УДК 004

Создание сервиса доставки еды на языке JavaScript

Халиманенков Андрей Сергеевич
Приамурский государственный университет имени Шолом-Алейхема
Студент

Аннотация
В данной статье рассматриваются разработка сервиса доставки еды на языке JavaScript для браузеров, в котором используется работа с RESTAPI при помощи AJAX запросов для получения ресторанов и их меню.
Ключевые слова: онлайн магазин, доставка еды, JavaScript, HTML, CSS, RESTAPI, веб-приложение.

Creating a food delivery service in JavaScript

Khalimanenkov Andrey Sergeevich
Sholom-Aleichem Priamursky State University
Student

Abstract
This article discusses the development of a food delivery service in JavaScript for browsers, which uses RESTAPI using AJAX requests to get restaurants and their menus.
Keywords: online store, food delivery, JavaScript, HTML, CSS, RESTAPI, web application.

Сервисы доставки еды получили огромный прирост активной базы покупателей после начале пандемии. Спрос на разработку новых и поддержку существующих сайтов вырос. Поэтому очень важным конкурентным преимуществом будет опыт создания подобных сайтов. «Лицо» магазина — это его интерфейс, то есть frontend часть, которая и будет затронута в статье.

А. В. Ермолик и А. П. Багаева рассмотрели этапы создания интернет-магазина и его преимуществ перед физическими точками [1]. Т. Г. Фомина и А. С. Соколова дали определение понятию «интернет-магазин», авторы предлагают рассматривать данное понятие не только с позиции маркетинговых коммуникаций, но и с технологической [2]. З. В. Галимзянов рассмотрел подход к созданию интернет-магазина [3].

В этой статье будет рассмотрено создание такой площадки на чистом JavaScript [4], HTML [5] и CSS [6] без использования серьёзных библиотек для frontend разработки, таких как React, Vue или Angular.

Главная страница выглядит следующим образом (рис. 1) – сверху находится кнопка входа в аккаунт, слайдер с акциями, а ниже список ресторанов.

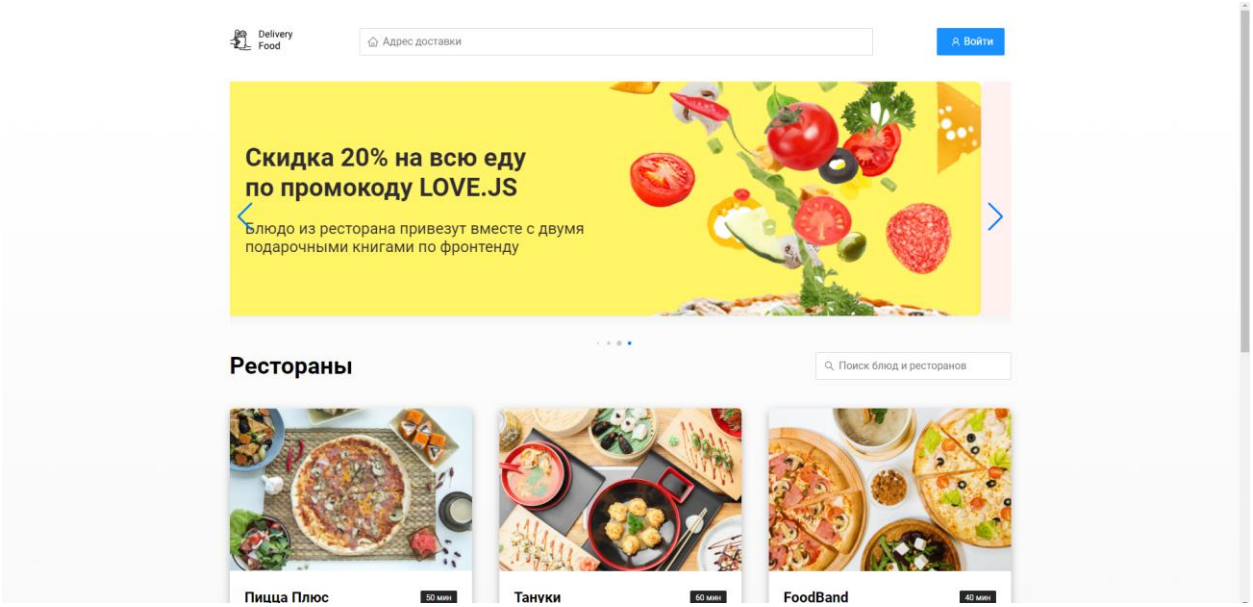

Рисунок 1 – Главная страница сайта

Для каждого ресторана создаётся отдельная карточка (рис. 2) с картинкой, названием, минимальной ценой, средней оценкой и средним временем доставки. Карточки генерируются из JSON файла, который приходит по AJAX запросу с сервера. Каждый ресторан имеет свой объект, в котором хранятся все перечисленные выше характеристики. Все эти объекты перебираются циклом forEach и вставляются в общий список ресторанов в виде контейнеров-ссылок, каждая из которых и является одной карточкой ресторана. Код выглядит следующим образом:

```
const partners = () => {
    const $cardsRestaurants = document.querySelector('.cards-restaurants');
    const $modalAuth = document.querySelector('.modal-auth');

        const   partnersFetch   =   fetch('https://delivery-service-da3cc-default-
rtdb.firebaseio.com/db/partners.json');

    async function checkLogin() {
        if (!localStorage.getItem('user')) {
            $modalAuth.style.display = 'flex';
        }
        else if (localStorage.getItem('user')) {
            window.location.href = 'restaurant.html';
        }
    }

    function renderItems (data, place) {
        data.forEach((item) => {
                const  {image,  name,  time_of_delivery,  stars,  price,  kitchen,
products} = item;
            console.log();
            const a = document.createElement('a');
```

```
            a.dataset.products = products;
            a.setAttribute('href', 'restaurant.html');
            a.classList.add('card', 'card-restaurant');
            a.innerHTML = `
                <img src="${image}" alt="${name}" class="card-image" />
                <div class="card-text">
                    <div class="card-heading">
                        <h3 class="card-title">${name}</h3>
                        <span class="card-tag tag">${time_of_delivery} мин</span>
                    </div>
                    <div class="card-info">
                        <div class="rating">
                            ${stars}
                        </div>
                        <div class="price">От ${price} ₽</div>
                        <div class="category">${kitchen}</div>
                    </div>
                </div>
            `;

            a.addEventListener('click', (e) => {
                e.preventDefault();
                localStorage.setItem('restaurant', JSON.stringify(item));
                checkLogin();
            })

            place.append(a);
        }
        );
    }

    async function appendFromFetch (fet, place) {
        try {
            let response = await fet;
            let data = await response.json();
            renderItems(data, place);
        }
        catch (error) {
            console.error(error);
        }
    }

    appendFromFetch(partnersFetch, $cardsRestaurants);
}
```

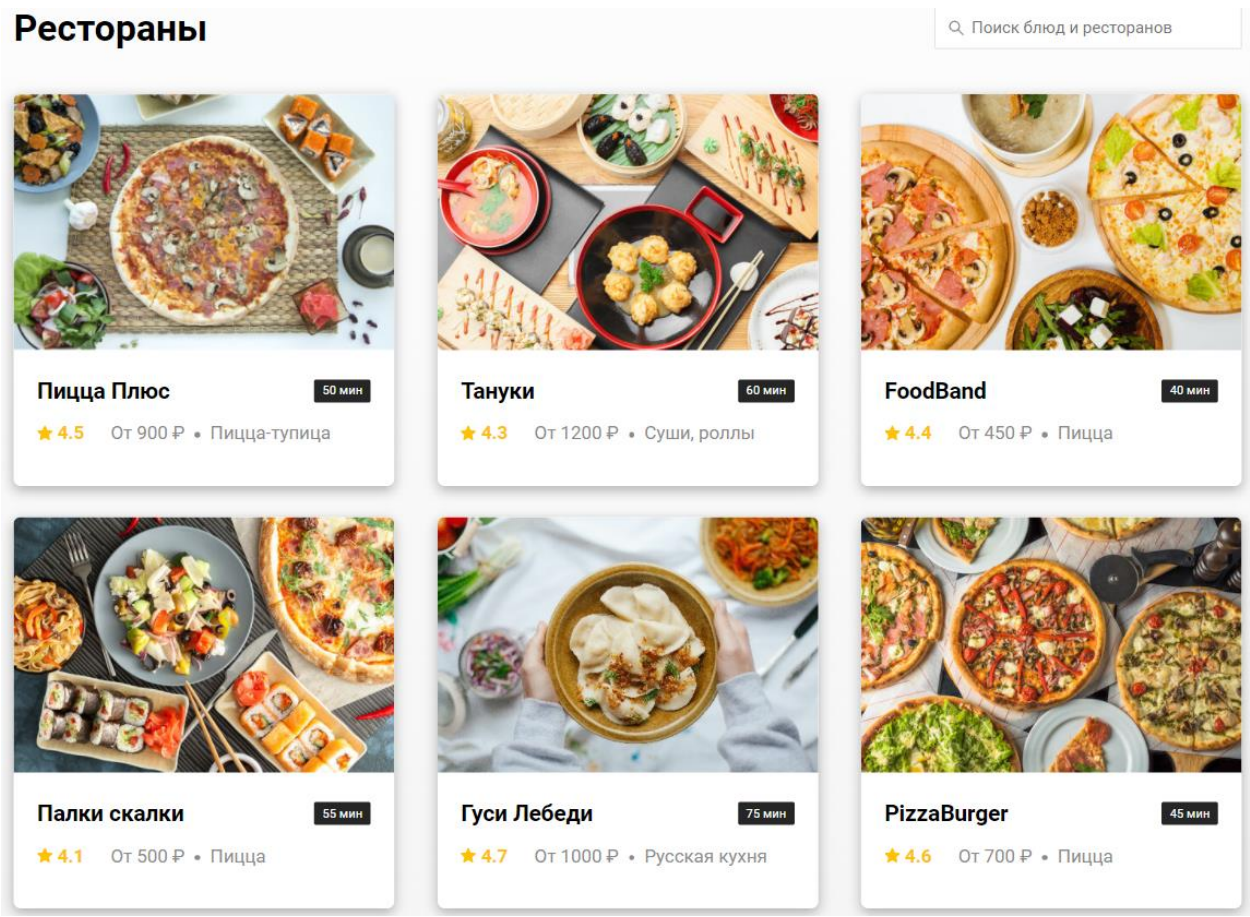

Рисунок 2 – Карточки ресторанов

При нажатии на ресторан срабатывает addEventListener, который передаёт объект из JSON файла в хранилище localStorage, после чего направляет на страницу /restaurant.html, где из записанного в хранилище объекта генерируется страница ресторана и его меню, который тоже берётся с сервера по запросу AJAX (рис. 3). Принцип похож на рендер ресторанов, только теперь вместо карточек заведений – карточки блюд. Код генерирующейся страницы следующий:

```
const menu = () => {
    const $cardsMenu = document.querySelector('.cards-menu');
    const $restaurantTitle = document.querySelector('.restaurant-title');
    const $rating = document.querySelector('.card-info .rating');
    const $price = document.querySelector('.card-info .price');
    const $category = document.querySelector('.card-info .category');

    function addToCart (cardItem) {
        const cartArray = localStorage.getItem('cart') ?
        JSON.parse(localStorage.getItem('cart')) :
        [];

          let checkTheSame = cartArray.findIndex((element) => element.id ===
cardItem.id);

        if (checkTheSame === -1) {
            cartArray.push(cardItem);
        } else {
            ++cartArray[checkTheSame].count;
```

```
        }

        localStorage.setItem('cart', JSON.stringify(cartArray));
    }

    function renderItems (data, place) {
        data.forEach( ({description, id, image, name, price}) => {
            const $divCard = document.createElement('div');
            $divCard.classList.add('card');
            $divCard.innerHTML = `
                <img src="${image}" alt="${name}" class="card-image" />
                <div class="card-text">
                    <div class="card-heading">
                        <h3 class="card-title card-title-reg">${name}</h3>
                    </div>
                    <!-- /.card-heading -->
                    <div class="card-info">
                        <div class="ingredients">${description}
                        </div>
                    </div>
                    <div class="card-buttons">
                        <button class="button button-add-cart">
                            <span class="button-card-text">В корзину</span>
                            <span class="button-cart-svg"></span>
                        </button>
                        <strong class="card-price-bold">${price} ₽</strong>
                    </div>
                </div>
            `;

            $divCard.querySelector('.button-card-text').addEventListener('click',
() => {
                addToCart({ name, price, id, count: 1 });
            })

            place.append($divCard);

        });
    }

    async function changeTitle (restaurant) {
        document.title = `${restaurant.name} — доставка еды на дом`;
        $restaurantTitle.textContent = restaurant.name;
        $rating.textContent = restaurant.stars;
        $price.textContent = `От ${restaurant.price} ₽`;
        $category.textContent = restaurant.kitchen;
    }

    async function appendFromFetch (fet, place) {
        try {
            let response = await fet;
            let data = await response.json();
            renderItems(data, place);
        }
        catch (error) {
            console.error(error);
        }
    }
```

```
    if (localStorage.getItem('restaurant')) {
        const restaurant = JSON.parse(localStorage.getItem('restaurant'));
        changeTitle(restaurant);
             const  menuFetch  =  fetch(`https://delivery-service-da3cc-default-
rtdb.firebaseio.com/db/${restaurant.products}`);
        appendFromFetch(menuFetch, $cardsMenu);
    } else {
        window.location.href = 'index.html';
    }
}

export default menu
```

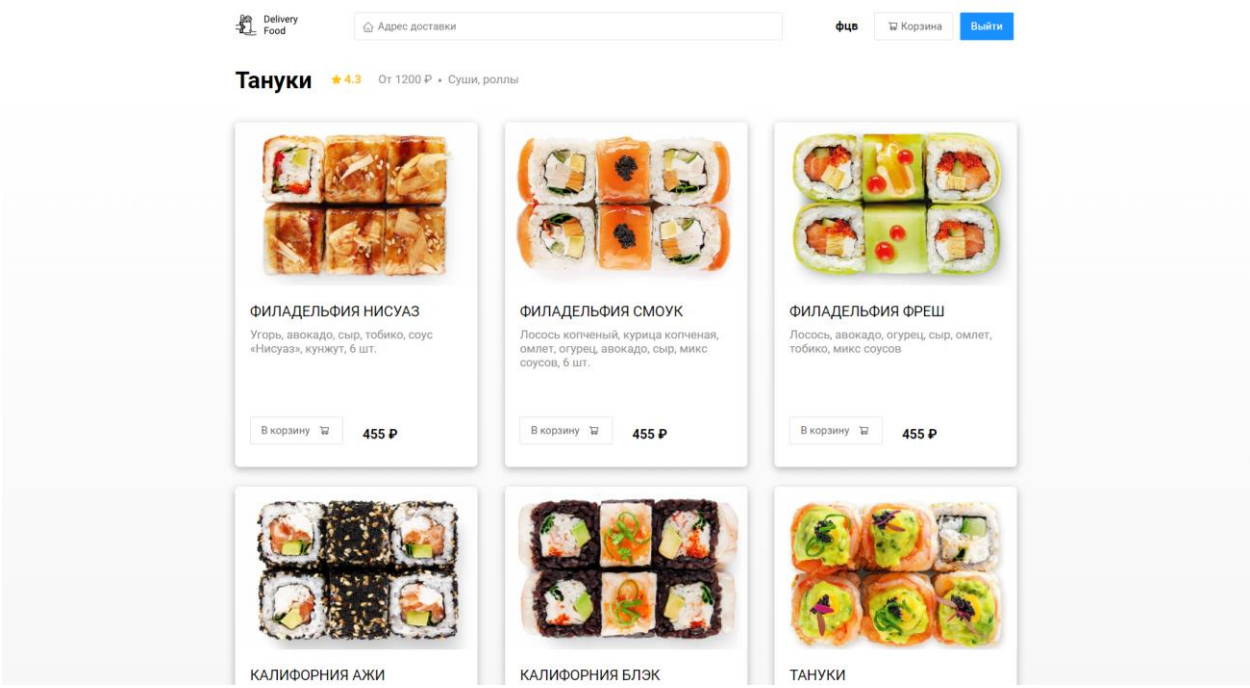

Рисунок 3 – Меню одного из ресторанов

При нажатии на кнопку «В корзину» блюда добавляется в корзину функцией addToCart, которая проверяет на наличие этого блюда в корзине. Если его там нет, то добавляется в корзину новым пунктом. Если уже есть, то просто увеличивается количество этого блюда в заказе на единицу.

В корзине перечислены все блюда (рис. 4), их количество и доступны кнопки по увеличению и уменьшению количества порций. Это регулируется функциями incrementCount и decrementCount соответственно. Также ведётся подсчёт общей суммы и есть кнопки заказа и отмены. При нажатии на эти кнопки корзина очищается (рис. 5). При пустой корзине есть кнопка «продолжить покупки», которая закрывает корзину, но не очищает её. В противоположном случае там находится кнопка «Оформить заказ». Это две разные кнопки со своими конкретными функциями, а не одна и та же кнопка, в которой просто меняется текст. Код корзины выглядит следующим образом:

```
const cart = () => {
    const $buttonCart = document.getElementById('cart-button');
    const $modalCart = document.querySelector('.modal-cart');
```

```
    const $closeModalCart = $modalCart.querySelector('.close');
    const $cartBody = $modalCart.querySelector('.modal-body');
    const $buttonSend = $modalCart.querySelector('.button-primary');
    const $buttonCancel = $modalCart.querySelector('.clear-cart');
    const $buttonContinue = $modalCart.querySelector('.button-continue');
    const $modalPricetag = $modalCart.querySelector('.modal-pricetag');

    function incrementCount(id) {
        const cartArray = JSON.parse(localStorage.getItem('cart'));

        console.log(cartArray);
        console.log(id);

        cartArray.map((item) => {
            if (item.id === id) {
                item.count++;
            }

            /* return item; */
        })

        localStorage.setItem('cart', JSON.stringify(cartArray));
        renderItems(cartArray);
    }

    function decrementCount(id) {
        const cartArray = JSON.parse(localStorage.getItem('cart'));

        console.log(cartArray);
        console.log(id);

        cartArray.map((item) => {
            if (item.id === id) {
                item.count <= 0 ? item.count = 0 : item.count--;
            }
            /* return item; */
        })
        localStorage.setItem('cart', JSON.stringify(cartArray));
        renderItems(cartArray);
    }

    let totalPrice = 0;
    function renderItems(data) {
        $cartBody.innerHTML = '';
        data.forEach(({name, price, id, count} ) => {
            const $foodRow = document.createElement('div');
            $foodRow.classList.add('food-row');
            $foodRow.innerHTML = `
                <span class="food-name">${name}</span>
                <strong class="food-price">${price * count} ₽</strong>
                <div class="food-counter">
                      <button class="counter-button btn-dec" data-index="${id}">-
</button>
                    <span class="counter">${count}</span>
                                <button  class="counter-button  btn-inc"  data-
index="${id}">+</button>
                </div>
            `;
```

```
            $cartBody.append($foodRow);
        });

         let totalPrice = data.reduce((sum, current) => sum + (current.price *
current.count), 0);
        console.log(totalPrice);
        $modalPricetag.textContent = `${totalPrice} ₽`;
    }

    function resetCart() {
        $cartBody.innerHTML = '';
        localStorage.removeItem('cart');
        $modalCart.classList.remove('is-open');

    }
     /*-----------ДЕЛЕГИРОВАНИЕ ОБРАБОТЧИКА СОБЫТИЙ. ОДИН ОБРАБОТЧИК НА ТЕЛЕ
КОРЗИНЫ, ВМЕСТО ОБРАБОТЧИКА НА КАЖДОЙ!!! КНОПКЕ - и + ------*/
    $cartBody.addEventListener('click', (e) => {
        if (e.target.classList.contains('btn-inc')) {
            incrementCount(e.target.dataset.index);
        } else if (e.target.classList.contains('btn-dec')) {
            decrementCount(e.target.dataset.index);
        }
    })
    $buttonSend.addEventListener('click', () => {
        const cartArray = localStorage.getItem('cart');

        fetch('https://jsonplaceholder.typicode.com/posts', {
            method: 'POST',
            body: cartArray,
        }).then(response => {
            if (response.ok) {
                resetCart();
            }
        })
        .catch( e => console.error(e));
    })

    $buttonCancel.addEventListener('click', () => {
        $cartBody.innerHTML = '';
        localStorage.removeItem('cart');
        $modalCart.classList.remove('is-open');
    })

    $buttonContinue.addEventListener('click', () => {
        $modalCart.classList.remove('is-open');
    })

    $buttonCart.addEventListener('click', () => {
        if (!localStorage.getItem('cart')) {
            $cartBody.innerHTML = '';

            const $emptyCartMessage = document.createElement('div');
            $emptyCartMessage.classList.add('empty-cart-message');
            $emptyCartMessage.innerHTML = 'Ваша корзина пуста. Вернитесь в меню и
выберите что-нибудь на ваш вкус :)';
            $cartBody.append($emptyCartMessage);

            $buttonContinue.classList.add('is-open');
```

```
            $buttonSend.style.display = 'none';
        }
        if (localStorage.getItem('cart')) {
            renderItems(JSON.parse(localStorage.getItem('cart')));
            $buttonContinue.classList.remove('is-open');
            $buttonSend.style.display = 'flex';
        }
        $modalCart.classList.add('is-open');
    })

    $closeModalCart.addEventListener('click', () => {
        $modalCart.classList.remove('is-open');

    })
}

export default cart
```

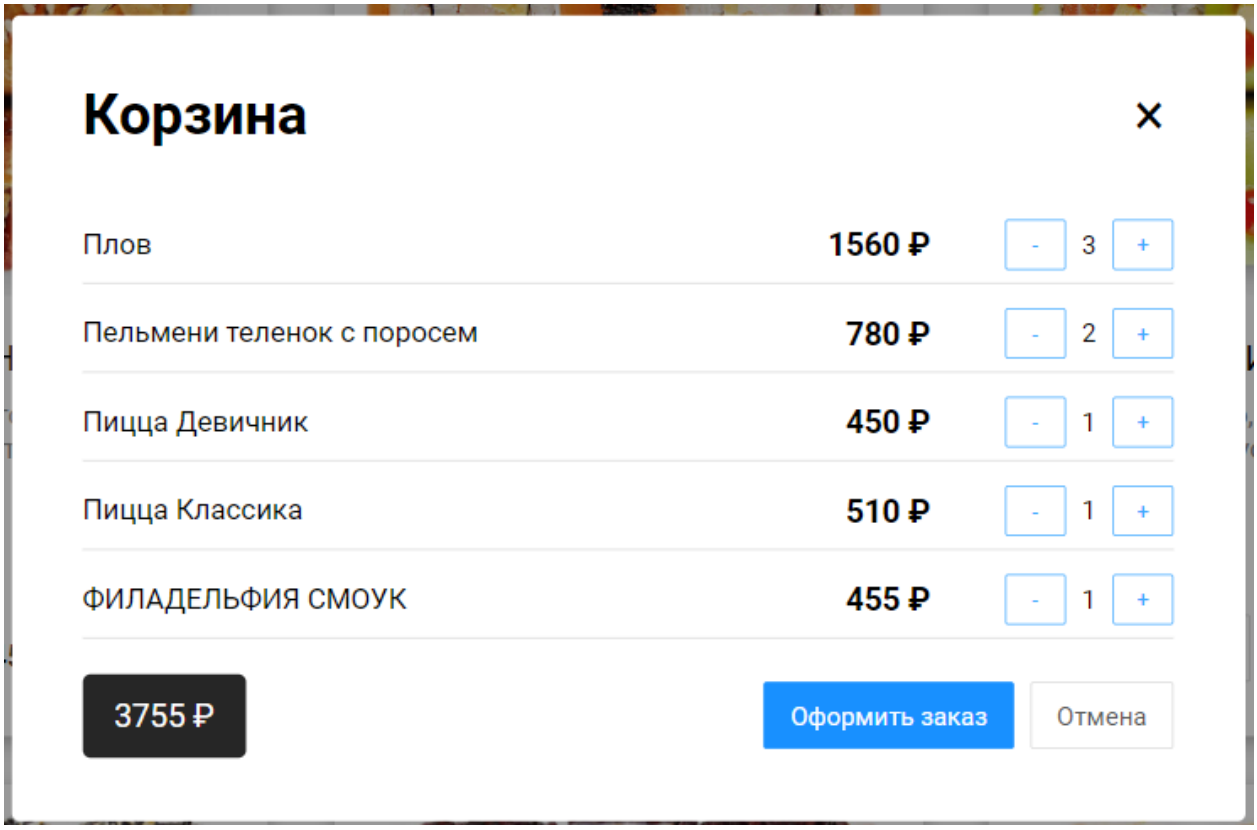

Рисунок 4 – Корзина блюд

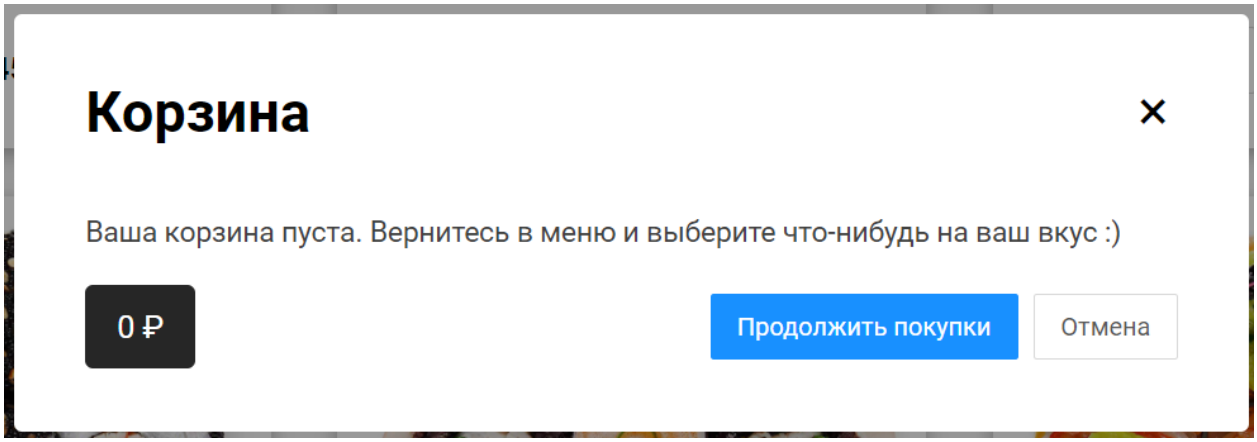

Рисунок 5 – Пустая корзина

Таким образом, разработана frontend часть веб-приложения сервиса по доставке еды. Были использованы функции для работы с сервером и локальным хранилищем браузера. Страницы формируются по содержимому ответа сервера. Преимущество такого подхода в том, что не нужно менять наполнение сайта вручную. Всё делается автоматически, если данные о ресторанах и блюдах поменяются на сервере. Это значит, что менеджеры ресторанов могут подключиться к такому сайту с доставкой и менять своё меню сами, без помощи программистов. А изменённые данные уже сами отобразятся в обновлённом виде на сервисе доставки еды.

Библиографический список

1. Ермолик А. В., Багаева А. П. Создание интернет-магазина //Актуальные проблемы авиации и космонавтики. 2013. Т. 1. №. 9.
2. Фомина Т. Г., Соколова А. С. Интернет-магазин: понятие и виды //Nauka-Rastudent. ru. 2014. №. 2. С. 20-20.
3. Галимзянов З. В. Разработка интернет-магазина //Научные междисциплинарные исследования. 2021. №. 3. С. 35-40.
4. JavaScript URL: https://ru.wikipedia.org/wiki/JavaScript (дата обращения: 03.01.2022).
5. HTML URL: https://ru.wikipedia.org/wiki/HTML (дата обращения: 03.01.2022).
6. CSS URL: https://ru.wikipedia.org/wiki/CSS (дата обращения: 03.01.2022).