САНКТ-ПЕТЕРБУРГСКИЙ НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ ИТМО

Дисциплина: Фронт-энд разработка

Отчет

Лабораторная работа №1

Выполнил:

Габов Михаил

K3440

Проверил:
Добряков Д. И.

Санкт-Петербург

2023 г.

Вариант: Сервис для обмена рецептами и кулинарных блогов

Требования:

1. Вход
2. Регистрация
3. Личный кабинет пользователя (сохраненные рецепты, публикации)
4. Поиск рецептов с фильтрацией по типу блюда, сложности, ингредиентам
5. Страница рецепта с фото, пошаговыми инструкциями и видео
6. Социальные функции (комментарии, лайки, подписки на кулинаров)

Каждый из сайтов обязательно должен включать в себя следующие страницы:

- Вход
- Регистрация
- Личный кабинет пользователя
- Страница для поиска с возможностью фильтрации

бэкенд взят из курса по бэкэнд разработке того же варианта

1 Вход

```
<main class="container">
    <div class="auth-container">
        <div class="auth-card">
            <h2 class="text-center mb-4">Login</h2>
            <form id="login-form">
                <div class="mb-3">
                    <label for="email" class="form-label">Email</label>
                    <input type="email" class="form-control" id="email" required>
                </div>
                <div class="mb-4">
                    <label for="password" class="form-label">Password</label>
                    <input type="password" class="form-control" id="password" required>
                </div>
                <div id="error-message" class="alert alert-danger d-none"></div>
                <button type="submit" class="btn btn-success w-100" id="submit-btn">Login</button>
            </form>
            <p class="text-center mt-4 mb-0 text-muted">
                Don't have an account? <a href="/register.html">Register</a>
            </p>
        </div>
    </div>
</main>
```

используется bootstrap класс container и отступы

Для полей используется form-control

```
<div class="profile-card mb-4">
    <h4 class="mb-3">Profile Info</h4>
    <form id="profile-form">
        <div class="mb-3">
            <label for="username" class="form-label">Username</label>
            <input type="text" class="form-control" id="username" required minlength="3">
        </div>
        <div class="mb-3">
            <label for="email" class="form-label">Email</label>
            <input type="email" class="form-control" id="email" required>
        </div>
        <div class="mb-3">
            <label for="bio" class="form-label">Bio</label>
            <textarea class="form-control" id="bio" rows="3" placeholder="Tell something about
yourself..."></textarea>
        </div>
        <div class="mb-3">
            <label for="avatarUrl" class="form-label">Avatar URL</label>
            <input type="url" class="form-control" id="avatarUrl"
placeholder="https://example.com/avatar.jpg">
        </div>
        <button type="submit" class="btn btn-success" id="save-profile-btn">Save Changes</button>
    </form>
</div>
```

но стили были немного переопределены

```
.form-control,
.form-select {
    border: 2px solid var(--border);
    border-radius: var(--radius-sm);
    padding: 0.75rem 1rem;
    font-size: 0.95rem;
    transition: var(--transition);
    background: var(--card-bg);
}

.form-control:focus,
.form-select:focus {
    border-color: var(--accent);
    box-shadow: 0 0 0 4px rgba(0, 184, 148, 0.1);
}
```

если пользователь уже залогинен, то скрипт направляет его дальше (после загрузки DOM)

```
document.addEventListener('DOMContentLoaded', () => {
    renderNavbar();

    if (auth.isAuthenticated()) {
        window.location.href = '/';
        return;
    }
```

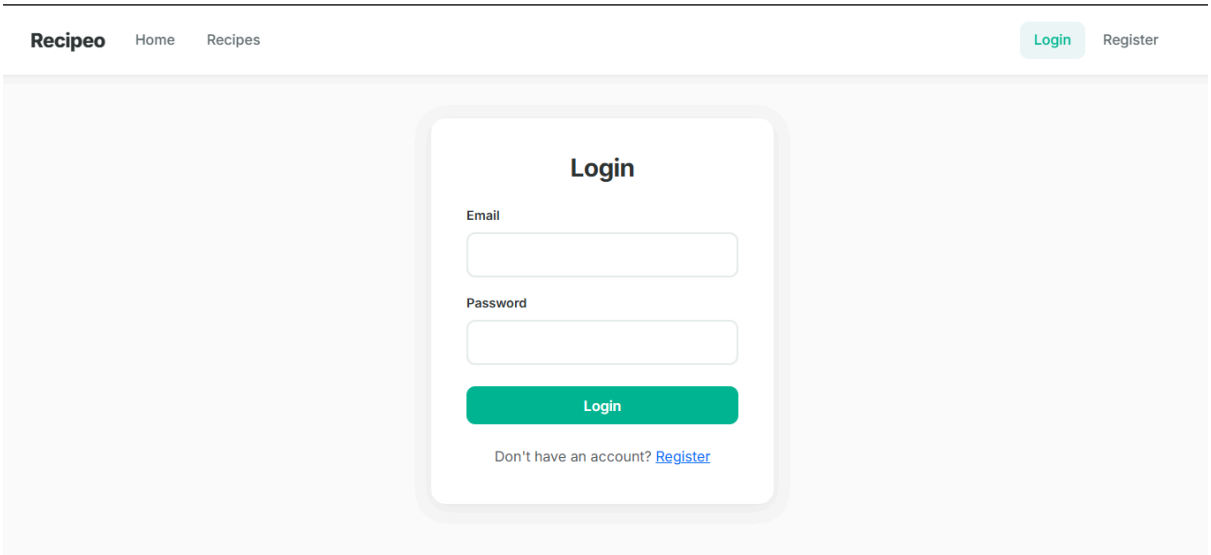
EventListener отслеживает отправку запроса на логин, собирает введенные данные и отправляет в authApi

```
    const form = document.getElementById('login-form');
    const errorMessage = document.getElementById('error-message');
    const submitBtn = document.getElementById('submit-btn');

    form.addEventListener('submit', async (e) => {
        e.preventDefault();
        errorMessage.classList.add('d-none');

        const email = document.getElementById('email').value.trim();
        const password = document.getElementById('password').value;

        if (!email || !password) {
            showError('Please fill in all fields');
            return;
        }

        submitBtn.disabled = true;
        submitBtn.textContent = 'Logging in...';

        try {
            const response = await authApi.login(email, password);
            auth.setToken(response.token);
            window.location.href = '/';
        } catch (error) {
            showError(error.message || 'Login failed. Please check your credentials.');
        } finally {
            submitBtn.disabled = false;
            submitBtn.textContent = 'Login';
        }
    });
```

authApi

```
export const authApi = {
    async register(username, email, password) {
        return await http.post('/users', { username, email, password });
    },

    async login(email, password) {
        return await http.post('/users/login', { email, password });
    },

    async getUserById(userId) {
        return await http.get(`/users/search/by?id=${userId}`);
    }
};
```

Страница логина



2 Регистрация

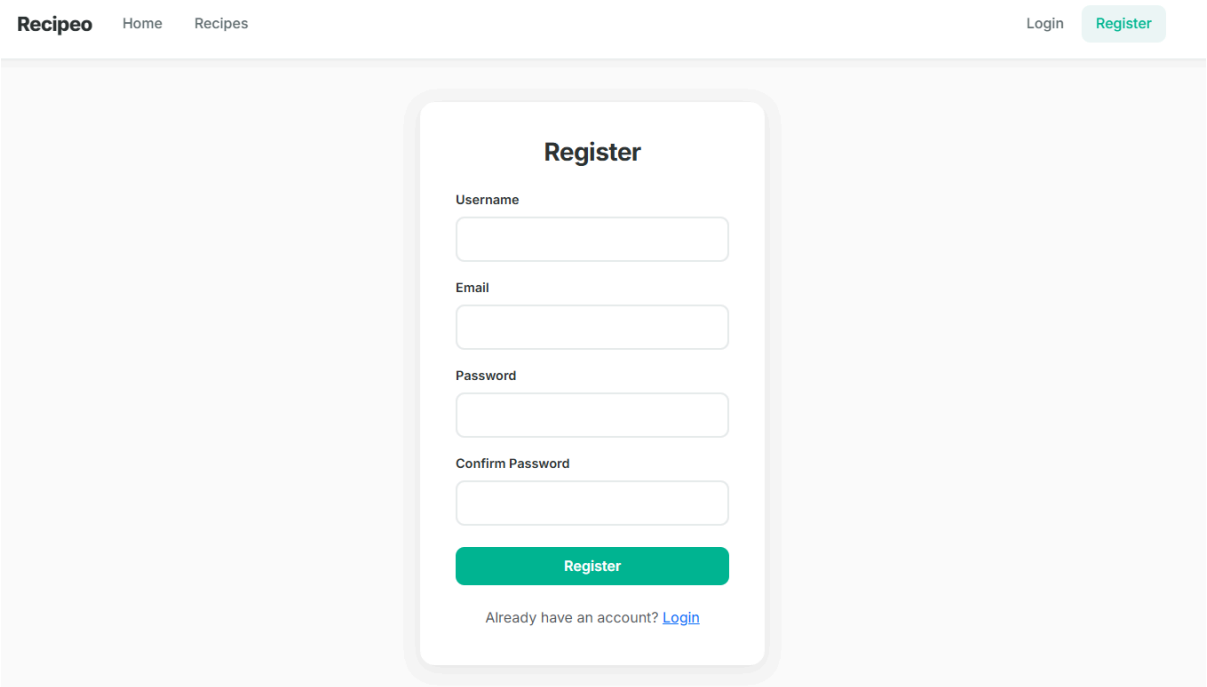
Используется аналогичный bootstrap контейнер

```
<main class="container">
    <div class="auth-container">
        <div class="auth-card">
            <h2 class="text-center mb-4">Register</h2>
            <form id="register-form">
                <div class="mb-3">
                    <label for="username" class="form-label">Username</label>
                    <input type="text" class="form-control" id="username" required minlength="3">
                </div>
                <div class="mb-3">
                    <label for="email" class="form-label">Email</label>
                    <input type="email" class="form-control" id="email" required>
                </div>
                <div class="mb-3">
                    <label for="password" class="form-label">Password</label>
                    <input type="password" class="form-control" id="password" required minlength="6">
                </div>
                <div class="mb-4">
                    <label for="confirm-password" class="form-label">Confirm Password</label>
                    <input type="password" class="form-control" id="confirm-password" required>
                </div>
                <div id="error-message" class="alert alert-danger d-none"></div>
                <button type="submit" class="btn btn-success w-100" id="submit-btn">Register</button>
            </form>
            <p class="text-center mt-4 mb-0 text-muted">
                Already have an account? <a href="/login.html">Login</a>
            </p>
        </div>
    </div>
</main>
```

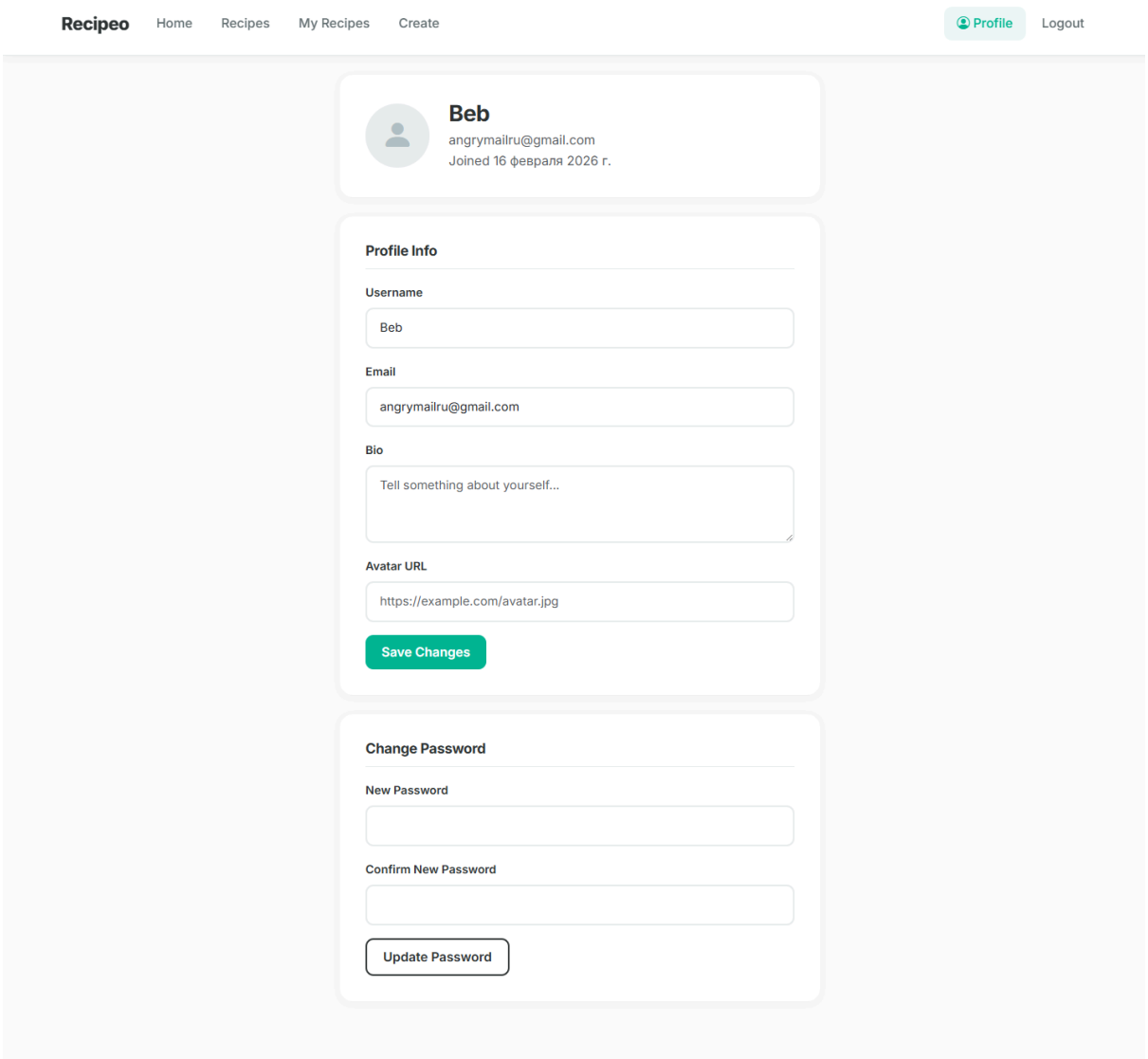
Аналогично логину, в скрипте есть EventListener на submit, но дополнительно присутствует валидация

```
        if (!username || !email || !password || !confirmPassword) {
            showError('Please fill in all fields');
            return;
        }

        if (username.length < 3) {
            showError('Username must be at least 3 characters');
            return;
        }

        if (!isValidEmail(email)) {
            showError('Please enter a valid email address');
            return;
        }

        if (password.length < 6) {
            showError('Password must be at least 6 characters');
            return;
        }

        if (password !== confirmPassword) {
            showError('Passwords do not match');
            return;
        }
```

страница регистрации



3 Личный кабинет



В личном кабинете можно изменить

```
{
  "username": "string",
  "email": "string",
  "password": "string",
  "bio": "string",
  "avatarUrl": "string"
}
```

если у пользователя нет картинки, используется bootstrap иконка

```
            if (updated.avatarUrl) {
                profileAvatar.innerHTML = `<img src="${updated.avatarUrl}" alt="Avatar">`;
            } else {
                profileAvatar.innerHTML = '<i class="bi bi-person-fill"></i>';
            }
```

К личному кабинету по заданию также относится страница с сохраненными рецептами и публикациями. Такой функционал присутствует, но он был вынесен на отдельную страницу — my recipes

рецепты загружаются сюда

```
<div id="recipes-container" class="row g-4">
    <!-- Recipes will be loaded here -->
</div>
```

сначала подтягиваются избранные рецепты пользователя

```
const favorites = await interactionsApi.getFavoritesByUser();
```

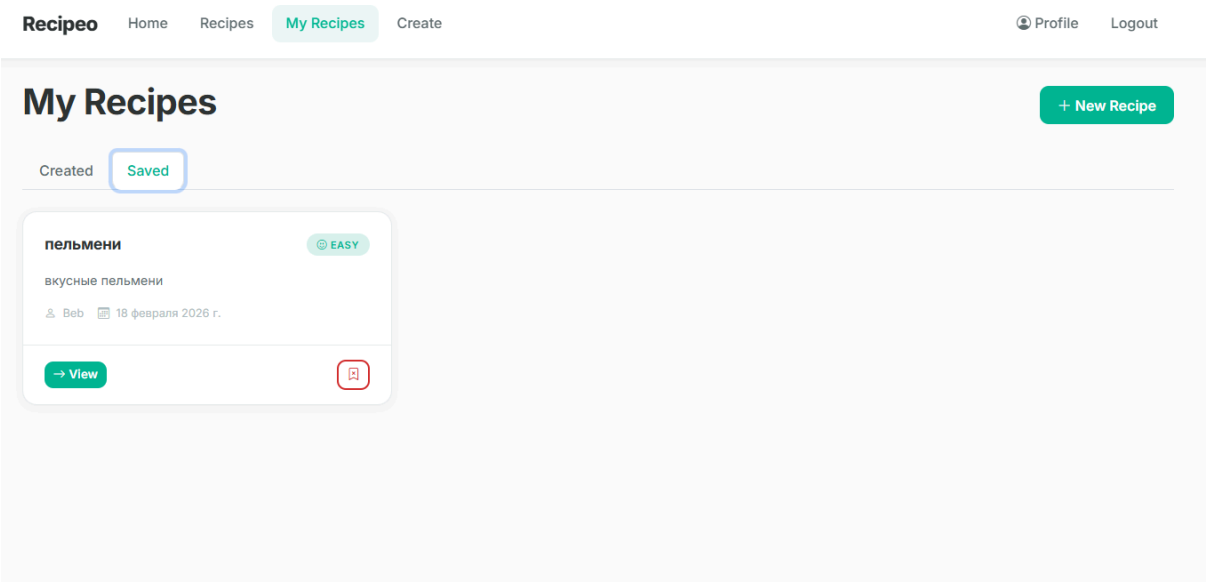
затем загружаются данные о рецепте и удаляются пустые значения

```
        const recipes = await Promise.all(
            favorites.map(fav => recipesApi.getRecipeById(fav.recipeId).catch(() => null))
        );
        const validRecipes = recipes.filter(r => r !== null);
```

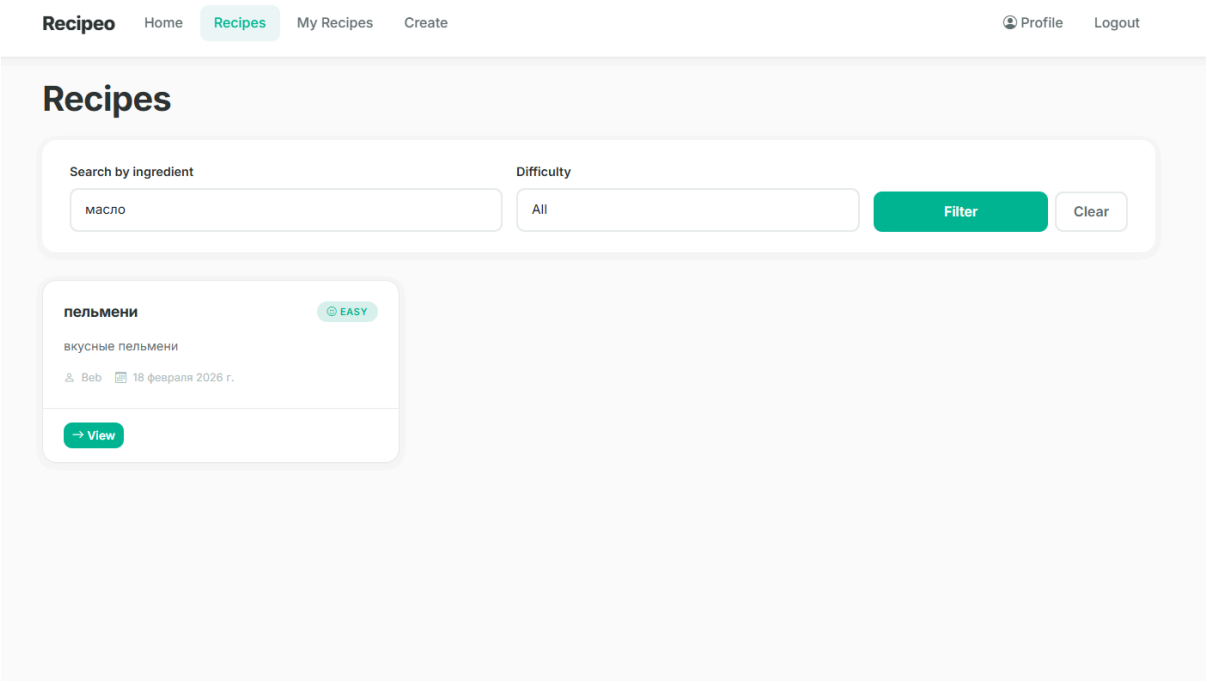
для валидных рецептов создаются контейнеры с карточками (createRecipeCard) для панели избранного

```
        container.innerHTML = '';
        validRecipes.forEach(recipe => {
            const card = createRecipeCard(recipe, {
                showRemoveFavorite: true,
                onRemoveFavorite: async (id) => {
                    try {
                        await interactionsApi.removeFromFavorites(id);
                        ui.showSuccess('Removed from saved');
                        await loadSavedRecipes();
                    } catch (err) {
                        ui.showError('Failed to remove from saved');
                    }
                }
            });
            container.appendChild(card);
        });
```

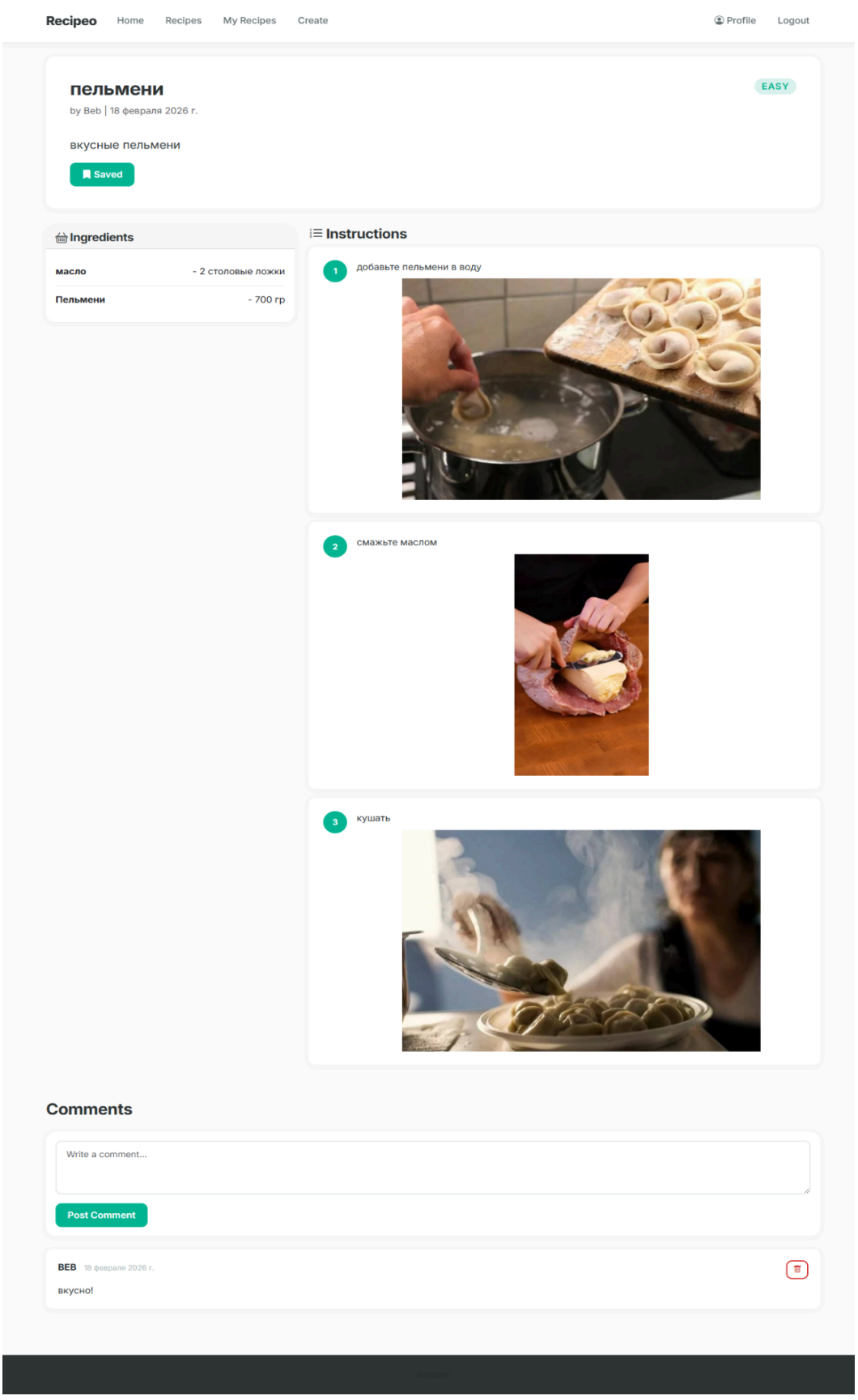
так выглядит панель избранного



фильтрация



4 Карточка рецепта



Тут же есть и система комментариев