МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ЗАПОРІЗЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ

МАТЕМАТИЧНИЙ ФАКУЛЬТЕТ
Кафедра програмної інженерії

КВАЛІФІКАЦІЙНА РОБОТА МАГІСТРА

на тему: **«РОЗРОБКА ГЕОІНФОРМАЦІЙНОЇ СИСТЕМИ УПРАВЛІННЯ СУДНОПЛАВСТВОМ»**

Виконала: студентка 2 курсу, групи 8.1210-з

спеціальності 121 інженерія програмного забезпечення
(шифр і назва спеціальності)

освітньої програми інженерія програмного забезпечення
(назва освітньої програми)

О.Є. Єгорова
(ініціали та прізвище)

Керівник доцент кафедри програмної інженерії, к.ф.-м.н., Мильцев О.М.
(посада, вчене звання, науковий ступінь, прізвище та ініціали)

Рецензент завідувач кафедри фундаментальної та прикладної математики, доцент, д.т.н., Гребенюк С.М.
(посада, вчене звання, науковий ступінь, прізвище та ініціали)

Запоріжжя – 2021

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
«ЗАПОРІЗЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ»

Факультет математичний

Кафедра програмної інженерії

Рівень вищої освіти магістр

Спеціальність 121 інженерія програмного забезпечення
(шифр і назва)

Освітня програма інженерія програмного забезпечення

ЗАТВЕРДЖУЮ
Завідувач кафедри програмної інженерії, к.ф.-м.н., доцент
_____ Лісняк А.О.
(підпис)

“ ____ ” _____ 2021 р.

З А В Д А Н Н Я

НА КВАЛІФІКАЦІЙНУ РОБОТУ СТУДЕНТЦІ

Єгоровій Оксані Євгенівні
(прізвище, ім’я та по-батькові)

1. Тема роботи (проекту) Розробка геоінформаційної системи управління судноплавством

керівник роботи (проекту) Мильцев Олександр Михайлович, к.ф.-м.н.
(прізвище, ім’я та по-батькові, науковий ступінь, вчене звання)

затверджені наказом ЗНУ від « 09 » 06 2021 року № 850-с

2. Строк подання студентом роботи 25.11.2021

3. Вихідні дані до роботи 1. Постановка задачі.
2. Перелік літератури.

4. Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити)
1. Аналіз предметної області.
2. Проектування системи.
3. Розробка системи.

5. Перелік графічного матеріалу (з точним зазначенням обов’язкових креслень)
Презентація

6. Консультанти розділів роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		Завдання видав	Завдання прийняв
1	Мильцев О.М., старший викладач		
2	Мильцев О.М., старший викладач		
3	Мильцев О.М., старший викладач		
4	Мильцев О.М., старший викладач		
Додатки	Мильцев О.М., старший викладач		

7. Дата видачі завдання 09.06.2021

КАЛЕНДАРНИЙ ПЛАН

№	Назва етапів кваліфікаційної роботи	Строк виконання етапів роботи	Примітка
1.	Розробка плану виконання кваліфікаційної роботи магістра.	14.06.2021	
2.	Збір вихідних даних та аналіз предметної області.	21.06.2021	
3.	Обробка методичних та теоретичних джерел.	01.07.2021	
4.	Специфікація вимог до системи. Робота над першим розділом.	05.07.2021	
5.	Моделювання та проектування системи. Робота над другим розділом.	02.08.2021	
6.	Реалізація системи. Робота над третім розділом.	16.08.2021	
7.	Робота над четвертим розділом та додатками.	01.11.2021	
8.	Оформлення та нормоконтроль кваліфікаційної роботи магістра.	25.11.2021	
9.	Захист кваліфікаційної роботи магістра.	09.12.2021	

Студент _____ О.Є. Єгорова
(підпис) (ініціали та прізвище)

Керівник роботи _____ О.М. Мильцев
(підпис) (ініціали та прізвище)

Нормоконтроль пройдено

Нормоконтролер _____ С.П. Швидка
(підпис) (ініціали та прізвище)

РЕФЕРАТ

Кваліфікаційна робота магістра «Розробка геоінформаційної системи управління судноплавством»: 89 с., 75 рис., 19 табл., 9 джерел, 3 додатки.

BACK-END, FRONT-END, LARAVEL, ГЕОІНФОРМАЦІЙНА СИСТЕМА, ЗАПЛАНОВАНИЙ МАРШРУТ, МАПА, МАРШРУТ, ПОЇЗДКА, СУДНО, ФАКТИЧНИЙ МАРШРУТ.

Об’єкт дослідження – процес судноплавства.

Мета роботи: розробка геоінформаційної системи управління судноплавством.

Методи дослідження – методи збору та аналізу вимог до програмного забезпечення, методи моделювання та проектування програмного забезпечення.

У кваліфікаційній роботі розглядається процес судноплавства, а саме: планування маршруту, відстеження фактичного маршруту та актуальної позиції судна, контроль виконання контрактів денної продуктивності.

У процесі розробки геоінформаційної системи було проведено аналіз предметної області, обрана архітектура і технології для реалізації, створено веб-додаток з використанням фреймворку Laravel. В результаті роботи отримано програмне забезпечення (геоінформаційна система) для управління судноплавством.

SUMMARY

Master’s qualifying paper «Development of Geoinformation System of Shipping Navigation»: 89 pages, 75 figures, 19 tables, 9 references, 3 supplements.

BACK-END, FRONT-END, LARAVEL, GEOINFORMATION SYSTEM, PLANNED ROUTE, MAP, ROUTE, TRIP, SHIP, ACTUAL ROUTE.

The object of the study is the process of shipping navigation.

The aim of the study is the development of a geoinformation system of shipping management.

The methods of research are collecting and analyzing software requirements, methods of modeling and designing software.

The qualification work considers the process of shipping navigation, namely: route planning, tracking the actual route and the current position of the vessel, monitoring the implementation of daily productivity contracts.

In the process of developing the geoinformation system, the subject area was analyzed, the architecture and technologies for implementation were selected, and a web application was created using the Laravel framework. As a result, software (geoinformation system) for shipping management was obtained.

ЗМІСТ

ВСТУП

Сучасне життя людини неможливо уявити без судноплавства.

Водним транспортом перевозиться більшість вантажів по всьому світу, а також здійснюється велика кількість пасажирських перевезень. Велику роль судноплавство грає і в риболовстві.

Сьогодні безпека, як і раніше, є головною турботою для судновласників та судноплавних компаній. Не зважаючи на те, що водний транспорт володіє якостями, які визначають його здатність безпечно здійснювати плавання за будь-якого стану моря та будь-якої погоди, а також забезпечують «живучість» у разі пошкодження, потрапляння судна в шторм є вкрай небезпечним та небажаним, тому дуже важливою задачею є зменшення цього ризику.

Крім цього, існує низка економічних факторів, пов'язаних з маршрутизацією судів. Вибір маршруту, що уникає штормів, сильних підводних течій та й просто злив та високих хвиль може доставити судно до місця призначення набагато швидше, заощадивши час та паливо.

Таким чином, процес планування маршруту судна є чи не найважливішим. Цей процес потребує прокладення маршруту за допомогою створення шляхових точок на мапі, розрахунку швидкості судна (на основі необхідного часу прибуття) або часу прибуття (використовуючи надані швидкості), часу, коли судно повинно дістатися кожної шляхової точки, координат у різних форматах, дистанції до попередньої та наступної шляхової точки у морських милях, курсу до наступної шляхової точки, а також відстані, отримання погодних даних для маршруту та прийняття остаточного рішення на основі розрахунків та отриманих погодних даних.

Отже, процес планування маршруту є об’ємним та досить складним, потребує уважності та значної кількості часу, тому автоматизація цього процесу є актуальною.

Таким чином, метою кваліфікаційної роботи є розробка геоінформаційної системи для управління судноплавством, яка скорочує ресурси часу, зменшує ризик виникнення помилок та спрощує процеси планування маршруту, відстеження фактичного маршруту та контролю виконання контрактів денної продуктивності.

1 ТЕХНІЧНЕ ЗАВДАННЯ

1.1 Терміни та визначення

Геоінформаційна система (ГІС) – комп'ютерна система, що забезпечує можливість використання, збереження, редагування, аналізу та відображення географічних даних.

База даних (БД) – місце збереження інформації ГІС.

Судно – вид транспорту, що виконує перевезення вантажів і пасажирів по водних шляхах, як природних (ріки, озера, моря, океани, протоки), так і штучних (канали, водосховища).

Маршрут – шлях, який судну необхідно подолати, щоб опинитися з пункту відправки в пункті призначення.

Поїздка – рух судна за запланованим маршрутом.

Порт – ділянка берега моря, озера, водосховища або ріки і прилегла водна площа, штучно або природно захищені від хвилювання, обладнані для стоянки та обслуговування суден.

Фронтенд (англ. front-end) – клієнтська сторона інтерфейсу користувача до програмно-апаратної частини сервісу.

Бекенд (англ. back-end) – програмно-апаратна частина сервісу, що відповідає за функціонування його внутрішньої частини.

Актуальний або фактичний маршрут – частина маршруту, який вже пройшло судно.

Запланований маршрут – частина маршруту, який ще залишилося пройти.

Шляхові точки – точки на мапі, за якими буде здійснена поїздка.

1.2 Загальні положення

1.2.1 Призначення і цілі створення системи

Функціональне призначення системи – планування маршруту судна, відстеження актуального маршруту судна, контроль виконання контрактів.

Експлуатаційне призначення системи: може експлуатуватися персоналом судноплавної компанії та персоналом інших підприємств водного транспорту.

Мета створення системи – спрощення процесу планування маршруту, скорочення ресурсів часу, зменшення ризику виникнення помилок.

1.2.2 Загальні функціональні можливості системи

ГІС управління судноплавством має надавати користувачам такі можливості [1]:

- управління (створення, редагування та видалення) користувачами;
- управління компаніями;
- управління країнами;
- управління портами;
- управління класифікаційними товариствами суден;
- управління паливом для судна;
- управління суднами;
- управління судноплавними шляхами, включаючи прокладання маршруту судна за допомогою шляхових точок на мапі;
- виконання необхідних розрахунків на основі даних судна та звітів (загальна відстань в милях, розрахунковий час прибуття, швидкість, відстань між точками маршруту, курс до наступної точки, відстань до місця прибуття та від місця відправки);
- відображення актуальних позицій судна на мапі на підставі наданих звітів капітана корабля;

- відображення різних погодних даних (вітру, температури, хвиль, течій тощо) на мапі;
- отримання даних погоди таких як:
    1) напрямок та швидкість (в вузлах) вітру;
    2) висота (в метрах) та напрямок хвилі;
    3) висота (в метрах), напрямок та період (в секундах) вітрової хвилі;
    4) напрямок, висота в метрах та період (в секундах) зибу (дрібних хвиль);
    5) тиск в гектопаскалях;
    6) температура повітря у градусах за цельсієм;
    7) напрямок та швидкість (в кілометрах за годину) дрейфу.
- розрахунок швидкості на основі необхідного часу прибуття;
- розрахунок часу прибуття, використовуючи надані швидкості;
- перегляд змін погодних умов протягом поїздки на мапі за допомогою кнопки плей та повзунка;
- відображення поїздок (активних маршрутів) на мапі;
- відправка звітів капітана, таких як:
    1) звіт про відправлення;
    2) щоденний обов’язковий звіт опівдні;
    3) звіт про прибуття;
    4) звіт про зупинку;
    5) звіт про відновлення руху.

Система повинна забезпечувати контроль виконання контрактів денної продуктивності та, у разі її зменшення, мати можливість попередити про це співробітників компанії, надіславши їм відповідного листа на пошту.

Крім цього, в системі має бути можливість збереження звітів про поїздку у вигляді файлів (з можливістю їх подальшого завантаження) таких типів:

1) прогноз погоди;
2) проміжний звіт;

3) фінальний звіт.

Також система має підтримувати експорт шляхових точок маршруту з отриманими погодними даними.

Для ефективного та зручного використання системи вона має надавати користувачам системи можливість фільтрації, сортування та пошуку даних.

1.3 Функціональні вимоги

У системі повинні бути такі ролі користувача [1]:

- адміністратор системи;
- адміністратор компанії;
- оператор компанії;
- капітан корабля.

Кожен користувач системи повинен мати доступ до відповідних функціональних можливостей згідно ролі.

Адміністратор системи повинен мати доступ до усіх функціональних можливостей без винятку.

Адміністратор компанії повинен мати доступ до управління користувачами та суднами компанії, а також повний доступ до управління маршрутами, включаючи відправку усіх звітів та контроль виконання контрактів денної продуктивності.

Оператор компанії та капітан корабля повинні мати доступ до перегляду списку суден та інформації щодо певного судна, а також повний доступ до управління маршрутами, включаючи відправку усіх звітів та контроль виконання контрактів денної продуктивності.

1.4 Нефункціональні вимоги

1.4.1 Інтерфейс користувача

Система повинна відображати коректно інтерфейс користувача на будь-якому пристрої.

1.4.2 Підтримка браузерів

Система повинна працювати для наступних браузерів останніх версій:

- MS Internet Explorer;
- Mozilla Firefox;
- Google Chrome;
- Safari;
- Opera.

1.5 Вимоги до продуктивності

Система повинна відображати будь-яку форму не довше, чим за 10 секунд.
Система повинна відображати будь-яку мапу не довше, чим за 10 секунд.

1.6 Вимоги до безпеки

Система повинна надавати доступ до функціоналу враховуючи роль користувача у системі.

Система не повинна надавати доступ до даних неавторизованим користувачам.

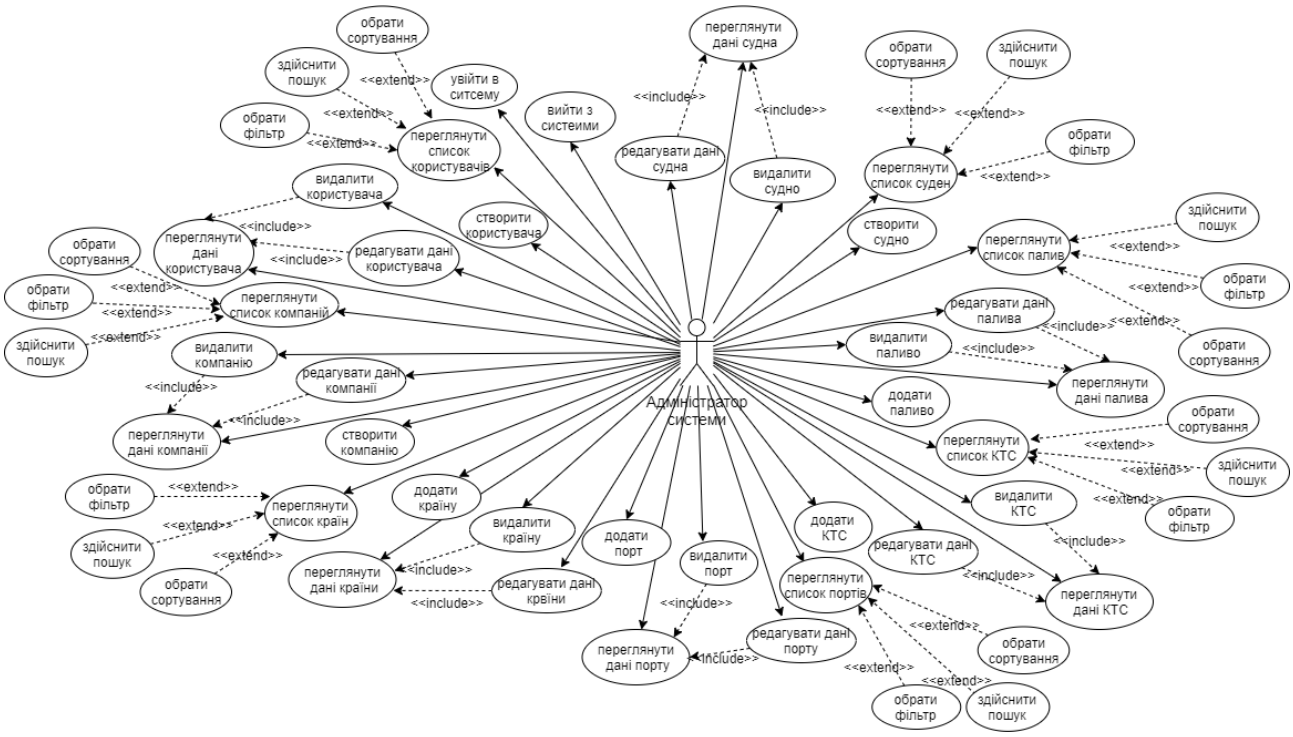
2 ПРОЕКТУВАННЯ

2.1 Проектування системи

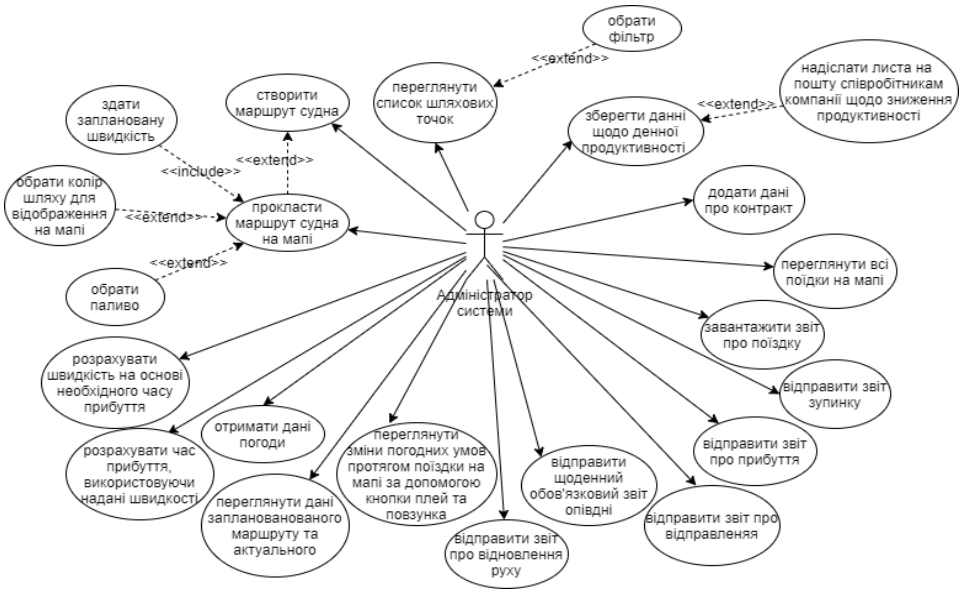
2.1.1 Етап концептуального проектування

Функціональні вимоги до системи представлені на діаграмах варіантів використання (див. рис. 2.1 – 2.4).

На ДДВ (діаграмі варіантів використання) адміністратором системи (див. рис. 2.1) зображені функціональні можливості, доступ до яких має адміністратор системи.



а) перша частина



б) друга частина

Рисунок 2.1 – ДВВ адміністратором системи

На ДДВ адміністратором компанії (див. рис. 2.2) зображені функціональні можливості, доступ до яких має адміністратор компанії.

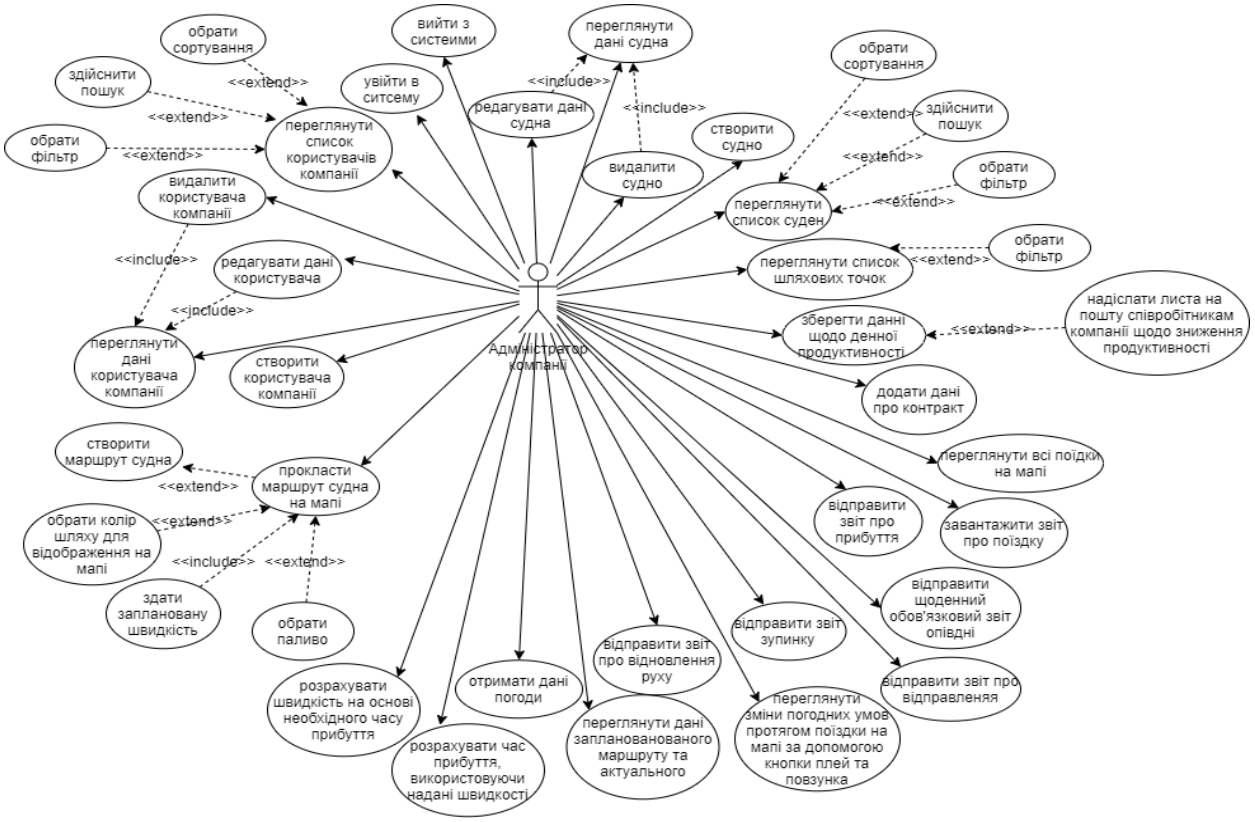


Рисунок 2.2 – ДВВ адміністратором компанії

На ДДВ оператором компанії (див. рис. 2.3) зображені функціональні можливості, доступ до яких має оператор компанії.

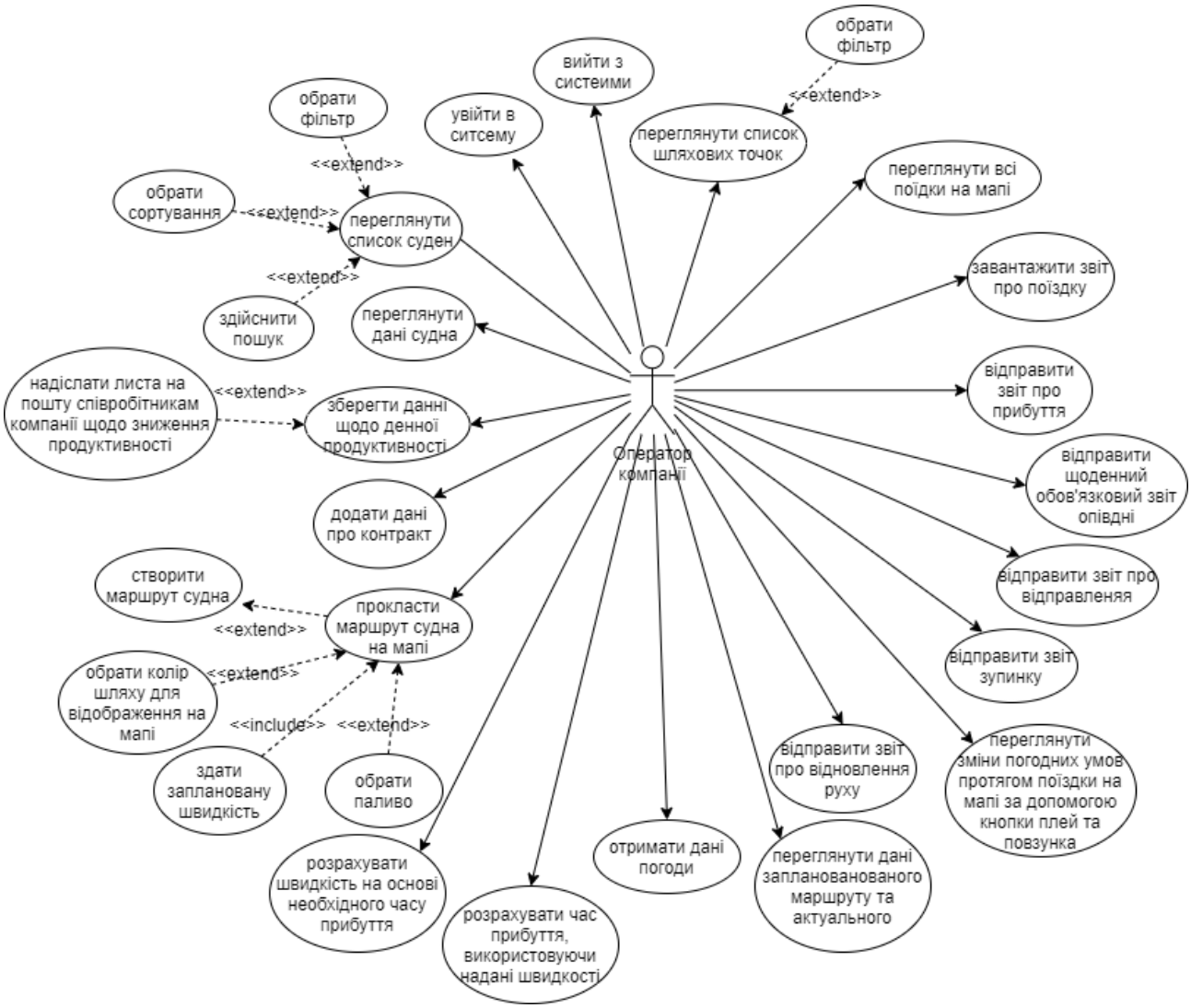


Рисунок 2.3 – ДВВ оператором компанії

На ДДВ капітаном корабля (див. рис. 2.4) зображені функціональні можливості, доступ до яких має капітан корабля.

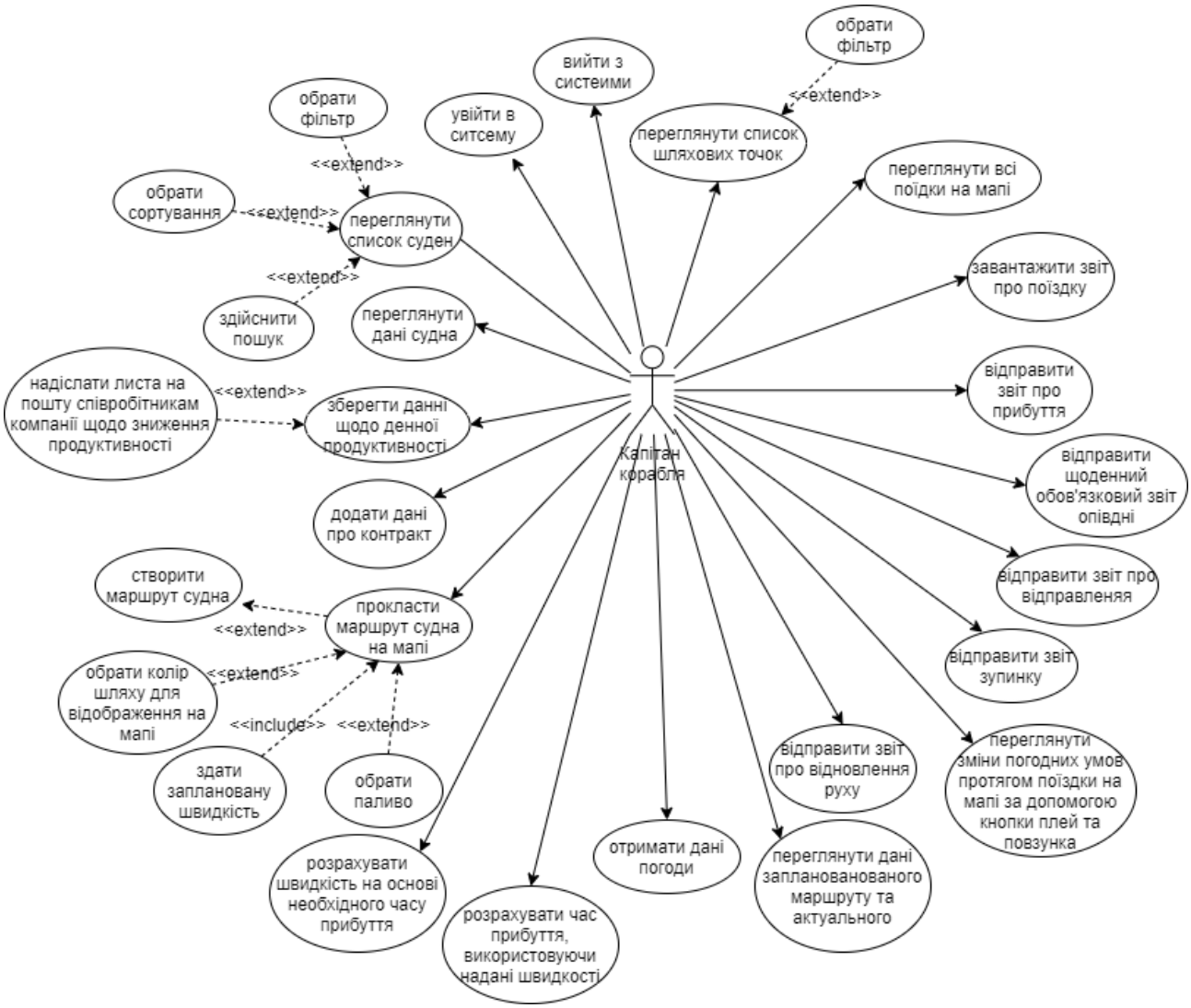


Рисунок 2.4 – ДВВ капітаном корабля

2.1.2 Етап логічного проектування

Для реалізації системи обрано фреймворк Laravel, тому при проектуванні враховуються архітектурні концепції цього фреймворку.

На діаграмі класів ядра HTTP (див. рис. 2.5) зображені класи, які містять основну логіку процесу обробки HTTP-запитів.

Клас Kernel – це ядро HTTP, яке визначає список проміжного програмного забезпечення HTTP, через яке всі запити повинні пройти, перш ніж вони будуть оброблені додатком.

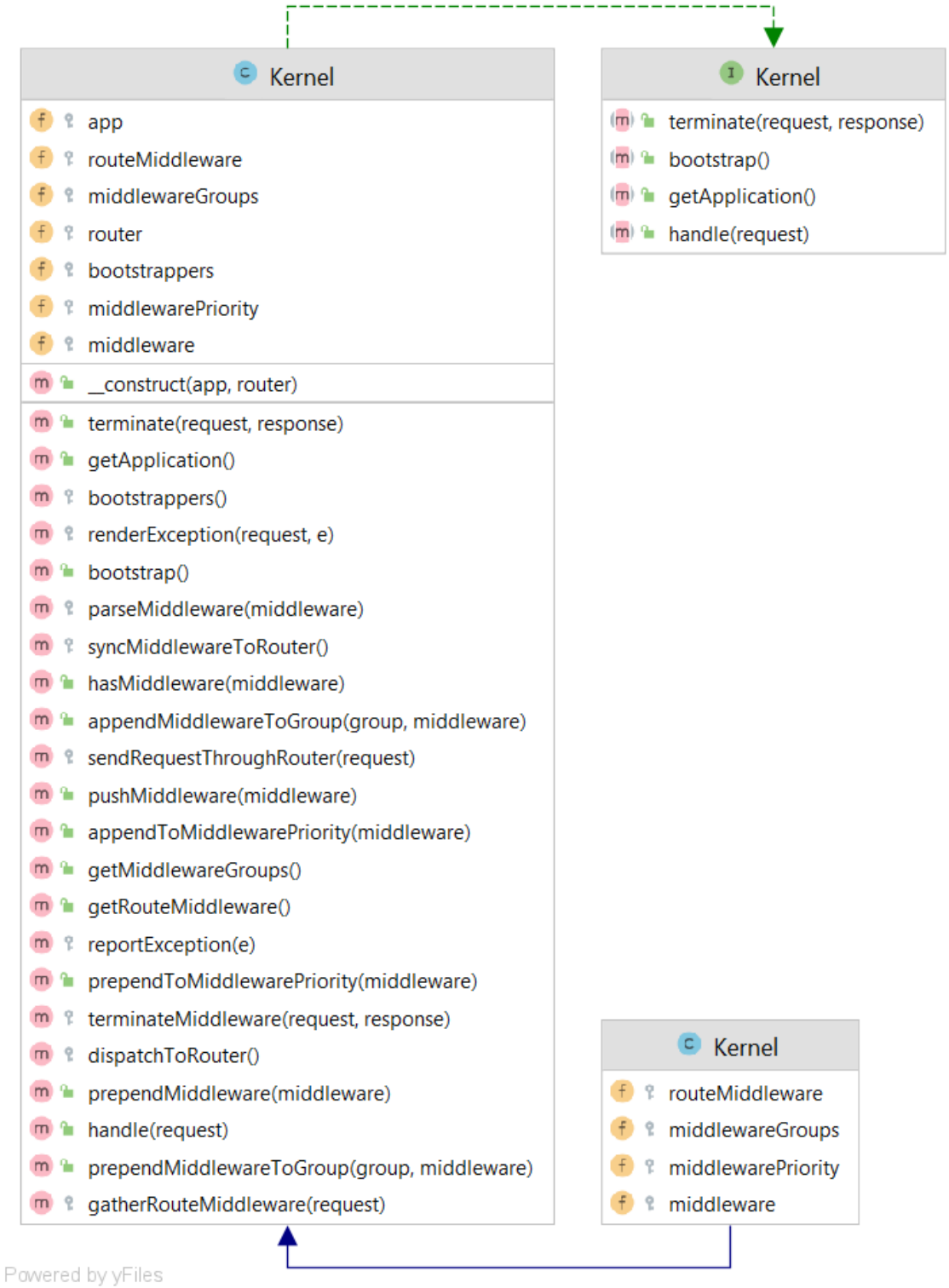


Рисунок 2.5 – Діаграма класів ядра HTTP

На діаграмі класів маршрутизації (див. рис. 2.6) зображено сам клас маршрутизації Router, який дозволяє реєструвати маршрути для будь-якого HTTP-запиту, а також допоміжні для цього класи.

Рисунок 2.6 – Діаграма класів маршрутизації

На діаграмі класів-посередників (див. рис. 2.7) зображені класи, які відповідають за обробку читання і запису сеансу HTTP, перевіряють токен CSRF і багато іншого.

Посередник забезпечує зручний механізм для перевірки та фільтрації HTTP-запитів, що надходять у систему.

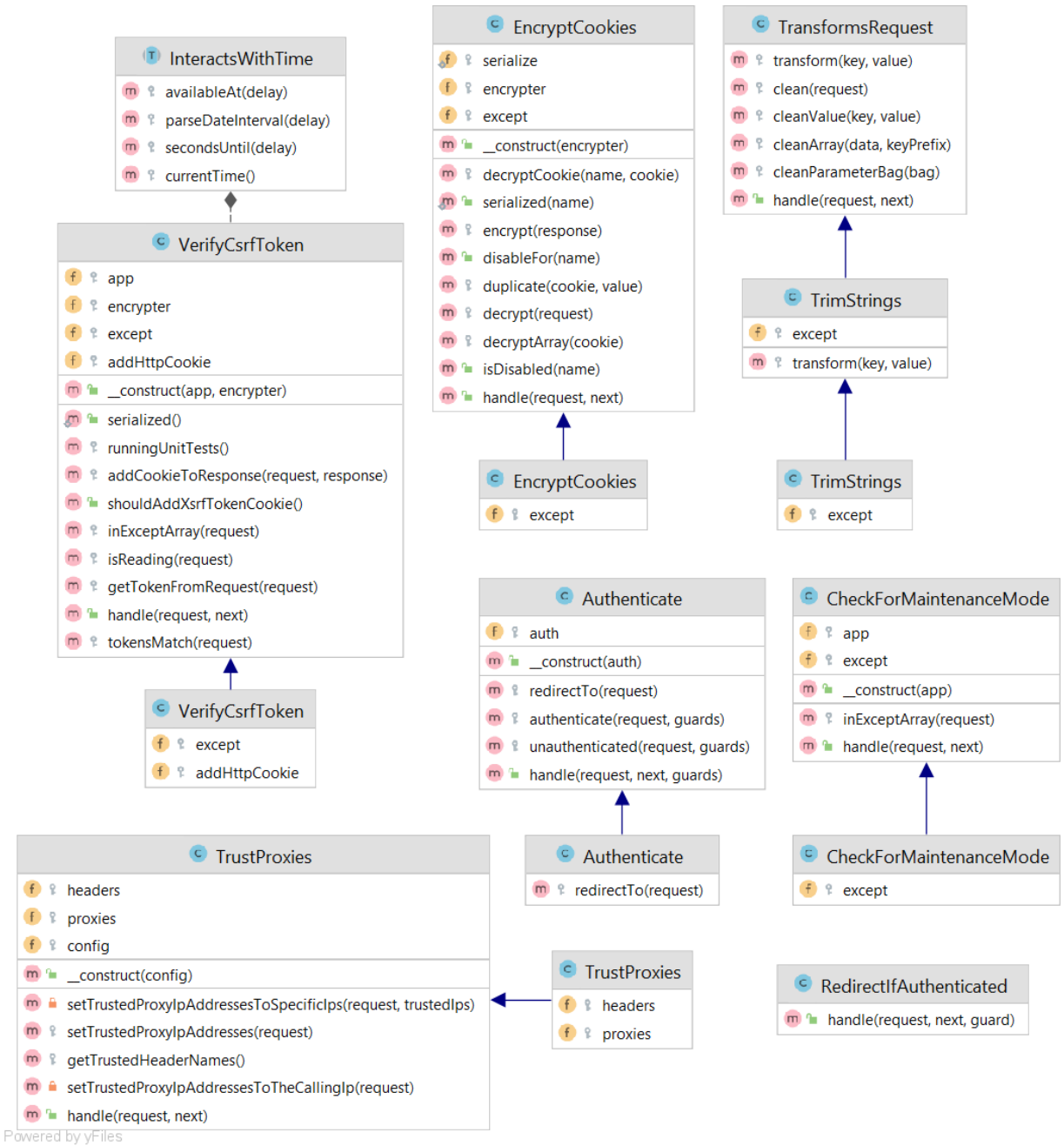


Рисунок 2.7 – Діаграма класів-посередників

На діаграмі класів авторизації (див. рис. 2.8) зображені класи, які відповідають за авторизацію користувача у системі, його реєстрацію у системі, верифікацію та інше.

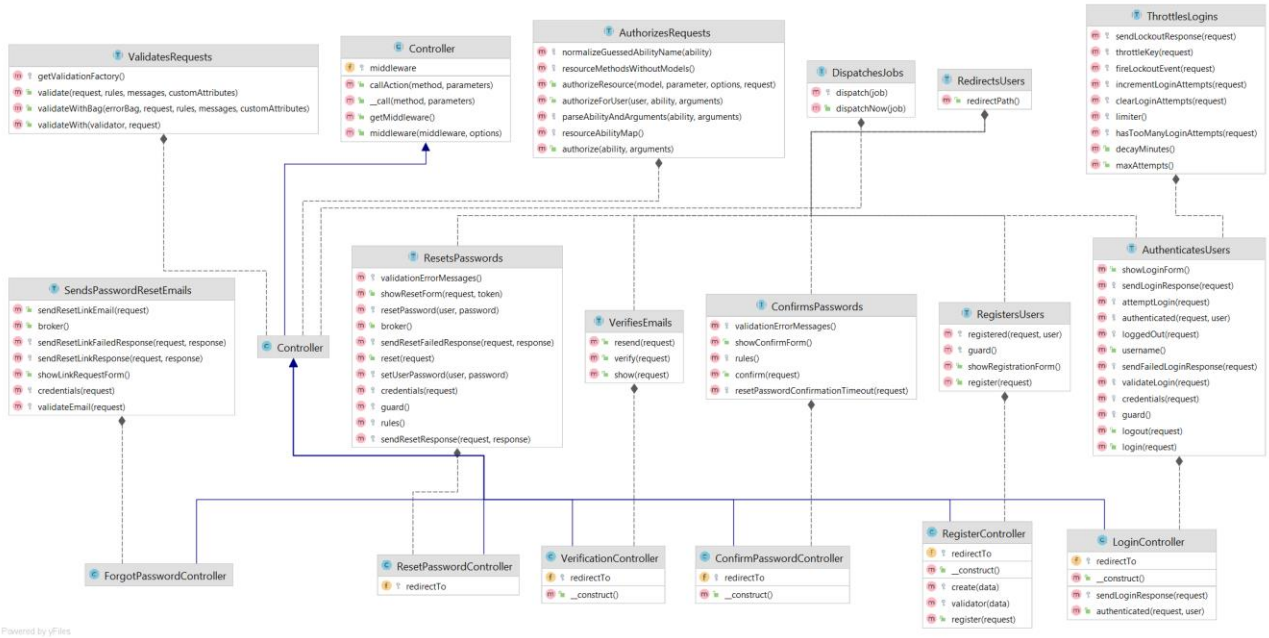


Рисунок 2.8 – Діаграма класів авторизації

Логіка обробки запитів знаходиться у контролерах (див. рис. 2.9).

Контролери можуть згрупувати пов'язану логіку обробки запитів до одного класу. Наприклад, клас UserController обробляє всі вхідні запити, що стосуються користувачів, включаючи відображення, створення, оновлення та видалення користувачів, а клас TripController обробляє всі вхідні запити, що стосуються маршруту, включаючи відображення, створення, оновлення та видалення маршруту, а також збереження шляхових точок.

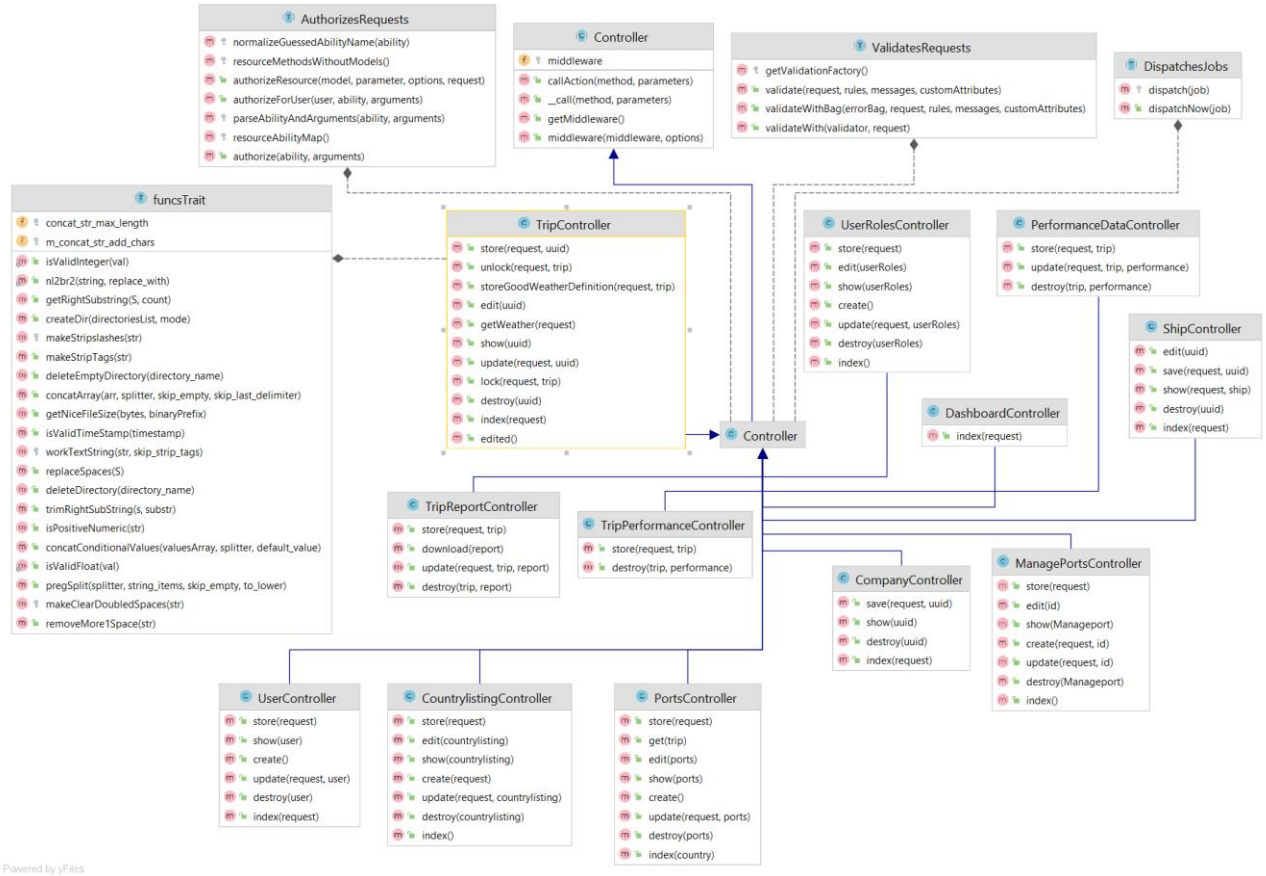


Рисунок 2.9 – Діаграма класів контролерів

Сервіс-провайдери займають центральне місце при початковому завантаженні всіх програм Laravel (див. рис. 2.10).

Під «початковим завантаженням» мається на увазі реєстрація елементів, включаючи реєстрацію зв'язувань контейнера служб (service container), слухачів подій (event listener), посередників (middleware) і навіть маршрутів (route).

Laravel містить ORM-бібліотеку Eloquent, яка надає можливість роботи з БД, яка часто зручніша за звичайний будівельник запитів. При використанні Eloquent кожна таблиця БД має відповідну модель, яка використовується для взаємодії з цією таблицею. Крім отримання записів з таблиці БД, моделі Eloquent також дозволють вставляти, оновлювати та видаляти записи з таблиці.

Всі моделі системи зображені на діаграмі класів моделей (див. рис. 2.11).

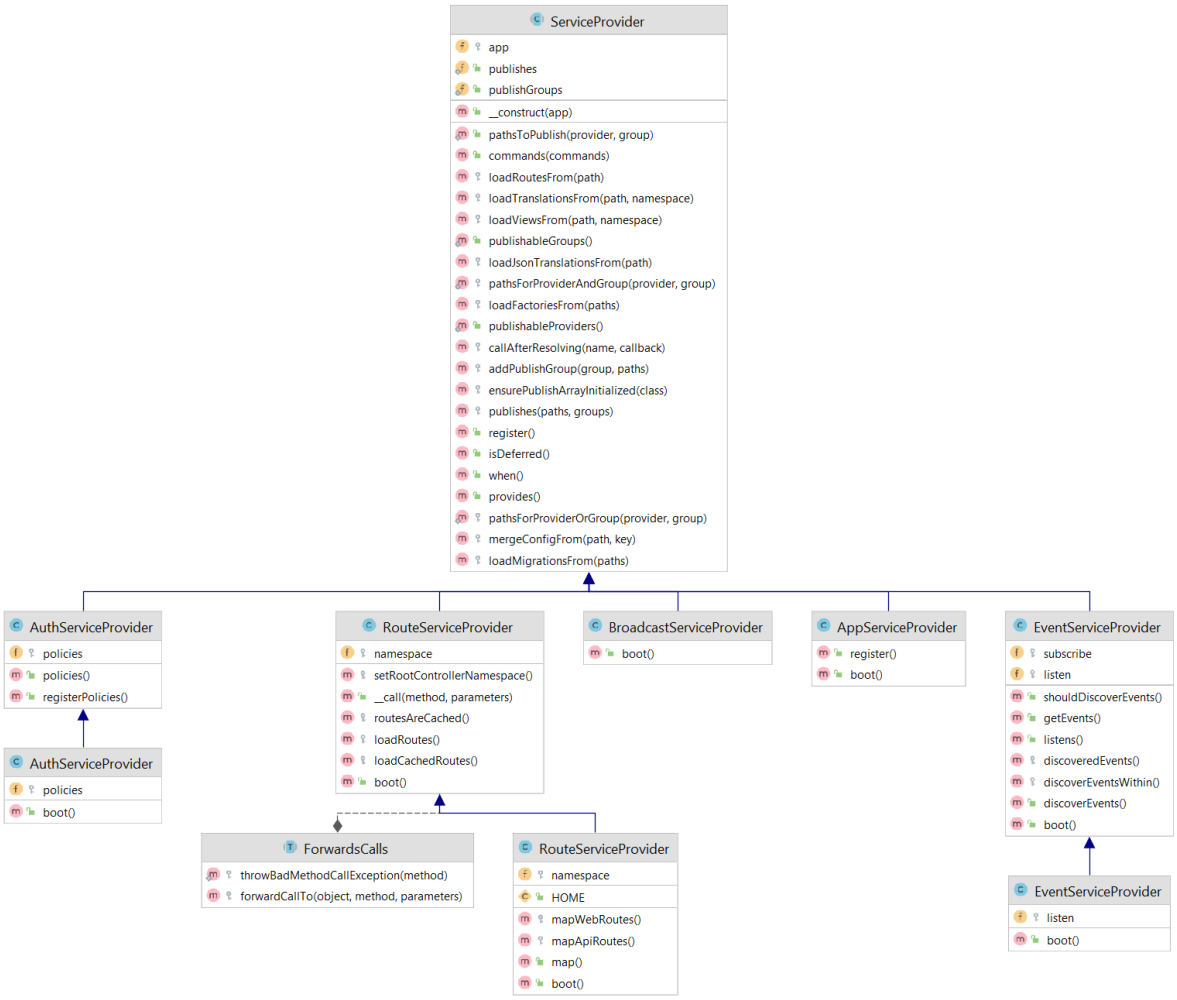


Рисунок 2.10 – Діаграма класів сервіс-провайдерів

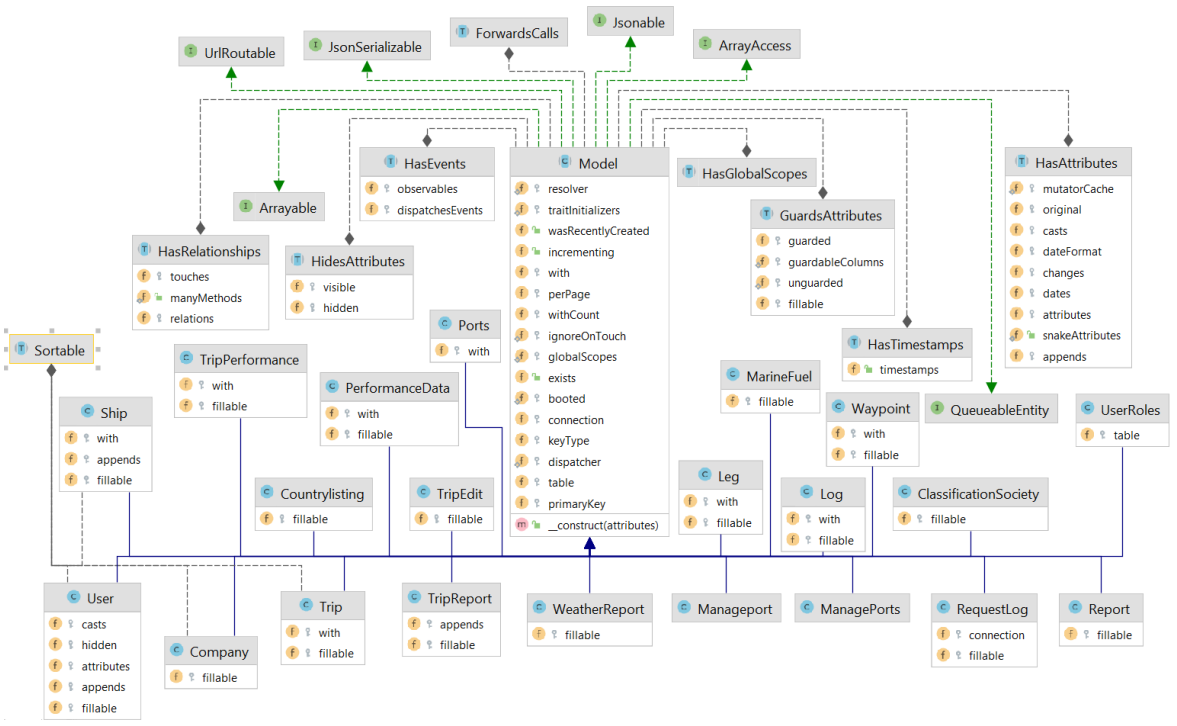


Рисунок 2.11 – Діаграма класів моделей

У фреймфорку Laravel обробка помилок та винятків вже налаштована (див. рис. 2.12). Клас Handler – це те місце, де всі винятки, створені програмою, реєструються і потім відображаються користувачу.

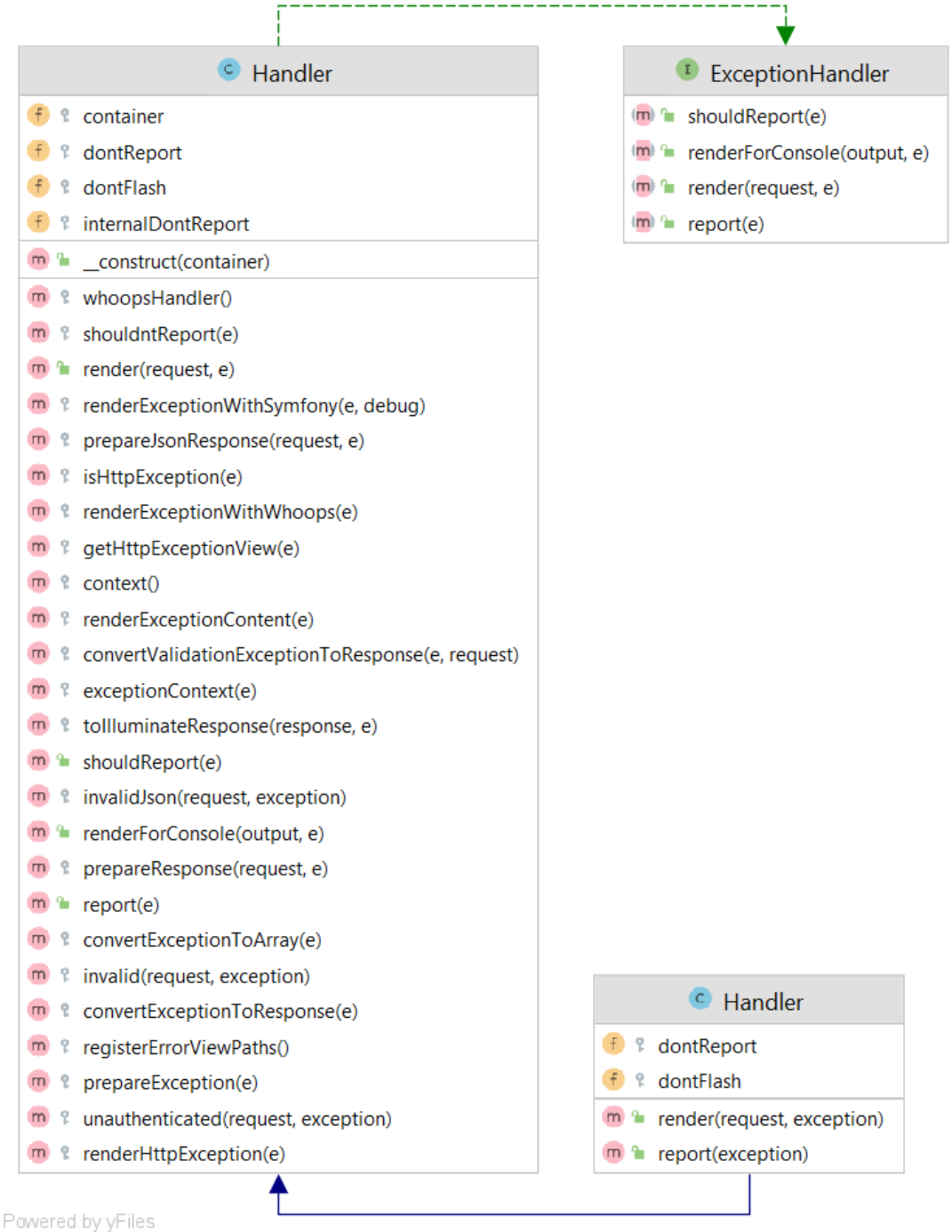


Рисунок 2.12 – Діаграма класів обробки помилок

На діаграмі класів повідомлень зображені класи, які відповідають за відправку повідомлень на пошту (див. 2.13).

Рисунок 2.13 – Діаграма класів повідомлень

Роботу з глобальним API погоди stormglass.io доцільно винести в окремий клас (див. рис. 2.14).

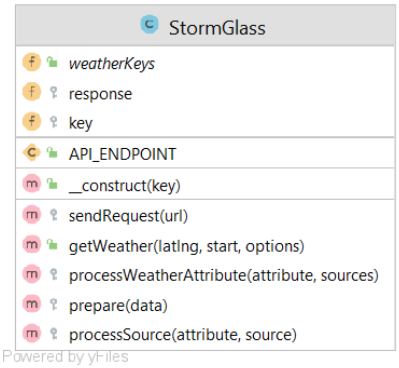


Рисунок 2.14 – Діаграма класу отримання погодних даних

2.1.3 Етап фізичного проектування

Структура програми Laravel (див. рис. 2.16) за умовчанням призначена для забезпечення відмінної відправної точки як для великих, так і для невеликих додатків.

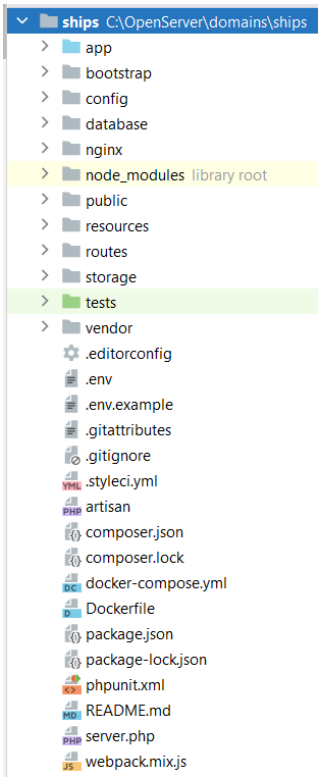


Рисунок 2.16 – Структура фреймворку Laravel

Розглянемо основні каталоги проекту:

- каталог app містить основний код програми;
- каталог bootstrap містить файл app.php, який завантажує фреймворк, а також каталог cache, що містить файли, згенеровані фреймворком для оптимізації продуктивності, наприклад, файли кеша маршрутів і служб;
- каталог config містить усі файли конфігурації;
- каталог database містить міграції баз даних, фабрики моделей та наповнювачі;
- каталог public містить файл index.php, який є точкою входу для всіх запитів, що надходять у програму, і конфігурує автозавантаження;
- каталог resources містить шаблони, а також необроблені, скомпільовані ресурси, наприклад, JavaScript або CSS;
- каталог routes містить всі визначення маршрутів для програми;
- каталог storage містить журнали (логи), скомпільовані шаблони Blade, файли сесій, кеша та інші файли, створені фреймворком;
- каталог tests містить автоматизовані тести;
- каталог vendor містить Composer-залежності.

Більшість програми знаходиться в каталозі app. За промовчанням цей каталог знаходиться в просторі імен App і автоматично завантажується Composer за допомогою автозавантажувача стандарту PSR-4.

Каталог App містить безліч додаткових каталогів:

- Console містить усі ваші Artisan-команди програми;
- у каталозі Events знаходяться класи подій;
- Exceptions містить обробник винятків програми, а також є гарним місцем для розміщення будь-яких винятків, що генеруються програмою;
- Http містить контролери, посередники та запити форм;

- Mail містить усі класи для роботи з електронними листами, що надсилаються додатком;
- Models містить усі класи моделей Eloquent;
- Providers містить усіх постачальників служб програми;
- Rules містить об'єкти користувача правил валідації програми.

Фізичне представлення системи не може бути повним, якщо відсутня інформація про те, на якій платформі і на яких обчислювальних засобах вона реалізована. Саме тому була розроблена діаграма розгортання (див. рис. 2.17).

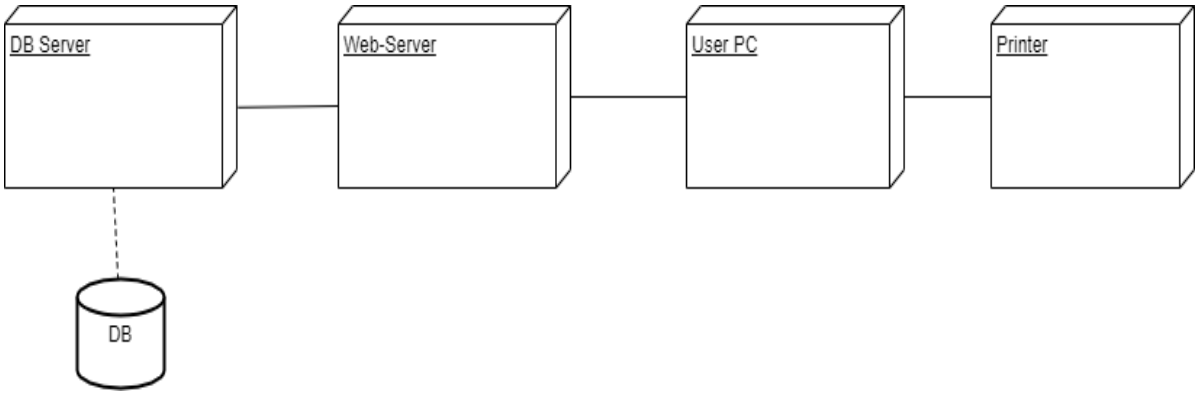


Рисунок 2.17 – Діаграма розгортання

На діаграмі розгортання зображені вузли, які є фізично існуючими елементами системи – сервер, сервер бази даних, персональний комп’ютер, принтер.

2.2 Проектування бази даних

2.2.1 Етап концептуального проектування

На цьому етапі була створена інформаційна модель даних системи (див. рисунок 2.18) [2].

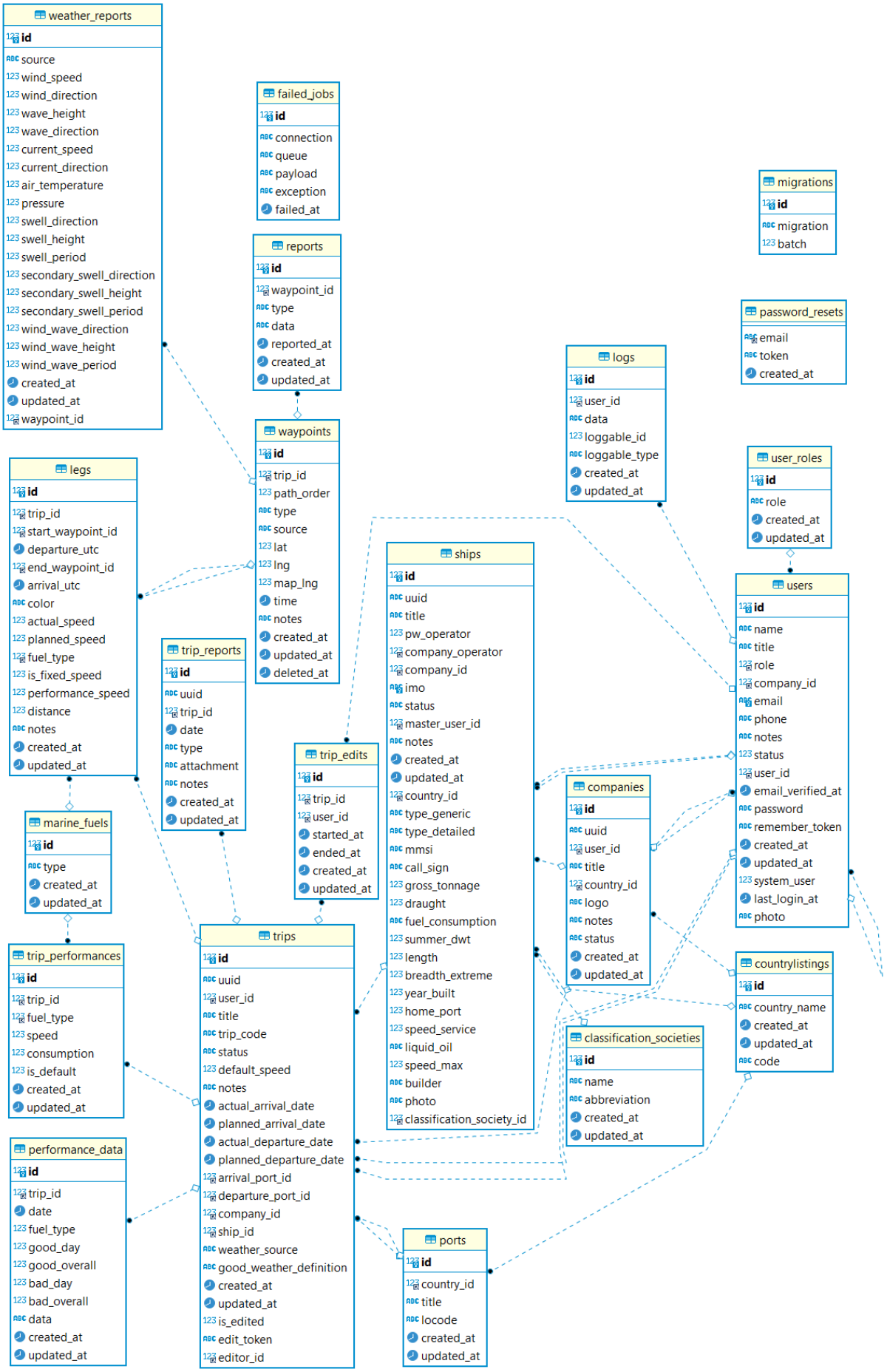


Рисунок 2.18 – ER-модель даних системи

2.2.2 Етап логічного проектування

Розглянемо детальніше кожну таблицю інформаційної моделі системи.

У таблиці «classification_societies» зберігається перелік установ, що займаються реєстрацією судів та оцінкою їх якостей за допомогою інституту сюрвеєрів на основі розроблюваних товариствами правил побудови судів різних типів (див. табл. 2.1).

Таблиця 2.1 – Опис таблиці БД «Класифікаційне товариство судів»

Назва поля	Тип даних	Властивість	Null	Опис
id	bigint(20)	Primary key (PK), Autoincrement	Ні	Ідентифікатор
name	varchar(255)	Unique key	Ні	Назва
abbreviation	varchar(255)		Ні	Абревіатура
created_at	timestamp		Ні	Дата створення
updated_at	timestamp		Ні	Дата оновлення

У таблиці «companies» зберігаються компанії (див. табл. 2.2).

Таблиця 2.2 – Опис таблиці БД «Компанія»

Назва поля	Тип даних	Властивість	Null	Опис
id	bigint(20)	Primary key (PK), Autoincrement	Ні	Ідентифікатор
uuid	char(36)	Unique key	Ні	Універсальний унікальний ідентифікатор
user_id	bigint(20)	Foreign key (FK)	Ні	Ідентифікатор користувача, який створив компанію
title	varchar(255)	Unique key	Ні	Назва
country_id	bigint(20)	Foreign key (FK)	Ні	Ідентифікатор країни
logo	varchar(255)		Так	Лого
notes	text		Так	Нотатки
status	enum('active', 'suspended', 'pending')		Ні	Статус
created_at	timestamp		Ні	Дата створення
updated_at	timestamp		Ні	Дата оновлення

У таблиці «countrylistings» зберігаються країни (див. табл. 2.3).

Таблиця 2.3 – Опис таблиці БД «Країна»

Назва поля	Тип даних	Властивість	Null	Опис
id	bigint(20)	Primary key (PK), Autoincrement	Ні	Ідентифікатор
country_name	varchar(255)	Unique key	Ні	Назва
code	varchar(2)	Unique key	Ні	Абревіатура
created_at	timestamp		Ні	Дата створення
updated_at	timestamp		Ні	Дата оновлення

У таблиці «legs» зберігається перелік частин маршрутів (див. табл. 2.4).

Таблиця 2.4 – Опис таблиці БД «Частина маршруту»

Назва поля	Тип даних	Властивість	Null	Опис
id	bigint(20)	Primary key (PK), Autoincrement	Ні	Ідентифікатор
trip_id	bigint(20)	Foreign key (FK)	Ні	Ідентифікатор маршруту
start_waypoint_id	bigint(20)	Foreign key (FK)	Ні	Ідентифікатор початкової маршрутної точки
departure_utc	datetime		Так	Дата та час відправлення
end_waypoint_id	bigint(20)	Foreign key (FK)	Ні	Ідентифікатор Кінцевої маршрутної точки
arrival_utc	datetime		Так	Дата та час прибуття
color	varchar(7)		Ні	Колір маршруту для відображення на мапі
actual_speed	double(8,2)		Так	Фактична швидкість
planned_speed	double		Ні	Запланована швидкість
fuel_type	bigint(20)		Так	Ідентифікатор палива
is_fixed_speed	tinyint(1)		Ні	Чи є фіксованою швидкість
distance	double(8,2)		Ні	Відстань
notes	text		Так	Нотатки
created_at	timestamp		Ні	Дата створення
updated_at	timestamp		Ні	Дата оновлення

У таблиці «logs» зберігаються логи системи (див. табл. 2.5).

Таблиця 2.5 – Опис таблиці БД «Логи»

Назва поля	Тип даних	Властивість	Null	Опис
id	bigint(20)	Primary key (PK), Autoincrement	Ні	Ідентифікатор
user_id	bigint(20)	Foreign key (FK)	Ні	Ідентифікатор користувача
data	longtext		Ні	Дані
loggable_id	bigint(20)		Ні	Ідентифікатор типу логів
loggable_type	varchar(255)		Ні	Тип логу
created_at	timestamp		Ні	Дата створення
updated_at	timestamp		Ні	Дата оновлення

У таблиці «performance_data» зберігаються дані контрактів денної продуктивності (див. табл. 2.6).

Таблиця 2.6 – Опис таблиці БД «Дані контрактів денної продуктивності»

Назва поля	Тип даних	Властивість	Null	Опис
id	bigint(20)	Primary key (PK), Autoincrement	Ні	Ідентифікатор
trip_id	bigint(20)	Unique key	Ні	Ідентифікатор маршруту
date	date	Foreign key (FK)	Так	Дата
fuel_type	bigint(20)		Так	Ідентифікатор палива
good_day	double(8,2)		Так	Швидкість для дня з хорошою погодою
good_overall	double(8,2)		Так	Швидкість для дня з в цілому хорошою погодою
bad_day	double(8,2)		Так	Швидкість для дня з поганою погодою
bad_overall	double(8,2)		Так	Швидкість для дня з в цілому поганою погодою
data	longtext		Так	(DC2Type:json)
created_at	timestamp		Ні	Дата створення
updated_at	timestamp		Ні	Дата оновлення

У таблиці «marine_fuels» зберігається перелік палив суден (див. табл. 2.7).

Таблиця 2.7 – Опис таблиці БД «Паливо»

Назва поля	Тип даних	Властивість	Null	Опис
id	bigint(20)	Primary key (PK), Autoincrement	Ні	Ідентифікатор
type	varchar(255)	Unique key	Ні	Назва
created_at	timestamp		Ні	Дата створення
updated_at	timestamp		Ні	Дата оновлення

У таблиці «ports» зберігається перелік всіх портів (див. табл. 2.8).

Таблиця 2.8 – Опис таблиці БД «Порт»

Назва поля	Тип даних	Властивість	Null	Опис
id	bigint(20)	Primary key (PK), Autoincrement	Ні	Ідентифікатор
country_id	bigint(20)	Foreign key (FK)	Ні	Ідентифікатор країни
title	varchar(255)	Unique key	Ні	Назва
locode	varchar(5)		Ні	Код за міжнародною системою класифікацією географічних об'єктів
created_at	timestamp		Ні	Дата створення
updated_at	timestamp		Ні	Дата оновлення

У таблиці «user_roles» зберігаються ролі користувачів системи (див. табл. 2.9).

Таблиця 2.9 – Опис таблиці БД «Ролі користувачів»

Назва поля	Тип даних	Властивість	Null	Опис
id	bigint(20)	Primary key (PK), Autoincrement	Ні	Ідентифікатор
role	varchar(45)		Ні	Назва
created_at	timestamp		Ні	Дата створення
updated_at	timestamp		Ні	Дата оновлення

У таблиці «reports» зберігається перелік всіх репортів (звітів) капітана (див. табл. 2.10).

Таблиця 2.10 – Опис таблиці БД «Репорт»

Назва поля	Тип даних	Властивість	Null	Опис
id	bigint(20)	Primary key (PK), Autoincrement	Ні	Ідентифікатор
waypoint_id	bigint(20)	Foreign key (FK)	Ні	Ідентифікатор точки маршруту
type	enum('Departure', 'Noon', 'Arrival', 'Stoppage', 'Resumption')	Unique key	Ні	Тип
data	text		Так	Дані
reported_at	datetime		Ні	Дата репортування
created_at	timestamp		Ні	Дата створення
updated_at	timestamp		Ні	Дата оновлення

У таблиці «trip_reports» зберігаються звіти для маршруту (див. табл. 2.11).

Таблиця 2.11 – Опис таблиці БД «Звіти маршруту»

Назва поля	Тип даних	Властивість	Null	Опис
id	bigint(20)	Primary key (PK), Autoincrement	Ні	Ідентифікатор
uuid	char(36)	Unique key	Ні	Універсальний унікальний ідентифікатор
trip_id	bigint(20)	Foreign key (FK)	Ні	Ідентифікатор маршруту
date	date		Ні	Дата
type	enum('Weather Forecast', 'Mid Report', 'Final Report')		Ні	Тип
attachment	varchar(255)		Ні	Файл
notes	text		Так	Нотатки
created_at	timestamp		Ні	Дата створення
updated_at	timestamp		Ні	Дата оновлення

У таблиці «trips» зберігаються всі маршрути суден (див. табл. 2.12).

Таблиця 2.12 – Опис таблиці БД «Маршрут»

Назва поля	Тип даних	Властивість	Null	Опис
id	bigint(20)	Primary key (PK), Autoincrement	Ні	Ідентифікатор
uuid	char(36)		Ні	Універсальний унікальний ідентифікатор
user_id	bigint(20)	Foreign key (FK)	Ні	Ідентифікатор користувача
title	varchar(128)	Unique key	Ні	Назва
trip_code	varchar(32)		Ні	Код
status	enum('Planning', 'Active', 'Completed', 'Canceled', 'Deleted')		Ні	Статус
default_speed	float(8,2)		Так	Швидкість за замовчуванням
notes	text		Так	Нотатки
actual_arrival_date	datetime		Так	Фактична дата прибуття
planned_arrival_date	datetime		Так	Запланована дата прибуття
actual_departure_date	datetime		Так	Фактична дата відправлення
planned_departure_date	datetime		Так	Запланована дата відправлення
arrival_port_id	bigint(20)	Foreign key (FK)	Так	Порт прибуття
departure_port_id	bigint(20)	Foreign key (FK)	Так	Порт відправки
company_id	bigint(20)	Foreign key (FK)	Ні	Ідентифікатор компанії
ship_id	bigint(20)	Foreign key (FK)	Ні	Ідентифікатор судна
weather_source	varchar(45)		Так	Ресурс погоди
good_weather_definition	text		Так	Визначення хорошої погоди
is_edited	tinyint(1)		Ні	Редагується
edit_token	varchar(255)		Ні	Токен редагування
editor_id	bigint(20)	Foreign key (FK)	Ні	Редактор
created_at	timestamp		Ні	Дата створення
updated_at	timestamp		Ні	Дата оновлення

У таблиці «trip_edits» зберігаються дані для редагування маршруту (див. табл. 2.13).

Таблиця 2.13 – Опис таблиці БД «Редагування маршруту»

Назва поля	Тип даних	Властивість	Null	Опис
id	bigint(20)	Primary key (PK), Autoincrement	Ні	Ідентифікатор
trip_id	bigint(20)		Ні	Ідентифікатор маршруту
user_id	bigint(20)		Ні	Ідентифікатор користувача
started_at	datetime		Ні	Дата та час початку редагування
ended_at	datetime		Ні	Дата та час закінчення редагування
created_at	timestamp		Ні	Дата створення
updated_at	timestamp		Ні	Дата оновлення

У таблиці «trip_performances» зберігаються дані продуктивності, визначені контрактом (див. табл. 2.14).

Таблиця 2.14 – Опис таблиці БД «Дані продуктивності за контрактом»

Назва поля	Тип даних	Властивість	Null	Опис
id	bigint(20)	Primary key (PK), Autoincrement	Ні	Ідентифікатор
trip_id	bigint(20)	Foreign key (FK)	Ні	Ідентифікатор маршруту
fuel_type	bigint(20)	Foreign key (FK)	Ні	Ідентифікатор топлива
speed	double(8,2)		Ні	Швидкість
consumption	double(8,2)		Ні	Споживання
is_default	tinyint(1)		Ні	За замовчуванням
created_at	timestamp		Ні	Дата створення
updated_at	timestamp		Ні	Дата оновлення

У таблиці «ships» зберігаються судна (див. табл. 2.15).

Таблиця 2.15 – Опис таблиці БД «Судно»

Назва поля	Тип даних	Властивість	Null	Опис
id	bigint(20)	Primary key (PK), Autoincrement	Ні	Ідентифікатор
uuid	char(36)	Unique key	Ні	Універсальний унікальний ідентифікатор
title	varchar(255)	Unique key	Ні	Назва
company_operator	bigint(20)	Foreign key (FK)	Так	Ідентифікатор оператора компанії
company_id	bigint(20)	Foreign key (FK)	Ні	Ідентифікатор компанії
imo	varchar(11)	Unique key	Так	Унікальний ідентифікатор судна
status	enum('P','A')		Ні	Статус
master_user_id	bigint(20)	Foreign key (FK)	Так	Ідентифікатор капітана корабля
notes	text		Так	Нотатки
country_id	bigint(20)	Foreign key (FK)	Так	Ідентифікатор країни
type_generic	varchar(255)		Так	Тип загальний
type_detailed	varchar(255)		Так	Тип детальний
mmsi	varchar(40)	Unique key	Так	Ідентифікатор морської мобільної служби
call_sign	varchar(255)		Так	Позивний
gross_tonnage	int(11)		Так	Валовий тоннаж
draught	double(8,2)		Так	Осадка корпусу судна
fuel_consumption	varchar(255)		Так	Витрата палива
summer_dwt	int(11)		Так	Літній дедвейт
length	double(8,2)		Так	Довжина
breadth_extreme	double(8,2)		Так	Максимальна ширина над крайніми точками між лівим бортом і правим бортом судна
year_built	int(11)		Так	Рік побудови
home_port	bigint(20)		Так	Порт, з якого прибуло судно або з якого воно зареєстроване
speed_service	double(8,2)		Так	Швидкість обслуговування
liquid_oil	varchar(255)		Так	Масло
speed_max	double(8,2)		Так	Максимальна швидкість
builder	varchar(255)		Так	Будівельник
photo	varchar(255)		Так	Фото
classification_society_id	bigint(20)	Foreign key (FK)	Так	Ідентифікатор класифікаційного товариства судів
created_at	timestamp		Ні	Дата створення
updated_at	timestamp		Ні	Дата оновлення

У таблиці «weather_reports» зберігаються дані погоди для точок маршруту (див. табл. 2.16).

Таблиця 2.16 – Опис таблиці БД «Репорти погоди»

Назва поля	Тип даних	Властивість	Null	Опис
id	bigint(20)	Primary key (PK), Autoincrement	Ні	Ідентифікатор
source	varchar(45)		Ні	Джерело погоди
wind_speed	double		Так	Швидкість вітру
wind_direction	double		Так	Напрямок вітру
wave_height	double		Так	Висота хвилі
wave_direction	double		Так	Напрямок хвилі
current_speed	double		Так	Швидкість дрейфу
current_direction	double		Так	Напрямок дрейфу
air_temperature	double		Так	Температура повітря
pressure	double		Так	Тиск
swell_direction	double		Так	Напрямок зибу
swell_height	double		Так	Висота зибу
swell_period	double		Так	Період зибу
secondary_swell_direction	double		Так	Вторинний напрямок зибу
secondary_swell_height	double		Так	Вторинна висота зибу
secondary_swell_period	double		Так	Вторинний напрямок зибу
wind_wave_direction	double		Так	Напрямок вітрової хвилі
wind_wave_height	double		Так	Висота вітрової хвилі
wind_wave_period	double		Так	Період вітрової хвилі
waypoint_id	bigint(20)	Foreign key (FK)	Ні	Ідентифікатор шляхової точки
created_at	timestamp		Ні	Дата створення
updated_at	timestamp		Ні	Дата оновлення

У таблиці «users» зберігається перелік користувачів (див. табл. 2.17).

Таблиця 2.17 – Опис таблиці БД « Користувачі»

Назва поля	Тип даних	Властивість	Null	Опис
id	bigint(20)	Primary key (PK), Autoincrement	Ні	Ідентифікатор
name	varchar(255)		Ні	Повне ім'я
title	varchar(255)		Ні	Звання
role	bigint(20)		Ні	Ідентифікатор ролі
company_id	bigint(20)	Foreign key (FK)	Так	Ідентифікатор компанії
email	varchar(55)	Unique key	Ні	Електронна адреса
phone	varchar(55)		Так	Номер телефону
notes	text		Так	Нотатки
status	int(11)		Ні	Статус (1=Active, 2=Suspended, 3=Pending)
user_id	bigint(20)	Foreign key (FK)	Ні	Ідентифікатор користувача, що створив юзера
email_verified_at	timestamp		Так	Дата та час верифікації електронної адреси
password	varchar(255)		Ні	Пароль
remember_token	varchar(100)		Так	Токен
system_user	tinyint(1)		Ні	Чи є системним юзером
last_login_at	datetime		Ні	Дата та час останнього входу
photo	varchar(255)		Так	Фото
created_at	timestamp		Ні	Дата створення
updated_at	timestamp		Ні	Дата оновлення

У БД присутні також таблиці «migrations», «failed_jobs», «password_resets». Вони є стандартними у фреймворку Laravel, тому детального опису не потребують.

У таблиці «waypoints» зберігаються точки маршруту (див. табл. 2.18).

Таблиця 2.18 – Опис таблиці БД «Шляхові точки»

Назва поля	Тип даних	Властивість	Null	Опис
id	bigint(20)	Primary key (PK), Autoincrement	Ні	Ідентифікатор
trip_id	bigint(20)	Foreign key (FK)	Ні	Ідентифікатор маршруту
path_order	int(11)		Ні	Порядок
type	enum('Planned', 'Actual')		Ні	Тип
source	enum('Added', 'Reported', 'Calculated')		Ні	Ресурс
lat	double(12,9)		Ні	Широта
lng	double(12,9)		Ні	Довгота
map_lng	double(12,9)		Ні	Довгота на мапі
time	datetime		Ні	Дата та час для точки
notes	text		Так	Нотатки
created_at	timestamp		Ні	Дата створення
updated_at	timestamp		Ні	Дата оновлення
deleted_at	timestamp		Так	Дата видалення

3 РЕАЛІЗАЦІЯ

3.1 Опис технологій

3.1.1 Тема Metronic

Важливою частиною у веб-розробці є дизайн – те, що у першу чергу вливає на перше враження користувача.

Саме тому для реалізації користувальницької частини була обрана тема Metronic.

Переваги використання Metronic [3]:

- можливість налаштування абсолютно усього глобально, що дозволяє забезпечити необмежені унікальні стилізовані проекти;
- велика кількість компонентів для підтримки проектів з останніми тенденціями UI / UX;
- плагін для збору даних Datatable зі всіма розширеними функціями CRUD;
- якість коду.

3.1.2 Mapbox

Для відображення мап використовується MapBox Styles API [4].

API стилів Mapbox дозволяє читати та змінювати стилі мапи, шрифти та зображення. Цей API є основою для Mapbox Studio.

Styles API використовується в Studio, Mapbox GL JS або Mapbox Mobile SDK.

Щоб використовувати API стилі, потрібно знати специфікацію стилів Mapbox. Специфікація стилів Mapbox визначає структуру стилів мап і є

відкритим стандартом, який допомагає Studio спілкуватися з API та створювати мапи, сумісні з бібліотеками Mapbox.

3.1.3 JavaScript бібліотека Leaflet

Для створення маршруту судна на мапі використовується бібліотека JavaScript – Leaflet [5].

Leaflet – провідна бібліотека JavaScript з відкритим кодом для інтерактивних мап, зручних для мобільних пристроїв. Вона важить близько 39 КБ і має всі функції відображення, які потрібні більшості розробників.

Leaflet розроблена з урахуванням простоти, продуктивності та зручності використання. Вона ефективно працює на всіх основних настільних та мобільних платформах, може бути розширена за допомогою безлічі плагінів, має гарний, простий у використанні та добре задокументований API, простий, читабельний вхідний код.

3.1.4 Windy API

Для візуалізації та прогнозування погоди на мапі використовується сервіс Windy [6] та його Map Forecast API, моделі погоди, шари і ізолінії якого накладено поверх Leaflet.

Map Forecast API дозволяє налаштовувати візуалізацію мап Windy за допомогою власного вмісту та зображень для відображення на веб-сторінках і мобільних пристроях. Map Forecast API містить шари, частинки, легенду, засіб вибору та ізоліній, а також основні елементи керування та мапу. Дає можливість вибрати, що потрібно відображати, і змінити все, використовуючи стилі, елементи керування та події, а також різні бібліотеки. Огляд API наведено у додатку А.

3.1.5 Фреймворк Laravel

У якості backend-складової системи був обраний PHP-фреймворк Laravel.

У нього є виражений синтаксис, який слідує архітектурі Model-View-Controller (MVC) [7].

Загалом в Laravel є багато корисних функцій, що дозволяють зробити процес розробки веб-додатків швидким, простим, якісним, а саме:

- зрозуміла документація;
- гнучка система маршрутизації;
- зручна система міграцій, яка дозволяє спростити розгортання та оновлення додатка;
- вбудована підтримка движка шаблонів Blade;
- при створенні додатка можна використовувати Artisan – інтерфейс командного рядка для виводу вбудованих та власних команд;
- зручний механізм обробки помилок та виключень;
- вбудовані механізми аутентифікції та авторизації користувачів;
- у Laravel багато синтаксичного цукру. Синтаксис API фреймворку досить простий і зрозумілий. Тут немає довгих і складних конструкцій, а тільки короткі і продумані назви функцій.

3.1.6 Глобальний API погоди stormglass.io

Для отримання даних погоди для точки маршруту використовується глобальний API погоди stormglass.io [8].

API Storm Glass надає прогнози з високою роздільною здатністю на термін до 10 днів вперед. Морські дані, включаючи припливи, доступні для всіх океанів і морів світу. Детальний огляд API наведено у додатку Б.

3.2 Створення маршруту

Розглянемо реалізацію з боку користувача системи.

Для початку потрібно зазначити деталі маршруту, заповнивши хоча б обов'язкові поля форми (див. рис. 3.1).

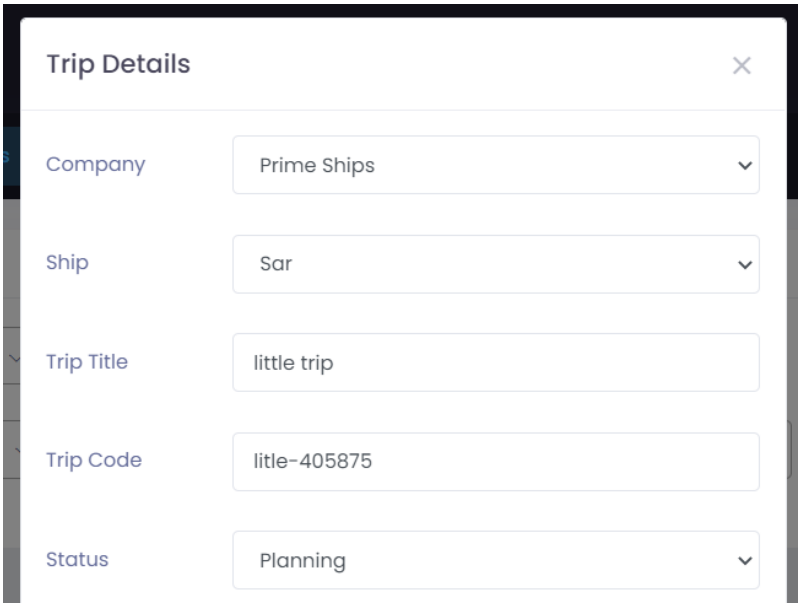


Рисунок 3.1 – Фрагмент форми деталей маршруту

Для створення маршруту треба перейти в режим редагування. Маршрут створюється вибором точок на мапі (див. рис. 3.2, 3.3).

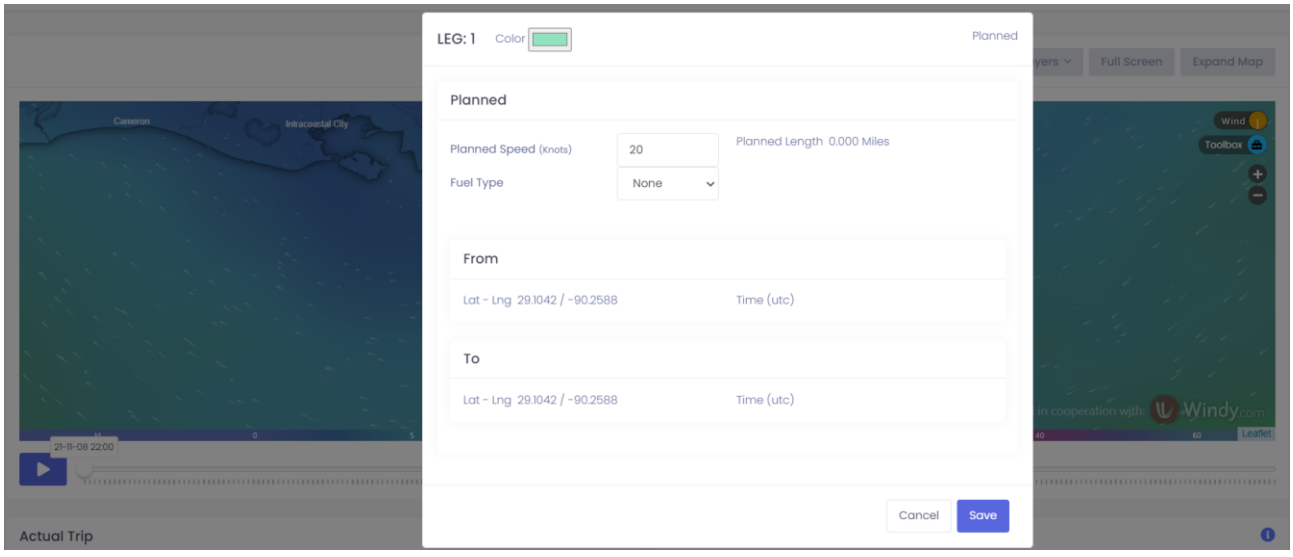


Рисунок 3.2 – Створення першої шляхової точки (місця відправлення)

Як можна побачити, при створенні першої шляхової точки пропонується обрати колір маршруту для відображення на мапі, заплановану швидкість та тип палива.

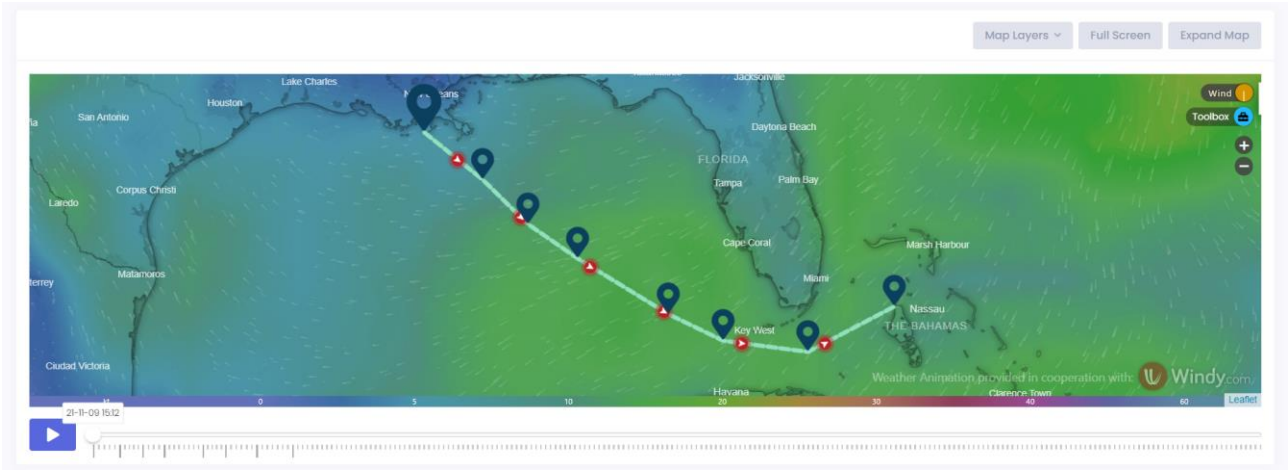


Рисунок 3.3 – Створення нових шляхових точок

При створенні подальших точок, вони просто відображаються на мапі вже зі стрілкою напряму руху.

Є можливість перегляду інформації щодо створеної шляхової точки (див. рис. 3.4). Для перегляду потрібно просто натиснути по маркеру точки.

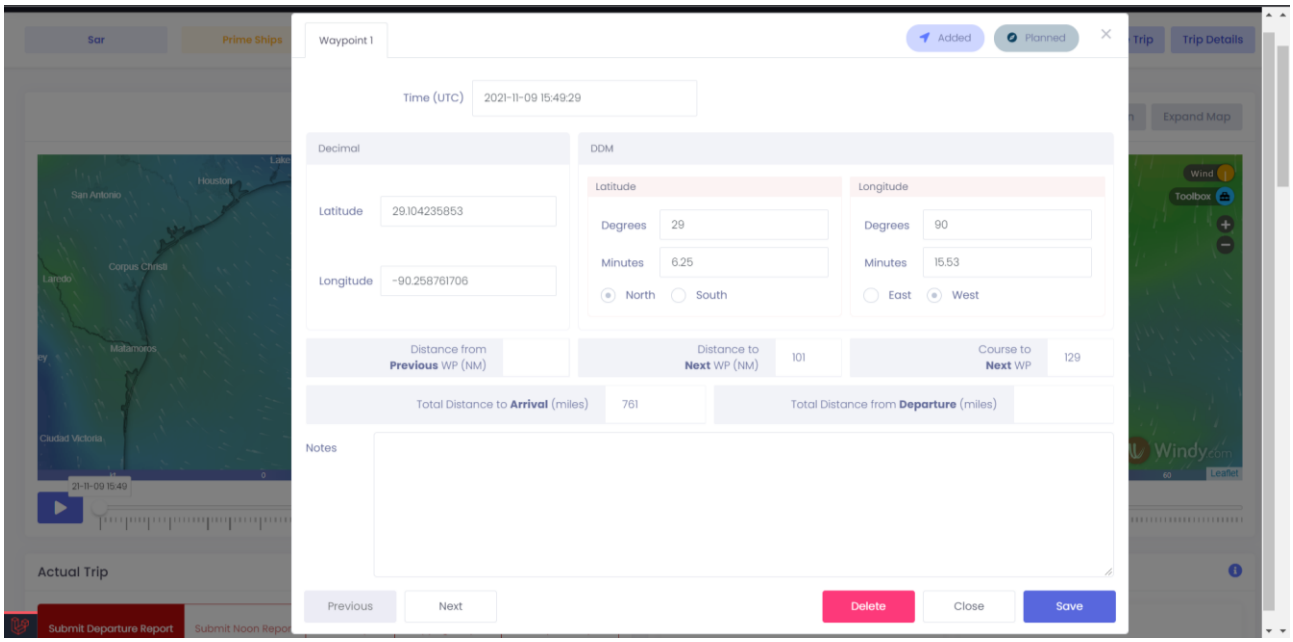


Рисунок 3.4 – Деталі шляхової точки

В деталях шляхової точки можна побачити час коли судно повинно дістатися цієї точки, координати точки у форматі градусів (DD) та у форматі градусів та десяткових хвилин (DDM), дистанцію до попередньої та наступної шляхової точки у морських милях, курс до наступної шляхової точки, а також відстань від місця відправки та до місця прибуття. Координати шляхової точки можна відредагувати та зберегти.

Увесь цей маршрут відображається у таблиці як одна частина з розрахованим запланованим часом прибуття, дистанцією усього маршруту у морських милях (див. рис. 3.5).

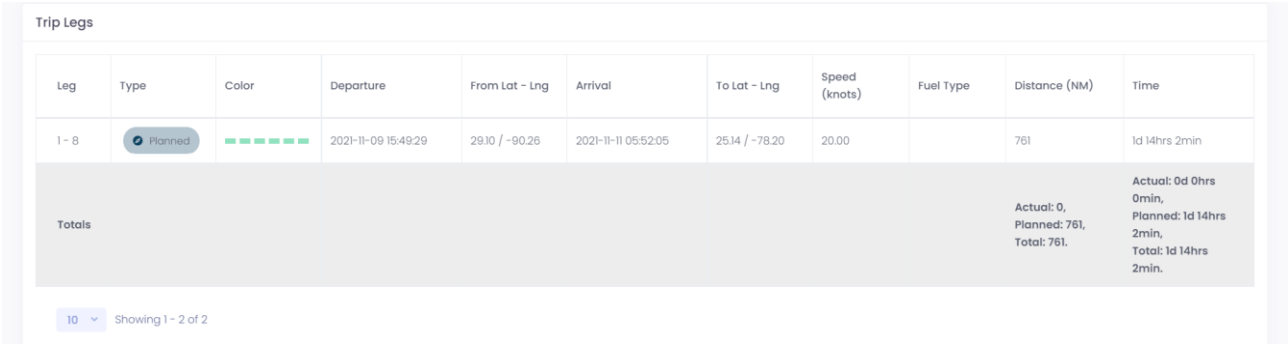


Рисунок 3.5 – Частина маршруту

На мапі можливо змінювати шари (погода чи звичайна мапа) за допомогою дропдауна «Map Layers» (див. рис. 3.6).

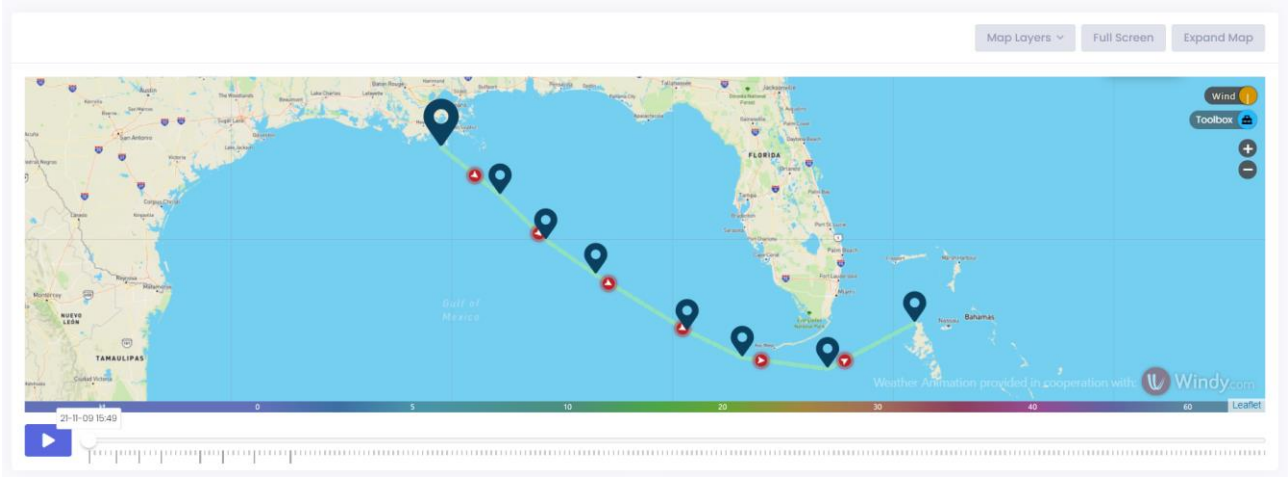


Рисунок 3.6 – Відображення звичайної мапи

Перейдемо детальніше до реалізації з боку розробника.

Для роботи мапи на сторінці підключені js-бібліотеки та js-плагіни (див. рис. 3.7).

```
<script src="https://unpkg.com/leaflet@1.4.0/dist/leaflet.js"></script>
<script src="https://cdnjs.cloudflare.com/ajax/libs/leaflet-polylinedecorator/1.1.0/leaflet.polylineDecorator.min.js"></script>
<script src="/js/vendor/leaflet.geometryutil.js"></script>
<script src="https://api.windy.com/assets/map-forecast/libBoot.js"></script>
<script src="https://cdnjs.cloudflare.com/ajax/libs/chartjs-plugin-annotation/0.5.7/chartjs-plugin-annotation.js"></script>
```

Рисунок 3.7 – Js-бібліотеки та js-плагіни для мапи

За відображення мапи на сторінці відповідає js-клас TripMap. У ньому відбувається отримання ключа Windy API та mapbox-token з відповідних тегів <meta>, ініціалізація мап та додавання шарів, оновлення стану часової шкали з наданими маршрутними точками, перемикання між шарами, а також додавання шляхових точок (див. рис. 3.8).

```
/**
 * Click handler for map
 * @param {object} event
 */
handleClick (event : Object ) {
  if (this.toolbox.tools.measure.active) {
    if (this.toolbox.tools.measure.start === null && event.originalEvent.shiftKey) {
      this.toolbox.tools.measure.line.setLatLngs([
        [event.latlng.lat, event.latlng.lng],
        [event.latlng.lat, event.latlng.lng]
      ]).addTo(this.map)
      this.toolbox.tools.measure.start = [event.latlng.lat, event.latlng.lng]
      this.toolbox.tools.measure.popup.setLatLng(this.toolbox.tools.measure.start)
        .setContent(`
        <p>
          <strong>Distance: </strong>0 (NM)<br/>
          <strong>Course: </strong>0
        </p>
      `)
      this.toolbox.tools.measure.popup.openOn(this.map)
    }
  } else {
    this.persistClickPosition(event.latlng)
  }
}
```

а) обробник кліку по мапі класу TripMap

```
/**
 * Notifies Trip controller about clicked position
 * @param {object} latlng
 */
persistClickPosition (latlng : Object ) {
  if (!this.data.adminView || !this.data.enabled) {
    return
  }

  if (latlng.lng >= -360 && latlng.lng <= 360) {
    window.actionsLogger.log('map-clicked')

    this.triggerEvent( eventName: 'clicked', latlng)
  }
}
```

б) метод класу TripMap для повідомлення контролеру поїздки про позицію, на якій було натиснуто

```
clicked: ({ detail }) => {
  if (this.legInProggress) {
    const newWaypoint = this.placeAddedWaypoint(detail)

    this.extendLeg(newWaypoint)
    this.saved = false
    _.last(this.legs).calculate()
    this.waypointsTable.reload( baseOnly: true)
    this.legsTable.reload( baseOnly: true)
  } else if (this.waypoints.length === 0) {
    const newWaypoint = this.placeAddedWaypoint(detail)

    this.startLeg(newWaypoint)
    this.saved = false
    _.last(this.legs).calculate()
    this.legEditor.open(
      _.last(this.legs),
      order: this.legs.length - 1,
      isNewLeg: true
    )
    this.waypointsTable.reload( baseOnly: true)
    this.legsTable.reload( baseOnly: true)
  } else {
    this.newLegDialog.open(detail)
  }
},
```

в) обробка кліку в класі Trip

Рисунок 3.8 – Фрагменти коду для створення шляхової точки

Таким чином, обробник кліку по мапі класу TripMap відповідає за отримання координат для шляхової точки, додавання шляхової точки до маршруту та створення лінії між точками за допомогою методу setLatLngs бібліотеки Leaflet (див. рис. 3.9, а). Також обробник кліку визиває метод для збереження позиції у контролері поїздки. У свою чергу, він запускає трігер кліку (див. рис. 3.9, б). Вже у класі Trip відбувається збереження шляхової точки, оновлення таблиць, відображення модального вікна у разі додавання першої шляхової точки для задання запланованої швидкості, вибору кольору маршруту для відображення на мапі та палива, виклик перерахунку даних для маршруту (часу, дистанції).

У js-класі TripMap також реалізовано метод для розрахунку градусів у форматі DDM (див. рис. 3.9).

```
calculateDmsValues (latlng) {
  let lng = latlng.lng

  if (latlng.lng > 180 || latlng.lng < -180) {
    lng = this.validate(latlng.lng,  range: 180)
  }

  const latScalar = Math.abs(latlng.lat)
  const lngScalar = Math.abs(lng)
  const latDirection = latlng.lat < 0 ? 'S' : 'N'
  const lngDirection = lng < 0 ? 'W' : 'E'
  const latDegrees = Math.floor(latScalar)
  const lngDegrees = Math.floor(lngScalar)
  const latDegDec = (latScalar % 1) * 60
  const lngDegDec = (lngScalar % 1) * 60

  return {
    lat: `${latDegrees.toString().padStart(2, '0')} ${latDegDec.toFixed( fractionDigits: 3)} ${latDirection}`,
    lng: `${lngDegrees.toString().padStart(3, '0')} ${lngDegDec.toFixed( fractionDigits: 3)} ${lngDirection}`
  }
}
```

Рисунок 3.9 – Фрагмент коду для розрахунку градусів у форматі DDM

Як можна побачити, ціле число – це градуси, а щоб отримати хвилини, потрібно просто помножити десяткову частину на 60. Щоб визначити напрям, потрібно просто порівняти значення з нулем. Якщо широта менша за нуль, тоді

напрямок північний, в інакшому випадку – південний. Якщо довгота менша за нуль, то напрям східний, в інакшому випадку – західний.

Для розрахунку відстані для всього маршруту (див. рис. 3.10) або між шляховими точками (див. рис. 3.11) використовується відповідно методи length та distance бібліотеки Turf.js.

```
if (this.waypoints.length > 1) {
  this.data.trip.full_distance = turf.length(
    turf.lineString(
      this.waypoints.map(waypoint => [waypoint.data.lng, waypoint.data.lat])
    )
  )
} else {
  this.data.trip.full_distance = 0
}
```

Рисунок 3.10 – Фрагмент коду для розрахунку відстані для всього маршруту

```
/**
 * Calculates distance between given coordinates
 * @param {array} latlng
 * @param {array} nextLatlng
 * @returns {string}
 */
getDistance (latlng :any[] , nextLatlng :any[] ) {
  return (turf.distance(
    turf.point(latlng),
    turf.point(nextLatlng)
  ) / 1.852).toFixed( fractionDigits: 0)
}
```

Рисунок 3.11 – Фрагмент коду для розрахунку відстані між двома координатами

Треба лише передати набір координат шляху чи набір двох координат, для яких потрібно розрахувати відстань. Так як відстань відображається в системі в морських милях, отриманий результат в кілометрах треба лише поділити на 1.852.

Для розрахунку курсу до наступної точки використовується метод bearing бібліотеки Turf.js (див. рис. 3.12).

```
/**
 * Calculates course between given coordinates
 * @param {array} latlng
 * @param {array} nextLatlng
 * @returns {string}
 */
getCourse (latlng :any[] , nextLatlng :any[] ) {
  let bearing = turf.bearing(
    turf.point(latlng),
    turf.point(nextLatlng)
  )

  bearing = bearing < 0 ? bearing + 360 : bearing

  return bearing.toFixed( fractionDigits: 0)
}
```

Рисунок 3.12 – Фрагмент коду для розрахунку курсу до наступної шляхової точки

Таким чином, за допомогою цих формул та методів розраховується дистанція до попередньої та наступної шляхової точки у морських милях, курс до наступної шляхової точки, а також відстань від місця відправки та до місця прибуття.

3.3 Отримання погодних даних

У системі є функціонал для отримання погодних даних як для запланованого маршруту, так і для актуального (фактичного). Як приклад, розглянемо отримання погодних даних для запланованого маршруту.

Для цього в режимі редагування маршруту треба перейти до блоку «Planned Trip» (див. рис. 3.13).

В блоці «Weather Data» потрібно заповнити інтервал отримання погодних даних та обрати дату і час початку. Далі потрібно натиснути на кнопку «Get Weather Data».

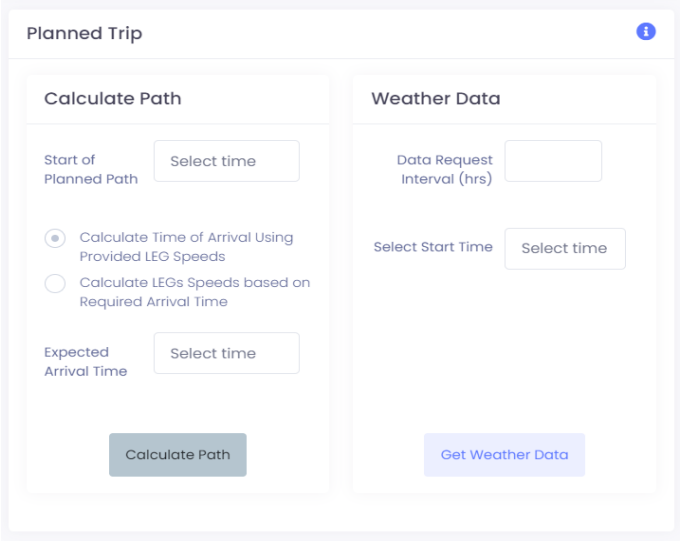


Рисунок 3.13 – Блок отримання даних для запланованого маршруту

Після цих дій, погодні дані з’являться у таблиці «Trip Waypoints» (див. рис. 3.14).

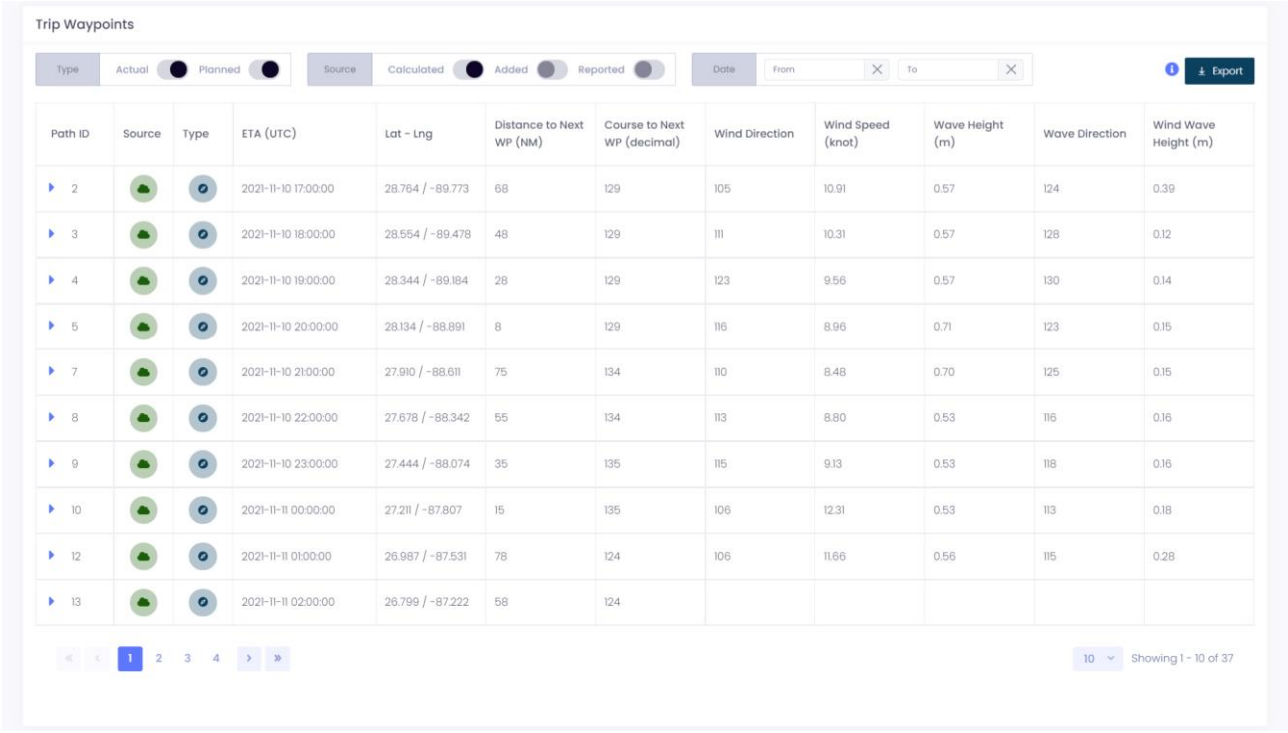


Рисунок 3.14 – Таблиця з розрахованими шляховими точками з погодними даними

У той же час на мапі з’являться розраховані шляхові точки, відображені червоними точками (див. рис. 3.15).

Рисунок 3.15 – Відображення розрахованих шляхових точок на мапі

Ці всі точки є клікабельними, тобто натиснувши на них з’явиться модальне вікно як для звичайної точки (див. рис. 3.16).

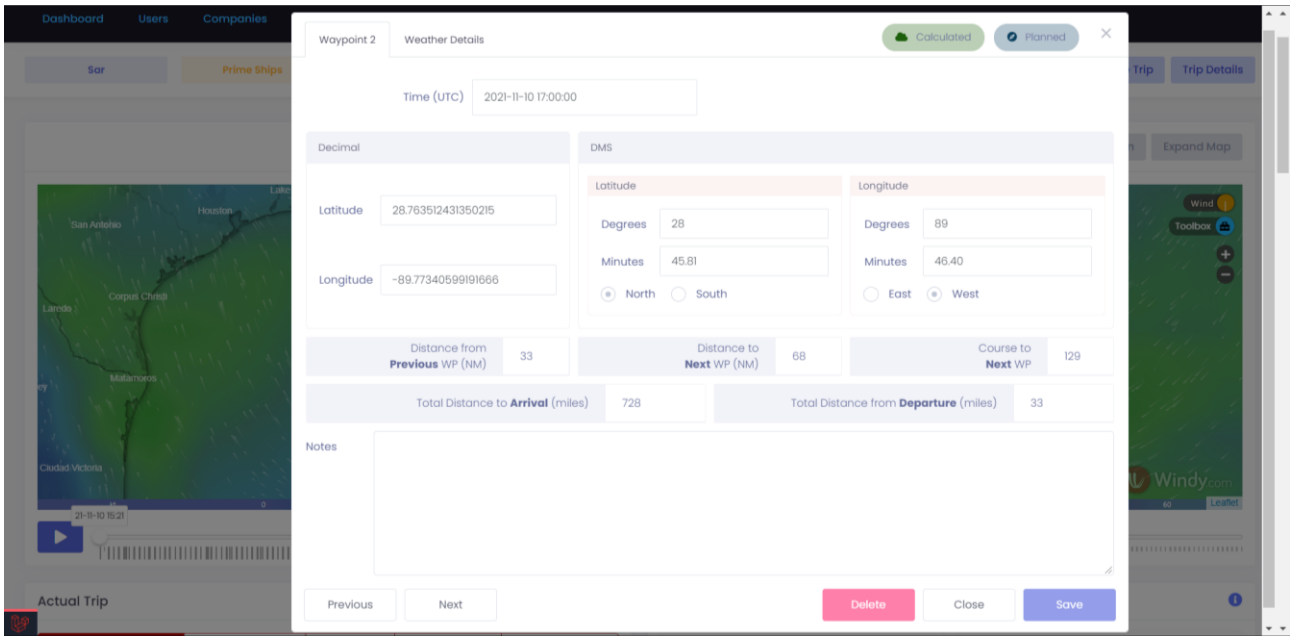


Рисунок 3.16 – Деталі першої розрахованої шляхової точки

Також ці точки вже мають вкладку с деталями погоди, в якій знаходяться отримані погодні дані з різних ресурсів (див. рис. 3.17).

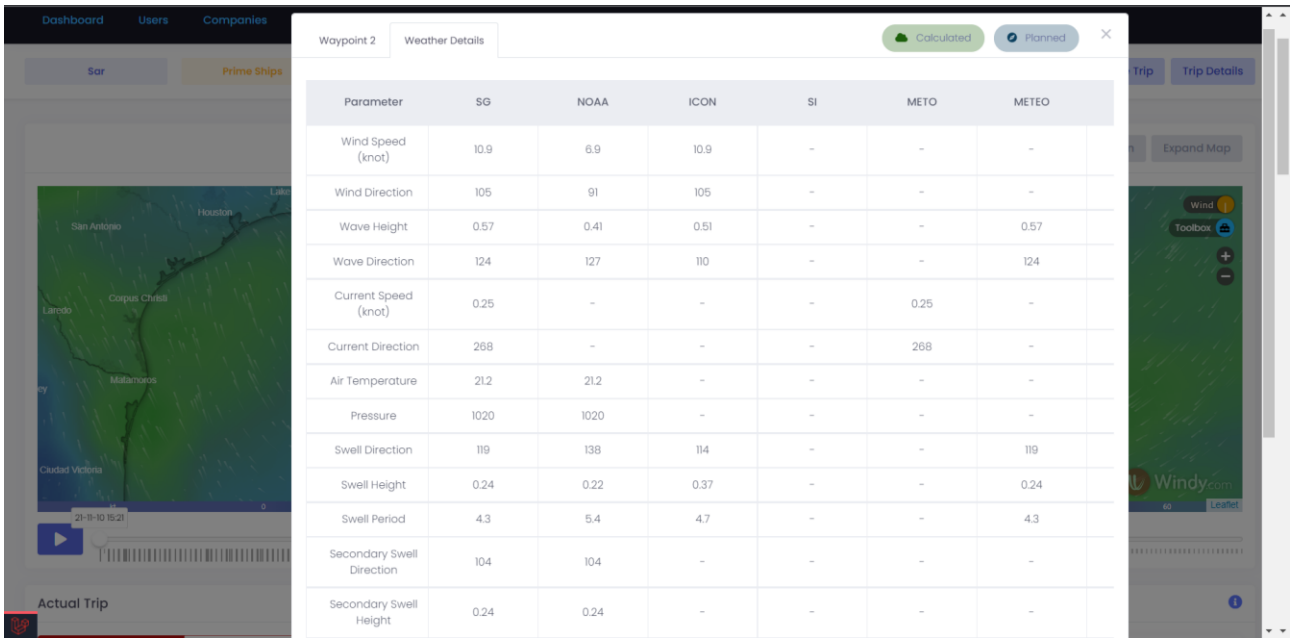


Рисунок 3.17 – Деталі погоди першої розрахованої шляхової точки

Розглянемо реалізацію з боку розробника (див. рис. 3.18).

Перш за все відбувається визначення з точками початку та кінця маршруту. Якщо вже є збережені репорти капітану корабля, то першою точкою вважатиметься координата останнього репорту, так як до цієї точки частина маршруту є вже пройденою та вважається актуальною. Далі відбувається розрахунок позиції судна за маршрутом з заданим інтервалом часу, тобто розрахунок нових шляхових точок.

```
/** Notifies trip controller about weather request **/
persistWeatherRequest () {
  window.actionsLogger.log('request-planned-weather', {
    interval: this.refs.weatherInterval.value,
    start: this.refs.$weatherStart.val()
  })

  if (this.refs.weatherInterval.value <= 0) {
    return
  }

  this.triggerEvent( eventName: 'weather',  detail: { interval: this.refs.weatherInterval.value })
}
```

а) обробка кліку по кнопці «Get Weather Data»

```
weather: ({ detail }) => {
  if (this.waypoints.length === 0) {
    return
  }

  let firstPoint = _.first(this.waypoints)
  const lastPoint = _.last(this.waypoints)

  if (this.lastReportedWaypoint !== null) {
    firstPoint = this.lastReportedWaypoint
  }

  this.wipeWeatherPoints(
    firstPoint.data.path_order,
    lastPoint.data.path_order
  )

  this.calculatePath()

  this.calculateWeatherPoints(
    detail.interval,
    firstPoint.data.path_order,
    lastPoint.data.path_order
  )
}
```

б) обробка кліку в класі Trip

Рисунок 3.18 – Фрагмент js-коду для отримання погодних даних

Наступним кроком є відправка запиту з розрахованими точками в метод контролеру TripController (див. рис. 3.19).

```
requestWeatherChunk () {
  const weatherPointsChunk = this.weatherPoints.splice( start: 0, this.weatherPointsPerCall)

  axios.post( url: '/weather', weatherPointsChunk)
    .then(({ data }) => {
      this.weatherPointsData = this.weatherPointsData.concat(data.data)

      if (this.weatherPoints.length) {
        this.requestWeatherChunk()
      } else {
        this.processWeatherData()
      }
    })
    .catch(error => {
      console.log(error)
      KTApp.unblock( target: '#trip')
    })
}
```

Рисунок 3.19 – Фрагмент коду для відправки запиту на отримання погодних даних

Контролер перебирає всі маршрутні точки та доповнює їх даними погоди (див. рис. 3.20). Сам же запит на отримання даних з API погоди stormglass.io реалізовано в класі StormGlass, код якого наведено у додатку В.

```php
/**
 * Call Storm Glass API for weather data for provided points
 *
 * @param Request $request
 * @return JsonResponse
 */
public function getWeather(Request $request)
{
    $stormGlass = new StormGlass(env( key: 'STORM_GLASS_KEY'));
    $points = $request->all();

    foreach ($points as $i => $point) {
        $weather = $stormGlass->getWeather(['lat' => $point['lat'], 'lng' => $point['lng']], $point['time']);
        $points[$i]['weather_reports'] = $weather;
    }

    return response()->json(['ok' => true, 'data' => $points]);
}
```

Рисунок 3.20 – Код методу контролера для отримання погодних даних

4 ОГЛЯД ОСНОВНИХ МОЖЛИВОСТЕЙ

4.1 Управління суднами

Функціонал управління суднами згідно зазначених функціональних вимог до системи доступний лише адміністратору системи та адміністратору компанії. На сторінці суден усі судна системи представлені у таблиці (див. рис. 4.1).

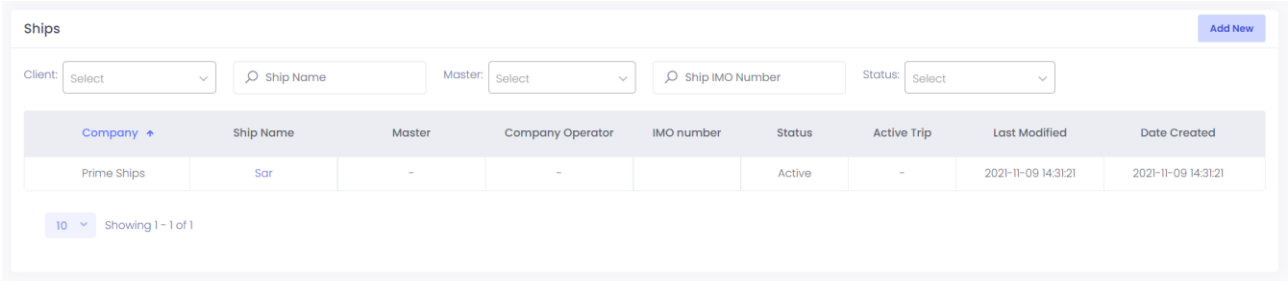


Рисунок 4.1 – Список суден

Щоб створити судно, потрібно натиснути кнопку «Add New» та заповнити форму (див. рис. 4.2).

Щоб перейти на сторінку з повною інформацією про судно, потрібно натиснути на назву судна у таблиці (див. рис. 4.3).

Щоб відредагувати інформацію про судно, потрібно натиснути кнопку «Edit Details», після чого вся інформація у блоці «Vessel Details» буде доступна для редагування.

Рисунок 4.2 – Форма створення судна

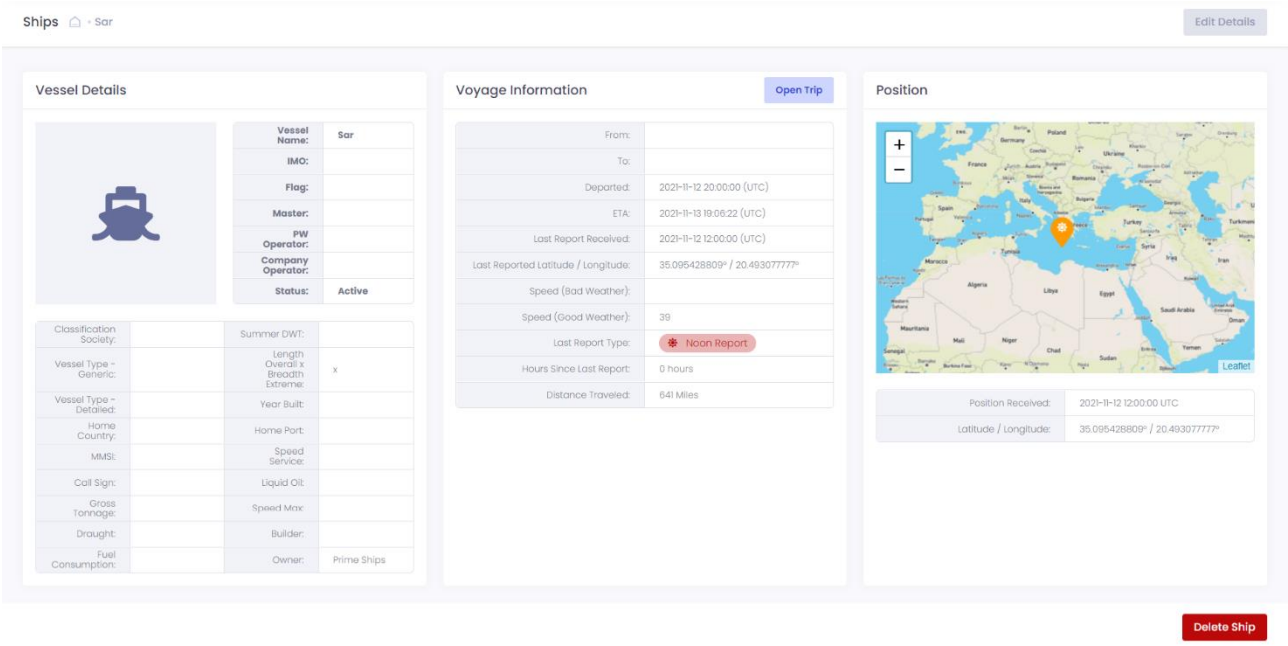


Рисунок 4.3 – Сторінка з повною інформацією про судно

4.2 Управління маршрутами

Функціонал управління маршрутами згідно зазначених функціональних вимог до системи доступний всім ролям користувачів у системі. На сторінці маршрутів усі маршрути суден представлені у таблиці (див. рис. 4.4).

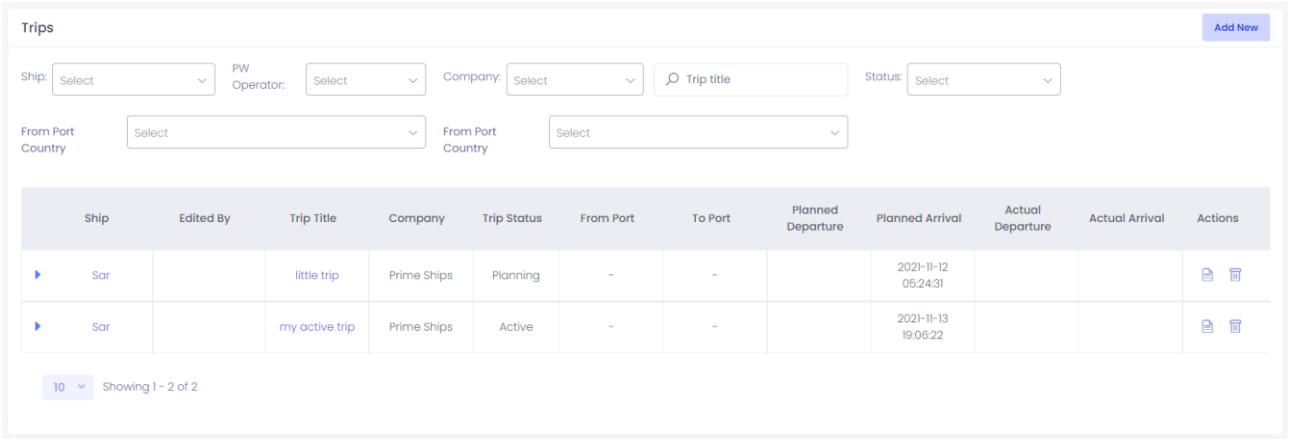


Рисунок 4.4 – Список маршрутів суден

Щоб створити маршрут, потрібно натиснути кнопку «Add New» та заповнити форму з деталями маршруту (див. рис. 4.5).

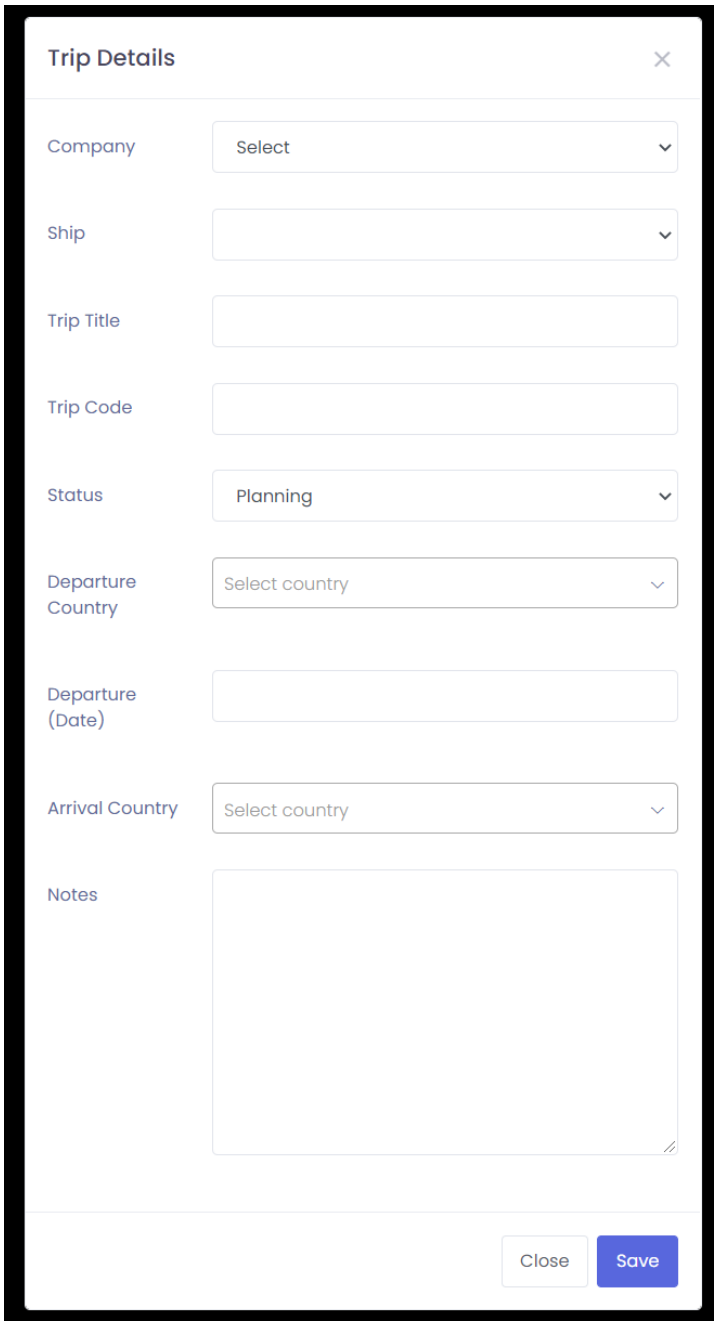


Рисунок 4.5 – Форма створення маршруту судна

Для редагування деталей чи видалення маршруту потрібно натиснути відповідну кнопку у колонці «Actions» таблиці.

Щоб перейти безпосередньо для редагування маршруту, потрібно натиснути на назві маршруту (див. рис. 4.6).

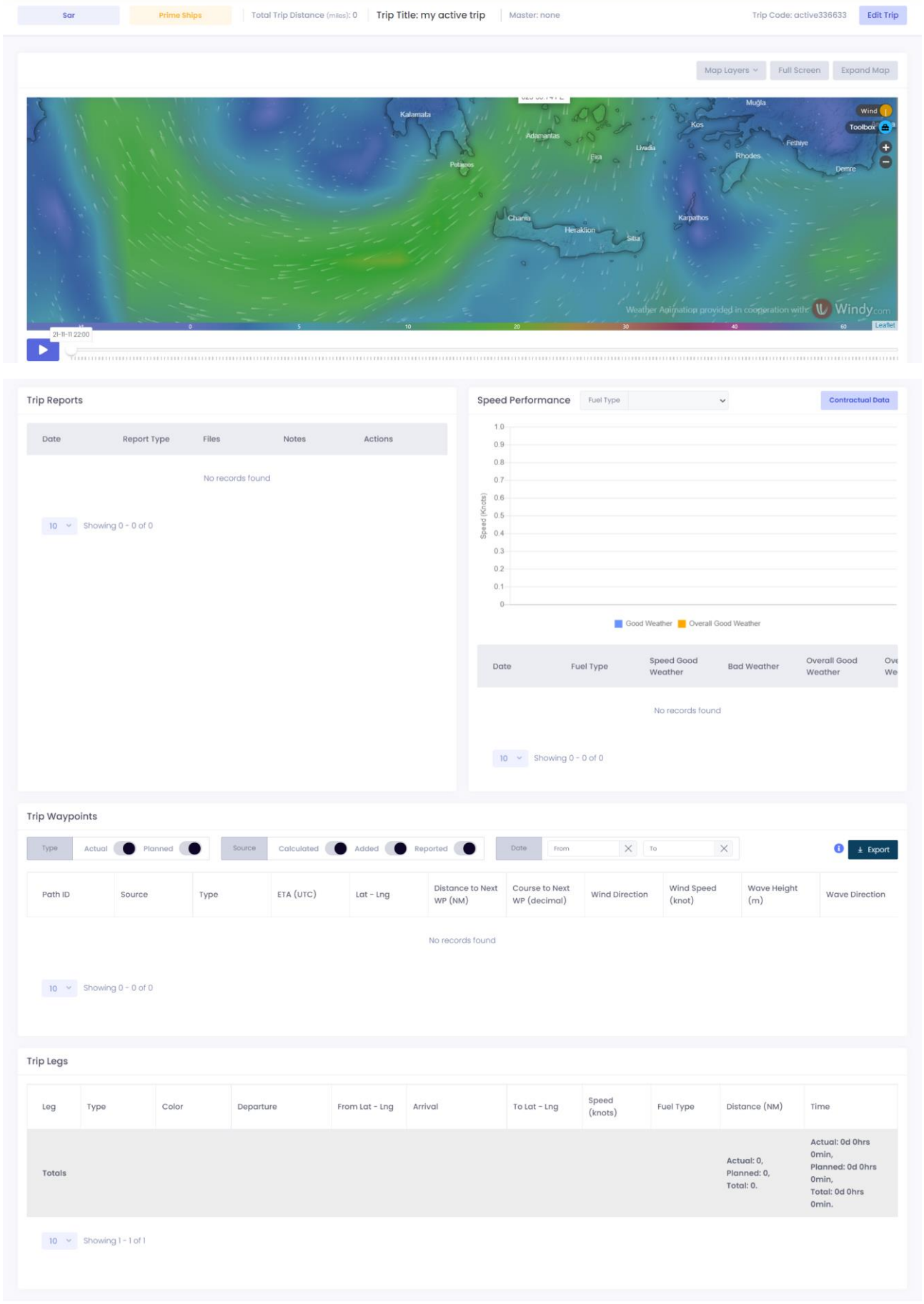
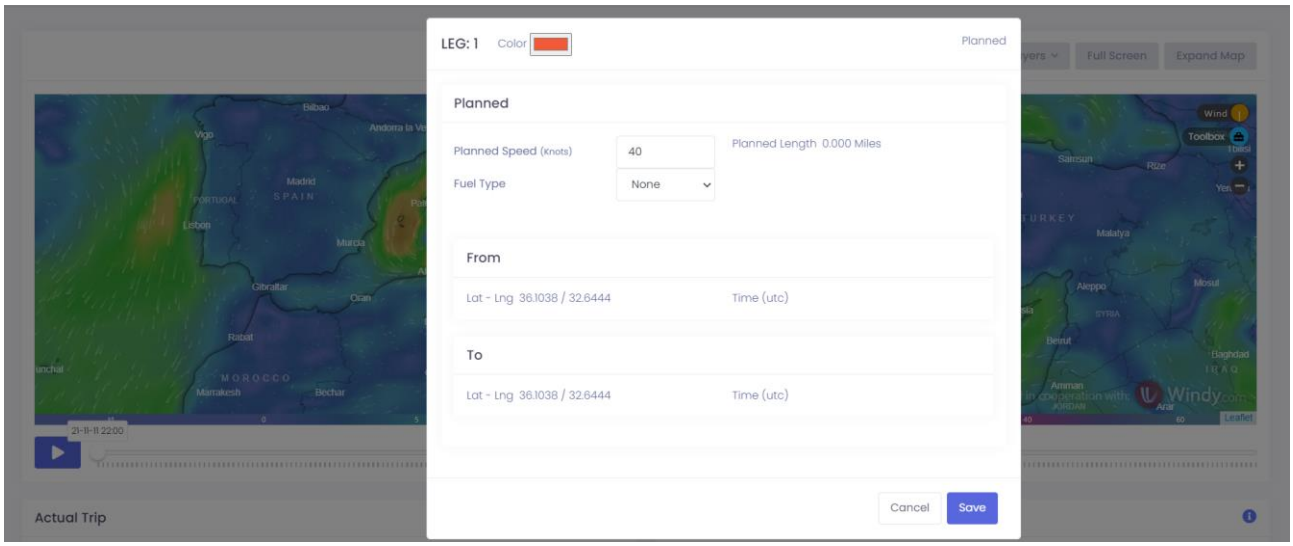
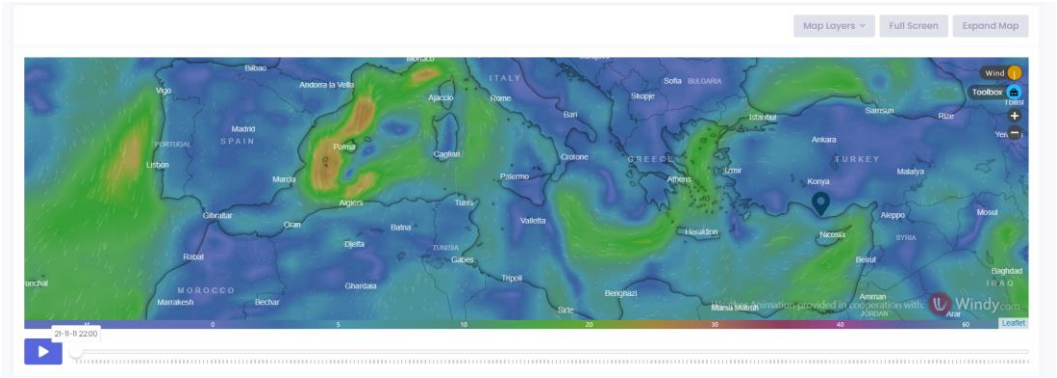


Рисунок 4.6 – Сторінка редагування маршруту

Для створення маршруту треба перейти в режим редагування. Маршрут створюється вибором точок на мапі (див. рис. 4.7, 4.8).



а) модальне вікно вибору кольору маршруту для відображення на мапі, швидкості та типу палива



б) створена точка на мапі

Рисунок 4.7 – Створення першої маршрутної точки

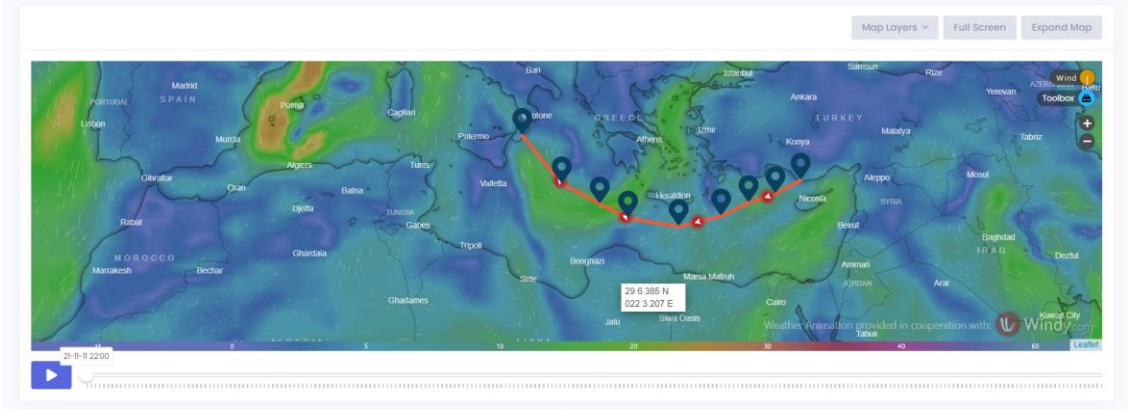


Рисунок 4.8 – Створення нових шляхових точок

При створенні подальших точок, вони просто відображаються на мапі вже зі стрілкою напряму руху.

Для мапи можливо обрати різні опції відображення: погодний шар, який продемонстровано на попередніх рисунках, та без шару (відображення звичайної мапи без шарів) (див. рис. 4.9).

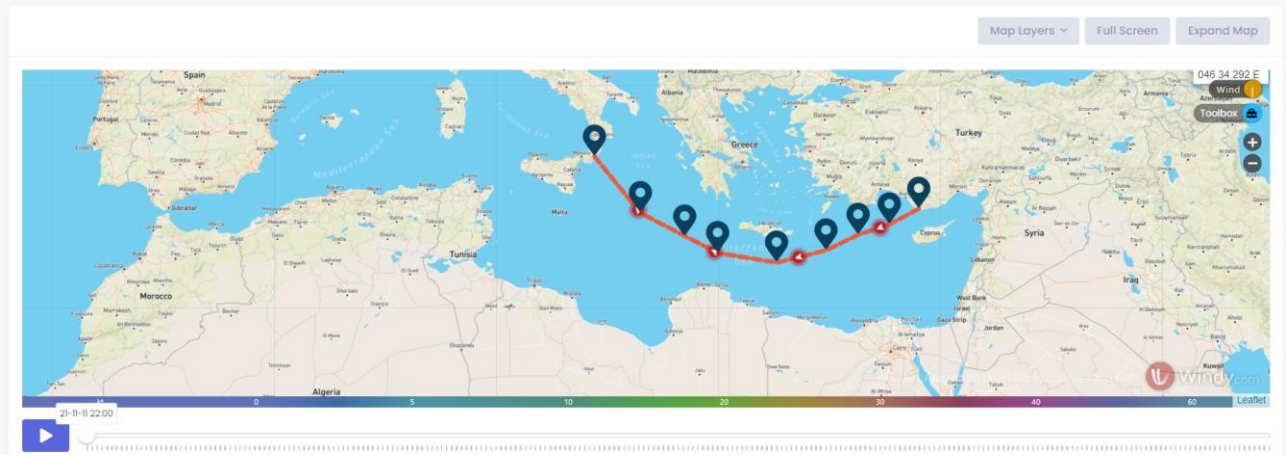


Рисунок 4.9 – Відображення звичайної мапи

У свою чергу на мапі можливо вибрати різні опції погодного шару для відображення (див. рис. 4.10 – 4.16). Для цього потрібно натиснути кнопку «Wind» на мапі та обрати потрібні дані для відображення.

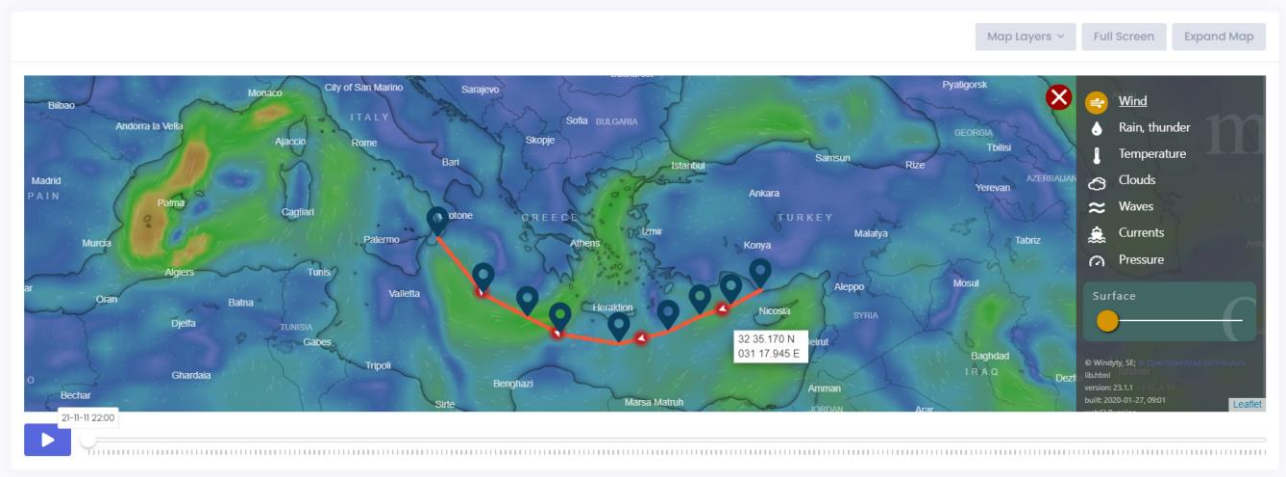


Рисунок 4.10 – Відображення вітру на мапі

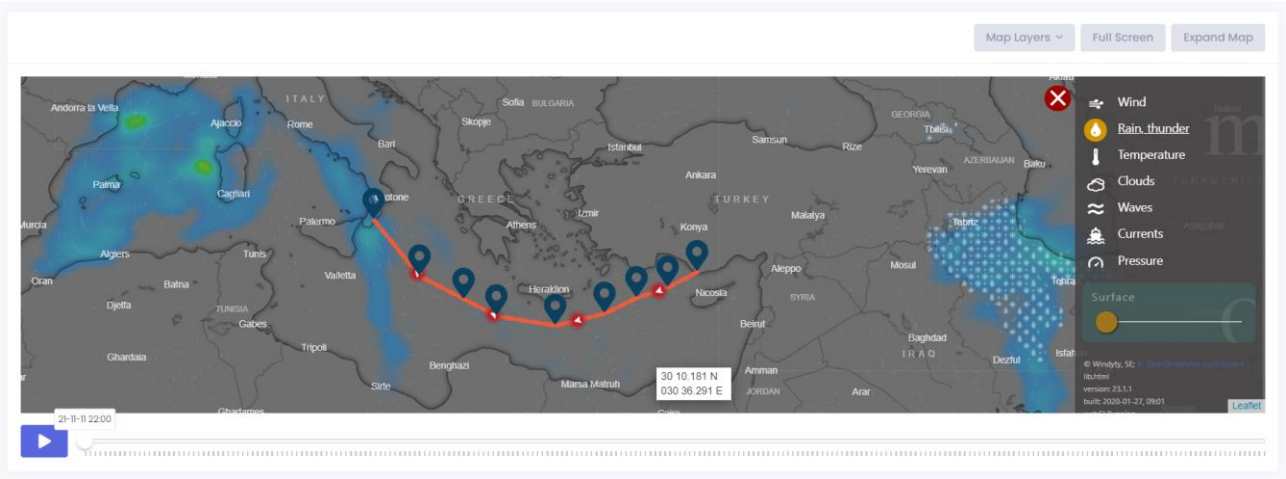


Рисунок 4.11 – Відображення опадів на мапі

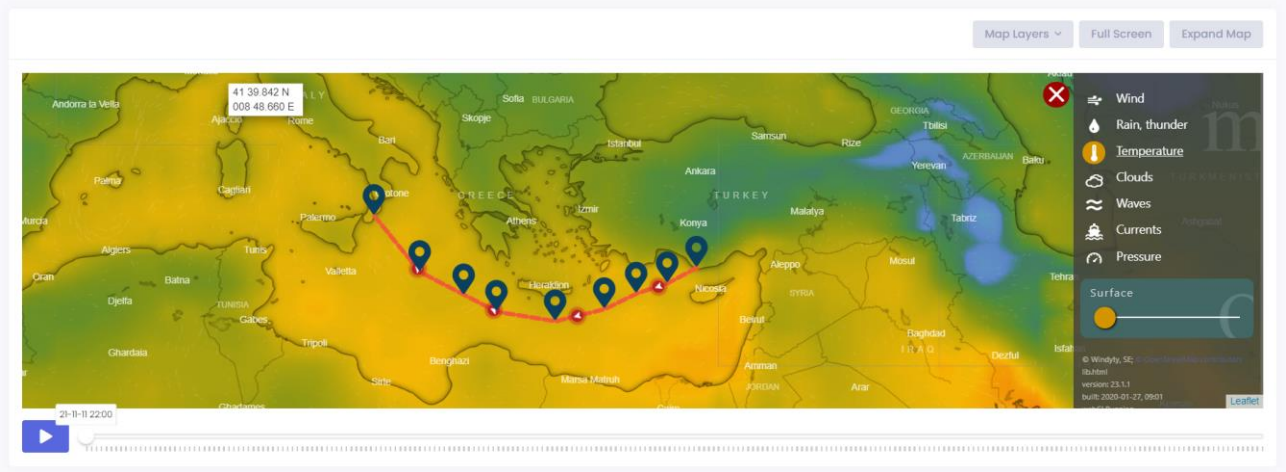


Рисунок 4.12 – Відображення температури на мапі

Рисунок 4.13 – Відображення хмар на мапі

Рисунок 4.14 – Відображення хвиль на мапі

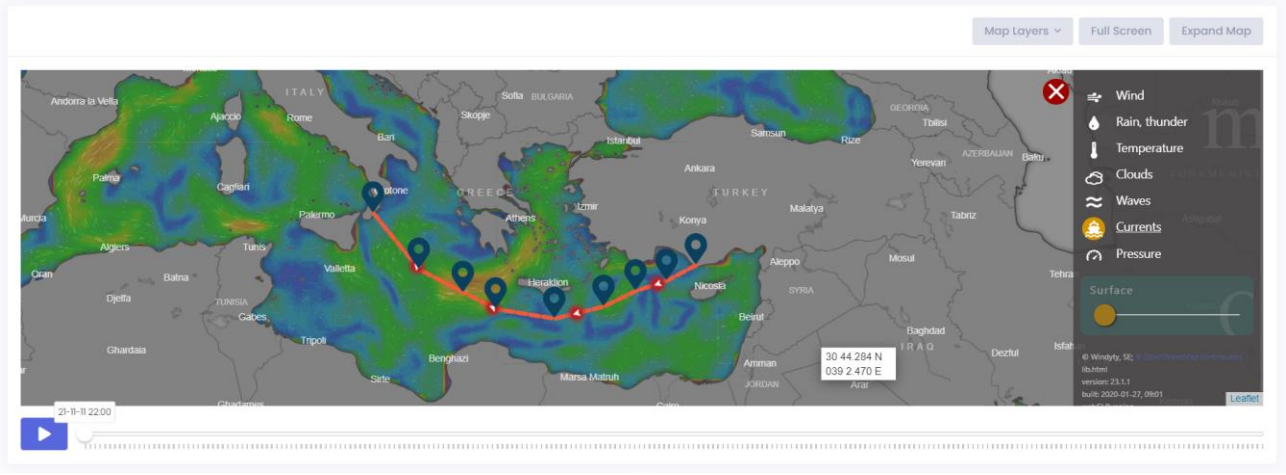


Рисунок 4.15 – Відображення течій на мапі

Рисунок 4.16 – Відображення тиску на мапі

Знизу мапи є повзунок з часовою шкалою, натиснувши на який можливо переглянути зміну погодних даних на мапі під час поїздки. Там буде змінюватись позиція судна, а якщо вийти з режиму редагування маршруту, то буде відображатись маркер з значеннями відповідно обраному шару погоди (див. рис. 4.17).

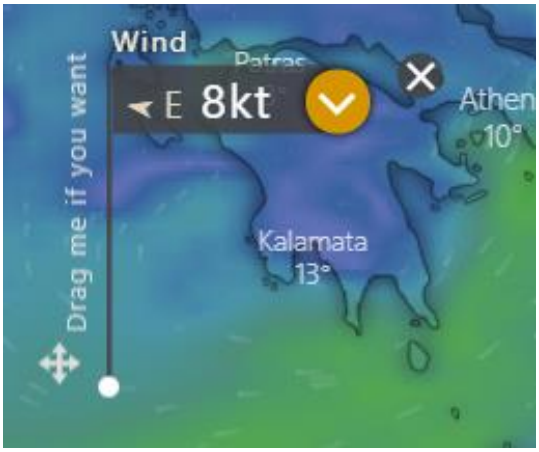


Рисунок 4.17 – Маркер з відповідними даними щодо вітру у точці

Для відстеження актуального маршруту судна та позиції є можливість збереження репортів капітана корабля п’ятьох видів (див. рис. 4.18).

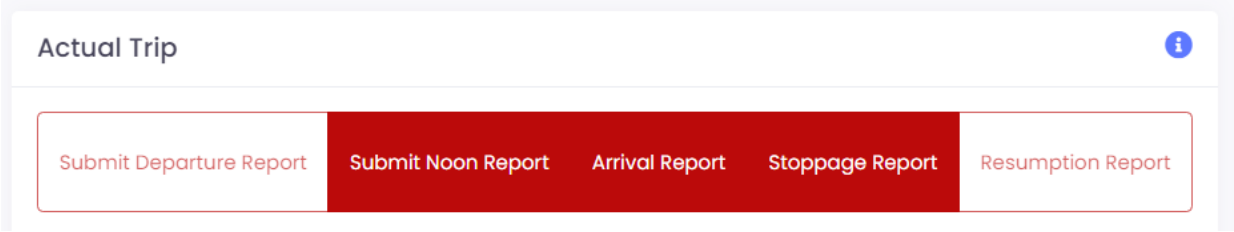


Рисунок 4.18 – Кнопки для збереження репортів капітана корабля

При створенні кожного репорту необхідно вказати час та позицію судна (див. рис. 4.19).

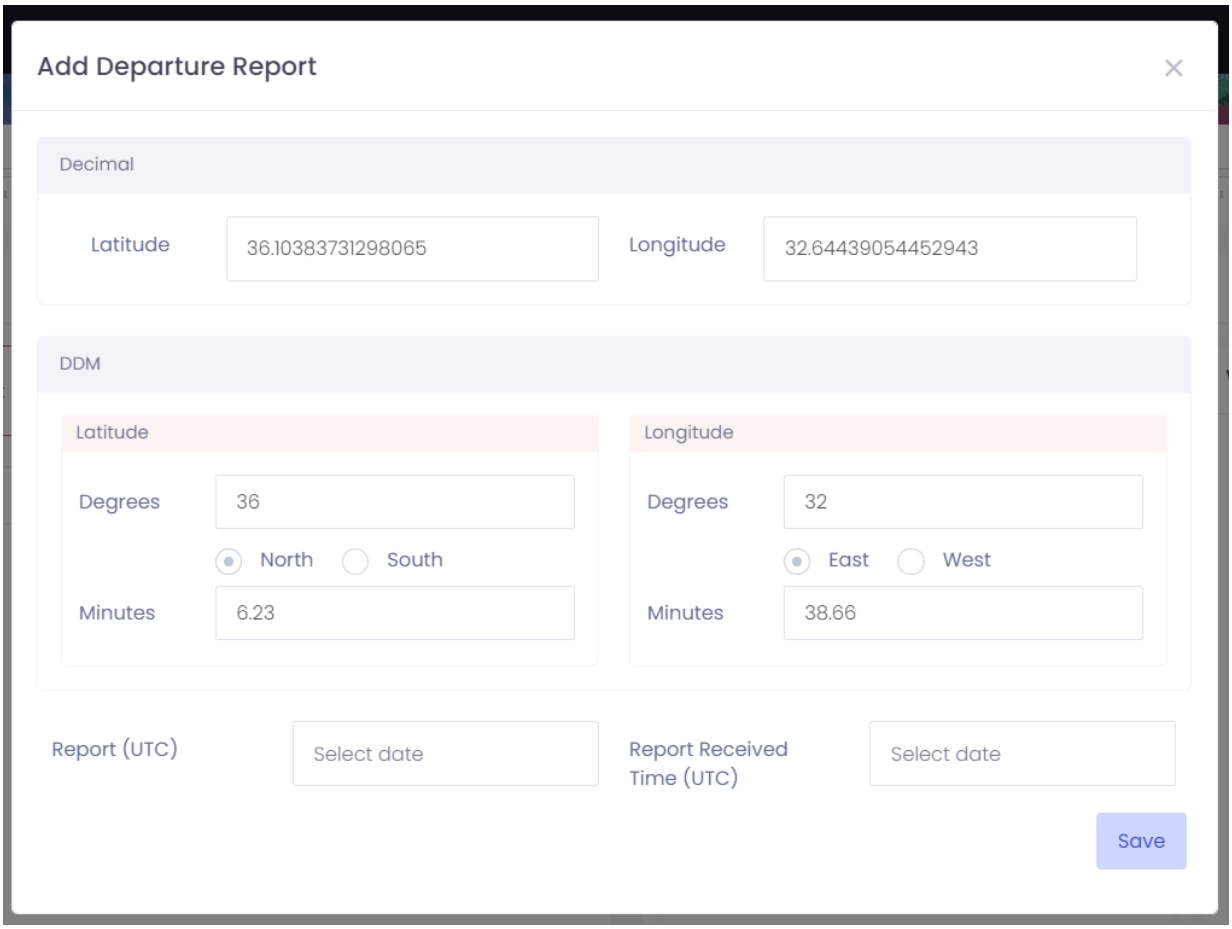


Рисунок 4.19 – Форма репорту про відправку

Після збереження кожного репорту на мапі будуть з’являтись відповідні позначки, дані будуть ураховуватися при розрахунках для решти маршруту (див. рис. 4.20, 4.21).

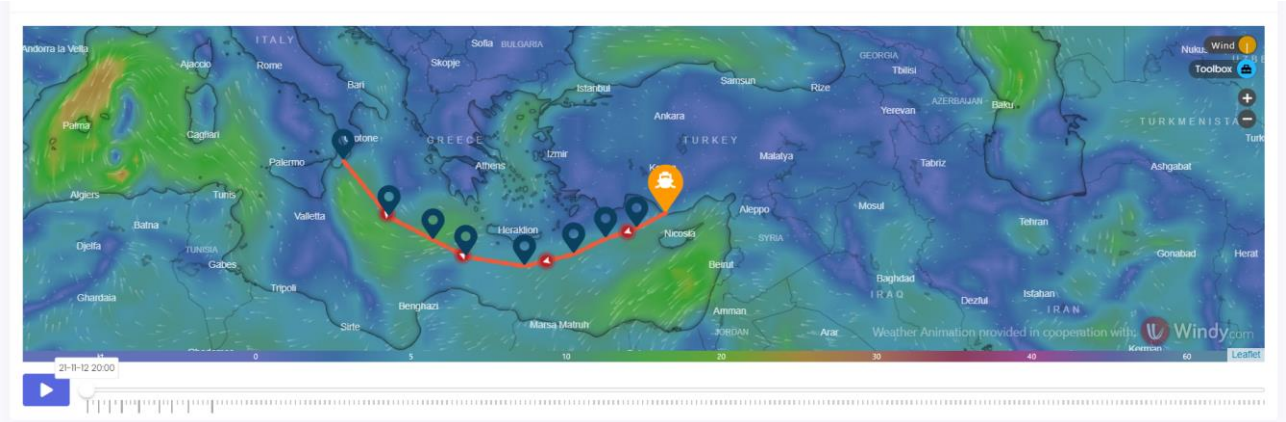


Рисунок 4.20 – Позначка репорту капітана про відправку на мапі

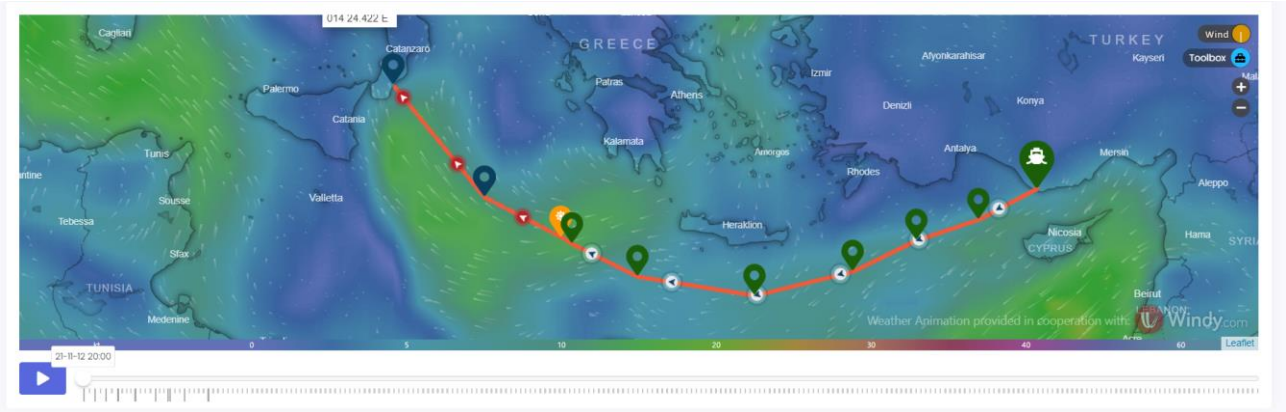
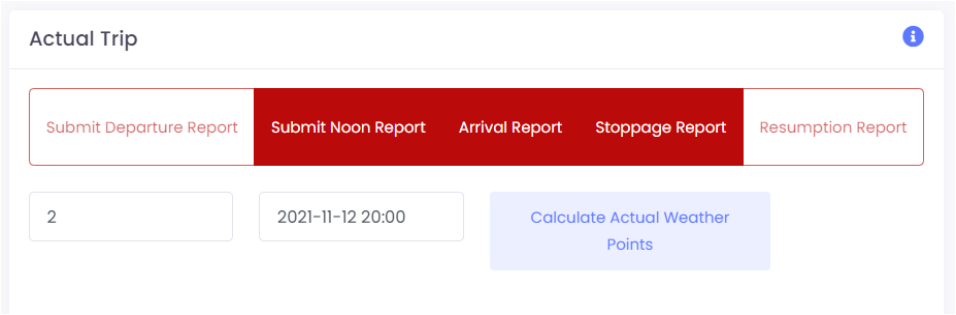


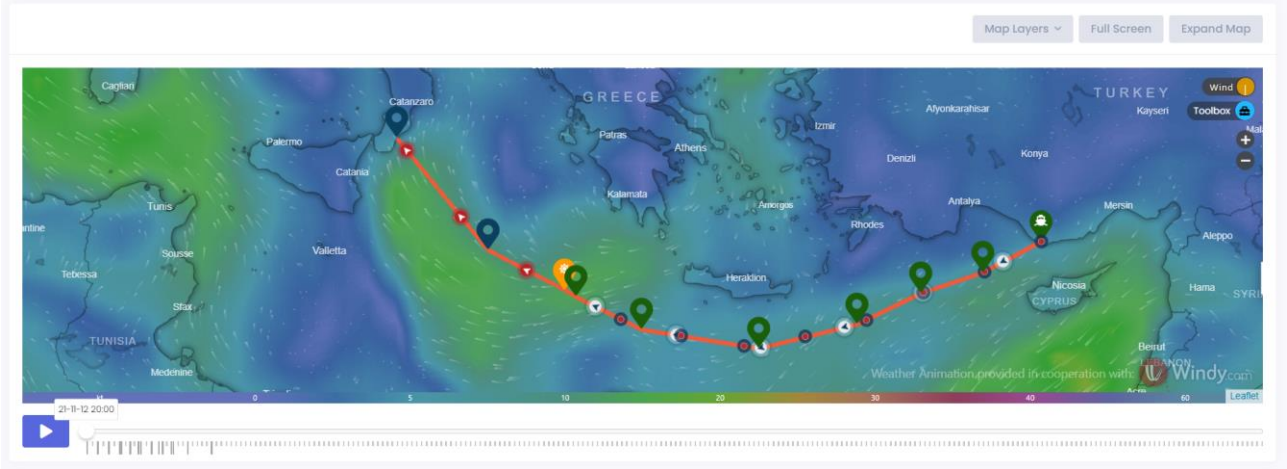
Рисунок 4.21 – Позначка репорту капітана опівдні на мапі

Увесь маршрут до репорту вважається актуальним (фактичним), його позначка – суцільна лінія та зелені маркери, запланований маршрут відображається переривчастою лінією та синіми маркерами.

Для фактичного шляху можливо отримати погодні дані з обраним інтервалом та часом і датою початку (див. рис. 4.22).



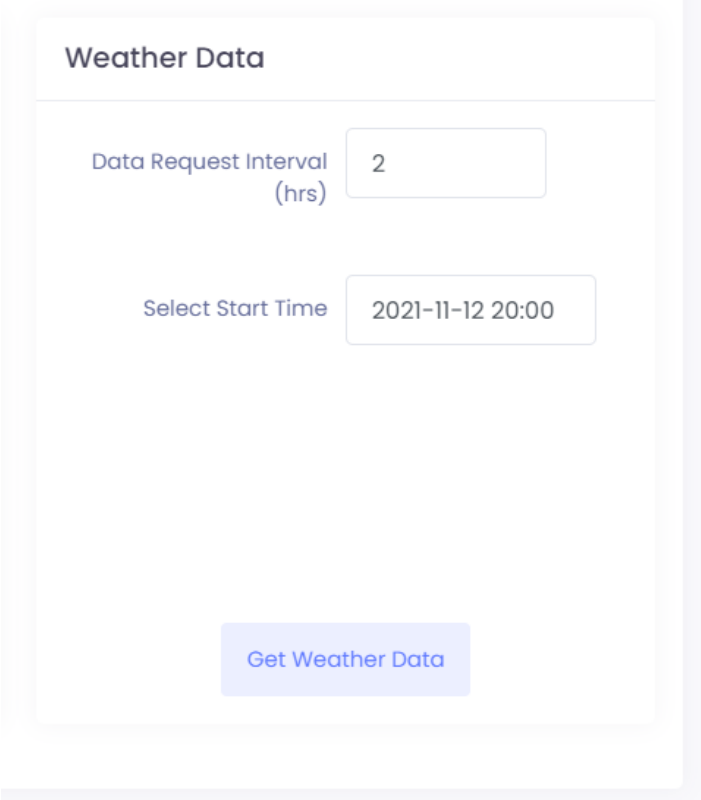
а) форма для отримання погодних даних



б) розраховані шляхові точки з погодними даними

Рисунок 4.22 – Отримання погодних даних для фактичного маршруту

Для запланованого маршруту також можливо отримати погодні дані (див. рис. 4.23).



а) форма для отримання погодних даних

б) розраховані шляхові точки з погодними даними

Рисунок 4.23 – Отримання погодних даних для запланованого маршруту

Розраховані шляхові точки містять багато даних (див. рис. 4.24, 4.25).

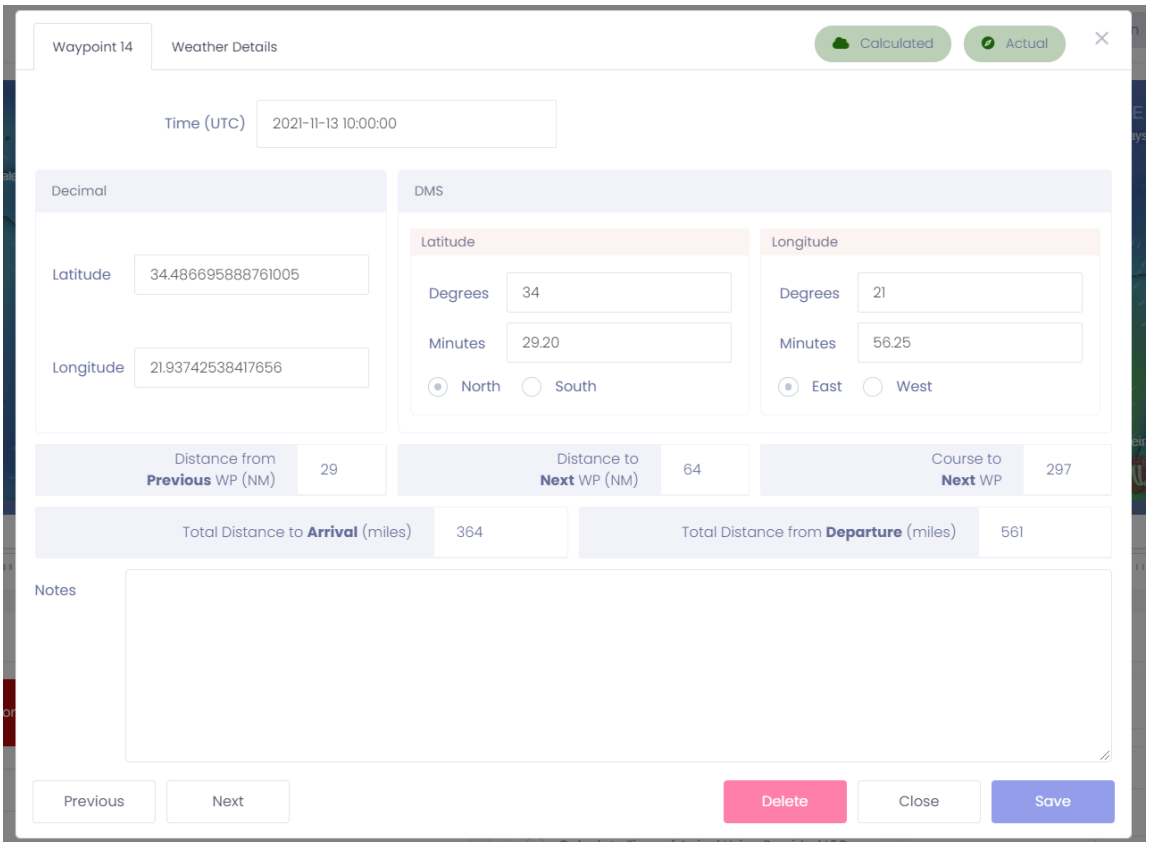
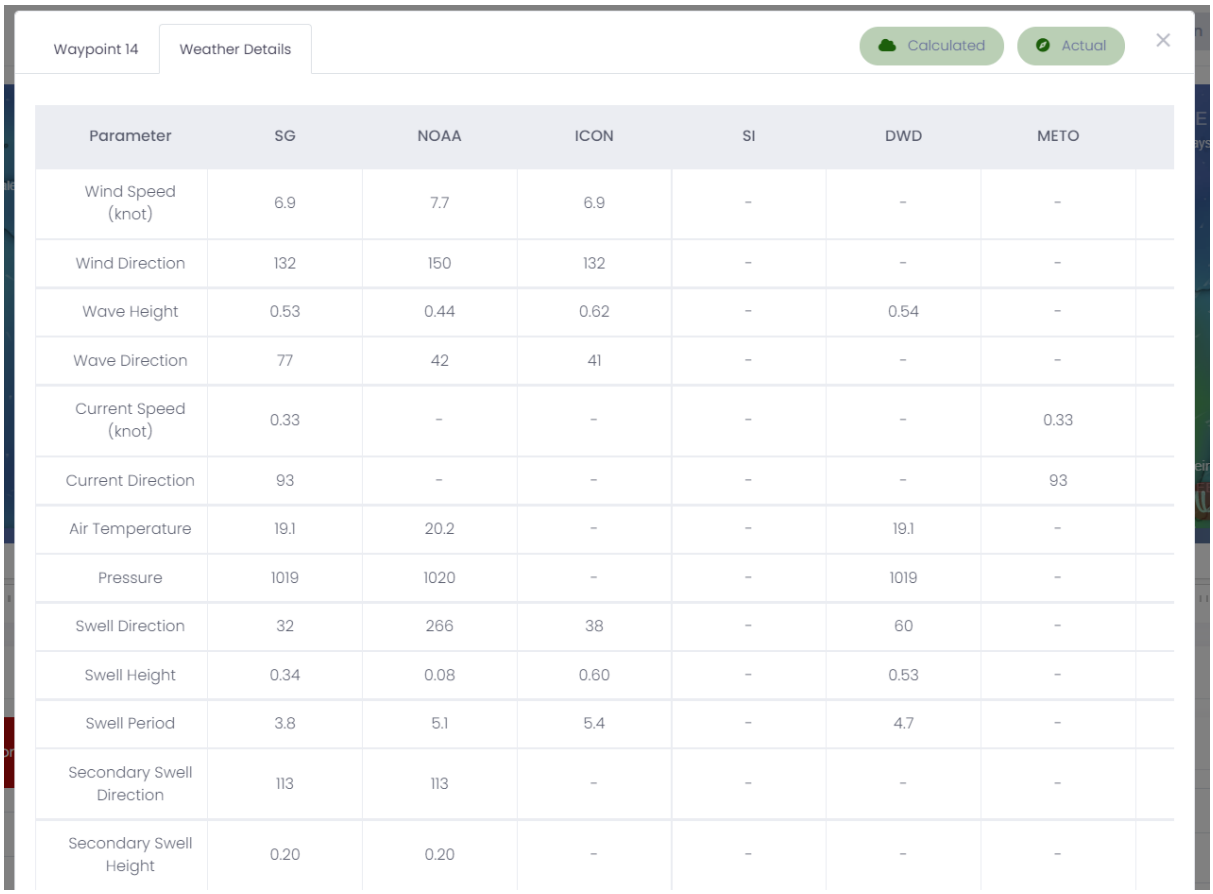


Рисунок 4.24 – Деталі розрахованої актуальної шляхової точки



Parameter	SG	NOAA	ICON	SI	DWD	METO
Wind Speed (knot)	6.9	7.7	6.9	-	-	-
Wind Direction	132	150	132	-	-	-
Wave Height	0.53	0.44	0.62	-	0.54	-
Wave Direction	77	42	41	-	-	-
Current Speed (knot)	0.33	-	-	-	-	0.33
Current Direction	93	-	-	-	-	93
Air Temperature	19.1	20.2	-	-	19.1	-
Pressure	1019	1020	-	-	1019	-
Swell Direction	32	266	38	-	60	-
Swell Height	0.34	0.08	0.60	-	0.53	-
Swell Period	3.8	5.1	5.4	-	4.7	-
Secondary Swell Direction	113	113	-	-	-	-
Secondary Swell Height	0.20	0.20	-	-	-	-

Рисунок 4.25 – Деталі погоди розрахованої актуальної шляхової точки

Окрім звітів капітану корабля в системі є звичайні текстові звіти, які завантаживши у системи потім можливо завантажити на свій пристрій з системи (див. рис. 4.26, 4.27).

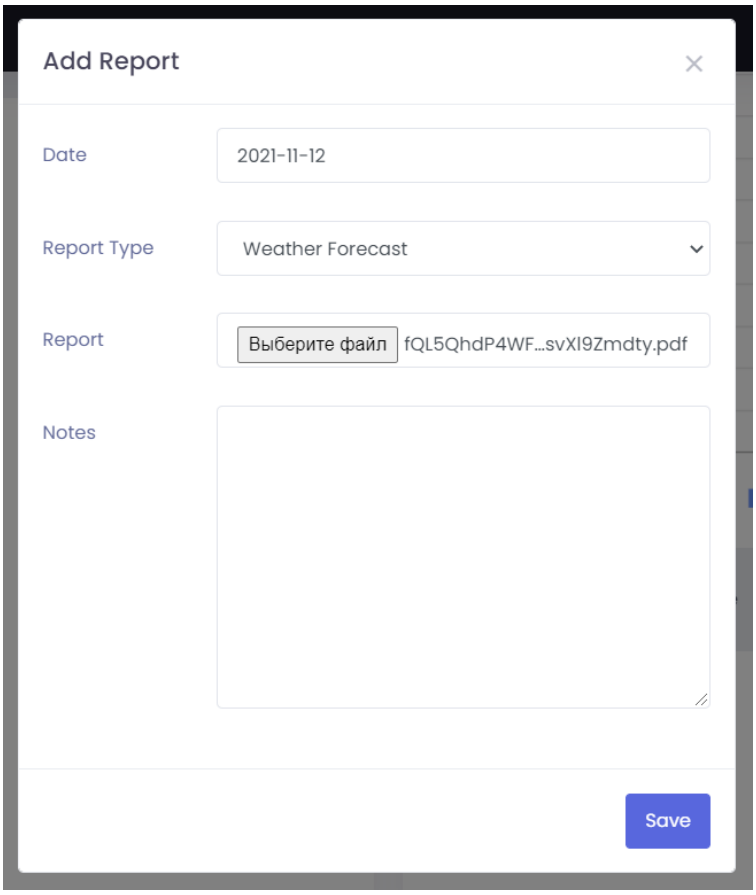


Рисунок 4.26 – Форма завантаження репорту

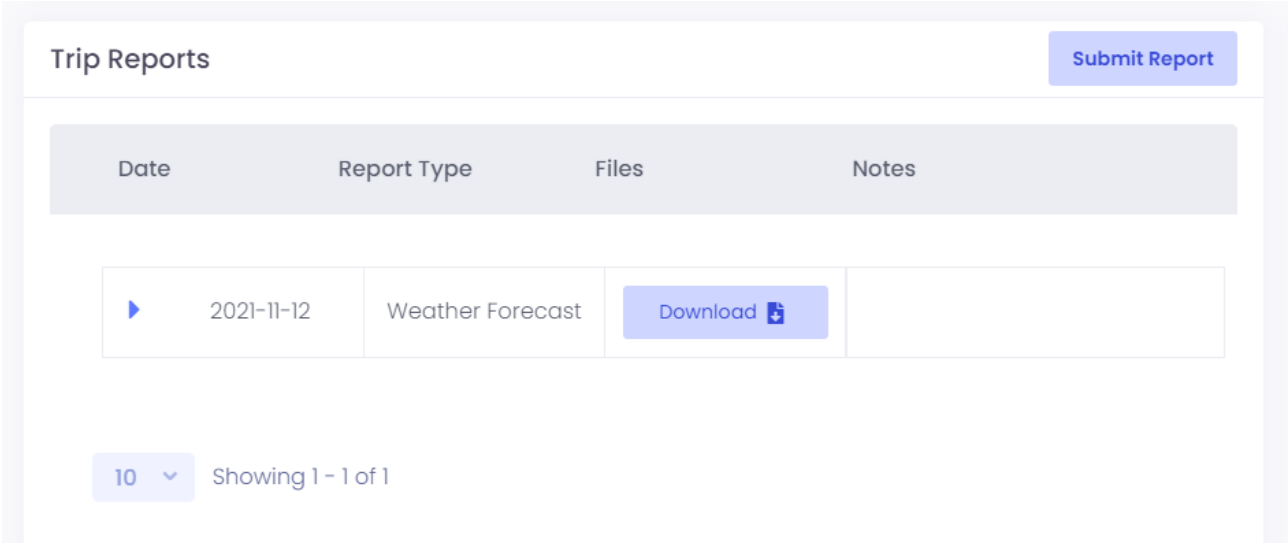


Рисунок 4.27 – Список репортів

Усі дані щодо шляхових точок представлені також у таблиці (див. рис. 4.28).

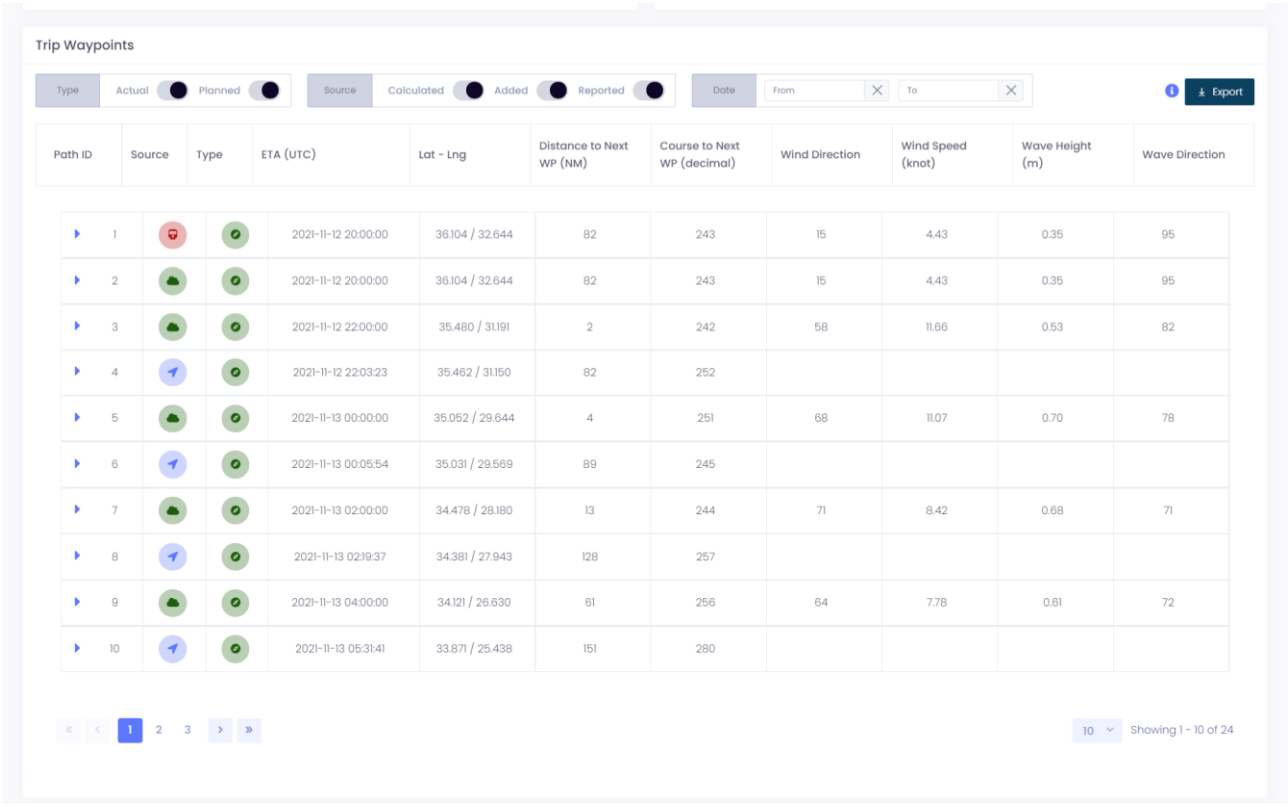


Рисунок 4.28 – Таблиця шляхових точок

Шляхові точки можливо експортувати у вигляді csv-файлу (див. рис. 4.29).

	A	B	C	D	E	F	G	H	I	J	K	L	M
1	path_order	source	type	time	lat	lng	distance_to_next	course_to_next	wind_direction	wind_speed	wave_height	wave_direction	wind_wave_height
2	1	Departure Report	Actual	2021-11-12 20:00:00	36,104	32,644	82	243	15	4,43	0,35	95	0,08
3	2	Calculated	Actual	2021-11-12 20:00:00	36,104	32,644	82	243	15	4,43	0,35	95	0,08
4	3	Calculated	Actual	2021-11-12 22:00:00	35,48	31,191	2	242	58	11,66	0,53	82	0,25
5	4	Added	Actual	2021-11-12 22:03:23	35,462	31,15	82	252					
6	5	Calculated	Actual	2021-11-13 00:00:00	35,052	29,644	4	251	68	11,07	0,7	78	0,37
7	6	Added	Actual	2021-11-13 00:05:54	35,031	29,569	89	245					
8	7	Calculated	Actual	2021-11-13 02:00:00	34,478	28,18	13	244	71	8,42	0,68	71	0,15
9	8	Added	Actual	2021-11-13 02:19:37	34,381	27,943	128	257					
10	9	Calculated	Actual	2021-11-13 04:00:00	34,121	26,63	61	256	64	7,78	0,61	72	0
11	10	Added	Actual	2021-11-13 05:31:41	33,871	25,438	151	280					
12	11	Calculated	Actual	2021-11-13 06:00:00	33,926	25,065	132	280	54	8,37	0,4	83	0,01
13	12	Calculated	Actual	2021-11-13 08:00:00	34,143	23,477	52	279	120	7,51	0,46	47	0,03
14	13	Added	Actual	2021-11-13 09:17:14	34,272	22,45	92	297					
15	14	Calculated	Actual	2021-11-13 10:00:00	34,487	21,937	64	297	132	6,86	0,53	77	0,22
16	15	Added	Actual	2021-11-13 11:35:28	34,959	20,781	16	300					
17	16	Noon Report	Actual	2021-11-13 12:00:00	35,095	20,493	108	298	97	5,99	0,49	78	0,01
18	Sum								83	8,48	0,55	74	0,13
19	16	Noon Report	Actual	2021-11-13 12:00:00	35,095	20,493	108	298					
20	17	Calculated	Planned	2021-11-13 13:00:00	35,408	19,772	68	298	112	9,4	0,58	91	0,09
21	18	Added	Planned	2021-11-13 14:41:32	35,926	18,54	177	322					
22	19	Calculated	Planned	2021-11-13 15:00:00	36,086	18,382	164	321	134	8,1	0,59	117	0
23	20	Calculated	Planned	2021-11-13 17:00:00	37,123	17,34	84	321	149	9,02	0,61	120	0
24	21	Calculated	Planned	2021-11-13 19:00:00	38,151	16,269	4	320	43	0,76	0,54	126	0,1
25	22	Added	Planned	2021-11-13 19:06:22	38,206	16,211			43	0,76	0,54	126	0,1

Рисунок 4.29 – Фрагмент файлу з експортом шляхових точок

Також в окремій таблиці можливо переглянути дані щодо фактичного та запланованого шляху (див. рис. 4.30).

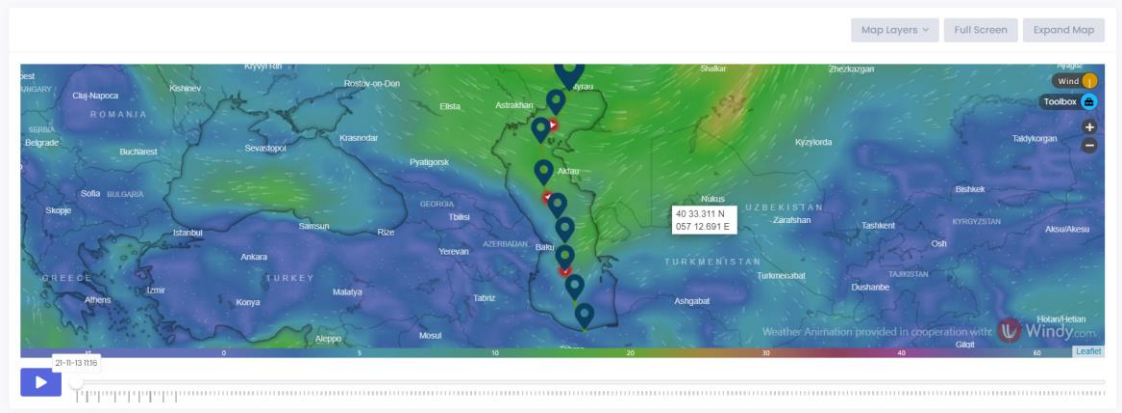
Trip Legs

Leg	Type	Color	Departure	From Lat - Lng	Arrival	To Lat - Lng	Speed (knots)	Fuel Type	Distance (NM)	Time
1 - 16	Actual		2021-11-12 20:00:00	36.10 / 32.64	2021-11-13 12:00:00	35.10 / 20.49	40.04		641	0d 16hrs 0min
16 - 22	Planned		2021-11-13 12:00:00	35.10 / 20.49	2021-11-13 19:06:22	38.21 / 16.21	40.00		284	0d 7hrs 6min
Totals									Actual: 641, Planned: 284, Total: 925.	Actual: 0d 16hrs 0min, Planned: 0d 7hrs 6min, Total: 0d 23hrs 6min.

10 Showing 1 - 3 of 3

Рисунок 4.30 – Таблиця з даними фактичного та актуального шляху

За замовчуванням час прибуття розраховується на підставі запланованої швидкості, але система має можливість розрахувати швидкість на підставі необхідного часу прибуття (див. рис. 4.31).



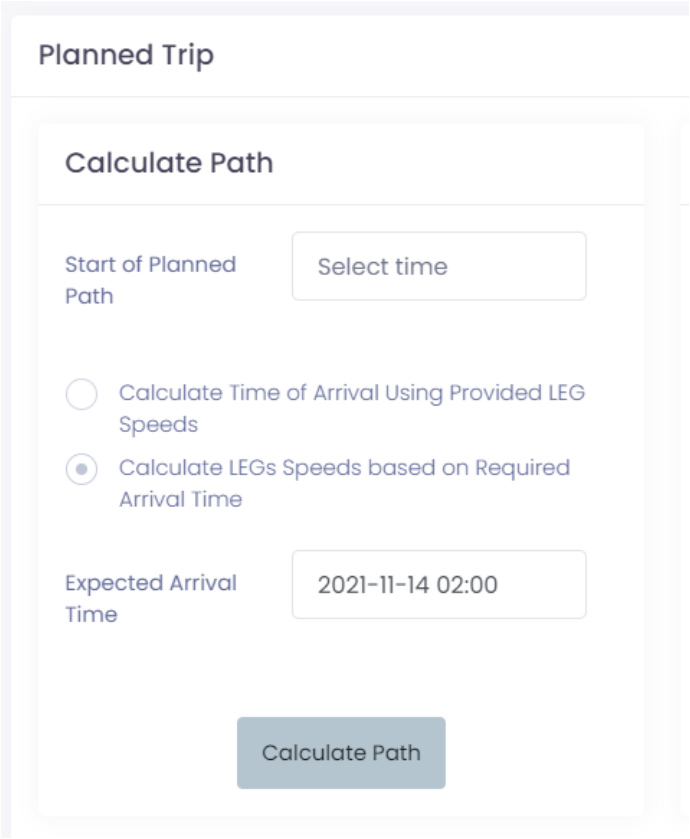
а) створений маршрут

Trip Legs

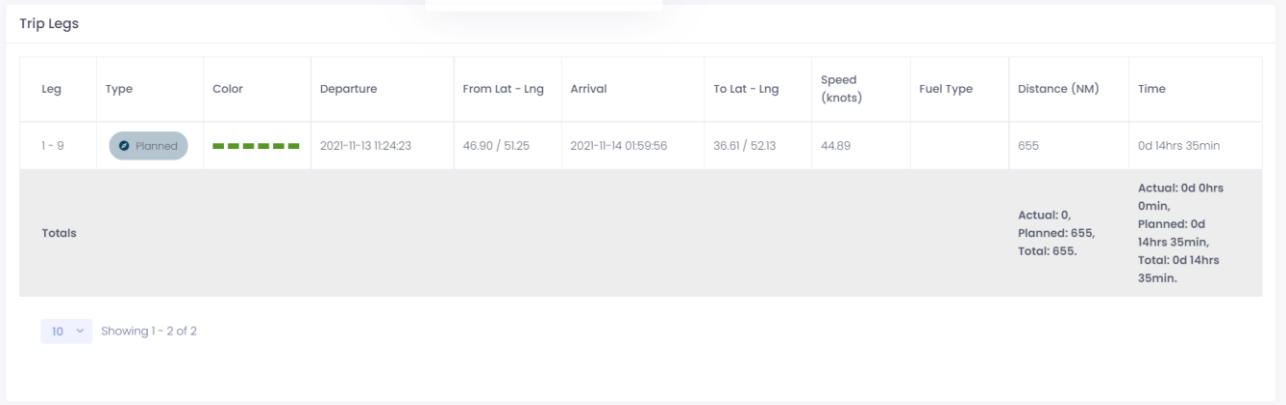
Leg	Type	Color	Departure	From Lat - Lng	Arrival	To Lat - Lng	Speed (knots)	Fuel Type	Distance (NM)	Time
1 - 9	Planned		2021-11-13 11:18:32	46.90 / 51.25	2021-11-14 09:08:36	36.61 / 52.13	30.00		655	0d 21hrs 50min
Totals									Actual: 0, Planned: 655, Total: 655.	Actual: 0d 0hrs 0min, Planned: 0d 21hrs 50min, Total: 0d 21hrs 50min.

10 Showing 1 - 2 of 2

б) розрахований час прибуття на підставі запланованої швидкості



в) форма для розрахунку



г) розрахована швидкість та час поїздки на підставі необхідного часу прибуття

Рисунок 4.31 – Розрахунки для маршруту

На головній сторінці також можливо переглянути усі активні поїздки на мапі, при наведенні на які відображається вікно з даними маршруту (див. рис. 4.32).

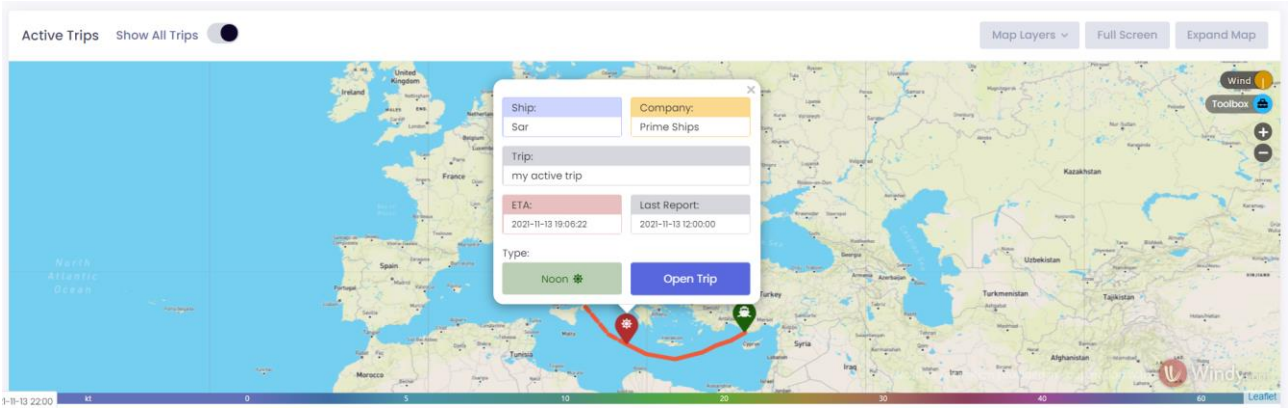


Рисунок 4.32 – Мапа активних поїздок

4.3 Контроль виконання контрактів денної продуктивності

Для контролю виконання контрактів денної продуктивності треба додати дані контракту до системи (див. рис. 4.33).

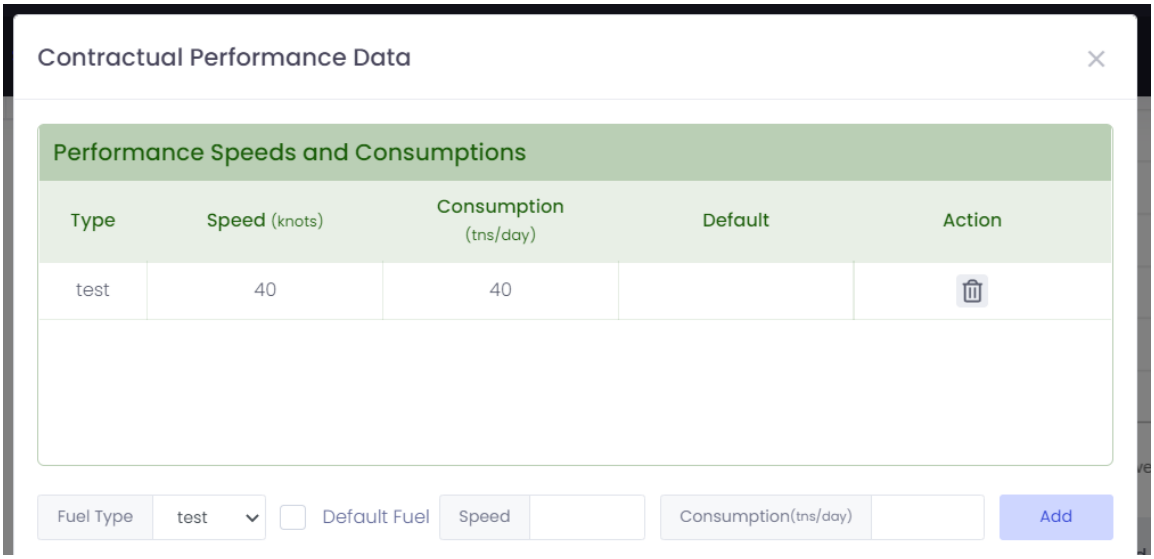
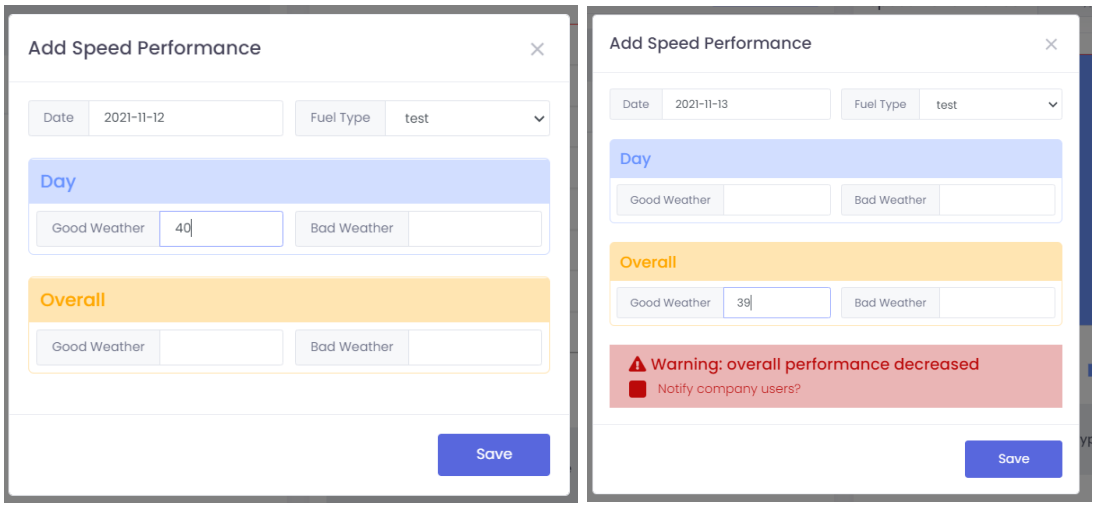


Рисунок 4.33 – Дані контракту

Необхідні дані щодо швидкості та оцінки погоди необхідно заносити у систему щоденно (див. рис. 4.34).



а) внесення задовільних даних б) внесення незадовільних даних

Рисунок 4.34 – Форма для збереження даних денної продуктивності

У разі якщо погода була хорошою або в цілому хорошою, але продуктивність менша ніж зазначена в контракті, то буде запропоновано сповістити про це співробітників компанії.

Усі дані щодо виконання контрактів денної продуктивності будуть представлені у вигляді таблиці та стовпчастої діаграми (див. рис. 4.35).

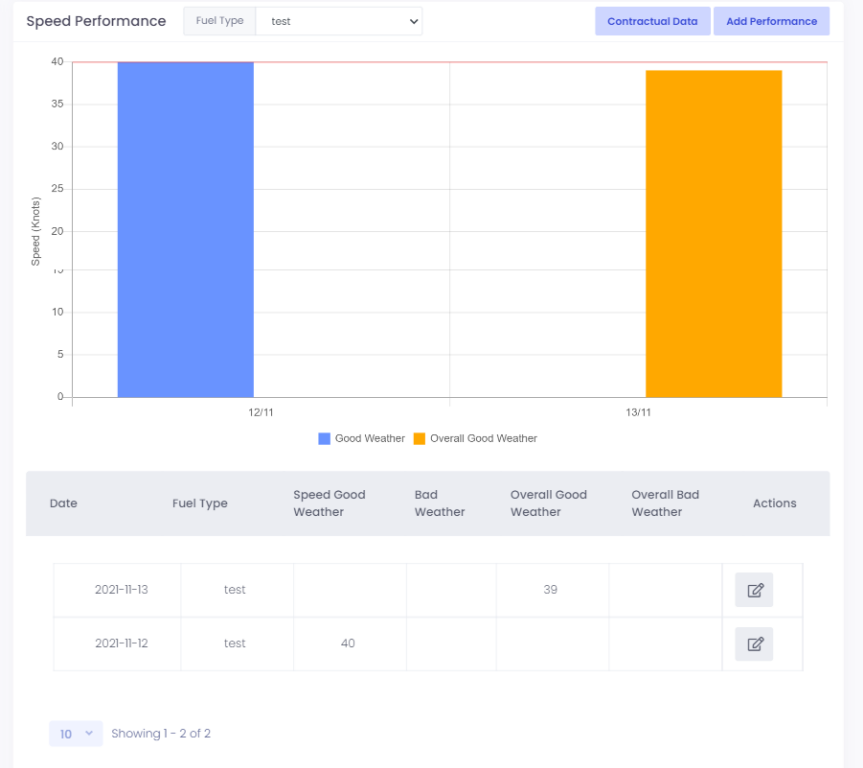


Рисунок 4.35 – Дані виконання контрактів

ВИСНОВКИ

У процесі розробки геоінформаційної системи управління судноплавством було проведено дослідження предметної області, розроблено технічне завдання, були створені діаграми прецедентів, класів та розгортання, а також побудована інформаційна модель.

Для створення цієї системи обрано один з найпопулярніших фреймворків – Laravel, навички роботи з яким було удосконалено. Крім цього, у ході розробки було отримані навички роботи з мапами та API для отримання погодних даних.

Розроблена геоінформаційна система дозволяє здійснювати:

- управління користувачами;
- управління компаніями;
- управління країнами;
- управління портами;
- управління класифікаційними товариствами суден;
- управління паливом для судна;
- управління суднами;
- управління судноплавними шляхами, включаючи планування маршруту та відстеження фактичного маршруту та актульної позиції;
- перегляд погодних даних на мапі та у вигляді таблиць;
- підтримку різних типів звітів;
- контроль виконання контрактів денної продуктивності.

Система має сучасний дизайн, зрозумілий інтерфейс користувачів системи. Після кожної дії користувача у системі відображається повідомлення успішного завершення операції або з помилкою, що унеможливлює будь-які непорозуміння при роботі з системою.

Геоінформаційна система значно скорочує ресурси часу, забезпечує надійність та точність, допомагає уникнути помилок при плануванні маршруту та загалом спростити його, надає зручний функціонал для відстеження

фактичного маршруту та актуальної позиції судна, контролю виконання контрактів денної продуктивності.

Таким чином мета кваліфікаційної роботи досягнута.

ПЕРЕЛІК ПОСИЛАНЬ

1. Вигерс К. Разработка требований к программному обеспечению. Петербург : БХВ-Петербург, 2014. 736 с.
2. MySQL Documentation. URL: https://dev.mysql.com/doc/ (дата звернення: 20.09.2021).
3. Metronic Documentation. URL: https://keenthemes.com/metronic/?page=docs (дата звернення: 10.10.2021).
4. Mapbox Documentation. URL: https://docs.mapbox.com/ (дата звернення: 21.10.2021).
5. Leaflet Documentation. URL: https://leafletjs.com/reference.html (дата звернення: 1.11.2021).
6. Windy API Documentation. URL: https://api.windy.com/map-forecast (дата звернення: 16.11.2021).
7. Laravel Documentation. URL: https://laravel.com/ (дата звернення: 11.08.2021).
8. Stormglass.io Documentation. URL: https://docs.stormglass.io/#/ (дата звернення: 20.11.2021).

ДОДАТОК А

Map Forecast API

Map Forecast API – це проста у використанні бібліотека, заснована на Leaflet 1.4.x. Вона дозволяє використовувати все, що пропонує Leaflet або JavaScript, а також візуалізацію мап Windy.com [7].

Щоб API працював, треба виконати 3 прості кроки:

1) отримати ключ Windy API;
2) додати бібліотеку Leaflet 1.4.0 на веб-сайт: https://unpkg.com/leaflet@1.4.0/dist/leaflet.js;
3) додати бібліотеку Windy API на веб-сайт: https://api.windy.com/assets/map-forecast/libBoot.js.

Існує велика різниця між самою мапою та візуалізаціями мапи Windy: мапа забезпечує базову інтерактивність та функціональні можливості, такі як масштабування, перетягування, переміщення, обробка клацань тощо, а візуалізація мапи Windy є лише розширенням мапи.

Мапа реалізована за допомогою бібліотеки Leaflet. Усе, що можливо робити з бібліотекою Leaflet, також можливо робити за допомогою API прогнозу мап.

Візуалізація Windy покращує мапу шарами, частинками та ізолініями. Цими візуалізаціями можна керувати програмно або за допомогою елементів керування UI. Для програмного керування візуалізаціями потрібно використовувати об’єкт JavaScript windyAPI, який повертається після ініціалізації API. Щоб керувати візуалізаціями за допомогою елементів керування інтерфейсу користувача, потрібно скористатися меню праворуч, індикатором перебігу, легендою або засобом вибору.

Функції, що надаються об’єктом JavaScript windyAPI:

- store (найважливіший клас API, він зберігає стан мапи та візуалізації. Це також дозволяє змінити стан. Усі значення зберігаються в структурі ключ-значення);
- map (це екземпляр класу Leaflet L.Map);
- picker (засіб вибору погоди, щоб отримати значення поточного шару за певними координатами. Засіб вибору працює лише на комп’ютері, але можливо налаштувати його працювати навіть на мобільних пристроях самостійно);
- utils (збірник корисних методів);
- broadcast (обробник подій);
- overlays (цей об’єкт містить усі екземпляри всіх шарів. Це стане в нагоді, коли потрібно змінити метрику шару);
- colors (цей об’єкт містить усі кольорові визначення всіх шарів. Це стане в нагоді, коли потрібно змінити визначення кольору будь-якого шару).

ДОДАТОК Б

Огляд глобального API погоди stormglass.io

API Storm Glass дозволяє отримувати дані про погоду для будь-якої координати на земній кулі у простий програмний спосіб, використовуючи звичайні HTTP-запити (див. рис. Б.1) [9]. Коли запит буде успішним, відповідь буде відправлена у вигляді об'єкта JSON.

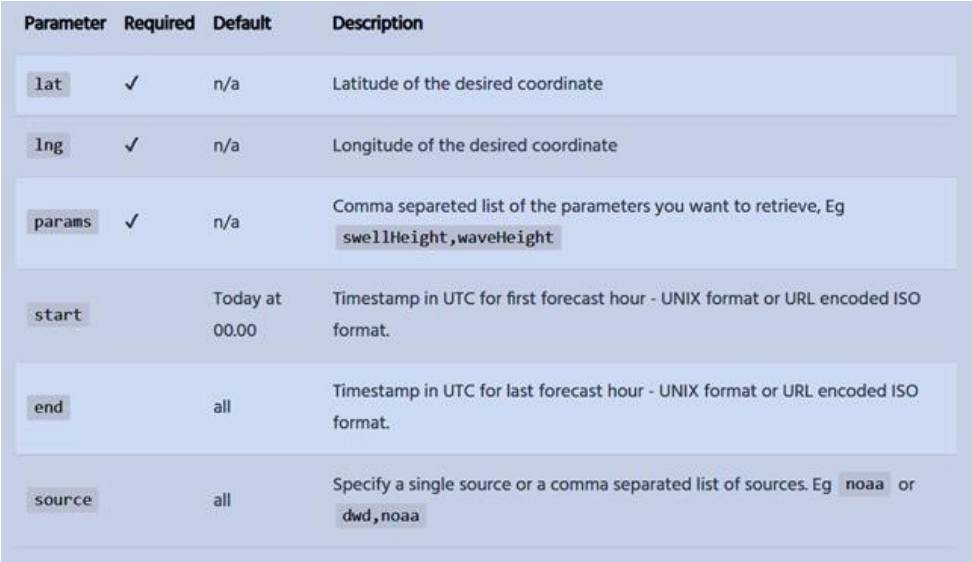
Рисунок Б.1 – Кінцева точка API

Storm Glass очікує, що ключ API буде включений у всі запити API до сервера в заголовку, який виглядає так: «Authorization: example-api-key».

Запит погоди використовується для отримання даних про погоду для точки. API Storm Glass забезпечує морську погоду, а також глобальну погоду для суші та озер (див. рис. Б.2). Щоб отримати морські дані, потрібно включити в запит координати точки моря (океану тощо), а для отримання даних для суші та озер – координати, розташовані на суші або на озері.

Відповідь буде відправлено у вигляді об'єкта JSON. Корінь ресурсу містить два об'єкти:

- мета (об'єкт містить інформацію про запит API. Наприклад, запитані широта та довгота, денна квота та кількість запитів, які ви зробили за сьогодні);
- дані (об'єкт даних містить фактичні дані про погоду на погодинній основі. Один елемент у масиві даних містить багато погодних даних (див. табл. Б.1)).



Parameter	Required	Default	Description
`lat`	✓	n/a	Latitude of the desired coordinate
`lng`	✓	n/a	Longitude of the desired coordinate
`params`	✓	n/a	Comma separeted list of the parameters you want to retrieve, Eg `swellHeight,waveHeight`
`start`		Today at 00.00	Timestamp in UTC for first forecast hour - UNIX format or URL encoded ISO format.
`end`		all	Timestamp in UTC for last forecast hour - UNIX format or URL encoded ISO format.
`source`		all	Specify a single source or a comma separated list of sources. Eg `noaa` or `dwd,noaa`

Рисунок Б.2 – Доступні параметри запиту

Кожен параметр є об'єктом, який містить key і value для кожного доступного джерела.

Storm Glass надає дані кількох погодних інститутів по всьому світу:

- SG (Storm Glass AI) — це інтелектуальна глобальна мережа, яка автоматично вибирає найкраще джерело погоди залежно від місця розташування;
- ICON — національна метеорологічна служба Німеччини;
- DWD — національна метеорологічна служба Німеччини;
- NOAA — національне управління океанічних і атмосферних досліджень;
- Météo-France (meteo) — національна метеорологічна служба Франції;
- UK MetOffice (meto) — національна служба погоди Великобританії;
- FCOO — данський оборонний центр оперативної океанографії;
- FMI — фінська метеорологічна установа;
- YR — норвезький метеорологічний інститут і NRK;
- SMHI — шведський метеорологічний і гідрологічний інститут.

Кожне джерело містить власний набір атрибутів, і дані оновлюються індивідуально з різними інтервалами.

Таблиця Б.1 – Опис даних відповіді API

Key	Value
time	Timestamp in UTC
airTemperature	Air temperature in degrees celsius
airTemperature80m	Air temperature at 80m above sea level in degrees celsius
airTemperature1000hpa	Air temperature at 1000hpa in degrees celsius
pressure	Air pressure in hPa
cloudCover	Total cloud coverage in percent
currentDirection	Direction of current. 0° indicates current coming from north
currentSpeed	Speed of current in meters per second
gust	Wind gust in meters per second
humidity	Relative humidity in percent
iceCover	Proportion, 0-1
precipitation	Mean precipitation in kg/m²/h = mm/h
snowDepth	Depth of snow in meters
seaLevel	Sea level relative to MSL
swellDirection	Direction of swell waves. 0° indicates swell coming from north
swellHeight	Height of swell waves in meters
swellPeriod	Period of swell waves in seconds
secondarySwellPeriod	Direction of secondary swell waves. 0° indicates swell coming from north
secondarySwellDirection	Height of secondary swell waves in meters
secondarySwellHeight	Period of secondary swell waves in seconds
visibility	Horizontal visibility in km
waterTemperature	Water temperature in degrees celsius
waveDirection	Direction of combined wind and swell waves. 0° indicates waves coming from north
waveHeight	Significant Height of combined wind and swell waves in meters
wavePeriod	Period of combined wind and swell waves in seconds
windWaveDirection	Direction of wind waves. 0° indicates waves coming from north
windWaveHeight	Height of wind waves in meters
windWavePeriod	Period of wind waves in seconds
windDirection	Direction of wind at 10m above sea level. 0° indicates wind coming from north
windDirection20m	Direction of wind at 20m above sea level. 0° indicates wind coming from north
windDirection1000hpa	Direction of wind at 1000hpa. 0° indicates wind coming from north
windSpeed	Speed of wind at 10m above sea level in meters per second.
windSpeed20m	Speed of wind at 20m above sea level in meters per second.
windSpeed30m	Speed of wind at 30m above sea level in meters per second.
windSpeed40m	Speed of wind at 40m above sea level in meters per second.
windSpeed50m	Speed of wind at 50m above sea level in meters per second.
windSpeed1000hpa	Speed of wind at 1000hpa in meters per second.
windSpeed800hpa	Speed of wind at 800hpa in meters per second.

ДОДАТОК В

Код класу StormGlass

```
<?php

namespace App\External;

class StormGlass
{
    const API_ENDPOINT = 'https://api.stormglass.io/v1';

    protected $key;
    protected $response;

    /**
     * StormGlass constructor.
     *
     * @param string $key API key
     */
    public function __construct($key)
    {
        $this->key = $key;

        $this->weatherKeys = [
            'windSpeed' => null,
            'windDirection' => null,
            'waveHeight' => null,
            'waveDirection' => null,
            'currentSpeed' => null,
            'currentDirection' => null,
            'airTemperature' => null,
            'pressure' => null,
            'swellDirection' => null,
            'swellHeight' => null,
            'swellPeriod' => null,
            'secondarySwellPeriod' => null,
            'secondarySwellDirection' => null,
            'secondarySwellHeight' => null,
            'windWaveDirection' => null,
            'windWaveHeight' => null,
            'windWavePeriod' => null
        ];

        $this->response = [
            'sources' => [],
            'time' => ""
        ];
    }

    /**
     * Will initiate the process to get the data from the stormglass API.
     *
     * @param array $latlng
     * @param null $start
     * @param null $options
     * @return array
```

```
     */
    public function getWeather($latlng, $start = null, $options = null)
    {
        while ($latlng['lng'] > 180) {
            $latlng['lng'] -= 360;
        }

        while ($latlng['lng'] < -180) {
            $latlng['lng'] += 360;
        }

        $url = '/weather/point?lat=' . $latlng['lat'] . '&lng=' .
$latlng['lng'];

        if ($start) {
            $url .= '&start=' . urlencode($start) . '&end=' . urlencode($start);
        }

        if ($options === null) {
            $url .=
'&params=windSpeed,windDirection,waveHeight,waveDirection,currentSpeed,currentDi
rection,airTemperature,pressure,swellDirection,swellHeight,swellPeriod,secondary
SwellPeriod,secondarySwellDirection,secondarySwellHeight,windWaveDirection,windW
aveHeight,windWavePeriod';
        }

        $response = $this->sendRequest($url);
        $data = json_decode($response);

        $this->response = [
            'sources' => [],
            'time' => ""
        ];

        if ($data) {
            $this->prepare($data);
        }

        return $this->response;
    }

    /**
     * Send the request to the StormGlass API with the prepared URL that
includes all the info needed.
     *
     * @param string $url
     * @return bool|string
     */
    protected function sendRequest($url)
    {
        $ch = curl_init();

        curl_setopt_array($ch, [
            CURLOPT_URL => self::API_ENDPOINT . $url,
            CURLOPT_RETURNTRANSFER => true,
            CURLOPT_ENCODING => '',
            CURLOPT_MAXREDIRS => 10,
            CURLOPT_TIMEOUT => 0,
            CURLOPT_FOLLOWLOCATION => true,
            CURLOPT_HTTP_VERSION => CURL_HTTP_VERSION_1_1,
            CURLOPT_CUSTOMREQUEST => 'GET',
            CURLOPT_HTTPHEADER => [
              "Authorization: {$this->key}"
            ],
```

```
        ]);

        $response = curl_exec($ch);
        $error = curl_error($ch);

        if ($error) {
            echo '<pre>' . print_r($error, true) . '</pre>';
            exit;
        }

        curl_close($ch);

        return $response;
    }

    /**
     * This will take the raw response from stormglass and prepare the response
object with the fixed data
     *
     * @param array $data
     */
    protected function prepare($data)
    {
        if (isset($data->hours[0])) {
            foreach ($data->hours[0] as $param => $value) {
                if ($value) {
                    if ($param === 'time') {
                        $this->response[$param] = $value;
                    } else {
                        $this->processWeatherAttribute($param, $value);
                    }
                }
            }
        }
    }

    /**
     * Process received weather attribute data
     *
     * @param string $attribute
     * @param array|int $sources
     */
    protected function processWeatherAttribute($attribute, $sources)
    {
        if (is_array($sources)) {
            foreach ($sources as $source) {
                $this->processSource($attribute, $source);
            }
        } else {
            $this->processSource($attribute, ['source' => 'sg', 'value' =>
$sources]);
        }
    }

    /**
     * Process weather attribute data from single source
     *
     * @param string $attribute
     * @param array $source
     */
    protected function processSource($attribute, $source)
    {
        if (!isset($this->response['sources'][$source->source])) {
            $this->response['sources'][$source->source] = $this->weatherKeys;
```

```
        }

        // Convert direction
        if ($attribute === 'currentDirection') {
            $tmpCurrent = round($source->value + 180);
            $source->value = $tmpCurrent >= 360
                ? $tmpCurrent - 360
                : $tmpCurrent;
        }

        // Convert speed form m/s to km/h
        $multiplier = 1;

        if ($attribute === 'windSpeed' || $attribute === 'currentSpeed') {
            $multiplier = 3.6;
        }

        $this->response['sources'][$source->source][$attribute] = $source->value
* $multiplier;
    }
}
```