Міністерство освіти і науки України
Національний технічний університет
«Дніпровська політехніка»

Інститут електроенергетики
(інститут)
Факультет інформаційних технологій
(факультет)
Кафедра інформаційних технологій та комп'ютерної інженерії
(повна назва)

ПОЯСНЮВАЛЬНА ЗАПИСКА
кваліфікаційної роботи ступеня магістра
(бакалавра, спеціаліста, магістра)

студента Чумичова Дениса Дмитровича
(ПІБ)
академічної групи 123м-20-1
(шифр)
спеціальності 123 «Комп'ютерна інженерія»
(код і назва спеціальності)
за освітньо-професійною програмою 123 «Комп'ютерна інженерія»
(офіційна назва)
на тему «Обґрунтування вибору параметрів комп'ютерної системи комплексу роботизованих аптек ТМ «Копійка»»
(назва за наказом ректора)

Керівники	**Прізвище, ініціали**	**Оцінка за шкалою**		**Підпис**
		рейтинговою	**інституційною**	
кваліфікаційної роботи	доц. Бешта Д.О.			
розділів:				
теоретичний розділ	доц. Бешта Д.О.			
синтез системи	доц. Ткаченко С.М.			
розроблення програмного забезпечення	ас. Бешта Л.В.			
Рецензент				
Нормоконтролер	проф. Цвіркун Л.І.			

Дніпро
2022

ЗАТВЕРДЖЕНО:

завідувач кафедри
інформаційних технологій
та комп'ютерної інженерії
(повна назва)

_____ Гнатушенко В.В.
(підпис) (прізвище,ініціали)

«6» вересня 2021 року

ЗАВДАННЯ
на кваліфікаційну роботу
ступеня бакалавр

студента Чумичова Д.Д. **академічної групи** 123м-20-1
(прізвище та ініціали) (шифр)

спеціальності 123 «Комп'ютерна інженерія»
за освітньо-професійною програмою 123 «Комп'ютерна інженерія»
(офіційна назва)

на тему «Обґрунтування вибору параметрів комп'ютерної системи комплексу роботизованих аптек ТМ «Копійка»» затверджену наказом ректора НТУ «Дніпровська політехніка» від 10.12.2021 №1036

Розділ	Зміст	Термін виконання
Стан питання та постановка завдання	На основі матеріалів виробничих практик, інших науково-технічних джерел сформулювати наукове завдання, конкретизувати предмет та мету досліджень	01.10.2021
Теоретичний	Обґрунтувати теоретичну базу розв'язання наукового завдання, якому присвячено роботу	15.10.2021
Синтез системи	Розробка комп'ютерної системи	12.11.2021
Розроблення програмного забезпечення	Розробка програмного забезпечення	10.12.2021
Експериментальний розділ	Проведення і обробка результатів експериментів	30.12.2021
Графічна частина	Графічні результати роботи подати у вигляді рисунків схем таблиць на 10 арк. формату А4.	07.01.2022

Завдання видано _____ доц. Бешта Д.О.
(підпис керівника) (прізвище, ініціали)

Дата видачі 6 вересня 2021 р.

Дата подання до екзаменаційної комісії 10.01.2022

Прийнято до виконання _____ Чумичов Д.Д.
(підпис студента) (прізвище, ініціали)

РЕФЕРАТ

Пояснювальна записка: 148 с., 35 рис., 20 табл., 2 додатки, 19 джерел.

Об’єкт розробки: комп’ютерна система, що виконує функції обліку кількості медикаментів в аптеках, обміну інформацією між аптеками та центральним офісом та реалізує інтерфейси користувачів та фармацевтів.

Мета: створення комп’ютерної системи для контролю кількості медикаментів в аптеках, обміну інформацією між аптеками та центральним офісом з реалізацією інтерфейсів користувачів та фармацевтів.

Розроблена комп’ютерної системи з можливістю гнучкої зміни числа і набору виконуваних функцій шляхом перепрограмування та апаратних оновлень, орієнтована на полегшення логістичних задач з обліку медикаментів та підвищення продуктивності роботи провізорів у аптеках.

Система виконана відкритою і дозволяє здійснювати технічну і програмну модернізацію системи, а так само забезпечує виконання наступних функцій:

– контроль та видачу медикаментів у кожній аптеці;

– зберігає дані про медикаменти аптеки у БД;

– надає фармацевтам інтерфейс для взаємодії з БД та роботизованою шафою аптеки;

– обмін даними між підсистемами аптек та підсистемою центрального офісу.

Розробка комп’ютерної системи виконана відповідно до завдання на кваліфікаційну роботу магістра.

Програмне забезпечення реалізовано у вигляді пакету програм написаних мовою JavaScript і перевірена його робота.

Результати перевірки у вигляді таблиць, графіків описані і наводяться у пояснювальній записці або додатках.

КОМП’ЮТЕРНА СИСТЕМА, КОНТРОЛЬ, МЕДИКАМЕНТИ, АПТЕКА, СКЛАД, МЕРЕЖА

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СИМВОЛІВ, ОДИНИЦЬ, СКОРОЧЕНЬ І ТЕРМІНІВ

Аптека – особлива спеціалізована організація системи охорони здоров'я, що займається виготовленням, фасуванням, аналізом і продажами лікарських засобів.

Роботизована шафа – місце, де у окремих комірках зберігаються різні товари, які завантажуються і розвантажуються за допомогою маніпуляторів, ведеться облік товару, відстежує терміну придатності в автоматичному режимі і збирає замовлення відповідно до команди з касового комп’ютера.

м. – місто.

ВАОС – відділ адміністрування обчислювальних систем.

ВОТІ – відділ обслуговування технічної інфраструктури.

ІМ – імітаційна модель.

РПС – реальні процеси та системи.

ЕОМ – електронна обчислювальна машина.

ПЕОМ – персональна електронна обчислювальна машина.

ПЗ – програмне забезпечення.

АСЗ – автоматизовані системи зберігання.

ВСТУП

Процес обслуговування відвідувача аптеки зсередини значно складніший, ніж здається зі сторони. Кожен раз, отримавши замовлення, провізор повинен знайти товар серед більш ніж 15 000 найменувань і принести його покупцеві. Весь цей процес може зайняти кілька хвилин, а покупці тим часом чекають в черзі. Однак аптеки вже кілька років використовують у своїй роботі фармацевтичного робота, що значно прискорює процес обслуговування. Він значно пришвидшує роботу по пошуку медкаментів.

Такі роботи можуть зберігати в собі одночасно більше 10 000 упаковок, а час видачі замовленого препарату складає менше 10 секунд. У свою чергу, провізор може приділити більше уваги покупцеві, надати детальну консультацію, а не витрачати час на пошук препаратів на складі.

Завдяки впровадженню фармацевтичних роботів в роботизованих аптеках покупці не витрачають багато часу в довгих чергах.

Переваги використання аптечних роботів:

− економія простору для зберігання препаратів. Роботизоване сховище може вмістити більше десяти тисяч коробок на площі в декілька кв.м. зберігання;

− прискорення пошуку препаратів. Робот знаходить і приносить препарат фармацевту в лічені секунди. Це значно прискорює обслуговування клієнтів, скорочує черги і підвищує якість обслуговування і задоволеність клієнтів;

− використання нестандартних просторів. Використання робота для зберігання препаратів дає можливість відкривати аптеки там, де раніше це було неможливо або безглуздо: у невеликих приміщеннях в людних місцях, або в багаторівневих приміщеннях;

− імідж аптеки. Необхідність купувати медикаменти – завжди певний стрес. Але якщо простір аптеки організовано гармонійно, а

обслуговування ведеться на високому рівні, це створює приємну атмосферу – і клієнт буде віддавати перевагу дану аптеку іншим.

Але для того щоб кінцевий покупець міг отримати необхідні медикаменти, потрібно спочатку доставити їх до аптеки. Фармацевтичний ланцюжок поставок – це складна логістична система, кінцевою метою якої є те, щоб відповідні лікарські засоби в установлені строки з дотриманням обов'язкових умов зберігання і транспортування були отримані очікують їх людьми. Аптеки продають щодня тисячі найменувань різних медикаментів, тому дуже важко забезпечити поставки необхідних медикаментів вчасно. Для вирішення цих проблем усі роботизовані аптеки об’єднані в єдину мережу.

Мета дослідження – обґрунтування та вибір апаратних засобів та розробка програмних інтерфейсів комп’ютерної системи комплексу роботизованих аптек ТМ «Копійка».

Завдання дослідження:

– виконати аналіз структури та складу аптечної комп’ютерної системи з роботизованими комплексами ТМ «Копійка»;

– дослідити влаштування та принципи роботи аптечних роботизованих шаф та автоматизованих складів;

– проаналізувати структуру підприємства та скласти вимоги до комп’ютерної системи комплексу роботизованих аптек ТМ «Копійка»;

– на основі проведеного аналізу розробити функціональну та принципову схеми роботизованої шафи, підібрати апаратне забезпечення для системи;

– розробити програмне забезпечення для комплексу роботизованих аптек ТМ «Копійка» з детальною розробкою інтерфейсів користувачів та фармацевтів, підключенням до БД та можливістю читання, запису, редагування та видалення інформації з неї і керуванням роботизованою шафою;

– розробити математичну модель мережі комплексу роботизованих аптек ТМ «Копійка» та отримати параметри роботи мережі у різних умовах;

– на основі отриманих параметрів виявити та скорегувати проблемні вузли.

Об’єкт дослідження – комп’ютерна система комплексу роботизованих аптек ТМ «Копійка».

Предмет дослідження – програмно-технічні засоби та мережевий зв'язок комплексу роботизованих аптек ТМ «Копійка».

Методи дослідження – для досягнення поставлених завдань використовуються елементи теорії масового обслуговування, комп’ютерних мереж, методів наукового дослідження, архітектури комп’ютерів та програмування інтерфейсів мовою JavaScript із бібліотекою React.

Наукові положення:

1. Використання роботизовнаних комплексів зберігання та видачі медикаментів сумісно із програмним інтерфейсом фармацевтів об’єднаних у єдину комп’ютерну систему оптимізує роботу співробітників аптеки, полегшує логістичні задачі обліку медикаментів та документообіг, що позитивно впливає на імідж і прибуток компанії.

2. Підключення великої кількості пристроїв до вузла мережі може викликати на ньому чергу пакетів, що може призвести до уповільнення роботи мережі.

Наукові результати:

1. Розроблений і реалізований у вигляді пакету програм алгоритм функціонування програмного забезпечення керування роботизованою шафою та взаємодією з БД.

2. Розроблені функціональна та принципова схема роботизованої шафи.

3. Розроблена математична модель комп’ютерної мережі та отримати параметри роботи мережі у різних умовах. Згідно отриманих параметрів скореговані параметри вузлів мережі.

1 СТАН ПИТАННЯ І ПОСТАНОВКА ЗАВДАННЯ

1.1 Стисла характеристика галузі та умов застосування системи, що проектується

Кожен мешканець міста повинен мати доступ до медикаментів, тому у таких великих містах, як Дніпро, дуже багато аптек по всьому місту. Для того щоб кожна людина отримала необхідні їй медикаменти важливо організувати вчасну доставку до кожної аптеки. У Дніпропетровській області дуже багато різних мереж аптек, що створює велику конкуренцію у цій сфері. Тому важлива не лише доставка, а й швидке обслуговування клієнтів, що створює гарний імідж для мережі аптек.

1.2 Характеристика і структура об’єкта дослідження

Об’єктом дослідження є мережа аптек ТМ «Копійка». Об’єкт представляє собою багато будівель: центральний офіс та аптеки, розподілені по місту Дніпро. Топологічне розміщення об’єктів у м. Дніпро представлене на рисунку 1.1 (зеленим кольором відмічені аптеки, червоним – центральний офіс). Центральний офіс знаходиться на десятому поверсі багатоповерхової офісної будівлі і складається з декількох кімнат. Топологічна схема розміщення структурних підрозділів компанії в головному офісі представлена на рисунку 1.2. Аптеки розподілені по території міста та мають один поверх. Топологічна схема розміщення структурних підрозділів компанії в аптеках та представлена на рисунку 1.3. Медикаменти у аптеках зберігаються у спеціальних роботизованих шафах. У центральному офісі розташований сервер на якому зберігаються дані компанії та дані по всім аптекам. У аптеках дані зберігаються на комп’ютері завідуючого аптекою. Усі зміни у БД аптеки раз на день автоматично надсилаються до сервера у центральному офісі.

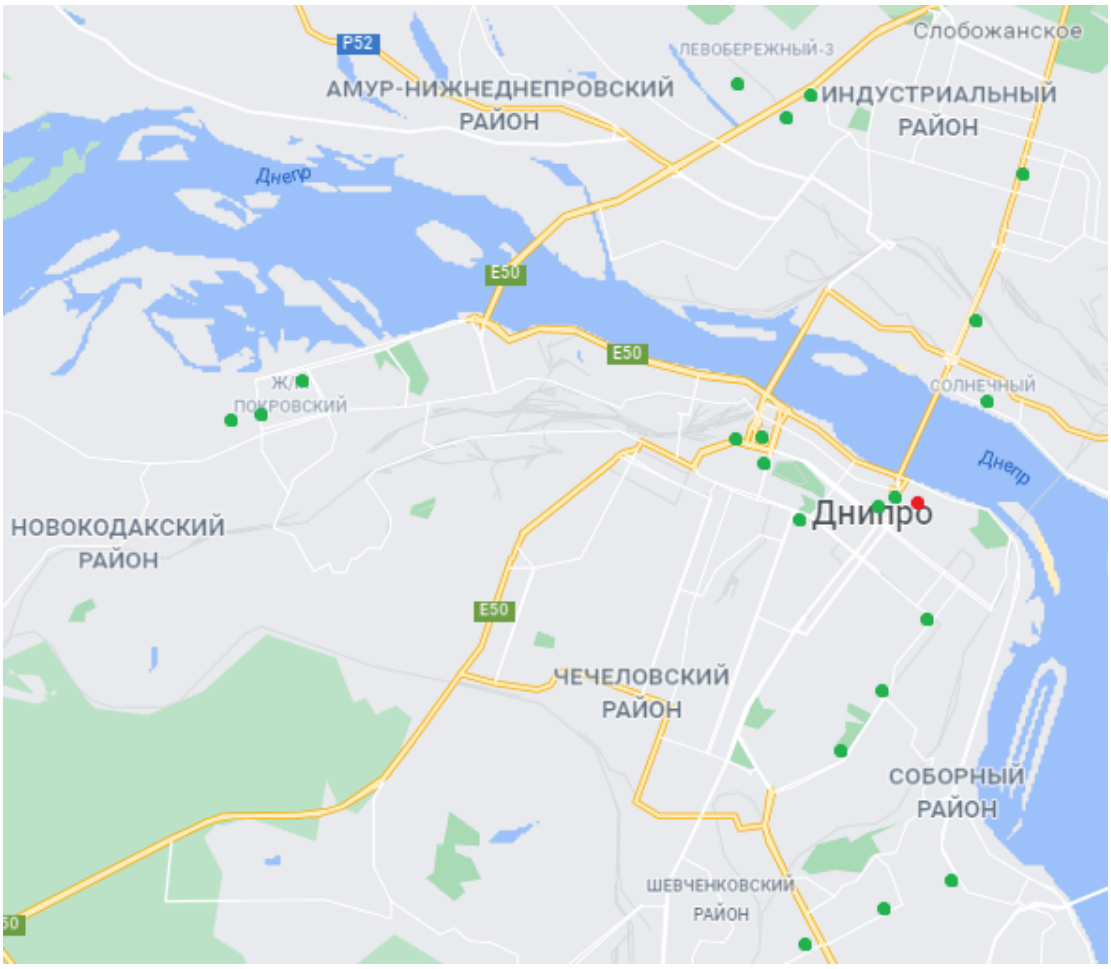


Рисунок 1.1 – Топологічне розміщення об’єктів у м. Дніпро

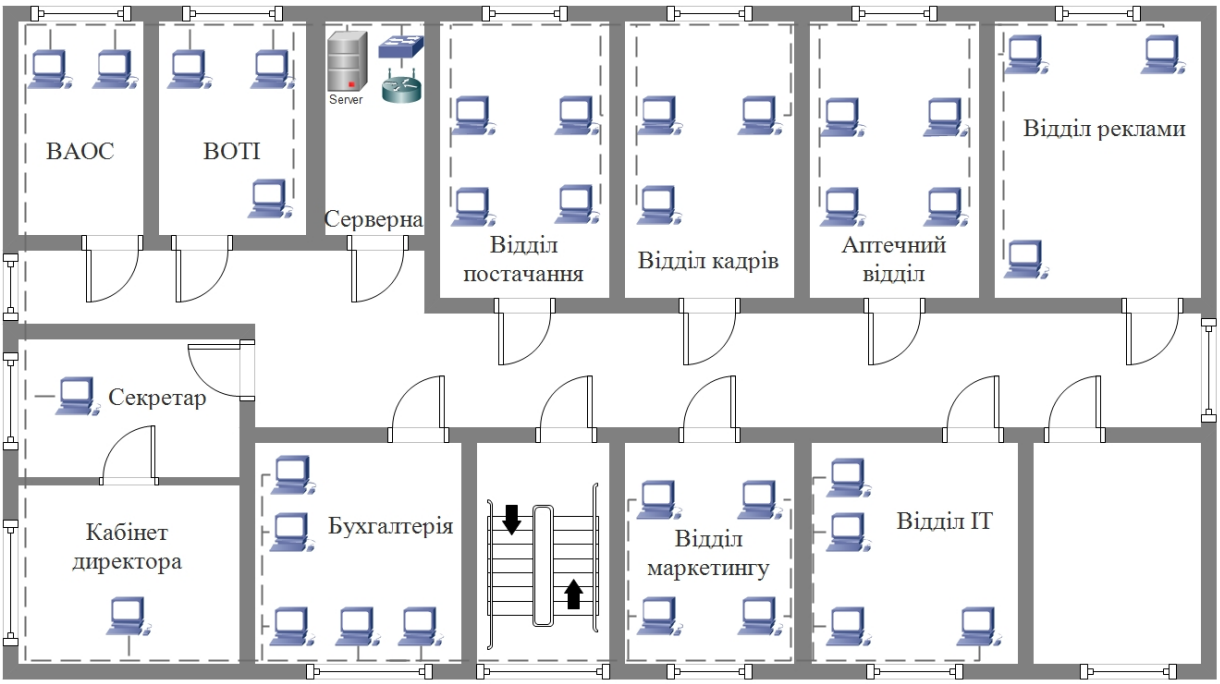


Рисунок 1.2 – Топологічна схема розміщення структурних підрозділів компанії у центральному офісі

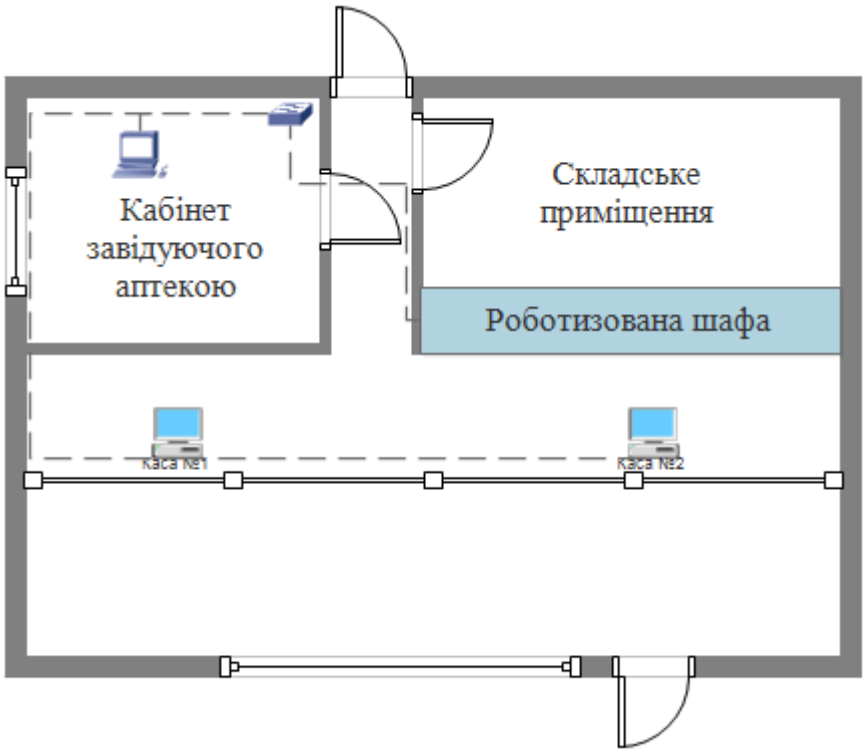


Рисунок 1.3 – Топологічна схема розміщення структурних підрозділів компанії в аптеці

Комплекс роботизованих аптек «Копійка» складається з таких об’єктів:

– центральний офіс – офіс з якого компанія керує усіма аптеками у м. Дніпро;

– аптеки – місця де місцеві жителі можуть отримати медикаменти.

Організаційно центральний офіс поділяється на такі підрозділи:

– дирекція роздрібної торгівлі – займається постачанням аптек необхідними товарами;

– дирекція по маркетингу і рекламі, який поділяється на відділ маркетингу та відділ реклами. Відділ маркетингу займається аналізом ринку та потреб споживачів. Відділ реклами відповідає за рекламну діяльність компанії у місті.

– відділ кадрів керує персоналом та підбирає нових співробітників;

– адміністративно технічна дирекція відповідає за програмну та апаратну складові компанії та складається з: відділу адміністрування обчислювальних систем, який відповідає за правильну роботу комп’ютерної мережі всередині компанії; відділу обслуговування технічної інфраструктури,

який займається ремонтом та профілактикою технічних засобів; відділу IT, який займається розробкою і підтримкою програмного забезпечення; бухгалтерії, яка займається документообігом.

Організаційна структура комплексу наведена на рисунку 1.4.

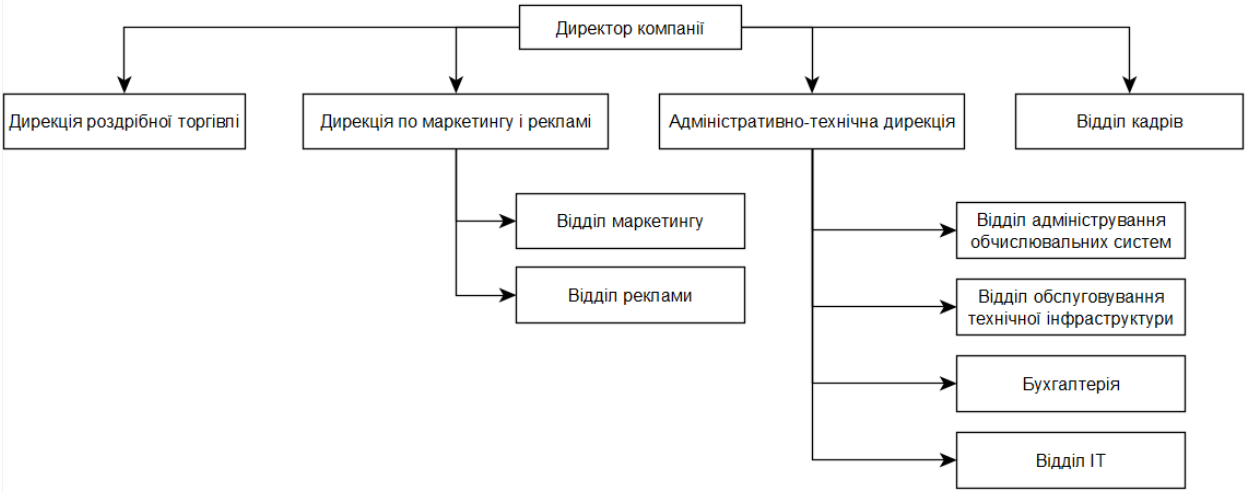


Рисунок 1.4 – Схема організаційної структури компанії

1.3 Принципи, технічні способи та математичні методи інформаційного забезпечення об’єкта впровадження

Кожна аптека і головний офіс представляють собою окрему підмережу, які об’єднані між собою з використанням Internet як транспортного середовище передачі даних, із застосуванням технології побудови VPN тунелів. Кожна підмережа має свій граничний маршрутизатор. У центральному офісі розташований головний сервер на якому зберігатимуться усі дані компанії, а також копії даних з усіх аптек. Дані в аптеках розміщуються локально на комп’ютерах та передаються на сервер у відповідному складі чи офісі.

1.4 Завдання і мета роботи, що виконується

Метою кваліфікаційної роботи є полегшити логістичні задачі з обліку документів та документообіг комплексу роботизованих аптек ТМ «Копійка»,

полегшити та збільшити продуктивність роботи провізорів у аптеках шляхом створення комп’ютерної мережі.

Задачі, які вирішуються у ході кваліфікаційної роботи:

− виконати аналіз топологічного розташування об’єктів мережі аптек ТМ «Копійка» та організаційної структури;

− дослідити влаштування та принципи роботи АСЗ різних типів та обрати тип як базу для розробки власної роботизованої шафи;

− проаналізувати структуру підприємства та скласти вимоги до комп’ютерної системи комплексу роботизованих аптек ТМ «Копійка»;

− на основі проведеного аналізу розробити функціональну та принципову схеми роботизованої шафи, підібрати апаратне забезпечення для системи;

− на основі проведеного підібрати мережеве апаратне забезпечення для системи, розробити функціональну схему структури аптеки, функціональну структуру роботизованої шафи. Підібрати контролер, датчики, виконуючі механізми та необхідні драйвери та на основі обраної елементної бази розробити принципову схему;

− розробити програмне забезпечення для комплексу роботизованих аптек ТМ «Копійка» з детальною розробкою інтерфейсів користувачів та фармацевтів, підключенням до БД та можливістю читання, запису, редагування та видалення інформації з неї і керуванням роботизованою шафою;

− розробити математичну модель мережі комплексу роботизованих аптек ТМ «Копійка» та отримати параметри інтенсивності потоку, середнього часу перебування пакета у вузлі, середньої кількості пакетів які знаходяться у вузлі у звичайному режимі роботи та у режимі та під впливом вірусних програм;

− на основі отриманих параметрів виявити та скорегувати проблемні вузли, перевірити параметри мережі із скорегованими параметрами вузлів.

1.5 Визначення можливих напрямків рішення поставлених завдань

Об’єднання всіх об’єктів компанії в єдину комп’ютерну систему припускає створення єдиної мережі. Об'єднання локальних мереж аптек та центрального офісу в єдину корпоративну мережу організації може здійснюватися:

– з використанням дротових мереж передачі даних;

– з використанням бездротових мереж (технологія Wi-Fi, сотовий зв’язок та ін.);

– з використанням Internet як транспортного середовище передачі даних, із застосуванням технології побудови VPN тунелів.

Для організації автоматизованих систем зберігання існує багато готових варіантів різних типів та розмірів. Деякі компанії пропонують розробку АСЗ для рішення конкретних задач у заданих приміщеннях.

1.6 Обґрунтування вибраного напрямку інженерного рішення

Найкращим варіантом організації мережі буде використання для зв’язку між об’єктами компанії мережі Internet. Кабелі мережі Internet розповсюджені по всьому місту, тому підключення до неї є простішим і дешевшим способом, ніж прокладання нових кабелів, та більш надійним ніж використання бездротових мереж.

АСЗ вирішено розробляти власноруч, так як готові рішення не відповідають поставленим вимогам, а замовлення розробки у спеціалізованих компаній є дуже дорогим з фінансової точки зору. Додатково обслуговування таких систем можливе тільки компаніями розробниками і також є не дешевим заходом.

2 ТЕОРЕТИЧНІ РОЗДІЛИ

2.1 Автоматизовані системи зберігання

Автоматизована система зберігання (АСЗ) – це комплекс, що складається зі стелажів та спеціальних підйомно-транспортних пристроїв, який дозволяє проводити розміщення та збирання вантажів без присутності людини у місці операції [4].

АСЗ забезпечують:

- ефективне використання площі приміщення;
- раціональне використання робочого часу персоналу;
- збереження товарів;
- точність видачі замовлень;
- контроль товарів (кількість та строки придатності);
- швидке виконання інвентаризації.

У більшості випадків АСЗ розробляються індивідуально в залежності від їх задач. Але переважна більшість із них заснована на механізмах наступних типів:

- ліфтовий;
- карусельний;
- гравітаційний.

2.1.1 Ліфтові АСЗ

Ліфтові АСЗ призначені для зберігання окремих предметів та коробок і представляють собою єдиний масив стелажів з комірками між якими вільно переміщується кран-штабелер (рис 2.1). За поданою командою кран-штабелер переносить товар від комірки зберігання до вікна видачі. Завантаження товару може відбуватися різними способами в залежності від реалізації як механізмами розвантаження, так і в ручну. Такі АСЗ виконують ряд задач, а саме: забезпечують безпечне зберігання, здійснюють комплектацію

замовлення відповідно до заданого списку та у повністю автоматичному режимі вивантажує коробки/вироби в потрібній послідовності.

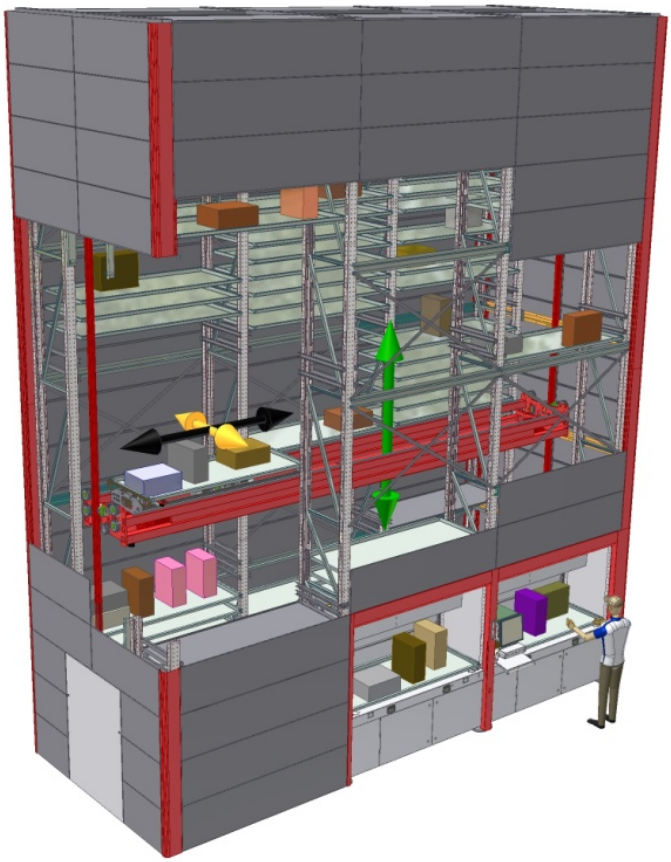


Рисунок 2.1 – АСЗ ліфтового типу

Для кожної коробки або виробу призначена індивідуальна комірка, що унеможливлює виникнення помилки в процесі видачі. В результаті замовлення може бути сформоване з будь-якою послідовністю виробів, а видача буде здійснюватися послідовно відповідно до листа замовлення. Замовлення на завантаження/вивантаження може надходити автоматично з програмного забезпечення верхнього рівня.

Переваги ліфтових АСЗ:

– блочна конструкція дозволяє легко масштабувати систему;

– можливість встановити будь-яку послідовність вивантаження коробок;

Недоліки ліфтових АСЗ:

– обмежена кількість вікон завантаження/видачі.

2.1.2 Карусельні АСЗ

Цей тип автоматизованих складських систем передбачає рухливу конструкцію комірок для зберігання товарів (рис. 2.1). Стелажі розбиті на секції, кожна з яких має власний електропривод та керування. У кожній вертикальній стійці або горизонтальному ряду є спеціальне місце для розвантаження товару.

Рисунок 2.2 – АСЗ карусельного типу

При необхідності завантаження або видачі товару включається електропривід, і комірки починають переміщуватись всередині стелажу по колу за допомогою ланцюгового механізму. Як тільки до місця вивантаження під'їжджає потрібний товар, то в залежності від реалізації він забирається в ручну або виштовхується і транспортується далі автоматично або за допомогою навантажувача.

Переваги карусельних АСЗ:

– можливість розміщення каруселі як горизонтально так і вертикально;

Недоліки карусельних АСЗ:

– неможливість зміни розміру комірок;

– важко масштабувати систему.

2.1.3 Гравітаційні АСЗ

Гравітаційні стелажі за конструкцією та принципом роботи схожі на системи ліфтового типу, але відрізняються тим, що завантажуються з одного боку, а вивантажуються з іншого. При цьому рух коробок або палет по всій глибині стелажу проводиться по роликах під впливом власної сили тяжіння (рис.2.3).

Рисунок 2.3 – гравітаційний стелаж

Автоматичні механізовані човники у гравітаційній системі відповідальні лише за навантаження та забір вантажів із крайніх точок ряду. При вивантаженні крайньої палети на її місце відразу ж, без застосування будь-якої сили, переміщається наступна. Такі АСЗ доцільно використовувати на оптових складах із невеликим асортиментом. Вони дозволяють використовувати дуже глибокі стелажі, максимально збільшуючи ефективну складську площу.

Переваги гравітаційних АСЗ:

– ефективне використання складської площі для великої кількості однотипних товарів;

– можливість масштабування.

Недоліки гравітаційних АСЗ:

– Необхідність двох окремих маніпуляторів для завантаження та для видачі, що підвищує складність обслуговування та ціну системи.

Найкращим варіантом реалізації системи буде гравітаційний тип системи. Такий тип має ряд переваг:

– максимально ефективне використання простору;

– можливість масштабування без істотних змін конструкції;

– гравітаційні АЗС дозволяють зберігати у одній комірці декілька однотипних товарів;

Для здешевлення системи кран-штабелер на завантаження прибирається і завантаження комірок відбувається в ручну.

2.2 Обґрунтування і вибір методів дослідження

Під час виконання роботи були використані наступні методи:

– метод аналізу та синтезу;

– метод порівняння;

– моделювання;

– експеримент.

Метод аналізу та синтезу використовується для розкладання об’єкту дослідження на більш прості складові та їх вивчення і на основі отриманих даних розробки апаратного та програмного забезпечення.

Метод порівняння використовується для:

– вибору типу АСЗ перед розробкою роботизованої шафи;

– реалізації апаратної частини системи, а саме при виборі обладнання та елементної бази, способів підключення та взаємодії елементів системи між собою;

– реалізації програмного забезпечення, а саме при виборі мови програмування, необхідних бібліотек, методів реалізації програмних модулів та взаємодії між ними.

Метод моделювання використовується для дослідження роботи системи у різних умовах.

Експерименти проводяться під час тестування розробленого ПЗ з різними вхідними даними та аналізом результатів роботи програми.

2.3 Програмне забезпечення

2.3.1 JavaScript

JavaScript – мова програмування яка застосовується до HTML документа і може забезпечити динамічну інтерактивність на веб-сайтах. JavaScript є універсальною і гнучкою мовою програмування функціонал якої може бути розширений використанням додаткових бібліотек. Завдяки їм можна розробляти не лише інтерфейс, а і підключення до БД, обміну інформації через комп'ютерну мережу, взаємодію з апаратним забезпеченням [6].

2.3.2 React

React – найпопулярніша бібліотека JavaScript для розробки користувальницького інтерфейсу (UI). Частіше всього React використовується при розробці інтерфейсів веб-сайтів. Данна бібліотека обрано для розробки користувацького інтерфейсу системи через ряд переваг:

1. Простота у використанні. Компонент React створити простіше ніж використовуючи інші бібліотеки, оскільки він використовує JSX – розширення синтаксису JavaScript, яке дозволяє комбінувати HTML з JavaScript.
2. React має безліч додаткових бібліотек розроблених спільнотою користувачів React які дозволяють вирішити практично будь-яке завдання зв'язане з відображенням інтерфейсу.
3. Можливість підключення бібліотеки Electron яка дозволяє перетворити веб-сайт у самостійний додаток.

2.3.3 Node.js

Node.js – це програмна платформа, що розширює можливості мови JavaScript. Node.js додає можливість JavaScript взаємодіяти з пристроями введення-виводу через свій API, написаний на C++, підключати інші зовнішні

бібліотеки, написані різними мовами, забезпечуючи виклики до них з JavaScript-коду. Частіше за все Node.js використовується для розробки серверних частин програмних продуктів, що працюють з використанням мережі Internet [7].

Використання Node.js дозволить підключитись до БД та до контролера роботизованої шафи.

2.4 Моделювання комп’ютерних систем

2.4.1 Комп'ютерне і статистичне імітаційне моделювання

Основою комп'ютерного моделювання як методу наукових досліджень є:

1. Побудова математичних моделей для опису процесів, що досліджуються;
2. Використання швидкодіючих обчислювальних машин (які можуть виконувати мільйони операцій в секунду) здатних взаємодіяти з людиною.

Сутність комп'ютерного моделювання полягає у наступному: на основі заздалегідь розробленої математичної моделі за допомогою ЕОМ проводиться серія обчислювальних експериментів, тобто досліджуються властивості об'єктів або процесів, знаходяться їх оптимальні параметри і режими роботи, уточнюється модель. Візьмемо деякий процес, який можна описати рівнянням. Тоді змінюючи коефіцієнти, початкові і граничні умови рівняння можна досліджувати поводження об'єкту. Проведені на ЕОМ обчислювальні експерименти з математичними моделями, що імітують поведінку реальних об'єктів, процесів або систем називають імітаційними моделями.

Існують два типи математичних моделей за допомогою яких можна досліджувати реальні процеси і системи: імітаційні та аналітичні.

При створенні аналітичних моделей поведінку реальних процесів і систем (РПС) задають у вигляді явних функціональних залежностей (лінійних, нелінійних, диференціальних або інтегральних рівнянь, систем цих рівнянь). Однак тільки для порівняно простих РПС можливо отримати такі залежності.

Коли РПС занадто складні доводиться їх спрощувати. В такому разі аналітична модель стає занадто грубим наближенням до дійсності. Якщо все ж для складних РПС вдається отримати аналітичні моделі, то найчастіше вони перетворюються в важко розв’язну задачу. В цьому випадку використовується імітаційне моделювання.

Імітаційне моделювання представляє собою проведення обчислювальних експериментів на математичних моделях, що імітують поведінку реальних об’єктів, процесів або систем протягом заданого проміжку часу. При цьому РПС розбиваються на елементарні складові. Функціонування цих складових може бути описане алгоритмами, які імітують явища зі збереженням їх логічної структури і послідовності протікання в часі.

Імітаційне моделювання – це сукупність методів алгоритмізації функціонування об'єктів досліджень, програмної реалізації алгоритмічних описів, організації, планування і виконання на ЕОМ обчислювальних експериментів з математичними моделями, що імітують функціонування РПС протягом заданого періоду.

Під алгоритмізацією функціонування РПС розуміється пооперaційний опис роботи всіх її функціональних підсистем окремих модулів з рівнем деталізації, відповідному комплексу вимог до моделі [8].

«Імітаційне моделювання» (ІМ) – подвійний термін, який складається з двох синонімів: імітація та моделювання. Математичне моделювання використовується у всіх областях науки, тому для того, щоб відрізняти математичні моделі, їм даються додаткові назви. Імітаційне моделювання дає можливість заздалегідь обчислити або передбачити поведінку системи, для чого необхідний обчислювальний експеримент (імітація) на математичній моделі при заданих вихідних даних.

Основні переваги ІМ:

1. Можливість опису поведінки компонента (елементів) процесів або систем на високому рівні деталізації.

2. Відсутність обмежень між параметрами ІМ і станом зовнішнього середовища РПС.

3. Можливість дослідження динаміки взаємодії компонент в часі і просторі параметрів системи [8].

Завдяки цим перевагам ІМ дуже поширені

ІМ використовуються у наступних випадках:

1. Якщо йде процес пізнання явища, ІМ служить засобом вивчення явища.

2. Якщо математичні процеси аналітичного методу є занадто складними, ІМ дає простіший спосіб вирішення завдання.

3. Коли реальне спостереження явищ неможливе умовах (реакції термоядерного синтезу, дослідження космічного простору), ІМ стає єдиним способом дослідження таких складних систем

4. Якщо окрім оцінки впливу параметрів на систему необхідне спостереження за процесом протягом певного періоду.

5. При дослідженні нових ситуацій у РПС. У цьому випадку вивчаються правила і умови проведення натурних експериментів.

6. При підготовці фахівців для експлуатації нової техніки ІМ допомагають набути необідніх навичок.

7. При проектуванні системи у яких послідовність подій має ключове значення моделі використовуються для пошуку вузьких місць у функціонуванні РПС

8. Якщо необхідне дослідження процесів шляхом прискорення або уповільнення явищ під час моделювання.

ІМ також має ряд недоліків:

1. Розробка ІМ вимагає більше часу і ресурсів ніж аналітичні моделі.

2. ІМ може виявитись не точною і ступінь неточності неможливо буде виміряти.

3. Можливість труднощів при розробленні ІМ і як наслідок виникнення методологічних помилок.

Але не зважаючи на перелічені недоліки ІМ є одним з найпоширеніших методів аналізу і синтезу складних процесів і систем

Статистичне імітаційне імітування є одним із видів ІМ і використовується при відтворенні на ЕОМ складних випадкових процесів.

При дослідженні складних систем, схильних до випадковим збурень використовуються імовірнісні аналітичні моделі та імовірнісні імітаційні моделі.

В імовірнісних аналітичних моделях випадкові фактори задаються імовірнісними характеристиками випадкових процесів (закони розподілу імовірностей, спектральні щільності або кореляційні функції). Побудова таких моделей є складною обчислювальною задачею, тому їх використовують для дослідження відносно простих систем.

Внесення випадкових збурень у ІМ не викликає великих ускладнень, тому складні випадкові процеси вивчаються за допомогою ІМ.

У імовірнісному ІМ використовуються конкретні випадкові числові значення параметрів системи замість імовірнісних характеристик випадкових процесів. При цьому результати отримані на таких ІМ є випадковими реалізаціями. Тому для отримання об'єктивних характеристик процесу необхідно статистично обробити результати багаторазового моделювання. Саме тому дослідження систем з випадковими характеристиками за допомогою ІМ називають статистичними моделюванням.

Статистична модель випадкового процесу – це алгоритм, за допомогою якого імітують роботу складної системи, схильною до випадковим збурень; імітують взаємодію елементів системи, що носять імовірнісний характер[2].

При статистичному ІМ на ЕОМ виникає необхідність у отриманні випадкових числових послідовностей за допомогою заданих імовірнісних характеристик. Чисельний метод генерації таких послідовностей отримав назву «метод Монте-Карло». Так як метод Монте-Карло використовується у інших чисельних методах (взяття інтегралів, рішення рівнянь), його також називають «метод статистичних випробувань».

Статистичне моделювання складається з декількох етапів:

1. Моделювання на ЕОМ методом Монте-Карло псевдовипадкових числових послідовностей параметрів системи або процесу для кожного експерименту.
2. Використання отриманих числових послідовностей у ІМ.
3. Статистична обробка результатів моделювання.

2.4.2 Методи моделювання комп’ютерних мереж

При моделюванні комп’ютерних мереж частіше за все використовуються аналітичні та імітаційні моделі.

Аналітичні моделі мереж будуються на основ математичних апаратів теорій масового обслуговування, ймовірностей і марковських процесів, а також методів дифузійної апроксимації. Також в якості методів аналітичного моделювання мереж можуть застосовуватися диференціальні і алгебраїчні рівняння, що описують поведінку мережі у часі.

При використанні аналітичного моделювання часто вдається отримати моделі для рішення досить широкого кола завдань дослідження комп'ютерних мереж. В той же час аналітичні моделі мережі мають ряд істотних недоліків, до числа яких слід віднести:

– значні спрощення, властиві більшості аналітичних моделей (Подібні спрощення ставлять іноді під сумнів результати аналітичного моделювання);

– громіздкість обчислень для складних моделей.

Таким чином, незважаючи на значні досягнення аналітичного моделювання, багато реальні ситуації неможливо адекватно представити за допомогою відповідних математичних моделей. В одних випадках цього заважає певна «жорсткість» математики як мови опису та подання подій і явищ. В інших випадках, навіть якщо є можливість формалізувати розглянуту життєву ситуацію за допомогою побудови математичної моделі, отримана на

її основі задача оптимізації може бути занадто складною для сучасних алгоритмів вирішення завдань цього класу.

Другий вид моделювання комп'ютерних мереж – імітаційне моделювання. цей вид моделювання часто є найкращим, а іноді і єдиним способом дослідження реальних систем, в тому числі і мереж.

Імітаційне моделювання застосовується у важко прогнозованих процесах і для передбачення і дослідження поведінки необхідний обчислювальний експеримент (імітація) при заданих вихідних даних.

Різниця між аналітичною і імітаційною моделями полягає в тому, що в останній замість явного математичного опису взаємини між вхідними та вихідними змінними реальна система розбивається на ряд досить малих (в функціональному відношенні) елементів або модулів. Потім поведінку вихідної системи імітується як поведінку сукупності цих елементів, певним чином пов'язаних (шляхом встановлення відповідних взаємозв'язків між ними) в єдине ціле. Обчислювальна реалізація такої моделі починається з вхідного елемента, далі проходить по всіх елементах, поки не буде досягнутий вихідний елемент моделі.

Таким чином імітаційне моделювання найкраще підходить для обчислення параметрів мережі і виявлення проблемних вузлів.

2.5 Висновки до розділу

У теоретичному розділі були розглянуті різні типи автоматизованих систем зберігання для розробки автоматизованої шафи та був обраний гравітаційний тип. Був розглянутий і обґрунтований вибір програмних засобів для розробки програмного забезпечення. Були розглянуті аналітичне та імітаційне моделювання комп’ютерних мереж та було вирішено використати імітаційне моделювання.

3 СИНТЕЗ СИСТЕМИ КОНТРОЛЮ

3.1 Розробка функціональної схеми структури аптеки системи комплексу роботизованих аптек ТМ «Копійка»

Система комплексу роботизованих аптек ТМ «Копійка» призначена для видачі та контролю медикаментів у аптеках, обміну інформації між підсистемами аптек та центрального офісу та всередині підсистем, збору статистики, покращення сервісу у мережі аптек.

Кожна аптека системи комплексу роботизованих аптек ТМ «Копійка» представляє собою окрему підсистему, яка містить у собі наступні вузли:

- маршрутизатор;
- ПЕОМ завідуючого аптекою;
- ПЕОМ фармацевта;
- касове обладнання;
- роботизована шафа.

Підсистема аптеки має виконувати наступні функції:

- зберігання БД товарів аптеки;
- відображення БД і можливість її редагування;
- зв’язок з іншими підсистемами системи;
- видачу товарів за командою фармацевта і внесення відповідних змін до БД;

Всі ПЕОМ підключаються до маршрутизатора по інтерфейсу Ethernet. Касове обладнання підключаються до ПЕОМ фармацевтів по інтерфейсу USB. Роботизована шафа підключається по інтерфейсу USB до ПЕОМ завідуючого аптекою.

Роботизована шафа складається з наступних складових:

- контролер;
- датчик нульового положення;
- енкодер горизонтального положення лотку;
- енкодер вертикального положення лотку;

– датчик наявності у лотку товару;

– кроковий двигун горизонтального положення лотку;

– кроковий двигун вертикального положення лотку;

– кроковий двигун завантаження у лоток;

– лінійний привід вивантаження з лотку.

Роботизована шафа має виконувати наступні функції:

– переміщення лотку до комірки з потрібним товаром та до позиції розвантаження;

– завантаження товару у лоток;

– вивантаження товару з лотку у позиції розвантаження.

Функціональна схема структури аптеки системи комплексу роботизованих аптек ТМ «Копійка» зображена на рисунку 3.1.

На основі функціональної схеми структури розробляється функціональна схема автоматизації.

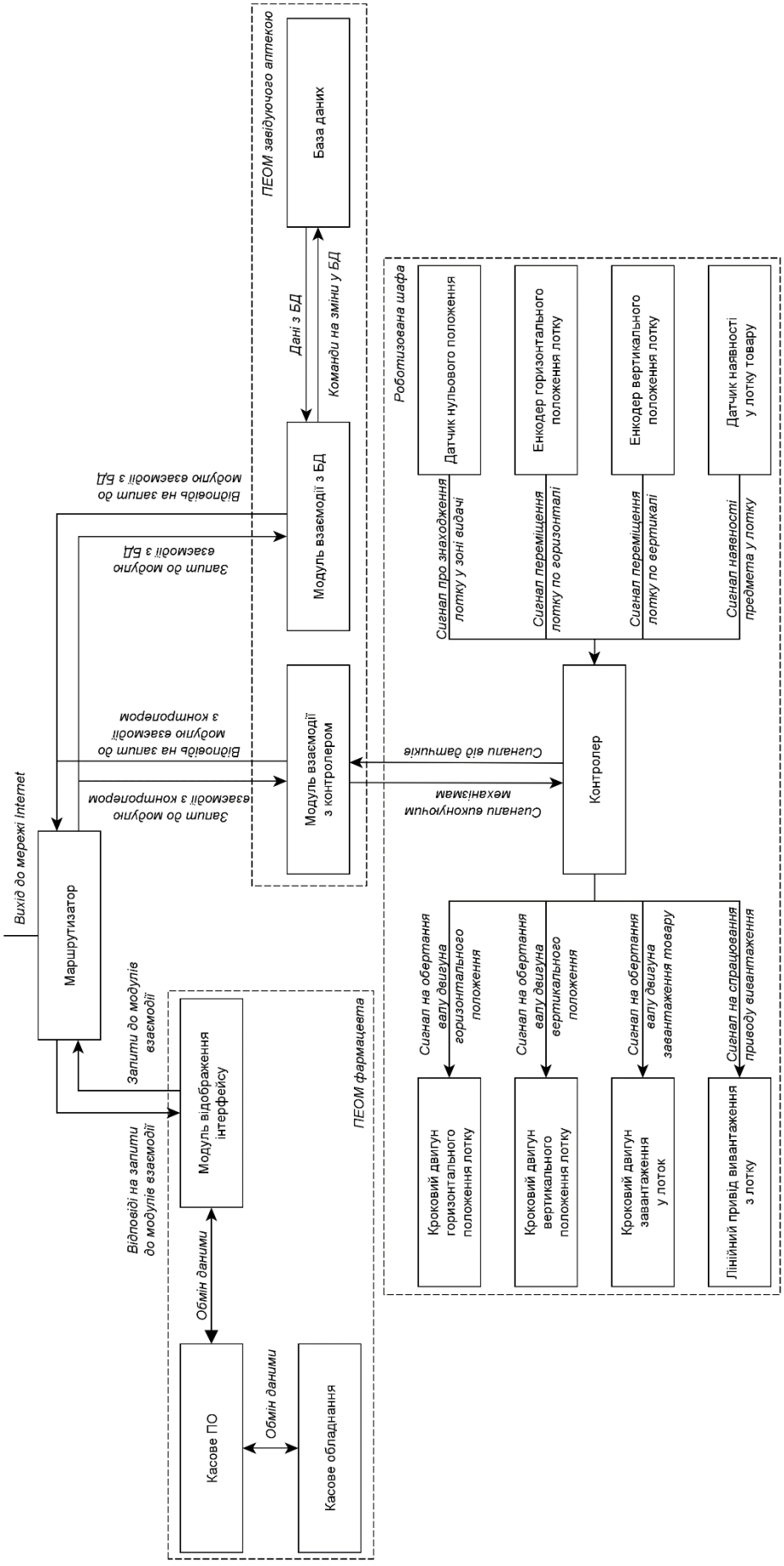


Рисунок 3.1 – Функціональна схема структури аптеки комплексу роботизованих аптек ТМ «Копійка»

3.2 Вибір апаратних засобів системи комплексу роботизованих аптек ТМ «Копійка»

Наступне обладнання було обрано виходячи з аналізу функціональної схеми системи комплексу роботизованих аптек ТМ «Копійка» та вимог замовника.

До маршрутизаторів замовником були висунуті вимоги: підтримка протоколів DHCP, Ipsec, NAT та наявність бездротового підключення. Додатково для маршрутизаторів у аптеках: через їх велику кількість, вартість маршрутизатора повинна бути мінімальною.

Враховуючи попередньо перераховані вимоги у якості аптечного маршрутизатора буде використано маршрутизатор D-Link DIR-841 (рис. 3.2).

Рисунок 3.2 – Маршрутизатор D-Link DIR-841

Основні технічні характеристики маршрутизатора D-Link DIR-841 вказані в таблиці 3.1 [9].

Таблиця 3.1 – Основні технічні характеристики маршрутизатора D-Link DIR-841

Опис	Технічні характеристики
Форм-фактор	Настільний
Версія протоколу Wi-Fi	802.11a, 802.11ac, 802.11b, 802.11g, 802.11n
Кількість антен	4 зовнішніх
Підтримка протоколів	DHCP, Ipsec, L2TP, NAT, PPPoE, PPTP
Швидкість Wi-Fi	867 Мбіт/с
Частота роботи Wi-Fi	2.4 ГГц, 5 ГГц

Продовження таблиці 3.1

Габарити	205 x 136 x 33 мм
Вага	285 г
Додатково Мережеві функції:	– Підтримка стандарту IEEE 802.1X для підключення до Інтернету; – DHCP-сервер/relay; – Автоматичне отримання LAN IP-адреси (у режимах точка доступу, повторювач, клієнт); – Статична IP-маршрутизація; – Статична IPv6-маршрутизація RIP; – Підтримка VLAN; – Створення M3T-тунелів; – Ручне налаштування швидкості та режиму дуплексу для кожного порту Ethernet; – Налаштування максимальної швидкості вихідного трафіку для кожного порту маршрутизатора
Процесор	RTL8197H (1 ГГц)
Оперативна пам'ять	64 МБ, DDR2
Flash-пам'ять	8 МБ, SPI
Діапазон частот бездротового модуля	2400 ~ 2483.5 МГц 5150 ~ 5350 МГц 5650 ~ 5725 МГц
Інтерфейс підключення	Gigabit Ethernet (10/100/1000)
Швидкість LAN портів	100 Мбіт/с
Кількість LAN портів	5
WAN-порт	Ethernet

До маршрутизатора у центральному офісі підключена велика кількість комп’ютерів та VPN-тунелів, тому у центральному офісі буде встановлено потужніший маршрутизатор Cisco RV160W Wireless-AC VPN Router (рис. 3.3).

Рисунок 3.3 – Маршрутизатор Cisco RV160W Wireless-AC VPN Router

Основні технічні характеристики маршрутизатора Cisco RV160W Wireless-AC VPN Router вказані в таблиці 3.2 [10].

Таблиця 3.2 – Основні технічні характеристики маршрутизатора Cisco RV160W Wireless-AC VPN Router

Опис	**Технічні характеристики**
Форм-фактор	настільний
Ethernet WAN	1 RJ-45 Комбінований порт Gigabit Ethernet
Ethernet LAN	4 порти Gigabit Ethernet RJ-45
Консольний порт	1 порт RJ-45
Операційна система	Linux
VLANs	16
WAN	DHCP, статичний IP, PPPoE, PPTP, L2TP
WLAN	2x2 бездротовий 802.11ac
Контроль доступу	Списки керування доступом (ACL) до IP
Безпечне управління	HTTPS, складність імені користувача/пароля
Права користувача	Два рівні доступу: адміністратор і гість
Мережеві протоколи	– DHCP – PPPoE – PPTP – L2TP – DNS-проксі – Агент ретрансляції DHCP – IGMP і багатоадресна переадресація – RSTP – Динамічний DNS (TZO, DynDNS, 3322.org, No-IP)

Продовження таблиці 3.2

	– NAT, PAT – NAT один на один – Керування портами – Віддзеркалення портів
Протоколи маршрутизації	– Статична маршрутизація, проксі-сервер IGMP – Динамічна маршрутизація – Динамічний DNS (ChangeIP, DynDNS, No-IP) – RIP v1 і v2 – RIP для IPv6 (RIPng)
IPsec від шлюзу до шлюзу VPN	10 тунелів IPsec
IPsec від клієнта до шлюзу VPN	10 тунелів IPsec
PPTP VPN	10 ТУНЕЛІВ PPTP VPN
OpenVPN	Підтримка сервера OpenVPN
Шифрування	Потрійний стандарт шифрування даних (3DES), розширений стандарт шифрування (AES) з шифруванням 128, 192 та 256-розрядних ключів
VPN прохідний	Прохідний прохід IPsec, PPTP і L2TP
Пропускна здатність NAT	600 Мбіт/с
Паралельні сеанси	15,000
Пропускна здатність IPsec VPN	50 Мбіт/с
Веб-інтерфейс користувача	Конфігурація на основі браузера (HTTP/HTTPS)

В центральному офісі необхідно розмістити комутатор для 36 пристроїв, тому обираємо комутатор Cisco C1000-48P-4X-L (рис. 3.4).

Рисунок 3.4 – комутатор Cisco C1000-48P-4X-L

Основні технічні характеристики маршрутизатора Cisco RV160W Wireless-AC VPN Router вказані в таблиці 3.3 [11].

Таблиця 3.3 – Основні технічні характеристики маршрутизатора Cisco RV160W Wireless-AC VPN Router

Опис	Технічні характеристики
Форм-фактор	Стоєчний
Ідентифікатор	C1000-48P-4G-L
Гігабітні порти Ethernet / FE	48 10/100/1000 RJ45 PoE+
Інтерфейси uplink	4 SFP
PoE + бюджет потужності	370W
Вага	5.43

У якості сервера у центральному офісі буде встановлено Cisco UCS B200 M6 Blade Server (рис. 3.5) з конфігурацією 2 SSD-накопичувачі по 3,6 ТБ кожен.

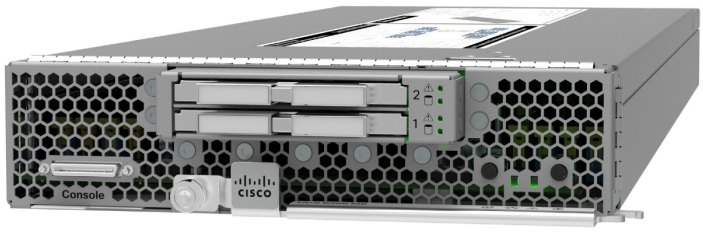


Рисунок 3.5 – Cisco UCS B200 M6 Blade Server

Основні технічні характеристики Cisco UCS B200 M6 Blade Server вказані в таблиці 3.4 [12].

Таблиця 3.4 – Основні технічні характеристики маршрутизатора Cisco RV160W Wireless-AC VPN Router

Опис	Технічні характеристики
Форм-фактор	Стоєчний
Центральний процесор	До 2 процесорів Intel Xeon третього покоління (1 або 2)
Пам'ять	32 слоти DDR4 DIMM: 16, 32, 64, 128 ГБ до 3200 МГц
Постійна пам'ять Intel Optane DC	16 слотів DIMM: 128, 256 і 512 ГБ при до 3200 МГц

Продовження таблиці 3.4

Внутрішня пам'ять	2 приводи з гарячим доступом передніх 7 мм форм-факторів: – SSD-накопичувачів з до 7,6 ТБ на диск; – NVMe: до 7,7 ТБ на диск; Або 4 – M.2: До 960 ГБ на диск

Для того, щоб підключити внутрішню пам’ять необхідно додатково обрати RAID-контролер Cisco FlexStorage 12G з 2 відсіками для SSD накопичувачів.

До робочих станцій фармацевтів та касового обладнання висуваються деякі вимоги.

Вимоги до робочих станцій фармацевтів:

– ПЕОМ з процесорами типу Intel Core під управлінням ОС Windows 7/8/8.1/10;

– Вільний дисковий простір не менше 1 ГБ.

– Оперативна пам'ять, не менше 1ГБ.

Вимоги до касового обладнання:

– Можливість підключення до робочих станцій під управлінням ОС Windows 7/8/8.1/10 по інтерфейсу USB або Ethernet.

– Можливість обміну інформацією з базою даних MongoDB або можливість роботи з NodeJS безпосередньо або через додаткове ПЗ;

3.3 Розробка функціональної схеми автоматизації роботизованої шафи

Керування роботизованою шафою виконується обладнанням, що розміщується у шафі КТЗ. Деякі датчики та виконуючі механізми не можуть бути підключені на пряму до контролера, тому необхідно передбачити спеціальні драйвери. Також треба враховувати, що контролер підключається

до ПЕОМ завідуючого аптекою. Таким чином буде використана ієрархічна топологія. Функціональна схема автоматизації роботизованої шафи зображена на рисунку 3.6.

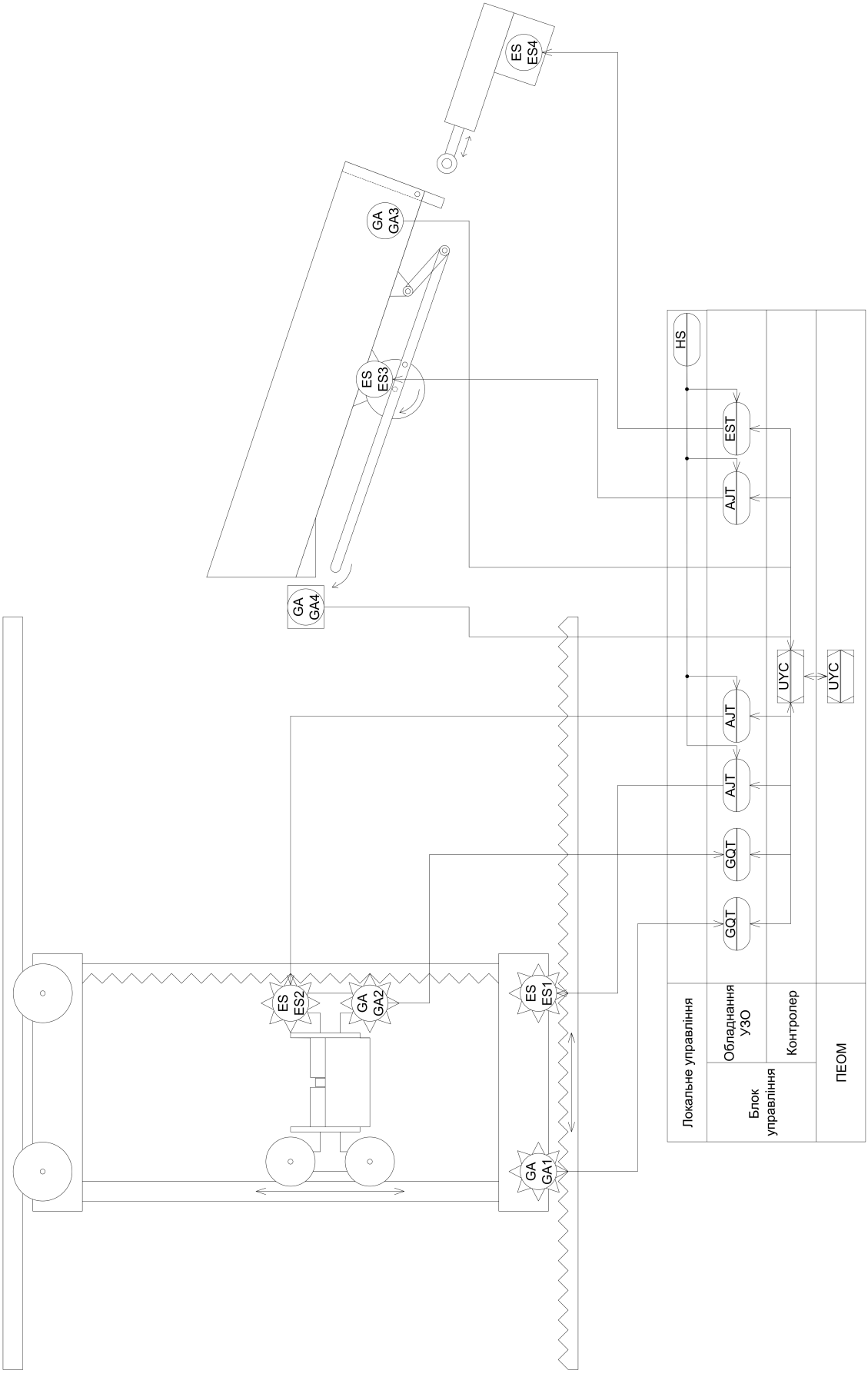


Рисунок 3.6 – Функціональна схема автоматизації роботизованої шафи

3.4 Розробка принципової схеми системи управління роботизованої шафи

3.4.1 Аналіз входів та виходів роботизованої шафи

Для визначення входів та виходів роботизованої шафи необхідно провести аналіз та класифікацію входів та виходів датчиків та виконавчих механізмів. В результаті аналізу отримано класифікацію входів та виходів представлені в таблиці 3.5.

Таблиця 3.5 – Перелік вхідних та вихідних сигналів контролера роботизованої шафи

№ п/п.	Найменування інформації (сигнали, дані)	Іденти-фікатор	Напр. вх./вих.	Функція	Вид	Джерело/ Отримувач	Форма представлення (розрядність, точність)		Період вв./вив., сек.
							Зовнішня	Внутрішня	
1.	Горизонтальна позиція лотку	GA1	Вхід	Вимір.	Частотний	Енкодер горизонт. положення	5 В	8 біт	0,2
2.	Вертикальна позиція лотку	GA2	Вхід	Вимір.	Частотний	Енкодер вертикал. положення	5 В	8 біт	0,2
3.	Наявність товару в лотку	GA3	Вхід	Вимір.	Неперер-вний	Оптичний датчик наявності товару	0,11-1,6 В	16 біт	0,2
4.	Лоток у позиції вивантаження	GA4	Вхід	Вимір.	Неперер-вний	Оптичний датчик нульового положення лотку	0,11-1,6 В	16 біт	0,2

Продовження таблиці 3.5.

5.	Переміщення лотку по горизонталі	ES1	Вихід	Управл.	Нормально відкритий	Кроковий двигун горизонт. переміщення	5 В	16 біт	0,2
6.	Переміщення лотку по вертикалі	ES2	Вихід	Управл.	Нормально відкритий	Кроковий двигун вертикал. переміщення	5 В	16 біт	0,2
7.	Завантаження товару в лоток	ES3	Вихід	Управл.	Нормально відкритий	Кроковий двигун завантаження у лоток	5 В	4 біт	0,2
8.	Вивантаження товару з лотку	ES4	Вихід	Управл.	Нормально відкритий	Лін. привід відкр. лотку	5 В	1 біт	0,2

3.4.2 Вибір елементної бази системи управління роботизованою шафою

У якості контролера роботизованої шафи використовується контролер Arduino Uno. При своїй низькій ціні, даний контролер дозволяє виконувати усі необхідні від нього функції та відповідає обмеженням, що накладені переліком у таблиці 3.5. Так як роботизована шафа розташована у аптеці, яка є опалюваним та кондиціонуємим приміщенням, у промисловому виконанні системи управління немає необхідності і рівня захисту IP22 буде достатньо. Основні технічні характеристики контролера Arduino Uno вказані в таблиці 3.6 [13].

Таблиця 3.6 – Основні технічні характеристики контролера Arduino Uno

Опис	Технічні характеристики
Мікроконтролер	ATmega328
Робоча напруга	5 В
Вхідна напруга (рекомендована)	7-12 В
Вхідна напруга (гранична)	6-20 В
Цифрові входи/виходи	14 (6 з яких можуть використовуватися як виходи ШІМ)
Інтерфейс підключення до ПЕОМ	USB
Аналогові входи	6
Постійний струм через вхід/вихід	40 мА
Постійний струм для виведення 3.3 В	50 мА
Флеш пам’ять	32 Кб (ATmega328) з яких 0.5 Кб використовуються для завантажувача
ОЗУ	2 Кб (ATmega328)
Тактова частота	16 МГц

Для взаємодії контролера з ПЕОМ, Arduino Uno має інтерфейс ICSP.

Для взаємодії контролера з кроковими двигунами був обраний драйвер для керування кроковим двигуном на мікросхемі L298N. Даний драйвер дозволяє працювати з більшістю крокових двигунів. Основні технічні характеристики драйвера вказані в таблиці 3.7 [14].

Таблиця 3.7 – Основні технічні характеристики драйвера для керування кроковим двигуном

Опис	Технічні характеристики
Мікросхема	L298N
Напруга живлення	5 В
Дискретні входи	4
Дискретні виходи	4
Діапазон робочих напруг	Від +5В до +46В

У якості енкодера використовується абсолютний енкодер Bourns ACE-128. Даний енкодер є досить точним при своїх невеликих розмірах і малій вазі. Основні технічні характеристики енкодера вказані в таблиці 3.8 [15].

Таблиця 3.8 – Основні технічні характеристики енкодера

Опис	Технічні характеристики
Вихід	8-розрядний код із 128 абсолютними станами
Обороти в хвилину (робочі)	120 максимум
Потужність підключення контактів	10 міліампер при 10 В постійного струму або максимум 0,1 Вт

Для взаємодії контролера з енкодерами була обрана плата розширення I2C на мікросхемі PCF8574T. Даний модуль має невеликі розміри і дозволяє розширити кількість дискретних виходів з 2 до 8. Основні технічні характеристики плати розширення I2C вказані в таблиці 3.9[16].

Таблиця 3.9 – Основні технічні характеристики плати розширення I2C

Опис	Технічні характеристики
Мікросхема	PCF8574T
Напруга живлення	5 В
Дискретні входи	8
Аналогові виходи	2
Інтерфейс	I2C

Для забезпечення руху штовхача лінійного приводу у необхідну сторону використовується подвійне реле. Лінійний привід сконструйовано так, що при досягненні штовхачем крайніх положень двигун лінійного приводу припиняє роботу поки не буде змінено полярність живлення.

У якості оптичних датчиків використовуються датчики HOA0149-001. Ці датчики є досить простими та мають малий розмір.

Для зменшення кількості дротів прокладених від контролера до лотку використовується плата розширення дискретних входів/виходів на мікросхемі PCA9534D. Основні технічні характеристики плати розширення дискретних входів/виходів вказані в таблиці 3.10 [17].

Таблиця 3.10 – Основні технічні характеристики плати розширення дискретних входів/виходів

Опис	**Технічні характеристики**
Мікросхема	PCA9534D
Напруга живлення	5 В
Дискретні виходи	8
Інтерфейс	I2C

У якості джерела струму для виконуючих механізмів та енкодерів використовується джерело постійного струму потужністю 500 Вт та напругою 24 В та силою струму 20А.

3.4.3 Реалізація принципової схеми системи управління роботизованою шафою

На основі функціональної схеми автоматизації роботизованої шафи, опису входів та виходів та обраної елементної бази була розроблена принципова схема зображена на рисунку 3.7. Кодування пристроїв на принциповій схемі роботизованої шафи вказане у таблиці 3.11.

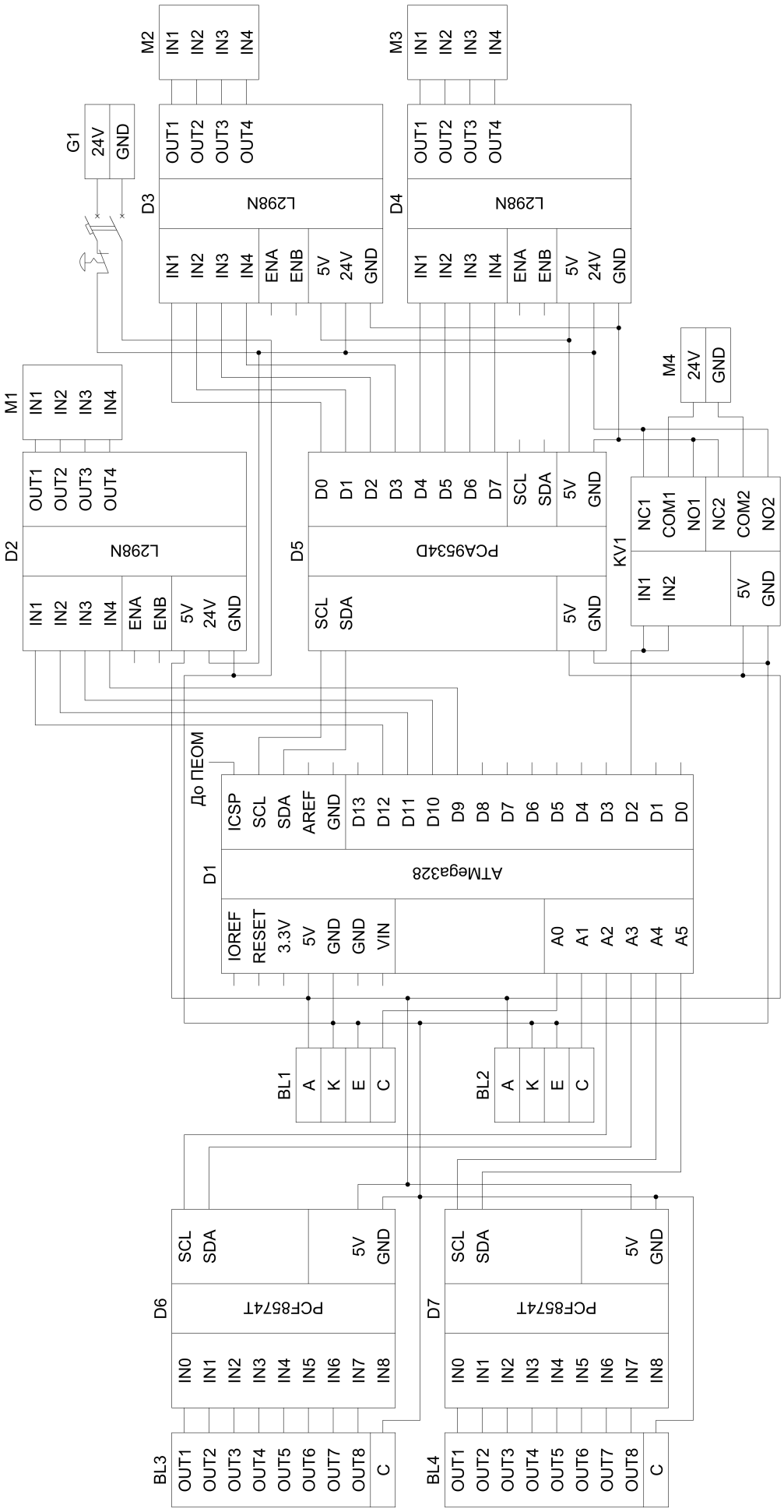


Рисунок 3.7 – Принципова схема роботизованої шафи

Таблиця 3.11 – Кодування пристроїв на принциповій схемі роботизованої шафи

Кодування	**Пристрій**
G1	Джерело постійного струму 24 В
D1	Контролер Arduino Uno
D2, D3, D4	Драйвер для керування кроковим двигуном
D5	Плата розширення дискретних входів/виходів Arduino Uno
D6, D7	Плата розширення I2C
M1, M2, M3	Крокові двигуни
M4	Лінійний привід
BL1, BL2	Оптичний датчик
BL3, BL4	Енкодери
KV1	Подвійне реле

3.5 Висновки до розділу

Під час синтезу системи було підібране обладнання до комп’ютерної мережі комплексу аптек та розроблена роботизована шафа у вигляді функціональної та структурної схеми. До роботизованої шафи була підібрана елементна база датчиків, виконуючих механізмів та драйверів.

4 РОЗРОБЛЕННЯ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

4.1 Призначення й сфера застосування програми

Програма призначена для збереження і редагування бази даних медикаментів, візуалізації даних, що зберігаються, та управління роботизованою шафою в аптеках системи комплексу роботизованих аптек ТМ «Копійка».

4.2 Обґрунтування технічних характеристик

4.2.1 Постановка завдання на розробку програми

Програма повинна виконувати наступні задачі:

– відображення таблиці всіх товарів;

– видалення обраного товару;

– редагування обраного товару;

– відправлення сигналів управління до контролера роботизованої шафи;

– редагування БД після видачі товару роботизованою шафою.

4.2.2 Опис алгоритму і функціонування програми

Виходячи із завдання на розробку, програму доцільно розділити на 3 модулі:

1. Відповідальний за відображення інтерфейсу та взаємодії з оператором. Цей модуль буде відображати головний екран з таблицею усіх товарів, кнопки взаємодії з товарами, вікна додавання та редагування товарів та функціонал формування масиву товарів для видачі. Цей модуль буде зв’язуватись з двома іншими.

2. Відповідальний за взаємодію з базою даних. Цей модуль буде посередником у взаємодії модулю відображення інтерфейсу та базою даних.

3. Відповідальний за взаємодію з контролером. Цей модуль буде посередником у взаємодії модулю відображення інтерфейсу та контролером.

Схема взаємодії модулів зображена на Рисунку 4.1.

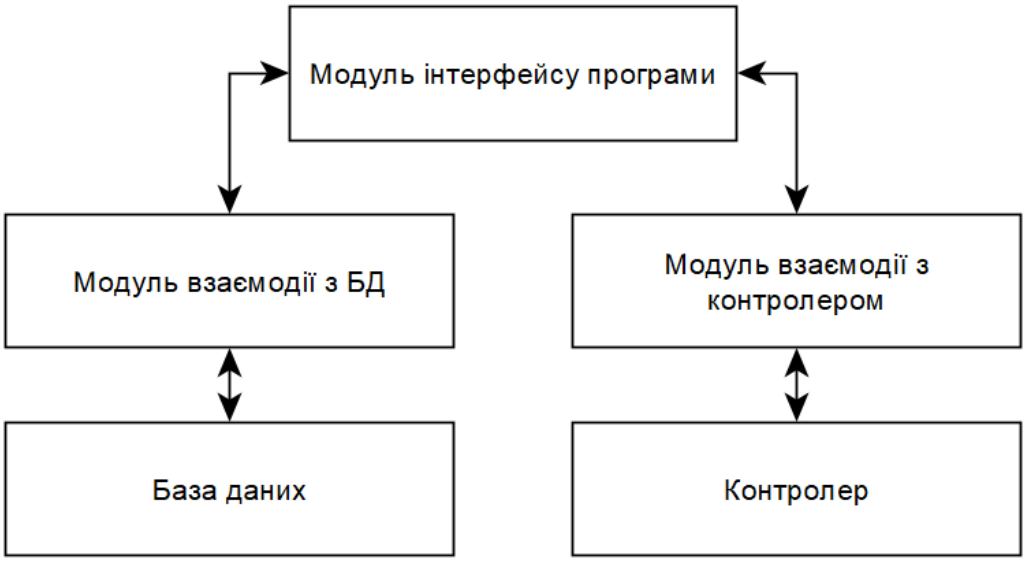


Рисунок 4.1 – Схема взаємодії модулів програми

4.2.3 Опис і обґрунтування вибору методу організації вхідних та вихідних даних

Кожен з трьох модулів описаних у попередньому розділі має свої вхідні та вихідні дані.

Модуль відображення інтерфейсу повинен приймати наступні вхідні дані:

– дані введені оператором для управління товарами;
– дані отримані з модулю взаємодії з БД;
– сигнали від модулю взаємодії з контролера.

У якості вихідних даних модулю відображення інтерфейсу використовуються:

– запити до модулю взаємодії з БД про отримання або редагування даних;
– запит до модулю взаємодії з контролером про видачу товарів;
– вивід даних про продукти у інтерфейсі програми.

Модуль взаємодії з БД повинен приймати наступні вхідні дані:

– запити від модулю відображення інтерфейсу;

– відповіді від БД на надіслані команди.

У якості вихідних даних модулю взаємодії з БД використовуються:

– команди до БД;

– отримані дані з БД або інформація про вдалу чи невдалу зміну даних в БД відправляються як відповідь на запит від модулю відображення інтерфейсу;

Модуль взаємодії з контролером повинен приймати наступні вхідні дані:

– запити від модулю відображення інтерфейсу;

– сигнали від контролера.

У якості вихідних даних модулю взаємодії з контролером використовуються:

– команди до контролера;

– відповідь на запит від модулю відображення інтерфейсу про успішну видачу товарів;

4.2.4 Опис і обґрунтування вибору та складу технічних та програмних засобів

Для розробки програмного забезпечення було використано мову JavaScript з бібліотеками React та Node.js. JavaScript має велику кількість бібліотек для організації мережевих запитів та логіки роботи програми. React дозволяє легко розробляти зручний інтерфейс. Node.js дозволяє працювати з базами даних та апаратними інтерфейсами.

Для тестування програми у якості контролера роботизованої шафи було обрано контролер Arduino Uno через його багатофункціональність та підтримку великої кількості мов програмування. На Arduino Uno завантажений стандартний скетч StandardFirmataPlus [18] який забезпечує зв’язок з програмним забезпеченням комп’ютера.

Програма працює на ПЕОМ з операційною системою Windows.

4.3 Опис розробленої програми

4.3.1 Загальні відомості

4.3.1.1 Позначення і найменування програми

Модуль відображення інтерфейсу має наступні атрибути:

Найменування виконуваного файлу	–	Pharmacy.exe
Розмір файлу	–	134 Мбайт
Версія файлу	–	1.1.0.0
Версія продукту	–	1.01.0001
Внутрішнє ім’я	–	Модуль відображення інтерфейсу
Назва початкового файлу	–	Pharmacy.exe
Назва продукту	–	Програма обліку медикаментів в аптеках та керування роботизованою шафою
Опис версії файлу	–	1.01.0001
Виробник	–	Чумичов Денис
Мова	–	Українська

Модуль взаємодії з БД має наступні атрибути:

Найменування виконуваного файлу	–	index.js
Розмір файлу	–	13,1 Мбайт
Версія файлу	–	1.1.0.0
Версія продукту	–	1.01.0001
Внутрішнє ім’я	–	Модуль взаємодії з БД
Назва початкового файлу	–	server.bat
Назва продукту	–	Модуль взаємодії з БД
Опис версії файлу	–	1.01.0001
Виробник	–	Чумичов Денис
Мова	–	Українська

Модуль взаємодії з контролером має наступні атрибути:

Найменування виконуваного файлу	–	index.js
Розмір файлу	–	21,3 Мбайт
Версія файлу	–	1.1.0.0
Версія продукту	–	1.01.0001
Внутрішнє ім’я	–	Модуль взаємодії з контролером
Назва початкового файлу	–	controller.bat
Назва продукту	–	Модуль взаємодії з контролером
Опис версії файлу	–	1.01.0001
Виробник	–	Чумичов Денис
Мова	–	Українська

4.3.1.2 Програмне забезпечення, необхідне для функціонування програми

Програма працює під управлінням ОС Windows 7/8/8.1/10. На ПЕОМ повинні бути встановлені система управління БД MongoDB та середовище Node.js.

4.3.1.3 Мови програмування, на яких написана програма

Вихідною мовою програмування є JavaScript з бібліотеками React та Node.js. Середовище розробки, компілятор – Visual Studio Code.

4.3.2 Функціональне призначення

4.3.2.1 Призначення програми

Програмний пакет «Pharmacy» призначений для роботи з БД, у якій зберігаються дані про товари, що знаходяться в шафі, візуального відображення даних оператору, взаємодії з контролером роботизованої шафи.

Програма виконує наступні задачі:

– відображення таблиці всіх товарів;
– видалення обраного товару;
– редагування обраного товару;
– відправлення сигналів управління до контролера роботизованої шафи;
– редагування БД після видачі товару роботизованою шафою;

4.3.2.2 Відомості про функціональні обмеження

При заповнені форми додавання або редагування продукту необхідно заповнити усі обов’язкові дані. Також треба враховувати, що код продукту повинен бути унікальним для кожного товару.

4.3.3 Опис логічної структури

4.3.3.1 Алгоритм роботи програми

Програмний пакет «Pharmacy» повинен забезпечувати можливість виконання перерахованих нижче функцій:

1. При першому запуску програми відкривається екран авторизації. Після введення оператором логіну та пароля, відправляється запит до БД та перевіряється існування логіну та правильність паролю. У випадку успішної перевірки йде перехід до основного екрану програми. У випадку, якщо перевірка не пройдена, оператору показується помилка, що інформує про неправильне введення логіна або пароля. Оператор залишається авторизованим після виходу з програми та подальших входів до неї. Вихід з облікового запису здійснюється лише по натисканню відповідної кнопки на головному екрані.

2. На головному екрані відображається таблиця з продуктами, що зберігаються в БД. Окрім інформаційних колонок (Зображення товару,

Найменування товару, Код товару, Категорія товару, Ціна за одиницю товару, Строк придатності, Номер комірки, Кількість одниць товару), в таблиці присутні три керуючі колонки, в яких відображаються кнопки напроти кожного продукту. Кнопки забезпечують наступні функції: видалення продукту, редагування продукту, додавання продукту до кошика. При натисканні на заголовок колонки відбувається сортування строк таблиці по цій колонці за зростанням, при другому натисканні: відбувається сортування строк таблиці по цій колонці за спаданням. При третьому натисканні колонки повертаються в початкове положення. При наступних натисканнях цикл сортування повторюється. Також на головному екрані відображаються кнопки виходу з облікового запису, додавання продукту, строка пошуку продукту, кнопка відкриття кошику.

3. Натиснувши на кнопку виходу з облікового запису відбувається перехід з головного екрану на екран авторизації.

4. Натиснувши на кнопку додавання продукту відкривається вікно, в якому необхідно заповнити інформацію про продукт, що додається. Після заповнення всіх даних та натискання на кнопку додавання продукту, данні записуються в базу даних, а таблиця оновлюється. При натисканні кнопки відміни вікно закривається.

5. Натиснувши на кнопку відкриття кошику (натискання можливе лише у випадку коли кошик не порожній) відкривається вікно, де відображається таблиця з товарами доданими у кошик. В таблиці можливо змінити кількість товару або видалити його. При натисканні кнопки видачі товару посилається сигнал на виконуючі механізми роботизованої шафи. Після закінчення видачі кошик автоматично очищується, вікно зачиняється, а у БД вносяться зміни відповідно до кількості виданих товарів. Очистити кошик та закрити вікно можна також натиснувши відповідну кнопку.

6. При введені тексту у строку пошуку строки головної таблиці фільтруються за всіма стовбцями після введення або видалення кожного символу.

7. При натисканні кнопки видалення продукту до БД відправляється запит про видалення продукту. Після видалення данні в таблиці оновлюються.

8. При натисканні кнопки редагування продукту відкривається вікно аналогічне вікну додавання продукту, але вже з заповненими полями. При натисненні кнопки редагування продукту відправляється запит про оновлення продукту. Після оновлення данні в таблиці оновлюються.

9. При натисканні кнопки додавання у кошик відбувається перевірка строку придатності і якщо товар прострочений з’являється попередження. Якщо товарне просрочено або якщо оператор згоден видати просрочений товар відповідний продукт додається до кошика.

Програма «Server» відправляє команди до системи MongoDB, вона повинна забезпечувати можливість виконання перерахованих нижче функцій:

1. Отримання даних про всі товари і передача їх до програми «Pharmacy»;

2. Додавання нового товару з даними отриманими від програми «Pharmacy»;

3. Редагувати товар або декілька товарів з даними отриманими від програми «Pharmacy»;

4. Видаляти продукт за ID отриманим від програми «Pharmacy»;

5. Перевіряти дані про оператора отриманими від програми «Pharmacy» та відмічати реєстрацію оператора;

6. Перевіряти авторизацію оператора за даними отриманими від програми «Pharmacy»;

Програма «Controller» відправляє команди до контролера, вона повинна забезпечувати видачу товарів отриманих від програми «Pharmacy».

4.3.3.2 Структура програми

Програма складається з трьох частин:

1. Pharmacy – модуль, що відповідає за інтерфейс взаємодії з оператором.

2. Server – модуль, що отримує та виконує запити від програми Pharmacy та взаємодіє з БД.

3. Controller – модуль що отримує та виконує запити від програми Pharmacy та взаємодіє з контролером роботизованої шафи.

Модуль «Pharmacy» має два основних компоненти SignIn та Products які відображають екран авторизації та екран основного інтерфейсу програми та ряд допоміжних компонентів та функцій.

Таблиця 4.1 – Перелік функцій програми «Pharmacy».

Ім'я	**Момент виклику**	**Призначення**
onSubmit	Оператор натиснув кнопку «Прийняти» у вікні додавання або редагування товару	Відправляє запит до програми «Server» про додавання (якщо initialValues дорівнює false) або редагування (якщо initialValues містить дані) товару
onDelete	Оператор натиснув кнопку видалення товару	Відправляє запит до програми «Server» про видалення товару
onEdit	Оператор натиснув кнопку редагування товару	В змінну initialValues записуються інформація про доданий товар, відкривається вікно редагування товару
onCancel	Оператор натиснув кнопку «Скасувати» у вікні додавання або редагування продукту	Вікно додавання або редагування товару зачиняється, змінна initialValues очищується
onAddToBasket	Оператор натиснув кнопку додавання товару до кошика	Додає у масив basket обраний товар
onCancelBasket	Оператор натиснув кнопку «Скасувати замовлення» у вікні кошика	Очищує масив basket, зачиняє вікно кошика

Продовження таблиці 4.1

onDeleteFrom-Basket	Оператор натиснув кнопку видалення товару із кошика » у вікні кошика	Відфільтровує видалений продукт з масиву basket. Якщо був видалений останній товар, то зачиняє вікно кошика
onChangeCount-ProductBasket	Оператор змінив кількість товару у кошику» у вікні кошика	Змінює відповідне значення у масиві basket
onOutput	Оператор натиснув кнопку «Видати замовлення» у вікні кошика	Відправляє запит до програми «ServerController» про видачу товару з кошика
onExit	Оператор натиснув кнопку «Вийти з облікового запису»	Видаляє токен у локальному сховищі, відправляє запит до програми «Server» про видалення токену, здійснює перехід до екрана авторизації

Модуль «Server» представляє собою набір функцій, які викликаються при отриманні відповідного запиту від програми «Pharmacy» та взаємодіє з БД.

Таблиця 4.2 – Перелік запитів до програми «Server».

Тип методу	**Посилання**	**Призначення**
Get	/api/products	Відправляє до програми «Pharmacy» інформацію про всі продукти
Post	/api/product	Записує у БД новий продукт
Post	/api/auth	Перевіряє існування запису про наданого оператора у БД і у випадку знаходження генерує токен, який записується оператору у БД та відправляється до програми «Pharmacy». Якщо оператора не було знайдено, відправляє помилку 403 до програми «Pharmacy»

Продовження таблиці 4.2

Post	/api/check-auth	Перевіряє чи належить токен, що надійшов, якому-небудь оператору. Якщо так, то відправляє до програми «Pharmacy» повідомлення про успіх, якщо ні, відправляє помилку 403 до програми «Pharmacy»
Post	/api/remove-token	Видаляє поле токен у оператора, якому належав токен, що надійшов
Patch	/api/product	Оновлює продукт, що надійшов у запиті
Patch	/api/products	Оновлює декілька продуктів, що надійшли у запиті
Delete	/api/product/:id	Видаляє продукт за вказаним id

Модуль «ServerController» представляє собою набір функцій, які викликаються при отриманні відповідного запиту від програми «Pharmacy» та взаємодіє з контролером.

Таблиця 4.3 – Перелік запитів до програми «ServerController».

Тип методу	**Посилання**	**Призначення**
Post	/api/output-product	Подає команди до контролера про видачу кожного товару з отриманого масиву товарів

4.3.3.3 Зв'язки між складовими частинами програми

Частини програм зв’язуються між собою за допомогою відправлення запитів програмою «Pharmacy» до програм «Server» та «Controller».

4.3.4 Використовувані технічні засоби

Програма експлуатується на персональному комп'ютері. Режим роботи – у формі віконного додатку, з яким взаємодіє оператор. При виникненні помилки використовується екран дисплея, клавіатура.

Вимоги до ПК:

– ПЕОМ з процесорами типу Intel Core під управлінням ОС Windows 7/8/8.1/10.

– Вільний дисковий простір не менше 1 ГБ.

– Оперативна пам'ять, не менше 1ГБ.

4.3.5 Виклик і завантаження

Після інсталяції програми на жорсткий диск необхідно запустити систему управління БД MongoDB, далі у вказаному порядку усі модулі: «controller.bat», «server.bat», «Pharmacy.exe».

4.3.6 Вхідні дані

Кожен програмний модуль має свої вхідні дані.

В якості вхідних даних модуль «Pharmacy» використовує відповіді на запити до інших модулів та дані введені оператором. У таблиці 4 представлений перелік змінних, в яких зберігаються вхідні дані модулю «Pharmacy».

Таблиця 4.4 – Перелік змінних, у яких зберігаються вхідні дані модулю «Pharmacy».

Ім'я	Тип	Призначення
searchText	String	Зберігає текст, який оператор вводить у строку пошуку
products	Array	Зберігає масив з усіма продуктами, що зберігаються в базі даних
basket	Array	Зберігає масив продуктів, які оператор додає до кошика
addProductDrawerVisible	bool	Змінна, що відповідає за відображення вікна з формою додавання або редагування продукту. Змінює своє значення залежно від дій оператора

Продовження таблиці 4.4

basketDrawerVisible	bool	Змінна, що відповідає за відображення вікна з таблицею продуктів доданих до кошика. Змінює своє значення залежно від дій оператора
initialValues	object	Зберігає об’єкт продукту, який оператор хоче відредагувати натиснувши на відповідну кнопку

В якості вхідних даних модуль «Server» використовує дані отримані у запитах від модулю «Pharmacy» та дані отримані як результат виконання команд до бази даних. У таблиці 5 представлений перелік змінних, в яких зберігаються вхідні дані до модулю «Server».

Таблиця 4.5 – Перелік змінних, в яких зберігаються вхідні дані до модулю «Server».

Тип методу	**Посилання**	**Змінна**	**Тип даних**	**Призначення**
Get	/api/products	products	Array	Масив, що містить дані про всі продукти у БД
Post	/api/product	data	Object	Об’єкт, який містить властивості товару, що додається
Post	/api/auth	userData	String	Строка, що містить об’єкт з логіном і паролем оператора у зашифрованому вгляді
		users	Array	Масив, що містить дані про всіх користувачів із БД
Post	/api/check-auth	token	String	Строка, що містить токен оператора для перевірки
		users	Array	Масив, що містить дані про всіх користувачів із БД
Post	/api/remove-token	token	String	Строка, що містить токен оператора, який необхідно видалити

Продовження таблиці 4.5

Patch	/api/product	data	Object	Об’єкт, який містить властивості товару, що редагується
Patch	/api/products	data	Array	Масив об’єктів, що редагуються
Delete	/api/product/:id	id	String	Строка, що містить ID товару, який необхідно видалити

В якості вхідних даних модуль «Controller» використовує дані отримані у запитах від модулю «Pharmacy» та дані отримані від контролера. У таблиці 6 представлений перелік змінних, в яких зберігаються вхідні дані із запитів до модулю «Controller».

Таблиця 4.6 – Перелік змінних, в яких зберігаються вхідні дані із запитів до модулю «Controller».

Тип методу	**Посилання**	**Змінна**	**Тип даних**	**Призначення**
Post	/api/output-product	data	Array	Масив об’єктів товарів для видачі

4.3.7 Вихідні дані

Кожен програмний модуль має свої вихідні дані.

В якості вихідних даних модуль «Pharmacy» використовує запити до інших модулів. У таблиці 7 представлений перелік змінних, в яких зберігаються вихідні дані, що відправляються до інших модулів.

Таблиця 4.7 – Перелік змінних, які відправляють вихідні дані від модулю «Pharmacy».

Тип методу	**Посилання**	**Змінна**	**Тип даних**	**Призначення**
Get	/api/products	products	Array	Масив, що містить дані про всі продукти у БД
Post	/api/product	data	Object	Об’єкт, який містить властивості товару, що додається
Post	/api/auth	userData	String	Строка, що містить об’єкт з логіном і паролем оператора у зашифрованому вигляді
Post	/api/check-auth	token	String	Строка, що містить токен оператора для перевірки
Post	/api/remove-token	token	String	Строка, що містить токен оператора, який необхідно видалити
Patch	/api/product	data	Object	Об’єкт, який містить властивості товару, що редагується
Patch	/api/products	item	Array	Об’єкт, який містить властивості товару, що редагується
Delete	/api/product/:id	id	String	Строка, що містить ID товару, який необхідно видалити
Post	/api/output-product	data	Array	Масив об’єктів товарів для видачі

В якості вихідних даних модуль «Server» використовує відповіді на запити та команди до БД. У таблиці 8 представлений перелік змінних, в яких зберігаються вихідні дані модулю «Server».

Таблиця 4.8 – Перелік змінних, в яких зберігаються вихідні дані модулю «Server».

Тип методу	**Посилання**	**Змінна**	**Тип даних**	**Призначення**
Get	/api/products	products	Array	Масив, що містить дані про всі продукти у БД
Post	/api/product	data	Object	Об’єкт, який містить властивості товару, що додається
Post	/api/auth	token	String	Строка, що містить токен оператора, що авторизувався
Post	/api/check-auth	isAuth	bool	Змінна, що показує зареєстрований оператор чи ні
Patch	/api/product	data	Object	Об’єкт, який містить властивості товару, що редагується
Patch	/api/products	data	Array	Масив об’єктів, що редагуються
Delete	/api/product/:id	id	String	Строка, що містить ID товару, який необхідно видалити

В якості вихідних даних модуль «Controller» використовує відповіді на запити та команди до контролера. У таблиці 9 представлений перелік змінних, в яких зберігаються вихідні дані модулю «Server».

Таблиця 4.9 – Перелік змінних, в яких зберігаються вихідні дані модулю «Controller».

Тип методу	**Посилання**	**Змінна**	**Тип даних**	**Призначення**
Post	/api/output-product	item	Array	Об’єкт, що містить властивості товару для видачі

4.4 Очікувані техніко-економічні показники

Впровадження даного програмного забезпечення дозволить оптимізувати роботу фармацевтів, що дозволить підвищити якість швидкість обслуговування клієнтів, що позитивно вплине на прибуток компанії.

4.5 Висновки до розділу

При розробленні програмного забезпечення був описаний алгоритм програми для збереження і редагування бази даних медикаментів, візуалізації даних, що зберігаються, та управління роботизованою шафою в аптеках системи комплексу роботизованих аптек ТМ «Копійка» і реалізований у вигляді пакету програм на мові JavaScript з бібліотеками React та Node.js. Була описана структура, функції і зміні програми, вхідні та вихідні дані.

5 ЕКСПЕРИМЕНТАЛЬНИЙ РОЗДІЛ

5.1 Розробка математичної моделі мережі як замкнутої системи масового обслуговування

Відповідно до структурної схеми комп'ютерної мережі та її імітаційної моделі розроблено структуру математичної моделі комп'ютерної мережі як замкнутої системи масового обслуговування (Рис. 5.1).

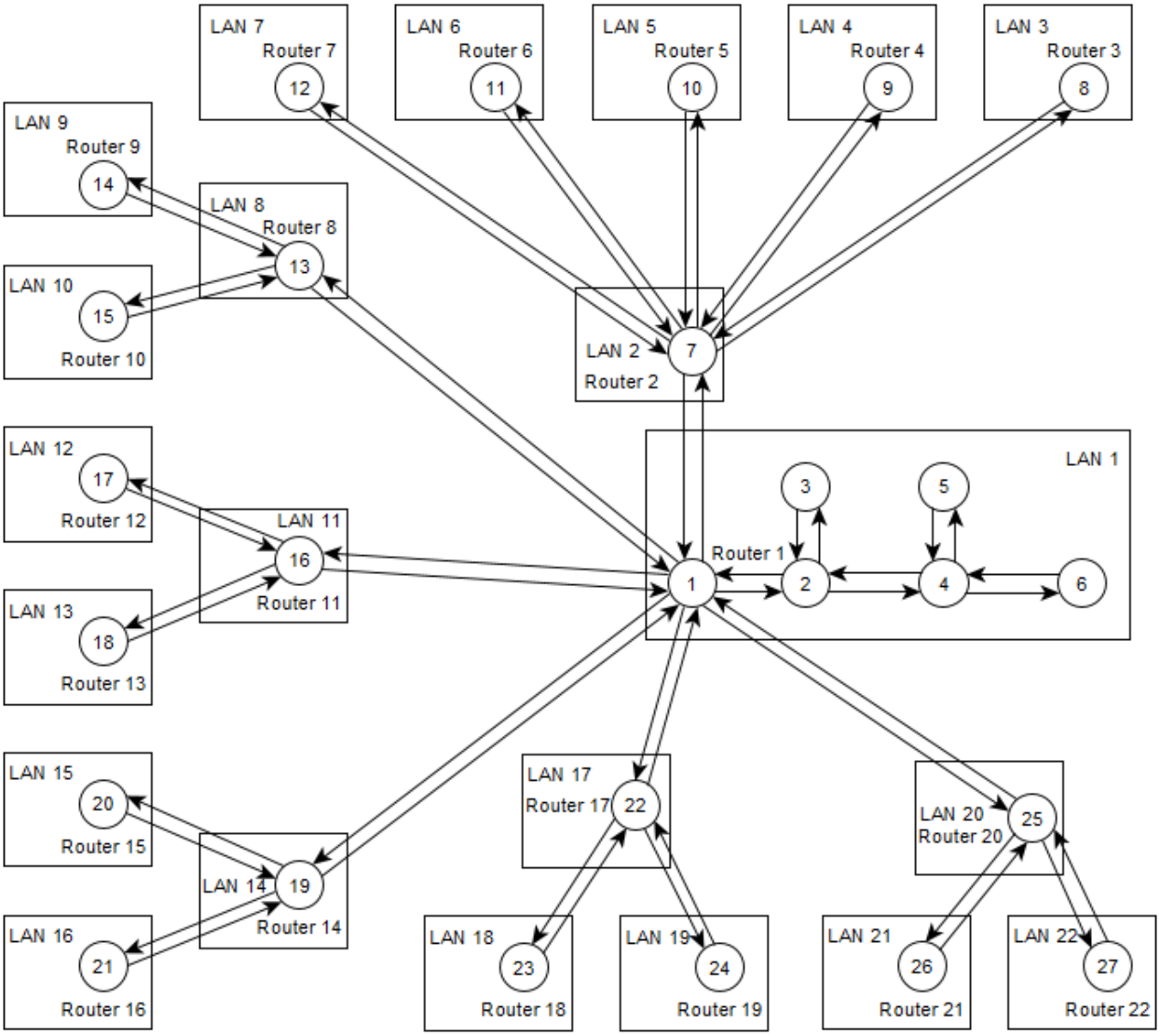


Рисунок 5.1 – Структура математичної моделі комп'ютерної мережі

В структурі моделі комп'ютерної мережі вузли 2,4,6 – це комутатори, що обслуговують локальні мережі.

Вузли 1, 7-27 – маршрутизатори.

Зв’язки між елементами структури – це вірогідність передачі пакета від одного до другого вузла. Кожен вузол – це система масового обслуговування.

Вірогідність того що вузол зв’язується сам з собою дорівнює нулю. Вірогідність того що вузли зв’яжуться між собою приблизно рівна.

Вірогідності зв’язку вузлів між собою описані у маршрутній матриці:

Pr :=

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26
0	0	0.16	0	0	0	0	0.14	0	0	0	0	0	0.14	0	0	0.14	0	0	0.14	0	0	0.14	0	0	0.14	0	0
1	0.4	0	0.3	0.3	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
2	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
3	0	0.4	0	0	0.3	0.3	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
4	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
5	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
6	0.2	0	0	0	0	0	0	0.16	0.16	0.16	0.16	0.16	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
7	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
8	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
9	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
10	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
11	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
12	0.5	0	0	0	0	0	0	0	0	0	0	0	0	0.25	0.25	0	0	0	0	0	0	0	0	0	0	0	0
13	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0
14	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0
15	0.5	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0.25	0.25	0	0	0	0	0	0	0	0	0
16	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0
17	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0
18	0.5	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0.25	0.25	0	0	0	0	0	0
19	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0
20	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0
21	0.5	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0.25	0.25	0	0	0
22	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0
23	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0
24	0.5	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0.25	0.25
25	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0
26	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0

Матриця стовпчик, що показує час обробки одного повідомлення в відповідному вузлі:

τ =

	0
0	5
1	5
2	5
3	5
4	5
5	5
6	5
7	5
8	5
9	5
10	5
11	5
12	5
13	5
14	5
15	5
16	5
17	5
18	5
19	5
20	5
21	5
22	5
23	5
24	5
25	5
26	5

Для початкових умов моделювання задаємо однаковий час обробки одного повідомлення для усіх вузлів.

5.2 Розрахунок параметрів мережі по її моделі

Далі методом Гауса розраховується матриця стовпчик з передаточними коефіцієнтами:

$$e := \begin{pmatrix} 1 \\ 0.4 \\ 0.12 \\ 0.3 \\ 0.09 \\ 0.09 \\ 0.7 \\ 0.112 \\ 0.112 \\ 0.112 \\ 0.112 \\ 0.112 \\ 0.28 \\ 0.07 \\ 0.07 \\ 0.28 \\ 0.07 \\ 0.07 \\ 0.28 \\ 0.07 \\ 0.07 \\ 0.28 \\ 0.07 \\ 0.07 \\ 0.28 \\ 0.07 \\ 0.07 \end{pmatrix}$$

Задаємо матрицю m, коефіцієнти якої означають кількість конвеєрів обробки пакетів в кожному із вузлів системи масового обслуговування:

m =

	0
0	1
1	1
2	1
3	1
4	1
5	1
6	1
7	1
8	1
9	1
10	1
11	1
12	1
13	1
14	1
15	1
16	1
17	1
18	1
19	1
20	1
21	1
22	1
23	1
24	1
25	1
26	1

Для розрахунків приймаємо, що в кожному пристрої знаходиться лише один конвеєр обробки пакетів.

Матриця B це матриця яка визначає з якою вірогідністю у відповідному вузлі (строчки) буде знаходитися в очікуванні обробки пакетів (номер стовпчика).

B =

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
0	$8.751\cdot10^{-5}$	$1.25\cdot10^{-4}$	$1.786\cdot10^{-4}$	$2.551\cdot10^{-4}$	$3.644\cdot10^{-4}$	$5.205\cdot10^{-4}$	$7.433\cdot10^{-4}$	$1.061\cdot10^{-3}$	$1.515\cdot10^{-3}$	$2.162\cdot10^{-3}$	$3.082\cdot10^{-3}$	$4.39\cdot10^{-3}$	$6.24\cdot10^{-3}$	$8.846\cdot10^{-3}$	0.012	0.018	0.024
1	0.6	0.24	0.096	0.038	0.015	$6.14\cdot10^{-3}$	$2.455\cdot10^{-3}$	$9.815\cdot10^{-4}$	$3.923\cdot10^{-4}$	$1.568\cdot10^{-4}$	$6.26\cdot10^{-5}$	$2.499\cdot10^{-5}$	$9.962\cdot10^{-6}$	$3.967\cdot10^{-6}$	$1.576\cdot10^{-6}$	$6.247\cdot10^{-7}$	$2.465\cdot10^{-7}$
2	0.88	0.106	0.013	$1.52\cdot10^{-3}$	$1.824\cdot10^{-4}$	$2.188\cdot10^{-5}$	$2.624\cdot10^{-6}$	$3.146\cdot10^{-7}$	$3.772\cdot10^{-8}$	$4.52\cdot10^{-9}$	$5.413\cdot10^{-10}$	$6.476\cdot10^{-11}$	$7.739\cdot10^{-12}$	$9.232\cdot10^{-13}$	$1.098\cdot10^{-13}$	$1.302\cdot10^{-14}$	$1.535\cdot10^{-15}$
3	0.7	0.21	0.063	0.019	$5.667\cdot10^{-3}$	$1.7\cdot10^{-3}$	$5.097\cdot10^{-4}$	$1.528\cdot10^{-4}$	$4.58\cdot10^{-5}$	$1.372\cdot10^{-5}$	$4.11\cdot10^{-6}$	$1.23\cdot10^{-6}$	$3.676\cdot10^{-7}$	$1.097\cdot10^{-7}$	$3.267\cdot10^{-8}$	$9.697\cdot10^{-9}$	$2.865\cdot10^{-9}$
4	0.91	0.082	$7.37\cdot10^{-3}$	$6.631\cdot10^{-4}$	$5.967\cdot10^{-5}$	$5.368\cdot10^{-6}$	$4.829\cdot10^{-7}$	$4.343\cdot10^{-8}$	$3.905\cdot10^{-9}$	$3.509\cdot10^{-10}$	$3.152\cdot10^{-11}$	$2.828\cdot10^{-12}$	$2.534\cdot10^{-13}$	$2.267\cdot10^{-14}$	$2.023\cdot10^{-15}$	0	0
5	0.91	0.082	$7.37\cdot10^{-3}$	$6.631\cdot10^{-4}$	$5.967\cdot10^{-5}$	$5.368\cdot10^{-6}$	$4.829\cdot10^{-7}$	$4.343\cdot10^{-8}$	$3.905\cdot10^{-9}$	$3.509\cdot10^{-10}$	$3.152\cdot10^{-11}$	$2.828\cdot10^{-12}$	$2.534\cdot10^{-13}$	$2.267\cdot10^{-14}$	$2.023\cdot10^{-15}$	0	0
6	0.3	0.21	0.147	0.103	0.072	0.05	0.035	0.025	0.017	0.012	$8.476\cdot10^{-3}$	$5.933\cdot10^{-3}$	$4.153\cdot10^{-3}$	$2.906\cdot10^{-3}$	$2.034\cdot10^{-3}$	$1.423\cdot10^{-3}$	$9.943\cdot10^{-4}$
7	0.888	0.099	0.011	$1.247\cdot10^{-3}$	$1.396\cdot10^{-4}$	$1.563\cdot10^{-5}$	$1.75\cdot10^{-6}$	$1.959\cdot10^{-7}$	$2.192\cdot10^{-8}$	$2.451\cdot10^{-9}$	$2.74\cdot10^{-10}$	$3.059\cdot10^{-11}$	$3.412\cdot10^{-12}$	$3.799\cdot10^{-13}$	$4.218\cdot10^{-14}$	$4.667\cdot10^{-15}$	0
8	0.888	0.099	0.011	$1.247\cdot10^{-3}$	$1.396\cdot10^{-4}$	$1.563\cdot10^{-5}$	$1.75\cdot10^{-6}$	$1.959\cdot10^{-7}$	$2.192\cdot10^{-8}$	$2.451\cdot10^{-9}$	$2.74\cdot10^{-10}$	$3.059\cdot10^{-11}$	$3.412\cdot10^{-12}$	$3.799\cdot10^{-13}$	$4.218\cdot10^{-14}$	$4.667\cdot10^{-15}$	0
9	0.888	0.099	0.011	$1.247\cdot10^{-3}$	$1.396\cdot10^{-4}$	$1.563\cdot10^{-5}$	$1.75\cdot10^{-6}$	$1.959\cdot10^{-7}$	$2.192\cdot10^{-8}$	$2.451\cdot10^{-9}$	$2.74\cdot10^{-10}$	$3.059\cdot10^{-11}$	$3.412\cdot10^{-12}$	$3.799\cdot10^{-13}$	$4.218\cdot10^{-14}$	$4.667\cdot10^{-15}$	0
10	0.888	0.099	0.011	$1.247\cdot10^{-3}$	$1.396\cdot10^{-4}$	$1.563\cdot10^{-5}$	$1.75\cdot10^{-6}$	$1.959\cdot10^{-7}$	$2.192\cdot10^{-8}$	$2.451\cdot10^{-9}$	$2.74\cdot10^{-10}$	$3.059\cdot10^{-11}$	$3.412\cdot10^{-12}$	$3.799\cdot10^{-13}$	$4.218\cdot10^{-14}$	$4.667\cdot10^{-15}$	0
11	0.888	0.099	0.011	$1.247\cdot10^{-3}$	$1.396\cdot10^{-4}$	$1.563\cdot10^{-5}$	$1.75\cdot10^{-6}$	$1.959\cdot10^{-7}$	$2.192\cdot10^{-8}$	$2.451\cdot10^{-9}$	$2.74\cdot10^{-10}$	$3.059\cdot10^{-11}$	$3.412\cdot10^{-12}$	$3.799\cdot10^{-13}$	$4.218\cdot10^{-14}$	$4.667\cdot10^{-15}$	0
12	0.72	0.202	0.056	0.016	$4.423\cdot10^{-3}$	$1.238\cdot10^{-3}$	$3.465\cdot10^{-4}$	$9.697\cdot10^{-5}$	$2.713\cdot10^{-5}$	$7.586\cdot10^{-6}$	$2.12\cdot10^{-6}$	$5.921\cdot10^{-7}$	$1.652\cdot10^{-7}$	$4.601\cdot10^{-8}$	$1.278\cdot10^{-8}$	$3.541\cdot10^{-9}$	$9.762\cdot10^{-10}$
13	0.93	0.065	$4.556\cdot10^{-3}$	$3.189\cdot10^{-4}$	$2.232\cdot10^{-5}$	$1.562\cdot10^{-6}$	$1.093\cdot10^{-7}$	$7.642\cdot10^{-9}$	$5.344\cdot10^{-10}$	$3.735\cdot10^{-11}$	$2.609\cdot10^{-12}$	$1.821\cdot10^{-13}$	$1.269\cdot10^{-14}$	0	0	0	0
14	0.93	0.065	$4.556\cdot10^{-3}$	$3.189\cdot10^{-4}$	$2.232\cdot10^{-5}$	$1.562\cdot10^{-6}$	$1.093\cdot10^{-7}$	$7.642\cdot10^{-9}$	$5.344\cdot10^{-10}$	$3.735\cdot10^{-11}$	$2.609\cdot10^{-12}$	$1.821\cdot10^{-13}$	$1.269\cdot10^{-14}$	0	0	0	0
15	0.72	0.202	0.056	0.016	$4.423\cdot10^{-3}$	$1.238\cdot10^{-3}$	$3.465\cdot10^{-4}$	$9.697\cdot10^{-5}$	$2.713\cdot10^{-5}$	$7.586\cdot10^{-6}$	$2.12\cdot10^{-6}$	$5.921\cdot10^{-7}$	$1.652\cdot10^{-7}$	$4.601\cdot10^{-8}$	$1.278\cdot10^{-8}$	$3.541\cdot10^{-9}$	...

Відповідно до алгоритму Бузена розраховуються середні значення по кожному із вузлів мережі.

Інтенсивність вхідного потоку пакетів у кожному вузлі:

$\lambda =$

	0
0	0.186
1	0.074
2	0.022
3	0.056
4	0.017
5	0.017
6	0.13
7	0.021
8	0.021
9	0.021
10	0.021
11	0.021
12	0.052
13	0.013
14	0.013
15	0.052
16	0.013
17	0.013
18	0.052
19	0.013
20	0.013
21	0.052
22	0.013
23	0.013
24	0.052
25	0.013
26	0.013

Середнє число пакетів що чекають на обробку в кожному вузлі:

$L =$

	0
0	21.91
1	0.667
2	0.136
3	0.429
4	0.099
5	0.099
6	2.329
7	0.126
8	0.126
9	0.126
10	0.126
11	0.126
12	0.389
13	0.075
14	0.075
15	0.389
16	0.075
17	0.075
18	0.389
19	0.075
20	0.075
21	0.389
22	0.075
23	0.075
24	0.389
25	0.075
26	0.081

Середній час обробки пакета в вузлі:

t =

	0
0	117.804
1	8.959
2	6.109
3	7.68
4	5.908
5	5.908
6	17.888
7	6.054
8	6.054
9	6.054
10	6.054
11	6.054
12	7.467
13	5.781
14	5.781
15	7.467
16	5.781
17	5.781
18	7.467
19	5.781
20	5.781
21	7.467
22	5.781
23	5.781
24	7.467
25	5.781
26	6.216

5.2.1 Параметри роботи мережі без впливу шкідливого програмного забезпечення

Роботу комп’ютерної мережі можна охарактеризувати наступними параметрами:

– інтенсивність потоку, що входить у вузол;

– середній час перебування пакета у вузлі;

– середня кількість пакетів які знаходяться у вузлі.

Припустимо, що у мережі циркулює 5 пакетів, а кожен вузол має один конвеєр обробки, тоді час обробки пакетів у всіх вузлах мережі буде однаковим і займатиме 5 часових одиниць, де одна часова одиниця дорівнює одній мілісекунді. За таких даних отримаємо усереднені характеристики кожного вузла показані на рисунках 5.2-5.4.

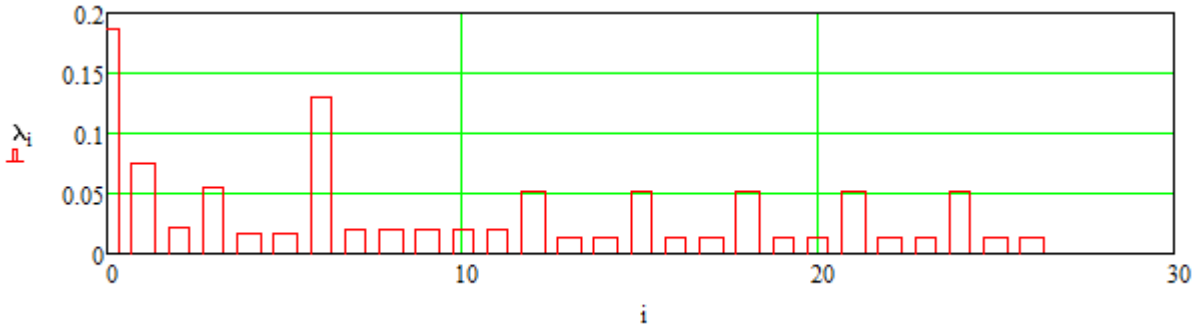


Рисунок 5.2 – Інтенсивність потоку, що входить у вузол

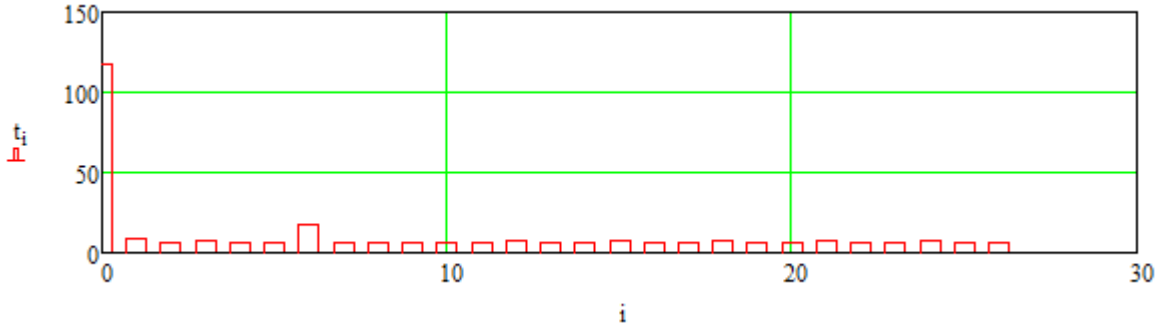


Рисунок 5.3 – Середній час перебування пакета у вузлі

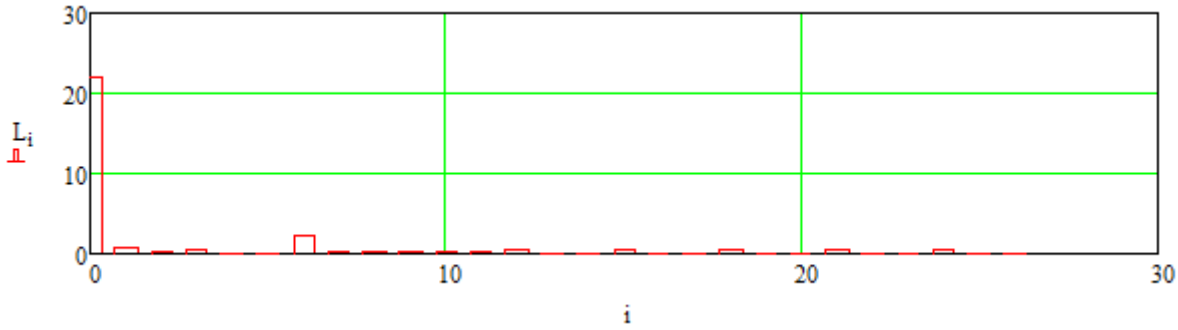


Рисунок 5.4 – Середня кількість пакетів які знаходяться у вузлі

Як бачимо, в цілому у всіх вузлах мережі, усереднені параметри показують, що усі повідомлення обробляються швидко і без черги. Виключенням є вузол 1 – маршрутизатор у центральному офісі.

Рисунок 5.5 показує з якою вірогідністю у вузлах мережі буде черга.

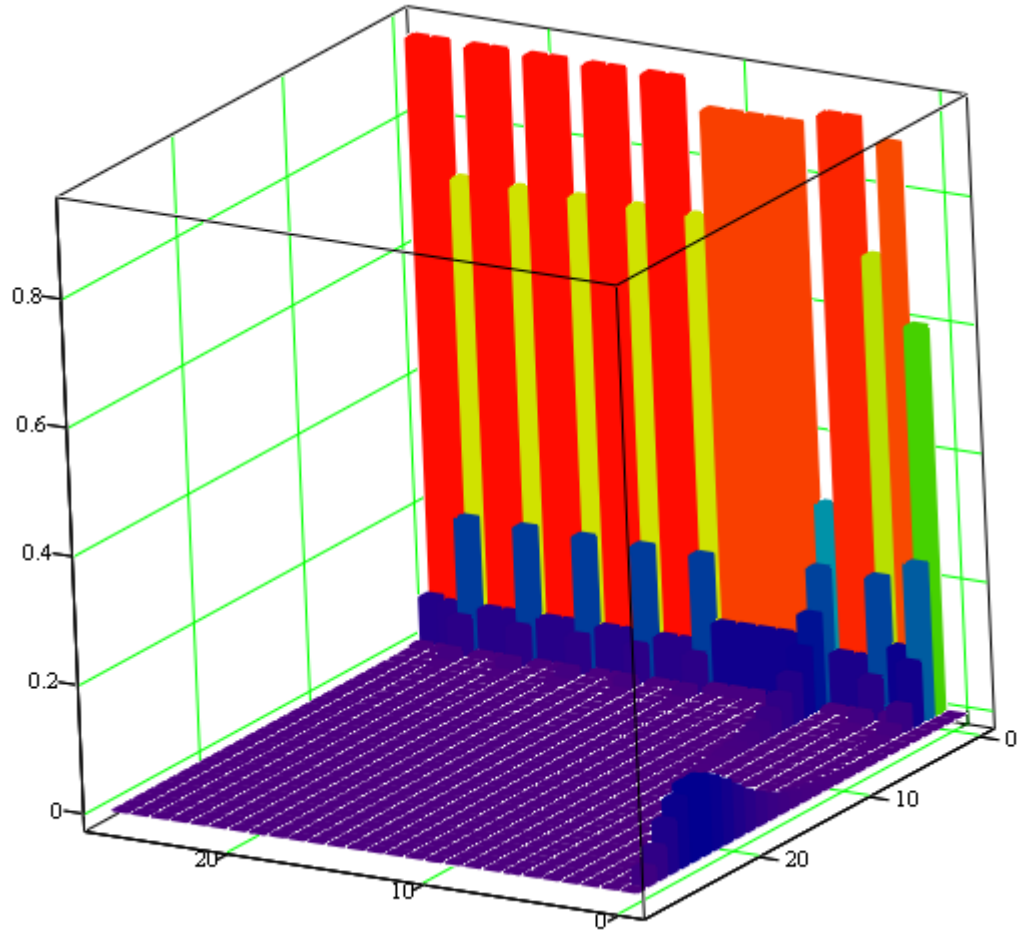


Рисунок 5.5 – Вірогідність черги у вузлах мережі

Можемо зробити попередній висновок про те, що при заданих параметрах маршрутизатори 1 та 7 мережі є найбільш проблемними.

5.2.2 Параметри роботи мережі під впливом вірусних програм

Деякі види вірусних програм можуть створювати додатковий трафік і таким чином устворювати додаткове навантаження на мережу уповільнюои її роботу. При генерації занадто великої кількості мережевих запитів комп’ютерна мережа може перестати функціонувати взагалі

Для моделювання подібної ситуації задаємо кількість запитів у мережі збільшуємо вдвічі, N=60.

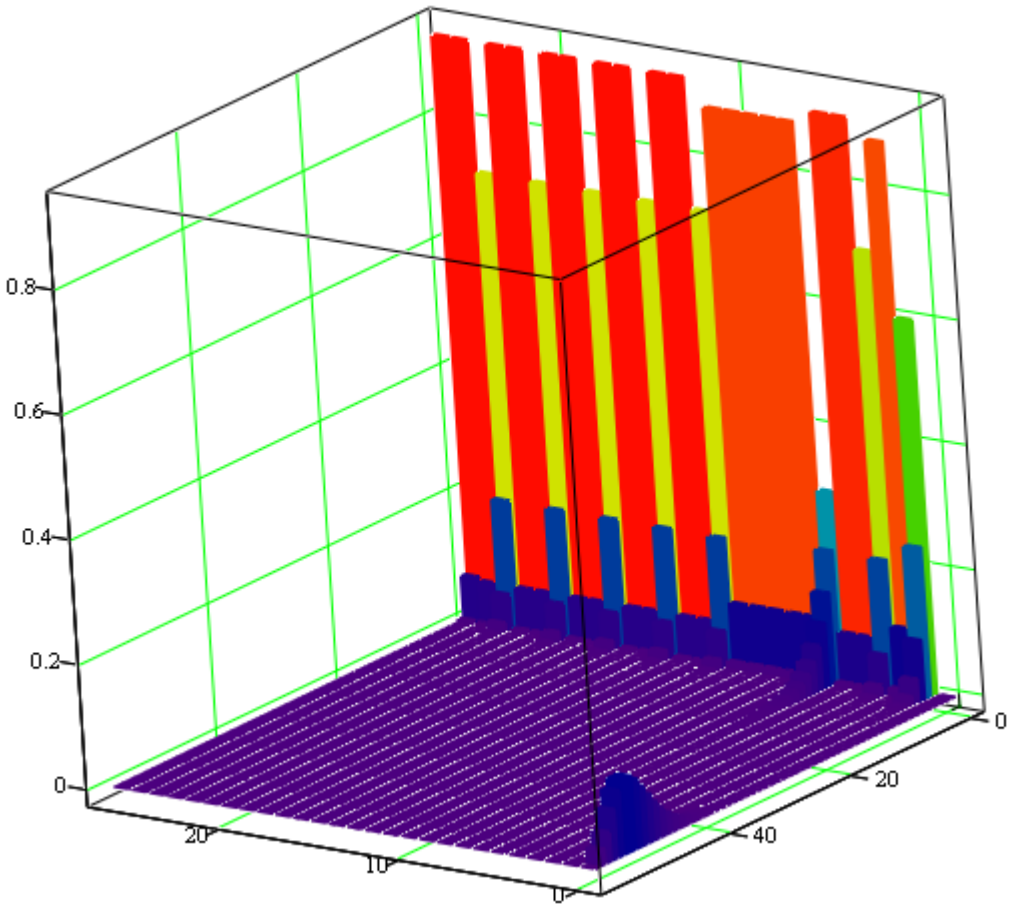


Рисунок 5.6 – Вірогідність черги у вузлах якщо в мережі циркулює 60 пакетів

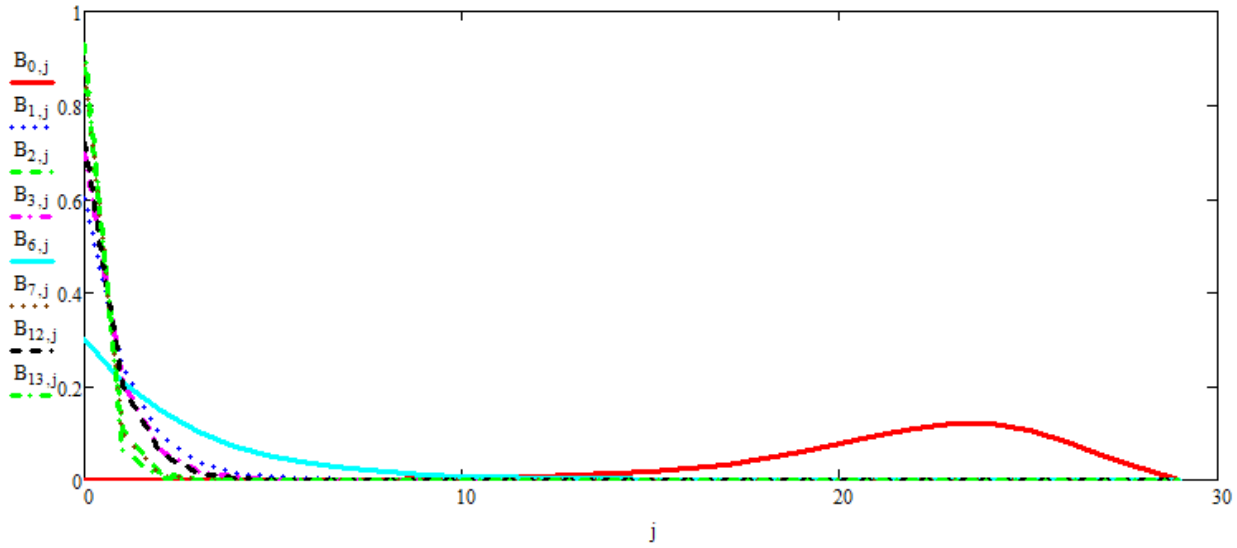


Рисунок 5.7 – Вірогідність черги у вузлах якщо в мережі циркулює 30 пакетів

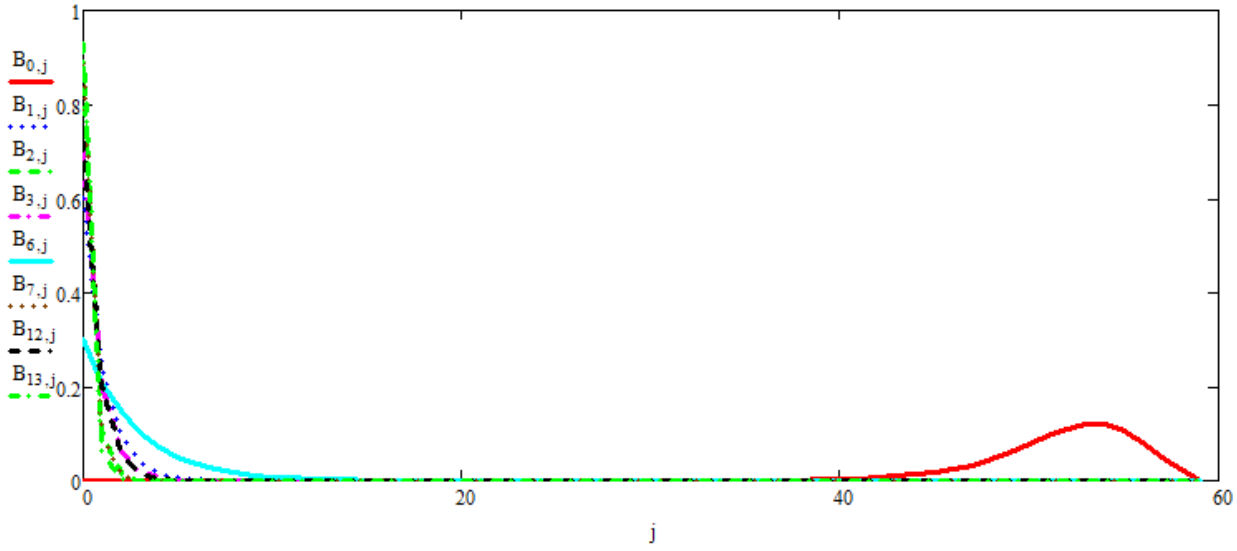


Рисунок 5.8 – Вірогідність черги у вузлах якщо в мережі циркулює 60 пакетів

Результати моделювання показують, що збільшення вдвічі кількості пакетів, що циркулюють в мережі приводить до того, що тільки у маршрутизатора №1 зростає вірогідність того що в черзі уже будуть знаходитися 2 пакета.

5.3 Робота мережі із скоригованими характеристиками проблемних вузлів

Корекція характеристик вузла №1 проводиться за рахунок підвищення швидкості обробки пакетів.

$\tau =$

	0
0	2
1	5
2	5
3	5
4	5
5	5
6	5
7	5
8	5
9	5
10	5
11	5
12	5
13	5
14	5
15	5
16	5
17	5
18	...

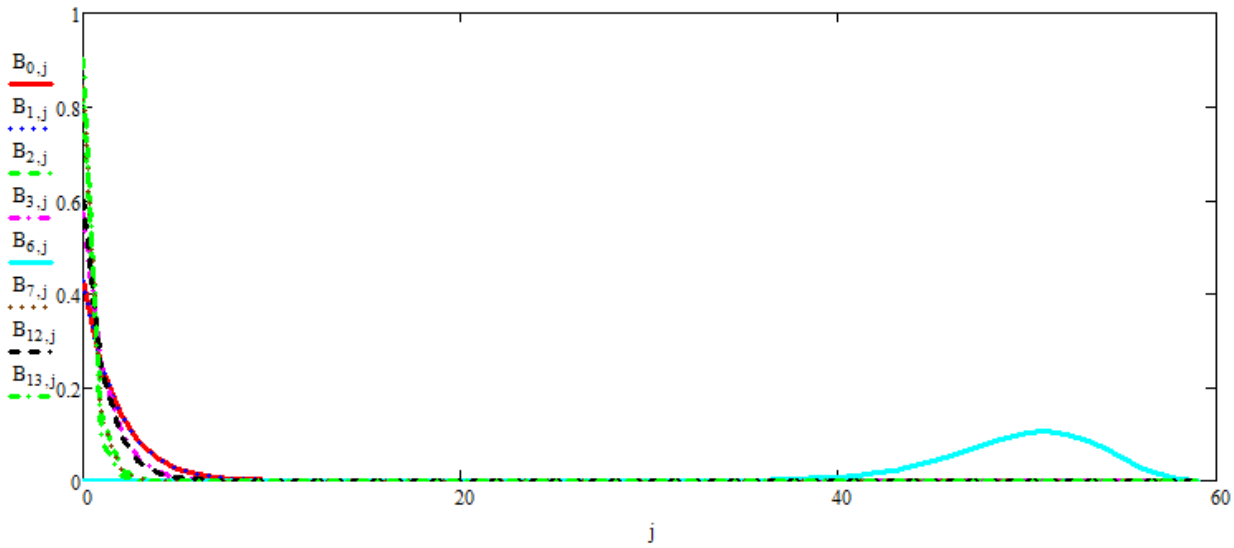


Рисунок 5.9 – Вірогідність черги у вузлах якщо в мережі циркулює 60 пакетів після першої корекції

Бачимо, що тепер у маршрутизатора №7 зростає вірогідність того що в черзі уже будуть знаходитися 2 пакета. Скорегуємо його характеристики.

$\tau =$

	0
0	2
1	5
2	5
3	5
4	5
5	5
6	3
7	5
8	5
9	5
10	5
11	5
12	5
13	5
14	5
15	5
16	5
17	5
18	...

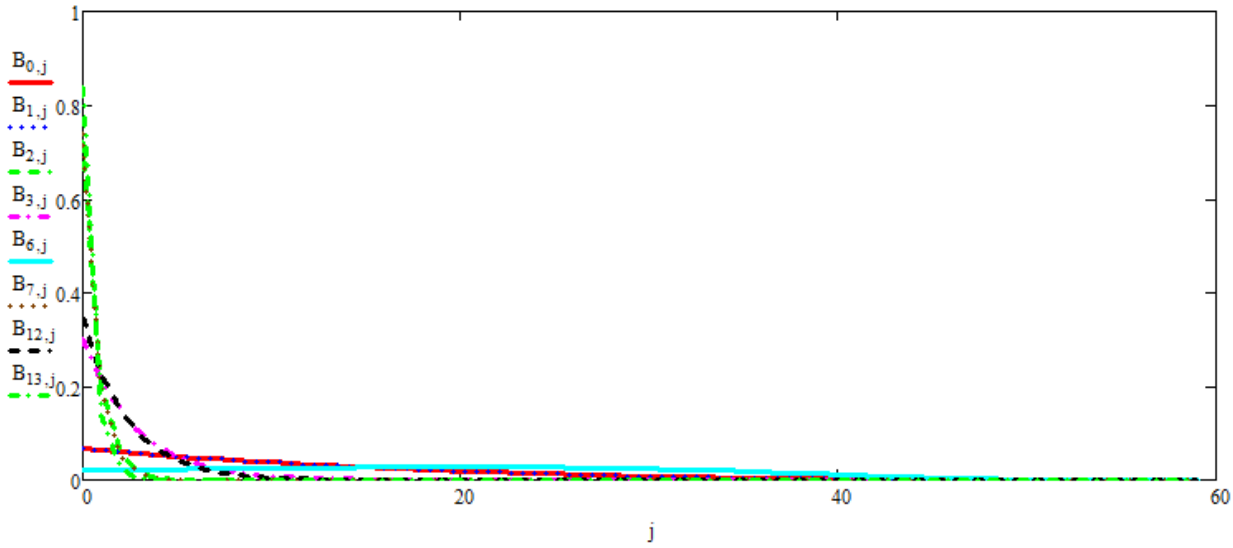


Рисунок 5.10 – Вірогідність черги у вузлах якщо в мережі циркулює 60 пакетів після другої корекції

Відповідно до змін розраховані як усереднені так і ймовірнісні характеристики.

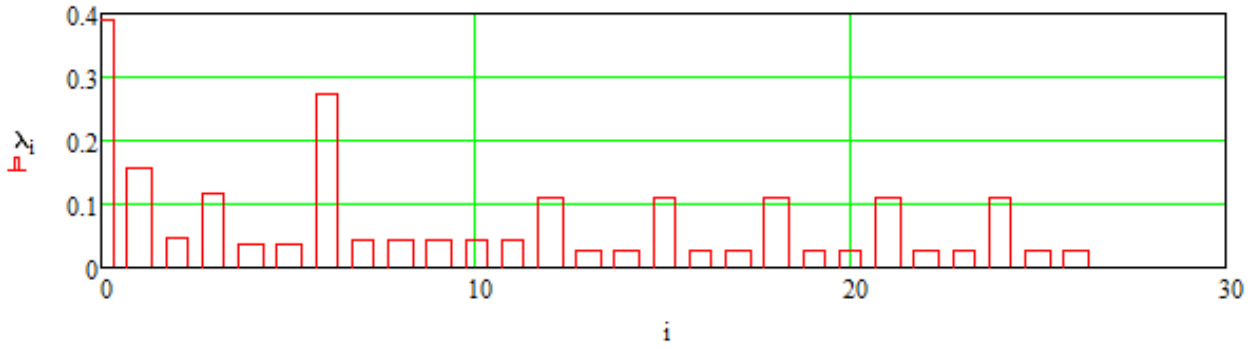


Рисунок 5.11 – Інтенсивність потоку, що входить у вузол після корекцій

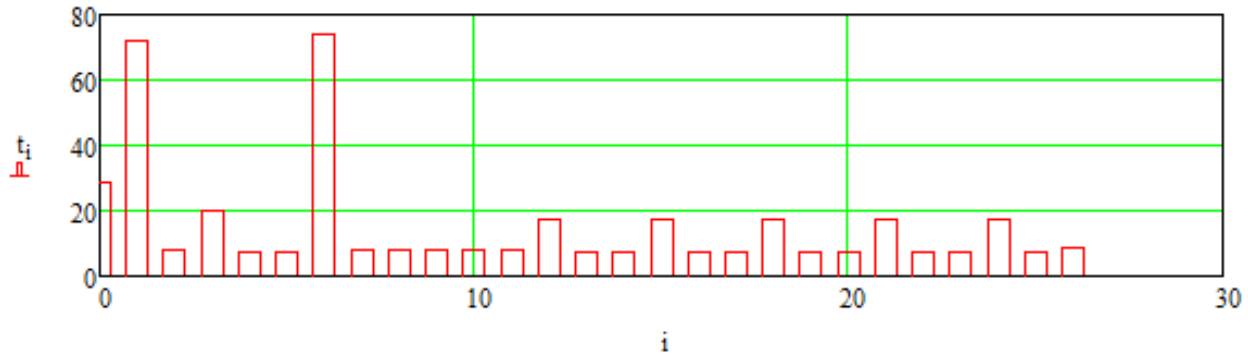


Рисунок 5.12 – Середній час перебування пакета у вузлі після корекцій

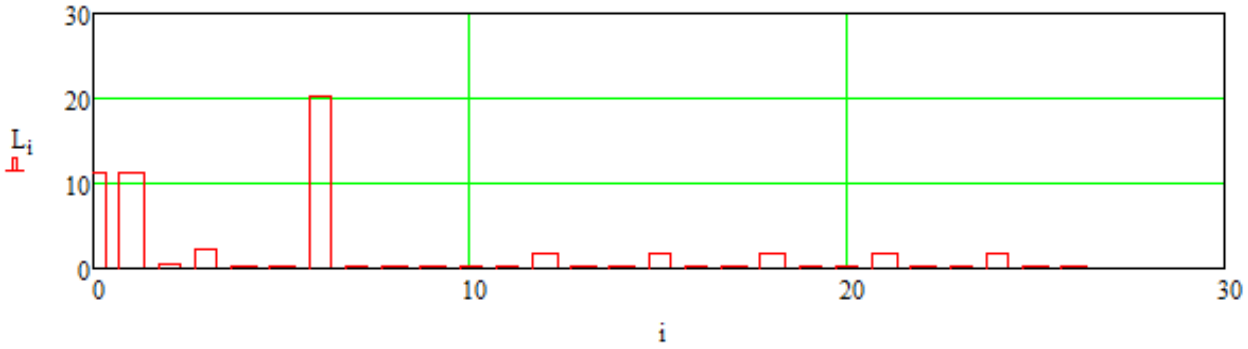


Рисунок 5.13 – Середня кількість пакетів які знаходяться у вузлі після корекцій

Вірогідність того, що в вузлах мережі може виникнути черга значно знизилася (Рис. 5.14)

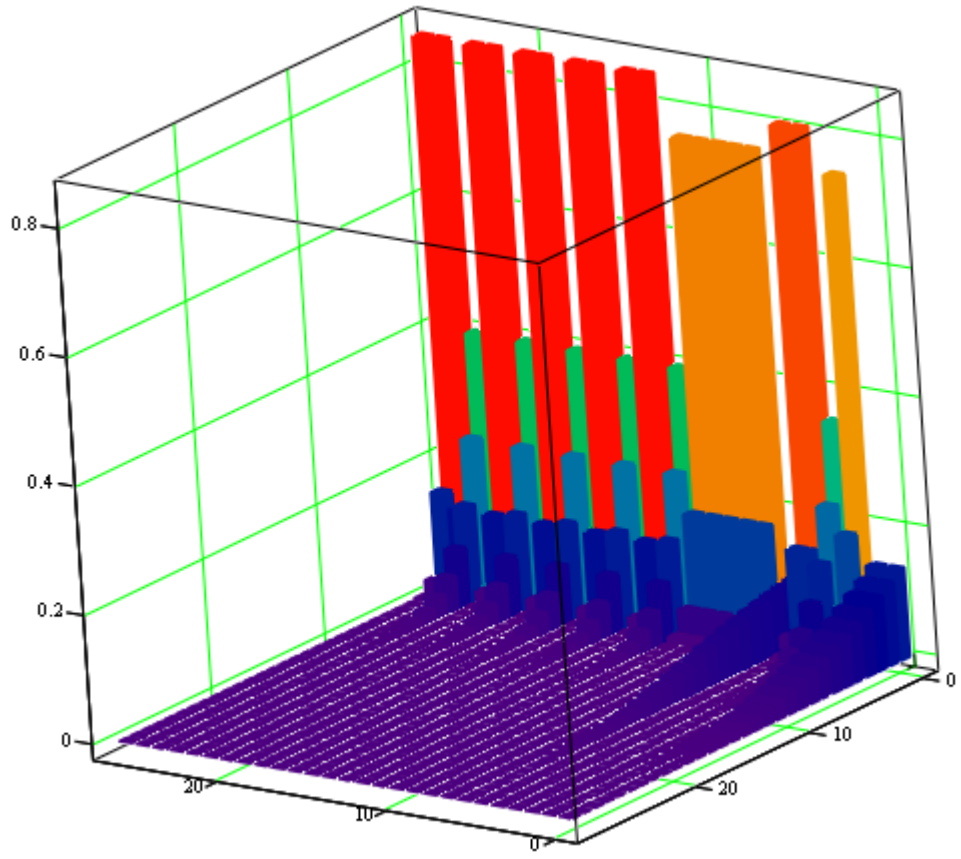


Рисунок 5.14 – Вірогідність черги у вузлах якщо в мережі циркулює 30 пакетів після корекцій

Якщо в мережі із скоригованими характеристиками буде циркулювати 20 пакетів тоді вірогідність того, що в вузлах не буде черги зменшується для роутерів (Рис. 5.15).

Підвищення швидкості обробки пакетів у вузлах, які показали найменшу стійкість до перевантаження дозволили певним чином покращити характеристики мережі.

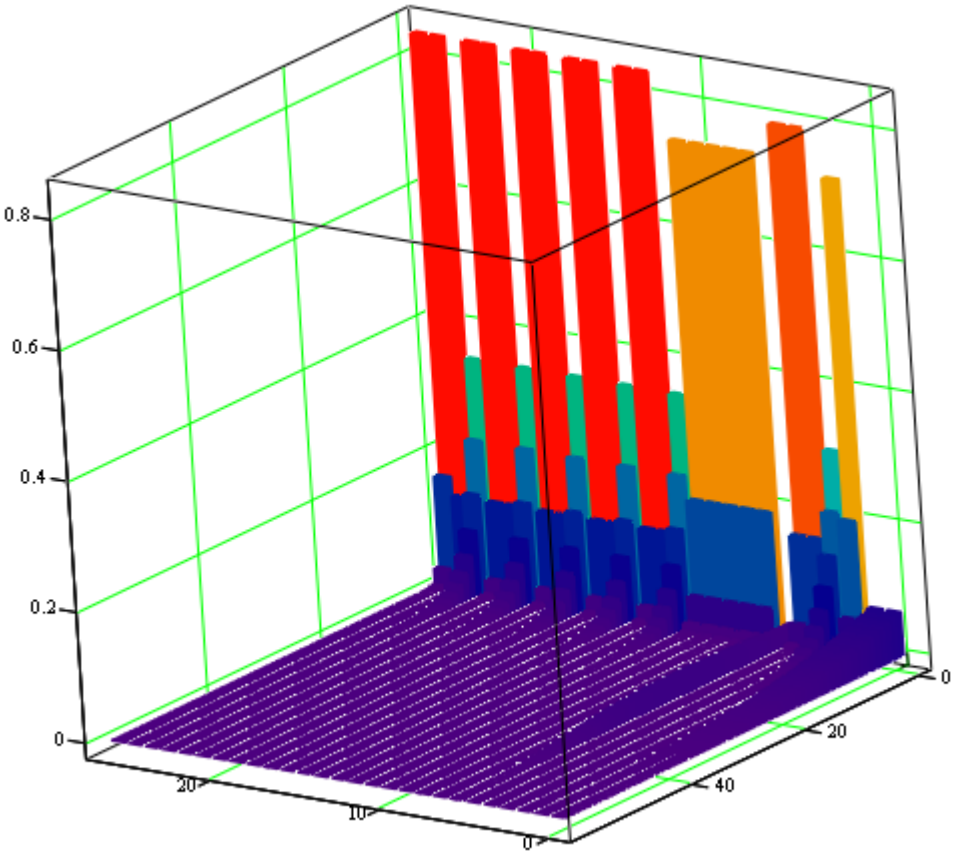


Рисунок 5.15 – Вірогідність черги у вузлах якщо в мережі циркулює 60 пакетів після корекцій

5.4 Висновки до розділу

Під час розробки математичної моделі мережі як замкнутої системи масового обслуговування були розраховані параметри інтенсивності потоку, середнього часу перебування пакета у вузлі, середньої кількості пакетів які знаходяться у вузлі у звичайному режимі роботи та у режимі та під впливом вірусних програм. Аналіз отриманих параметрів дозволив виявити проблемні вузли та скорегувати їх покращивши роботу комп’ютерної мережі.

ВИСНОВКИ

Під час виконання кваліфікаційної роботи була вирішена науково-практична задача розробки та обґрунтування програмно-технічних засобів комп’ютерної системи комплексу роботизованих аптек ТМ «Копійка».

Основні висновки і результати роботи полягають у наступному:

1. Досліджені основні типи АСЗ, їх влаштування та принципи роботи. На основі проведених досліджень було вирішено та обґрунтовано за основу майбутньої роботизованої шафи взяти гравітаційний тип.

2. Для комп’ютерної системи комплексу підібрано обладнання з необхідними характеристиками

3. Розроблені функціональна та принципова схеми роботизованої шафи, підібрана та обґрунтована відповідна елементна база.

4. Згідно розробленого алгоритму роботи був створений пакет програм мовою JavaScript, який призначений для керування роботизованою шафою та взаємодією з БД.

5. Проведено моделювання комп’ютерної мережі комплексу, виявлені проблемні вузли та скореговано їх параметри.

Кваліфікаційна робота виконана повністю відповідно до теми і завдання, оформлена відповідно до нормативних документів і методичного керівництва.

Цілі, поставлені перед кваліфікаційною роботою, повністю виконані.

ПЕРЕЛІК ПОСИЛАНЬ

1. Дипломування. Методичні рекомендації до виконання кваліфікаційної роботи магістра студентами галузі знань 12 Інформаційні технології спеціальності 123 Комп'ютерна інженерія / Л.І. Цвіркун, С.М. Ткаченко. ; М-во освіти і науки України, Нац. техн. ун-т «Дніпровська політехніка». – Дніпро: НТУ «ДП», 2019. – 41 с.

2. ГОСТ 19.201-78. ЕСПД. Єдина система програмної документації. Технічне завдання. Вимоги до змістом і оформленням. - М.: Держстандарт, 1982. - 128 с.

3. ГОСТ 19.404-79. ЕСПД Єдина система програмної документації. Пояснювальна записка. Вимоги до змісту та оформлення. - М.: Держстандарт, 1982. - 128 с.

4. Автоматизированные складские системы: виды, преимущества, особенности и сферы применения [Електронний ресурс]. URL: https://www.ekam.ru/blogs/pos/avtomatizirovannye-skladskie-sistemy.

5. Современный СКЛАД: снижение издержек, повышение эффективности, оптимизация производства [Електронний ресурс]. URL: https://sovtest-ate.com/news/publications/sovremennyy-sklad_-snizhenie-izderzhek_-povyshenie-effektivnosti_-optimizatsiya-proizvodstva/.

6. Основы JavaScript [Електронний ресурс]. URL: https://developer.mozilla.org/ru/docs/Learn/Getting_started_with_the_web/JavaScript_basics.

7. Node.js [Електронний ресурс]. URL: https://ru.wikipedia.org/wiki/Node.js.

8. Реферат – Життєвий цикл проекту, моделі життєвого циклу [Електронний ресурс]. URL: https://ukrdoc.com.ua/text/3918/index-1.html.

9. D-link DIR-841 [Електронний ресурс]. URL: https://www.dlink.ru/ru/products/5/2350.html.

10. Cisco RV160 VPN Router and RV160W Wireless-AC VPN Router Data Sheet [Електронний ресурс]. URL: https://www.cisco.com/c/en/us/products/collateral/routers/rv160-vpn-router/datasheet-c78-741410.html.

11. Cisco Catalyst 1000 Series Switches Data Sheet [Електронний ресурс]. URL: https://www.cisco.com/c/en/us/products/collateral/switches/catalyst-1000-series-switches/nb-06-cat1k-ser-switch-ds-cte-en.html?oid=otren019232.

12. Spec Sheet Cisco UCS B200 M6 Blade Server [Електронний документ]. URL: https://www.cisco.com/c/dam/en/us/products/collateral/servers-unified-computing/ucs-b-series-blade-servers/b200m6-specsheet.pdf.

13. Arduino Uno Data Sheet [Електронний документ]. URL: https://www.farnell.com/datasheets/1682209.pdf.

14. L298 Data Sheet [Електронний документ]. URL: https://www.sparkfun.com/datasheets/Robotics/L298_H_Bridge.pdf.

15. EAW – Absolute Contacting Encoder (ACE™) Data Sheet [Електронний документ]. URL: https://www.bourns.com/pdfs/ace.pdf.

16. PCF8574 Remote 8-Bit I/O Expander for I2C Bus Data Sheet [Електронний документ]. URL: https://www.ti.com/lit/ds/symlink/pcf8574.pdf.

17. Расширитель на 8 Входов/Выходов (Trema-модуль) [Електронний ресурс]. URL: https://wiki.iarduino.ru/page/rasshiritel-na-8-vhodov-vyhodov-trema-modul/#h3_2.

18. StandardFirmataPlus.ino [Електронний ресурс]. URL: https://github.com/firmata/arduino/blob/master/examples/StandardFirmataPlus/StandardFirmataPlus.ino.

19. Огляд сучасних методів моделювання та аналізу мереж [Електронний ресурс]. URL: https://megapredmet.ru/1-40267.html.

ДОДАТОК А – ТЕКСТИ ПРОГРАМИ ОБЛІКУ МЕДИКАМЕНТІВ В АПТЕКАХ ТА КЕРУВАННЯ РОБОТИЗОВАНОЮ ШАФОЮ

Міністерство освіти і науки України
НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ
“ДНІПРОВСЬКА ПОЛІТЕХНІКА”

ПРОГРАМНЕ ЗАБЕЗПЕЧЕННЯ ОБЛІКУ МЕДИКАМЕНТІВ В АПТЕКАХ ТА КЕРУВАННЯ РОБОТИЗОВАНОЮ ШАФОЮ

Текст програми

804.02070743.22022-01 12 01

Листів 46

2022

АНОТАЦІЯ

Даний документ містить в собі програмний код обліку медикаментів в аптеках та керування роботизованою шафою в підсистемі аптеки комп’ютерної системи комплексу роботизованих аптек ТМ «Копійка».

Програма призначена для вконання наступних функцій:

– відображення таблиці всіх товарів;

– видалення обраного товару;

– редагування обраного товару;

– відправлення сигналів управління до контролера роботизованої шафи;

– редагування БД після видачі товару роботизованою шафою.

ЗМІСТ

1 PHARMACY

1.1 App.jsx

```
import React from 'react';
import { Provider } from 'react-redux';
import { PersistGate } from 'redux-persist/integration/react';
import AppNavigator from './navigation';
import { store, persistor } from './store/store';

function App() {
  return (
    <Provider store={store}>
      <PersistGate loading={null} persistor={persistor}>
        <AppNavigator />
      </PersistGate>
    </Provider>
  );
}

export default App;
```

1.2 Products/index.jsx

```
import React, { useEffect, useState } from 'react';
import { Typography, Button, Input, Popconfirm } from 'antd';
import PropTypes from 'prop-types';
import { createUseStyles } from 'react-jss';
import { connect } from 'react-redux';
import {
  addProduct,
  editProduct,
  deleteProduct,
  sortTable,
  getProducts,
} from '../../store/actions/data';
import { outputProduct } from '../../store/actions/controller';
import ProductsTable from './ProductsTable';
import BasketTable from './BasketTable';
import Drawer from '../../components/common/Drawer';
import ProductForm from './ProductForm';
import { searchProduct } from '../../services/utils';
import { products } from '../../store/selectors/data';
import { removeToken } from '../../store/actions/users';

const { Text } = Typography;

const Products = ({
  products,
  getProducts,
  addProduct,
  deleteProduct,
```

```
  removeToken,
  outputProduct,
  editProduct,
  sortTable,
  history,
}) => {
  const style = useStyles();
  const [addProductDrawerVisible, setAddProductDrawerVisible] = useState(false);
  const [basketDrawerVisible, setBasketDrawerVisible] = useState(false);
  const [
    isBasketDrawerButtonsDisabled,
    setIsBasketDrawerButtonsDisabled,
  ] = useState(false);
  const [initialValues, setInitialValues] = useState(null);
  const [basket, setBasket] = useState([]);
  const [searchText, setSearchText] = useState('');
  const [loadingStatus, setLoadingStatus] = useState(false);

  useEffect(() => {
    setLoadingStatus(true);
    getProducts().then(() => setLoadingStatus(false));
  }, []);

  const onSubmit = async data => {
    setLoadingStatus(true);
    if (initialValues) {
      await editProduct({ ...initialValues, ...data });
    } else {
      await addProduct(data);
    }
    setLoadingStatus(false);
    setInitialValues(null);
    setAddProductDrawerVisible(false);
  };

  const onDelete = async data => {
    setLoadingStatus(true);
    await deleteProduct(data);
    setLoadingStatus(false);
  };

  const onEdit = async ({ image, ...data }) => {
    setInitialValues(image ? { ...data, image: [image] } : { ...data });
    setAddProductDrawerVisible(true);
  };

  const onCancel = () => {
    setAddProductDrawerVisible(false);
    setInitialValues(null);
  };

  const onAddToBasket = data => {
    setBasket(prevBasket => {
```

```
    const clonedBasket = [...prevBasket];
    if (!clonedBasket.some(item => item._id === data._id)) {
      clonedBasket[clonedBasket.length] = { ...data, outputCount: 1 };
    }
    return clonedBasket;
  });
};

const onCancelBasket = () => {
  setBasketDrawerVisible(false);
  setBasket([]);
};

const onDeleteFromBasket = id => {
  setBasket(prevBasket => {
    const clonedBasket = [...prevBasket];
    const filteredClonedBasket = clonedBasket.filter(item => item._id !== id);
    if (filteredClonedBasket.length < 1) {
      setBasketDrawerVisible(false);
    }
    return filteredClonedBasket;
  });
};

const onChangeCountProductBasket = data => {
  setBasket(prevBasket => {
    const clonedBasket = [...prevBasket];
    return clonedBasket.map(item => {
      if (item._id === data.id) {
        if (data.outputCount) {
          return { ...item, outputCount: data.outputCount.toFixed() };
        }
      }
      return item;
    });
  });
};

const onOutput = () => {
  setIsBasketDrawerButtonsDisabled(true);
  outputProduct(basket).then(() => {
    setBasket([]);
    setBasketDrawerVisible(false);
    setIsBasketDrawerButtonsDisabled(false);
  });
};

const onExit = async () => {
  removeToken({ token: await localStorage.getItem('token') }).then(() => {
    localStorage.setItem('token', '');
    history.push('/sign-in');
  });
};
```

```
return products ? (
 <>
  <Drawer
   title={initialValues ? 'Редагувати товар' : 'Додати товар'}
   isVisible={addProductDrawerVisible}
   placement="left"
   onClose={onCancel}
   destroyOnClose
   footer={
    <div className={style.footer}>
     <Button form="product" type="primary" htmlType="submit">
      Прийняти
     </Button>
     <Button onClick={onCancel} style={{ marginLeft: 8 }}>
      Скасувати
     </Button>
    </div>
   }
  >
   <ProductForm
    onFinish={onSubmit}
    initialValues={initialValues}
    products={products}
   />
  </Drawer>

  <Drawer
   title="Кошик"
   isVisible={basketDrawerVisible}
   onClose={() => setBasketDrawerVisible(false)}
   destroyOnClose
   width={600}
   footer={
    <div className={style.footer}>
     <Button
      disabled={basket.length === 0 || isBasketDrawerButtonsDisabled}
      form="product"
      type="primary"
      onClick={onOutput}
     >
      Видати замовлення
     </Button>
     <Button
      disabled={isBasketDrawerButtonsDisabled}
      onClick={onCancelBasket}
      style={{ marginLeft: 8 }}
     >
      Скасувати замовлення
     </Button>
    </div>
   }
  >
```

```
    <BasketTable
     source={basket}
     isController
     onDelete={onDeleteFromBasket}
     onChangeCount={onChangeCountProductBasket}
    />
   </Drawer>

   <div className={style.container}>
    <div className={style.buttonContainer}>
     <div />
     <Popconfirm
      title="Ви впевнені, що хочете вийти з облікового запису?"
      placement="bottomLeft"
      onConfirm={() => onExit()}
      okText="Так"
      cancelText="Ні"
     >
      <Button type="secondary">Вийти з облікового запису</Button>
     </Popconfirm>
    </div>
    <div className={style.buttonContainer}>
     <Button
      type="primary"
      onClick={() => setAddProductDrawerVisible(true)}
     >
      Додати товар
     </Button>
     <Button
      disabled={basket.length === 0}
      onClick={() => setBasketDrawerVisible(true)}
     >
      Кошик
     </Button>
    </div>
    <div className={style.searchContainer}>
     <div className={style.searchTitle}>
      <Text>Пошук товару:</Text>
     </div>
     <Input onChange={e => setSearchText(e.target.value)} />
    </div>
    <ProductsTable
     source={searchProduct(products, searchText)}
     sortTable={sortTable}
     isController
     onEdit={onEdit}
     onDelete={onDelete}
     onAddToBasket={onAddToBasket}
     loadingStatus={loadingStatus}
    />
   </div>
  </>
 ) : (
```

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

const useStyles = createUseStyles({
  container: {
    padding: 50,
  },
  buttonContainer: {
    width: '100%',
    paddingBottom: 30,
    display: 'flex',
    justifyContent: 'space-between',
  },
  searchContainer: {
    display: 'flex',
    paddingBottom: 30,
    width: '50%',
  },
  searchTitle: {
    width: 130,
    display: 'flex',
    alignItems: 'center',
  },
  footer: {
    display: 'flex',
    justifyContent: 'space-between',
    padding: '10px 8px',
  },
});

const mapStateToProps = state => ({
  products: products(state),
});

const mapDispatchToProps = {
  getProducts,
  addProduct,
  deleteProduct,
  removeToken,
  outputProduct,
  editProduct,
  sortTable,
};

Products.propTypes = {
  products: PropTypes.arrayOf(PropTypes.shape({})).isRequired,
  getProducts: PropTypes.func.isRequired,
  addProduct: PropTypes.func.isRequired,
  deleteProduct: PropTypes.func.isRequired,
  removeToken: PropTypes.func.isRequired,
  outputProduct: PropTypes.func.isRequired,
  editProduct: PropTypes.func.isRequired,
```

```
  sortTable: PropTypes.func.isRequired,
  history: PropTypes.shape({}).isRequired,
};

export default connect(mapStateToProps, mapDispatchToProps)(Products);
```

1.3 Products/ProductForm

```
import React from 'react';
import { Input, Form, InputNumber, Button, Upload } from 'antd';
import PropTypes from 'prop-types';
import dayjs from 'dayjs';
import isSameOrAfter from 'dayjs/plugin/isSameOrAfter';
import DatePicker from '../../components/common/DatePicker';

dayjs.extend(isSameOrAfter);

const { form } = Form.useForm;

const ProductForm = ({ onFinish, initialValues, products }) => {
  const imageFormats = 'image/png, image/jpeg';

  const checkCell = value => {
    let errorCount = 0;
    products.forEach(item => {
      if (value === item.cellNumber && initialValues?._id !== item._id) {
        errorCount++;
      }
    });
    if (errorCount > 0) {
      return Promise.reject(new Error('Вказана комірка вже зайнята'));
    }
    if (value && (!Number.isInteger(Number(value)) || value > 100)) {
      return Promise.reject(new Error('Вказаної комірки не існує'));
    }
    return Promise.resolve();
  };

  const checkCode = value => {
    let errorCount = 0;
    products.forEach(item => {
      if (value === item.code && initialValues?._id !== item._id) {
        errorCount++;
      }
    });
    if (errorCount > 0) {
      return Promise.reject(new Error('Товар з таким кодом вже існує'));
    }
    if (value && !Number.isInteger(Number(value))) {
      return Promise.reject(new Error('Введіть коректний код товару'));
    }
    return Promise.resolve();
```

```
};

const checkExpirationDate = value => {
  if (value) {
    const dataArray = value.split('.');
    const convertedDate = `${dataArray[1]}.${dataArray[0]}.${dataArray[2]}`;
    if (dayjs().isSameOrAfter(convertedDate, 'day')) {
      return Promise.reject(new Error('Строк придатності товару закінчився'));
    }
  }
  return Promise.resolve();
};

const checkCount = value => {
  if (value && !Number.isInteger(value) && value < 1) {
    return Promise.reject(new Error('Введіть коректну кількість товарів'));
  }
  return Promise.resolve();
};

const checkSize = value => {
  if (value && value[0]?.size / 1024 / 1024 > 10) {
    return Promise.reject(
      new Error('Зображення не повинно бути більше 10 МБ'),
    );
  }
  return Promise.resolve();
};

return (
  <Form
    form={form}
    layout="vertical"
    onFinish={onFinish}
    name="product"
    hideRequiredMark
    initialValues={initialValues}
  >
    <Form.Item
      name="name"
      label="Найменування товару"
      rules={[
        { required: true, message: 'Найменування повинно бути заповненим' },
      ]}
    >
      <Input placeholder="Найменування товару" />
    </Form.Item>

    <Form.Item
      name="code"
      label="Код товару"
      rules={[
        {
```

```
    required: true,
    message: 'Код товару повиннен бути заповнений',
   },
   {
    validator: (_, value) => checkCode(value),
   },
  ]}
>
  <InputNumber
   type="number"
   placeholder="Код товару"
   min={0}
   style={{ width: '100%' }}
  />
</Form.Item>

<Form.Item name="category" label="Категорія товару">
  <Input placeholder="Категорія товару" />
</Form.Item>

<Form.Item name="description" label="Рекомендації по використанню">
  <Input placeholder="Рекомендації по використанню" />
</Form.Item>

<Form.Item
  name="image"
  label="Зображення товару"
  valuePropName="fileList"
  getValueFromEvent={({ fileList }) => fileList}
  rules={[
   {
    validator: (_, value) => checkSize(value),
   },
  ]}
>
  <Upload accept={imageFormats} beforeUpload={() => false} maxCount={1}>
   <Button>
    {initialValues?.image?.length > 0
     ? 'Редагувати зображення товару'
     : 'Додати зображення товару'}
   </Button>
  </Upload>
</Form.Item>

<Form.Item
  name="price"
  label="Ціна за одиницю товару"
  rules={[
   {
    required: true,
    message: 'Ціна за одиницю товару повинна бути заповнена',
   },
   { pattern: /^\d+(?:\.\d{1,2})?$/, message: 'Введіть коректну ціну' },
```

```
  ]}
>
  <InputNumber
    type="number"
    placeholder="Ціна за одиницю товару"
    min={0}
    step="0.01"
    style={{ width: '100%' }}
  />
</Form.Item>

<Form.Item
  name="expirationDate"
  label="Строк придатності"
  rules={[
    {
      required: true,
      message: 'Строк придатності повинен бути заповнений',
    },
    {
      validator: (_, value) => checkExpirationDate(value),
    },
  ]}
>
  <DatePicker />
</Form.Item>

<Form.Item
  name="cellNumber"
  label="Номер комірки"
  rules={[
    { required: true, message: 'Номер комірки повинен бути заповнений' },
    {
      validator: (_, value) => checkCell(value),
    },
  ]}
>
  <InputNumber
    type="number"
    placeholder="Номер комірки"
    min={0}
    style={{ width: '100%' }}
  />
</Form.Item>

<Form.Item
  name="count"
  label="Кількість одиниць товару"
  rules={[
    {
      required: true,
      message: 'Кількість одиниць товару повинна бути заповнена',
    },
```

```
        {
          validator: (_, value) => checkCount(value),
        },
       ]}
     >
      <InputNumber
       type="number"
       placeholder="Номер комірки"
       min={0}
       style={{ width: '100%' }}
      />
     </Form.Item>
    </Form>
  );
};

ProductForm.propTypes = {
  products: PropTypes.arrayOf(PropTypes.shape({})).isRequired,
  onFinish: PropTypes.func.isRequired,
  initialValues: PropTypes.shape({}),
};

ProductForm.defaultProps = {
  initialValues: null,
};

export default ProductForm;
```

1.4 Products/ProductsTable/index.jsx

```
import React, { useCallback, useEffect } from 'react';
import { Table } from 'antd';
import PropTypes from 'prop-types';

import useTableController from '../../../hooks/useTableController';
import { getBaseColumns, getControlColumns } from './columns';

const ProductsTable = ({
  onSelect,
  selectedProducts,
  source,
  isController,
  onEdit,
  onAddToBasket,
  onDelete,
  sortTable,
  loadingStatus,
}) => {
  const getColumns = useCallback(
    () =>
      isController
        ? [
```

```
        ...getBaseColumns(),
        ...getControlColumns({
          onEdit,
          onDelete,
          onAddToBasket,
        }),
      ]
    : [...getBaseColumns()],
  [isController],
);

const {
  sortDirection,
  sortField,
  setSortDirection,
  setSortField,
} = useTableController({
  dataSource: source,
  getColumns,
});

useEffect(() => {
  sortTable({ sortDirection, sortField });
}, [sortDirection, sortField]);

const rowSelection = {
  onChange: selectedRowKeys => {
    onSelect(selectedRowKeys);
  },
  selectedRowKeys: selectedProducts || [],
};

const onChange = (pagination, filters, sorter) => {
  const { field, order } = sorter;
  setSortDirection(order);
  setSortField(field);
};

return (
  <Table
    columns={getColumns()}
    dataSource={source}
    scroll={{ x: true }}
    rowKey={({ _id }) => _id}
    onChange={onChange}
    loading={loadingStatus}
    rowSelection={onSelect ? { ...rowSelection } : null}
    expandable={{
      expandedRowRender: record => (
        <p style={{ margin: 0 }}>{record.description}</p>
      ),
      rowExpandable: record => !!record.description,
    }}
```

```
    />
  );
};

ProductsTable.propTypes = {
  source: PropTypes.arrayOf(PropTypes.shape({})).isRequired,
  onSelect: PropTypes.func,
  selectedProducts: PropTypes.arrayOf(PropTypes.string),
  isController: PropTypes.bool,
  onEdit: PropTypes.func,
  onAddToBasket: PropTypes.func,
  onDelete: PropTypes.func,
  sortTable: PropTypes.func,
  loadingStatus: PropTypes.bool,
};

ProductsTable.defaultProps = {
  onSelect: null,
  selectedProducts: null,
  isController: false,
  onEdit: () => {},
  onDelete: () => {},
  sortTable: () => {},
  onAddToBasket: () => {},
  loadingStatus: false,
};

export default ProductsTable;
```

1.5 Products/ProductsTable/column.js

```
import React from 'react';
import { Popconfirm, Typography } from 'antd';
import {
  EditOutlined,
  DeleteOutlined,
  ShoppingCartOutlined,
} from '@ant-design/icons';
import dayjs from 'dayjs';
import ImageMaster from '../../../components/common/ImageMaster';

const { Text } = Typography;

const checkExpirationDate = value => {
  if (value) {
    const dataArray = value.split('.');
    const convertedDate = `${dataArray[1]}.${dataArray[0]}.${dataArray[2]}`;
    return dayjs().isSameOrAfter(convertedDate, 'day');
  }
};

export const getBaseColumns = () => {
```

```
return [
  {
    title: 'Зображення товару',
    dataIndex: 'image',
    key: 'image',
    render: image => (
      <div style={{ width: '300px' }}>
        {image && <ImageMaster source={image} height={150} />}
      </div>
    ),
  },
  {
    title: 'Найменування товару',
    dataIndex: 'name',
    key: 'name',
    sorter: true,
  },
  {
    title: 'Код товару',
    dataIndex: 'code',
    key: 'code',
    sorter: true,
  },
  {
    title: 'Категорія товару',
    dataIndex: 'category',
    key: 'category',
    sorter: true,
  },
  {
    title: 'Ціна за одиницю товару',
    dataIndex: 'price',
    key: 'price',
    sorter: true,
    render: price => price.toFixed(2),
  },
  {
    title: 'Строк придатності',
    dataIndex: 'expirationDate',
    key: 'expirationDate',
    sorter: true,
    render: expirationDate => (
      <Text style={checkExpirationDate(expirationDate) && { color: 'red' }}>
        {expirationDate}
      </Text>
    ),
  },
  {
    title: 'Номер комірки',
    dataIndex: 'cellNumber',
    key: 'cellNumber',
    sorter: true,
  },
```

```
    {
      title: 'Кількість одниць товару',
      dataIndex: 'count',
      key: 'count',
      sorter: true,
    },
  ];
};

export const getControlColumns = ({ onEdit, onDelete, onAddToBasket }) => [
  {
    title: '',
    dataIndex: 'edit',
    key: 'edit',
    fixed: 'right',
    render: (edit, product) => (
      <div
        onClick={() => {
          onEdit(product);
        }}
      >
        <EditOutlined style={{ cursor: 'pointer', fontSize: 20 }} />
      </div>
    ),
  },
  {
    title: '',
    dataIndex: 'delete',
    key: 'delete',
    fixed: 'right',
    render: (del, product) => (
      <Popconfirm
        title="Ви впевнені, що хочете видалити цей товар?"
        onConfirm={() => onDelete(product._id)}
        okText="Так"
        cancelText="Ні"
      >
        <div>
          <DeleteOutlined style={{ cursor: 'pointer', fontSize: 20 }} />
        </div>
      </Popconfirm>
    ),
  },
  {
    title: '',
    dataIndex: 'output',
    key: 'output',
    fixed: 'right',
    render: (edit, product) =>
      product.count > 0 &&
      (checkExpirationDate(product.expirationDate) ? (
        <Popconfirm
          title="Товар просрочено. Ви впевнені що хоете видати його?"
```

```
        onConfirm={() => onAddToBasket(product)}
        okText="Так"
        cancelText="Ні"
       >
        <div>
         <ShoppingCartOutlined style={{ cursor: 'pointer', fontSize: 20 }} />
        </div>
       </Popconfirm>
      ) : (
       <div
        onClick={() => {
         onAddToBasket(product);
        }}
       >
        <ShoppingCartOutlined style={{ cursor: 'pointer', fontSize: 20 }} />
       </div>
      )),
  },
];
```

1.6 Products/BasketTable/index.jsx

```
import React, { forwardRef, useCallback } from 'react';
import { Table, Typography } from 'antd';
import PropTypes from 'prop-types';

import { getBaseColumns, getControlColumns } from './columns';

const { Text } = Typography;

const BasketTable = ({ source, isController, onDelete, onChangeCount }) => {
  const getColumns = useCallback(
    () =>
      isController
        ? [...getBaseColumns(onChangeCount), ...getControlColumns({ onDelete })]
        : [...getBaseColumns()],
    [isController],
  );

  return (
    <Table
      columns={getColumns()}
      dataSource={source}
      pagination={false}
      scroll={{ x: true, y: 290 }}
      rowKey={({ _id }) => _id}
      summary={() => (
        <Table.Summary fixed>
          <Table.Summary.Row>
            <Table.Summary.Cell index={0}>
              <Text style={{ fontWeight: '500' }}>Сумарна вартість</Text>
            </Table.Summary.Cell>
```

```
          <Table.Summary.Cell index={1} colSpan={3}>
            <Text style={{ fontWeight: '500' }}>
              {source
                .reduce((previousValue, currentValue) => {
                  return (
                    previousValue +
                    currentValue.price * currentValue.outputCount
                  );
                }, 0)
                .toFixed(2)}
            </Text>
          </Table.Summary.Cell>
        </Table.Summary.Row>
      </Table.Summary>
    )}
  />
 );
};

BasketTable.propTypes = {
 source: PropTypes.arrayOf(PropTypes.shape({})).isRequired,
 isController: PropTypes.bool,
 onDelete: PropTypes.func,
 onChangeCount: PropTypes.func,
};

BasketTable.defaultProps = {
 isController: false,
 onDelete: () => {},
 onChangeCount: () => {},
};

export default BasketTable;
```

1.7 Products/BasketTable/column.js

```
import React from 'react';
import { InputNumber, Popconfirm } from 'antd';
import { DeleteOutlined } from '@ant-design/icons';

export const getBaseColumns = onChangeCount => {
 return [
  {
   title: 'Найменування товару',
   dataIndex: 'name',
   key: 'name',
  },
  {
   title: 'Ціна за одиницю товару',
   dataIndex: 'price',
   key: 'price',
   render: price => price.toFixed(2),
```

```
    },
    {
      title: 'Кількість одниць товару',
      dataIndex: 'outputCount',
      key: 'outputCount',
      render: (outputCount, product) => (
        <>
          <InputNumber
            value={outputCount}
            min={1}
            max={product.count}
            onChange={outputCount => {
              onChangeCount({ outputCount, id: product._id });
            }}
          />
        </>
      ),
    },
  ];
};

export const getControlColumns = ({ onDelete }) => [
  {
    title: '',
    dataIndex: 'delete',
    key: 'delete',
    fixed: 'right',
    render: (del, product) => (
      <Popconfirm
        title="Ви впевнені, що хочете видалити цей товар?"
        onConfirm={() => onDelete(product._id)}
        okText="Так"
        cancelText="Ні"
      >
        <div>
          <DeleteOutlined style={{ cursor: 'pointer', fontSize: 20 }} />
        </div>
      </Popconfirm>
    ),
  },
];
```

1.8 SignIn/index.jsx

```
import React from 'react';
import { message, Form, Input, Button } from 'antd';
import { createUseStyles } from 'react-jss';
import { connect } from 'react-redux';
import PropTypes from 'prop-types';

import { createUser, signIn } from '../../store/actions/users';
```

```
const SignIn = ({ signIn, createUser, history }) => {
 const style = useStyles();

 const [form] = Form.useForm();

 const onSignIn = async (login, password) => signIn({ login, password });

 const onSignInSuccess = ({ data }) => {
  message.success('Вхід виконано успішно');
  localStorage.setItem('token', data.token);
  history.push('/products');
 };

 const onSignInFailed = () =>
  message.error('Вхід не виконано. Перевірте логін та пароль!');

 const onFinish = data => {
  const { login, password } = data;
  onSignIn(login, password)
   .then(onSignInSuccess)
   .catch(onSignInFailed);
 };

 const onCreateUser = () => {
  const { login, password } = form.getFieldValue();
  createUser({ login, password }).then(() =>
   message.success('Користувача створено успішно'),
  );
 };

 const layout = {
  labelCol: { span: 6 },
  wrapperCol: { span: 14 },
 };

 return (
  <div className={style.container}>
   <Form
    {...layout}
    name="signIn"
    initialValues={{ remember: true }}
    onFinish={onFinish}
    form={form}
   >
    <Form.Item
     label="ЛОГІН"
     name="login"
     rules={[{ required: true, message: 'Будь-ласка введіть логін!' }]}
    >
     <Input />
    </Form.Item>

    <Form.Item
```

```
        label="ПАРОЛЬ"
        name="password"
        rules={[{ required: true, message: 'Будь-ласка введіть пароль!' }]}
       >
        <Input.Password />
       </Form.Item>

       <Button type="primary" htmlType="submit">
        УВІЙТИ
       </Button>
       <Button type="secondary" onClick={onCreateUser}>
        СТВОРИТИ КОРИСТУВАЧА
       </Button>
     </Form>
   </div>
  );
};

const useStyles = createUseStyles({
  container: {
    flex: 1,
    minHeight: '100vh',
    display: 'flex',
    flexDirection: 'column',
    justifyContent: 'center',
    '& .ant-form': {
      minHeight: '100vh',
      display: 'flex',
      flexDirection: 'column',
      justifyContent: 'center',
      width: '75%',
      margin: '0 auto',
    },
    '& .ant-row .ant-form-item-label label': {
      fontWeight: '400',
      fontSize: 16,
      lineHeight: '16px',
    },
    '& .ant-btn': {
      height: 36,
      width: 250,
      margin: '10px auto',
    },
  },
});

SignIn.propTypes = {
  signIn: PropTypes.func.isRequired,
  createUser: PropTypes.func.isRequired,
  history: PropTypes.shape({
    push: PropTypes.func.isRequired,
  }).isRequired,
};
```

```
export default connect(null, { signIn, createUser })(SignIn);
```

1.9 hoc/withNotVerifiedProtection.js

```
import React, { useEffect, useState } from 'react';
import { connect } from 'react-redux';
import { Redirect } from 'react-router';
import { checkAuth } from '../store/actions/users';

const withNotVerifiedProtection = Component => {
  const Wrapped = ({ checkAuth, ...props }) => {
    const [isChecked, setCheckStatus] = useState(false);

    useEffect(() => {
      checkAuth(localStorage.getItem('token'))
        .then(() => props.history.push('products'))
        .catch(() => setCheckStatus(true));
    }, []);

    return isChecked ? <Component {...props} /> : null;
  };

  return connect(null, { checkAuth })(Wrapped);
};

export default withNotVerifiedProtection;
```

1.10 hoc/withVerifiedProtection.js

```
import React, { useEffect, useState } from 'react';
import { connect } from 'react-redux';
import { Redirect } from 'react-router';
import { checkAuth } from '../store/actions/users';

const withVerifiedProtection = Component => {
  const Wrapped = ({ checkAuth, ...props }) => {
    const [isChecked, setCheckStatus] = useState(false);

    useEffect(() => {
      checkAuth(localStorage.getItem('token'))
        .then(() => setCheckStatus(true))
        .catch(() => props.history.push('403'));
    }, []);

    return isChecked ? <Component {...props} /> : null;
  };

  return connect(null, { checkAuth })(Wrapped);
};
```

```
export default withVerifiedProtection;
```

1.11 hooks/useTableController.js

```
import { useState } from 'react';

const mapOrders = {
  ascend: 'asc',
  descend: 'desc',
};

const useTableController = () => {
  const [sortDirection, setSortDirection] = useState(null);
  const [sortField, setSortField] = useState(null);

  return {
    sortDirection,
    sortField,
    setSortDirection: order => {
      setSortDirection(mapOrders[order]);
    },
    setSortField,
  };
};

export default useTableController;
```

1.12 navigation/index.jsx

```
import React from 'react';
import { BrowserRouter, Route, Switch, Redirect } from 'react-router-dom';
import Products from '../containers/Products';
import NotFoundPage from '../components/NotFoundPage';
import ForbiddenPage from '../components/ForbiddenPage';
import SignIn from '../containers/SignIn';
import withVerifiedProtection from '../hoc/withVerifiedProtection';
import withNotVerifiedProtection from '../hoc/withNotVerifiedProtection';

const AppNavigator = () => {
  return (
    <BrowserRouter>
      <Switch>
        <Route
          path="/sign-in"
          component={withNotVerifiedProtection(SignIn)}
          exact
        />
        <Route path="/" render={() => <Redirect to="/sign-in" />} exact />
        <Route
          path="/products"
          component={withVerifiedProtection(Products)}
```

```
        exact
       />
       <Route path="/403" component={ForbiddenPage} />
       <Route path="*" component={NotFoundPage} />
     </Switch>
   </BrowserRouter>
  );
};
export default AppNavigator;
```

1.13 services/utils.js

```
import * as dayjs from 'dayjs';

export const sortArray = (array, { sortDirection, sortField }) => {
  if (sortDirection === 'asc') {
    if (sortField === 'createAt') {
      return sortByDate(array, 'up');
    }
    return array.sort((a, b) => (a[sortField] > b[sortField] ? 1 : -1));
  }
  if (sortDirection === 'desc') {
    if (sortField === 'createAt') {
      return sortByDate(array, 'down');
    }
    return array.sort((a, b) => (a[sortField] < b[sortField] ? 1 : -1));
  }
  return sortByDate(array, 'up');
};

export const sortByDate = (array, sortType) => {
  if (sortType === 'up') {
    return array.sort((a, b) => {
      if (dayjs(a.createAt).isAfter(dayjs(b.createAt))) {
        return 1;
      }
      if (dayjs(a.createAt).isBefore(dayjs(b.createAt))) {
        return -1;
      }
      return 0;
    });
  }
  return array.sort((a, b) => {
    if (dayjs(a.date).isBefore(dayjs(b.date))) {
      return 1;
    }
    if (dayjs(a.date).isAfter(dayjs(b.date))) {
      return -1;
    }
    return 0;
  });
};
```

```
export const searchProduct = (array, searchText) => {
  return array.filter(item => {
    let sum = 0;
    Object.entries(item).forEach(objectItem => {
      if (
        !['__v', '_id'].includes(objectItem[0]) &&
        String(objectItem[1])
          .toUpperCase()
          .includes(searchText.toUpperCase())
      ) {
        sum++;
      }
    });
    if (sum > 0) {
      return item;
    }
  });
};
```

1.14 network/controller/index.js

```
import axios from 'axios';

const instance = axios.create({
  baseURL: 'http://localhost:8888/api',
});

const outputProduct = data => {
  return instance.post(`/output-product`, { data });
};

export default { outputProduct };
```

1.15 network/server/index.js

```
import axios from 'axios';
import CryptoJS from 'crypto-js';

const instance = axios.create({
  baseURL: 'http://localhost:9999/api',
});

const encodeImageToBase64 = async ({ originFileObj, ...image }) => {
  return typeof originFileObj === 'string'
    ? { originFileObj, ...image }
    : { ...image, originFileObj: await toBase64(originFileObj) };
};

const toBase64 = file =>
  new Promise((resolve, reject) => {
```

```
    const reader = new FileReader();
    reader.readAsDataURL(file);
    reader.onload = () => resolve(reader.result);
    reader.onerror = error => reject(error);
  });

const addProduct = async ({ image, ...data }) => {
  return instance.post(`/product`, {
    ...data,
    image: image && image.length > 0 && (await encodeImageToBase64(image[0])),
  });
};

const editProduct = async ({ image, ...data }) => {
  return instance.patch(`/product`, {
    ...data,
    image: image.length > 0 && (await encodeImageToBase64(image[0])),
  });
};

const editProducts = async data => {
  return instance.patch(
    `/products`,
    await Promise.all(
      data.map(async ({ image, count, outputCount, ...item }) => ({
        ...item,
        count: count - outputCount,
        image: image.length > 0 ? await encodeImageToBase64(image[0]) : image,
      })),
    ),
  );
};

const deleteProduct = id => {
  return instance.delete(`/product/${id}`);
};

const getProducts = () => {
  return instance.get('/products');
};

const signIn = data => {
  return instance.post('/auth', {
    userData: CryptoJS.AES.encrypt(
      JSON.stringify(data),
      process.env.REACT_APP_SECRET_KEY,
    ).toString(),
  });
};

const createUser = data => {
  return instance.post('/create-user', {
    userData: CryptoJS.AES.encrypt(
```

```
    JSON.stringify(data),
    process.env.REACT_APP_SECRET_KEY,
   ).toString(),
  });
};

const checkAuth = token => {
  return instance.post('/check-auth', {
    token,
  });
};

const removeToken = token => {
  return instance.post('/remove-token', token);
};

export default {
  addProduct,
  getProducts,
  editProduct,
  deleteProduct,
  editProducts,
  signIn,
  createUser,
  checkAuth,
  removeToken,
};
```

1.16 common/DatePicker.jsx

```
import React, { useMemo } from 'react';
import { DatePicker as AntdDatePicker } from 'antd';
import * as moment from 'moment';
import PropTypes from 'prop-types';
import dayjs from 'dayjs';
import customParseFormat from 'dayjs/plugin/customParseFormat';

dayjs.extend(customParseFormat);

const DatePicker = ({ value, onChange, ...props }) => {
  const computedValue = useMemo(() => {
    return value ? moment(value, 'DD.MM.YYYY') : null;
  }, [value]);

  const onChangeDate = value => {
    onChange(value ? dayjs(value).format('DD.MM.YYYY') : null);
  };

  return (
    <AntdDatePicker
      {...props}
      value={computedValue}
```

```
      onChange={onChangeDate}
      format="DD.MM.YYYY"
    />
  );
};

DatePicker.propTypes = {
  value: PropTypes.string,
  onChange: PropTypes.func,
};

DatePicker.defaultProps = {
  value: null,
  onChange: () => {},
};

export default DatePicker;
```

1.17 common/Drawer.jsx

```
import React from 'react';
import { Drawer as BasicDrawer } from 'antd';
import PropTypes from 'prop-types';

const Drawer = ({
  title,
  placement,
  isVisible,
  onClose,
  children,
  width,
  ...props
}) => {
  return (
    <BasicDrawer
      title={title}
      placement={placement}
      closable
      onClose={onClose}
      visible={isVisible}
      width={width}
      {...props}
    >
      {children}
    </BasicDrawer>
  );
};

Drawer.propTypes = {
  title: PropTypes.string.isRequired,
  placement: PropTypes.string,
  isVisible: PropTypes.bool.isRequired,
```

```
  onClose: PropTypes.func.isRequired,
  children: PropTypes.node.isRequired,
  width: PropTypes.PropTypes.oneOfType([PropTypes.string, PropTypes.number]),
};

Drawer.defaultProps = {
  placement: 'right',
  width: 350,
};

export default Drawer;
```

1.18 common/ImageMaster.jsx

```
import React, { useState } from 'react';
import { Modal } from 'antd';
import PropTypes from 'prop-types';

const ImageMaster = ({ source, height }) => {
  const [isFullScreen, setFullScreenStatus] = useState(false);
  const onFullScreenRequest = () => {
    if (source) {
      setFullScreenStatus(true);
    }
  };

  return (
    <>
      <div
        style={{
          width: '100%',
          height: '100%',
          display: 'flex',
        }}
        onClick={onFullScreenRequest}
      >
        <img
          src={source.originFileObj}
          alt="video"
          style={{
            height,
            maxWidth: '100%',
            margin: 'auto',
          }}
        />
      </div>
      <Modal
        visible={isFullScreen}
        destroyOnClose
        width="60%"
        onCancel={() => setFullScreenStatus(false)}
        footer={null}
```

```
    >
     <div style={{ paddingTop: 20 }}>
      <img
       src={source.originFileObj}
       alt="img"
       style={{
        width: '100%',
        height: '100%',
        margin: 'auto',
       }}
      />
     </div>
    </Modal>
   </>
  );
};

ImageMaster.propTypes = {
  source: PropTypes.shape({}),
  height: PropTypes.number,
};

ImageMaster.defaultProps = {
  source: null,
  height: 150,
};

export default ImageMaster;
```

1.19 store/store.js

```
import { createStore, applyMiddleware } from 'redux';
import storage from 'redux-persist/lib/storage';
import thunk from 'redux-thunk';
import createSaga from 'redux-saga';
import { persistStore, persistReducer } from 'redux-persist';
import reducers from './reducers';
import sagas from './sagas';

const createStoreFn = (initialData = {}) => {
  const saga = createSaga();
  const middlewares = [thunk, saga];
  const persistConfig = {
    key: 'root',
    storage,
  };
  const persistedReducer = persistReducer(persistConfig, reducers);
  const store = createStore(
    persistedReducer,
    initialData,
    applyMiddleware(...middlewares),
  );
```

```
  saga.run(sagas);

  return store;
};

export const store = createStoreFn();

export const persistor = persistStore(store);

export { createStoreFn as createStore };
```

1.20 store/types.js

```
export const successAction = action => `${action}_SUCCESS`;
export const failureAction = action => `${action}_FAILURE`;

export const GET_PRODUCTS = 'GET_PRODUCTS';
export const ADD_PRODUCT = 'ADD_PRODUCT';
export const DELETE_PRODUCT = 'DELETE_PRODUCT';
export const EDIT_PRODUCT = 'EDIT_PRODUCT';
export const EDIT_PRODUCTS = 'EDIT_PRODUCTS';
export const SORT_TABLE = 'SORT_TABLE';

export const ADD_INFO = 'ADD_INFO';
export const OUTPUT_PRODUCT = 'OUTPUT_PRODUCT';

export const SIGN_IN = 'SIGN_IN';
export const CREATE_USER = 'CREATE_USER';
export const CHECK_AUTH = 'CHECK_AUTH';
export const REMOVE_TOKEN = 'REMOVE_TOKEN';
```

1.21 store/actions/controller.js

```
import { OUTPUT_PRODUCT } from '../types';
import { callApi } from './utils';
import controllerNetwork from '../../services/network/controller';

export const outputProduct = data => {
  const action = () => controllerNetwork.outputProduct(data);
  return callApi(action, OUTPUT_PRODUCT);
};
```

1.22 store/actions/data.js

```
import {
  ADD_PRODUCT,
  DELETE_PRODUCT,
  EDIT_PRODUCT,
  SORT_TABLE,
  ADD_INFO,
  GET_PRODUCTS,
```

```
  OUTPUT_PRODUCT,
  EDIT_PRODUCTS,
} from '../types';
import { callApi } from './utils';
import serverNetwork from '../../services/network/server';
import controllerNetwork from '../../services/network/controller';

export const addProduct = data => {
  const action = () => serverNetwork.addProduct(data);
  return callApi(action, ADD_PRODUCT);
};

export const editProduct = data => {
  const action = () => serverNetwork.editProduct(data);
  return callApi(action, EDIT_PRODUCT);
};

export const editProducts = data => {
  const action = () => serverNetwork.editProducts(data);
  return callApi(action, EDIT_PRODUCTS);
};

export const deleteProduct = id => {
  const action = () => serverNetwork.deleteProduct(id);
  return callApi(action, DELETE_PRODUCT);
};

export const getProducts = () => {
  const action = () => serverNetwork.getProducts();
  return callApi(action, GET_PRODUCTS);
};

export const sortTable = payload => {
  return {
    type: SORT_TABLE,
    payload,
  };
};
```

1.23 store/actions/users.js

```
import { CHECK_AUTH, CREATE_USER, REMOVE_TOKEN, SIGN_IN } from '../types';
import { callApi } from './utils';
import serverNetwork from '../../services/network/server';

export const signIn = data => {
  const action = () => serverNetwork.signIn(data);
  return callApi(action, SIGN_IN);
};

export const createUser = data => {
  const action = () => serverNetwork.createUser(data);
```

```
  return callApi(action, CREATE_USER);
};

export const checkAuth = token => {
  const action = () => serverNetwork.checkAuth(token);
  return callApi(action, CHECK_AUTH);
};

export const removeToken = token => {
  const action = () => serverNetwork.removeToken(token);
  return callApi(action, REMOVE_TOKEN);
};
```

1.24 store/actions/utils.js

```
import { successAction, failureAction } from '../types';

export const callApi = (action, actionType, options = {}) => dispatch => {
  dispatch({ type: actionType, ...options });

  return new Promise((resolve, reject) => {
    const actionPromise = action();
    actionPromise.then(payload => {
      dispatch({
        type: successAction(actionType),
        payload,
        ...options,
      });
      resolve(payload);
    });
    actionPromise.catch(error => {
      dispatch({
        type: failureAction(actionType),
        error,
        ...options,
      });
      reject(error);
    });
  });
};
```

1.25 store/reducers/data.js

```
import { sortArray } from '../../services/utils';
import { GET_PRODUCTS, SORT_TABLE, successAction } from '../types';

const initialState = {
  products: [],
};

export default (state = initialState, action) => {
  switch (action.type) {
```

```
    case successAction(GET_PRODUCTS): {
      console.log(action.payload.data.products);
      return {
        ...state,
        products: action.payload.data || [],
      };
    }

    case SORT_TABLE: {
      const newProducts = Array.from(state.products);
      return {
        ...state,
        products: sortArray(newProducts, action.payload),
      };
    }
    default:
      return state;
  }
};
```

1.26 store/reducers/index.js

```
import { combineReducers } from "redux";

import data from "./data";

const reducers = combineReducers({
  data,
});

export default reducers;
```

1.27 store/sagas/data.js

```
import { takeLatest, call, put } from 'redux-saga/effects';
import { editProducts, getProducts } from '../actions/data';
import {
  ADD_PRODUCT,
  DELETE_PRODUCT,
  EDIT_PRODUCT,
  EDIT_PRODUCTS,
  OUTPUT_PRODUCT,
  successAction,
} from '../types';

function* handleGetData() {
  yield call(console.log, ADD_PRODUCT);
}

function* handleChangeData() {
  yield put(getProducts());
}
```

```
function* handleOutputProduct({ payload }) {
  console.log('handleOutputProduct');
  console.log(payload.data);
  yield put(editProducts(payload.data));
}

function* watchGetData() {
  yield takeLatest(ADD_PRODUCT, handleGetData);
}

function* watchChangeData() {
  yield takeLatest(
    [
      successAction(EDIT_PRODUCT),
      successAction(EDIT_PRODUCTS),
      successAction(DELETE_PRODUCT),
      successAction(ADD_PRODUCT),
    ],
    handleChangeData,
  );
}

function* watchOutputProduct() {
  yield takeLatest([successAction(OUTPUT_PRODUCT)], handleOutputProduct);
}

export default [watchGetData, watchChangeData, watchOutputProduct];
```

1.28 store/sagas/index.js

```
import { fork, all } from 'redux-saga/effects';
import data from './data';

export default function* root() {
  const watchers = [...data];
  yield all(watchers.map(fork));
}
```

1.29 store/selectors/data.js

```
export const data = state => state.data;
export const products = state => state.data.products;
```

2 SERVER

2.1 index.js

```
const express = require("express");
const cors = require("cors");
const { v4: uuidv4 } = require("uuid");
const bodyParser = require("body-parser");
const mongoose = require("./mongoose");
const { decoder, encoder } = require("./utils");
const Product = require("./schemaProduct");
const User = require("./schemaUser");

const app = express();

app.use(cors());
app.use(
  bodyParser.urlencoded({
    extended: true,
  })
);
app.use(bodyParser.json({ limit: "50mb" }));

app.listen(9999, () => {
  console.log("Visit http://localhost:9999");
});

app.post("/api/product", async (req, res) => {
  const data = req.body;
  console.log("data", data);
  const newProduct = new Product(data);
  await newProduct.save((err, result) => {
    console.log(err, result);
  });
  res.json({ message: "success!" });
});

app.patch("/api/product", async (req, res) => {
  const data = req.body;
  console.log("data", data);

  await Product.replaceOne({ _id: data._id }, { ...data });
  res.json({ message: "success!" });
});

app.patch("/api/products", async (req, res) => {
  const data = req.body;
  console.log("productsData", data);

  await data.forEach(
    async (item) => await Product.replaceOne({ _id: item._id }, { ...item })
  );
  res.json({ message: "success!" });
```

```
});

app.delete("/api/product/:id", async (req, res) => {
  const { id } = req.params;

  console.log("delete product", id);
  await Product.deleteOne({ _id: id });
  console.log("product was deleted");
  res.send();
});

app.get("/api/products", async (req, res) => {
  const products = await Product.find();
  res.type("application/json").send(JSON.stringify(products));
});

app.post("/api/auth", async (req, res) => {
  const data = req.body;
  console.log("data", data);

  const users = await User.find();
  const decodedData = decoder(data.userData);
  const authUser = users.find((user) => {
    const decodedUser = decoder(user.userData);
    return (
      decodedUser.login === decodedData.login &&
      decodedUser.password === decodedData.password
    );
  });
  if (!!authUser) {
    let token = "";
    do {
      token = uuidv4();
    } while (users.some((user) => user.token === token));
    await User.updateOne({ _id: authUser._id }, { token: encoder(token) });
    res.json({ token: encoder(token) });
  } else {
    res.status(403).send({ error: "Wrong login or password!" });
  }
});

app.post("/api/create-user", async (req, res) => {
  const data = req.body;
  console.log("data", data.userData);
  const newUser = new User(data);
  await newUser.save((err, result) => {
    console.log(err, result);
  });

  res.json({ message: "success!" });
});

app.post("/api/check-auth", async (req, res) => {
```

```
 const { token } = req.body;
 const users = await User.find();
 console.log("token", token);
 const isAuth = token
  ? users.some(
     (user) => user?.token && decoder(user.token) === decoder(token)
   )
  : false;
 if (isAuth) {
  res.json({ message: "success!" });
 } else {
  res.status(403).send({ error: "Wrong token!" });
 }
});

app.post("/api/remove-token", async (req, res) => {
 const { token } = req.body;
 const users = await User.find();
 const removeTokenUser = users.find(
  (user) => user?.token && decoder(user.token) === decoder(token)
 );
 await User.updateOne({ _id: removeTokenUser._id }, { $unset: { token: "" } });
 res.json({ message: "success!" });
});
```

2.2 key.js

```
const key = "KIIT12320m1";

module.exports = key;
```

2.3 mongoose.js

```
const mongoose = require("mongoose");

mongoose.connect("mongodb://localhost:27017/baseR", {
 useNewUrlParser: true,
});

module.exports = mongoose;
```

2.4 schemaProduct.js

```
const mongoose = require("mongoose");

const productSchema = new mongoose.Schema({
 name: String,
 code: Number,
 category: String,
 description: String,
 price: Number,
```

```
  expirationDate: String,
  cellNumber: String,
  count: Number,
  image: {},
});

module.exports = mongoose.model("Product", productSchema);
```

2.5 schemaUser.js

```
const mongoose = require("mongoose");

const userSchema = new mongoose.Schema({
  userData: String,
  token: String,
});

module.exports = mongoose.model("User", userSchema);
```

2.6 server.bat

```
node index.js
```

2.7 utils.js

```
const CryptoJS = require("crypto-js");
const key = require("./key");

const decoder = (data) => {
  console.log(data);
  const bytes = CryptoJS.AES.decrypt(data, key);
  return JSON.parse(bytes.toString(CryptoJS.enc.Utf8));
};

const encoder = (data) => {
  return CryptoJS.AES.encrypt(JSON.stringify(data), key).toString();
};

module.exports = { decoder, encoder };
```

3 CONTROLLER

3.1 controller.bat

```
node index.js
```

3.2 index.js

```
const { Board, Led, LCD, Servo, Button } = require("johnny-five");
const express = require("express");
const cors = require("cors");
const bodyParser = require("body-parser");
const CyrillicToTranslit = require("cyrillic-to-translit-js");
const Queue = require("./Queue");

const board = new Board();
const app = express();

app.use(cors());
app.use(
  bodyParser.urlencoded({
    extended: true,
  })
);
app.use(bodyParser.json({ limit: "50mb" }));

const startServer = () => {
  app.listen(8888, () => {
    console.log("Visit http://localhost:8888");
  });
};

board.on("ready", startServer);

app.post("/api/output-product", async (req, res) => {
  const data = req.body.data;

  var lcd = new LCD({
    controller: "PCF8574AT",
  });

  button = new Button(2);

  lcd.useChar("heart");

  data.forEach((item) => {
    Queue.enqueue(
      () =>
        new Promise((resolve, reject) => {
          lcd.clear();
          lcd
            .cursor(0, 0)
            .print(CyrillicToTranslit({ preset: "uk" }).transform(item.name));
```

```
        lcd
          .cursor(1, 0)
          .print(`count ${item.outputCount} cell ${item.cellNumber}`);
        button.on("down", () => {
          resolve();
          console.log("down");
        });
        setTimeout(() => resolve(), 3000);
      })
  );
 });

 Queue.enqueue(
  () =>
    new Promise((resolve, reject) => {
      lcd.clear();
      resolve();
      res.json(data);
    })
 );

});
```

3.2 Queue.js

```
module.exports = class Queue {
 static queue = [];
 static pendingPromise = false;

 static enqueue(promise) {
  return new Promise((resolve, reject) => {
    this.queue.push({
      promise,
      resolve,
      reject,
    });
    this.dequeue();
  });
 }

 static dequeue() {
  if (this.workingOnPromise) {
    return false;
  }
  const item = this.queue.shift();
  if (!item) {
    return false;
  }
  try {
    this.workingOnPromise = true;
    item
      .promise()
```

```
      .then((value) => {
        this.workingOnPromise = false;
        item.resolve(value);
        this.dequeue();
      })
      .catch((err) => {
        this.workingOnPromise = false;
        item.reject(err);
        this.dequeue();
      });
    } catch (err) {
      this.workingOnPromise = false;
      item.reject(err);
      this.dequeue();
    }
    return true;
  }
};
```

4 ІНТЕРФЕЙС ПРОГРАМИ

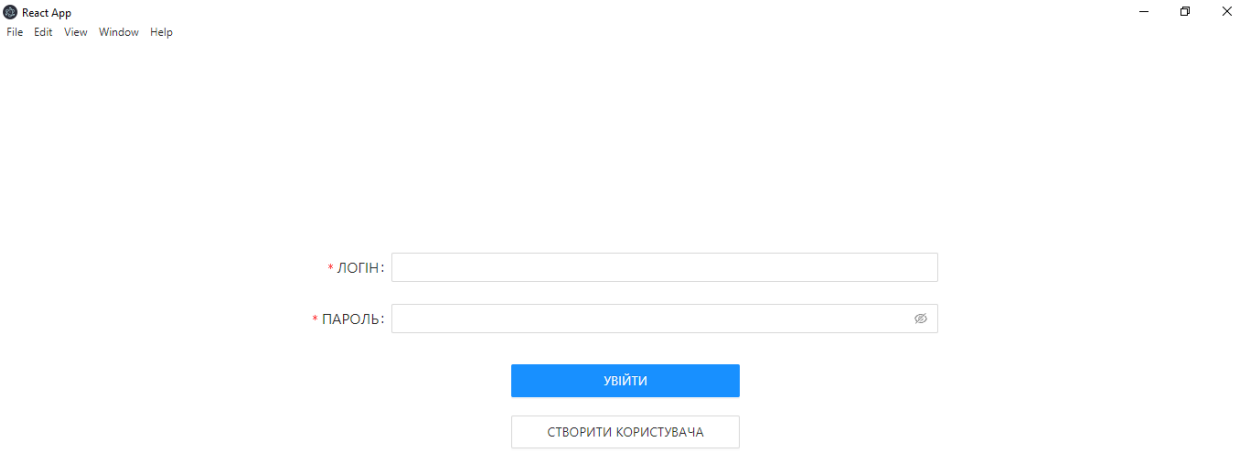


Рисунок 4.1 – Вікно авторизації

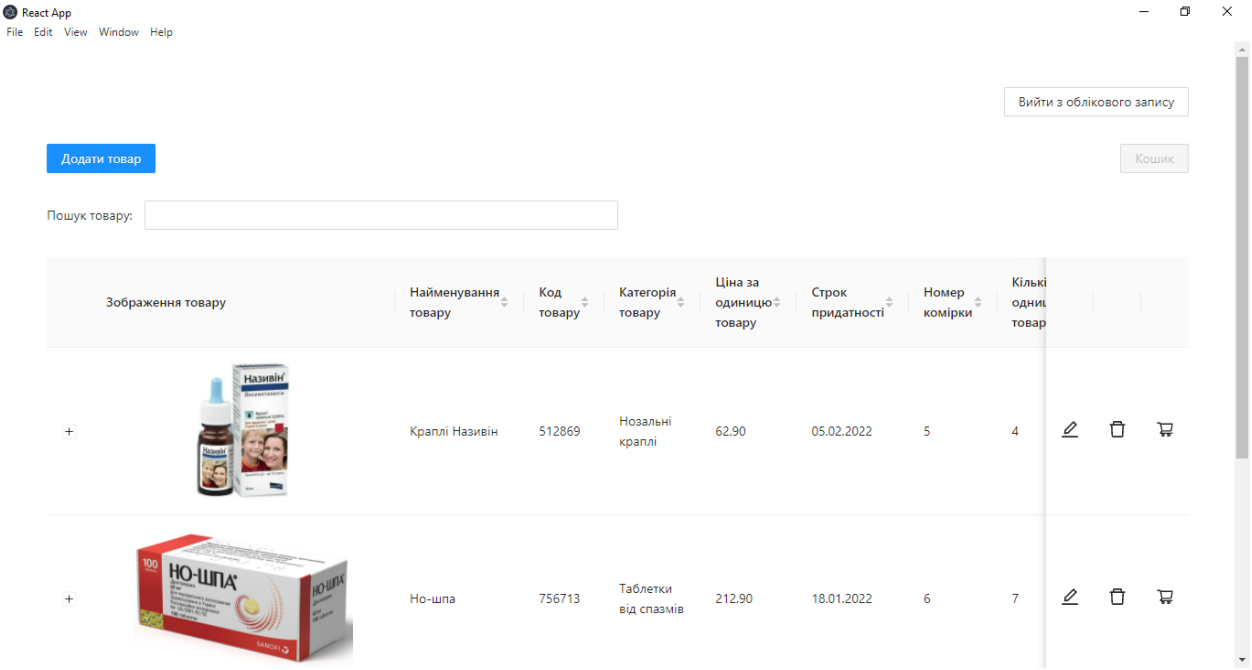


Рисунок 4.2 – Основне вікно програми

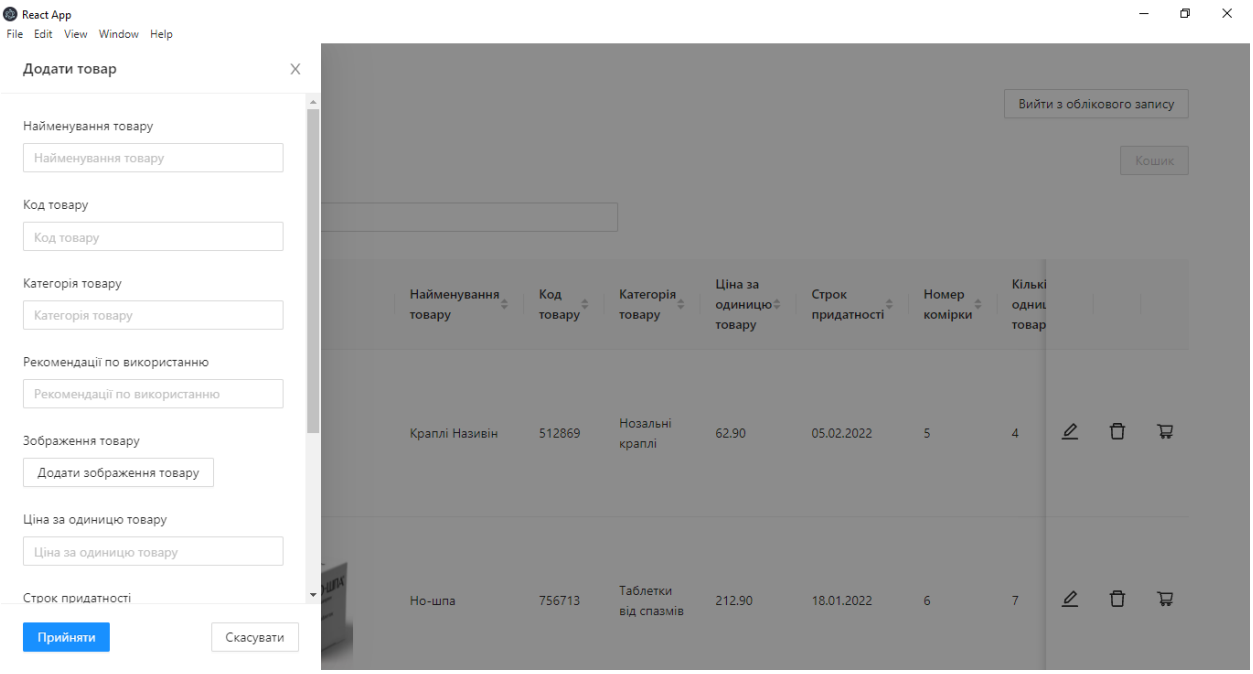


Рисунок 4.3 – Вікно додавання товару

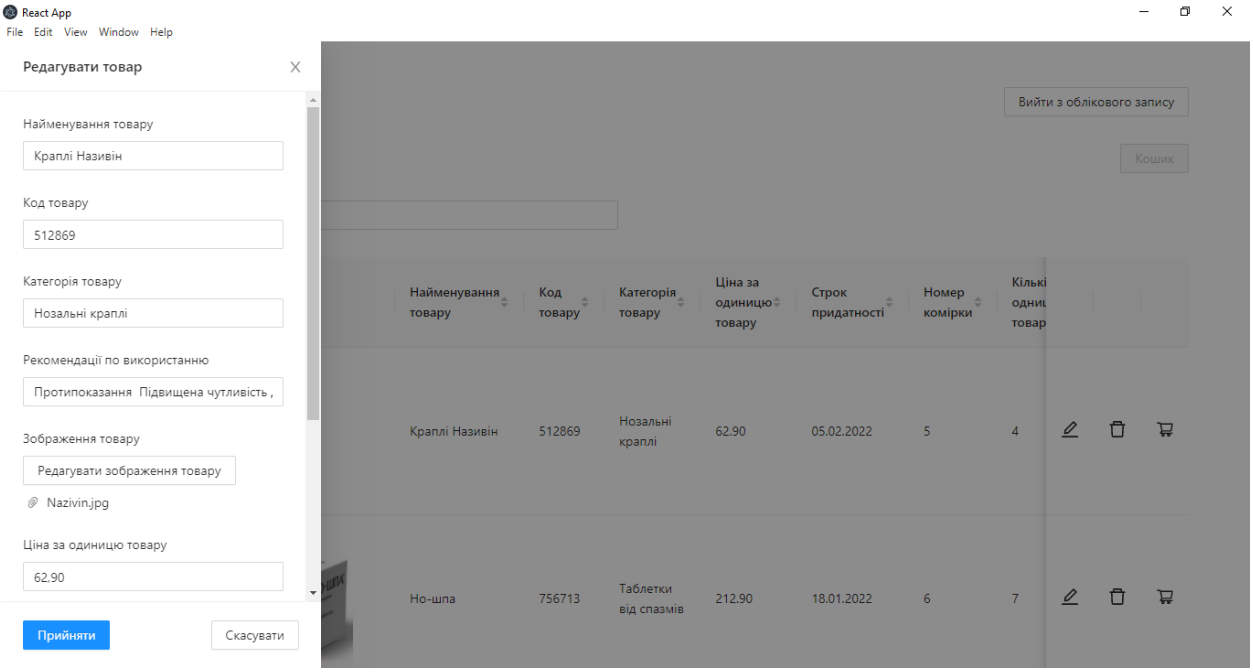


Рисунок 4.4 – Вікно редагування товару

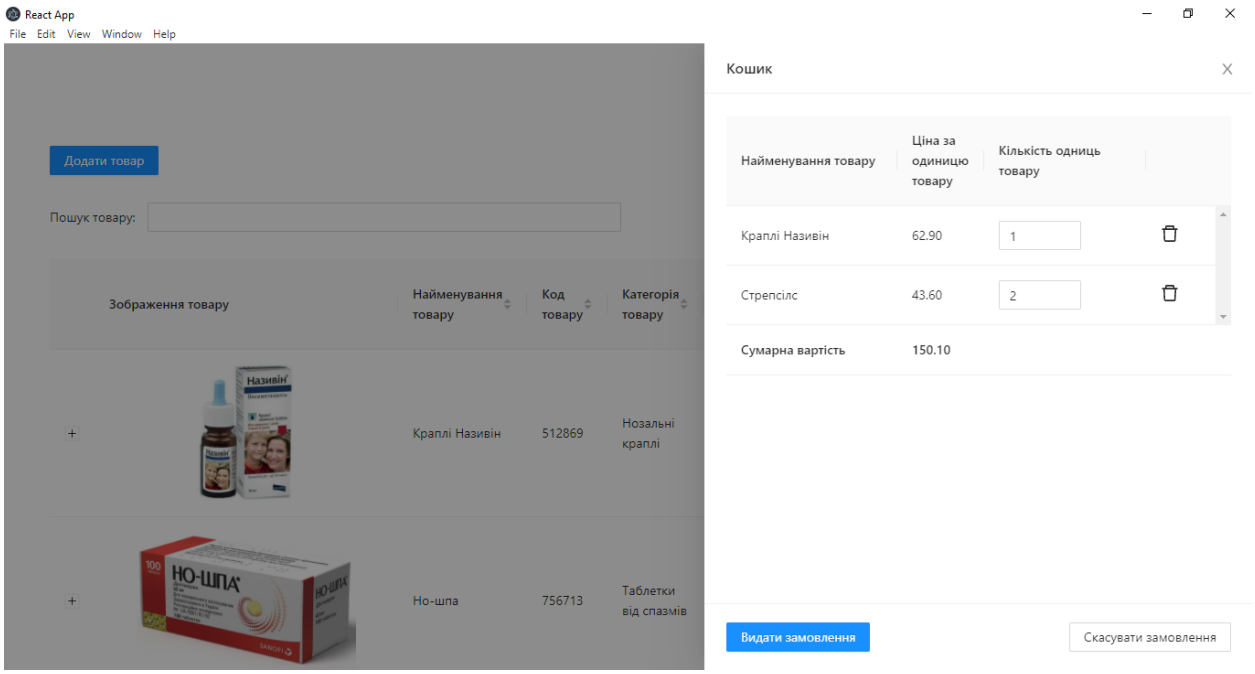


Рисунок 4.5 – Вікно кошику

ДОДАТОК Б – РОЗРАХУНОК ПАРАМЕТРІВ КОМП'ЮТЕРНОЇ МЕРЕЖІ ПО ЇЇ МАТЕМАТИЧНІЙ МОДЕЛІ ЯК ЗАМКНУТОЇ СИСТЕМИ МАСОВОГО ОБСЛУГОВУВАННЯ

Міністерство освіти і науки України
НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ
“ДНІПРОВСЬКА ПОЛІТЕХНІКА”

РОЗРАХУНОК ПАРАМЕТРІВ КОМП’ЮТЕРНОЇ МЕРЕЖІ ПО ЇЇ МАТЕМАТИЧНІЙ МОДЕЛІ ЯК ЗАМКНУТОЇ СИСТЕМИ МАСОВОГО ОБСЛУГОВУВАННЯ

Текст програми

804.02070743.22022-01 12 01

Листів 16

2022

АНОТАЦІЯ

Даний документ містить в собі програмний код розрахунку параметрів інтенсивності потоку, що входить у вузол, середнього часу перебування пакета у вузлі комп’ютерної мережі та середньої кількісті пакетів які знаходяться у вузлі по її математичній моделі як замкнутої системи масового обслуговування у програмі MathCad.

ЗМІСТ

1 ПЕРЕЛІК ВИКОРИСТАНИХ ЗМІННИХ

N_n – кількість вузлів мережі.

τ – час обробки одного пакета у вузлі.

P_r – матриця перехідних ймовірностей.

e – матриця перехідних коефіцієнтів.

m – кількість конвеєрів у вузлах.

N – кількість пакетів що циркулюють в мережі.

B – матриця ймовірностей черги у вузлах.

λ – середня інтенсивність запитів на вході у вузол.

L – середня черга пакетів у вузлі.

t – середній час перебування пакета у вузлі.

2 ТЕКСТ ПРОГРАМИ

$Nn := 26 \qquad i := 0..Nn \qquad j := 0..Nn \qquad \tau_i :=$

5
5
5
5
5
5
5
5
5
5
5
5
5
5
5
5
5
5
5
5
5
5
5
5
5
5

$\tau =$

	0
0	5
1	5
2	5
3	5
4	5
5	5
6	5
7	5
8	5
9	5
10	5
11	5
12	5
13	5
14	5
15	5
16	5
17	5
18	...

$\mu_i := \frac{1}{\tau_i}$

$\mu =$

	0
0	0.2
1	0.2
2	0.2
3	0.2
4	0.2
5	0.2
6	0.2
7	0.2
8	0.2
9	0.2
10	0.2
11	0.2
12	0.2
13	0.2
14	0.2
15	0.2
16	0.2
17	0.2
18	...

Pr :=

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26
0	0	0.16	0	0	0	0	0.14	0	0	0	0	0	0.14	0	0	0.14	0	0	0.14	0	0	0.14	0	0	0.14	0	0
1	0.4	0	0.3	0.3	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
2	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
3	0	0.4	0	0	0.3	0.3	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
4	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
5	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
6	0.2	0	0	0	0	0	0	0.16	0.16	0.16	0.16	0.16	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
7	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
8	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
9	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
10	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
11	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
12	0.5	0	0	0	0	0	0	0	0	0	0	0	0	0.25	0.25	0	0	0	0	0	0	0	0	0	0	0	0
13	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0
14	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0
15	0.5	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0.25	0.25	0	0	0	0	0	0	0	0	0
16	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0
17	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0
18	0.5	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0.25	0.25	0	0	0	0	0	0
19	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0
20	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0
21	0.5	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0.25	0.25	0	0	0
22	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0
23	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0
24	0.5	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0.25	0.25
25	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0
26	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0

$$SumPr_i := \sum_{j=0}^{Nn} Pr_{i,j}$$

SumPr =

	0
0	1
1	1
2	1
3	1
4	1
5	1
6	1
7	1
8	1
9	1
10	1
11	1
12	1
13	1
14	1
15	1
16	1
17	1
18	...

$P := Pr^T$

P =

	0	1	2	3	4	5	6	7	8	9
0	0	0.4	0	0	0	0	0.2	0	0	0
1	0.16	0	1	0.4	0	0	0	0	0	0
2	0	0.3	0	0	0	0	0	0	0	0
3	0	0.3	0	0	1	1	0	0	0	0
4	0	0	0	0.3	0	0	0	0	0	0
5	0	0	0	0.3	0	0	0	0	0	0
6	0.14	0	0	0	0	0	0	1	1	1
7	0	0	0	0	0	0	0.16	0	0	0
8	0	0	0	0	0	0	0.16	0	0	0
9	0	0	0	0	0	0	0.16	0	0	0
10	0	0	0	0	0	0	0.16	0	0	0
11	0	0	0	0	0	0	0.16	0	0	0
12	0.14	0	0	0	0	0	0	0	0	0
13	0	0	0	0	0	0	0	0	0	0
14	0	0	0	0	0	0	0	0	0	0
15	0.14	0	0	0	0	0	0	0	0	0
16	0	0	0	0	0	0	0	0	0	0
17	0	0	0	0	0	0	0	0	0	...

D :=

	0	1	2	3	4	5	6	7	8	9
0	1	0	0	0	0	0	0	0	0	0
1	0	1	0	0	0	0	0	0	0	0
2	0	0	1	0	0	0	0	0	0	0
3	0	0	0	1	0	0	0	0	0	0
4	0	0	0	0	1	0	0	0	0	0
5	0	0	0	0	0	1	0	0	0	0
6	0	0	0	0	0	0	1	0	0	0
7	0	0	0	0	0	0	0	1	0	0
8	0	0	0	0	0	0	0	0	1	0
9	0	0	0	0	0	0	0	0	0	1
10	0	0	0	0	0	0	0	0	0	0
11	0	0	0	0	0	0	0	0	0	0
12	0	0	0	0	0	0	0	0	0	0
13	0	0	0	0	0	0	0	0	0	0
14	0	0	0	0	0	0	0	0	0	0
15	0	0	0	0	0	0	0	0	0	0
16	0	0	0	0	0	0	0	0	0	0
17	0	0	0	0	0	0	0	0	0	...

$P1 := P - D$

P1 =

	0	1	2	3	4	5	6	7	8	9
0	-1	0.4	0	0	0	0	0.2	0	0	0
1	0.16	-1	1	0.4	0	0	0	0	0	0
2	0	0.3	-1	0	0	0	0	0	0	0
3	0	0.3	0	-1	1	1	0	0	0	0
4	0	0	0	0.3	-1	0	0	0	0	0
5	0	0	0	0.3	0	-1	0	0	0	0
6	0.14	0	0	0	0	0	-1	1	1	1
7	0	0	0	0	0	0	0.16	-1	0	0
8	0	0	0	0	0	0	0.16	0	-1	0
9	0	0	0	0	0	0	0.16	0	0	-1
10	0	0	0	0	0	0	0.16	0	0	0
11	0	0	0	0	0	0	0.16	0	0	0
12	0.14	0	0	0	0	0	0	0	0	0
13	0	0	0	0	0	0	0	0	0	0
14	0	0	0	0	0	0	0	0	0	0
15	0.14	0	0	0	0	0	0	0	0	0
16	0	0	0	0	0	0	0	0	0	0
17	0	0	0	0	0	0	0	0	0	...

$j := 1..Nn \qquad i := 0..Nn$

$P2_{(j-1),i} := P1_{0,i} + P1_{j,i}$

P2 =

	0	1	2	3	4	5	6	7	8	9
0	-0.84	-0.6	1	0.4	0	0	0.2	0	0	0
1	-1	0.7	-1	0	0	0	0.2	0	0	0
2	-1	0.7	0	-1	1	1	0.2	0	0	0
3	-1	0.4	0	0.3	-1	0	0.2	0	0	0
4	-1	0.4	0	0.3	0	-1	0.2	0	0	0
5	-0.86	0.4	0	0	0	0	-0.8	1	1	1
6	-1	0.4	0	0	0	0	0.36	-1	0	0
7	-1	0.4	0	0	0	0	0.36	0	-1	0
8	-1	0.4	0	0	0	0	0.36	0	0	-1
9	-1	0.4	0	0	0	0	0.36	0	0	0
10	-1	0.4	0	0	0	0	0.36	0	0	0
11	-0.86	0.4	0	0	0	0	0.2	0	0	0
12	-1	0.4	0	0	0	0	0.2	0	0	0
13	-1	0.4	0	0	0	0	0.2	0	0	0
14	-0.86	0.4	0	0	0	0	0.2	0	0	0
15	-1	0.4	0	0	0	0	0.2	0	0	0
16	-1	0.4	0	0	0	0	0.2	0	0	0
17	-0.86	0.4	0	0	0	0	0.2	0	0	...

$j := 0..Nn - 1 \qquad i := 0..Nn - 1 \qquad PP2_{j,i} := P2_{j,i+1}$

PP2 =

	0	1	2	3	4	5	6	7	8	9
0	-0.6	1	0.4	0	0	0.2	0	0	0	0
1	0.7	-1	0	0	0	0.2	0	0	0	0
2	0.7	0	-1	1	1	0.2	0	0	0	0
3	0.4	0	0.3	-1	0	0.2	0	0	0	0
4	0.4	0	0.3	0	-1	0.2	0	0	0	0
5	0.4	0	0	0	0	-0.8	1	1	1	1
6	0.4	0	0	0	0	0.36	-1	0	0	0
7	0.4	0	0	0	0	0.36	0	-1	0	0
8	0.4	0	0	0	0	0.36	0	0	-1	0
9	0.4	0	0	0	0	0.36	0	0	0	-1
10	0.4	0	0	0	0	0.36	0	0	0	0
11	0.4	0	0	0	0	0.2	0	0	0	0
12	0.4	0	0	0	0	0.2	0	0	0	0
13	0.4	0	0	0	0	0.2	0	0	0	0
14	0.4	0	0	0	0	0.2	0	0	0	0
15	0.4	0	0	0	0	0.2	0	0	0	0
16	0.4	0	0	0	0	0.2	0	0	0	0
17	0.4	0	0	0	0	0.2	0	0	0	...

$Q_{j,0} := P2_{j,0}$

Q =

	0
0	-0.84
1	-1
2	-1
3	-1
4	-1
5	-0.86
6	-1
7	-1
8	-1
9	-1
10	-1
11	-0.86
12	-1
13	-1
14	-0.86
15	-1
16	-1
17	...

$E := lsolve(PP2, Q)$

E =

	0
0	-0.4
1	-0.12
2	-0.3
3	-0.09
4	-0.09
5	-0.7
6	-0.112
7	-0.112
8	-0.112
9	-0.112
10	-0.112
11	-0.28
12	-0.07
13	-0.07
14	-0.28
15	-0.07
16	-0.07
17	...

e :=

1
0.4
0.12
0.3
0.09
0.09
0.7
0.112
0.112
0.112
0.112
0.112
0.28
0.07
0.07
0.28
0.07
0.07
0.28
0.07
0.07
0.28
0.07
0.07
0.28
0.07
0.07

$m_i :=$ $N := 60$ $i := 0..Nn$ $j := 0..N - 1$

1
1
1
1
1
1
1
1
1
1
1
1
1
1
1
1
1
1
1
1
1
1
1
1
1
1
1
1
1
1

$m =$

	0
0	1
1	1
2	1
3	1
4	1
5	1
6	1
7	1
8	1
9	1
10	1
11	1
12	1
13	1
14	1
15	1
16	1
17	1
18	1
19	...

$$A_{i,j} := \begin{cases} j! & \text{if } m_i \geq N - 1 \\ 1 & \text{if } m_i = 1 \\ j! & \text{if } 1 < m_i < N - 1 \wedge j \leq m_i \\ m_i! \cdot (m_i)^{j - m_i} & \text{if } 1 < m_i < N - 1 \wedge j > m_i \end{cases}$$

A =

	0	1	2	3	4	5	6	7	8	9
0	1	1	1	1	1	1	1	1	1	1
1	1	1	1	1	1	1	1	1	1	1
2	1	1	1	1	1	1	1	1	1	1
3	1	1	1	1	1	1	1	1	1	1
4	1	1	1	1	1	1	1	1	1	1
5	1	1	1	1	1	1	1	1	1	1
6	1	1	1	1	1	1	1	1	1	1
7	1	1	1	1	1	1	1	1	1	1
8	1	1	1	1	1	1	1	1	1	1
9	1	1	1	1	1	1	1	1	1	1
10	1	1	1	1	1	1	1	1	1	1
11	1	1	1	1	1	1	1	1	1	1
12	1	1	1	1	1	1	1	1	1	1
13	1	1	1	1	1	1	1	1	1	1
14	1	1	1	1	1	1	1	1	1	1
15	1	1	1	1	1	1	1	1	1	1
16	1	1	1	1	1	1	1	1	1	1
17	1	1	1	1	1	1	1	1	1	...

$$X_i := \frac{e_i}{\mu_i}$$

X =

	0
0	5
1	2
2	0.6
3	1.5
4	0.45
5	0.45
6	3.5
7	0.56
8	0.56
9	0.56
10	0.56
11	0.56
12	1.4
13	0.35
14	0.35
15	1.4
16	0.35
17	...

$$T_{i,j} := \frac{(X_i)^j}{A_{i,j}} \qquad T_{i,0} := 1$$

$T =$

	0	1	2	3	4	5	6	7	8	9
0	1	5	25	125	625	$3.125 \cdot 10^3$	$1.563 \cdot 10^4$	$7.813 \cdot 10^4$	$3.906 \cdot 10^5$	$1.953 \cdot 10^6$
1	1	2	4	8	16	32	64	128	256	512
2	1	0.6	0.36	0.216	0.13	0.078	0.047	0.028	0.017	0.01
3	1	1.5	2.25	3.375	5.062	7.594	11.391	17.086	25.629	38.443
4	1	0.45	0.202	0.091	0.041	0.018	$8.304 \cdot 10^{-3}$	$3.737 \cdot 10^{-3}$	$1.682 \cdot 10^{-3}$	$7.567 \cdot 10^{-4}$
5	1	0.45	0.202	0.091	0.041	0.018	$8.304 \cdot 10^{-3}$	$3.737 \cdot 10^{-3}$	$1.682 \cdot 10^{-3}$	$7.567 \cdot 10^{-4}$
6	1	3.5	12.25	42.875	150.062	525.219	$1.838 \cdot 10^3$	$6.434 \cdot 10^3$	$2.252 \cdot 10^4$	$7.882 \cdot 10^4$
7	1	0.56	0.314	0.176	0.098	0.055	0.031	0.017	$9.672 \cdot 10^{-3}$	$5.416 \cdot 10^{-3}$
8	1	0.56	0.314	0.176	0.098	0.055	0.031	0.017	$9.672 \cdot 10^{-3}$	$5.416 \cdot 10^{-3}$
9	1	0.56	0.314	0.176	0.098	0.055	0.031	0.017	$9.672 \cdot 10^{-3}$	$5.416 \cdot 10^{-3}$
10	1	0.56	0.314	0.176	0.098	0.055	0.031	0.017	$9.672 \cdot 10^{-3}$	$5.416 \cdot 10^{-3}$
11	1	0.56	0.314	0.176	0.098	0.055	0.031	0.017	$9.672 \cdot 10^{-3}$	$5.416 \cdot 10^{-3}$
12	1	1.4	1.96	2.744	3.842	5.378	7.53	10.541	14.758	20.661
13	1	0.35	0.123	0.043	0.015	$5.252 \cdot 10^{-3}$	$1.838 \cdot 10^{-3}$	$6.434 \cdot 10^{-4}$	$2.252 \cdot 10^{-4}$	$7.882 \cdot 10^{-5}$
14	1	0.35	0.123	0.043	0.015	$5.252 \cdot 10^{-3}$	$1.838 \cdot 10^{-3}$	$6.434 \cdot 10^{-4}$	$2.252 \cdot 10^{-4}$	$7.882 \cdot 10^{-5}$
15	1	1.4	1.96	2.744	3.842	5.378	7.53	10.541	14.758	20.661
16	1	0.35	0.123	0.043	0.015	$5.252 \cdot 10^{-3}$	$1.838 \cdot 10^{-3}$	$6.434 \cdot 10^{-4}$	$2.252 \cdot 10^{-4}$	$7.882 \cdot 10^{-5}$
17	1	0.35	0.123	0.043	0.015	$5.252 \cdot 10^{-3}$	$1.838 \cdot 10^{-3}$	$6.434 \cdot 10^{-4}$	$2.252 \cdot 10^{-4}$	...

$i := 1..Nn \qquad k := 0..N-1 \qquad G_{0,j} := T_{0,j}$

$$G_{i,k} := \sum_{j=0}^{k} \left(T_{i,j} \cdot G_{i-1,k-j}\right)$$

$G =$

	0	1	2	3	4	5	6	7	8	9
0	1	5	25	125	625	$3.125 \cdot 10^3$	$1.563 \cdot 10^4$	$7.813 \cdot 10^4$	$3.906 \cdot 10^5$	$1.953 \cdot 10^6$
1	1	7	39	203	$1.031 \cdot 10^3$	$5.187 \cdot 10^3$	$2.6 \cdot 10^4$	$1.301 \cdot 10^5$	$6.509 \cdot 10^5$	$3.255 \cdot 10^6$
2	1	7.6	43.56	229.136	$1.168 \cdot 10^3$	$5.888 \cdot 10^3$	$2.953 \cdot 10^4$	$1.478 \cdot 10^5$	$7.396 \cdot 10^5$	$3.699 \cdot 10^6$
3	1	9.1	57.21	314.951	$1.641 \cdot 10^3$	$8.349 \cdot 10^3$	$4.206 \cdot 10^4$	$2.109 \cdot 10^5$	$1.056 \cdot 10^6$	$5.283 \cdot 10^6$
4	1	9.55	61.508	342.629	$1.795 \cdot 10^3$	$9.157 \cdot 10^3$	$4.618 \cdot 10^4$	$2.317 \cdot 10^5$	$1.16 \cdot 10^6$	$5.805 \cdot 10^6$
5	1	10	66.008	372.333	$1.963 \cdot 10^3$	$1.004 \cdot 10^4$	$5.069 \cdot 10^4$	$2.545 \cdot 10^5$	$1.275 \cdot 10^6$	$6.378 \cdot 10^6$
6	1	13.5	113.257	768.734	$4.653 \cdot 10^3$	$2.633 \cdot 10^4$	$1.428 \cdot 10^5$	$7.545 \cdot 10^5$	$3.915 \cdot 10^6$	$2.008 \cdot 10^7$
7	1	14.06	121.131	836.567	$5.122 \cdot 10^3$	$2.919 \cdot 10^4$	$1.592 \cdot 10^5$	$8.436 \cdot 10^5$	$4.388 \cdot 10^6$	$2.254 \cdot 10^7$
8	1	14.62	129.318	908.986	$5.631 \cdot 10^3$	$3.235 \cdot 10^4$	$1.773 \cdot 10^5$	$9.429 \cdot 10^5$	$4.916 \cdot 10^6$	$2.529 \cdot 10^7$
9	1	15.18	137.819	986.164	$6.183 \cdot 10^3$	$3.581 \cdot 10^4$	$1.974 \cdot 10^5$	$1.053 \cdot 10^6$	$5.506 \cdot 10^6$	$2.838 \cdot 10^7$
10	1	15.74	146.633	$1.068 \cdot 10^3$	$6.781 \cdot 10^3$	$3.961 \cdot 10^4$	$2.195 \cdot 10^5$	$1.176 \cdot 10^6$	$6.164 \cdot 10^6$	$3.183 \cdot 10^7$
11	1	16.3	155.762	$1.156 \cdot 10^3$	$7.428 \cdot 10^3$	$4.377 \cdot 10^4$	$2.44 \cdot 10^5$	$1.313 \cdot 10^6$	$6.9 \cdot 10^6$	$3.569 \cdot 10^7$
12	1	17.7	180.542	$1.408 \cdot 10^3$	$9.4 \cdot 10^3$	$5.693 \cdot 10^4$	$3.237 \cdot 10^5$	$1.766 \cdot 10^6$	$9.372 \cdot 10^6$	$4.881 \cdot 10^7$
13	1	18.05	186.859	$1.474 \cdot 10^3$	$9.916 \cdot 10^3$	$6.04 \cdot 10^4$	$3.449 \cdot 10^5$	$1.887 \cdot 10^6$	$1.003 \cdot 10^7$	$5.232 \cdot 10^7$
14	1	18.4	193.299	$1.541 \cdot 10^3$	$1.046 \cdot 10^4$	$6.406 \cdot 10^4$	$3.673 \cdot 10^5$	$2.016 \cdot 10^6$	$1.074 \cdot 10^7$	$5.608 \cdot 10^7$
15	1	19.8	221.019	$1.851 \cdot 10^3$	$1.305 \cdot 10^4$	$8.232 \cdot 10^4$	$4.826 \cdot 10^5$	$2.691 \cdot 10^6$	$1.451 \cdot 10^7$	$7.639 \cdot 10^7$
16	1	20.15	228.072	$1.931 \cdot 10^3$	$1.372 \cdot 10^4$	$8.712 \cdot 10^4$	$5.13 \cdot 10^5$	$2.871 \cdot 10^6$	$1.551 \cdot 10^7$	$8.182 \cdot 10^7$
17	1	20.5	235.247	$2.013 \cdot 10^3$	$1.443 \cdot 10^4$	$9.217 \cdot 10^4$	$5.453 \cdot 10^5$	$3.062 \cdot 10^6$	$1.658 \cdot 10^7$	...

$i := 0..Nn - 1 \qquad j := 0..N - 1 \qquad B_{i,j} := \frac{T_{i,j}}{G_{Nn,N-1}} Gn_{i,N-1-j}$

B =

	0	1	2	3	4	5	6	7	8	9
0	$1.972 \cdot 10^{-9}$	$2.817 \cdot 10^{-9}$	$4.025 \cdot 10^{-9}$	$5.749 \cdot 10^{-9}$	$8.214 \cdot 10^{-9}$	$1.173 \cdot 10^{-8}$	$1.676 \cdot 10^{-8}$	$2.395 \cdot 10^{-8}$	$3.421 \cdot 10^{-8}$	$4.887 \cdot 10^{-8}$
1	0.6	0.24	0.096	0.038	0.015	$6.144 \cdot 10^{-3}$	$2.458 \cdot 10^{-3}$	$9.83 \cdot 10^{-4}$	$3.932 \cdot 10^{-4}$	$1.573 \cdot 10^{-4}$
2	0.88	0.106	0.013	$1.521 \cdot 10^{-3}$	$1.825 \cdot 10^{-4}$	$2.19 \cdot 10^{-5}$	$2.628 \cdot 10^{-6}$	$3.153 \cdot 10^{-7}$	$3.784 \cdot 10^{-8}$	$4.541 \cdot 10^{-9}$
3	0.7	0.21	0.063	0.019	$5.67 \cdot 10^{-3}$	$1.701 \cdot 10^{-3}$	$5.103 \cdot 10^{-4}$	$1.531 \cdot 10^{-4}$	$4.593 \cdot 10^{-5}$	$1.378 \cdot 10^{-5}$
4	0.91	0.082	$7.371 \cdot 10^{-3}$	$6.634 \cdot 10^{-4}$	$5.971 \cdot 10^{-5}$	$5.373 \cdot 10^{-6}$	$4.836 \cdot 10^{-7}$	$4.353 \cdot 10^{-8}$	$3.917 \cdot 10^{-9}$	$3.526 \cdot 10^{-10}$
5	0.91	0.082	$7.371 \cdot 10^{-3}$	$6.634 \cdot 10^{-4}$	$5.971 \cdot 10^{-5}$	$5.373 \cdot 10^{-6}$	$4.836 \cdot 10^{-7}$	$4.353 \cdot 10^{-8}$	$3.917 \cdot 10^{-9}$	$3.526 \cdot 10^{-10}$
6	0.3	0.21	0.147	0.103	0.072	0.05	0.035	0.025	0.017	0.012
7	0.888	0.099	0.011	$1.248 \cdot 10^{-3}$	$1.397 \cdot 10^{-4}$	$1.565 \cdot 10^{-5}$	$1.753 \cdot 10^{-6}$	$1.963 \cdot 10^{-7}$	$2.199 \cdot 10^{-8}$	$2.462 \cdot 10^{-9}$
8	0.888	0.099	0.011	$1.248 \cdot 10^{-3}$	$1.397 \cdot 10^{-4}$	$1.565 \cdot 10^{-5}$	$1.753 \cdot 10^{-6}$	$1.963 \cdot 10^{-7}$	$2.199 \cdot 10^{-8}$	$2.462 \cdot 10^{-9}$
9	0.888	0.099	0.011	$1.248 \cdot 10^{-3}$	$1.397 \cdot 10^{-4}$	$1.565 \cdot 10^{-5}$	$1.753 \cdot 10^{-6}$	$1.963 \cdot 10^{-7}$	$2.199 \cdot 10^{-8}$	$2.462 \cdot 10^{-9}$
10	0.888	0.099	0.011	$1.248 \cdot 10^{-3}$	$1.397 \cdot 10^{-4}$	$1.565 \cdot 10^{-5}$	$1.753 \cdot 10^{-6}$	$1.963 \cdot 10^{-7}$	$2.199 \cdot 10^{-8}$	$2.462 \cdot 10^{-9}$
11	0.888	0.099	0.011	$1.248 \cdot 10^{-3}$	$1.397 \cdot 10^{-4}$	$1.565 \cdot 10^{-5}$	$1.753 \cdot 10^{-6}$	$1.963 \cdot 10^{-7}$	$2.199 \cdot 10^{-8}$	$2.462 \cdot 10^{-9}$
12	0.72	0.202	0.056	0.016	$4.426 \cdot 10^{-3}$	$1.239 \cdot 10^{-3}$	$3.47 \cdot 10^{-4}$	$9.715 \cdot 10^{-5}$	$2.72 \cdot 10^{-5}$	$7.616 \cdot 10^{-6}$
13	0.93	0.065	$4.557 \cdot 10^{-3}$	$3.19 \cdot 10^{-4}$	$2.233 \cdot 10^{-5}$	$1.563 \cdot 10^{-6}$	$1.094 \cdot 10^{-7}$	$7.659 \cdot 10^{-9}$	$5.361 \cdot 10^{-10}$	$3.753 \cdot 10^{-11}$
14	0.93	0.065	$4.557 \cdot 10^{-3}$	$3.19 \cdot 10^{-4}$	$2.233 \cdot 10^{-5}$	$1.563 \cdot 10^{-6}$	$1.094 \cdot 10^{-7}$	$7.659 \cdot 10^{-9}$	$5.361 \cdot 10^{-10}$	$3.753 \cdot 10^{-11}$
15	0.72	0.202	0.056	0.016	$4.426 \cdot 10^{-3}$	$1.239 \cdot 10^{-3}$	$3.47 \cdot 10^{-4}$	$9.715 \cdot 10^{-5}$	$2.72 \cdot 10^{-5}$	$7.616 \cdot 10^{-6}$
16	0.93	0.065	$4.557 \cdot 10^{-3}$	$3.19 \cdot 10^{-4}$	$2.233 \cdot 10^{-5}$	$1.563 \cdot 10^{-6}$	$1.094 \cdot 10^{-7}$	$7.659 \cdot 10^{-9}$	$5.361 \cdot 10^{-10}$	$3.753 \cdot 10^{-11}$
17	0.93	0.065	$4.557 \cdot 10^{-3}$	$3.19 \cdot 10^{-4}$	$2.233 \cdot 10^{-5}$	$1.563 \cdot 10^{-6}$	$1.094 \cdot 10^{-7}$	$7.659 \cdot 10^{-9}$	$5.361 \cdot 10^{-10}$	...

$i := 0..Nn \qquad j := 0..N - 1$

$SumB_i := \sum_j B_{i,j}$

SumB =

	0
0	1
1	1
2	1
3	1
4	1
5	1
6	1
7	1
8	1
9	1
10	1
11	1
12	1
13	1
14	1
15	1
16	1
17	...

$$\lambda_i := e_i \cdot \frac{G_{Nn-1,N-2}}{G_{Nn,N-1}} \qquad L_i := \sum_{n=0}^{N-1} \left(n \cdot B_{i,n}\right)$$

$\lambda =$

	0
0	0.186
1	0.074
2	0.022
3	0.056
4	0.017
5	0.017
6	0.13
7	0.021
8	0.021
9	0.021
10	0.021
11	0.021
12	0.052
13	0.013
14	0.013
15	0.052
16	0.013
17	0.013
18	...

$L =$

	0
0	51.905
1	0.667
2	0.136
3	0.429
4	0.099
5	0.099
6	2.333
7	0.126
8	0.126
9	0.126
10	0.126
11	0.126
12	0.389
13	0.075
14	0.075
15	0.389
16	0.075
17	0.075
18	...

$$\lambda_i := e_i \cdot \frac{G_{Nn-1,N-2}}{G_{Nn,N-1}} \qquad L_i := \sum_{n=0}^{N-1} \left(n \cdot B_{i,n}\right)$$

$\lambda =$

	0
0	0.186
1	0.074
2	0.022
3	0.056
4	0.017
5	0.017
6	0.13
7	0.021
8	0.021
9	0.021
10	0.021
11	0.021
12	0.052
13	0.013
14	0.013
15	0.052
16	0.013
17	0.013
18	...

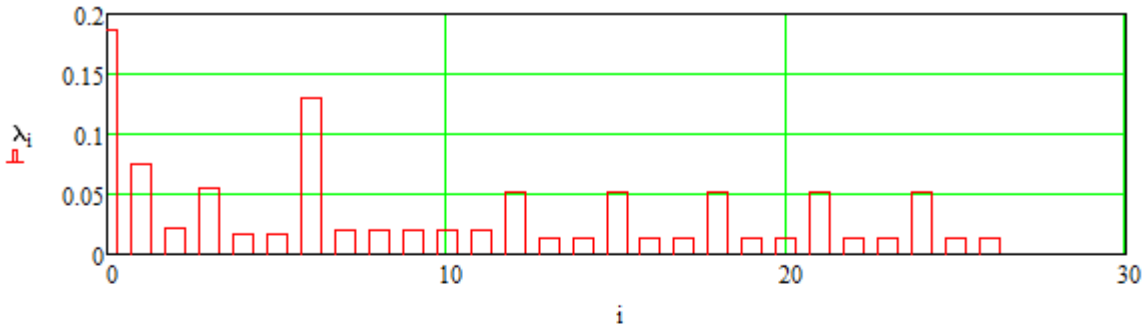
$L =$

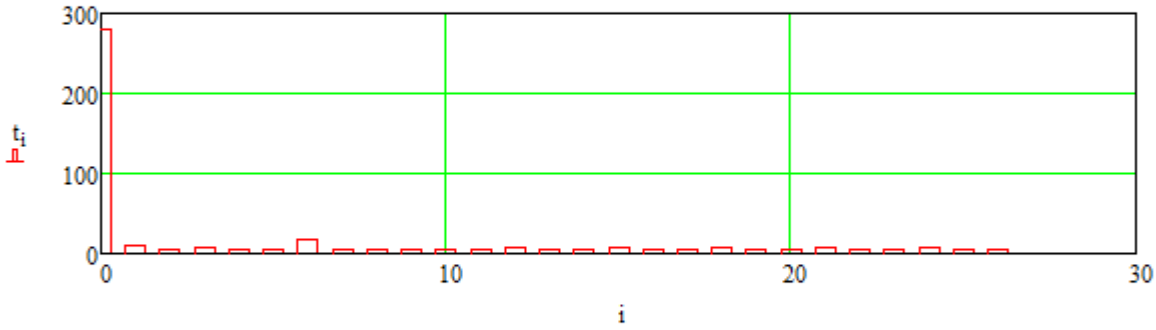
	0
0	51.905
1	0.667
2	0.136
3	0.429
4	0.099
5	0.099
6	2.333
7	0.126
8	0.126
9	0.126
10	0.126
11	0.126
12	0.389
13	0.075
14	0.075
15	0.389
16	0.075
17	0.075
18	...

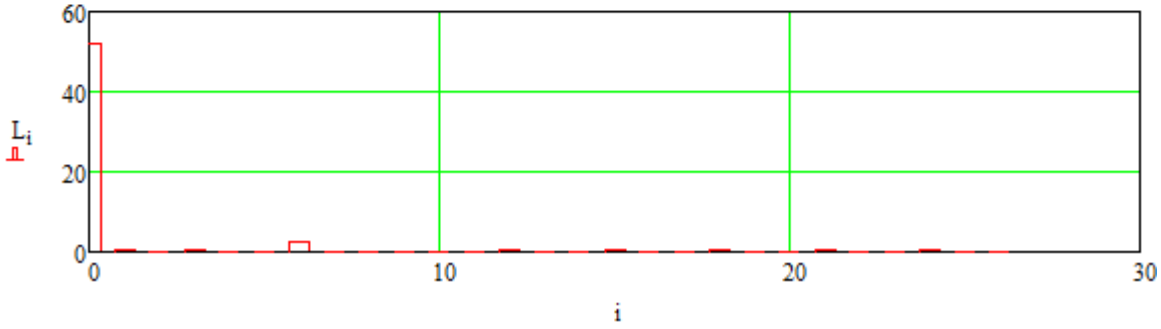
$$t_i := \frac{L_i}{\lambda_i}$$

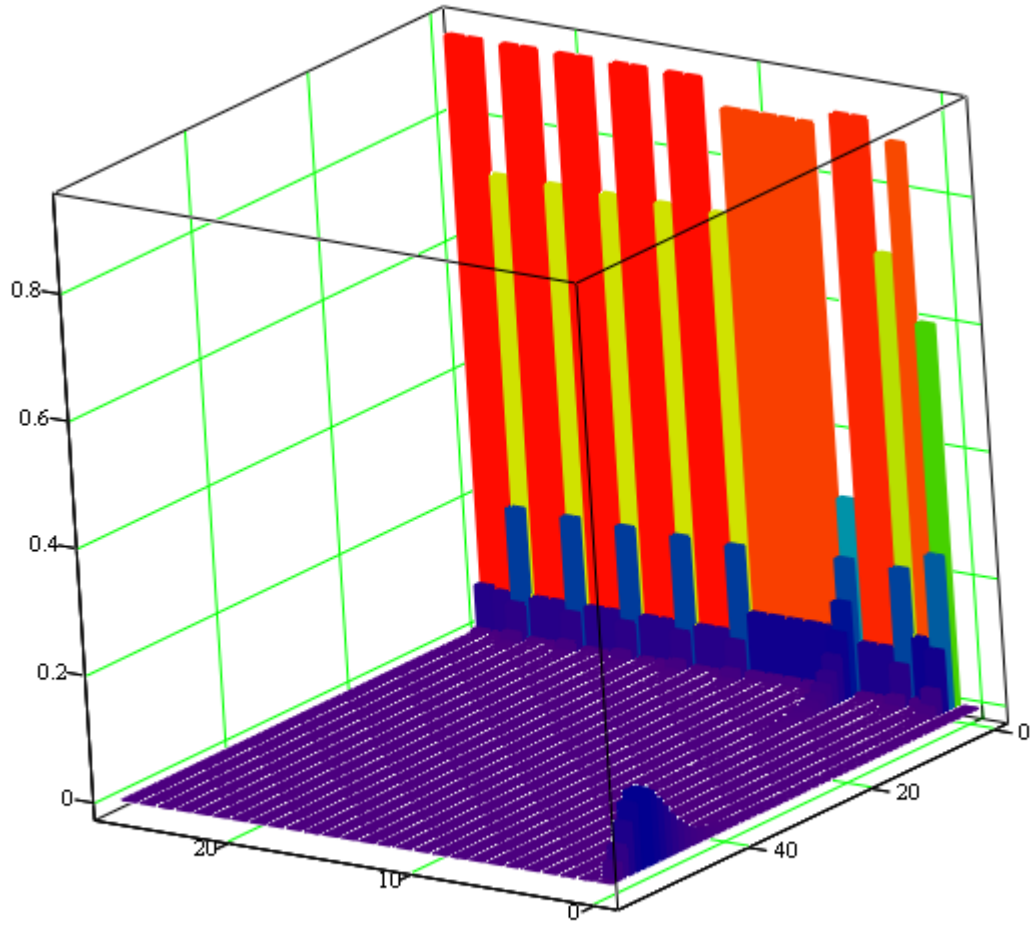
$t =$

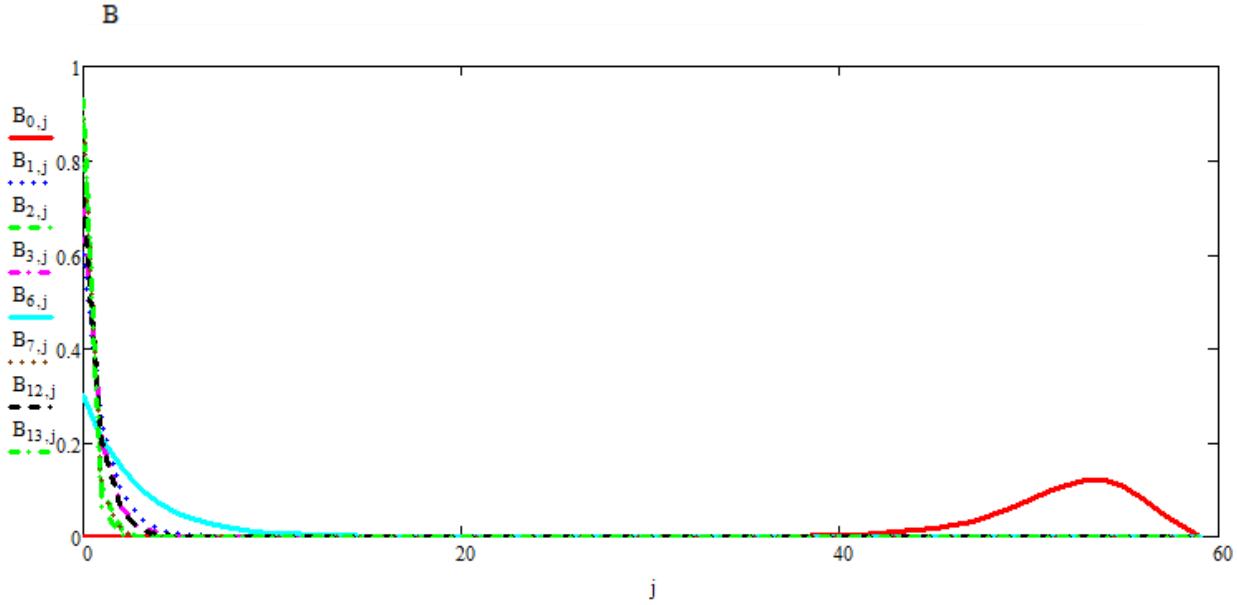
	0
0	279.058
1	8.961
2	6.109
3	7.68
4	5.908
5	5.908
6	17.921
7	6.054
8	6.054
9	6.054
10	6.054
11	6.054
12	7.467
13	5.781
14	5.781
15	7.467
16	5.781
17	5.781
18	...


0.2
0.15
0.1
0.05
0
λi
0
10
20
30
i


300
200
100
0
ti
0
10
20
30
i


60
40
20
0
Li
0
10
20
30
i


0.8
0.6
0.4
0.2
0
20
10
0
0
20
40


B
B0,j
B1,j
B2,j
B3,j
B6,j
B7,j
B12,j
B13,j
1
0.8
0.6
0.4
0.2
0
0
20
40
60
j