Министерство транспорта Российской Федерации
Федеральное агентство железнодорожного транспорта
Федеральное государственное бюджетное
Образовательное учреждение высшего образования
«Дальневосточный государственный университет путей сообщения»

Кафедра «Высшая математика»

Курсовая работа по дисциплине:

«Базы данных»

Вариант № 3

Тема: «Разработка программного обеспечения для работы с базой данных в PostgreSQL на примере судоходной компании «Балтика»»

Выполнила:

Студент группы БО921ПМИ (2 курс, 4 семестр)

Направления 01.03.02

«Прикладная математика и информатика»

Белькович Анжелика Игоревна

Проверил:

Доцент кафедры «Высшая математика», ДВГУПС
Коломийцева Светлана Валерьевна

РЕЙТИНГОВЫЙ БАЛЛ ЗА КУРСОВУЮ РАБОТУ _____
ОЦЕНКА ЗА КУРСОВУЮ РАБОТУ_____

Хабаровск

2024

СОДЕРЖАНИЕ

ВВЕДЕНИЕ

В современном мире, характеризующемся стремительным развитием технологий, потребность в обработке и хранении огромных объемов информации постоянно растет. Базы данных предоставляют обширные возможности для решения этой задачи. Их преимущество заключается в эффективном управлении данными и обеспечении быстрого доступа к ним. Таким образом, использование баз данных способствует оптимизации рабочих процессов и решает множество проблем, с которыми сталкиваются организации.

В данной курсовой работе рассматривается разработка приложения для работы с базой данных в PostgreSQL на примере разработки прикладного программного обеспечения деятельности судоходной компании «Балтика». Эта крупная компания занимается перевозками грузов между континентами и её деятельность требует эффективной системы управления данными.

Целью данной работы является разработка приложения, которое обеспечит эффективное хранение и управление данными, а также автоматизирует процессы работы с базой данных компании. В рамках исследования будут изучены основные принципы работы с PostgreSQL и методы разработки приложений на его основе. Мы также изучим передовые методы создания баз данных и разработки приложений, чтобы создать надежное и эффективное решение, способное удовлетворить актуальные потребности в управлении и хранении данных.

Рассматривая данную цель, следует поставить следующие задачи:

1) Провести анализ деятельности судоходной компании «Балтика»;
2) Изучить требования судоходной компании «Балтика» к базе данных и приложению;
3) Разработать концептуальное проектирования базы данных;
4) Спроектировать базы данных с использованием PostgreSQL;
5) Создать прикладное программное обеспечение.

Новизна данной работы заключается в использовании передовых методов создания баз данных и разработки приложений на основе PostgreSQL. Реализация этих методов даст существенное преимущество компании за счет улучшения процессов составления маршрутов для судов, а также предоставления подробной информации о партиях грузов, их отправителях и получателях. Управление этой информацией будет способствовать повышению эффективности работы компании, что подчеркивает значимость данного проекта.

Объектом исследования является разработка базы данных PostgreSQL и соответствующего приложения для судоходной компании «Балтика», а предметом исследования – проектирование автоматизированной системы управления её деятельностью.

Для реализации поставленной задачи использовались следующие **инструменты**:

1) **Draw.io:** Для создания визуальных схем и диаграмм. Draw.io предоставляет возможность легко разрабатывать блок-схемы, диаграммы процессов и архитектуры системы, что способствует лучшему пониманию и планированию проектов.
2) **StarUML:** Для проектирования базы данных. StarUML позволяет создавать диаграммы классов, диаграммы сущность-связь и другие виды диаграмм, которые необходимы для проектирования структуры базы данных.
3) **PostgreSQL:** В качестве реляционной базы данных для хранения информации о судах, маршрутах, партиях грузов и других сущностях компании «Балтика». PostgreSQL обеспечивает высокую надежность и масштабируемость, а также богатый функционал. Она поддерживает сложные запросы и транзакции, что делает её идеальной для работы с большими объемами данных.
4) **Python:** Для создания прикладного программного обеспечения, включая интерфейс пользователя и работу с базой данных. Python популярен благодаря своей простоте и читаемости кода, а также богатой библиотеке модулей. Это ускоряет разработку и упрощает интеграцию с другими системами.

5) **PyCharm:** В качестве интегрированной среды разработки (IDE) для Python. PyCharm предоставляет мощные инструменты для редактирования кода, отладки, тестирования и развертывания приложений. Он поддерживает различные фреймворки и библиотеки, что делает его идеальным выбором для разработки сложных программных проектов на Python, обеспечивая продуктивность и качество кода.

1. ТЕОРЕТИЧЕСКАЯ ЧАСТЬ

1.1 Постановка задачи

Необходимо разработать прикладное программное обеспечение деятельности судоходной компании «Балтика». Эта крупная компания занимается перевозками грузов между континентами.

В ее собственности несколько десятков судов различного класса и грузоподъемности. К услугам этой компании обращаются тысячи клиентов из различных стран мира. На судне может находиться несколько партий грузов для различных грузополучателей из различных стран и городов. Одна партия груза может состоять из нескольких разновидностей грузов. У одной партии груза может быть только один отправитель и только один получатель. Судно следует по маршруту. Маршрут разрабатывается главным менеджером компании и проходит через несколько портов.

В очередном порту назначения производится лишь частичная погрузка и выгрузка грузов, и судно следует дальше.

В базе данных должна храниться информация о:

- Судах;
- Отправителях;
- Получателях;
- Партиях грузов;
- Грузах;
- Маршрутах.

Набор данных для прикладного программного обеспечения деятельности судоходной компании «Балтика» представлен в следующей таблице:

Набор данных для прикладного программного обеспечения деятельности судоходной компании «Балтика» представлен в таблице 1.1.

Таблица 1.1 – Данные для ПО

№	Поле	Тип	Размер	Описание
1	RegNumber	**Числовой**	10	Регистрационный номер судна
2	Name	**Текстовый**	60	Название судна
3	Skipper	**Текстовый**	60	ФИО капитана судна
4	Type	**Текстовый**	15	Тип судна (танкер, сухогруз)
5	Capacity	**Числовой**	10	Грузоподъемность судна
6	Year	**Числовой**	4	Год постройки судна
7	Picture	**Поле объекта OLE**	Авто	Фотография судна
8	Dockyard	**Текстовый**	15	Порт приписки
9	CustomValue	**Числовой**	10	Таможенный номер партии груза
10	DepartureDate	**Дата/время**	Авто	Дата убытия груза
11	ArriveDate	**Дата/время**	Авто	Дата прибытия груза
12	Origin	**Текстовый**	20	Пункт отправления
13	Destination	**Текстовый**	20	Пункт назначения
14	CustomClearance	**Логический**	1	Необходимость таможенной декларации
15	Number	**Числовой**	4	Номер груза в партии
16	Shipment	**Текстовый**	30	Название груза
17	DeclareValue	**Числовой**	8	Заявленная величина груза
18	Unit	**Текстовый**	10	Единица измерения груза
19	InsureValue	**Числовой**	8	Застрахованная величина груза
20	Sender	**Текстовый**	30	Отправитель груза
21	INNsender	**Числовой**	10	ИНН отправителя груза

22	BankSender	**Текстовый**	60	Банк отправителя груза
23	AddressSender	**Текстовый**	80	Юридический адрес отправителя груза
24	Consignee	**Текстовый**	30	Получатель груза
25	INNconsignee	**Числовой**	10	ИНН получателя груза
26	BankConsignee	**Текстовый**	60	Банк получателя груза
27	AddressConsign	**Текстовый**	80	Юридический адрес получателя груза
28	Comment	**Поле Memo**	Авто	Примечания

1.2 Анализ деятельности судоходной компании «Балтика»

Анализ схемы потоков данных

Диаграмма потоков данных (DFD) — это графическое представление потока данных через информационную систему. Она показывает, как данные вводятся, обрабатываются, хранятся и выводятся системой. DFD используются для анализа, проектирования и документирования информационных систем [1].

Метод DFD разбивает высокоуровневую диаграмму потока данных на набор более подробных диаграмм, обеспечивая общее представление о всей системе, а также более подробную декомпозицию. Дает общее представление о системе в целом, а также более подробную декомпозицию и, при необходимости, более подробную разбивку и описание отдельных действий для облегчения разъяснения и понимания.

Отправитель груза обращается в компанию, предоставляет необходимые данные для формирования заявки.

Сначала в базу заносится основная информация об отправителе груза (такие данные, как название организации, ИНН, банк и юридический адрес), затем составляется партия грузов по предпочтениям клиента, в котором указаны:

номер партии, номер судна, номер отправителя, название организации, номер получателя и др. В базе так же отдельно хранятся данные о получателях.

Далее составляется маршрут для каждой партии грузов и подбирается определенное судно, а данные по маршрутам и о судах так же храниться отдельно в базе.

При этом у одного клиента может существовать неограниченное количество заявок.

На основе имеющейся информации была построена DFD-диаграмма (Рис.1)

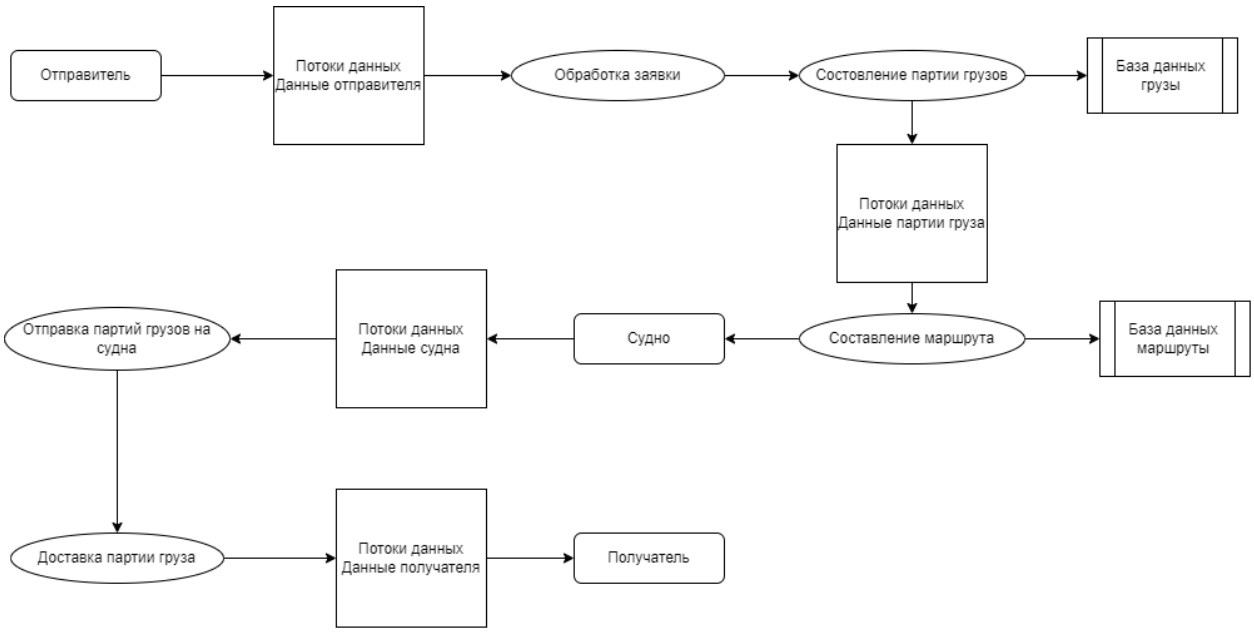

Рис. 1 - DFD- диаграмма

1.3 ER – диаграмма сущностей

ER-диаграмма сущностей — это разновидность блок-схемы, где показано, как разные «сущности» (люди, объекты, концепции и так далее) связаны между собой внутри системы. ER-диаграммы чаще всего применяются для проектирования и отладки реляционных баз данных в сфере образования, исследования и разработки программного обеспечения и информационных систем для бизнеса [2].

В данной работе ER-диаграмма является полезным инструментом для наглядного представления сущностей и их отношений в базе данных, для дальнейшего понимания работы с этой базой данных.

В данной базе данных было выделено 6 основных сущностей:

1. Суда;
2. Маршруты;
3. Отправители;
4. Получатели;
5. Партии грузов;
6. Грузы.

- **Суда** (sydno) обладает следующими атрибутами:

reg_number_id – id судна, является первичным ключом для сущности Sydno,

name – название судна,

skipper – ФИО капитана,

type – тип судна (танкер, сухогруз),

capacity - грузоподъемность судна,

year – год постройки судна,

picture – фотография судна,

dockyard – порт приписки;

- **Маршруты** (marshryt) обладает следующими атрибутами:

route_id – id маршрута, является первичным ключом для сущности Marshryt,

reg_number_id – id судна, является вторичным ключом для сущности Sydno,

departure_date – дата убытия груза,

arrive_date – дата прибытия груза,

origin – пункт отправления,

destination – пункт назначения;

- **Отправители** (otpravitel) обладает следующими атрибутами:

sender_id - id отправителя, является первичным ключом для сущности Otpravitel,

sender_name – название организации отправителя,

inn_sender – ИНН отправителя груза,

bank_sender – банк отправителя груза,

address_sender – юридический адрес отправителя груза;

- **Получатели** (polychatel) обладает следующими атрибутами:

recipient_id - id получателя, является первичным ключом для сущности Polychatel,

consignee_name – название организации получателя,

inn_consignee – ИНН получателя груза,

bank_consignee – банк получателя груза,

address_consign – юридический адрес получателя груза;

- **Партии грузов** (partia_gryza) обладает следующими атрибутами:

custom_value_id – id партии грузов, является первичным ключом для сущности Partia_gryza,

reg_number_id – id судна, является вторичным ключом для сущности Sydno,

sender_id – id отправителя, является вторичным ключом для сущности Otpravitel,

sender_name – название организации отправителя,

recipient_id – id получателя, является вторичным ключом для сущности Polychatel,

consignee_name – название организации получателя,

custom_clearance – необходимость таможенной декларации;

- **Грузы** (gryz) обладает следующими атрибутами:

custom_value_id – id партии грузов, является вторичным ключом для сущности Partia_gryza,

number_id – id груза, является первичным ключом для сущности Gryz,

shipment – название груза,

declare_value – заявленная величина груза,

unit – единица измерения груза,

insure_value – застрахованная величина груза,

number_cargo – примечание.

На основе имеющейся информации была построена ER-диаграмма (Рис.2)

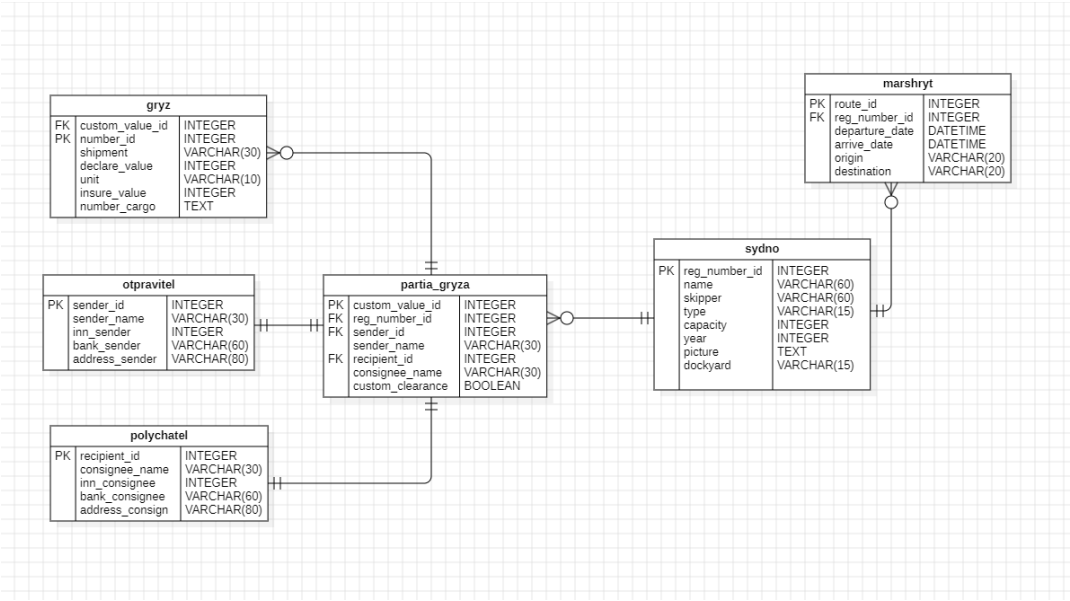

Рис. 2 – ER-диаграмма

Для каждой сущности ER-модели (Рис.2) нужно создать базовое отношение, каждому простому атрибуту сущности соответствует столбец таблицы. Ключевой атрибут сущности становится первичным ключом отношения. Если ключ составной, то для каждой его части создается отдельный столбец, а затем этот набор столбцов объявляется первичным ключом отношения.

Созданные отношения, представленные в виде таблиц, должны быть нормализованы.

Нормализация – это процесс организации данных в базе данных, включающий создание таблиц и установление отношений между ними в

соответствии с правилами, которые обеспечивают защиту данных и делают базу данных более гибкой, устраняя избыточность и несогласованные зависимости.

После соблюдения всех требований нормализации будут получены следующие отношения:

Таблица 1 – Отношение «Суда»

```
CREATE TABLE kostochka.sydno (
    reg_number_id integer NOT NULL,
    name varchar(100) NOT NULL,
    skipper varchar(60) NOT NULL,
    type varchar(100) NOT NULL,
    capacity integer NOT NULL,
    year integer NOT NULL,
    picture text NOT NULL,
    dockyard varchar(100) NOT NULL,
    PRIMARY KEY (reg_number_id)
);
```

Таблица 2 – Отношение «Маршруты»

```
CREATE TABLE kostochka.marshryt (
    route_id integer NOT NULL,
    reg_number_id integer NOT NULL,
    departure_date timestamp with time zone NOT NULL,
    arrive_date timestamp with time zone NOT NULL,
    origin varchar(100) NOT NULL,
```

destination varchar(100) NOT NULL,

PRIMARY KEY (route_id)

);

Для данного отношения также были созданы внешние ограничения ключей:

ALTER TABLE kostochka.marshryt ADD CONSTRAINT FK_marshryt__reg_number_id FOREIGN KEY (reg_number_id) REFERENCES kostochka.sydno(reg_number_id);

Этот запрос также добавляет внешний ключ к таблице marshryt, но на этот раз он ссылается на столбец reg_number_id в таблице sydno. Таким образом, он создает связь между таблицами marshryt и sydno через внешний ключ.

Таблица 3 – Отношение «Отправители»

CREATE TABLE kostochka.otpravitel (

sender_id integer NOT NULL,

sender_name varchar(100) NOT NULL,

inn_sender integer NOT NULL,

bank_sender varchar(100) NOT NULL,

address_sender varchar(100) NOT NULL,

PRIMARY KEY (sender_id)

);

Таблица 4 – Отношение «Получатели»

CREATE TABLE kostochka.polychatel (

recipient_id integer NOT NULL,

consignee_name varchar(100) NOT NULL,

inn_consignee integer NOT NULL,

```
    bank_consignee varchar(100) NOT NULL,
    address_consign varchar(100) NOT NULL,
    PRIMARY KEY (recipient_id)
);
```

Таблица 5 – Отношение «Партии грузов»

```
CREATE TABLE kostochka.partia_gryza (
    custom_value_id integer NOT NULL,
    reg_number_id integer NOT NULL,
    sender_id integer NOT NULL,
    sender_name varchar(100) NOT NULL,
    recipient_id integer NOT NULL,
    consignee_name varchar(100) NOT NULL,
    custom_clearance boolean NOT NULL,
  PRIMARY KEY (custom_value_id)
);
```

Для данного отношения также были созданы внешние ограничения ключей:

1) ALTER TABLE kostochka.partia_gryza ADD CONSTRAINT FK_partia_gryza__sender_id FOREIGN KEY (sender_id) REFERENCES kostochka.otpravitel(sender_id);

   Этот запрос добавляет внешний ключ к таблице partia_gryza, который ссылается на столбец custom_ sender_id в таблице otpravitel, обеспечивая таким образом целостность данных и связь между этими двумя таблицами.

2) ALTER TABLE kostochka.partia_gryza ADD CONSTRAINT FK_partia_gryza__recipient_id FOREIGN KEY (recipient_id) REFERENCES kostochka.polychatel(recipient_id);

Этот запрос также добавляет внешний ключ к таблице partia_gryza, который ссылается на столбец custom_ recipient_id в таблице polychatel, обеспечивая таким образом целостность данных и связь между этими двумя таблицами.

3) ALTER TABLE kostochka.partia_gryza ADD CONSTRAINT FK_partia_gryza__reg_number_id FOREIGN KEY (reg_number_id) REFERENCES kostochka.sydno(reg_number_id);

Еще один запрос также добавляет внешний ключ к таблице partia_gryza, который ссылается на столбец custom_ reg_number_id в таблице sydno, обеспечивая таким образом целостность данных и связь между этими двумя таблицами.

Таблица – 6 Отношение «Грузы»

```
CREATE TABLE kostochka.gryz (
    custom_value_id integer NOT NULL,
    number_id integer NOT NULL,
    shipment varchar(100) NOT NULL,
    declare_value integer NOT NULL,
    unit varchar(100) NOT NULL,
    insure_value integer NOT NULL,
    number_cargo text NOT NULL,
    PRIMARY KEY (number_id)
);
```

Для данного отношения также были созданы внешние ограничения ключей:

ALTER TABLE kostochka.gryz ADD CONSTRAINT FK_gryz__custom_value_id FOREIGN KEY (custom_value_id) REFERENCES kostochka.partia_gryza(custom_value_id);

Данный запрос добавляет внешний ключ к таблице gryz, который ссылается на столбец custom_value_id в таблице partia_gryza, обеспечивая таким образом целостность данных и связь между этими двумя таблицами.

Для ускорения поиска данных в базе данных созданы индексы:

```
CREATE INDEX ON kostochka.gryz
  (custom_value_id);
CREATE INDEX ON kostochka.partia_gryza
   (reg_number_id);
CREATE INDEX ON kostochka.partia_gryza
   (sender_id);
CREATE INDEX ON kostochka.partia_gryza
   (recipient_id);
CREATE INDEX ON kostochka.marshryt
   (reg_number_id)
```

2. ПРАКТИЧЕСКАЯ ЧАСТЬ

2.1 Реализация базы данных

Для создания прикладного программного обеспечения следует создать базу данных в SQLShell, заполнение таблиц производилось в данной среде.

В созданную ранее базу данных persik добавляем основные таблицы:

Суда (sydno), Маршруты (marshryt), Отправители (otpravitel), Получатели (polychatel), Партии грузов (partia_gryza), Грузы (gryz).

В результате будет создана база данных (Рис. 3)

```
Server [localhost]:
Database [postgres]:
Port [5432]:
Username [postgres]:
Пароль пользователя postgres:

psql (16.3)
ПРЕДУПРЕЖДЕНИЕ: Кодовая страница консоли (866) отличается от основной
                страницы Windows (1251).
                8-битовые (русские) символы могут отображаться некорректно.
                Подробнее об этом смотрите документацию psql, раздел
                "Notes for Windows users".
Введите "help", чтобы получить справку.

postgres=# \c persik;
Вы подключены к базе данных "persik" как пользователь "postgres".
```

Рис. 3 – База данных

Далее необходимо заполнить базу данных информацией.

Таблица 1 – Отношение «Суда»

	reg_number_id [PK] integer	name character varying (100)	skipper character varying (60)	type character varying (100)	capacity integer	year integer	picture text
1	11	Черная жемчужина	Иванов Петр Сергеевич	танкер	5000	2000	https://upload.wikimedia.org/wikipedia/commons/thumb/a/aa/JuliusBeckmannMainFr
2	12	Сокол тысячелетия	Пухов Кирилл Александрович	сухогруз	10000	1991	https://upload.wikimedia.org/wikipedia/ru/d/dc/Melfal.jpg
3	13	Монарх	Петров Василий Николаевич	танкер	6000	1995	https://upload.wikimedia.org/wikipedia/commons/thumb/9/9d/Thane_Creek_and_Elepl
4	14	Черное солнце	Кузнецов Игорь Александрович	танкер	5500	2005	https://rgk-palur.ru/fileadmin/_processed_/csm_3-tanker_BU_SAMRA_8955bfcd0d.jpg
5	15	Плавучий барин	Смирнов Олег Владимирович	сухогруз	8000	1989	https://3dnews.ru/assets/external/illustrations/2024/03/20/1102033/sail_00.jpg
6	16	Белый шторм	Федоров Дмитрий Викторович	танкер	4800	2010	https://rgk-palur.ru/fileadmin/_processed_/csm_4-Tanker_Mekaines_e943921494.jpg
7	17	Летучий Голландец	Козлов Андрей Александрович	сухогруз	12000	2001	https://korabley.net/upload/_nw/2/83045.jpg
8	18	Гигант	Громов Артем Сергеевич	сухогруз	15000	2003	https://img.tourister.ru/files/2/7/6/7/8/0/3/0/clones/870_580_fixedwidth.jpg
9	19	Титаник	Лебедев Сергей Павлович	танкер	6200	2012	https://maritimeforum.net/data/spravochnik/img/product-tanker.jpg
10	20	Шепот Волн	Игнатьев Владимир Михайлович	танкер	5300	2009	https://rgk-palur.ru/fileadmin/_processed_/csm_2-Tanker_Moza_-_pervyi_korabl_v_dann

Таблица 2 – Отношение «Маршруты»

	route_id [PK] integer	reg_number_id integer	departure_date timestamp with time zone	arrive_date timestamp with time zone	origin character varying (100)	destination character varying (100)
1	121	11	2024-04-01 08:00:00+10	2024-04-05 16:00:00+10	Порт Северодвинск	Порт Восточный
2	122	11	2024-04-02 10:00:00+10	2024-04-06 12:00:00+10	Порт Владивосток	Порт Мурманск
3	123	11	2024-04-03 12:00:00+10	2024-04-07 08:00:00+10	Порт Мурманск	Порт Калининград
4	124	11	2024-04-04 14:00:00+10	2024-04-08 10:00:00+10	Порт Калининград	Порт Ростов-на-Дону
5	125	12	2024-04-05 16:00:00+10	2024-04-09 08:00:00+10	Порт Ростов-на-Дону	Порт Новороссийск
6	126	12	2024-04-06 08:00:00+10	2024-04-10 10:00:00+10	Порт Новороссийск	Порт Магадан
7	127	12	2024-04-07 10:00:00+10	2024-04-11 12:00:00+10	Порт Магадан	Порт Одесса
8	128	12	2024-04-08 12:00:00+10	2024-04-12 14:00:00+10	Порт Одесса	Порт Севастополь
9	129	13	2024-04-09 14:00:00+10	2024-04-13 16:00:00+10	Порт Севастополь	Порт Новороссийск
10	130	13	2024-04-10 16:00:00+10	2024-04-14 08:00:00+10	Порт Новороссийск	Порт Владивосток
11	131	13	2024-04-11 08:00:00+10	2024-04-15 10:00:00+10	Порт Восточный	Порт Мурманск
12	132	13	2024-04-12 10:00:00+10	2024-04-16 12:00:00+10	Порт Мурманск	Порт Калининград
13	133	14	2024-04-13 12:00:00+10	2024-04-17 14:00:00+10	Порт Калининград	Порт Ростов-на-Дону
14	134	14	2024-04-14 14:00:00+10	2024-04-18 16:00:00+10	Порт Ростов-на-Дону	Порт Новороссийск
15	135	14	2024-04-15 16:00:00+10	2024-04-19 08:00:00+10	Порт Новороссийск	Порт Магадан
16	136	14	2024-04-16 08:00:00+10	2024-04-20 10:00:00+10	Порт Магадан	Порт Одесса
17	137	15	2024-04-17 10:00:00+10	2024-04-21 12:00:00+10	Порт Одесса	Порт Севастополь
18	138	15	2024-04-18 12:00:00+10	2024-04-22 14:00:00+10	Порт Севастополь	Порт Новороссийск
19	139	15	2024-04-19 14:00:00+10	2024-04-23 16:00:00+10	Порт Новороссийск	Порт Владивосток
20	140	15	2024-04-20 16:00:00+10	2024-04-24 08:00:00+10	Порт Владивосток	Порт Мурманск
21	141	16	2024-04-21 08:00:00+10	2024-04-25 10:00:00+10	Порт Мурманск	Порт Калининград
22	142	16	2024-04-22 10:00:00+10	2024-04-26 12:00:00+10	Порт Калининград	Порт Ростов-на-Дону
23	143	16	2024-04-23 12:00:00+10	2024-04-27 14:00:00+10	Порт Ростов-на-Дону	Порт Новороссийск
24	144	16	2024-04-24 14:00:00+10	2024-04-28 16:00:00+10	Порт Новороссийск	Порт Магадан

Продолжение Таблицы – 2 Отношение «Маршруты»

25	145	17	2024-04-25 16:00:00+10	2024-04-29 08:00:00+10	Порт Магадан	Порт Одесса
26	146	17	2024-04-26 08:00:00+10	2024-04-30 10:00:00+10	Порт Одесса	Порт Севастополь
27	147	17	2024-04-27 10:00:00+10	2024-05-01 12:00:00+10	Порт Севастополь	Порт Новороссийск
28	148	17	2024-04-28 12:00:00+10	2024-05-02 14:00:00+10	Порт Новороссийск	Порт Владивосток
29	149	18	2024-04-29 14:00:00+10	2024-05-03 16:00:00+10	Порт Владивосток	Порт Мурманск
30	150	18	2024-04-30 16:00:00+10	2024-05-04 08:00:00+10	Порт Мурманск	Порт Калининград
31	151	18	2024-05-01 08:00:00+10	2024-05-05 10:00:00+10	Порт Калининград	Порт Ростов-на-Дону
32	152	18	2024-05-02 10:00:00+10	2024-05-06 12:00:00+10	Порт Ростов-на-Дону	Порт Новороссийск
33	153	19	2024-05-03 12:00:00+10	2024-05-07 14:00:00+10	Порт Новороссийск	Порт Магадан
34	154	19	2024-05-04 14:00:00+10	2024-05-08 16:00:00+10	Порт Магадан	Порт Одесса
35	155	19	2024-05-05 16:00:00+10	2024-05-09 08:00:00+10	Порт Одесса	Порт Севастополь
36	156	19	2024-05-06 08:00:00+10	2024-05-10 10:00:00+10	Порт Севастополь	Порт Новороссийск
37	157	20	2024-05-07 10:00:00+10	2024-05-11 12:00:00+10	Порт Новороссийск	Порт Владивосток
38	158	20	2024-05-08 12:00:00+10	2024-05-12 14:00:00+10	Порт Владивосток	Порт Мурманск
39	159	20	2024-05-09 14:00:00+10	2024-05-13 16:00:00+10	Порт Мурманск	Порт Калининград
40	160	20	2024-05-10 16:00:00+10	2024-05-14 08:00:00+10	Порт Калининград	Порт Ростов-на-Дону

Таблица 3 – Отношение «Отправители»

	sender_id [PK] integer	sender_name character varying (100)	inn_sender integer	bank_sender character varying (100)	address_sender character varying (100)
1	401	ООО "Морские перевозки"	123456789	Банк "Морской"	ул. Морская, д. 1, г. Москва
2	402	ОАО "Транспортные грузоперевозки"	987654321	Банк "Транспортный"	пр. Грузовой, д. 5, г. Санкт-Петербург
3	403	ИП Иванов Иван Иванович	267890123	Банк "Финансовый"	ул. Первомайская, д. 10, г. Новосибирск
4	404	ООО "Грузовые перевозки"	432109876	Банк "Грузовой"	пр. Транспортный, д. 15, г. Екатеринбург
5	405	ЗАО "Морской экспресс"	187612345	Банк "Морской экспресс"	ул. Портовая, д. 20, г. Владивосток
6	406	ФГУП "Морские перевозки России"	987987651	Банк "Морской ФГУП"	пр. Морской, д. 30, г. Севастополь
7	407	ООО "Транзит"	123487650	Банк "Транзитный"	ул. Транзитная, д. 25, г. Калининград
8	408	ИП Петров Петр Петрович	987698741	Банк "Финансовый"	пр. Финансовый, д. 35, г. Астрахань
9	409	ОАО "Перевозки и логистика"	32348765	Банк "Перевозки"	ул. Логистическая, д. 40, г. Мурманск
10	410	ЗАО "Грузовик"	187612345	Банк "Грузовой"	пр. Грузовиков, д. 45, г. Ростов-на-Дону
11	411	ООО "Морские грузоперевозки"	223456789	Банк "Морской"	ул. Морская, д. 11, г. Москва
12	412	ОАО "Транзитные перевозки"	287654321	Банк "Транспортный"	пр. Транзитный, д. 12, г. Санкт-Петербург
13	413	ИП Петров Петр Петрович	167890123	Банк "Финансовый"	ул. Финансовая, д. 13, г. Новосибирск
14	414	ООО "Грузоперевозки и поставки"	432109876	Банк "Грузовой"	пр. Поставочный, д. 14, г. Екатеринбург
15	415	ЗАО "Морской грузовик"	187612345	Банк "Морской экспресс"	ул. Морская, д. 15, г. Владивосток
16	416	ФГУП "Грузоперевозки России"	187698765	Банк "Морской ФГУП"	пр. Грузовой, д. 16, г. Севастополь
17	417	ООО "Транспортные поставки"	123487650	Банк "Транзитный"	ул. Поставочная, д. 17, г. Калининград
18	418	ИП Петров Петр Петрович	187698741	Банк "Финансовый"	пр. Финансовый, д. 18, г. Астрахань
19	419	ОАО "Грузоперевозки и логистика"	123487651	Банк "Перевозки"	ул. Логистическая, д. 19, г. Мурманск
20	420	ЗАО "Грузовая компания"	187612345	Банк "Грузовой"	пр. Грузовиков, д. 20, г. Ростов-на-Дону

Таблица 4 – Отношение «Получатели»

	recipient_id [PK] integer	consignee_name character varying (100)	inn_consignee integer	bank_consignee character varying (100)	address_consign character varying (100)
1	101	ООО "Грузополучатели"	123456789	Банк "Грузовой"	ул. Грузовая, д. 1, г. Москва
2	102	ОАО "Транспортные поставки"	987654321	Банк "Транспортный"	пр. Грузовой, д. 5, г. Санкт-Петербург
3	103	ИП Сидоров Сидор Сидорович	567890123	Банк "Финансовый"	ул. Сидорова, д. 10, г. Новосибирск
4	104	ООО "Товарополучатели и грузоперевозки"	432109876	Банк "Товарный"	пр. Товарный, д. 15, г. Екатеринбург
5	105	ЗАО "Грузоперевозки и поставки"	987612345	Банк "Грузовые поставки"	ул. Поставочная, д. 20, г. Владивосток
6	106	ФГУП "Транспортные перевозки России"	987698765	Банк "Транспортный ФГУП"	пр. Транспортный, д. 30, г. Севастополь
7	107	ООО "Поставки и грузоперевозки"	123487650	Банк "Поставки"	ул. Доставочная, д. 25, г. Калининград
8	108	ИП Иванов Иван Иванович	987698741	Банк "Финансовый"	пр. Финансовый, д. 35, г. Астрахань
9	109	ОАО "Грузовики и поставки"	123487651	Банк "Грузовой"	ул. Грузовиков, д. 40, г. Мурманск
10	110	ЗАО "Транспорт"	987612345	Банк "Транспортный"	пр. Транспортный, д. 45, г. Ростов-на-Дону
11	111	ООО "Грузополучатели"	123456789	Банк "Грузовой"	ул. Грузовая, д. 11, г. Москва
12	112	ОАО "Транспортные поставки"	987654321	Банк "Транспортный"	пр. Грузовой, д. 12, г. Санкт-Петербург
13	113	ИП Сидоров Сидор Сидорович	567890123	Банк "Финансовый"	ул. Сидорова, д. 13, г. Новосибирск
14	114	ООО "Товарополучатели и грузоперевозки"	432109876	Банк "Товарный"	пр. Товарный, д. 14, г. Екатеринбург
15	115	ЗАО "Грузоперевозки и поставки"	987612345	Банк "Грузовые поставки"	ул. Поставочная, д. 15, г. Владивосток
16	116	ФГУП "Транспортные перевозки России"	987698765	Банк "Транспортный ФГУП"	пр. Транспортный, д. 16, г. Севастополь
17	117	ООО "Поставки и грузоперевозки"	123487651	Банк "Поставки"	ул. Доставочная, д. 17, г. Калининград
18	118	ИП Иванов Иван Иванович	987698741	Банк "Финансовый"	пр. Финансовый, д. 18, г. Астрахань
19	119	ОАО "Грузовики и поставки"	123487651	Банк "Грузовой"	ул. Грузовиков, д. 19, г. Мурманск
20	120	ЗАО "Транспорт"	987612345	Банк "Транспортный"	пр. Транспортный, д. 20, г. Ростов-на-Дону

Таблица 5 – Отношение «Партии грузов»

	custom_value_id [PK] integer	reg_number_id integer	sender_id integer	sender_name character varying (100)	recipient_id integer	consignee_name character varying (100)	custom_clearance boolean
1	201	11	401	ООО "Морские перевозки"	101	ООО "Грузополучатели"	true
2	202	11	402	ОАО "Транспортные грузоперевозки"	102	ОАО "Транспортные поставки"	false
3	203	12	403	ИП Иванов Иван Иванович	103	ИП Сидоров Сидор Сидорович	true
4	204	12	404	ООО "Грузовые перевозки"	104	ООО "Товарополучатели и грузоперевозки"	false
5	205	13	405	ЗАО "Морской экспресс"	105	ЗАО "Грузоперевозки и поставки"	true
6	206	13	406	ФГУП "Морские перевозки России"	106	ФГУП "Транспортные перевозки России"	false
7	207	14	407	ООО "Транзит"	107	ООО "Поставки и грузоперевозки"	true
8	208	14	408	ИП Петров Петр Петрович	108	ИП Иванов Иван Иванович	false
9	209	15	409	ОАО "Перевозки и логистика"	109	ОАО "Грузовики и поставки"	true
10	210	15	410	ЗАО "Грузовик"	110	ЗАО "Транспорт"	false
11	211	16	411	ООО "Морские грузоперевозки"	111	ООО "Грузополучатели"	true
12	212	16	412	ОАО "Транзитные перевозки"	112	ОАО "Транспортные поставки"	false
13	213	17	413	ИП Иванов Иван Иванович	113	ИП Сидоров Сидор Сидорович	true
14	214	17	414	ООО "Грузоперевозки и поставки"	114	ООО "Товарополучатели и грузоперевозки"	false
15	215	17	414	ООО "Грузоперевозки и поставки"	114	ООО "Товарополучатели и грузоперевозки"	false
16	216	18	415	ЗАО "Морской грузовик"	115	ЗАО "Грузоперевозки и поставки"	true
17	217	18	415	ЗАО "Морской грузовик"	115	ЗАО "Грузоперевозки и поставки"	true
18	218	18	416	ФГУП "Грузоперевозки России"	116	ФГУП "Транспортные перевозки России"	false
19	219	18	416	ФГУП "Грузоперевозки России"	116	ФГУП "Транспортные перевозки России"	false
20	220	19	417	ООО "Транспортные поставки"	117	ООО "Поставки и грузоперевозки"	true
21	221	19	417	ООО "Транспортные поставки"	117	ООО "Поставки и грузоперевозки"	true
22	222	19	418	ИП Петров Петр Петрович	118	ИП Иванов Иван Иванович	false
23	223	20	419	ОАО "Грузоперевозки и логистика"	119	ОАО "Грузовики и поставки"	true
24	224	20	419	ОАО "Грузоперевозки и логистика"	119	ОАО "Грузовики и поставки"	true

Таблица – 6 Отношение «Грузы»

	custom_value_id integer	number_id [PK] integer	shipment character varying (100)	declare_value integer	unit character varying (100)	insure_value integer	number_cargo text
1	201	51	Мука	10000	кг	12000	Срок годности: 6 месяцев
2	201	52	Сахар	8000	кг	10000	Без добавок
3	202	53	Рис	12000	кг	14000	Премиум качество
4	202	54	Кофе	500	кг	600	Арабика
5	203	55	Чай	300	кг	400	Зеленый
6	203	56	Масло	200	л	300	Оливковое
7	204	57	Соль	100	кг	150	Поваренная
8	204	58	Перец	50	кг	70	Черный
9	205	59	Мука	15000	кг	17000	Для выпечки
10	205	60	Сахар	10000	кг	12000	Песок
11	206	61	Рис	18000	кг	20000	Овсяный
12	206	62	Кофе	700	кг	900	Робуста
13	207	63	Чай	400	кг	500	Черный
14	207	64	Масло	300	л	400	Растительное
15	208	65	Соль	200	кг	250	Морская
16	208	66	Перец	100	кг	120	Белый
17	209	67	Мука	20000	кг	22000	Кукурузная
18	209	68	Сахар	15000	кг	17000	Кристаллический
19	210	69	Рис	25000	кг	27000	Белый
20	210	70	Кофе	900	кг	1100	Арабика/Робуста
21	211	71	Чай	500	кг	600	Зеленый/Черный
22	211	72	Масло	400	л	500	Оливковое/Подсолнечное
23	212	73	Соль	250	кг	300	Каменная/Поваренная
24	212	74	Сок	2500	л	2750	Апельсиновый

Продолжение Таблицы – 6 Отношение «Грузы»

	custom_value_id integer	number_id [PK] integer	shipment character varying (100)	declare_value integer	unit character varying (100)	insure_value integer	number_cargo text
25	212	75	Перец	120	кг	150	Черный/Белый
26	213	76	Мука	25000	кг	27000	Пшеничная/Кукурузная
27	213	77	Сахар	20000	кг	22000	Тростниковый/Свекловичный
28	214	78	Рис	35000	кг	37000	Дикий/Круглозерный
29	214	79	Сок	3500	л	3700	Мультифрукт
30	214	80	Кофе	1200	кг	1400	Молотый/Зерновой
31	215	81	Чай	700	кг	800	Листовой/Саше
32	215	82	Масло	600	л	700	Сливочное/Растительное
33	216	83	Соль	350	кг	400	Морская/Каменная
34	216	84	Перец	180	кг	200	Черный/Красный
35	217	85	Мука	35000	кг	37000	Пшеничная/Кукурузная
36	217	86	Сахар	25000	кг	27000	Тростниковый/Свекловичный
37	218	87	Рис	45000	кг	47000	Дикий/Круглозерный
38	218	88	Кофе	1500	кг	1700	Молотый/Зерновой
39	219	89	Чай	800	кг	900	Листовой/Саше
40	219	90	Масло	700	л	800	Сливочное/Растительное
41	220	91	Соль	400	кг	450	Морская/Каменная
42	220	92	Чай	990	кг	1200	Черный/Каркаде
43	221	93	Перец	225	кг	285	Черный/Красный
44	221	94	Консервы	3400	кг	3650	Тушонка
45	221	95	Кофе	1500	кг	1400	Молотый/Зерновой
46	222	96	Кофе	1000	кг	1100	Молотый/Зерновой
47	222	97	Рис	49002	кг	50000	Дикий/Круглозерный
48	222	98	Чай	2000	л	2100	Липтон

Продолжение Таблицы – 6 Отношение «Грузы»

49	223	99	Перец	290	кг	300	Черный/Красный
50	223	100	Мука	13000	кг	13900	Кукурузная
51	223	301	Сок	1000	л	1300	Вишневый/Земляничный
52	224	302	Перец	500	кг	550	Черный/Красный
53	224	303	Масло	550	л	600	Растительное
54	225	304	Сок	1000	л	1200	Персиковый/Яблочный
55	225	305	Сахар	2200	кг	2500	Тростниковый/Свекловичный

Первичные и внешние ключи таблиц базы данных были определены еще на моменте создания таблиц, то есть в теле кода с помощью операторов PRIMARY KEY и REFERENCES.

Таким образом, была создана база данных, таблицы в этой базе данных, установлены первичные ключи и установлена связь между ними с помощью внешних ключей.

2.2 Создание прикладного ПО

Разработка программного обеспечения реализуется в среде разработки PyCharm на языке Python.

При создании приложения нужна связь между программой и базой данных для эффективной работы с таблицами напрямую. При вызове баз данных для доступа к ее ресурсам требуется прописать следующие команды в коде (Рис.4). Для этого используем библиотеку Psycopg2 и функции фреймворка Flask. Psycopg2 широко используется и протестирован в Python для подключения к базам данных PostgreSQL. Он обеспечивает высокую производительность благодаря бинарному протоколу. Flask используется для импорта баз данных в Python.

```python
import psycopg2 as pg
from flask import Flask, request, render_template

app = Flask(__name__)

# Параметры подключения к базе данных
DATABASE = {
    'user': 'postgres',
    'password': '105100105',
    'host': 'localhost',
    'port': '5432',
    'dbname': 'persik'
}

try:
    # Подключение к базе данных
    conn = pg.connect(**DATABASE)
    print("Connected to PostgreSQL (УСПЕШНО)")
    cur = conn.cursor()
except:
    print("Unable to connect to PostgreSQL (ОШИБКА)")
```

Рис. 4 – Подключение к базе данных

Для получения списка таблиц баз данных (для примера взята таблица marshryt) была создана отдельная функция (Рис. 6). Предварительно создадим html-файл index.html, в который пропишем информацию о структуре и контенте веб-страницы (Приложение 1).

```python
@app.route('/')
def index():
        # Получение данных из таблицы
        cur.execute("SELECT * FROM kostochka.marshryt")
        rows = cur.fetchall()

        return render_template( template_name_or_list: 'index.html', data=rows)
```

Рис. 6 – Функция для получения таблицы

Результат работы будет получен с помощью ссылки (Рис. 7)

Рис. 7– Ссылка при запуске приложения

Осуществив переход по ссылке, после чего открывается отдельная вкладка в браузере, где отображается таблица (Рис. 8).

Идентификатор маршрута	Идентификатор судна	Дата убывания судна	Дата пребывания судна	Пункт отправления	Пункт назначения
121	11	2024-04-01 08:00:00+10:00	2024-04-05 16:00:00+10:00	Порт Северодвинск	Порт Восточный
122	11	2024-04-02 10:00:00+10:00	2024-04-06 12:00:00+10:00	Порт Владивосток	Порт Мурманск
123	11	2024-04-03 12:00:00+10:00	2024-04-07 08:00:00+10:00	Порт Мурманск	Порт Калининград
124	11	2024-04-04 14:00:00+10:00	2024-04-08 10:00:00+10:00	Порт Калининград	Порт Ростов-на-Дону
125	12	2024-04-05 16:00:00+10:00	2024-04-09 08:00:00+10:00	Порт Ростов-на-Дону	Порт Новороссийск
126	12	2024-04-06 08:00:00+10:00	2024-04-10 10:00:00+10:00	Порт Новороссийск	Порт Магадан
127	12	2024-04-07 10:00:00+10:00	2024-04-11 12:00:00+10:00	Порт Магадан	Порт Одесса
128	12	2024-04-08 12:00:00+10:00	2024-04-12 14:00:00+10:00	Порт Одесса	Порт Севастополь
129	13	2024-04-09 14:00:00+10:00	2024-04-13 16:00:00+10:00	Порт Севастополь	Порт Новороссийск
130	13	2024-04-10 16:00:00+10:00	2024-04-14 08:00:00+10:00	Порт Новороссийск	Порт Владивосток

Рис. 8 – Отображение таблицы во вкладке в браузере

Реализуем возможность добавления новых строк в окне браузера. Для этого необходимо создать еще одну функцию с использованием GET и POST запросов (Рис. 9). Для этой функции создадим файл ADD.html (Приложение Б).

```
@app.route( rule: '/add', methods=['POST', 'GET'])
def add():

    if request.method == 'GET':
        return render_template('add.html')
    # Добавление новой строки
    if request.method == 'POST':
        route_id = request.form['route_id']
        reg_number_id = request.form['reg_number_id']
        departure_date = request.form['departure_date']
        arrive_date = request.form['arrive_date']
        origin = request.form['origin']
        destination = request.form['destination']


    cur.execute("INSERT INTO kostochka.marshryt (route_id, reg_number_id, departure_date, arrive_date, origin, destination) "
                "VALUES (%s,%s,%s,%s,%s,%s)", (route_id, reg_number_id, departure_date, arrive_date, origin, destination))
    conn.commit()
    return render_template('add.html')
```

Рис. 9 – Функция добавления строк

На странице вывода таблицы появляется окно, имеющее поля для ввода новой строки (Рис. 10).

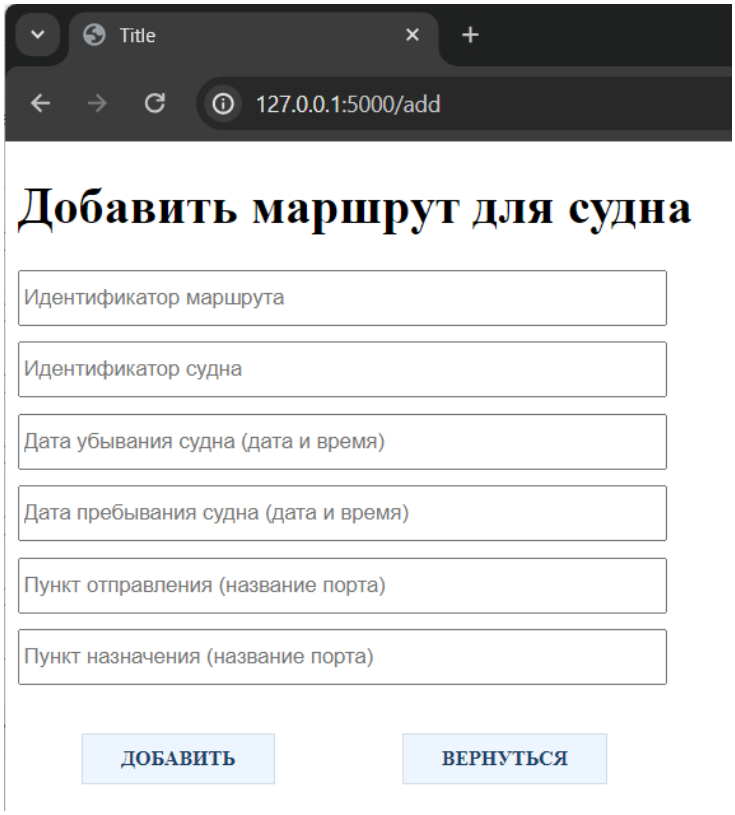

Рис. 10 – Окно ввода новой строки

После ввода данных и обновления страницы появляется возможность увидеть новые данные.

Добавим более сложные запросы:

1) Информация об общей величине грузов по категории

```
SELECT shipment, SUM(declare_value) AS total_declare_value, unit
FROM kostochka.gryz
GROUP BY shipment, unit;
```

2) Информация о наличии таможенной декларации у партии грузов

```
SELECT pg.reg_number_id, s.name, s.skipper, pg.custom_value_id, pg.custom_clearance
FROM kostochka.partia_gryza pg
JOIN kostochka.sydno s ON pg.reg_number_id = s.reg_number_id
WHERE pg.custom_clearance = true;
```

3) Информация о количестве партий грузов на судне

```
SELECT
   s.reg_number_id,
   s.name,
   s.skipper,
   COUNT(pg.reg_number_id) AS num_of_shipments
FROM
   kostochka.sydno s
LEFT JOIN
   kostochka.partia_gryza pg ON s.reg_number_id = pg.reg_number_id
GROUP BY
   s.reg_number_id, s.name, s.skipper
ORDER BY
   s.reg_number_id ASC;
```

Для выполнения запроса необходимо добавить три новые функции (Рис. 11 - 13). Для каждого запроса был создан отдельный html-файл (Приложение 3, 4, 5).

```python
@app.route( rule: '/req1', methods=['GET'])
def req1():
    if request.method == 'GET':
        cur.execute("SELECT shipment, SUM(declare_value) AS total_declare_value, unit FROM kostochka.gryz GROUP BY shipment, unit;")
        rows = cur.fetchall()
        return render_template( template_name_or_list: 'Request1.html', data=rows)
```

Рис. 11 – Информация об общей величине грузов по категории

```python
@app.route( rule: '/req2', methods=['GET'])
def req2():
    if request.method == 'GET':
        cur.execute("SELECT pg.reg_number_id, s.name, s.skipper, pg.custom_value_id, pg.custom_clearance FROM kostochka.partia_gryza pg JOIN kostochka.sydno
        rows = cur.fetchall()
        return render_template( template_name_or_list: 'Request2.html', data=rows)
```

Рис. 12 – Информация о наличии таможенной декларации у партии грузов

```python
@app.route( rule: '/req3', methods=['GET'])
def req3():
    if request.method == 'GET':
        cur.execute("SELECT s.reg_number_id, s.name, s.skipper, COUNT(pg.reg_number_id) AS num_of_shipments FROM  kostochka.sydno
        rows = cur.fetchall()
        return render_template( template_name_or_list: 'Request3.html', data=rows)
```

Рис. 13 – Информация о количестве партий грузов на судне

Результаты запросов в браузере (Рис. 14 – 16):

127.0.0.1:5000/req1

Информация об общей величине грузов по категории

Название груза	Общая заявленая величина	Единица измерения груза
Перец	1465	кг
Соль	1300	кг
Сок	8000	л
Чай	2000	л
Сахар	80200	кг
Чай	3690	кг
Масло	2750	л
Кофе	7300	кг
Рис	184002	кг
Консервы	3400	кг
Мука	118000	кг

ВЕРНУТЬСЯ

Рис. 14 – Первый запрос

Информация о наличии таможенной декларации у партии грузов

ID судна	Название судна	ФИО капитана	ID партии грузов	Наличие таможенной декларации
11	Черная жемчужина	Иванов Петр Сергеевич	201	True
12	Сокол тысячелетия	Пухов Кирилл Александрович	203	True
13	Монарх	Петров Василий Николаевич	205	True
14	Черное солнце	Кузнецов Игорь Александрович	207	True
15	Плавучий барин	Смирнов Олег Владимирович	209	True
16	Белый шторм	Федоров Дмитрий Викторович	211	True
17	Летучий Голландец	Козлов Андрей Александрович	213	True
18	Гигант	Громов Артем Сергеевич	216	True
18	Гигант	Громов Артем Сергеевич	217	True
19	Титаник	Лебедев Сергей Павлович	220	True
19	Титаник	Лебедев Сергей Павлович	221	True
20	Шепот Волн	Игнатьев Владимир Михайлович	223	True
20	Шепот Волн	Игнатьев Владимир Михайлович	224	True

ВЕРНУТЬСЯ

Рис. 15 – Второй запрос

Информация о количестве партий грузов на судне

ID судна	Название судна	ФИО капитана	Количество партий грузов
11	Черная жемчужина	Иванов Петр Сергеевич	2
12	Сокол тысячелетия	Пухов Кирилл Александрович	2
13	Монарх	Петров Василий Николаевич	2
14	Черное солнце	Кузнецов Игорь Александрович	2
15	Плавучий барин	Смирнов Олег Владимирович	2
16	Белый шторм	Федоров Дмитрий Викторович	2
17	Летучий Голландец	Козлов Андрей Александрович	3
18	Гигант	Громов Артем Сергеевич	4
19	Титаник	Лебедев Сергей Павлович	3
20	Шепот Волн	Игнатьев Владимир Михайлович	3

ВЕРНУТЬСЯ

Рис. 16 – Третий запрос

Таким образом, мы разработали программное обеспечение для судоходной компании «Балтика». Полный код программы находится в Приложении 6.

ЗАКЛЮЧЕНИЕ

В результате выполнения курсовой была достигнута поставленная цель – это разработка приложения для работы с базой данных в PostgreSQL на примере разработки прикладного программного обеспечения для судоходной компании «Балтика».

В процессе работы над приложением был проведен анализ деятельности предприятия, спроектирована архитектура и реализована база данных и создано прикладное ПО.

В соответствии с поставленной целью, были выполнены следующие задачи:

- Была спроектирована база данных, соответствующей требованиям конкретной предметной области.
- Была создана база данных с использованием PostgreSQL.
- Было разработано прикладное программное обеспечение.

Выполнение данной курсовой работы позволило углубить теоретические знания и практические навыки в области разработки программного обеспечения с использованием баз данных. Полученные результаты могут быть использованы для дальнейшего совершенствования и развития системы управления деятельностью судоходной компании.

В целом, разработанное приложение является эффективным инструментом для автоматизации процессов составления заявок и управления информацией, что способствует повышению производительности и качества работы организации. Наличие данной разработки ускоряет процесс работы предприятия и значительно сокращает время, необходимое для поиска и упорядочивания физических документов, улучшает эффективность и снижает операционные расходы.

Для реализации поставленной задачи были использованы следующие средства разработки: Draw.io, StarUML, PostgreSQL, PyCharm и язык разработки Python.

Список использованных источников

1) DFD: примеры и правила построения диаграмм потоков данных // GB URL: https://gb.ru/blog/chto-takoe-baza-dannyh/ (дата обращения: 29.05.2024).
2) Что такое ER-диаграмма и как ее создать // Lucidchart URL:https://www.lucidchart.com/pages/ru/erd%D0%B4%D0%B8%D0%B0%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%B0 (дата обращения: 29.05.2024) .
3) Кара-Ушанов В.Ю., «SQL — язык реляционных баз данных», 2018г.
4) Лузанов Павел Вениаминович, Рогов Егор Валерьевич, «POSTGRES. Первое знакомство», 2022 г.
5) Документация к Postgres Pro Standard 14.12.1 // PostgresPro URL: https://postgrespro.ru/docs/postgrespro/14/index (дата обращения: 29.05.2024).
6) Аллен Б. Дауни, «Основы Python», 2021г.
7) Моргунов Е.П., «ЯЗЫК SQL. Базовый курс: учебно-практическое пособие», 2018 г.

ПРИЛОЖЕНИЕ 1

Код файла ADD.html

```
lang="en">
<head>
    <meta charset="UTF-8">
    <title>Title</title>
    <style>
  .form-group {
    display: flex;
    flex-direction: column;
  }
  .form-control {
    height: 30px;
    width: 40%;
    margin-bottom: 10px;
  }
  .button {
                    background-color: #edf6ff; /* Цвет фона */
                    color: #1b4268; /* Цвет текста */
                    font: bold 0.8em Times New Roman, sans-serif; /* Параметры
шрифта */
                    border: 1px solid #d4dde5; /* Параметры рамки */
                    padding: 0.5rem 1.5rem; /* Расстояние от рамки до текста */
                    text-decoration: none; /* Убираем подчёркивание */
                    text-transform: uppercase; /* Все буквы большие */
                   /*  border-radius: 0px; Радиус скругления рамки */
                    transition: 0.5s; /* Время перехода */
                    margin-top: 20px; /*  Отступ сверху  */
                    margin-bottom: 20px; /*  Отступ снизу  */
                    margin-left: 40px;
                    margin-right: 40px;
                    float: left;
                    display: inline-block;
                   }

</style>
</head>
{#<body>#}
{#    <div class = "container">#}
{#    <form method="post" action="/add">#}
{#        <legend>Добавить маршрут для судна</legend>#}
{#        <input type="text" name="route_id" placeholder="Идентификатор
маршрута">#}
{#         <input type="text" name="reg_number_id" placeholder="Идентификатор
судна">#}
{#         <input type="text" name="departure_date" placeholder="Дата убывания
судна (дата и время)">#}
{#         <input type="text" name="arrive_date" placeholder="Дата пребывания
судна (дата и время)">#}
{#         <input type="text" name="origin" placeholder="Пункт отправления
(название порта)">#}
{#         <input type="text" name="destination" placeholder="Пункт назначения
(название порта)">#}
{#        <button type="submit">Добавить</button>#}
{#    </form>#}
{#            <form method="get", action="/">#}
{#            <a href="/">Вернуться</a>#}
```

```
{#            </form>#}
{#    </div>#}
{#</body>#}
<body class="container">
  <form method="post" action="/add">
    <h1>Добавить маршрут для судна</h1>
    <div class="form-group">
      <input type="text" name="route_id" placeholder="Идентификатор маршрута "
class="form-control">
    </div>
   <div class="form-group">
      <input type="text" name="reg_number_id" placeholder="Идентификатор судна"
class="form-control">
    </div>
    <div class="form-group">
      <input type="text" name="departure_date" placeholder="Дата убывания судна
(дата и время)" class="form-control">
    </div>
    <div class="form-group">
      <input type="text" name="arrive_date" placeholder="Дата пребывания судна
(дата и время)" class="form-control">
    </div>
    <div class="form-group">
      <input type="text" name="origin" placeholder="Пункт отправления (название
порта)" class="form-control">
    </div>
    <div class="form-group">
      <input type="text" name="destination" placeholder="Пункт назначения
(название порта)" class="form-control">
    </div>
    <button class = "button"type="submit">Добавить</button>
  </form>
  <a href="/" class="button">Вернуться</a>
</body>
</html>
```

ПРИЛОЖЕНИЕ 2

Код файла index.html

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>Таблица данных о отправителях</title>
</head>
<body>
    <style>
        table {
            border-collapse: collapse; /*   Убираем двойные границы между ячейками   */
        }
        th, td {
            border: 1px solid black; /*   Добавляем границу вокруг каждой ячейки */
            padding: 8px; /*   Добавляем немного пространства между содержимым и границей   */
        }
```



```
           .button {
                    background-color: #edf6ff; /* Цвет фона */
                    color: #1b4268; /* Цвет текста */
                    font: bold 0.8em Times New Roman, sans-serif; /* Параметры шрифта */
                    border: 1px solid #d4dde5; /* Параметры рамки */
                    padding: 0.5rem 1.5rem; /* Расстояние от рамки до текста */
                    text-decoration: none; /* Убираем подчёркивание */
                    text-transform: uppercase; /* Все буквы большие */
                   /*  border-radius: 0px; Радиус скругления рамки */
                    transition: 0.5s; /* Время перехода */
                    margin-top: 20px; /*  Отступ сверху  */
                    margin-bottom: 20px; /*  Отступ снизу  */
                    margin-left: 40px;
                    margin-right: 40px;
                    float: left;
                    display: inline-block;
                   }
           .button:hover {
                        background-color: #f60; /* Цвет фона */
                        color: #fff; /* Цвет текста */
                        border-color: #f60; /* Цвет рамки */
                       }
    </style>
    <table>
        <tr>
            <th>Идентификатор маршрута</th>
            <th>Идентификатор судна</th>
            <th>Дата убывания судна</th>
            <th>Дата пребывания судна</th>
            <th>Пункт отправления</th>
            <th>Пункт назначения</th>
        </tr>
        {% for row in data %}
            <tr>
                {% for cell in row %}
                    <td>{{ cell }}</td>
```

```
                {% endfor %}
            </tr>
        {% endfor %}
    </table>
    <form method="post", action="/add">
            <a href="/add" class="button">Добавить</a>
    </form>
    <form method="get", action="/req1">
            <a href="/req1" class="button">Информация об общей величине грузов
по категории</a>
    </form>
    </form>
    <form method="get", action="/req2">
            <a href="/req2" class="button">Информация о наличии таможенной
декларации у партии грузов</a>
    </form>
    </form>
    <form method="get", action="/req3">
            <a href="/req3" class="button">Информация о количестве партий грузов
на судне</a>
    </form>
</body>
</html>
```

ПРИЛОЖЕНИЕ 3

Код файла Request1.html

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>Данные о суднах</title>

</head>
<body>
    <style>
        table {
            border-collapse: collapse; /*   Убираем двойные границы между
ячейками   */
        }
        th, td {
            border: 1px solid black; /*   Добавляем границу вокруг каждой ячейки
*/
            padding: 8px; /*   Добавляем немного пространства между содержимым и
границей   */
        }
             .button {
                    background-color: #edf6ff; /* Цвет фона */
                    color: #1b4268; /* Цвет текста */
                    font: bold 0.8em Times New Roman, sans-serif; /* Параметры
шрифта */
                    border: 1px solid #d4dde5; /* Параметры рамки */
                    padding: 0.5rem 1.5rem; /* Расстояние от рамки до текста */
                    text-decoration: none; /* Убираем подчёркивание */
                    text-transform: uppercase; /* Все буквы большие */
                   /*  border-radius: 0px; Радиус скругления рамки */
                    transition: 0.5s; /* Время перехода */
                    margin-top: 20px; /*  Отступ сверху  */
                    margin-bottom: 20px; /*  Отступ снизу  */
                    margin-left: 40px;
                    margin-right: 40px;
                    float: left;
                    display: inline-block;
                   }
    </style>
    <table>
        <h1>Информация об общей величине грузов по категории</h1>
        <tr>
            <th>Название груза</th>
            <th>Общая заявленая величина</th>
            <th>Единица измерения груза</th>
        </tr>
        {% for row in data %}
            <tr>
                {% for cell in row %}
                    <td>{{ cell }}</td>
                {% endfor %}
            </tr>
        {% endfor %}
    </table>
    <form method="get", action="/">
        <a href="/" class="button">Вернуться</a>
```

```
    </form>
</body>
</html>
```

ПРИЛОЖЕНИЕ 4

Код файла Request2.html

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>Информация о мастерах</title>
</head>
<body>
  <style>
        table {
            border-collapse: collapse; /*   Убираем двойные границы между
ячейками   */
        }
        th, td {
            border: 1px solid black; /*   Добавляем границу вокруг каждой ячейки
*/
            padding: 8px; /*   Добавляем немного пространства между содержимым и
границей   */
        }
            .button {
                    background-color: #edf6ff; /* Цвет фона */
                    color: #1b4268; /* Цвет текста */
                    font: bold 0.8em Times New Roman, sans-serif; /* Параметры
шрифта */
                    border: 1px solid #d4dde5; /* Параметры рамки */
                    padding: 0.5rem 1.5rem; /* Расстояние от рамки до текста */
                    text-decoration: none; /* Убираем подчёркивание */
                    text-transform: uppercase; /* Все буквы большие */
                   /*  border-radius: 0px; Радиус скругления рамки */
                    transition: 0.5s; /* Время перехода */
                    margin-top: 20px; /*  Отступ сверху  */
                    margin-bottom: 20px; /*  Отступ снизу  */
                    margin-left: 40px;
                    margin-right: 40px;
                    float: left;
                    display: inline-block;
                   }
    </style>
  <table>
        <h1>Информация о наличии таможенной декларации у партии грузов</h1>
        <tr>
            <th>ID судна</th>
            <th>Название судна</th>
            <th>ФИО капитана</th>
            <th>ID партии грузов</th>
            <th>Наличие таможенной декларации</th>
        </tr>
        {% for row in data %}
            <tr>
                {% for cell in row %}
                    <td>{{ cell }}</td>
```

Продолжение приложения 4

```
                {% endfor %}
            </tr>
        {% endfor %}
```

```
    </table>
    <form method="get", action="/">
        <a href="/" class="button">Вернуться</a>
    </form>
</body>
</html>
```

ПРИЛОЖЕНИЕ 5

Код файла Request3.html

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>Информация о технике</title>
</head>
<body>
    <style>
        table {
            border-collapse: collapse; /*   Убираем двойные границы между
ячейками   */
        }
        th, td {
            border: 1px solid black; /*   Добавляем границу вокруг каждой ячейки
*  /
            padding: 8px; /*   Добавляем немного пространства между содержимым и
границей   */
        }
            .button {
                    background-color: #edf6ff; /* Цвет фона */
                    color: #1b4268; /* Цвет текста */
                    font: bold 0.8em Times New Roman, sans-serif; /* Параметры
шрифта */
                    border: 1px solid #d4dde5; /* Параметры рамки */
                    padding: 0.5rem 1.5rem; /* Расстояние от рамки до текста */
                    text-decoration: none; /* Убираем подчёркивание */
                    text-transform: uppercase; /* Все буквы большие */
                   /*  border-radius: 0px; Радиус скругления рамки */
                    transition: 0.5s; /* Время перехода */
                    margin-top: 20px; /*  Отступ сверху  */
                    margin-bottom: 20px; /*  Отступ снизу  */
                    margin-left: 40px;
                    margin-right: 40px;
                    float: left;
                    display: inline-block;
                   }
    </style>
<table>
        <h1>Информация о количестве партий грузов на судне</h1>
        <tr>
            <th>ID судна</th>
            <th>Название судна</th>
            <th>ФИО капитана</th>
            <th>Количество партий грузов</th>
```

Продолжение приложения 5

```
        </tr>
        {% for row in data %}
            <tr>
                {% for cell in row %}
                    <td>{{ cell }}</td>
                {% endfor %}
            </tr>
```

```
        {% endfor %}
    </table>
    <form method="get", action="/">
        <a href="/" class="button">Вернуться</a>
    </form>
</body>
</html>
```

ПРИЛОЖЕНИЕ 6

Код файла Request3.html

```
import psycopg2 as pg
from flask import Flask, request, render_template

app = Flask(__name__)

# Параметры подключения к базе данных
DATABASE = {
    'user': 'postgres',
    'password': '105100105',
    'host': 'localhost',
    'port': '5432',
    'dbname': 'persik'
}

try:
    # Подключение к базе данных
    conn = pg.connect(**DATABASE)
    print("Connected to PostgreSQL (УСПЕШНО)")
    cur = conn.cursor()
except:
    print("Unable to connect to PostgreSQL (ОШИБКА)")

@app.route('/')
def index():
        # Получение данных из таблицы
        cur.execute("SELECT * FROM kostochka.marshryt")
        rows = cur.fetchall()

        return render_template('index.html', data=rows)
@app.route('/add', methods=['POST', 'GET'])
def add():

    if request.method == 'GET':
        return render_template('add.html')
    # Добавление новой строки
    if request.method == 'POST':
        route_id = request.form['route_id']
        reg_number_id = request.form['reg_number_id']
```



```
        departure_date = request.form['departure_date']
        arrive_date = request.form['arrive_date']
        origin = request.form['origin']
        destination = request.form['destination']


    cur.execute("INSERT INTO kostochka.marshryt (route_id, reg_number_id,
departure_date, arrive_date, origin, destination) "
                "VALUES (%s,%s,%s,%s,%s,%s)", (route_id, reg_number_id,
departure_date, arrive_date, origin, destination))
    conn.commit()
    return render_template('add.html')

@app.route('/req1', methods=['GET'])
def req1():
```

```
    if request.method == 'GET':
        cur.execute("SELECT shipment, SUM(declare_value) AS total_declare_value,
unit FROM kostochka.gryz GROUP BY shipment, unit;")
        rows = cur.fetchall()
        return render_template('Request1.html', data=rows)

@app.route('/req2', methods=['GET'])
def req2():
    if request.method == 'GET':
        cur.execute("SELECT pg.reg_number_id, s.name, s.skipper,
pg.custom_value_id, pg.custom_clearance FROM kostochka.partia_gryza pg JOIN
kostochka.sydno s ON pg.reg_number_id = s.reg_number_id WHERE
pg.custom_clearance = true;;")
        rows = cur.fetchall()
        return render_template('Request2.html', data=rows)

@app.route('/req3', methods=['GET'])
def req3():
    if request.method == 'GET':
        cur.execute("SELECT s.reg_number_id, s.name, s.skipper,
COUNT(pg.reg_number_id) AS num_of_shipments FROM  kostochka.sydno s LEFT JOIN
kostochka.partia_gryza pg ON s.reg_number_id = pg.reg_number_id GROUP BY
s.reg_number_id, s.name, s.skipper ORDER BY s.reg_number_id ASC;")
        rows = cur.fetchall()
        return render_template('Request3.html', data=rows)


if __name__ == '__main__':
    app.run(debug=True)
```