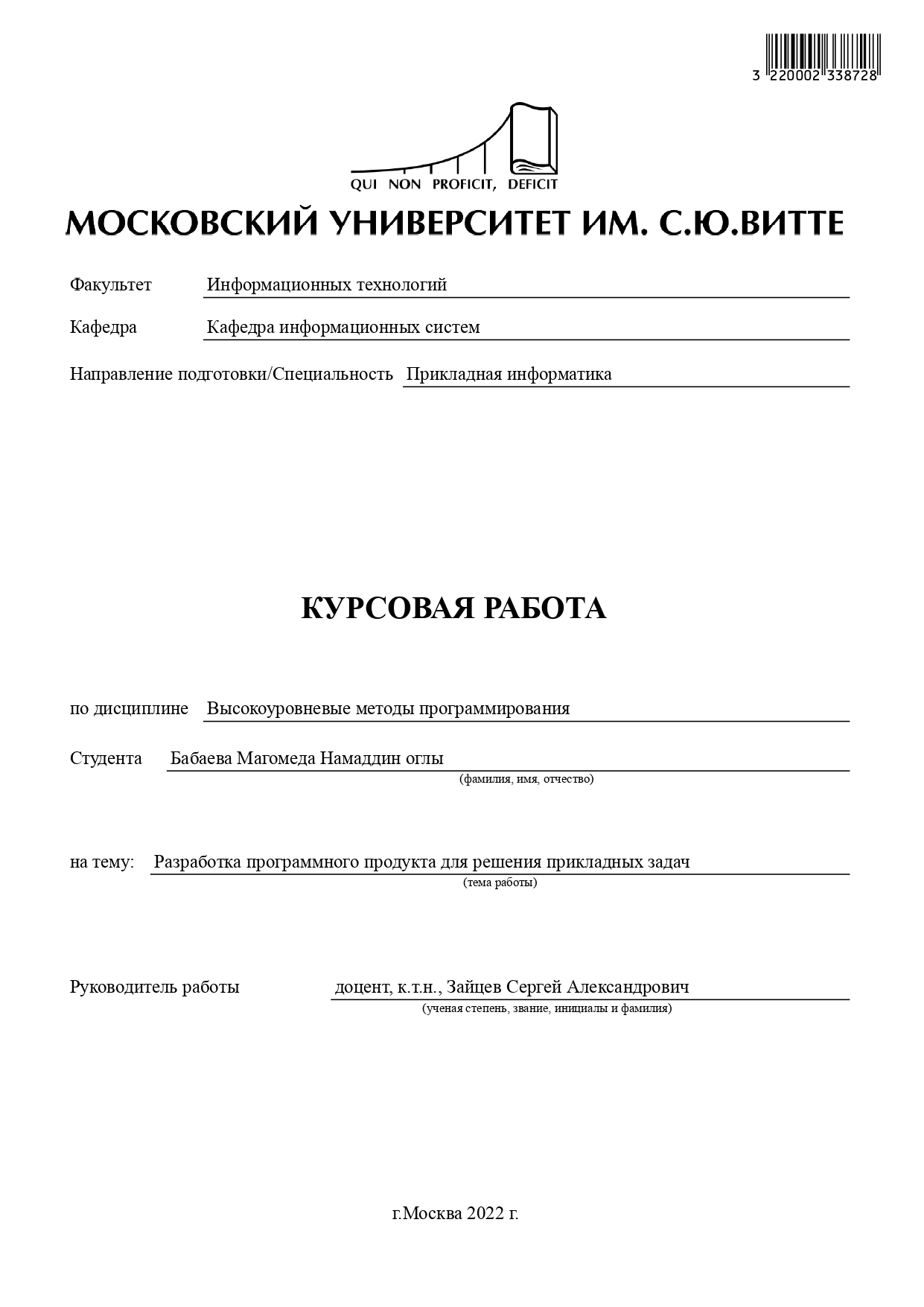
**

[Введение: 2](#_Toc128415440)

[1. Постановка задачи проектной работы: 2](#_Toc128415441)

[1.1 Выводы 3](#_Toc128415442)

[2. Выбор языков программирования для написания проекта 3](#_Toc128415443)

[3. Подключение базы данных 3](#_Toc128415444)

[3.1 Вывод по 4 главе 4](#_Toc128415445)

[4. Регистрация пользователей 5](#_Toc128415446)

[5. Авторизация 8](#_Toc128415447)

[6. Проверка авторизации 10](#_Toc128415448)

[7. Деавторизация 10](#_Toc128415449)

[8. Главная страница 11](#_Toc128415450)

[9. Личный кабинет 13](#_Toc128415451)

[9.1. Настройка и редактирование данных пользователя 13](#_Toc128415452)

[9.2. Корзина пользователя 13](#_Toc128415453)

[9.3. Отложенные товары 14](#_Toc128415454)

[10. Страница товара 14](#_Toc128415455)

[11. Выводы 15](#_Toc128415456)

[12. Приложение 15](#_Toc128415457)

# Введение:

В условиях быстро развивающейся современной высокотехнологичной экономики актуальной задачей является разработка различных автоматизированных систем. Они позволяют наиболее эффективно использовать различные современные средства информационной вычислительной техники и решать прикладные задачи. Для успешного решения таких задач нужно уметь использовать высокоуровневые средства разработки и декомпозировать задачу. В данной работе решаются задачи высокоуровневого программирования, согласно индивидуальному заданию. Актуальность данной работы связана с наличием большого количества прикладных задач, которые необходимо эффективно решать с помощью инструментов высокоуровневых методов программирования.

# Постановка задачи проектной работы:

Написать сайт, для онлайн продажи парфюмерной продукции. Он должен обладать современным дизайном, а также соответствовать современным стандартам безопасности и защиты данных клиентов (ФЗ-152). Должен быть реализован следующий функционал:

1. Регистрация
2. Авторизация
3. Выход
4. Страница всех товаров
5. Страница конкретного товара с блоком похожих товаров
6. Поиск по товарам
7. Корзина
8. Отложенные товары
9. Админ панель
10. Возможность назначить администратором любого пользователя
11. Добавление товаров

## 1.1 Выводы

Проанализировав построенную задачу из главы 1, можно сделать вывод, что задания включают наиболее часто используемые прикладные задачи высокоуровневого программирования – это работа с файлами, работа с базами данных, использование структур данных и алгоритмов, построение сайта с графическим пользовательским интерфейсом, работа с наиболее популярной среде разработки. Успешное выполнение задания будет свидетельствовать о полноте усвоенного материала в рамках дисциплины «Методы высокоуровневого программирования» и умение решать задачи низкой и средней сложности.

# 2. Выбор языков программирования для написания проекта

Для реализации поставленной цели, нужно выбрать языки программирования для последующей разработки. Для написания backend-а, был выбран язык Node js в связке с NPM модулем Express поскольку данный язык соответствует всем современным стандартам и обладает рядом преимуществ над языком PHP. В настоящее время поддерживается подавляющим большинством хостинг-провайдеров и является одним из лидеров среди языков, применяющихся для создания динамических веб-сайтов. Frontend был написан с помощью HTML, CSS, Java Script. Работа с базами данных проводилась с помощью sqlite3.

# 3. Подключение базы данных

Весь веб-сайт будет взаимодействовать с базой данных SQL, это означает, что на каждой странице, в каждом файле, мы должны подключаться к базам данных. Как было указано выше, средой разработки базы данных был выбран npm модуль sqlite3 поскольку он позволяет без лишних проблем и сложностей взаимодействовать с базой данных. Для его установки требуется прописать в консоле команду «npm i sqlite3». Для подключения этого модуля в нужном месте требуется прописать строку (Рис. 1).



Рис. 1

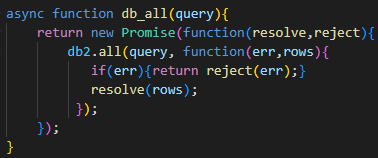
Для удобства запросов к базе данных была написана следующая функция (Рис.2).

Рис. 2

Таким образом, для осуществления SQL запроса нам нужно вызвать созданную функцию в нужном нам месте и в качестве аргумента передать сам запрос (Рис. 3)

Рис.3

## 3.1 Вывод по 4 главе

Поскольку весь веб-сайт будет взаимодействовать с базой данных SQL, а также с точки зрения объектно-ориентированного программирования, лучшим решением явлется использование npm модуля «sqlite3». Это позволит нам в будущем быстро подключаться к базе данных, а также быстро взаимодействовать с базой данных. Структура будет выглядеть следующим образом:

|  |
| --- |
| Const db2 = new sqlite3.Database(‘путь к файлу базы данных’)  Function db\_all (Запрос к базе данных) {  return new Promise(function(resolve,reject){          db2.all(query, function(err,rows){             if(err){return reject(err);}             resolve(rows);           });      });  }  Const sqlReq = ‘запрос к базе данных’  db\_all(sqlReq) |

# 4. Регистрация пользователей

Поскольку веб-сайт ориентирован в основном на работу с пользователями, первым делом реализуем регистрацию. Для этого создадим несколько каталогов, в которых будут лежать все последующие файлы для регистрации, авторизации и выхода. Создаём новый роутер с названием “auth.js”, а также необходимо создать модуль с названием “sign-up”, который будет подгружаться в случае когда пользователь не авторизован. Регистрация пользователей была реализована всплывающем окном авторизации. На данном этапе требуется продумать все требования к регистрации нового пользователя:

1. Предусмотреть не возможность регистрации пользователей с одинаковым логином.
2. Регистрация должна обладать реактивностью т.е. в случае удачной или неудачной регистрации, пользователь должен оставаться на одной и той же странице.
3. Обработка запроса на регистрацию должна быть реализована с помощью JavaScript асинхронных запросов.
4. В момент обработки пользователь должен наблюдать “loader”.

Таким образом был создан модуль с полями ‘Login’ ‘Email’ ‘Password’ и кнопкой ‘Sign-up’. Все поля обязательны для заполнения. На поле “Login” присвоен слушатель событий “onInput”, который запускает функцию проверки поля (Рис. 4).

Рис. 4

Функция является асинхронной. В начале создаются переменная, куда помещается значение поле введённого пользователем. Необходимо предотвратить бесконечные запросы к базе данных при вводе каждого символа поскольку это очень сильно ударит по производительности сайта. Для этого вначале существует проверка на существование переменной, которую мы создаём ниже и помещаем туда таймер срабатывания функции запроса к базе данных. В случае существования этой переменной мы её удаляем. Таким образом, при вводе пользователем символов логина создаётся таймер, который выполняет функцию через 5 секунд, но в случае если пользователь ввёл ещё один символ или его удалил, то проверка удаляет созданный ранее таймер и создаёт новый, в результате чего мы обеспечили корректную работу данного поля. Сама функция проверки посылает асинхронный запрос на сервер, где обрабатывается введённое пользователем значение логина, сравнивает с уже имеющимися значениями в базе пользователей. В случае совпадения, сервер возвращает негативный ответ и поле становится не валидным, что запрещает отправку формы регистрации, соответственно мы предотвратили вариант создания нескольких пользователей с одинаковым логином, а пользователь получает сообщение о том, что такой логин уже занят. В случае положительного ответа от сервера, поле становится валидным.

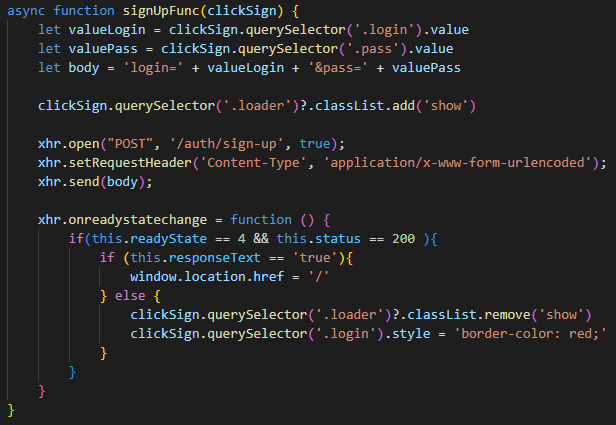
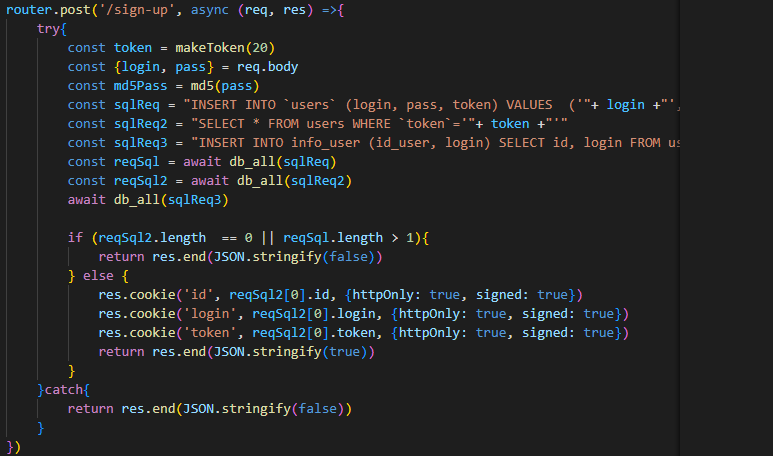
В случае нажатия на кнопку регистрации запускается функция (Рис.5) куда передаётся форма регистрации.

Рис. 5

Функция является асинхронной. В начале создаются переменные, куда помещаются введённые пользователем значения. После создаётся переменная с “POST” запросом куда подставляются введённые значения. Затем создается запрос с указанием метода, пути выполнения запроса и назначением его асинхронности. После запрос отправляется на сервер (Рис. 6), а пользователь видит “loader”.

Рис. 6

В случае удачной регистрации создаются подписанные сервером куки авторизации и возвращается ответ true в json формате. Если на этапе регистрации произошли какие-либо ошибки, возвращается ответ false в json формате. Если сервер прислал ответ true, то страница перезагружается и кнопки регистрации и входа пропадают. Если сервер прислал ответ false, то все поля отчищаются становятся красными.

# 5. Авторизация

Создадим компонент “sign-in”. Это всплывающее меню, в котором пользователь вводит свой логин и пароль. По нажатию на кнопку войти, запускается функция, аргументом которой является форма входа с данными, которые ввёл пользователь (Рис. 7).

Рис. 7

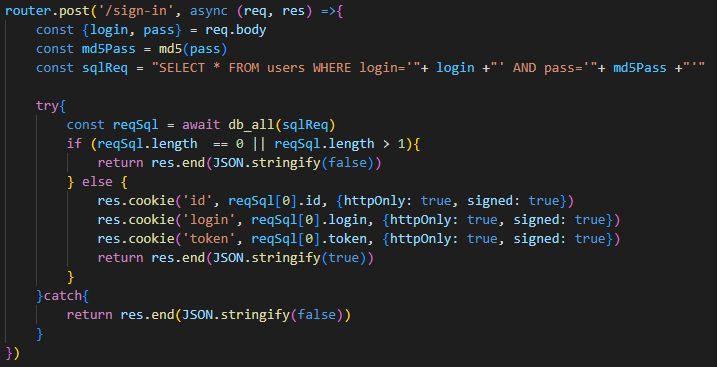
Данная функция посылает POST запрос на сервер. Телом запроса являются введённые пользователем данные, которые обрабатываются на сервере (Рис. 8). В результате обработки данных на сервере, сервер присылает ответ. В случае совпадения данных, авторизация считается успешной, сервер создаёт куки авторизации (id, login, token) и возвращает ответ true. В случае не совпадения данных, авторизация считается не успешной, а сервер возвращает ответ false.

Рис. 8

Как только сервер прислал ответ, запускается функция, которая обрабатывает ответ. Если сервер прислал ответ true, пользователя переадресует на главную страницу. Таким образом Страница обновляется, кнопки входа или регистрации пропадают, появляются кнопки профиля пользователя. В случае если сервер прислал ответ false, поля логина и пароля подсвечиваются красным цветом, поле пароля отчищается и появляется сообщение «Invalid username or password».

# 6. Проверка авторизации

Проверка авторизации представляет собой проверку наличия куков авторизации, а именно куки с названием id, login, token. Важным фактором является проверка только подписанных сервером куки, поскольку пользователь может сам создать вышеуказанные куки и авторизоваться под чужим профилем. Во избежание таких ситуаций, все куки должны быть подписаны и работа должна проводиться только с ними. В случае наличия подписанных куки, кнопки авторизации перестают существовать, на их место встают кнопки пользователя (личный кабинет, корзина и т.д.).

# 7. Деавторизация

Существует необходимость деавторизации пользователей. Для этого создадим кнопку, которая будет перенаправлять пользователя на страницу деевторизации по пути «/auth/sign-out». Для деавторизации требуется удалить подписанные куки пользователя. Для этого создадим новый роутер, во избежание ошибок (например перехода пользователем по данной ссылке будучи не авторизованным и т.п.) поместим код для удаления куки в оператор try catch. Таким образом, в случае попытки деавторизации не авторизованного пользователя произойдёт ошибка, но сервер продолжит работу, а пользователь увидит страницу ошибки «404: Not found». Если пользователь был авторизован, все куки авторизации удаляются, а пользователя переадресуют на главную страницу (Рис. 9).



Рис. 9

# 8. Главная страница

Главная страница представляет собой список товаров, с примитивной фильтрацией по типу товаров. Все товары отсортированы по популярности, а именно по количеству просмотров на товаре. Главная страница должна обладать следующим функционалом:

* Header (поиск по товарам, кнопки авторизации или кнопки профиля)
* Меню фильтрации по типу товаров
* Список товаров, представляющий собой карточки товаров
* Возможность добавления и удаления товара в корзину или отложенные товары
* По нажатию на карточку товаров открывается страница товара
* Footer (ссылка на GitHub работы)

Меню сортировки представляет собой кнопки с типом, по которому происходит фильтрация. По нажатию на них создаётся GET запрос (в ссылку добавляется «?type=номер выбранного типа», страница обновляется, на сервере расшифровывается ссылка командой url.parse(req.url, true) и помещается в переменную. После чего переменная проверяется на наличие типа сортировки. Если тип не указан, выводится весь список товаров, если тип присутствует, то выводятся только те товары, которые относятся к выбранному типу фильтрации (Рис. 10).

Рис. 10

Также на странице имеется возможность помещать товары в корзину или в отложенные товары, но только в случае когда пользователь авторизован. Данный функционал должен быть реализован без перезагрузки страницы т.е. реактивно. Поскольку в данной работе не используется никакой фреймворк, весь функционал был реализован с помощью JavaScript, а именно асинхронных запросов.

По нажатию на кнопку в корзину или кнопку в форме сердца, запускаются соответствующие функции. В качестве аргумента передаётся id товара и id пользователя. Формируется тело POST запроса с вышеуказанными данными и посылается запрос на сервер. На сервере запрос обрабатывается соответствующим роутером и данные заносятся в базу данных отложенных товаров или корзины, в зависимости от того, на что нажал пользователь. В случае повторного нажатия на эти же кнопки, соответствующие строки в базе данных удаляются.

# 9. Личный кабинет

Личный кабинет является основополагающей частью данного проекта, поскольку цель данного проекта - коммерческая деятельность, а все заказы пользователь должен осуществлять через личный кабинет. Учитывая всё вышеперечисленное, личный кабинет должен включать в себя следующие страницы:

* Настройка и редактирование своих данных
* Страница корзины
* Страница отложенных товаров
* Страница заказов
  1. **Настройка и редактирование данных пользователя**

Поскольку для отправки заказа пользователю нам требуется информация о нем (адрес, ФИО и т.д.), требуется создать страницу, на которой пользователь может указать, обновить или отредактировать информацию о себе. Учитывая всё вышеперечисленное, была создана страница пользователя. На ней пользователь может сменить ФИО, поменять контактные данные, а также указать адрес доставки. Также на этой странице находятся ссылки на страницы корзины пользователя, отложенных товары и список его заказов.

## Корзина пользователя

На данной странице отображается список товаров, находящихся в корзине пользователя. Около каждого из товаров находится кнопка удаления этого товара из корзины. По нажатию на карточку товара, пользователя перенаправляет на страницу выбранного товара. Также реализована кнопка отчистки всей корзины пользователя. По нажатию на неё, посылается запрос на сервер, который отчищает все поля из таблицы cart в базе данных для конкретного пользователя. Таким образом корзина полностью отчищается. А пользователь видит надпись «корзина пуста». Если пользователь хочет сделать заказ, то он нажимает на кнопку «Заказать» и в случае наличия данных о пользователе (адрес доставки, контактные данные и т.д.), заказ считается успешным и заказ отображается на странице заказов пользователя.

* 1. **Отложенные товары**

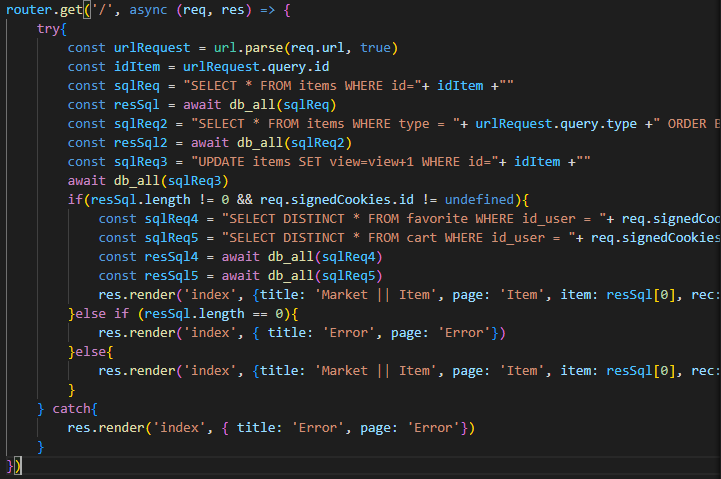
Страница отложенных товаров является важной частью любого сервиса, который занимается коммерческой деятельностью, поскольку данный функционал направлен на удобство пользователя и позволяет пользователю отложить понравившийся товар, чтобы его не потерять, заказать позже и т.д.

Учитывая всё вышеперечисленное, была создана страница отложенных товаров, которая работает схожим образом с корзиной. Отличия заключаются в том, что вместо кнопки «заказать» находится кнопка «поместить всё в корзину» с соответствующим функционалом.

**10. Страница товара**

Пользователю необходимо перед заказом оценить товар по достоинству, посмотреть фотографии, прочитать описание и т.д. Для этого была создана страница товара.

При переходе на данную страницу происходит посредствам ссылки и GET запроса на сервер, в котором находится id товара и его тип. По id сервер ищет в всю информацию о товаре в базе данных и посылает её на страницу. С помощью типа товара происходит поиск по похожим товарам и возвращает на страницу массив с данными о похожих товарах. Это нужно для того, чтобы пользователь мог посмотреть похожие товары и сделать выбор. (Рис. 11)

Рис. 11

**11. Выводы**

Основной целью данной курсовой работы являлось решение прикладных задач. В ходе выполнения проектной деятельности были закреплены навыки взаимодействия с базами данных SQL, были углублены знания Node JS, NPM модуля Express и JavaScript. Помимо этого, было уделено внимание тестированию и отладке программного кода, что является неотъемлемой частью разработки программного обеспечения. Получившиеся в результате разработки веб-приложение успешно выполняется. Все поставленные задачи были выполнены.

# Приложение

Ссылка на GitHub: <https://github.com/EV1L46rus/babaev>

Сайт: