Одеський національний університет імені І. І. Мечникова
Факультет математики, фізики та інформаційних технологій
Кафедра комп'ютерних систем та технологій

Кваліфікаційна робота

на здобуття ступеня вищої освіти «бакалавр»
«Інформаційна система фінансово-технічного управління для судновласницьких компаній»
«Information system of financial and technical management for shipowners companies»

Виконав: здобувач денної форми навчання
спеціальності 123 – Комп'ютерна інженерія
Освітня програма «Комп'ютерна інженерія»

Вдовіченко Володимир Олександрович

Керівник

__д. т.н., доц. кафедри КСТ, Михайленко В. С.__

(науковий ступінь, вчене звання, прізвище та ініціали, підпис)

Рецензент

_____

(науковий ступінь, вчене звання, прізвище та ініціали)

Рекомендовано до захисту:
Протокол засідання кафедри
№ ___ від _____ р.

Завідувач кафедри

_____ Юрій ГУНЧЕНКО
(підпис) (прізвище, ім'я)

Захищено на засіданні ЕК №_____
протокол № ____ від _____ р.
Оцінка_____/___ ___/_____
(за національною шкалою, шкалою ECTS, бали)
Голова ЕК

_____ Алла КОБОЗЄВА
(підпис) (прізвище, ім'я)

Одеса – 2023

АНОТАЦІЯ

Ціль проекту полягає у створенні інформаційної системи фінансово-технічного управління для судновласницьких компаній, що постає у вигляді веб-додатку із зрозумілим інтерфейсом для певних груп службовців. Користувачами даної системи є капітани суден, службовці технічного відділу, крюїнгового відділу, фінансового відділу, менеджери (оператори суден) та адміністратори системи. Реалізація проекту виконана з використанням мови програмування JavaScript і СУБД PostgreSQL. Архітектура системи відповідає шаблону MVC.

Ця система дозволяє оптимізувати управління фінансовими ресурсами, планувати і контролювати технічне обслуговування та ремонт суден, здійснювати ефективний моніторинг витрат на паливо й інші матеріали, а також здійснювати заміну членів екіпажу суден.

Одним із ключових елементів інформаційної системи є автоматизація фінансового обліку, що дозволяє відстежувати фінансові операції компанії та забезпечує швидкий аналіз фінансової діяльності.

Крім того, інформаційна система фінасово-технічного управління забезпечує збір та аналіз даних про технічний стан суден, що дозволяє планувати та проводити ремонтні роботи вчасно та ефективно.

Застосування інформаційної системи фінасово-технічного управління допомагає судновласницьким компаніям знизити витрати на управління та технічне обслуговування суден, підвищити ефективність управління фінансовими ресурсами та забезпечити безперебійну експлуатацію суден.

ABSTRACT

The goal of the project is to create an information system for financial and technical management of shipowning companies. It takes the form of a web application with a user-friendly interface for specific groups of employees. The users of this system include ship captains, technical department staff, crewing department staff, finance department staff, vessel operators (managers), and system administrators. The project implementation is done using JavaScript programming language and PostgreSQL database management system. The system architecture follows the MVC pattern.

This system allows optimizing the management of financial resources, planning and controlling vessel maintenance and repairs, effectively monitoring fuel and other material expenses, as well as replacing vessel crew members.

One of the key elements of the information system is the automation of financial accounting, which allows tracking the company's financial transactions and provides quick analysis of financial performance.

Furthermore, the financial and technical management information system collects and analyzes data on the technical condition of vessels, enabling timely and efficient planning and execution of repair works.

The application of the financial and technical management information system helps shipowning companies reduce costs associated with management and vessel maintenance, improve the efficiency of financial resource management, and ensure uninterrupted vessel operations.

ЗМІСТ

ВСТУП

Морські транспортні перевезення є необхідною складовою світової економіки, і компанії-власники суден мають нести відповідальність за ефективне управління транспортними процесами та оптимізацію витрат.

У сучасних умовах автоматизація та комп’ютеризація є ключовими факторами в успішному управлінні бізнесом. Застосування спеціальних програм та сервісів може значно полегшити процес управління, зменшити ризики та помилки, а також забезпечити високу якість обслуговування.

Враховуючі великі об’єми даних та застосування, наприклад, звичайних поштових програм судновласницькі компанії не можуть забезпечити швидкий, надійний обмін даними та належне структурування інформації, тим самим організувати оптимальну роботу. Виникає логічна потреба у створенні програмного сервісу, що полегшить роботу та вирішіть вищезазначені проблеми, тому робота є актуальна на даний час.

Створення програмного сервісу у вигляді веб-додатку для автоматизації бізнес-процесів є актуальною задачею та може бути важливим кроком для компаній-власників суден. Такий сервіс допоможе у керуванні різними процесами: від відслідковування маршрутів до управління фінансовими показниками та технічним станом судна.

Метою проекту є створення інформаційної системи для ведення облікових записів та фінансово-технічного управління судновласницької компанії у вигляді веб-додатку із зручним інтерфейсом для капітанів суден, службовців технічного відділу, службовців крюїнгового відділу, службовців фінансового відділу та менеджерів (операторів суден).

Система допомагає вирішувати користувачам наступні задачі:

а) капітани суден можуть переглядати всі запити й вимоги компанії та відповідати на них. Вони можуть писати щоденний звіт оператору судна, сповіщати крюїнговий відділ щодо незапланованої заміни членів екіпажу судна, інформувати технічний відділ про необхідність заміни певних деталей

та обладнання, поповнення витратних матеріалів, бункерування (заправка суден паливом) тощо. Також сповіщати оператора судна про різноманітні фінансові потоки, такі як: портові збори, штрафи, витрати на провізію, премії членам екіпажу, поповнення суднових коштів та інше;

б) технічний відділ здійснює моніторинг технічного стану суден. Вони переглядають та відповідають на позиви капітанів щодо технічних проблем суден та поповнення витратних матеріалів. Також проводять контроль запланованих технічних обслуговувань судна. Технічний відділ виставляє рахунки фінансовому відділу щодо витрат на деталі обладнання та витратні матеріали;

в) крюїнговой відділ відповідає за заміну членів екіпажів суден відповідно до закінчення їх терміну контракту та нараховують зарплатню членам екіпажів. Також вони відповідають на прохання капітанів, щодо незапланованої заміни певних членів екіпажу. Крюїнговий відділ виставляє рахунки фінансовому відділу щодо виплат зарплат членам екіпажів та все, що стосується членів екіпажу;

г) фінансовий відділ здійснює контроль та рух фінансових потоків компанії. Вони відповідають та взаємодіють із технічним відділом, крюїнговим відділом та менеджерами (операторами суден) для отримання рахунків на оплату та здійснення переказів кошт компанії згідно цих рахунків. Фінансовий відділ веде облік загальних витрат та проводить порівняльний аналіз;

д) менеджери (оператори суден) визначають рейсові завдання для кожного капітану судна, надають контакти та вимоги портів капітанам, у які планується прибуття. Переглядають та відповідають на запроси від капітанів суден. Виставляють рахунки фінансовому відділу щодо витрат капітанів, які передчасно узгоджують із капітанами безпосередньо;

е) адміністратори системи добавляють, видаляють та редагують користувачів системи.

Для досягнення зазначеної мети необхідно вирішити наступні задачі:

а) провести аналіз області «фінансово-технічне управління судновласницької компанії» та сформулювати задачі користувачів;

б) обрати архітектуру та шаблон проектування для розробки інформаційної системи;

в) спроектувати БД для інформаційної системи;

г) обрати технології та засоби для реалізації інформаційної системи;

д) розробити БД та клієнтську програму, яка наддасть можливість різним категоріям користувачів ефективно маніпулювати даними предметної області відповідно до їхніх повноважень;

е) забезпечити захист системи від несанкціонованого доступу та розмежування повноважень між різними категоріями користувачів;

ж) протестувати програмну систему.

1 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ

1.1 Опис предметної області

Історія морських транспортних перевезень сягає віддалених часів. Людство завжди використовувало море для переміщення людей та вантажів. Вже в давніх цивілізаціях, таких як фінікійці, греки та римляни, морська торгівля відігравала важливу роль у розвитку економіки та обміну культурними цінностями.

У середньовіччі, з появою великих кораблів, зокрема каравел, морські транспортні перевезення стали ще популярнішими. Відкриття нових морських шляхів, таких як шлях до Індії Васко да Гами, розширили торгівельні маршрути та змінили картину світової торгівлі. Із часом, з появою парових суден та суден із поршневими двигунами внутрішнього згорання, морські перевезення стали ще більш швидкими, ефективними та складними.

На сьогоднішній день морські транспортні перевезення відіграють важливу роль у світовій економіці. Вони забезпечують міжнародну торгівлю, дозволяючи перевозити великі обсяги товарів між країнами та континентами. Морські перевезення є найбільш ефективним та економічним способом перевезення великих вантажів, зокрема сировини, машин та інших продуктів.

Крім того, морські перевезення забезпечують інтеграцію світових ринків, стимулюють економічний розвиток та сприяють розширенню глобальної торгівлі. Вони є необхідною складовою ланки логістичного ланцюжка, забезпечуючи доставку товарів в порти та звідти до кінцевих пунктів. [1]

У сучасному управлінні компаніями, що надають послуги морських перевезень, спостерігаються наступні тенденції та вимоги:

а) обробка великих обсягів інформації. У галузі морських перевезень виникає потреба в обробці та керуванні великим обсягом різноманітної інформації. Це включає дані про рух суден, розклади, логістику, фінансову

інформацію, дані про екіпаж та багато іншого. Управління цими обсягами даних стає складною задачею, яка вимагає ефективних систем для збору, обробки, зберігання та аналізу інформації;

б) покращення комунікації та збільшення швидкості обміну інформацією. Завдяки швидкому розвитку технологій інформаційного сектора, морські компанії мають доступ до потужних інструментів для ефективного управління та взаємодії всередині компанії. Це включає використання хмарних технологій для зберігання та обміну даними, систем управління віддаленим доступом, програмного забезпечення для автоматизації бізнес-процесів та аналітичних інструментів;

в) надійний захист даних. На сьогоднішній день, коли автоматизація та комп'ютеризація проникають у всі сфери життя, включаючи морські транспортні перевезення, існує ризик зловживання недосвідченими користувачами, які ще не добре ознайомлені з цифровою безпекою, і можуть розкрити конфіденційні дані компанії. Наприклад, використання фішингових електронних листів, що відправляються через звичайні поштові клієнти, є одним із найпростіших видів атак та водночас однією з найбільших загроз для компанії. Ці листи маскуються під звичайні робочі повідомлення від співробітників або партнерів компанії і надсилаються з метою збору інформації, яка може використовуватися у злочинних цілях. Під такі атаки можуть потрапляти як капітани суден, так й інші співробітники компанії. [2]

Отже, постає важливе питання у надані прикладного програмного забезпечення, що буде задовольняти вищенаведені вимоги та покращить введення бізнес процесів у цій галузі. Таким чином створення такої програми є доволі актуальним завданням на даний момент.

1.2 Аналіз існуючих конкурентів

Морські компанії та кораблі використовують різні методи зв'язку для обміну інформацією та підтримки зв'язку один з одним.

Одним із найпоширеніших методів є радіозв'язок. Кораблі використовують радіозв'язок для спілкування між собою та для зв'язку з береговими станціями. Зазвичай кораблі використовують радіостанції, які працюють на певних діапазонах частот для обміну інформацією.

Крім радіозв'язку, морські компанії та кораблі можуть використовувати засоби супутникового зв'язку, такі як системи Inmarsat або VSAT. Ці системи дозволяють кораблям підтримувати зв'язок з береговими станціями та іншими кораблями незалежно від їхнього розташування на океані.

Також існують спеціальні програми, які використовуються для обміну інформацією та керування кораблями. Наприклад, програми для керування дорожніми листами, вантажами, розкладами й так далі.

Розглянемо найбільш поширені програми, що надають можливість взаємодії або управління з командою кораблями, котрі знаходяться у морі.

AMOS (Advanced Maintenance and Operations System) є однією з найпопулярніших програмних систем для управління технічним обслуговуванням в морській індустрії від компанії SpecTec.

AMOS дозволяє ефективно управляти обслуговуванням та ремонтом кораблів, автоматизуючи та оптимізуючи процеси планування, замовлення запчастин, виконання ремонтних робіт та обліку витрат. Ця система надає комплексні функції для управління запасами, контролю якості, обміну даними, планування ресурсів та багато іншого.

AMOS використовується відомими морськими операторами, власниками та менеджерами флоту для підтримки ефективного управління технічним обслуговуванням кораблів та морського обладнання. Вона дозволяє автоматизувати та спростити процеси планування ремонтних робіт, управління запасами, відстеження вартості та аналізу ефективності ремонтних дій.

Крім того, AMOS надає можливість зберігати велику кількість даних про обладнання та ремонтні роботи, що дозволяє підвищити безпеку, планувати

профілактичні ремонти та забезпечувати відповідність з вимогами міжнародних стандартів.

Узагальнюючи, AMOS є потужним інструментом для автоматизації та ефективного управління технічним обслуговуванням та ремонтом кораблів, сприяючи збільшенню ефективності та надійності морського флоту. [3]

ShipManager – це комп'ютерна програма, розроблена компанією DNV GL, яка надає рішення для керування дорожніми листами та вантажоперевезеннями в морській індустрії. Цей додаток допомагає операторам суден та власникам флоту ефективно планувати, виконувати та відстежувати процеси вантажоперевезень.

ShipManager забезпечує контроль над дорожніми листами, що включає планування та координацію вантажоперевезень, оптимізацію маршрутів, відстеження вантажу та контроль витрат. Це дозволяє знизити час і зусилля, пов'язані з організацією та виконанням вантажоперевезень, і забезпечує оптимальне використання ресурсів.

За допомогою ShipManager можна автоматизувати процеси відстежування вантажу та документообігу, забезпечуючи швидкий та точний обмін інформацією між всіма зацікавленими сторонами, такими як оператори суден, порти, термінали та клієнти. Це сприяє покращенню комунікації, зниженню ризиків помилок та підвищенню ефективності вантажоперевезень.

Узагальнюючи, ShipManager є додатком, який спрощує та автоматизує процеси управління дорожніми листами та вантажем. [4]

Navis N4 – це інтегрована система управління портовими операціями та судновими стоянками, розроблена компанією Navis. Цей додаток дозволяє портовим операторам ефективно керувати всіма аспектами портових операцій, включаючи суднові стоянки, рух суден, завантаження та розвантаження вантажів та координацію роботи всіх зацікавлених сторін.

Navis N4 забезпечує централізовану систему управління, яка дозволяє в реальному часі контролювати всі етапи портових операцій. Вона включає такі

функції, як планування робочих місць, призначення ресурсів, управління потоками суден, моніторинг вантажів, обробка документів та звітність.

За допомогою Navis N4 портові оператори можуть оптимізувати використання ресурсів, мінімізувати час очікування суден та вантажів, забезпечити оптимальний рух транспортних засобів у порту та підвищити продуктивність операцій. Вона також дозволяє автоматизувати інформаційний обмін між різними сторонами, такими як портові влади, судновласники, лінії перевезень та логістичні партнери, сприяючи покращенню комунікації та співпраці.

Navis N4 забезпечує також підтримку безпеки та контролю над вантажами, зокрема відстеження інвентаризації, перевірку заходів безпеки та виконання регуляторних вимог. Вона допомагає забезпечити точність, якість та надійність портових операцій.

Узагальнюючи, Navis N4 є інструментом для управління портовими операціями та судновими стоянками.[5]

Зробимо порівняльний аналіз вищезазначених програмних додатків та виділимо їх основні переваги та недоліки (табл. 1.1).

Таблиця 1.1 – Результати порівняння програмних додатків

Переваги	Недоліки
AMOS	
- підтримує ефективне управління обслуговуванням та ремонтом кораблів; - забезпечує контроль та оптимізацію процесів запасів, контролю якості та обліку витрат	- обмежена функціональність в порівнянні з іншими системами управління; - може вимагати тривалого часу та ресурсів для впровадження та навчання персоналу
ShipManager	
- забезпечує зберігання великої кількості даних про обладнання та ремонтні роботи; - автоматизує та оптимізує процеси планування, виконання та відстеження вантажоперевезень; - забезпечує ефективний обмін інформацією між всіма зацікавленими сторонами	- вимагає підтримки технічної інфраструктури для безперебійної роботи; - обмежена спеціалізація на керуванні дорожніми листами вантажоперевезеннями; - може потребувати інтеграції з іншими системами для повноцінного функціонування

Продовження таблиці 1.1

Переваги	Недоліки
Navis N4	
- надає аналітичні можливості та звітність для прийняття обґрунтованих рішень; - забезпечує централізоване управління портовими операціями та судновими стоянками	- вимагає навчання та ознайомлення персоналу з системою; - може бути дещо складним у використанні для непідготовленого персоналу

За результатами аналізу можна зробити висновок про те, що на даний момент існують потужні інформаційні системи, що надають можливість здійснення комунікації та ефективного обміну інформації, що безпосередньо покращує та прискорює роботу клієнта. Основною проблемою в таких сервісах, на першому плані – не зручний або не зрозумілий інтерфейс, що ускладнює застосування додатку та збільшує час навчання нових працівників. Також відсутність інструкцій до використання та в деяких випадках необхідність інтеграції сторонніх програмних продуктів для повноцінної роботи.

1.3 Постановка мети та задачі проекту

Мета даного проекту полягає в розробці веб-додатку, який буде використовуватися судновласницькими компаніями для ефективного ведення облікових записів та фінансово-технічного управління.

Цей додаток буде мати зручний інтерфейс, що дозволить різним користувачам, таким як капітани суден, співробітники технічного відділу, крюїнгового відділу, фінансового відділу та менеджери (оператори суден), здійснювати свої обов'язки та завдання в зручний спосіб.

Ця інформаційна система буде забезпечувати зберігання та обробку даних про фінансовий стан компанії, технічний стан суден, керування екіпажем та управління робочими процесами. Капітани суден матимуть можливість вносити дані про рейси, витрати на пальне, обслуговування та інші

фактори, пов'язані з експлуатацією судна. Службовці технічного відділу зможуть вести облік і планування технічних робіт, ремонтів та запасних частин. Крюїнговий відділ зможе керувати найманням та розстановкою екіпажу на суднах. Фінансовий відділ отримає інструменти для ведення обліку фінансових операцій, розрахунків з партнерами та заробітної плати екіпажу. Менеджери (оператори суден) матимуть можливість відстежувати та координувати роботу суден, планувати рейси та контролювати їх виконання.

Ця інформаційна система буде сприяти автоматизації та оптимізації робочих процесів у судновласницькій компанії, забезпечуючи більш точний облік та аналіз даних, полегшуючи комунікацію між різними відділами компанії та забезпечуючи доступ до актуальної інформації для прийняття рішень.

Створення такої інформаційної системи є актуальним завданням, оскільки вона дозволить поліпшити ефективність управління, знизити витрати, покращити координацію та співпрацю між різними відділами. Крім того, це забезпечить компанії конкурентну перевагу на ринку, сприятиме зростанню та розвитку її бізнесу.

2 СПЕЦИФІКАЦІЯ ВИМОГ ДО ІНФОРМАЦІЙНОЇ СИСТЕМИ

2.1 Загальний погляд на вимоги до системи

Після аналізу предметної області дозволяє визначити наступні поняття (глосарій) у інформаційній системі.

Глосарій технічної сфери надає визначення та пояснення спеціалізованих термінів, які можуть бути незрозумілими для людей, не знайомих з цією галуззю. Він допомагає стандартизувати термінологію та забезпечує однозначне розуміння між фахівцями та учасниками проекту, якщо вони було незрозумілі.

Тому створимо глосарій нашої інформаційній системи:

а) капітан судна – це людина, яка виконує функції керівника цілого судна і відповідає за усіх членів команди та за судно у цілому;

б) крюїнговий відділ – це відділ, котрий відповідає за штат співробітників судна (членів екіпажу);

в) технічний відділ – це відділ, котрий відповідає за технічний стан судна та його ремонт;

г) оператор судна – це відділ, котрий відповідає за щоденний зв’язок з капітаном судна та будь-які інші речі, що турбує капітана;

д) звіт – це письмовий документ, який представляє інформацію, результати аналізу, огляду або дослідження про певну ситуацію, подію;

е) повідомлення – це короткий текст або комунікативний засіб, який використовується для передачі інформації між людьми, у нашому випадку буде зв’язок між капітаном судна та відділами;

ж) адміністратор – це людина, яка керує внутрішніми функціями інформаційної системи як реєстрація користувачів, видалення, редагування;

и) база даних – сукупність даних, організованих за певними правилами, що дозволяють їх зберігання, модифікацію та отримання.

2.2 Функціональні вимоги

ІС передбачено шість типів користувачів:

а) капітани суден;

б) технічний відділ;

в) крюінговий відділ;

г) фінансовий відділ;

д) менеджери (оператори суден);

е) адміністратори.

У таблиці 2.1. наведено перелік основних задач для кожного з користувачів згідно опису предметної області із зазначенням вхідної та вихідної інформації.

Таблиця 2.1 – Задачі користувачів ІС «Фінансово-технічного управління для судновласницьких компаній»

Задача	Вхідна інформація	Вихідна інформація
Капітан судна		
Ка.1 Переглядання запитів від відділів		Інформація звіту, як текст звіту, коментар, тип звіту, відділ
Ка.2 Переглядання підтверджених звітів		Інформація звіту, як текст звіту, коментар, тип звіту, відділ
Ка.3 Переглядання відхилених звітів		Інформація звіту, як текст звіту, коментар, тип звіту, відділ
Ка.4 Відповідь на запит оператора	Підтвердження/скасування запиту Коментар	Відповідь на запит
Ка.5 Написання щоденного звіту оператору	Текст звіту Оператор судна Тип звіту	Відправлений звіт
Ка.6 Запит на заміну членів екіпажу судна	Текст звіту з ФІО члена екіпажу Крюїнговий відділ Тип запиту заміну члена екіпажу	Відправлений запит

Продовження таблиці 2.1

Задача	Вхідна інформація	Вихідна інформація
Ка.7 Запит на деталь або/та обладнання	Текст звіту, що включає в себе деталь, котру потрібно замінити Технічний відділ Тип звіту на деталь	Відправлений запит
Ка.8 Запит на бункерування (постачання палива)	Текст звіту, що включає в себе запит на бункерування Технічний відділ Тип звіту на бункерування	Відправлений запит
Ка.9 Звіт грошових витрат	Тип витрат Сума витрат Дата витрат Коментар Оператор судна	Відправлений звіт
Ка.10 Написання відгуків/характеристики стосовно члену екіпажу	Текст звіту з ФІО члена екіпажу Крюїнговий відділ Тип звіту на відгук	Відправлений відгук
Ка.11 Узгодження рейсового завдання із менеджером	Текст звіту Оператор судна Тип звіту на узгодження	Відправлене повідомлення
Ка.12 Переглянути повідомлення від оператору		Діалог з повідомленнями
Ка.13 Переглянути повідомлення від технічного відділу		Діалог з повідомленнями
Ка.14 Переглянути повідомлення від крюїнгового відділу		Діалог з повідомленнями
Ка.15 Переглянути повідомлення від фінансового відділу		Діалог з повідомленнями
Ка.16 Написати повідомлення оператору	Текст повідомлення	
Ка.17 Написати повідомлення технічному відділу	Текст повідомлення	
Ка.18 Написати повідомлення крюїнговому відділу	Текст повідомлення	
Ка.16 Написати повідомлення фін відділу	Текст повідомлення	

Продовження таблиці 2.1

Задача	Вхідна інформація	Вихідна інформація
Технічний відділ		
Т.1 Перевірка технічного стану судна	Текст звіту, що включає дату запланованої перевірки та ім’я суперінтенданта Назва судна	Відправлене повідомлення
Т.2 Переглядання запитів від капітанів суден		Інформація звіту, як текст звіту, коментар, тип звіту, назва судна
Т.3 Переглядання підтверджених запитів		Інформація звіту, як текст звіту, коментар, тип звіту, назва судна
Т.4 Переглядання відхилених запитів		Інформація звіту, як текст звіту, коментар, тип звіту, назва судна
Т.5 Відповідь на запит капітану судна	Підтвердження/скасування запиту Коментар	Відповідь на запит
Т.6 Надсилання фінансового звіту до фінансового відділу	Тип витрат Сума витрат Текст звіту Фінансовий відділ	Відправлений звіт
Т.7 Переглянути повідомлення від капітана		Діалог з повідомленнями
Т.8 Написати повідомлення капітану	Текст повідомлення	
Крюїнговий відділ		
Кр.1 Надсилання запиту до капітану щодо зміни членів екіпажу	ФІО членів екіпажу Дата планування заміни Назва судна	Відправлений запит
Кр.2 Переглядання запитів від капітанів щодо заміни члена екіпажу		Запит капітану судна
Кр.3 Переглядання підтверджених запитів		Інформація звіту, як текст звіту, коментар, тип звіту, назва судна
Кр.4 Переглядання відхилених запитів		Інформація звіту, як текст звіту, коментар, тип звіту, назва судна
Кр.5 Відповідь на запит капітану судна	Підтвердження/скасування запиту Коментар	Відповідь на запит

Продовження таблиці 2.1

Задача	Вхідна інформація	Вихідна інформація
Кр.6 Надсилання звіту фінансовому відділу щодо зарплатні членів екіпажу	Текс звіту, що включає ФІО члена екіпажу Сума зарплатні Фінансовий відділ	Відправлений звіт
Кр.7 Переглянути повідомлення від капітана		Діалог з повідомленнями
Кр.8 Написати повідомлення капітану	Текст повідомлення	
Фінансовий відділ		
Ф.1 Переглядання фінансових звітів від технічного відділу		Інформація звіту від технічного відділу, як текст звіту, тип звіту, сума витрат
Ф.3 Переглядання фінансових звітів від крюїнгового відділу		Інформація звіту від крюїнгового віддіду, як текст звіту, тип звіту, сума витрат
Ф.4 Переглядання фінансових звітів від менеджерів суден		Інформація звіту від менеджера суден, як текст звіту, тип звіту, сума витрат
Ф.5 Аналіз витрат за фільтром усіх суден	Проміжок дат (початок дати та кінець дати)	Сумарні витрати всіх суден
Ф.6 Аналіз кількості витрат усіх суден	Проміжок дат (початок дати та кінець дати) Назва судна	Сумарні витрати конкретного судна
Менеджер (оператор судна)		
М.1 Надіслання рейсового завдання капітану судна	Текст звіту, що включає завдання капітану Тип звіту Назва судна	Відправлене повідомлення
М.2 Надання контактів та вимог портів	Текст звіту, що включає назву порту Тип звіту Назва судна	Відправлене повідомлення
М.3 Переглядання запитів від капітанів		Інформація звіту від капітану, як текст звіту, тип звіту, сума витрат або комент
М.4 Переглядання підтверджених запитів		Інформація звіту, як текст звіту, тип звіту, сума витрат або комент

Продовження таблиці 2.1

Задача	Вхідна інформація	Вихідна інформація
М.5 Переглядання відхилених запитів		Інформація звіту , як текст звіту, тип звіту, сума витрат або комент
М.6 Відповідь на запит капітана	Підтвердження/скасування запиту Коментар	Відповідь на запит
М.7 Надіслання звітів щодо незапланованих витрат капітанів	Сума витрат Коментар Фінансовий відділ Тип звіту	Відправлений звіт
М.8 Переглянути повідомлення від капітана		Діалог з повідомленнями
М.9 Написати повідомлення капітану	Текст повідомлення	
Адміністратор		
А.1 Створення нового користувача	Ім’я Прізвище Адреса Дата народження Телефон Пароль Роль в системі Ранг членів судна	Новий користувач
А.2 Видалення користувача	Ідентифікатор користувача	Користувач помічений як видалений
А.3 Редагування користувача	Ідентифікатор Ім’я Прізвище Адреса Дата народження Телефон Пароль	Оновлені дані по користувачу

2.3 Вимоги користувачів

Для кращого ілюстрування програмних вимог була створена діаграма варіантів використання. Діаграма описує роботу з додатком усіх користувачів та можливості системи та зовнішніх сервісів.

Найбільш детально проведено пошук акторів та випадки використання, і вони представлені в наступних таблицях (табл. 2.2).

Таблиця 2.2 – Пошук та аналіз акторів

Актор	Короткий опис
Адміністратор	Адміністратор – це особа, що здійснює контроль та адміністрування усієї інформаційної системи. До його обов'язків входить робота з користувачами, а саме реєстрація, оновлення та видалення з системи
Капітан	Капітан – це керівник на судні і саме він є відповідальний за обмін звітами та повідомленнями між відділами та судном у додатку. Після обміну звітами або повідомленнями він буде виконувати певні дії та керувати своїми підопічними
Менеджер (оператор судна)	Відділ, що повинен тримати контакт з капітаном судна та при будь-яких незвичайних ситуаціях рекомендувати капітану ті чи інші дії. Так цей відділ отримує кожен день звіт від капітана, щоб бути у курсі справ судна
Технічний відділ	Відділ, обов'язки якого – це слідкування за технічним станом судна та усі інші технічні можливості як заправка судна або заміна обладнання
Крюїнговий відділ	Відділ, обов'язки якого – це люди,а саме персонал на судні, що виконує роботу як технічну так і іншу. Відділ також відповідає за заміну членів екіпажу або відправка звітів для заробітної плати члені судна
Фінансовий відділ	Відділ, обов'язки якого – отримувати звіти від усіх інших звітів та підраховувати загальні витрати у інформаційній системі (можливо додати різноманітну аналітику)

Створимо діаграми варіантів використання для користувачів системи, щоб розуміти у візуальному плані, які прецеденти виконують вони у інформаційній системі. Загальний вигляд доступного функціоналу користувача вказано у вигляді Unified Modeling Language (UML) діаграми прецедентів на рисунках нижче.

Переглянемо діаграму варіантів використання для користувача адміністратор див. рис. 2.1.

Рисунок 2.1 – Діаграма варіантів використання для адміністратора

Переглянемо діаграму варіантів використання для користувача капітан див. рис. 2.2 та 2.3.

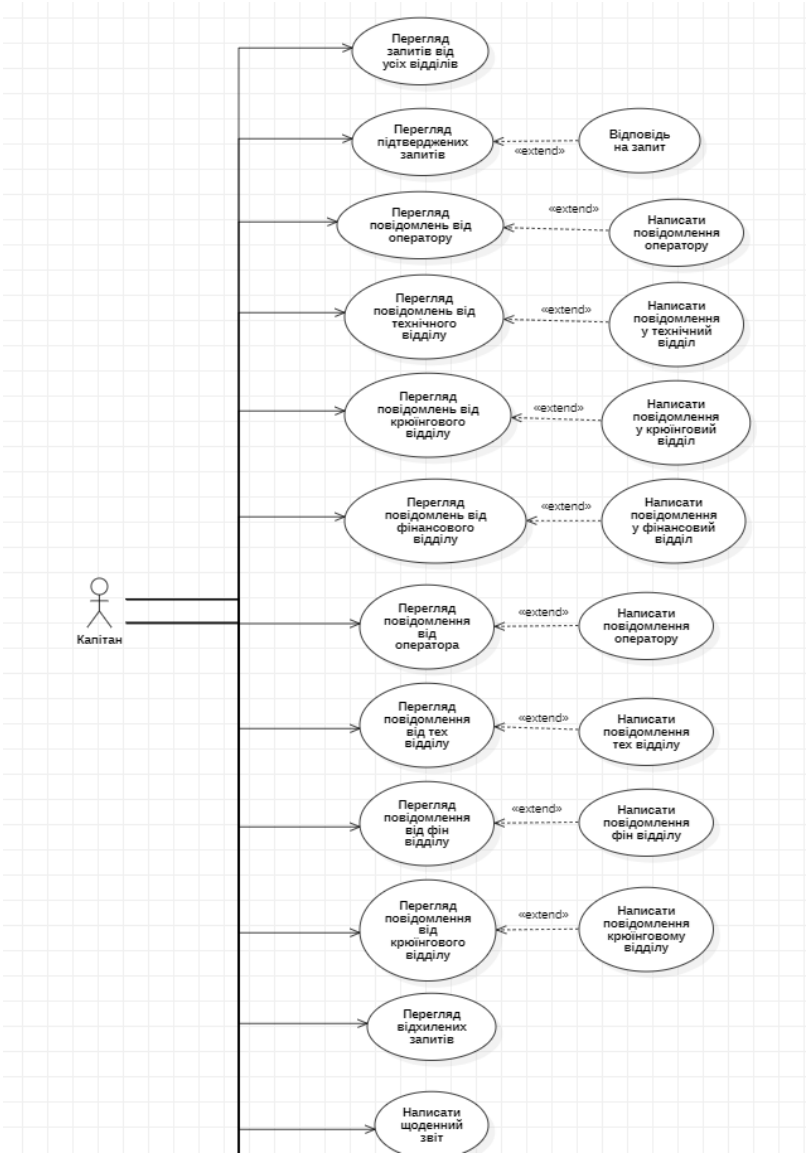

Рисунок 2.2 – Діаграма варіантів використання для капітана, лист 1

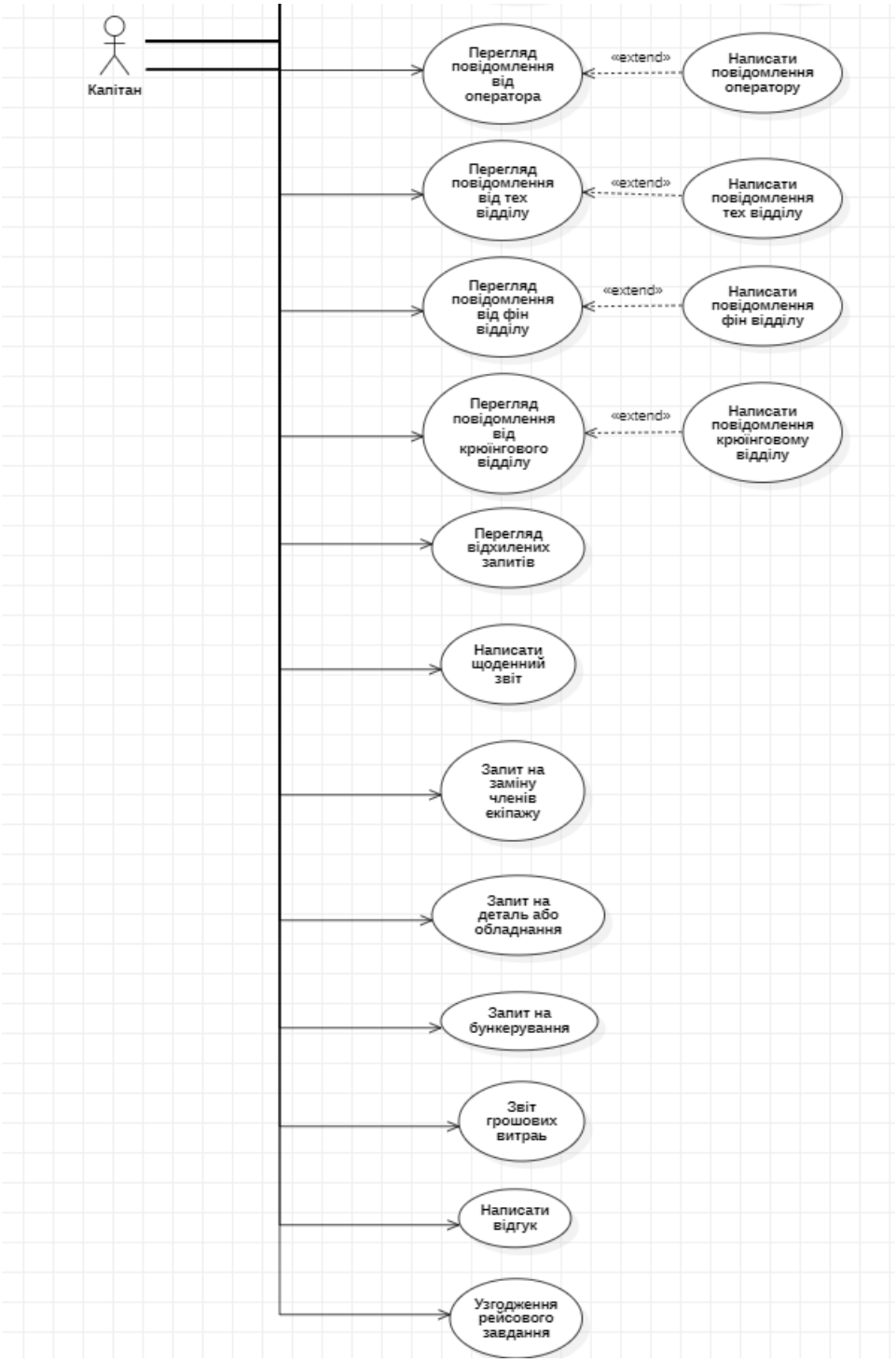

Рисунок 2.3 – Діаграма варіантів використання для капітана, лист 2

Більше діаграм варіантів використання можливо переглянути у Додатку Б.

2.4 Атрибути якості програми

Для зручного користування інформаційної системи користувачам потрібно слідувати певним атрибутам якості, таким як: зручність використання, безпека, швидкість роботи додатку та інше (табл. 2.3).

Таблиця 2.3 – Нефункціональні вимоги

Вимога	Підтримка системою
Інтерфейс програмного забезпечення	Кросплатформність – веб-додаток може працювати з усіма можливими веб-браузерами: Google Chrome, Internet Explorer та т. д.
Апаратні вимоги	Для взаємодії з системою потрібні лише самі мінімальні налаштування, для підтримки браузера
Програмні вимоги	Використання веб-додатку можливо на: Windows, Linux, Mac
Завантаження сайту	Швидкість завантаження сайту становить секунду часу
Дружелюбність інтерфейсу	На веб-сайті повинен бути інтуїтивно зрозумілий інтерфейс для використання користувачем
Юзабіліті сайту	Усю потрібно інформацію можливо знайти у меню сайту
Безпека на сайті	Додаток не збирає інформації, уся інформація зберігається у базі даних

2.5 Модель користувацького інтерфейсу

У користувачів інформаційної системи може бути схожий користувацький інтерфейс, проте різний функціонал, тому потрібно змоделювати користувацький інтерфейс для усіх користувачів інформаційної системи з деякими виправленнями під кожного користувача. Модель користувацького інтерфейсу побудована у вигляді діаграми ієрархії веб-сторінок [6], що відображає послідовність переходів по сторінках додатку.

Переглянемо користувацький інтерфейс адміністратора на (рис. 2.4).

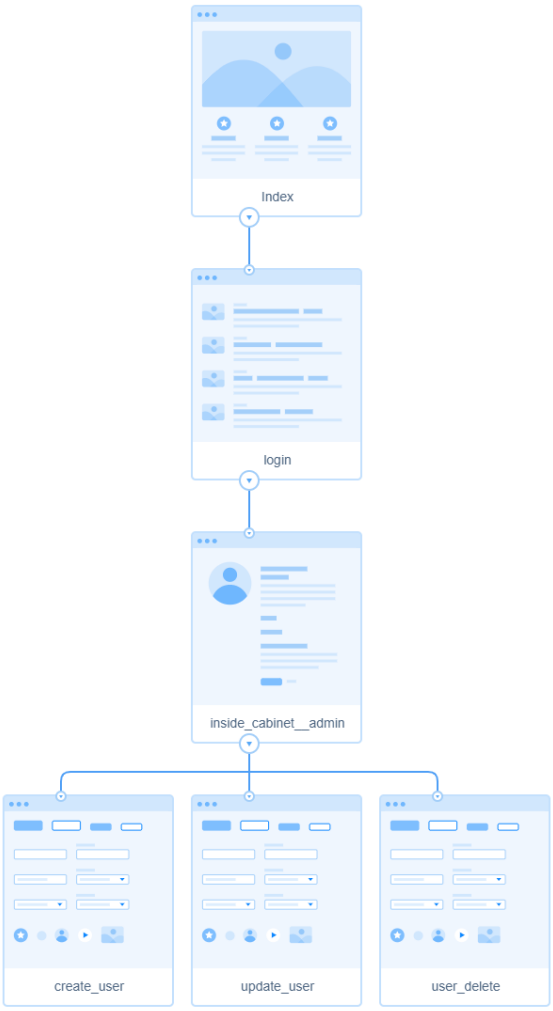

Рисунок 2.4 – Діаграма користувацького інтерфейсу для адміністратора

Більше діаграм ієрархії користувацького інтерфейсу можливо переглянути у додатку Б.

3 ПРОЕКТУВАННЯ ІНФОРМАЦІЙНОЇ СИСТЕМИ

3.1 Вибір шаблону проектування

Сучасні ІС зазвичай будуються відповідно до N-рівневої архітектури клієнт-сервер, яка дозволяє створювати гнучкі і повторно використовувані додатки. Найбільшого поширення набули дворівнева та трирівнева клієнт-серверні архітектури. В якості розподіленої архітектури інформаційної системи обрано трирівневу клієнт-серверну архітектуру (рис. 3.1).

Рисунок 3.1 – Трирівнева архітектура клієнт-сервер

Триланкова архітектура складається з серверу бази даних (рівень управління ресурсами), серверу додатків (рівень прикладного компоненту) та клієнтів (рівень представлення даних) Усе це можна також представити у вигляді діаграми розгортання див. рис. 3.2.

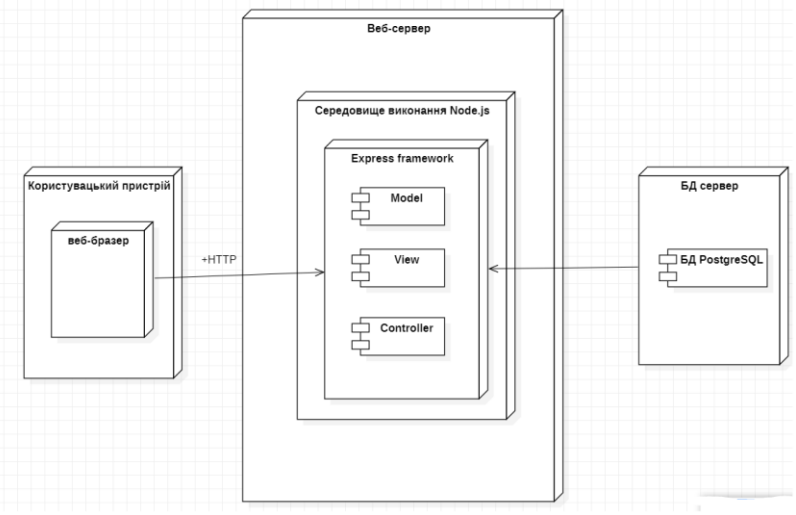

Рисунок 3.2 – Діаграма розгортування для архітектури

Такий підхід до архітектури системи значно мінімізує потік даних між клієнтською програмую та сервером застосунку, адже передається лише необхідний потік даних – аргументи функцій та результати їх виконання. А ізольованість рівня серверу додатку або серверу бази даних дозволить швидко допрацювати систему при виникненні збоїв, а також полегшити додавання нового функціоналу системи. Роботу системи можна описати наступним чином: клієнт посилає запит до серверу, який оброблює запит та повертає відповідь клієнту, які формуються як результати запитів до бази даних.

Для розробки складних інформаційних систем доцільно використовувати шаблони проектування, які дозволяють розділити систему на окремі компоненти та визначити коло їх відповідальності, що дозволяє ефективно вирішувати проблеми в проектуванні програмного забезпечення, припускають багаторазове використання коду, забезпечують надійність, дозволяють зменшити кількість помилок і прискорюють процес проектування. Для проектування логічної структури ІС як правило використовують шаблони трьох шарового архітектурного стилю: MVC (Model-View-Controller), MVP (Model-View-Presenter) та MVVM (Model-View-ViewModel). Варто зазначити, що останні два є подальшими більш спеціалізованими розвитками MVC. Так, наприклад MVP більш придатний для створень клієнтських додатків, коли потрібно реагувати на події інтерфейсу користувача, а MVVM використовується у фреймворках, які підтримують «зв’язування даних».

В якості шаблону проектування обрано MVC [7], тому що даний шаблон більш підходить для проектування ІС, де бізнес-логіка повністю реалізується на стороні серверу. MVC складається з Моделі, Представлення та Контролера:

– Модель здійснює маніпулювання даними додатка, а вже потім надає дані Представленню і реагує на команди Контролера шляхом зміни свого стану;

– Представлення (призначений для користувача інтерфейс) відповідає за отримання необхідних даних з Моделі і відображення їх користувачеві, а

також за отримання від користувача команд для роботи з даними і відправку цих команд Контролеру;

– Контролер, який відповідає за перетворення команд користувача, отриманих від Представлення, в набір дій над Моделлю з метою її зміни. Для забезпечення виклику необхідного Controller виконується маршрутизація – сайт має спільну точку входу для кожного введеного URL, де проводиться його аналіз та виклик відповідного контролеру.

Див. рис. 3.3. зображено взаємодія компонентів MVC.

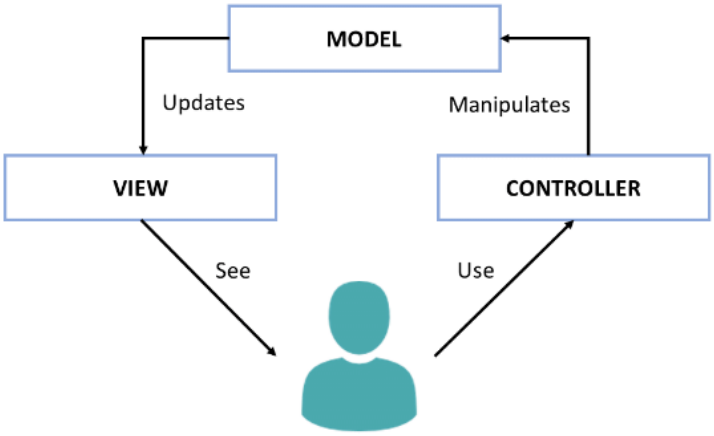

Рисунок 3.3 – Схема взаємодії компонентів MVC

Підсумовуючи вищенаведену інформацію, MVC є вдалим вибором через його гнучкий потенціал.

Після того як були обрані архітектура та шаблон проектування, спроектуємо діаграму компонентів [8] інформаційної системи, вона дозволить відобразити структуру та залежності (зв’язки) між компонентами програмного забезпечення (рис. 3.4).

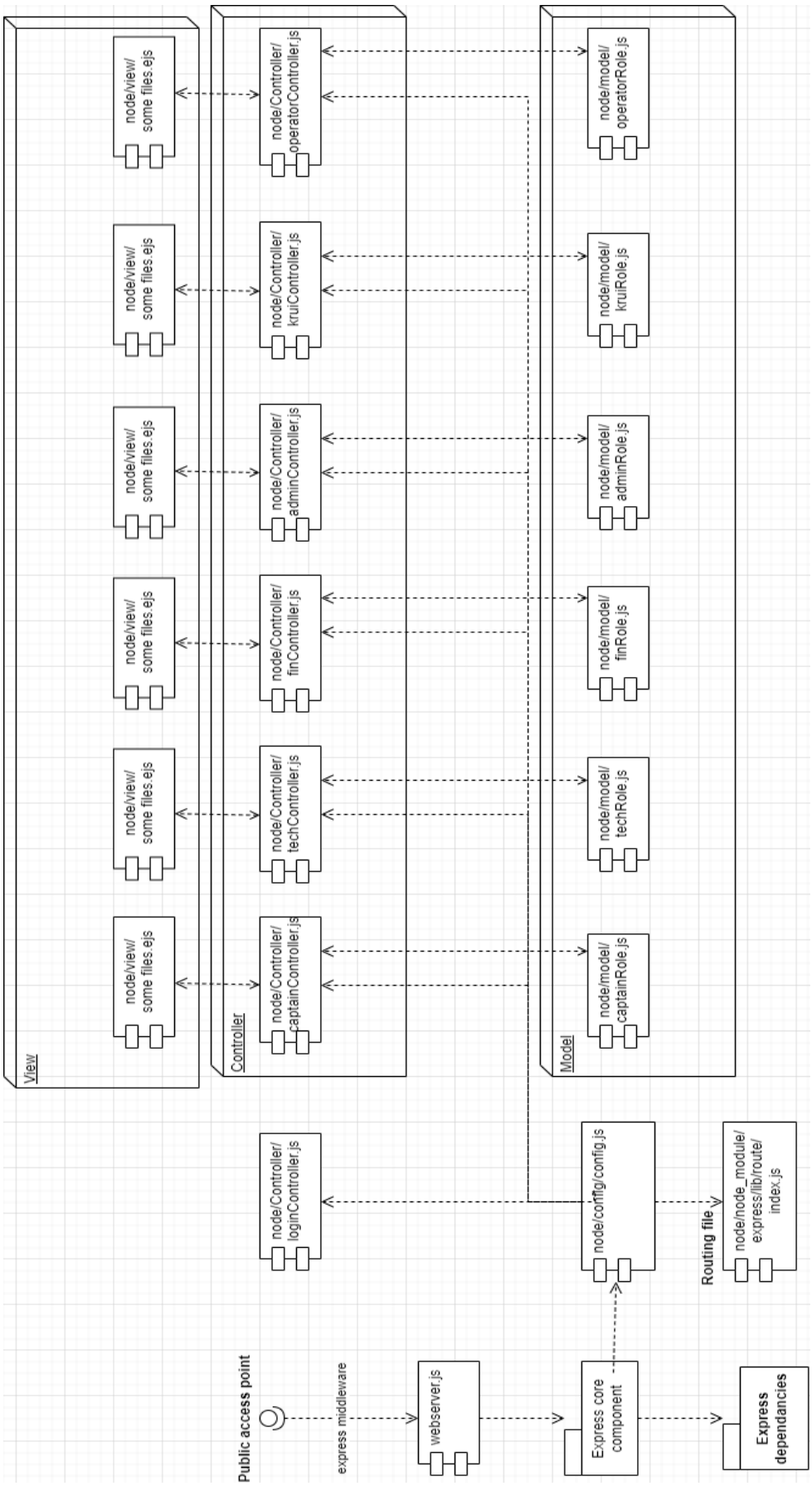

Рисунок 3.4 – Діаграма компонентів для інформаційної системи

3.2 Інформаційна модель предметної області

Сутності та їх властивості з описом обмежень, що потрібні для розв’язання поставлених задач, див. табл 3.1. В таблиця 3.1 використані наступні позначення обмежень на атрибути: первинний ключ таблиці PRIMARY KEY (PK), обов’язкове ненульове значення атрибуту NOT NULL (NN), зовнішній ключ FOREIGN KEY (FK), унікальний атрибут UNIQUE (U).

Таблиця 3.1 – Опис сутностей та їх властивостей

Властивість	Призначення атрибута	Обмеження
Об’єкт «role_type» («Роль»)		
id	Ідентифікатор ролі	PK
name	Назва ролі	NN, U
Об’єкт «employee» («Робітник»)		
id	Ідентифікатор робітника	PK
role_type_id	Ідентифікатор ролі	FK, NN
first_name	Ім’я	NN
last_name	Прізвище	NN
address	Адреса	NN
phone	Телефон	NN
password	Пароль	NN
data_of_birth	Дата народження	NN, CHECK >=18 або <= 60
Is_delete	Чи видалено?	NN, DEFAUL FALSE
Об’єкт «rank» («Ранг»)		
id	Ідентифікатор рану	PK
name	Назва рангу	U, NN
Об’єкт «sea_worker» («Робітник у морі»)		
id	Ідентифікатор робітника у морі	PK
rank_id	Ідентифікатор рангу	FK, NN
employee_id	Ідентифікатор робітника	FK, NN
Об’єкт «department» («Відділ»)		
id	Ідентифікатор відділу	PK
name	Назва відділу	NN
Об’єкт «office_worker» («працівник офісу»)		
id	Ідентифікатор працівника офісу	PK
department_id	Ідентифікатор відділу	FK, NN
employee_id	Ідентифікатор робітника	FK, NN
Об’єкт «ship» («судно»)		
id	Ідентифікатор судна	PK

Продовження таблиці 3.1

Властивість	Призначення атрибута	Обмеження
name	Назва корабля	NN
sea_worker_id	Ідентифікатор робітника	FK, NN
year_of_building	Рік будівництва	NN, >30 років зі старту
weight	Вага	NN
length	Довжина	NN
Об’єкт «team» («Команда»)		
sea_worker_id	Ідентифікатор робітника у морі	FK, PK
ship_id	Ідентифікатор судна	FK, PK
Об’єкт «type» («Тип»)		
id	Ідентифікатор типу	PK
name	Назва типу	U, NN
Об’єкт «message» («Повідомлення»)		
id	Ідентифікатор типу	PK
ship_id	Ідентифікатор судна	FK, NN
department_id	Ідентифікатор відділу	FK, NN
date_time	Дата та час	NN
message_text	Повідомлення	
how_send	Хто відправив	NN
Об’єкт «message» («Повідомлення»)		
id	Ідентифікатор типу	PK
ship_id	Ідентифікатор судна	FK, NN
department_id	Ідентифікатор відділу	FK, NN
type_id	Ідентифікатор типу	FK, NN
sum	Сума	
date_time	Дата та час	NN
message_text	Повідомлення	
isCheck	Прийнято або відхилено	
comment	Коментар	
how_send	Хто відправив	NN

Між сутностями наявні 2 типи зв’язків:

– «один-до-багатьох»;

– «багато-до-багатьох».

Зв’язок «один-до-багатьох» присутній між таблицями:

– role_type та employee;

– employee та sea_worker;

– employee та office_worker;

– rank та sea_worker;

– sea_worker та ship;

– department та office_worker;

– department та message;

– department та report;

– type та report.

Для формалізації наведених зв’язків, до N-зв’язаної таблиці додається первинний ключ однозв’язної таблиці у якості зовнішнього ключа. Наприклад, первинний ключ «id» таблиці «rank» додається до таблиці «sea_worker» – «rank_id».

Зв’язок «багато-до-багатьох» виникає між таблицями:

– sea_worker та ship умовний із обох кінців, для формалізації даного відношення створено допоміжну таблицю team;

Кожна допоміжна таблиця містить первині ключі двох таблиць між якими є зв’язок «багато-до-багатьох». Наприклад, атрибути «sea_worker_id», «ship_id» таблиці «team» є первинними ключами таблиць «sea_worker», та «ship» відповідно.

Одним із засобів графічного представлення предметної області є діаграма «сутність-зв’язок» (ER-діаграма, ERD). Для візуалізації ER-діаграм існує ряд нотацій, таких як нотація Чена, нотація Баркера, нотація «воронячі лапки» та інші. Для побудови ER-діаграми можна скористатися різними програмними засобами.. ER-діаграма для предметної області див. рис. 3.5.

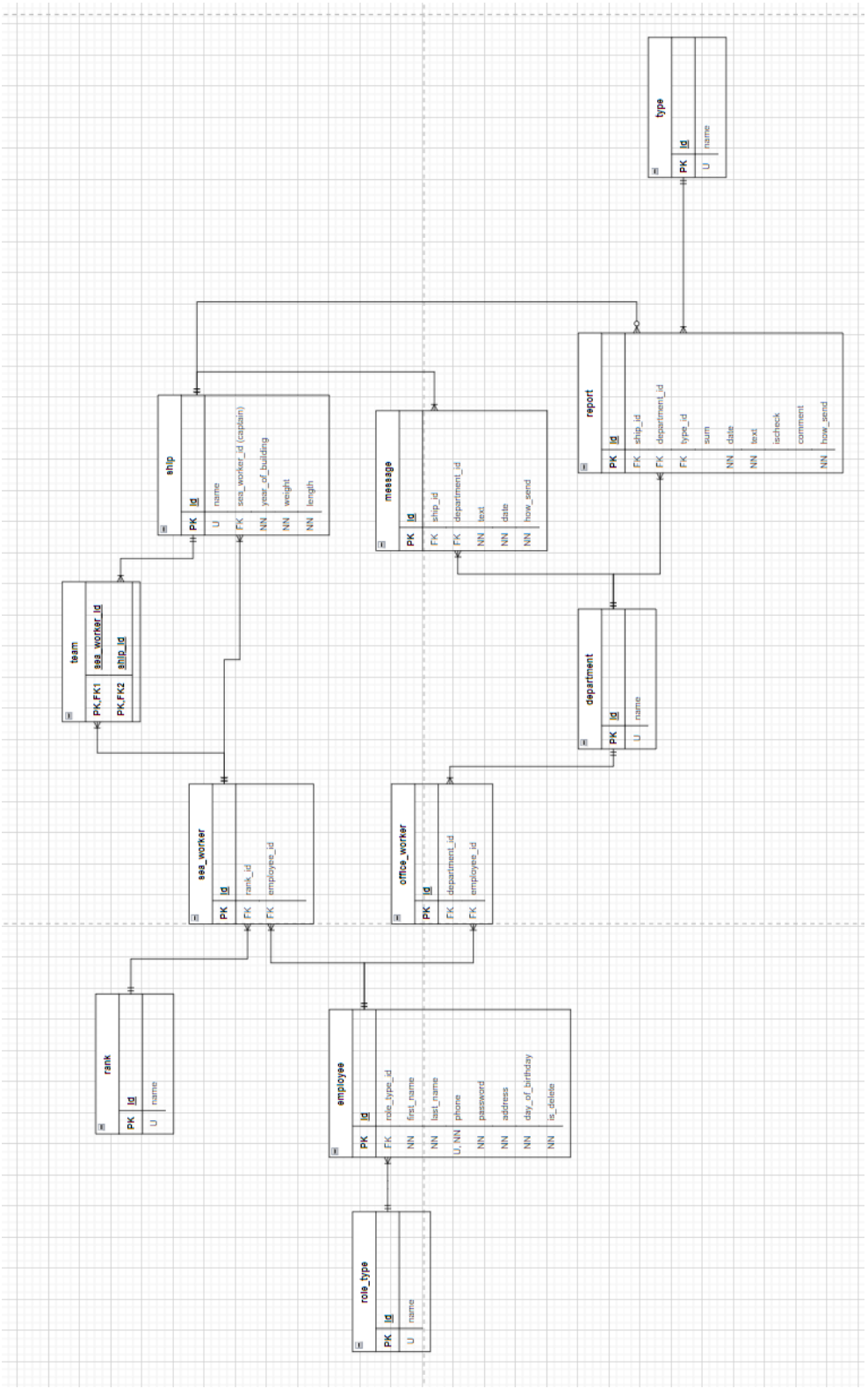

Рисунок 3.5 – Модель сутність зв’язок для предметної області

4 ВИБІР ЗАСОБІВ ДЛЯ РОЗРОБКИ

Архітектурним шаблоном, який буде використовуватися для впровадження системи, було обрано MVC, тепер необхідно визначити мову програмування, фреймворк та інструменти, які дозволять виконати завдання у повній мірі.

Як засоби реалізації інформаційної системи для предметної області були обрані наступні інструменти:

а) СУБД PostgreSQL;

б) мова програмування JavaScript;

в) середовище розробки Visual Studio Code;

г) серверна частина додатку створювалась за допомогою фреймворку Express використовуючи сервер додатку Node.js та менеджер пакетів nmp;

е) інтерфейс додатку створювався за допомогою бібліотеки Bootstrap.

4.1 PostgreSQL

PostgreSQL – це потужна система з об’єктно-реляційною базою даних з відкритим кодом, яка використовує та розширює мову SQL у поєднанні з багатьма функціями, які дозволяють безпечно зберігати та масштабувати найскладніші навантаження на дані [9]. Перевагами PostgreSQL є надійність, цілісність даних, розширюваність, працює на всіх основних операційних системах, допомагає управляти даними, незалежно від того, наскільки великий чи малий набір даних, є сумісним з ACID (Atomicity, Consistency, Isolation, Durability). Наприклад, можна визначити власні типи даних, створювати власні функції, навіть писати код з різних мов програмування без перекомпіляції бази даних.

До основних можливостей (переваг) входять:

а) open source (відкриті джерела для використання безкоштовно);

б) розширюваність;

в) можливості ACID;

г) розширена підтримка SQL;

д) реплікація та висока доступність;

е) підтримка типу даних JSON та NoSQL систем.

PostgreSQL є потужним та гнучким рішенням для управління базами даних, яке широко використовується як у веб-розробці, так і в інших сферах програмування. Він має активну спільноту користувачів та постійно розвивається, що робить його одним з найпопулярніших виборів для багатьох проектів див. рис 4.1.

Рисунок 4.1 – Логотип СУБД PostgreSQL

4.2 Node.js

Node.js – це платформа, заснована з використанням двигуна V8 (він здійснює переклад мови JavaScript в зрозумілий комп’ютеру), що вже перетворює сам JavaScript з вузької мови програмування в мову широкого призначення. Node дає можливість взаємодіяти з таким як введення-виведення за допомогою API (Application Programming Interface), підключати до свого проекту стороні пакети, які написані на різних існуючих вже мовах програмування [10]. Node здебільшого використовується на стороні сервер. В

основі Node.js покладено асинхронне програмування з неблокуючим введенням-виводом див. рис 4.2.

Особливості та переваги:

а) асинхронність або необлокаючий ввід/вивід;

б) широка підтримка пакетів;

в) висока продуктивність;

г) серверна сторона та мережеві додатки.

Рисунок 4.2 – Логотип Node.js

4.3 Менеджер пакетів nmp

NMP – це менеджер пакетів, який використовується у Node.js для завантаження сторонніх пакетів до себе проект.

Менеджер пакетів по замовчуванням вже вбудований в Node.js і для того, щоб його використати потрібно в консолі прописати команду для встановлення пакету у додаток. З цим ми можемо використовувати пакети, які були написані на інших мовах програмування за допомогою JavaScript. Після встановлення пакету його версія та назва одразу записується у файл package.json з цим програміст може легко оновлювати свої пакети та уникати якихось змін див. рис 4.3.

Рисунок 4.3 – Логотип менеджера пакетів npm

4.4 Express

Express – це фреймворк для Node.js, який реалізовує шар функцій, необхідних для створення ефективних програм та API. Він скорочує написання коду, а, значить, зменшуються витрати на розробку див. рис 4.4.

Node.js Express має готові функції обробки HTTP запитів, причому для кожного HTTP методу є своя функція, що особливо зручно при створенні REST – це далеко не єдина причина використання Express.

Для досягнення архітектурного шаблону MVC з використанням фреймворку Express потрібно реалізувати наступне:

а) модель – це те, що буде обробляти дані нашого додатку. У неї повинен бути доступ до об’єкта бази даних, що повертається об’єктом PostgresClient. Наша модель також має мати метод для її розширення у разі необхідності.

б) представлення – це те, що буде виводити інформацію на екран. Говорячи простою мовою, представлення – це клас, який відправляє браузеру відповідь. Express надає просту можливість це робити.

в) контролер – це маршрут, який направляється на middleware-функції, яка вже приймає в себе request, response чи callback-функцію next. Ці три об’єкти вже оброблюються в самій функції.

Рисунок 4.4 – Логотип фреймворку express

4.5 Мова програмування JavaScript

JavaScript або JS – це об’єктно орієнтована-динамічна мова програмування [11]. Зазвичай JS використовується при розробці сценарію у

веб-сторінках, які будуть використовувати в браузері користувач, асинхронний обмін чи вигляду веб-сторінки. JavaScript є основною мовою для програмування на Node.js. Завдяки його гнучкості та асинхронності, можливо створювати різні дії циклічно при роботі. JS зазвичай класифікують як скриптову мову з динамічною типізацією. Також підтримує архітектурні властивості як динамічна та слабка типізація, автоматичний збір сміття для керування пам’яттю див. рис 4.5.

Мова JS використовується для:

а) написання коду у веб-сторінках для надання їм гнучкості;

б) розробка з фреймворками як (Express, React, AngularJS);

в) розробка на сервері Node.js.

Рисунок 4.5 – Логотип мови програмування JavaScript

4.6 Bootstrap

Користувацький інтерфейс інформаційної системи подається у вигляді сайту за допомогою Bootstrap (HTML- та CSS-шаблон, включаючи JavaScript розширення) див. рис 4.6.

Bootstrap є найпопулярнішою бібліотекою компонентів у світі Bootstrap є відкритим вихідним кодом для розробки з HTML, CSS і JS [12]. Швидкий прототип ідей користувачів, гнучка система з великими готовими компонентами і потужними плагінами, побудованими на jQuery.

Рисунок 4.6 – Логотип Bootstrap

4.7 Visual studio code

Visual Studio Code (VS Code) – це безкоштовне, відкрите та легковагове середовище розробки, розроблене компанією Microsoft. Воно надає широкі можливості для редагування та налагодження коду, підтримку різних мов програмування та розширення, що робить його популярним. (рис. 4.7).

Особливості та переваги:

а) розширюваність та легкість у використані;

б) потужність та продуктивність;

в) кросплатформеність та робота з Git.

Рисунок 4.7 – Логотип VS code

4.8 Starlink

Starlink – це глобальна мережа супутників, що належить SpaceX, компанії, заснованої Ілоном Маском. Метою Starlink є надання доступу до

швидкісного та доступного Інтернету по всьому світу, зокрема в регіонах з обмеженим доступом до інфраструктури зв'язку (рис. 4.8).

Система Starlink базується на мережі тисяч супутників, розташованих на низьких орбітах. Коли супутник розташовується на низькій орбіті, його відстань до Землі набагато менша, ніж у випадку геостаціонарних супутників, що зазвичай використовуються для забезпечення супутникового зв'язку. Це дозволяє зменшити затримки сигналу та забезпечити вищу швидкість передачі даних.

Користувачі можуть отримати доступ до Інтернету Starlink за допомогою антени, встановленої на землі, яка підключається до супутникової мережі. Коли антена налаштована на прийом сигналу від супутників, вона забезпечує Інтернет-підключення до підключених пристроїв у будинку або офісі.

Starlink розгортається поступово, з метою створення констелляції з близько 12 000 супутників, що охоплюватимуть усю планету. Це дозволить забезпечити покриття в будь-якому місці на Землі та надати доступ до швидкого Інтернету людям, які проживають у віддалених регіонах або тим, хто має обмежений доступ до існуючих мереж зв’язку.

Рисунок 4.8 – Логотип Starlink

5 РЕАЛІЗАЦІЯ ТА ТЕСТУВАННЯ

5.1 Створення бази даних

В цьому розділі описані основні об’єкти бази даних та права доступу користувачів до неї. Повний код створення бази даних наведено у додатку В.

Усі перераховані нижче об’єкти бази даних ship_company зберігаються у схемі sea.

Створення таблиць для реалізації концептуальної моделі:

– role_type – містить дані про ролі користувачів. Тобто в нас є такі ролі: «Адмін», «Капітан», «Технічний відділ», «Фінансовий відділ», «Крюїнковий відділ» та «Оператор». Усі поля є обов’язковими до заповнення;

– role_type – зберігає дані про ролі у інформаційній системі;

– employee – зберігає дані про всіх користувачів системи;

– rank – зберігає дані про назви рангів для членів команди судна;

– sea_worker – містить дані про робітників на судні;

– office_worker – містить дані про робітників у офісі;

– message – таблиця, що зберігає записи повідомлень, які створюють капітан або відділ;

– report – таблиця, що зберігає записи звітів, які створюють капітан або відділ один одному;

– team – містить дані про членів команди на судні;

– ship – таблиця, що зберігає інформацію про судна та капітана судна;

– department – містить дані про відділ;

– type – містить дані про тип звітів.

```
CREATE TABLE IF NOT EXISTS sea.role_type (
     id serial PRIMARY KEY NOT NULL,
     name text NOT NULL UNIQUE
);
```

Лістинг 5.1 – SQL-запит створення таблиці role_type

Ключова фраза CREATE TABLE визначає безпосередньо створення таблиці. Створення відбувається якщо ця таблиця ще не фігурує в базі. За це відповідає IF NOT EXISTS. Кожному полю таблиці зіставляється певний тип даних та задається певне обмеження за допомогою службового слова CONSTRAINT. Таким чином атрибути таблиці id визначаються як первинний та зовнішній ключ. Всі поля таблиці мають бути не нульовими, тобто вони не можуть містити порожні значення.

Типи даних атрибутів таблиці role_type:

– id має тип даних serial, тобто це є просто зручний засіб для створення стовпців із унікальними ідентифікаторами;

– name має текстовий тип даних;

Повний код створення таблиць бази даних наведено у додатку В.

Представлення – віртуальна таблиця, що представляє собою запит який буде підставлений як підзапит при використанні. Представлення були створені для того, щоб звернення проходило не напряму до таблиць, так як це не є безпечно, тому що можна змінити дані таблиці.

Створені такі представлення в базі даних, код яких більш детально можна подивитися у додатку В:

– employee_info– зберігає дані усіх робітників за паролями у зашифрованому виді;

– show_department – містить інформацію про назву на ідентифікатори усіх відділів;

– show_fin_report_oper – переглянути фінансовому відділу звіти за певним типом від оператора;

– show_fin_report_tech – переглянути фінансовому відділу звіти за певним типом від технічного відділу;

– show_fin_report_krui – переглянути фінансовому відділу звіти за певним типом від крюїнгового відділу;

– show_message_captain –дані для перегляду повідомлень капітаном від відділів;

– show_report_null – переглянути інформацію про звіт, на який ще не була дана відповідь;

– show_report_true – переглянути інформацію про звіт, на який вже була дана відповідь і вона є позитивною;

– show_report_false – переглянути інформацію про звіт, на який вже була дана відповідь і вона є негативною;

– show_ship – переглянути дані про судна;

– show_type – переглянути дані про тип звітів;

– signin – зберігає дані, а саме логін, пароль та роль для входу у свій персональний кабінет.

Детально розглянемо код одного з перерахованих вище представлень – представлення sea.signin.

Створення представлення відбувається за допомогою наступного SQL-запиту дивитись лістинг 5.2:

```
CREATE OR REPLACE VIEW sea.signin AS
SELECT
    e.phone AS login,
    e.password AS password,
    r.name AS position
FROM sea.employee e
    INNER JOIN sea.role_type r ON (r.id = e.role_type_id)
WHERE TRUE
    AND (e.is_delete = FALSE);
```

Лістинг 5.2 – SQL-запит створення представлення sea.signin

Ключова фраза CREATE VIEW визначає безпосередньо створення представлень є CREATE OR REPLACE VIEW. Представлення будується на основі двох таблиць: sea.employee та sea.role_type, що об’єднуються за допомогою INNER JOIN після чого робимо SELECT потрібних нам рядків. Для виклику представлень потрібно написати SELECT * FROM sea.signin.

Збережені процедура – об'єкт бази даних, що представляє собою набір SQL-інструкцій, який компілюється один раз і зберігається на сервері.

Збережені процедури можуть також містити оголошені змінні для обробки даних і курсорів, які дозволяють організувати цикл з кількох рядків в таблиці.

Збережені процедури, які були реалізовані у інформаційній системі (код можливо переглянути у Додатку В):

– sea._analysis_all_ship(date, date) – функція для аналізу витрат за певний період часу, а саме сума та кількість їх;

– sea._answer_report(integer,boolean, text) – функція для відповіді на звіт;

– sea._show_message_by_department(text, integer) – функція для перегляду повідомлень відділам;

– sea_show_report_by_department (text) – функція для перегляду звітів відділам;

– sea_show_report_by_department_true (text) – функція для перегляду звітів відділам, які вже підтверджені;

– sea_show_report_by_department_false (text) – функція для перегляду звітів відділам, які були відхилені;

– sea._show_report_by_department_answer (text, integer) – функція для перегляду відповіді на звіти відділам;

– sea._user_create(text,text,text,date,text,text,text,text) – функція для реєстрації робітників;

– sea._user_delete(integer[]) – функція для видалення робітників;

– sea._user_update(integer,text,text,text,date,text, text) – функція для оновлення робітників;

– sea._write_report(text, integer, text, integer,text,integer) – функція для написання звітів капітану;

– sea._write_report_to_captain(text, integer, text, integer,text,integer) – функція для написання звітів капітану від відділів;

– sea._write_report_to_department(text, text, integer, text,integer) – функція для написання звітів від відділу до відділа;

– sea._write_text(text, text, text) – функція для написання повідомлення від капітана до відділів;

– sea._write_text_to_captain(text, text, text) – функція для написання повідомлення від відділів до капітана.

Розглянемо детальніше наступну збережену процедуру, код SQL якої виглядає наступним чином, дивись лістинг 5.3:

```
CREATE OR REPLACE FUNCTION sea._write_report(_phone
text,_department integer,_type text,_sum integer,_text
text,_how_send integer)
RETURNS void
SECURITY DEFINER
LANGUAGE plpgsql
AS
$BODY$
DECLARE
_id_ship integer;
_id_type integer;
_id_department integer;
BEGIN
 SELECT
        t.id
    INTO
        _id_type
    FROM sea.type t
    WHERE TRUE
        AND (t.name = _type);
SELECT
        s.id
    INTO
        _id_ship
    FROM sea.employee e
        INNER JOIN sea.sea_worker sw ON (e.id = sw.employee_id)
INNER JOIN sea.ship s ON (sw.id = s.sea_worker_id)
WHERE TRUE
    AND (phone = _phone);
INSERT INTO sea.report (_id,
department_id,type_id,sum,date_time,message_text,how_send)
VALUES(_id_ship,_department,_id_type,_sum,CURRENT_TIMESTAMP,_tex
t,_how_send);
END;
$BODY$;
```

Лістинг 5.3 – SQL-запит створення функції sea._write_report

Для того щоб створити функцію потрібно написати ключову фразу CREATE OR REPLACE FUNCTION. Далі нам потрібно визначити схему, де вона буде знаходитися та її назву. Після чого нам потрібно передати в функцію

один аргумент – це номер телефону користувача, відділ, тип, сума, текс звіту та хто відправив. Дана функція нічого не повертає, тому тип значення, що вертається буде RETURNS VOID. Коли було визначено тип значення, що вертається, то далі дефінуємо мову програмування функції та безпечний доступ до неї. Після чого оголосимо дві змінні: _id_type та _id_ship для подальшого отримання ідентифікаторів запитів.

Починаємо розглядати основний блок коду функції. Перше, що робимо беремо номер телефону, що був переданий нам через аргумент для пошуку капітана корабля та робимо пошук за допомогою запиту SQL до таблиці sea.ship, де по номеру телефона знайдемо ідентифікатор судна. Далі знаходими ідентифікатор тип за допомогою переданої назви типу у аргументі. У самому кінці робимо вставку до таблиці sea.report та вносимо усі передані нам аргумент.

5.2 Запити для вирішення задач

У цьому розділі переглянемо вирішення поставлених завдань, які ми визначили у другому розділі. Усі збережені процедури (функції) та представлення можна переглянути у Додатку В.

Спочатку вирішимо запити до користувача – адміністратор.

Для вирішення задачі А.1 нам потрібно створити збережену процедуру для реєстрації нового робітника sea._user_create(text, text, text, date, text, text, text) для звернення до якої необхідний запит SELECT * FROM sea._ user_create ($1,$2,$3,$4,$5,$6,$7,$8), як можемо побачити тут ми передаємо такі аргументи, де $1 – ім’я, $2 – прізвище, $3 – адреса, $4 – дату народження, $5 – логін (номер телефону), $6 – пароль, $7 – роль, $8 – ранг.

Для вирішення задачі А.2 нам потрібно створити збережену процедуру для видалення робітника sea._user_delete(integer[]) для звернення до якої необхідний запит SELECT * FROM sea._user_delete ($1), як можемо побачити тут ми передаємо аргумент, де $1 – ідентифікатор користувача.

Для вирішення задачі А.3 нам потрібно створити збережену процедуру для редагування користувача sea._user_update(integer,text,text,text,date,text, text) для звернення до якої необхідний запит SELECT * FROM sea._user_update($1,$2,$3,$4,$5,$6,$7) як можемо побачити тут ми передаємо такі аргументи, де $1 – ідентифікатор, $2 – ім’я, $3 –прізвище, $4 – адреса, $5 – логін (номер телефона), $6 –дата народження, $7 – пароль.

Вирішимо запити до користувача – капітан.

Для вирішення задачі Ка.1 нам потрібно створити представлення для перегляду звітів sea.show_report_null для звернення до якої необхідний запит SELECT * FROM sea.show_report_null.

Для вирішення задачі Ка.1 нам потрібно створити представлення для перегляду звітів sea.show_report_null для звернення до якої необхідний запит SELECT * FROM sea.show_report_null.

Для вирішення задачі Ка.2 нам потрібно створити представлення для перегляду звітів sea.show_report_true для звернення до якої необхідний запит SELECT * FROM sea.show_report_true.

Для вирішення задачі Ка.3 нам потрібно створити представлення для перегляду звітів sea.show_report_false для звернення до якої необхідний запит SELECT * FROM sea.show_report_false.

Для вирішення задачі Ка.4 нам потрібно створити збережену процедуру для відповіді на звіт sea._answer_report(integer, boolean, text) для звернення до якої необхідний запит SELECT * FROM sea._user_update($1, $2 ,$3 ,$4, $5, $6, $7) як можемо побачити тут ми передаємо такі аргументи, де $1 – ідентифікатор, $2 – TRUE або FALSE, $3 – коментар.

Для вирішення задачі Ка.5 нам потрібно створити збережену процедуру для написання звіту оператору sea._write_report(text, integer, text, integer, text, integer) для звернення до якої необхідний запит SELECT * FROM sea._write_report ($1, $2 ,$3 ,$4, $5, $6) як можемо побачити тут ми передаємо такі аргументи, де $1 – телефон, $2 – відділ (оператор), $3 –тип (щоденний звіт), $4 – NULL, $5 – текст звіту, $6 – хто відправив.

Для вирішення задачі Ка.6 нам потрібно створити збережену процедуру для написання звіту на заміну членів екіпажу sea._write_report(text, integer, text, integer, text, integer) для звернення до якої необхідний запит SELECT * FROM sea._write_report ($1, $2 ,$3 ,$4, $5, $6) як можемо побачити тут ми передаємо такі аргументи, де $1 – телефон, $2 – відділ (крюїнговий), $3 –тип (заміна членів екіпажу), $4 – NULL, $5 – текст звіту, $6 – хто відправив.

Для вирішення задачі Ка.7 нам потрібно створити збережену процедуру для написання звіту на заміну деталі sea._write_report(text, integer, text, integer, text, integer) для звернення до якої необхідний запит SELECT * FROM sea._write_report ($1, $2 ,$3 ,$4, $5, $6) як можемо побачити тут ми передаємо такі аргументи, де $1 – телефон, $2 – відділ (технічний), $3 –тип (заміна деталі), $4 – NULL, $5 – текст звіту, $6 – хто відправив.

Для вирішення задачі Ка.8 нам потрібно створити збережену процедуру для написання звіту на бункерування sea._write_report(text, integer, text, integer, text, integer) для звернення до якої необхідний запит SELECT * FROM sea._write_report ($1, $2 ,$3 ,$4, $5, $6) як можемо побачити тут ми передаємо такі аргументи, де $1 – телефон, $2 – відділ (технічний), $3 –тип, $4 – NULL, $5 – текст звіту, $6 – хто відправив.

Для вирішення задачі Ка.9 нам потрібно створити збережену процедуру для написання звіту з грошових витрат sea._write_report(text, integer, text, integer, text, integer) для звернення до якої необхідний запит SELECT * FROM sea._write_report ($1, $2 ,$3 ,$4, $5, $6) як можемо побачити тут ми передаємо такі аргументи, де $1 – телефон, $2 – відділ (оператор), $3 –тип (грошовий звіт), $4 – сума, $5 – текст звіту, $6 – хто відправив.

Для вирішення задачі Ка.10 нам потрібно створити збережену процедуру для написання звіту відгука/характеристики на члена екіпажу sea._write_report(text, integer, text, integer, text, integer) для звернення до якої необхідний запит SELECT * FROM sea._write_report ($1, $2 ,$3 ,$4, $5, $6) як можемо побачити тут ми передаємо такі аргументи, де $1 – телефон, $2 – відділ (крюїнговий), $3 –тип (відгук), $4 – NULL, $5 – текст звіту, $6 – хто відправив.

Для вирішення задачі Ка.11 нам потрібно створити збережену процедуру для написання звіту відгука/характеристики на члена екіпажу sea._write_report(text, integer, text, integer, text, integer) для звернення до якої необхідний запит SELECT * FROM sea._write_report ($1, $2 ,$3 ,$4, $5, $6) як можемо побачити тут ми передаємо такі аргументи, де $1 – телефон, $2 – відділ (крюїнговий), $3 –тип (відгук), $4 – NULL, $5 – текст звіту, $6 – хто відправив.

Для вирішення задачі Ка.12 нам потрібно створити збережену процедуру для написання звіту відгука/характеристики на члена екіпажу sea._write_report(text, integer, text, integer, text, integer) для звернення до якої необхідний запит SELECT * FROM sea._write_report ($1, $2 ,$3 ,$4, $5, $6) як можемо побачити тут ми передаємо такі аргументи, де $1 – телефон, $2 – відділ (крюїнговий), $3 –тип (відгук), $4 – NULL, $5 – текст звіту, $6 – хто відправив.

Для вирішення задачі Ка.13 нам потрібно створити представлення для перегляду повідомлень від технічного відділу sea.show_message_captain для звернення до якої необхідний запит SELECT * FROM sea.show_message_captain WHERE department = ‘Технічний відділ’.

Для вирішення задачі Ка.14 нам потрібно створити представлення для перегляду повідомлень від крюїнгового відділу sea.show_message_captain для звернення до якої необхідний запит SELECT * FROM sea.show_message_captain WHERE department = ‘Крюінковий відділ’.

Для вирішення задачі Ка.15 нам потрібно створити представлення для перегляду повідомлень від фінансового відділу sea.show_message_captain для звернення до якої необхідний запит SELECT * FROM sea.show_message_captain WHERE department = ‘Фінансовий відділ’.

Для вирішення задачі Ка.16 нам потрібно створити представлення для перегляду повідомлень від оператора sea.show_message_captain для звернення до якої необхідний запит SELECT * FROM sea.show_message_captain WHERE department = ‘Менеджер (оператор судна).

Вирішимо запити до користувача – технічний відділ.

Для вирішення задачі Т.1 нам потрібно створити збережену процедуру для написання звіту на перевірку технічного стану судна sea._write_report_to_captain(text, integer, text, integer, text, integer) для звернення до якої необхідний запит SELECT * FROM sea._write_report_to_captain ($1, $2 ,$3 ,$4, $5, $6) як можемо побачити тут ми передаємо такі аргументи, де $1 – телефон, $2 – ідентифікатор судна $3 –тип (перевірка технічного стану), $4 – NULL, $5 – текст звіту, $6 – хто відправив.

Для вирішення задачі Т.2 нам потрібно створити збережену процедуру для перегляду звітів від капітанів суден sea._show_report_by_department(text) для звернення до якої необхідний запит SELECT * FROM sea._show_report_by_department_true ($1) як можемо побачити тут ми передаємо такі аргументи, де $1 – телефон.

Для вирішення задачі Т.3 нам потрібно створити збережену процедуру для перегляду звітів від капітанів суден, які були прийняті sea._show_report_by_department_true(text) для звернення до якої необхідний запит SELECT * FROM sea._show_report_by_department_true ($1) як можемо побачити тут ми передаємо такі аргументи, де $1 – телефон.

Для вирішення задачі Т.4 нам потрібно створити збережену процедуру для перегляду звітів від капітанів суден, які були відхилені sea._show_report_by_department_false(text) для звернення до якої необхідний запит SELECT * FROM sea._show_report_by_department_false ($1) як можемо побачити тут ми передаємо такі аргументи, де $1 – телефон.

Для вирішення задачі Т.5 нам потрібно створити збережену процедуру для відповіді на запит від капітана sea._answer_report(integer, boolean, text) для звернення до якої необхідний запит SELECT * FROM sea._answer_report ($1, $2, $3) як можемо побачити тут ми передаємо такі аргументи, де $1 – ідентифікатор, $2 – TRUE/FALSE $3 – коментар відповіді.

Для вирішення задачі Т.6 нам потрібно створити збережену процедуру для написання звіту для фінансового відділу sea _write_report_to_department (text, text, integer, text, integer) для звернення до якої необхідний запит SELECT

* FROM sea _write_report_to_department ($1, $2, $3, $4, $5) як можемо побачити тут ми передаємо такі аргументи, де $1 – телефон, $2 – тип (фінансовий звіт) $3 – сума, $4 – текст звіту, $5 – хто відправив.

Для вирішення задачі Т.7 нам потрібно створити збережену процедуру для перегляду повідомлень від капітана sea._show_message_by_department (text, integer) для звернення до якої необхідний запит SELECT * FROM sea._show_message_by_departmenti ($1, $2) як можемо побачити тут ми передаємо такі аргументи, де $1 – телефон, $2 – ідентифікатор судна.

Для вирішення задачі Т.8 нам потрібно створити збережену процедуру для написання повідомлень до капітана sea._write_text_to_captain(text, text, integer, integer) для звернення до якої необхідний запит SELECT * FROM sea._write_text_to_captain ($1, $2, $3, $4) як можемо побачити тут ми передаємо такі аргументи, де $1 – текст, $2 –телефон, ідентифікатор судна< хто відправив.

Вирішимо запити до користувача – Крюїнговий відділ.

Для вирішення задачі Кр.1 нам потрібно створити збережену процедуру для написання звіту до капітана щодо заміни члена команди sea._write_report_to_captain(text, integer, text, integer, text, integer) для звернення до якої необхідний запит SELECT * FROM sea._write_report_to_captain ($1, $2 ,$3 ,$4, $5, $6) як можемо побачити тут ми передаємо такі аргументи, де $1 – телефон, $2 – ідентифікатор судна $3 –тип (заміна члена екіпажу), $4 – NULL, $5 – текст звіту, $6 – хто відправив.

Для вирішення задачі Кр.2 нам потрібно створити збережену процедуру для перегляду звітів від капітанів суден щодо заміни члена екіпажу sea._show_report_by_department(text) для звернення до якої необхідний запит SELECT * FROM sea._show_report_by_department_true ($1) як можемо побачити тут ми передаємо такі аргументи, де $1 – телефон.

Для вирішення задачі Кр.3 нам потрібно створити збережену процедуру для перегляду звітів від капітанів суден, які були прийняті sea._show_report_by_department_true(text) для звернення до якої необхідний

запит SELECT * FROM sea._show_report_by_department_true ($1) як можемо побачити тут ми передаємо такі аргументи, де $1 – телефон.

Для вирішення задачі Кр.4 нам потрібно створити збережену процедуру для перегляду звітів від капітанів суден, які були відхилені sea._show_report_by_department_false(text) для звернення до якої необхідний запит SELECT * FROM sea._show_report_by_department_false ($1) як можемо побачити тут ми передаємо такі аргументи, де $1 – телефон.

Для вирішення задачі Кр.5 нам потрібно створити збережену процедуру для відповіді на запит від капітана sea._answer_report(integer, boolean, text) для звернення до якої необхідний запит SELECT * FROM sea._answer_report ($1, $2, $3) як можемо побачити тут ми передаємо такі аргументи, де $1 – ідентифікатор, $2 – TRUE/FALSE $3 – коментар відповіді.

Для вирішення задачі Кр.6 нам потрібно створити збережену процедуру для написання звіту для фінансового відділу щодо заробітної плати sea _write_report_to_department (text, text, integer, text, integer) для звернення до якої необхідний запит SELECT * FROM sea _write_report_to_department ($1, $2, $3, $4, $5) як можемо побачити тут ми передаємо такі аргументи, де $1 – телефон, $2 – тип (фінансовий звіт) $3 – сума, $4 – текст звіту, $5 – хто відправив.

Для вирішення задачі Кр.7 нам потрібно створити збережену процедуру для перегляду повідомлень від капітана sea._show_message_by_department (text, integer) для звернення до якої необхідний запит SELECT * FROM sea._show_message_by_departmenti ($1, $2) як можемо побачити тут ми передаємо такі аргументи, де $1 – телефон, $2 – ідентифікатор судна.

Для вирішення задачі Кр.8 нам потрібно створити збережену процедуру для написання повідомлень до капітана sea._write_text_to_captain(text, text, integer, integer) для звернення до якої необхідний запит SELECT * FROM sea._write_text_to_captain ($1, $2, $3, $4) як можемо побачити тут ми передаємо такі аргументи, де $1 – текст, $2 –телефон, ідентифікатор судна< хто відправив.

Вирішимо запити до користувача – менеджера (оператора судна).

Для вирішення задачі М.1 нам потрібно створити збережену процедуру для написання звіту до капітана щодо рейсового завдання sea._write_report_to_captain(text, integer, text, integer, text, integer) для звернення до якої необхідний запит SELECT * FROM sea._write_report_to_captain ($1, $2 ,$3 ,$4, $5, $6) як можемо побачити тут ми передаємо такі аргументи, де $1 – телефон, $2 – ідентифікатор судна $3 –тип (заміна члена екіпажу), $4 – NULL, $5 – текст звіту, $6 – хто відправив.

Для вирішення задачі М.2 нам потрібно створити збережену процедуру для написання звіту до капітана щодо вимог портів, які йому потрібні sea._write_report_to_captain(text, integer, text, integer, text, integer) для звернення до якої необхідний запит SELECT * FROM sea._write_report_to_captain ($1, $2 ,$3 ,$4, $5, $6) як можемо побачити тут ми передаємо такі аргументи, де $1 – телефон, $2 – ідентифікатор судна $3 –тип (заміна члена екіпажу), $4 – NULL, $5 – текст звіту, $6 – хто відправив.

Для вирішення задачі М.3 нам потрібно створити збережену процедуру для перегляду звітів від капітанів sea._show_report_by_department(text) для звернення до якої необхідний запит SELECT * FROM sea._show_report_by_department_true ($1) як можемо побачити тут ми передаємо такі аргументи, де $1 – телефон.

Для вирішення задачі М.4 нам потрібно створити збережену процедуру для перегляду звітів від капітанів суден, які були прийняті sea._show_report_by_department_true(text) для звернення до якої необхідний запит SELECT * FROM sea._show_report_by_department_true ($1) як можемо побачити тут ми передаємо такі аргументи, де $1 – телефон.

Для вирішення задачі М.5 нам потрібно створити збережену процедуру для перегляду звітів від капітанів суден, які були відхилені sea._show_report_by_department_false(text) для звернення до якої необхідний запит SELECT * FROM sea._show_report_by_department_false ($1) як можемо побачити тут ми передаємо такі аргументи, де $1 – телефон.

Для вирішення задачі М.6 нам потрібно створити збережену процедуру для відповіді на запит від капітана sea._answer_report(integer, boolean, text) для звернення до якої необхідний запит SELECT * FROM sea._answer_report ($1, $2, $3) як можемо побачити тут ми передаємо такі аргументи, де $1 – ідентифікатор, $2 – TRUE/FALSE $3 – коментар відповіді.

Для вирішення задачі М.7 нам потрібно створити збережену процедуру для написання звіту для фінансового відділу щодо незапланованих витрат капітана судна sea _write_report_to_department (text, text, integer, text, integer) для звернення до якої необхідний запит SELECT * FROM sea _write_report_to_department ($1, $2, $3, $4, $5) як можемо побачити тут ми передаємо такі аргументи, де $1 – телефон, $2 – тип (фінансовий звіт) $3 – сума, $4 – текст звіту, $5 – хто відправив.

Для вирішення задачі М.8 нам потрібно створити збережену процедуру для перегляду повідомлень від капітана sea._show_message_by_department (text, integer) для звернення до якої необхідний запит SELECT * FROM sea._show_message_by_departmenti ($1, $2) як можемо побачити тут ми передаємо такі аргументи, де $1 – телефон, $2 – ідентифікатор судна.

Для вирішення задачі М.9 нам потрібно створити збережену процедуру для написання повідомлень до капітана sea._write_text_to_captain(text, text, integer, integer) для звернення до якої необхідний запит SELECT * FROM sea._write_text_to_captain ($1, $2, $3, $4) як можемо побачити тут ми передаємо такі аргументи, де $1 – текст, $2 –телефон, ідентифікатор судна< хто відправив.

Вирішимо запити до користувача – фінансовий відділ.

Для вирішення задачі Ф.1 нам потрібно створити представлення для перегляду фінансових звітів від технічного відділу sea.show_fin_report_tech, до якої необхідний запит SELECT * FROM show_fin_report_tech.

Для вирішення задачі Ф.2 нам потрібно створити представлення для перегляду фінансових звітів від крюїнгового відділу sea.show_fin_report_krui, до якої необхідний запит SELECT * FROM show_fin_report_krui.

Для вирішення задачі Ф.3 нам потрібно створити представлення для перегляду фінансових звітів від менеджера суден sea.show_fin_report_oper, до якої необхідний запит SELECT * FROM show_fin_report_oper.

Для вирішення задачі Ф.4 нам потрібно створити збережену процедуру для перегляду суми витрат за фільтром дат sea._analysis_all_ship(date, date) для звернення до якої необхідний запит SELECT * FROM sea._analysis_all_ship ($1, $2) як можемо побачити тут ми передаємо такі аргументи, де $1 – дата від, $2 – дата до.

Для вирішення задачі Ф.5 нам потрібно створити збережену процедуру для перегляду кількості витрат за фільтром дат sea._analysis_all_ship(date, date) для звернення до якої необхідний запит SELECT * FROM sea._analysis_all_ship ($1, $2) як можемо побачити тут ми передаємо такі аргументи, де $1 – дата від, $2 – дата до.

5.3 Безпека інформаційної системи

Безпека на рівні БД забезпечується шляхом створення ролей і наділення їх відповідними привілеями. В інформаційній системі реалізовано шість ролей: «Адміністратор», «Капітан», «Технічний відділ», «Крюїнговий відділ», «Фінансовий відділ», «Оператор судна» та «Логін».

Потрібно зауважити, що безпосереднє звернення запитами до таблиці є неправильним та може нанести шкоду БД, тому було визначено, що ролям не буде наданий доступ до таблиці у ніякому вигляді, а тільки до представлень та збережених процедур (функцій).

Див. табл. 5.1, де наведемо привілеї ролей. При цьому будемо використовувати наступні позначення:

– R (Read) – читання даних;

– E (Execute) – виконання процедури (функції);

– Адміністратор – А;

– Капітан – К;

– Технічний відділ – Т;

– Крюїнговий відділ – Кр;

– Менеджер (оператор суден) – М;

– Фінансовий відділ – Ф;

– Логін – Л.

Таблиця 5.1 – Привілеї ролей на об’єкти БД

Об’єкти бази даних	Ролі						
	А	К	Т	Кр	М	Ф	Л
Представлення							
employee_info	R	R	R	R	R	R	-
show_department	R	-	-	-	R	-	-
show_fin_report_krui	-	-	-	-	-	R	-
show_fin_report_tech	-	-	-	-	-	R	-
show_fin_report_oper	-	-	-	-	-	R	-
show_message_captain	-	R	-	-	-	-	-
show_report_null	-	R	-	-	-	-	-
show_report_true	-	R	-	-	-	-	-
show_report_false	-	R	-	-	-	-	-
show_ship	-	-	R	R	R	-	-
show_type	R	R	R	R	R	R	-
signin	-	-	-	-	-	-	R
Функції							
_analysis_all_ship	-	-	-	-	-	E	-
_answer_report	-	E	-	-	-	-	-
_show_message_by_department	-	-	E	E	E	E	-
_show_report_by_department	-	-	E	E	E	E	-
_show_report_by_department_true	-	-	E	E	E	-	-
_show_report_by_department_false	-	-	E	E	E	-	-
_show_report_by_department_answer	-	-	E	E	E	-	-
_user_create	E	-	-	-	-	-	-
_user_delete	E	-	-	-	-	-	-
_user_update	E	-	-	-	-	-	-
_write_report	-	E	-	-	-	-	-
_write_report_to_captain	-	-	E	E	E	-	-
_write_text	-	E	-	-	-	-	-
_write_text_to_captain	-	-	E	E	E	E	-

Створення ролей і наділення їх привілеями відповідно до наведеної вище таблиці здійснюється за допомогою певних SQL-запитів, які подивитися можна у Додатку В.

Паролі користувачів зберігаються в зашифрованому вигляді (у вигляді хеша) в таблиці employee. Також зашифровані паролі можна переглянути у представленнях: signin та show_employee. Шифрування паролів здійснюється безпосередньо у базі даних за допомогою розширення PostgreSQL – pgcrypto. Паролі шифруються у функціях _user_create() та _user_update().

5.4 Програмна реалізація інформаційної системи

Бекенд сайту написаний мовою JavaScript із допомогою фреймворку Express, фронтенд – HTML/CSS із використанням фреймворку Bootstrap.

Фрагменти програмного коду знаходяться у Додатку Г.

Файлова система проекту складається з папок, див рис. 5.1:

а) view – у папці знаходяться усі ejs (html) сторінки;

б) public – у папці знаходяться css стилі, javascript скрипти, фото, календар з датами та bootstrap стилі;

в) controller – в папці можна знаходяться сім контролерів, для взаємодію у системі: loginController, techController, kruiController, finController, operatorController, captainController, captainController, adminController;

г) model – у папці знаходяться файли для з’єднується з базою даних: techRole, kruiRole, finRole, operatorRole, captainRole, adminRole;

д) config – знаходиться файл з параметрами для підключення до бази даних та порту серверу;

е) node_modules – одна з основних папок, де знаходиться усі модули та пакети які були підключені до мого проекту;

ж) package.json – файл, де знаходиться версії та імена усіх використовуючих пакеті у системі;

к) server.js – точна старту системи.

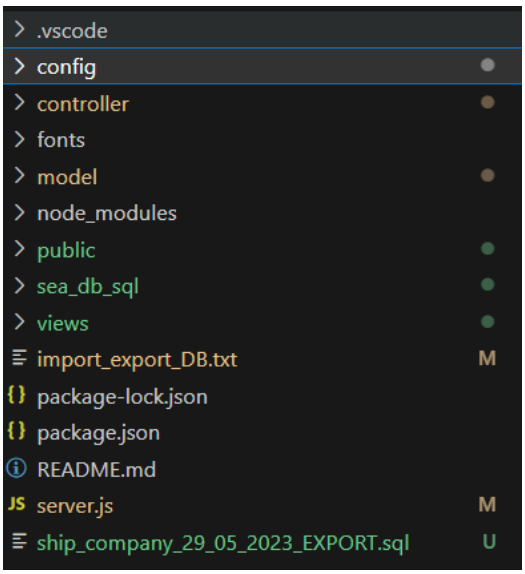

Рисунок 5.1 – Вміст папки з кодом програми

Давайте розробимо (розглянемо) як користувач буде заходити до особистого кабінету. Для цього була реалізована функція на Javascript searchUser(req, res). Детально її розглянути можна на лістингу 5.4.

```
searchUser(req, res) {
  const db = new pg.Pool(client_login.client_login)
  db.connect(function(err, client, done){
  console.log('db.connect')
  if(err){
  return console.error('З'єднання не вдалося')}
    client.query("SELECT * FROM finances.login_phone WHERE login
= $1 AND password = replace(crypt($2, password), ' ',
'')",[req.body.login, req.body.password], function(err, result){
  console.log('query') done()
  if (err || result.rowCount <= 0){
  res.render('error_login_choice')
  return console.error("Запрос не удался")}
if(result.rows[0].position === "Captain"){
cap.getClients(req,res)
console.log("Вошел как Captain")}
else if  (result.rows[0].position === "Crewing_department"){
krui.getClients(req,res)
console.log("Вошел как Crewing_department")}
else if  (result.rows[0].position === "Financial_department"){
fin.getClients(req,res)
console.log("Вошел как Financial_department")}
else if  (result.rows[0].position === "Ship_operator"){
oper.getClients(req,res)
console.log("Вошел как Ship_operator")}
```

Лістинг 5.4 – Javascript код для входу до особистого кабінету, лист 1

```
else if  (result.rows[0].position === "Technical_department"){
tech.getClients(req,res)
console.log("Вошел как Technical_department")}
else if  (result.rows[0].position === "Team"){
fam.getClients(req,res)
console.log("Вошел как Team")}
else if  (result.rows[0].position === "Admin"){
adm.getClients(req,res)
console.log("Вошел как адмнн")}  else {
  console.log("не вішов")}})
  console.log("Увійшли у Postgres")})}
```

Лістинг 5.4 – Javascript код для входу до особистого кабінету, лист 2

Функція реалізована для того, щоб користувач міг зручно увійти до свого особистого кабінету. Перше, що ми робимо при виклику функції – це приєднуємося до бази даних за допомогою певної ролі ця запис const db = new pg.Pool(client_login.client_login) нам демонструє, що приєднання в нас буде передане через зміну client_login.client_login. Сама вона знаходиться у проекті серверу додатку у файлі config.js та містить таку інформацію по цій змінній та певній ролі бази даних, переглянемо її на лістингу 5.5.

```
const client_login = {
    user:'login11',
    database:'family_finances',
    password: '12',
    host:'localhost',
    port:5432,
    max:10,
    idleTimeoutMillis: 30000
}
```

Лістинг 5.5 – Підключення до БД за роллю login11

Сама ця роль має права на доступ у базі даних тільки на представлення signin тому після вдалого приєднання до БД робимо запит, з переданими аргументами користувача логін та пароль щоб визначити, який користувач буде входити та які йому видати права доступу. Як можете бачити сам запит бази даних виконується за допомогою ключового слова client.query і в середені вказуємо запит, де ми співвідношуємо введені дані [req.body.login,

req.body.password] з наявними у базі, а також розшифровуємо пароль. Якщо в нас не вірні данні, то виконується код res.render('error_login_choice') return console.error("Запрос не удался"). А якщо в нас є такий користувач, то ми визначаємо, які привілеї йому видати через умову та за допомогою поля у представлені signin як name у ролі хто входить.

Для того, щоб користувач міг переглядати інформацію з БД на сайті, потрібно у файлах EJS додати вихідну інформацію, яку було виконано на контролері, використовуючи запит БД та передано до файлу EJS. Розглянемо приклад цього, дивись лістинг 5.6.

```
<form action = "/message_captain1"  autocomplete="off"
method="POST">
<button class="nav-link text-white btn btn-success "
style="position: relative;top:15px; left: 450px;"
name ="knop1" type="submit"><h5><b>Обрати</b></h5></button> <br>
<table class="table table-striped table-dark table-hover table-
bordered ">
<thead class="thead-light">
<tr class="active">
<th>Обрати</th>
<th>Ім'я корабля</th>
<th>Ім'я капітану</th>
<th>Рік побудови корабля</th>
<th>Вага</th>
<th>Довжина</th>
</tr></thead>
<tbody>
<% sh_ship.rows.forEach(function(user1){%>
<tr>
<td class="text-center">
<input  type="radio"  name="id_ship" value="<%= user1.id %>">
</td><td >
<%= user1.name %>
</td><td >
<%= user1.last_name %> <%= user1.first_name %></td><td >
<%= user1.year_of_building %></td><td>
<%= user1.weight %></td><td>
<%= user1.length %></td>
<%} ) %></tr></tr></tbody></table></form>
```

Лістинг 5.6 – Вихідна інформація, що сформулюється з запиту БД та показується повідомлення на сторінці

Увесь код програмної реалізації можливо переглянути у Додатку Г.

5.5 Тестування функціоналу

Існує дуже багато видів тестувань, які вже розрізняють та яким вигадали свої визначення, такі як:

– функціональне тестування – це аналіз функціонального характеру у додатку та перевірки на несправність між реальною дією функцій та очікуваним результатом відповідно до специфікації та вимог. Функціональне тестування можна сказати імітує використання у додатку реальним користувачем, що користувався додатком;

– юзабіліті тестування – це перевірка з якою легкістю користувач зможе адаптуватися і користуватися цим веб-додатком. Проводиться для оптимізації, поліпшення і спрощення інтерфейсу до інтуїтивно зрозумілого. Сюди можуть входити всілякі підказки, помічники, зручний дизайн елементів, мінімізація кроків проходження для доступу до функції.

Функціональне тестування може проводитися на усіх рівнях тестування додатку таких як: системному, інтеграційному, компонентному та приймальному. Вище вже згадувалось, як правильно, ця функціональність дій описуються у вимогах, функціональних специфікаціях.

Тестування функцій у мене відбувалось завдяки тест кейсам. В тест кейсах ми перевіримо будемо тестувати функціонал який було розроблено у додатку.

Таблиця 5.2 Тестування відправки повідомлення

Опис	Кроки	Тестові дані	Пріоритет	Очікуємий результат	Реальний результат
Перевіряємо відправку повідомлення капітаном	1. Перейти у меню на сторінку діалогу. 2. Ввести текст. 3. Перевірити, що ввів	Текст якийсь	Середній	Повідомлення повинно бути відправлене	Здійснена відправка

Продовження таблиці 5.2

Опис	Кроки	Тестові дані	Пріоритет	Очікуємий результат	Реальний результат
	4.Натиснути відправити				
Перевіряємо відправку повідомлення відділом до капітана	1. Перейти у меню на сторінку діалогу. 2. Ввести текст. 3.Натиснути відправити	Текст якийсь	Середній	Повідомлення повинно бути відправлене	Здійснена відправка

Таблиця 5.3 Тестування відправки звіту

Опис	Кроки	Тестові дані	Пріоритет	Очікуємий результат	Реальний результат
Перевіряємо відправку звіту капітаном до відділів	1. Обрати у меню вкладку для відправки звіту конкретному відділу. 2. Вести необхідні дані для відправки звіту відділам 3. Перевірити 4. Натиснути кнопку подати звіт	Текс звіту, сума (якщо потрібна), тип	Середній	Звіт відправлено до відділу	Звіт успішно відіслано
Перевіряємо відправку звіту відділом до капітана	1. Обрати у меню вкладку для відправки звіту до капітана. 2. Обрати корабель. 3. Написати звіт. 4.Відправити	Текст звіту, сума, тип, корабель	Середній	Звіт відправлено капітану	Звіт успішно відіслано

Таблиця 5.4 Тестування відповіді на звіт

Опис	Кроки	Тестові дані	Пріоритет	Очікуємий результат	Реальний результат
Перевіряємо відповідь на звіт для капітана	1. Обрати у меню вкладку для відповіді на звіт. 2. Обрати звіт на який будете відповідати. 3.Підтвердити/ відхилити. 4. Коментар	Коментар та Так/Ні	Середній	Відповідь на звіт дана	Відповідь на звіт дана
Перевіряємо відповідь на звіт для відділів	1. Обрати у меню вкладку для відповіді на звіт. 2. Обрати звіт на який будете відповідати. 3. Підтвердити/відхилити. 4. Коментар	Коментар та Так/Ні	Середній	Відповідь на звіт дана	Відповідь на звіт дана

Ось таким чином відбувається перевірка функціональних вимог. У даному випадку були перевірені основні функції у додатку, що потрібен для капітана судна та для відділів.

5.6 Критерії якості інформаційної системи

Таблиця 5.5 Тестування критеріїв якості

Опис	Пріоритет	Результат
Можливість використовувати додаток кросплатформено у будь-якому популярному веб-браузері	Середній	Веб-додаток можливо використовувати в усіх популярних веб-браузерах
Функціональність поставлених задач	Високий	Були реалізовані усі поставлені задачі в додатку

Продовження таблиці 5.5

Опис	Пріоритет	Результат
надійність при використані веб-сайту	Високий	Надійність на сайті при вході та дій забезпечується як на рівні додатку так і на рівні бази даних
Зручність використання додатку	Низький	Сайт був реалізовано з можливістю його зручного використання для усіх можливих користувачів
Ефективність роботи сайту	Низький	Під час роботи додаток видає результат швидко з високим рівнем відповідно до виділених ресурсів

За допомогою функціонального тестування можливо перевірити роботу будь-яких функцій у додатку – це можливо робити автоматичним тестування та мануальним, у нашому випадку – мануальним.

6 ІНСТРУКЦІЯ КОРИСТУВАЧА

Коли ми запускаємо сервер та заходимо на головну сторінку сайту, ми бачимо наступне, головна сторінка інформаційної системи (рис. 6.1).

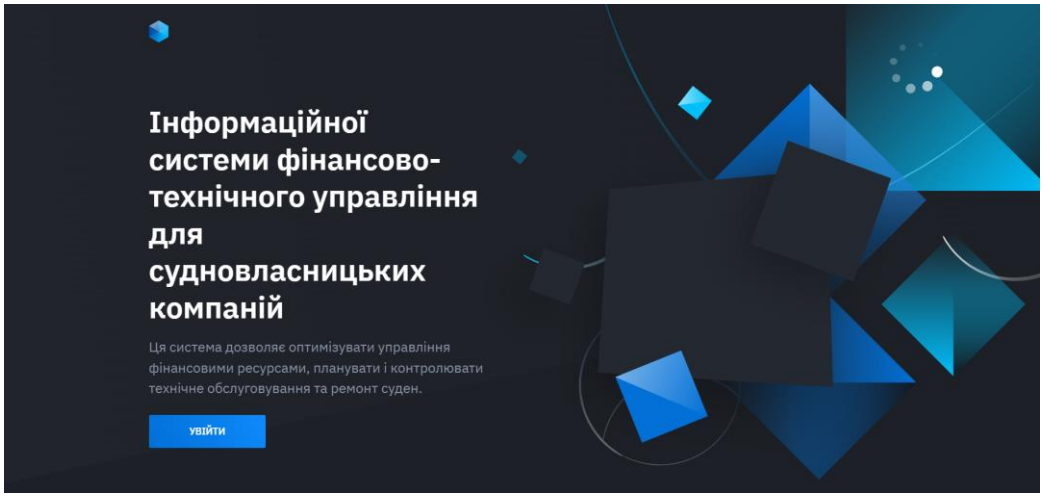

Рисунок 6.1 – Головна сторінка сайту

Головна сторінка сайту не перевантажена непотрібними функціями чи спливаючими вікнами. А зразу пропонує увійти до свого кабінету.

Після натискання кнопки «УВІЙТИ» ми потрапляємо на сторінку авторизації, де потрібно ввести наші дані, а саме номер телефону та пароль для входу до персонального кабінету (рис. 6.2).

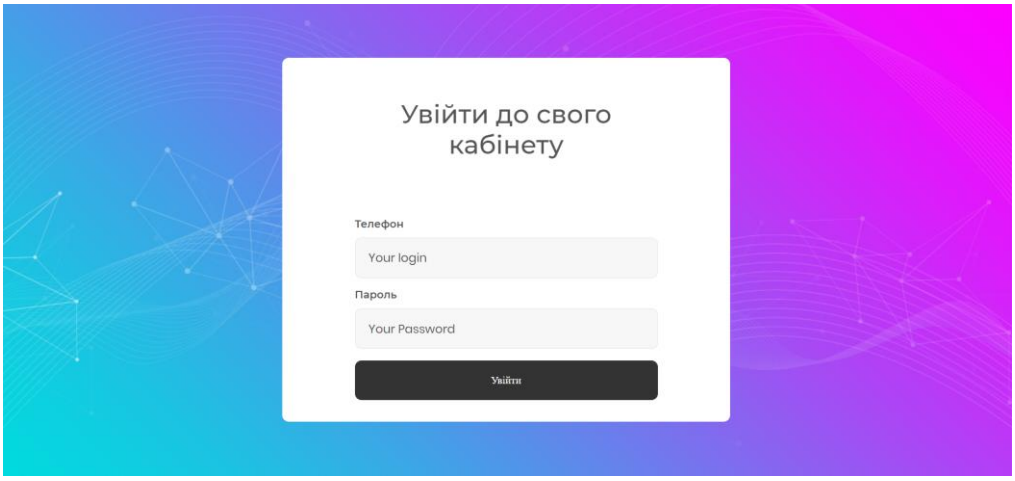

Рисунок 6.2 – Сторінка авторизації

6.1 Адміністратор

Після проходження авторизації як адміністратор, система надає певних привілей адміністратора та ми потрапляємо до профілю адміністратора, сторінка з інформацією про адміністратора (рис. 6.3).

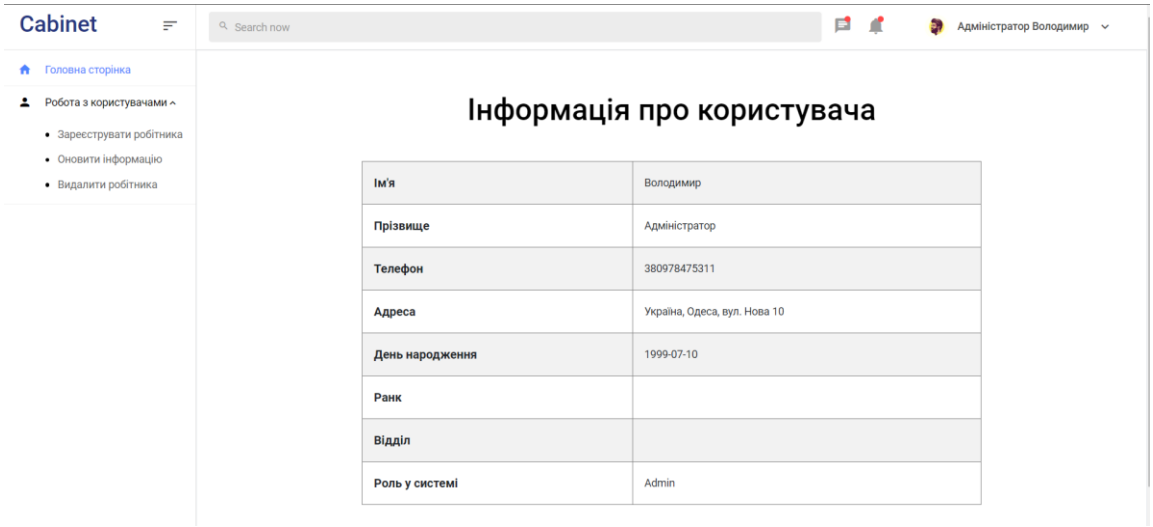

Рисунок 6.3 – Персональний кабінет адміністратора

Далі, ми можемо перейти до сторінки реєстрації користувача для цього натиснемо на відповідний пункт у меню та потрапимо на неї, зареєструвати нового робітника (рис. 6.4).

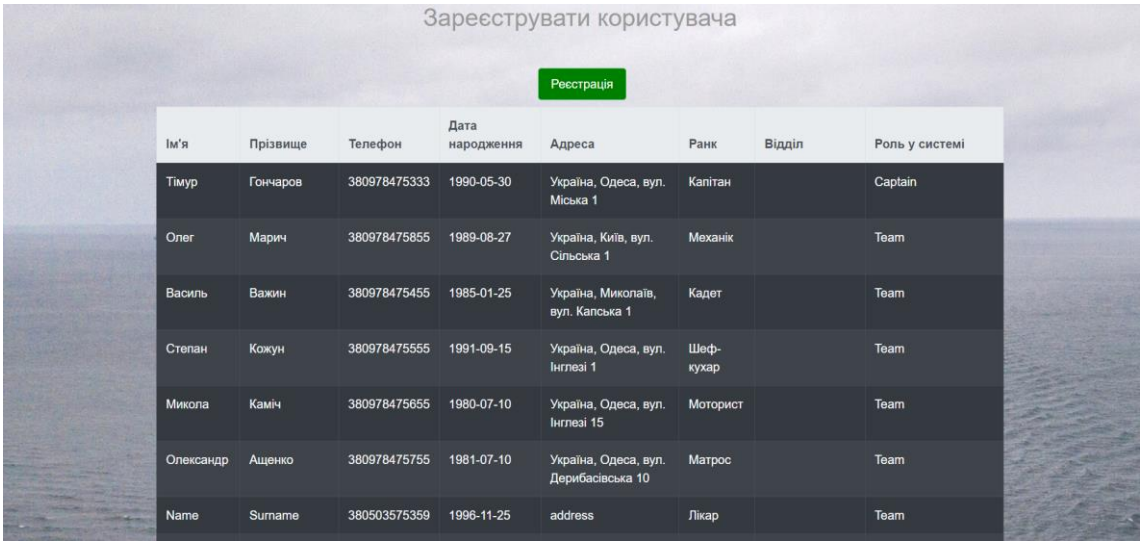

Рисунок 6.4 – Сторінка реєстрації та перегляду існуючих робітників

Для того, щоб додати нового користувача, потрібно натиснути на кнопку реєстрація, де відкриється форма і в котрій треба вписати дані нового користувача (рис. 6.5).

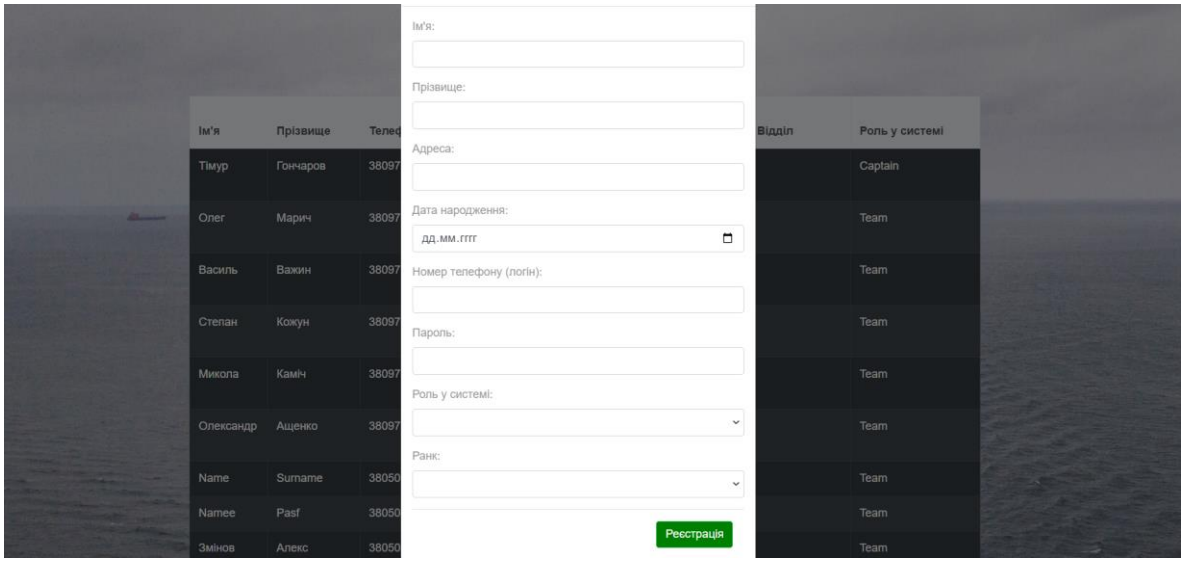

Рисунок 6.5 – Форма реєстрації нового користувача

Далі розглянемо оновлення інформації користувача (рис. 6.6). Де спочатку потрібно обрати інформацію кого будемо змінювати, а далі вже потрібно оновлювати.

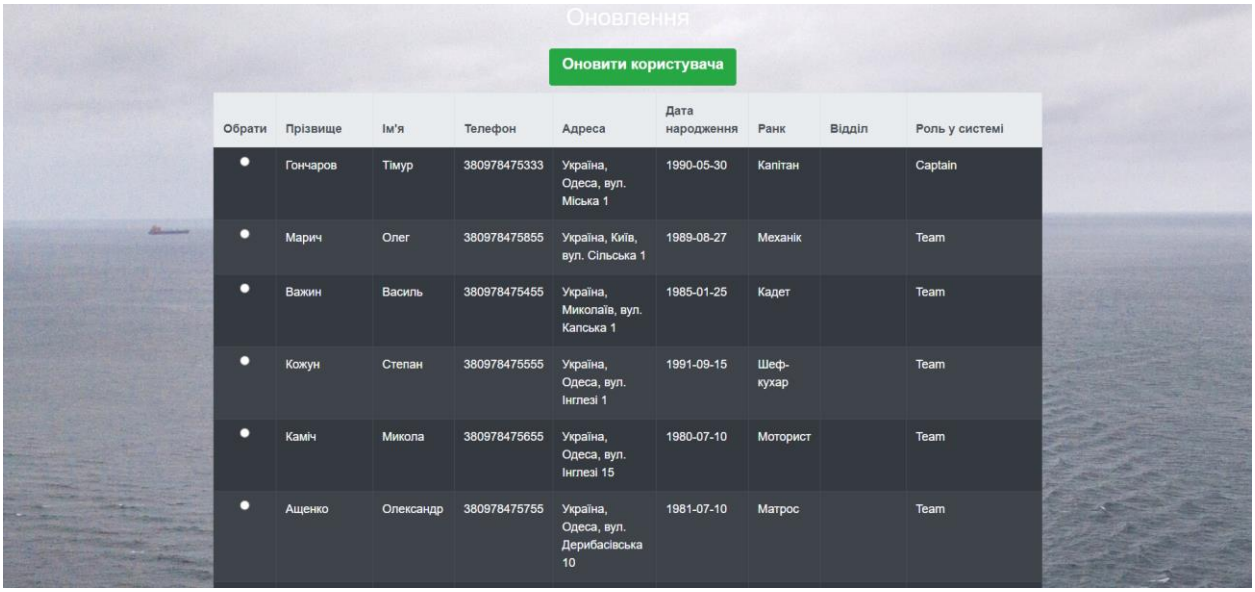

Рисунок 6.6 – Оновлення інформації про користувача

Та останнє для користувача адміністратор – це видалення робітників. Їх можливо видалити по-одному, а бо одразу декілька (рис. 6.7).

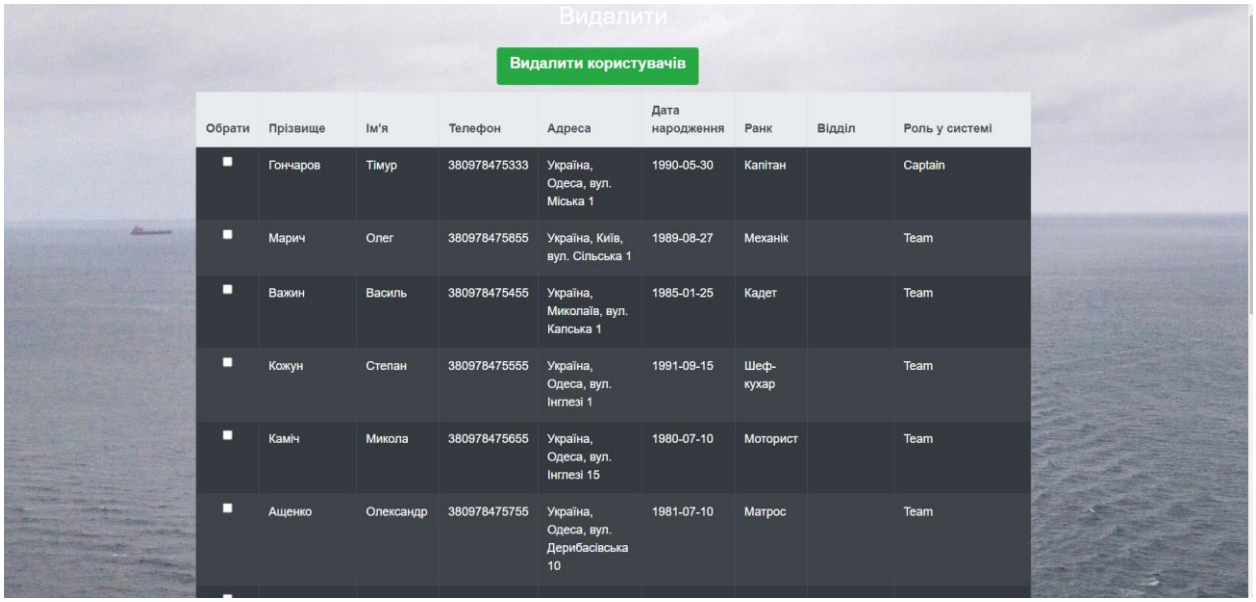

Рисунок 6.7 – Сторінка видалення робітників

6.2 Капітан судна

Після проходження авторизації як капітан, система надає певних привілей капітану та ми потрапляємо до профілю капітана (рис. 6.8).

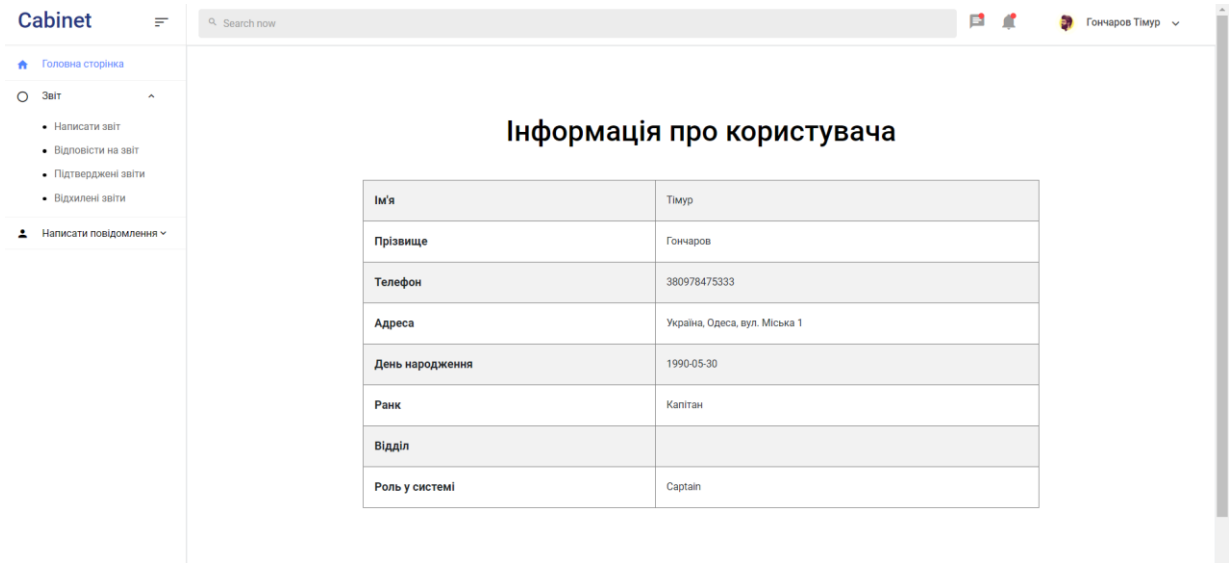

Рисунок 6.8 – Персональний кабінет капітана судна

Основна функція для капітана судна – це написання звітів до відділ – це можливо зробити після переходу у головному меню по посиланню – написати звіт (рис. 6.9).

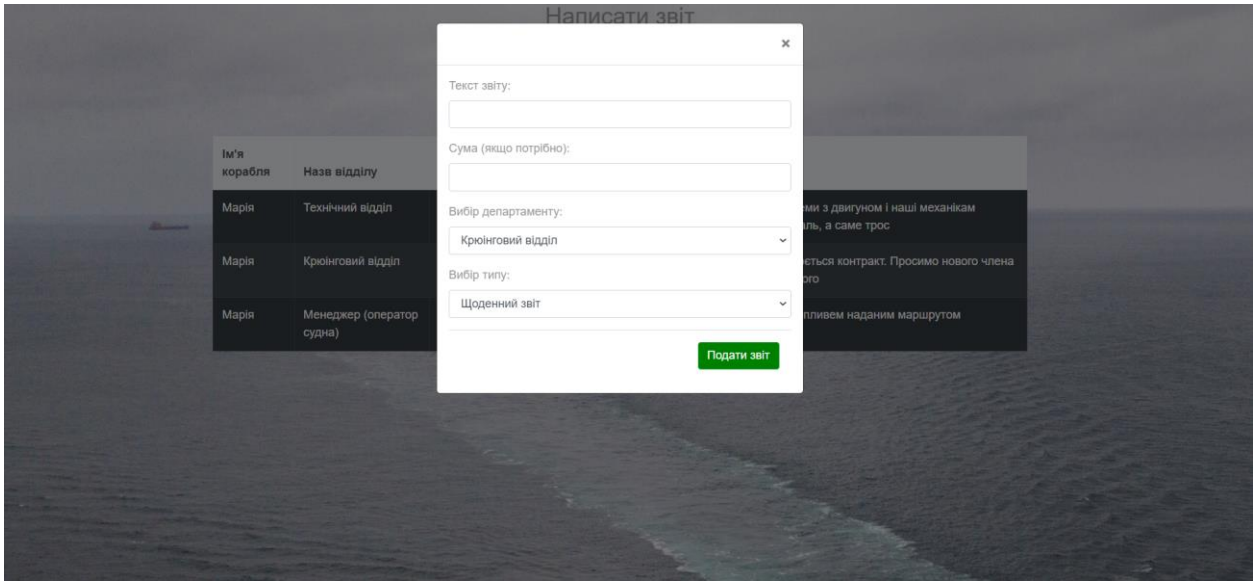

Рисунок 6.9 – Написати звіт до відділів

Коли капітану судна надсилають звіт з якогось відділ, то він може прийняти цей звіт та написати коментар або відмовитись прийняти, проте перед цим потрібно обрати, на який звіт відповідати буде (рис. 6.10).

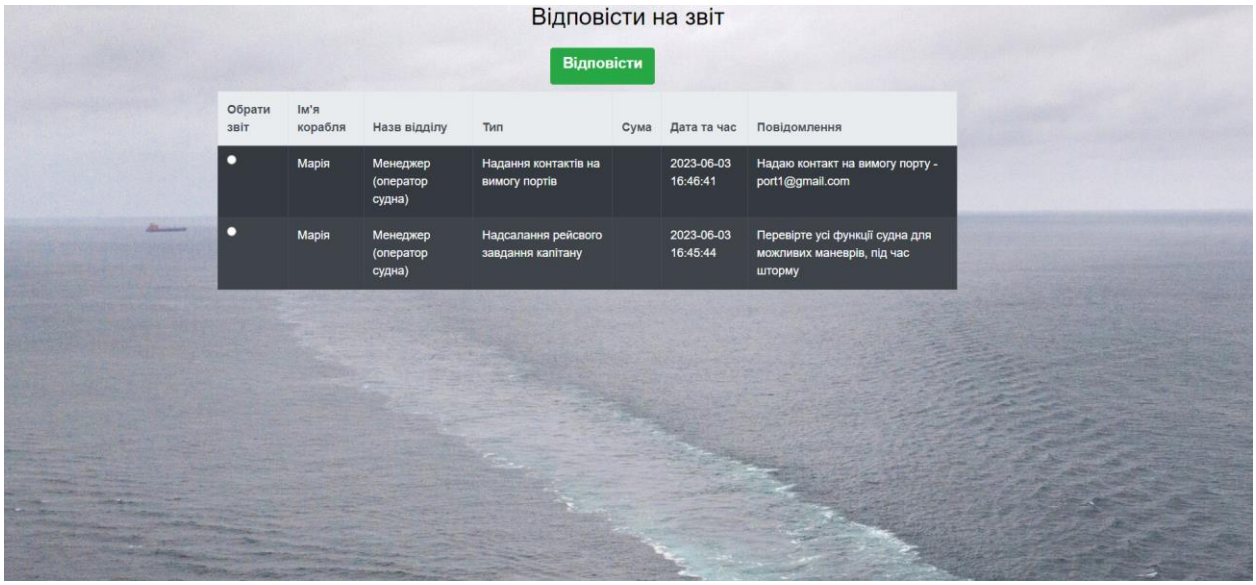

Рисунок 6.10 – Надати відповідь на звіт

Після того як капітан дав відповідь на запит підтвердженням, то ми можемо перевірити це – натиснувши на посилання: «підтверджені звіти» та переглянути їх (рис. 6.11).

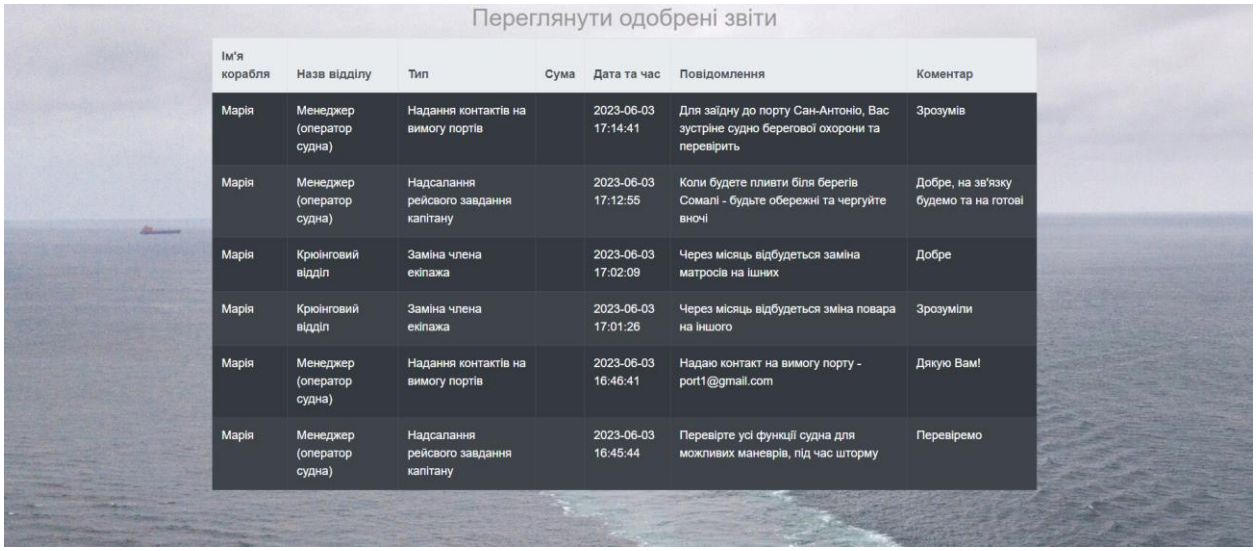

Рисунок 6.11 – Підтверджені звіти

Після того як капітан дав відповідь на запит відмовою, то ми можемо перевірити це – натиснувши на посилання: «відхилені звіти» та переглянути їх (рис. 6.12).

Рисунок 6.12 – Відхилені звіти

Далі розглянемо можливість вести неформальну бесіду між капітаном судна та усіма відділами у інформаційній системі. У персональному кабінеті зліва можливо побачити посилання «написати повідомлення» і потрібно обрати відділ кому писати (рис. 6.13).

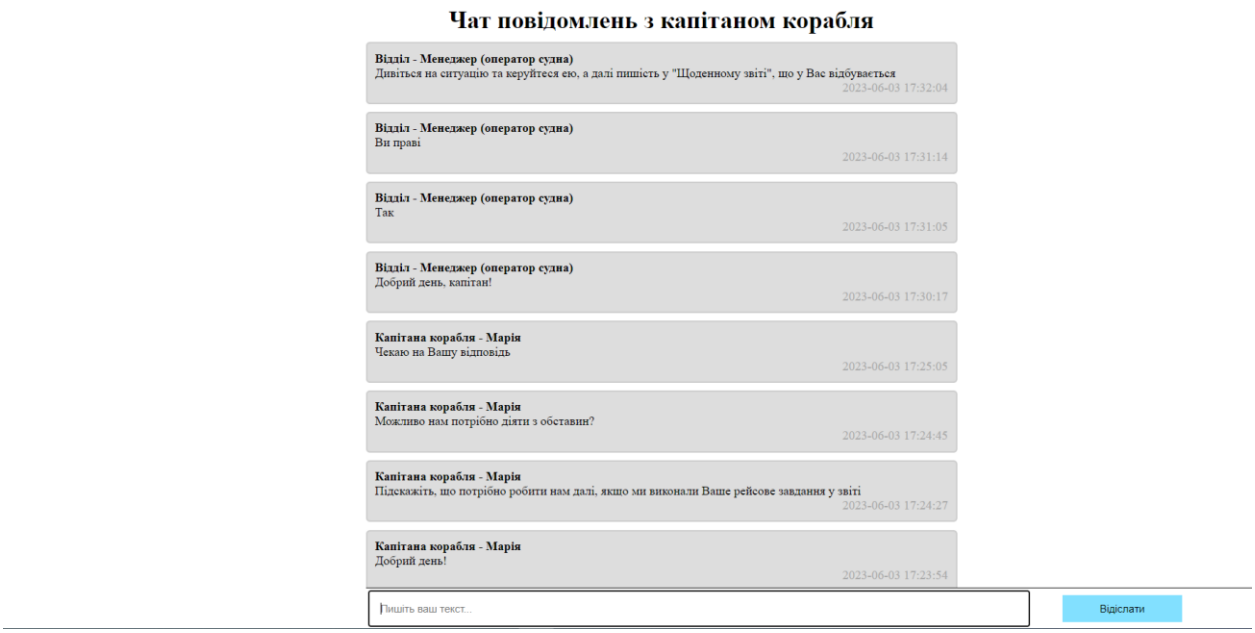

Рисунок 6.13 – Обмін повідомленнями

Для інших відділів ми побачимо таку саму ситуацію, тому не потрібно робити скріншоти з іншими відділами.

6.3 Менеджер (оператор судна)

Після проходження авторизації як менеджер, система надає певних привілей йому та ми потрапляємо до профілю менеджера (рис. 6.14).

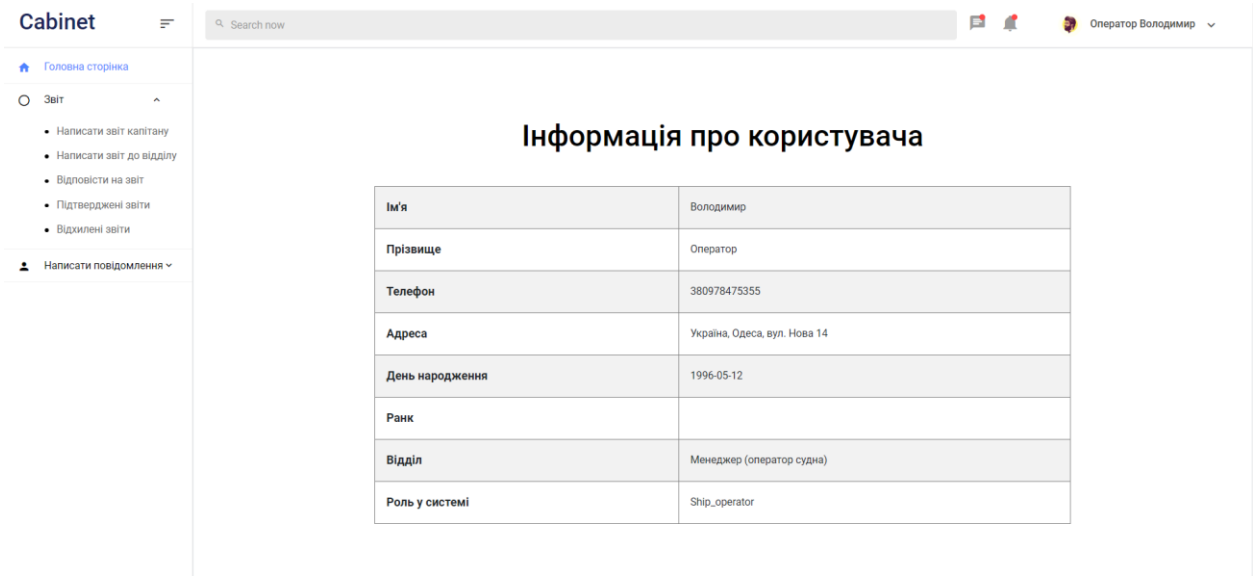

Рисунок 6.14 – Персональний кабінет менеджера

Менеджер може написати капітану судна та видати йому рейсове завдання або надати дані про порти для цього треба перейти по посиланню «Написати звіт капітану», де ми вже можемо написати (рис. 6.15).

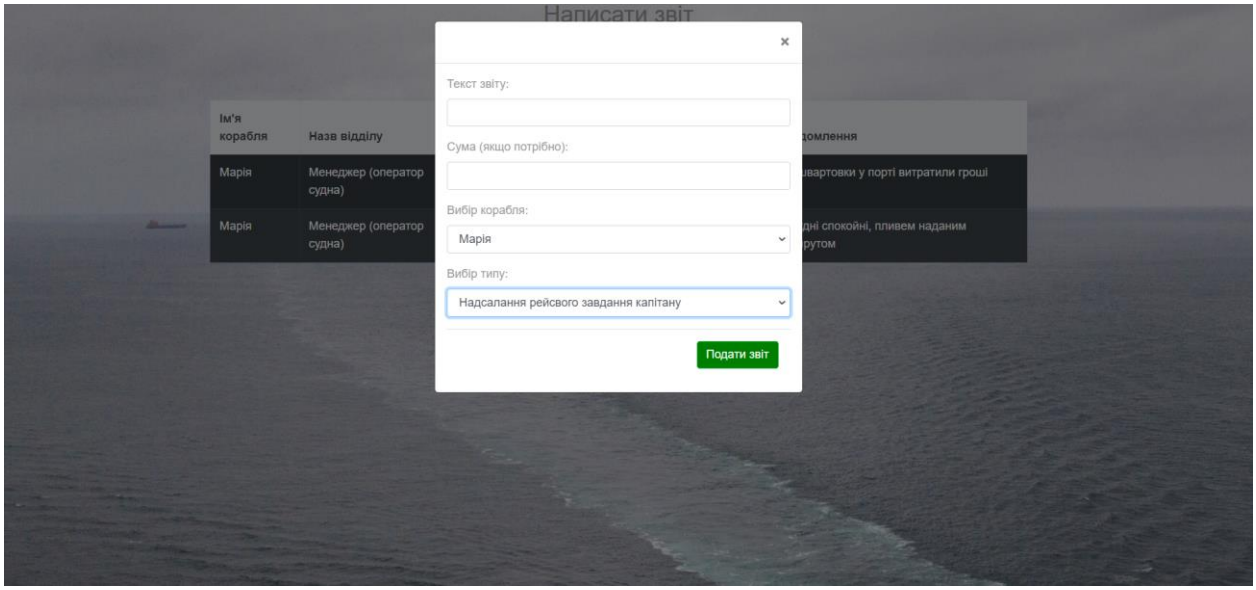

Рисунок 6.15 – Написання звіту до капітана від менеджера

Далі менеджер можете написати звіт до фінансового відділу, щодо незапланованих витрат капітаном (рис. 6.16).

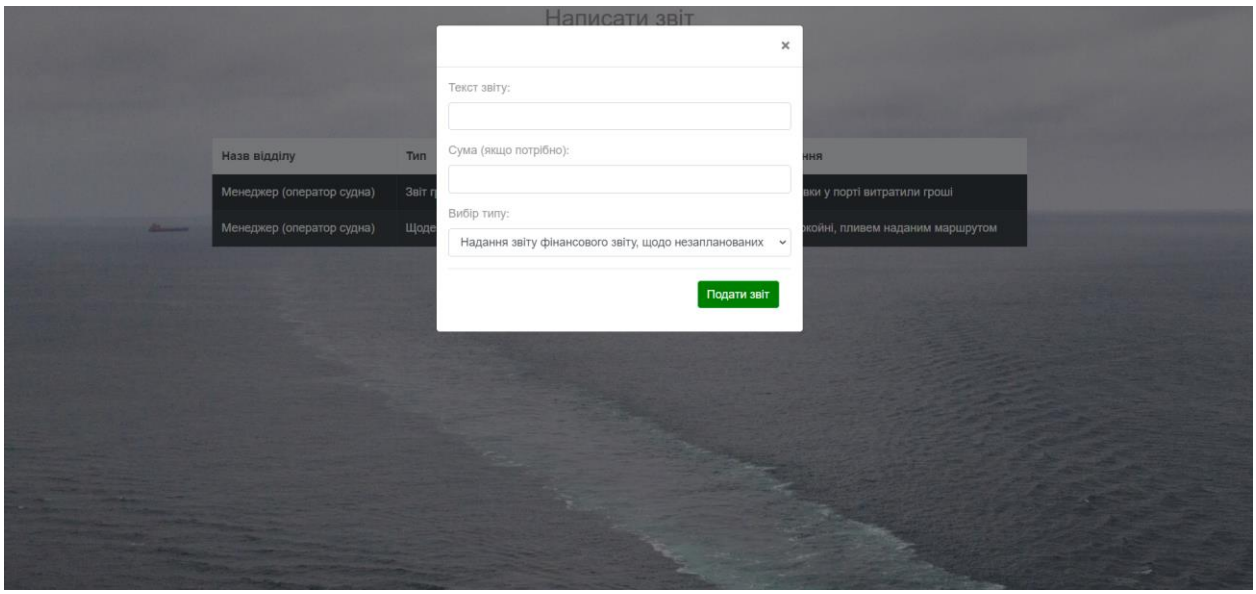

Рисунок 6.16 – Написання звіту до фінансового відділу

Перегляд підтверджених звітів капітаном або відділами іншими (рис. 6.17).

Рисунок 6.17 – Підтверджені звіти

Перегляд відхилених звітів капітаном або відділами іншими (рис. 6.18).

Рисунок 6.18 – Відхилені звіти

6.4 Технічний відділ

Після проходження авторизації як технічний відділ, система надає певних привілей йому та ми потрапляємо до профілю технічного відділу (рис. 6.19).

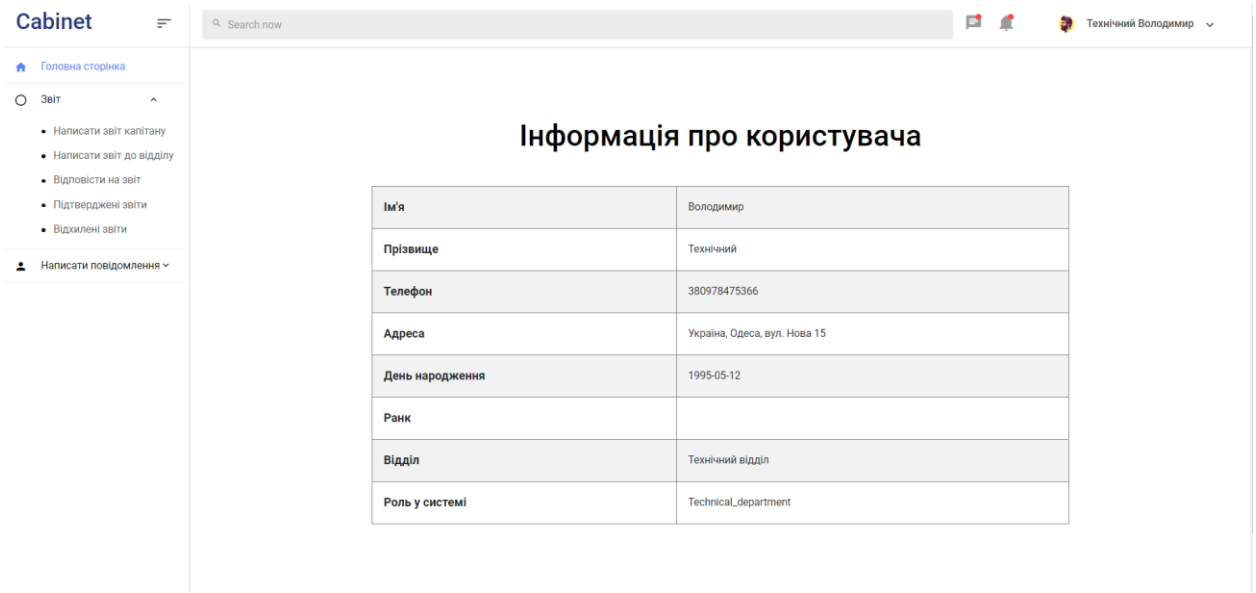

Рисунок 6.19 – Персональний кабінет технічного відділу

Технічний відділ може написати звіт капітану судна щодо технічної перевірки судна для цього треба перейти по посиланню «Написати звіт капітану», де ми вже можемо написати (рис. 6.20).

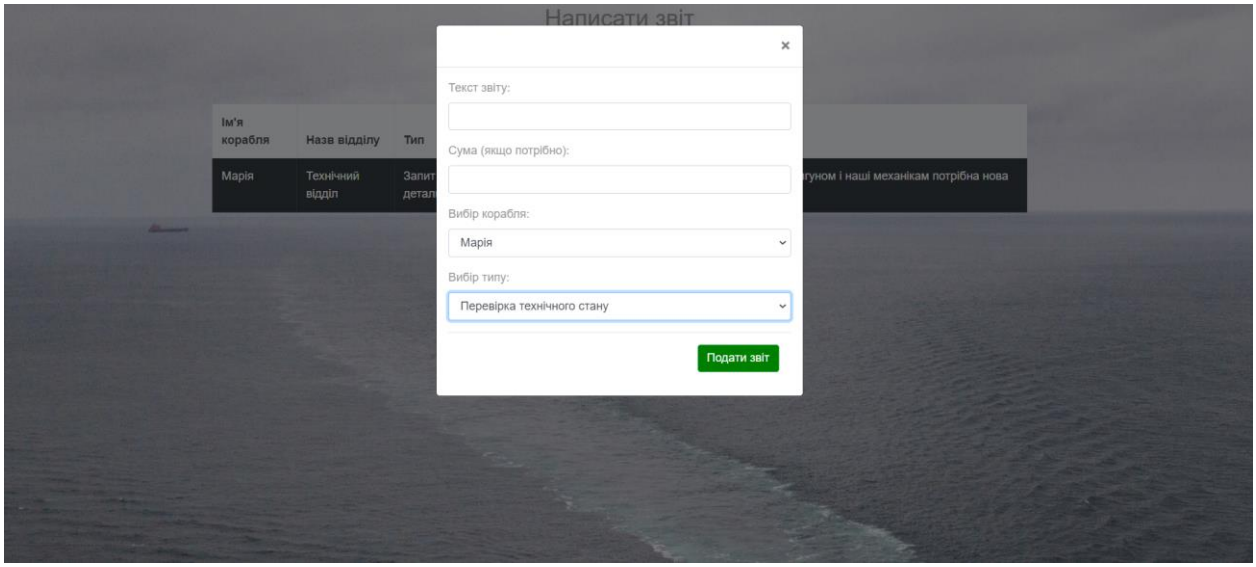

Рисунок 6.20 – Написання звіту до капітана від технічного відділу

Далі технічний відділ може написати звіт до фінансового відділу, щодо витрат та ремонту суден (рис. 6.21).

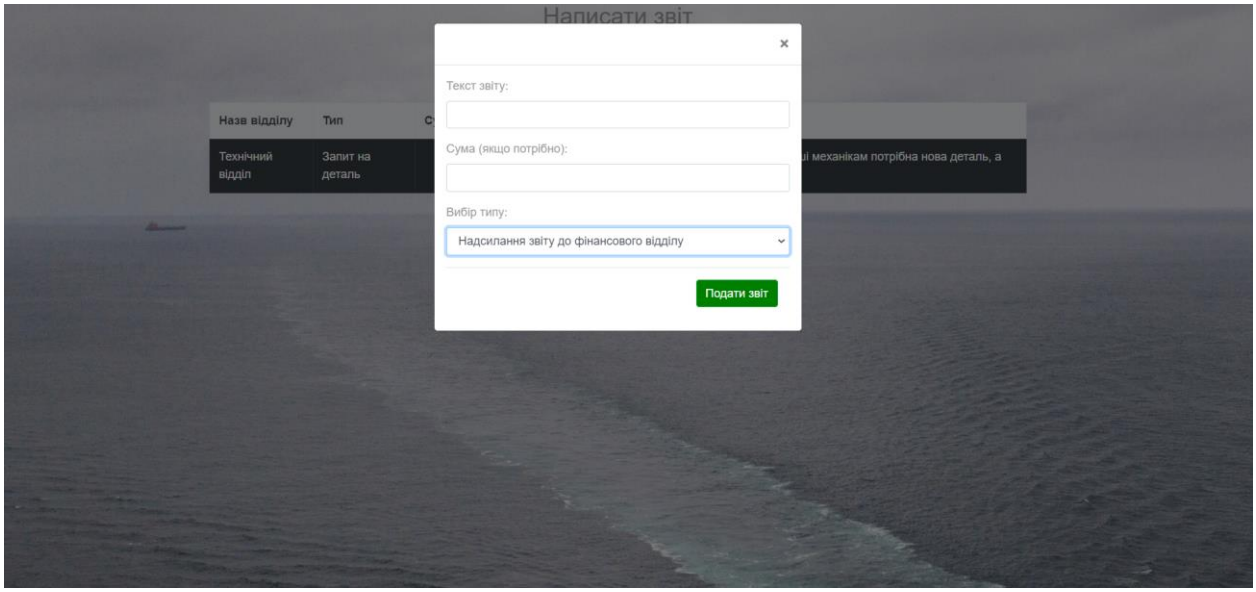

Рисунок 6.21 – Написання звіту до фінансового відділу від технічного

Перегляд підтверджених звітів капітаном або відділами іншими (рис. 6.22).

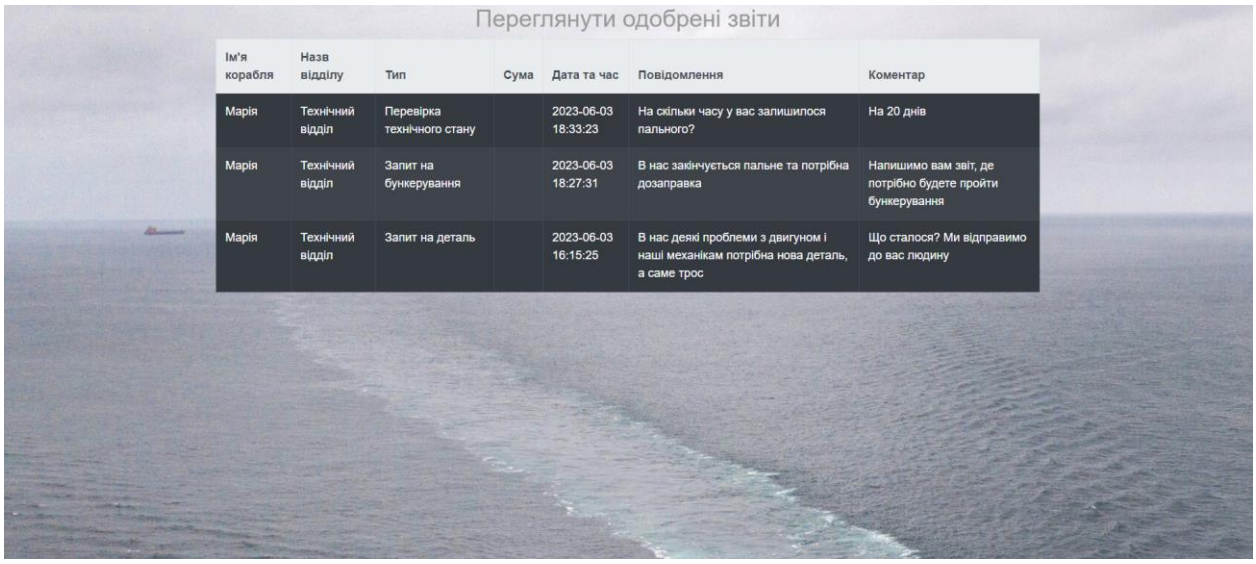

Рисунок 6.22 – Підтверджені звіти

Перегляд відхилених звітів капітаном або відділами іншими (рис. 6.23).

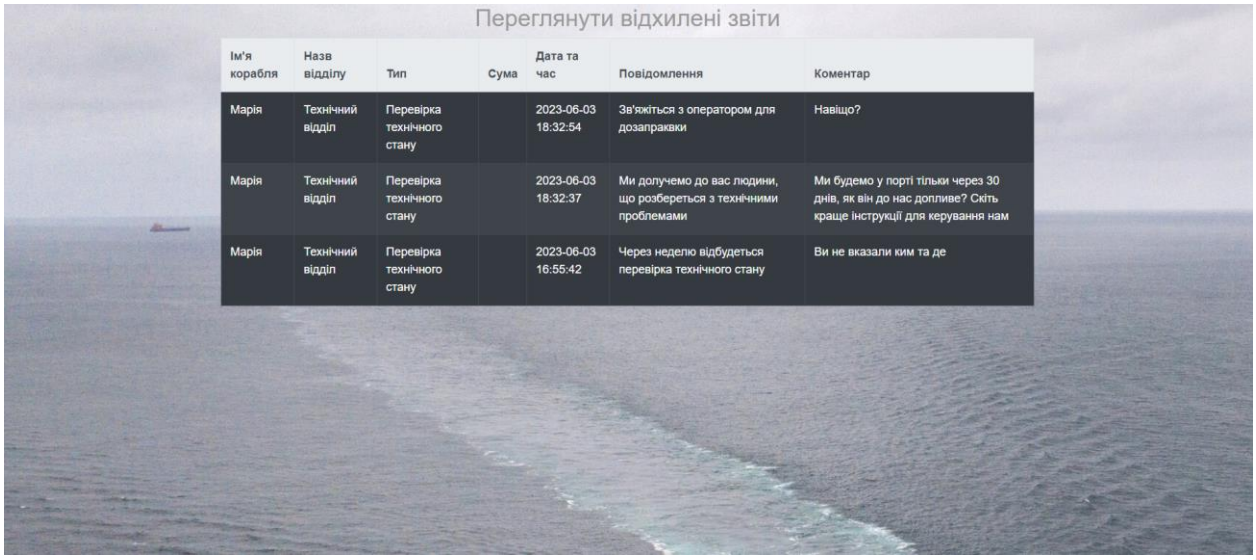

Рисунок 6.23 – Відхилені звіти

6.5 Крюїнговий відділ

Після проходження авторизації як крюїнговий відділ, система надає певних привілей йому та ми потрапляємо до профілю крюїнгового відділу (рис. 6.24).

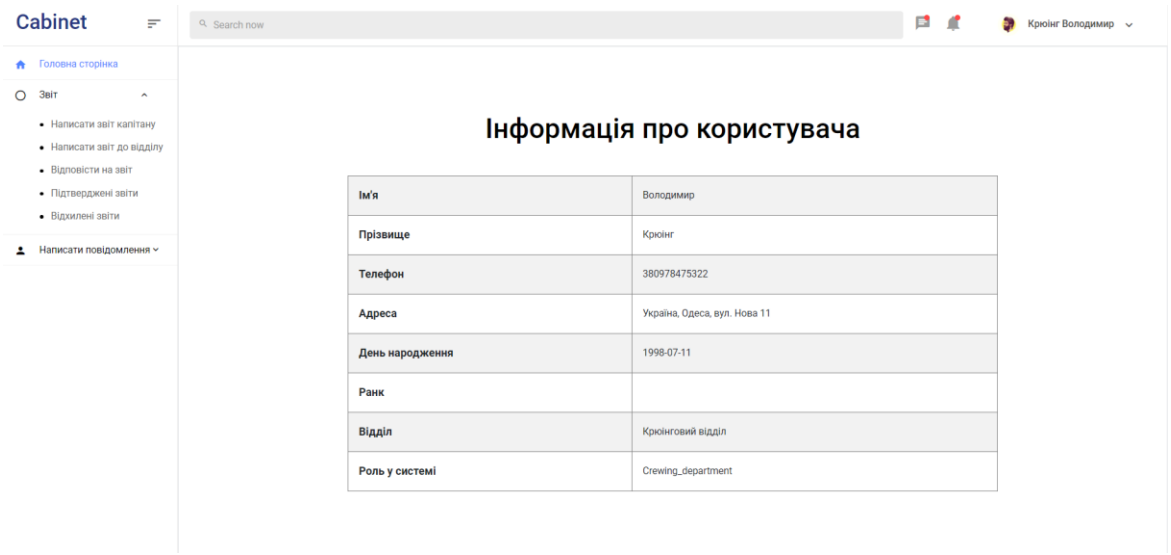

Рисунок 6.24 – Персональний кабінет крюїнгового відділу

Крюїнговий відділ може написати звіт капітану судна щодо заміни ленів екіпажу для цього треба перейти по посиланню «Написати звіт капітану», де ми вже можемо написати (рис. 6.25).

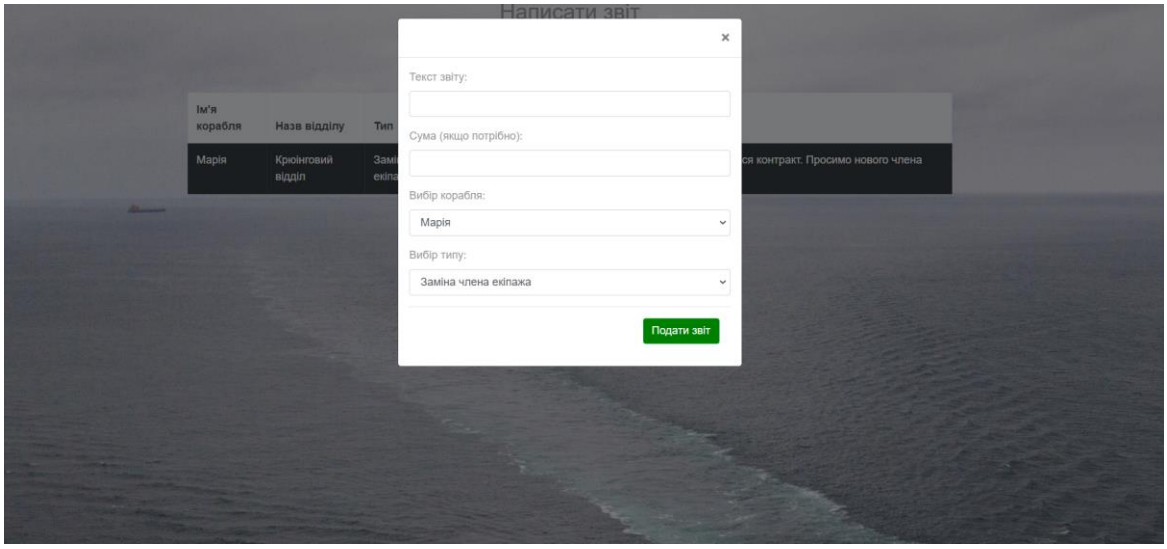

Рисунок 6.25 – Написання звіту до капітана від крюїнгового відділу

Далі крюїнговий відділ може написати звіт до фінансового відділу, щодо заробітної платні екіпажу суден (рис. 6.26).

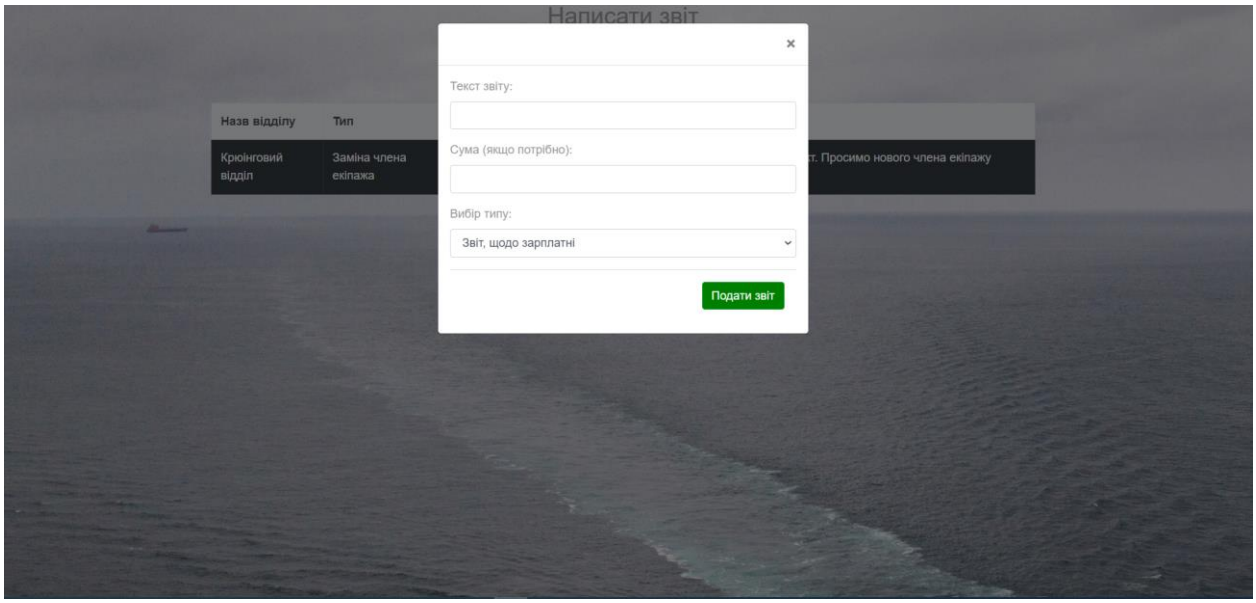

Рисунок 6.26 – Написання звіту до фінансового відділу від крюїнгового

Перегляд підтверджених звітів капітаном або відділами іншими (рис. 6.27).

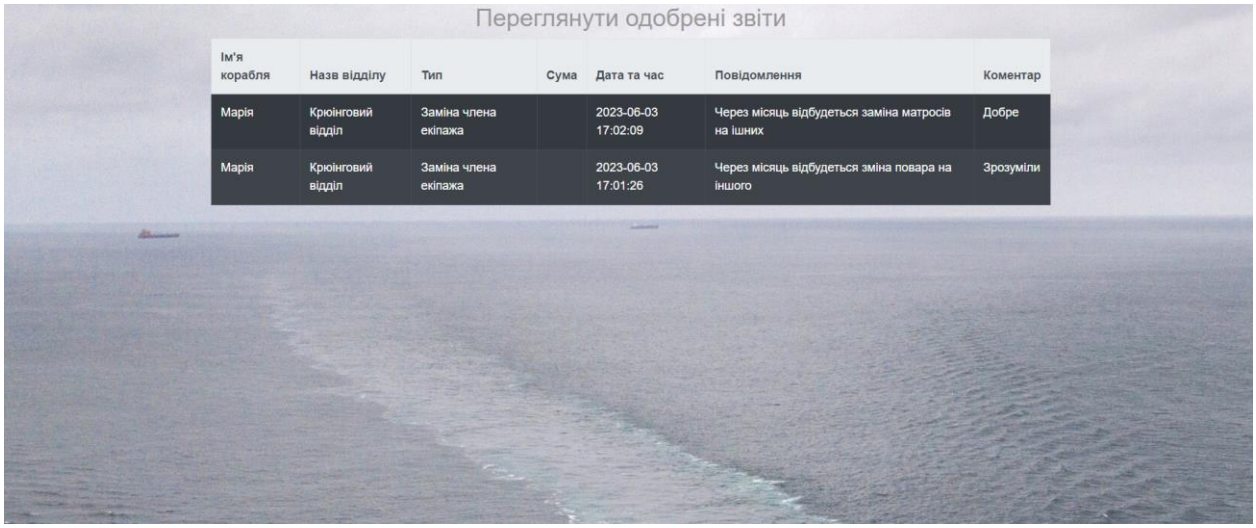

Рисунок 6.27 – Підтверджені звіти

Перегляд відхилених звітів капітаном або відділами іншими (рис. 6.28).

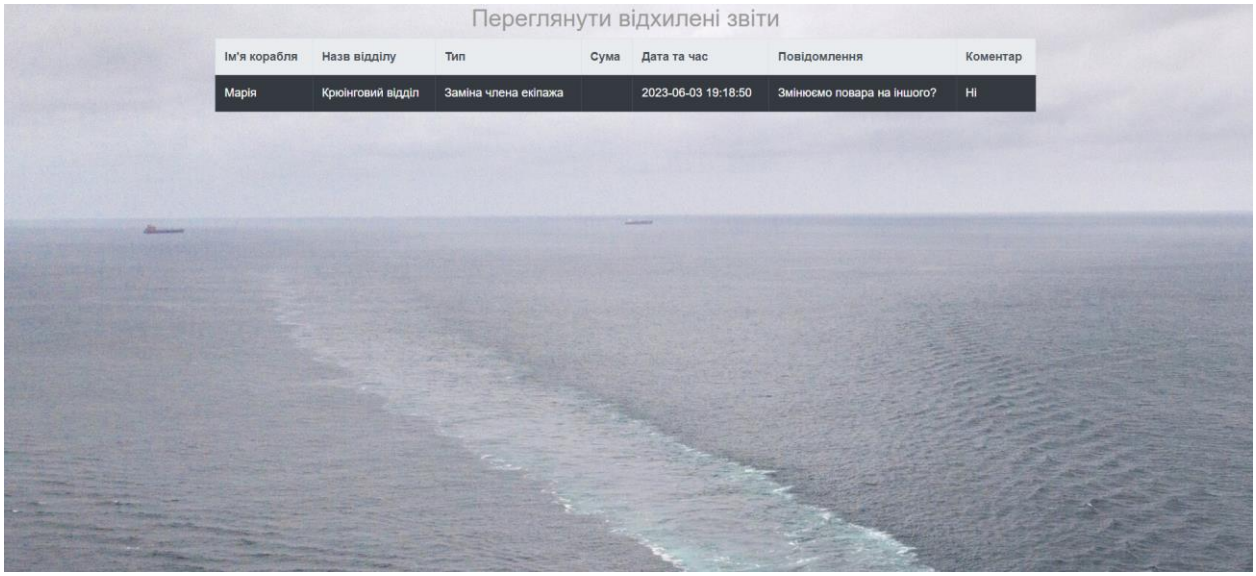

Рисунок 6.28 – Відхилені звіти

6.6 Фінансовий відділ

Після проходження авторизації як фінансовий відділ, система надає певних привілей йому та ми потрапляємо до профілю фінансового відділу (рис. 6.29).

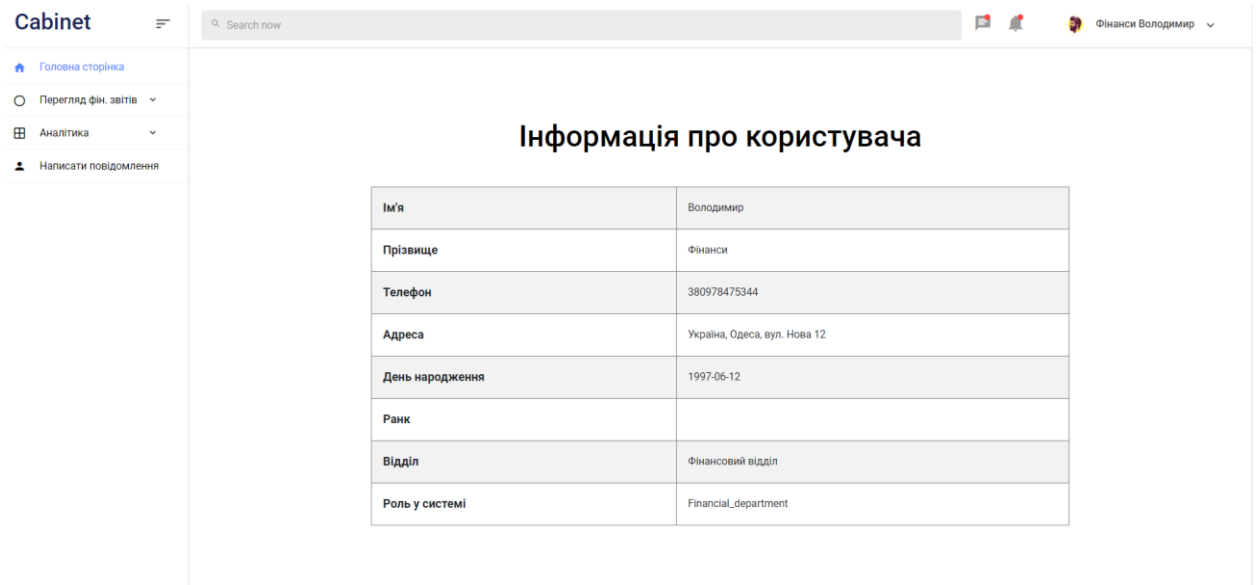

Рисунок 6.29 – Персональний кабінет фінансового відділу

Фінансовий відділ – відповідає за перегляд фінансовий звітів та їх підрахунок (рис. 6.30, 6.31, 6.32).

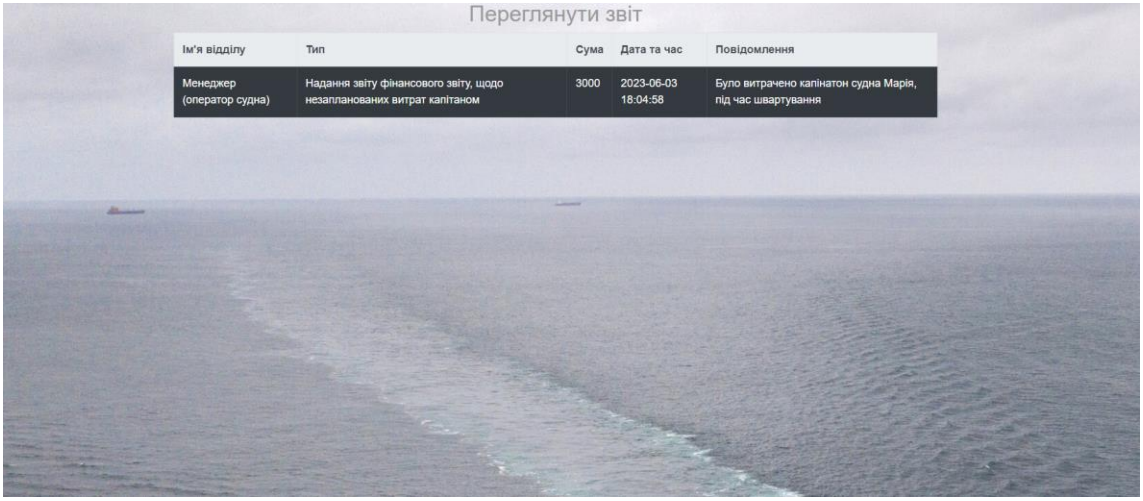

Рисунок 6.30 – Фінансовий звіт від менеджера судна

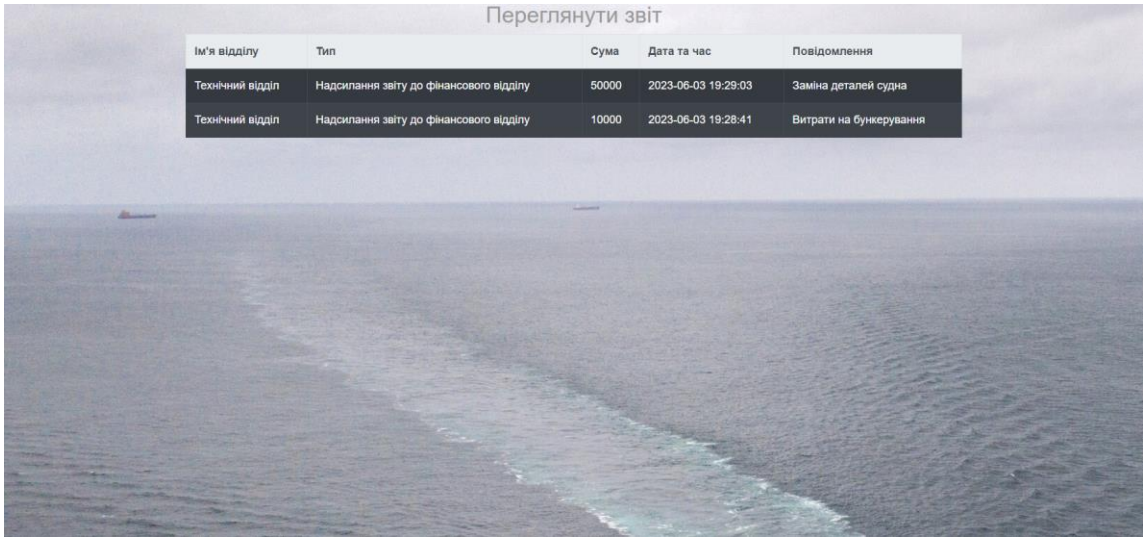

Рисунок 6.31 – Фінансовий звіт від технічного відділу

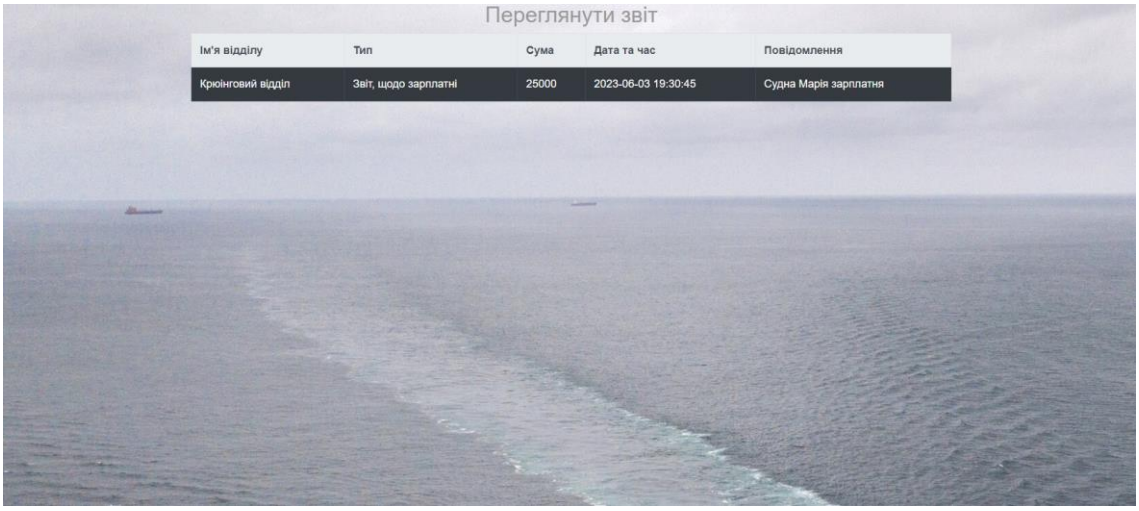

Рисунок 6.32 – Фінансовий звіт від крюїногового відділу

Далі – це аналітика, ми можемо на даний момент проводити загальний аналіз витрат та кількість їх (рис. 6.33).

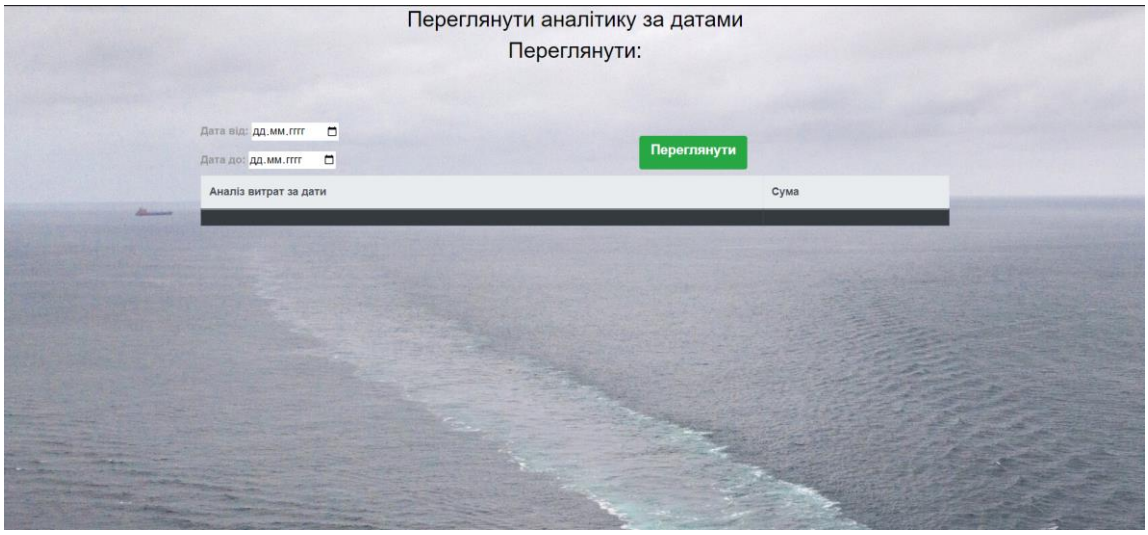

Рисунок 6.33 – Перегляд фінансових витрат відділами

ВИСНОВКИ

Представлена дипломна робота спрямована на проектування і реалізацію інформаційної системи фінансово-технічного управління для судновласницької компанії, за допомогою якої, морські компанії зможуть швидко та не застосовуючи зовнішні додатки зв’язуватися між суднами та відділами, щоб передати інформацію. Саме тому, робота є актуальна, щоб компанії перестали використовувати зовнішні застосунки.

Під час аналізу предметної області виконано аналіз та порівняння аналогів, визначено проблеми цієї області та необхідні задачі, які потребують користувачам інформаційної системи фінансово-технічного управління для судновласницької компанії.

В результаті проведеного аналізу були розроблені специфікації вимог до інформаційної системи, а саме: загальний погляд до вимог, функціональні вимоги, вимоги користувачів, атрибути якості програми та модель користувацького інтерфейсу.

Після визначення вимог для розробки інформаційної системи обраний та спроектований трирівневий шаблон проектування MVC, що можливо подивитися на діаграмі розгортання і спроектовано, а у подалі розроблено базу даних на основі СУБД PostgreSQL, яка реалізує реляційну модель даних. Також було спроектована і розроблена серверна частина ІС за допомогою Node.js з використанням фреймворку Express.js, який спрощує взаємодію при розробці сервера. Клієнтська частина, у свою чергу, розроблена за допомогою HTML, CSS та фреймворку вільних інструментів Bootstrap, який допомагає швидше розробляти інтерфейс сайту.

Подальший розвиток системи має перспективи в покращені клієнтського інтерфейсу, та реалізації додаткових функцій, що сприятимуть кращої візуалізації даних на екрані. Також потрібно буде реалізувати мобільний додаток, щоб не прив’язувати капітана судна до комп’ютера, а дати йому можливість бути з командою та робити звіти.

СПИСОК ВИКОРИСТАННИХ ДЖЕРЕЛ

1. Технологія морських перевезень. [Електронний ресурс]. – Режим доступу: https://www.sworld.com.ua/files/uchebnik/ucheb-ua-020.pdf
2. Як хакери можуть «зламувати» судна [Електронний ресурс]. – Режим доступу: https://usm.media/yak-hakeri-mozhut-zlamuvati-sudna-ta-do-chogo-tut-chatgpt/
3. Програмне забезпечення AMOS [Електронний ресурс]. – Режим доступу: https://www.spectec.net/amos-maintenance-and-procurement
4. Програмне забезпечення ShipManager [Електронний ресурс]. – Режим доступу: https://www.dnv.com/services/ship-management-operations-and-ship-design-software-14032
5. Програмне забезпечення Navis N4 [Електронний ресурс]. – Режим доступу: https://www.navis.com
6. Онлайн інструмент для моделювання діаграм ієрархії сторінок [Електронний ресурс]. – Режим доступу: https://www.flowmapp.com/
7. Що таке MVC? Робота та переваги [Електронний ресурс]. – Режим доступу: https://uk.education-wiki.com/3886982-what-is-mvc
8. Повне розуміння діаграми компонентів [Електронний ресурс]. – Режим доступу: https://www.mindonmap.com/uk/blog/uml-component-diagram/
9. Документація для PostgreSQL [Електронний ресурс]. – Режим доступу: https://www.postgresql.org/docs/
10. Документація для Node.js [Електронний ресурс]. – Режим доступу: https://nodejs.org/en/docs
11. Мова програмування JavaScript [Електронний ресурс]. – Режим доступу: https://developer.mozilla.org/ru/docs/Web/JavaScript
12. Bootstrap documentation by Bootstrap 4 [Електронний ресурс]. – Режим доступу: https://getbootstrap.com/docs/4.3/getting-started/introduction/
13. Організація баз даних: Конспект лекцій // Є.В. Малахов / Одеса: Одес. нац. ун-т ім. І. І. Мечникова, 2021. – 196 с.

ДОДАТОК А

Перелік копій демонстраційного матеріалу

Рисунок А.1 – Титульна сторінка презентації

Рисунок А.2 – Актуальність теми

Рисунок А.3 – Мета та постановк задачі

Рисунок А.4 – Завдання та вимоги до користувачів

Пошук конкурентів та визначення їх переваг та недоліків

5

Назва додатку	Переваги	Недоліки
AMOS	- підтримує ефективне управління обслуговуванням та ремонтом кораблів; - забезпечує контроль та оптимізацію процесів запасів, контролю якості та обліку витрата	- обмежена функціональність в порівнянні з іншими системами управління; - може вимагати тривалого часу та ресурсів для впровадження та навчання персоналу
ShipManager	- забезпечує зберігання великої кількості даних про обладнання та ремонтні роботи; - автоматизує та оптимізує процеси планування, виконання та відстеження вантажоперевезень;	- вимагає підтримки технічної інфраструктури для безперебійної роботи; - може потребувати інтеграції з іншими системами для повноцінного функціонування
Navis N4	- надає аналітичні можливості та звітність для прийняття обґрунтованих рішень; - забезпечує централізоване управління портовими операціями та судновими стоянками	- вимагає навчання та ознайомлення персоналу з системою; - може бути дещо складним у використанні для непідготовленого персоналу

Таблиця 1 – Аналіз конкурентів

Рисунок А.5 – Пошук конкурентів та визначення їх переваг а недоліків

Проектування діаграми ієрархії сторінок

6

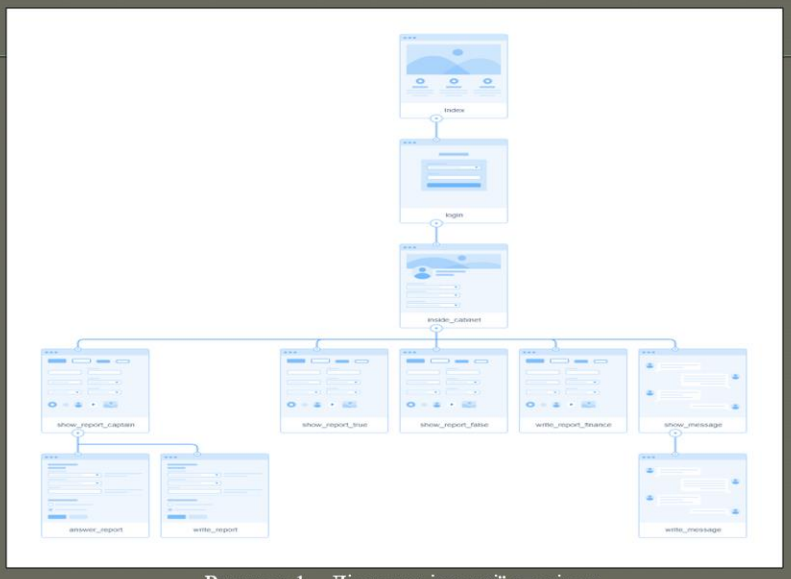

Рисунок 1 – Діаграми ієрархії сторінок

Рисунок А.6 – Проектування діаграми ієрархії сторінок

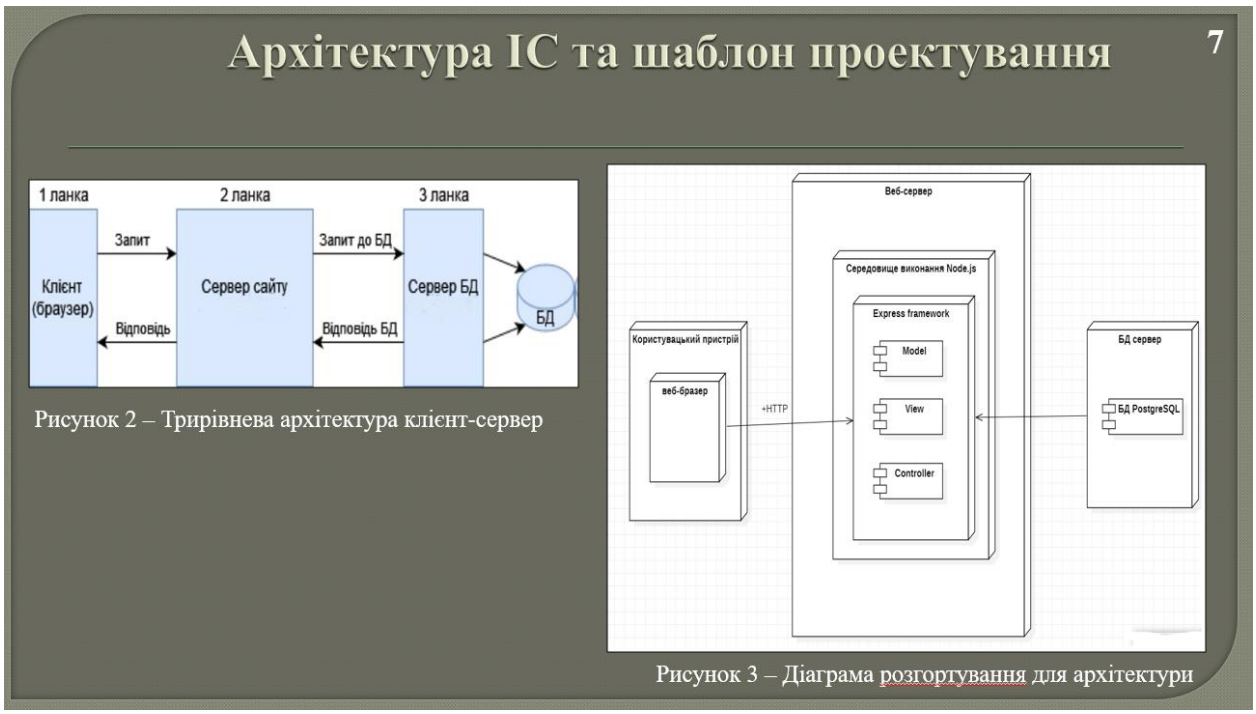

Рисунок А.7 – Проектування діаграми ієрархії сторінок

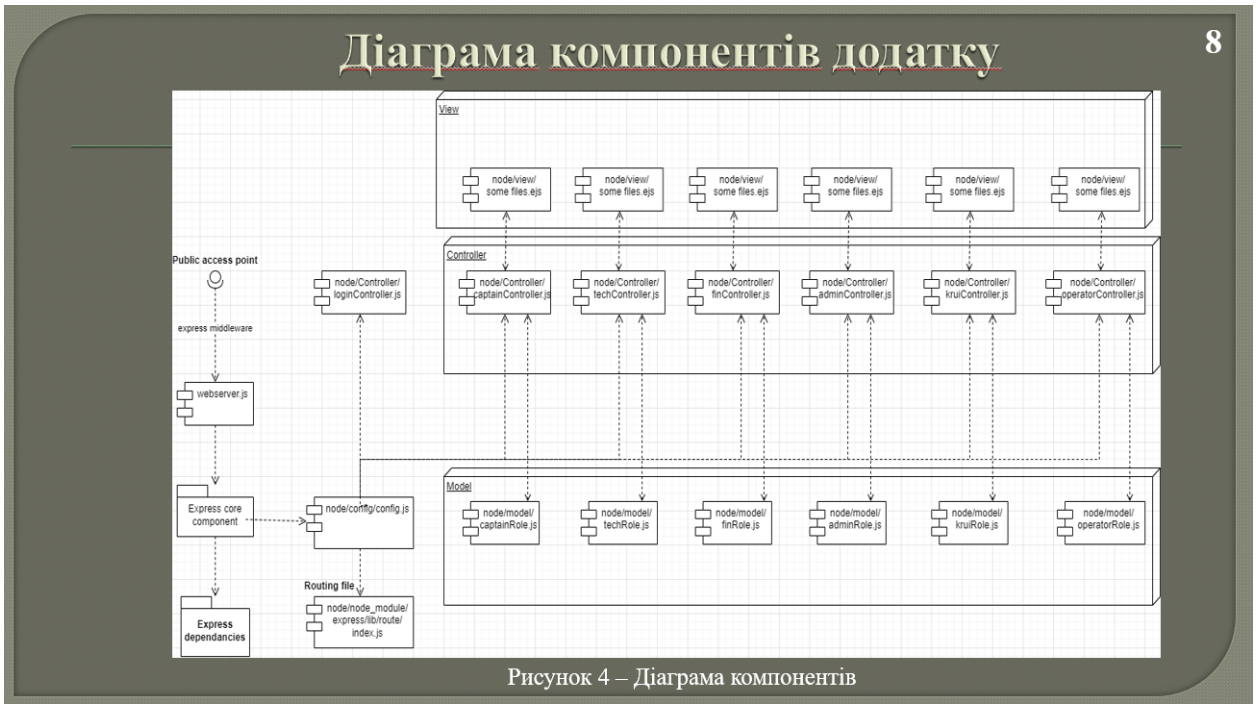

Рисунок А.8 – Діаграма компонентів додатку

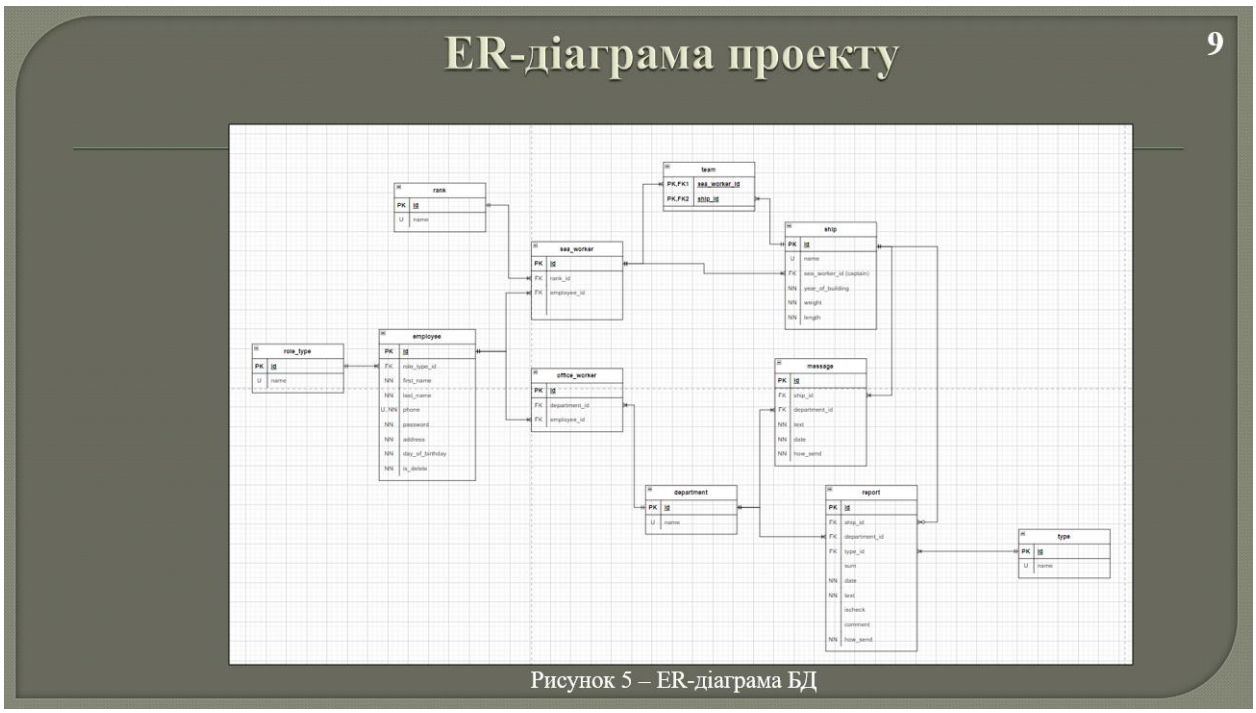

Рисунок А.9 – ER-діаграма проекту

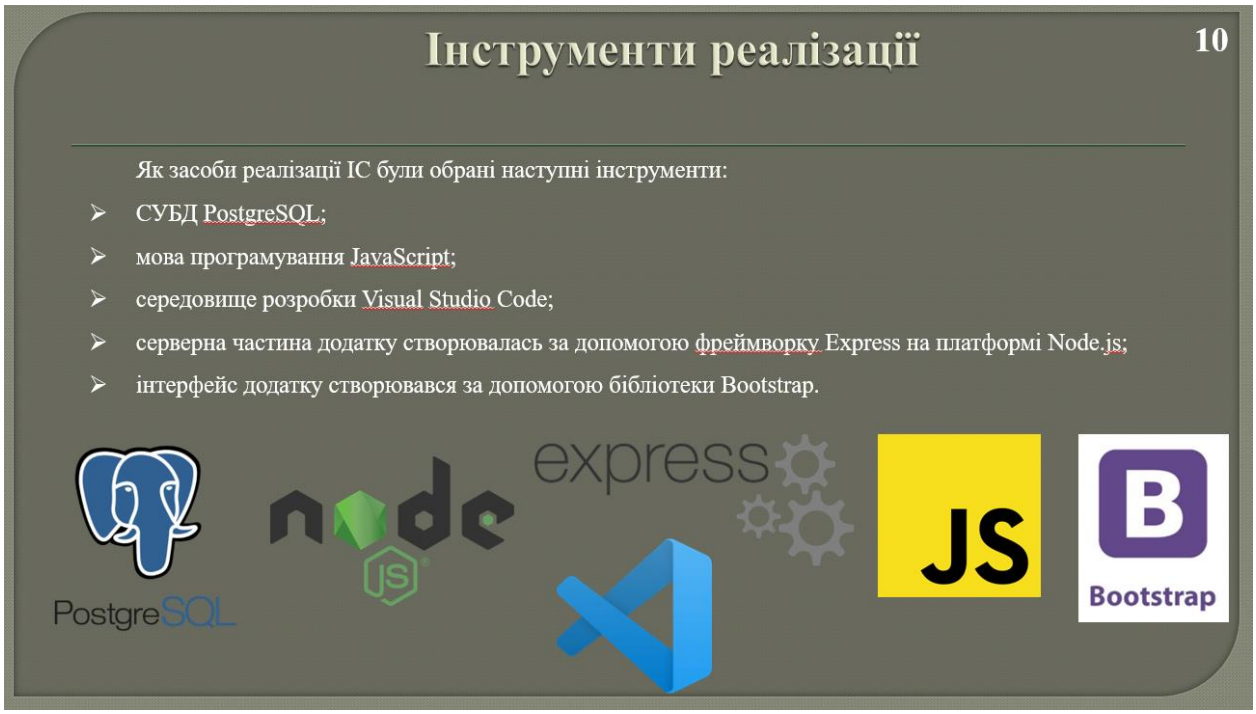

Рисунок А.10 – Інструменти реалізації

11

Передача інформації

Для передачі повідомлень між кораблем та відділами було обрана система зв'язку – Starlink, яку легко налаштувати та використовувати у будь-якій точці земного шару.

Рисунок 6 – Starlink, інтернет для зв'язок

Рисунок А.11 – Передача інформації

12

Розробка бази даних

Була створена БД з такими сутностями що включають 11 таблиць 10 представлень 15 функцій для зберігання переданої інформації між відділами та капітаном у системі.

Таблиць	Представлень	Функцій
11	10	15

Таблиця 2 – Сутності БД

Рисунок А.12 – Розробка бази даних

13

Безпека інформаційної системи

- Шифрування паролів користувачів за допомогою бібліотеки **pgcrypto**;
- У **pgcrypto** обрано алгоритм шифрування **bf** (на рисунку перераховані можливі алгоритми);
- Безпечний доступ до БД за допомогою ролей, що були створені;
- Проведено тестування програми.

Algorithm	Max Password Length	Adaptive?	Salt Bits	Output Length	Description
bf	72	yes	128	60	Blowfish-based, variant 2a
md5	unlimited	no	48	34	MD5-based crypt
xdes	8	yes	24	20	Extended DES
des	8	no	12	13	Original UNIX crypt

Рисунок 7 – Різновиди алгоритмів шифрування

Рисунок А.13 – Безпека інформаційної системи

14

Інструменти користувача

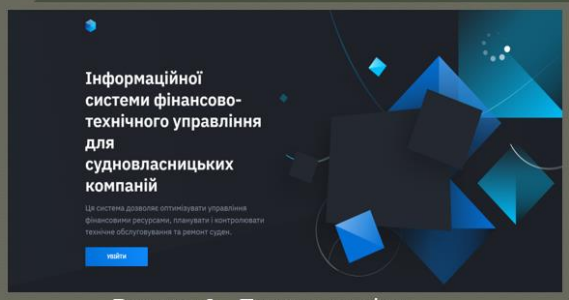

Рисунок 8 – Головна сторінка

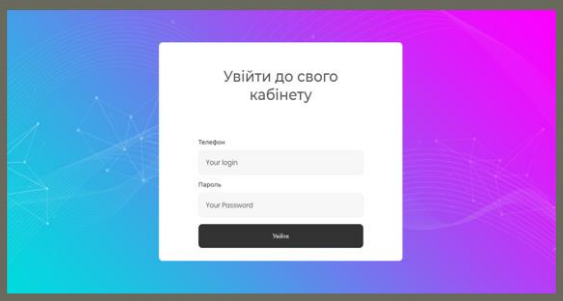

Рисунок 9 – Сторінка авторизації

Рисунок А.14 – Інструменти користувача

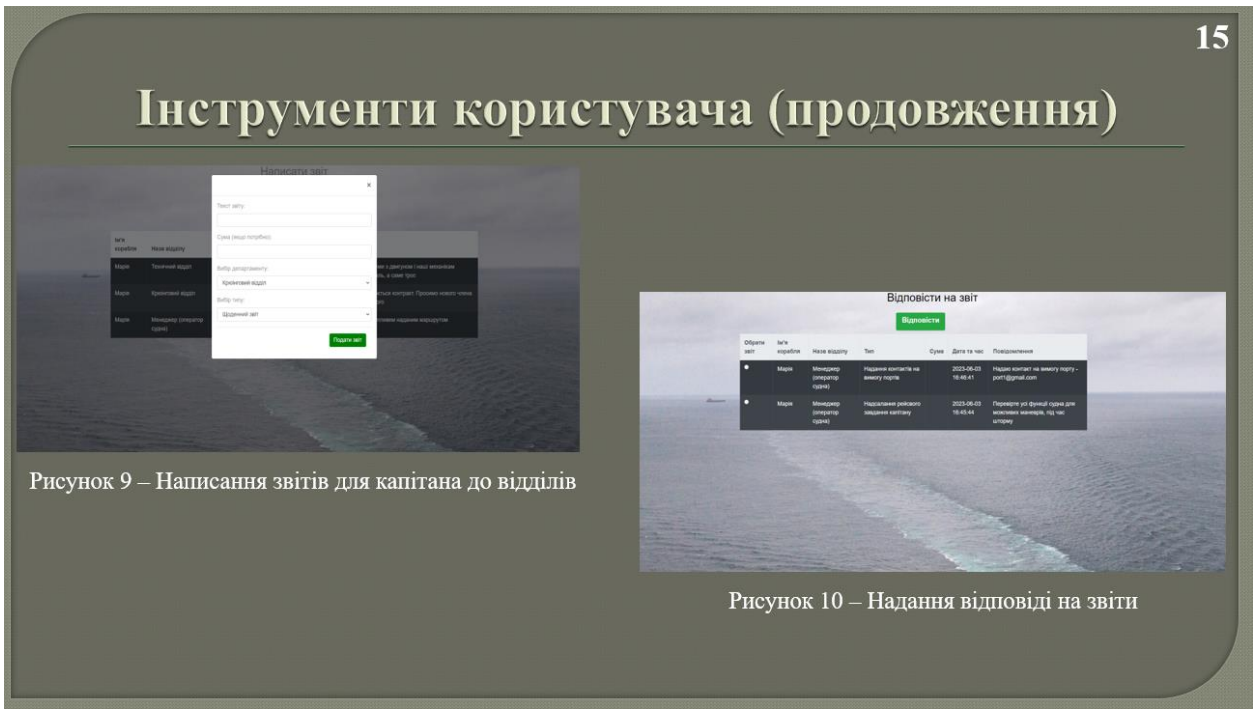

Рисунок 9 – Написання звітів для капітана до відділів

Рисунок 10 – Надання відповіді на звіти

Рисунок А.15 – Інструменти користувача (продовження)

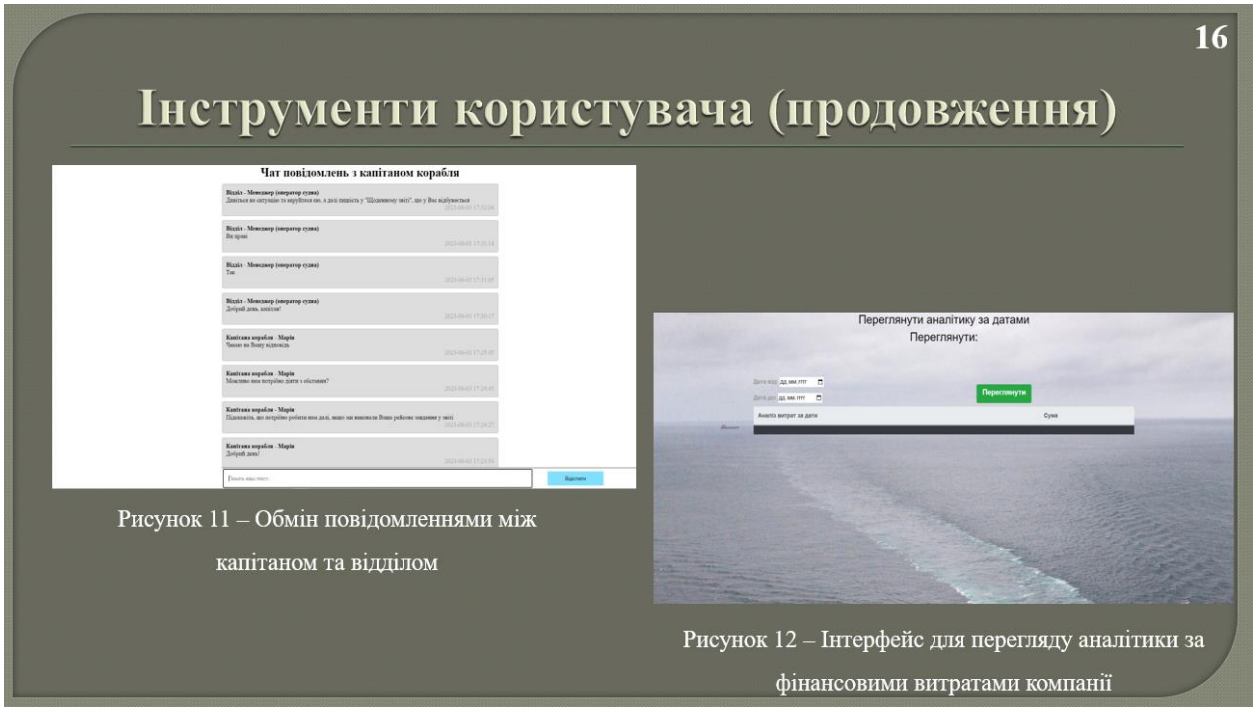

Рисунок 11 – Обмін повідомленнями між капітаном та відділом

Рисунок 12 – Інтерфейс для перегляду аналітики за фінансовими витратами компанії

Рисунок А.16 – Інструменти користувача (продовження)

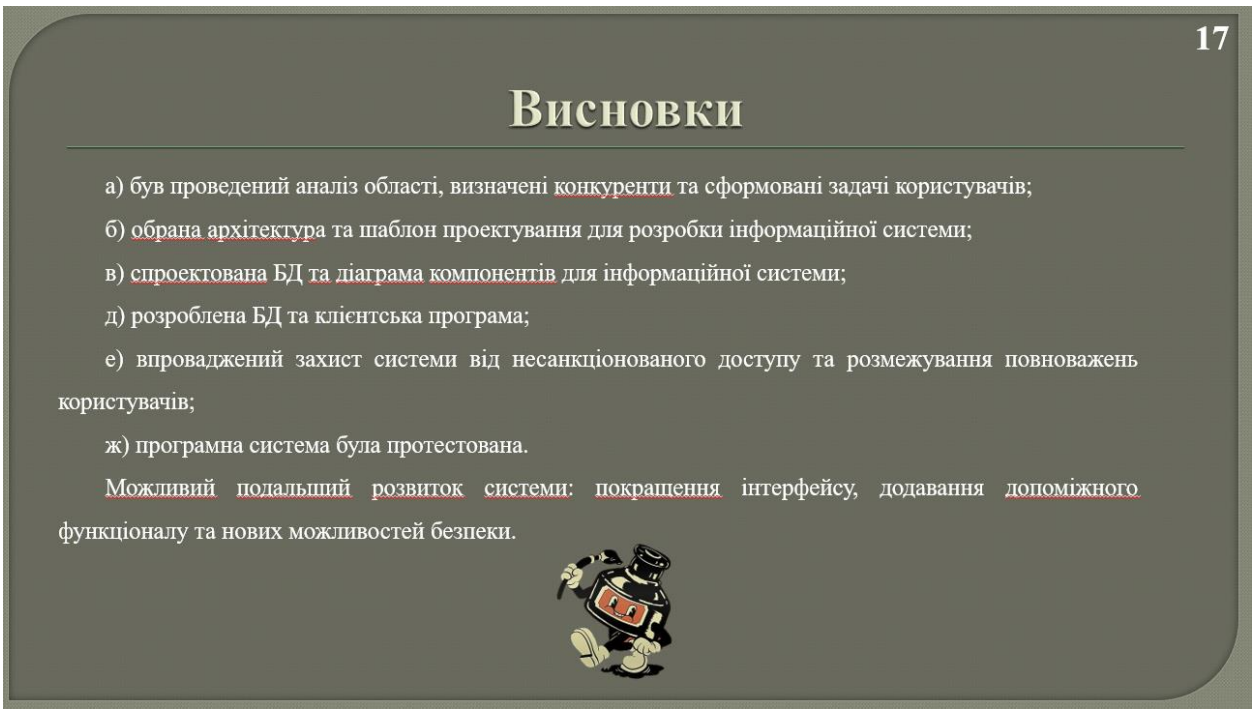

Рисунок А.17 – Висновки

ДОДАТОК Б

Перелік діаграм

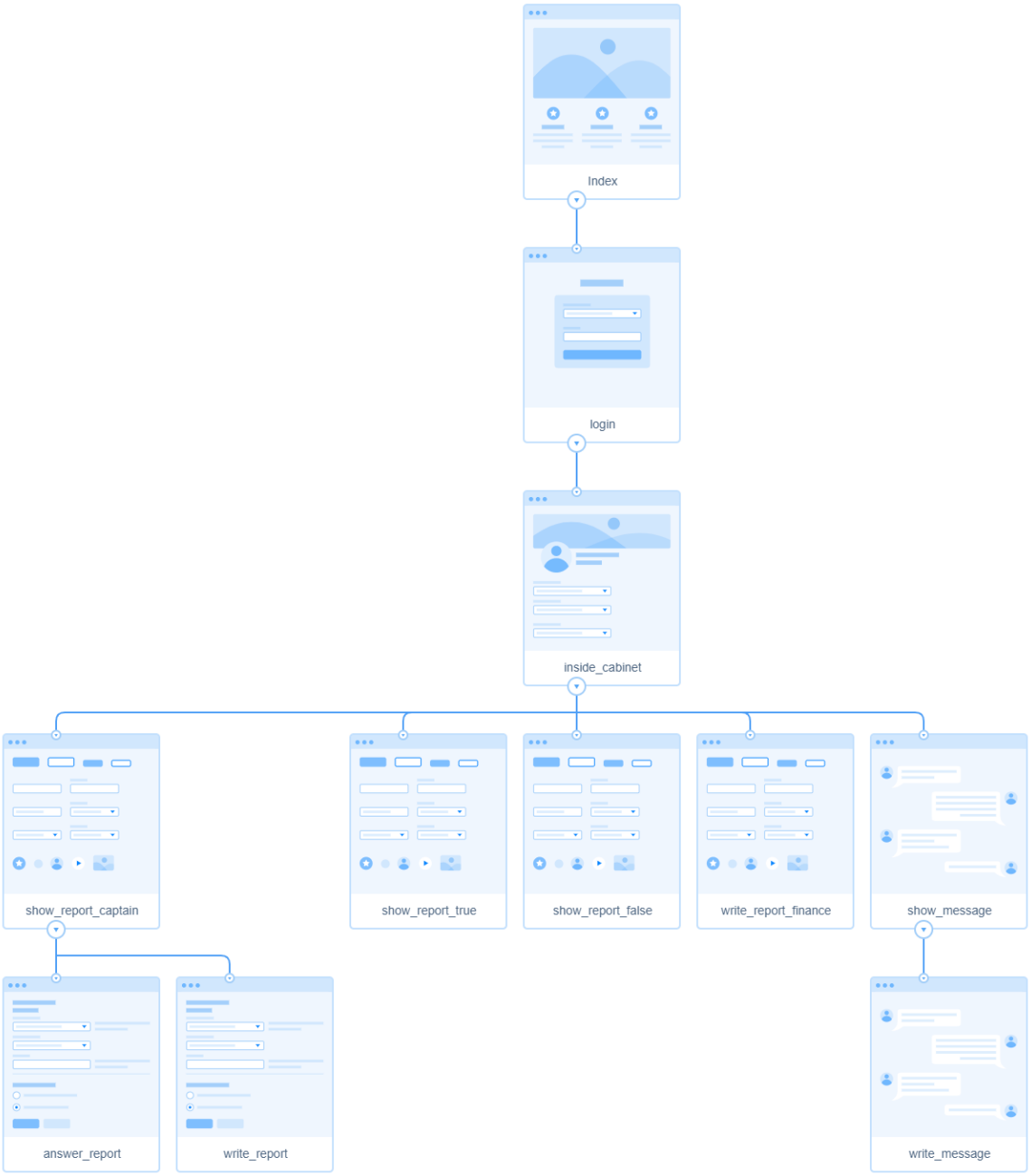

Рисунок Б.1 – Діаграма користувацького інтерфейсу для технічного та крюїнгового відділів та менеджерів суден

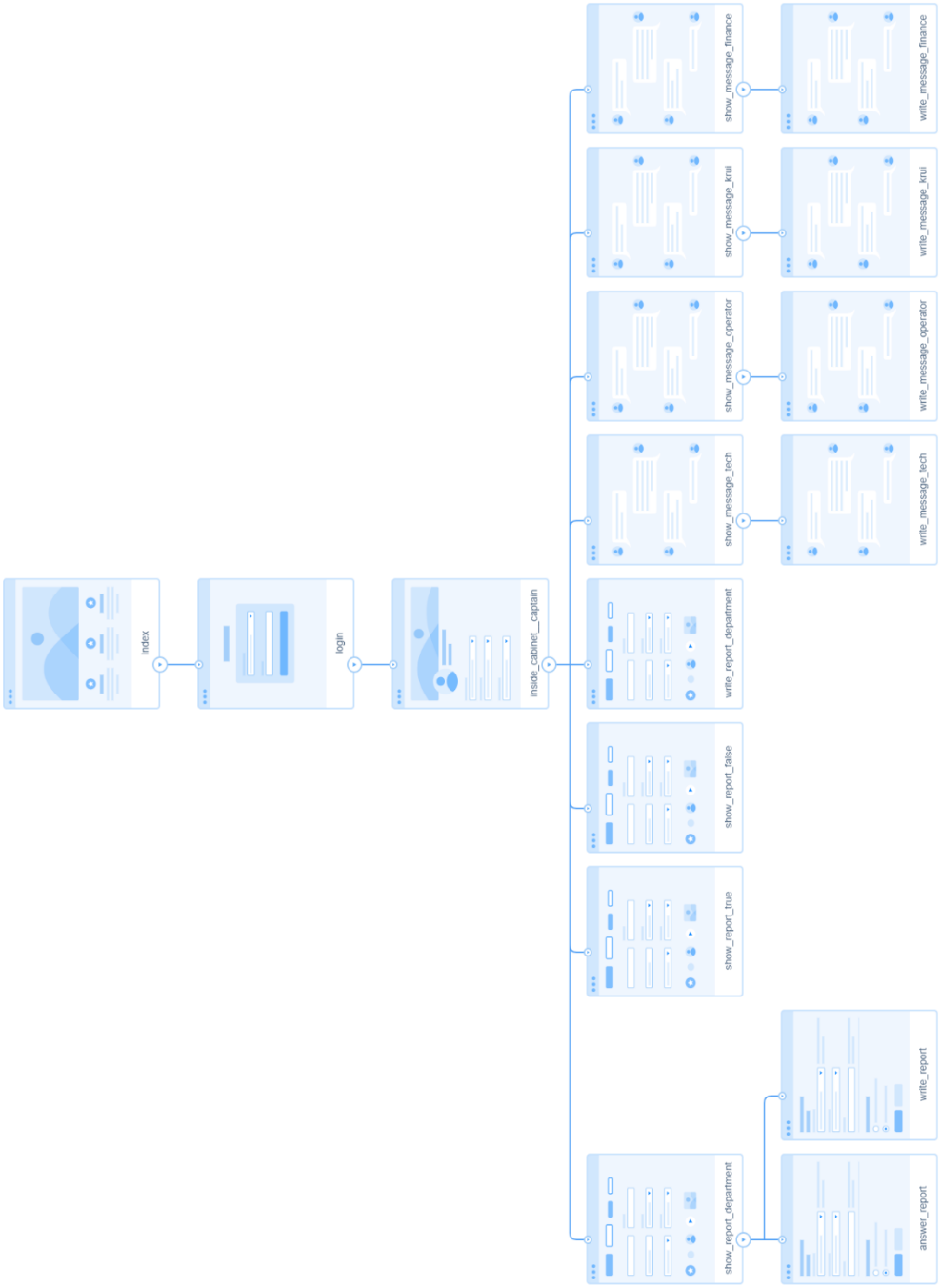

Рисунок Б.2 – Діаграма користувацького інтерфейсу для капітану судна

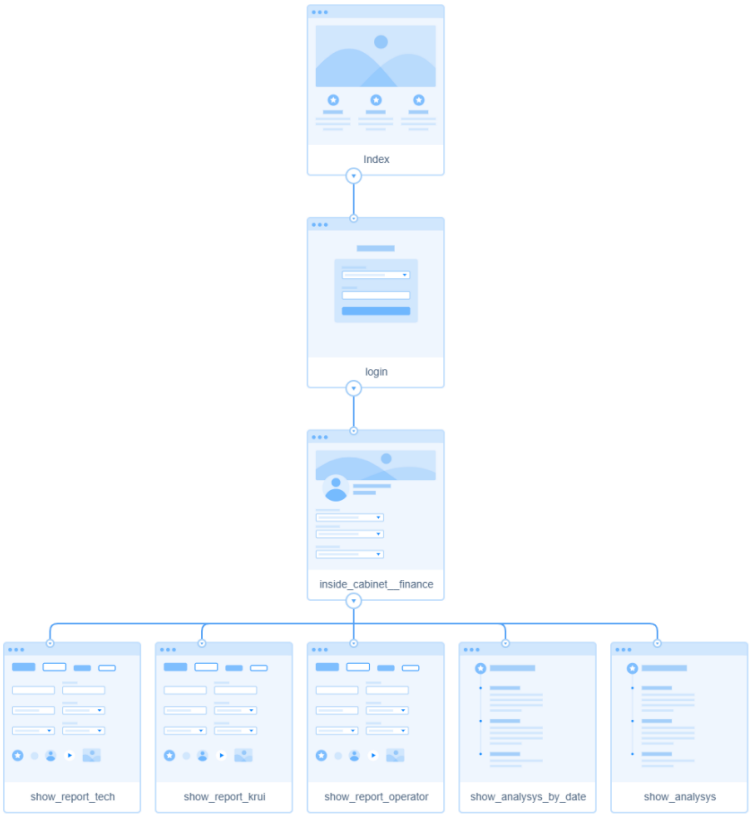

Рисунок Б.3 – Діаграма користувацького інтерфейсу фінансового відділу

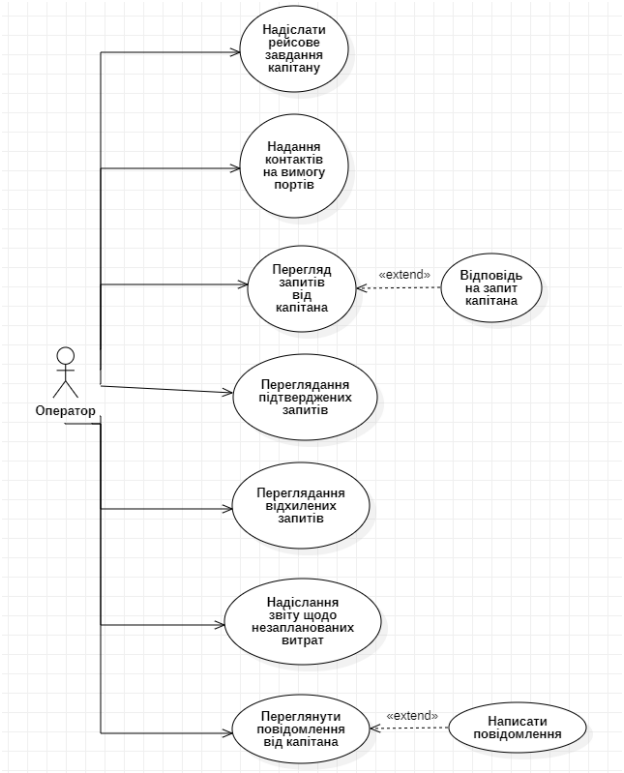

Рисунок Б.4 – Діаграма варіантів використання для оператора

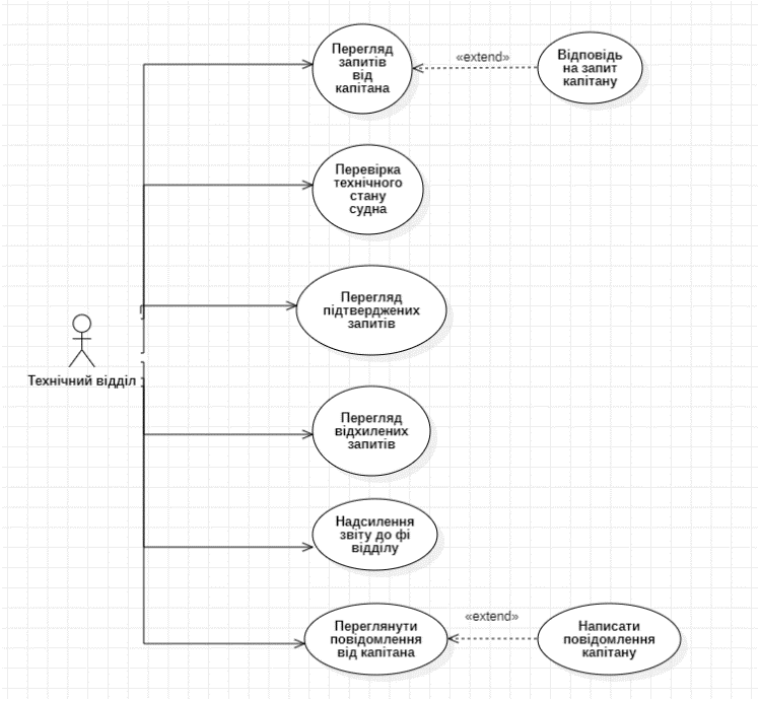

Рисунок Б.5 – Діаграма варіантів використання для технічного відділу

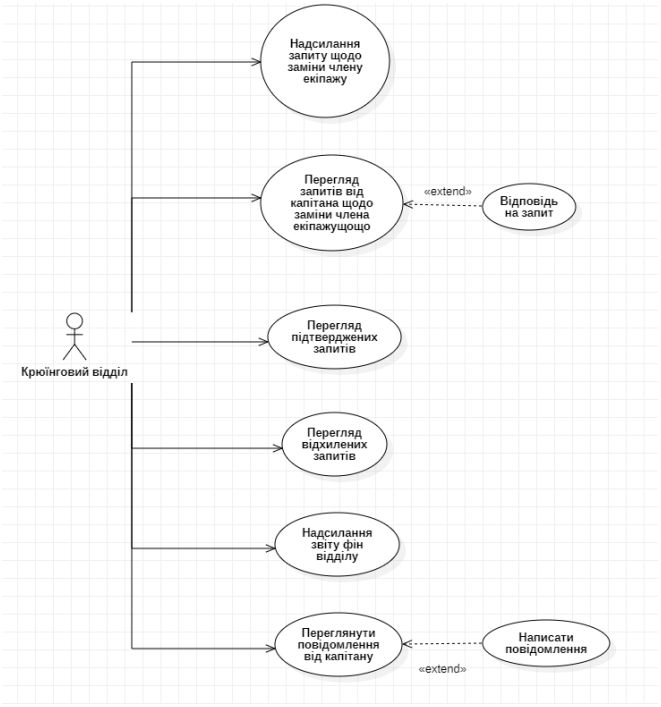

Рисунок Б.6 – Діаграма варіантів використання для крюїнгового відділу

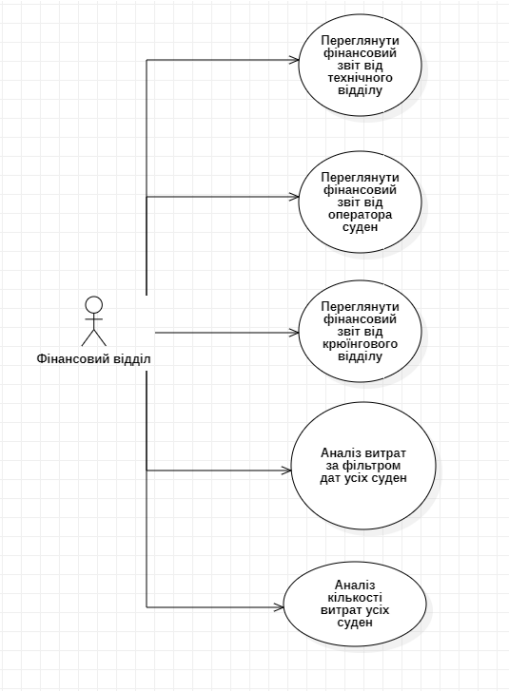

Рисунок Б.7 – Діаграма варіантів використання для фінансового відділу

ДОДАТОК В

Код бази даних

```
CREATE DATABASE ship_company;
DROP SCHEMA IF EXISTS sea;
CREATE SCHEMA IF NOT EXISTS sea;
```

Лістинг В.1 – Створення бази даних та схеми

```
DROP TABLE IF EXISTS sea.role_type CASCADE;
CREATE TABLE IF NOT EXISTS sea.role_type (
id serial PRIMARY KEY NOT NULL,
name text NOT NULL UNIQUE
);
```

Лістинг В.2 – Створення таблиці role_type

```
DROP TABLE IF EXISTS sea.employee CASCADE;
CREATE TABLE IF NOT EXISTS sea.employee (
id serial PRIMARY KEY NOT NULL,
role_type_id integer NOT NULL,
first_name text NOT NULL,
last_name text NOT NULL,
address text NOT NULL,
phone text NOT NULL UNIQUE,
password text NOT NULL,
date_of_birth date NOT NULL,
is_delete boolean NOT NULL DEFAULT FALSE,
CONSTRAINT employee_role_type_id_fk FOREIGN KEY (role_type_id)
REFERENCES sea.role_type(id),
CONSTRAINT younger_then_eighteen check (date_of_birth <
current_timestamp - interval '18 year' and date_of_birth >
current_timestamp - interval '60 Year')
);
```

Лістинг В.3 – Створення таблиці employee

```
DROP TABLE IF EXISTS sea.rank CASCADE;
CREATE TABLE IF NOT EXISTS sea.rank (
id serial PRIMARY KEY NOT NULL,
name text NOT NULL UNIQUE
);
```

Лістинг В.4 – Створення таблиці rank

```
DROP TABLE IF EXISTS sea.sea_worker CASCADE;
CREATE TABLE IF NOT EXISTS sea.sea_worker (
```

Лістинг В.5 – Створення таблиці sea_worker, лист 1

```
id serial PRIMARY KEY NOT NULL,
rank_id integer NOT NULL,
employee_id integer NOT NULL,
CONSTRAINT sea_worker_rank_id_fk FOREIGN KEY (rank_id) REFERENCES
sea.rank(id),
CONSTRAINT sea_worker_employee_id_fk FOREIGN KEY (employee_id)
REFERENCES sea.employee(id)
);
```

Лістинг В.5 – Створення таблиці sea_worker, лист 2

```
DROP TABLE IF EXISTS sea.department CASCADE;
CREATE TABLE IF NOT EXISTS sea.department (
id serial PRIMARY KEY NOT NULL,
name text NOT NULL UNIQUE
);
```

Лістинг В.6 – Створення таблиці department

```
DROP TABLE IF EXISTS sea.office_worker CASCADE;
CREATE TABLE IF NOT EXISTS sea.office_worker (
id serial PRIMARY KEY NOT NULL,
department_id integer NOT NULL,
employee_id integer NOT NULL,
CONSTRAINT office_worker_department_id_fk FOREIGN KEY
(department_id) REFERENCES sea.department(id),
CONSTRAINT office_worker_employee_id_fk FOREIGN KEY (employee_id)
REFERENCES sea.employee(id)
);
```

Лістинг В.7 – Створення таблиці office_worker

```
DROP TABLE IF EXISTS sea.ship CASCADE;
CREATE TABLE IF NOT EXISTS sea.ship (
id serial PRIMARY KEY NOT NULL,
name text NOT NULL UNIQUE,
sea_worker_id integer NOT NULL,
year_of_building integer NOT NULL,
weight integer NOT NULL,
length integer NOT NULL,
CONSTRAINT ship_sea_worker_id_fk FOREIGN KEY (sea_worker_id)
REFERENCES sea.sea_worker(id),
CONSTRAINT ship_older_or_younger_30_year CHECK (year_of_building
>= EXTRACT(YEAR FROM CURRENT_DATE)::integer - 30 AND
year_of_building <= EXTRACT(YEAR FROM CURRENT_DATE)::integer)
);
```

Лістинг В.8 – Створення таблиці ship

```
DROP TABLE IF EXISTS sea.team CASCADE;
CREATE TABLE IF NOT EXISTS sea.team (
sea_worker_id integer NOT NULL,
ship_id integer NOT NULL,
CONSTRAINT    team__sea_worker_id_ship_id__pk    PRIMARY    KEY
(sea_worker_id, ship_id)
);
```

Лістинг В.9 – Створення таблиці team

```
DROP TABLE IF EXISTS sea.type CASCADE;
CREATE TABLE IF NOT EXISTS sea.type (
id serial PRIMARY KEY NOT NULL,
name text NOT NULL UNIQUE
);
```

Лістинг В.9 – Створення таблиці type

```
DROP TABLE IF EXISTS sea.message CASCADE;
CREATE TABLE IF NOT EXISTS sea.message (
id serial PRIMARY KEY NOT NULL,
ship_id integer NOT NULL,
department_id integer NOT NULL,
date_time timestamp NOT NULL,
message_text text NOT NULL,
how_send integer,
CONSTRAINT message_ship_id_id_fk FOREIGN KEY (ship_id) REFERENCES
sea.ship(id),
CONSTRAINT message_department_id_fk FOREIGN KEY (department_id)
REFERENCES sea.department(id)
);
ALTER TABLE IF EXISTS sea.message
ADD COLUMN how_send integer;
```

Лістинг В.10 – Створення таблиці message

```
DROP TABLE IF EXISTS sea.report CASCADE;
CREATE TABLE IF NOT EXISTS sea.report (
id serial PRIMARY KEY NOT NULL,
ship_id integer,
department_id integer NOT NULL,
type_id integer NOT NULL,
sum integer,
date_time timestamp NOT NULL,
message_text text NOT NULL,
ischeck boolean,
comment text,
how_send integer,
```

Лістинг В.11 – Створення таблиці report, лист 1

```
CONSTRAINT report_ship_id_id_fk FOREIGN KEY (ship_id) REFERENCES
sea.ship(id),
CONSTRAINT report_department_id_fk FOREIGN KEY (department_id)
REFERENCES sea.department(id),
CONSTRAINT report_type_id_fk FOREIGN KEY (type_id) REFERENCES
sea.type(id)
);
```

Лістинг В.11 – Створення таблиці report, лист 2

```
CREATE OR REPLACE VIEW sea.employee_info AS
SELECT
    e.id,
    e.first_name,
    e.last_name,
    e.address,
    e.phone,
    e.password,
    e.date_of_birth,
    r.name AS role_type,
    ra.name AS rank,
    d.name AS department
FROM sea.employee e
    INNER JOIN sea.role_type r ON (r.id = e.role_type_id)
    LEFT JOIN sea.sea_worker w ON (e.id = w.employee_id)
    LEFT JOIN sea.rank ra ON (w.rank_id = ra.id)
    LEFT JOIN sea.office_worker o ON (e.id = o.employee_id)
    LEFT JOIN sea.department d ON (o.department_id = d.id)
WHERE TRUE
    AND (is_delete = FALSE);
```

Лістинг В.12 – Створення представлення employee_info

```
CREATE OR REPLACE VIEW sea.show_department AS
SELECT
    d.id,
    d.name
FROM sea.department d
WHERE TRUE;
```

Лістинг В.13 – Створення представлення show_department

```
CREATE OR REPLACE VIEW sea.signin AS
SELECT
    e.phone AS login,
    e.password AS password,
    r.name AS position
FROM sea.employee e
    INNER JOIN sea.role_type r ON (r.id = e.role_type_id)
WHERE TRUEAND (e.is_delete = FALSE);
```

Лістинг В.14 – Створення представлення signin

```
CREATE OR REPLACE VIEW sea.show_fin_report_oper AS
SELECT
    r.id,
    d.name AS department_name,
    t.name AS type_name,
    r.sum,
    r.date_time,
    r.message_text
FROM sea.report r
    LEFT JOIN sea.department d ON (d.id = r.department_id)
    INNER JOIN sea.type t ON (t.id = r.type_id)
WHERE TRUE
    AND (t.name = 'Надання звіту фінансового звіту, щодо
незапланованих витрат капітаном')
ORDER BY
    r.date_time DESC
;

CREATE OR REPLACE VIEW sea.show_fin_report_krui AS
SELECT
    r.id,
    d.name AS department_name,
    t.name AS type_name,
    r.sum,
    r.date_time,
    r.message_text
FROM sea.report r
    LEFT JOIN sea.department d ON (d.id = r.department_id)
    INNER JOIN sea.type t ON (t.id = r.type_id)
WHERE TRUE
    AND (t.name = 'Звіт, щодо зарплатні')
ORDER BY
    r.date_time DESC
;

CREATE OR REPLACE VIEW sea.show_fin_report_tech AS
SELECT
    r.id,
    d.name AS department_name,
    t.name AS type_name,
    r.sum,
    r.date_time,
    r.message_text
FROM sea.report r
    LEFT JOIN sea.department d ON (d.id = r.department_id)
    INNER JOIN sea.type t ON (t.id = r.type_id)
WHERE TRUE
    AND (t.name = 'Надсилання звіту до фінансового відділу')
ORDER BY
    r.date_time DESC;
```

Лістинг В.15 – Створення представлення show_fin_report_oper, show_fin_report_krui, show_fin_report_tech

```
CREATE OR REPLACE VIEW sea.show_message_captain AS
SELECT
CASE
WHEN (how_send = 1) THEN
'Капітана корабля - ' || s.name
WHEN (how_send <> 2) THEN
'Відділ - ' || d.name
END AS name,
m.message_text,
m.date_time,
e.phone,
d.name AS department
FROM sea.employee e
INNER JOIN sea.role_type ro ON (ro.id = e.role_type_id)
INNER JOIN sea.sea_worker w ON (e.id = w.employee_id)
INNER JOIN sea.ship s ON (w.id = s.sea_worker_id)
INNER JOIN sea.message m ON (s.id = m.ship_id)
INNER JOIN sea.department d ON (m.department_id = d.id)
WHERE TRUE
AND (is_delete = FALSE)
ORDER BY
m.date_time DESC
```

Лістинг В.16 – Створення представлення show_message_captain

```
CREATE OR REPLACE VIEW sea.show_type AS

SELECT
    d.id,
    d.name
FROM sea.type d
WHERE TRUE
;
```

Лістинг В.17 – Створення представлення show_type

```
CREATE OR REPLACE VIEW sea.show_ship AS
SELECT
    s.id,
    s.name,
    s.sea_worker_id,
    s.year_of_building,
    s.weight,
    s.length,
    e.first_name,
    e.last_name
FROM sea.ship s
    INNER JOIN sea.sea_worker ON (s.sea_worker_id = sea_worker.id)
    INNER JOIN sea.employee e ON (e.id = sea_worker.employee_id)
WHERE TRUE
;
```

Лістинг В.18 – Створення представлення show_ship

```
CREATE OR REPLACE VIEW sea.show_type AS

SELECT
    d.id,
    d.name
FROM sea.type d
WHERE TRUE
;
```

Лістинг В.19 – Створення представлення show_type

```
CREATE OR REPLACE VIEW sea.show_report_null AS
SELECT
    r.id,
    r.ship_id,
    s.name AS ship_name,
    r.department_id,
    d.name AS department_name,
    r.type_id,
    t.name AS type_name,
    r.sum,
    r.date_time,
    r.message_text,
    r.ischeck,
    r.comment,
    e.phone,
    r.how_send
FROM sea.report r
    LEFT JOIN sea.ship s ON (s.id = r.ship_id)
    LEFT JOIN sea.department d ON (d.id = r.department_id)
    LEFT JOIN sea.employee e ON (e.id = s.sea_worker_id)
    INNER JOIN sea.type t ON (t.id = r.type_id)
WHERE TRUE
    AND (ischeck IS NULL)
ORDER BY
    r.date_time DESC
;
```

Лістинг В.20 – Створення представлення show_report_null

```
CREATE OR REPLACE FUNCTION sea._analysis_all_ship(
_date_from date,
_date_to date
)
RETURNS TABLE (
sum bigint,
count bigint
)
SECURITY DEFINER
LANGUAGE plpgsql
```

Лістинг В.21 – Створення функції analysis_all_ship, лист 1

```
AS
$BODY$
DECLARE
_sum bigint;
_count bigint;
BEGIN
SELECT
SUM(r.sum) AS sum
INTO
_sum
FROM sea.report r
WHERE TRUE
AND date_time BETWEEN _date_from AND _date_to
;
SELECT
COUNT(r.id) AS count
INTO
_count
FROM sea.report r
WHERE TRUE
AND (r.sum IS NOT NULL)
AND date_time BETWEEN _date_from AND _date_to
;
RETURN QUERY
SELECT
_sum,
_count
;
END;
$BODY$;
```

Лістинг В.21 – Створення функції analysis_all_ship, лист 2

```
CREATE OR REPLACE FUNCTION sea._answer_report(
_id integer,
_check boolean,
_text text
)
RETURNS void
SECURITY DEFINER
LANGUAGE plpgsql
AS
$BODY$
DECLARE
BEGIN
UPDATE sea.report SET
ischeck = _check,
comment = _text
WHERE TRUE
AND (id = _id)
;
END;$BODY$;
```

Лістинг В.22 – Створення функції _answer_report

```
CREATE OR REPLACE FUNCTION sea._show_message_by_department(
_phone text,
_ship_id integer
)
RETURNS TABLE (
name1 text,
message_text1 text,
date_time1 timestamp
)
SECURITY DEFINER
LANGUAGE plpgsql
AS
$BODY$
DECLARE
BEGIN

RETURN QUERY
SELECT
CASE
WHEN (how_send = 1) THEN
'Капітана корабля - ' || s.name
WHEN (how_send <> 1) THEN
'Відділ - ' || d.name
END AS name,
m.message_text,
m.date_time
FROM sea.message m
INNER JOIN sea.ship s ON (s.id = m.ship_id)
INNER JOIN sea.department d ON (m.department_id = d.id)
WHERE TRUE
AND m.department_id = (
SELECT
d.id
FROM sea.employee e
INNER JOIN sea.office_worker o ON (o.employee_id = e.id)
INNER JOIN sea.department d ON (o.department_id = d.id)
WHERE TRUE
AND (e.phone = _phone)
)
AND (m.ship_id = _ship_id)
ORDER BY
m.date_time DESC
;
END;
$BODY$;
```

Лістинг В.23 – Створення функції _show_message_by_department

```
CREATE OR REPLACE FUNCTION sea._show_report_by_department(
_phone text)
```

Лістинг В.24 – Створення функції _show_report_by_department, _show_report_by_department_true, _show_report_by_department_false, лист 1

```
RETURNS TABLE (
comment1 text,
ischeck1 boolean,
ship_name1 text,
type_name1 text,
dep_name1 text,
message_text1 text,
sum1 integer,
date_time1 timestamp
)
SECURITY DEFINER
LANGUAGE plpgsql
AS
$BODY$
DECLARE
BEGIN
RETURN QUERY
SELECT
r.comment,
r.ischeck,
s.name AS ship_name,
t.name AS type_name,
d.name AS dep_name,
r.message_text,
r.sum,
r.date_time
FROM sea.report r
LEFT JOIN sea.ship s ON (s.id = r.ship_id)
INNER JOIN sea.department d ON (r.department_id = d.id)
INNER JOIN sea.type t ON (r.type_id = t.id)
WHERE TRUE
AND r.department_id = (
SELECT
d.id
FROM sea.employee e
INNER JOIN sea.office_worker o ON (o.employee_id = e.id)
INNER JOIN sea.department d ON (o.department_id = d.id)
WHERE TRUE
AND (e.phone = _phone)
)
AND r.ischeck IS NULL
ORDER BY
r.date_time DESC
;
END;
$BODY$;
CREATE OR REPLACE FUNCTION sea._show_report_by_department_true(
_phone text
)
RETURNS TABLE (comment1 text,ischeck1 boolean,
```

Лістинг В.24 – Створення функції _show_report_by_department, _show_report_by_department_true, _show_report_by_department_false, лист 2

```
ship_name1 text,
type_name1 text,
dep_name1 text,
message_text1 text,
sum1 integer,
date_time1 timestamp
)
SECURITY DEFINER
LANGUAGE plpgsql
AS
$BODY$
DECLARE
BEGIN
RETURN QUERY
SELECT
r.comment,
r.ischeck,
s.name AS ship_name,
t.name AS type_name,
d.name AS dep_name,
r.message_text,
r.sum,
r.date_time
FROM sea.report r
LEFT JOIN sea.ship s ON (s.id = r.ship_id)
INNER JOIN sea.department d ON (r.department_id = d.id)
INNER JOIN sea.type t ON (r.type_id = t.id)
WHERE TRUE
AND r.department_id = (
SELECT
d.id
FROM sea.employee e
INNER JOIN sea.office_worker o ON (o.employee_id = e.id)
INNER JOIN sea.department d ON (o.department_id = d.id)
WHERE TRUE
AND (e.phone = _phone)
)
AND (r.ischeck = TRUE)
ORDER BY
r.date_time DESC
;
END;
$BODY$;
CREATE OR REPLACE FUNCTION sea._show_report_by_department_false(
_phone text
)
RETURNS TABLE (
comment1 text,
ischeck1 boolean,
ship_name1 text,
```

Лістинг В.24 – Створення функції _show_report_by_department, _show_report_by_department_true, _show_report_by_department_false, лист 3

```
type_name1 text,
dep_name1 text,
message_text1 text,
sum1 integer,
date_time1 timestamp
)
SECURITY DEFINER
LANGUAGE plpgsql
AS
$BODY$
DECLARE
BEGIN
RETURN QUERY
SELECT
r.comment,
r.ischeck,
s.name AS ship_name,
t.name AS type_name,
d.name AS dep_name,
r.message_text,
r.sum,
r.date_time
FROM sea.report r
LEFT JOIN sea.ship s ON (s.id = r.ship_id)
INNER JOIN sea.department d ON (r.department_id = d.id)
INNER JOIN sea.type t ON (r.type_id = t.id)
WHERE TRUE
AND r.department_id = (
SELECT
d.id
FROM sea.employee e
INNER JOIN sea.office_worker o ON (o.employee_id = e.id)
INNER JOIN sea.department d ON (o.department_id = d.id)
WHERE TRUE
AND (e.phone = _phone)
)
AND (r.ischeck = FALSE)
ORDER BY
r.date_time DESC
;
END;
$BODY$;
```

Лістинг В.24 – Створення функції _show_report_by_department, _show_report_by_department_true, _show_report_by_department_false, лист 4

```
CREATE OR REPLACE FUNCTION sea._show_report_by_department_answer(
_phone text,
_how_send integer)RETURNS TABLE (
id1 integer,
```

Лістинг В.25 – Створення функції _show_report_by_department_answer, лист 1

```
comment1 text,
ischeck1 boolean,
ship_name1 text,
type_name1 text,
dep_name1 text,
message_text1 text,
sum1 integer,
date_time1 timestamp
)
SECURITY DEFINER
LANGUAGE plpgsql
AS
$BODY$
DECLARE
BEGIN
RETURN QUERY
SELECT
r.id,
r.comment,
r.ischeck,
s.name AS ship_name,
t.name AS type_name,
d.name AS dep_name,
r.message_text,
r.sum,
r.date_time
FROM sea.report r
LEFT JOIN sea.ship s ON (s.id = r.ship_id)
INNER JOIN sea.department d ON (r.department_id = d.id)
INNER JOIN sea.type t ON (r.type_id = t.id)
WHERE TRUE
AND r.department_id = (
SELECT
    d.id
FROM sea.employee e
    INNER JOIN sea.office_worker o ON (o.employee_id = e.id)
    INNER JOIN sea.department d ON (o.department_id = d.id)
WHERE TRUE
    AND (e.phone = _phone)
)
AND (r.ischeck IS NULL)
AND (how_send <> _how_send)
ORDER BY
r.date_time DESC
;
END;
$BODY$;
```

Лістинг В.25 – Створення функції _show_report_by_department_answer, лист 2

```
CREATE OR REPLACE FUNCTION sea._user_create(
_first_name text,_last_name text,
```

Лістинг В.26 – Створення функції _user_create, лист 1

```
_address text,
_data_of_birthday date,
_login text,
_password text,
_role_type text,
_rank text
)
RETURNS void
SECURITY DEFINER
LANGUAGE plpgsql
AS
$BODY$
DECLARE
_employee_id integer;
_role_id integer;
_rank_id integer;
_department_id integer;
BEGIN
_first_name := regexp_replace(_first_name, '[ ]*[0-9]+[ ]*|[0-
9]+','','g');
_last_name := regexp_replace(_last_name, '[ ]*[0-9]+[ ]*|[0-
9]+','','g');
IF ((_password = '') OR (_password IS NULL)) THEN
RAISE EXCEPTION 'password can not be empty or null';
END IF;
IF ((_data_of_birthday) > NOW()) THEN
RAISE EXCEPTION 'DATE OUT OF RANGE';
END IF;
_login := regexp_replace(_login, '[^0-9]', '', 'g');
IF ((_login ~* '^380(97|98|50|67|93|63|68|96|95|99|94|66|73)([0-
9]{7})$') = FALSE) THEN
RAISE EXCEPTION 'FORMAT IS UNDEFINED';
END IF;
_password := replace(_password, ' ', '');
IF ((NULLIF(trim(_first_name), '') IS NULL) OR
(NULLIF(trim(_last_name), '') IS NULL) OR (NULLIF(trim(_address),
'') IS NULL) OR (NULLIF(trim(_password), '') IS NULL) ) THEN
RAISE EXCEPTION 'MUST BE NOT NULL';
END IF;
SELECT
id
INTO
_role_id
FROM sea.role_type r
WHERE TRUE
AND (r.name = _role_type)
;
SELECT
id
INTO
_rank_id
```

Лістинг В.26 – Створення функції _user_create, лист 2

```
FROM sea.rank ra
WHERE TRUE
AND (ra.name = _rank)
;
SELECT
id
INTO
_department_id
FROM sea.department d
WHERE TRUE
AND (d.name = CASE
    WHEN(_role_type = 'Crewing_department') THEN
        'Крюінговий відділ'
    WHEN(_role_type = 'Financial_department') THEN
        'Фінансовий відділ'
    WHEN(_role_type = 'Ship_operator') THEN
        'Менеджер (оператор судна)'
    WHEN(_role_type = 'Technical_department') THEN
        'Технічний відділ'
    END
)
;
IF EXISTS(
SELECT
login
FROM sea.signin
WHERE TRUE
AND login = _login
)

THEN RAISE EXCEPTION 'Already exists number';

ELSIF length (cast(_login AS text)) > 12 OR length (cast(_login AS
text)) < 12

THEN RAISE EXCEPTION 'More or less 12';

ELSIF (NULLIF(trim(_rank), '') IS NOT NULL AND _role_type NOT IN
('Crewing_department', 'Financial_department', 'Ship_operator',
'Technical_department')) THEN

INSERT INTO
sea.employee (role_type_id, first_name, last_name, address, phone,
password, date_of_birth, is_delete)
VALUES
(_role_id,    _first_name,    _last_name,    _address,    _login,
crypt(_password, gen_salt('bf')), _data_of_birthday, FALSE)
RETURNING id INTO _employee_id;

INSERT INTO
sea.sea_worker (rank_id, employee_id)
```

Лістинг В.26 – Створення функції _user_create, лист 3

```
VALUES
(_rank_id, _employee_id);

ELSIF (NULLIF(trim(_rank), '') IS NULL AND _role_type IN
('Crewing_department', 'Financial_department', 'Ship_operator',
'Technical_department')) THEN
INSERT INTO
sea.employee (role_type_id, first_name, last_name, address, phone,
password, date_of_birth, is_delete)
VALUES
(_role_id, _first_name, _last_name, _address, _login,
crypt(_password, gen_salt('bf')), _data_of_birthday, FALSE)
RETURNING id INTO _employee_id;
INSERT INTO
sea.office_worker (department_id, employee_id)
VALUES
(_department_id, _employee_id);
END IF;
END;
$BODY$;
```

Лістинг В.26 – Створення функції _user_create, лист 4

```
CREATE OR REPLACE FUNCTION sea._user_delete(
_id integer[]
)
RETURNS void
SECURITY DEFINER
LANGUAGE plpgsql
AS
$BODY$
DECLARE
BEGIN
UPDATE sea.employee s
SET
is_delete = TRUE
WHERE
id = ANY(_id)
;
END;
$BODY$;
```

Лістинг В.27 – Створення функції _user_delete

```
CREATE OR REPLACE FUNCTION sea._user_update(
_id integer,
_first_name text,
_last_name text,
_address text,
_login text,
_data_of_birthday date,_password text)
```

Лістинг В.28 – Створення функції _user_update, лист 1

```
RETURNS void
SECURITY DEFINER
LANGUAGE plpgsql
AS
$BODY$
DECLARE BEGIN
_first_name := regexp_replace(_first_name, '[ ]*[0-9]+[ ]*|[0-
9]+','','g');
_last_name := regexp_replace(_last_name, '[ ]*[0-9]+[ ]*|[0-
9]+','','g');
IF ((_password = '') OR (_password IS NULL)) THEN
RAISE EXCEPTION 'password can not be empty or null';
END IF;
IF ((_data_of_birthday) > NOW()) THEN
RAISE EXCEPTION 'DATE OUT OF RANGE';
END IF;
_login := regexp_replace(_login, '[^0-9]', '', 'g');
IF ((_login ~* '^380(97|98|50|67|93|63|68|96|95|99|94|66|73)([0-
9]{7})$') = FALSE) THEN
RAISE EXCEPTION 'FORMAT IS UNDEFINED';
END IF;
_password := replace(_password, ' ', '');
IF ((NULLIF(trim(_first_name), '') IS NULL) OR
(NULLIF(trim(_last_name), '') IS NULL) OR (NULLIF(trim(_address),
'') IS NULL) ) THEN
RAISE EXCEPTION 'MUST BE NOT NULL';
END IF;
IF length(_password::text) > 20 THEN
RAISE INFO '%', '1';
UPDATE sea.employee
SET
phone = _login,
first_name = _first_name,
last_name = _last_name,
password = _password,
date_of_birth = _data_of_birthday,
address = _address
WHERE TRUE
AND id = _id;
ELSE
RAISE INFO '%', '2';
UPDATE sea.employee
SET
phone = _login,
first_name = _first_name,
last_name = _last_name,
password = crypt(_password, gen_salt('bf')),
date_of_birth = _data_of_birthday,
address = _address
WHERE TRUE
AND id = _id;END IF;END;$BODY$;
```

Лістинг В.28 – Створення функції _user_update, лист 2

```
CREATE OR REPLACE FUNCTION sea._write_report(
    _phone text,
    _department integer,
    _type text,
    _sum integer,
    _text text,
    _how_send integer
)
    RETURNS void
    SECURITY DEFINER
    LANGUAGE plpgsql
AS
$BODY$
DECLARE
    _id_ship integer;
    _id_type integer;
BEGIN
    SELECT
        t.id
    INTO
        _id_type
    FROM sea.type t
    WHERE TRUE
        AND (t.name = _type)
    ;
    SELECT
        s.id
    INTO
        _id_ship
    FROM sea.employee e
        INNER JOIN sea.sea_worker sw ON (e.id = sw.employee_id)
        INNER JOIN sea.ship s ON (sw.id = s.sea_worker_id)
    WHERE TRUE
        AND (phone = _phone)
    ;
    INSERT INTO sea.report (
        ship_id,
        department_id,
        type_id,
        sum,
        date_time,
        message_text,
        how_send
    )
    VALUES(
        _id_ship,
        _department,
        _id_type,
        _sum,
        CURRENT_TIMESTAMP,
        _text,
        _how_send);END;$BODY$;
```

Лістинг В.29 – Створення функції _write_report

```
CREATE OR REPLACE FUNCTION sea._write_report_to_captain(
_phone text,
_ship_id integer,
_type text,
_sum integer,
_text text,
_how_send integer
)
RETURNS void
SECURITY DEFINER
LANGUAGE plpgsql
AS
$BODY$
DECLARE
_id_type integer;
_id_department integer;
BEGIN
SELECT
t.id
INTO
_id_type
FROM sea.type t
WHERE TRUE
AND (t.name = _type)
;
SELECT
d.id
INTO
_id_department
FROM sea.employee e
INNER JOIN sea.office_worker o ON (o.employee_id = e.id)
INNER JOIN sea.department d ON (o.department_id = d.id)
WHERE TRUE
AND (e.phone = _phone)
;
INSERT INTO sea.report (
ship_id,
department_id,
type_id,
sum,
date_time,
message_text,
how_send
)
VALUES(
_ship_id,
_id_department,
_id_type,
_sum,
CURRENT_TIMESTAMP,
_text,
_how_send);END;$BODY$;
```

Лістинг В.30 – Створення функції _write_report_to_captain

```
CREATE OR REPLACE FUNCTION sea._write_report_to_department(
    _phone text,
    _type text,
    _sum integer,
    _text text,
    _how_send integer
)
    RETURNS void
    SECURITY DEFINER
    LANGUAGE plpgsql
AS
$BODY$
DECLARE
    _id_type integer;
    _id_department integer;
BEGIN
    SELECT
        t.id
    INTO
        _id_type
    FROM sea.type t
    WHERE TRUE
        AND (t.name = _type)
    ;
    SELECT
        d.id
    INTO
        _id_department
    FROM sea.employee e
        INNER JOIN sea.office_worker o ON (o.employee_id = e.id)
        INNER JOIN sea.department d ON (o.department_id = d.id)
    WHERE TRUE
        AND (e.phone = _phone);
    INSERT INTO sea.report (
        ship_id,
        department_id,
        type_id,
        sum,
        date_time,
        message_text,
        how_send,
        ischeck
    )
    VALUES(
        NULL,
        _id_department,
        _id_type,
        _sum,
        CURRENT_TIMESTAMP,
        _text,
        _how_send,
        TRUE);END;$BODY$;
```

Лістинг В.31 – Створення функції _write_report_to_department

```
CREATE OR REPLACE FUNCTION sea._write_text(
_text text,
_phone text,
_department text,
_how_send integer
)
RETURNS void
SECURITY DEFINER
LANGUAGE plpgsql
AS
$BODY$
DECLARE
_id_ship integer;
_id_department integer;
BEGIN
SELECT
s.id
INTO
_id_ship
FROM sea.employee e
INNER JOIN sea.sea_worker sw ON (e.id = sw.employee_id)
INNER JOIN sea.ship s ON (sw.id = s.sea_worker_id)
WHERE TRUE
AND (phone = _phone)
;
SELECT
id
INTO
_id_department
FROM sea.department
WHERE TRUE
AND (name = _department)
;
INSERT INTO sea.message (
ship_id,
department_id,
date_time,
message_text,
how_send
)

VALUES(
_id_ship,
_id_department,
CURRENT_TIMESTAMP,
_text,
_how_send
);

END;
$BODY$;
```

Лістинг В.32 – Створення функції _write_text

```
CREATE OR REPLACE FUNCTION sea._write_text_to_captain(
_text text,
_phone text,
_ship_id integer,
_how_send integer
)
RETURNS void
SECURITY DEFINER
LANGUAGE plpgsql
AS
$BODY$
DECLARE
_id_department integer;
BEGIN
SELECT
d.id
INTO
_id_department
FROM sea.employee e
INNER JOIN sea.office_worker o ON (o.employee_id = e.id)
INNER JOIN sea.department d ON (o.department_id = d.id)
WHERE TRUE
AND (e.phone = _phone)
;
INSERT INTO sea.message (
ship_id,
department_id,
date_time,
message_text,
how_send
)
VALUES(
_ship_id,
_id_department,
CURRENT_TIMESTAMP,
_text,
_how_send
);END;
$BODY$;
```

Лістинг В.33 – Створення функції _write_text_to_captain

```
DROP ROLE ship_captain, ship_admin, ship_tech, ship_krui,
ship_fin, ship_manager, ship_user;
CREATE ROLE ship_captain WITH PASSWORD '12' LOGIN;
CREATE ROLE ship_tech WITH PASSWORD '123' LOGIN;
CREATE ROLE ship_admin SUPERUSER LOGIN PASSWORD '12345';
CREATE ROLE ship_krui WITH PASSWORD '1' LOGIN;
CREATE ROLE ship_fin WITH PASSWORD '1' LOGIN;
CREATE ROLE ship_manager WITH PASSWORD '1' LOGIN;
CREATE ROLE ship_user WITH PASSWORD '1' LOGIN;
```

Лістинг В.34 – Створення ролей

```
GRANT CONNECT ON DATABASE ship_company TO ship_captain,
ship_tech, ship_admin, ship_krui, ship_fin, ship_manager,
ship_user;
GRANT USAGE ON SCHEMA sea TO ship_captain, ship_tech,
ship_admin, ship_krui, ship_fin, ship_manager, ship_user;
```

Лістинг В.35 – Права ролям на вхід до бази даних та схему

```
REVOKE ALL ON FUNCTION
    sea._analysis_all_ship(date, date),
    sea._answer_report(integer,boolean, text),
    sea._show_message_by_department(text, integer),
    sea._show_report_by_department_false(text),
    sea._show_report_by_department_true(text),
    sea._show_report_by_department(text),
    sea._show_report_by_department_answer(text, integer),
    sea._user_create(text,text,text,date,text,text,text,text),
    sea._user_delete(integer[]),
    sea._user_update(integer,text,text,text,date,text, text),
    sea._write_report(text, integer, text,
integer,text,integer),
    sea._write_report_to_captain(text, integer, text,
integer,text,integer),
    sea._write_report_to_department(text, text, integer,
text,integer),
    sea._write_text(text, text, text, integer),
    sea._write_text_to_captain(text, text, integer, integer)
FROM PUBLIC;
```

Лістинг В.36 – Видалення прав на функції зі схеми public

```
GRANT SELECT ON
    sea.signin
TO ship_user;

GRANT SELECT ON
    sea.employee_info,
    sea.department,
    sea.rank,
    sea.role_type

TO ship_admin;

GRANT EXECUTE ON FUNCTION
    sea._user_create(text,text,text,date,text,text,text,text),
    sea._user_delete(integer[]),
    sea._user_update(integer,text,text,text,date,text, text)

TO ship_admin;
```

Лістинг В.37 – Права для ролі ship_admin

```
GRANT SELECT ON
    sea.show_message_captain,
    sea.show_department,
    sea.show_type,
    sea.show_report_null,
    sea.show_report_true,
    sea.show_report_false
TO ship_captain;

GRANT EXECUTE ON FUNCTION
    sea._write_text(text, text, text, integer),
    sea._write_report(text, integer, text,
integer,text,integer),
    sea._answer_report(integer,boolean, text)
TO ship_captain;
```

Лістинг В.38 – Права для ролі ship_captain

```
GRANT SELECT ON
    sea.show_ship,
    sea.show_type,
    sea.show_department,
    sea.show_report_null
TO ship_krui;

GRANT EXECUTE ON FUNCTION
    sea._show_message_by_department(text, integer),
    sea._write_text_to_captain(text, text, integer, integer),
    sea._show_report_by_department(text),
    sea._show_report_by_department_false(text),
    sea._show_report_by_department_true(text),
    sea._write_report_to_captain(text, integer, text,
integer,text,integer),
    sea._write_report_to_department(text, text, integer,
text,integer),
    sea._answer_report(integer,boolean, text),
    sea._show_report_by_department_answer(text, integer)
TO ship_krui;
```

Лістинг В.39 – Права для ролі ship_krui

```
GRANT SELECT ON
    sea.show_ship,
    sea.show_type,
    sea.show_department,
    sea.show_report_null
TO ship_tech;

GRANT EXECUTE ON FUNCTION
    sea._show_message_by_department(text, integer),
    sea._write_text_to_captain(text, text, integer, integer),
    sea._show_report_by_department(text),
```

Лістинг В.40 – Права для ролі ship_tech, лист 1

```
    sea._show_report_by_department_false(text),
    sea._show_report_by_department_true(text),
    sea._write_report_to_captain(text, integer, text,
integer,text,integer),
    sea._write_report_to_department(text, text, integer,
text,integer),
    sea._answer_report(integer,boolean, text),
    sea._show_report_by_department_answer(text, integer)
TO ship_tech;
```

Лістинг В.40 – Права для ролі ship_tech, лист 2

```
GRANT SELECT ON
    sea.show_ship,
    sea.show_type,
    sea.show_department,
    sea.show_report_null
TO ship_manager;

GRANT EXECUTE ON FUNCTION
    sea._show_message_by_department(text, integer),
    sea._write_text_to_captain(text, text, integer, integer),
    sea._show_report_by_department(text),
    sea._show_report_by_department_false(text),
    sea._show_report_by_department_true(text),
    sea._write_report_to_captain(text, integer, text,
integer,text,integer),
    sea._write_report_to_department(text, text, integer,
text,integer),
    sea._answer_report(integer,boolean, text),
    sea._show_report_by_department_answer(text, integer)
TO ship_manager;
```

Лістинг В.41 – Права для ролі ship_manager

```
GRANT SELECT ON
    sea.show_ship,
    sea.show_fin_report_krui,
    sea.show_fin_report_oper,
    sea.show_fin_report_tech
TO ship_fin;

GRANT EXECUTE ON FUNCTION
    sea._show_message_by_department(text, integer),
    sea._write_text_to_captain(text, text, integer, integer),
    sea._analysis_all_ship(date, date)
TO ship_fin;
```

Лістинг В.42 – Права для ролі ship_fin

ДОДАТОК Г

Код програмної реалізації

```
const express = require ("express")
const app = express ()
const pg = require("pg")
const db_operator =
require("./controller/operatorController.js")
const db_customer = require("./controller/captainController.js")
const db_tech = require("./controller/techController.js")
const db_krui = require("./controller/kruiController.js")
const db_fin = require("./controller/finController.js")
const db_admin = require("./controller/adminController.js")
const listen1 = require("./config/config_db");
const client_user1 = require("./config/config_db")
const client_user = client_user1.client_user1
app.use(express.static("public"))
app.set("views","./views")
app.set('view engine','ejs')
app.listen(listen1.listen1)
const PDFDocument = require('pdfkit');
var file = require('file-system');
var fs = require('fs');
const session = require("express-session")
var cookieParser = require('cookie-parser')
app.use(session({ secret: '111', resave:false,
saveUninitialized:false, cookie: { maxAge: 600000 }}));
app.use(cookieParser())
const bodyParser= require('body-parser')
const urlencodedParser = bodyParser.urlencoded({extended:
false});
var multer = require('multer');
const upload = multer({ dest: '/tmp/' });
```

Лістинг Г.1 – Підключення бібліотек та файлів

```
app.post('/cabinet_inside_operator',urlencodedParser,db_operator
.operator)
app.get('/cabinet_inside_operator',urlencodedParser,db_operator.
get_operator)
app.post('/cabinet_inside_operator1',urlencodedParser,db_operato
r.oper1)
app.get('/cabinet_inside_operator1',urlencodedParser,db_operator
.get_oper1)
app.post('/message_captain',urlencodedParser,db_operator.message
_captain)
app.get('/message_captain',urlencodedParser,db_operator.get_mess
age_captain)
```

Лістинг Г.2 – POST та GET запити до контролера для оператору, лист 1

```
app.post('/message_captain1',urlencodedParser,db_operator.messag
e_captain1)
app.post('/message_captain2',urlencodedParser,db_operator.messag
e_captain2)
app.post('/report_captain',urlencodedParser,db_operator.report_c
aptain)
app.get('/report_captain',urlencodedParser,db_operator.get_repor
t_captain)
app.post('/report_captain1',urlencodedParser,db_operator.report_
captain1)
app.post('/report_department_operator',urlencodedParser,db_opera
tor.report_department_operator)
app.get('/report_department_operator',urlencodedParser,db_operat
or.get_report_department_operator)
app.post('/report_department_operator1',urlencodedParser,db_oper
ator.report_department_operator1)
app.post('/show_report_by_operator_answer',urlencodedParser,db_o
perator.show_report_by_operator_answer)
app.get('/show_report_by_operator_answer',urlencodedParser,db_op
erator.get_show_report_by_operator_answer)
app.post('/show_report_by_operator_answer1',urlencodedParser,db_
operator.show_report_by_operator_answer1)
app.post('/show_report_by_operator_answer2',urlencodedParser,db_
operator.show_report_by_operator_answer2)
app.post('/report_department_operator_true',urlencodedParser,db_
operator.report_department_operator_true)
app.get('/report_department_operator_true',urlencodedParser,db_o
perator.get_report_department_operator_true)
app.post('/report_department_operator_false',urlencodedParser,db
_operator.report_department_operator_false)
app.get('/report_department_operator_false',urlencodedParser,db_
operator.get_report_department_operator_false)
```

Лістинг Г.3 – POST та GET запити до контролера для оператору, лист 2

```
app.post('/cabinet_inside_tech',urlencodedParser,db_tech.tech)
app.get('/cabinet_inside_tech',urlencodedParser,db_tech.get_tec)
app.post('/cabinet_inside_tech1',urlencodedParser,db_tech.tech1)
app.get('/cabinet_inside_tech1',urlencodedParser,db_tech.get_tec
h1)
app.post('/message_captain_tech',urlencodedParser,db_tech.messag
e_captain_tech)
app.get('/message_captain_tech',urlencodedParser,db_tech.get_mes
sage_captain_tech)
app.post('/message_captain_tech1',urlencodedParser,db_tech.messa
ge_captain_tech1)
app.post('/message_captain_tech2',urlencodedParser,db_tech.messa
ge_captain_tech2)
app.post('/report_captain_tech',urlencodedParser,db_tech.report_
captain_tech)
```

Лістинг Г.4 – POST та GET запити до контролера для технічного відділу,

лист 1

```
app.get('/report_captain_tech',urlencodedParser,db_tech.get_repo
rt_captain_tech)
app.post('/report_captain_tech1',urlencodedParser,db_tech.report
_captain_tech1)
app.post('/report_department_tech',urlencodedParser,db_tech.repo
rt_department_tech)
app.get('/report_department_tech',urlencodedParser,db_tech.get_r
eport_department_tech)
app.post('/report_department_tech1',urlencodedParser,db_tech.rep
ort_department_tech1)
app.post('/show_report_by_tech_answer',urlencodedParser,db_tech.
show_report_by_tech_answer)
app.get('/show_report_by_tech_answer',urlencodedParser,db_tech.g
et_show_report_by_tech_answer)
app.post('/show_report_by_tech_answer1',urlencodedParser,db_tech
.show_report_by_tech_answer1)
app.post('/show_report_by_tech_answer2',urlencodedParser,db_tech
.show_report_by_tech_answer2)
app.post('/report_department_tech_true',urlencodedParser,db_tech
.report_department_tech_true)
app.get('/report_department_tech_true',urlencodedParser,db_tech.
get_report_department_tech_true)
app.post('/report_department_tech_false',urlencodedParser,db_tec
h.report_department_tech_false)
app.get('/report_department_tech_false',urlencodedParser,db_tech
.get_report_department_tech_false)
```

Лістинг Г.4 – POST та GET запити до контролера для технічного відділу, лист 2

```
app.post('/cabinet_inside_krui',urlencodedParser,db_krui.krui)
app.get('/cabinet_inside_krui',urlencodedParser,db_krui.get_krui
)
app.post('/cabinet_inside_krui1',urlencodedParser,db_krui.krui1)
app.get('/cabinet_inside_krui1',urlencodedParser,db_krui.get_kru
i1)
app.post('/message_captain_krui',urlencodedParser,db_krui.messag
e_captain_krui)
app.get('/message_captain_krui',urlencodedParser,db_krui.message
_captain_krui)
app.post('/message_captain_krui1',urlencodedParser,db_krui.messa
ge_captain_krui1)
app.post('/message_captain_krui2',urlencodedParser,db_krui.messa
ge_captain_krui2)
app.post('/report_captain_krui',urlencodedParser,db_krui.report_
captain_krui)
app.get('/report_captain_krui',urlencodedParser,db_krui.get_repo
rt_captain_krui)
app.post('/report_captain_krui1',urlencodedParser,db_krui.report
_captain_krui1)
```

Лістинг Г.5 – POST та GET запити до контролера для крюїнгового відділу, лист 1

```
app.post('/report_department_krui',urlencodedParser,db_krui.repo
rt_department_krui)
app.get('/report_department_krui',urlencodedParser,db_krui.get_r
eport_department_krui)
app.post('/report_department_krui1',urlencodedParser,db_krui.rep
ort_department_krui1)
app.post('/show_report_by_krui_answer',urlencodedParser,db_krui.
show_report_by_krui_answer)
app.get('/show_report_by_krui_answer',urlencodedParser,db_krui.g
et_show_report_by_krui_answer)
app.post('/show_report_by_krui_answer1',urlencodedParser,db_krui
.show_report_by_krui_answer1)
app.post('/show_report_by_krui_answer2',urlencodedParser,db_krui
.show_report_by_krui_answer2)
app.post('/report_department_krui_true',urlencodedParser,db_krui
.report_department_krui_true)
app.get('/report_department_krui_true',urlencodedParser,db_krui.
get_report_department_krui_true)
app.post('/report_department_krui_false',urlencodedParser,db_kru
i.report_department_krui_false)
app.get('/report_department_krui_false',urlencodedParser,db_krui
.get_report_department_krui_false)
```

Лістинг Г.5 – POST та GET запити до контролера для крюїнгового відділу, лист 2

```
app.post('/cabinet_inside_fin',urlencodedParser,db_fin.fin)
app.get('/cabinet_inside_fin',urlencodedParser,db_fin.get_fin)
app.post('/cabinet_inside_fin1',urlencodedParser,db_fin.fin1)
app.get('/cabinet_inside_fin1',urlencodedParser,db_fin.get_fin1)
app.post('/show_fin_report_oper',urlencodedParser,db_fin.show_fi
n_report_oper)
app.get('/show_fin_report_oper',urlencodedParser,db_fin.get_show
_fin_report_oper)
app.post('/show_fin_report_krui',urlencodedParser,db_fin.show_fi
n_report_krui)
app.get('/show_fin_report_krui',urlencodedParser,db_fin.get_show
_fin_report_krui)
app.post('/show_fin_report_tech',urlencodedParser,db_fin.show_fi
n_report_tech)
app.get('/show_fin_report_tech',urlencodedParser,db_fin.get_show
_fin_report_tech)
app.post('/analysis_all_ship',urlencodedParser,db_fin.analysis_a
ll_ship)
app.get('/analysis_all_ship',urlencodedParser,db_fin.get_analysi
s_all_ship)
app.post('/analysis_all_ship1',urlencodedParser,db_fin.analysis_
all_ship1)
app.post('/analysis_all_ship_count',urlencodedParser,db_fin.anal
ysis_all_ship_count)
```

Лістинг Г.6 – POST та GET запити до контролера для фінансового відділу, лист 1

```
app.get('/analysis_all_ship_count',urlencodedParser,db_fin.analy
sis_all_ship_count)
app.post('/analysis_all_ship_count1',urlencodedParser,db_fin.ana
lysis_all_ship_count1)
app.post('/message_captain_fin',urlencodedParser,db_fin.message_
captain_fin)
app.get('/message_captain_fin',urlencodedParser,db_fin.message_c
aptain_fin)
app.post('/message_captain_fin1',urlencodedParser,db_fin.message
_captain_fin1)
app.post('/message_captain_fin2',urlencodedParser,db_fin.message
_captain_fin2)
```

Лістинг Г.6 – POST та GET запити до контролера для фінансового відділу, лист 2

```
app.post('/cabinet_inside',urlencodedParser,db_admin.admin)
app.get('/cabinet_inside',urlencodedParser,db_admin.get_admin)
app.post('/create_user',urlencodedParser,db_admin.show_employe)
app.get('/create_user',urlencodedParser,db_admin.get_show_employ
e)
app.post('/create_user_success',urlencodedParser,db_admin.create
_user_success)
app.post('/update_user',urlencodedParser,db_admin.show_employe_u
p)
app.get('/update_user',urlencodedParser,db_admin.get_show_employ
e_up)
app.post('/update_user1',urlencodedParser,db_admin.show_employe_
up1)
app.post('/update_user2',urlencodedParser,db_admin.show_employe_
up2)
app.post('/delete_user',urlencodedParser,db_admin.show_employe_d
el)
app.get('/delete_user',urlencodedParser,db_admin.get_show_employ
e_del)
app.post('/delete_user_success',urlencodedParser,db_admin.show_e
mploye_del_success)
```

Лістинг Г.7 – POST та GET запити до контролера для адміністратора

```
app.post('/cabinet_inside_capitan',urlencodedParser,db_customer.
captain)
app.get('/cabinet_inside_capitan',urlencodedParser,db_customer.g
et_captain)
app.post('/cabinet_inside_capitan1',urlencodedParser,db_customer
.cap1)
app.get('/cabinet_inside_capitan1',urlencodedParser,db_customer.
get_cap1)app.post('/message_oper',urlencodedParser,db_customer.m
sg_oper)app.get('/message_oper',urlencodedParser,db_customer.get
_msg_орят)
```

Лістинг Г.8 – POST та GET запити до контролера для капітана, лист 1

```
app.post('/message_oper1',urlencodedParser,db_customer.msg_oper)
app.post('/message_krui',urlencodedParser,db_customer.msg_krui)
app.get('/message_krui',urlencodedParser,db_customer.get_msg_kru
i)
app.post('/message_krui1',urlencodedParser,db_customer.msg_krui1
)
app.post('/message_finance',urlencodedParser,db_customer.msg_fin
ance)
app.get('/message_finance',urlencodedParser,db_customer.get_msg_
finance)
app.post('/message_finance1',urlencodedParser,db_customer.msg_fi
nance1)
app.post('/message_tech',urlencodedParser,db_customer.msg_tech)
app.get('/message_tech',urlencodedParser,db_customer.get_msg_tec
h)
app.post('/message_tech1',urlencodedParser,db_customer.msg_tech1
)
app.post('/show_report',urlencodedParser,db_customer.show_report
)
app.get('/show_report',urlencodedParser,db_customer.get_show_rep
ort)
app.post('/show_report1',urlencodedParser,db_customer.show_repor
t1)
app.post('/show_report_answer',urlencodedParser,db_customer.show
_report_answer)
app.get('/show_report_answer',urlencodedParser,db_customer.get_s
how_report_answer)
app.post('/show_report_answer1',urlencodedParser,db_customer.sho
w_report_answer1)
app.post('/show_report_answer2',urlencodedParser,db_customer.sho
w_report_answer2)
app.post('/show_report_true',urlencodedParser,db_customer.show_r
eport_true)
app.get('/show_report_true',urlencodedParser,db_customer.get_sho
w_report_true)
app.post('/show_report_false',urlencodedParser,db_customer.show_
report_false)
app.get('/show_report_false',urlencodedParser,db_customer.get_sh
ow_report_false)
```

Лістинг Г.8 – POST та GET запити до контролера для капітана, лист 2

```
const pg = require("pg")
const client_operator1 = {
    user:'super',
    database:'ship_company',
    password: '12345',
    host:'localhost',
    port:5432,
    max:10,
    idleTimeoutMillis: 30000}
```

Лістинг Г.9 – Підключення до бази даних, лист 1

```
const client_captain1 = {
    user:'super',
    database:'ship_company',
    password: '12345',
    host:'localhost',
    port:5432,
    max:10,
    idleTimeoutMillis: 30000
}
const client_user1 = {
    user:'super',
    database:'ship_company',
    password: '12345',
    host:'localhost',
    port:5432,
    max:10,
    idleTimeoutMillis: 30000
}
const client_admin1 = {
    user:'super',
    database:'ship_company',
    password: '12345',
    host:'localhost',
    port:5432,
    max:10,
    idleTimeoutMillis: 30000
}
const listen1 =  4000

const client_login = {
    user:'super',
    database:'ship_company',
    password: '12345',
    host:'localhost',
    port:5432,
    max:10,
    idleTimeoutMillis: 30000
}

const client_tech1 = {
    user:'super',
    database:'ship_company',
    password: '12345',
    host:'localhost',
    port:5432,
    max:10,
    idleTimeoutMillis: 30000
}

const client_krui1 = {
    user:'super',
```

Лістинг Г.9 – Підключення до бази даних, лист 2

```
    database:'ship_company',
    password: '12345',
    host:'localhost',
    port:5432,
    max:10,
    idleTimeoutMillis: 30000
}

const client_fin1 = {
    user:'super',
    database:'ship_company',
    password: '12345',
    host:'localhost',
    port:5432,
    max:10,
    idleTimeoutMillis: 30000
}
module.exports ={
    client_operator1:client_operator1,
    client_fin1:client_fin1,
    client_krui1:client_krui1,
    client_tech1:client_tech1,
    client_captain1:client_captain1,
    client_user1:client_user1,
    client_admin1:client_admin1,
    listen1,
    client_login:client_login}
```

Лістинг Г.9 – Підключення до бази даних, лист 3

```
searchUser(req, res) {
const db = new pg.Pool(client_login.client_login)
db.connect(function(err, client, done){
console.log('db.connect')
if(err){
return console.error('Не вдаось з'єднання)
}
console.log('db.connect')
console.log(req.body.login)
console.log(req.body.password)
client.query("SELECT * FROM sea.signin WHERE login = $1 AND
password = replace(crypt($2, password), ' ',
'')",[req.body.login, req.body.password], function(err, result){
console.log('query')
done()
if (err || result.rowCount <= 0){
var chi = "Помилка. Можливо Ви ввели невірний логін або пароль.
Спробуйте ще раз"
res.render('error_login')
return console.error("запит не вийшов")
}
```

Лістинг Г.9 – Функцій для входу у персональний кабінет, лист 1

```
if(result.rows[0].position === "Captain")
{
cap.getClients(req,res)
}
else if  (result.rows[0].position === "Crewing_department"){
krui.getClients(req,res)
}
else if  (result.rows[0].position === "Financial_department"){
fin.getClients(req,res)
}
else if  (result.rows[0].position === "Ship_operator"){
oper.getClients(req,res)
}
else if  (result.rows[0].position === "Technical_department"){
tech.getClients(req,res)
}
else if  (result.rows[0].position === "Team"){
fam.getClients(req,res)
}
else if  (result.rows[0].position === "Admin"){
adm.getClients(req,res)
console.log("Увійшов як адмін")
}
else {
console.log("не увійшов")}})})}}
```

Лістинг Г.9 – Функцій для входу у персональний кабінет, лист 2