? $value : '';
return $this->view('productCreate', [
    'user' => $user,
    'producerName' => $producerName,
    'productName' => $productName,
    'productId' => $productId
]);
}
```

```
public function POST_send(): ActionResult {
    $productName = $this->getParam('pn');
    $producerName = $this->getParam('pc');
    $productId = $this->tryGetParam('p', $value) ? (int)$value : null;
```

```
    $result = $this->productCreationCommand->execute($productId, $productName, $producerName, $_FILES['img']['tmp_name']);
```

```
    //error occurred
    if($result != 0) {
        $errors = [];
        if($result & \Application\ProductCreationCommand::Error_NotAuthenticated) {
            $errors[] = "You need to be logged in to create ratings";
        }
    }
}
```



```
if($result & \Application\ProductCreationCommand::Error_InvalidProductName) {
    $errors[] = "Enter product name";
}
if($result & \Application\ProductCreationCommand::Error_InvalidProducer) {
    $errors[] = "Enter producer";
}
if($result & \Application\ProductCreationCommand::Error_InvalidImage) {
    $errors[] = "Select valid image";
}
if($result & \Application\ProductCreationCommand::Error_DbErrorOccured) {
    $errors[] = "Error_DbErrorOccured";
}
if(sizeof($errors) == 0) {
    $errors[] = 'Something went wrong.';
}

return $this->view('productCreate', [
    'user' => $this->signedInUserQuery->execute(),
    'producerName' => $productName,
    'productName' => $producerName,
    'productId' => $productId,
    'errors' => $errors
]);
} else {
    return $this->redirect('Product', 'Index');
}
}
```

Controllers/Rating.php

<?php

```
namespace Presentation\Controllers;

use Presentation\MVC\ViewResult;
use Presentation\MVC\Controller;
use Presentation\MVC\ActionResult;

class Rating extends Controller {
    public function __construct(
        private \Application\SignedInUserQuery $signedInUserQuery,
        private \Application\RatingsQuery $ratingsQuery,
        private \Application\ProductQuery $productQuery,
        private \Application\RatingCreationCommand $ratingCreationCommand,
        private \Application\RatingDeleteCommand $ratingDeleteCommand
    )
    {

    }

    public function GET_index(): ViewResult {
        $selectedProductId = $this->getParam('p');
        $selectedProduct = $this->productQuery->execute($selectedProductId);
        $user = $this->signedInUserQuery->execute();
        $ratings = $this->ratingsQuery->execute($selectedProductId);
        $rating = null;

        if($user !== null) {
            foreach($ratings as $r) {
                if($r->getCreatorName() == $user->getUserName()) {
                    $rating = $r;
                }
            }
        }

        return $this->view('ratingList', [
            'user' => $user,
            'product' => $selectedProduct,
            'ratings' => $ratings,
            'rating' => $rating
        ]);
    }
}
```

```
public function POST_send(): ActionResult {
    $rating = $this->getParam('rt');
    $comment = $this->getParam('ct');
    $selectedProductId = $this->getParam('p');
    $selectedProduct = $this->productQuery->execute($selectedProductId);

    $result = $this->ratingCreationCommand->execute($selectedProductId, $rating, $comment);

    //error occurred
    if($result != 0) {
        $errors = [];
        if($result & \Application\RatingCreationCommand::Error_NotAuthenticated) {
            $errors[] = "You need to be logged in to create ratings";
        }
        if($result & \Application\RatingCreationCommand::Error_DbErrorOccured) {
            $errors[] = "Error_DbErrorOccured";
        }
        if($result & \Application\RatingCreationCommand::Error_InvalidComment) {
            $errors[] = "Comment is too long";
        }
        if($result & \Application\RatingCreationCommand::Error_InvalidRating) {
            $errors[] = "Only Ratings from 1 to 5 are valid";
        }
        if(sizeof($errors) == 0) {
            $errors[] = 'Something went wrong.';
        }
    }

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

    $ratings = $this->ratingsQuery->execute($selectedProductId);
    $rating = null;

    if($user !== null) {
        foreach($ratings as $r) {
            if($r->getCreatorName() == $user->getUserName()) {
                $rating = $r;
            }
        }
    }

    return $this->view('ratingList', [
        'user' => $user,
    ]
}
```

```
        'product' => $selectedProduct,
        'ratings' => $this->ratingsQuery->execute($selectedProductId),
        'rating' => $rating,
        'errors' => $errors
    ]);
} else {
    return $this->redirect('Rating', 'Index', ['p' => $selectedProductId]);
}
}

public function POST_delete(): ActionResult {

    $selectedProductId = $this->getParam('p');
    $selectedProduct = $this->productQuery->execute($selectedProductId);

    $result = $this->ratingDeleteCommand->execute($selectedProductId);

    //error occurred
    if($result != 0) {
        $errors = [];
        if($result & \Application\RatingDeleteCommand::Error_NotAuthenticated) {
            $errors[] = "You need to be logged in to create ratings";
        }
        if($result & \Application\RatingDeleteCommand::Error_DbErrorOccured) {
            $errors[] = "Error_DbErrorOccured";
        }
        if(sizeof($errors) == 0) {
            $errors[] = 'Something went wrong.';
        }

        return $this->view('ratingList', [
            'user' => $this->signedInUserQuery->execute(),
            'product' => $selectedProduct,
            'ratings' => $this->ratingsQuery->execute($selectedProductId),
            'errors' => $errors
        ]);
    } else {
        return $this->redirect('Rating', 'Index', ['p' => $selectedProductId]);
    }
}
}
```

Controllers/User.php

<?php

namespace Presentation\Controllers;

class User extends \Presentation\MVC\Controller

{

public function __construct(

private \Application\RegisterCommand \$registerCommand,

private \Application\SignOutCommand \$signOutCommand,

private \Application\SignInCommand \$signInCommand,

private \Application\SignedInUserQuery \$signedInUserQuery

) {

}

public function GET_LogIn(): \Presentation\MVC\ActionResult

{

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

// only show login view when there is no authenticated user

if(\$user != null) {

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

}

return \$this->view('login', [

'user' => \$this->signedInUserQuery->execute(),

'userName' => ''

]);

}

public function POST_LogIn(): \Presentation\MVC\ActionResult

{

\$ok = \$this->signInCommand->execute(\$this->getParam('un'), \$this->getParam('pwd'));

if(!\$ok) {

return \$this->view('login', [

'user' => \$this->signedInUserQuery->execute(),

'userName' => \$this->getParam('un'),

'errors' => ['The combination of username and password you have entered is in correct']

]);

}

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

}

public function POST_LogOut(): \Presentation\MVC\ActionResult

{

```
$this->signOutCommand->execute();
return $this->redirect('Home', 'Index');
}
public function POST_Register(): \Presentation\MVC\ActionResult
{
    $result = $this->registerCommand->execute($this->getParam('un'), $this->getParam('pwd'));

    if($result != 0) {
        $errors = [];
        if($result & \Application\RegisterCommand::Error_UserNameTooShort) {
            $errors[] = "Username needs to have between 3 and 30 characters";
        }
        if($result & \Application\RegisterCommand::Error_InvalidPassword) {
            $errors[] = "Password needs to have atleast 1 character";
        }
        if($result & \Application\RegisterCommand::Error_UserNameUsed) {
            $errors[] = "Entered username is already used";
        }
        if(sizeof($errors) == 0) {
            $errors[] = 'Something went wrong.';
        }
        return $this->view('login', [
            'user' => $this->signedInUserQuery->execute(),
            'userName' => $this->getParam('un'),
            'errors' => $errors
        ]);
    }
    return $this->redirect('Home', 'Index');
}
}
```

Infrastructure

Repository.php

```
<?php
```

```
namespace Infrastructure;
```

```
use Application\Entities\User;
```

```
class Repository implements
```

```
\Application\Interfaces\UserRepository,  
\Application\Interfaces\RatingRepository,  
\Application\Interfaces\ProductRepository
```

```
{
```

```
    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 getUser(int $id): ?User
{
    $user = null;
    $con = $this->getConnection();
    $stat = $this->executeStatement(
        $con,
        'SELECT id, username FROM users WHERE id = ?',
        function ($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): ?
User
```



```
{
    $user = null;
    $con = $this->getConnection();
    $stat = $this->executeStatement(
        $con,
        'SELECT id, passwordHash FROM users WHERE username = ?',
        function ($s) use ($username) {
            $s->bind_param('s', $username);
        }
    );
    $stat->bind_result($id, $passwordHash);
    if ($stat->fetch() && password_verify($password, $passwordHash)) {
        $user = new \Application\Entities\User($id, $username);
    }
    $stat->close();
    $con->close();
    return $user;
}

public function createUser(string $username, string $password): ?User
{
    $con = $this->getConnection();
    $stat = $con->prepare(
        'INSERT INTO users (username, passwordHash)
        VALUES (?, ?);'
    );
    $passwordHash = password_hash($password, PASSWORD_DEFAULT);
    $stat->bind_param("ss", $username, $passwordHash);
    $stat->execute();

    $newUserCreated = $stat->insert_id != 0;
    $stat->close();
    $con->close();

    return $newUserCreated === false ? null : $this-
>getUserForUserNameAndPassword($username, $password);
}

public function getProducts(): array
{
    $products = [];
    $con = $this->getConnection();
    $stat = $this->executeStatement(
        $con,
```

```
'SELECT products.id, name, producer, creatorId, username, image
FROM products JOIN users ON (users.id = products.creatorId)',
function () {
}
);
$stat->bind_result($id, $name, $producer, $creatorId, $username, $image);
while ($stat->fetch()) {
    $products[] = new \Application\Entities\Product($id, $name, $producer, $creatorI
d, $username, $image);
}
$stat->close();
$con->close();
return $products;
}

public function getProductById(int $productId): ?\Application\Entities\Product
{
    $product = null;
    $con = $this->getConnection();
    $stat = $this->executeStatement(
        $con,
        'SELECT products.id AS pid, name, producer, creatorId, username, image
FROM products JOIN users ON (users.id = products.creatorId)
WHERE products.id = ?',
        function ($s) use ($productId) {
            $s->bind_param('i', $productId);
        }
    );
    $stat->bind_result($id, $name, $producer, $creatorId, $username, $image);
    if ($stat->fetch()) {
        $product = new \Application\Entities\Product($id, $name, $producer, $creatorId,
$username, $image);
    }
    $stat->close();
    $con->close();
    return $product;
}

public function getProductsForFilter(string $filter): array
{
    $filter = "%$filter%";
    $products = [];
    $con = $this->getConnection();
```

```

    $stat = $this->executeStatement(
        $con,
        'SELECT products.id, name, producer, creatorId, username, image
        FROM products JOIN users ON (users.id = products.creatorId)
        WHERE name LIKE ? OR producer LIKE ?',
        function ($s) use ($filter) {
            $s->bind_param('ss', $filter, $filter);
        }
    );
    $stat->bind_result($id, $name, $producer, $creatorId, $username, $image);
    while ($stat->fetch()) {
        $products[] = new \Application\Entities\Product($id, $name, $producer, $creatorId, $username, $image);
    }
    $stat->close();
    $con->close();
    return $products;
}

public function getRatingsForProduct(int $productId): array
{
    $ratings = [];
    $con = $this->getConnection();
    $stat = $this->executeStatement(
        $con,
        'SELECT ratings.id, ratings.creatorId, username, productId, name, creationDate,
rating, comment
        FROM ratings JOIN users ON (users.id = ratings.creatorId)
        JOIN products ON (products.id = ratings.productId)
        WHERE productId = ?
        ORDER BY creationDate DESC',
        function ($s) use ($productId) {
            $s->bind_param('i', $productId);
        }
    );
    $stat->bind_result($id, $creatorId, $username, $productId, $name, $creationDate, $
rating, $comment);
    while ($stat->fetch()) {
        $ratings[] = new \Application\Entities\Rating($id, $creatorId, $username, $produ
ctId, $name, $creationDate, $rating, $comment);
    }
    $stat->close();
    $con->close();
}

```

```
        return $ratings;
    }
    public function getAvgRatingForProduct(int $productId): float
    {
        $ratingsAvg = 0;
        $con = $this->getConnection();
        $stat = $this->executeStatement(
            $con,
            'SELECT AVG(rating) AS avg
            FROM ratings
            WHERE productId = ?',
            function ($s) use ($productId) {
                $s->bind_param('i', $productId);
            }
        );
        $stat->bind_result($ratingsAvg);
        $stat->fetch();

        $stat->close();
        $con->close();
        return $ratingsAvg ?? 0;
    }
    public function getCountOfRatingsForProduct(int $productId): int
    {
        $ratingsCount = 0;
        $con = $this->getConnection();
        $stat = $this->executeStatement(
            $con,
            'SELECT COUNT(*) AS cnt
            FROM ratings
            WHERE productId = ?',
            function ($s) use ($productId) {
                $s->bind_param('i', $productId);
            }
        );
        $stat->bind_result($ratingsCount);
        $stat->fetch();

        $stat->close();
        $con->close();
        return $ratingsCount;
    }
}
```

```
public function createOrUpdateRatingForProduct(int $creatorId, int $productId, int $
rating, string $comment): ?int
{
    $con = $this->getConnection();
    $con->autocommit(false);

    //check if rating exists
    $stat = $this->executeStatement(
        $con,
        'SELECT id
        FROM ratings
        WHERE creatorId = ? AND productId = ? ',
        function ($s) use ($creatorId, $productId) {
            $s->bind_param('ii', $creatorId, $productId);
        }
    );
    $stat->bind_result($id);
    while ($stat->fetch());

    //No rating exists -> can create a new one
    if ($id == null) {
        $stat = $this->executeStatement(
            $con,
            'INSERT INTO ratings (creatorId, productId, creationDate, rating, comment)
            VALUES (?, ?, CURDATE(), ?, ?)',
            function ($s) use ($creatorId, $productId, $rating, $comment) {
                $s->bind_param('iiis', $creatorId, $productId, $rating, $comment);
            }
        );
        $id = $stat->insert_id;
    } else { //rating exists and needs to be updated
        $stat = $this->executeStatement(
            $con,
            'UPDATE ratings
            SET creationDate = CURDATE(), rating = ?, comment = ?
            WHERE id = ?',
            function ($s) use ($rating, $comment, $id) {
                $s->bind_param('isi', $rating, $comment, $id);
            }
        );
    }
    $con->commit();
}
```

```
$con->close();
return $id;
}
public function deleteRatingForProduct(int $creatorId, int $productId): ?int
{
    $con = $this->getConnection();

    //check if rating exists
    $stat = $this->executeStatement(
        $con,
        'DELETE
        FROM ratings
        WHERE creatorId = ? AND productId = ?',
        function ($s) use ($creatorId, $productId) {
            $s->bind_param('ii', $creatorId, $productId);
        }
    );
    $affectedRows = $con->affected_rows;
    $con->close();
    return $affectedRows;
}
public function createProduct(int $userId, string $productName, string $producerName
, string $imgBlob): ?int
{
    $con = $this->getConnection();
    $stat = $this->executeStatement(
        $con,
        'INSERT INTO products (creatorId, name, producer, image) VALUES (?, ?, ?, ?)',
        function ($s) use ($userId, $productName, $producerName, $imgBlob) {
            $s->bind_param('iiss', $userId, $productName, $producerName, $imgBlob);
        }
    );
    $productId = $stat->insert_id;
    $stat->close();
    $con->close();
    return $productId;
}
public function updateProduct(int $userId, int $productId, string $productName, stri
ng $producerName, string $imgBlob): ?int
{
    $con = $this->getConnection();
    $stat = $this->executeStatement(
        $con,
```

```

        'UPDATE products
        SET name = ?,
        producer = ?,
        image = ?
        WHERE id = ? AND creatorId = ?',
        function ($s) use ($productName, $producerName, $imgBlob, $productId, $userId) {
            $s->bind_param('sssii', $productName, $producerName, $imgBlob, $productId, $u
serId);
        }
    );
    $affectedRows = $con->affected_rows;
    $stat->close();
    $con->close();
    return $affectedRows;
}

public function updateProductWithoutImage(int $userId, int $productId, string $produ
ctName, string $producerName): ?int {
    $con = $this->getConnection();
    $stat = $this->executeStatement(
        $con,
        'UPDATE products
        SET name = ?,
        producer = ?
        WHERE id = ? AND creatorId = ?',
        function ($s) use ($productName, $producerName, $productId, $userId) {
            $s->bind_param('ssii', $productName, $producerName, $productId, $userId);
        }
    );
    $affectedRows = $con->affected_rows;
    $stat->close();
    $con->close();
    return $affectedRows;
}
}

```

Session.php

<?php

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]);
    }
}
```


Application

Entities/Product.php

```
<?php
```

```
namespace Application\Entities;
```

```
class Product
```

```
{  
    public function __construct(  
        private int $id,  
        private string $name,  
        private string $producer,  
        private int $creatorId,  
        private string $creatorName,  
        private string $image  
    ) {  
    }  
    public function getId(): int  
    {  
        return $this->id;  
    }  
    public function getName(): string  
    {  
        return $this->name;  
    }  
    public function getProducer(): string  
    {  
        return $this->producer;  
    }  
    public function getCreatorId(): int  
    {  
        return $this->creatorId;  
    }  
    public function getCreatorName(): string  
    {  
        return $this->creatorName;  
    }  
    public function getImage(): string  
    {  
        return $this->image;  
    }  
}
```

```
}
```

Entities/Rating.php

```
<?php
```

```
namespace Application\Entities;
```

```
class Rating
```

```
{
```

```
    public function __construct(
```

```
        private int $id,
```

```
        private int $creatorId,
```

```
        private string $creatorName,
```

```
        private int $productId,
```

```
        private string $productName,
```

```
        private string $date,
```

```
        private int $rating,
```

```
        private string $comment
```

```
    ) {
```

```
    }
```

```
    public function getId(): int
```

```
    {
```

```
        return $this->id;
```

```
    }
```

```
    public function getCreatorId(): int
```

```
    {
```

```
        return $this->creatorId;
```

```
    }
```

```
    public function getProductId(): int
```

```
    {
```

```
        return $this->productId;
```

```
    }
```

```
    public function getDate(): string
```

```
    {
```

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

```
    }
```

```
    public function getRating(): int
```

```
    {
```

```
        return $this->rating;
```

```
    }
```

```
    public function getComment(): string
```

```
    {
```

```
        return $this->comment;
    }
    public function getCreatorName(): string
    {
        return $this->creatorName;
    }
    public function getProductName(): string
    {
        return $this->productName;
    }
}
```

Entities/User.php

```
<?php
```

```
namespace Application\Entities;
```

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

Interfaces/ProductRepository.php

<?php

namespace Application\Interfaces;

interface ProductRepository

{

public function getProducts(): array; //array of \Application\Entities\Product

public function getProductById(int \$productId): ?\Application\Entities\Product;

public function getProductsForFilter(string \$filter): array; //array of \

\Application\Entities\Product

public function createProduct(int \$userId, string \$productName, string \$producerName, string \$imgBlob): ?int;

public function updateProduct(int \$userId, int \$productId, string \$productName, string \$producerName, string \$imgBlob): ?int;

public function updateProductWithoutImage(int \$userId, int \$productId, string \$productName, string \$producerName): ?int;

}

Interfaces/RatingRepository.php

<?php

namespace Application\Interfaces;

interface RatingRepository

{

public function getRatingsForProduct(int \$productId): array; //array of \Application\Entities\Rating

public function getAvgRatingForProduct(int \$productId): float;

public function getCountOfRatingsForProduct(int \$productId): int;

public function createOrUpdateRatingForProduct(int \$creatorId, int \$productId, int \$rating, string \$comment): ?int;

public function deleteRatingForProduct(int \$creatorId, int \$productId): ?int;

}

Interfaces/UserRepository.php

<?php

namespace Application\Interfaces;

interface UserRepository

{

public function getUser(int \$id): ?\Application\Entities\User;

public function getUserForUserNameAndPassword(string \$username, string \$password): ?

\Application\Entities\User;

public function createUser(string \$username, string \$password): ?\Application\

Entities\User;

}

Interfaces/Session.php

<?php

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;

}

Services/AuthenticationService.php

<?php

namespace Application\Services;

class AuthenticationService

{

const SESSION_USER_ID = 'userId';

public function __construct(

private \Application\Interfaces\Session \$session

) {

}

public function signOut(): void

{

\$this->session->delete(self::SESSION_USER_ID);

}

public function signIn(int \$userId): void

{

\$this->session->put(self::SESSION_USER_ID, \$userId);

}

public function getUserId(): ?int

{

return \$this->session->get(self::SESSION_USER_ID);

}

}

ProductData.php

<?php

```
namespace Application;

class ProductData
{
    public function __construct(
        private int $id,
        private string $name,
        private string $producer,
        private string $creatorName,
        private string $image,
        private int $ratingCount,
        private float $avgRating
    ) {
    }
    public function getId(): int
    {
        return $this->id;
    }
    public function getName(): string
    {
        return $this->name;
    }
    public function getProducer(): string
    {
        return $this->producer;
    }
    public function getCreatorName(): string
    {
        return $this->creatorName;
    }
    public function getRatingCount(): int
    {
        return $this->ratingCount;
    }
    public function getAvgRating(): float
    {
        return $this->avgRating;
    }
    public function getImage(): string
```

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

RatingData.php

```
<?php
```

```
namespace Application;
```

```
class RatingData
```

```
{  
    public function __construct(  
        private int $id,  
        private int $creatorId,  
        private string $creatorName,  
        private int $productId,  
        private string $productName,  
        private string $date,  
        private int $rating,  
        private string $comment  
    ) {  
    }  
    public function getId(): int  
    {  
        return $this->id;  
    }  
    public function getCreatorId(): int  
    {  
        return $this->creatorId;  
    }  
    public function getProductId(): int  
    {  
        return $this->productId;  
    }  
    public function getDate(): string  
    {  
        return $this->date;  
    }  
    public function getRating(): int  
    {  
        return $this->rating;  
    }  
}
```



```
}  
public function getComment(): string  
{  
    return $this->comment;  
}  
public function getCreatorName(): string  
{  
    return $this->creatorName;  
}  
public function getProductName(): string  
{  
    return $this->productName;  
}  
}
```

UserData.php

```
<?php
```

```
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;  
    }  
}
```

ProductCreationCommand.php

<?php

namespace Application;

class ProductCreationCommand

{

const Error_NotAuthenticated = 0x01;

const Error_DbErrorOccured = 0x02;

const Error_InvalidProductName = 0x04;

const Error_InvalidProducer = 0x08;

const Error_InvalidImage = 0x10;

public function __construct(

private \Application\Interfaces\ProductRepository \$productRepository,

private \Application\Services\AuthenticationService \$authenticationService

) {

}

 public function execute(?int \$productId, string \$productName, string \$producerName,
string \$image): int

{

\$errors = 0;

//check for authenticated user

\$userId = \$this->authenticationService->getUserId();

if(\$userId === null) {

\$errors |= self::Error_NotAuthenticated;

}

if(strlen(\$productName) == 0) {

\$errors |= self::Error_InvalidProductName;

}

if(strlen(\$producerName) == 0) {

\$errors |= self::Error_InvalidProducer;

}

if(!\$errors) {

\$result = null;

if(\$productId === null || \$productId <= 0) {

if(strlen(\$image) == 0) {

\$errors |= self::Error_InvalidImage;

return \$errors;

}

```

        $base64 = base64_encode(file_get_contents($image));
        $result = $this->productRepository->createProduct($userId, $productName, $producerName, $base64);
    } else {
        $base64 = '';
        if(strlen($image) !== 0) {
            $base64 = base64_encode(file_get_contents($image));
            $result = $this->productRepository->updateProduct($userId, $productId, $productName, $producerName, $base64);
        } else {
            $result = $this->productRepository->updateProductWithoutImage($userId, $productId, $productName, $producerName);
        }
    }
    if($result === null) {
        $errors |= self::Error_DbErrorOccured;
    }
}
return $errors;
}
}

```

ProductQuery.php

```
<?php
```

```

namespace Application;

class ProductQuery
{
    public function __construct(
        private \Application\Interfaces\ProductRepository $productRepository,
        private \Application\Interfaces\RatingRepository $ratingRepository
    ) {
    }
    public function execute(int $productId): ?\Application\ProductData
    {
        $p = $this->productRepository->getProductById($productId);
        if($p == null) {
            return null;
        }
    }
}

```

```
        return new ProductData(
            $p->getId(),
            $p->getName(),
            $p->getProducer(),
            $p->getCreatorName(),
            $p->getImage(),
            $this->ratingRepository->getCountOfRatingsForProduct($p->getId()),
            $this->ratingRepository->getAvgRatingForProduct($p->getId())
        );
    }
}
```

ProductSearchQuery.php

<?php

```
namespace Application;

class ProductSearchQuery
{
    public function __construct(
        private Interfaces\ProductRepository $productRepository,
        private \Application\Interfaces\RatingRepository $ratingRepository
    ) {
    }

    public function execute(string $filter): array
    {
        $res = [];
        foreach ($this->productRepository->getProductsForFilter($filter) as $p) {
            $res[] = new ProductData(
                $p->getId(),
                $p->getName(),
                $p->getProducer(),
                $p->getCreatorName(),
                $p->getImage(),
                $this->ratingRepository->getCountOfRatingsForProduct($p->getId()),
                $this->ratingRepository->getAvgRatingForProduct($p->getId())
            );
        }
        return $res;
    }
}
```

ProductsQuery.php

<?php

```
namespace Application;

class ProductsQuery
{
    public function __construct(
        private \Application\Interfaces\ProductRepository $productRepository,
        private \Application\Interfaces\RatingRepository $ratingRepository
    ) {
    }
    public function execute(): array
    {
        $res = [];
        foreach ($this->productRepository->getProducts() as $p) {
            $res[] = new ProductData(
                $p->getId(),
                $p->getName(),
                $p->getProducer(),
                $p->getCreatorName(),
                $p->getImage(),
                $this->ratingRepository->getCountOfRatingsForProduct($p->getId()),
                $this->ratingRepository->getAvgRatingForProduct($p->getId())
            );
        }
        return $res;
    }
}
```

RatingCreationCommand.php

<?php

```
namespace Application;

class RatingCreationCommand
{
    const Error_NotAuthenticated = 0x01;
    const Error_DbErrorOccured = 0x02;
    const Error_InvalidRating = 0x04;
    const Error_InvalidComment = 0x08;

    public function __construct(
        private \Application\Interfaces\ProductRepository $productRepository,
        private \Application\Interfaces\RatingRepository $ratingRepository,
        private \Application\Services\AuthenticationService $authenticationService
    ) {
    }

    public function execute(int $productId, int $rating, string $comment): int
    {
        $errors = 0;
        //check for authenticated user
        $userId = $this->authenticationService->getUserId();
        if($userId === null) {
            $errors |= self::Error_NotAuthenticated;
        }
        if($rating < 1 || $rating > 5) {
            $errors |= self::Error_InvalidRating;
        }
        if(strlen($comment) > 2000) {
            $errors |= self::Error_InvalidComment;
        }

        if(!$errors) {
            $rid = $this->ratingRepository->createOrUpdateRatingForProduct($userId, $product
Id, $rating, $comment);
            if($rid === null) {
                $errors |= self::Error_DbErrorOccured;
            }
        }
        return $errors;
    }
}
```

```
}
```

RatingDeleteCommand.php

```
<?php
```

```
namespace Application;
```

```
class RatingDeleteCommand
```

```
{
```

```
    const Error_NotAuthenticated = 0x01;
```

```
    const Error_DbErrorOccured = 0x02;
```

```
    public function __construct(
```

```
        private \Application\Interfaces\ProductRepository $productRepository,
```

```
        private \Application\Interfaces\RatingRepository $ratingRepository,
```

```
        private \Application\Services\AuthenticationService $authenticationService
```

```
) {
```

```
}
```

```
    public function execute(int $productId): int
```

```
{
```

```
    $errors = 0;
```

```
    //check for authenticated user
```

```
    $userId = $this->authenticationService->getUserId();
```

```
    if($userId === null) {
```

```
        $errors |= self::Error_NotAuthenticated;
```

```
    }
```

```
    if(!$errors) {
```

```
        $rows = $this->ratingRepository->deleteRatingForProduct($userId, $productId);
```

```
        if($rows === null) {
```

```
            $errors |= self::Error_DbErrorOccured;
```

```
        }
```

```
    }
```

```
    return $errors;
```

```
}
```

```
}
```

RatingsQuery.php

<?php

namespace Application;

class RatingsQuery

```
{
    public function __construct(
        private \Application\Interfaces\RatingRepository $ratingRepository
    ) {
    }
    public function execute(int $productId): array
    {
        $res = [];
        foreach ($this->ratingRepository->getRatingsForProduct($productId) as $p) {
            $res[] = new RatingData(
                $p->getId(),
                $p->getCreatorId(),
                $p->getCreatorName(),
                $p->getProductId(),
                $p->getProductName(),
                $p->getDate(),
                $p->getRating(),
                $p->getComment()
            );
        }
        return $res;
    }
}
```


RegisterCommand.php

<?php

```
namespace Application;

class RegisterCommand
{
    const Error_UserNameToShort = 0x01;
    const Error_InvalidPassword = 0x02;
    const Error_UserNameUsed = 0x04;

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

    public function execute(string $userName, string $password): int
    {
        $errors = 0;
        if (strlen($userName) < 3 || strlen($userName) > 30) {
            $errors |= self::Error_UserNameToShort;
        }
        if (strlen($password) == 0) {
            $errors |= self::Error_InvalidPassword;
        }
        if ($errors == 0) {
            $this->authenticationService->signOut();
            $user = $this->userRepository->createUser($userName, $password);
            if ($user == null) {
                $errors |= self::Error_UserNameUsed;
            } else {
                $this->authenticationService->signIn($user->getId());
            }
        }
        return $errors;
    }
}
```

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

SignInCommand.php

<?php

```
namespace Application;

class SignInCommand
{
    public function __construct(
        private \Application\Services\AuthenticationService $authenticationService,
        private \Application\Interfaces\UserRepository $userRepository
    ) {
    }
    public function execute(string $userName, string $password): bool
    {
        $this->authenticationService->signIn($userName, $password);
    }
}
```

```
$user = $this->userRepository->getUserForUserNameAndPassword($userName, $password)
;
if($user != null) {
    $this->authenticationService->signIn($user->getId());
    return true;
}
return false;
}
```

SignOutCommand.php

```
<?php

namespace Application;

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

Testfälle

Im Folgenden wird für jede geforderte funktionale Anforderung ein Screenshots gezeigt. Im selben Zug werde ich versuchen die Robustheit der Applikation zu demonstrieren.

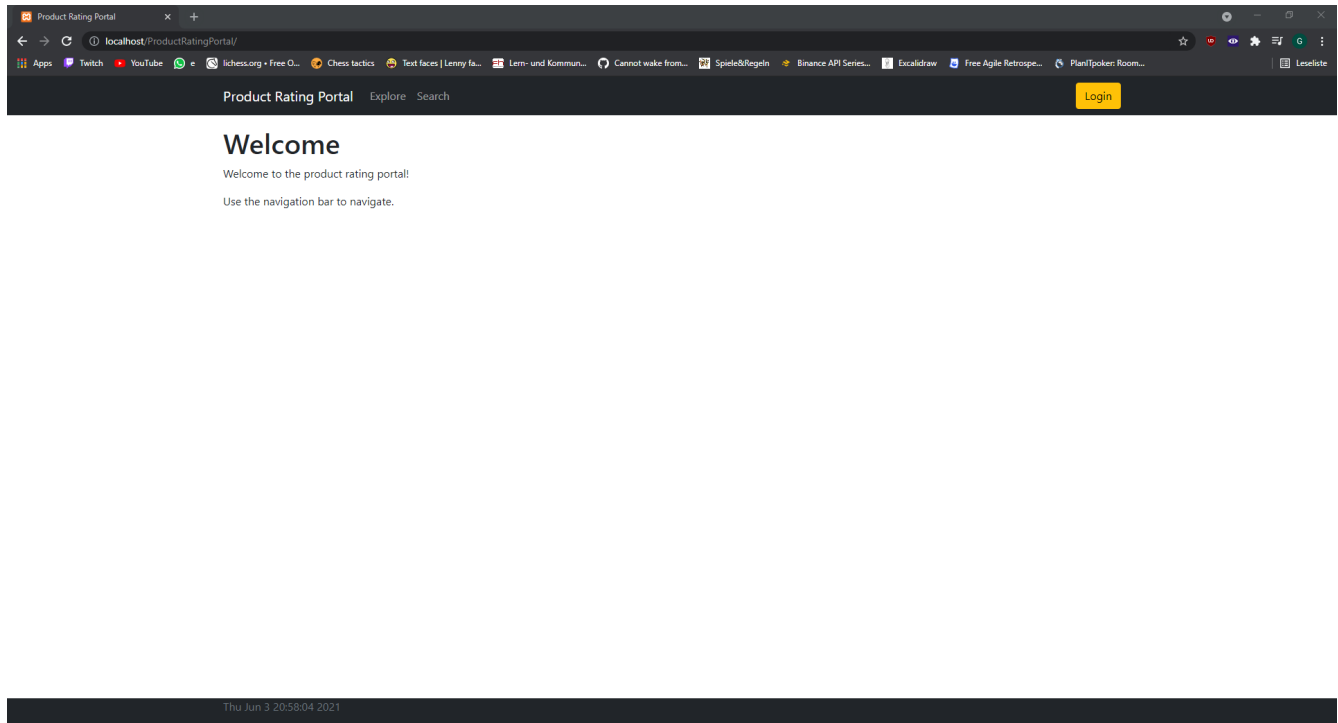


Abbildung 1: Welcome-View

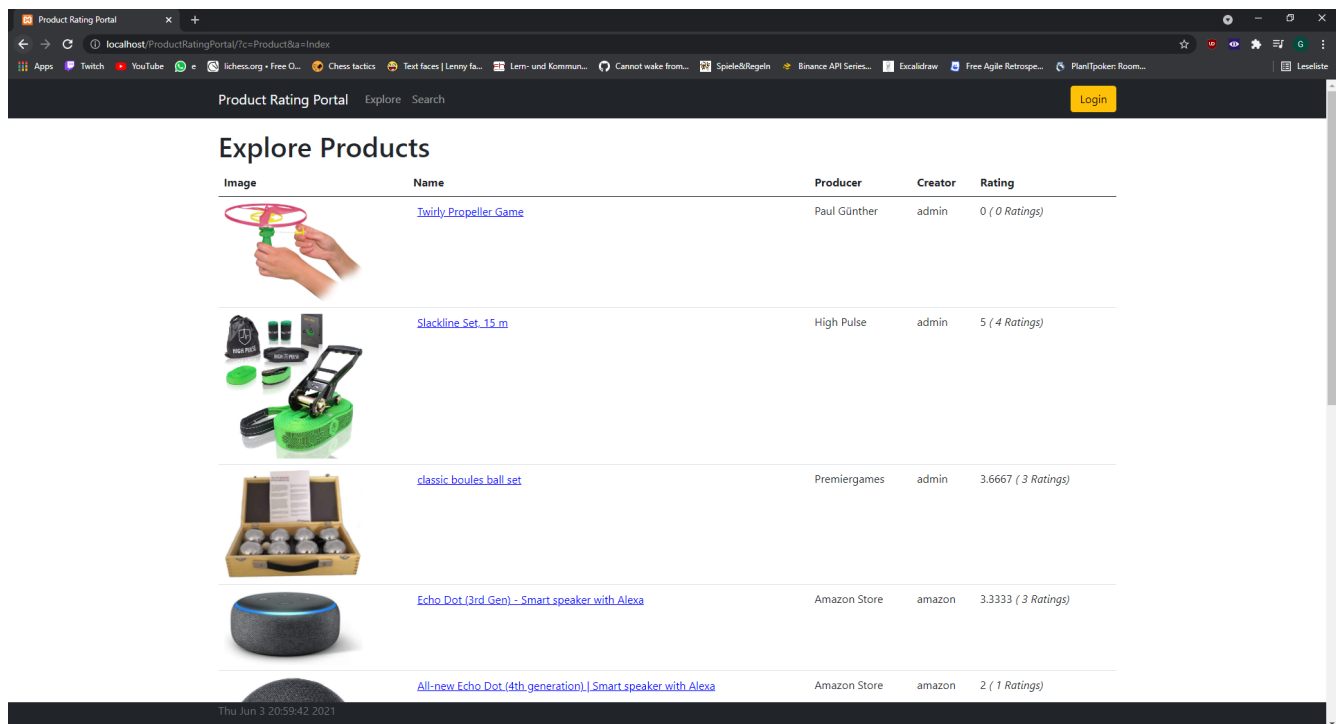


Abbildung 2: Explore: Übersicht der Produkte.

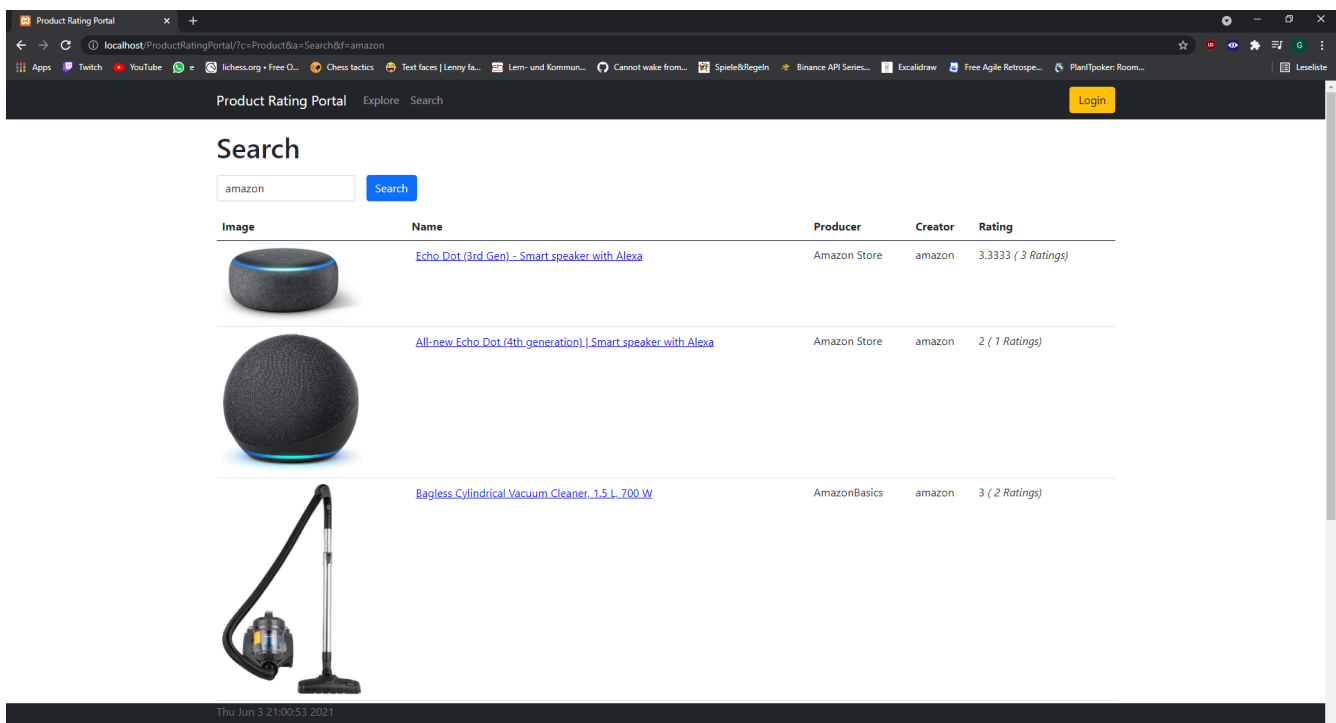
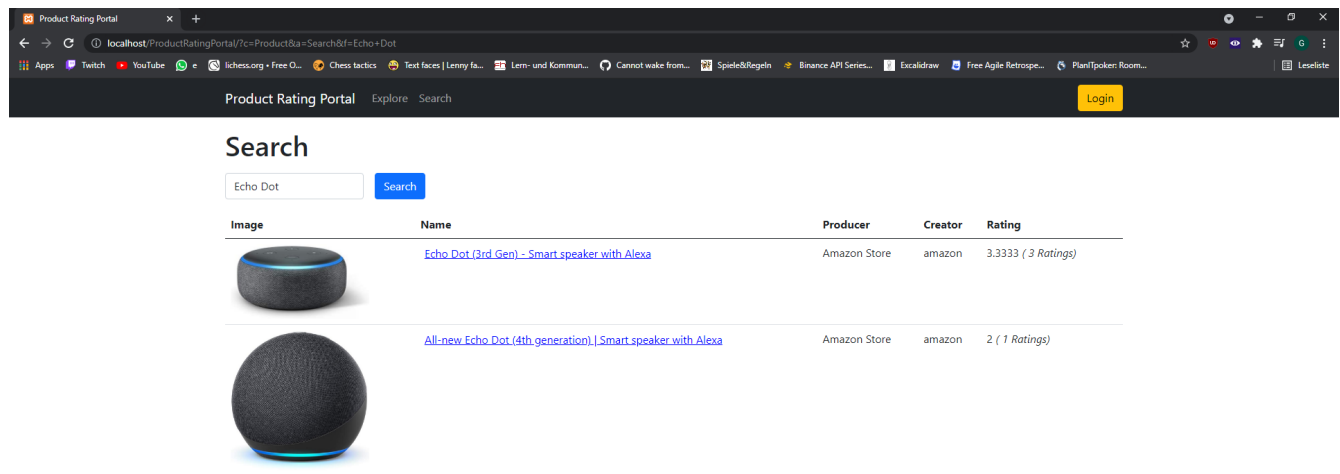
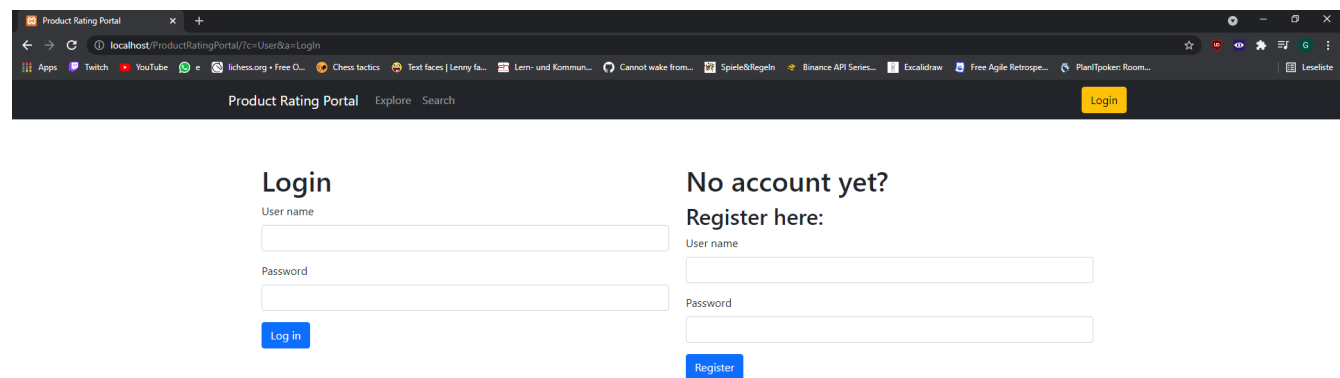


Abbildung 3: Suchfunktionalität im Reiter "Search"



Thu Jun 3 21:02:37 2021

Abbildung 4: Weiteres Beispiel zur Suchfunktionitt



Thu Jun 3 21:03:00 2021

Abbildung 5: Login-Screen

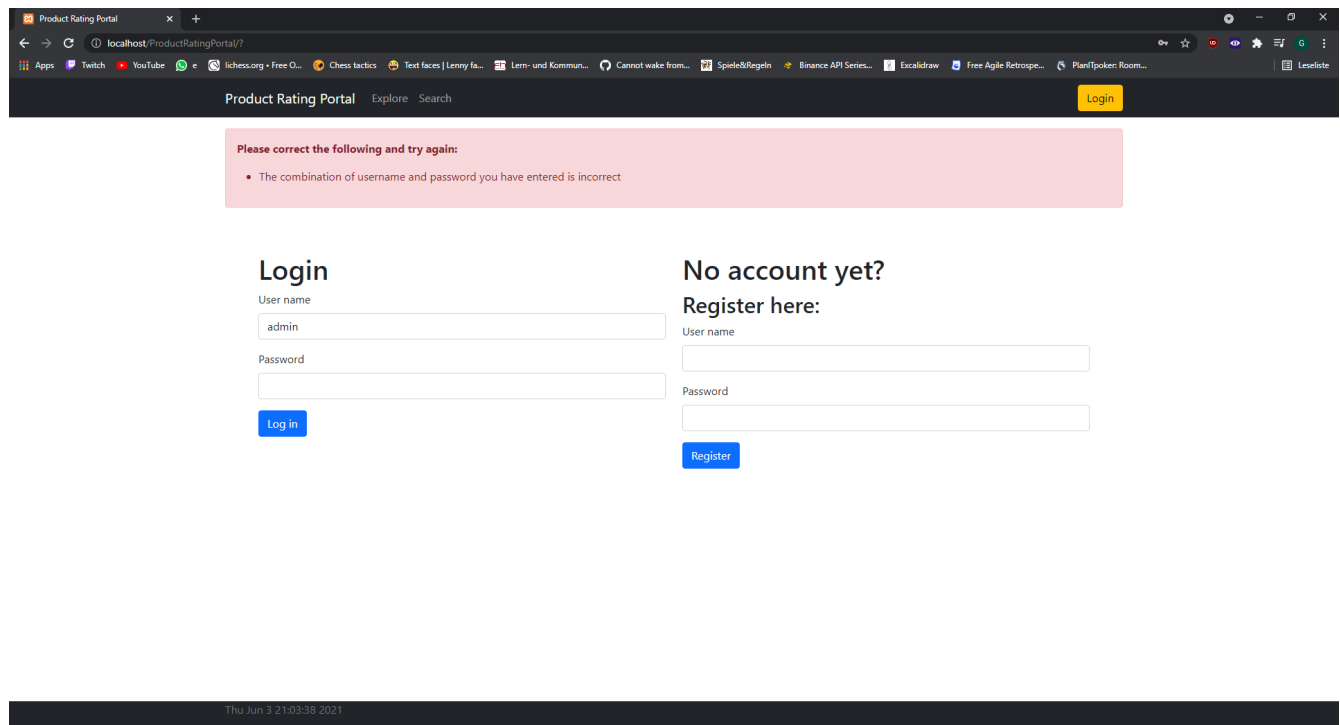


Abbildung 6: Error-Meldung bei falschen Login-Daten

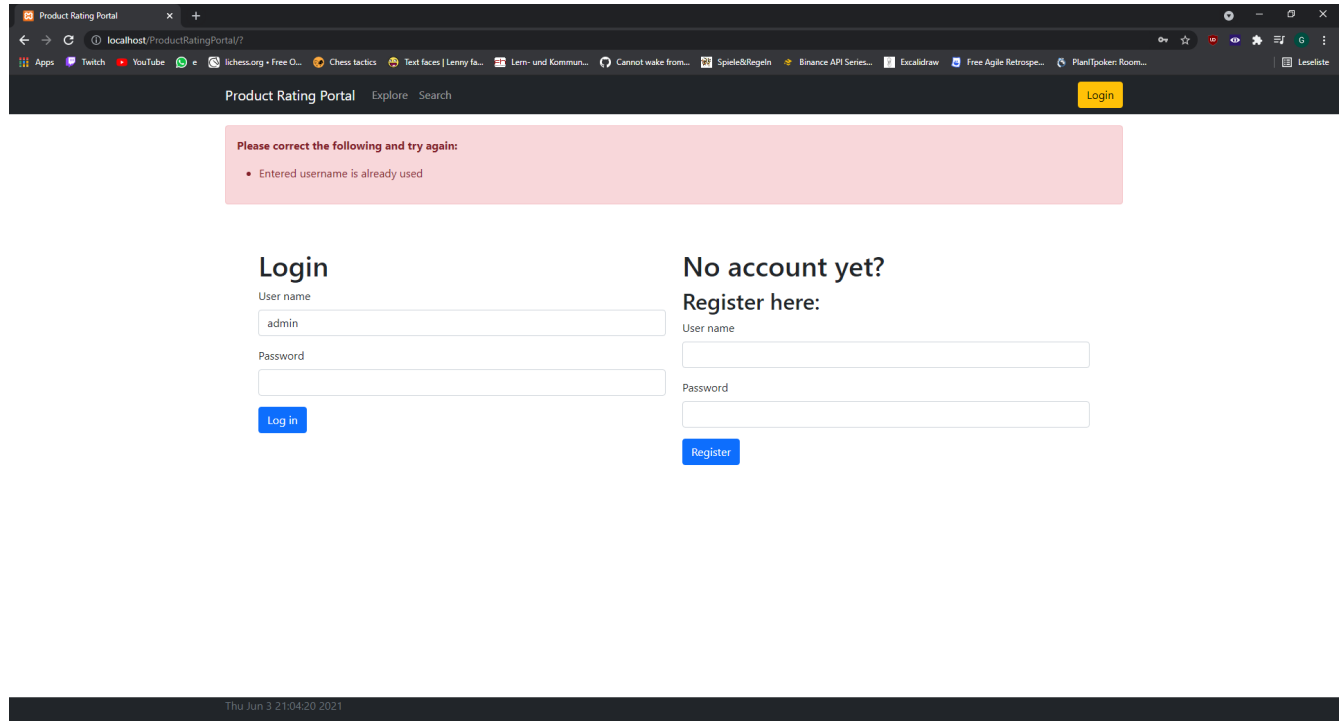


Abbildung 7: Versuch der Registrierung mit einem bereits existierenden Usernamen

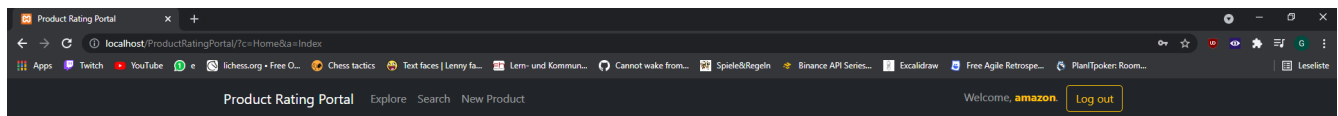


Abbildung 8: Erfolgreiches Login -> Username wird angezeigt. Neuer Reiter "New Product" wird angezeigt




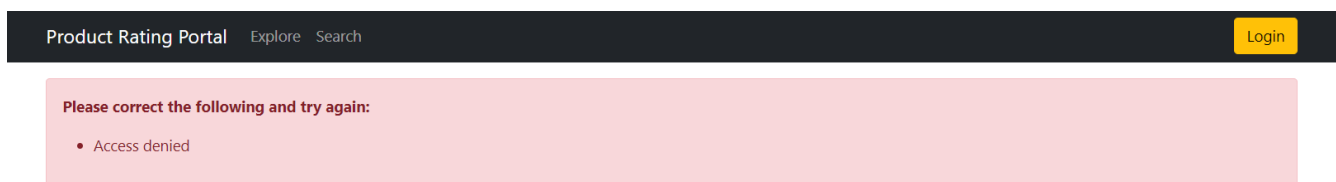
| | | | | | |
|--|--|--------------|--------|---------------------|----------------------|
|  | classic boules ball set | Premiergames | admin | 3.6667 (3 Ratings) | |
|  | Echo Dot (3rd Gen) - Smart speaker with Alexa | Amazon Store | amazon | 3.3333 (3 Ratings) | Edit |
|  | All-new Echo Dot (4th generation) Smart speaker with Alexa | Amazon Store | amazon | 2 (1 Ratings) | Edit |

Abbildung 9: Der eingeloggte User "amazon" sieht neben seinen Produkten einen Edit-Button

Dieser Edit-Button führt zb zu `http://localhost/ProductRatingPortal/?p=23&pn=Echo+Dot+%283rd+Gen%29+-+Smart+speaker+with+Alexa&pc=Amazon+Store&c=Product&a=Edit`



Explore Products

Abbildung 10: Ein nicht-eingeloggter User kann darauf nicht zugreifen, auch wenn er den Link manuell eingibt

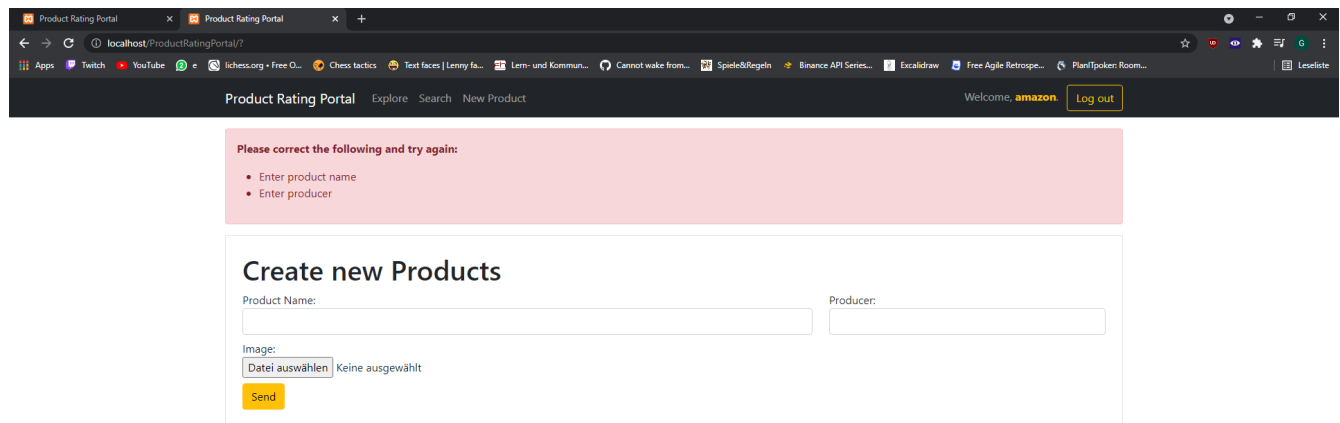


Abbildung 11: Validierung der Usereingabe bei neuen Produkten

The screenshot shows the "Create new Products" form with the following data: "Product Name:" is "Beispiel", "Producer:" is "Hersteller 1", and "Image:" is "Datei auswählen yZlqh.png". A yellow "Send" button is at the bottom of the form.

Abbildung 12: Erstellung eines validen Produkts

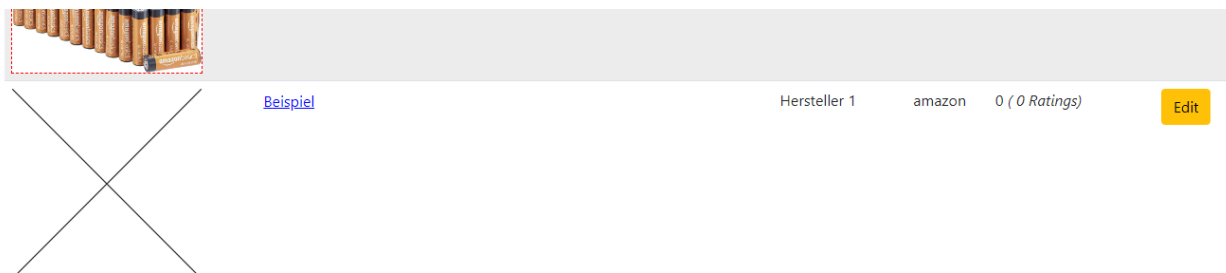


Abbildung 13: Weiterleitung zu Explore. Das Produkt wird angezeigt

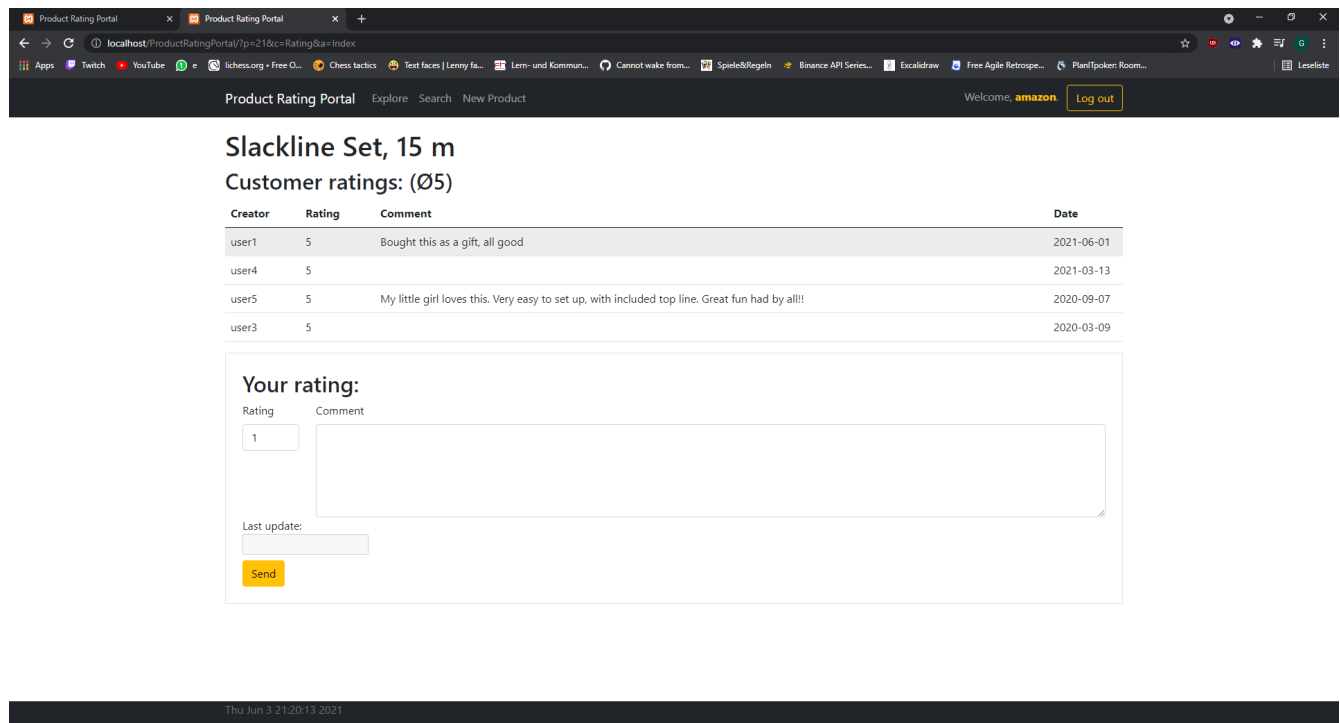


Abbildung 14: Detailansicht eines Produkts. Nachdem ein User eingeloggt ist, wird hier auch ein Formular mit dein eigenen Rating angezeigt

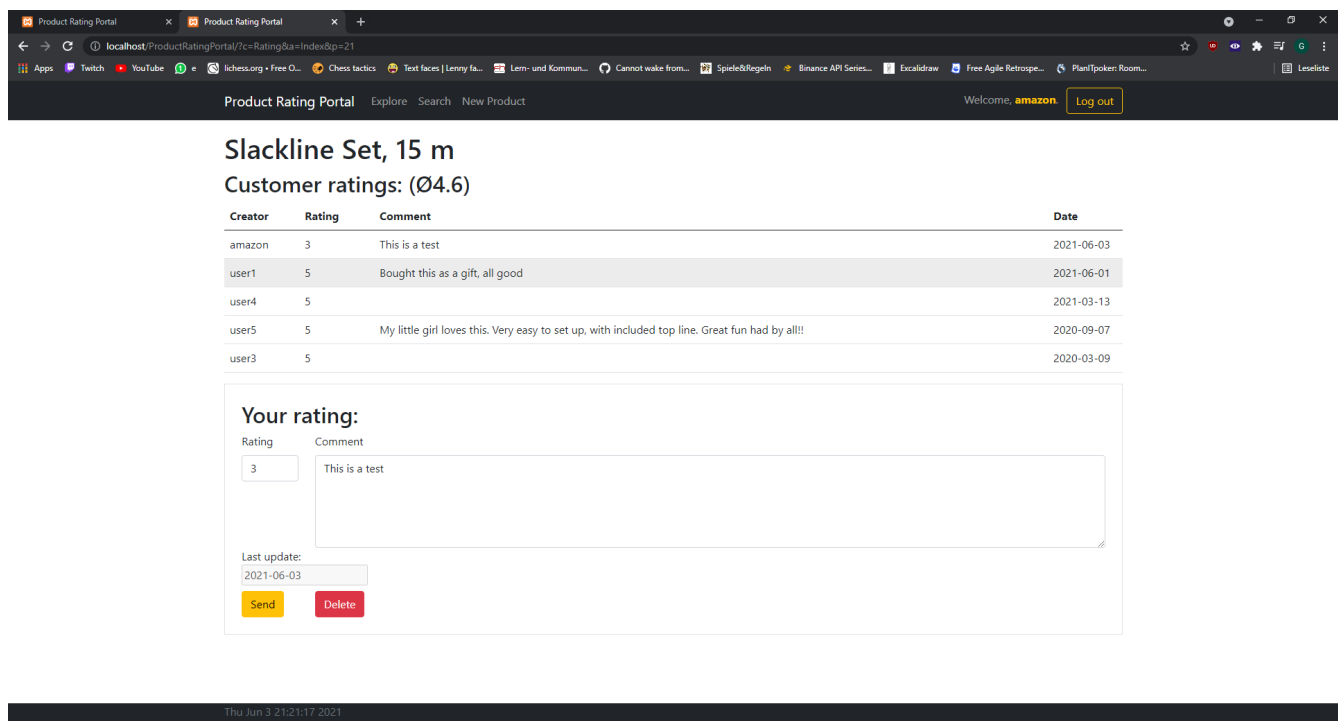
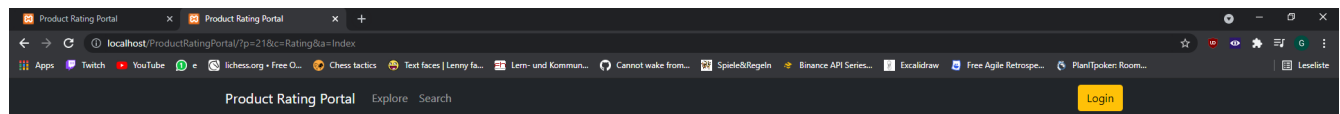
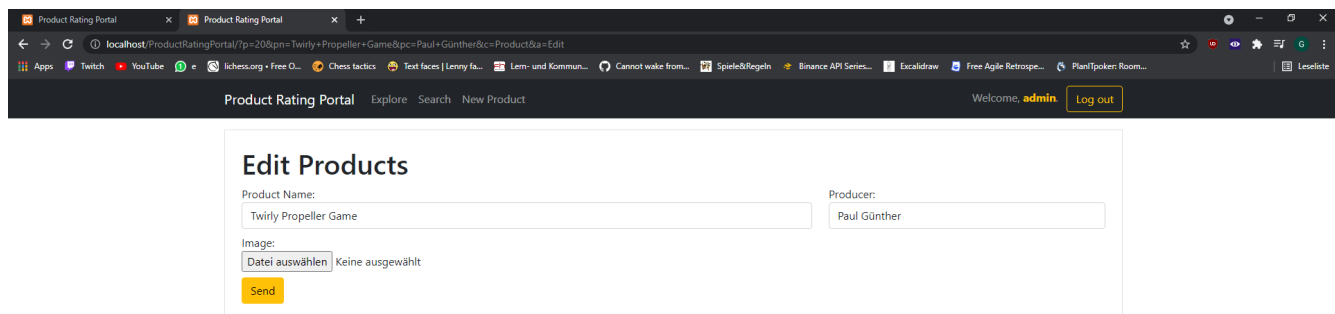


Abbildung 15: Wurde ein Rating abgegeben, so kann diese über den Delete Button gelöscht werden



Thu Jun 3 21:22:25 2021

Abbildung 16: Ausgeloggte User sehen alle Ratings aber können keine eigenen anlegen



Thu Jun 3 21:24:26 2021

Abbildung 17: Bearbeiten eines Produkts. Wenn kein Bild ausgewählt wird, wird das alte übernommen