**Поволжский Государственный Университет Телекоммуникаций и Информатики**Кафедра «Программная инженерия»

|  |  |
| --- | --- |
| Сдана на проверку  «\_\_» \_\_\_\_2024 г. | Допустить к защите  «\_\_» \_\_\_\_\_ 2024 г.  Защищена с оценкой  «\_\_\_» \_\_\_\_\_\_\_\_ 2024 г. |

**КУРСОВАЯ РАБОТА**По дисциплине: «Прикладное программирование»  
На тему: «Разработка клиент-серверного web-приложения InventoryVista»  
  
Пояснительная записка

|  |  |
| --- | --- |
| Студент группы ИВТ-11\_\_\_\_\_\_\_ Вязков А. А.  (роспись) (ФИО)  Руководитель \_\_\_\_\_\_\_\_ к.т.н., доц. Ахметшина Э. Г.  (роспись) (ФИО) | 0 210225 0  (№ зачетной книжки) |

Самара 2024

Рецензия

Содержание

[Описание предметной области. Актуальность 4](#_Toc1183589675)

[Описание программы. Общие сведения 6](#_Toc282498409)

[Функциональное назначение 6](#_Toc1021157000)

[Физическая модель базы данных 11](#_Toc819131022)

[Диаграмма классов 12](#_Toc498394935)

[Диаграмма компонентов 13](#_Toc1945372171)

[Диаграмма вариантов использования 15](#_Toc1214014259)

[Диаграмма последовательности 16](#_Toc1411144081)

[Демонстрация работы приложения 18](#_Toc130837166)

[Список используемых источников 32](#_Toc1897641439)

[Приложение А – Листинг программного кода 33](#_Toc1249231300)

Описание предметной области. Актуальность

В современной динамичной экономике и технологической среде, где потребители становятся все более требовательными, системы учета и управления запасами становятся стратегическим инструментом для предприятий. В этом контексте, проект "InventoryVista" представляет собой ключевое решение для организаций любого масштаба.

По мере увеличения объемов товарооборота и разнообразия продуктов, внедрение эффективной системы инвентаризации товаров становится неотъемлемым элементом для бизнеса. "InventoryVista" способствует оперативной реакции на изменения спроса, предоставляет точные данные для принятия стратегических решений и создает условия для повышения уровня обслуживания клиентов.

Проект акцентирует внимание на стремлении предприятий к оптимизации своей деятельности. "InventoryVista" автоматизирует процессы учета, минимизирует ошибки и снижает риски, связанные с управлением запасами и сроками годности товаров.

Дополнительно, "InventoryVista" отвечает требованиям соблюдения норм и стандартов в сфере управления складскими запасами. Система инвентаризации товаров автоматизирует контроль за соблюдением сроков годности, условий хранения и других регуляторных требований, что критически важно для предприятий, работающих с продуктами, подлежащими специальным требованиям.

Таким образом, система "InventoryVista" ориентирована не только на повышение эффективности бизнес-процессов, но и на соответствие современным стандартам, что делает ее востребованным инструментом для предприятий, стремящихся к совершенствованию своей деятельности в условиях быстро меняющегося бизнес-окружения.

Описание программы. Общие сведения

Приложение – веб-ресурс. Установка не требуется.

Языки программирования: Java, JavaScript.

Фреймворки/Библиотеки: Spring, React.js

Протокол взаимодействия: HTTP

СУБД: PostgreSQL

Среда разработки: Intellij IDEA.

Объем проекта: 663,3 МБ (вместе с исходным кодом).

Исходный код (Java классы и компоненты React): 5151 строк.

Функциональное назначение

Проект "InventoryVista" разработан с целью эффективного учета товаров на складе и предоставления пользователям возможности полноценного управления информацией о продукции. В основе функционала приложения лежат задачи, направленные на упрощение и оптимизацию процессов управления запасами. Проект охватывает широкий спектр функций, среди которых выделены следующие:

1. **Добавление Товаров: возможность добавления новых товаров в инвентарь, с указанием основных характеристик, таких как наименование, количество и срок годности.**
2. **Редактирование Данных: пользователи имеют доступ к функции редактирования информации о товарах, что обеспечивает актуальность и точность данных.**
3. **Отслеживание Товаров: встроенные механизмы позволяют отслеживать перемещение товаров, изменения в количестве и сроке годности.**
4. **Управление Сроками Годности: функционал** по контролю сроков годности товаров помогает предотвращать списание просроченной продукции и обеспечивает безопасность для потребителя.

Описание используемых технологий

**React.js** — это библиотека JavaScript для разработки интерактивных пользовательских интерфейсов. Созданная Facebook, она позволяет эффективно строить многокомпонентные веб-приложения, где каждый компонент управляет своим состоянием. React также обеспечивает виртуальный DOM, что повышает производительность при обновлении данных.

**Spring Boot** — фреймворк для разработки веб-приложений на языке программирования Java. Он предоставляет удобные средства для создания микросервисов и веб-приложений с минимальными усилиями разработчика. С Spring Boot легко создавать самостоятельные, автономные приложения, что ускоряет процесс разработки.

**RESTful (Representational State Transfer)** — стиль архитектуры взаимодействия компонентов распределенного приложения. Он основан на принципах, описанных в ограниченном и однозначном интерфейсе, ресурсах, представлении ресурсов и управлении состоянием. RESTful API предоставляет простой и эффективный способ связи между клиентом и сервером.

**PostgreSQL** — это мощная система управления реляционными базами данных с открытым исходным кодом. Она предоставляет надежное хранение данных и поддерживает множество расширений, что делает ее превосходным выбором для приложений, требующих структурированные данные. PostgreSQL обеспечивает высокую производительность и расширенные возможности запросов.

Этот технологический стек обеспечивает надежность, производительность и гибкость при разработке приложения "Инвентаризация товаров". Все компоненты взаимодействуют гармонично, обеспечивая пользовательскому опыту высший уровень удовлетворения.

Логическая модель базы данных

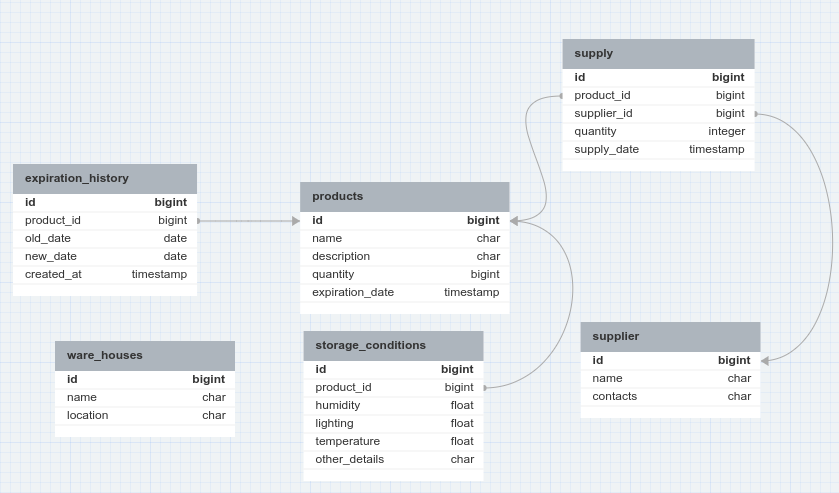


Рис. 1- Логическая модель базы данных

Таблица 1 – Описание сущностей логической модели данных

|  |  |
| --- | --- |
| **Название** | **Назначение** |
| Products | Представляет отдельный товар в системе |
| Склад | Отражает информацию о месте хранения товарных запасов |
| Поставщик | Содержит данные о компаниях-поставщиках продукции |
| Поставка | Представляет информацию о поступлении товаров на склад |
| История годности | Содержит данные об изменениях сроков годности продукта |

Продолжение табл. 1

|  |  |
| --- | --- |
| Условия хранения | Включает в себя информацию о требованиях к условиям хранения продукта |

**Products**

* id (Primary Key): Уникальный идентификатор товара
* name: Наименование товара
* quantity: Количество единиц товара в инвентаре
* expiration\_date: Дата окончания срока годности

**StorageConditions**

* id (Primary Key): Уникальный идентификатор условий хранения
* product\_id (Foreign Key): Ссылка на товар из таблицы Products
* temperature: Температура хранения товара
* humidity: Влажность, влияющая на сохранность товара
* lighting: Уровень освещения, важный фактор для определенных продуктов

**Expiration\_History**

* id (Primary Key): Уникальный идентификатор истории срока годности
* product\_id (Foreign Key): Ссылка на товар из таблицы Products
* old\_expiration\_date: Предыдущая дата срока годности
* new\_expiration\_date: Новая дата срока годности
* created\_at: Дата создания записи о изменении срока годности

**Ware\_Houses**

* id (Primary Key): Уникальный идентификатор склада
* name: Название склада
* location: Местоположение склада

**Suppliers**

* id (Primary Key): Уникальный идентификатор поставщика
* name: Наименование поставщика

**Supply**

* id (Primary Key): Уникальный идентификатор поставки
* product\_id (Foreign Key): Ссылка на товар из таблицы Products
* warehouse\_id (Foreign Key): Ссылка на склад из таблицы Ware\_Houses
* supplier\_id (Foreign Key): Ссылка на поставщика из таблицы Suppliers
* supply\_date: Дата поставки товара
* quantity: Количество поставленного товара

**Связи между таблицами:**

* Products → StorageConditions: Отношение один к одному (One-to-One) через внешний ключ product\_id
* Products → Expiration\_History: Отношение один ко многим (One-to-Many) через внешний ключ product\_id
* Products → Supply: Отношение один ко многим (One-to-Many) через внешний ключ product\_id
* Suppliers → Supply: Отношение один ко многим (One-to-Many) через внешний ключ supplier\_id

Физическая модель базы данных

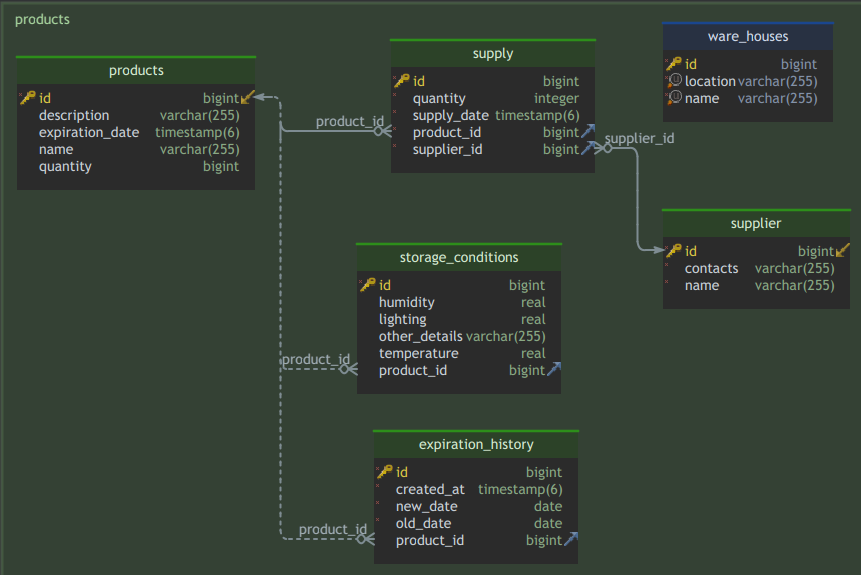


Рис. 2 - Физическая модель базы данных

Диаграмма классов

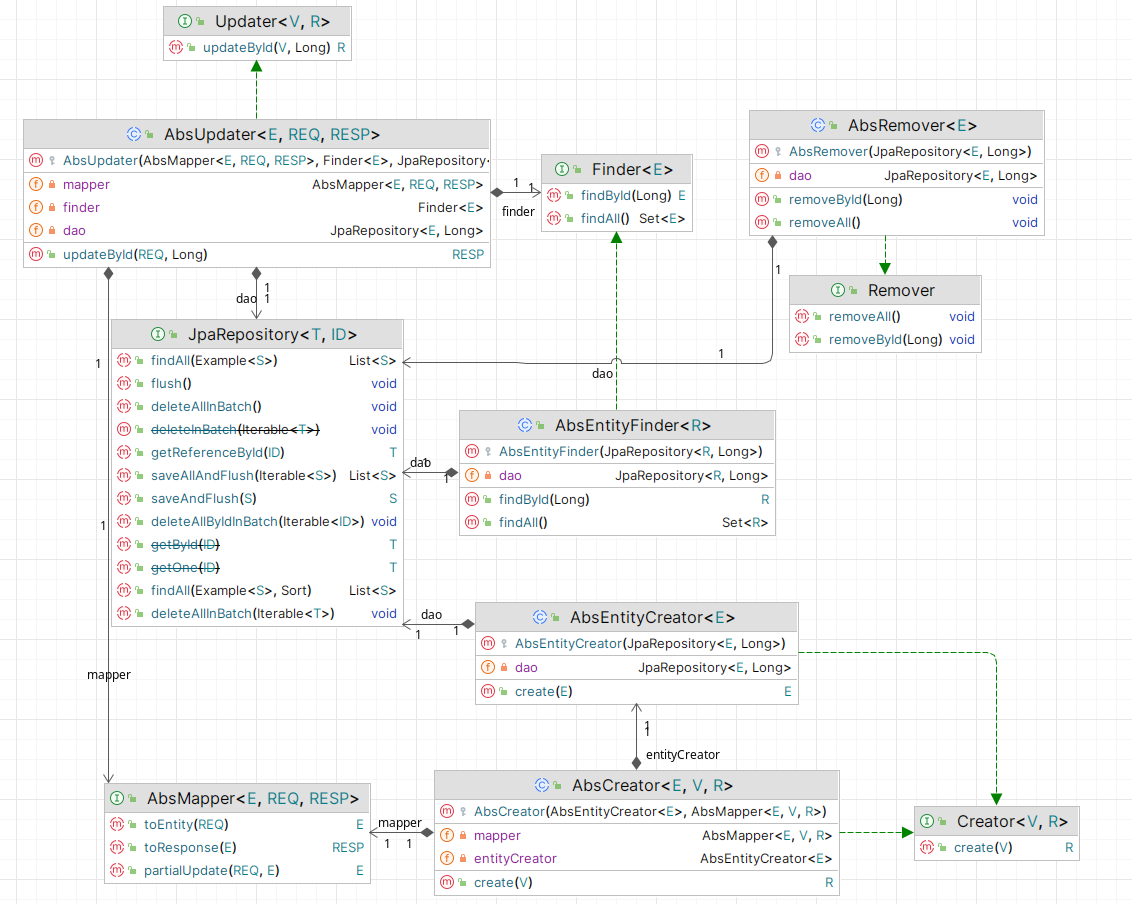


Рис. 3 - Диаграмма классов

Диаграмма компонентов

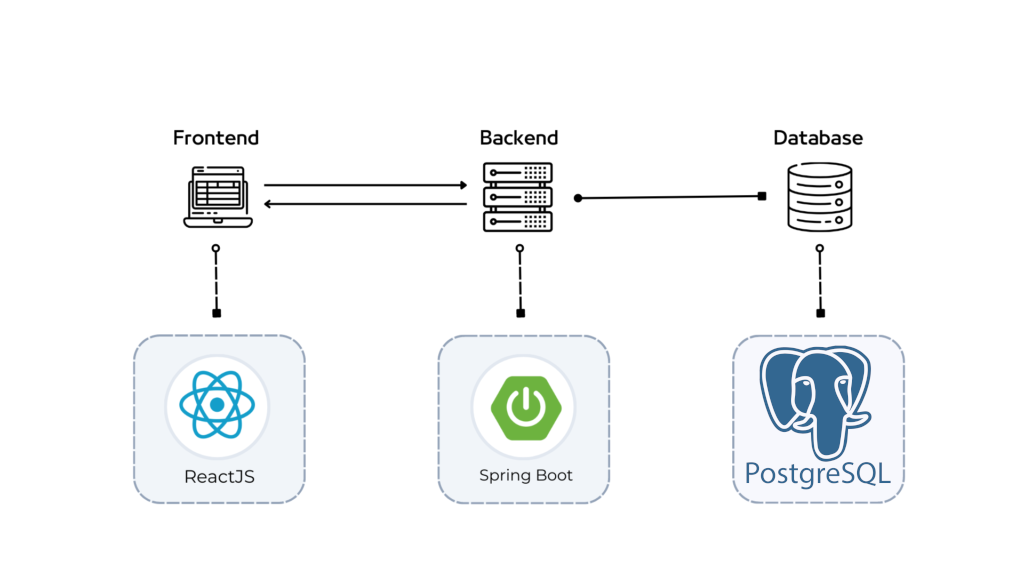


Рис. 4 — Диаграмма компонентов системы

В нашей системе JSON используется для передачи структурированных данных между компонентами. Например, фронтэнд может отправлять HTTP-запросы с данными в формате JSON на бэкенд, который в свою очередь может возвращать ответы в этом формате. Данные, такие как пользовательские настройки или результаты запросов к базе данных, могут быть представлены в виде JSON.

Fetch API используется на фронтэнде для взаимодействия с бэкендом. Например, при загрузке данных или отправке обновлений, фронтэнд может использовать Fetch API для выполнения асинхронных HTTP-запросов к API на бэкенде. Полученные данные в формате JSON могут быть затем обработаны и отображены пользователю.

Hibernate используется на бэкенде для взаимодействия с базой данных. Компоненты бэкенда, такие как сервисы или репозитории, могут использовать JPA-аннотации для маппинга Java-объектов на записи в базе данных. Например, сущность "Пользователь" может быть сохранена в таблице базы данных через Hibernate.

HTTP используется для обеспечения взаимодействия между фронтэндом и бэкендом. Фронтэнд отправляет HTTP-запросы на бэкенд для выполнения операций, таких как получение данных или отправка обновлений. Бэкенд обрабатывает эти запросы и возвращает соответствующие HTTP-ответы, содержащие данные в формате JSON.

Диаграмма вариантов использования

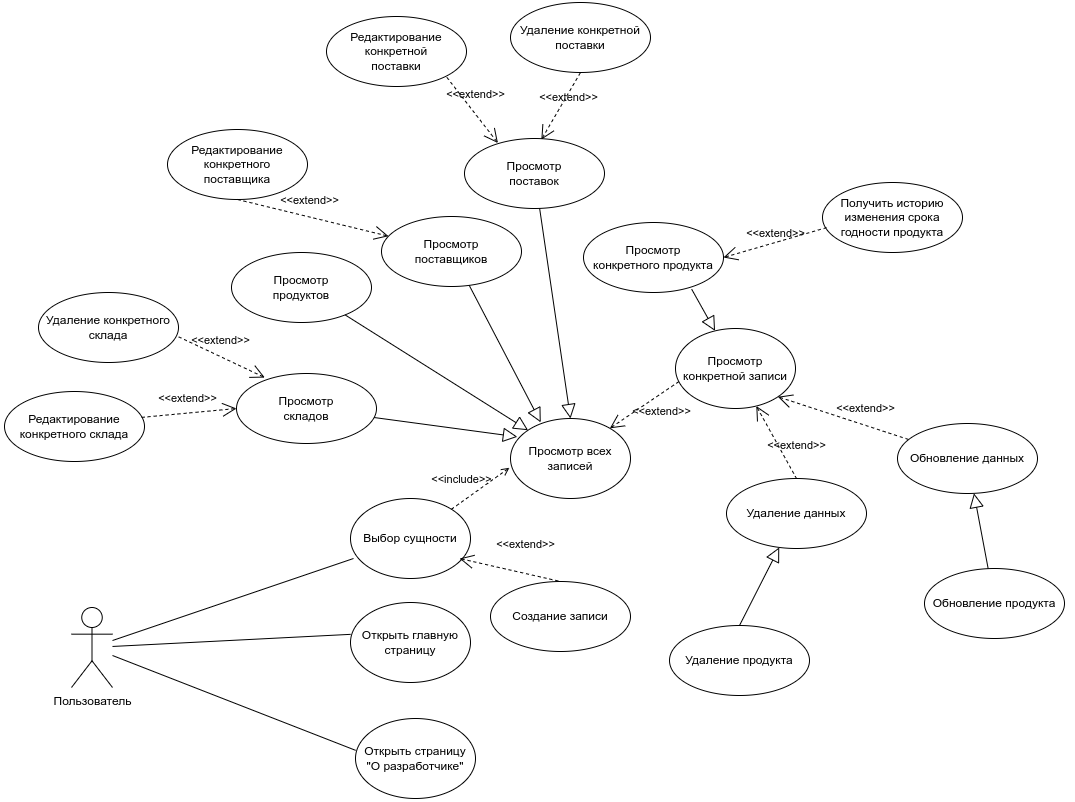


Рис. 5- Диаграмма вариантов использования

Диаграмма последовательности

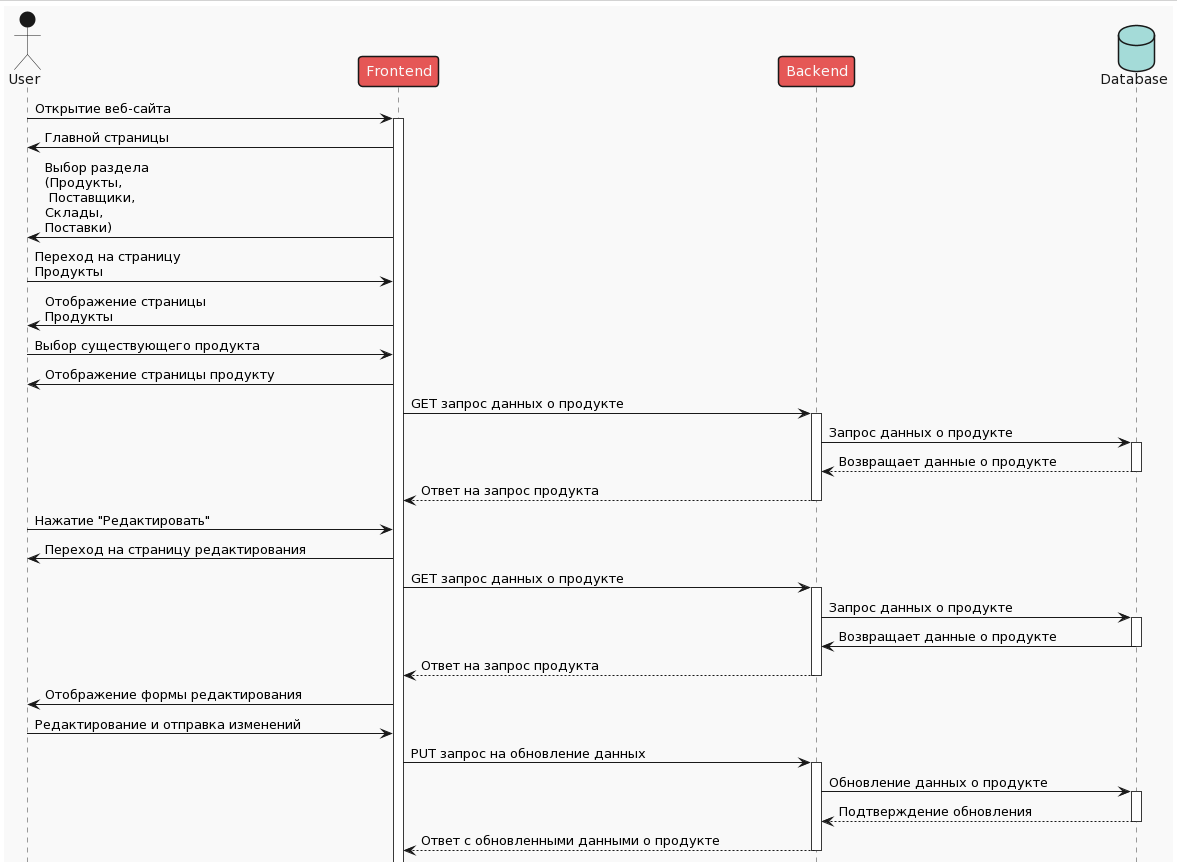


Рис. 6.1 - Диаграмма последовательности

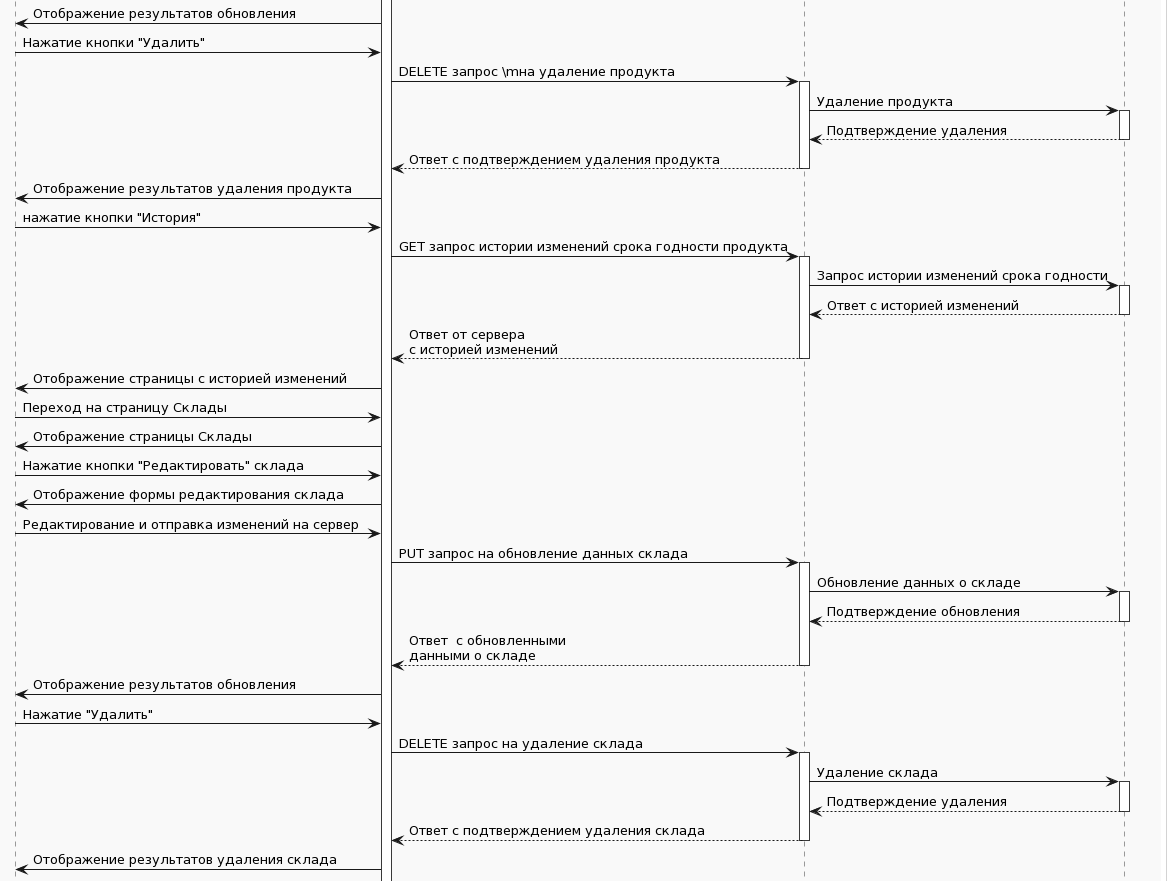


Рис. 6.2 - Продолжение диаграммы последовательности

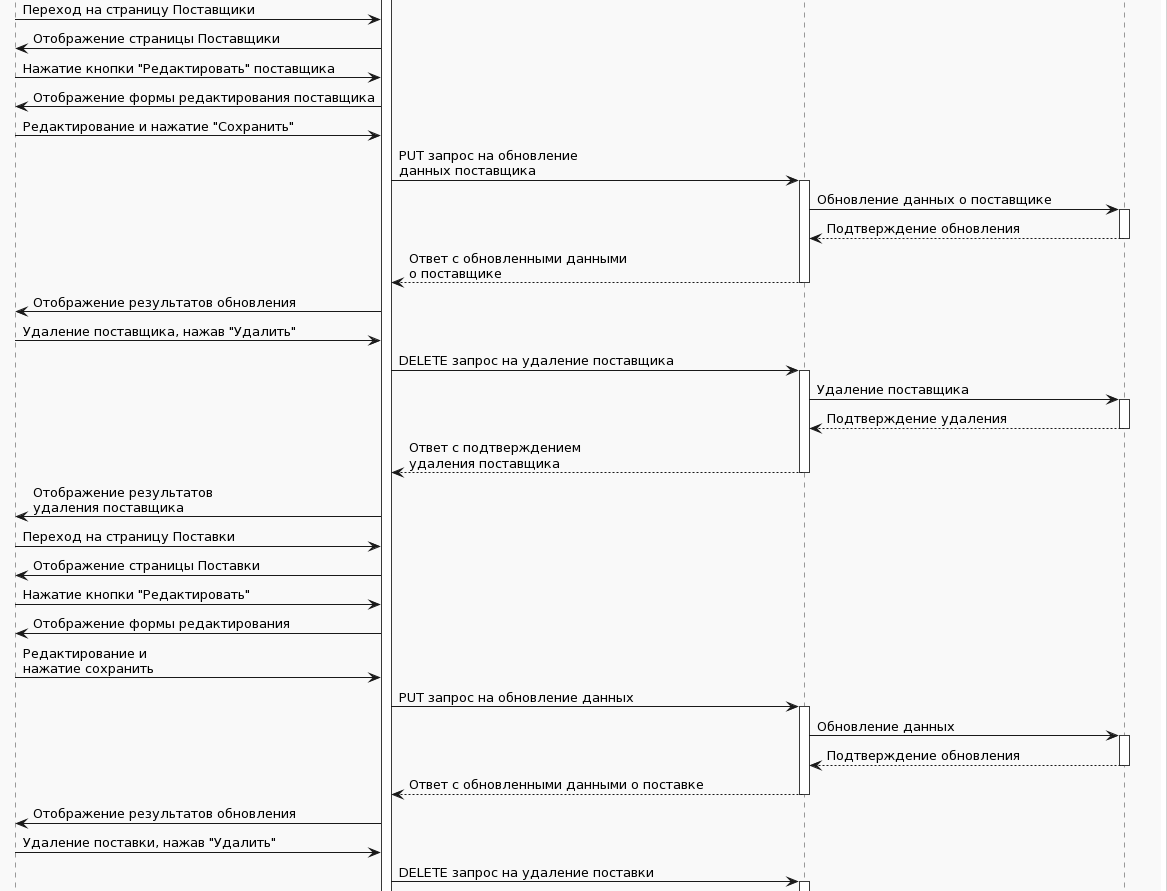


Рис. 6.3 - Продолжение диаграммы последовательности

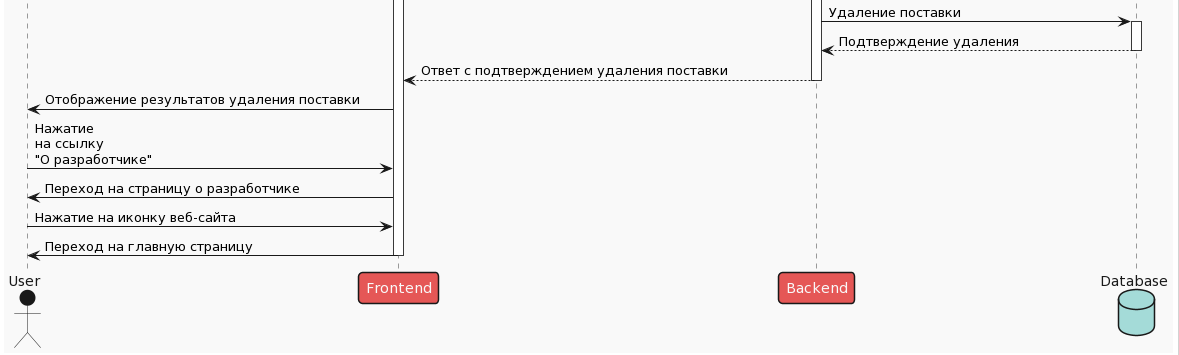


Рис. 6.4 - Продолжение диаграммы последовательности

Демонстрация работы приложения

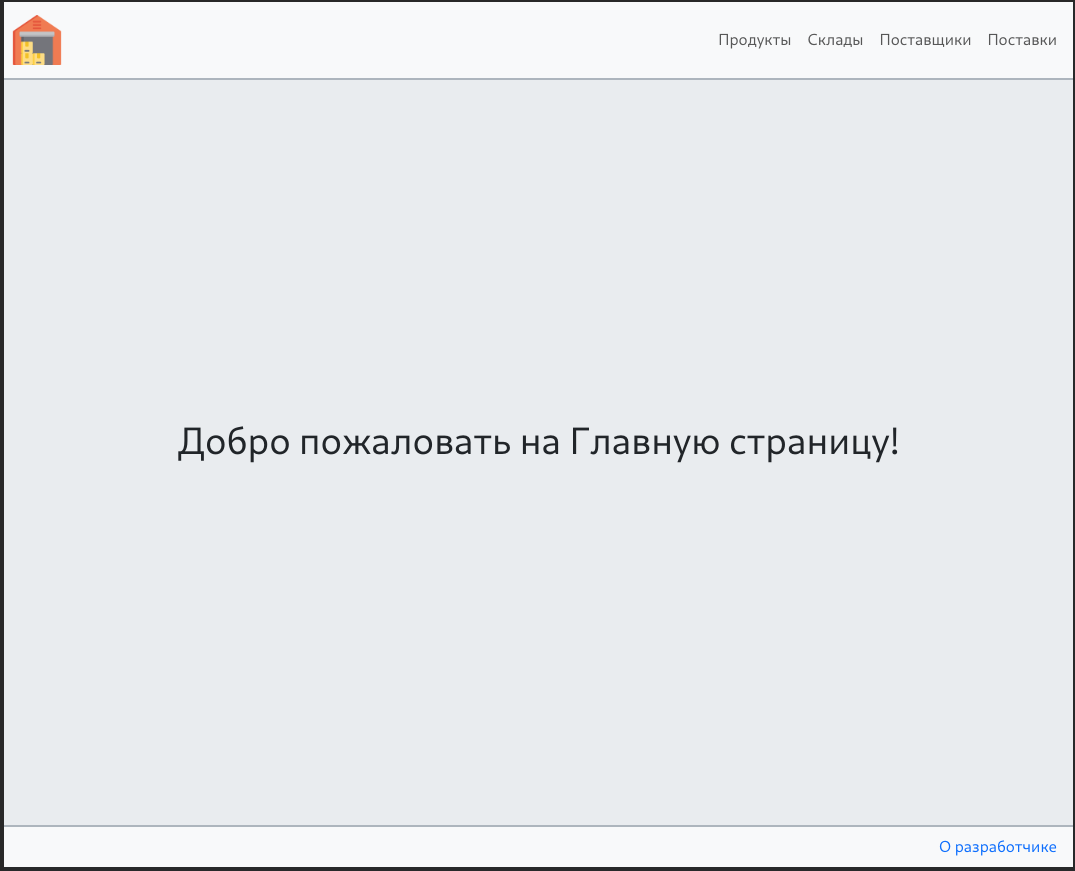


Рис. 7.1 - Главная страница сайта

При открытии приложения в браузере пользователь попадает на главную страницу. (Рис. 7.1)

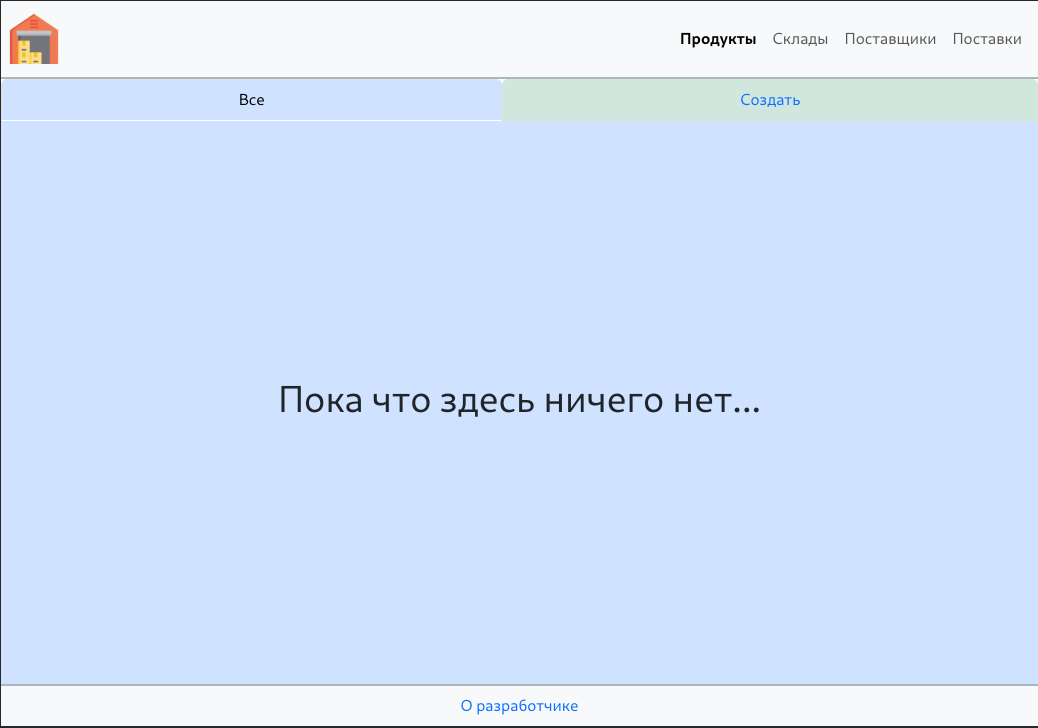


Рис. 7.2 - Страница продуктов - вкладка “Все”

При переходе на страницу продуктов, мы переходим во вкладку все, где выводится список существующих продуктов. Т.к. продуктов на данный момент нет, выводится соответствующее сообщение. (Рис. 7.2)

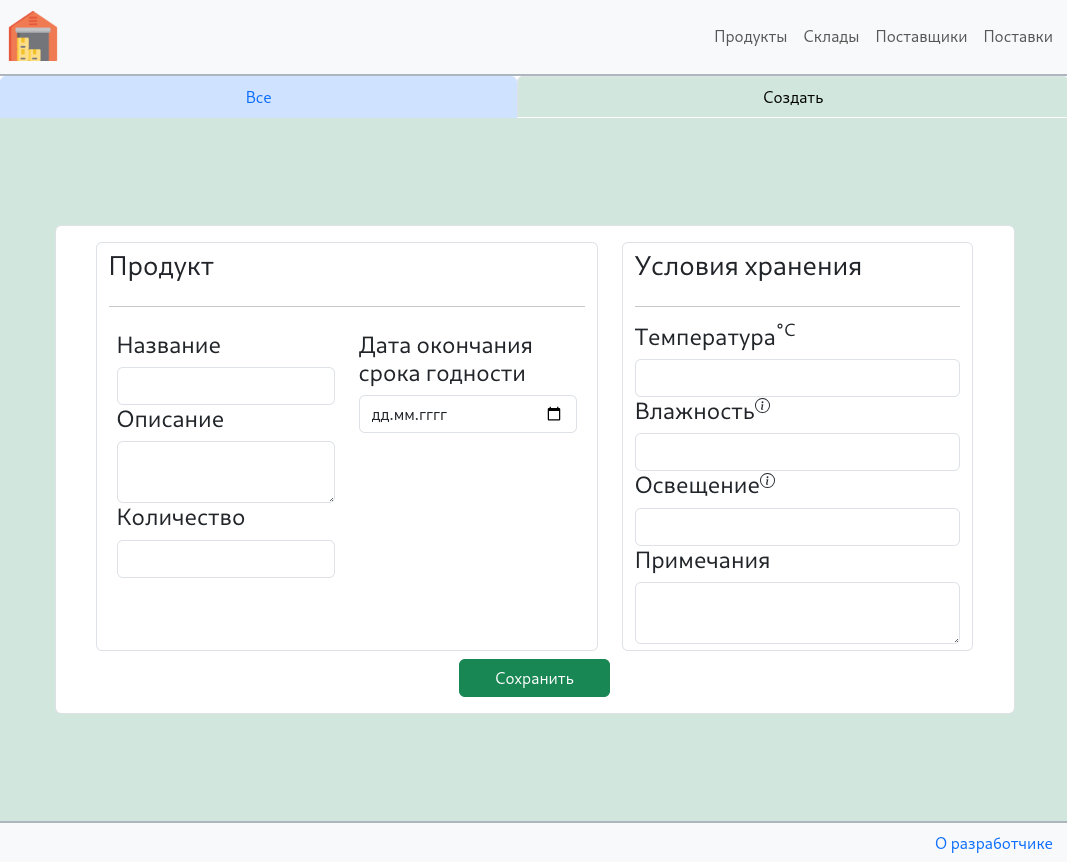


Рис. 7.3 - Страница продуктов - форма создания продукта

При переходе на вкладку «Создать», нас приветствует форма для создания нового продукта. (Рис. 7.3)

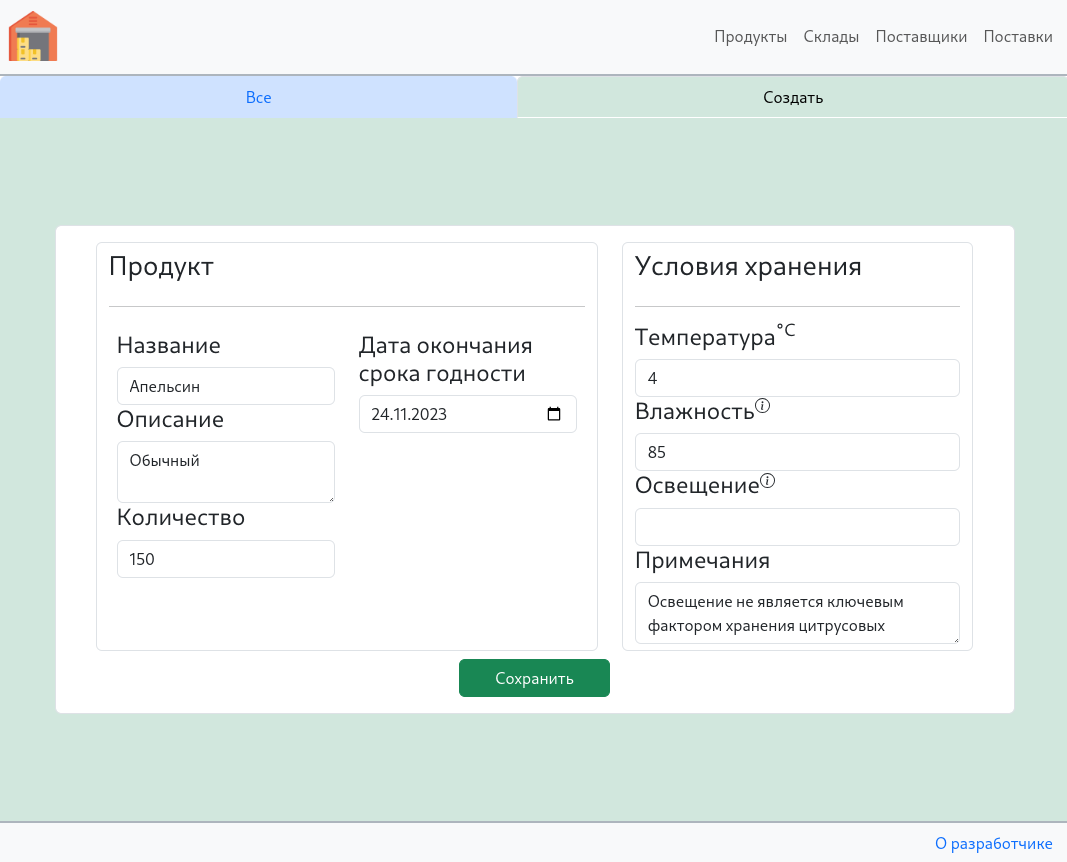


Рис. 7.4 - Страница продуктов - заполненная форма создания

Заполнили форму данными о новом товаре. (Рис. 7.4)

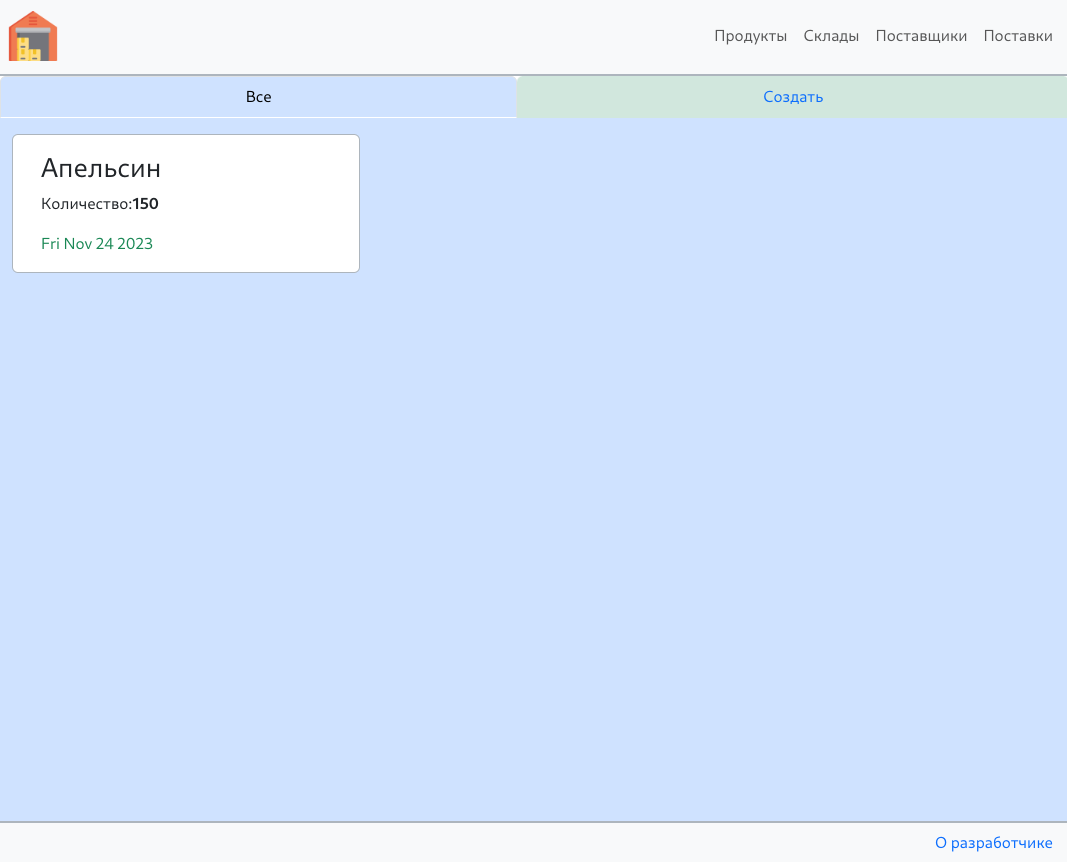


Рис. 7.5 - Страница продуктов - после создания продукта

После заполнения формы нажали на кнопку «Сохранить» и пользователя отправляет на вкладку с добавленными продуктами. (Рис. 7.5)

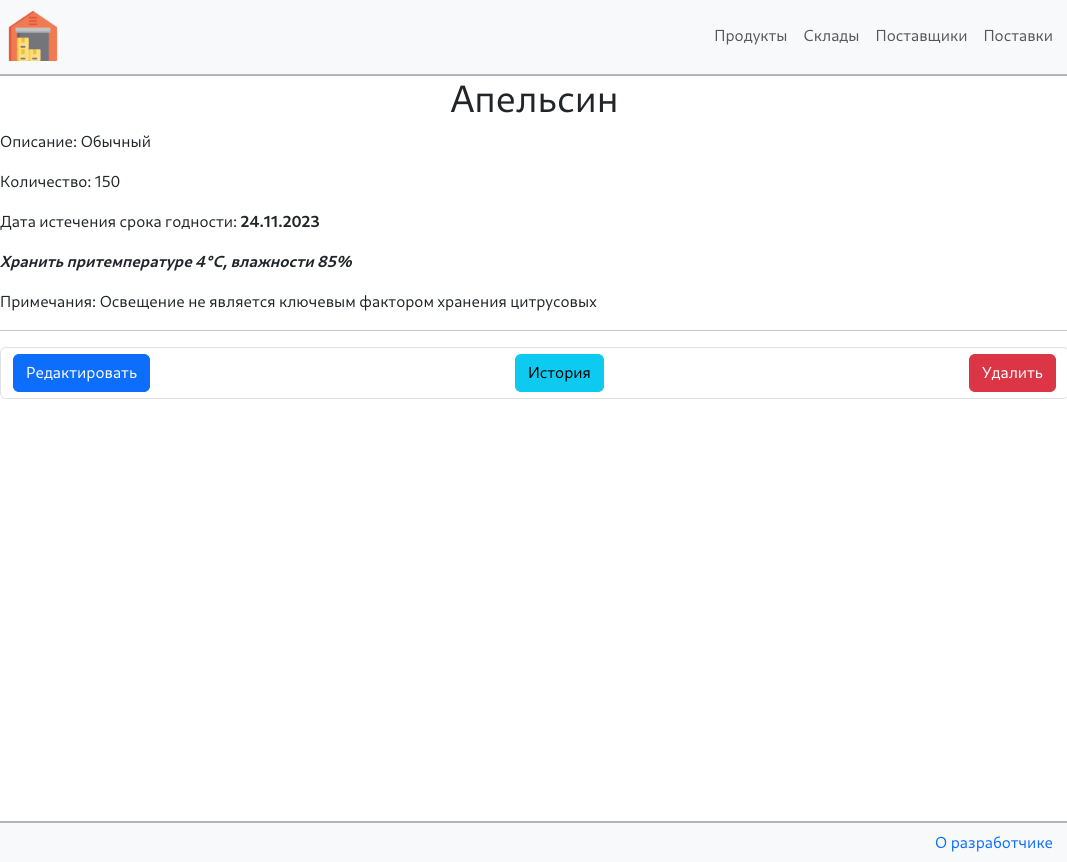


Рис. 7.6 - Страница созданного продукта

После создания пользователь может подробнее ознакомиться с характеристиками товара, перейдя на его страницу. (Рис. 7.6)

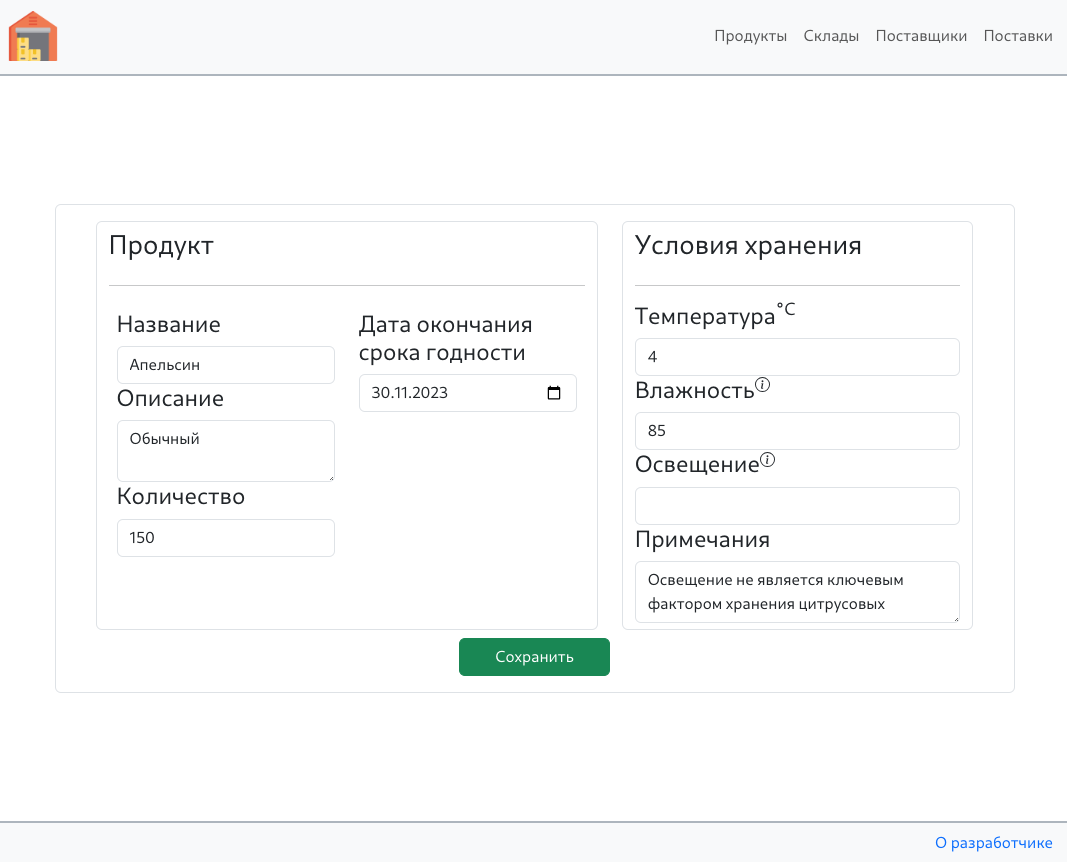


Рис. 7.7 - страница редактирования продукта

Из страницы описания товара мы можем перейти в процесс редактирования товара, нажав на кнопку «Редактировать». В последствии мы увидим, заполненную исходными данными, форму. (Рис. 7.7)

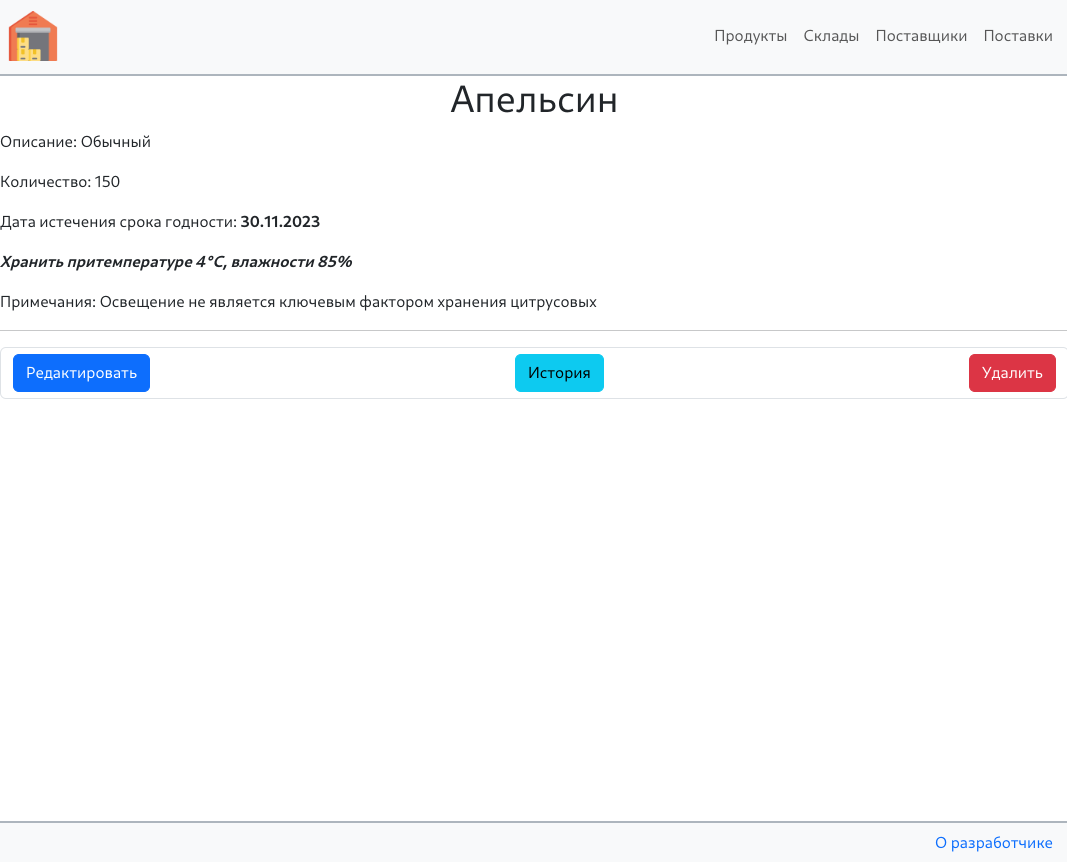


Рис. 7.8 - продукт после редактирования

После редактирования мы снова переходим на страницу продукта и убеждаемся, что редактирование прошло успешно. (Рис. 7.8)

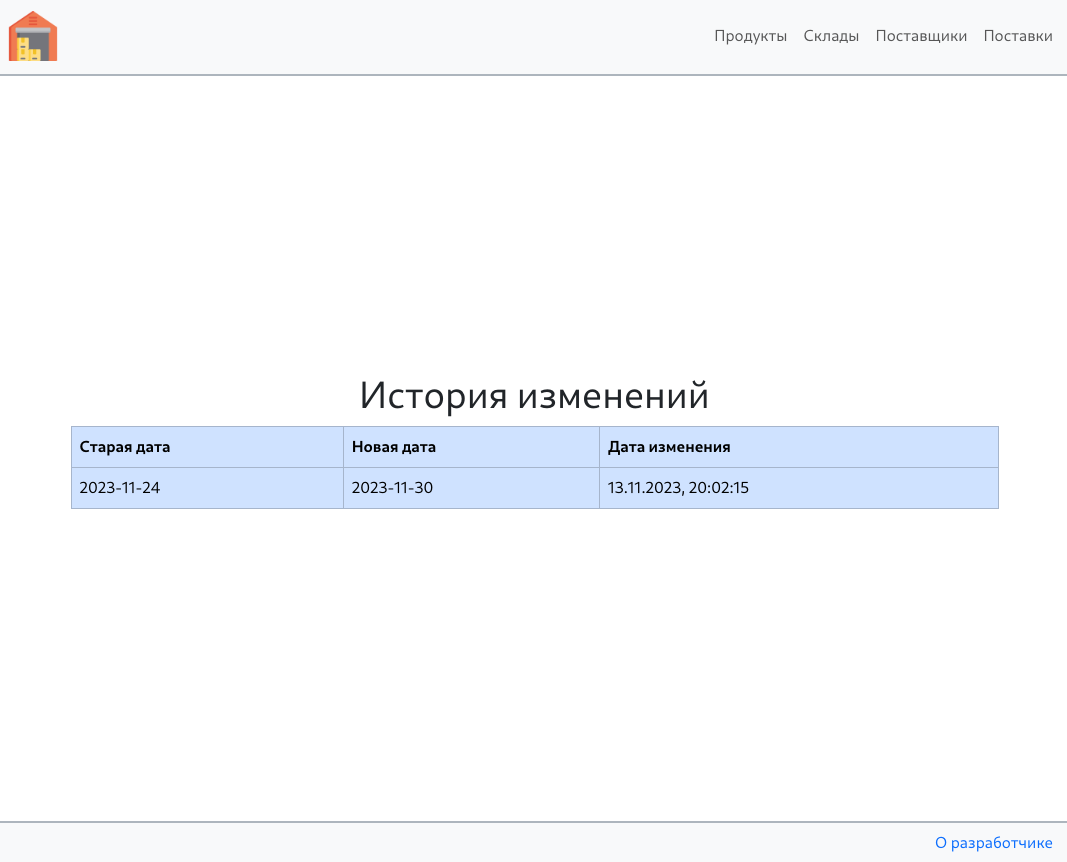


Рис. 7.9 - Страница истории изменения истечения срока годности продукта

После изменения даты истечения срока годности продукта мы можем просмотреть историю изменения, нажав на кнопку «История». (Рис. 7.9)

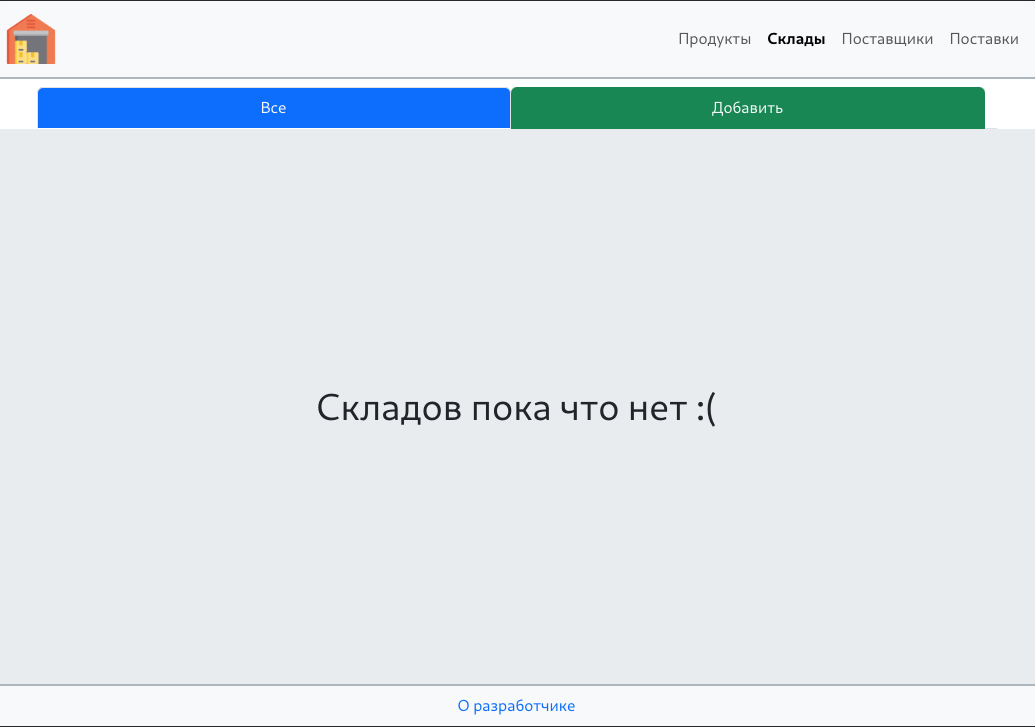


Рис. 7.10 - Cтраница складов

Перейдя на страницу «Склады» из панели навигации, мы увидим страницу со списком складов, который на данном этапе пуст. (Рис. 7.10)

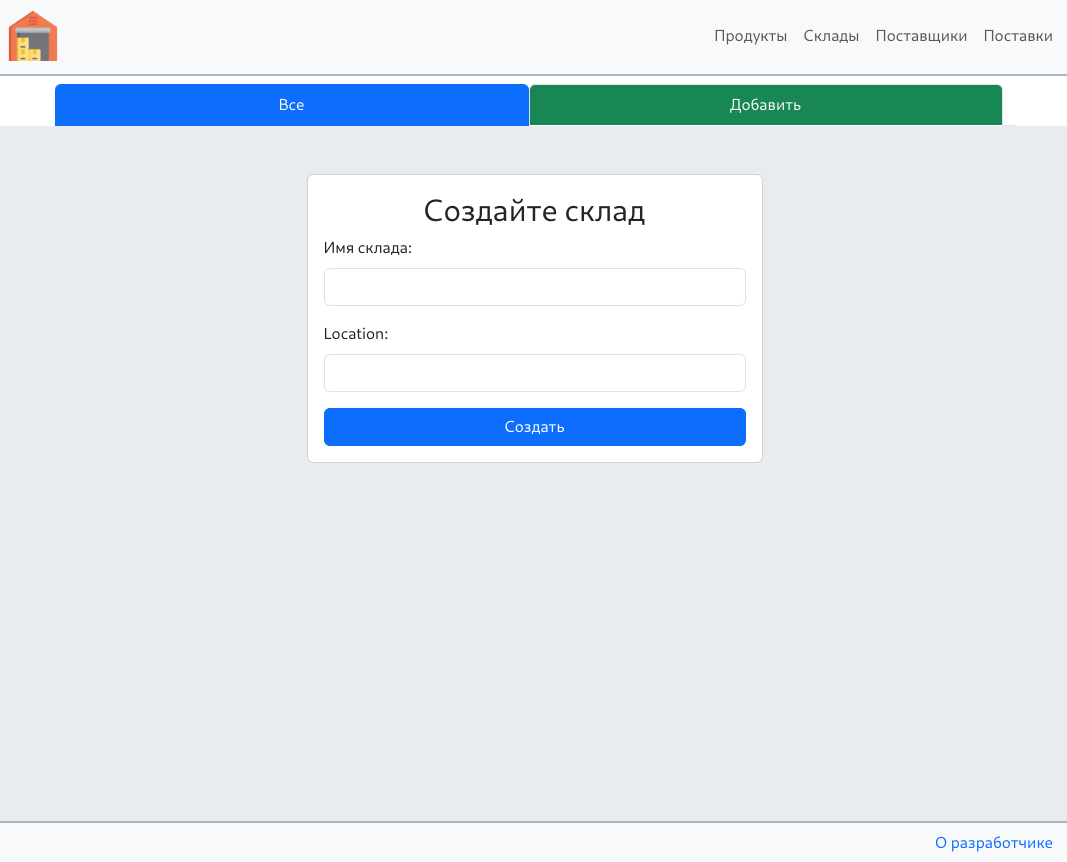


Рис. 7.11 - Страница складов - создание склада

Нажатием на вкладку «Добавить» пользователь видит форму, которую заполняет для создания склада. (Рис. 7.11)

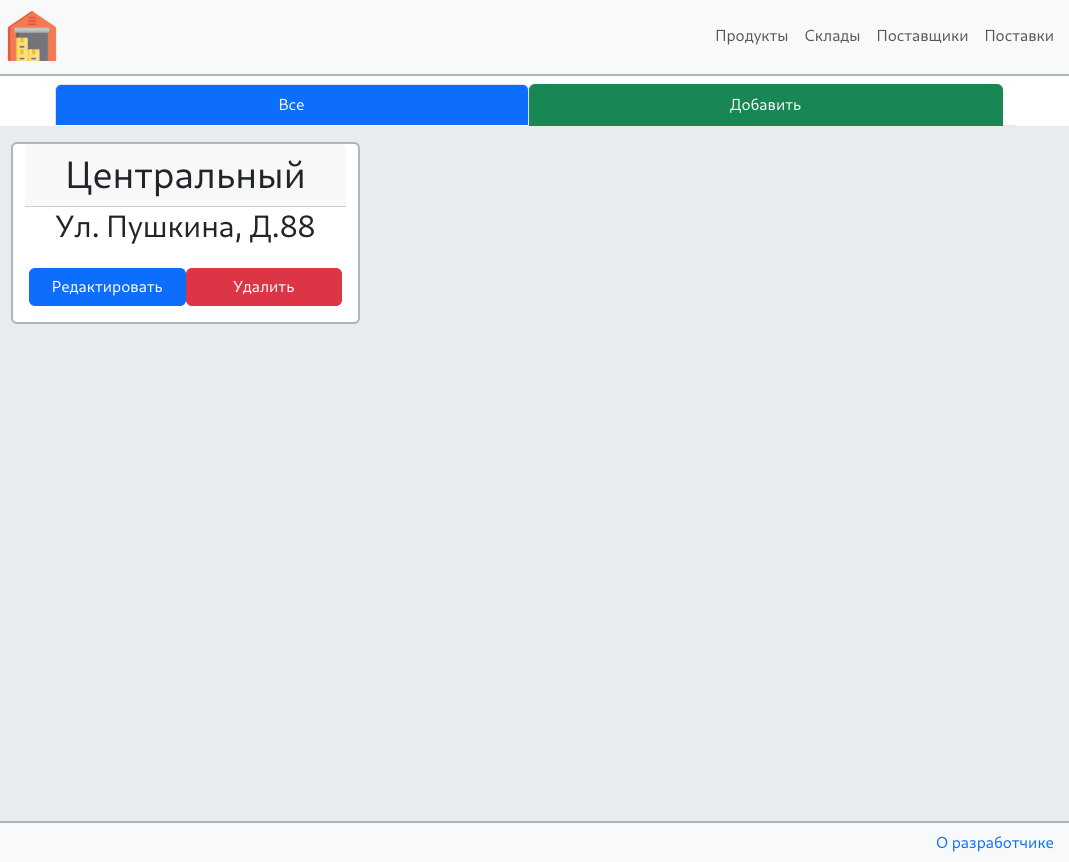


Рис. 7.12 - Страница складов с новым складом

После создания склада видим карточку со складом в общем списке. (Рис. 7.12)

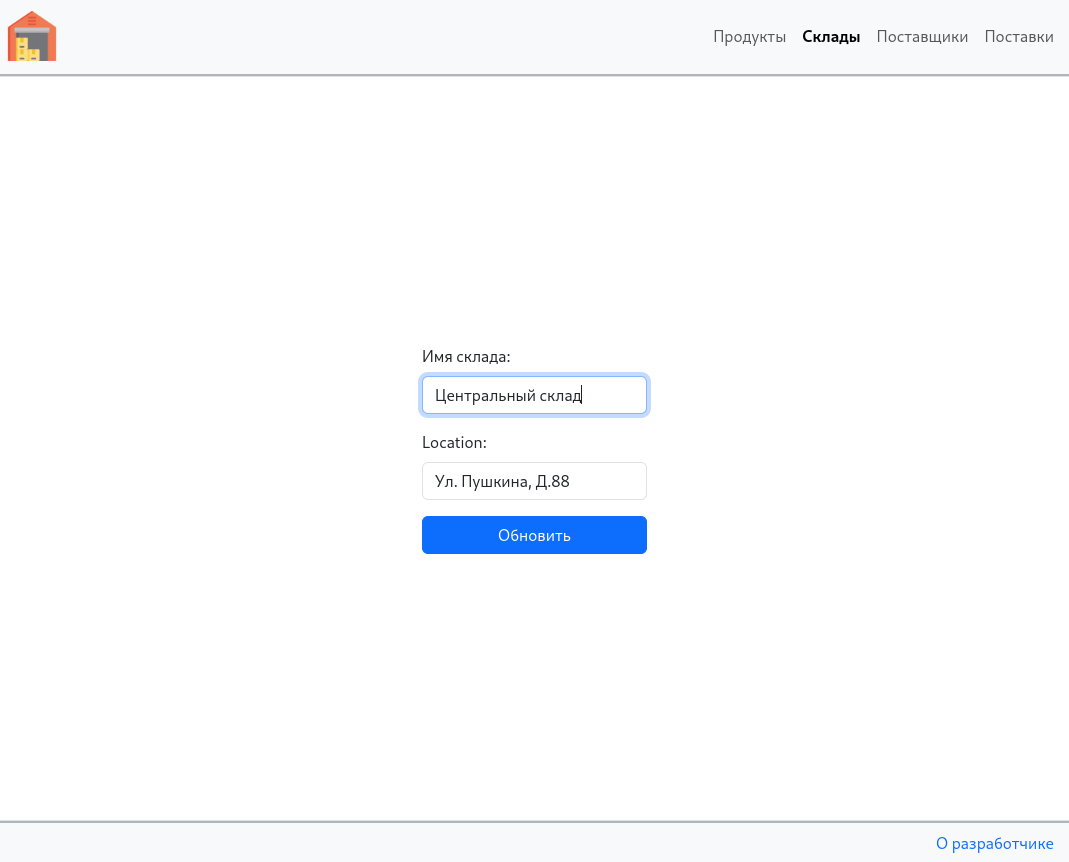


Рис. 7.13 - Страница изменения склада

Затем мы меняем название склада нажав кнопку «Редактировать». (Рис. 7.13)

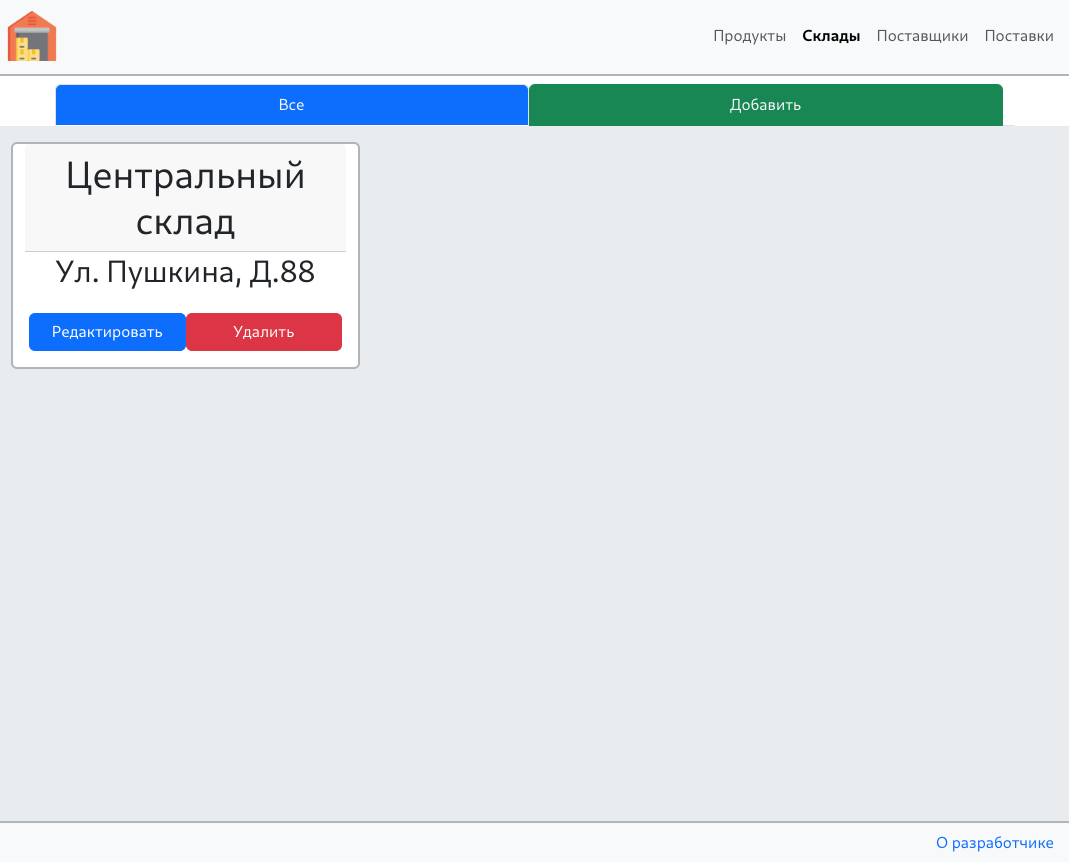


Рис. 7.14 - Измененный склад

После изменения в общем списке можно увидеть измененный склад. (Рис. 7.14)

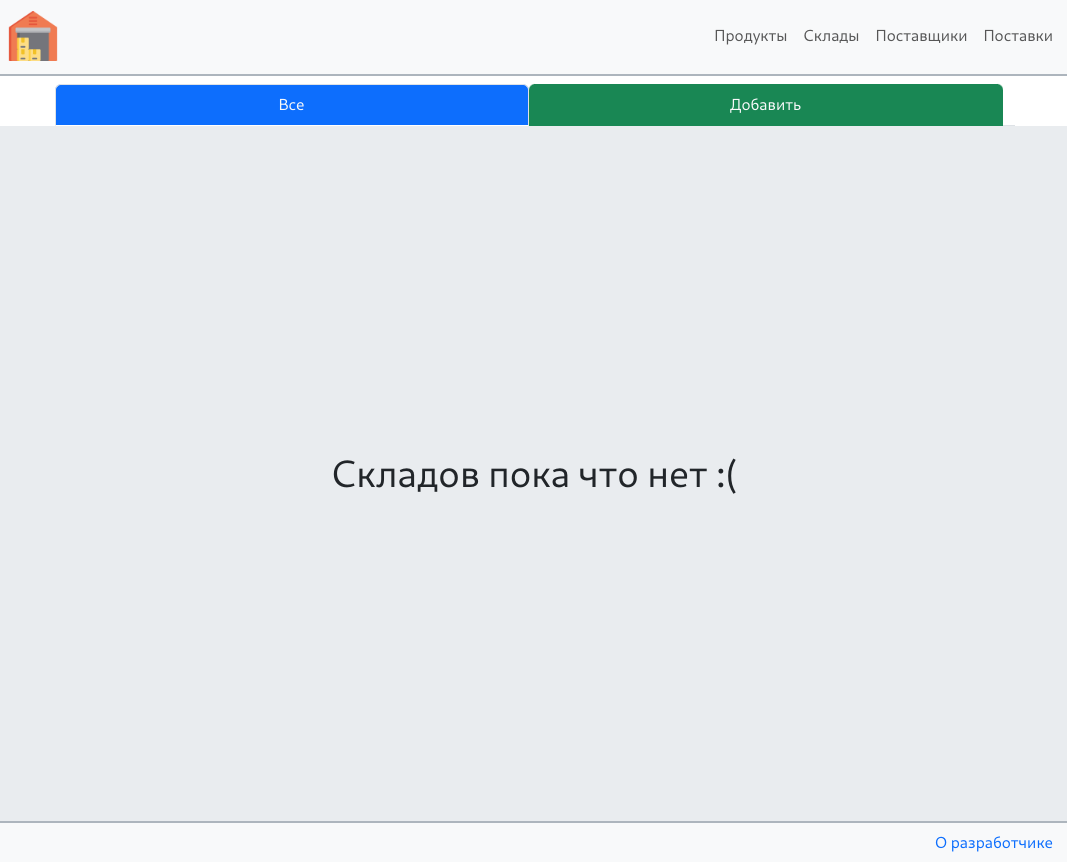


Рис. 7.15 - Удаленный склад

Пользователь удаляет единственный склад и снова возвращается в состояние, когда складов в системе нет. (Рис. 7.15)

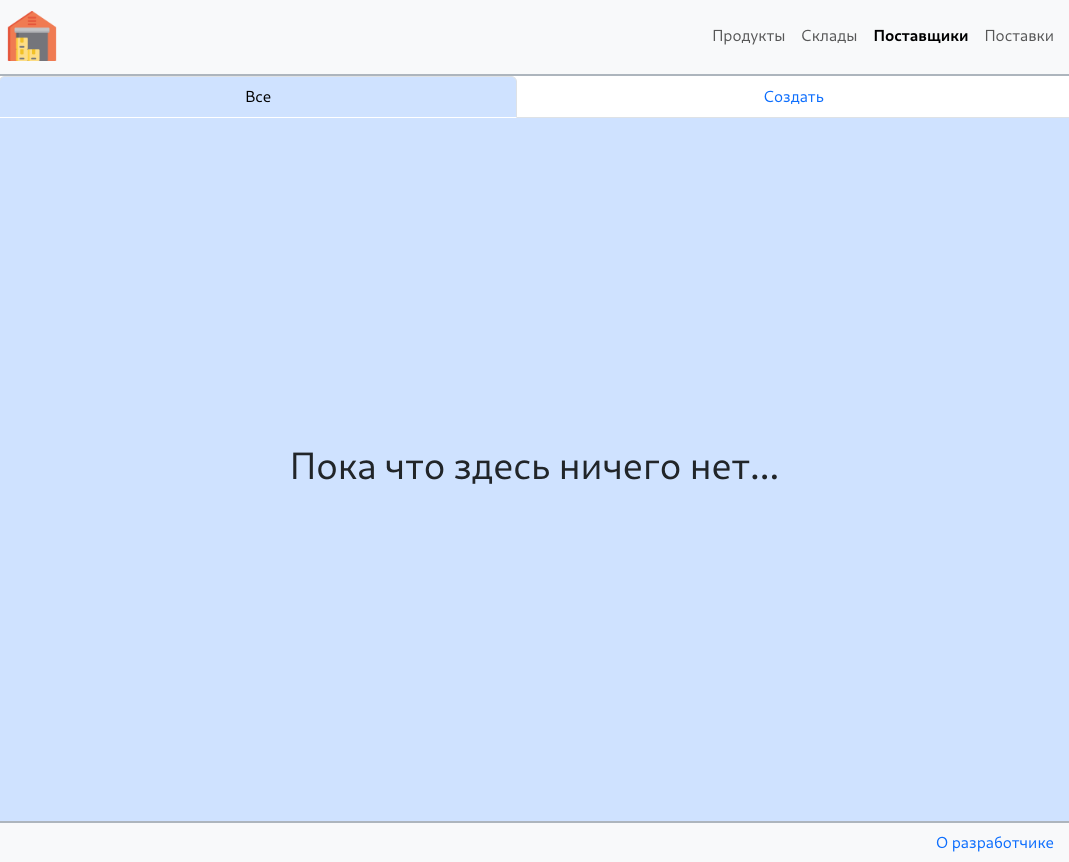


Рис. 7.16 - Страница поставщиков

Теперь пользователь переходит на страницу поставщиков и видит привычные две вкладки на странице. Изначально он попадает на вкладку с общим списком, где пока что нет данных. (Рис. 7.16)

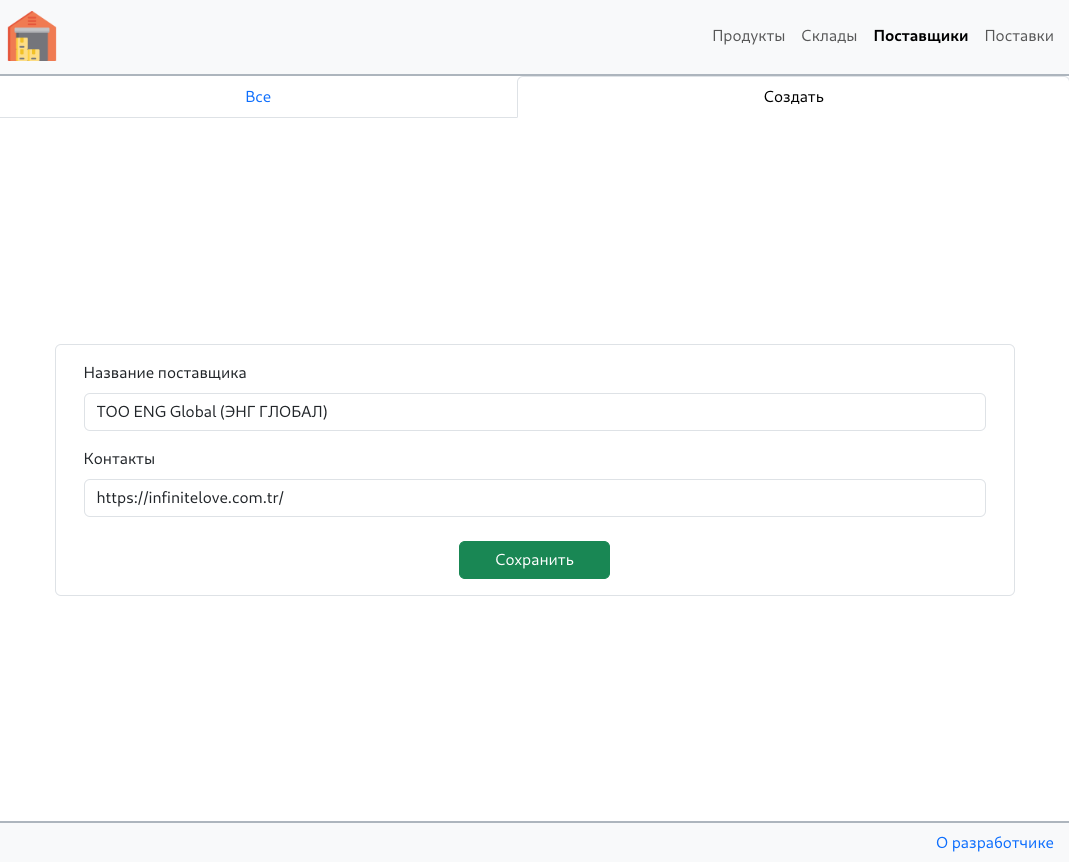


Рис. 7.17 - Создание поставщика

Пользователь переходит на вкладку «Создать» и видит форму, которую заполняет нужными данными. (Рис. 7.17)

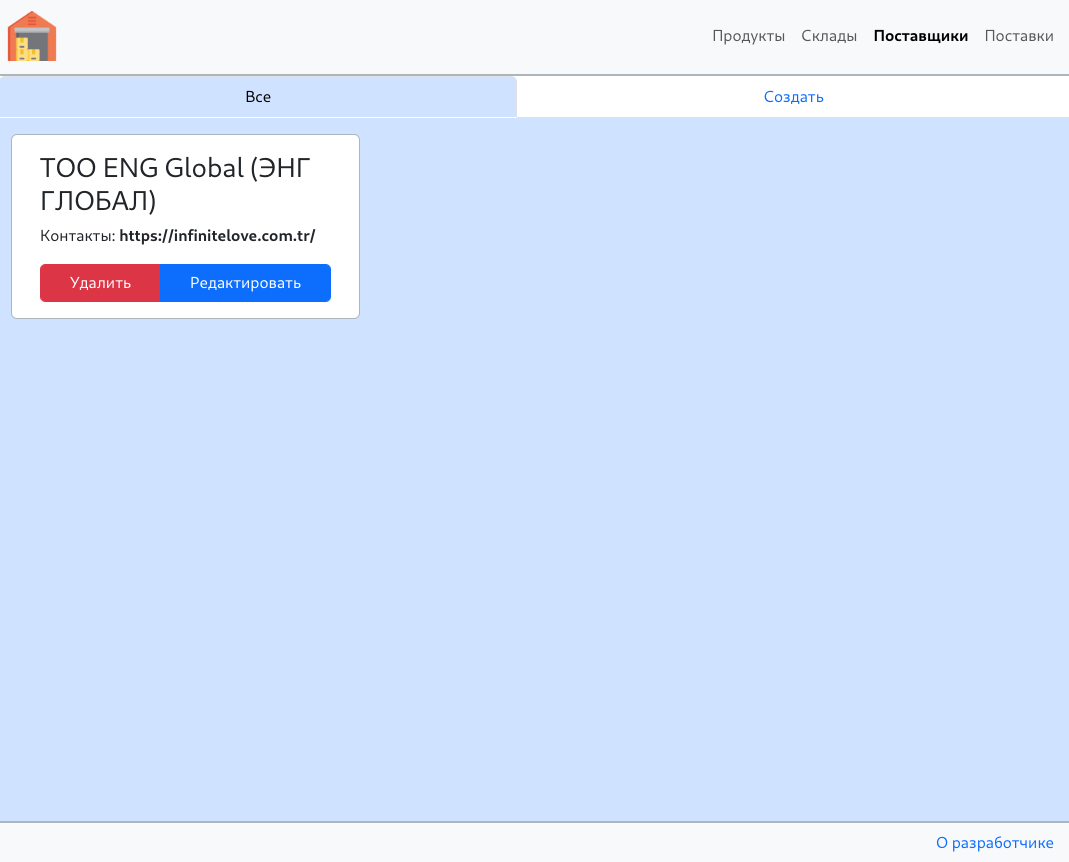


Рис. 7.18 - Созданный поставщик

Затем пользователь попадает на вкладку с общим списком поставщиков, где видит созданного им. (Рис. 7.18)

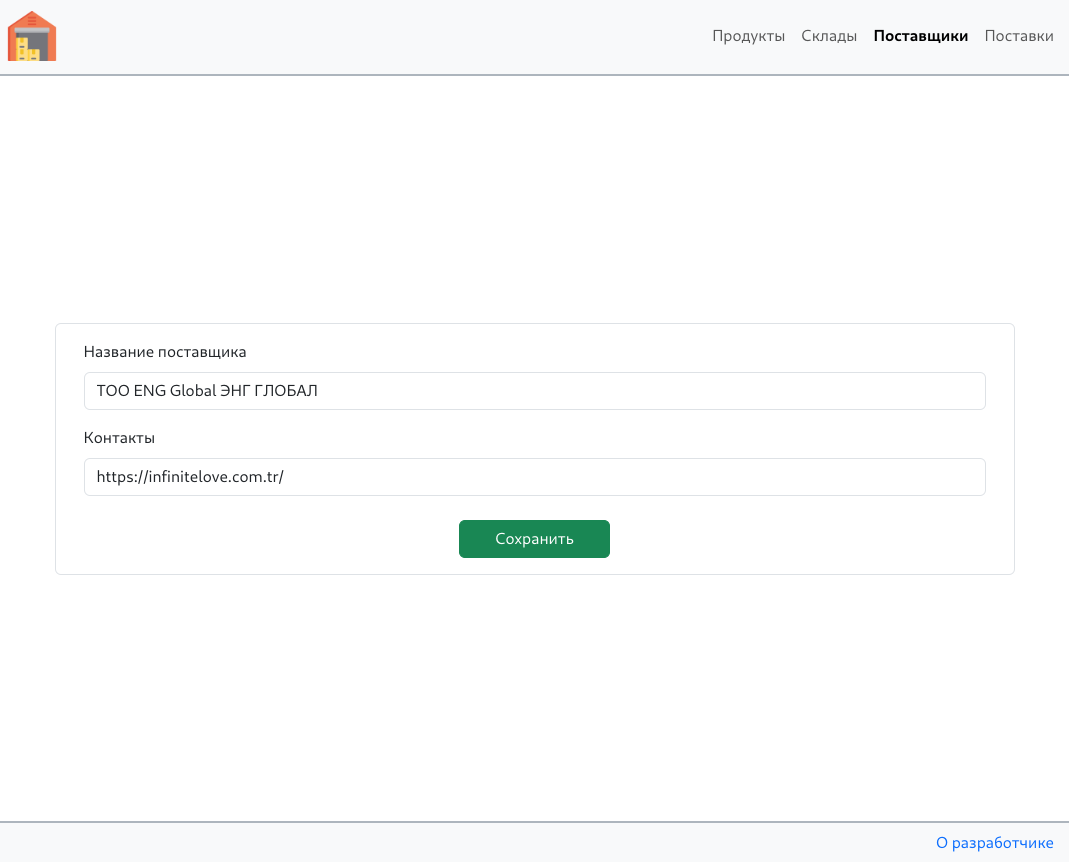


Рис. 7.19 - Редактирование поставщика

После создания пользователь способен отредактировать данные поставщика. (Рис. 7.19)

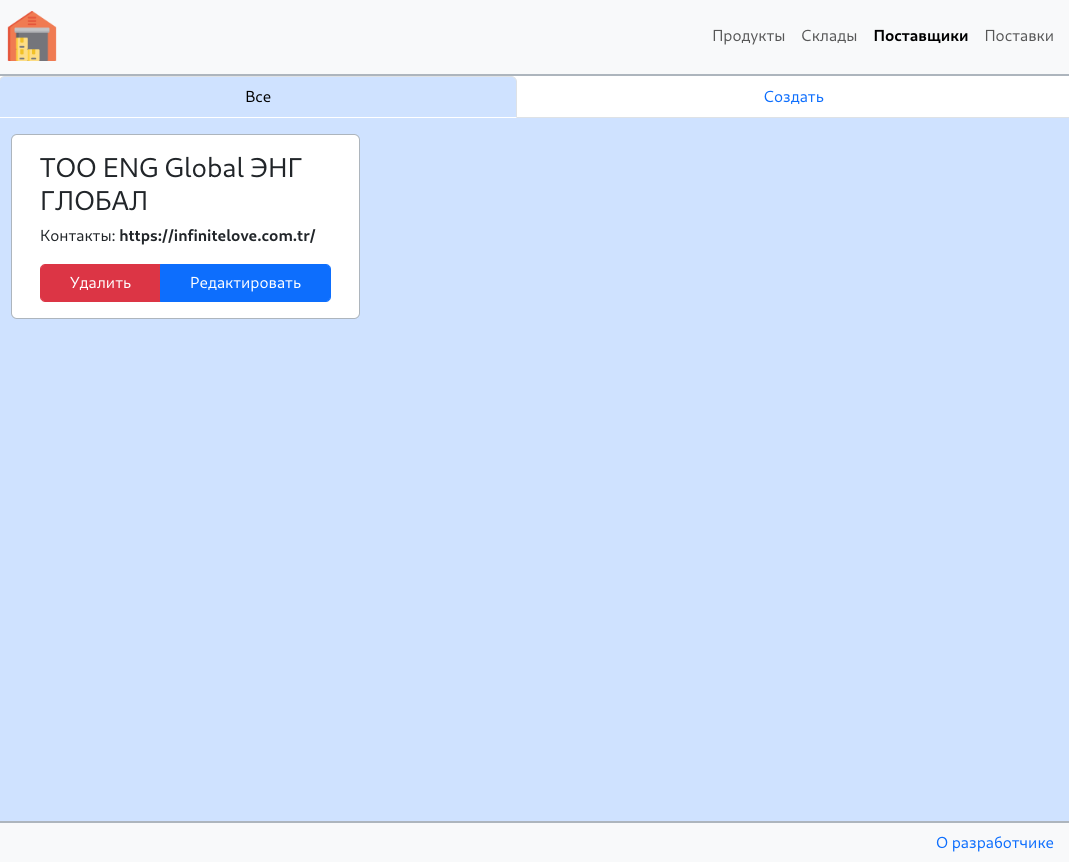


Рис. 7.20 - Отредактированный поставщик

После редактирования пользователь попадает на вкладку с общим списком поставщиков. (Рис. 7.20)

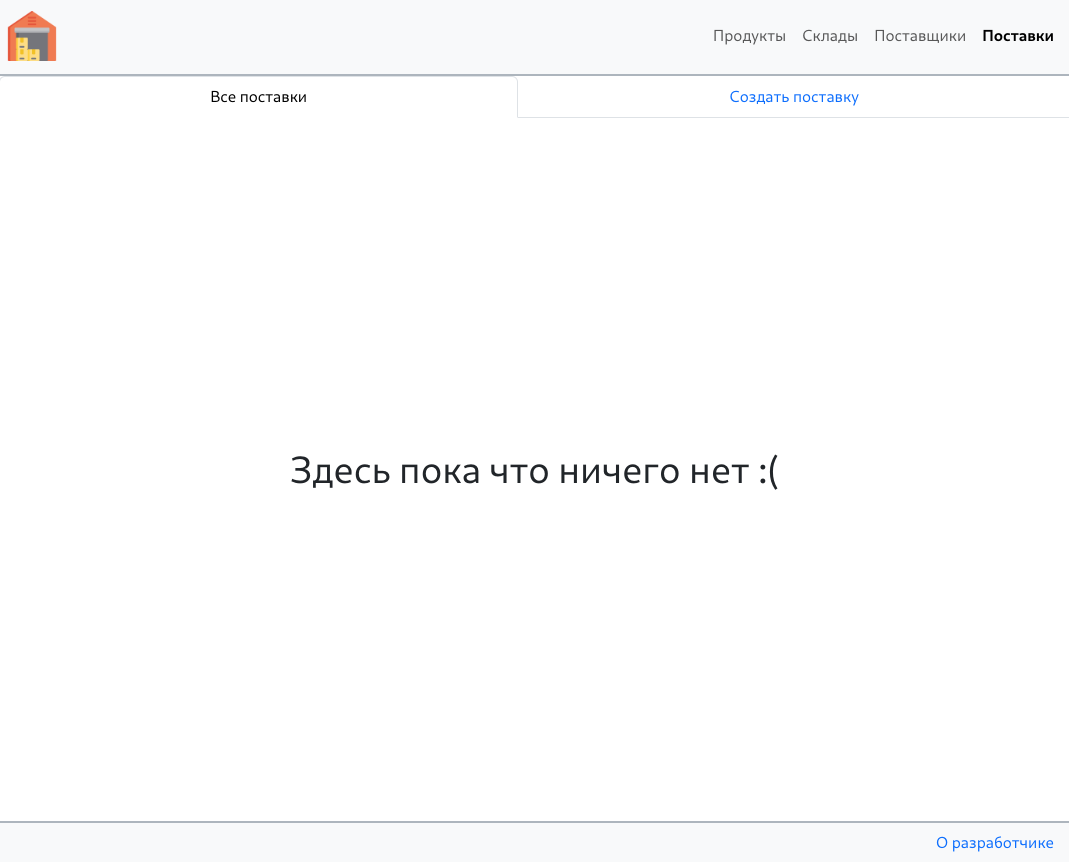


Рис. 7.21 - Страница поставок

Пользователь нажимает на «Поставки» на панели навигации и попадает на страницу с поставками. (Рис. 7.21)

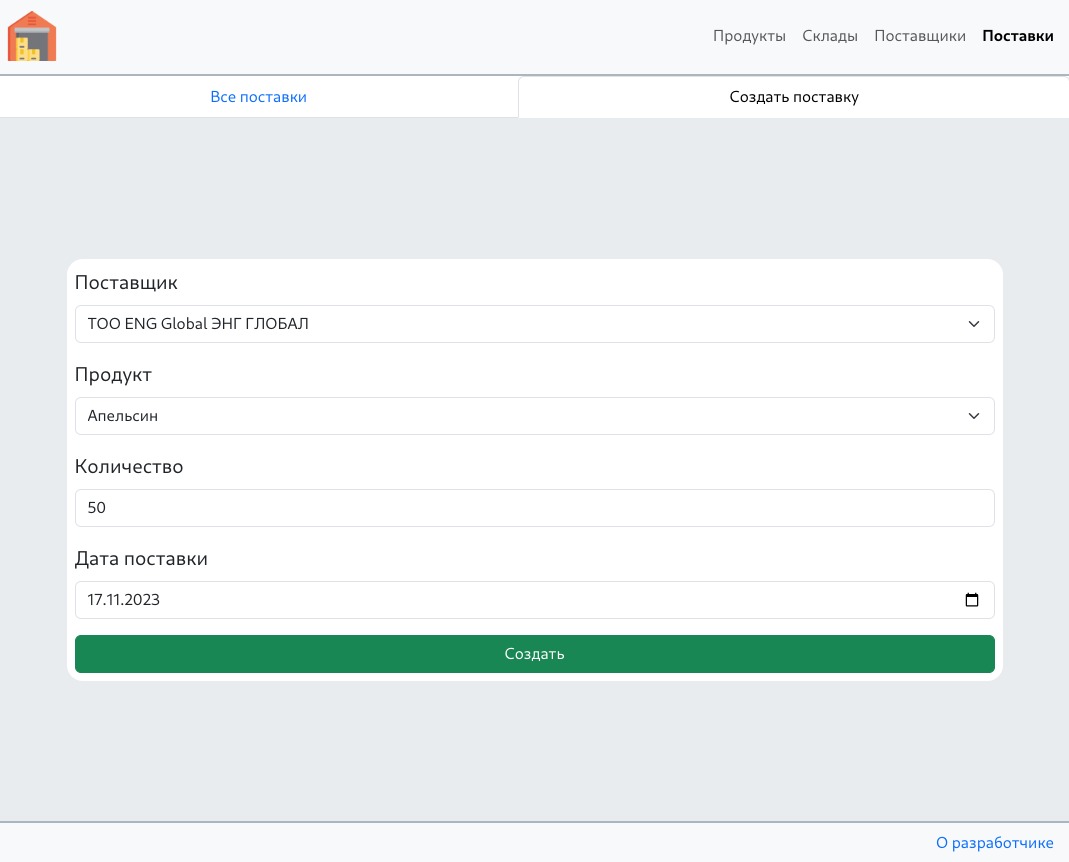


Рис. 7.22 - Страница поставок - создание поставки

Пользователь создает поставку, заполняя форму данными о существующих продукте и поставщике. (Рис. 7.22)

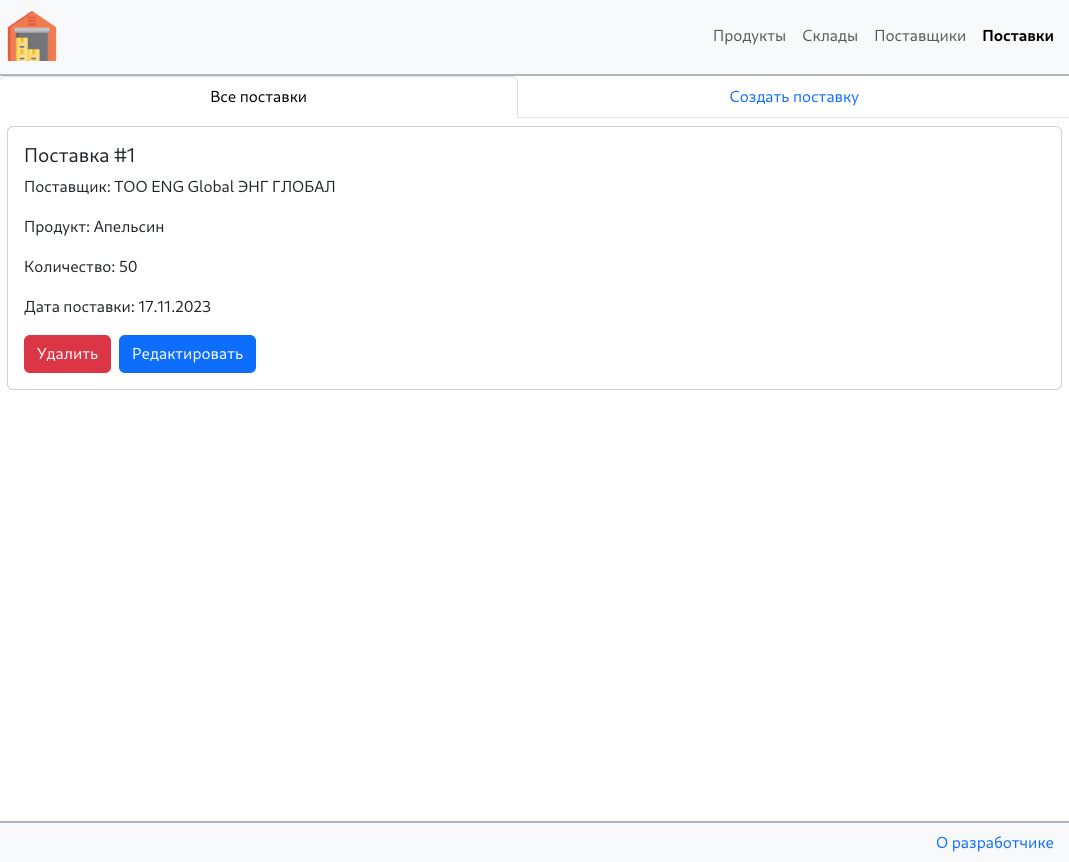


Рис. 7.23 - Созданная поставка

После создания пользователь видит созданную поставку в общем списке поставок. (Рис. 7.23)

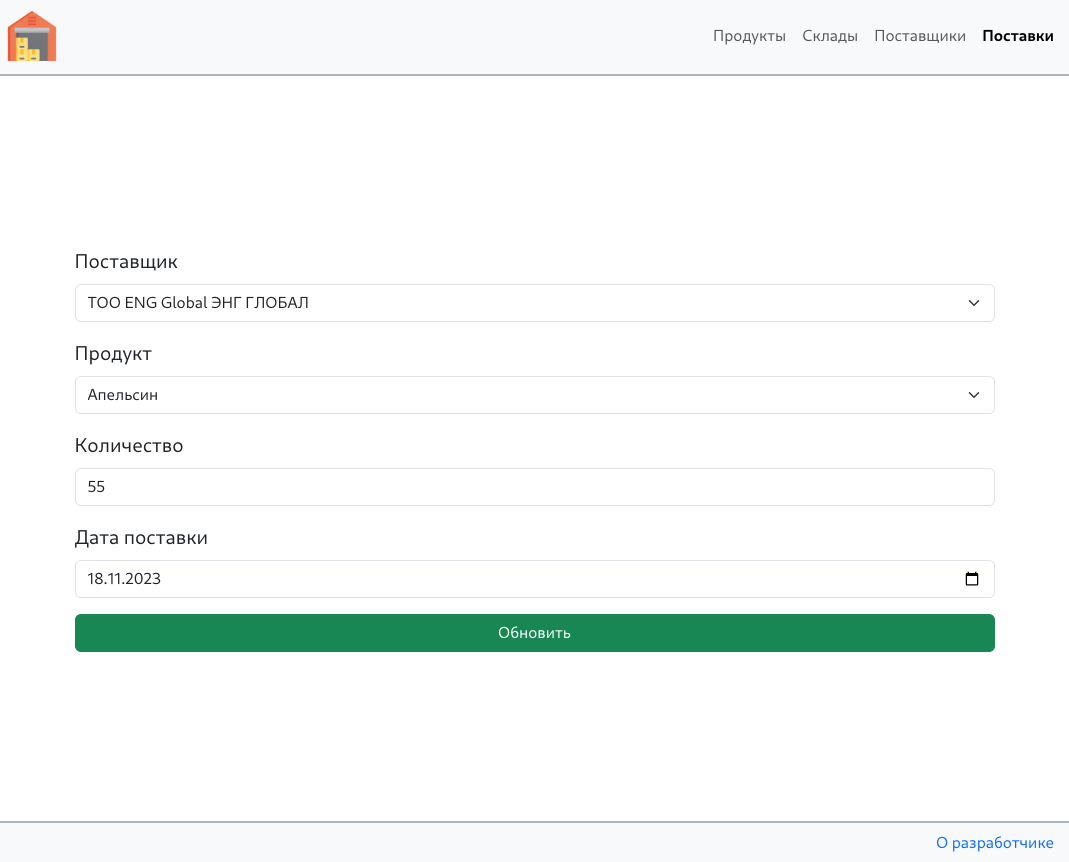


Рис. 7.24 - Редактирование поставки

Пользователь может поступить с созданной поставкой также как и с другими сущностями. В данном примере редактируется поставка с изменением количества продуктов.

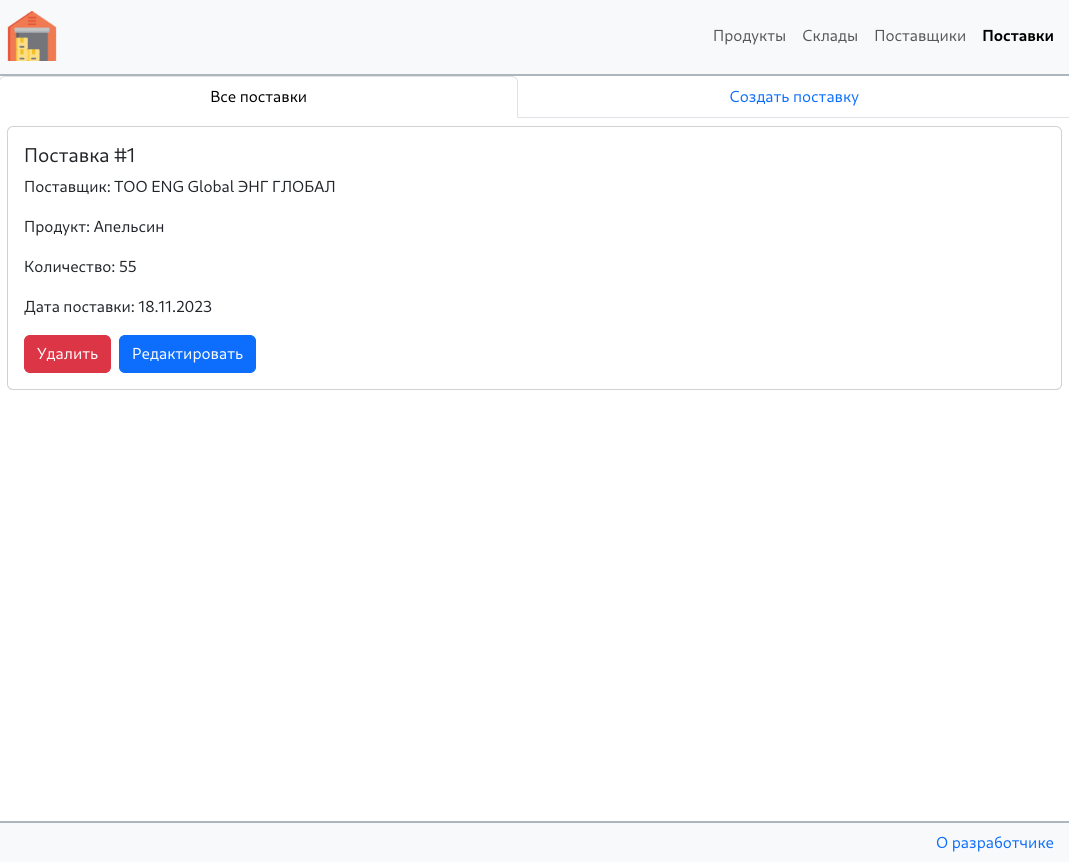


Рис. 7.25 - Отредактированная поставка

После отправки формы редактирования пользователь попадает обратно на страницу с общим списком поставок. (Рис. 7.25)

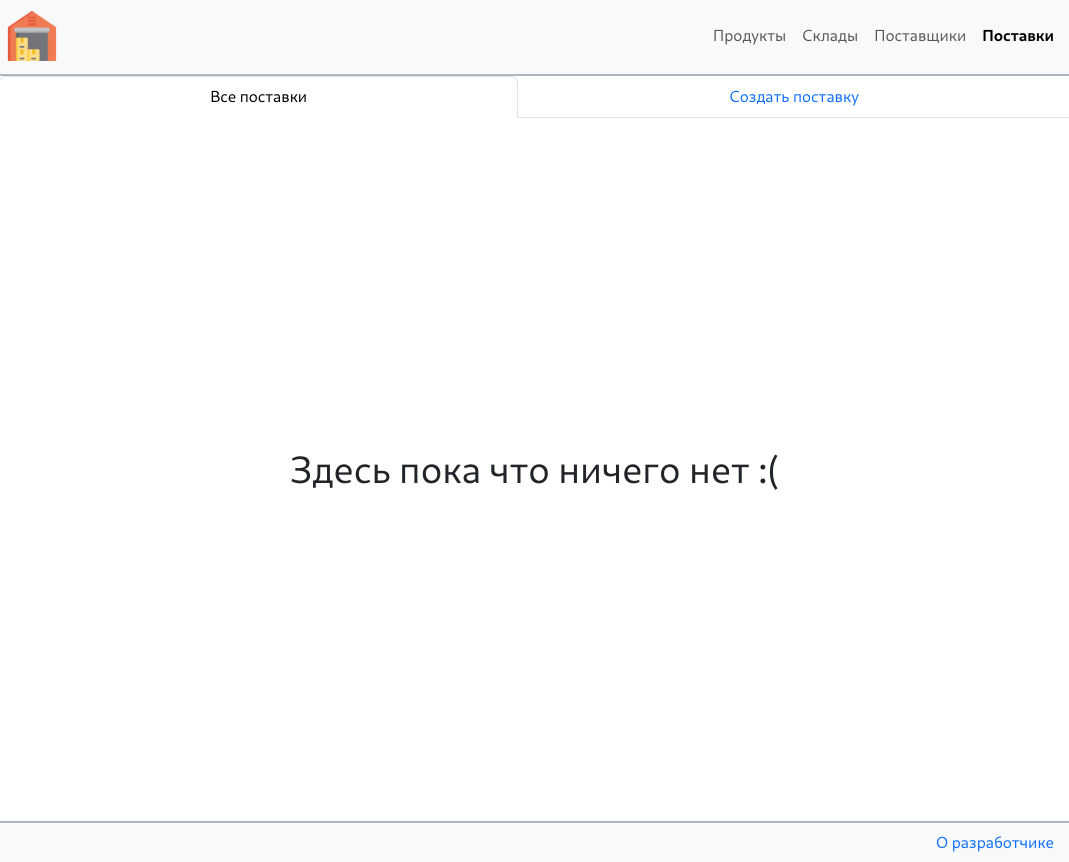


Рис. 7.26 – Удаление поставки

После проделанной работы можно также и удалить поставку. Нажатием на кнопку удалить. После удаления поставка исчезнет из общего списка. (Рис. 7.26)

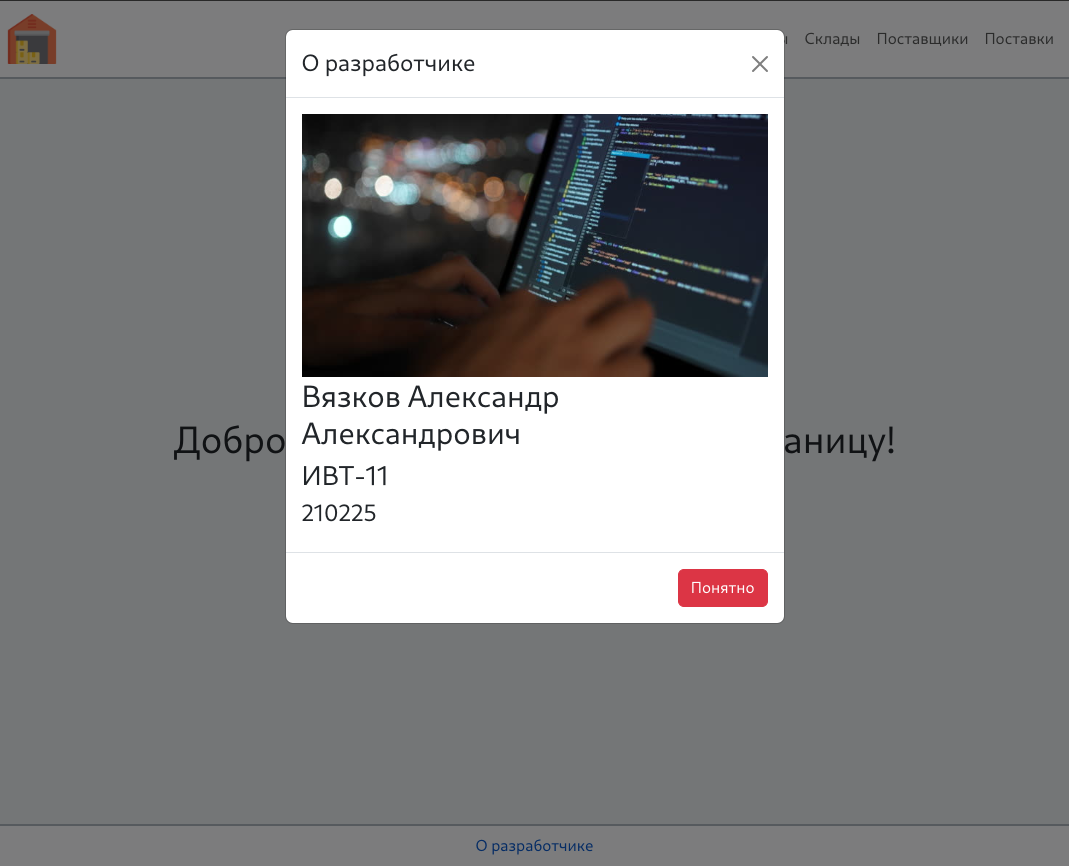


Рис. 7.27 – Всплывающее окно «О разработчике»

В футере сайта можно обнаружить ссылку, которая после нажатия приведет пользователя на всплывающее окно с данными о разработчике. (Рис. 7.27)

# **Список используемых источников**

1. Джон Дакетт - "HTML и CSS. Разработка и дизайн веб-сайтов", Эксмо, 2017, 480 стр. (ориг. название: "HTML и CSS: Design and Build Websites", John Wiley & Sons)
2. Робин Вирух (Robin Wieruch) - "Путь к React: Ваш путь к освоению простого, но прагматичного React.js", БХВ-Петербург, 2018, 242 стр. (ориг. название: "React: Up and Running: Building Web Applications", Packt Publishing)
3. Максимилиан Шварцмюллер (Maximilian Schwarzmüller) - "React - Полное руководство (включая Hooks, React Router, Redux)", Udemy, 2021, 523 стр. (ориг. название: "React - The Complete Guide (incl Hooks, React Router, Redux) - English version")
4. Джошуа Блох (Joshua Bloch) - "Эффективная Java", Addison-Wesley, 2017, 432 стр. (ориг. название: "Effective Java")
5. Билл Венц (Bill Venkataraman) - "Spring Boot: Введение в работу", O'Reilly Media, 2020, 356 стр. (ориг. название: "Spring Boot: Up and Running")
6. Нил Гэнди (Neil Gandhi) - "Освоение Spring Boot 2.0", Packt Publishing, 2018, 548 стр. (ориг. название: "Mastering Spring Boot 2.0")
7. Марк Томас (Mark Thomas) - "Spring в действии", Manning Publications, 2017, 624 стр. (ориг. название: "Spring in Action")
8. Дэвид Макфарланд (David McFarland) - "HTML и CSS: Полное руководство", Диалектика, 2018, 992 стр. (ориг. название: "HTML & CSS: The Complete Reference", Fifth Edition, McGraw-Hill Education)

# **Приложение А – Листинг программного кода**

HomePage.js

class HomePage extends React.Component{  
 render() {  
 return (  
 <PageWithHeaderAndFooter content={  
 <div className="  
 bg-body-secondary  
 d-flex flex-column  
 justify-content-center  
 align-items-center  
 h-100  
 ">  
 <h1>Добро пожаловать на Главную страницу!</h1>  
 <p></p>  
 </div>  
 }/>  
 );  
 }  
};  
export default HomePage;

ProductPage.js

const ProductPageWrapper = () => {  
 const {id} = useParams();  
 return <ProductPage id={id}/>  
}  
class ProductPage extends React.Component{  
  
 constructor(props) {  
 super(props);  
 this.state = {  
 isLoaded: false,  
 data:{},  
 }  
 }  
  
 componentDidMount() {  
  
 new Getter({  
 url: api.api.baseUrl+"/products"  
 }).getById(this.props.id)  
 .then(data=> {  
 this.setState(prev => ({  
 ...prev,  
 data: {  
 ...data.body  
 }  
 }))  
 })  
 .catch(error=>this.setState({error}));  
  
  
 new Getter({  
 url: api.api.baseUrl+`/products/${this.props.id}/storage-condition`  
 })  
 .getAll()  
 .then(data=>{  
 if(data.code !== 200){  
 this.setState(  
 prev=>(  
 {  
 ...prev,  
 isLoaded: true,  
 error: data.message  
  
 }  
 )  
 );  
 }  
 else {  
 this.setState(  
 prev => ({  
 ...prev,  
 isLoaded: true,  
 storageCondition: {  
 ...data.body  
 }  
 }), () => console.log(JSON.stringify(this.state))  
 );  
 }  
 });  
 }  
  
 #storageConditionData = (storageCondition)=>{  
 return(  
 <div>  
 <p className="fw-bold fst-italic">Хранить при температуре {storageCondition.temperature}°C,  
 влажности {storageCondition.humidity}% и освещении {storageCondition.lighting} люкс</p>  
 </div>  
 );  
 }  
  
 #storageConditionError = () =>{  
 return(<Error message={this.state.error}/>);  
 }  
  
 onRemove = (e)=>{  
 console.log("removing clicked!");  
 new Remover({url: api.api.baseUrl+`/products`})  
 .removeById(this.state.data.id)  
 .then(resp=>{  
 if(resp.ok){  
 console.log("removed...");  
 window.location.replace("/products");  
 }  
 else{  
 console.log(resp);  
 this.setState({  
 error: resp.message  
 }  
 )  
 }  
  
 })  
 }  
 onUpdate = (e) =>{  
 window.location.replace(window.location.href+'/update');  
 }  
  
 onClickHistory = (e) =>{  
 window.location.replace(window.location.href+'/history');  
 }  
 render() {  
  
 const {isLoaded, data, storageCondition} = this.state;  
  
 if (!isLoaded) {  
 return (  
 <PageWithHeaderAndFooter content={  
 <div className="h-100 w-100 d-flex justify-content-center align-items-center">  
 <div className="spinner-border text-success"></div>  
 </div>  
 }/>  
 );  
 }  
  
 return (  
 <PageWithHeaderAndFooter content={  
 <div className="">  
 <h1 className="text-center">{data.name}</h1>  
 <div>  
 <p className="text-wrap text-break">  
 Описание: {data.description}  
 </p>  
 <p>Количество: {data.quantity}</p>  
 <p className="">Дата истечения срока годности: {" "}  
 <span  
 className={"fw-bold " + ((new Date(Date.parse(data.expirationDate) - new Date().getTime()).getDate()) > 7 ? "" : "text-danger")}>  
 {new Date(data.expirationDate.split('T')[0]).toLocaleDateString()}  
 </span>  
 </p>  
 </div>  
 {  
 !this.state.error  
 ? this.#storageConditionData(storageCondition)  
 : this.#storageConditionError()  
 }  
 <hr/>  
 <div className="d-flex flex-row-reverse justify-content-between form-control">  
 <button className="btn btn-danger" onClick={(e) => {  
 console.log("removing...");  
 this.onRemove(e);  
 }}>Удалить</button>  
 <button className="btn btn-info" onClick={this.onClickHistory}>История</button>  
 <button className="btn btn-primary" onClick={this.onUpdate}>Редактировать</button>  
 </div>  
 </div>  
 }/>  
 );  
 }  
  
  
}  
  
export default *ProductPageWrapper*;

Products.js

class ProductsPage extends React.Component {  
 constructor(props) {  
 super(props);  
 this.state = {  
 currentTab: undefined  
 }  
 }  
  
 changeTab = (event)=>{  
 console.log("updating selected tab...");  
 const id = event.target.id;  
 this.setState(prevState=>({  
 ...prevState,  
 currentTab: id  
 }),()=>{console.log(this.state.currentTab)});  
 }  
 render() {  
 return (  
 <PageWithHeaderAndFooter content={this.#productsNav()}/>  
 );  
 }  
  
 #productsNav() {  
 return (  
 <div className="h-100 d-flex flex-column">  
 <ul className="nav nav-tabs nav-fill">  
 <li className="nav-item">  
 <a className="nav-link active bg-primary-subtle"  
 data-bs-toggle="tab" href="#products" onClick={this.changeTab}  
 id="all">Все</a>  
 </li>  
 <li className="nav-item">  
 <a className="nav-link bg-success-subtle"  
 data-bs-toggle="tab"  
 href="#create"  
 onClick={this.changeTab}  
 id="create-tab">Создать</a>  
 </li>  
 </ul>  
  
 <div className="tab-content h-100">  
 <div className="tab-pane active bg-primary-subtle h-100" id="products">  
 <AllProducts kek={this.state.currentTab}/>  
 </div>  
 <div className="tab-pane fade h-100 bg-success-subtle" id="create">  
 <ProductCreator kek={this.state.currentTab}/>  
 </div>  
 </div>  
 </div>  
  
 );  
 }  
}