\* эту пояснительную записку лучше всего смотреть сразу с перезентацией, поскольку все скриншоты будут предоставленны там, а здесь только текст с пояснениями.

# Использованные технологии

API написано с использованием **Spring Framework**. Для взаимодействия с базой данный используется **Spring Data JPA** – это библиотека, которая добавляет дополнительный уровень абстракции поверх ORM реализации (дополнительный удобный механизм для взаимодействия с сущностями базы данных, организации их в репозитории, извлечении и изменении данных). По умолчанию Spring Data JPA использует **Hibernate**, в качестве ORM провайдера (чтобы выполнять запросы). Spring Data JPA под капотом использует Hibernate, аHibernate под капотом использует JDBC API. Также использовался **Spring Boot** - еще один компонент Spring Framework,который упрощает конфигурацию и разработку. И **Spring Security** - компонент Spring Framework, который является стандартом для обеспечения безопасности в Spring приложениях, помогает реализовать функционал аутентификации и авторизации.

Для клиентской части приложения был использован **Angular** вместе с **Bootstrap.**

# Возможности неаутентифицированного пользователя

## Главная страница

С главной страници приложения пользователь может выполнить такие действия:

- Просмотреть продукты по выбранной категории (в левой панеле категорий)

- так же просмотр полной информации о любом из продуктов

- Перейти на страницу Filter (расширенный поиск продуктов с использованием фильтра)

- Аутентифицироваться

- Зарегистрироваться

- Найти продукт по названию в правой верхней палели поиска

## Продукты по выбранной категории

При переключении по категориям (в левой панеле) у нас будут отображаться продукты, относящиеся к этой категории с пагинацией. Нужно заметить, что будут показаны только доступные продукты (isAvailable = true) и те, которые есть в наличии (amountInStorage > 0).

Метод в репозитории, который будет нам возвращать нужную страицу с нужными продуктами:

Page<Product> findByCategoryIdAndIsAvailableAndAmountInStorageGreaterThan(

int categoryId,

boolean isAvailable,

int amount,

Pageable pageable);

Здесь мы возвращаем нужную страницу со списком продуктов благодаря переданному в параметрах объекту Pageable, который включает в себя такие данные как номер текущей страницы и кол-во продуктов на странице. Имя метода должно быть четко определено, порядок слов и ключевые слова должны быть указаны в соответствии с документацией, названия “CategoryId”, “IsAvailable”, “AmountInStorage” должны строго соответствовать названиям полей в сущности Product. Так же передаваемые переменные должны быть в таком же порядке как и названия этих полей и иметь такой же тип данных, само же название переменной в праметрах может быть произвольным. Исходя из этих всех данных Spring Data JPA формирует соответствующий sql запрос, данные из таблицы маппит в сущности и возвращает нам нужную страницу с перечнем этих сущностей, кторую мы в последствии отправляем на клиент и отображаем.

## Просмотр полной информации о продукте

При нажатии на иконку глаза мы можем просмотреть полную информацию о продукте, где будет вся доступная пользователю информация. Так же под основной информацией мы видим список продуктов, которые «могут заинтересовать пользователя». На самом деле это список из продуктов с той же категории, что и продукт, который просматривает пользователь, но список отсортировам по кол-ву лайков (количество людей, которые добавили этот продукт в список своих избранных продуктов), и отсортированны они так, что бы самые непопулярные или новые (у новых тоже мало лайков) были самые первые в списке, и мы взяли первые 4 продукта с этого отсортированного списка и показали только их (кол-во приходит к нам с клиента, так что можем в дальнейшем это количество изменить, если что).

## Страница Filter

На этой странице пользователь может увидеть расширенный поиск продуктов с использованием фильтра, здесь мы можем отфильтровать проодукты по стране производства, по бренду, цене, показать только продукты по скидке или отсортировать так, что бы самые популярные продукты были первыми.

Алгоритм фильтрации: Обычно самый первый и простой алгоритм, который приходит в голову это вытащить все продукты из БД, затем проверять каждую переменную и если она не пустая (фильтр задействован) – мы удаляем продукты, которые совпадают по этому фильтру (например если переменная country != “” - удаляем все продукты у которых поле country соответствует значению переданной переменной). Но этот алгоритм имеет недостаток – в любом случае нам нужно выгрузить из БД абсолютно все продукты, а потом только удалять лишнее (а продуктов может быть миллион или более), да и фильтры могут быть задействованы все, а значит что из миллиона потом придется удалить процентов 95. Это не эффективно и очень ресурсоемко. Вместо того что бы вытягивать сразу все товары из БД мы сначала проверим переменную “brand” и если этот фильтр задействован – вытащим пвсе продукты, которые соответствуют этому бренду (то что бы выбрали именно бренд первым важно, по скольку это поможет максимально уменьшить объем выборки), затем проверяем по порядку переменные country, discount и price, удаляя совпадения уже от этой выборки. Если переменная brand пустая и не участвует в выборке, мы ее игнорируем и пытаемся отобрать товары по другим критериям. Если не выбранно ничего – мы по крайней мене фильтруем по цене – в общем суть в том, что бы по-любому вытаскиваем не все товары сразу, а только часть их. После чего в конце сортируем по популярности, если нужно и возвращяем резулятат с пагинацией (нужную страницу и нужное ко-во элементов).

## Панель поиска

Тут все просто – находим продукты, в названии которых находится введенная в поиске подстрока, игнорируя регистр.

## Логин и регистрация

На этих страницах останавливаться особого смысла нет, там просто формы входа и регистрации. Но на счет входа, я использовал для аутентификации Spring Security, после успешного входа пользователю выдается jwt token, на клиенте мы этот токен сохраняем в local storage и потом с каждым запросом передаем этот токен в хедере, а Spring Security конфигурируем таким образом что бы он перехватывал каждый запрос к апи, и если запрос идет к тому методу в контрллере, к которому может иметь доступ только аутентифицированный пользователь то Spring Security, а именно jwt-filter должен проверить хедеры на наличие токена и сам токен должен быть валидным и не просроченым – тогда пользователь получит доступ к этому методу.

# Как изменится страничка, если мы залогинимся

Теперь мы можем добавлять товары в корзину (так же менять их количество или удалить с корзины) и добавлять продукты в избранные (или убирать) – все действия сразу же меняют данные в БД.

Так же, как только мы залогинимся – мы увидим две новые доступные ссылки в хедере – это профиль пользователя и корзина. Начнем с корзины.

# Корзина

Как мы видим, в корзине у пользователя уже что-то есть (корзина хранится в БД так что мы будем абсолютно уверены что пользователь не потеряет свои товары, через какое время он бы не зашел на сайт). Откроем ее и увидим список добавленных товаров. Здесь мы можем изменить кол-во товара (цена его сразу же будет прересчитана, как и общая цена покупки). Колличество товара и цена будет сразу же измененна в БД, а общая цена – это вычисляемое значение (мы всегда можем сложить все цены товаров и получить это значение), по этому общую цену мы нигде сохранять не будем, а будем только на клиенте пересчитывать с каждым изменением кол-ва какого-либо товара (что бы не травмировать лишний раз базу данных).

Так же мы можем удалить товар из корзины, либо удалить все товары – в обоих случаях так же удалятся они и в БД.

Так же стоит заметить что этот бэйдж в header над корзиной постоянно меняется при каждом изменении товаров в корзине, либо при их удалении – мы это значение бэйджа храним в local storage (записываем его туда при аутентификации пользователя и изменяем каждый раз в local storage, и привязанно это значение в бэйдже именно к значению в local storage, так что оно на бэйдже обновляется автоматически)

# Делаем заказ

Для того что бы сделать заказ обязательно нужно ввести номер телефона. Почему этот номер не хранится в БД? Потому что пользователь не обязательно себе может заказывать эти продукты и в этом поле нужно указать именно номер того человека, который будет получать продукты (это могут быть родители, дедушка или бабушка, родственники, кто угодно). После нажатия на кнопку «Подтвердить заказ» мы увидим что все товары были удалены с корзины (они так же удаены из БД – из таблицы basket), мы так же видим сообщение об успехе и ссылку, где можно просмотреть список наших заказов и отследить статус этого заказа. Чтож, перейдем по ней.

# Профиль пользователя / его заказы

Когда мы попадаем в профиль пользователя мы сразу видим карточку с полной информацией о пользователе, и открытую вкладку с его заказами. Как видим в карточке – заказов 12, но они не выгруженны все ниже, они выгружаются с пагинацией по скроллу вниз – это бесконечный скролл. Получается что когда мы скроллим вниз страницу, заказы выгружаются еще перед тем как мы их увидим и у нас будет складываться впечатление что они там были всегда. В информации о заказе мы видим дату, когда заказ был сделан (кстати все заказы отсортированны по дате – последний заказ - был показан на предыдущем слайде), так же видим цену заказа и статус (всего статуса – 4, это: «обрабатывается», «собирается» (когда сборщик будет собирирать эти товары), дальше статус «ожидает оплаты» и «завершен»).

Если мы просмотрим информацию о заказе (нажмем на значек глаза) – мы увидим список всех продуктов, которые находятся в этом заказе, и каждый из этих продуктов мы так же сможем просмотреть (модальное окно с полной информацией о продукте).

# Избранные продукты

На вкладке избранных продуктов все подукты отображаются по категориям для удобства, ведь избранных продуктов может быть много и пользователю будет сложно найти нужный товар, что бы заново добавить его в корзину, например. Так же список продуктов имеет пагинацию, конечно же. Если мы снимем лайк – продукт пропадет с этого списка (и с базы данных так же).

# Редактирование информации пользователя

Ну и вкладка с возможностью редактирования информации пользователя, здесь важно, что если пользователь изменит username – он сразу разлогинется, поскольку jwt token формируется на основании username и если username меняется то токен автоматически становится невалидным и нам нужно разлогинить пользователя, что бы избежать ошибки. Как только пользователь залогинется заново – ему будет выдан новый валидный токен на основании нового username и он сможет дальше без проблем пользоваться приложением.

# Роль администратор

Когда мы залогинимся под администратором у нас появляется возможность редактировать продукты в окне просмотра полной информации об этих подуктах. Достаточно большая форма редактирования получилась, потому что много связанных таблиц с таблицей продуктов и нужно много информации ввести. При выборе, например, новой категории администратор может не только выбрать новую категорию из выпадающего списка всех доступных категорий, но и сразу же здесь добавить новую категорию, которая сохранится в базе данных и затем уже образуется связь с этим продуктом. Из-за того что база данных получилась максимально нормализованной то при редактированиии продукта, конечно, пришлось заморочиться, но что поделать.

# Админ панель / управление пользователями

О вкладке добавления продукта говорить нет смысла, там такая же форма как и при редактировании и логика практически такая же. Для добавления и редактирования был использован один и тот же общий компонент в ангуларе.

На вкладке управления пользователями админ может найти пользователя по юзернейму, просмотреть его полную информацию и кол-во заказов, заблокировать или дать ему права администратора. При блокировке пользователя, он конечно же, все равно может заходить на сайт и делать заказы, потому что глупо запрещать человеку давать нам деньги, это же бизнес. Однако поле isBlocked все же меняется в базе данных на true и мы как-то можем это использовать в будущем, например блокировать рекламные сообщения пользователю или что-то подобное. Ну в общем главное что бы можно было что-то заблокировать, а что и зачем – с этим уже можно и потом разобраться.

# Следим за кол-вом продуктов на складе

Администратору важно понимать когда продукты заканчиваются и сколько осталось конкретного товара что бы вовремя сделать заказ у поставщика и пополнить запасы. На этой вкладке будут показаны товары, которых осталось на складе меньше 50 единиц. Этот список с пагинацией, конечно же.