|  |  |
| --- | --- |
| **­­** | **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\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**РАСЧЕТНО-ПОЯСНИТЕЛЬНАЯ ЗАПИСКА**

***К ДОМАШНЕМУ ЗАДАНИЮ***

***НА ТЕМУ:***

***Севморпуть, заявки на проводку ледоколом \_\_\_\_\_\_\_***

***\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_***

***\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_***

***\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_***

***\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_***

Студент \_\_ИУ5Ц-71Б\_\_\_\_\_\_\_ **\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_Марков И.С.\_\_\_**

(Группа) (Подпись, дата) (И.О.Фамилия)

Руководитель курсовой работы **\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_А.И. Канев\_\_\_\_\_\_**

(Подпись, дата) (И.О.Фамилия)

*2024 г.*

АННОТАЦИЯ

Расчётно-пояснительная содержит 26 страниц. С приложениями объем составляет 44 страницы. Работа включает в себя 8 диаграмм и 17 изображений системы. В процессе выполнения было использовано 10 источников.

Объектом разработки является система создания заявок на проводку кораблей ледоколом. Данный программный модуль позволяет обрабатывать созданные заявки с целью дальнейшей проводки.

Цель работы заключается в исследовании различных сервисов кораблей, а также создание набора программного обеспечения и сопутствующих веб сервисов для обработки изображений.

В работе была разработана архитектура веб сервиса, рассчитанного на создания заявок, разработан интерфейс для общения с данным веб сервисом, разработан и развернут веб сервер, нативное приложение и прогрессивное веб приложение, способные общаться с данным веб сервисом.

Пояснительная записка содержит 2 приложения.

СОДЕРЖАНИЕ

Оглавление

[АННОТАЦИЯ 2](#_Toc185840670)

[СОДЕРЖАНИЕ 3](#_Toc185840671)

[ВВЕДЕНИЕ 4](#_Toc185840672)

[1 ПРЕДМЕТНАЯ ОБЛАСТЬ 6](#_Toc185840673)

[2 АРХИТЕКТУРА 12](#_Toc185840674)

[3 АЛГОРИТМЫ 16](#_Toc185840675)

[4 ОПИСАНИЕ ИНТЕРФЕЙСА 18](#_Toc185840676)

[ЗАКЛЮЧЕНИЕ 26](#_Toc185840677)

[СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ 27](#_Toc185840678)

[ПРИЛОЖЕНИЕ А ТЕХНИЧЕСКОЕ ЗАДАНИЕ 28](#_Toc185840679)

[1. Введение 29](#_Toc185840680)

[2. Назначение разработки 29](#_Toc185840681)

[3. Стадии и этапы разработки 29](#_Toc185840682)

[4. Требования к функциональным характеристикам 30](#_Toc185840683)

[5. Требования к составу и параметрам технических средств 34](#_Toc185840684)

[6. Требования к информационной и программной совместимости 35](#_Toc185840685)

[ПРИЛОЖЕНИЕ Б СПИСОК HTTP МЕТОДОВ 36](#_Toc185840686)

ВВЕДЕНИЕ

Корабли играют ключевую роль в обеспечении транспортных и логистических операций, особенно в арктических и северных районах, где использование ледоколов становится необходимым для безопасного прохождения судов через замерзшие участки водных путей. С увеличением объема международных перевозок и усложнением навигационных условий растет потребность в эффективной координации и управлении проводкой судов ледоколами. Особенно важным в этих обстоятельствах становится использование специализированных сервисов, помогающих создавать эти проводки.

Целью данного проекта является разработка системы, предоставляющей интерфейс для учета заявок на проводку кораблей ледоколами, включающей в себя веб-сервис, веб-приложение и десктопное приложение Tauri.

Данная система предназначена для компаний-судовладельцев и навигационных служб, занимающихся организацией безопасного прохождения кораблей через сложные участки водных путей при помощи ледоколов. Пользователям предлагается возможность создания заявок на услуги проводки, управления ими и отслеживания их статуса. Система обеспечивает автоматизированный способ создания, обработки и мониторинга заявок, упрощая взаимодействие между судовладельцами и операторами ледоколов.

Нефункциональные требования к разрабатываемой системе:

1. Должна поддерживаться кроссплатформенность.
2. Интерфейс системы и текст ошибок должны быть русифицированы.

В ходе работы необходимо выполнить следующие задачи:

* 1. Создать MVP и базового дизайна на основе shipsforsale.su;
  2. Создать базу данных для хранения информации о кораблях и заявках на их проводку ледоколом;
  3. Создать веб-сервис в бэкенде на Django (DRF);
  4. Реализовать авторизацию и хранение сессий в Redis;
  5. Разработать базовый SPA на React для гостя;
  6. Внедрить адаптивность, менеджер состояний Redux Toolkit, PWА, разработать Tauri приложение;
  7. Завершить разработку интерфейса пользователя в React, использовать для обращений к методам веб-сервиса Axios;
  8. Реализовать интерфейс оператора ледоколов в React;
  9. Разработать десктопное приложение Tauri
  10. Развернуть приложение при помощи GitHub Pages
  11. Подготовить набор документации, включающий РПЗ, ТЗ и набор диаграмм.
  12. Оформить git-репозиторий на сервисе GitHub, содержащий исходный код проекта.

1. ПРЕДМЕТНАЯ ОБЛАСТЬ

В последние годы наблюдается значительное увеличение судоходных операций в арктических водах, что связано с изменениями климата, растущими объемами добычи природных ресурсов и развитием транспортных маршрутов, таких как Северный морской путь (СМП). С 2013 по 2023 годы количество судов в арктических водах увеличилось на 111%, что обусловлено ростом объемов добычи углеводородов и расширением морских транспортных путей [1].

С увеличением судоходства в Арктике также возрастает потребность в более мощных и специализированных ледоколах, чтобы справляться с изменениями ледовых условий. Например, несмотря на таяние льдов в западной части СМП, восточная часть остаётся более сложной для судоходства из-за охлаждения температур и увеличения толщины льдов, что требует обновления и увеличения ледокольного флота, в том числе для обеспечения круглогодичного судоходства.

Флот ледоколов России ограничен, что снижает её возможности в Арктическом регионе. В связи с этим актуальным становится развитие технологий прогнозирования ледовых условий, особенно с использованием методов машинного обучения для предсказания взаимодействия судов с морским льдом. В статье «Автономная навигация в водах, покрытых льдом, с использованием предсказаний на основе взаимодействия судов с льдом» предложен инновационный подход, основанный на глубокой нейронной сети, который позволяет прогнозировать динамику льда в ответ на маневры судна [2]. Этот метод учитывает такие ключевые параметры, как водоизмещение судна (displacement), осадка (draft), скорость (speed) и ледовый класс (ice class), что делает прогнозы более точными и адаптированными к реальным условиям.

Система, описанная в статье, использует карты занятости (occupancy maps) для представления ледового поля и прогнозирования изменений в его структуре после взаимодействия с судном. Это позволяет минимизировать столкновения с льдом, снизить потерю кинетической энергии (kinetic energy loss) и импульс ударов (impulse), что особенно важно для судов с ограниченной ледовой прочностью. Кроме того, предложенный подход учитывает динамику льда, что делает его эффективным даже в условиях высокой концентрации льда (до 50%).

Основные метрики, используемые для оценки эффективности системы, включают расстояние пути (travel distance) — минимизация длины маршрута для экономии времени и топлива, потерю кинетической энергии (kinetic energy loss) — снижение энергии, теряемой при столкновениях, что уменьшает повреждения судна, импульс столкновений (impulse) — мера силы ударов, которую система стремится минимизировать, и приближенную работу (approximated work) — оценка изменений в ледовом поле, вызванных маневрами судна [2].

Результаты, представленные в статье, показывают, что предложенная система значительно превосходит традиционные методы навигации, такие как прямолинейное планирование (straight), планирование по скелету (skeleton) и решеточное планирование (lattice). В частности, в условиях высокой концентрации льда (40% и 50%) система демонстрирует устойчивую эффективность, минимизируя столкновения без значительного увеличения длины маршрута.

Пользователи системы (system users) включают операторов автономных судов (autonomous vessel operators), судовладельцев (ship owners), логистические компании (logistics companies), исследователей и разработчиков (researchers and developers), а также организации, занимающиеся мониторингом ледовой обстановки (ice monitoring organizations). Для операторов и судовладельцев система предлагает повышение безопасности навигации и снижение рисков повреждений судов. Логистические компании могут использовать её для оптимизации маршрутов и минимизации задержек. Исследователи и разработчики получают инструмент для дальнейшего совершенствования технологий автономной навигации, а организации, занимающиеся мониторингом льда, могут улучшить точность своих прогнозов.

Таким образом, использование глубокого обучения для прогнозирования взаимодействия судов с льдом позволяет не только повысить безопасность навигации, но и оптимизировать использование ресурсов, что особенно важно для операторов автономных судов, судовладельцев и логистических компаний, работающих в Арктике. В будущем планируется расширить модель, включив такие факторы, как волны (waves), океанские течения (ocean currents) и ветер (wind), для ещё более точного прогнозирования.

Кроме того, растущие геополитические и экологические вызовы требуют внедрения экологически чистых технологий и соблюдения Полярного кодекса, который регулирует использование тяжелых топлив и других аспектов судоходства в арктических водах [1].

Эти вызовы также подчеркивают необходимость разработки новых решений для автоматизации и управления ледокольным флотом, что будет способствовать улучшению навигации и сокращению рисков в регионе.

Пользователь (user), являющийся судовладельцем (ship owner), может выбрать корабль (ship) для проводки ледоколом (icebreaker wiring), указать порядок кораблей и отправить заявку (request) на рассмотрение оператора ледоколов (icebreaker operator). Каждая заявка (request) в начале создается как черновик, в который можно добавлять и из которого можно удалять корабли. Одна заявка может содержать несколько кораблей; тогда, в случае согласования заявки, будет оформлен заказ на проводку всех кораблей в заявке.

Судовладелец может посмотреть список всех созданных им заявок. В случае, если заявку составлена корректно, оператор ледоколов утверждает её. В противном случае – отклоняет. Оператор ледоколов может только либо согласовать, либо отклонить заявку.

Операторы ледоколов компании, предоставляющей услуги проводки, могут редактировать список кораблей, доступных для проводки: добавлять новый корабль, удалять устаревший, а также редактировать информацию о каждом отдельном корабле. Операторы ледоколов имеют возможность просматривать все заявки, созданные пользователями, а также фильтровать их по имени судовладельца, дате формирования и статусу. После того, как заявка была согласована или отклонена, оператор не может изменять её статус.

Функции пользователей с различными ролями описаны на диаграмме прецедентов (рисунок 1).

Гостям доступен просмотр списка доступных для проводки кораблей. Гости, прошедшие этап регистрации, являются судовладельцами. Судовладельцы могут добавлять корабль в заявку на проводку ледоколом, формировать заявку из черновика, а также просматривать список своих заявок.

Судовладелец выбирает корабли для проводки, затем на основе выбранного списка кораблей формирует заявку. После чего оператор ледоколов может просмотреть её, а также согласовать или отклонить. В случае некорректного составления заявки судовладелец имеет возможность удалить заявку. Возможные состояния статуса заявки отображены на диаграмме состояний (рисунок 2).

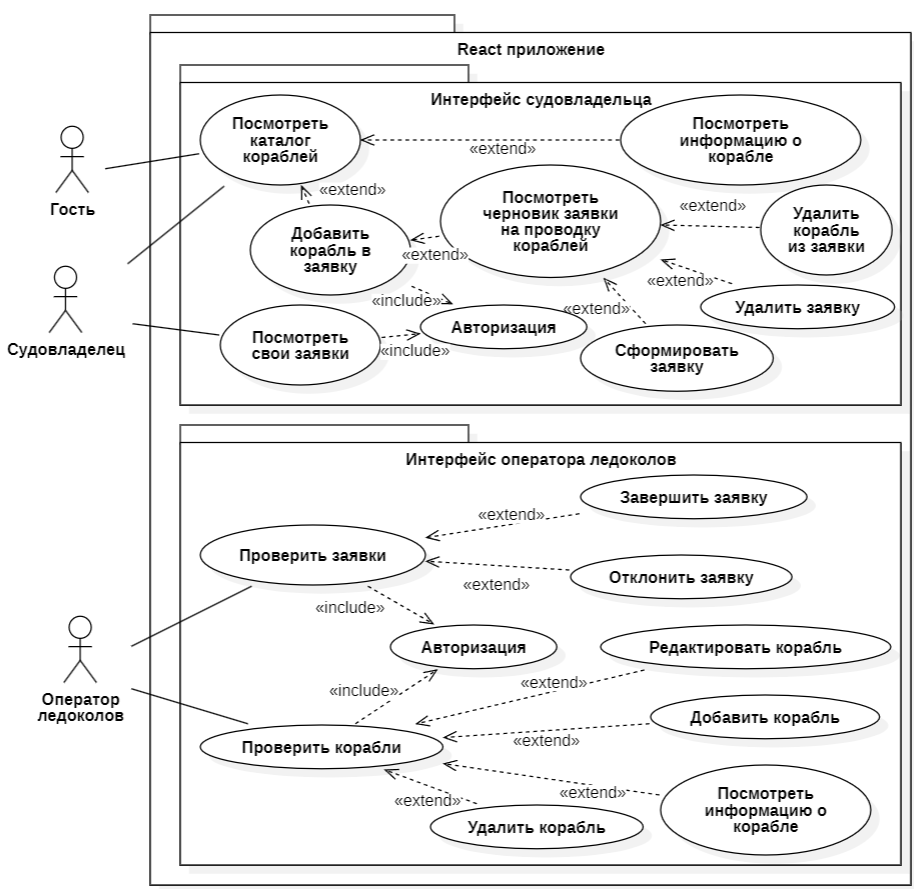


Рисунок 1 – Диаграмма прецедентов

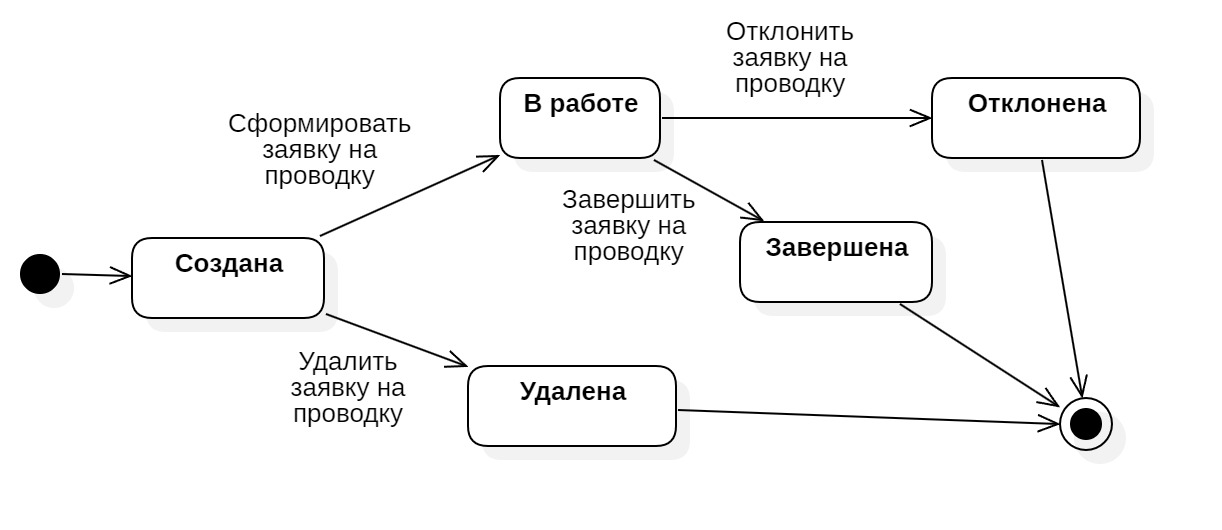


Рисунок 2 – Диаграмма состояний заявок

Заявки обрабатываются операторами компании. В результате обработки заявки её либо одобряют, либо отклоняют. Оператору доступны операции для работы с кораблями: просмотр всех кораблей, редактирование, создание и удаление кораблей, а также просмотр списка всех кораблей табличном виде.

Процесс оформления заявки отражен на диаграмме деятельности (рисунок 3).

В начале взаимодействия с системой оформления заявок на проводку кораблей судовладелец запрашивает список доступных для проводки кораблей. При необходимости судовладелец может добавить конкретный корабль в текущую черновую заявку. После чего может продолжить выбор кораблей из каталога или перейти к оформлению заявки. В случае, если судовладелец выбрал оформление заявки на проводку кораблей, ему необходимо указать порядок кораблей в заявке. После чего указать дату, когда необходимо провести проводку, начальную точку проводки и конечную точку проводки. По нажатии кнопки «Сформировать заявку» заявка формируется и становится доступна в интерфейсе оператора.

Оператор имеет возможность просматривать заявки пользователей, а также принимать решение об одобрении заявки, если она может быть выполнена компанией или отклонении заявки, если она не может быть выполнена компанией.

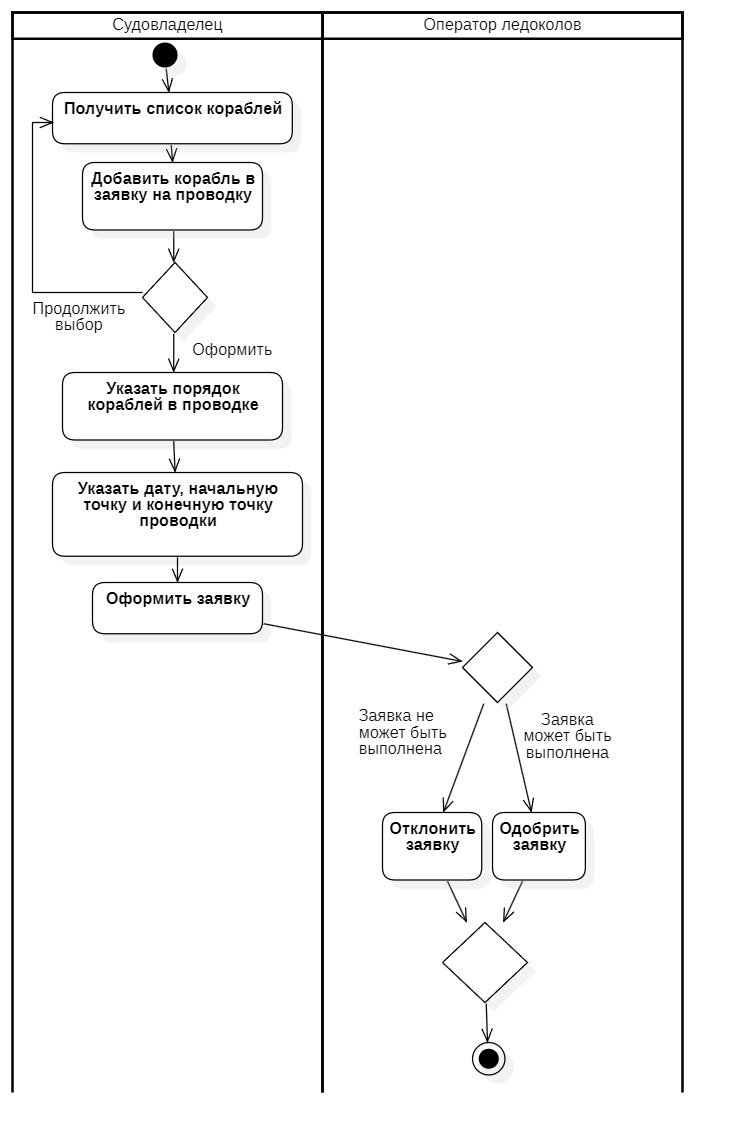


Рисунок 3 – Диаграмма деятельности

1. АРХИТЕКТУРА

Архитектура системы отображена на диаграмме развертывания (рисунок 4).

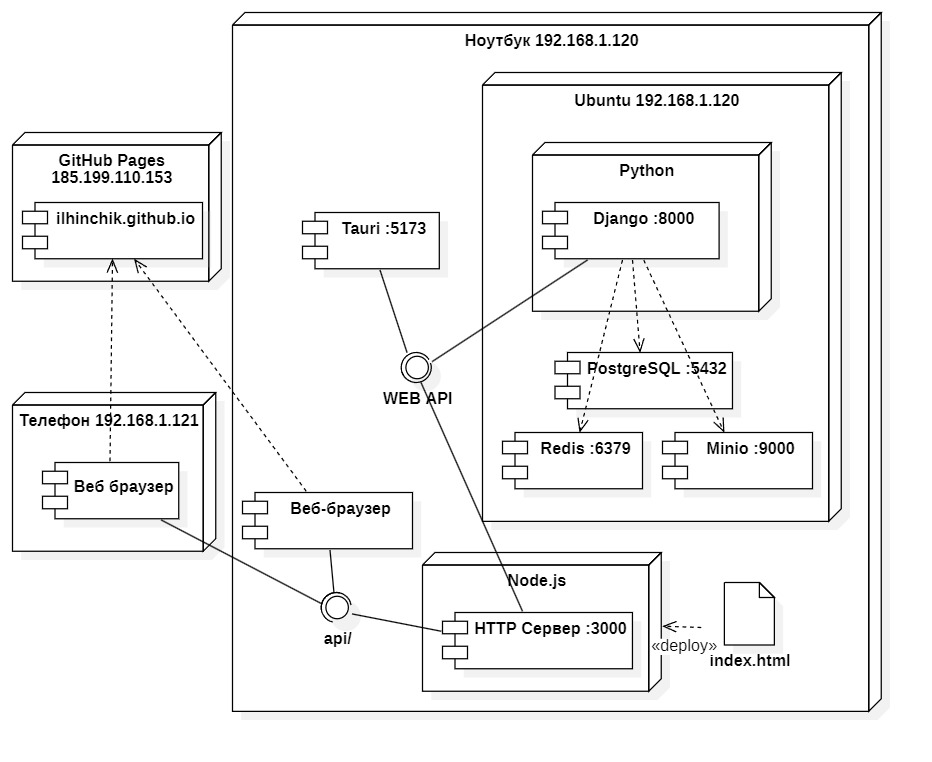


Рисунок 4 – Диаграмма развертывания

Веб-сервис, реализованный на фреймворке Django [3] с использованием DRF [4] связан с серверами Minio [5], Redis [6] и СУБД PostgreSQL [7]. В Redis хранятся активные сессии пользователей; при выходе пользователя из системы – удаляются.

Язык программирования Python [8] был выбран в силу его простоты, а также популярности. Использование Django обусловлено тем, что этот фреймворк имеет достаточно широкий возможности и является стандартом индустрии. Веб-сервис на Django является общим для веб-сервера и десктопного приложения Tauri [9].

Данные хранятся в СУБД PostgreSQL. Она была выбрана, поскольку, на ровне с Python и Django она является стандартом современной индустрии разработки.

Структура данных отражена на ER диаграмме (рисунок 5). Модель программ (услуг) представляет собой набор полей, отражающих свойства кораблей. Данные о кораблях хранятся в таблице Ship. Для хранения в одной заявке на проводку нескольких кораблей используется промежуточная таблица связи ShipIcebreaker, которая реализует связь М-М. Таблица Icebreaker представляет собой список заявок на проводку кораблей. Данные о пользователях система – судовладельцах и операторах ледоколов – хранятся в таблице User.

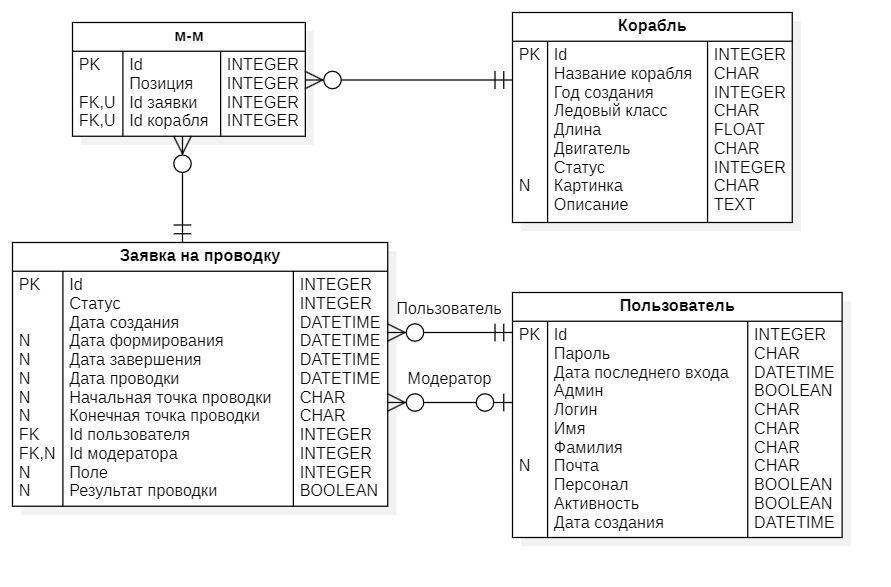


Рисунок 5 – ER-диаграмма

Веб-север реализован с использованием языка TypeScript и фреймворка React [10]. Устройство бэкенда приложения разработанной системы приведено на диаграмме классов бэкенда (рисунок 6). Пользователи взаимодействуют с доменами. Домены связаны с моделями. Модели имеют связи с таблицами в базе данных.

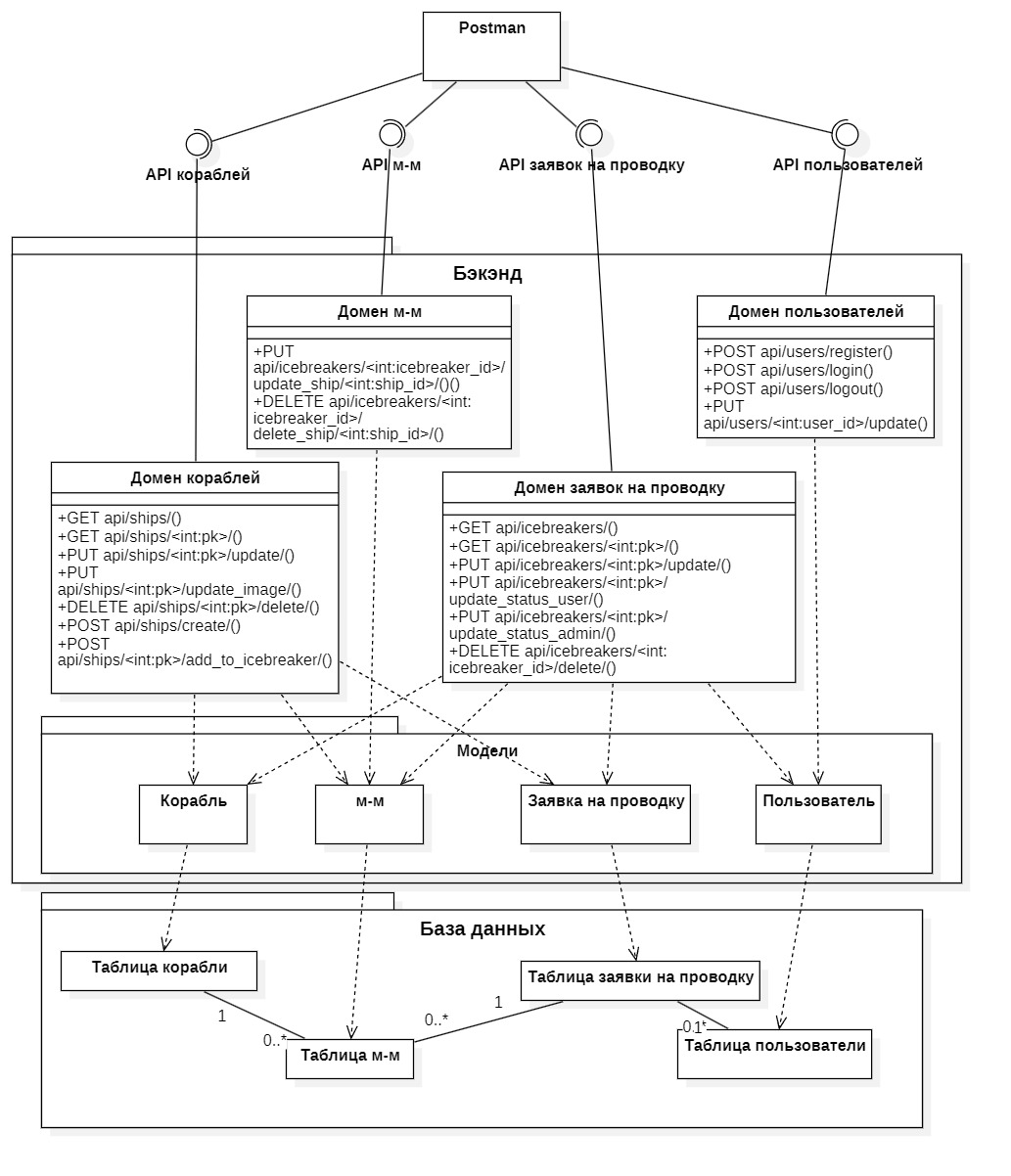


Рисунок 6 – Диаграмма классов бэкенда

Связь фронтенда и бэкенда отражена на диаграмме классов фронтенда (рисунок 7). Каждая страница связана с API, которое используется для взаимодействия с данными на соответствующей странице.

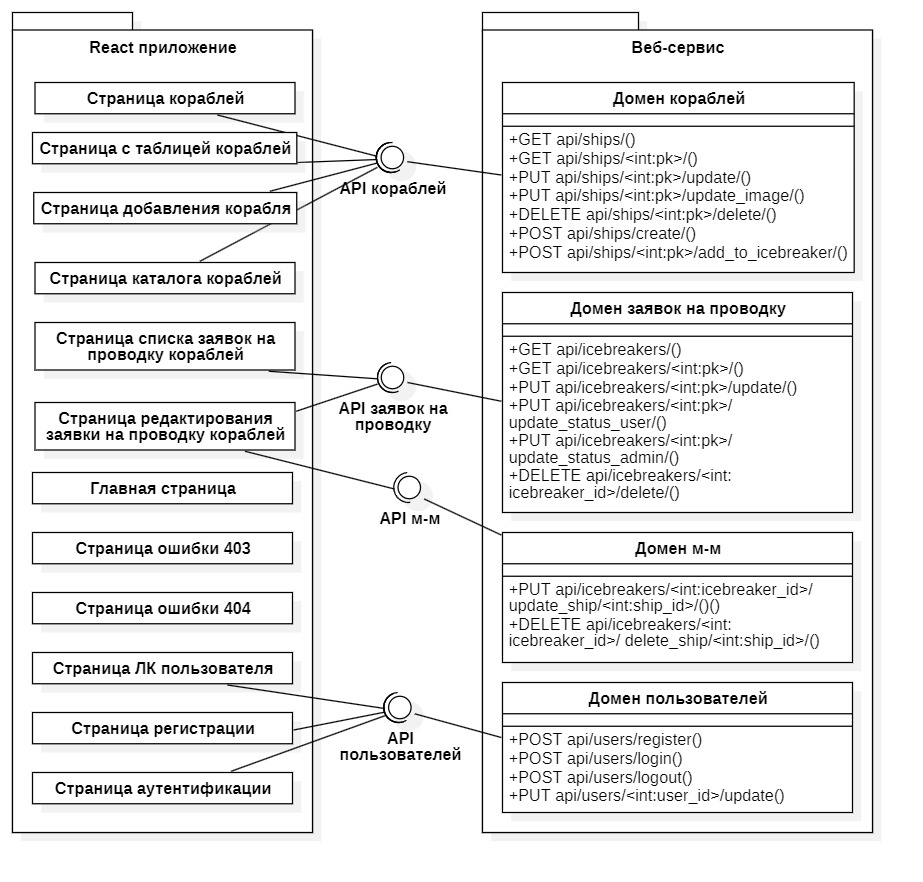


Рисунок 7 – Диаграмма классов фронтенда

1. АЛГОРИТМЫ

Алгоритм работы разработанной системы отображен на диаграмме последовательности (рисунок 8). В основе системы лежит веб-сервис, реализующий внутри себя всю бизнес-логику. Он предоставляет доступ к методам из следующих доменов: корабли, заявки на проводку кораблей, корабль в заявке, пользователи. Методы следуют правилам REST API.

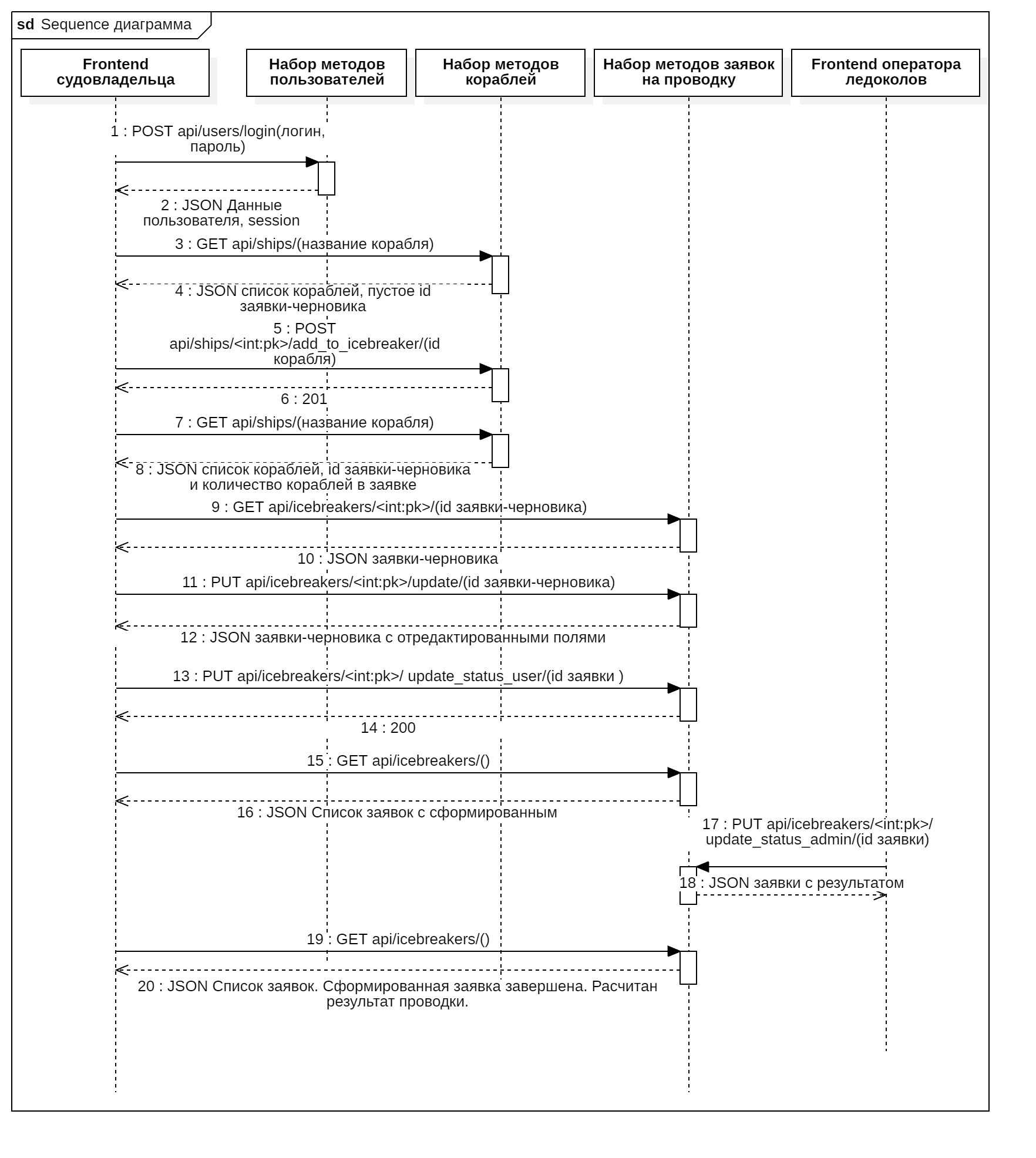


Рисунок 8 – Диаграмма последовательности

В начале бизнес-процесса происходит аутентификация пользователя. При помощи графического интерфейса пользователь отправляет запрос, передавая в нем логин и пароль. Если аккаунт с указанными данными существует в базе, на клиент возвращается информация о пользователе и устанавливаются куки с полем session\_id (идентификатор текущей сессии) в ответном запросе. Если же аккаунта с введённым логином не существует, или пароль введен неверно, на клиент возвращается ошибка. В таком случае пользователю надо пройти регистрацию, либо ввести верный пароль. На этом же этапе происходит проверка: является пользователь судовладельцем или оператором ледоколов. Графический интерфейс пользователя запрашивает у веб-сервиса список услуг (программ), которые возвращаются в JSON формате. Судовладелец выбирает корабль, который хочет добавить в заявку на проводку, и, нажимая на кнопку «добавить» в графическом интерфейсе, отправляет запрос на добавление корабля в свою черновую заявку. Этот процесс может продолжаться несколько раз.

Когда пользователь определится со списком необходимого для проводки кораблей, а также указал порядок кораблей, дату, начальную точку и конечную точку проводки, он нажимает кнопку «сформировать» в графическом интерфейсе. После чего приложение отправляет на веб-сервис запрос на формирование заявки. Клиент может отслеживать статус сформированных заявок на соответствующей странице в графическом интерфейсе.

Процесс рассмотрения заявок операторами так же происходит при помощи графического интерфейса. Операторы могут просматривать список всех созданных пользователями заявок; согласовывать или отклонять их при помощи соответствующих кнопок. Доступна фильтрация по логинам пользователей, а также статусам и дате формирования заявок. Операторы имеют возможность редактировать список кораблей, доступного для проводки, а также изменять данные о каждом корабле. Для каждой из этих возможностей существует соответствующий метод, отправляемый на веб-сервис.

1. ОПИСАНИЕ ИНТЕРФЕЙСА

Главное меню приложения включает пункты, которые доступны в зависимости от роли пользователя (рисунки 9, 10, 11).



Рисунок 9 – Меню приложения (для гостя)



Рисунок 10 – Меню приложения (для судовладельца)



Рисунок 11 – Меню приложения (для оператора)

На странице с формой регистрации (рисунок 12) отображается форма, при помощи которой гость может создать новый аккаунт. После успешной регистрации открывается форма аутентификации.

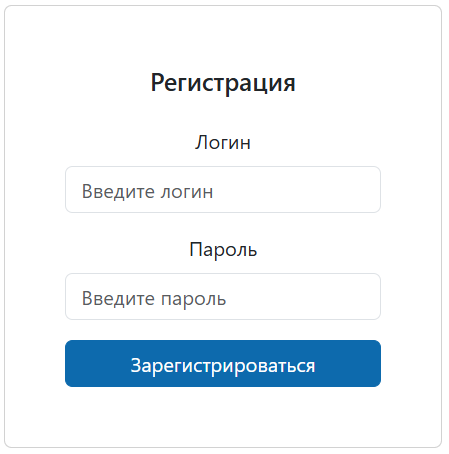


Рисунок 12 – Страница регистрации

На странице с формой входа (рисунок 13) отображается форма, через которую пользователь может войти в свой аккаунт. При успешном вводе данных аккаунта на клиент приходят куки с идентификатором текущей сессии.

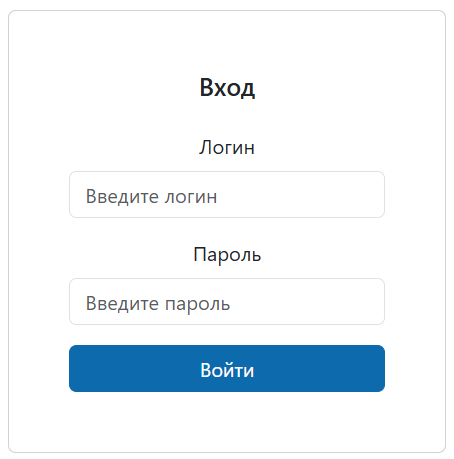
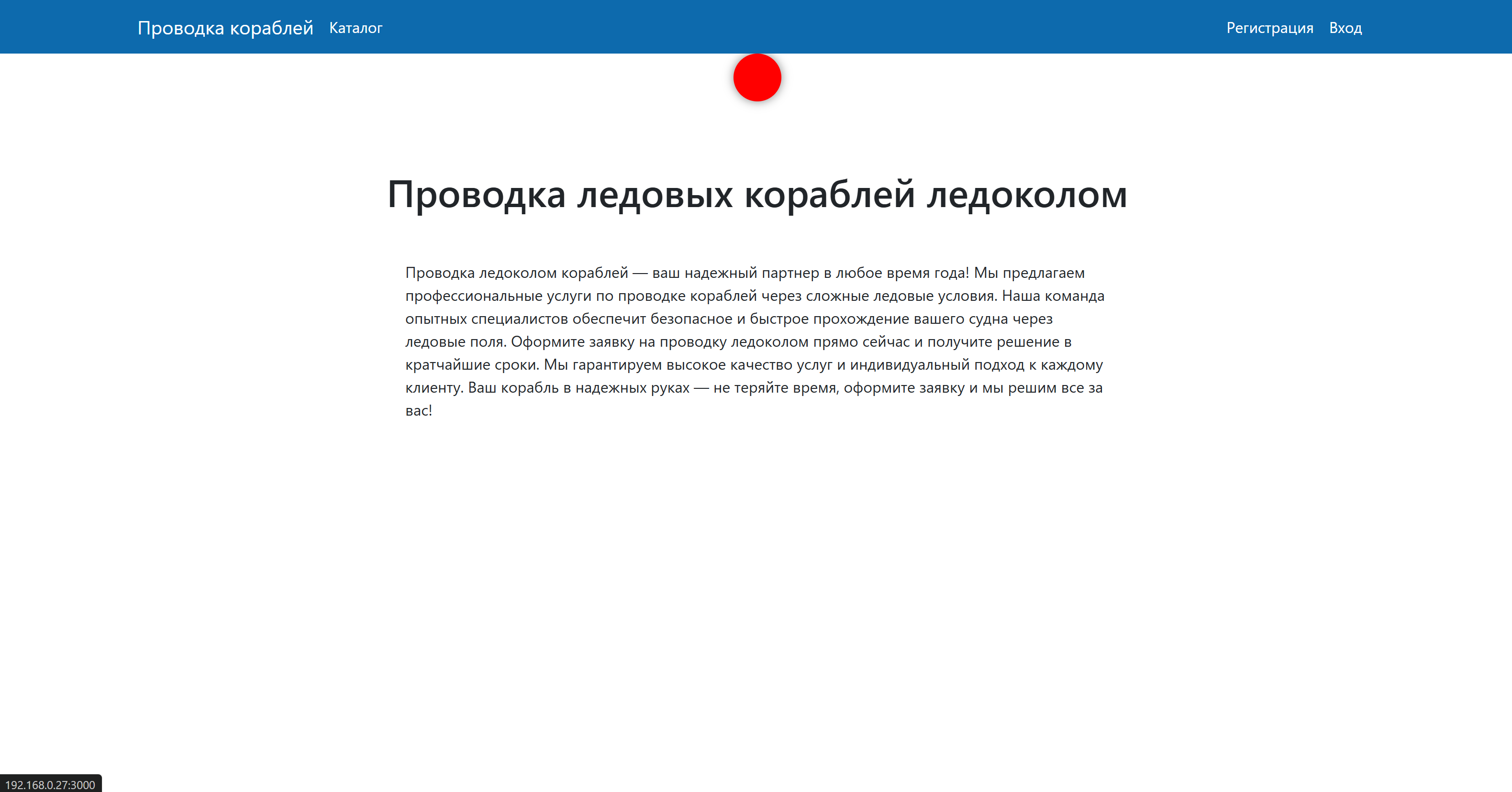


Рисунок 13 – Страница входа

После входа в аккаунт пользователь попадает на главную страницу (рисунок 14). На этой странице располагается описание разработанной системы.

 Рисунок 14 – Главная страница

Страница каталога кораблей (рисунок 15) содержит список программ в виде карточек. При нажатии на кнопку «Подробнее» на любой карточке, открывается страница с подробным описанием выбранного корабля. Также с помощью нажатия на кнопку «Добавить» судовладелец может добавлять программы в черновик заявки. Вверху страницы находится строка для поиска корабля по названию.

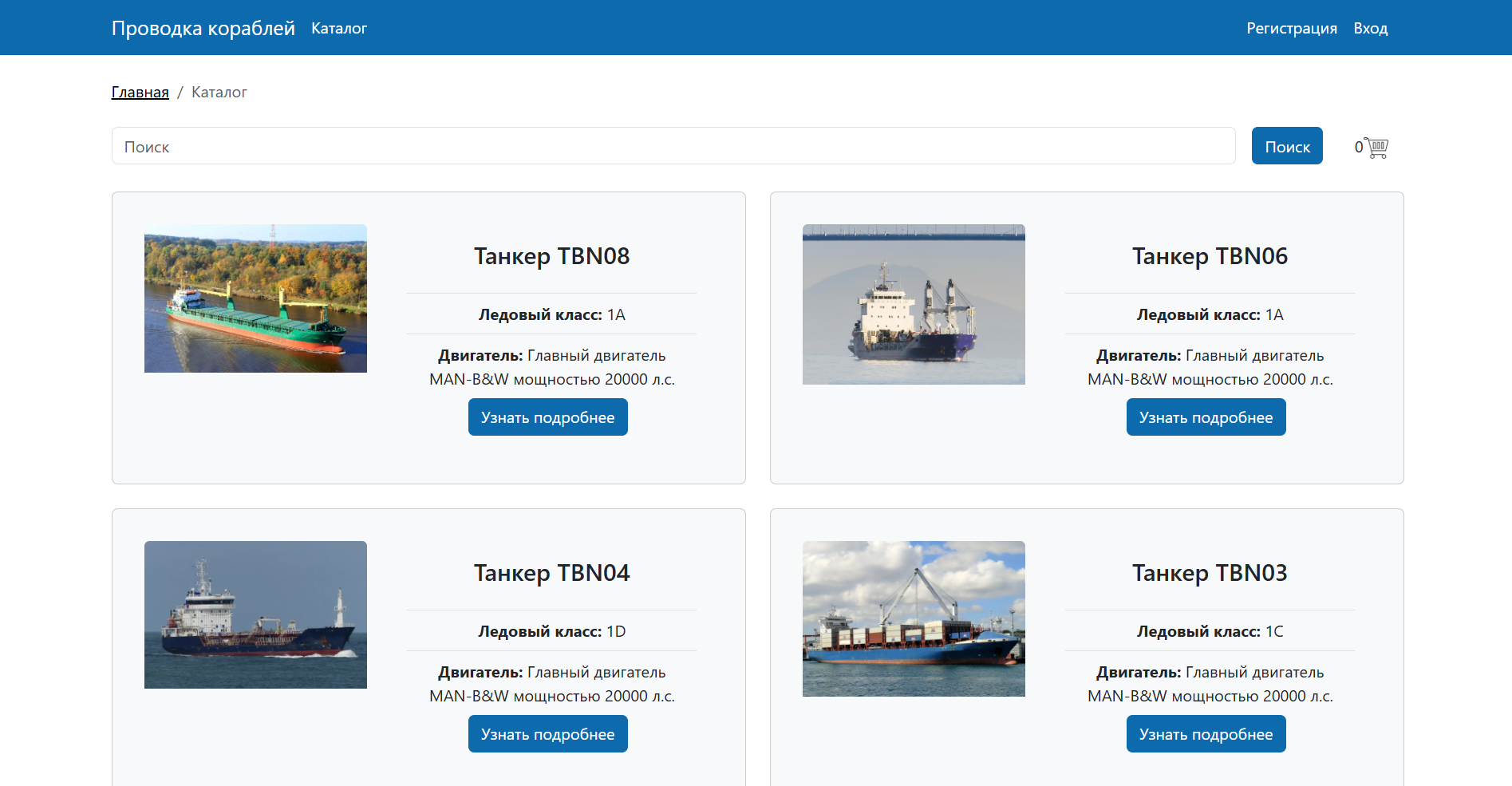


Рисунок 15 – Страница каталога кораблей

На странице с подробным описанием программы выводится основная информация о корабля (рисунок 16).

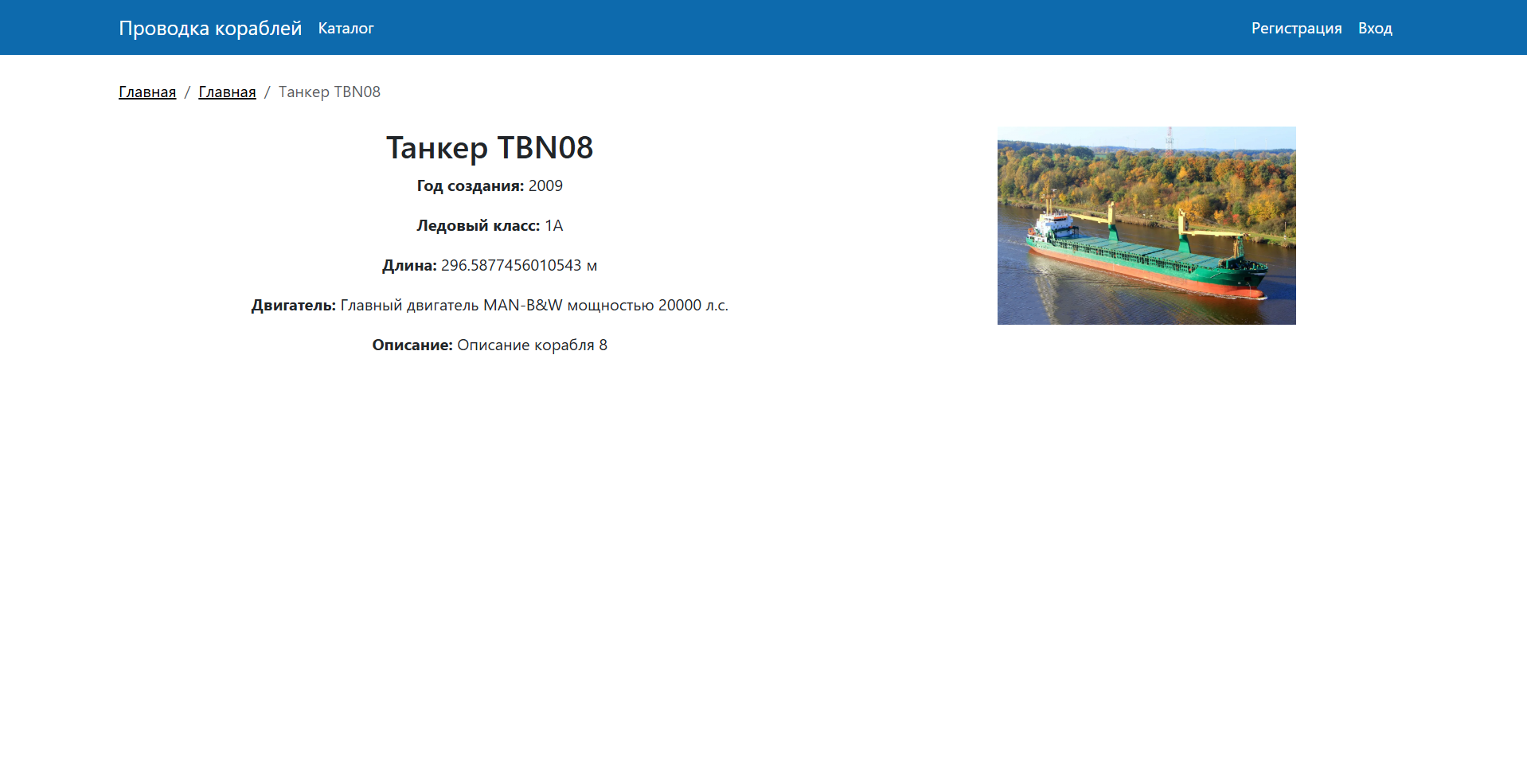


Рисунок 16 – Страница с описанием корабля

На странице одной заявки (рисунок 17) отображается текущая заявка пользователя. У пользователя есть возможность удалить корабля из заявки, поменять порядок кораблей, а также оформить заявку на проверку или удалить ее. Редактирование оформленных заявок недоступно

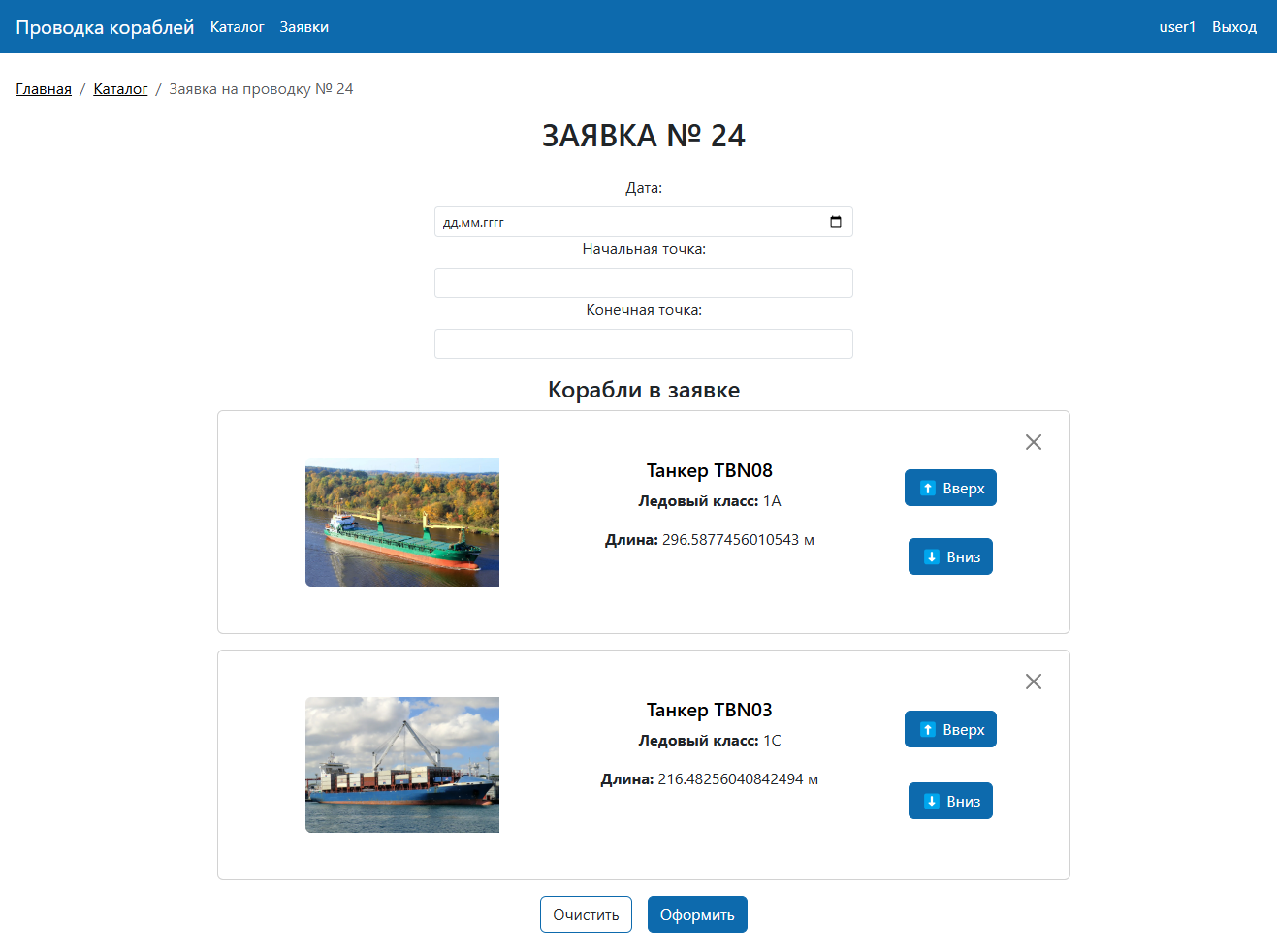


Рисунок 17 – Страница заявки на проводку кораблей

На странице списка заявок (рисунок 18) судовладельцы могут просматривать созданные ими заявки в виде таблицы. На этой странице можно посмотреть подробную информацию о заявке, нажав на её номер.

Пользователь может изменить свои данные (логин и пароль) на странице редактирования данных пользователя (рисунок 19). Она открывается при нажатии на логин пользователя в меню.

Оператор ледоколов может просматривать информацию о всех кораблях, занесённом в систему на странице списка кораблей (рисунок 20). Оператор имеет возможность добавить новый корабль, отредактировать или удалить существующий.

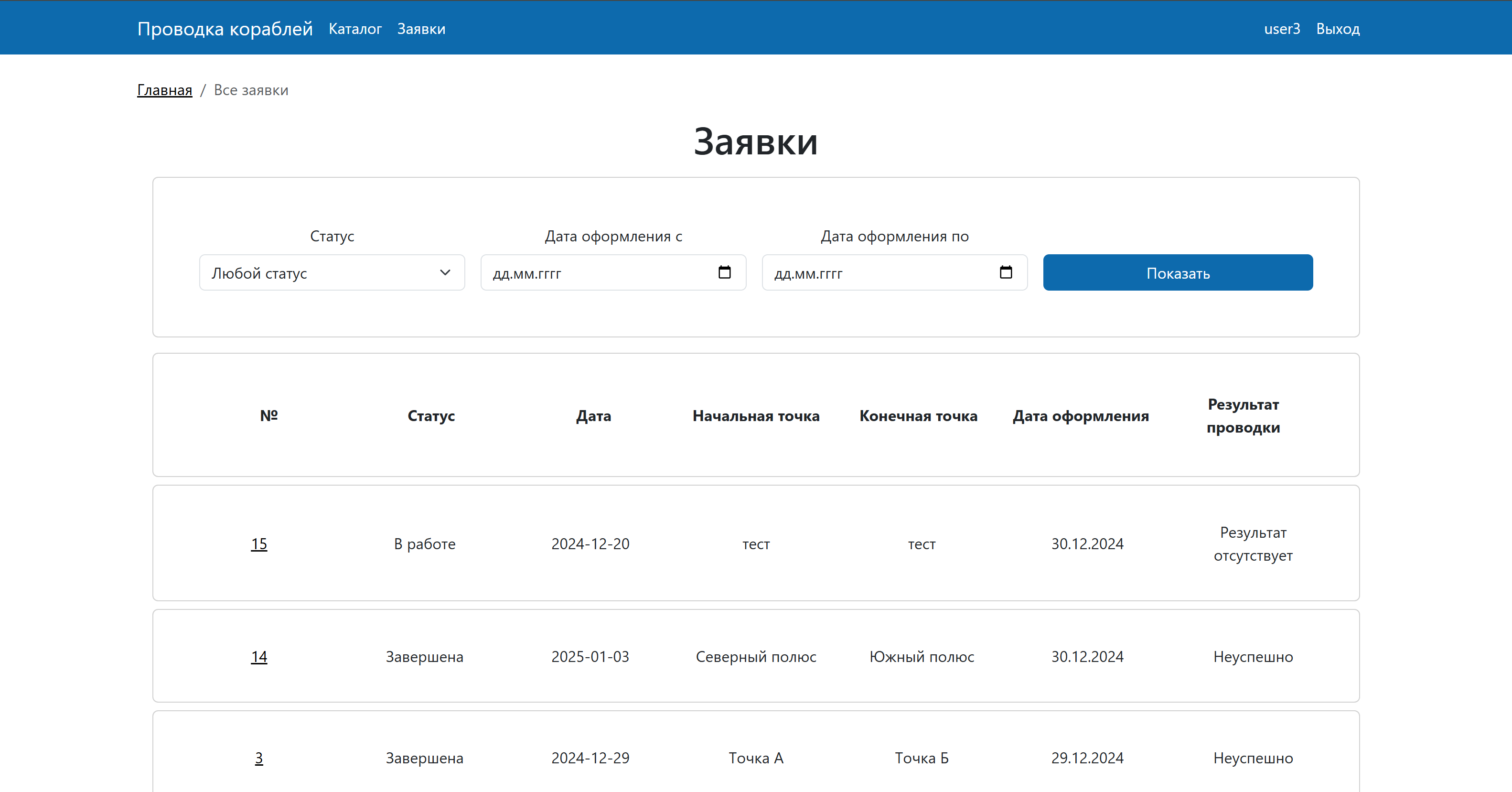


Рисунок 18 – Страница списка заявок на проводку кораблей

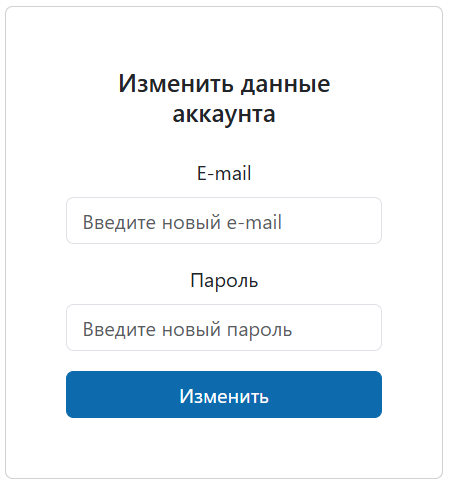
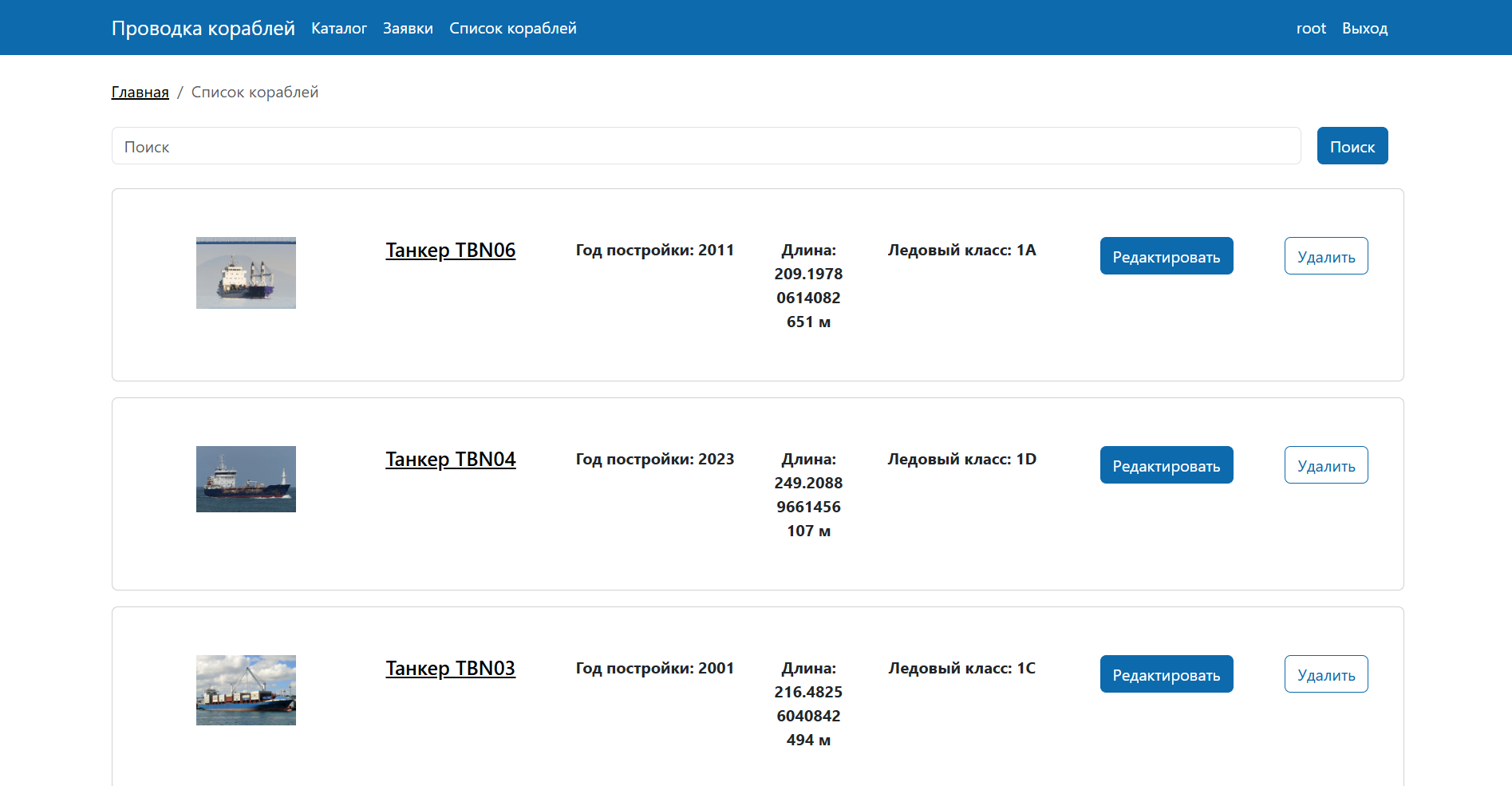


Рисунок 19 – Страница редактирования информации о пользователе

На странице списка всех заявок пользователей (рисунок 21), оператор может согласовывать или отклонять заявки пользователей путём нажатия соответствующих кнопок в интерфейсе. Менеджер может просматривать подробную информацию о заявке путём нажатия на её номер (рисунок 22). Доступны поля фильтрации по логину пользователя, статусу и датам формирования заявки.

 Рисунок 20 – Страница списка кораблей

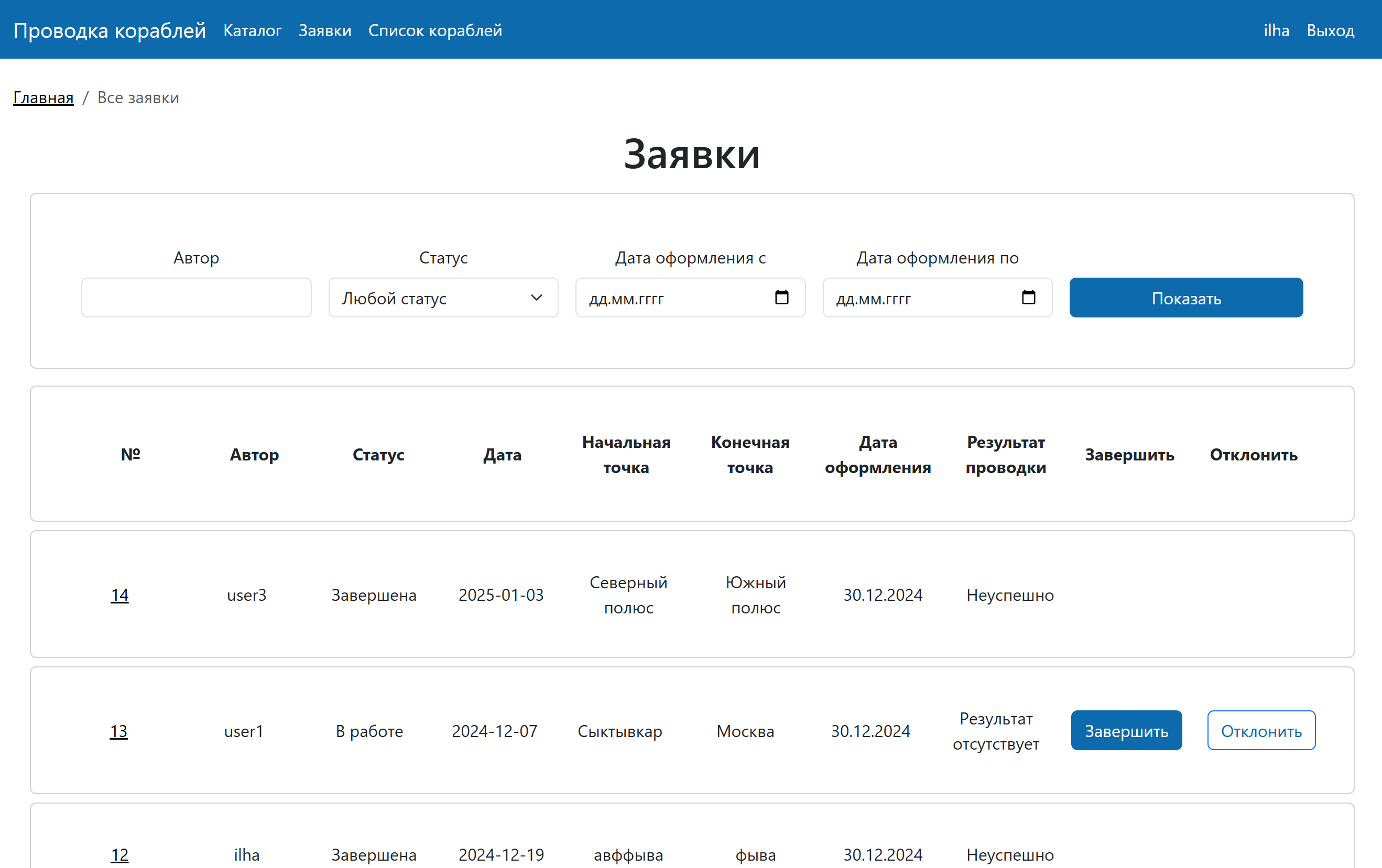


Рисунок 21 – Страница списка заявок на проводку кораблей

С помощью формы редактирования и создания корабля (рисунок 23) сотрудник может добавлять новые программы или изменять существующие.

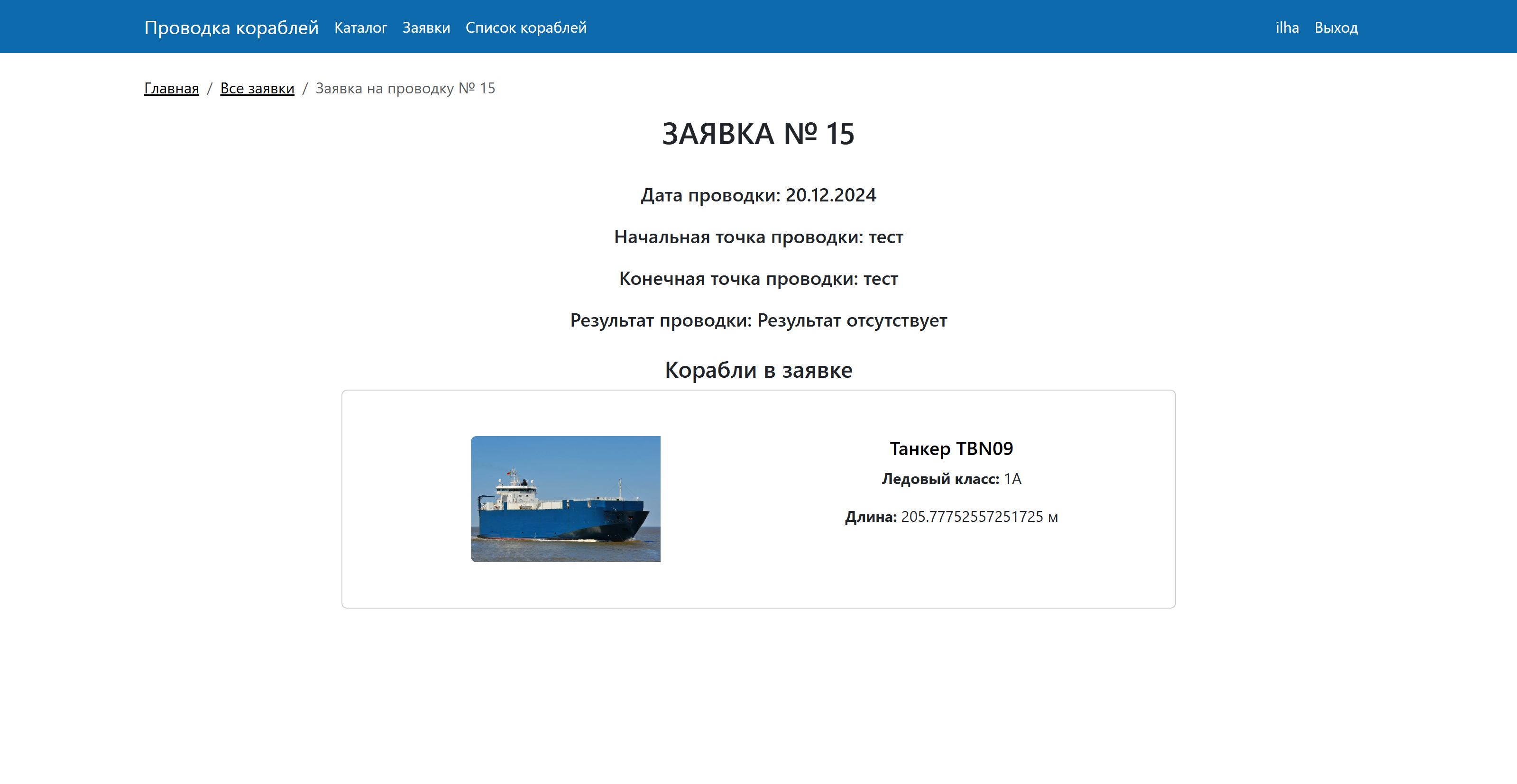


Рисунок 22 – Страница подробной информации о заявке

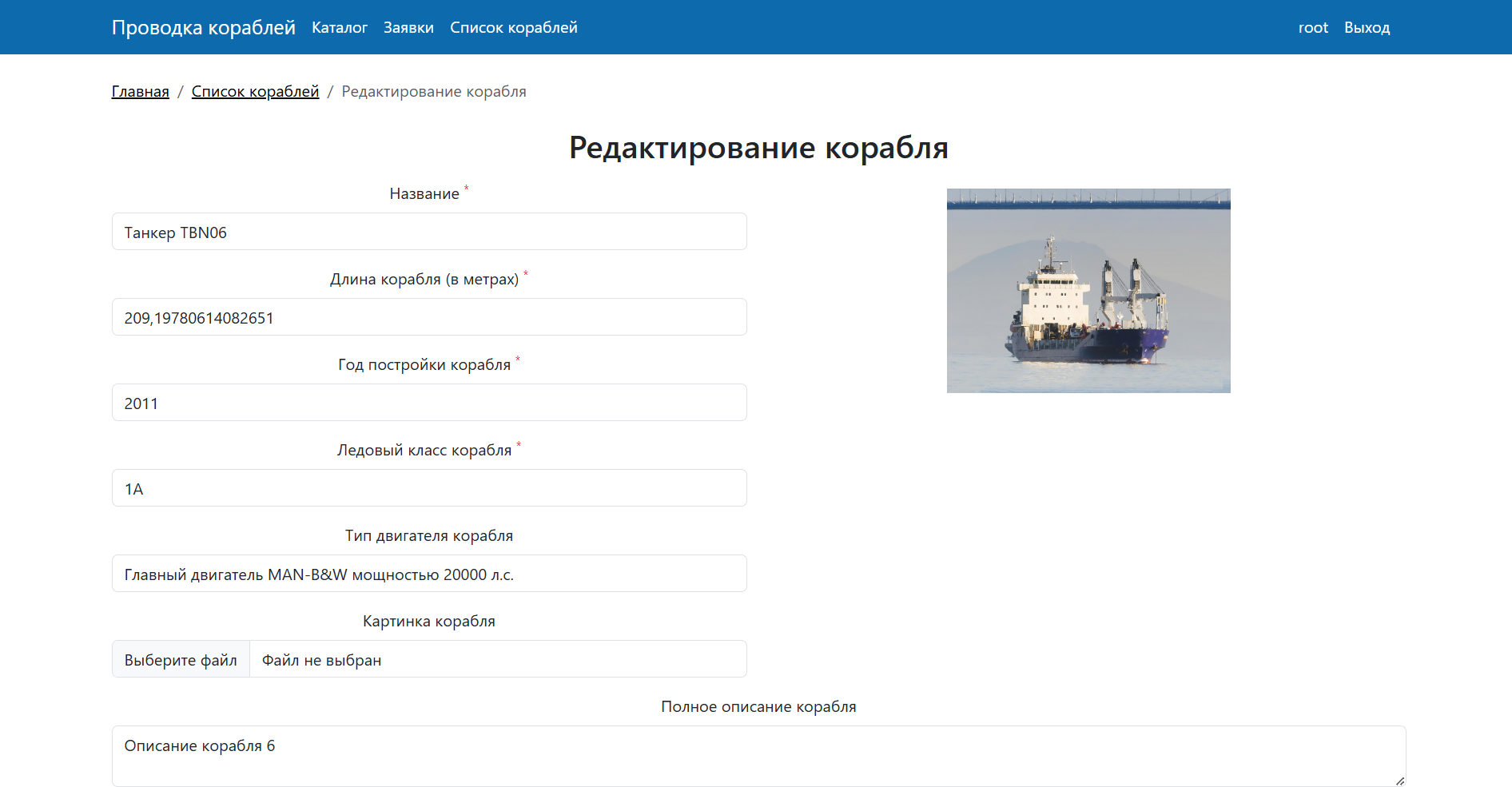


Рисунок 23 – Страница редактирования и создания корабля

В случае, если пользователь запрашивает несуществующую станицу, он перенаправляется на страницу ошибки 404 (рисунок 24).

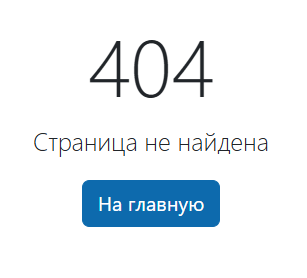


Рисунок 24 – Страница 404

В случае, если пользователь пытается получить доступ к страницам, для просмотра которых ему не хватает прав, он перенаправляется на страницу ошибки 403 (рисунок 25).

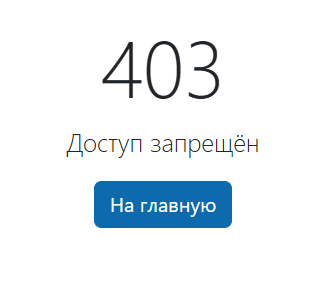


Рисунок 25 – Страница 403

ЗАКЛЮЧЕНИЕ

В ходе работы были достигнуты следующие результаты:

* 1. Создан MVP и базовый дизайн приложения на основе shipsforsale.su;
  2. Создана базу данных для хранения информации о корабля и заявках на проводку;
  3. Создан веб-сервис на Django (DRF);
  4. Реализована авторизация и хранение сессий в Redis;
  5. Разработан базовый SPA на React для гостя;
  6. Внедрена адаптивность, менеджер состояний Redux Toolkit, PWА, разработано Tauri приложение;
  7. Завершена разработка интерфейса пользователя в React;
  8. Реализован интерфейс оператора ледоколов;
  9. Разработано десктопное приложение Tauri
  10. Приложение развернуто при помощи сервиса GitHub Pages и доступно по ссылке: https://ilhinchik.github.io/Ice\_ships\_frontend/
  11. Подготовлен набор документации, включающий РПЗ, ТЗ и набор диаграмм.
  12. Оформлен git-репозиторий на сервисе GitHub, содержащий исходный код проекта: https://github.com/Ilhinchik/Ice\_ships\_frontend

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. Russia Lacks Icebreakers Its Arctic Fleet Needs To Function In Eastern Arctic – Analysis [Электронный ресурс] // EurasiaReview. URL: https://www.eurasiareview.com/27112024-russia-lacks-icebreakers-its-arctic-fleet-needs-to-function-in-eastern-arctic-analysis/ (дата обращения: 24.12.2024);
2. Autonomous Navigation in Ice-Covered Waters with Learned Predictions on Ship-Ice Interactions [Электронный ресурс] // Arxiv. URL: https://arxiv.org/pdf/2409.11326 (дата обращения: 24.12.2024);
3. Документация по Django [Электронный ресурс] // djangoproject. URL: https://www.djangoproject.com/ (дата обращения: 20.10.2024);
4. Документация по DRF [Электронный ресурс] // Django-rest-framework. URL: https://www.django-rest-framework.org/topics/documenting-your-api/ (дата обращения: 20.10.2024);
5. Документация по Redis [Электронный ресурс] // Netlify. URL: https://master--redis-doc.netlify.app/docs/ (дата обращения: 10.10.2024);
6. Документация Minio [Электронный ресурс] // Min. URL: https://min.io/docs/minio/kubernetes/upstream/index.html (дата обращения: 09.09.2024);
7. Документация PostgreSQL [Электронный ресурс] // Postgresql. URL: https://www.postgresql.org/docs/ (дата обращения: 15.09.2024);
8. Документация по Python [Электронный ресурс] // Python. URL: https://docs.python.org/3/index.html/ (дата обращения: 20.10.2024);
9. Документация по Tauri [Электронный ресурс] // Tauri. URL: https://v2.tauri.app/develop/ (дата обращения: 20.11.2024);
10. Документация по React [Электронный ресурс] // React. URL: https://react.dev/learn (дата обращения: 01.11.2024);

ПРИЛОЖЕНИЕ А ТЕХНИЧЕСКОЕ ЗАДАНИЕ

МОСКОВСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ

им. Н.Э. Баумана

Кафедра «Системы обработки информации и управления»

|  |  |  |
| --- | --- | --- |
| Утверждаю  Заведующий кафедрой ИУ-5 |  | Согласовано  Научный руководитель |
| \_\_\_\_\_\_\_\_\_\_\_В.И.Терехов  "\_\_"\_\_\_\_\_\_\_\_\_\_2024 г. |  | \_\_\_\_\_\_\_\_\_\_А.И. Канев  "\_\_"\_\_\_\_\_\_\_\_\_\_2024 г. |

**Севморпуть, заявки на проводку ледоколами**

Техническое задание

(вид документа)

писчая бумага

(вид носителя)

7

(количество листов)

|  |  |
| --- | --- |
| ИСПОЛНИТЕЛЬ: |  |
| \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ | Марков Илья Сергеевич |
| "\_\_"\_\_\_\_\_\_\_\_\_\_\_2024 г. |  |

Москва - 2024

1. Введение

Реализовать систему для автоматизации процесса подачи и обработки заявок на проводку судов ледоколами, которая будет состоять из веб-приложения, веб-сервиса и десктопного приложения.

1. Назначение разработки

Основное назначение разработанной системы заключается в автоматизации и оптимизации процесса подачи и обработки заявок на проводку судов ледоколами. Система позволит судовладельцам подавать заявки на проводку, отслеживать их статус, а также просматривать всю необходимую информацию о кораблях. Администраторы и операторы ледоколов смогут оперативно обрабатывать заявки, принимать или отклонять их, планировать маршруты, что сделает процесс проводки судов более прозрачным и эффективным.

1. Стадии и этапы разработки
   1. Создание MVP и базового дизайна в Figma на основе shipsforsale.su
   2. Спроектировать базу данных PostgreSQL
   3. Создать веб-сервис на Django Rest Framework
   4. Реализовать авторизацию и хранение сессий в Redis
   5. Разработать SPA на React для гостя
   6. Внедрить адаптивность, а также возможность работы в режиме PWA
   7. Развернуть веб-приложение в GitHub Pages
   8. Разработать Tauri приложение
   9. Реализовать интерфейс пользователя на React с оператором ледоколов состояний Redux Toolkit
   10. Добавить в React приложение интерфейс модератора
   11. Подготовка всей документации (РПЗ, ТЗ и набор диаграмм)
   12. Подготовка репозитория на GitHub
2. Требования к функциональным характеристикам
   1. Методы HTTP
      1. GET Получение всех кораблей
      2. POST Создание корабля
      3. GET Получение одного корабля
      4. PUT Изменение корабля
      5. DELETE Удаление корабля
      6. POST Добавление корабля в заявку на проводку
      7. POST Изменение/добавление картинки корабля
      8. GET Получение всех заявок на проводку
      9. GET Получение одного заявки на проводку
      10. PUT Изменение полей заявки на проводку
      11. DELETE Удаление заявки на проводку
      12. PUT Сохранение заявки на проводку
      13. PUT Модерирование заявки на проводку
      14. DELETE Удаление корабля из заявки на проводку
      15. PUT Изменение значения м-м корабля в заявке на проводку
      16. POST Регистрация
      17. PUT Личный кабинет
      18. POST Аутентификация
      19. POST Деавторизация
   2. Меню
      1. Главная - перенаправляет на страницу 4.5
      2. Список кораблей – перенаправляет на страницу 4.7
      3. Список заявок на проводку – перенаправляет на страницу 4.10, доступно только авторизованным пользователям
      4. Редактирование корабля – перенаправляет на страницу 4.12, доступно только модератору
      5. Зарегистрироваться – перенаправляет на страницу 4.3 доступно только для гостей
      6. Личный кабинет – перенаправляет на страницу 4.5, доступно только авторизованным пользователям
      7. Войти – перенаправляет на страницу 4.4, доступно только для гостей
      8. Выйти – перенаправляет на страницу 4.6, доступно только авторизованным пользователям, выход из аккаунта (метод 4.1.19)
   3. Регистрация
      1. Доступно только гостям
      2. Отображает форму регистрации
         1. Поле логина
         2. Поле почты
         3. Поле пароля
      3. Действия
         1. Регистрация пользователя – (вызывается метод 4.1.16)
         2. Вернуться к аутентификации – перенаправляет на страницу 4.4
   4. Аутентификация
      1. Доступно только гостям
      2. Отображает форму аутентификации
         1. Поле логина
         2. Поле пароля
      3. Действия
         1. Войти – (вызывается метод 4.1.18)
         2. Регистрация – перенаправляет на страницу 4.3
   5. Личный кабинет
      1. Доступно авторизованному пользователю
      2. Действия
         1. Изменить данные пользователя – (вызывается метод 4.1.17)
   6. Главная
      1. Доступна всем
      2. Отображается статическое курс сервиса
   7. Список кораблей
      1. Доступна всем
      2. Отображаются карточки кораблей (метод 4.1.1)
         1. Название
         2. Картинка
         3. Ледовый класс
         4. Год
      3. Действия
         1. Поиск – перенаправляет на страницу 4.6, (используется метод 4.1.1), с фильтрующем параметром
         2. Подробнее – перенаправляет на страницу 4.7 (используется метод 4.1.3)
         3. Добавить в заявку на проводку – добавляет корабль в заявку-черновик, (вызывается метод 4.1.2), только аутентифицированные пользователи.
         4. Кнопка корзины – перенаправляет на страницу 4.9, только аутентифицированные пользователи.
   8. Один корабль
      1. Доступна всем
      2. Отображается подробная информация выбранного корабля, (вызывается метод 4.1.3)
   9. Одна заявка
      1. Доступно только аутентифицированным пользователям
      2. Отображает текущую заявку-черновик, (метод 4.1.9)
         1. Список добавленных кораблей
      3. Действия, доступны только в случае, если статус «черновик»
         1. Убрать корабль – удалят корабль из заявки, (вызывается метод 4.1.14)
         2. Сохранить – сохраняет текущую заявку-черновик, (вызывается метод 4.1.12)
         3. Очистить – удаляет заявку-черновик, (вызывается метод 4.1.1)
         4. Сформировать – вносит данные заявки (формирует заявку, вызывается метод 4.1.10)
   10. Список заявок
       1. Доступно авторизированному пользователю
       2. Отображается список заявок (метод 4.1.8)
          1. Для пользователя - только заявки пользователя
          2. Для модератора - все заявки
       3. Действия
          1. Фильтрация – фильтрует заявку по дате создания и статусу (вызывается метод 4.1.8)
          2. Сформировать – формирует заявку, выполняется метод 4.1.13, доступно только пользователю
          3. Отклонить – отклоняет заявку, вызывается метод 4.1.13, доступно только пользователю
          4. Посмотреть подробную информацию о корабле – перенаправляет на страницу 4.9 (вызывается метод 4.1.9)
   11. Список кораблей таблицей
       1. Доступно только модератору
       2. Отображаются все корабли (вызывается метод 4.1.1)
       3. Действия
          1. Удалить – удаляет корабль (вызывается метод 4.1.5)
          2. Редактирование/создание – переход на страницу 4.12
   12. Редактирование/создание корабля
       1. Доступно только модератору
       2. Отображается информация о корабле (метод 4.1.3)
          1. Название
          2. Ледовый класс
          3. Год создания
          4. Картинка
       3. Действия
          1. Сохранить – обновляет существующий корабль (вызывается метод 4.1.4)
          2. Добавить – добавляет новый корабль (метод 4.1.2)
          3. Картинка – добавляет/изменяет картинку корабля (вызывается метод 4.1.7)
   13. 404
       1. Доступно всем
       2. Отображается в случае отсутствия ресурса
   14. 403
       1. Доступно всем
       2. Отображается в случае запрета на использование ресурса
3. Требования к составу и параметрам технических средств
   1. Сервер
      1. Процессор минимум 2-ядерный с частотой от 2 ГГц
      2. Оперативная память от 4 Гб
      3. Место на жестком диске от 2 Гб
   2. Клиент
      1. Процессор с частотой от 2ГГц
      2. Оперативная память от 4 Гб
      3. Свободное пространство на диске 10 Гб
4. Требования к информационной и программной совместимости
   1. Сервер
      1. ОС Linux (6.4.12)
      2. Redis (7.2)
      3. Minio (RELEASE 2022-10-15T19-57-03Z)
      4. PostgreSQL (16)
      5. Docker
      6. Node JS
      7. Python3.12
      8. Django 5.1
   2. Клиент
      1. Chrome (119.0.6045 и выше)

ПРИЛОЖЕНИЕ Б СПИСОК HTTP МЕТОДОВ

Таблица 1 – HTTP методы разрабатываемого веб-сервиса

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **№** | **Тип** | **URL** | **Курс** | **Входные данные** | **Выходные данные** |
| 4.1.1 | GET | /api/ships/ | Возвращает список кораблей, а также id черновой заявки  Доступно всем пользователям | {  “ship\_name”: char(100)  } | {  “draft\_icebreaker\_id”: int,  “ships\_count”: int,  “ships”: [  {  “id”: int,  “ship\_name”: char(100),  “year”: int,  “ice\_class”: char(10),  “length”: float,  “engine”: char(100),  “status”: int,  “image”: char(100),  “description”: text  },  …  ]  } |
| 4.1.2 | POST | api/ships/create/ | Добавляет корабль в черновую заявку  Доступно только авторизованным пользователям | {  “ship\_name”: char(100),  “ice\_class”: char(10),  “year”: int,  } | 201 / 403 / 404 |
| 4.1.3 | GET | api/ships/<ship\_id>/ | Получает информацию о корабле  Доступно всем пользователям | {  “ship\_id”: int  } | {  “id”: int,  “ship\_name”: char(100),  “year”: int,  “ice\_class”: char(10),  “length”: float,  “engine”: char(255),  “status”: int,  “image”: char(100),  “description”: text  } |
| 4.1.4 | PUT | api/ships/<ship\_id>/update/ | Изменяет данные корабля (кроме картинки)  Доступно только авторизованным пользователям | {  “ship\_id”: int  } | {  “id”: int,  “ship\_name”: char(100),  “year”: int,  “ice\_class”: char(10),  “length”: float,  “engine”: char(255),  “status”: int,  “image”: char(100),  “description”: text  } |
| 4.1.5 | DELETE | api/ships/<ship\_id> /delete/ | Удаляет корабль  Доступно только авторизованным пользователям | {  “ship\_id”: int  } | [  {  “id”: int,  “ship\_name”: char(100),  “year”: int,  “ice\_class”: char(10),  “length”: float,  “engine”: char(255),  “status”: int,  “image”: char(100),  “description”: text  },  …  ] |
| 4.1.6 | POST | api/ships/<ship\_id> /add\_to\_icebreaker/ | Добавляет корабль в черновую заявку  Доступно только авторизованным пользователям | {  “ship\_id”: int  } | 201 / 403 / 404 |
| 4.1.7 | POST | api/ships/<ship\_id> /update\_image / | Добавляет/изменяет картинку корабля  Доступно только авторизованным пользователям | {  “ship\_id”: int,  “image”: bytes  } | {  “id”: int,  “ship\_name”: char(100),  “year”: int,  “ice\_class”: char(10),  “length”: float,  “engine”: char(100),  “status”: int,  “image”: char(100),  “description”: text  } |
| 4.1.8 | GET | api/icebreakers/ | Возвращает список заявок. Доступно только авторизованным пользователям | {  “status”: int,  “date\_start”: char(100),  “date\_end”: char(100)  } | [  {  "id": int,  "status": int,  "date\_created": datetime,  “date\_formation": datetime,  "date\_complete": datetime,  "owner": char(100),  “moderator”: char(100),  “date”: date,  “start\_point”: char(100),  “finish\_point”: chat(100)  },  …  ] |
| 4.1.9 | GET | api/icebreakers/<icebreaker\_id>/ | Возвращает информацию о заявках  Доступно только авторизованным пользователям | {  “icebreaker\_id”: int  } | {  "id": int,  "status": int,  "date\_created": datetime,  “date\_formation": datetime,  "date\_complete": datetime,  "owner": char(100),  “moderator”: char(100),  “date”: date,  “start\_point”: char(100),  “finish\_point”: chat(100),  “ships”: [  {  “id”: int,  “ship\_name”: char(100),  “year”: int,  “ice\_class”: char(10),  “length”: float,  “engine”: char(255),  “status”: int,  “image”: char(100),  “description”: text  },  …  ]  } |
| 4.1.10 | PUT | api/icebreakers/<icebreaker\_id>/update/ | Изменяет информацию о заявке  Доступно только авторизованным пользователям | {  “icebreaker\_id”: int  } | {  "id": int,  "status": int,  "date\_created": datetime,  “date\_formation": datetime,  "date\_complete": datetime,  "owner": char(100),  “moderator”: char(100),  “date”: date,  “start\_point”: char(100),  “finish\_point”: chat(100),  “ships”: […]  } |
| 4.1.11 | DELETE | api/icebreakers/<icebreaker\_id>/delete/ | Удаляет заявку  Доступно только авторизованным пользователям | {  “icebreaker\_id”: int  } | 200 / 404 / 403 |
| 4.1.12 | PUT | api/icebreakers/<icebreaker\_id>/update\_status\_user/ | Формирует заявку  Доступно только авторизованным пользователям | {  “icebreaker\_id”: int  } | {  "id": int,  "status": int,  "date\_created": datetime,  “date\_formation": datetime,  "date\_complete": datetime,  "owner": char(100),  “moderator”: char(100),  “date”: date,  “start\_point”: char(100),  “finish\_point”: chat(100),  "ships": […]  } |
| 4.1.13 | PUT | api/icebreakers/<icebreaker\_id>/update\_status\_admin/ | Модерирует заявку  Доступно только модераторам | {  “icebreaker\_id”: int  } | {  "id": int,  "status": int,  "date\_created": datetime,  “date\_formation": datetime,  "date\_complete": datetime,  "owner": char(100),  “moderator”: char(100),  “date”: date,  “start\_point”: char(100),  “finish\_point”: chat(100),  "ships": […]  } |
| 4.1.14 | DELETE | api/icebreakers/<icebreaker\_id>/delete\_ship/<ship\_id>/ | Удаляет корабль из заявки  Доступно только авторизованным пользователям | {  “icebreaker\_id”: int,  “ship\_id”: int  } | 200 / 404 / 403 |
| 4.1.15 | PUT | api/icebreakers/<int:icebreaker\_id>/update\_ship/<int:ship\_id>/ | Обновляет значение м-м корабля в заявке  Доступно только авторизованным пользователям | {  “icebreaker\_id”: int,  “ship\_id”: int,  “order”: int  } | {  “id”: int,  “ship\_name”: char(100),  “year”: int,  “ice\_class”: char(10),  “length”: float,  “engine”: char(100),  “status”: int,  “image”: char(100),  “description”: text  “order”: int  } |
| 4.1.16 | POST | api/users/register/ | Регистрирует пользователя  Доступно только гостям | {  “username”: char(100),  “email”: char(100),  “password”: char(100)  } | {  “user\_id”: int,  “username”: char(100),  “username”: char(100),  “password”: char(100)  } |
| 4.1.17 | PUT | api/users/update/<int:user\_id> | Обновляет данные пользователя.  Доступно только авторизованным пользователям | {  “user\_id”: int,  “email”: char(100),  “username”: char(100),  “password”: char(100),  } | {  “user\_id”: int,  “username”: char(100),  “username”: char(100),  “password”: char(100)  } |
| 4.1.18 | POST | /api/users/login/ | Аутентификация  Доступно всем | {  “user\_id”: int,  “username”: char(100),  “password”: char(100)  } | {  “user\_id”: int,  “username”: char(100),  “username”: char(100),  “password”: char(100)  } |
| 4.1.19 | POST | api/users/logout/ | Деавторизация  Доступно только авторизованным пользователям |  | 200 / 403 |