Министерство образования и науки РФ

Федеральное государственное бюджетное образовательное учреждение высшего образования «Ивановский государственный энергетический университет имени В.И. Ленина»

Кафедра программного обеспечения компьютерных систем

Отчёт по лабораторной работе №6

Конструирование интернет-приложений

Разграничение прав

Выполнила студент гр. 3-42 \_\_\_\_\_\_\_\_\_\_\_\_\_\_ Шарабанов Н.А.

Проверил \_\_\_\_\_\_\_\_\_\_\_\_\_\_ Садыков А.М.

Иваново 2022

**Цель лабораторной работы:** добавить возможности разграничения прав пользователей

**Задания:**

1. Добавить создание предопределённых пользователей и ролей

2. Добавить роль для нового пользователя

3. Добавить разграничение прав по ролям

4. Добавить информацию о роли текущего пользователя

5. Добавить возврат 401 кода в авторизации

6. Настроить параметры неудачных входов

7. Добавить разграничение прав в клиентской части

Вывод

**Ход Работы**

1. **Добавлено создание предопределенных пользователей и ролей**

Было добавлено создание двух ролей и двух базовых пользователей системы при старте приложения.

В Data создан IdentitySeed.cs

using InternetShopWebApp.Models;

using Microsoft.AspNetCore.Identity;

namespace InternetShopWebApp.Data

{

public class IdentitySeed

{

public static async Task CreateUserRoles(IServiceProvider serviceProvider)

{

var roleManager = serviceProvider.GetRequiredService<RoleManager<IdentityRole>>();

var userManager = serviceProvider.GetRequiredService<UserManager<User>>();

// Создание ролей администратора и пользователя

if (await roleManager.FindByNameAsync("admin") == null)

{

await roleManager.CreateAsync(new IdentityRole("admin"));

}

if (await roleManager.FindByNameAsync("user") == null)

{

await roleManager.CreateAsync(new IdentityRole("user"));

}

// Создание Администратора

string adminEmail = "admin@mail.com";

string adminPassword = "Aa123456!";

if (await userManager.FindByNameAsync(adminEmail) == null)

{

User admin = new User { Email = adminEmail, UserName = adminEmail };

IdentityResult result = await userManager.CreateAsync(admin, adminPassword);

if (result.Succeeded)

{

await userManager.AddToRoleAsync(admin, "admin");

}

}

// Создание Пользователя

string userEmail = "user@mail.com";

string userPassword = "Aa123456!";

if (await userManager.FindByNameAsync(userEmail) == null)

{

User user = new User { Email = userEmail, UserName = userEmail };

IdentityResult result = await userManager.CreateAsync(user, userPassword);

if (result.Succeeded)

{

await userManager.AddToRoleAsync(user, "user");

}

}

}

}

}

В program.cs были внесены следующие изменения.

using (var scope = app.Services.CreateScope())

{

var shopInternetContext =

scope.ServiceProvider.GetRequiredService<ShopContext>();

await ShopContextSeed.SeedAsync(shopInternetContext);

await IdentitySeed.CreateUserRoles(scope.ServiceProvider);

}

1. **Добавлена роль для нового пользователя**

Была добавлена строка в AccountController.cs.

[AllowAnonymous]

public async Task<IActionResult> Register([FromBody] RegisterViewModel model)

{

if (ModelState.IsValid)

{

User user = new() { Email = model.Email, UserName = model.Email };

// Добавление нового пользователя

var result = await \_userManager.CreateAsync(user, model.Password);

if (result.Succeeded)

{

// Установка роли User

await \_userManager.AddToRoleAsync(user, "user");

// Установка куки

await \_signInManager.SignInAsync(user, false);

return Ok(new { message = "Добавлен новый пользователь: " + user.UserName });

}

else

{

foreach (var error in result.Errors)

{

ModelState.AddModelError(string.Empty, error.Description);

}

var errorMsg = new

{

message = "Пользователь не добавлен",

error = ModelState.Values.SelectMany(e => e.Errors.Select(er => er.ErrorMessage))

};

return Created("", errorMsg);

}

}

else

{

var errorMsg = new

{

message = "Неверные входные данные",

error = ModelState.Values.SelectMany(e => e.Errors.Select(er => er.ErrorMessage))

};

return Created("", errorMsg);

}

}

1. **Добавлено разграничение прав по ролям**

Для ограничения доступа к методам был добавлен атрибут Authorize во всех контроллеры.

В моей системе две роли: пользователь и администратор. В зависимости от этого, есть ограничения в функционале для каждого:

1. Пользователь может добавлять и удалять товар в корзину.

2. Администратор может просматривать товары в корзине пользователя

Пересечений в функционале нет.

// DELETE: api/OrderItem/5

[HttpDelete("{id}")]

[Authorize(Roles = "admin")]

public async Task<IActionResult> DeleteOrderItem(int id)

{

var blog = await \_context.OrderItem.FindAsync(id);

if (blog == null)

{

return NotFound();

}

\_context.OrderItem.Remove(blog);

await \_context.SaveChangesAsync();

return NoContent();

}

1. **Добавить информацию о роли текущего пользователя**

Для получения информации о роли пользователя были внесена следующие изменения в AccountController:

[HttpPost]

[Route("api/account/login")]

//[AllowAnonymous]

public async Task<IActionResult> Login([FromBody] LoginViewModel model)

{

if (ModelState.IsValid)

{

var result =

await \_signInManager.PasswordSignInAsync(model.Email, model.Password, model.RememberMe, false);

if (result.Succeeded)

{

User usr = await GetCurrentUserAsync();

if (usr == null)

{

return Unauthorized(new { message = "Ошибка выполнения авторизации" });

}

IList<string> roles = await \_userManager.GetRolesAsync(usr);

string? userRole = roles.FirstOrDefault();

return Ok(new { message = "Выполнен вход", userName = model.Email, userRole });

}

else

{

ModelState.AddModelError("", "Неправильный логин и (или) пароль");

var errorMsg = new

{

message = "Вход не выполнен",

error = ModelState.Values.SelectMany(e => e.Errors.Select(er => er.ErrorMessage))

};

return Created("", errorMsg);

}

}

else

{

var errorMsg = new

{

message = "Вход не выполнен",

error = ModelState.Values.SelectMany(e => e.Errors.Select(er => er.ErrorMessage))

};

return Created("", errorMsg);

}

}

[HttpGet]

[Route("api/account/isauthenticated")]

public async Task<IActionResult> IsAuthenticated()

{

User usr = await GetCurrentUserAsync();

if (usr == null)

{

return Unauthorized(new { message = "Вы Гость. Пожалуйста, выполните вход" });

}

IList<string> roles = await \_userManager.GetRolesAsync(usr);

string? userRole = roles.FirstOrDefault();

return Ok(new { message = "Сессия активна", userName = usr.UserName, userRole });

}

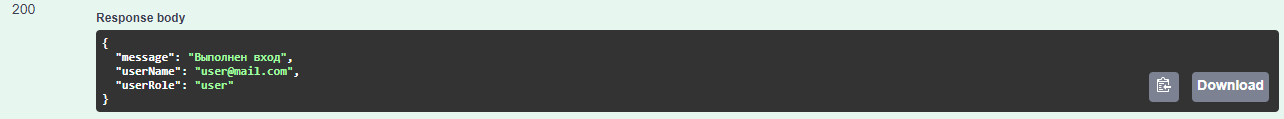


Рисунок 1 – Результат добавления вывода роил пользователя

1. **Добавлен возврат 401 кода в авторизации**

В Program.cs был добавлен следующий код:

builder.Services.ConfigureApplicationCookie(options =>

{

options.Cookie.Name = "BloggingApp";

options.LoginPath = "/";

options.AccessDeniedPath = "/";

options.LogoutPath = "/";

options.Events.OnRedirectToLogin = context =>

{

context.Response.StatusCode = 401;

return Task.CompletedTask;

};

// 401

options.Events.OnRedirectToAccessDenied = context =>

{

context.Response.StatusCode = 401;

return Task.CompletedTask;

};

});

1. **Настроены параметры неудачных входов**

В Program.cs был добавлен следующий код:

builder.Services.Configure<IdentityOptions>(options =>

{

options.Lockout.DefaultLockoutTimeSpan = TimeSpan.FromMinutes(5);

options.Lockout.MaxFailedAccessAttempts = 5;

options.Lockout.AllowedForNewUsers = true;

});

1. **Добавлено разграничение прав в клиентской части**

Было изменено в index.js:

const [user, setUser] = useState({isAuthenticated: false, userName: '', userRole: ''});

Внесены изменения в Register, LogIn, LoggOut:

import React, {useState} from 'react';

import {useNavigate} from 'react-router-dom';

const LogIn = ({user, setUser}) => {

  const [errorMessages, setErrorMessages] = useState([]);

  const navigate = useNavigate();

  const logIn = async (event) => {

    event.preventDefault();

    const {email, password} = document.forms[0];

    // console.log(email.value, password.value)

    const requestOptions = {

      method: 'POST',

      headers: {'Content-Type': 'application/json'},

      body: JSON.stringify({

        email: email.value,

        password: password.value,

        passwordConfirm: password.value,

      }),

    };

    return await fetch(

        "api/account/login",

        requestOptions,

    )

        .then((response) => {

        // console.log(response.status)

          response.status === 200 &&

          setUser({isAuthenticated: true, userName: '', userRole: ''});

          return response.json();

        })

        .then(

            (data) => {

              console.log('Data:', data);

              if (

                typeof data !== 'undefined' &&

            typeof data.userName !== 'undefined'&&

            typeof data.userRole !== 'undefined'

              ) {

                // eslint-disable-next-line max-len

                setUser({isAuthenticated: true, userName: data.userName, userRole: data.userRole});

                navigate('/');

              }

              typeof data !== 'undefined' &&

            typeof data.error !== 'undefined' &&

            setErrorMessages(data.error);

            },

            (error) => {

              console.log(error);

            },

        );

  };

  const renderErrorMessage = () =>

    errorMessages.map((error, index) => <div key={index}>{error}</div>);

  return (

    <>

      {user.isAuthenticated ? (

        <h3>Пользователь {user.userName} с ролью {user.userRole} успешно вошел в систему</h3>

      ) : (

        <>

          <h3>Вход</h3>

          <form onSubmit={logIn}>

            <label>Пользователь </label>

            <input type="text" name="email" placeholder="Логин" />

            <br />

            <label>Пароль </label>

            <input type="text" name="password" placeholder="Пароль" />

            <br />

            <button type="submit">Войти</button>

          </form>

          {renderErrorMessage()}

        </>

      )}

    </>

  );

};

export default LogIn;

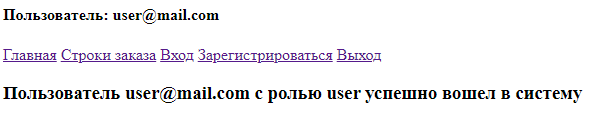


Рисунок 2 – Результат авторизации под аккаунтом пользователя

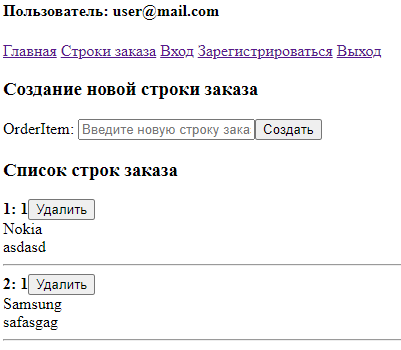


Рисунок 3 – Результат разделения ролей

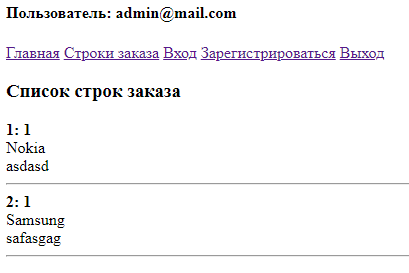


Рисунок 4 – Результат разделения ролей

**Вывод**

В ходе лабораторной работы я добавил возможность разграничения прав пользователей.