

ОПИСАНИЕ ЗАДАНИЯ

К Вам обратилась компания «Просто купить» для создания веб-приложения (SPA) для их сервиса заказа товаров. Заказчик предоставляет рабочее API (коллекция запросов ниже).

При разработке сайта используется HTML — язык гипертекстовой разметки документов для создания и размещения элементов страниц в браузере. Чтобы стилизовать элементы HTML используется CSS («каскадные таблицы стилей») — язык описания внешнего вида документа. Но эти языки *не являются языками программирования*.

Программирование используется для работы с запросами, приходящими с API, динамического отображения элементов (например, всплывающего окна).

Язык программирования, который используется для работы с элементами HTML - JavaScript. Чем больше функций и страниц в приложении, тем сложнее становится его разработка и объединение его разных компонентов. Для облегчения этих задач используются различные фреймворки и библиотеки, которые также написаны на JavaScript. Вы можете использовать любой из них: Vue, React и т.д.

Ниже дана документация с подробным разъяснением основ HTML, CSS, JS, Vue, React. Перед выполнением задания рекомендуется ознакомиться как минимум с HTML, CSS, JS.

Рекомендуется для написания проекта использовать IDE (интегрированная среда разработки), такие как Webstorm.

Что такое Single Page Application (SPA) - <https://thecode.media/spa/>

При выполнении данного задания ставятся следующие задачи:

1. Сверстать все необходимые страницы и элементы управления.
2. Разработать приложение (SPA), согласно требованиям, описанным в задании.
3. Улучшить пользовательский опыт (UX) взаимодействия с реализованным SPA приложением с помощью анимирования взаимодействий с интерфейсом, интерактивными сообщениями и т.д.
4. Проект разместить в github/gitlab репозитории. Проект должен содержать подробную инструкцию по его запуску (она создается автоматически).
5. Рекомендуется также развернуть docker контейнер с проектом. Настроить окружение с использованием docker-compose.

Проект может содержать:

- Папку deploy с docker-compose file
- Папку project с проектом

Навигация по заданию:

- [Описание проекта и задач](#)
- [Функциональные требования](#)
- [Общее описание API](#)
- [Специфические особенности API](#)
- [Источники по технологиям](#)
- [Предоставление результатов работы](#)

Описание проекта и задач

Ваше SPA должно состоять из следующих компонентов:

- [Каталог товаров](#)
- [Регистрация](#)
- [Вход в систему](#)
- [Корзина](#)
- [Оформленные заказы](#)

Для улучшения пользовательского опыта (UX) все взаимодействия с интерфейсом приложения должны сопровождаться анимацией, интерактивными сообщениями и т.д.

Функциональные требования

Каталог товаров

На данном домашнем экране находится список всех товаров из каталога. В зависимости от роли пользователя, отображаются те или иные дополнительные элементы интерфейса:

Если пользователь не авторизован:

- Присутствуют ссылки на регистрацию и авторизацию в системе;

Если пользователь авторизован:

- Присутствует ссылка на выход из аккаунта;
- Возле каждого товара отображается кнопка для добавления его в корзину;
- Присутствует ссылка для просмотра ранее оформленных заказов;

Интерактивные элементы страницы

- По щелчку на **ссылки регистрации и авторизации** осуществляется переход на соответствующие страницы или модальные окна;
- По щелчку на **ссылку выхода из аккаунта** пользователь становится неавторизованным;
- По щелчку на **ссылку для оформленных заказов** осуществляется переход на страницу ранее оформленных заказов;
- По щелчку на **кнопку добавления товара**, соответствующий товар добавляется в корзину.

Регистрация

На данном экране представлена форма для регистрации нового пользователя.

При вводе **некорректных значений** у соответствующих полей формы отображаются тексты ошибок, а сами поля подсвечиваются красным.

При **успешной регистрации** происходит переход на страницу входа в систему.

Интерактивные элементы страницы

- По щелчку на **кнопку регистрации**, происходит попытка зарегистрировать нового пользователя;
- По щелчку на **кнопку назад**, происходит переход на домашнюю страницу.

Вход в систему

На данном экране представлена форма для входа пользователя в систему.

- При вводе **некорректных значений** у соответствующих полей формы отображаются тексты ошибок, а сами поля подсвечиваются красным.
- При **успешной авторизации** происходит переход на домашнюю страницу, **иначе** выводится соответствующее сообщение, объясняющее причину по которой пользователь не смог авторизоваться.

Интерактивные элементы экрана

- По щелчку на **кнопку входа**, происходит попытка авторизации пользователя;
- По щелчку на **кнопку назад**, происходит переход на домашнюю страницу.

Корзина

На данном экране отображаются список добавленных в корзину товаров.

- Одинаковые товары в корзине **сгруппированы** с указанием количества.
- Возле каждого из товаров **есть кнопки** увеличения и уменьшения его количества, а также удаления его из корзины.
- Если в корзине есть товары, то также присутствует **кнопка** для оформления заказа. В том случае, если их нет то выводится сообщение “Корзина пуста”.

Интерактивные элементы страницы

- По щелчку на **кнопки увеличения/уменьшения** количества товара, соответственно количество товара увеличивается/уменьшается;
- По щелчку на **кнопку удаления** товара из корзины, соответствующий товар удаляется из корзины;
- По щелчку на **кнопку оформления** заказа, происходит оформление заказа и переход на страницу с заказами;
- По щелчку на **кнопку назад**, происходит переход на домашнюю страницу.

Оформленные заказы

На данной странице отображается список оформленных заказов пользователя.

Интерактивные элементы страницы

- По щелчку на **кнопку назад**, происходит переход на домашнюю страницу.

Общие описание API

Идентификация реализована посредством **Bearer Token**.

Bearer Token - это *токен на предъявителя*. Соответственно, кто его предъявит, тот и является авторизованным пользователем. Поскольку токен доступен на клиенте, это и используется: обычно клиент отправляет к серверу запросы с заголовком:

Authorization: Bearer <token>.

API - это набор адресов, которые отдают какие то определённые данные. (Что такое API: <https://habr.com/ru/articles/464261/>)

При попытке доступа к защищенным авторизацией функциям системы во всех запросах возвращается ответ следующего вида:

Status: 403

Content-Type: application/json

Body:

```
{
  "error": {
    "code": 403,
    "message": "Login failed"
  }
}
```

При попытке доступа авторизованным пользователем к функциям недоступным для своей группы во всех запросах возвращается ответ следующего вида:

Status: 403

Content-Type: application/json

Body:

```
{
  "error": {
    "code": 403,
    "message": "Forbidden for you"
  }
}
```

```
}
```

При попытке получить несуществующий ресурс возвращается ответ следующего вида:

Status: 404

Content-Type: application/json

Body:

```
{
  "error": {
    "code": 404,
    "message": "Not found"
  }
}
```

В случае ошибок связанных с валидацией данных во всех запросах возвращается следующее тело ответа:

```
{
  "error": {
    "code": 422,
    "message": "Validation error",
    "errors": {
      "<key>": [ <error message>]
    }
  }
}
```


Специфические особенности API

В примерах будет использоваться переменная `{{host}}` которая обозначает адрес <http://lifestealer86.ru/api-shop/>.

Функционал гостя

Аутентификация

При успешной аутентификации возвращается сгенерированный токен пользователя.

Request	Response
URL: <code>{{host}}/login</code> Method: POST Headers - Content-Type: application/json Body: { "email": "admin@admin.ru", "password": "admin" }	<div><u>Успех</u> Status: 200 Content-Type: application/json Body: { "data": { "user_token": <сгенерированный token> } }</div> <hr/> <div><u>Неправильные логин или пароль</u> Status: 401 Content-Type: application/json Body: { "error": { "code": 401, "message": "Authentication failed" } }</div>

Регистрация

При успешной регистрации возвращается сгенерированный токен добавленного пользователя

Request	Response
URL: {{host}} /signup Method: POST Headers - Content-Type: application/json Body: { "flo": "Иванов Иван Иванович", "email": "admin@admin.ru", "password": "admin" }	<div>Успех</div> Status: 201 Content-Type: application/json Body: { "data": { "user_token": <сгенерированный token> } } <hr/> <div>Ошибки валидации полей</div> <div>Формат ответа из общих требований</div>

Просмотр списка товаров

Возвращается массив data, содержащий список объектов товаров.

Request	Response
URL: {{host}} /products Method: GET	<div>Успех</div> Status: 200 Content-Type: application/json Body: { "data": [{ "id": 1, "name": "Product name 1", "description": "Product description 1", "price": 100 }, { "id": 2, "name": "Product name 2", "description": "Product description 2", "price": 200 }] }

Функционал клиента (Авторизованного пользователя)

Добавление товара в корзину

Request	Response
URL: {{host}} /cart/{product_id} Method: POST Headers - Authorization: Bearer {user_token} Примечание: {product_id} - идентификатор товара	<div>Успех</div> Status: 201 Content-Type: application/json Body: { "data": { "message": "Product add to card" } }

Просмотр своей корзины

Возвращается массив data, содержащий список товаров в корзине.

Request	Response
URL: {{host}} /cart Method: GET Headers - Authorization: Bearer {user_token}	<div>Успех</div> Status: 200 Content-Type: application/json Body: { "data": [{ "id": 1, "product_id": 1, "name": "Product name 1", "description": "Product description 1", "price": 100 }, { "id": 2, "product_id": 1, "name": "Product name 1", "description": "Product description 1", "price": 100 }, { "id": 3, "product_id": 2, "name": "Product name 2", "description": "Product description 2", "price": 200 }] } Примечание: "id" - идентификатор товара в корзине "product_id" - идентификатор товара

Удаление товара из корзины

Request	Response
<p>URL: {{host}}/cart/{id}</p> <p>Method: DELETE</p> <p>Headers</p> <ul style="list-style-type: none">- Authorization: Bearer {user_token} <p>Примечание: {id} - идентификатор товара в корзине</p>	<p><u>Успех</u></p> <p>Status: 200 Content-Type: application/json Body:</p> <pre>{ "data": { "message": "Item removed from cart" } }</pre> <hr/> <p><u>Попытка удалить товар не из своей корзины</u></p> <p>Status: 403 Content-Type: application/json Body:</p> <pre>{ "error": { "code": 403, "message": "Forbidden for you" } }</pre>

Оформления заказа

Request	Response
<p>URL: {{host}}/order</p> <p>Method: POST</p> <p>Headers</p> <ul style="list-style-type: none">- Authorization: Bearer {user_token}	<p><u>Успех</u></p> <p>Status: 201 Content-Type: application/json Body:</p> <pre>{ "data": { "order_id": 1 "message": "Order is processed" } }</pre> <hr/> <p><u>Попытка оформить заказ с пустой корзиной</u></p> <p>Status: 422 Content-Type: application/json Body:</p> <pre>{ "error": { "code": 422, "message": "Cart is empty" } }</pre>

Просмотр своих оформленных заказов

Request	Response
URL: {{host}} /order Method: GET Headers - Authorization: Bearer {user_token}	<div>Успех</div> Status: 200 Content-Type: application/json Body: { "data": [{ "id": 1, "products": [1, 1, 2], "order_price": 400 }, { "id": 5, "products": [1, 2], "order_price": 300 }] } Примечание: Массив products содержит идентификаторы товаров в заказе

Выход

Запрос предназначен для очистки значение токена пользователя.

Request	Response
URL: {{host}} /logout Method: GET Headers - Authorization: Bearer {user_token}	<div>Успех</div> Status: 200 Content-Type: application/json Body: { "data": { "message": "logout" } }

Источники по технологиям

HTML (HyperText Markup Language) - это язык разметки, который используется для создания структуры и содержимого веб-страницы. Он определяет элементы и их расположение на странице, такие как заголовки, параграфы, изображения и ссылки.

CSS (Cascading Style Sheets) - это язык стилей, который используется для оформления веб-страниц. Он определяет внешний вид элементов HTML, такие как цвет фона, шрифт, размеры и позиционирование.

JavaScript (JS) - это язык программирования, который используется для добавления интерактивности и динамического поведения на веб-странице. Он может изменять содержимое и стили элементов HTML и реагировать на действия пользователя, такие как клики и наведение курсора.

Таким образом, HTML определяет структуру страницы, CSS задает ее внешний вид, JavaScript добавляет интерактивность и динамическое поведение. Все три языка работают вместе для создания полноценных и функциональных веб-страниц.

Чтобы увидеть элементы HTML на странице сайта нужно открыть сайт и нажать F12. Откроются инструменты разработчика, которые также рекомендуется изучить.

HTML

- [Изучение HTML: руководства и уроки - Изучение веб-разработки | MDN](#)
- [HTML и HTML5. Описание тегов по основным разделам](#)
- [HTML Tutorial](#)

CSS

- [Руководство по Flexbox/Grid](#)
- [Flexbox Froggy](#)
- [Flexbox - Изучение веб-разработки | MDN](#)

Препроцессор SASS/SCSS

- [Основы Sass](#)

Tailwind CSS

- [Установка Tailwind CSS с Vite](#)
- [Официальная документация Tailwind CSS](#)

JavaScript

- [Современный учебник JavaScript](#)
- [JavaScript - Изучение веб-разработки | MDN](#)
- <https://www.freecodecamp.org/learn/javascript-algorithms-and-data-structures/>
- [JavaScript | MDN](#)
- [LocalStorage, sessionStorage](#)

Axios/Fetch (Взаимодействие с API):

- [Fetch](#)
- [Axios](#)

TypeScript

- [Введение в TypeScript](#)
- <https://www.typescriptlang.org>

Vue.js


- [Официальная документация VueJS](#)
- [Тutorial по основам фреймворка Vue](#)
- [Vue Router](#)
- [Pinia](#)
- [Vuex](#)
- Форма регистрации с отправкой запроса и валидацией (Composition API)
 - ☰ Валидация формы(vee-validate) и отправка данных(axios) на...
- ☰ Методическое пособие Знакомство с фреймворком Vue.js (Option API)
- Пример формы авторизации с отправкой запроса (Option API)
 - ☰ Authentication vue
- Пример формы регистрации с валидацией (Option API)
 - ☰ Registration vue

React.js

- [Официальная документация React](#)
- [Начало работы с React - Изучение веб-разработки | MDN](#)

Git/Github

- [Git и GitHub - Изучение веб-разработки | MDN](#)
- <https://githowto.com/ru>

- https://learngitbranching.js.org/?locale=ru_RU
-  1.1 Git – Введение – Что такое Git?

Docker

- [Изучаем Docker, часть 1: основы / Хабр](#)

Предоставление результатов работы

Результаты прикрепить в moodle в курс задания в виде файла .docx

Укажите:

Telegram: ссылка ваш аккаунт в Telegram

Github: ссылка на github с выполненным заданием

Почему решили пойти на Кафедру: Описать почему решили пойти на кафедру в этой роли (не более 2-3 предложений)

Немного о себе: Описать свои личностные качества, интересы и увлечения (не более 2-3 предложений)