|  | **Министерство науки и высшего образования Российской Федерации**  **Федеральное государственное бюджетное образовательное учреждение**  **высшего образования**  **«Московский государственный технический университет**  **имени Н.Э. Баумана**  **(национальный исследовательский университет)»**  **(МГТУ им. Н.Э. Баумана)** |
| --- | --- |

**Факультет «Информатика и системы управления»**

**Кафедра ИУ5 «Системы обработки информации и управления»**

Дисциплина «Разработка интернет-приложений»

Расчетно-пояснительная записка

Тема: «Грузовые корабли к МКС»

Студент: Иванченко М.Д.

Группа: ИУ5-53Б

Преподаватель: Канев А.И.

2024 г.

# **ВВЕДЕНИЕ**

Современное развитие космической индустрии предоставляет новые возможности для транспортировки грузов к международной космической станции (МКС). С увеличением числа космических миссий и ростом интереса к исследованию космоса актуальность эффективных и надежных средств доставки грузов становится более явной. В свете этого, создание системы мониторинга и автоматизированной логистики для грузовых кораблей, предназначенных для поставок к МКС, представляет собой важный шаг в улучшении этого процесса.

Целью работы является реализация системы для контроля доставки различных грузов к МКС, включающую в себя веб-сервис, веб-приложение, десктопное приложение и выделенный сервис расчета статуса доставки.

Система предназначена для заказчиков и модераторов (руководства государственных организаций, занимающихся доставкой грузов к МКС). Предусмотрен доступ к списку грузов. Для реализации доставки грузов пользователю необходимо создать заявку. Система предоставляет автоматизированный способ создания, учета и ведения полетов. Также она позволяет модераторам принимать или отклонять заявки. Модераторы имеют возможность редактировать существующие и добавлять новые грузы.

Нефункциональные требования к разрабатываемой системе:

* 1. Должна поддерживаться кроссплатформенность.
  2. Интерфейс системы должны быть русифицированным.

В ходе работы необходимо выполнить следующие задачи:

* 1. Разработать дизайн приложения.
  2. Создать базу данных в PostgreSQL.
  3. Создать веб-сервис на golang 1.21.0.
  4. Реализовать интерфейс гостя на технологии React.
  5. Задеплоить на Github Pages.
  6. Добавить авторизацию и аутентификацию в веб-сервис.
  7. Реализовать интерфейс пользователя в React.
  8. Реализовать интерфейс модератора в React.
  9. Создать десктопное приложение на React Native.
  10. Создать выделенный сервис для расчета статуса доставки на Django.
  11. Подготовить набор документации, включающий РПЗ, ТЗ и набор диаграмм.

# **БИЗНЕС-ПРОЦЕСС**

В бизнес-процессе участвуют заказчики. Они могут иметь роль пользователя (все заказчики), модератора (руководство государственных организаций), или не иметь роль вообще – тогда они считаются гостями.

Система хранит список грузов, которые могут быть отправлены на международную космическую станцию. Грузы имеют название, категорию, описание, фотографию, цену, массу, объем. Заказчики выбирают грузы, которые могут быть доставлены на МКС [1], добавляя их в полет-черновик. Черновой полет можно удалить в любой момент. Когда заказчик выберет все желаемые грузы и установит желаемое количество каждого из них, он формирует полет. После этого изменение состава полета невозможно. Список истории заявок заказчика доступен в соответствующем пункте меню. Сформированные полеты затем рассматриваются госорганизациями. Они принимают решение об одобрении или отклонении сформированного полета. Решение госорганизации зависит от результата доставки грузов на МКС. Каков результат доставки – решает внешний сервис. После вынесения вердикта у заказчика обновляется статус полета на «завершен» или «отклонен».

Грузы могут обладать разными размерами и объемом, будь это пакет с сублимированной водой или баллон с сжатым воздухом. Особенности груза видны по его изображению, категории и описанию. Космонавтам на МКС могут требоваться новые продукты для разнообразия питания, лекарства для лечения новых болезней, новое оборудование для экспериментов, топливо [2]. Именно поэтому госорганизация может отредактировать какую-либо груз или создать совершенно новый.

Функции пользователей с разными ролями описаны на диаграмме прецедентов (рис. 1).

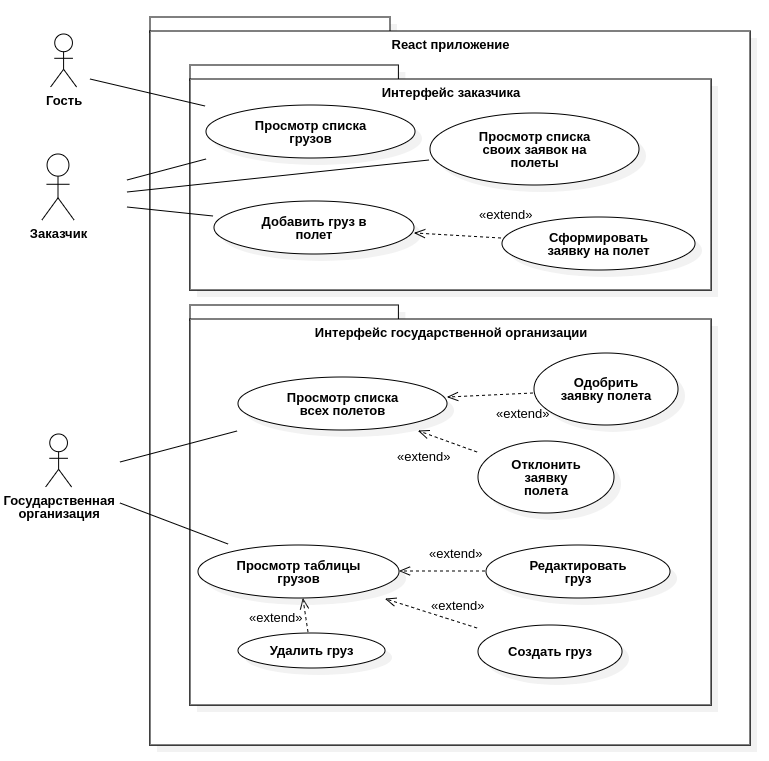


Рисунок 1 - Диаграмма прецедентов

Гостям доступен только просмотр грузов. Зарегистрированные гости – заказчики. Они могут добавлять грузы в полет, просматривать список своих полетов и формировать текущий полет-черновик. Полеты обрабатываются модераторами (госорганизациями). В результате обработки полета её либо одобряют, либо отклоняют. Процесс оформления полета отражен на диаграмме деятельности (рис. 2).

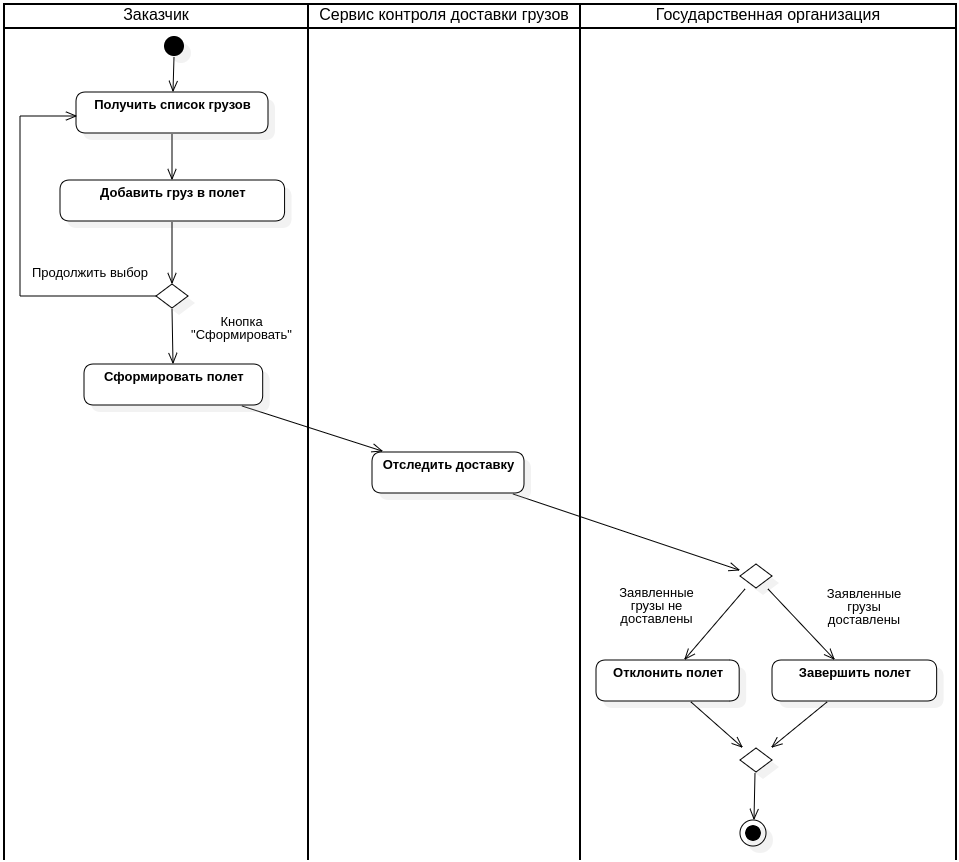


Рисунок 2 - Диаграмма деятельности

Заказчик выбирает грузы, затем формирует на основе выбранных грузов полет. После формирования этот полет обрабатывает асинхронный сервис, а затем и госорганизация. Возможные состояния полетов отражены на диаграмме состояний для статусов полета (рис. 3).

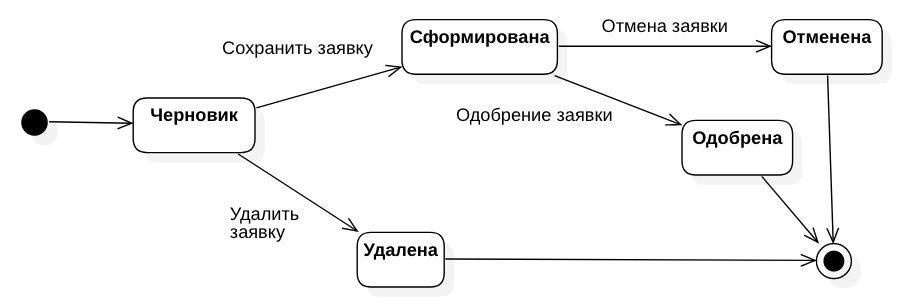


Рисунок 3 - Диаграмма состояний статусов полета

При выборе первого груза формируется полет-черновик. Последующие выбранные грузы добавляются в этот черновик. Заказчик затем формирует полет, удаляет ее или выходит из приложения. Сформированный полет обрабатывает гос организация. Она может одобрить или отклонить ее.

# **АРХИТЕКТУРА**

Архитектура системы представлена на диаграмме развертывания (рис. 4). Бэкенд разворачивается на локальном ноутбуке. Такое решение было принято, поскольку бэкенд не требует масштабируемости и может успешно функционировать в отдельном окружении. В дополнение к бэкенду, в системе присутствует асинхронный сервис, который разворачивается в контейнере Docker. Этот сервис предоставляет дополнительные функциональности и может быть масштабирован в зависимости от потребностей. Дополнительно, в системе используются следующие Docker-образы, описанные в docker-compose.yml файле [3]: PostgreSQL, Redis, Minio. Этот подход позволяет легко развертывать и управлять каждым компонентом системы, а также обеспечивает гибкость в выборе платформы для запуска Docker-контейнера. Асинхронный сервис [2] разворачивается отдельно и необязательно на том же компьютере, что и основной сервис.

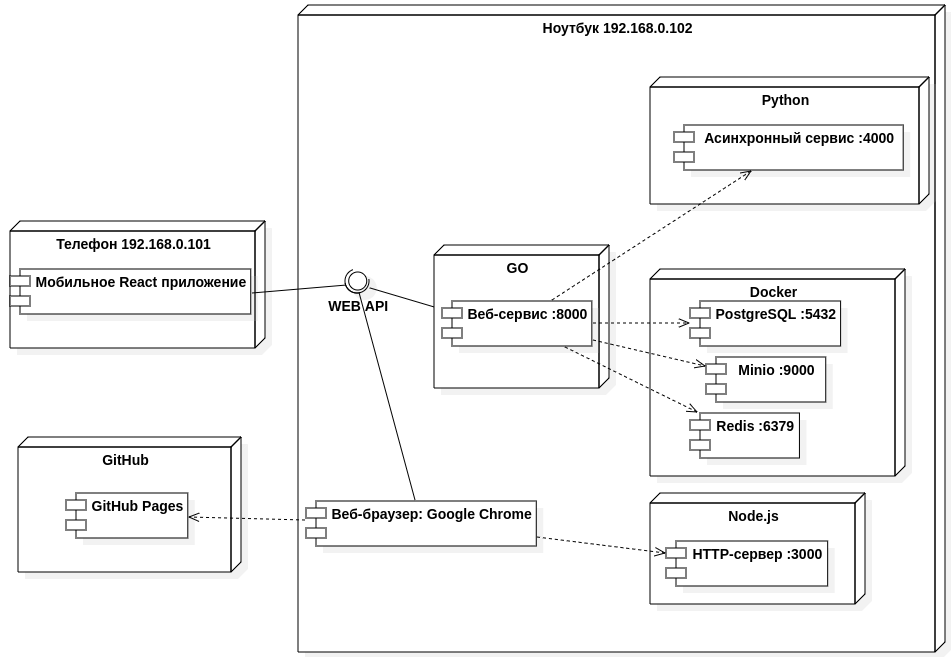


Рисунок 4 - Диаграмма развертывания

Нативное [4] и браузерное приложения [5, 6] обращаются к веб-сервису, основанному на технологии Gin v1.9.1 [7], через REST Web-API. Использование фреймворка Gin обусловлено тем, что эта технология является проверенным и быстродействующим решением, стандартом индустрии. Современность языка GoLang также склоняет к выбору технологии. Данные хранятся в СУБД PostgreSQL [8], их структура отражена на ER диаграмме (рис. 5). СУБД PostgreSQL является одним из стандартов индустрии, поэтому было решено использовать её. Структура данных довольна проста. Помимо базовых полей, заказчик также имеет поле роль, отражающее его уровень доступа к информации сервиса. Модель груза представляет собой набор полей, необходимых исключительно для бизнес-логики. Для хранения в одном полете нескольких грузов используется промежуточная таблица полеты\_с\_грузами, реализующая связь М-М. Устройство бекенда приложения изображено на диаграмме классов бекенда (рис. 6).

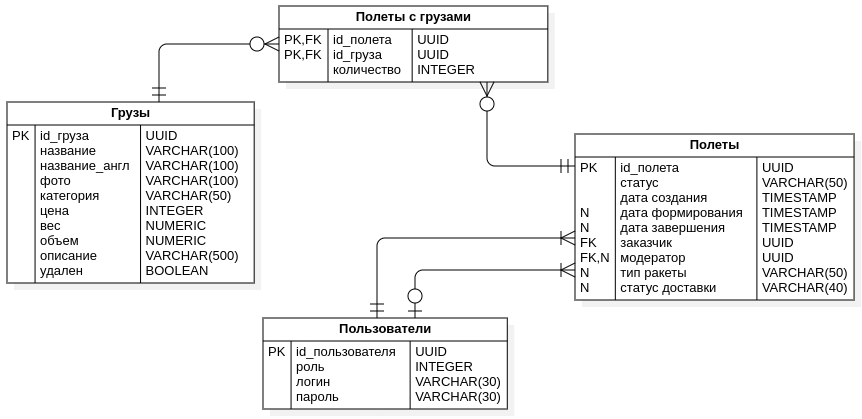


Рисунок 5 - ER диаграмма

Устройство бекенда приложения изображено на диаграмме классов бекенда (рис. 6). Модели имеют связи с таблицами в базе данных. Также некоторые модели имеют связи с внешними сервисами. В частности, грузы имеют связь с сервером статических файлов, т.к. в грузах хранится ссылка на их изображение, хранимое на сервере статических файлов.

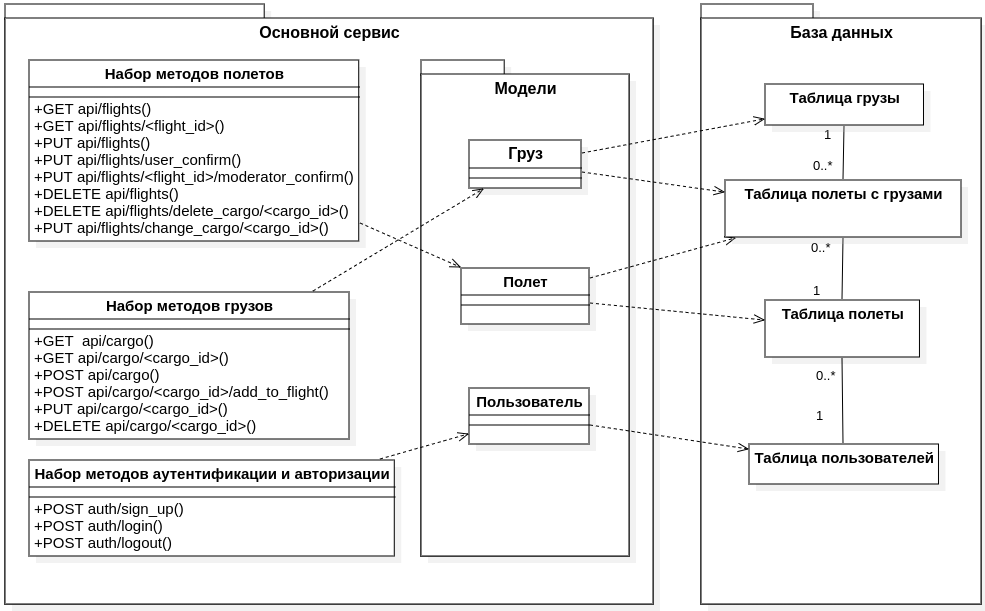


Рисунок 6 *-* Диаграмма классов бекенда

Связь фронтенда и бекенда отражена на диаграмме классов фронтенда (рис. 7). Ключевые страницы имеют связь с API аутентификации, т.к. доступ к ним осуществляется только для авторизированных пользователей (заказчиков) с определенными правами (ролями).

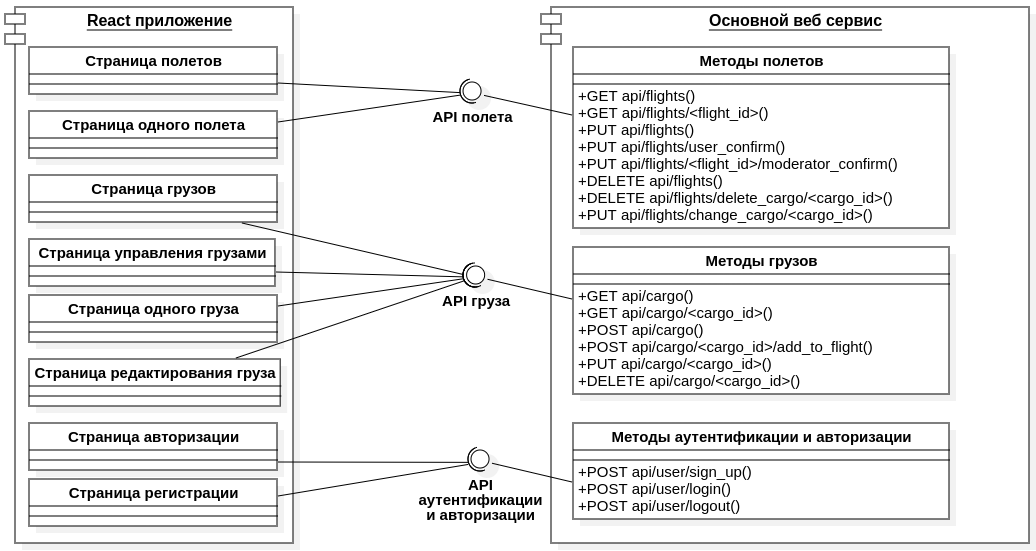


Рисунок 7 *-* Диаграмма классов фронтенда

# **АЛГОРИТМЫ**

Алгоритм работы системы отображен на диаграмме последовательности (рис. 8). В основе системы лежит веб-сервис, реализующий внутри себя всю бизнес-логику. Он предоставляет доступ к методам из следующих доменов: грузы, полеты, пользователь. Методы следуют правилам REST API.

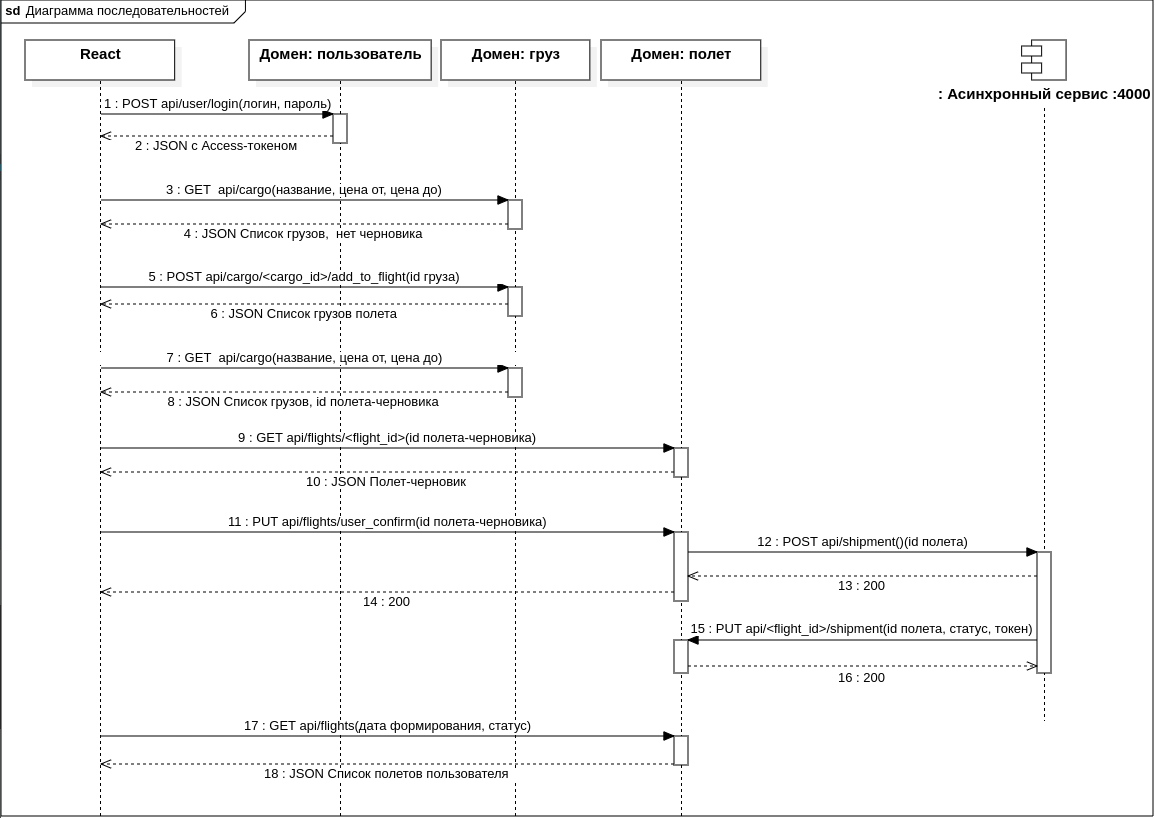


Рисунок 8 - Диаграмма последовательности

В начале бизнес-процесса происходит аутентификация заказчика. Для этого он отправляет через графический интерфейс запрос, передавая в нем логин и пароль. Если аккаунт с такими данными существует, то пользователь получает JWT в ответном запросе. Если же такого аккаунта не существует, или пароль введен неверно, пользователь получит ошибку. В таком случае ему надо либо пройти регистрацию, либо ввести пароль верно. Затем графический интерфейс заказчика запрашивает у веб-сервиса список грузов, которые возвращаются в JSON формате. Пользователь выбирает груз, которую хочет доставить на МКС, и, нажимая на кнопку «добавить в корзину» в графическом интерфейсе, отправляет запрос на добавление груза в свой черновой полет. Этот процесс может продолжаться несколько раз.

Когда пользователь определится с выбором, он нажимает на кнопку «сформировать» в графическом интерфейсе. После этого приложение запрашивает id чернового полета заказчика и затем отправляет запрос на формирование этого полета. В этот момент основной веб-сервис выполняет запрос к асинхронному сервису на то, чтобы тот рассчитал результат доставки.

Процесс рассмотрения полетов происходит также через графический интерфейс. Государственные организации могут просматривать списки всех полетов и, нажимая на соответствующие кнопки, отправлять запросы на их одобрение или отклонение в основной веб-сервис. В эти запросы также можно включить фильтры по имени создателя полета и диапазону дат, в котором должны были быть созданы полеты. У госорганизаций функционал еще больше, через графический интерфейс, они могут управлять непосредственно грузами. Им доступны такие функции, как создание и редактирование грузов, просмотр списка грузов и их удаление. Для каждой из этих функций присутствует свой метод, отправляемый на основной веб-сервис.

# **ОПИСАНИЕ ИНТЕРФЕЙСА**

Главное меню приложения включает пункты, которые доступны в зависимости от роли пользователя (рис. 9, 10).

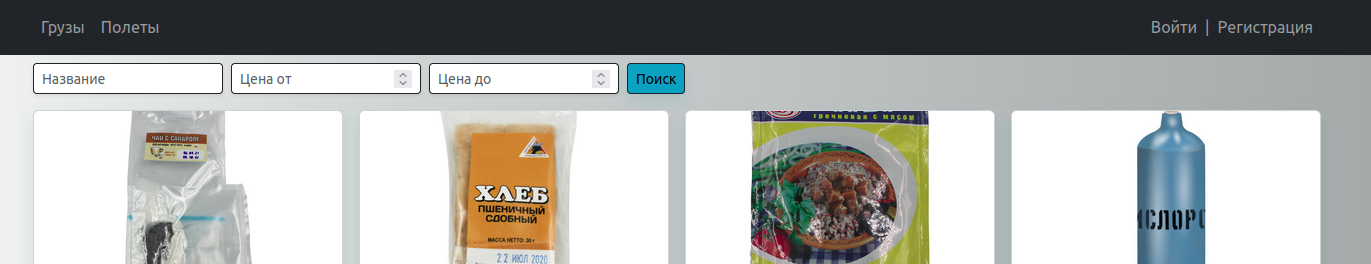


Рисунок 9 - Главное меню (гость)

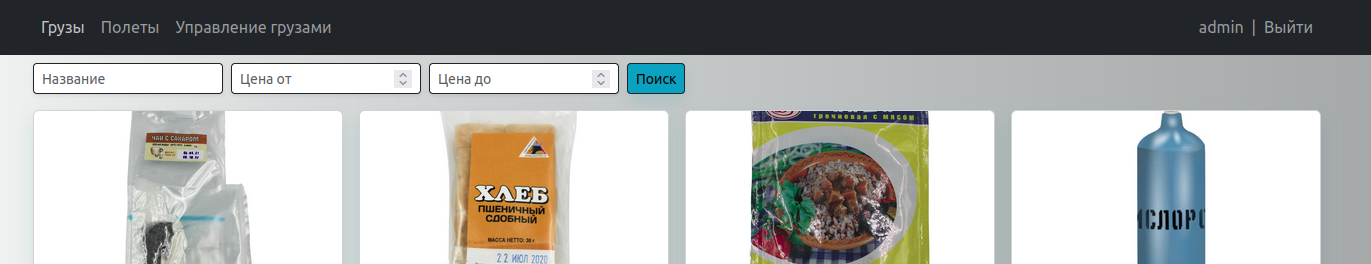


Рисунок 10 - Главное меню (госорганизация)

На странице с формой авторизации (рис. 11) отображается форма, через которую гость входит в свой аккаунт. При успешном вводе логина и пароля гость получает JWT, который используется при отправлении запросов.

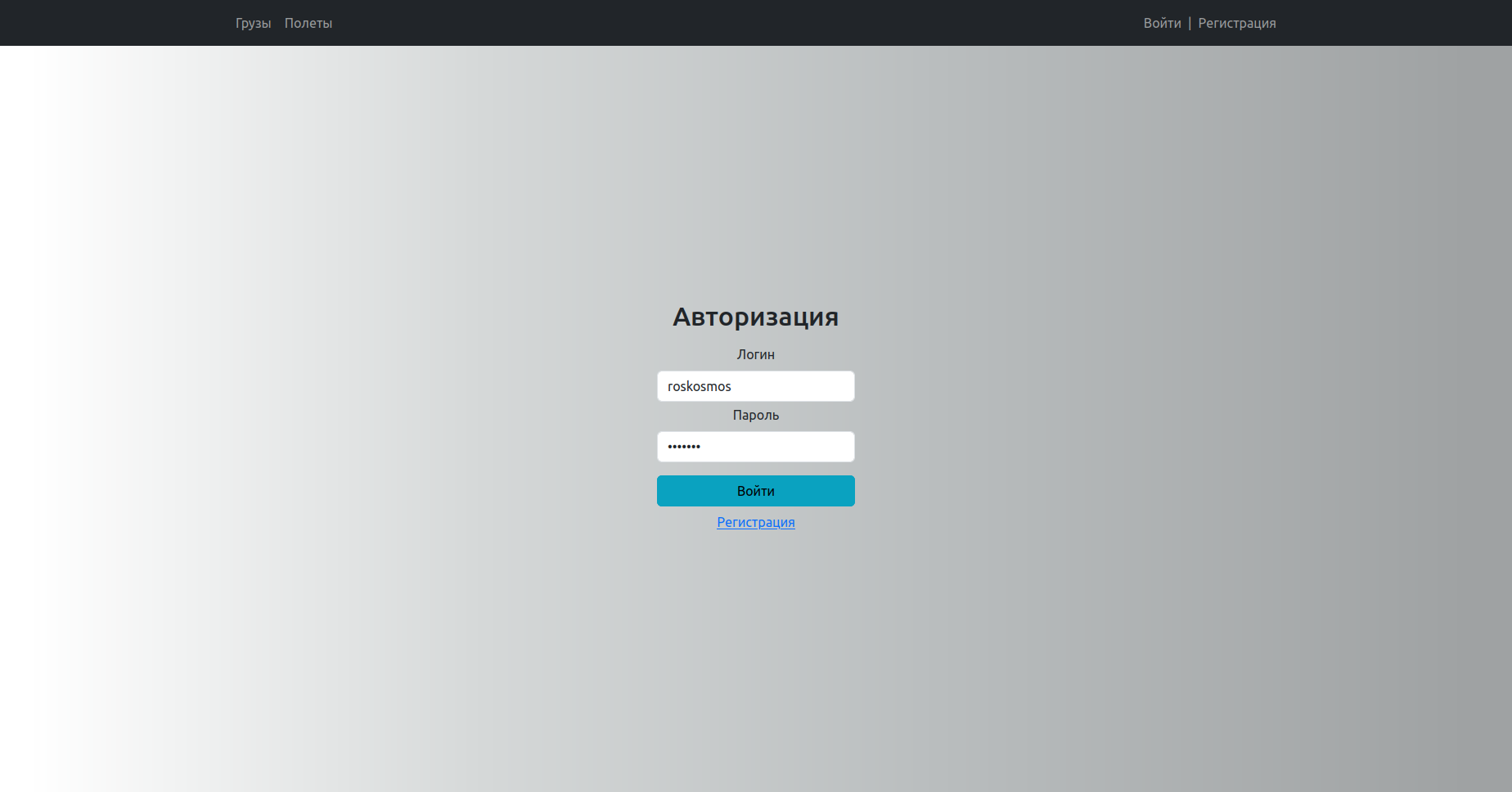
****

Рисунок 11 - Страница авторизации

На странице с формой регистрации (рис. 12) гости могут завести аккаунт. Для этого нужно указать логин, и пароль.

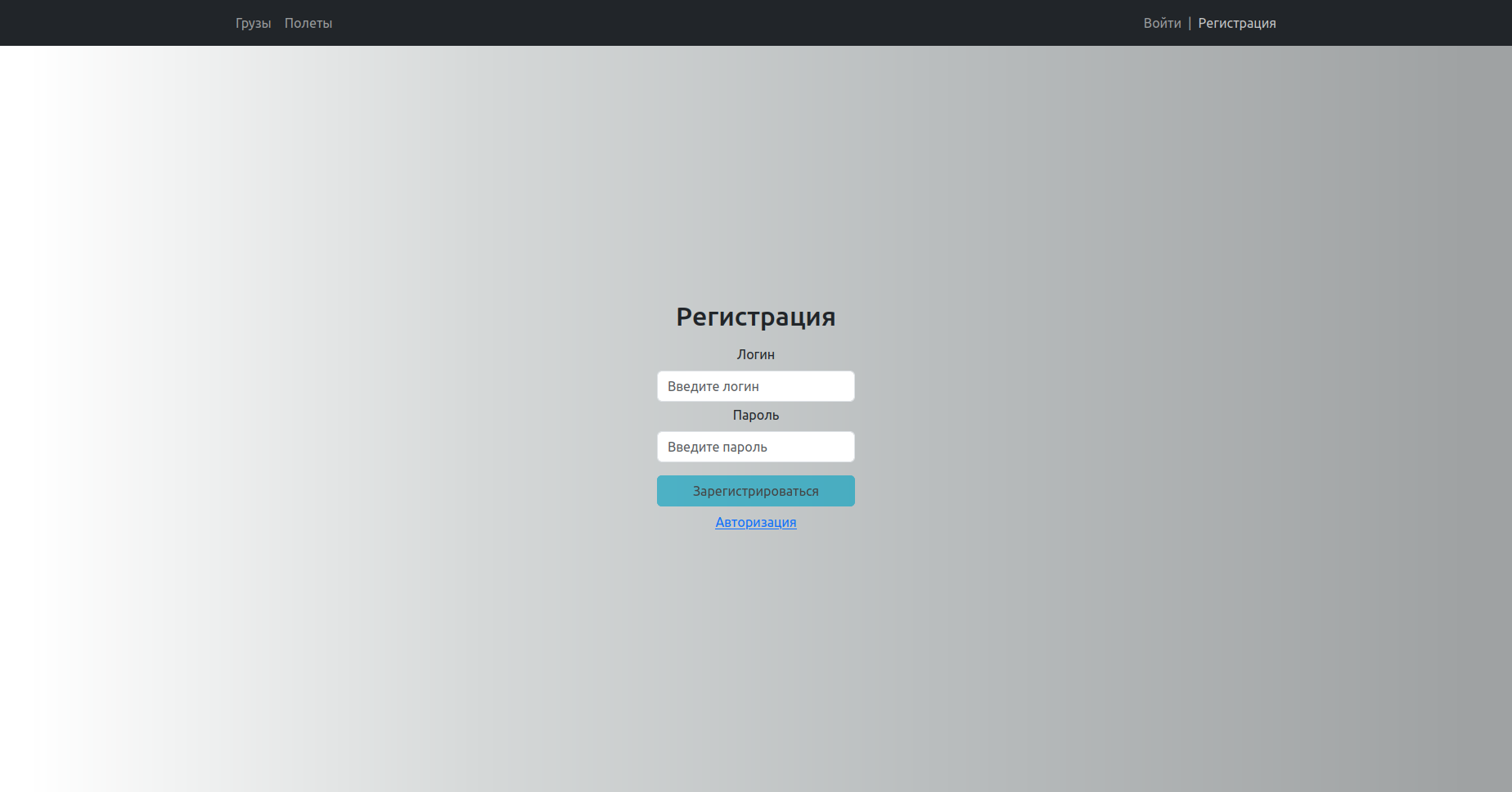
****

Рисунок 12 - Страница регистрации

На странице со списком грузов (рис. 13) отображается список грузов в виде карточек. У каждой карточки есть кнопка «Подробнее», переносящая пользователя на страницу с подробной информацией о грузе, и кнопка «добавить в корзину». Сверху находится фильтр грузов, там можно ввести фрагмент названия груза, который должен присутствовать в названии грузов, а также диапазон цен.

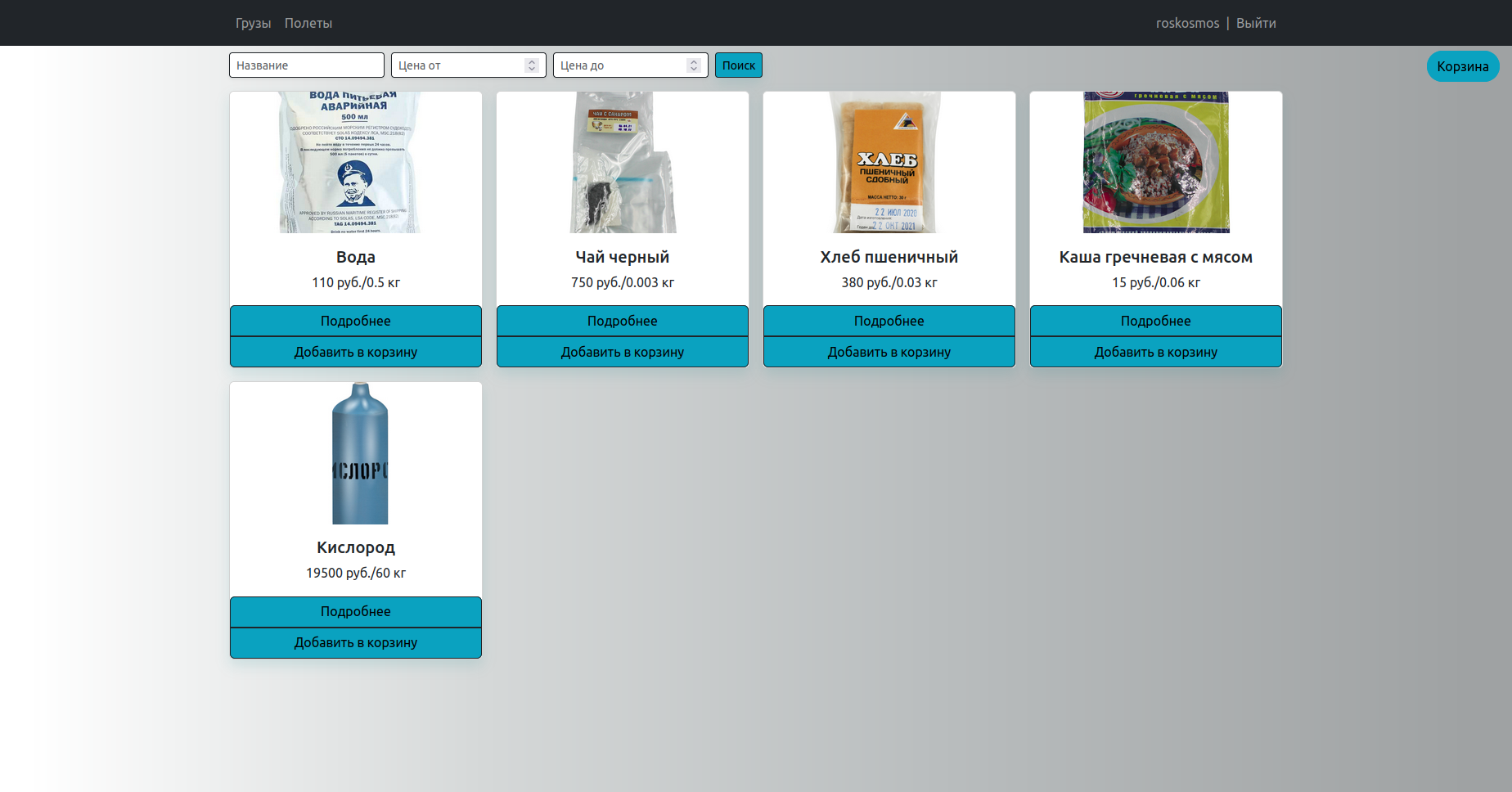
****

Рисунок 13 - Страница со списком грузов

На странице с подробным описанием груза (рис. 14) отображается подробная информация о грузе: название, категория, описание, цена, масса, объем и изображение.

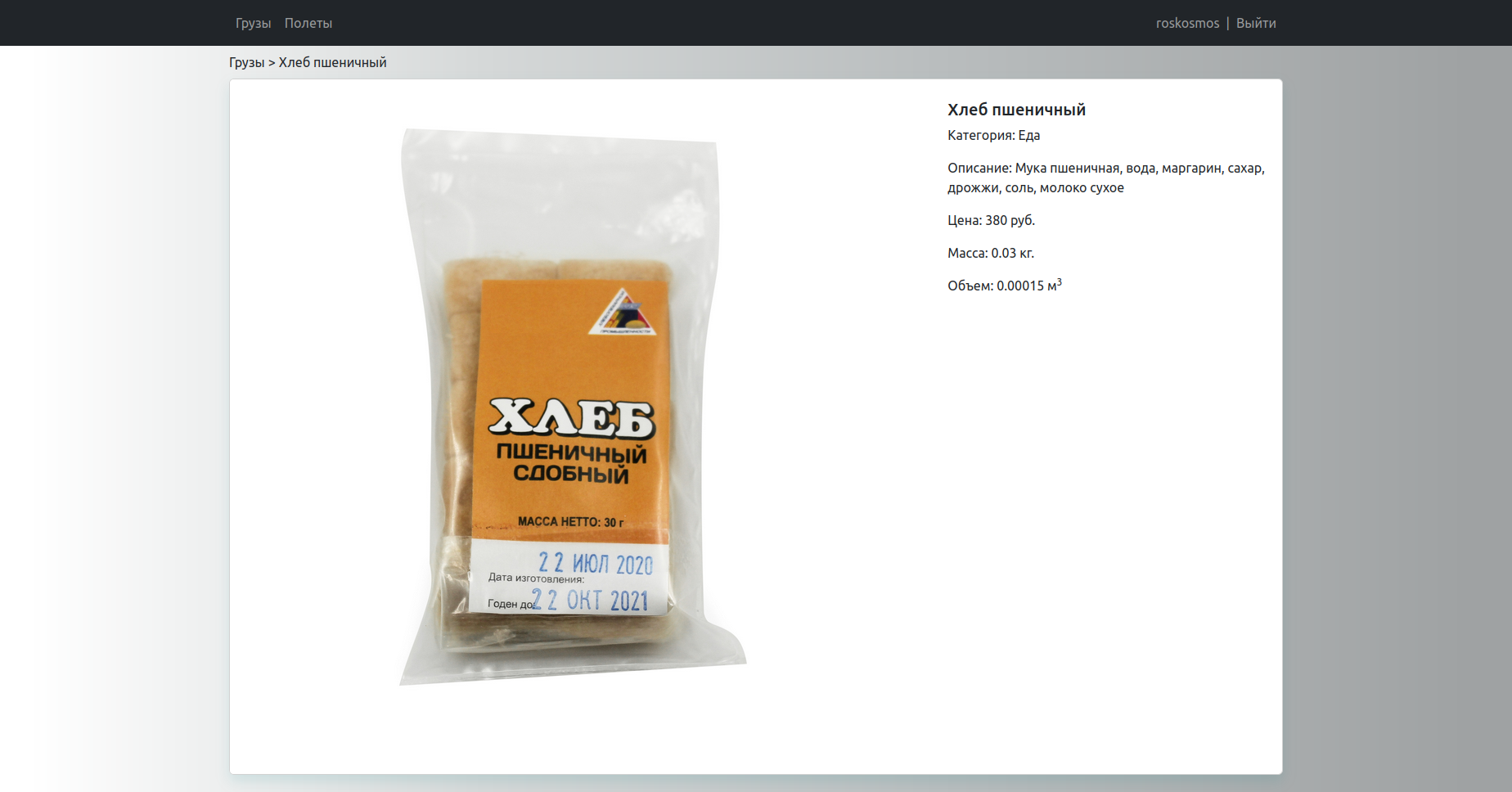
****

Рисунок 14 - Страница с подробным описанием груза

На странице со списком полетов (рис. 15, 16) отображается список полетов. В зависимости от типа пользователя этот список будет функционально отличаться. Так, для заказчиков отображается список созданных ими полетов: статус, дата создания, дата формирования, дата завершения, тип ракеты и кнопка для открытия подробной информации.

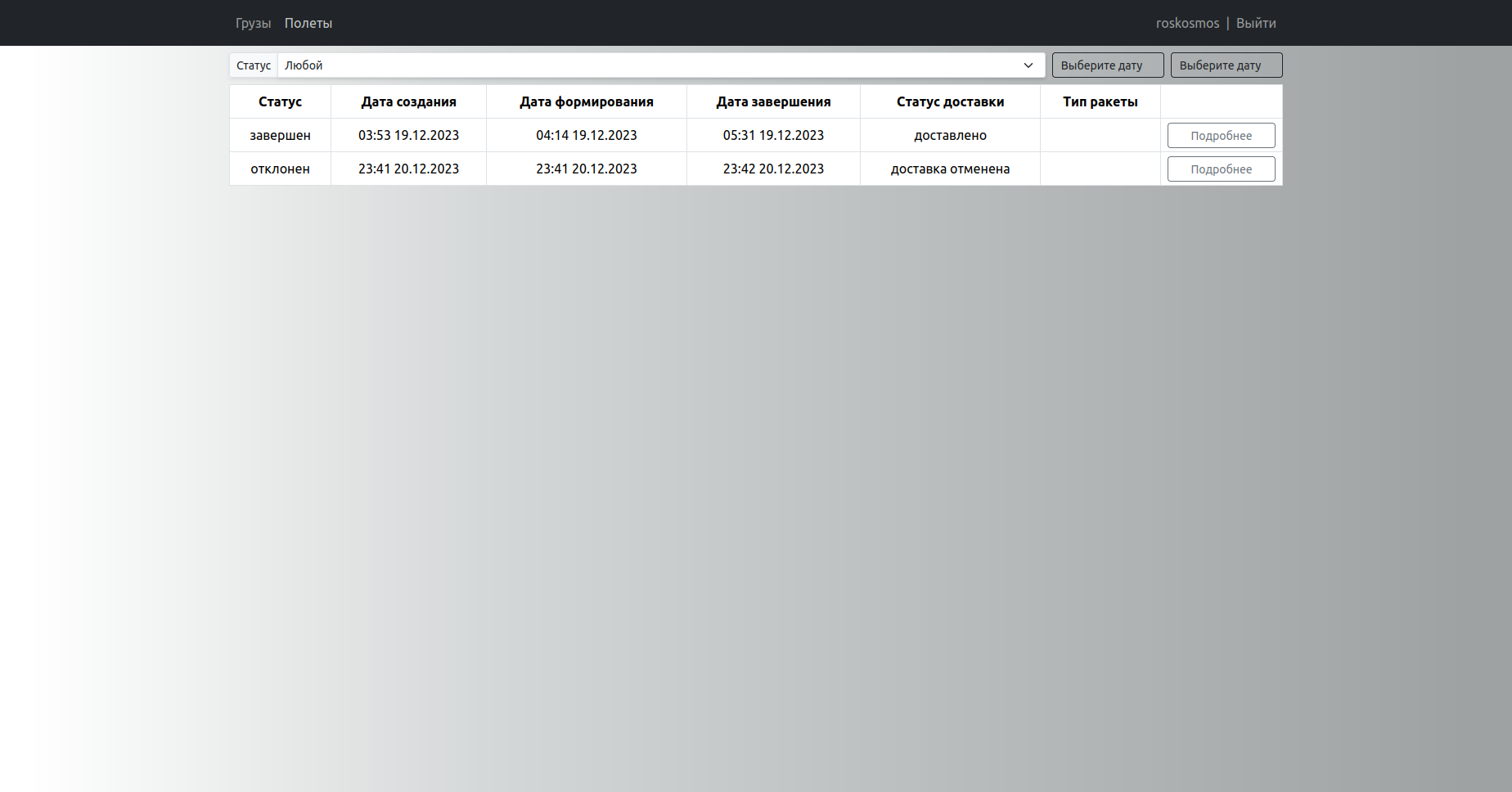
****

Рисунок 15 - Страница со списком полетов (заказчик)

Для госорганизаций функция этой страницы шире. Для них отображается список всех полетов всех заказчиков с подробной информацией: автор полета, статус, дата создания, дата формирования, дата завершения, тип ракеты, статус доставки. У полетов со статусом «черновик» также присутствует кнопка для установления статуса. Есть фильтрация по заказчику.

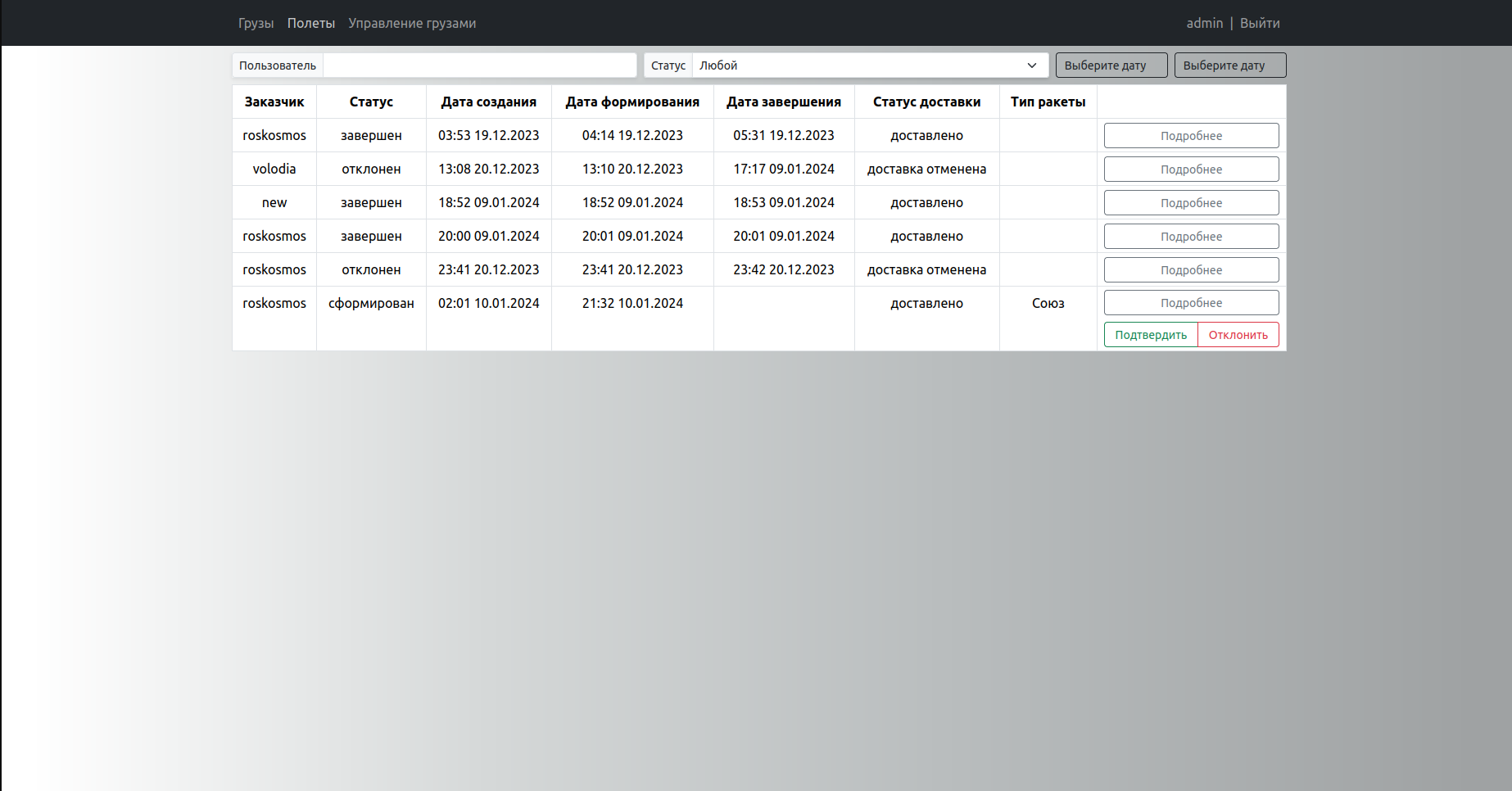
****

Рисунок 16 - Страница со списком полетов (госорганизация)

На странице с подробным описанием полета (рис. 17) отображается подробная информация о полете: список выбранных грузов и их количества в виде карточек, а также статус полета.

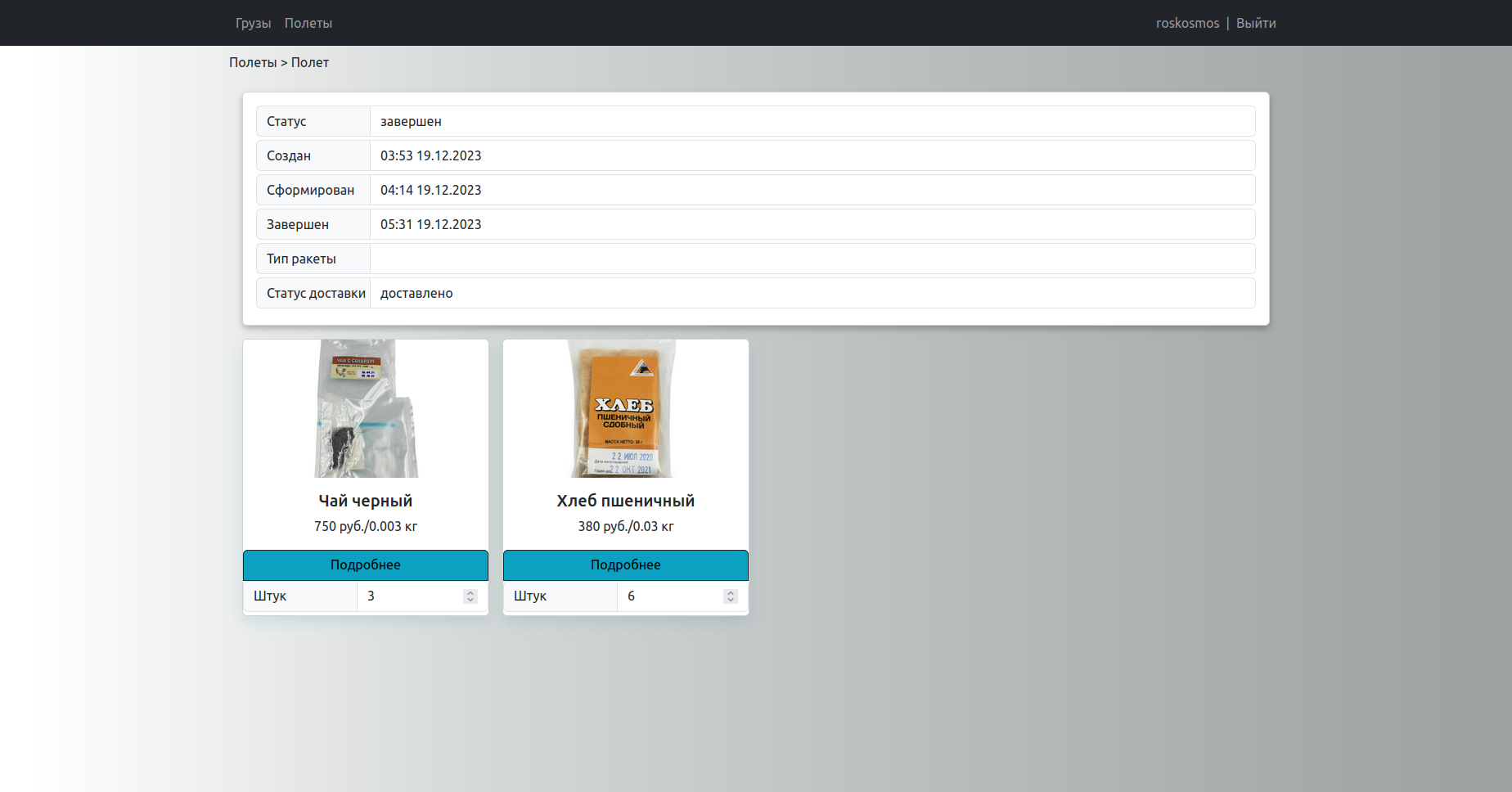
****

Рисунок 17 - Страница с подробным описанием полета

На странице с таблицей грузов (рис. 18) модератор может в компактном и удобном формате просмотреть список всех грузов, существующих в системе. Отображаются следующие поля: название, категория, изображение, цена, масса. Также можно открыть существующий груз или перейти на страницу создания нового (рис. 19).

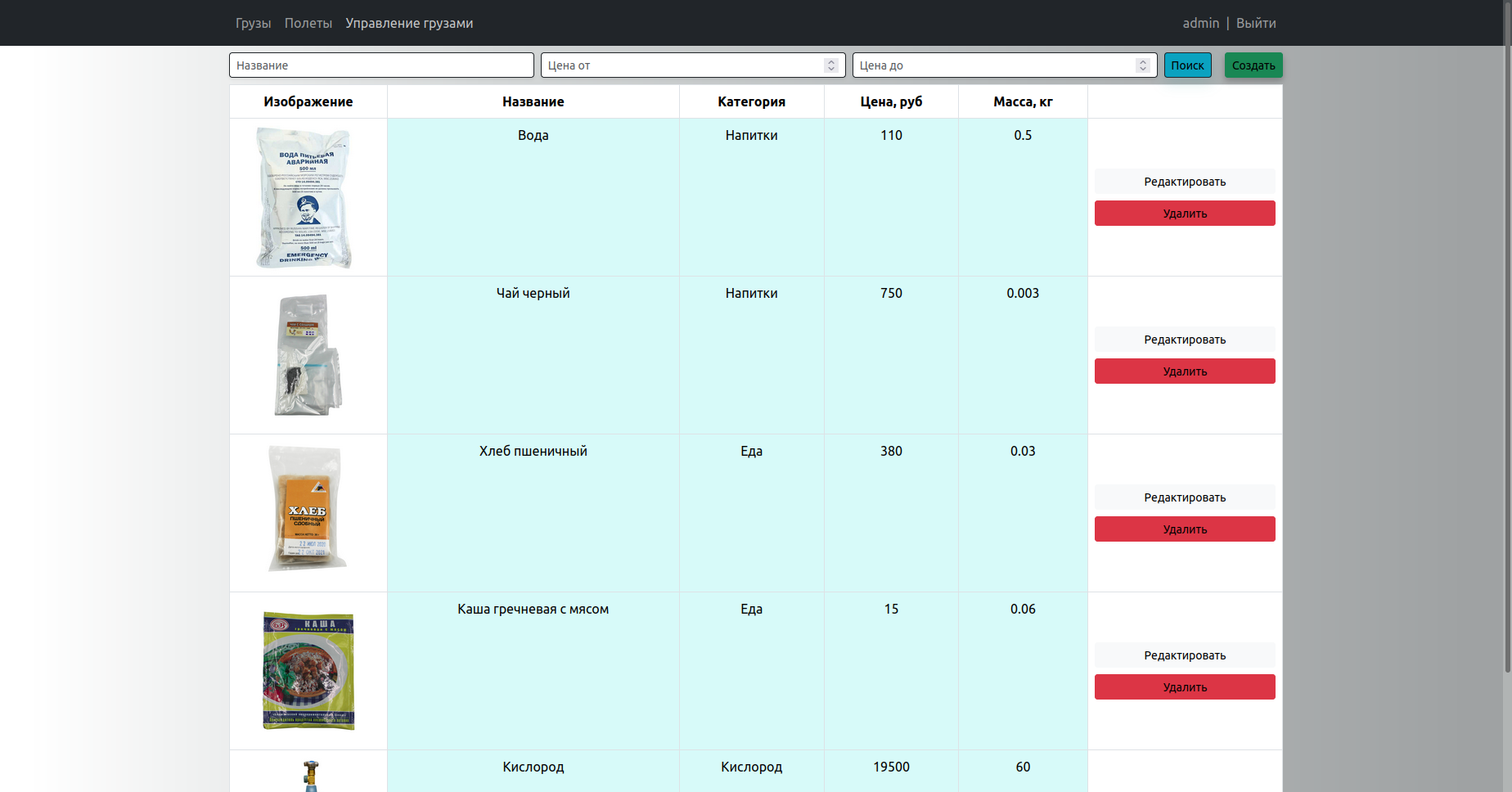
****

Рисунок 18 - Страница с таблицей грузов

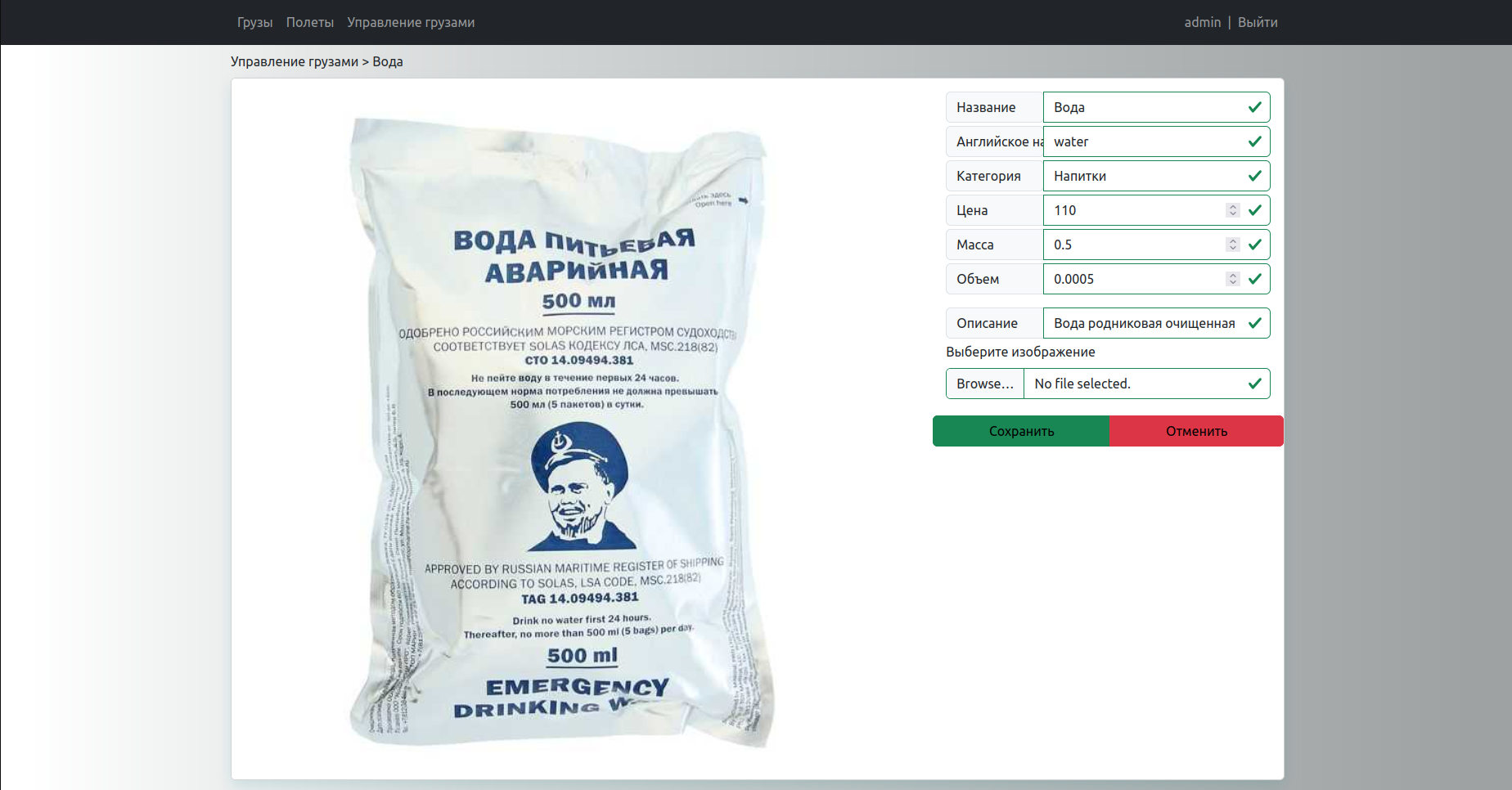
****

Рисунок 19 - Страница редактирования/создания груза

На странице редактирования/создания груза (рис. 19) госорганизация может отредактировать существующий или создать новый груз.

# **ЗАКЛЮЧЕНИЕ**

В ходе работы были достигнуты следующие результаты:

1. Был разработан дизайн приложения с помощью набора стилей CSS и HTML тегов.
2. База данных была создана и расположена в docker контейнере.
3. Был создан веб-сервис на golang 1.21.0, с использованием веб-фреймворка Gin.
4. Разработан интерфейс гостя с использованием технологии React Framework и подключен к веб-сервису.
5. Приложение интерфейса было развернуто на сервисе Github Pages по ссылке https://maximivanchenko18.github.io/InternetApps\_5sem\_Frontend/.
6. В веб-сервис добавлена авторизация через JWT, а методы задокументированы через Swagger.
7. Реализован интерфейс заказчика. Доступ к нему имеют только авторизованные пользователи.
8. Выделенный сервис был разработан и развернут в отдельном виртуальном окружении Python.
9. Реализован интерфейс гос организации для подтверждения новых полетов и редактирования грузов.
10. Было реализовано десктопное приложение на React Native, повторяющее интерфейс веб-приложения на React.
11. Подготовлен набор документации, включающий РПЗ, ТЗ и набор UML диаграмм.
12. Исходный код проекта доступен в GitHub https://github.com/MaximIvanchenko18/InternetApps\_5sem.

# **СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ**

* 1. Онлайн-магазин CosmoFeed.Ru космических сублимированных продуктов [Электронный ресурс] // CosmoFeed. URL: https://cosmofeed.ru/kosmicheskaja-produkcija/ (дата обращения: 15.09.2023).
  2. Грузы для МКС. Запуски 2021 года [Статья] // Pikabu. URL: https://pikabu.ru/story/gruzyi\_dlya\_mks\_zapuski\_goda\_106\_vsego\_18\_ot\_rossii\_8575393?ysclid=lrb59rkdij27495805 (дата обращения: 21.09.2023).
  3. Полное практическое руководство по Docker [Электронный ресурс] // Habr. URL: https://habr.com/ru/articles/310460/ (дата обращения: 20.10.2023).
  4. Руководство по React Native [Электронный ресурс] // Habr. URL: https://habr.com/ru/companies/plarium/articles/458118/ (дата обращения: 03.11.2023).
  5. Руководство по React [Электронный ресурс] // Metanit. URL: https://metanit.com/web/react/ (дата обращения: 10.10.2023).
  6. Quick Start – React [Электронный ресурс] // React. URL: https://react.dev/learn (дата обращения: 12.10.2023).
  7. Руководство по Gin Web Framework [Электронный ресурс] // GoLang. URL: https://github.com/gin-gonic/gin (дата обращения: 11.09.2023).
  8. Руководство по PostgreSQL [Электронный ресурс] // Metanit. URL: https://metanit.com/sql/postgresql/ (дата обращения: 03.08.2023).

**Приложение. Техническое задание**

| Gerb-BMSTU_01 | **Министерство науки и высшего образования Российской Федерации Федеральное государственное бюджетное образовательное учреждение высшего образования**  **«Московский государственный технический университет**  **имени Н.Э. Баумана**  **(национальный исследовательский университет)»**  **(МГТУ им. Н.Э. Баумана)** |
| --- | --- |

ФАКУЛЬТЕТ **ИНФОРМАТИКА И СИСТЕМЫ УПРАВЛЕНИЯ**

КАФЕДРА **СИСТЕМЫ ОБРАБОТКИ ИНФОРМАЦИИ И УПРАВЛЕНИЯ**

Техническое задание

по дисциплине «Разработка интернет приложений»

Тема предметной области «Грузовые корабли к МКС»

Выполнил:

Студент группы ИУ5-53Б

Иванченко М.Д.

(дата, подпись)

Проверил:

Преподаватель

Канев А.И.

(дата, подпись)

Москва, 2024 г.

# **Цель**

Разработка системы доставки грузов к международной космической станции (МКС), состоящей из веб-сервиса, веб-приложения, кроссплатформенного приложения и сервиса по определению результата доставки грузов, для обеспечения эффективной работы и координации системы отправок грузов.

# **Назначение**

Система предназначена для управления доставками грузов к МКС, которые были сформированы зарегистрировавшимися пользователями (заказчиками). Система предоставляет интерфейс для просмотра и изменения информации как о самих грузах, так и о полетах. Модераторы (главы госкорпораций) могут изменять список доступных для доставки грузов и выполнять модерацию полетов, то есть завершать и отклонять их. Главная задача системы – автоматизация процесса регистрации полетов и контроль за их осуществление к МКС.

# **Задачи**

* 1. Разработать дизайн приложения.
  2. Создать базу данных в PostgreSQL.
  3. Создать веб-сервис на golang 1.21.0.
  4. Реализовать интерфейс гостя на технологии React.
  5. Задеплоить на Github Pages.
  6. Добавить авторизацию и аутентификацию в веб-сервис.
  7. Реализовать интерфейс пользователя в React.
  8. Реализовать интерфейс модератора в React.
  9. Создать десктопное приложение на React Native.
  10. Создать выделенный сервис для расчета статуса доставки на Django.
  11. Подготовить набор документации, включающий РПЗ, ТЗ и набор диаграмм.

# **Методы веб-сервиса**

| № | Метод | URL | Описание | Входные данные | Выходные данные |
| --- | --- | --- | --- | --- | --- |
| 4.1 Грузы | | | | | |
| 4.1.1 | GET | api/cargo?name=&low\_price=&high\_price= | Список грузов, отфильтрованный по названию (name), диапазону цен (от low\_price до high\_price) | Параметры uri (name, low\_price, high\_price) | {“draft\_flight”: string,  [{“cargo\_id”: string, ”name”: varchar,  “category”: varchar, “photo”: varchar,  “price”: integer, “weight”: float, “capacity”: float, “description”: varchar},  …  ]} |
| 4.1.2 | POST | api/cargo/ | Добавление груза.  Доступно модератору | {”name”: varchar,  “category”: varchar, “photo”: varchar,  “price”: integer, “weight”: float, “capacity”: float, “description”: varchar},  Auth jwt | {“cargo\_id”: string, ”name”: varchar,  “category”: varchar, “photo”: varchar, “price”: integer, “weight”: float, “capacity”: float, “description”: varchar} |
| 4.1.3 | GET | api/cargo /<cargo\_id> | Возвращает груз по уникальному id | Параметр uri (cargo\_id) | {“cargo\_id”: string, ”name”: varchar,  “category”: varchar, “photo”: varchar, “price”: integer, “weight”: float, “capacity”: float, “description”: varchar} |
| 4.1.4 | PUT | api/cargo /<cargo\_id> | Обновляет информацию о грузе.  Доступно модератору | Параметр uri (cargo\_id)  {”name”: varchar,  “category”: varchar, “photo”: varchar,  “price”: integer, “weight”: float, “capacity”: float, “description”: varchar},  Auth jwt | {“cargo\_id”: string, ”name”: varchar,  “category”: varchar, “photo”: varchar, “price”: integer, “weight”: float, “capacity”: float, “description”: varchar} |
| 4.1.5 | DELETE | api/cargo /<cargo\_id> | Меняет статус груза на «Удален». Доступно модератору | Auth jwt |  |
| 4.1.6 | POST | api/cargo/<cargo\_id>/add\_to\_flight | Добавляет груз в полет-черновик. Доступно пользователю и модератору | Auth jwt | [{“flight\_id”: string} :  {“cargo\_id”: string, ”name”: varchar,  “category”: varchar, “photo”: varchar, “price”: integer, “weight”: float, “capacity”: float, “description”: varchar}  …] |
| 4.2 Полеты | | | | | |
| 4.2.1 | GET | api/flights?status=&form\_date\_start=&form\_date\_end= | Возвращает список полетов.  Доступно пользователю, модератору | Параметры uri (Status, form\_date\_star, form\_date\_end),  Auth jwt | [{  “flight\_id”: string,  “status”: varchar, “creation\_date”:datetime, “formation\_date”:datetime, “completion\_date”:datetime, “moderator\_id”: string,  “customer\_id”: string, “rocket\_type”:varchar, “shipment\_status”: varchar },  …] |
| 4.2.2 | GET | api/flights/<flight\_id> | Возвращает информацию об одном полете. Доступно пользователю, модератору | Параметр uri (flight\_id),  Auth jwt | {  {“flight\_id”: string,  “status”: varchar, “creation\_date”:datetime, “formation\_date”:datetime, “completion\_date”:datetime, “moderator\_id”: string,  “customer\_id”: string, “rocket\_type”:varchar}, “shipment\_status”: varchar},  [  {{“cargo\_id”: string, ”name”: varchar,  “category”: varchar, “photo”: varchar, “price”: integer, “weight”: float, “capacity”: float, “description”: varchar}, “cargo\_quantity”: integer}, ...]  } |
| 4.2.3 | PUT | api/flights/ | Обновление информации о полете. Доступно пользователю, модератору | { “rocket\_type”: varchar  },  Auth jwt |  |
| 4.2.4 | DELETE | api/flights/ | Меняет статус полета на «Удален». Доступно пользователю, модератору | Auth jwt |  |
| 4.2.5 | DELETE | api/flights/delete\_cargo/<cargo\_id> | Удаление груза из полета. Доступно пользователю, модератору | Параметр uri (cargo\_id),  Auth jwt |  |
| 4.2.6 | PUT | api/flights/change\_cargo/<cargo\_id> | Изменение количества груза в полете. Доступно пользователю, модератору | Параметр uri (cargo\_id),  {“quantity”: integer},  Auth jwt |  |
| 4.2.7 | PUT | api/flights/<flight\_id>/moderator\_confirm | Меняет статус с «Сформирован» на «Отклонен» или «Завершен». Доступно модератору | {  “confirm”: boolean  },  Auth jwt |  |
| 4.2.8 | PUT | api/flights/user\_confirm | Меняет статус с «Черновик» на «Сформирован».  Доступно пользователю, модератору | Auth jwt |  |
| 4.2.9 | PUT | api/flights/{flight\_id}/shipment | Устанавливает статус доставки грузов в полете | {  “flight\_id”: string, “shipment\_status”: boolean, “token”: string  } |  |
| 4.3 Пользователь | | | | | |
| 4.3.1 | POST | api/user/sign\_up | Регистрация | {  “login”: varchar,  “password”: varchar  } |  |
| 4.3.2 | POST | api/user/login | Авторизация | {  “login”: varchar,  “password”: varchar  } | Auth Jwt |
| 4.3.3 | POST | api/user/logout | Выход из аккаунта | Auth jwt |  |
| 4.4 Внешний сервис | | | | | |
| 4.4.1 | POST | api/shipment | Рассчитывает статус доставки грузов | {“flight\_id”: string} |  |

# **Функциональные требования**

* 1. Главное меню.
     1. Доступна всем пользователям.
     2. Действия
        1. Регистрация – переход на страницу 5.2. Для гостей.
        2. Войти – переход на страницу 5.3. Для гостей.
        3. Список грузов – переход на страницу 5.4. Для всех.
        4. Список выбранных грузов для доставки – переход на страницу 5.7. Для авторизованных пользователей.
        5. Список полетов – переход на страницу 5.6. Для авторизованных пользователей.
        6. Добавить груз – переход на страницу 5.8. Для модераторов.
        7. Таблица грузов – переход на страницу 5.9. Для модераторов.
  2. Страница регистрации.
     1. Доступна гостям.
     2. Действия
        1. Регистрация – отправляет запрос (метод 4.3.1), в котором передаются введенные данные.
        2. Войти – перенаправление на страницу 5.3.
  3. Страница авторизации.
     1. Доступна гостям.
     2. Действия
        1. Войти – отправляет запрос (метод 4.3.2).
        2. Регистрация – перенаправление на страницу 5.2.
  4. Страница со списком грузов (список грузов и панель фильтрации).
     1. Доступна всем пользователям.
     2. Информация о каждом грузе в виде карточки (метод 4.1.1)
        1. Изображение груза.
        2. Название груза.
        3. Цена груза.
        4. Масса груза.
     3. Действия
        1. Фильтрация грузов по названию и диапазону цены (метод 4.1.1).
        2. «Корзина» – перенаправление на страницу полета 5.7.
        3. Добавить груз в полет (метод 4.1.6). Для авторизованных пользователей.
        4. Подробнее – перенаправление на страницу 5.5.
  5. Страница с описанием груза.
     1. Доступна всем пользователям.
     2. Вывод информации о грузе (метод 4.1.3).
        1. Изображение груза.
        2. Название груза.
        3. Категория груза.
        4. Описание груза.
        5. Цена груза.
        6. Масса груза.
        7. Объем груза.
  6. Страница со списком полетов.
     1. Доступна только авторизованным пользователям.
     2. Вывод информации о полетах (метод 4.2.1).
        1. Статус полета.
        2. Статус доставки.
        3. Дата создания.
        4. Дата формирования.
        5. Дата завершения.
        6. Кем сформирована. Для модераторов.
        7. Список добавленных грузов.
     3. Действия
        1. Открыть полет – перенаправление на страницу 5.7
        2. Установить статус «завершен» или «отклонен» (метод 4.2.7). Для модераторов.
        3. Фильтр полетов по статусу и/или диапазону даты формирования (метод 4.2.1). Для авторизованных пользователей. А также по и/или создателю. Для модераторов.
  7. Страница с описанием полета.
     1. Доступна только авторизованным пользователям.
     2. Информация о полете и списке грузов в нем (метод 4.2.2).
        1. Изображение груза.
        2. Название груза.
        3. Цена груза.
        4. Масса груза.
     3. Действия над черновиком
        1. Сформировать полет (метод 4.2.8).
        2. Удалить полет (4.2.4).
        3. Удалить груз из полета (4.2.5).
        4. Изменить количество груза в полете (4.2.6).
        5. Открыть груз – перенаправление на страницу 5.5
  8. Страница изменения/создания грузов. Модератор может изменить, удалить или создать груз.
     1. Доступна только модераторам.
     2. Информация о грузе (метод 4.1.3).
        1. Изображение груза.
        2. Название груза.
        3. Категория груза.
        4. Описание груза.
        5. Цена груза.
        6. Масса груза.
        7. Объем груза.
     3. Изменение полей груза (метод 4.1.4).
  9. Страница с таблицей грузов.
     1. Доступна только модераторам.
     2. Информация о грузах (метод 4.2.2).
        1. Изображение груза.
        2. Название груза.
        3. Категория груза.
        4. Описание груза.
        5. Цена груза.
        6. Масса груза.
        7. Объем груза.
     3. Действия
        1. Удалить груз (метод 4.1.5).
        2. Редактировать груз – перенаправление на страницу 5.8.
        3. Добавить груз – перенаправление на страницу 5.8.

# **Требования к аппаратному обеспечению**

* 1. Серверная часть
     1. Процессор минимум 2-ядерный от 2 Ггц
     2. Оперативная память от 8 Гб.
     3. Место на жестком диске от 2 Гб.
  2. Клиентская часть
     1. Процессор частотой от 1 ГГц.
     2. Оперативная память от 2 Гб.

# **Требования к программному обеспечению**

* 1. Серверная часть
     1. ОС: Linux Ubuntu 22.04.3+/Windows 10+
     2. Python 3.11.7
     3. Django 5.0.1
     4. Docker 18.09.0 и выше
     5. СУБД: PostgreSQL 15.0
     6. Golang 1.21.0 и выше
     7. minio RELEASE.2022-10-15T19-57-03Z
     8. nginx 1.19.2
     9. redis 7.2.2
     10. Node.js 20.10.0 и выше
  2. Клиентская часть
     1. OC: Windows 10+, Linux Ubuntu 22.04.3+, MacOs 14.2.1+
     2. Браузер: Google Chrome 100+, Firefox 44+, Safari 12+, Yandex 20+