Міністерство освіти і науки України
Криворізький національний університет
Кафедра моделювання та програмного забезпечення

КВАЛІФІКАЦІЙНА РОБОТА

на здобуття ступеня вищої освіти магістра

за спеціальністю 121 – Інженерія програмного забезпечення

На тему: Розробка веб-застосунку із використанням алгоритмів предиктивної аналітики

Засвідчую, що в цій кваліфікаційній роботі немає запозичень із праць інших авторів без відповідних посилань.
Студент гр. ІПЗ-24м
_____ *О.С. Покровський*

Керівник
кваліфікаційної роботи _____ / *О.В. Шамрай* /

Завідувач кафедри: _____ / *А.М. Стрюк* /

Кривий Ріг

2025

Криворізький національний університет

Факультет: Інформаційних технологій
Кафедра: Моделювання та програмного забезпечення
Ступінь вищої освіти: магістр
Напрям підготовки: 121 – Інженерія програмного забезпечення

ЗАТВЕРДЖУЮ:
Зав. кафедри
_____ *Стрюк А.М.*
«___» _____ 20__р.

ЗАВДАННЯ
на кваліфікаційну роботу

студенту групи ІПЗ-24м Покровському Олексію Станіславовичу

- Тема: Розробка веб-застосунку з використанням алгоритмів прогнозної аналітики затверджено наказом по КНУ № ____ від « »_____ 20__р.
- Термін подання студентом закінченої роботи: «_» _____20__ р.
- Вихідні дані по роботі: У роботі використано синтетично сформований набір даних, що моделює структуру військових підрозділів, їхню техніку, ресурси, нормативи споживання та умови виконання операцій. До вихідних даних включено: перелік бригад, типи техніки та їх характеристики, ресурсні норми для особового складу та озброєння, поточні запаси, а також множину коефіцієнтів, що відображають вплив зовнішніх факторів (сезон, інтенсивність операції, віддаленість від складу).

- Зміст пояснювальної записки (перелік питань, що їх треба розробити): Пояснювальна записка містить обґрунтування мети та актуальності розробки, аналіз підходів до прогнозування логістичних потреб із залученням базових та умовних ML-методів, опис предметної області та факторів, що впливають на споживання ресурсів. У роботі викладено моделювання

прогнозних процесів, проектування архітектури програмного забезпечення на базі Laravel, реалізацію алгоритмів розрахунку та порівняння результатів різних моделей, а також наведено узагальнення отриманих результатів і висновки.

- Перелік графічного демонстраційного матеріалу: Графічний матеріал включає діаграму варіантів використання системи, блок-схеми реалізованих алгоритмів прогнозування, структурну схему розробленої бази даних, а також знімки екранних форм веб-інтерфейсу, що демонструють роботу програмного забезпечення.

РЕФЕРАТ

ПРОГНОЗУВАННЯ ЛОГІСТИЧНИХ ПОТРЕБ, МОДЕЛІ СПОЖИВАННЯ РЕСУРСІВ, ВОЄННІ ПІДРОЗДІЛИ, НОРМАТИВНІ МОДЕЛІ, МАШИННЕ НАВЧАННЯ, АНАЛІТИКА, LARAVEL, MYSQL.

Пояснювальна записка: 77 с., 10 табл., 7 рис., 18 джерел.

У сучасних умовах високої динаміки бойових дій та зростаючих вимог до оперативного забезпечення, планування логістичних потреб військових підрозділів набуває критичного значення. Підрозділи володіють різнорідною технікою, мають різний склад особового складу та працюють у мінливих умовах, що визначають рівень споживання ресурсів, зокрема пального, боєприпасів та засобів забезпечення. Ефективне прогнозування цих потреб є ключовим чинником стійкості та боєздатності військових формувань.

У роботі розглянуто підходи до побудови системи прогнозування, що поєднує нормативну модель розрахунку («норма × коефіцієнти») та умовну ML-модель, яка дозволяє імітувати аналітичні можливості машинного навчання на основі історичних спостережень та закономірностей споживання ресурсів. Проаналізовано фактори впливу, такі як сезонність, інтенсивність операцій, втрати техніки та віддаленість від пунктів постачання, що суттєво змінюють обсяг прогнозованих потреб.

У процесі дослідження спроектовано та реалізовано програмне рішення на базі Laravel з використанням MySQL. Система включає модуль прогнозування, інтерфейс для введення вихідних даних та блок візуалізації результатів. Проведено порівняння ефективності базового та умовного ML-підходів, наведено висновки щодо їхнього практичного застосування.

Отримані результати демонструють можливість підвищення точності планування логістики та скорочення часових витрат на аналітичні операції, що є важливим для забезпечення стабільної роботи військових підрозділів у складних умовах.

ABSTRACT

LOGISTICS DEMAND FORECASTING, RESOURCE CONSUMPTION MODELS, MILITARY UNITS, NORMATIVE MODELS, CONDITIONAL MACHINE LEARNING, ANALYTICS, LARAVEL, MYSQL.

Thesis in 77 p., 10 table, 7 figure, 18 source.

In modern operational environments, where military activities are highly dynamic and resource-intensive, the ability to accurately forecast logistical needs becomes critically important. Military units operate with diverse equipment, varying personnel structures, and under constantly changing conditions that directly affect the consumption of essential resources such as fuel, ammunition, and supplies. Effective forecasting of these demands is a key factor in maintaining operational readiness and sustainability.

This work examines approaches to building a forecasting system that combines a normative calculation model ("norm × coefficients") with a conditional machine-learning-inspired model designed to simulate analytical capabilities using historical data and observed consumption patterns. The study analyzes key influencing factors such as seasonality, operational intensity, equipment losses, and distance to supply points, all of which significantly affect predicted resource requirements.

A software solution was designed and implemented using the Laravel framework with MySQL as the primary data storage. The system includes a forecasting module, an interface for input data management, and visualization components for viewing analytical results. Comparative analysis of the baseline and conditional ML approaches is carried out, and conclusions regarding their practical applicability are presented.

The results demonstrate improved accuracy of logistical planning and reduced analytical workload, contributing to more reliable support of military units operating in demanding conditions.

Вступ

У сучасних військових конфліктах успіх операцій значною мірою залежить від ефективності системи матеріально-технічного забезпечення (логістики). Забезпечення військ всім необхідним – від боєприпасів та палива до продовольства й медичних засобів – повинно здійснюватися своєчасно і безперервно навіть в умовах динамічних бойових дій. Одним із ключових факторів тут є здатність прогнозувати потреби підрозділів наперед. Серед уроків війни в Україні – небезпечна застарілість традиційних реактивних (постфактум) моделей логістики. Недостатнє планування наперед призводить до ситуацій, коли війська залишаються без пального чи боєприпасів у критичний момент, як це трапилося в російських підрозділах на початку вторгнення 2022 року. Натомість українська сторона за підтримки партнерів продемонструвала ефективність підходу з використанням аналітики даних для прогнозування потреб та завчасного підвезення ресурсів. Це підтвердило: майбутнє військової логістики має бути проактивним, з опорою на прогнозування, а не реактивним реагуванням на дефіцити постфактум.

Отже, проблема прогнозування потреб в військовій логістиці є надзвичайно актуальною. Від здатності точно оцінити наперед, скільки і яких ресурсів знадобиться тому чи іншому підрозділу, залежить успішність бойових операцій, стійкість оборони та збереження життя військовослужбовців. Традиційні методи, засновані лише на нормах споживання чи інтуїції офіцерів тилу, нерідко не враховують динаміку сучасного бою та великий масив впливових факторів (темп операцій, рельєф, погода, ворожий вплив тощо). Тому впровадження систем підтримки прийняття рішень (СППР) для прогнозування логістичних потреб із застосуванням сучасних інформаційних технологій, зокрема методів машинного навчання (ML) та аналітики великих даних, є перспективним шляхом підвищення ефективності військової логістики.

Метою даної кваліфікаційної роботи є дослідження та проектування програмної системи для підтримки прийняття рішень щодо прогнозування потреб військових підрозділів у матеріально-технічних ресурсах. Для досягнення цієї мети у роботі вирішуються такі завдання:

- провести аналіз предметної галузі (структура підрозділів, організація логістики, існуючі проблеми та особливості прогнозування потреб);
- виконати огляд підходів і існуючих досліджень з теми прогнозування потреб та логістичних СППР, а також застосувань ML/аналітики в оборонній сфері;

- визначити вимоги до розроблюваної програмної системи (функціональні, нефункціональні, вимоги до інтерфейсу, даних та інтеграції через API);

- розробити концептуальну модель системи та архітектуру її реалізації, включно з основними компонентами та базовою моделлю даних (ER-діаграмою).

Структурно пояснювальна записка містить чотири розділи. У першому розділі досліджено предметну область військової логістики та обґрунтовано актуальність проблеми прогнозування матеріальних потреб підрозділів. У другому розділі проведено аналіз літературних джерел і існуючих рішень з тематик прогнозування логістичних потреб та використання інформаційних технологій і штучного інтелекту в оборонній логістиці. Третій розділ присвячено визначенню вимог до програмного засобу, що розробляється, з урахуванням особливостей предметної області. У четвертому розділі здійснено моделювання і проектування системи – визначено її основні компоненти, архітектуру та структуру бази даних. Наприкінці роботи сформульовано висновки та окреслено напрями подальшого розвитку системи.

Розділ 1. Дослідження предметної галузі

1.1 Структура військових підрозділів

Військова організація має ієрархічну структуру підрозділів, що впливає на характер забезпечення та обсяг потреб. Базовою одиницею є **відділення** (група солдат чисельністю 5–10 осіб). Кілька відділень утворюють **взвод** (~30 осіб), взводи об’єднуються в **роту** (за штатами ЗСУ близько 100 осіб). Роти входять до складу **батальйону** (як правило, 300–500 військовослужбовців), а батальйони – до **бригади** (близько 2–4 тисяч осіб). Вище бригад можуть існувати об’єднання – **дивізії**, **корпуси**, проте в сучасній українській армії основною тактичною одиницею є бригада. Кожен рівень має визначені підрозділи бойового, забезпечувального та штабного характеру.

Для ефективної логістики важливо враховувати цю структуру, адже потреби формуються на нижчих рівнях (солдати, відділення) і агрегуються на верхніх рівнях для планування. Наприклад, потреби взводу є сумою потреб його відділень, проте при переході на рівень роти чи батальйону додаються потреби підрозділів забезпечення (ремонтних, медичних тощо) і збільшується масштаб перевезень. Структура підрозділів визначає і штатну техніку: так, механізована рота має бойові машини піхоти, танкова – танки, артилерійська батарея – гармати тощо. Від цього залежить і номенклатура потрібних ресурсів

(паливо для техніки, боєприпаси відповідних калібрів, запчастини до техніки тощо).

Організаційно тилове (логістичне) забезпечення здійснюється через спеціальні підрозділи забезпечення. На рівні батальйону існує **рота матеріального забезпечення**, на рівні бригади – **батальйон логістичного забезпечення** або окремі роти постачання, перевезення та ремонту. Вони відповідають за отримання матеріальних засобів зі складів вищого рівня та доставку їх до підрозділів бригади. Вищі ланки (оперативне командування, Генеральний штаб) планують логістичні операції в масштабах театру воєнних дій, керують складами і базами постачання та координують постачання між бригадами. Таким чином, ланцюг постачання у військах проходить від стратегічного рівня (центральні склади, бази), через оперативний (склади оперативного командування, бригадні тилові бази) до тактичного рівня (батальйонні і ротні постачальники, аж до взводних відділень постачання на передовій).

1.2 Система військової логістики

Військова логістика охоплює планування, закупівлю, зберігання, транспортування та розподіл матеріальних ресурсів для забезпечення бойової діяльності військ. На відміну від цивільних ланцюгів постачання, військова логістика діє в умовах підвищеної невизначеності, ворожого протидії та жорстких вимог до надійності. Логістична система Збройних Сил включає декілька класів матеріальних засобів:

- **Боєприпаси та озброєння.** Включає усі типи боєприпасів (патрони, артилерійські снаряди, ракети), вибухові речовини, а також стрілецьку і важку зброю. Забезпечення боєприпасами є критично важливим: їх витрати залежать від інтенсивності боїв і можуть різко зростати під час наступальних операцій.

- **Паливо та мастильні матеріали.** Паливо (бензин, дизель, авіаційний гас) потрібне для роботи техніки – від транспортних машин до танків і літаків. Норми витрат пального залежать від пробігу та кількості техніки, а також від умов місцевості. Планування постачання пального потребує врахування відстаней, які долають підрозділи, та тривалості операцій.

- **Продовольство і вода.** Сюди входять харчові пайки, продовольство для польових кухонь, питна вода. Споживання продовольства доволі передбачуване (норматив на одного бійця на добу), проте в екстремальних умовах потреби можуть зростати. Вода критично необхідна на полі бою як для пиття, так і для санітарних потреб.

- **Матеріально-технічне майно та спорядження.** Це включає форму, засоби захисту, обмундирування, інженерне обладнання (лопати, мішки, будматеріали для укриттів), запасні частини до техніки, акумулятори, шини тощо. Споживання цих матеріалів залежить від інтенсивності експлуатації техніки та тривалості перебування в польових умовах.

- **Медичні засоби та обладнання.** Аптечки, медикаменти, перев'язувальні матеріали, медичне обладнання для польових госпіталів. Їх потреба зростає при інтенсивних боях та наявності втрат, тому прогнозування медичного забезпечення є частиною логістичного планування.

- **Спеціальні та інші категорії.** До них можна віднести, наприклад, матеріали для зв'язку (батареї, радіодеталі), паливно-мастильні матеріали спеціального призначення, засоби обігріву, а також інші класи постачання, прийняті в НАТО (усього розрізняють 10 класів постачання, що охоплюють всі види матеріальних засобів).

Логістична система працює за принципом безперервного циклу: **планування – постачання – споживання – поповнення**. На етапі планування визначаються потреби підрозділів на певний період чи операцію. Потім здійснюється постачання – доставка ресурсів від складів до підрозділів. У процесі бою або повсякденної діяльності відбувається споживання ресурсів (витрата боєприпасів, палива, продовольства тощо). Логістичні органи відслідковують залишки і своєчасно ініціюють поповнення запасів.

В умовах активних бойових дій логістика стикається з такими проблемами, як руйнування інфраструктури, ризик втрати вантажів через обстріли, необхідність маневрувати склади та маршрути постачання задля безпеки. Проте навіть у мирний час система повинна враховувати ротацію підрозділів, навчання, переміщення військ – усе це впливає на потреби в ресурсах.

1.3 Виклики забезпечення та логістичні ризики

Забезпечення військ належними ресурсами ускладнене низкою викликів, які слід враховувати при плануванні та прогнозуванні:

- **Нестабільність постачання та залежність від зовнішніх джерел.** У сучасному світі армії часто покладаються на імпортні комплектуючі, пальне або боєприпаси. Залежність від іноземних постачальників несе ризики для стабільності ланцюгів постачання. Геополітичні конфлікти

або санкції можуть призвести до раптового дефіциту критичних матеріалів. Отже, планування повинно враховувати альтернативні джерела та створення резервів.

- **Недостатня прозорість і контроль ланцюга постачання.** Один з головних викликів – неповна видимість руху матеріальних засобів від складу до фронту. Без наскрізного відстеження партій вантажів виникають затримки, втрачається частина вантажів або інформації, що може спричинити як надлишкові, так і недостатні поставки. Відсутність оперативної інформації про стан постачання ускладнює прийняття рішень на всіх рівнях.

- **Невизначеність потреб та коливання попиту.** Військові операції характеризуються різкими змінами інтенсивності боїв та відповідно споживання ресурсів. Наприклад, під час наступу витрати боєприпасів і пального стрімко зростають, тоді як у період перегрупування – падають. Такі коливання попиту ускладнюють прогнозування. Додатково, обмежені бюджети вимагають ефективного розподілу ресурсів: неможливо тримати необмежені запаси, отже потрібно точно знати потребу, щоб не було ні дефіциту, ні надлишків.

- **Кібербезпека та надійність систем.** Сучасна логістика залежить від інформаційних систем управління поставками, баз даних та засобів зв’язку. Це робить її вразливою до кібератак. У разі успішної атаки противник може паралізувати постачання або викрасти критичні дані. Таким чином, захист інформаційних систем та резервні плани на випадок відмов ІТ-систем є невід’ємною частиною логістичного планування.

- **Великі відстані та інфраструктурні обмеження.** Театр воєнних дій може охоплювати значні території, а транспортна інфраструктура (мости, дороги) може бути зруйнована. Пересування вантажів на великі відстані вимагає значних транспортних спроможностей і часу. Як зазначають військові логісти США, для театрів типу Тихоокеанського (де відстані можуть сягати 5000 миль) прогнозування і попереднє розгортання запасів набуває критичного значення. Необхідно враховувати "тиранію відстані": що далі знаходяться війська від баз постачання, то більше часу й ресурсів потребує доставка.

- **Протидія противника.** Ворог навмисно намагається порушити наші ланцюги постачання – обстрілює склади, атакує транспорт (колони постачання), проводить диверсії. Це створює ризик втрати вантажів і

потребує додаткових заходів безпеки, дублювання маршрутів, створення резервів на випадок втрат. Прогнозування має включати сценарії втрат і відповідні запаси для компенсації.

Перелічені виклики зумовлюють необхідність впровадження інноваційних підходів до управління постачанням. Зокрема, критично важливо забезпечити більшу **гнучкість та адаптивність** логістичної системи: можливість швидко перенаправляти ресурси, змінювати плани постачання в режимі реального часу за зміною обстановки. Для цього потрібні сучасні інформаційні інструменти з функціями моніторингу та прогнозування.

1.4 Особливості прогнозування потреб

Прогнозування потреб у військовій сфері має свою специфіку порівняно з цивільним сектором. По-перше, як зазначалося, **історичні дані** про споживання ресурсів не завжди є надійним орієнтиром для майбутніх операцій. Якщо в бізнесі попередній попит може повторюватися циклічно, то у військових діях кожна операція унікальна за характером. Наприклад, досвід боїв в лісистій місцевості мало застосовний для прогнозування потреб у пустелі. Тому прогнози мусять базуватися не лише на статистиці минулого, але й на **оперативних планах та сценаріях**.

При прогнозуванні логісти використовують нормативи споживання – середні показники витрат на одну особу чи одиницю техніки. Приміром, норматив продовольства – певна кількість кілокалорій на бійця в добу, боєприпасів – певна кількість на зброю за день бою, пального – літрів на 100 км пробігу для кожного виду машини. Ці нормативи стають відправною точкою. Далі вони коригуються з урахуванням конкретної обстановки:

- **Схема маневру і інтенсивність бойових дій.** Якщо плануються наступальні дії високої інтенсивності, очікується підвищена витрата боєприпасів та пального. Навпаки, оборонні дії на підготовлених рубежах можуть потребувати більше матеріалів для інженерного обладнання, але менше пального (якщо техніка менше рухається).

- **Час та тривалість операції.** Прогноз робиться на визначений період. Якщо операція розрахована на тиждень, логістика повинна забезпечити запас ресурсів щонайменше на цей термін із певним запасом. Якщо ж готується тривала кампанія – потрібне поступове нарощування поставок і створення тилових баз.

- **Кліматичні умови та місцевість.** Холодна погода підвищує потребу в

паливі (для обігріву) та калорійності харчування, пустельна місцевість – у воді і технічному обслуговуванні (через зношення техніки піском). Гірська місцевість обмежує вантажопідйомність транспорту, а значить потребує більшої кількості дрібних перевезень. Усе це впливає на витрати ресурсів і мусить враховуватись у прогнозах.

- **Стан техніки та особового складу.** Підрозділи з зношеною технікою можуть потребувати більше запасних частин і ремонтних матеріалів. Якщо особовий склад виснажений, можуть зрости потреби в медичних засобах, тонізуючих препаратах, висококалорійному харчуванні.

- **Взаємодія з союзниками.** У сучасних умовах військові операції часто проводяться коаліціями. Так, ЗСУ отримують значну частину боєприпасів і техніки від партнерів. Це означає, що прогнозування має враховувати постачання ззовні (наприклад, графіки поставок від союзників) і розподіл ресурсів між підрозділами коаліції. Також треба забезпечити **сумісність стандартів**: паливо різних марок, боєприпаси різних калібрів повинні відповідати техніці, що використовується.

Для підтримки прогнозування застосовуються спеціальні розрахункові методики та інструменти. Наприклад, для пального у бригаді може використовуватись формула: *потреба пального = кількість техніки × середня витрата (л/100 км) × відстань маршу (км)* – з корекцією на складність дороги. Для боєприпасів оцінюють передбачувану інтенсивність вогню (пострілів на добу) для кожної гармати чи стрільця і множать на їх кількість. Такі розрахунки можуть виконуватися вручну або в електронних таблицях, але це трудомісткий процес, схильний до помилок, особливо коли сценарій складний і включає багато змінних.

Як свідчить практика навчань, коли відсутній якісний аналіз потреб, підрозділи часто замовляють "наосліп" однакові обсяги постачання щодня, не враховуючи зміни обставин. На Національному тренувальному центрі (США) спостерігали, що бригадні тилові офіцери просто надсилають стандартний запит замість того, щоб вивчити оперативний план і спрогнозувати потреби залежно від майбутніх дій. Як наслідок, виникають ситуації, коли в одних місцях нагромаджуються надлишкові запаси, а в інших – виникає брак критичних ресурсів. Також це веде до зайвого навантаження на транспорт: доводиться возити назад невикористані вантажі, витрачаючи час і пальне. Нестача прогнозування призводить до того, що логістична система не встигає за різкою зміною обстановки. Отже, застосування науково обґрунтованих методів прогнозування є необхідною умовою успішного тилового забезпечення.

Висновком цього розділу є розуміння, що для ефективного управління військовою логістикою потрібно глибоке врахування структури військ та особливостей бойових дій. Прогнозування потреб має здійснюватися з використанням усіх доступних даних – як нормативів і історичної інформації, так і прогнозів розвитку ситуації – з метою мінімізації ризиків дефіциту або надлишку ресурсів.

Розділ 2. Аналіз досліджень з теми

2.1 Підходи до прогнозування потреб

Проблема прогнозування матеріальних потреб у військовій сфері привертала увагу дослідників протягом десятиліть, особливо в галузі військово-логістичного планування. Класичні підходи базувалися на статистичних методах та експертних оцінках. Зазвичай використовувався метод "нормативів та коефіцієнтів": для кожного типу підрозділу й ситуації встановлювалися норми витрат (напр. боєприпасів на день бою, пального на 100 км маршу тощо) на підставі аналізу минулих воєн і навчань. Потім планувальники коригували ці норми на очікувані умови (підвищували або знижували). Такий підхід застосовувався ще з часів Другої світової війни і залишається основою для попередніх оцінок.

В сучасну епоху, з появою великих обсягів даних і обчислювальних потужностей, з’явилися нові підходи:

- **Статистичне моделювання та прогнозні моделі.** Використання математичної статистики для обробки даних про витрати ресурсів. Наприклад, моделі часових рядів (ARIMA, експоненційне згладжування) можуть застосовуватися для прогнозування споживання окремих видів матеріалів на основі трендів. Однак у військовій сфері часові ряди часто нерегулярні через зміну фаз операцій.

- **Імітаційне моделювання сценаріїв.** Планувальники можуть використовувати системи, що моделюють перебіг операції (включно з логістикою). На основі сценарних параметрів (кількість військ, інтенсивність боїв, тривалість) така модель “програє” постачання і видає прогноз, скільки ресурсів буде витрачено. Існують як комплексні військові моделі (наприклад, моделювання бойових дій з урахуванням витрати матеріалів), так і спеціалізовані логістичні симулятори.

- **Оптимізаційні моделі.** Застосування методів дослідження операцій, де прогнозування потреб є частиною задачі оптимального забезпечення. Наприклад, розв’язуються задачі типу “динамічний рюкзак” чи

"проблема транспорту" з урахуванням прогнозних витрат: оптимальний розподіл обмежених ресурсів між запитами підрозділів так, щоб максимізувати бойову готовність.

- **Експертні системи і правила.** У 1990-2000-х роках набули поширення експертні системи – програмні комплекси з базою знань, що містить правила типу "Якщо ... то ...". У контексті прогнозування потреб такі системи могли містити знання досвідчених логістів: наприклад, "якщо температура нижче -10°C, збільшити потребу в паливі на 20%" тощо.

- **Машинне навчання та штучний інтелект.** Сучасний тренд – застосування алгоритмів ML, які можуть навчатися на великих масивах даних (зокрема історичних записах про витрати ресурсів у різних операціях) і виявляти приховані закономірності. Наприклад, нейронні мережі або алгоритми типу Random Forest можуть будувати прогноз споживання на підставі множини факторів (погодні умови, тип місцевості, склад сил, інтенсивність операції тощо). Такі моделі потребують значного обсягу достовірних даних для навчання, що не завжди доступно у військовій сфері через секретність та унікальність умов. Проте у комерційній логістиці вони демонструють хороші результати, тому їх адаптація до військових needs – питання часу.

Варто зазначити, що в реальних умовах застосовується **комбінація методів**. Первинний прогноз може робитися за нормативами (для швидкості), потім уточнюватися за допомогою моделювання сценаріїв або з використанням даних розвідки про противника. Високий ступінь невизначеності змушує планувати з певним запасом (safety stock), тобто навмисно завищувати потреби на випадок непередбачуваного розвитку подій.

2.2 Системи підтримки прийняття рішень у логістиці

Системи підтримки прийняття рішень (СППР) у військовій логістиці – це програмні інструменти, що допомагають офіцерам-тиловикам в плануванні та управлінні постачанням. Вони можуть включати бази даних, модулі аналізу та прогнозування, інтерфейси для введення сценарних даних і виводу рекомендацій.

Історично, військові планувальники використовували в основному офлайн-інструменти – таблиці постачання, друковані керівництва з нормами тощо. З переходом на цифрові технології армії почали запроваджувати автоматизовані системи управління логістикою (логістичні інформаційні системи). Деякі приклади:

- **LOGFAS (NATO Logistics Functional Area Services).** Це комплекс програмних модулів, прийнятий в НАТО для планування і координації багатонаціональної логістики. Він дозволяє планувати перевезення, відстежувати переміщення військ і вантажів, вести облік запасів. Україна станом на 2023 рік впроваджує систему LOGFAS у своїх Збройних Силах на рівні бригад для покращення обліку й координації постачання. Ця система надає командуванню можливість бачити загальну картину логістики і оперативно приймати рішення щодо розподілу ресурсів.

- **Національні логістичні системи.** Наприклад, США мають систему **GCSS-Army (Global Combat Support System-Army)** – це ERP-система, що об'єднує облік матеріальних засобів, заявок на постачання, управління складами та ремонтами в єдиному інформаційному просторі. Подібні системи існують і в інших країнах, інтегруючи різні аспекти тилового забезпечення.

- **Спеціалізовані прогностичні інструменти.** Окрім великих ERP-систем, військові можуть використовувати окремі модулі або програми для розрахунку потреб. Наприклад, програму для розрахунку витрат пального на марш (враховує кількість техніки і відстань), чи калькулятор боєприпасів для артилерійської батареї залежно від вогневої задачі. Такі інструменти можуть бути окремими додатками чи входити в складніші системи.

- **DSS для логістики з елементами оптимізації.** Існують дослідні розробки, які пропонують оптимізаційні поради: скажімо, якщо ресурсів недостатньо для покриття всіх заявок, система може рекомендувати, які підрозділи забезпечити в першу чергу, виходячи з пріоритетів операції. Також DSS може генерувати маршрути постачання, оптимізовані за часом або мінімальною загрозою.

- **Інтерактивні тренажери та системи підтримки навчань.** Для навчання тилових органів застосовуються системи, які в режимі реального часу моделюють логістичну обстановку на навчаннях і дають учасникам можливість відпрацьовувати прийняття рішень. Ці системи також виконують роль DSS, пропонуючи варіанти дій або оцінюючи рішення офіцерів.

Загалом, впровадження СППР у логістиці спрямоване на те, щоб знизити навантаження на людей у процесі аналізу великих обсягів інформації та

підвищити обґрунтованість рішень. Проте якісна робота таких систем залежить від **достовірності та повноти даних**: якщо дані про залишки, витрати або поставки неактуальні, то і рекомендації системи будуть хибними.

2.3 Інформаційні технології у військовій логістиці

Останніми роками військова логістика зазнає суттєвих змін під впливом цифрових технологій. До ключових напрямів впровадження IT-рішень належать:

- **Діджиталізація обліку та складських операцій.** Перехід від паперових документів до електронних систем обліку забезпечує швидший обмін даними та кращий контроль. Наприклад, використання штрих-кодів або RFID-міток на ящиках із боєприпасами дозволяє швидко сканувати вантажі при відправці та отриманні, знижуючи кількість помилок і втрат.

- **Геоінформаційні системи (GIS) та відстеження в реальному часі.** Сучасна логістика впроваджує трекери і датчики для моніторингу місцеположення конвоїв, стану техніки (температура, вібрація) тощо. Інтеграція цих даних у GIS дає логістам можливість бачити на електронній карті, де знаходяться ресурси і коли вони прибудуть. Це підвищує прозорість ланцюга постачання та дозволяє швидше реагувати на відхилення.

- **Big Data та аналітика.** Військові накопичують величезні обсяги даних: про споживання матеріалів у різних умовах, про надійність техніки, про часові затримки в доставці тощо. Аналіз цих "великих даних" вручну неможливий, тому застосовуються платформи для обробки даних та генерації аналітичних звітів. Наприклад, в США Пентагон запроваджує практики прогнозної аналітики для планування логістики Одна з ініціатив – створення системи, яка агрегує дані з різних джерел (склади, датчики, плани операцій) і видає прогноз потреб на декілька тижнів уперед.

- **Штучний інтелект (AI) та машинне навчання.** В оборонній логістиці почали з’являтися AI-рішення. Наприклад, **Defense Logistics Agency (DLA)** – головна логістична агенція Міноборони США – вже використовує понад 55 моделей штучного інтелекту на різних етапах, включаючи планування попиту та управління ризиками у постачанні. Створено центр передового досвіду з AI, що координує інтеграцію таких технологій. Мета – побудувати єдину екосистему, де AI допомагає виявляти закономірності та підказувати рішення (наприклад,

передбачати можливий дефіцит певної деталі через 2 місяці і вчасно ініціювати замовлення).

- **Хмарні технології та інтеграція даних.** Союзницькі операції вимагають обміну логістичними даними між різними країнами та відомствами. Для цього створюються захищені платформи (часто на базі хмарних рішень), що дозволяють стикувати різні системи. Приклад – інтеграція національних систем США з НАТО-стандартом LOGFAS, щоб автоматично обмінюватися даними про переміщення і запаси. Україна також створила власні програми ("Коровай" та модуль на базі системи SOTA) для взаємодії із західними донорами щодо постачання озброєнь, що забезпечує прозорість і довіру між партнерами.

- **Спеціалізовані платформи для прогнозування.** Наприклад, DLA Energy (агенція, що відповідає за пальне) розробила платформу **PLUTO** для прогнозування та оптимізації постачання палива. Ця система забезпечує прозорість даних про споживання палива в глобальній мережі, моделює ризики та дозволяє планувати запаси палива проактивно, до того як виникне дефіцит.

Таким чином, IT проникають у всі аспекти військової логістики – від складу до штабу. Вони надають інструментарій для переходу від реактивного реагування (коли проблеми вирішуються після того, як сталися) до **прогностичного управління**, коли проблемі запобігають завдяки завчасно отриманому прогнозу. Особливо це стало очевидним під час війни в Україні, де західні технології аналітики даних допомагають нашому тилу ефективніше розподіляти ресурси, а російська армія, покладаючись на застарілі методи, зазнає збоїв у постачанні.

2.4 Застосування ML та аналітики в обороні

Методи машинного навчання (ML) і штучного інтелекту дедалі активніше застосовуються в оборонному секторі, і логістика – одна з ключових сфер, де вони можуть принести значні вигоди. Приклади застосувань:

- **Прогнозування відмов техніки (predictive maintenance).** ML-моделі аналізують дані датчиків і журнали технічного обслуговування, щоб передбачити, коли військова техніка вийде з ладу. Це дозволяє заздалегідь замовити і доставити потрібні запчастини. Вже згаданий комплекс **AMC Predictive Analytics Suite** в армії США аналізує дані про напрацювання вузлів машин, умови експлуатації тощо, щоб передбачити відмови і спланувати ремонт до поломки. Таким чином,

логістика стає більш проактивною – деталь надходить ще до того, як техніка зламалась.

- **Аналіз споживання боєприпасів.** На основі даних про інтенсивність боїв ML може виявляти закономірності: наприклад, що при певній тактиці противника наші підрозділи витрачають більше боєприпасів певного типу. Знаючи це, командування може змінити пріоритети постачання або тактику застосування військ. В майбутньому можливо створення моделей, які майже в реальному часі будуть коригувати прогноз витрати боєприпасів, підлаштовуючись під стиль бою поточного противника.

- **Оптимізація маршрутів постачання з урахуванням ризиків.** Алгоритми можуть оцінювати різні маршрути транспортування вантажів (протяжність, наявність мостів, загроза обстрілу) і рекомендувати найбезпечніший або найшвидший шлях. Також вони можуть динамічно переназначати маршрути, якщо надходить інформація про небезпеку (наприклад, розвідка повідомила про засідку).

- **Розподіл ресурсів за пріоритетністю.** Коли ресурсу на всіх не вистачає, постає задача вирішити, кому дати в першу чергу. ML може допомогти, оцінивши бойову обстановку: які підрозділи ведуть найважчі бої, де очікується наступ, де потрібна підтримка. На основі цього система може підказати, як розподілити доступні ресурси оптимально для успіху операції.

- **Виявлення аномалій та ризиків.** В логістичних даних алгоритми можуть автоматично помічати аномалії – наприклад, різке зниження запасів на складі, яке не було заплановане (можливий сигнал про крадіжку чи про помилку в обліку), або незвично довгий час у дорозі для певного транспорту (можливі проблеми на маршруті). Таким чином, командири отримують сигнали про потенційні проблеми раніше, ніж вони переростуть у кризу.

- **Навчання і підтримка рішень.** На основі накопичених даних про ухвалені рішення в минулих операціях (що спрацювало, а що ні) ML-моделі можуть навчитися пропонувати рішення, подібні до дій досвідчених командирів тилу. Це свого роду "цифровий консультант": молодий офіцер може отримати підказку від системи, яка зважила багато факторів.

В усіх цих випадках слід пам’ятати, що ML – це інструмент допомоги, а не заміна командирського рішення. Особливістю військової сфери є вимоги безпеки і надійності. Алгоритм повинен бути прозорим і пояснюваним, щоб офіцери довіряли рекомендаціям. Також дані для навчання повинні бути захищені та представницькі для реальних ситуацій, інакше модель може видати хибні прогнози.

На сьогодні багато проектів знаходяться на стадії випробувань. Проте загальна тенденція зрозуміла: **оборонна логістика стає “керованою даними” (data-driven)**. Країни-лідери інвестують в аналітику та ML як у ключ до підвищення стійкості своїх збройних сил. Застосування цих технологій дає змогу компенсувати навіть більші матеріальні ресурси противника за рахунок більш розумного, ефективного їх використання. Україна, отримуючи сучасні засоби від партнерів, також опановує і ці підходи, що є важливою складовою зміцнення обороноздатності.

Розділ 3. Вибір технологічного стеку

3.1 Вимоги до технологій

Важливим етапом розробки веб-застосунку прогнозування логістичних потреб є формування чітких вимог до технологічного стеку. З огляду на специфіку проекту необхідно забезпечити надійне зберігання та обробку даних, можливість побудови інтуїтивного інтерфейсу з динамічними графіками, а також інтеграцію з моделями машинного навчання. Стек має відповідати таким ключовим вимогам:

- **Робота з реляційною базою даних (MySQL):** обрана СКБД MySQL слугує основним сховищем даних про логістичні показники. Фреймворк повинен мати стабільну ORM-абстракцію для зручної роботи з MySQL (виконання запитів, міграції, зв’язки між таблицями) та підтримувати транзакції і ефективні індекси для масштабування.

- **Побудова RESTful API:** веб-застосунок повинен надавати програмний інтерфейс для взаємодії з фронтендом або іншими сервісами (наприклад, для отримання результатів моделі прогнозування). Тому обрана технологія має мати вбудовані механізми маршрутизації REST, контролери для обробки HTTP-запитів, форматування даних у JSON і засоби автентифікації/авторизації (API-токени).

- **Інтерактивна візуалізація даних (Chart.js):** для аналізу прогнозів логістичних потреб необхідні наочні графіки. Стек повинен забезпечувати безперешкодну інтеграцію зі сторонніми JavaScript-бібліотеками візуалізації, зокрема з Chart.js. Це вимагає наявності

потужного шаблонізатора на стороні сервера та можливості передавати дані в HTML-сторінки для відображення графіків на клієнті.

- **Безпека і контроль доступу:** оскільки застосунок оперує важливими даними військового призначення, фреймворк має пропонувати вбудовані засоби захисту від поширених вразливостей вебу – CSRF, XSS, SQL-ін'єкцій. Необхідні також готові рішення для реєстрації, автентифікації та управління ролями користувачів.

- **Продуктивність і масштабованість:** система повинна ефективно працювати з великими обсягами даних і витримувати навантаження одночасних користувачів. Вимоги включають підтримку кешування даних, можливість розподілу навантаження (кластеризація сервера, мікросервісна архітектура), а також оптимізоване виконання серверного коду.

- **Інтеграція з моделями ML (Python):** для реалізації прогнозування планується використання алгоритмів машинного навчання на Python. Технологічний стек має надати можливість легко інтегрувати Python-моделі: наприклад, через виклик зовнішніх скриптів чи сервісів, обмін даними у форматі JSON, а також підтримувати обробку великих обчислювальних задач (чергові завдання, багатопотоковість).

- **Зручність розробки і підтримки:** обрані інструменти мають забезпечити високу швидкість розробки та зручність підтримки коду. Важливі наявність добре структурованої архітектури (MVC-поділ), простота налаштування, наявність документації та активної спільноти розробників. Ідеально, якщо фреймворки є open-source і не мають ліцензійних обмежень.

Таким чином, технологічний стек повинен поєднувати функціональність для роботи з даними, засоби побудови веб-інтерфейсу з графіками, компоненти безпеки та інтеграцію з Python, забезпечуючи при цьому високу продуктивність і надійність.

3.2 Порівняння фреймворків

Для вибору оптимального серверного фреймворку проведено порівняння Laravel, Symfony, Node.js (з урахуванням популярних доповнень, наприклад Express) та Django за ключовими параметрами.

Таблиця 3.1 Основні характеристики кожного фреймворку:

Критерій	Laravel	Symfony	Node.js (Express)	Django
Мова	PHP (версія 7–8+)	PHP (7–8+)	JavaScript (Node.js)	Python
Архітектурний стиль	MVC (Model-View-Controller)	MVC (компонентний підхід)	Подієво-орієнтована, неблокуюча	MTV (Model-Template-View), близько до MVC
Шаблонізатор	Blade (функціональний шаблонізатор)	Twig (гнучкий шаблонізатор)	Залежить від вибору (наприклад, Pug/EJS або SPA-фреймворки)	Django templates (система шаблонів)
ORM / БД	Eloquent ORM (підтримка MySQL)	Doctrine ORM (підтримка SQL)	Власного немає; використовуються пакети (Sequelize, Mongoose тощо)	Вбудований ORM (реляційні БД)
Продуктивність	Висока при оптимізації (PHP 8+, опкешування)	Висока при масштабуванні	Дуже висока при обробці одночасних запитів (асинхронна модель)	Висока для типових веб-завдань (залежить від налаштування сервера)
Масштабованість	Підтримує великі системи (черги, кешування)	Дуже висока (портфоліо великих проектів)	Широкі можливості (кластеризація, мікросервіси)	Висока (командний додаток можна масштабувати горизонтально)

Безпека	Вбудований захист CSRF, XSS, SQL-ін'єкцій; готові middleware	Високий рівень захисту (Security-компонент Symfony)	Залежить від реалізації; потрібно використовувати додаткові бібліотеки (Helmet, express-validator)	Вбудований захист CSRF, XSS, SQL-ін'єкцій; гнучка система аутентифікації
Екосистема та спільнота	Широка екосистема пакетів (Composer); велика спільнота	Розвинена екосистема компонентів; корпоративні проекти	Дуже велика (npm); багато бібліотек для різних завдань	Широка екосистема Python-бібліотек; спільнота, орієнтована на веб і наукові застосунки
Інтеграція з ML	Через зовнішні сервіси/скрипти (через REST/CLI)	Так само, як і в Laravel, через API	Є JS-бібліотеки (TensorFlow.js), або через зовнішні сервіси	Пряма інтеграція з Python-бібліотеками ML (pandas, scikit-learn)

Наведену таблицю доповнюють такі ключові спостереження:

- **Laravel (PHP):** фреймворк «з коробки» надає багато інструментів для веб-розробки (Eloquent ORM, шаблонізатор Blade, маршрутизацію, готові рішення для аутентифікації, обробки черг тощо). Він підходить для швидкої реалізації CRUD-операцій і REST API. Великий плюс – зрозуміла структура проекту і активна спільнота, але застаріла слава PHP може сприйматися критично. При цьому сучасні версії PHP (8+) пропонують високу продуктивність та оптимізацію коду. Laravel вибивається як «золота середина» між простотою використання і гнучкістю, особливо придатний для типових веб-застосунків і прототипування.

- **Symfony (PHP):** більш «важковаговий» фреймворк з компонентним підходом. Він дає величезну свободу налаштування і високий рівень контролю над кожним шаром програми. Завдяки цьому Symfony добре підходить для великих корпоративних систем із суворими

архітектурними вимогами. Однак розробка вимагає більше зусиль і часу (більше конфігурацій), що може уповільнювати швидкість прототипування. Symfony часто обирають для проектів, де потрібно глибоке доопрацювання і масштабування, але для стартапних або невеликих застосунків це може бути надмірно.

- **Node.js (Express):** серверне середовище на базі JavaScript із подієво-орієнтованою моделлю, що забезпечує високопродуктивну роботу з великою кількістю одночасних з'єднань. Підходить для реальних часовых застосунків і мікросервісів завдяки неблокуючому вводу-виводу. Відсутність стандартного фреймворку означає, що потрібно самостійно зібрати необхідні модулі (Express, ORM, безпека тощо). Мова JavaScript дозволяє використовувати ті ж технології на клієнті та сервері. Для створення REST API підійде Express або фреймворки вищого рівня (Nest.js). Головні недоліки – відсутність єдиної стуктури проекту за замовчуванням і необхідність ретельно підбирати та налаштовувати бібліотеки для аутентифікації та безпеки.

- **Django (Python):** «батарейки в комплекті» фреймворк, який забезпечує швидкий старт: вбудовані ORM, адміністративна панель, шаблонізатор і готові засоби аутентифікації. Python широко використовується в науці про дані та машинному навчанні, тож Django природно добре інтегрується з ML-модулями. Це робить його сильним кандидатом для проектів з активними аналітичними компонентами. Django забезпечує високу захищеність за замовчуванням, але може вимагати оптимізації продуктивності (наприклад, налагодження кешування) при великих навантаженнях. В цілому Django добре підходить для веб-додатків із великим обсягом обчислень на боці сервера та коли важлива швидка розробка за рахунок «готових рішень».

Кожен із розглянутих фреймворків має свої переваги і обмеження. Laravel і Symfony – обидва на PHP, але Laravel дає швидший старт і простіше освоюється, тоді як Symfony орієнтований на масштабовані рішення. Node.js забезпечує найкращу продуктивність при високій конкуренції та є гнучким, але потребує більше ручного налаштування. Django пропонує неперевершені можливості для інтеграції з Python-екосистемою ML, але може поступатися в швидкості чистого CRUD-процесу. Отже, вибір технологій залежить від балансу між функціональними вимогами, ресурсами команди та майбутніми планами масштабування.

3.3 Обґрунтування вибору Laravel та суміжних технологій

Після аналізу вимог (п. 3.1) та порівняння фреймворків (п. 3.2) обґрунтування

вибору припадає на фреймворк **Laravel** разом із суміжними технологіями (MySQL, Blade, Chart.js, REST API та Python). Основні причини такого вибору:

- **Структурована MVC-архітектура і REST API:** Laravel реалізує архітектуру MVC, чітко розділяючи модель, представлення та контролер. Це забезпечує прозору організацію коду і простоту підтримки. У поєднанні з можливістю визначати ресурсні маршрути, контролери і API-ресурси Laravel дозволяє легко створити RESTful API для обміну даними між сервером та клієнтом. Завдяки цьому проекту забезпечується чиста взаємодія між фронтендом і бекендом.

- **Робота з базою даних MySQL:** Laravel має вбудований **Eloquent ORM**, що забезпечує зручний та декларативний доступ до MySQL. ORM спрощує написання запитів, визначення зв'язків між таблицями (наприклад, «один-до-багатьох», «багато-до-багатьох»), а також керування міграціями (версіями схеми бази). Для логістичних даних, які зберігаються в реляційному форматі, це важливо – розробники можуть оперувати моделями даних на високому рівні, не турбуючись про складні SQL-запити. При цьому Laravel забезпечує оптимізовану взаємодію з MySQL (автозавантаження запитів, кешування) для підтримки продуктивності.

- **Шаблонізатор Blade і інтеграція Chart.js:** для формування інтерфейсу застосунку обрано серверний шаблонізатор **Blade**, який дозволяє впроваджувати динамічний HTML з простим PHP-кодом. Blade спрощує передачу даних від контролерів у вигляді змінних безпосередньо у вигляди, де їх можна відобразити. Це дає змогу легко вбудовувати у сторінки сторонні JavaScript-бібліотеки. Зокрема, **Chart.js** – популярна бібліотека для побудови інтерактивних графіків – інтегрується в Blade-шаблони шляхом імпорту скрипту і передачі в нього даних з контролера. Така комбінація дозволяє будувати наочні графіки (діаграми витрат, прогнозів тощо) без складних налаштувань.

- **Безпека веб-застосунку:** Laravel містить вбудовані механізми захисту: автоматичне генерування CSRF-токенів у формах, екранування виводу для запобігання XSS, параметризовані запити для захисту від SQL-ін'єкцій. Крім того, фреймворк пропонує готові рішення для аутентифікації й авторизації (наприклад, пакети Laravel Jetstream або Breeze), що значно полегшує реалізацію системи входу користувачів і ролей доступу. Таким чином, усі вимоги до безпеки даних і контролю доступу можуть бути задоволені стандартним функціоналом Laravel або легко доповненими пакетами.

- **Інтеграція з Python і ML-сервісами:** хоча Laravel базується на PHP,

передбачені механізми для взаємодії з Python-модулями. Зазвичай це реалізується через виклик зовнішнього REST API, створеного на Flask або Django, або через запуск скриптів Python з командного рядка. Завдяки чітко визначеній REST-архітектурі Laravel може надсилати дані до ML-сервісу та отримувати результати прогнозів у форматі JSON. Крім того, в Laravel є система завдань (queues), яка може запускати фонові процеси (наприклад, обчислення статистики або оновлення моделі). Це відповідає вимозі виконувати важкі обчислення на Python і передавати результати у веб-додаток.

- **Швидкість розробки та підтримка:** Laravel відомий своєю простотою освоєння і зрозумілою документацією. Вбудований CLI-інструмент Artisan дозволяє створювати шаблони класів, міграції бази, контролери та інші компоненти за допомогою команд. Є велика кількість готових пакетів (через Composer) для типових завдань: управління користувачами, відправлення пошти, інтернаціоналізація тощо. Активна спільнота та популярність фреймворку гарантують, що на будь-яке питання можна знайти рішення або приклади. Всі ці фактори пришвидшують розробку та забезпечують легкість подальшої підтримки і розширення застосунку.

Таким чином, комбінація Laravel, MySQL, Blade, Chart.js і REST API оптимально відповідає сформованим вимогам. Laravel забезпечує необхідну архітектуру і безпечну роботу з даними, Blade дозволяє формувати адаптивний інтерфейс, а Chart.js – будувати наочну візуалізацію прогнозів. Підтримка REST API і можливості інтеграції з Python-модулями дозволяють під’єднати необхідні алгоритми машинного навчання. Завдяки відкритій ліцензії та широкій спільноті такий стек є економічно вигідним і технологічно надійним для реалізації проекту веб-застосунку в логістиці.

3.4 Структура взаємодії компонентів застосунку

Розроблений веб-застосунок прогнозування логістичних потреб базується на класичній клієнт-серверній архітектурі. Користувацький інтерфейс (фронтенд) забезпечує взаємодію користувача із системою та надсилає запити до серверної частини. Серверна частина застосунку на основі Laravel обробляє вхідні запити, виконує бізнес-логіку та формує звернення до бази даних для збереження або отримання необхідних даних, включно з історичною інформацією та результатами прогнозів. За потреби бекенд звертається до зовнішнього Python-сервісу, який реалізує алгоритми прогнозування та повертає результати для подальшої обробки. Уся структура взаємодії компонентів системи наведена на рисунку 3.1.

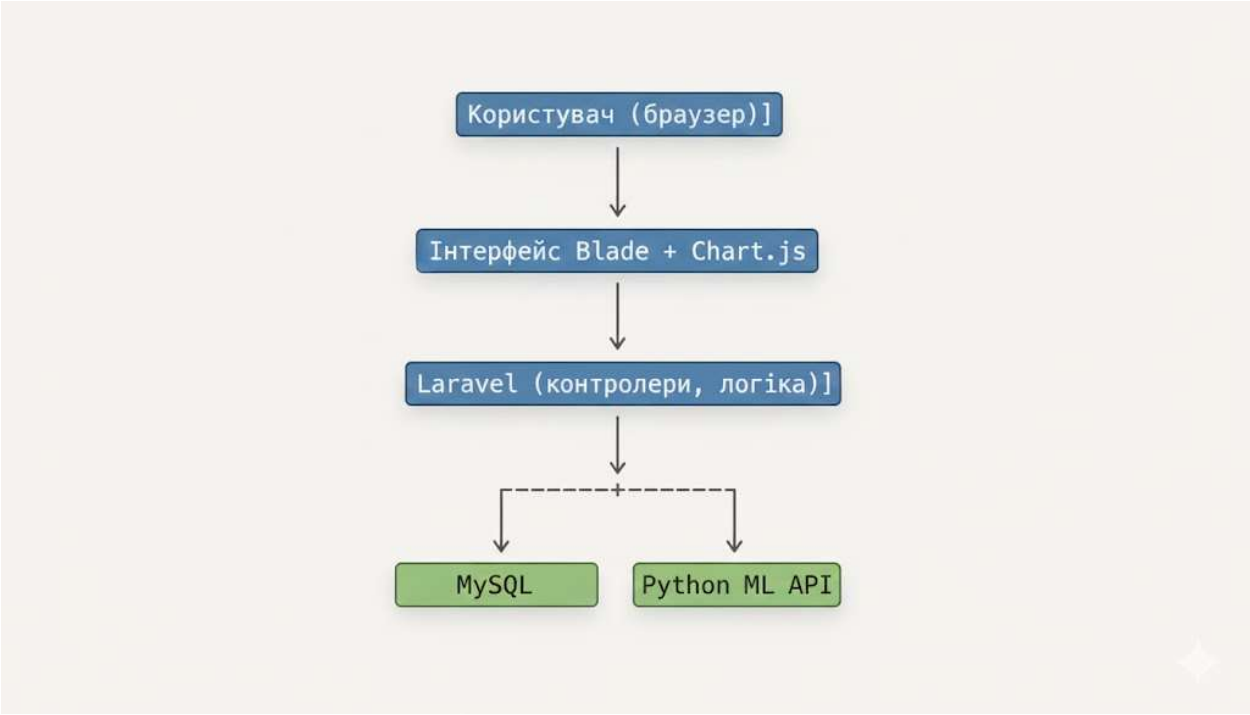

Рисунок 3.1 — Архітектура системи прогнозування.

3.5 Інструменти супроводу та автоматизації

Для ефективної розробки та супроводу Laravel-проекту застосовується набір інструментів, призначених для автоматизації рутинних задач. Ключовими компонентами інструментарію проекту є командний інтерфейс (CLI), менеджер пакетів Composer, консоль Artisan, механізм черг (Queue), HTTP-клієнт Guzzle та система міграцій (Migrations). Кожен з цих інструментів виконує свою роль у процесі розробки, про що йдеться далі.

- **CLI (Command Line Interface)** забезпечує взаємодію з проектом через командний рядок операційної системи. Через консольні команди розробник може запускати скрипти, керувати процесами проекту та виконувати інші утиліти, що автоматизують розробку та адміністрування.

- **Composer** — менеджер пакетів PHP, що відповідає за встановлення залежностей проекту та автоматичне завантаження класів. Цей інструмент спрощує додавання та оновлення сторонніх бібліотек, забезпечує контроль версій і підтримує узгоджене середовище розробки.

- **Artisan** — вбудована консольна оболонка Laravel, яка надає набір команд для генерації коду (створення моделей, контролерів тощо), запуску міграцій, управління чергами задач та виконання інших стандартних операцій. Завдяки Artisan розробка прискорюється за рахунок автоматизації повторюваних дій.

- **Queue (система черг)** використовується для асинхронного виконання довготривалих задач у фоновому режимі. Завдання, що потребують значних ресурсів (наприклад, обчислення прогнозів або надсилання великих обсягів даних), ставляться в чергу та обробляються окремими процесами, що покращує масштабованість системи та відчуття швидкодії інтерфейсу.

- **Guzzle** — HTTP-клієнт для PHP, який дозволяє відправляти HTTP-запити до зовнішніх сервісів. За допомогою Guzzle бекенд-застосунок може здійснювати звернення до Python-сервісу прогнозування або інших API, отримуючи дані чи результати обробки. Цей інструмент спрощує інтеграцію із зовнішніми мікросервісами та надає зручні засоби обробки відповідей і помилок.

- **Migrations (міграції)** — система керування версіями схеми бази даних у Laravel. Міграції дозволяють у коді визначати структуру таблиць та індексів, застосовувати зміни до схеми БД і відкотити їх за потреби. Вони забезпечують контроль історії змін бази даних, полегшують командну розробку та гарантують узгодженість структури даних.

Таблиця 3.2 — Характеристика інструментів автоматизації в Laravel-проекты

Інструмент	**Призначення**	**Переваги**
CLI	запуск командних утиліт через командний рядок ОС	автоматизація рутинних задач, підвищення ефективності розробки
Composer	керування залежностями PHP-проекту та автозавантаження класів	спрощення встановлення і оновлення пакетів, контроль версій бібліотек
Artisan	консоль Laravel із командами для генерації коду, запуску міграцій та інших задач	автоматизація створення типових компонентів, прискорення розробки

Queue	асинхронне виконання відкладених фонових задач	підвищення продуктивності та масштабованості, безперервна робота інтерфейсу
Guzzle	HTTP-клієнт для відправки запитів до зовнішніх API	спрощена інтеграція з зовнішніми сервісами, гнучкі можливості обробки запитів
Migrations	версіонування та застосування змін структури бази даних	контроль історії змін БД, можливість відкату оновлень, узгодженість структури серед розробників

Розділ 4. Визначення вимог до системи

На основі проведеного аналізу предметної області та існуючих підходів можна сформулювати вимоги до програмної системи підтримки прийняття рішень для прогнозування логістичних потреб. Система буде призначена для використання офіцерами логістики (тилу) бригади або аналогічного підрозділу з метою планування забезпечення операцій. Розглянемо категорії вимог.

4.1 Функціональні вимоги

Функціональні вимоги описують, що саме система повинна робити – тобто її функції та можливості з точки боку користувача і зовнішніх систем. Основні функціональні можливості розроблюваної системи визначені так:

Таблиця 4.1. Функціональні вимоги

ID	**Функціональна вимога**	**Опис**
FR1	Введення даних про підрозділи та ресурси	Система повинна надавати інтерфейс для введення або імпорту даних про структуру

		підрозділу (чисельність, типи техніки), а також поточні запаси матеріальних засобів і дані про споживання.
FR2	Встановлення параметрів сценарію операції	Система дозволяє користувачу задавати параметри майбутньої операції: тривалість, тип (оборона, наступ), сезон/погодні умови, особливі умови місцевості, очікувана інтенсивність боїв тощо.
FR3	Розрахунок прогнозованих потреб по категоріях	На основі вхідних даних (структури сил, сценарію) система автоматично розраховує потребу в ключових ресурсах (боєприпаси, пальне, продовольство тощо) на заданий період. Розрахунок враховує нормативи споживання та коригувальні коефіцієнти.
FR4	Оновлення прогнозу в реальному часі	Під час виконання операції система повинна мати можливість оновлювати прогноз на основі фактичних даних про витрати і змін обстановки. Користувач може вводити фактичні витрати за день, після чого прогноз на наступні періоди автоматично уточнюється.
FR5	Виявлення дефіцитів та критичних ресурсів	Система аналізує результати прогнозу та виявляє позиції, де прогнозована потреба перевищує наявні запаси або встановлені ліміти. Повинно генеруватися попередження про можливий дефіцит певних ресурсів.

FR6	Рекомендації щодо розподілу ресурсів	На основі пріоритетів підрозділів та важливості ресурсів система надає рекомендації: наприклад, якщо ресурс обмежений, запропонувати, яким ротам/батальйонам розподілити його в першу чергу. Також можуть надаватися рекомендації щодо необхідності поповнення запасів з вищих складів.
FR7	Генерація звітів та візуалізація	Система повинна формувати зрозумілі звіти: текстові і графічні (діаграми витрат, графіки залишків тощо) для представлення прогнозованих потреб. Наприклад, графік потреби пального по днях операції або діаграма складу вантажів для перевезення.
FR8	Експорт та обмін даними	Система надає можливості експорту даних прогнозу і рекомендацій у стандартних форматах (Excel, PDF), а також імпорту даних з інших систем (напр. імпорт фактичних залишків зі складу). Передбачена можливість обміну даними з коаліційними системами (через API, див. вимоги до API).
FR9	Управління користувачами та правами доступу	Система має підтримувати декілька ролей користувачів (наприклад, планувальник бригади, начальник штабу, оператор введення даних). Залежно від ролі, обмежувати доступ до певних функцій (перегляд/редагування даних,

		затвердження прогнозів тощо).
FR10	Журнал подій і рішень	Всі ключові дії користувачів (введення даних, зміна параметрів, затвердження розподілу) та генеровані системою попередження/рекомендації повинні протоколюватися. Це дозволить відслідковувати хід планування та в разі потреби проаналізувати, які рішення були прийняті.

4.2 Нефункціональні вимоги

Нефункціональні вимоги задають критерії якості, обмеження та характеристики, яким має відповідати система:

- **Надійність.** Система повинна працювати стійко навіть у польових умовах. Збій системи під час операції є неприпустимим, тому необхідно забезпечити механізми резервування даних (автоматичне збереження введених даних, резервне копіювання бази даних). Середній час безвідмовної роботи має бути якомога вищим; у випадку збою відновлення роботи повинно займати мінімум часу.

- **Безпека.** Дані про логістичні потреби та запаси є чутливими (можуть бути таємними), тому система має забезпечувати шифрування даних як у сховищі, так і при передаванні. Аутентифікація користувачів – обов’язкова, з використанням надійних паролів або двофакторної автентифікації. Повинна бути реалізована система розмежування доступу до особливо важливих даних (наприклад, інформація про запаси боєприпасів доступна лише обмеженому колу осіб).

- **Продуктивність.** Система повинна обробляти розрахунки прогнозу швидко. Наприклад, після введення вихідних даних розрахунок потреб на місяць по бригаді (кілька сотень одиниць техніки, тисячі людей) має виконуватися за секунди, максимум десятки секунд. Інтерфейсні операції (перегляд, фільтрація даних) мають відбуватися майже миттєво.

Це критично, оскільки у військових рішення часто треба приймати оперативно.

- **Масштабованість.** Система повинна підтримувати масштабування: як вгору (на випадок використання на рівні більшого з'єднання, наприклад дивізії – тобто більше даних), так і вшир (багато паралельних користувачів чи розгорнення в багатьох бригадах). Архітектура має дозволяти додавання нових модулів, розширення функцій без повної переробки системи.

- **Зручність користування (Usability).** Цільові користувачі – офіцери логістики, які можуть не мати глибоких IT-навичок. Інтерфейс повинен бути інтуїтивно зрозумілим, бажано рідною мовою (українською), з використанням знайомої термінології. Система повинна працювати на типових пристроях (ноутбук, планшет) без потреби спеціального обладнання. Важливо, щоб інформація подавалася в наочному вигляді (графіки, іконки) і не перенавантажувала користувача зайвими деталями.

- **Сумісність та інтеграція.** Якщо в бригаді вже використовуються деякі інформаційні системи (наприклад, облік на складах), наша система має бути сумісна з ними або легко інтегруватися через обмін файлами/даними. Також важливо врахувати, що система може функціонувати в умовах обмеженого інтернет-зв’язку, тобто повинна мати можливість роботи автономно, синхронізуючись з іншими вузлами, коли зв'язок є.

- **Вимоги до експлуатації.** Система повинна бути достатньо легкою у розгортанні і підтримці. Ідеально – можливість працювати на звичайному ноутбуці без потреби в потужному серверному обладнанні на передовій. Оновлення системи мають проводитися централізовано в умовах, коли це дозволяє ситуація (наприклад, під час ротації чи на вищому рівні командування).

- **Регуляторні вимоги.** Система повинна відповідати стандартам і керівним документам, які існують у ЗСУ щодо захисту інформації, сумісності з системами НАТО (якщо такі вимоги є), стандартам представлення даних (наприклад, єдині класифікатори предметів постачання).

4.3 Вимоги до інтерфейсу

Вимоги до інтерфейсу користувача визначають, яким чином інформація і функції системи мають бути представлені користувачу:

- **Головна панель і інформаційна панорама.** Після входу в систему користувач повинен бачити загальний стан: ключові показники (наприклад, % забезпеченості по основних ресурсах, найбільш критичні потреби). Може бути реалізовано у вигляді дашборду з індикаторами ("зеленим/жовтим/червоним" для рівнів запасів).

- **Навігація по розділах.** Інтерфейс має містити зрозуміле меню або вкладки для доступу до основних розділів: введення даних (структура, запаси), налаштування сценарію, результати прогнозу, звіти, адміністративні налаштування. Меню повинно бути логічно структурованим.

- **Форми введення даних.** Екрани, де офіцер вводить чисельність підрозділів, перелік техніки, поточні залишки, мають бути простими. Використовувати випадаючі списки для вибору типів техніки чи матеріалів (щоб уніфікувати дані і зменшити помилки вводу). Поля повинні мати підказки (наприклад, в одиницях виміру – "літрів", "кілограми", "штук" тощо).

- **Настройки параметрів прогнозу.** Для введення умов операції доцільно мати спеціальне діалогове вікно або форму, де користувач вибирає тип операції (випадаючий список), задає дати початку/кінця (календарний вибір), відмічає галочками особливі умови ("мороз", "бездоріжжя", "висока інтенсивність"). Інтерфейс може одразу показувати, як ці фактори вплинуть на нормативи (наприклад, при виборі "мороз" може з’являтися текст "нормативи палива будуть збільшені на 15%" для прозорості).

- **Відображення результатів.** Результати прогнозу мають відображатися у читабельній формі. Наприклад, таблиця: рядки – категорії ресурсів, колонки – дні операції (або фази операції), значення – потреба/залишки. Ключові цифри (де дефіцит) виділити кольором. Можна перемикатися між загальним переглядом (по бригаді) і деталізацією по батальйонах/ротах.

- **Графічні елементи.** Бажано відобразити графіки витрат: наприклад, лінійний графік, що показує прогнози і фактичні витрати по днях. Так офіцеру буде зрозуміліше тенденція (чи зростає витрата, чи в межах плану). Також може бути діаграма розподілу ваги вантажів по категоріях

для розуміння, що найбільше впливає на транспортне навантаження.

- **Повідомлення та підтвердження.** Якщо система генерує попередження ("Нестача боєприпасів через 3 дні"), це має бути помітно – наприклад, червона іконка з текстом. Коли користувач виконує критичну дію (видалення даних, затвердження остаточного плану), потрібно запитувати підтвердження, щоб уникнути випадкових помилок.

- **Локалізація.** Інтерфейс – українською мовою. Всі терміни повинні відповідати військовій доктрині (наприклад, слово "батальйон" повністю, а не скорочення тощо). Якщо планується використання з партнерами – може бути опція перемикання мови на англійську.

- **Доступність та UI/UX.** Інтерфейс має бути побудований за принципами, зручними для швидкого використання: контрастні кольори, великі читабельні шрифти (можливо система буде використовуватися в польових умовах, на сонці чи вночі – тож важлива видимість). Команди, які використовуються часто, мають бути легко доступні (наприклад, кнопка "Оновити прогноз" прямо на головному екрані). В цілому UX повинен бути інтуїтивним, щоб навчання користувачів займало мінімум часу.

4.4 Вимоги до даних

Ця система оперує значними обсягами даних про підрозділи, ресурси і споживання. Вимоги до даних включають:

- **Структура даних підрозділів.** Система повинна зберігати інформацію про організаційну структуру: список підрозділів (бригада, батальйони, роти), їхні характеристики (тип, чисельність особового складу, кількість одиниць техніки за типами). Повинні бути уніфіковані довідники типів підрозділів та техніки, щоб дані були цілісними (напр. тип техніки: "танк Т-64", "БМП-2" тощо з відповідними параметрами витрат).

- **Нормативно-довідкові дані.** База нормативів споживання ресурсів: наприклад, норми витрати пального на 100 км для кожного типу техніки; норми витрати боєприпасів на одну одиницю зброї за добу інтенсивного бою; норми продовольства на людину тощо. Ці дані мають бути налаштовані і за потреби редаговані адміністратором системи, оскільки вони можуть змінюватися або уточнюватися.

- **Дані про запаси і постачання.** Система зберігає поточні залишки

ресурсів по категоріях (скільки є на складі бригади, можливо, з деталізацією по місцях зберігання). Також вона повинна вміти враховувати заплановані поставки (наприклад, якщо відомо, що завтра прибуде конвой з пальним – це має бути відображено для коректного прогнозу).

- **Оперативні дані про витрати.** Під час виконання операції потрібно вводити фактичні дані про витрати ресурсів (щоб система могла коригувати прогноз). Ці дані можуть надходити щоденно з батальйонів (по ключових позиціях). Система повинна агрегувати їх та зберігати хронологію витрат.

- **Історичні дані та база знань.** Бажано, щоб система накопичувала історію виконаних прогнозів і фактичних витрат для подальшого аналізу. Це дозволить удосконалювати моделі прогнозування (наприклад, порівнювати прогноз і факт, визначати середню похибку, підлаштовувати нормативи). Також історичні дані можуть служити тренувальним набором для ML-модулів системи.

- **Обсяг даних і продуктивність.** Передбачається, що система оперуватиме даними про декілька тисяч найменувань ресурсів (якщо деталізувати до кожного типу боєприпасів чи запчастин). Необхідно оптимізувати зберігання (наприклад, використовувати реляційну базу даних з індексами) та звернення до даних, щоб це не сповільнювало роботу.

- **Якість даних.** Дуже важливо підтримувати актуальність даних. Це не стільки технічна вимога, скільки методична: повинні бути процедури регулярного оновлення інформації (наприклад, після кожного бою оновити витрати, щотижня оновити реальні залишки на складах). Система може допомагати, нагадуючи про необхідність оновлення або блокуючи застарілі прогнози, якщо дані не оновлені.

4.5 Вимоги до API інтеграції

Для того, щоб система могла взаємодіяти з іншими інформаційними системами (як національними, так і коаліційними), слід передбачити програмний інтерфейс (API):

- **REST API для доступу до даних.** Система повинна мати захищений RESTful API, через який авторизовані зовнішні системи можуть отримувати ключові дані: наприклад, запит "отримати прогноз потреб

на наступні 7 днів по пальному" або "оновити поточні залишки за категоріями". Формат обміну – JSON або XML, в залежності від стандартів, прийнятих у Міноборони.

- **Сумісність з NATO стандартами.** Якщо планується обмін даними з системою типу LOGFAS або іншими, API повинен підтримувати конверсію даних в потрібні формати. Наприклад, експортувати зведення про потреби у форматі, який може бути імпортовано до LOGFAS як заявка на постачання.

- **Інтеграція з складськими системами.** Можливий варіант – підключення до систем обліку запасів (складу). API може дозволяти отримувати фактичні залишки автоматично, без ручного введення, якщо на складі є електронний облік. Так само і навпаки – передавати рішення про розподіл ресурсів назад у систему складу для виконання (формування накладних).

- **Аутентифікація та безпека API.** Обмін даними має бути захищеним. Використання API – тільки через HTTPS, з обов'язковою автентифікацією (наприклад, по токенах або сертифікатах). Права на API-виклики теж слід обмежувати (окремі ключі з правом тільки читати дані, або читати+записувати).

- **Документованість.** Для API потрібно надати документацію (специфікацію кінцевих точок, параметрів). Можливо, реалізувати стандарт OpenAPI (Swagger) для зручності інтеграції сторонніми розробниками.

- **Продуктивність та ліміти.** Якщо дані запитуються зовнішніми системами часто, API має справлятися без значного впливу на основну роботу системи. Можливо, потрібно передбачити кешування відповідей на запити, що часто повторюються. Також – ліміти на запити (rate limiting), щоб запобігти перевантаженню чи зловмисному впливу.

Визначені вимоги стануть основою для подальшого проектування системи, зокрема для вибору архітектурних рішень та технологій.

Розділ 5. Моделювання і проектування

На цьому етапі на основі вимог здійснюється побудова моделі майбутньої системи: визначаються її основні складові (компоненти), загальна архітектура та структура даних. Проектування буде виконано на концептуальному рівні, із представленням ключових діаграм.

5.1 Основні компоненти системи

Розроблювана система СППР для прогнозування логістичних потреб матиме наступні основні компоненти:

- **Модуль вводу та управління даними.** Відповідає за створення і підтримку актуальної бази даних про підрозділи, ресурси, нормативи та фактичні дані. Реалізує інтерфейси введення інформації, а також імпорт/експорт даних. Цей модуль забезпечує цілісність довідникової інформації (типи ресурсів, підрозділів) та актуальність записів про залишки.

- **Модуль прогнозування та аналітики.** Ядро системи, яке на основі вхідних даних здійснює розрахунки потреб. Він включає реалізацію алгоритмів прогнозування: від простих формул по нормативам до складніших (можливо, ML-моделей) для уточнення прогнозу. Також цей компонент відповідає за аналіз наявних запасів і виявлення потенційних дефіцитів. Входом модулю є сценарні параметри та поточний стан, виходом – набір прогнозних даних та рекомендацій.

- **Модуль підтримки рішень (рекомендацій).** Пов’язаний з попереднім, він формулює на основі результатів прогнозу конкретні поради для користувача: які дії слід виконати (поповнити такий-то ресурс, перерозподілити між складами і т.д.). Фактично, це рівень бізнес-логіки, що "розуміє" контекст: наприклад, якщо прогноз показав нестачу пального, модуль рекомендацій може згенерувати варіанти – зменшити інтенсивність використання техніки або запросити додаткове пальне з оперативного складу.

- **База даних (репозиторій даних).** Центральне сховище всіх необхідних даних. Швидше за все, реалізоване у вигляді реляційної БД з таблицями: "Підрозділи", "Техніка", "Ресурси", "Нормативи", "Залишки", "Прогнози", "Витрати" тощо. База даних має забезпечувати цілісність (наприклад, не допустити видалення даних про ресурс, якщо по ньому є запис прогнозу).

- **Модуль інтерфейсу користувача.** Відповідає за взаємодію з користувачем. Це сукупність екранів, форм, діаграм, про які йшлося в вимогах до інтерфейсу. Цей модуль може бути реалізований як веб-інтерфейс (в браузері) або як окремий застосунок з графічним інтерфейсом. Він звертається до інших модулів (через API або внутрішні виклики) для отримання/відправлення даних.

- **API/Integration модуль.** Надає програмний інтерфейс для зовнішніх систем. Він взаємодіє з базою даних та модулем логіки, але виставляє тільки ті операції, які дозволені зовні (з дотриманням безпеки). Може бути реалізований як окремий веб-сервіс (наприклад, RESTful service), що працює на тому ж сервері.

- **Модуль безпеки та управління доступом.** Відповідає за автентифікацію, авторизацію користувачів, шифрування даних. Він інтегрується з інтерфейсним модулем (для авторизації при вході) і з API (перевірка ключів). Також може забезпечувати логування подій безпеки (наприклад, невдалих спроб входу).

- **Журнал подій та аудиту.** Механізм, який збирає записи про дії користувачів та ключові зміни системи. Може бути реалізований як частина БД або окремий сервіс. Цей компонент важливий для відстеження, хто і коли змінив дані чи отримав прогноз.

Взаємодія між компонентами здійснюється за чітко визначеними інтерфейсами. Зокрема, модуль інтерфейсу та API звертаються до модуля прогнозування через його інтерфейси (методи) або через звернення до БД (в залежності від обраної архітектури – може бути багатошарова архітектура з виділенням окремого шару сервісів).

5.2 Загальна архітектура системи

Архітектуру програмної системи можна охарактеризувати як класичну клієнт-серверну багаторівневу структуру. Вона включає:

- **Клієнтську частину**, що реалізована у вигляді веб-інтерфейсу, доступного через браузер (розроблена за допомогою Blade-шаблонів Laravel та JavaScript);

- **Серверну частину**, яка відповідає за обробку бізнес-логіки, розрахунки прогнозів, обробку запитів API;

- **Базу даних**, яка зберігає структуровану інформацію про підрозділи, ресурси, норми споживання, прогнози та інше.

Система реалізована з використанням таких технологій:

- **Backend**: PHP-фреймворк Laravel — забезпечує маршрутизацію, обробку запитів, авторизацію, бізнес-логіку та API;

- **Frontend**: Blade (шаблонізатор Laravel), JavaScript, Chart.js — для побудови інтерфейсу та візуалізації даних (графіки витрат, прогнози тощо);

- **База даних**: MySQL — реляційна СУБД для зберігання основних сутностей системи (бригади, ресурси, прогнози тощо);

- **Безпека**: вбудовані засоби Laravel для аутентифікації, авторизації, CSRF-захисту;

- **API**: REST API для обміну даними з потенційними зовнішніми системами (наприклад, складськими обліковими системами чи логістичними платформами).

Ця архітектура забезпечує чітке розділення функцій: клієнт відповідає за взаємодію з користувачем та відображення інформації, а сервер — за логіку, розрахунки й зберігання даних. Такий підхід сприяє гнучкості, масштабованості та спрощує майбутнє розширення функціональності системи. Крім того, система здатна працювати в обмежених умовах (наприклад, на ноутбуці бригади), що критично важливо для військових застосувань.

5.3 ER-модель даних

Для проектування структури бази даних побудовано спрощену ER-діаграму, що відображає основні сутності (entity) системи та зв’язки між ними.

[Діаграма 5.1. ER-модель даних]

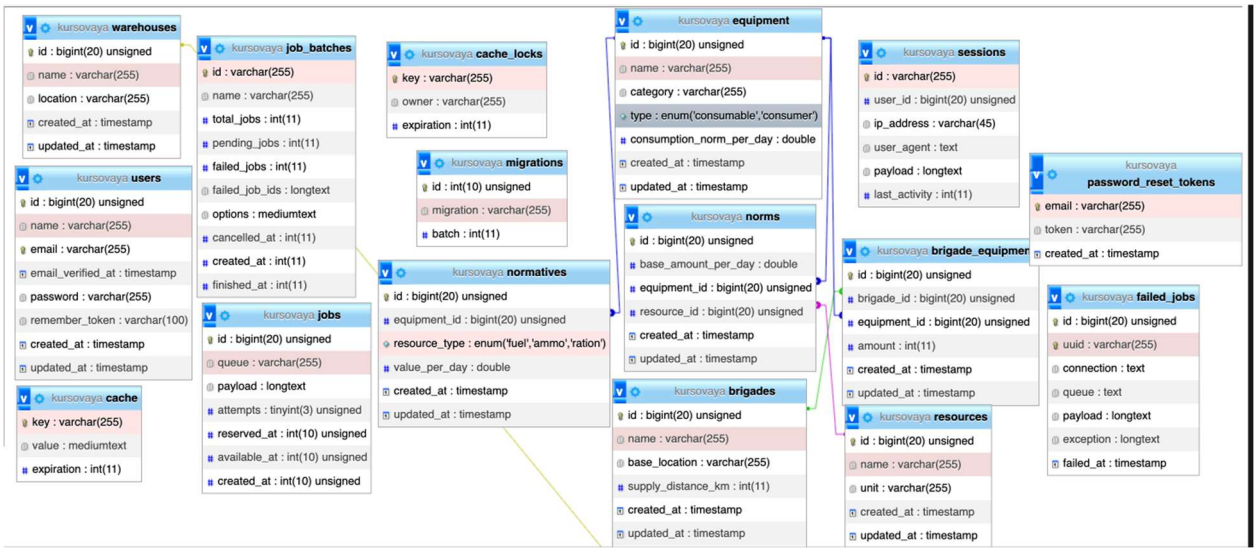

Рис.5.1 - Реалізована структура бази даних системи (MySQL Workbench)

Опис ER-діаграми: можна виділити такі основні сутності:

- **Підрозділ** (Unit) – характеризує елемент структури військ. Атрибути: код/назва підрозділу, тип (бригада, батальйон, рота), належність до вищого підрозділу (зв'язок "підрозділ – батьківський підрозділ" рекурсивний для ієрархії). Зв'язки: підрозділ може мати багато записів у сутності "Споживання" (витрати ресурсів цим підрозділом).

- **Матеріальний ресурс** (Resource) – довідник усіх видів ресурсів. Атрибути: код ресурсу, назва (наприклад, 120-мм міна, дизельне паливо, сухпайок), категорія (боєприпаси, паливо, продовольство тощо), одиниця виміру. Зв'язки: ресурс пов'язаний з нормативами і зі залишками.

- **Норматив** (ConsumptionRate) – зберігає норму витрати певного ресурсу в певному контексті. Атрибути: величина (кількість на одиницю за одиницю часу), умови (наприклад, тип операції = "наступ", чи клімат = "зима"). Зв'язки: пов'язана з Resource (для якого ресурсу), можливо з типом підрозділу чи техніки (якщо норма специфічна).

- **Техніка/Обладнання** (Equipment) – довідник зразків техніки. Атрибути: назва, тип (наприклад, вантажівка, танк, кухня польова), характеристика споживання (наприклад, витрата палива на 100 км). Зв'язки: може бути використано в розрахунках норм; підрозділ може мати перелік техніки.

- **Залишок** (Stock) – таблиця, що відображає, скільки якого ресурсу є на складі. Атрибути: кількість, дата оновлення. Зв'язки: зв'язана з Resource (якого ресурсу), можливо, з підрозділом (якщо модель враховує розподілені запаси по підрозділах, хоча найчастіше йдеться про один

основний склад бригади).

- **Прогноз** (Forecast) – сутність, що зберігає згенерований прогноз. Атрибути: ідентифікатор прогнозу, період дії (дата початку, дата кінця), створений користувачем, дата створення. Зв'язки: має підпорядковані записи "ForecastItem" (позиції прогнозу).

- **Позиція прогнозу** (ForecastItem) – окрема потреба. Атрибути: дата (або етап), кількість потреби. Зв'язки: пов'язана з Forecast (належить прогнозу), з Resource (якого ресурсу стосується), і можливо з підрозділом (якщо прогноз деталізовано по підрозділах).

- **Споживання (витрата)** (ConsumptionLog) – журнал фактичного споживання. Атрибути: дата, кількість. Зв'язки: з Resource (що витрачено), з підрозділом (ким витрачено, якщо відомо) – таким чином можна буде порівнювати з прогнозом.

- **Користувач** (User) – для управління доступом. Атрибути: ім'я, роль, хеш паролю. Зв'язки: можливо пов'язаний з підрозділом (наприклад, щоб обмежити доступ даними свого підрозділу).

- **Рекомендація** (Recommendation) – опціонально, зберігає сформульовані системою поради. Атрибути: текст рекомендації, пріоритет, статус (прийнято/відхилено). Зв'язки: може прив'язуватися до прогнозу або окремого ресурсу.

Зв'язки між сутностями включають:

- Unit – Unit (ієрархія, наприклад, поле ParentUnitID).

- Unit – Equipment (можливо, через проміжну сутність типу "UnitEquipment" із зазначенням кількості одиниць техніки в підрозділі).

- Resource – ConsumptionRate (1 до багатьох, один ресурс може мати кілька нормативів під різні умови).

- Resource – Stock (1 до 1: для кожного ресурсу може бути один запис залишку на складі; або 1 до багатьох, якщо множинні місця зберігання).

- Resource – ForecastItem (1 до багатьох, ресурс представлений у багатьох позиціях прогнозу по датах).

- Forecast – ForecastItem (1 до багатьох, прогноз містить багато позицій).
- Resource – ConsumptionLog (1 до багатьох, по кожному ресурсу багато записів витрат).
- Unit – ConsumptionLog (1 до багатьох, кожен запис витрати віднесено до певного підрозділу-споживача).
- User – (можливі зв'язки з Unit або просто атрибут role).
- Forecast – Recommendation (1 до багатьох, прогноз може породити кілька рекомендацій).

Наведена ER-модель є базовою. В реальній БД можна додати додаткові атрибути (наприклад, для Resource – щільність для палива, щоб врахувати об'єм, для Unit – вид військ тощо). Але у рамках проектування дипломної роботи цього рівня деталізації достатньо, щоб зрозуміти, як інформація структурована.

В результаті виконаного моделювання визначено, як система буде побудована та які дані обробляти. Наступним етапом розробки може бути детальне технічне проектування (вибір конкретних технологій, розробка інтерфейсів користувача) та реалізація прототипу системи.

5.4 DFD – Data Flow Diagram

На рівні 1 діаграми потоків даних (DFD) відображено взаємодію основних компонентів системи з точки зору обміну інформацією. Основні модулі системи включають користувацький інтерфейс, модуль обробки прогнозів та модуль збереження даних (базу даних). Нижче наведено текстовий опис потоків даних між цими модулями:

- **`Інтерфейс користувача`** → **`Модуль обробки прогнозів`:** передача параметрів нового прогнозу, введених користувачем (наприклад, діапазону дат та критеріїв прогнозу), до сервера для обчислення.
- **`Модуль обробки прогнозів`** → **`База даних`:** запит історичних даних, необхідних для побудови прогнозу. Модуль обробки отримує записані дані про минулі показники з бази даних і використовує їх для розрахунків.

- **`Модуль обробки прогнозів` → `Інтерфейс користувача`:** передача результатів обчисленого прогнозу назад до інтерфейсу для відображення користувачу. Після обробки запиту система надсилає сформований прогноз у вигляді даних до клієнтської частини.

- **`Інтерфейс користувача` → `База даних`:** зберігання нового або відкоригованого прогнозу. Коли користувач створює чи редагує прогноз, зміни передаються до сервера і зберігаються в базі даних для подальшого використання та аналізу.

- **`База даних` → `Інтерфейс користувача`:** видача запитуваних прогнозів для перегляду. Користувач може отримувати список наявних прогнозів або конкретний прогноз, дані якого дістаються з бази даних і відображаються в інтерфейсі.

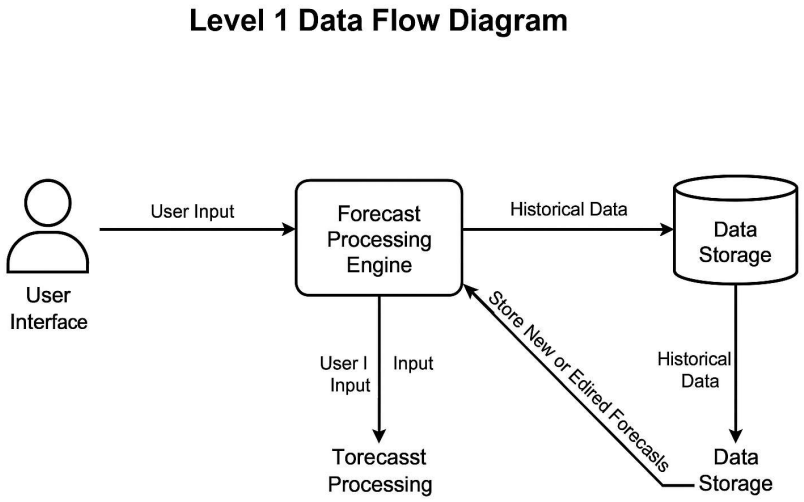

Рис.5.2 DFD діаграма

5.5 UML – Діаграма варіантів використання

Діаграма варіантів використання (Use Case Diagram) ілюструє основні сценарії взаємодії користувача з системою. Головним актором у системі є користувач (наприклад, аналітик або менеджер), який працює з функціоналом прогнозування. Основними сценаріями (варіантами використання) є **створення прогнозу**, **перегляд прогнозу** та **редагування прогнозу**. Опис цих сценаріїв наведено нижче:

- **Створення прогнозу:** користувач вводить необхідні параметри для

побудови прогнозу (наприклад, часовий інтервал та критерії) та запускає обчислення. Система обробляє введені дані, генерує прогноз на основі історичних даних і зберігає отриманий результат у базі даних для подальшого використання.

- **Перегляд прогнозу:** користувач отримує список раніше створених прогнозів або обирає конкретний прогноз для перегляду. Система витягує обрані прогнозні дані з бази даних і відображає їх у зручному форматі, надаючи можливість аналізу отриманих результатів.

- **Редагування прогнозу:** користувач може відкрити існуючий прогноз, внести зміни до його параметрів чи критеріїв та повторно ініціювати його обчислення. Після внесення змін система повторно обчислює оновлений прогноз та оновлює відповідний запис у базі даних.

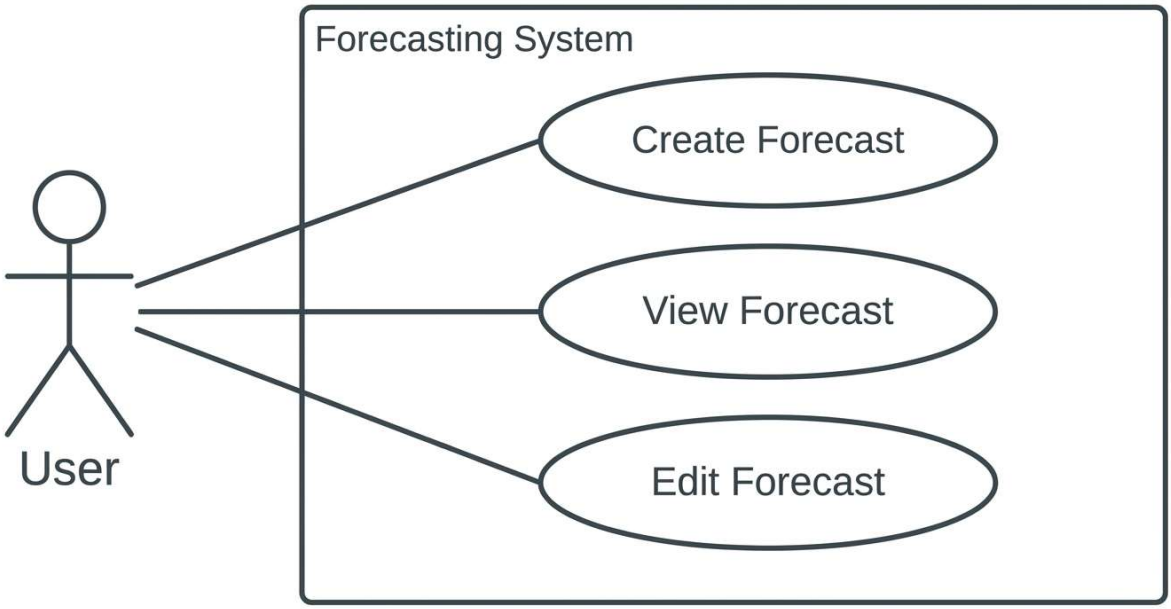

Рис.5.3 UML діаграма

Розділ 6. Реалізація системи прогнозування логістичних потреб

6.1 Архітектура веб-додатку

Веб-додаток реалізовано на базі фреймворку **Laravel** з використанням шаблонів **Blade** для побудови інтерфейсу користувача. Архітектура додатку дотримується шаблону Model-View-Controller (MVC): логіка роботи з даними зосереджена в моделях і контролерах, а представлення реалізоване у вигляді Blade-шаблонів. Це забезпечує чітке розділення функціональності і полегшує підтримку коду.

У системі передбачено лише один тип користувача – **адміністратор**, що спрощує управління доступом. Адміністратор має повні права для роботи з додатком: введення вихідних даних, запуск розрахунків та перегляд результатів. Відсутність декількох ролей знижує складність реалізації контролю доступу і логіки інтерфейсу.

Серверна частина додатку написана мовою PHP. Логіка застосунку зосереджена у контролерах Laravel – зокрема, основні розрахунки виконує, наприклад, контролер ForecastController. Шаблони **Blade** використовуються для генерації HTML-сторінок інтерфейсу адміністратора. Такий підхід дозволяє відокремити програмну логіку від представлення: Blade-шаблони містять тільки мінімальну кількість PHP-коду для відображення даних (цикли, умови), що робить код інтерфейсу чистішим і зрозумілішим.

Система зберігає дані в реляційній базі даних (наприклад, MySQL). Laravel надає ORM **Eloquent**, тож кожна сутність (бригада, техніка, ресурс тощо) представлена відповідною модельною класою і пов'язаною таблицею в БД. Це спрощує виконання запитів до бази і забезпечує зручне маніпулювання даними без написання сирих SQL-запитів.

6.2 Моделі даних та структура бази даних

Для реалізації предметної області визначено низку основних моделей (сутностей). Назви моделей та їх атрибутів наведено англійською мовою згідно з практикою розробки. Ключові моделі даних у системі такі:

- **Brigade** – модель, що представляє військову бригаду (підрозділ). Містить інформацію про сам підрозділ: назву, тип, можливо локацію та інші характеристики. Бригада пов'язана з наявною в ній технікою і особовим складом, від яких залежать логістичні потреби.

- **Equipment** – модель техніки та обладнання. Визначає типи техніки (наприклад, танк, БТР, вантажівка тощо) та кількість одиниць техніки в кожній бригаді. Кожен запис **Equipment** може містити посилання на **Brigade** (до якої належить) і, опціонально, на нормативи споживання ресурсів.

- **Resource** – модель, що представляє тип ресурсу або матеріалу. Це можуть бути пальне (різних видів), боєприпаси, продовольство, медичні засоби та інші види постачання. Модель містить назву ресурсу і одиниці виміру (літри, кілограми, штуки тощо).

- **ConsumptionNorm** – модель нормативів споживання. Визначає базові

норми витрат ресурсів певним типом техніки за одиницю часу або відстані. Наприклад, запис **ConsumptionNorm** може містити, що танк споживає 300 л дизельного пального на 100 км або 10 л на годину роботи двигуна. Такі нормативи використовуються як база для прогнозування.

- **Warehouse** – модель складів постачання. Містить інформацію про логістичні склади чи бази забезпечення: назву складу, його місце розташування (координати або адресу). Ці дані застосовуються при плануванні постачання, щоби врахувати відстані та маршрути.

- **Distance** – умовна сутність для зберігання відстаней між точками постачання і підрозділами. В системі можна зберігати відстань від кожного **Warehouse** до кожної **Brigade** (наприклад, як окрема таблиця або як поле в моделі **Brigade**). Цей параметр може враховуватися при розрахунку потреб у пальному для транспортування ресурсів.

Додайте діаграму класів для основних моделей даних, щоб наочно показати структуру бази даних та зв'язки між сутностями. На діаграмі варто відобразити класи **Brigade**, **Equipment**, **Resource**, **ConsumptionNorm**, **Warehouse** та зв'язки між ними (наприклад, «багато-до-одного» між **Equipment** і **Brigade**, «багато-до-багатьох» між **Brigade** і **Resource** через споживання тощо).

Кожна з перелічених моделей має відповідну таблицю в базі даних. Зв'язки між таблицями забезпечують узгодженість даних. Наприклад, **Brigade** пов'язана з **Equipment** (бригада може мати багато записів техніки), а **Equipment** може посилатися на **ConsumptionNorm** для отримання норми споживання відповідних ресурсів. **Resource** може бути пов'язаний із **ConsumptionNorm** (щоб знати, яка норма належить до якого ресурсу). Сутність **Warehouse** може мати зв'язок з **Brigade** через таблицю відстаней (наприклад, кожна пара «бригада–склад» має числове значення відстані).

6.3 Інтерфейс користувача адміністратора

Інтерфейс користувача (адміністратора) представлений у вигляді веб-сторінок, створених за допомогою шаблонів Blade. Інтерфейс розроблено з акцентом на простоту і зручність, враховуючи, що адміністратор – єдиний користувач системи. Після автентифікації (простий логін для адміністратора) користувач отримує доступ до головної панелі керування. Звідти доступні розділи для введення даних та запуску прогнозування.

Основним екраном є форма введення параметрів прогнозу. Адміністратор може задати необхідні умови сценарію, за яким буде проводитись прогноз логістичних потреб. Зокрема, через інтерфейс передбачено введення та вибір таких даних:

- **Вибір бригади**: зі списку наявних у системі бригад адміністратор обирає конкретний підрозділ (або групу підрозділів) для прогнозування. Це визначає, для якої бригади будуть розраховані потреби.

- **Параметри сценарію**: введення змінних умов, що впливають на споживання. Наприклад, тривалість місії (у днях), інтенсивність бойових дій (коефіцієнт, що коригує норму споживання, де 1.0 – стандартна інтенсивність, 1.2 – підвищена і т.д.), тип місцевості або пори року (що теж може бути враховано коефіцієнтом, наприклад, для складних умов постачання). Кожен з цих параметрів задається або чисельно, або шляхом вибору із наперед визначених опцій.

- **Обрання моделі прогнозування**: оскільки система підтримує дві моделі прогнозування (базову та ML), інтерфейс може містити перемикач або список для вибору моделі розрахунку. За замовчуванням обрана базова модель (нормативна), але адміністратор має змогу переключитися на умовну ML-модель, якщо вона доступна.

Після заповнення вихідних даних адміністратор ініціює розрахунок, натиснувши кнопку (наприклад, **Calculate** / “Розрахувати”). Далі відбувається обробка введених умов на сервері і обчислення прогнозованих потреб.

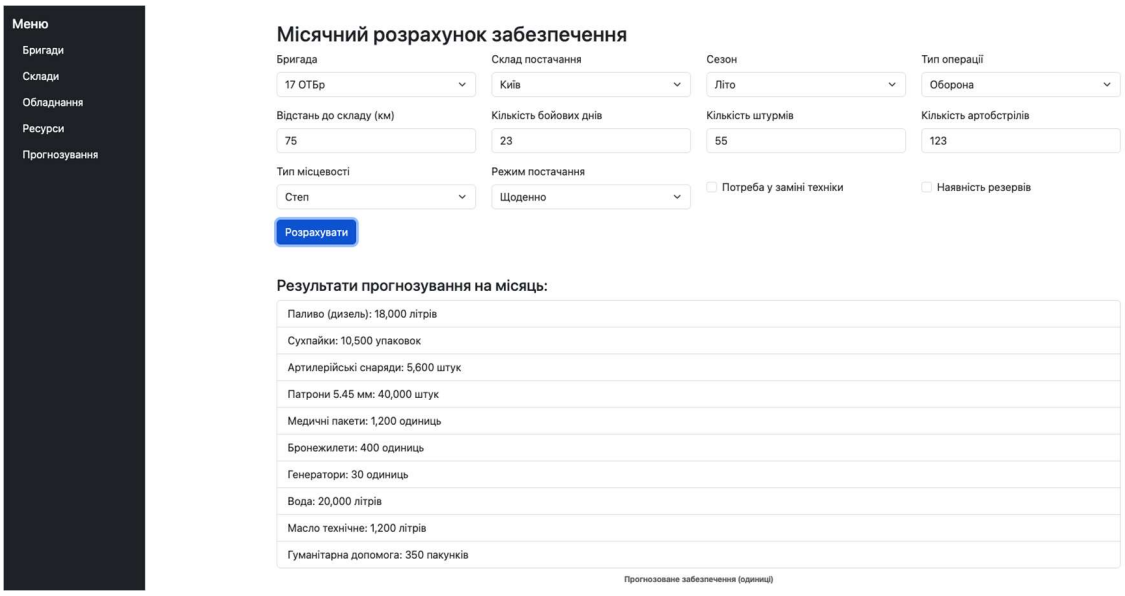

Рис. 6.1 - Демонстрація інтерфейсу системи

Після запуску обчислення система перенаправляє адміністратора на сторінку результатів. На ній відображаються результати прогнозу у зручному для аналізу вигляді. Зазвичай результати включають:

- **Таблицю з прогнозованими потребами**: перелік ресурсів та необхідна кількість кожного для заданого сценарію. Наприклад, для бригади може бути вказано, що потрібно 5000 літрів дизельного пального, 200 шт. артилерійських снарядів, 100 продовольчих пайків тощо – залежно від введених умов. Таблиця може бути згрупована по типах ресурсів або по підрозділах (якщо обрано кілька бригад), з підсумковими значеннями.

- **Графічні матеріали (діаграми)**: для кращого сприйняття система будує графіки. Це можуть бути стовпчикові діаграми або кругові діаграми, що ілюструють структуру потреб. Наприклад, діаграма, яка показує у відсотках, яку частину від загального обсягу займає пальне, боєприпаси, продовольство тощо. Інший графік може порівнювати потреби різних бригад (якщо аналізується більше одного підрозділу).

Результати відображаються інтерактивно: адміністратор бачить числові значення та може візуально оцінити їх через графіки. Інтерфейс також може дозволяти експортувати ці дані (наприклад, у формат Excel або PDF) для подальшого використання у звітах.

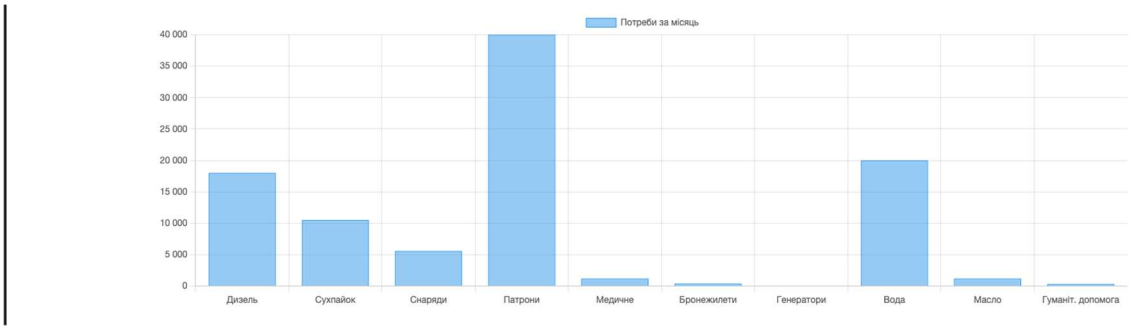

Рис.7.2 - Графічне відображення потреб

Крім основної функції прогнозування, інтерфейс адміністратора може містити підрозділи для управління довідниковими даними. Наприклад, окремі сторінки або модальні вікна для редагування списку бригад (додавання нової бригади, оновлення її складу), управління довідником техніки та норм

споживання (внесення нових типів техніки і нормативів), а також управління списком складів. Це дозволяє підтримувати актуальність даних без прямого втручання в базу. Всі ці дії також виконуються через Blade-формами, а контролери забезпечують збереження змін до бази даних.

6.4 Алгоритм прогнозування логістичних потреб

Основою системи є модуль прогнозування, який здійснює розрахунок потреб у ресурсах на базі введених даних. У поточній реалізації використано **базову прогнозну модель**, що базується на нормативних показниках споживання (формула типу «норма × коефіцієнти»). Передбачено також можливість використання альтернативної, умовної **ML-моделі** (моделі з застосуванням методів машинного навчання) для більш точної оцінки, причому система дозволяє легко переключатися між двома підходами.

Базова модель прогнозування. Цей підхід полягає у розрахунку потреб виходячи з наперед визначених норм споживання і коригувальних коефіцієнтів. Алгоритм роботи базової моделі можна описати такими кроками:

1. **Збір даних.** Після того, як адміністратор задав сценарій (обрав бригаду і ввів параметри), система отримує з бази даних всю необхідну інформацію про вибраний підрозділ. Зокрема, завантажуються дані про наявну техніку в цій бригаді: перелік типів **Equipment** та кількість одиниць кожного типу. Також завантажуються відповідні нормативи споживання **ConsumptionNorm** для цієї техніки і дані про ресурси **Resource**, які потрібні для експлуатації кожного виду техніки.

2. **Розрахунок для кожної одиниці техніки.** Алгоритм ітерується по всіх типах техніки в бригаді. Для кожного типу визначається, які ресурси він споживає і в якій кількості за стандартних умов. Ці базові норми (наприклад, літрів пального на годину роботи або снарядів на день бойових дій) беруться з моделі **ConsumptionNorm**. Потім норма множиться на кількість одиниць цього типу техніки в бригаді та на коригувальні коефіцієнти, що відповідають заданим умовам. Коефіцієнти можуть включати:

    - **Коефіцієнт інтенсивності** бойових дій: якщо сценарій передбачає підвищену активність, цей коефіцієнт > 1 збільшує витрату (наприклад, 1.2 для інтенсивного режиму, 0.8 для навчань і т.д.).

    - **Коефіцієнт умов місцевості/сезону**: враховує вплив середовища (наприклад, у зимових умовах витрати пального можуть бути

більші, або в гірській місцевості техніка споживає більше ресурсів).

- **Тривалість операції**: у разі якщо прогноз робиться на певний період (наприклад, на 7 днів), обчислені добові потреби множаться на цю кількість діб.

3. **Агрегація результатів.** Розраховані витрати для кожного виду техніки по кожному виду ресурсу сумуються між собою, формуючи загальні потреби бригади. На цьому етапі система може також врахувати додаткові потреби, не пов'язані прямо з технікою – наприклад, базові потреби особового складу (вода, їжа) за аналогічною схемою (норма на одну людину × кількість військовослужбовців × час × коефіцієнти).

4. **Врахування відстаней та логістичних втрат (опціонально).** Якщо залучено дані про **Warehouse** та **Distance**, алгоритм може скоригувати потреби з урахуванням доставки. Наприклад, для доставки пального з найближчого складу на відстані 100 км може знадобитися витратити певну частину самого пального (для роботи транспорту), що додається до загальної потреби. В базовій моделі ці моменти можуть враховуватися простим коефіцієнтом логістичних втрат (наприклад, +5% до пального на кожні 100 км перевезення).

5. **Формування виходу.** Система готує структурований результат – перелік потрібних ресурсів та їх кількість. Ці дані передаються у вигляді масивів/колекцій до представлення (Blade-шаблону), де відображаються у вигляді таблиць і графіків, як було описано вище.

Алгоритм базової моделі є детермінованим та прозорим: при однакових введених даних результат завжди буде однаковим, оскільки розрахунок базується на фіксованих нормах і заданих коефіцієнтах.

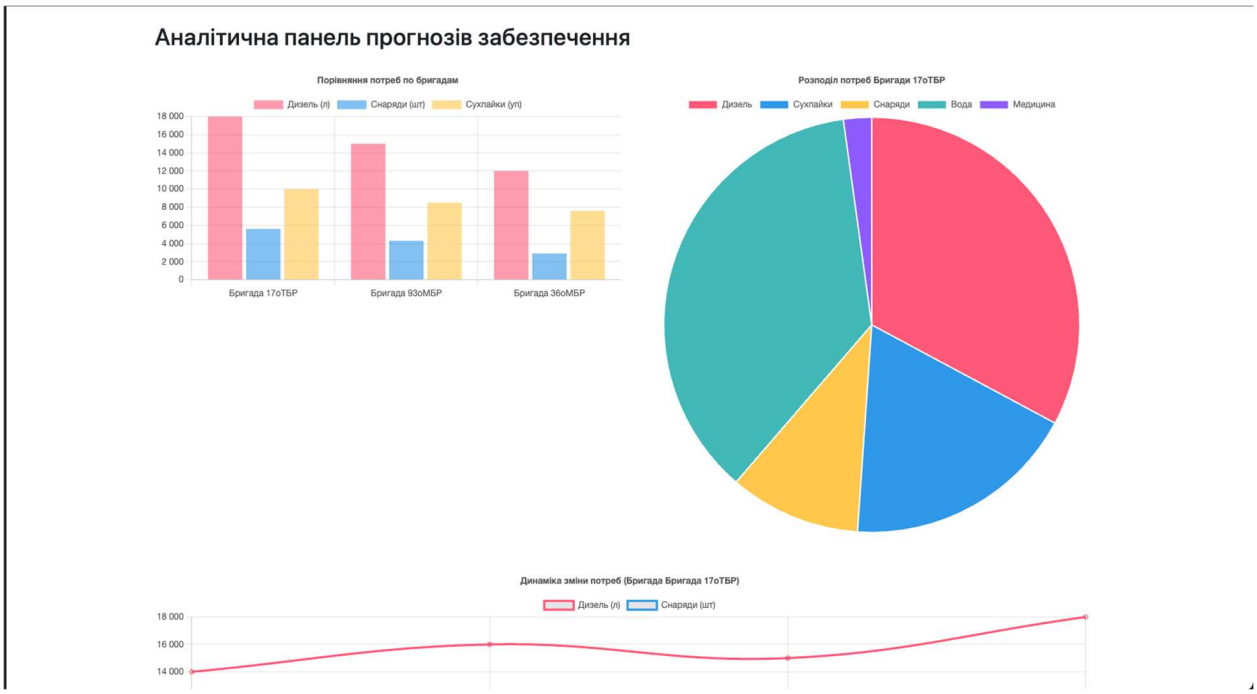

Рис.6.3 - Аналітична панель прогнозів забезпечення

Розширення до ML-моделі. Система спроектована таким чином, що прогнозний модуль можна замінити або доповнити моделлю машинного навчання. **ML-модель** могла б враховувати більшу кількість факторів і виявляти нелінійні залежності на основі історичних даних споживання. Наприклад, маючи дані про фактичне споживання ресурсів різними бригадами в реальних умовах, така модель може навчитися прогнозувати потреби більш точно, ніж це робить лінійна нормативна формула.

З точки зору реалізації, підтримка ML-моделі означає, що додаток має абстрактний інтерфейс або сервіс для обчислення прогнозу. Базова та ML-моделі реалізують один і той самий інтерфейс (метод розрахунку потреб) або дотримуються єдиного формату вводу/виводу. Це дає змогу **легко переключатись між двома моделями** – достатньо змінити налаштування або вибір у інтерфейсі, і система замість нормативної формули виконає прогноз через ML-алгоритм. У практичному плані це зроблено як конфігураційний параметр (напряму в коді або в .env файлі Laravel) або як опція в UI, яку обирає адміністратор перед запуском прогнозу.

Архітектура системи вже готова до цього: передбачено окремий сервіс або клас **ForecastService**, що може викликати або BasicForecastCalculator, або MLForecastCalculator в залежності від вибору.

Сумуючи, система забезпечує гнучкість у використанні алгоритмів прогнозування. Базова модель гарантує простоту і зрозумілість розрахунків, тоді як інтеграція моделі машинного навчання в дозволяє підвищити точність прогнозів. Адміністратор може в залежності від ситуації перемикатися між

цими двома підходами, що робить програму адаптивною до різних вимог і доступних даних.

Розділ 6.4.2. Машинне навчання (ML) у прогнозуванні логістичних потреб

Машинне навчання — це підхід до побудови прогностичних моделей, в яких комп'ютер самостійно «вчиться» від даних і знаходить залежності між вхідними факторами (ознаками) та прогнозованим результатом. Зазвичай це означає, що маємо набір історичних даних, де для різних бригад і умов відомі фактичні витрати ресурсів (цільова змінна *y*) і пов'язані з ними числові або категоріальні характеристики (ознаки *X*). Процес машинного навчання можна описати покроково:

- **Підготовка даних.** Збираються історичні дані: наприклад, кількість одиниць техніки певного типу у підрозділі, чисельність особового складу, параметри середовища (тип місцевості, сезон), інтенсивність дій тощо, а також реальні витрати ресурсів (паливо, боєприпаси, продукти тощо) за відповідний період. Ці дані очищуються (усуваються пропуски, аномалії) і кодуються: числові ознаки залишаються числами, а категоріальні перетворюються у числові представлення (наприклад, one-hot-кодування або підстановка числових індексів).

- **Навчання моделі.** Обирається алгоритм машинного навчання (лінійна регресія, деревовидний метод, бустінг тощо) і здійснюється налаштування його параметрів (тренування) на підготовлених даних. Модель оптимізує внутрішні параметри (ваги, правила розбиття тощо) таким чином, щоб прогнозовані значення $\hat{y}$ якнайточніше відповідали реальним витратам y. Наприклад, у випадку лінійної регресії формується модель виду:

$$\hat{y} = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + \cdots + \beta_n x_n,$$

- де x_i – вхідні ознаки (наприклад, кількість одиниць техніки, коефіцієнти інтенсивності, показник сезону), а β_i – вагові коефіцієнти, які налаштовуються в процесі навчання. У нелінійних моделях (деревні чи ансамблі) формула складніша, але суть залишається: модель будує внутрішню математичну структуру, яка відображає залежність $X \to y$.

- **Оцінка якості.** Навчена модель тестується на невикористаних для тренування даних, обчислюються метрики точності (наприклад, середньоквадратична похибка або коефіцієнт детермінації). Якщо точність недостатня, може бути проведено повторне навчання з іншими гіперпараметрами або відбір найважливіших ознак.

- **Прогнозування.** Після успішного навчання модель приймає на вхід нові дані *X* (параметри сценарію) і видає прогнозовані потреби *y*. Наприклад, якщо надходять дані про поточний склад бригади, коефіцієнти інтенсивності та сезон проведення операції, модель обчислить очікувані витрати ресурсів, наприклад пального і боєприпасів, використовуючи набуті під час навчання залежності.

На відміну від регламентної (rule-based) моделі, яка використовує фіксовані формули та встановлені норми, ML-модель **навчається на історичних даних** і враховує більше факторів одночасно. Rule-based підхід спирається на заздалегідь задану формулу виду «норма × кількість × коефіцієнт», де всі складові визначаються експертно. Натомість ML-модель може врахувати нетривіальні взаємозв'язки між ознаками. Наприклад, вона може виявити, що в певних комбінаціях типів техніки і погодних умов витрати зростають нерівномірно, що за статичного підходу врахувати складно. ML автоматично скоригує ваги (коефіцієнти) у прогнозній формулі так, щоб мінімізувати помилку на історії.

Використання машинного навчання стає доцільним, коли: - **Наявні великі обсяги історичних даних.** Наприклад, зібрані дані про реальні споживання ресурсів різними бригадами у різні сезони і ситуації. Чим більше даних — тим точніша модель. - **Бажаємо враховувати складні нелінійні ефекти.** Якщо залежності між факторами і витратами складні (запаси вливаються один в одного, є сезонні сплески, ефект кумуляції), модель ML може їх навчитися. - **Необхідна адаптивність.** ML-модель може перенавчатися на нових даних (змінивши ваги) без переписування формул; rule-based підхід потребує ручної корекції норм.

Приклади ознак (факторів), які може враховувати ML-модель при прогнозі, включають: - *Структурні ознаки:* кількість одиниць техніки кожного типу в бригаді, чисельність особового складу. - *Тактичні ознаки:* коефіцієнт інтенсивності (стандартний, навчання, бойовий) та тривалість операції (дні). - *Ознаки середовища:* тип місцевості (пустеля, ліс, гори), погодні умови, **змінна «сезон»** (зима/літо/перехідний). - *Історичні ознаки:* середні минулі витрати ресурсів за аналогічних умов у попередні періоди або бойових зіткненнях.

Наприклад, змінна **«season»** може кодуватися числом чи набором бінарних ознак (зимовий/літній період) і моделювати підвищене споживання пального в холодну пору року через прогрівання двигунів або сповільнення руху техніки. Формально роль сезону у прогнозній формулі може виглядати так: $\hat{y} = \beta_0 + \beta_1 \cdot (\textit{кількість техніки}) + \beta_2 \cdot (\textit{інтенсивність}) + \beta_3 \cdot (\textit{сезон}) + \cdots$. Наприклад, якщо $season = 1$ для зими і 0 для літа, коефіцієнт β_3 покаже, наскільки зросла середня потреба в ресурсах через зимові умови. У нелінійних методах сезон може враховуватися через змінні, що дозволяють

немонотонні залежності (наприклад, деревоподібна модель може окремо прогнозувати для кожної пори року).

Таким чином, ML-модель – це не просто ще один «калькулятор» із фіксованою формулою. Головна відмінність у тому, що її **прогнозуюча функція** формується на основі аналізу даних і може приймати більш загальний вигляд (наприклад, дерева рішень або нейронної мережі), яка **самостійно знаходить оптимальні математичні співвідношення** між X та y. За сталої кількості вхідних даних вона може дати різні результати, якщо модель перенавчена на новій інформації, тоді як жорсткі нормативи завжди призводять до одного результату. Завдяки цьому ML-модель може коригуватись і підлаштовуватись під нові умови, що важливо для військової логістики, де ситуація може змінюватися.

Розділ 6.4.3. Технічна реалізація ML-моделі прогнозування

Для впровадження машинного навчання в систему прогнозування логістичних потреб використовується мова програмування **Python** та бібліотека **scikit-learn**. Цей інструментарій дозволяє швидко реалізувати повний pipeline машинного навчання – від завантаження даних до отримання прогнозів – і легко інтегрувати модель у веб-застосунок. Технічна реалізація складається з кількох основних етапів:

- **Збір і підготовка даних.** Створюється набір даних для навчання: історичні записи про споживання ресурсів за попередні періоди разом з відповідними параметрами сценаріїв (властивості бригади, інтенсивність, сезон тощо). Дані можна отримувати безпосередньо з бази даних веб-системи або з зовнішніх джерел. На цьому кроці виконується очищення даних (видалення аномалій, заповнення пропусків) та зведення їх до єдиного формату. Наприклад, стовпчики можуть містити: num_tanks, personnel_count, intensity, season, terrain_type, а цільовий стовпчик – fuel_consumption.

- **Формування ознак.** На основі зібраних даних формується матриця ознак *X* та вектор відповідей *y*. Категоріальні змінні (наприклад, terrain_type чи season) кодуються у вигляді числових (one-hot чи порядкових) полів. При цьому можна також створити похідні ознаки: наприклад, добуток num_tanks * intensity, або ввести лаг-змінні (споживання попереднього дня) для збереження часових залежностей.

- **Навчання моделі.** Обирається алгоритм зі sklearn (наприклад, RandomForestRegressor, GradientBoostingRegressor, LinearRegression тощо) і здійснюється навчання на підготовлених даних. За допомогою функцій бібліотеки ми розбиваємо дані на тренувальну та тестову вибірки, ініціалізуємо модель з необхідними гіперпараметрами і виконуємо метод fit(X_train, y_train). Після тренування модель

зберігається (наприклад, засобами joblib.dump) у вигляді об’єкта, готового до прогнозування.

- **Оцінка та налаштування.** Проводиться оцінка точності роботи моделі на тестовій вибірці – обчислюється метрика (MSE, R2 тощо). Якщо необхідно, здійснюється підбір кращих гіперпараметрів або методів відбору ознак. Наприклад, можна перебирати кількість дерев у випадковому лісі (n_estimators) або максимальну глибину дерева (max_depth) для пошуку оптимального результату.
- **Прогнозування нових сценаріїв.** Коли адміністрація системи вводить новий сценарій (наприклад, параметри майбутньої операції), дані передаються до ML-моделі. Код на Python (у вигляді скрипта або сервісу) приймає вхідні значення ознак та застосовує метод predict() навченого об’єкта. На виході отримуємо прогнозовані кількості ресурсів для кожного виду (паливо, боєприпаси, тощо).

Нижче наведено спрощений **JSON-приклад** даних, які надсилаються до сервісу прогнозування (або передаються через API) та отримуються з нього у відповідь:

```
{
 "model": "ML",
 "scenario": {
  "brigade_id": 3,
  "num_tanks": 10,
  "num_personnel": 500,
  "intensity": 1.2,
  "season": "Winter",
  "days": 7
 },
 "predictions": {
  "fuel_consumption": 12500,
  "ammo_consumption": 4800,
  "food_consumption": 3500
 }
}
```

У цьому прикладі поле "model": "ML" вказує вибір машинонавчальної моделі (альтернативою може бути "Baseline" для регламентного підходу). Сценарій містить усі необхідні ознаки, а predictions – прогнозовані витрати ресурсів.

Архітектурно система може бути побудована так, що сам ML-процес виконується в окремому сервісному компоненті або модулі. Наприклад, Laravel-застосунок може викликати Python-скрипт через консоль або REST-запит до спеціального мікросервісу прогнозування. Як тільки користувач вказує в інтерфейсі, що потрібно використати ML-модель (через

налаштування або прапорець у веб-формі), застосунок передає дані до ML-сервісу. Цей сервіс обробляє дані (підготовка ознак, масштабування), застосовує модель та повертає результат у вигляді JSON. Якщо ж обрано rule-based модель, дані передаються назад у класичний a ForecastService, що використовує нормативні формули.

Перемикання між моделями організовано гнучко: у коді наявний єдиний інтерфейс або фабрика прогнозування. У простому варіанті, в конфігураційному файлі (або у виборі користувача) встановлюється параметр forecast_model = 'baseline' або 'ml'. Наприклад, у псевдокоді:

```
if (config('forecast_model') == 'ml') {
   $forecast = MLForecastCalculator::calculate($scenario);
} else {
   $forecast = BasicForecastCalculator::calculate($scenario);
}
```

Таким чином адміністратор може переключитися між режимами без змін у основній логіці системи: базова модель гарантує прозорість і швидкість, а ML-модель (при наявності даних) підвищує точність прогнозів.

Наприкінці порівняємо ключові аспекти використання rule-based та ML-підходів у формі таблиці:
Таблиця 6.4. Порівняння rule-based та ML

Критерій	Rule-Based (регламентна модель)	Машинне навчання (ML)
Точність прогнозів	Зазвичай обмежена встановленими нормами; може пропускати нелінійні ефекти.	Вища при наявності якісних даних; вчиться на реальних витратах, враховує складні закономірності.
Гнучкість	Низька: зміни (нові фактори) треба додавати вручну у формули.	Висока: модель можна перенавчати на нових даних; автоматично пристосовується до нових умов.
Зрозумілість	Висока: логіка прозора, формули явні, легко пояснити, як розраховуються результати.	Низька/середня: алгоритми можуть бути «чорним ящиком» (особливо складні), важче інтерпретувати

Критерій	Rule-Based (регламентна модель)	Машинне навчання (ML)
		внутрішні залежності.
Необхідний обсяг даних	Невеликий: достатньо експертних нормативів та коефіцієнтів.	Великий: потрібні історичні дані про фактичне споживання та параметри бойових дій/умов.
Складність впровадження	Низька: проста реалізація у вигляді арифметичних формул, не потребує окремих модулів ML.	Вища: потребує побудови повного ML-пайплайну (скрипти на Python, бібліотеки, інтеграція через API), а також обслуговування (оновлення даних, перенавчання).

Ця таблиця демонструє, що rule-based підхід підходить для сценаріїв із мінімальною доступністю даних та необхідністю високої прозорості, тоді як ML-моделі виправдані, коли є багаті історичні дані і потрібно максимізувати точність прогнозу навіть за складних нелінійних взаємозв'язків між факторами.

6.5 Фрагменти програмної реалізації веб-застосунку

У цьому підрозділі наведено фрагменти програмної реалізації розробленого веб-застосунку на базі фреймворку Laravel. Кодові приклади ілюструють ключові аспекти побудови системи: моделювання даних, реалізацію бізнес-логіки прогнозування, REST-інтерфейси, ініціалізацію бази даних та підготовку даних для візуалізації результатів за допомогою Chart.js.

6.5.1. Моделі даних та зв'язки між сутностями

Нижче наведено спрощені моделі Brigade, Resource, Equipment та ConsumptionNorm, які відображають основні сутності предметної області та їх зв'язки в базі даних.

```php
<?php

namespace App\Models;

use Illuminate\Database\Eloquent\Model;
use Illuminate\Database\Eloquent\Relations\HasMany;

class Brigade extends Model
{
    protected $fillable = [
        'name',
        'personnel_count',
    ];

    public function equipment(): HasMany
    {
        return $this->hasMany(Equipment::class);
    }

    public function forecasts(): HasMany
    {
        return $this->hasMany(Forecast::class);
    }
}
```

```php
<?php

namespace App\Models;

use Illuminate\Database\Eloquent\Model;
use Illuminate\Database\Eloquent\Relations\HasMany;

class Resource extends Model
{
    protected $fillable = [
        'code',
        'name',
        'category',   // fuel, ammo, food, medical, etc.
        'unit',       // l, kg, pcs
    ];

    public function norms(): HasMany
    {
        return $this->hasMany(ConsumptionNorm::class);
    }
```

```
}
```

```
<?php

namespace App\Models;

use Illuminate\Database\Eloquent\Model;
use Illuminate\Database\Eloquent\Relations\BelongsTo;

class Equipment extends Model
{
    protected $fillable = [
        'brigade_id',
        'name',
        'type',             // tank, IFV, truck, etc.
        'quantity',
    ];

    public function brigade(): BelongsTo
    {
        return $this->belongsTo(Brigade::class);
    }
}
```

6.5.2. Сервіс прогнозування логістичних потреб

Сервісний клас ForecastService інкапсулює базовий алгоритм прогнозування за формулою «норма × кількість техніки × інтенсивність × тривалість». Передбачений інтерфейс для потенційної ML-реалізації.

```
<?php

namespace App\Services;

use App\Models\Brigade;
use App\Models\Resource;
use App\Models\ConsumptionNorm;
use Illuminate\Support\Collection;

class ForecastService
{
    /**
     * Розрахунок потреб бригади за базовою нормативною моделлю.
     *
     * @param  Brigade  $brigade
     * @param  int      $days
```

```
     * @param  float    $intensityFactor   // коеф. інтенсивності бою (1.0 = стандарт)
     * @param  float    $environmentFactor // коеф. умов (зима, бездоріжжя тощо)
     * @return Collection<Resource, float>  // ресурс -> потрібна кількість
     */
    public function calculateBaseline(
        Brigade $brigade,
        int $days,
        float $intensityFactor,
        float $environmentFactor
    ): Collection {
        $result = collect();

        foreach ($brigade->equipment as $equipment) {
            $norms = ConsumptionNorm::where('equipment_type', $equipment->type)->get();

            foreach ($norms as $norm) {
                /** @var Resource $resource */
                $resource = $norm->resource;

                $base = $norm->base_amount_per_day;
                $amount = $base
                    * $equipment->quantity
                    * $days
                    * $intensityFactor
                    * $environmentFactor;

                $result[$resource->id] = ($result[$resource->id] ?? 0) + $amount;
            }
        }

        return $result;
    }

    /**
     public function calculateWithMl(Brigade $brigade, array $scenario): Collection
    {
        // TODO: інтеграція з ML-службою (наприклад, через HTTP API)
        return $this->calculateBaseline(
            $brigade,
            $scenario['days'] ?? 7,
            $scenario['intensity'] ?? 1.0,
            $scenario['environment'] ?? 1.0
```

```
        );
    }
}
```

6.5.3. Контролер та REST-інтерфейс для розрахунку прогнозу

Контролер ForecastController приймає параметри сценарію з HTTP-запиту, викликає сервіс прогнозування та зберігає результат у таблицях forecasts та forecast_items.

```
<?php

namespace App\Http\Controllers;

use App\Models\Brigade;
use App\Models\Forecast;
use App\Models\ForecastItem;
use App\Services\ForecastService;
use Illuminate\Http\Request;

class ForecastController extends Controller
{
    public function __construct(
        protected ForecastService $forecastService
    ) {}

    /**
     * POST /api/forecasts
     */
```

```
public function store(Request $request)
{
    $data = $request->validate([
        'brigade_id'        => ['required', 'exists:brigades,id'],
        'days'              => ['required', 'integer', 'min:1'],
        'intensity_factor'  => ['required', 'numeric', 'min:0.1'],
        'environment_factor'=> ['required', 'numeric', 'min:0.1'],
        'mode'              => ['required', 'in:baseline,ml'],
    ]);

    $brigade = Brigade::findOrFail($data['brigade_id']);

    $forecast = Forecast::create([
        'brigade_id' => $brigade->id,
        'days'       => $data['days'],
        'mode'       => $data['mode'],
    ]);

    $resources = $data['mode'] === 'ml'
        ? $this->forecastService->calculateWithMl($brigade, $data)
        : $this->forecastService->calculateBaseline(
            $brigade,
            $data['days'],
            $data['intensity_factor'],
            $data['environment_factor']
```

```
            );

        foreach ($resources as $resourceId => $amount) {
            ForecastItem::create([
                'forecast_id' => $forecast->id,
                'resource_id' => $resourceId,
                'amount'      => $amount,
            ]);
        }

        return response()->json([
            'id'       => $forecast->id,
            'message'  => 'Forecast created',
        ], 201);
    }
}
```

6.5.4. Міграції та початкові дані (сидери)

Фрагмент міграції для таблиці forecasts та forecast_items, а також приклад сидера для створення тестових даних.

```
<?php

use Illuminate\Database\Migrations\Migration;
use Illuminate\Database\Schema\Blueprint;
use Illuminate\Support\Facades\Schema;
```

```
class CreateForecastsTables extends Migration
{
    public function up(): void
    {
        Schema::create('forecasts', function (Blueprint $table) {
            $table->id();
            $table->foreignId('brigade_id')->constrained()->cascadeOnDelete();
            $table->unsignedInteger('days');
            $table->string('mode'); // baseline | ml
            $table->timestamps();
        });

        Schema::create('forecast_items', function (Blueprint $table) {
            $table->id();
            $table->foreignId('forecast_id')->constrained()->cascadeOnDelete();
            $table->foreignId('resource_id')->constrained()->cascadeOnDelete();
            $table->decimal('amount', 15, 3);
            $table->timestamps();
        });
    }

    public function down(): void
    {
        Schema::dropIfExists('forecast_items');
```

```
        Schema::dropIfExists('forecasts');
    }
}
<?php

namespace Database\Seeders;

use App\Models\Brigade;
use App\Models\Equipment;
use App\Models\Resource;
use App\Models\ConsumptionNorm;
use Illuminate\Database\Seeder;

class DemoDataSeeder extends Seeder
{
    public function run(): void
    {
        $brigade = Brigade::create([
            'name'            => 'Механізована бригада №1',
            'personnel_count' => 3500,
        ]);

        Equipment::create([
            'brigade_id' => $brigade->id,
            'name'       => 'Танк Т-64',
```

```
            'type'       => 'tank',
            'quantity'   => 30,
        ]);

        $diesel = Resource::create([
            'code'     => 'FUEL_DIESEL',
            'name'     => 'Дизельне пальне',
            'category' => 'fuel',
            'unit'     => 'l',
        ]);

        ConsumptionNorm::create([
            'equipment_type'      => 'tank',
            'resource_id'         => $diesel->id,
            'base_amount_per_day' => 600, // літрів на танк за добу бою
        ]);
    }
}
```

6.5.5. Підготовка даних для візуалізації прогнозів у Chart.js

Метод контролера, який завантажує прогноз, агрегує дані за категоріями ресурсів та передає їх у подання Blade у форматі, зручному для побудови діаграми у Chart.js.

```
<?php

namespace App\Http\Controllers;
```

```
use App\Models\Forecast;
use Illuminate\View\View;

class ForecastViewController extends Controller
{
    /**
     * GET /forecasts/{forecast}
     */
    public function show(Forecast $forecast): View
    {
        $items = $forecast->items()
            ->with('resource')
            ->get()
            ->groupBy(fn ($item) => $item->resource->category)
            ->map(fn ($group) => $group->sum('amount'));

        $labels = $items->keys()->values();      // категорії ресурсів
        $data   = $items->values()->map(fn ($v) => round($v, 2)); // значення

        return view('forecasts.show', [
            'forecast' => $forecast,
            'chartLabels' => $labels,
            'chartData'   => $data,
        ]);
```

```
    }
}
```

Фрагмент Blade-шаблону, який використовує передані дані для побудови діаграми:

```
<canvas id="resourcesChart"></canvas>


<script src="https://cdn.jsdelivr.net/npm/chart.js"></script>
<script>
    const ctx = document.getElementById('resourcesChart').getContext('2d');


    new Chart(ctx, {
        type: 'bar',
        data: {
            labels: @json($chartLabels),
            datasets: [{
                label: 'Потреба за категоріями ресурсів',
                data: @json($chartData),
            }]
        },
        options: {
            responsive: true,
            scales: {
                y: { beginAtZero: true }
            }
```

```
    }
  });
</script>
```

Розділ 7. Економічне обґрунтування проекту

7.1. Обґрунтування доцільності розробки системи

Сучасні військові операції створюють надзвичайно складні логістичні умови, що вимагають гнучких та цифрово підтриманих рішень. Військовий конфлікт в Україні зумовив необхідність модернізації підходів до логістики та переходу від традиційних централізованих схем забезпечення до гнучких, децентралізованих систем з використанням ІТ. Відсутність ефективної системи прогнозування логістичних потреб призводить до надлишкових запасів або дефіциту ресурсів, нераціонального використання транспорту й обладнання, затримок постачання та зростання витрат. Неефективне планування ресурсів також сприяє збільшенню людських помилок та переплатам за термінові закупівлі. Як свідчить практика, впровадження автоматизованих систем підтримки прийняття рішень дозволяє значно скоротити експлуатаційні витрати (до 20–30%) і підвищити оперативність процесів, забезпечуючи швидший аналіз даних і зменшуючи кількість помилок.

Ключові проблеми, які вирішує розроблювана система:

- Відсутність точного прогнозування потреб призводить до закупівлі надмірних запасів амуніції, пального та медичних засобів.
- Несвоєчасне забезпечення підрозділів ресурсами призводить до простоїв і зриву планових операцій.
- Ручне планування створює високе навантаження на логістичний персонал і багаторазову обробку звітної інформації.
- Брак аналітичних інструментів унеможливлює швидке та обґрунтоване прийняття рішень в умовах бойової невизначеності.

Автоматизація логістичних процесів сприяє оптимізації управлінських процедур та зменшенню витрат на обробку даних. Розробка web-системи підтримки рішень для прогнозування військових потреб дозволить не лише

вирішити вказані проблеми, а й звільнити ресурси для критично важливих завдань, підвищуючи загальну ефективність забезпечення підрозділів та обороноздатність країни.

7.2. Розрахунок витрат на розробку

Загальні витрати на розробку програмного забезпечення (K) визначаються сумою витрат на безпосередню розробку (проектування, програмування, тестування) (K_1) та витрат на відлагодження і пробну експлуатацію (K_2):

$$K = K\_1 + K\_2$$

До складу витрат на розробку (K_1) входять: оплата праці розробників, соціальні відрахування до фондів, купівля обладнання та компонентів, накладні витрати (управління проектом, оренда тощо) та інші виробничі витрати. Витрати на оплату праці (Z) розраховуються як сума добутків числа працівників, їхньої тарифної ставки та трудомісткості (кількості робочих днів):

$$Z = \sum_{(i)} n_i \cdot (O_i \cdot D_i) / D_m$$

де (n_i) – кількість працівників (i)-ої категорії, (O_i) – місячний оклад, (D_i) – дні, затрачені одним працівником, (D_m) – середня кількість робочих днів у місяці (наприклад, 22).

Таблиця 7.1. Вихідні дані і розрахунок витрат на оплату праці розробників

№	Посада	К-ть осіб	Місячний оклад, грн	Дні на проект	Витрати на оплату праці, грн
1	Програміст	2	25 000	66	150 000
2	Аналітик	1	20 000	66	60 000
3	Тестувальник	1	18 000	22	18 000

	Разом:				**228 000**

Загальна сума базової зарплати за цей період склала 228 000 грн. До неї додаються нарахування та інші витрати:

- **Відрахування до фондів (22%)** – 50 160 грн,
- **Накладні витрати** – приймаються на рівні 150% від зарплатного фонду:

$$H = 1.5 \cdot 228\,000 = 342\,000$$

- **Інші витрати (10%)** – 22 800 грн (ремонт, тестове обладнання тощо).

Таким чином, витрати на розробку (проектування та програмування) (K_1 = 228,000 + 50,160 + 342,000 + 22,800 = 642,960) грн. Витрати на відлагодження і пілотну експлуатацію (K_2) оцінюємо в ≈ 20 000 грн.
Загальні витрати на розробку системи:

$$K = K_1 + K_2 \approx 662\,960\ грн$$

7.3. Витрати на впровадження та експлуатацію

Після завершення розробки система вимагає додаткових витрат на впровадження, налаштування, навчання користувачів та підтримку. Основні категорії цих витрат:

- **Технічна інфраструктура:** серверне обладнання або оренда віртуального сервера. Наприклад, потрібен виділений сервер вартістю ≈ 100 000 грн.
- **Навчання персоналу:** навчання 5 співробітників із середньою вартістю курсу 6 000 грн складе ≈ 30 000 грн.
- **Технічна підтримка:** річні витрати на підтримку можуть бути 10–20% вартості впровадження (приблизно 20 000–50 000 грн).

Таблиця 7.2. Витрати на впровадження та експлуатацію

Стаття витрат	**Одиниця**	**Вартість, грн**
Сервер / Хостинг (1 рік)	1 шт.	100 000
Навчання персоналу (5 чол.)	1 курс/5 осіб	30 000
Технічна підтримка (1 рік)	1 рік	30 000
Разом (орієнтовно)		**160 000**

Витрати на впровадження та перший рік експлуатації оцінюються приблизно в 160 000 грн. Таким чином, **сукупні інвестиції** у систему (розробка + впровадження) складатимуть близько **822 960 грн**.

7.4. Оцінка економічного ефекту

Основні джерела економічного ефекту:

- **Зменшення надмірного споживання ресурсів.** Завдяки точним прогнозам знижуються втрати. Наприклад, прогнозування дозволяє скоротити річне споживання пального на 20–30%, що дає зекономлені 20 000 л/рік. При собівартості пального ~60 грн/л це ≈ 1 200 000 грн на рік.

- **Економія часу логістів.** Якщо річне робоче навантаження скорочується на 200 годин, при тарифі 200 грн/год ця економія становить 40 000 грн/рік.

- **Зниження людських помилок.** Автоматизовані системи значно

зменшують кількість помилкових рішень.

Таблиця 4.1. Оцінка річної економії ресурсів (приклад)

№	Заходи економії	До впровадження	Після	Заощаджено (в одиницях)	Економія, грн/рік
1	Споживання пального, л	100 000	80 000	20 000 (20%)	(20,000 * 60 = 1,200,000) грн
2	Робочі години логіста, годин	1 000	800	200 (20%)	(200 * 200 = 40,000) грн
Разом:	–	–	–	–	**1 240 000 грн/рік**

7.5. Розрахунок строку окупності

$$T_{ок} = \frac{K_{заг}}{E_{річ}} = \frac{822960}{1240000} \approx 0.66$$

7.6 Висновки

Розробка веб-системи підтримки прийняття рішень для прогнозування логістичних потреб військових підрозділів є обґрунтованою та актуальною. Орієнтовні витрати на розробку та впровадження проекту становлять близько 823 тис. грн. Економічний ефект від впровадження проекту полягає в економії паливно-мастильних матеріалів, оптимізації використання персоналу та зниженні кількості помилок. Теоретичні оцінки показують можливе скорочення експлуатаційних витрат на 20–30%. Наприклад, за розрахунками, щорічна економія може перевищувати 1,24 млн грн лише за рахунок зниження витрат пального і часу логістів. Розрахунковий строк

окупності системи (≈ 0,66 року) дозволяє вважати проект інвестиційно привабливим. У результаті проект забезпечить підвищення продуктивності логістичних процесів, оптимізує бізнес–процеси забезпечення військ і сприятиме загальному зростанню обороноздатності.

Висновок: Розробка та впровадження зазначеної web-системи є економічно обгрунтованим рішенням. Незважаючи на значні початкові вкладення, очікуваний економічний ефект внаслідок зменшення витрат і зростання ефективності обґрунтовує проект як вигідний у довгостроковій перспективі.

Висновки

У ході виконання дипломної роботи було проведено комплексне дослідження проблематики прогнозування матеріально-технічних потреб військових підрозділів та розроблено концепцію програмної системи для підтримки прийняття рішень у цій сфері. Проаналізувавши структуру та особливості функціонування військової логістики, визначено ключові фактори, що впливають на ефективність забезпечення військ – зокрема, важливість проактивного (прогнозного) підходу замість реактивного реагування на дефіцити. Огляд сучасних досліджень та технологій показав, що провідні армії світу активно впроваджують системи аналізу даних і штучного інтелекту, здатні істотно підвищити точність прогнозів та швидкість логістичних рішень.

В рамках роботи сформульовано вимоги до програмного продукту, що забезпечує автоматизацію процесу прогнозування. Вимоги охоплюють як функціональні аспекти (конкретні функції системи: розрахунок потреб, виявлення дефіцитів, рекомендації з розподілу), так і нефункціональні (надійність, безпека, зручність, інтеграція), враховуючи умови військового застосування.

На основі вимог здійснено проектування системи: запропоновано модульну архітектуру з виділенням компонентів для роботи з даними, бізнес-логіки прогнозування, інтерфейсу користувача та зовнішніх API. Розроблена ER-модель даних відображає основні сутності домену – підрозділи, ресурси, нормативи, запаси, прогнози тощо – і їх взаємозв’язки, що забезпечить створення ефективної бази даних для системи. Представлені у роботі діаграмні рішення (архітектури і даних) є концептуальними шаблонами, які можуть бути деталізовані на етапі реалізації.

Практична цінність отриманих результатів полягає у тому, що запропонована система здатна підвищити оперативність і обґрунтованість рішень у військовій логістиці. Завдяки їй офіцери зможуть швидко оцінювати потреби у ресурсах для різних сценаріїв, заздалегідь виявляти потенційні "вузькі місця" в забезпеченні та вживати заходів для їх нейтралізації. Це особливо актуально в умовах сучасних бойових дій, де темп операцій дуже високий і ціна помилки в забезпеченні – надзвичайно велика.

Подальший розвиток проекту може включати реалізацію дослідного зразка системи та його тестування на сценаріях, наближених до реальних. Цікавою задачею є інтеграція алгоритмів машинного навчання – наприклад, навчання моделі на історичних даних про бої на Донбасі для прогнозування витрати боєприпасів. Також необхідно врахувати особливості впровадження системи в існуючу інфраструктуру ЗСУ, зокрема, її сумісність з натівськими стандартами та національними системами (типу "Дозор" чи інші системи управління).

Отже, поставлені в роботі завдання виконані: проведено предметне дослідження, узагальнено передовий досвід, визначено вимоги та спроектовано програмну систему. Реалізація запропонованої СППР у перспективі сприятиме підвищенню ефективності військової логістики та здатності Збройних Сил забезпечувати свої підрозділи усім необхідним вчасно і в повному обсязі.

Список використаних джерел

1. NATO Logistics Handbook. – NATO Standardization Office, 2020.

2. Van Creveld, M. Supplying War: Logistics from Wallenstein to Patton. – Cambridge University Press, 2004.

3. Фіногенов А.І. Військова логістика: підручник. – Київ: НАОУ, 2020.

4. Бутко М.І., Чичкань С.В. Основи військової логістики. – Харків: ХНУПС, 2016.

5. Адаменко В.Ф. Логістика. Теорія та практика. – КНЕУ, 2019.

6. Goodfellow, I., Bengio, Y., Courville, A. Deep Learning. – MIT Press, 2016.

7. James, G., Witten, D., Hastie, T., Tibshirani, R. An Introduction to Statistical Learning. – Springer, 2021.

8. Alpaydin, E. Introduction to Machine Learning. – MIT Press, 2020.

9. Войтенко В.М. Штучний інтелект і машинне навчання в управлінні. – Київ, 2022.

10. Laravel Documentation. – https://laravel.com/docs

11. Chart.js Documentation. – https://www.chartjs.org/docs/

12. MySQL 8.0 Reference Manual – https://dev.mysql.com/doc/

13. Bootstrap 5 Documentation – https://getbootstrap.com/docs/5.0/

14. REST API Design Guide – https://restfulapi.net/

15. ISO/IEC 25010:2011 – Systems and software engineering — System and software quality models.

16. Fowler, M. Patterns of Enterprise Application Architecture. – Addison-Wesley, 2002.

17. Rozanski, N., Woods, E. Software Systems Architecture: Working with Stakeholders Using Viewpoints and Perspectives. – Addison-Wesley, 2011.

18. Sommerville, I. Software Engineering. – 10th ed., Pearson, 2015.