SCR4x Projektarbeit SS 2017/18

☐ **Gruppe 1** DI (FH) G. Horn, MSc

☐ **Gruppe 2** DI (FH) I. Krammer

Abgabetermin: Mittwoch, 03.06.2018, 24:00 Uhr

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 funktionalen und technischen Anforderungen 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 allerdings 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.
- Alle zu speichernden Daten sollen in einer MySQL-Datenbank abgelegt werden. Entwickeln Sie dazu ein entsprechendes Datenbankmodell und dokumentieren Sie es ausführlich (Diagramme etc.).
- Geschäftslogik (Datenbankzugriff, Prüfen von Berechtigungen etc.) und Präsentation müssen möglichst gut entkoppelt werden. Sämtliche nötige Sicherheitsprüfungen müssen immer (auch) direkt in der Geschäftslogik erfolgen.
- Passwörter dürfen aus Sicherheitsgründen nicht im Klartext 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 manuelle Aufrufen von Skripts über entsprechend abgeänderte URLs 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 laut Stundenplan vorgesehenen Termin in Form einer Live-Demonstration zu präsentieren.

Abzugebende Komponenten

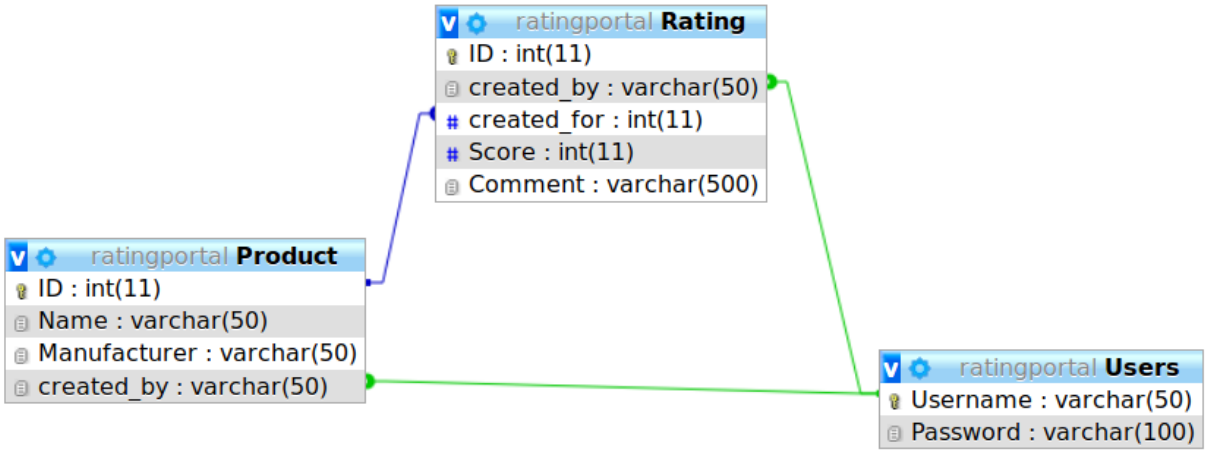
- Datenbankskript zur Erstellung der Datenbank als Textdatei
- Datenbankskript zur Erstellung von Testdaten in der Datenbank, mit denen auch bei der Entwicklung bereits getestet wurde, als Textdatei
- Ordner der entwickelten Webseite mit allen relevanten Daten (HTML-Seiten, PHP-Skripts, Stylesheets, andere Ressourcen wie Bilder etc.)
- 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

1 Produktwebertungportal

1.1 Lösungsidee

Wie in der Angabe vorgegeben wurde das MVC Muster angewandt. Sowohl die Struktur als auch zum Teil der Code wurden von der Übung übernommen. Das Model besteht aus den Entitäten User, Product und Rating. Die Repräsentation in der Datenbank kann im Bild weiter unten entnommen werden. Auf der Hauptseite befindet sich eine Liste mit allen Produkten sowie die Möglichkeit die angezeigten Produkte zu filtern. Wird auf ein Produkt geklickt, so wird die Detailansicht geöffnet. Erst wenn sich ein Benutzer registriert und angemeldet hat, erscheint die Schaltfläche um ein neues Produkt anzulegen. In der Detailansicht wird nach der Anmeldung ebenfalls eine Schaltfläche aktiviert um ein Rating abzugeben.

1.2 Datenbank




```
CREATE DATABASE IF NOT EXISTS`ratingportal`

CREATE TABLE `Users` (
  `Username` varchar(50) NOT NULL,
  `Password` varchar(100) NOT NULL,
  PRIMARY KEY (`Username`)
) ENGINE=InnoDB DEFAULT CHARSET=utf8

CREATE TABLE `Product` (
  `ID` int(11) NOT NULL AUTO_INCREMENT,
  `Name` varchar(50) NOT NULL,
  `Manufacturer` varchar(50) NOT NULL,
  `created_by` varchar(50) NOT NULL,
  PRIMARY KEY (`ID`),
  KEY `created_by` (`created_by`),
  CONSTRAINT `Product_ibfk_1` FOREIGN KEY (`created_by`) REFERENCES `Users` (`Username`)
) ENGINE=InnoDB AUTO_INCREMENT=4 DEFAULT CHARSET=utf8

CREATE TABLE `Rating` (
  `ID` int(11) NOT NULL AUTO_INCREMENT,
  `created_by` varchar(50) NOT NULL,
  `created_for` int(11) NOT NULL,
  `Score` int(11) NOT NULL,
  `Comment` varchar(500) DEFAULT NULL,
  PRIMARY KEY (`ID`),
  KEY `rating_created_by` (`created_by`),
  KEY `rating_created_for` (`created_for`),
```

```
 CONSTRAINT `Rating_ibfk_1` FOREIGN KEY (`created_by`) REFERENCES `Users` (`Username`),
 CONSTRAINT `Rating_ibfk_2` FOREIGN KEY (`created_for`) REFERENCES `Product` (`ID`) ON
    DELETE CASCADE
) ENGINE=InnoDB AUTO_INCREMENT=18 DEFAULT CHARSET=utf8

/* If you cant execute this script try every table table individually */
```

../database/create.txt

```
INSERT INTO `Users` (`Username`, `Password`) VALUES
('chris', 'chris'),
('elisabeth', 'elisabeth'),
('hans', 'hans'),
('lisa', 'lisa'),
('sepp', 'sepp'),
('theresa', 'theresa'),
('tom', 'tom');

INSERT INTO `Product` (`ID`, `Name`, `Manufacturer`, `created_by`) VALUES
(8, 'IPhone 5', 'Apple', 'chris'),
(9, 'IPhone 6', 'Apple', 'chris'),
(10, 'Thinkpad T460s', 'Lenovo', 'elisabeth'),
(11, 'T-800', 'Cyberdyne Systems', 'hans'),
(12, 'Kerbal Space Program', 'Squad', 'hans'),
(13, 'Civic Type R', 'Honda', 'lisa'),
(14, 'MSA 120', 'Stihl', 'sepp'),
(15, 'F-1', 'Rockedyne', 'theresa'),
(16, 'Thinkpad T460', 'Lenovo', 'tom');

INSERT INTO `Rating` (`ID`, `created_by`, `created_for`, `Score`, `Comment`) VALUES
(21, 'chris', 8, 3, 'could be better...'),
(22, 'chris', 9, 5, 'shit is lit fam'),
(23, 'elisabeth', 10, 4, ''),
(24, 'hans', 11, 5, 'it will get you man'),
(25, 'hans', 12, 5, ''),
(26, 'lisa', 13, 3, ''),
(27, 'sepp', 14, 4, ''),
(28, 'theresa', 15, 5, 'it will blow you away'),
(29, 'tom', 16, 4, ''),
(30, 'tom', 9, 4, ''),
(31, 'tom', 11, 5, ''),
(32, 'tom', 12, 5, 'played it for years.'),
(33, 'tom', 13, 4, ''),
(34, 'tom', 14, 4, ''),
(35, 'theresa', 8, 4, ''),
(36, 'theresa', 10, 4, ''),
(37, 'theresa', 11, 1, ''),
(38, 'theresa', 13, 4, ''),
(39, 'theresa', 16, 2, ''),
(40, 'sepp', 8, 4, ''),
(41, 'sepp', 9, 5, ''),
(42, 'sepp', 10, 5, ''),
(43, 'sepp', 11, 5, ''),
(44, 'sepp', 12, 4, ''),
(45, 'sepp', 13, 5, ''),
(46, 'sepp', 15, 5, ''),
(47, 'lisa', 8, 4, ''),
(48, 'lisa', 10, 4, ''),
(49, 'lisa', 11, 4, ''),
(50, 'lisa', 14, 5, ''),
(51, 'lisa', 16, 4, ''),
(52, 'hans', 9, 2, ''),
(53, 'hans', 14, 4, ''),
(54, 'elisabeth', 8, 4, ''),
(55, 'elisabeth', 16, 5, ''),
```

```
(56, 'elisabeth', 13, 4, ''),
(57, 'chris', 10, 4, ''),
(58, 'chris', 11, 5, ''),
(59, 'chris', 12, 5, ''),
(60, 'chris', 13, 5, ''),
(61, 'chris', 14, 5, ''),
(62, 'chris', 15, 2, ''),
(63, 'chris', 16, 2, '');
```

../database/testdata.txt

1.3 Sourcecode

```
<?php
// index.php acts as router
spl_autoload_register(function ($class) {
    $file = __DIR__ . '/src/' . str_replace('\\', '/', $class) . '.php';
    if (file_exists($file)) {
        require_once ($file);
    }
});
\Framework\Injector::register(\DataLayer\DataLayer::class, false, \DataLayer\DBDataLayer::
    class, array(
    'server' => 'localhost',
    'userName' => 'root',
    'password' => '',
    'database' => 'ratingportal'
));
\Framework\Injector::register(\BusinessLogic\Session::class, true);
\Framework\MVC::handleRequest();
```

../index.php

```
<?php
namespace BusinessLogic;

final class AuthenticationManager
{
    private $dataLayer;
    private $session;
    const SESSION_USER_ID = 'userId';
    public function __construct(\DataLayer\DataLayer $dataLayer, \BusinessLogic\Session
    $session)
    {
        $this->dataLayer = $dataLayer;
        $this->session = $session;
    }
    public function authenticate($userName, $password)
    {
        $user = $this->dataLayer->getUserForUserNameAndPassword($userName, $password);
        if ($user != null) {
           $this->session->storeValue(self::SESSION_USER_ID, $user->getId());
           return true;
        }
        self::signOut();
        return false;
    }
    public function signOut()
    {
        $this->session->deleteValue(self::SESSION_USER_ID);
    }
    public function isAuthenticated()
```

```
    {
30          return $this->session->hasValue(self::SESSION_USER_ID);
    }
32      public function getAuthenticatedUser()
    {
34          return $this->isAuthenticated() ? $this->dataLayer->getUser($this->session->getValue
    (self::SESSION_USER_ID)) : null;
    }
36  }
```

../src/BusinessLogic/AuthenticationManager.php

```
<?php
2  namespace BusinessLogic;

4  final class Session
   {
6      public function __construct()
       {
8          session_start();
       }
10     public function hasValue($key)
       {
12         return isset($_SESSION[$key]);
       }
14     public function getValue($key, $defaultValue = null)
       {
16         return $this->hasValue($key) ? $_SESSION[$key] : $defaultValue;
       }
18     public function storeValue($key, $value)
       {
20         $_SESSION[$key] = $value;
       }
22     public function deleteValue($key)
       {
24         unset($_SESSION[$key]);
       }
26 }
```

../src/BusinessLogic/Session.php

```
<?php
2  namespace Controllers;

4  class Home extends \Framework\Controller
   {
6      private $authenticationManager;
       private $dataLayer;
8      public function __construct(\DataLayer\DataLayer $dataLayer, \BusinessLogic\
       AuthenticationManager $authenticationManager)
       {
10         $this->dataLayer = $dataLayer;
           $this->authenticationManager = $authenticationManager;
12     }
       public function GET_Index()
14     {

16         $products = array();

18         foreach ($this->dataLayer->getProducts() as $product) {
               $d = array();
20             $d[0] = $this->dataLayer->getAvarageRating($product->getId());
               $d[1] = $this->dataLayer->getNumRatings($product->getId());
22             $d[2] = $product;
```

```
            $d[3] = $this->dataLayer->getUser($product->getUserID());
            array_push($products, $d);
        }

        return $this->renderView('home', array(
            'user' => $this->authenticationManager->getAuthenticatedUser(),
            'products' => $products,
        ));
    }

    public function POST_Index()
    {
        $products = array();

        foreach ($this->dataLayer->getProductsWithFilter($this->getParam('f')) as $product)
    {
            $d = array();
            $d[0] = $this->dataLayer->getAvarageRating($product->getId());
            $d[1] = $this->dataLayer->getNumRatings($product->getId());
            $d[2] = $product;
            $d[3] = $this->dataLayer->getUser($product->getUserID());
            array_push($products, $d);
        }

        return $this->renderView('home', array(
            'user' => $this->authenticationManager->getAuthenticatedUser(),
            'products' => $products,
        ));
    }
}
```

../src/Controllers/Home.php

```
<?php
namespace Controllers;

class NewProduct extends \Framework\Controller
{
    private $authenticationManager;
    private $dataLayer;
    public function __construct(\DataLayer\DataLayer $dataLayer, \BusinessLogic\
    AuthenticationManager $authenticationManager)
    {
        $this->dataLayer = $dataLayer;
        $this->authenticationManager = $authenticationManager;
    }

    public function GET_Index()
    {
        return $this->renderView('newproduct', array(
            'user' => $this->authenticationManager->getAuthenticatedUser(),
        ));
    }

    public function GET_Update()
    {
        return $this->renderView('newproduct', array(
            'user' => $this->authenticationManager->getAuthenticatedUser(),
            'product' => $this->dataLayer->getProductById($this->getParam('pid'))
        ));
    }



    public function POST_Create(){
```

```
        if ($this->getParam('pn') == "" || $this->getParam('manu') == "") {
            return $this->renderView('newproduct', array(
                'user' => $this->authenticationManager->getAuthenticatedUser(),
                'errors' => array('No empty fields allowed'),
            ));
        }


        $p = $this->dataLayer->createProduct(
            $this->getParam('pn'),
            $this->getParam('manu'),
            $this->authenticationManager->getAuthenticatedUser()->getUserName()
        );

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

    public function POST_Update(){
        if ($this->getParam('pn') == "" || $this->getParam('manu') == "") {
            return $this->renderView('newproduct', array(
                'user' => $this->authenticationManager->getAuthenticatedUser(),
                'product' => $this->dataLayer->getProductById($this->getParam('pid')),
                'errors' => array('No empty fields allowed')
            ));
        }


        $p = $this->dataLayer->updateProduct(
            $this->getParam('pid'),
            $this->getParam('pn'),
            $this->getParam('manu'),
            $this->authenticationManager->getAuthenticatedUser()->getUserName()
        );

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

    public function GET_Delete(){

        $this->dataLayer->deleteProduct($this->getParam('pid'));

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

../src/Controllers/NewProduct.php

```
<?php
namespace Controllers;

class NewRating extends \Framework\Controller
{
    private $authenticationManager;
    private $dataLayer;
    public function __construct(\DataLayer\DataLayer $dataLayer, \BusinessLogic\
    AuthenticationManager $authenticationManager)
    {
        $this->dataLayer = $dataLayer;
        $this->authenticationManager = $authenticationManager;
    }

```

```
    public function GET_Index()
    {
        return $this->renderView('newrating', array(
            'user' => $this->authenticationManager->getAuthenticatedUser(),
            'product' => $this->dataLayer->getProductById($this->getParam('pid')),
        ));
    }

    public function GET_Update()
    {
        return $this->renderView('newrating', array(
            'user' => $this->authenticationManager->getAuthenticatedUser(),
            'product' => $this->dataLayer->getProductById($this->getParam('pid')),
            'rating' => $this->dataLayer->getRatingById($this->getParam('rid')),
        ));
    }

    public function POST_Create()
    {
        if ($this->getParam('rating') < 1 || $this->getParam('rating') > 5) {
            return $this->renderView('newrating', array(
                'user' => $this->authenticationManager->getAuthenticatedUser(),
                'product' => $this->dataLayer->getProductById($this->getParam('pid')),
                'errors' => array("Invalid data provided"),
            ));
        }

        $p = $this->dataLayer->createRating(
            $this->getParam('pid'),
            $this->authenticationManager->getAuthenticatedUser()->getUserName(),
            $this->getParam('rating'),
            $this->getParam('cm')
        );

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

    public function POST_Update()
    {

        if ($this->getParam('rating') < 1 || $this->getParam('rating') > 5) {
            return $this->renderView('newrating', array(
                'user' => $this->authenticationManager->getAuthenticatedUser(),
                'product' => $this->dataLayer->getProductById($this->getParam('pid')),
                'rating' => $this->dataLayer->getRatingById($this->getParam('rid')),
                'errors' => array("Invalid data provided"),
            ));
        }

        $this->dataLayer->updateRating(
            $this->getParam('rid'),
            $this->authenticationManager->getAuthenticatedUser()->getUserName(),
            $this->getParam('rating'),
            $this->getParam('cm')
        );

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

    public function GET_Delete()
    {
```

```
79
        $this->dataLayer->deleteRating($this->getParam('rid'));
81
        return $this->redirect('Index', 'UserHome');
83    }
}
```

../src/Controllers/NewRating.php

```
<?php
2 namespace Controllers;

4 class Product extends \Framework\Controller
{
6     private $authenticationManager;
      private $dataLayer;
8     public function __construct(\DataLayer\DataLayer $dataLayer, \BusinessLogic\
      AuthenticationManager $authenticationManager)
      {
10        $this->dataLayer = $dataLayer;
          $this->authenticationManager = $authenticationManager;
12    }
      public function GET_Index()
14    {
          return $this->renderView('product', array(
16            'user' => $this->authenticationManager->getAuthenticatedUser(),
              'product' => $this->dataLayer->getProductById($this->getParam('pid')),
18            'score' => $this->dataLayer->getAvarageRating($this->getParam('pid')),
              'ratings' => $this->dataLayer->getRatingsByProductId($this->getParam('pid'))
20        ));
      }
22 }
```

../src/Controllers/Product.php

```
<?php
2 namespace Controllers;

4 class User extends \Framework\Controller
{
6
      private $authenticationManager;
8     private $dataLayer;

10    const PARAM_USER_NAME = 'un';
      const PARAM_PASSWORD = 'pwd';
12    public function __construct(\DataLayer\DataLayer $dataLayer, \BusinessLogic\
      AuthenticationManager $authenticationManager)
      {
14        $this->dataLayer = $dataLayer;
          $this->authenticationManager = $authenticationManager;
16    }

18    public function GET_LogIn()
      {
20        if ($this->authenticationManager->isAuthenticated()) {
              return $this->redirect('Index', 'Home');
22        }
          return $this->renderView('login', array(
24            'user' => $this->authenticationManager->getAuthenticatedUser(),
              'userName' => $this->getParam(self::PARAM_USER_NAME),
26        ));
      }
28    public function POST_LogIn()
```

```
    {
        if (!$this->authenticationManager->authenticate(
            $this->getParam(self::PARAM_USER_NAME),
            $this->getParam(self::PARAM_PASSWORD)
        )) {
            return $this->renderView('login', array(
                'user' => $this->authenticationManager->getAuthenticatedUser(),
                'userName' => $this->getParam(self::PARAM_USER_NAME),
                'errors' => array('Invalid user name or password.'),
            ));
        }
        return $this->redirect('Index', 'Home');
    }
    public function POST_LogOut()
    {
        $this->authenticationManager->signOut();
        return $this->redirect('Index', 'Home');
    }

    public function GET_Register()
    {
        return $this->renderView('register', array(
            'user' => $this->authenticationManager->getAuthenticatedUser(),
            'userName' => $this->getParam(self::PARAM_USER_NAME),
        ));
    }

    public function POST_Register()
    {
        if ($this->getParam(self::PARAM_USER_NAME) == "" || $this->getParam(self::
    PARAM_PASSWORD) == ""){
            return $this->renderView('register', array(
                'user' => $this->authenticationManager->getAuthenticatedUser(),
                'userName' => $this->getParam(self::PARAM_USER_NAME),
                'errors' => array('invalid username or password'),
            ));
        }

        if ($this->dataLayer->createUser(
            $this->getParam(self::PARAM_USER_NAME),
            $this->getParam(self::PARAM_PASSWORD))) {
            return $this->redirect('LogIn', 'User');
        } else {
            return $this->renderView('register', array(
                'user' => $this->authenticationManager->getAuthenticatedUser(),
                'userName' => $this->getParam(self::PARAM_USER_NAME),
                'errors' => array('User already exists'),
            ));
        }
    }
}
```

../src/Controllers/User.php

```
<?php
namespace Controllers;

class UserHome extends \Framework\Controller
{
    private $authenticationManager;
    private $dataLayer;
    public function __construct(\DataLayer\DataLayer $dataLayer, \BusinessLogic\
    AuthenticationManager $authenticationManager)
    {
        $this->dataLayer = $dataLayer;
```

```
        $this->authenticationManager = $authenticationManager;
    }
    public function GET_Index()
    {
        $productRatings = array();
        $ratings = $this->dataLayer->getRatingsByUser($this->authenticationManager->
    getAuthenticatedUser()->getUserName());


        foreach ($ratings as $rating) {
            $d = array();
            $d[0] = $this->dataLayer->getProductById($rating->getProductID());
            $d[1] = $rating;
            array_push($productRatings, $d);
        }

        return $this->renderView('userhome', array(
            'user' => $this->authenticationManager->getAuthenticatedUser(),
            'products' => $this->dataLayer->getProductsByUser($this->authenticationManager->
    getAuthenticatedUser()->getUserName()),
            'ratings' => $productRatings,
        ));
    }
}
```

../src/Controllers/UserHome.php

```
<?php
namespace DataLayer;

interface DataLayer
{
    public function getUser($id);
    public function getUserForUserNameAndPassword($userName, $password);
    public function createUser($userName, $password);

    public function getProducts();
    public function getProductById($id);
    public function getProductsWithFilter($filter);
    public function getRatingsByProductId($productID);
    public function getProductsByUser($userName);
    public function getRatingsByUser($userName);
    public function createProduct($productName, $manufacturer, $uid);
    public function createRating($pid, $uid, $score, $comment);
    public function getAvarageRating($pid);
    public function getNumRatings($pid);
    public function deleteRating($rid);
    public function deleteProduct($pid);
    public function updateProduct($pid, $productName, $manufacturer, $uid);
    public function getRatingById($rid);
    public function updateRating($rid, $uid, $score, $comment);
}
```

../src/DataLayer/DataLayer.php

```
<?php
namespace DataLayer;

use \Domain\Product;
use \Domain\Rating;
use \Domain\User;

class DBDataLayer implements DataLayer
{
```

```
    private $server;
    private $userName;
    private $password;
    private $database;

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

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

    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`: ' . $connection->error);
        }
        return $statement;
    }

    public function getUser($id)
    {
        $user = null;

        $conn = $this->getConnection();
        $stat = $this->executeStatement(
            $conn,
            'SELECT Username, Password FROM Users WHERE Username = ?',
            function ($s) use ($id) {
                $s->bind_param('s', $id);
            }
        );
        $stat->bind_result($userName, $password);
        if ($stat->fetch()) {
            $user = new User($userName, $userName, $password);
        }
        $stat->close();
        $conn->close();
        return $user;
    }
```

```
75      public function getUserForUserNameAndPassword($userName, $password)
        {
77          $user = null;
            $conn = $this->getConnection();
79          $stat = $this->executeStatement(
                $conn,
81              'SELECT Username, Password FROM Users WHERE Username = ?',
                function ($s) use ($userName) {
83                  $s->bind_param('s', $userName);
                }
85          );
            $stat->bind_result($un, $pw);
87
            if ($stat->fetch() && $password == $pw) {
89              $user = new User($un, $un, $pw);
            }
91          $stat->close();
            $conn->close();
93          return $user;
        }
95
        public function createUser($userName, $password)
97      {
            if($this->getUser($userName) != null){
99              return false;
            }
101
            $conn = $this->getConnection();
103         $stat = $this->executeStatement(
                $conn,
105             'INSERT INTO Users (Username, Password) VALUES (?, ?)',
                function ($s) use ($userName, $password) {
107                 $s->bind_param('ss', $userName, $password);
                }
109         );
            return true;
111     }

113     public function getProducts()
        {
115         $products = array();

117         $conn = $this->getConnection();
            $res = $this->executeQuery($conn, 'SELECT ID, Name, Manufacturer, created_by FROM
        Product');
119         while ($p = $res->fetch_object()) {
                $products[] = new Product($p->ID, $p->Name, $p->created_by, $p->Manufacturer);
121         }
            $res->close();
123         $conn->close();
            return $products;
125     }

127     public function getProductById($id)
        {
129         $products = array();

131         $conn = $this->getConnection();
            $res = $this->executeStatement($conn, 'SELECT ID, Name, Manufacturer, created_by
        FROM Product WHERE ID = ?',
133             function ($s) use ($id) {
                    $s->bind_param('i', $id);
135             }
            );
```

```
137         $res->bind_result($ID, $Name, $Manufacturer, $created_by);
            while ($res->fetch()) {
139             $products[] = new Product($ID, $Name, $created_by, $Manufacturer);
            }
141         $res->close();
            $conn->close();
143         return $products[0];
        }
145
        public function getProductsByUser($userName)
147     {
            $products = array();
149
            $conn = $this->getConnection();
151         $res = $this->executeStatement($conn, 'SELECT ID, Name, Manufacturer, created_by
    FROM Product WHERE created_by = ?',
                function ($s) use ($userName) {
153                 $s->bind_param('s', $userName);
                }
155         );
            $res->bind_result($ID, $Name, $Manufacturer, $created_by);
157         while ($res->fetch()) {
                $products[] = new Product($ID, $Name, $created_by, $Manufacturer);
159         }
            $res->close();
161         $conn->close();
            return $products;
163     }

165     public function getProductsWithFilter($filter)
        {
167         $sqlfilter = "%" . $filter . "%";
            $products = array();
169
            $conn = $this->getConnection();
171         $res = $this->executeStatement($conn, 'SELECT ID, Name, Manufacturer, created_by
    FROM Product WHERE Name LIKE ? OR Manufacturer LIKE ?',
                function ($s) use ($sqlfilter) {
173                 $s->bind_param('ss', $sqlfilter, $sqlfilter);
                }
175         );
            $res->bind_result($ID, $Name, $Manufacturer, $created_by);
177         while ($res->fetch()) {
                $products[] = new Product($ID, $Name, $created_by, $Manufacturer);
179         }
            $res->close();
181         $conn->close();
            return $products;
183     }

185     public function createProduct($productName, $manufacturer, $uid){
            $conn = $this->getConnection();
187         $res = $this->executeStatement($conn, 'INSERT INTO Product (Name, Manufacturer,
    created_by) VALUES (?, ?, ?)',
                function ($s) use ($productName, $manufacturer, $uid) {
189                 $s->bind_param('sss', $productName, $manufacturer, $uid);
                }
191         );
            $pid = $res->insert_id;
193         $res->close();
            $conn->close();
195         return $pid;
        }
197
        public function createRating($pid, $uid, $score, $comment){
```

```
199         $conn = $this->getConnection();
            $res = $this->executeStatement($conn, 'INSERT INTO Rating (created_by, created_for,
    Score, Comment) VALUES (?, ?, ?, ?)',
201             function ($s) use ($uid, $pid, $score, $comment) {
                    $s->bind_param('siis', $uid, $pid, $score, $comment);
203             }
            );
205         $res->close();
            $conn->close();
207         return $pid;
        }
209
        public function getRatingsByProductId($productID)
211     {
            $ratings = array();
213
            $conn = $this->getConnection();
215         $res = $this->executeStatement($conn, 'SELECT ID, created_by, created_for, Score,
    Comment FROM Rating WHERE created_for = ?',
                function ($s) use ($productID) {
217                 $s->bind_param('i', $productID);
                }
219         );
            $res->bind_result($ID, $created_by, $created_for, $Score, $Comment);
221         while ($res->fetch()) {
                $ratings[] = new Rating($ID, $created_for, $created_by, $Score, $Comment);
223         }
            $res->close();
225         $conn->close();
            return $ratings;
227     }

229     public function getRatingsByUser($userName)
        {
231         $ratings = array();

233         $conn = $this->getConnection();
            $res = $this->executeStatement($conn, 'SELECT ID, created_by, created_for, Score,
    Comment FROM Rating WHERE created_by = ?',
235             function ($s) use ($userName) {
                    $s->bind_param('s', $userName);
237             }
            );
239         $res->bind_result($ID, $created_by, $created_for, $Score, $Comment);
            while ($res->fetch()) {
241             $ratings[] = new Rating($ID, $created_for, $created_by, $Score, $Comment);
            }
243         $res->close();
            $conn->close();
245         return $ratings;
        }
247
        public function getAvarageRating($pid){
249         $s = 0;
            $conn = $this->getConnection();
251         $res = $this->executeStatement($conn, 'SELECT AVG(Score) FROM `Rating` WHERE
    created_for = ?',
                function ($s) use ($pid) {
253                 $s->bind_param('i', $pid);
                }
255         );
            $res->bind_result($score);
257         if ($res->fetch()) {
                $s = $score;
259         }
```

```
        $res->close();
        $conn->close();
        return $s;
    }

    public function getNumRatings($pid){
        $s = 0;
        $conn = $this->getConnection();
        $res = $this->executeStatement($conn, 'SELECT COUNT(Score) FROM `Rating` WHERE created_for = ?',
            function ($s) use ($pid) {
                $s->bind_param('i', $pid);
            }
        );
        $res->bind_result($cnt);
        if ($res->fetch()) {
            $s = $cnt;
        }
        $res->close();
        $conn->close();
        return $s;
    }

    public function deleteRating($rid){
        $conn = $this->getConnection();
        $res = $this->executeStatement($conn, 'DELETE FROM `Rating` WHERE ID = ?',
            function ($s) use ($rid) {
                $s->bind_param('i', $rid);
            }
        );
        $res->close();
        $conn->close();
    }

    public function deleteProduct($pid){
        $conn = $this->getConnection();
        $res = $this->executeStatement($conn, 'DELETE FROM `Product` WHERE ID = ?',
            function ($s) use ($pid) {
                $s->bind_param('i', $pid);
            }
        );
        $res->close();
        $conn->close();
    }

    public function updateProduct($pid, $productName, $manufacturer, $uid){
        $conn = $this->getConnection();
        $res = $this->executeStatement($conn, 'UPDATE `Product` SET Name = ?, Manufacturer = ? WHERE ID = ? AND created_by = ?',
            function ($s) use ($pid, $productName, $manufacturer, $uid) {
                $s->bind_param('ssis', $productName, $manufacturer, $pid, $uid);
            }
        );
        $res->close();
        $conn->close();
        return $pid;
    }

    public function getRatingById($rid){
        $ratings = array();

        $conn = $this->getConnection();
        $res = $this->executeStatement($conn, 'SELECT ID, created_by, created_for, Score, Comment FROM Rating WHERE ID = ?',
            function ($s) use ($rid) {
```

```
                $s->bind_param('i', $rid);
            }
        );
        $res->bind_result($ID, $created_by, $created_for, $Score, $Comment);
        while ($res->fetch()) {
            $ratings[] = new Rating($ID, $created_for, $created_by, $Score, $Comment);
        }
        $res->close();
        $conn->close();
        return $ratings[0];
    }

    public function updateRating($rid, $uid, $score, $comment){
        $conn = $this->getConnection();
        $res = $this->executeStatement($conn, 'UPDATE `Rating` SET Score = ?, Comment = ?
    WHERE ID = ? AND created_by = ?',
            function ($s) use ($score, $comment, $rid, $uid) {
                $s->bind_param('isis', $score, $comment, $rid, $uid);
            }
        );
        $res->close();
        $conn->close();
    }

}
```

../src/DataLayer/DBDataLayer.php

```
<?php
namespace Domain;

class Entity
{
    private $id;
    public function getId()
    {
        return $this->id;
    }
    public function __construct($id)
    {
        $this->id = $id;
    }
}
```

../src/Domain/Entity.php

```
<?php
namespace Domain;

class Product extends Entity
{
    private $productName;
    private $userID;
    private $manufacturer;

    public function __construct($id, $productName, $userID, $manufacturer)
    {
        parent::__construct($id);
        $this->productName = $productName;
        $this->userID = $userID;
        $this->manufacturer = $manufacturer;
    }

    public function getProductName()
```

```
19      {
            return $this->productName;
21      }

23      public function getUserID(){
            return $this->userID;
25      }

27      public function getManufacturer()
        {
29          return $this->manufacturer;
        }
31  }
```

../src/Domain/Product.php

```
1   <?php
    namespace Domain;
3
    class Rating extends Entity
5   {
        private $userID;
7       private $score;
        private $productId;
9       private $description;

11      public function __construct($id, $productID, $userID, $score, $description)
        {
13          parent::__construct($id);
            $this->productId = $productID;
15          $this->userID = $userID;
            $this->score = $score;
17          $this->description = $description;
        }
19
        public function getProductID()
21      {
            return $this->productId;
23      }

25      public function getUserID()
        {
27          return $this->userID;
        }
29
        public function getScore()
31      {
            return $this->score;
33      }

35      public function getDescription()
        {
37          return $this->description;
        }
39  }
```

../src/Domain/Rating.php

```
1   <?php
    namespace Domain;
3
    class User extends Entity
5   {
        private $userName;
```

```
7       private $password;
9       public function __construct($id, $userName, $password)
        {
11          parent::__construct($id);
            $this->userName = $userName;
13          $this->password = $password;
        }
15
        public function getUserName()
17      {
            return $this->userName;
19      }
21      public function getPassword()
        {
23          return $this->password;
        }
25  }
```

../src/Domain/User.php

```
1   <?php
    namespace Framework;
3
    class Controller
5   {
        final public function hasParam($id)
7       {
            return isset($_REQUEST[$id]);
9       }
11      final public function getParam($id, $defaultValue = null)
        {
13          return isset($_REQUEST[$id]) ? $_REQUEST[$id] : $defaultValue;
        }
15
        final public function renderView($view, $model = array())
17      {
            ViewRenderer::renderView($view, $model);
19      }
21      final public function redirectToUrl($url)
        {
23          header("Location: $url");
        }
25
        final public function redirect($action, $controller, $params = null)
27      {
            $this->redirectToUrl($this->buildActionLink($action, $controller, $params));
29      }
31      final public function buildActionLink($action, $controller, $params)
        {
33          return MVC::buildActionLink($action, $controller, $params);
        }
35  }
```

../src/Framework/Controller.php

```
1   <?php
    namespace Framework;
3
    final class Injector
```

```
{
    private function __construct()
    {}
    private static $instances;
    private static $singletonFlags;
    private static $classNames;
    private static $ctorParameters;
    public static function register($serviceName, $isSingleton = false, $className = null,
    $ctorParameters = null)
    {
        self::$singletonFlags[$serviceName] = $isSingleton;
        self::$classNames[$serviceName] = $className;
        self::$ctorParameters[$serviceName] = $ctorParameters;
    }
    public static function resolve($serviceName)
    {
        if (isset(self::$instances[$serviceName])) {
            return self::$instances[$serviceName];
        }
        $className = isset(self::$classNames[$serviceName]) && self::$classNames[
    $serviceName] != null ? self::$classNames[$serviceName] : $serviceName;
        $actualParams = array();
        $rClass = (new \ReflectionClass($className));
        if ($rClass == null) {
            die("Cannot find class '$className'.");
        }
        $rCtor = $rClass->getConstructor();
        if ($rCtor != null) {
            foreach ($rCtor->getParameters() as $rParam) {
                if (isset(self::$ctorParameters[$serviceName]) && isset(self::
    $ctorParameters[$serviceName][$rParam->getName()])) {
                    $actualParams[] = self::$ctorParameters[$serviceName][$rParam->getName()
    ];
                } else if ($rParam->isOptional()) {
                    $actualParams[] = $rParam->getDefaultValue();
                } else if ($rParam->getClass() != null) {
                    $actualParams[] = self::resolve($rParam->getClass()->name);
                } else {
                    die("Cannot resolve constructor parameter '{$rParam->getName()}' for
    class '$className'.");
                }
            }
        }
        $instance = new $className(...$actualParams);
        if (isset(self::$singletonFlags[$serviceName]) && self::$singletonFlags[$serviceName
    ] == true) {
            self::$instances[$serviceName] = $instance;
        }
        return $instance;
    }
}
```

../src/Framework/Injector.php

```
<?php
namespace Framework;

final class MVC
{
    private function __construct()
    {}

    const PARAM_CONTROLLER = 'c';
    const PARAM_ACTION = 'a';
    const DEFAULT_CONTROLLER = 'Home';
```

```
    const DEFAULT_ACTION = 'Index';
    const CONTROLLER_NAMESPACE = '\\Controllers';
    private static $viewPath = 'views';
    public static function getViewPath()
    {
        return self::$viewPath;
    }
    public static function buildActionLink($action, $controller, $params)
    {
        $res = '?' . self::PARAM_ACTION . '=' . rawurlencode($action) . '&' . self::
    PARAM_CONTROLLER . '=' . rawurlencode($controller);
        if (is_array($params)) {
            foreach ($params as $name => $value) {
                $res .= '&' . rawurlencode($name) . '=' . rawurlencode($value);
            }
        }
        return $res;
    }
    public static function handleRequest()
    {
        $controllerName = isset($_REQUEST[self::PARAM_CONTROLLER]) ? $_REQUEST[self::
    PARAM_CONTROLLER] : self::DEFAULT_CONTROLLER;
        $controller = self::CONTROLLER_NAMESPACE . "\\$controllerName";
        $method = $_SERVER['REQUEST_METHOD'];
        $action = isset($_REQUEST[self::PARAM_ACTION]) ? $_REQUEST[self::PARAM_ACTION] :
    self::DEFAULT_ACTION;
        $m = $method . '_' . $action;
        //(new $controller)->$m();
        \Framework\Injector::resolve($controller)->$m();
    }
}
```

../src/Framework/MVC.php

```
<?php
namespace Framework;

final class ViewRenderer
{
    private function __construct()
    {

    }
    public static function renderView($view, $model)
    {
        require MVC::getViewPath() . "/$view.inc";
    }
    private static function htmlOut($string)
    {
        echo (nl2br(htmlentities($string)));
    }
    private static function beginActionForm($action, $controller, $params = null, $method =
    'get', $cssClass = null)
    {
        $cc = $cssClass != null ? " class=\"$cssClass\"" : "";
        $form = <<<FORM
<form method="$method" action="?"$cc>
    <input type="hidden" name="c" value="$controller">
    <input type="hidden" name="a" value="$action">
FORM;
        echo ($form);
        if (is_array($params)) {
            foreach ($params as $name => $value) {
                $form = <<<FORM
<input type="hidden" name="$name" value="$value">
```

```
FORM;
                echo ($form);
            }
        }
    }
    private static function endActionForm()
    {
        echo ("</form>");
    }
    private static function actionLink($content, $action, $controller, $params = null,
    $cssClass = null)
    {
        $cc = $cssClass != null ? " class=\"$cssClass\"" : "";
        $url = MVC::buildActionLink($action, $controller, $params);
        $link = <<<LINK
<a href="$url"$cc>
LINK;
        echo ($link);
        echo ($content);
        echo ('</a>');
    }
}
```

../src/Framework/ViewRenderer.php

```
<?php self::renderView('partial/header', $model);?>
<div class="container">
<?php
if (isset($model['user'])) { ?>
    <?php self::actionLink("Create new product", 'Index', 'NewProduct'); ?>
    </br>
<?php } ?>
    <?php self::beginActionForm('Index', 'Home', null, 'post');?>
        <div class="form-row">
            <div class="col">
                <input class="form-control" id="filter" name="f" placeholder="your filter
    here...">
            </div>
            <div class="col">
                <button class="btn btn-primary">Apply</button>
            </div>
        </div>
    <?php self::endActionForm();?>

    <h1>Products</h1>
    <p>
        <table class="table">
            <tr>
                <th>Name</th>
                <th>Manufacturer</th>
                <th>Rating</th>
                <th># Rated</th>
                <th>created by</th>
            </tr>
            <?php foreach ($model['products'] as $product) {?>
            <tr>
                <td>
                    <?php self::actionLink($product[2]->getProductName(), 'Index', 'Product'
    , array('pid' => $product[2]->getId()));?>
                </td>
                <td>
                    <?php self::htmlout($product[2]->getManufacturer());?>
                </td>
                <td>
                    <?php self::htmlout($product[0]);?>
```

```
                </td>
                <td>
                    <?php self::htmlout($product[1]);?>
                </td>
                <td>
                    <?php self::htmlout($product[3]->getUserName());?>
                </td>
            </tr>
            <?php }?>
        </table>
    </p>
</div>
<?php self::renderView('partial/footer', $model);?>
```

../views/home.inc

```
<?php self::renderView('partial/header', $model); ?>
<div class="container">
    <h1>Login</h1>

    <?php self::beginActionForm('LogIn', 'User', null, 'post'); ?>
        <div class="form-group">
            <label for="userName">Username</label>
            <input class="form-control" id="userName" name="un" value="<?php self::htmlOut(
    $model['userName']); ?>">
        </div>
        <div class="form-group">
            <label for="password">Password</label>
            <input class="form-control" type="password" id="password" name="pwd">
        </div>
        <button class="btn btn-primary">Login</button>
    <?php self::endActionForm(); ?>
</div>
<?php self::renderView('partial/footer', $model); ?>
```

../views/login.inc

```
<?php self::renderView('partial/header', $model);?>
<div class="container">
    <?php if (!isset($model['product'])) {?>
    <h1>New Product</h1>
    <?php } else {?>
    <h1>Update Product</h1>
    <?php }?>

    <?php if (!isset($model['product'])){ ?>
    <?php self::beginActionForm('Create', 'NewProduct', null, 'post'); ?>
        <div class="form-group">
            <label for="productName">Product name</label>
            <input class="form-control" id="productName" name="pn">
        </div>
        <div class="form-group">
            <label for="manufacturer">Manufacturer</label>
            <input class="form-control" id="manufacturer" name="manu">
        </div>
        <button class="btn btn-primary">Create</button>
    <?php self::endActionForm();?>
    <?php } else { ?>
        <?php self::beginActionForm('Update', 'NewProduct', array('pid' => $model['product'
    ]->getId()), 'post'); ?>
        <div class="form-group">
            <label for="productName">Product name</label>
            <input class="form-control" id="productName" name="pn" value="<?php self::
    htmlout($model['product']->getProductName());?>">
```

```
        </div>
        <div class="form-group">
            <label for="manufacturer">Manufacturer</label>
            <input class="form-control" id="manufacturer" name="manu" value="<?php self::
    htmlout($model['product']->getManufacturer());?>">
        </div>
        <button class="btn btn-primary">Update</button>
    <?php self::endActionForm();?>
    <?php } ?>

</div>
<?php self::renderView('partial/footer', $model);?>
```

../views/newproduct.inc

```
<?php self::renderView('partial/header', $model);?>
<div class="container">
    <?php if(!isset($model['rating'])){ ?>
    <h1>New rating for <?php self::htmlout($model['product']->getProductName())?></h1>
    <?php } else { ?>
    <h1>Update rating for <?php self::htmlout($model['product']->getProductName())?></h1>
    <?php } ?>
    <?php if (!isset($model['rating'])) {?>
    <?php self::beginActionForm('Create', 'NewRating', array('pid' => $model['product']->
    getId()), 'post');?>
        <div class="form-group">
            <label for="comment">Comment</label>
            <input class="form-control" id="comment" name="cm">
        </div>
        <div class="radio">
            <label><input type="radio" name="rating" value="1">1</label>
        </div>
        <div class="radio">
            <label><input type="radio" name="rating" value="2">2</label>
        </div>
        <div class="radio">
            <label><input type="radio" name="rating" value="3" checked="checked">3</label>
        </div>
        <div class="radio">
            <label><input type="radio" name="rating" value="4">4</label>
        </div>
        <div class="radio">
            <label><input type="radio" name="rating" value="5">5</label>
        </div>
        <button class="btn btn-primary">Create</button>
    <?php self::endActionForm();?>
    <?php } else {?>
        <?php self::beginActionForm('Update', 'NewRating', array('pid' => $model['product'
    ]->getId(),'rid' => $model['rating']->getId()), 'post');?>
        <div class="form-group">
            <label for="comment">Comment</label>
            <input class="form-control" id="comment" name="cm" value="<?php self::htmlout(
    $model['rating']->getDescription());?>">
        </div>
        <div class="radio">
            <label><input type="radio" name="rating" value="1"  <?php if ($model['rating']->
    getScore() == 1) {?>checked="checked" <?php }?>>1</label>
        </div>
        <div class="radio">
            <label><input type="radio" name="rating" value="2"  <?php if ($model['rating']->
    getScore() == 2) {?>checked="checked" <?php }?>>2</label>
        </div>
        <div class="radio">
    <label><input type="radio" name="rating" value="3"  <?php if ($model['rating']->getScore
    () == 3) {?>checked="checked" <?php }?>>3</label>
```

```
        </div>
        <div class="radio">
            <label><input type="radio" name="rating" value="4" <?php if ($model['rating']->
    getScore() == 4) {?>checked="checked" <?php }?>>4</label>
        </div>
        <div class="radio">
            <label><input type="radio" name="rating" value="5" <?php if ($model['rating']->
    getScore() == 5) {?>checked="checked" <?php }?>>5</label>
        </div>
        <button class="btn btn-primary">Update</button>
    <?php self::endActionForm();?>

    <?php }?>
</div>
<?php self::renderView('partial/footer', $model);?>
```

../views/newrating.inc

```
<?php self::renderView('partial/header', $model);?>
<div class="container">
    <h1><?php echo ($model['product']->getProductName()) ?></h1>
    By <em><?php echo ($model['product']->getManufacturer()) ?></em></br>
    Avarage score: <?php self::htmlout($model['score'])?></br>
    <?php
if (isset($model['user'])) {
    self::actionLink('create rating for this product', 'Index', 'NewRating', array('pid' =>
    $model['product']->getId()));
}
?>
    <h2>Ratings</h2>
    <table class="table">
        <tr>
            <th>Score</th>
            <th>Comment</th>
            <th>rating by</th>
        </tr>
        <?php foreach ($model['ratings'] as $rating) {?>
        <tr>
            <td>
                <?php self::htmlout($rating->getScore())?>
            </td>
            <td>
                <?php self::htmlout($rating->getDescription())?>
            </td>
            <td>
                <?php self::htmlout($rating->getUserID()); ?>
            </td>
        </tr>
        <?php }?>
    </table>
</div>
<?php self::renderView('partial/footer', $model);?>
```

../views/product.inc

```
<?php self::renderView('partial/header', $model); ?>
<div class="container">
    <h1>Register</h1>

    <?php self::beginActionForm('Register', 'User', null, 'post'); ?>
        <div class="form-group">
            <label for="userName">Username</label>
            <input class="form-control" id="userName" name="un" value="<?php self::htmlOut(
    $model['userName']); ?>">
```

```
        </div>
        <div class="form-group">
            <label for="password">Password</label>
            <input class="form-control" type="password" id="password" name="pwd">
        </div>
        <button class="btn btn-primary">Login</button>
    <?php self::endActionForm(); ?>
</div>
<?php self::renderView('partial/footer', $model); ?>
```

../views/register.inc

```
<?php self::renderView('partial/header', $model);?>
<div class="container">
    <h1>Your products</h1>
    <p>
        <table class="table">
            <tr>
                <th>Name</th>
                <th>Manufacturer</th>
                <th>Action</th>
            </tr>
            <?php foreach ($model['products'] as $product) {?>
            <tr>
                <td>
                    <?php self::actionLink($product->getProductName(), 'Index', 'Product',
    array('pid' => $product->getId()));?>
                </td>
                <td>
                    <?php self::htmlout($product->getManufacturer()); ?>
                </td>
                <td>
                    <?php self::actionLink("delete", 'Delete', 'NewProduct', array('pid' =>
    $product->getId()));?>
                    <?php self::actionLink("update", 'Update', 'NewProduct', array('pid' =>
    $product->getId()));?>
                </td>
            </tr>
            <?php }?>
        </table>
    </p>

    <h1>Your ratings</h1>
    <p>
        <table class="table">
            <tr>
                <th>Product</th>
                <th>Comment</th>
                <th>Rating</th>
                <th>Action</th>
            </tr>
            <?php foreach ($model['ratings'] as $rating) {?>
            <tr>
                <td>
                    <?php self::actionLink($rating[0]->getProductName(), 'Index', 'Product',
     array('pid' => $rating[0]->getId()));?>
                </td>
                <td>
                    <?php self::htmlout($rating[1]->getDescription()); ?>
                </td>
                <td>
                    <?php self::htmlout($rating[1]->getScore()); ?>
                </td>
                <td>
```

```
                <?php self::actionLink("delete", 'Delete', 'NewRating', array('rid' =>
    $rating[1]->getId()));?>
                <?php self::actionLink("update", 'Update', 'NewRating', array('rid' =>
    $rating[1]->getId(), 'pid' => $rating[0]->getId()));?>
                </td>
            </tr>
            <?php }?>
        </table>
    </p>
</div>
<?php self::renderView('partial/footer', $model);?>
```

../views/userhome.inc

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

../views/partial/errors.inc

```
<footer>
            <div class="container">
                <p class="textmuted"><? strftime('%c'); ?></p>
            </div>
        </footer>
        <script src="js/jquery.min.js"></script>
        <script src="js/bootstrap.min.js"></script>
    </body>
</html>
```

../views/partial/footer.inc

```
<!DOCTYPE html>
<html>
    <head>
        <meta charset="utf-8">
        <title>Product rating portal</title>
        <link href="css/bootstrap.min.css" rel="stylesheet">
        <link href="css/bootstrap-theme.min.css" rel="stylesheet">
        <link href="css/ratingportal.css" rel="stylesheet">
    </head>
    <body>
        <nav class="navbar navbar-inverse navbar-fixed-top">
            <div class="container">
                <div class="navbar-header">
                    <button type="button" class="navbar-toggle collapsed" data-toggle="
    collapse" data-target="#navbar" aria-expanded="false" aria-controls="navbar">
                        <span class="sr-only">Toggle navigation</span>
                        <span class="icon-bar"></span>
                        <span class="icon-bar"></span>
                        <span class="icon-bar"></span>
                    </button>
                    <a class="navbar-brand" href="?">RATE</a>
                </div>
                <div id="navbar" class="collapse navbar-collapse">
                    <ul class="nav navbar-nav">
```

```
                        <?php if(isset($model['user'])) { ?>
                        <li><?php self::actionLink('Home', 'Index', 'UserHome'); ?></li>
                        <?php } ?>
                    </ul>
                    <div class="navbar-right">
                        <?php self::renderView('partial/user', $model['user']); ?>
                    </div>
                </div>
            </div>
        </nav>
        <?php if(isset($model['errors'])) {
            self::renderView('partial/errors', $model['errors']);
} ?>
```

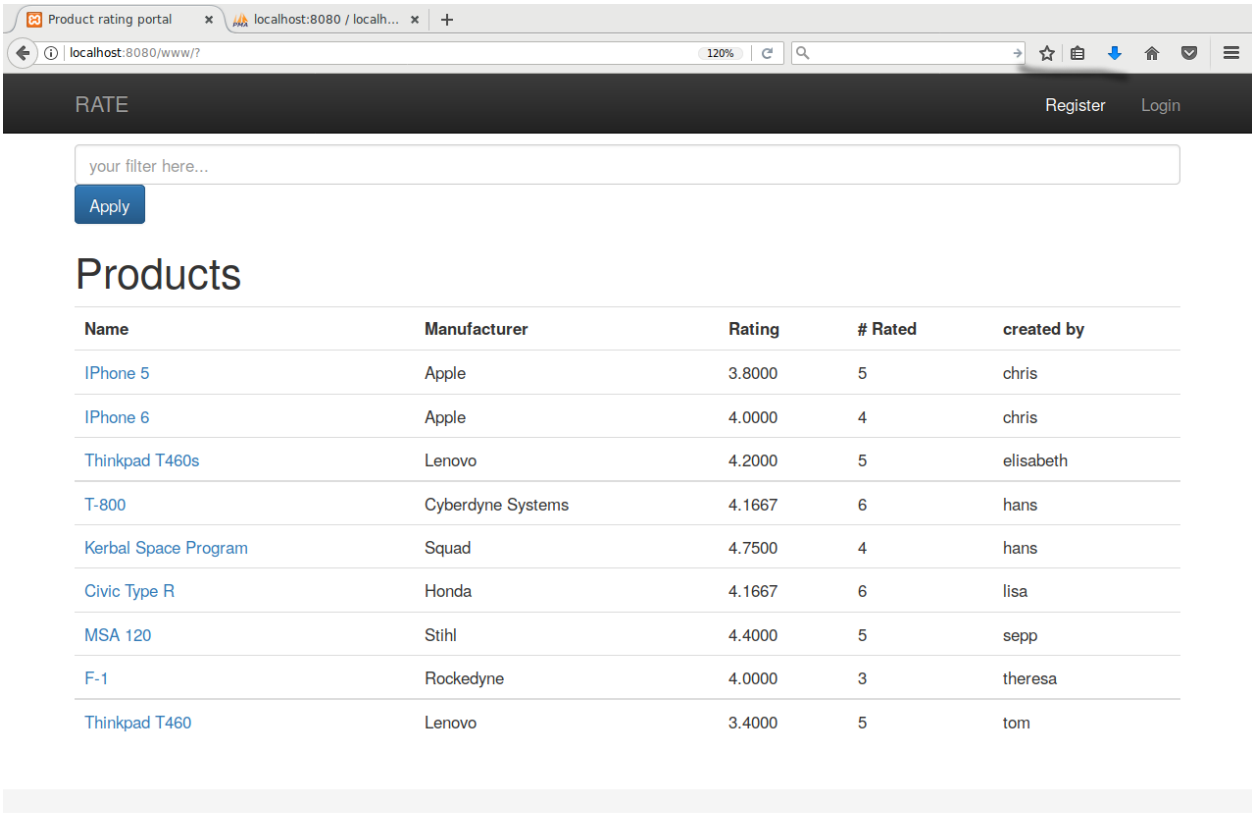
../views/partial/header.inc

```
<ul class="nav navbar-nav">
<?php if(!isset($model)): ?>
    <li><?php self::actionLink('Register', 'Register', 'User'); ?></li>
    <li><?php self::actionLink('Login', 'LogIn', 'User'); ?></li>
<?php else: ?>
    <li class="navbar-text">Welcome, <?php self::htmlOut($model->getUserName()); ?></li>
    <li>
        <?php self::beginActionForm('LogOut', 'User', null, 'post', 'navbar-form'); ?>
            <button class="btn btn-link">Log out</button>
        <?php self::endActionForm(); ?>
    </li>
<?php endif; ?>
</ul>
```

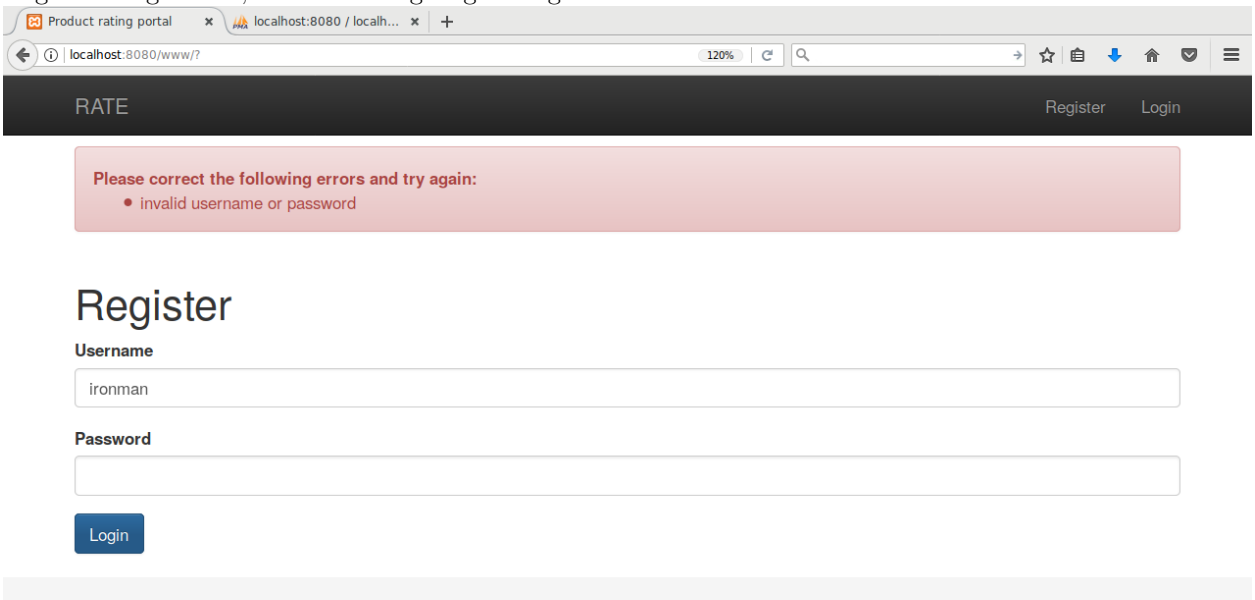
../views/partial/user.inc

1.4 Tests

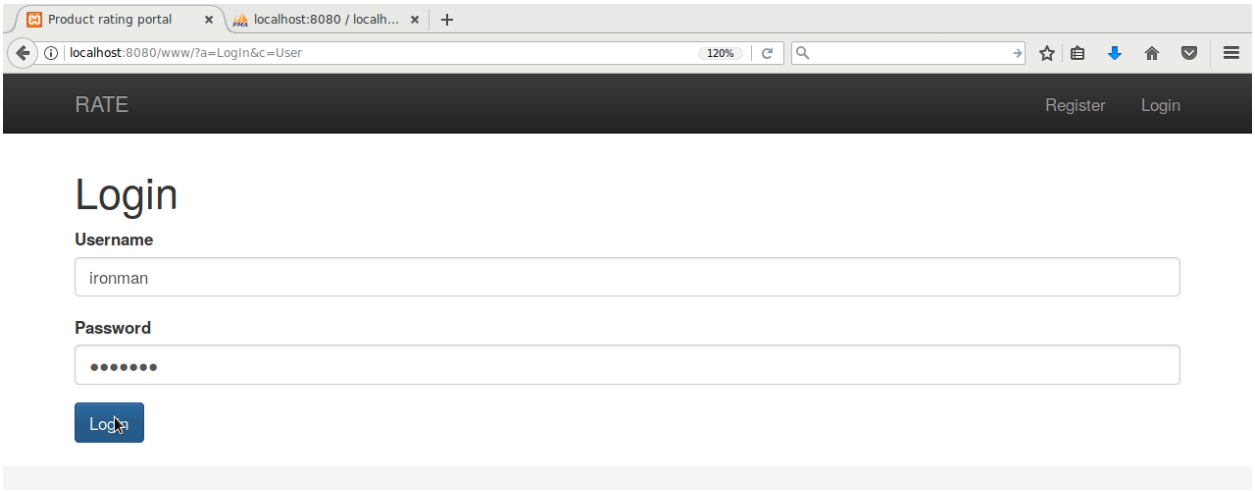
Übersichtsseite der Produkte, nicht eingeloggt.



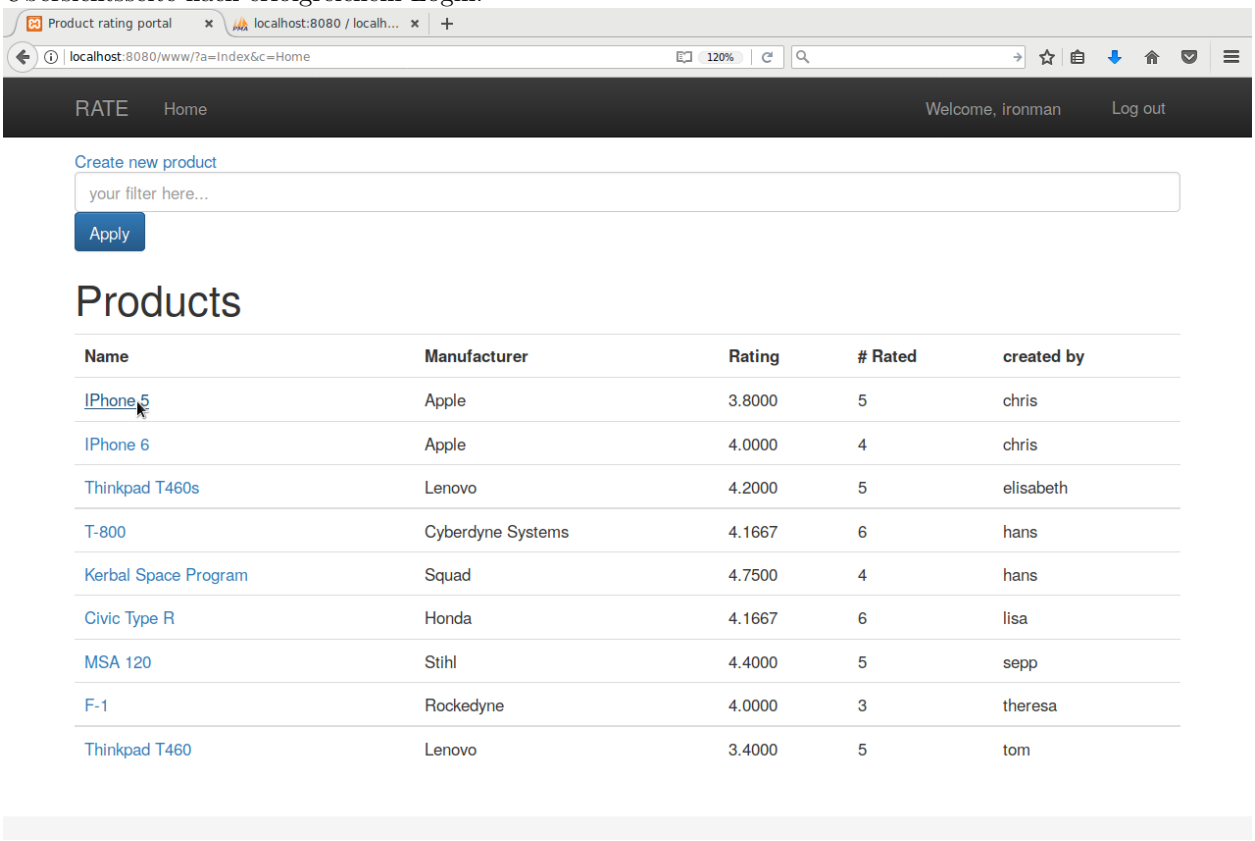
Registrierungsmaske; Fehler bei ungültiger Eingabe.



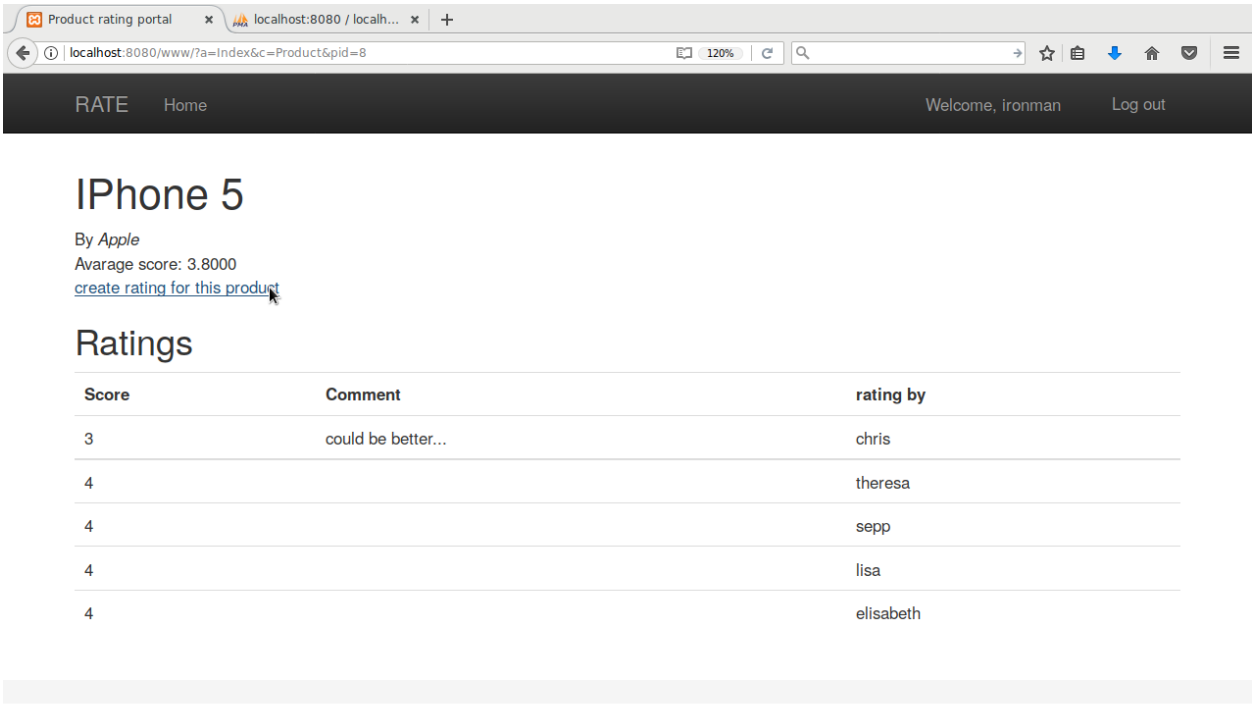
Nach erfolgreicher registrierung Weiterleitung auf Login-Maske.



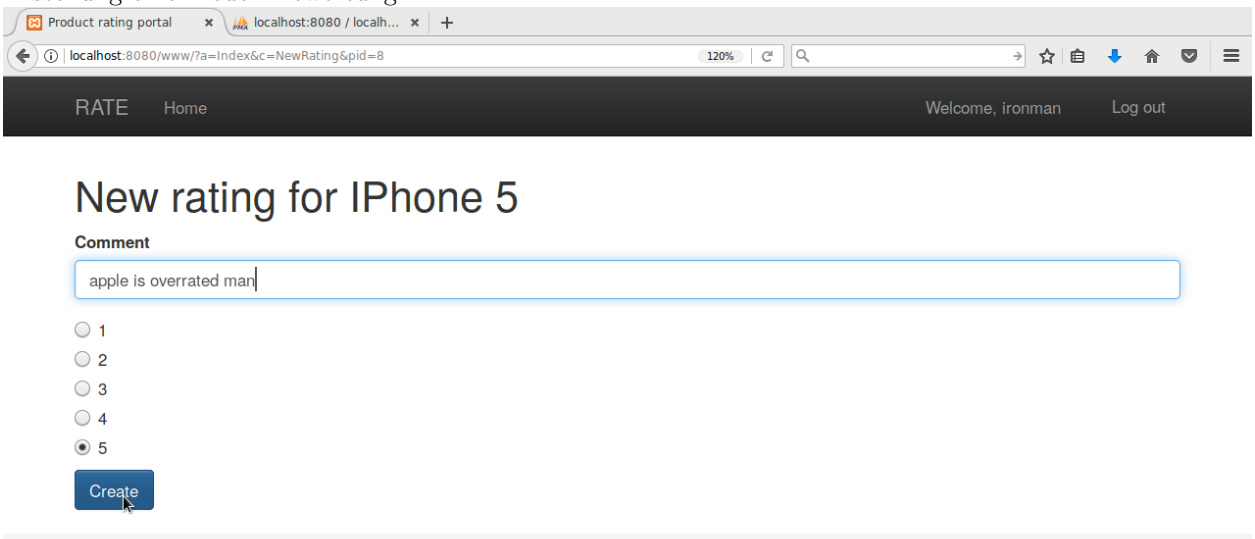
Übersichtsseite nach erfolgreichem Login.



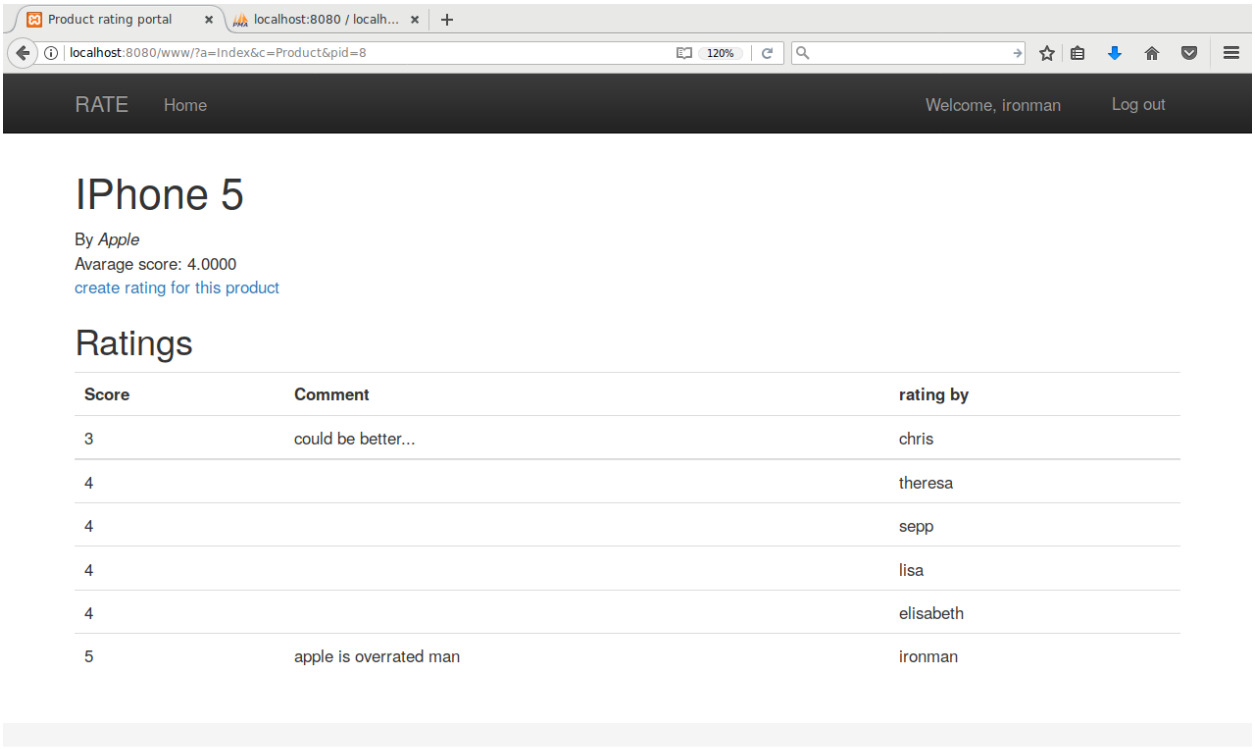
Detailansicht eines Produkts.



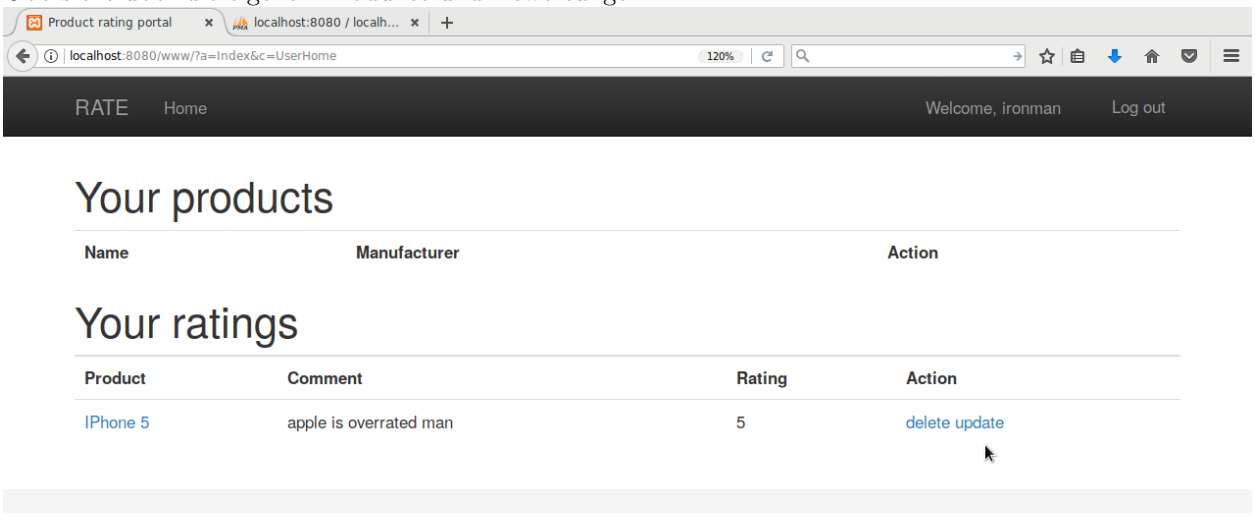
Erstellung einer neuen Bewertung.



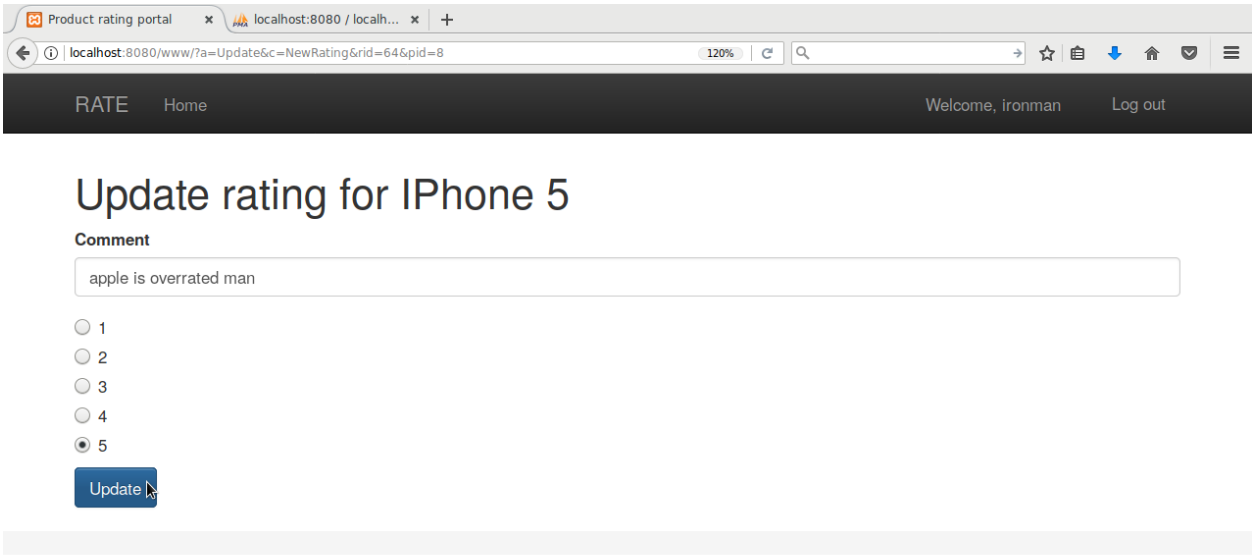
Detailansicht mit neuer Bewertung.



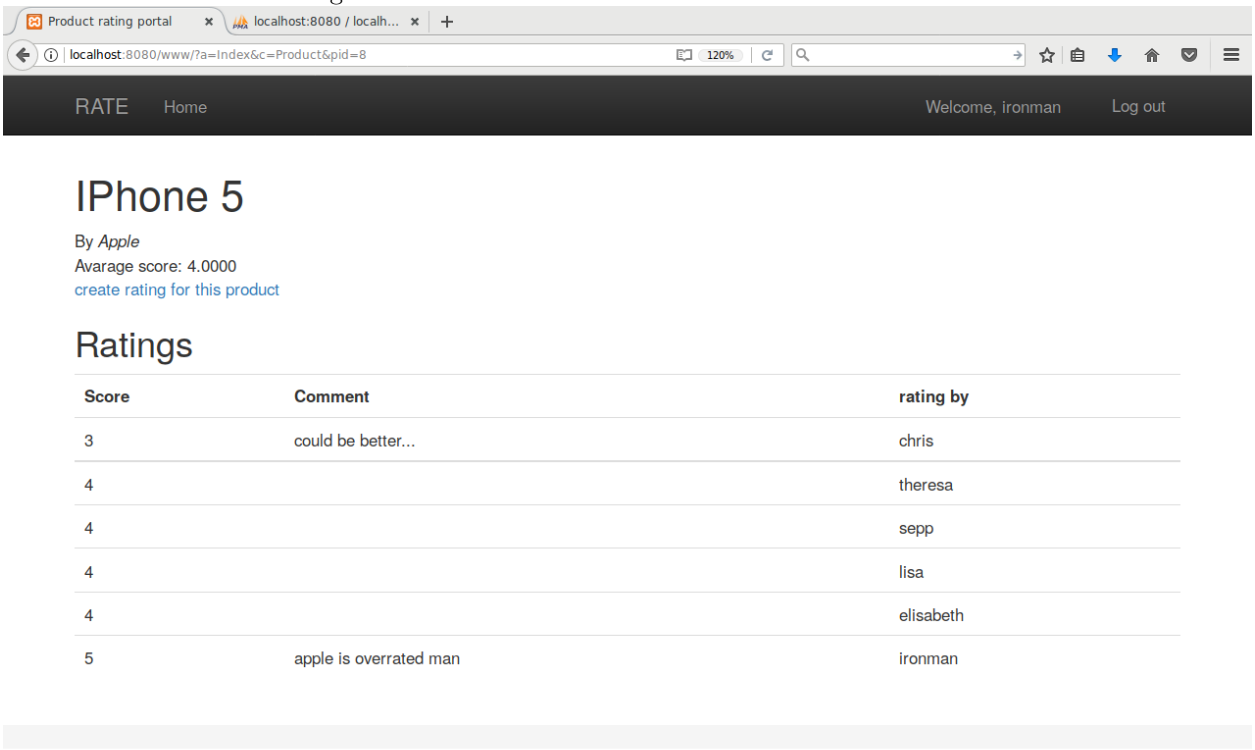
Übersicht über die eigenen Produkte und Bewertungen.



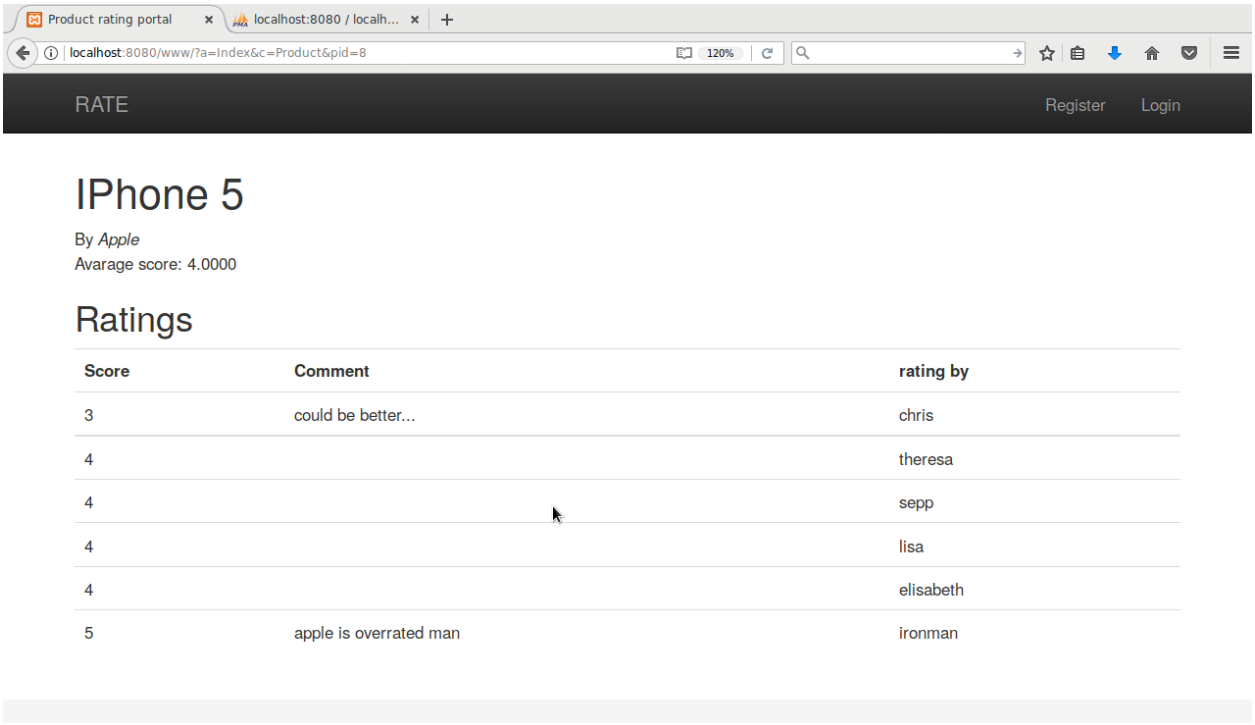
Änderung einer Bewertung.



Datailansicht nach Änderung.



Detailansicht im ausgeloggtem Zustand.


Product rating portal
localhost:8080 / localh...
localhost:8080/www/?a=Index&c=Product&pid=8
120%
RATE
Register
Login
IPhone 5
By Apple
Avarage score: 4.0000
Ratings
Score
Comment
rating by
3
could be better...
chris
4
theresa
4
sepp
4
lisa
4
elisabeth
5
apple is overrated man
ironman