НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
“КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ
імені ІГОРЯ СІКОРСЬКОГО”

Факультет інформатики та обчислювальної техніки

Кафедра обчислювальної техніки

До захисту допущено:
Завідувач кафедри
_____ Сергій СТІРЕНКО
(підпис)
“__” _____2022 р.

Дипломний проєкт

на здобуття ступеня бакалавра

за освітньо-професійною програмою “ Комп’ютерні системи та мережі ”

спеціальності 123 “Комп’ютерна інженерія”

на тему: Система керування чергами в супермаркетах

Виконав : студент __4__ курсу, групи _ІВ-81_
(шифр групи)

_____Савічев Денис Анатолійович_____
(прізвище, ім’я, по батькові) (підпис)

Керівник _____доцент, д.т.н. Ткаченко В.В._____
(посада, науковий ступінь, вчене звання, прізвище та ініціали) (підпис)

Консультант(нормоконтроль) професор, д.т.н., ___Сімоненко В. П._____
(назва розділу) (посада, вчене звання, науковий ступінь, прізвище та ініціали) (підпис)

Рецензент _____
(посада, науковий ступінь, вчене звання, прізвище та ініціали) (підпис)

Засвідчую, що у цьому дипломному проєкті немає запозичень з праць інших авторів без відповідних посилань.
Студент _____
(підпис)

Київ – 2022 р.

НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
“КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ
імені ІГОРЯ СІКОРСЬКОГО”

Факультет інформатики та обчислювальної техніки

Кафедра обчислювальної техніки

Рівень вищої освіти – перший (бакалавр)

Освітньо-професійна програма

“Комп’ютерні системи та мережі”

спеціальності 123 “Комп’ютерна інженерія”

ЗАТВЕРДЖУЮ
Завідувач кафедри
_____ Сергій СТІРЕНКО
(підпис)
“__” _____2022 р.

ЗАВДАННЯ
на бакалаврський дипломний проєкт студента

Савічева Дениса Анатолійовича

1. Тема проєкту *Система для керування чергами в супермаркетах*
керівник проєкту Ткаченко Валентина Василівна, доцент, к.т.н.,
(прізвище, ім’я, по батькові, науковий ступінь, вчене звання)
затверджені наказом по університету від 11 травня 2022 року №1139-с

2. Термін здачі студентом закінченого проєкту *11 червня 2022 р.*

3. Вихідні дані до проєкту *технічна документація, теоретичні дані.*

4. Зміст розрахунково-пояснювальної записки (перелік питань, які розробляються)

Розділ 1. Огляд існуючих рішень для побудови систем керування чергами.

Розділ 2. Огляд алгоритмів та технологій для розробки системи.

Розділ 3. Деталі розробки системи.

Розділ 4. Дослідження та аналіз розробленої системи.

5. Перелік графічного матеріалу (з точним позначенням обов'язкових креслень) структурна схема системи, функціональна схема (діаграма класів), алгоритм дій програмного забезпечення.

6. Консультанта проєкту, з вказівкою розділів проєкту, які до них вносяться

Розділ	Консультант	Підпис, дата	
		Завдання видав	Завдання прийняв
Нормоконтроль	Сімоненко В. П.		

7. Дата видачі завдання «30» серпня 2021 р.

Календарний план

№ п/п	Найменування етапів дипломного проєкту	Терміни виконання етапів	Примітки
1	*Затвердження теми проєкту*	*10.12.2021-15.12.2022*	
2	*Вивчення та аналіз завдання*	*15.12.2022-15.03.2022*	
3.	*Розробка архітектури та загальної структури системи*	*15.03.2022-25.03.2022*	
4.	*Розробка структур окремих підсистем*	*25.03.2022-5.04.2022*	
5	*Програмна реалізація системи*	*5.04.2022-15.04.2022*	
6	*Оформлення пояснювальної*	*15.04.2022-*	
7	*Захист програмного продукту*	*25.04.2022*	
8	*Передзахист*	*23.05.2022*	
9	*Захист*	*14.06.2022*	

Студент-дипломник _____ Денис САВІЧЕВ
(підпис)

Керівник проєкту _____ Валентина ТКАЧЕНКО
(підпис)

АНОТАЦІЯ

У даній роботі було детально розглянуто найвідоміші системи керування чергами для супермаркетів а також проведений детальний аналіз ринку цих систем.Були проаналізовані їхні слабкі та сильні сторони. В результаті роботи було розроблено систему керування чергами, що вирішує проблему скупчення людей в супермаркетах та інших людних місцях, що в свою чергу вирішує питання оптимізації для бізнесу та загальної гігієни. Програмний продукт був розроблений на мові Python.

Ключові слова: керування чергами, веб-додаток, нейронна мережа, Python.

ANNOTATION

This paper examines in detail the most well-known steering wheel control systems for supermarkets, as well as a detailed analysis of the market for these systems. Their weaknesses and strengths were analyzed. As a result of the work, a queue management system was developed, which solves the problem of crowding in supermarkets and other crowded places, which in turn solves the issue of business and general hygiene. The software product is developed in Python.

Keywords: queue management, web application, neural network, Python

Позначення	Найменування	Кіл-ть	№ екз.	Прим
	Документація загальна			
А4	ІАЛЦ.467200.002 ТЗ	Система керування чергами в	3	
		глобальної пошукової оптимізації		
		технічне завдання		
А4	ІАЛЦ.467200.003 ПЗ	Система керування чергами в	61	
		Оптимізації		
		Пояснювальна записка		
А4	ІАЛЦ.467200.004 Д1	Система керування чергами в	1	
		глобальної пошукової оптимізації		
		Структурна схема		
		Системи керування чергами для		
А4	ІАЛЦ.4672008.005 Д2	Система керування чергами в	1	
		глобальної пошукової		
		оптимізації		
		Функціональна схема		
		(Схема сховища даних)		
А4	ІАЛЦ.467200.006 Д3	Система керування чергами в	1	
		супермаркетах		
		Принципова схема авторизації		

					ІАЛЦ.467800.001 ВП
А		№ докум.	Підп	Д	
Розроб.		Савічев Д.А.			Відомість проекту
Перевір.		Ткаченко В.В.			
Н. Контр.		Симоненко В.П.			
атверд.		Стіренко			

Літ.	Арк.	Акрушів
	1	1

НТУУ КПІ ім. Ігоря Сікорського, ФІОТ,

ТЕХНІЧНЕ ЗАВДАННЯ
ДО ДИПЛОМНОГО ПРОЄКТУ

на тему: «*Система керування чергами в супермаркетах*»

Київ – 2022

ЗМІСТ

					ІАЛЦ.467200.002 ТЗ		
Зм.	Арк.	№ докум.	Підпис	Дата			
Розробив		Савічев Д.А.			**Еволюційні алгоритми глобальної пошукової оптимізації** Технічне завдання	Літ.	Аркуш / Аркушів
Перевірив		Ткаченко В.В.					
Н. Контр.		Симоненко В.П.				НТУУ КПІ ім. Ігоря Сікорського, ФІОТ ІВ-81	
атверд		Стіренко					

1 НАЙМЕНУВАННЯ ТА ОБЛАСТЬ ЗАСТОСУВАННЯ

Дане технічне завдання поширюється на розробку системи керуванння чергами в супермаркетах та аналізу виникнення інцидентів, пов’язаних з натовпом людей.

Областю застосування цієї системи є проектування акустичного обладнання, аналіз даних, економічне та фінансове прогнозування, оцінка та управління екологічним ризиком, дизайн промислового продукту, оптимізація в числовій математиці, оптимальна робота "закритих" (конфіденційних) інженерних або інших систем, для яких поставлена задача глобальної оптимізації.

2 ПІДСТАВИ ДЛЯ РОЗРОБКИ

Підставою для розробки даної системи є завдання для виконання роботи кваліфікаційно-освітнього рівня «бакалавр інженерії програмного забезпечення», який був затверджений факультетом “Інформатики та обчислювальної техніки” кафедрою обчислювальної техніки Національного технічного Університету України «Київський Політехнічний інститут ім. Ігоря Сікорського».

3 МЕТА ТА ПРИЗНАЧЕННЯ РОЗРОБКИ

Метою та призначенням даної роботи є розробка системи пошуку глобальних екстремумів багатомірних функцій із використанням еволюційних алгоритмів, що дозволить ефективно вирішувати дану задачу.

4 ДЖЕРЕЛА РОЗРОБКИ

Джерелом розробки даного дипломного проекту є офіційні документації, публікації та статті в мережі Інтернет на дану тему, науково-технічна література

5 ТЕХНІЧНІ ВИМОГИ

5.1. Вимоги до розробленого продукту

Розроблена система має виконувати такі вимоги:

- Простий і інтуїтивно-зрозумілий інтерфейс системи.
- Надати можливість користувачам передавати вхідні дані для налаштувань системи.
- Надати можливість користувачам власноруч конфігурувати специфічні параметри обчислення для деяких алгоритмів.
- Надати можливість користувачам власноруч вибирати бажаний алгоритм для обчислення.
- Надати вичерпну та зрозумілу документацію для розробленого продукту.

5.2. Вимоги до програмного забезпечення

- **ОС Windows, Mac чи Linux.**
- **Visual Studio 2017 IDE версії або вище.**

5.3. Вимоги до апаратної частини

- **ЦП не менше ніж Intel® Core (TM) i3-2100T.**
- **ROM не менше ніж 64 ГБ.**
- **RAM не менше ніж 4 ГБ.**

6 ЕТАПИ РОЗРОБКИ

Назва етапів виконання	Термін виконання
Затвердження теми роботи	10.12.2021-15.12.2021

Вивчення та аналіз завдання	15.12.2022-15.03.2022
Розробка архітектури та загальної структури системи	15.03.2022-25.03.2022
Розробка структур окремих частин системи	25.03.2022-5.04.2022
Програмна реалізація системи	5.04.2022-15.04.2022
Виправлення помилок	15.04.2022-20.05.2022
Оформлення пояснювальної записки	25.04.2022

ПОЯСНЮВАЛЬНА ЗАПИСКА
ДО ДИПЛОМНОГО ПРОЄКТУ

на тему: «*Система керування чергами в супермаркетах*»

Київ – 2022

ЗМІСТ

ВСТУП

В даній дипломній роботі розроблено і описано систему для керування чергнами в супермаркетах та інших закладах. Це особливо є актуальним в контексті пандемії останніх років а також для покращення бізнесу з точки зору покращення досвіду клієнтів. Також проведений аналіз ринку та порівняння існуючих рішень в сфері систем управління чергами для різних випадків.

З розвитком інтернет-технологій все більшого масштабу та популярності набирають веб додатки через свою доступність для пересічного користувача. Система, розробка якої описана в дипломній роботі, буде розроблена саме як веб-додаток для касирів та управління супермаркетів для оптимізації та контролю їх роботи.

ґ

РОЗДІЛ 1. ОГЛЯД ІСНУЮЧИХ РІШЕНЬ ПОБУДОВИ СИСТЕМ КЕРУВАННЯ ЧЕРГАМИ

1.1 Черга, її наслідки та їх вирішення

Черга — це група людей, які стали один за одним для одержання або здійснення чого-небудь. Далі буде розглядатися в контексті черги на касі в супермаркеті. Це являється одночасно непродуктивною витратою часу для людини, а також — зменшенням прибутку для супермаркету. Деякі експерти вважають черги — одною з найбільших проблем сервісу у сфері рітейлу.

1.1.1 Передумови виникнення ідеї

Керування чергами являє собою очевидний і невідворотній крок для оптимізації роботи супермаркетів, магазинів тощо. Понад 40% людей, які стоять у чергах, відчувають негативні емоції. Одна з найчастіших спонтанних реакцій є відмова від покупок, коли відвідувач бачить чергу в 5 чи більше осіб. Так більше третини людей змінюють магазини на нові — де відбуваються врегулювання черг. Очевидно, що це несе колосальні збитки для рітейл сектору.

В наші дні, в умовах пандемії, питання санітарії у людних місцях набуло особливого значення. Великі скупчення людей є неприпустимими для бізнесу і унеможливлює розвиток і існування, як такого. Це створює ще більш міцне підгрунтя для розгляду автоматизації керування чергами і вбудову таких систем в свій бізнес.

ґ

1.1.2 Відомі підходи до вирішення проблеми

Проблема суто полягає у вчасному відкритті нових кас, при переповненні черг вже відкритих, до того, як це стане впливати на відвідувачів.

Принципово існує два підходи: автоматичний та з наглядачем, тобто де існує людина, яка напряму чи в напів автоматичному режимі(за допомогою камер спостереження) управляє персоналом та касами.

1.1.2.1 Ручний підхід

Ручше управління являє собою першу сходинку до автоматизації, так як, фактично, з нього все почалося. Основним недоліком, безумовно, являється людський фактор, але, в свою чергу, це може здатися дешевшим і, безумовно, легшим підходом для вбудови в існуючий бізнес.

1.1.2.2 Автоматизація

Навідміну від ручного управління (в основі якого стоїть людина — управляючий) з автоматичним підходом можна уникнути багатьох проблем та покращити існуючі процеси. Далі будуть розгянуті автоматичні підходи для управління чергами

1.1.3 Розмір і прогноз ринку систем управління чергою

Розмір ринку системи керування чергами зростає помірними темпами із значними темпами зростання протягом останніх кількох років, і, за оцінками, ринок значно зросте в прогнозований період, тобто з 2021 по 2028 рік.

ґ

Розмір ринку глобальної системи управління чергою (QMS) зросте з 0,5 мільярда доларів США в 2020 році до 0,6 мільярда доларів США до 2026 року зі складним річним темпом зростання (CAGR) 4,0% протягом прогнозованого періоду. Такі фактори, як зростаюча потреба в управлінні трафіком клієнтів і переміщенням клієнтів для підвищення продуктивності, а також зростаюча потреба в підвищенні ефективності персоналу та посиленні взаємодії з клієнтами під час COVID-19 сприяють поширенню ринку QMS по всьому світу. СМК можна використовувати в будь-якому місці, де зазнає значного впливу. Більшість організацій віддають перевагу рішенням і послугам QMS, щоб скоротити час очікування клієнтів на місці та скоротити кількість пішоходів. Розмір ринку глобальної системи керування чергою (QMS) зросте з 0,5 мільярда доларів США в 2020 році до 0,6 мільярда доларів США до 2026 року за підсумками Compound Annual Темп зростання (CAGR) 4,0% протягом прогнозного періоду. Такі фактори, як зростаюча потреба в управлінні трафіком клієнтів і переміщенням клієнтів для підвищення продуктивності, а також зростаюча потреба в підвищенні ефективності персоналу та посиленні взаємодії з клієнтами під час COVID-19 сприяють поширенню ринку СУЯ по всьому світу. СМК можна використовувати в будь-якому місці, де зазнає значного впливу. Більшість організацій віддають перевагу рішенням та послугам QMS, щоб скоротити час очікування клієнтів на місці та зменшити кількість прогулянок.

Зростаюча потреба в управлінні трафіком клієнтів і переміщенням клієнтів для підвищення продуктивності, а також зростаюча потреба в підвищенні ефективності роботи персоналу та посиленні взаємодії з клієнтами є основними факторами, які прискорять зростання ринку. Звіт ринку Global Queue Management System надає цілісну оцінку ринку. Звіт пропонує всебічний аналіз ключових сегментів, тенденцій, драйверів, обмежень, конкурентного середовища та факторів, які відіграють істотну роль на ринку.

г

1.1.3.1 Визначення ринку глобальної системи керування чергою

Це інтелектуальна система, яка допомагає керувати чергами людей для більш ефективного обслуговування клієнтів. Більшість роздрібних магазинів і аеропортів використовують цю систему для контролю черг з різних місць і в різних ситуаціях. Крім того, система управління чергою складається з кількох модулів для ефективнішого управління запитами клієнтів і їх обробки. Систему можна застосовувати для структурованих черг, неструктурованих черг, черг на основі терміналів та мобільних черг. Основний принцип систем управління чергою полягає у кількісній оцінці потреб у чергах у будь-який момент часу та інформуванні персоналу в режимі реального часу. Датчики підрахунку людей, розміщені над кожною касою, підраховують кількість клієнтів, яких обслуговують, кількість клієнтів, які чекають на обслуговування, і вимірюють, скільки часу вони чекали.

1.1.3.2 Ринок системи управління чергою, за типом

- Віртуальна черга
- Лінійна черга

На основі типу ринок поділяється на віртуальні черги та лінійні черги. Сегмент системи керування віртуальними чергами очікував значного зростання протягом прогнозованого періоду. Оскільки організації висувають вимоги, щоб уникнути скупчення філій і захистити клієнтів / персонал. У процесі системи віртуальної черги клієнти можуть або зареєструватися, або ідентифікуватися після прибуття, або отримати квиток для зустрічі з постачальником послуг у відповідний час. Багато компаній пропонують рішення для управління віртуальними чергами, які прискорять зростання ринку.

г

1.1.3.3 Динаміка ринку

- Зростаюча потреба в управлінні трафіком клієнтів і рухом клієнтів для підвищення продуктивності. QMS – це автоматизована система, призначена для управління послугами або потоком клієнтів. Він в основному використовується для керування взаємодією з клієнтами, будь то особисто або за допомогою інформації, що відображається на екрані. Ця система допомагає організаціям керувати чергами людей і контролювати їх. Він пропонує ряд модулів, які користувачі можуть використовувати для ефективної обробки запитів клієнтів. QMS дозволяє компанії точно зрозуміти, де є неефективність у їхніх чергах, і вирішувати ці проблеми на постійній основі, щоб забезпечити стабільно позитивний досвід. Переваги QMS можуть поширюватися на весь бізнес від підвищення рівня задоволеності клієнтів до підвищення продуктивності бізнесу. Згідно з опитуванням VersionX, 78% клієнтів погоджуються, що їм подобається контролювати, коли вони хочуть отримати обслуговування.
- Високі витрати на встановлення системи керування чергою. QMS широко використовується в мережі громадського транспорту. Розгортання їх у мережі вимагає спільних зусиль таких організацій, як постачальники інфраструктури, виробники компонентів, постачальники послуг, державний сектор та групи користувачів. Залучення різних організацій для успішного впровадження квиткових систем тягне за собою складну структуру, що призводить до величезних інвестицій. Отже, успішне впровадження QMS вимагає великого фінансування з боку органів влади, що може уповільнити зростання цього ринку. Будуть високі початкові витрати на налаштування та постійні надбавки, пов’язані з QMS, що може перешкодитисdc зростанн цього ринку.

г

- Поява передових технологій, таких як штучний інтелект і аналітика

У бурхливому технологічному світі спостерігається швидке збільшення використання підключених пристроїв через поширення тенденцій штучного інтелекту (ШІ) та аналітики. В індустрії гостинності ці технології дали можливість підвищити безпеку відвідувачів і гостей і зменшити кількість робочої сили, необхідної для підтримки захисних заходів. Зростаюче дотримання нормативних вимог також призводить до зростання попиту на автоматизовані системи керування відвідувачами. Технології штучного інтелекту та аналітики відіграють важливу роль у запровадженні глобального ринку систем керування чергами. Аналітика — це комбінація зберігання й обробки даних датчиків і обчислень із використанням аналітики даних для досягнення та ефективного керування чергами клієнтів. З появою штучного інтелекту та аналітики великі підприємства тепер можуть надавати безпечні рішення QMS з різними ключовими функціями, такими як онлайн-бронювання, інтелектуальна система зворотних дзвінків, самообслуговування входу, відстеження продуктивності, а також сповіщення та сповіщення в режимі реального часу.

1.3.4 Ключові гравці ринку

Ключові гравці ринку Постачальники QMS впровадили різні типи стратегій як органічного, так і неорганічного зростання, таких як запуск нових продуктів, оновлення продуктів, партнерства та угоди, розширення бізнесу, а також злиття та поглинання, щоб посилити свої пропозиції на ринку. Деякі з ключових гравців, що працюють на ринку QMS, включають Advantech (Тайвань), Wavetec (Дубай), Aurionpro (Індія), Lavi Industries (США), QLess (США), Qmatic (Європа), SEDCO (OAE), Q-nomy (США), Core Mobile (США), MaliaTec (Ліван), JRNI (Англія), Qudini (Англія), Qminder (Великобританія),

г

ATT Systems (Індія), XIPHIAS (Індія), AKIS Technologies (Європа), AwebStar (Сінгапур), Xtreme Media (Індія), Skiplino (Бахрейн), Business Automation (Бангладеш), Udentify (Туреччина), 2Meters (Німеччина), OnlineToken (США), Hate2wait (Індія), VersionX (Індія). Дослідження включає поглиблений конкурентний аналіз цих ключових гравців на ринку систем управління чергою з профілями їх компаній, останніми розробками та ключовими ринковими стратегіями.

1.1.3.5 Класифікація ринку QMS на основі компонента, типу, програми, режиму розгортання, розміру організації, вертикалі та регіону.

За компонентами:

1. Рішення
2. Програмне забеспечення
3. Платформа
4. Сервічи
5. Пофесійні сервіси
6. Консалтинг
7. Системи інтеграції та імплементаії
8. Підтримка
9. Сервіси управління

За типом черги:

- Структурована черга
- Не структурована черга
- Kiosk-Based
- Мобільні черги

За типом розгортання:

ґ

1. Локальне програмне забезпечення
2. Хмара

За розміром організації

- Великі підприємства
- Маленькі та середні підприємства (SMEs)

За типом додатку:

- Репорт та аналіз
- Управління призначенням
- Клієнтський сервіс
- Перехоплення черг
- Управління в магазині
- Оптимізація робочої сили
- Моніторинг в реальному часі

По вертикалі:

- Уряд і державний сектор
- Рітейл та споживчі товари
- Охорона здоров'я та науки про життя
- IT та Телеком
- Подорожі та гостинність
- Комунальні послуги
- Інші (Освіта, виробництво, медіа та розваги)

За регіоном:

- Північна Америка
  - Сполучені Штати
  - Канада
- Європа
  - Велика Британія
  - Німеччина

ґ

- Франція
- Азіатсько-Тихоокеанський регіон
- Китай
- Індія
- Японія
- Близький Схід та Північна Африка
- Північна Африка
- UAE
- KSA
- Латинська Америка
- Бразилія
- Мексика

1.1.3.6 Останні події на ринку

- У березні 2021 року SEDCO представляє нову серію кіосків самообслуговування, що забезпечує кращий досвід роботи з клієнтами та більш ефективну роботу з меншими витратами. Багатофункціональний кіоск представлений у двох сериях, FASTSERV та CONSULTA, що дозволяє організаціям вибирати машини на основі запитів, розміру філії та бюджету. Машини самообслуговування можуть бути інтегровані з удосконаленою QMS, щоб забезпечити розумну маршрутизацію, коли клієнти спрямовуються до потрібного каналу обслуговування на основі типу обслуговування, часу обслуговування, демографії клієнтів та наявності персоналу або машин.
- У грудні 2020 року Advantech додала дві нові моделі до своєї лінійки UTC-500 — 15,6-дюймові UTC-515H та 21,5-дюймові сенсорні комп’ютери «все в одному». UTC-515H і UTC-520H ідеально підходять для HMI і систем управління об’єктами на фабриках, кухонних дисплеїв і кіосків

г

самообслуговування. Вони забезпечують інтегровані програми роздрібної торгівлі, гостинності та державних послуг.

- У грудні 2020 року Qmatic та лабораторія Каролінського університету підписали угоду про впровадження Qmatic Mobile Ticket у 47 своїх лабораторних підрозділах по всій Швеції. Впровадження мобільного рішення Qmatic для управління чергою в цифровому вигляді змінить досвід пацієнтів і забезпечить захисні заходи для запобігання поширенню COVID-19 у його лабораторних підрозділах.
- У вересні 2020 року Qmatic оголосила про нову пропозицію пакетів для охорони здоров'я з випуском останнього та покращеного модуля проміжного програмного забезпечення Qmatic Level Seven (HL7). Нова медична пропозиція від Qmatic підтримує три найпоширеніші потоки пацієнтів у сфері охорони здоров'я, перед прибуттям, прибуттям та обслуговуванням, що дозволяє постачальникам медичних послуг керувати прийомами, відвідуваннями та пацієнтами невідкладної допомоги від прибуття до виписки.
- У квітні 2019 року Aurionpro підписав угоду з одним із провідних банків державного сектору Індії щодо створення кіосків із самообслуговуванням. Компанія виграла замовлення від одного з провідних банків державного сектору Індії на створення кіосків із самообслуговуванням у різних місцях по всій Індії. Передбачалося, що Aurionpro постачає, встановлює та підтримує KIOSKS з самообслуговуванням у більш ніж 500 відділеннях банку по всій Індії.

1.1.4 Актуальність

Очікувалося, що ринок систем управління чергами у 2020 році буде свідком незначного сповільнення через глобальний карантин. Пандемія

ґ

COVID-19 підвищила рівень відтоку та вразила майже кожну галузь. Блокування впливає на світове виробництво, ланцюги поставок і логістику, оскільки безперервність операцій у різних секторах сильно впливає. Секторами, які стикаються з найбільшими недоліками, є виробництво, транспорт і логістика, а також роздрібна торгівля та споживчі товари. На доступність основних речей вплинула нестача робочої сили для роботи на виробничих лініях, ланцюгах поставок і транспортуванні, хоча предмети першої необхідності звільнені від блокування. Очікується, що цей стан вийде під контроль на початку 2022 року, тоді як попит на рішення та послуги QMS, як очікується, зросте, що пов’язано зі збільшенням попиту на покращений досвід роботи з клієнтами та побудову персоналізованих відносин із потенційними клієнтами. Кілька вертикалей вже планують розгорнути різноманітний набір рішень і сервісів QMS, щоб забезпечити ініціативи цифрової трансформації, які стосуються критично важливих процесів, покращують операції та автентифікують ідентифікацію користувача. Зменшення операційних витрат, кращий досвід роботи з клієнтами, виявлення та запобігання шахрайству, покращені процеси та операції аутентифікації та покращене прийняття рішень у режимі реального часу є ключовими бізнесовими та операційними пріоритетами, які, як очікується, сприятимуть запровадженню ринку систем керування чергами.

1.2 Огляд існуючих систем керування чергами

Усі галузі сфери послуг завжди шукають способи покращити якість обслуговування клієнтів та шляхи клієнтів у своїх компаніях, офісах чи

ґ

центрах обслуговування клієнтів. Задоволені клієнти означають хорошу репутацію бренду та більшу утримання клієнтів, що зрештою призводить до сталого зростання бізнесу. Найкращий спосіб – оптимізувати потік клієнтів за допомогою цифрової системи управління чергою. У Дубаї, ОАЕ, якщо бренд фізично взаємодіє зі своїми клієнтами та відвідувачами, система управління чергою є обов'язковою. Незалежно від того, наскільки малий чи великий бізнес, якщо вони взаємодіють із клієнтами та відвідувачами у своїх приміщеннях, у наші дні наявність системи управління чергою є нормою. Клієнти та відвідувачі також очікують системи управління чергою, а не довгих черг, що контролюються персоналом вручну. Системи управління чергою будуються як для певних сценаріїв, так і загального призначення, а технологія, що використовується для побудови систем масового обслуговування, також широкодоступна, що призводить до великої кількості нововведень та типів системи управління чергою. Це іноді ускладнює для деяких підприємств вибір найефективнішої системи управління чергою, яка може задовольнити їхні потреби.

1.2.1 Онлайн запис

Система онлайн-запису приймання дуже поширена в секторах охорони здоров'я. Багато інших галузей також використовують систему онлайн-запису приймання. Система управління чергою може бути або інтегрована з існуючою онлайн-системою запису на прийом або системою управління, або модуль запису приймання також може поставлятися в комплекті з системою управління чергою. Система запису на прийом може надаватися або через програму для смартфона, або через онлайн-портал. В основному використовуються обидва методи. Клієнт може отримати доступ до панелі керування записом на прийом із входом або без входу на клієнтський портал, додаток для смартфона або навіть через веб-сайт. Клієнт може вибрати

г

конкретний день і час на свій вибір, система може надати йому очікуваний час надання послуги або кількість активних черг у цей конкретний час. Також може бути надано багато інших функцій для подальшого покращення якості обслуговування клієнтів. Онлайн-запис на прийом із системою управління чергою не лише допомагає скоротити час очікування клієнта та час надання послуги, але також знижує навантаження на персонал та дає їм найкращий контроль над кількістю клієнтів або відвідувачів у будь-який момент часу. Система онлайн-запису на прийом також допомагає організаціям збирати цінні дані та іншу системну статистику, а також відгуки клієнтів. Ці дані дуже допомагають бізнесу приймати життєво важливі рішення та розробляти майбутні стратегії.

1.2.2 Лінійна черга

Лінійна система управління чергою або лінійна система управління чергою - це найпростіший тип управління потоком клієнтів. Це тип організації черг у порядку живої черги. Ідентифікуються найстаріші та найчастіше використовувані методи обробки черг. Він складається з однієї лінії та одного сервера. Хоча сервери могли бути і більшими, і лінії могли бути аналітичними. Зазвичай такий вид чергового центру використовується для максимізації пропускної спроможності обслуговування клієнтів. Підхід «першим надійшов — першим обслужений» або FIFO найкраще підходить для невеликих послуг, що базуються на час доставки або швидших послуг. В основному невеликі сервісні центри з привабливими клієнтами користуються лінійною організацією черг, що максимізують ефективність системи та статистики. Лінійна організація черг - це найпростіший тип систем управління чергами, який також потребує меншої кількості системних компонентів та меншої складності. Декілька ліній були розділені на основі необхідної послуги, система управління дозволяє швидко вибирати клієнтські послуги, які він хоче

ґ

вибирати. Найчастіше в лінійній організації черговості, реалізованої у сценаріях, де доступно кілька послуг, може бути створено кілька пропорцій ліній, клієнтів і відвідувачів, які можна розділити з урахуванням послуг, у яких користуються послугами. Наприклад, всі клієнти, які скористалися послугою з вищим часом доставки, повинні бути поставлені в чергу на перевищення лічильника.

1.2.3 Нелінійна черга

Здебільшого у сфері послуг виявляється нелінійна організація черг. Зазвичай, коли у вас є більше 4 лічильників та/або кілька сервісів для обслуговування, нелінійна організація черг виявляється набагато кращою, ніж лінійна організація черг. Наприклад, у нелінійній черзі; є два лічильники для послуг А та чотири лічильники для послуг В. Послуга А вимагає близько 10 хвилин, а послуга В вимагає близько 20 хвилин для надання однієї послуги. Система управління автоматично направляє клієнтів до необхідної стійки незалежного обслуговування від часу їх виділення, якщо потрібна стійка вільна, вони прямують до цієї стійки без затримки. Ось як працює нелінійна організація звинувачень, звинувачень у змінах та умовах, які викликаються у слухачів, щоб визначити, звідки вони отримують послугу та скільки часу їм необхідно, щоб отримати послугу. Усі типи черг підпадають під визначення нелінійних черг, у яких не дотримується правило «першим прийшов – першим обслужений». Наприклад, ексклюзивні лічильники для клієнтів VIP або Premium є прикладами нелінійної черги. Іншим дуже бажано залучити центри підтримки та обслуговування клієнтів телекомунікацій, які зазвичай оплачуються різними послугами, та майже всі стійки також доступні для всіх видів послуг. Тим не менш, деякі послуги надаються швидше, ніж інші, це залежить від політики компанії, наприклад, ми можемо сказати, що компанія

ґ

хоче віддавати пріоритет новим лідам та потенційним клієнтам, які вирішують придбати новий план чи послугу тощо. Усі ці типи черг є нелінійними чергами.

1.2.4 Черга з кількома лічильниками

Черга з кількома лічильниками є дещо складнішою, ніж інше лінійне та нелінійне керування потоками клієнтів. У черзі з кількома прилавками клієнту доведеться звернутися до кількох або більше ніж однієї стійки, щоб отримати повну послугу. У черзі з кількома лічильниками кількома чергами потрібно керувати паралельно, щоб забезпечити менший час очікування. Наприклад, візові та імміграційні послуги, клініка, банк чи щось інше, існує дуже багато різних застосувань черги з кількома прилавками. У черзі з кількома прилавками клієнт реєструється в черзі та отримує номер квитка, коли настає їхня черга, вони приходять до першого лічильника, де виконується більшість даних, або ініціюється послуга. Тоді їм або присвоюють новий номер квитка, або той самий буде повторно використаний для наступної зупинки. Клієнта можна направити до наступної стійки за допомогою різних вказівників і покажчиків напрямку. Потім клієнт підійшов до другої стійки, дані та/або деталі клієнта можуть бути автоматично перенесені на наступну стійку після їх прибуття. Це заощаджує багато часу та покращує якість обслуговування клієнтів. Або повний обсяг послуг надається на другому прилавку, або клієнт може бути направлений до наступного прилавка тощо. Ось як працює багатостійкова черга.

1.3 Проблеми проектування систем контролю чергами

ґ

Наступні проблеми будуть розглянуть з позиції моделювання системи контролю чергами для супермаркетів з використанням IOT (Internet of things) «розумних» камер.

1.3.1 Одночасне управління багатьма касами у великих супермаркетах

При встановленні камер над касами для подальшої відправки фото по HTTP на сервер для обробки цих фотографій, виникає проблема у відслідковувванні джерала конкретного фото. Для вирішення проблеми має бути відповідний дизайн системи, а саме — введення сутності «локації» камери.

1.3.2 Розпізнавання кількості людей, не включаючи співробітників

Нейронна мережа, яка буде розпізнавати кількість людей в черзі має «знати» в якому регіоні фотографії знаходяться покупці, а в якомі співробітники (касири). Для вирішення проблеми слід ввести додаткову логіку, а саме — виділення регіонів черг і касирів для кожної локації.

г

ВИСНОВОК ДО РОЗДІЛУ 1

В даному розділі розглянута проблематика виникнення черг та їх вплив в сфері рітейлу. Проаналізовано актуальність QMS, також в контексті пандемії. Описані основні тенденції та прогнози ринку QMS. Приведений розгорнутий порівняльний аналіз типів QMS, також проблематика при проєктуванні таких систем та усунення цих недоліків.

г

РОЗДІЛ 2. ОГЛЯД ТЕХНОЛОГІЙ ДЛЯ ПРОГРАМУВАННЯ QMS

2.1 Архітектура додатку

Складність корпоративних програмних систем зростає зі збільшенням підтримувані функції та можливості. Як ви бачили у розділі 1, кожна корпоративна система сьогодні необхідно без перешкод передавати інформацію з багатьма іншими системами, як внутрішніми, та зовнішні стосовно організації. Традиційно ми розробляємо програмні системи як "модульний моноліт". Під модульністю мається на увазі, що вони слідують принципам модулів, шарів та рівнів і, отже, існує логічна модульність для елементів усередині програмної системи. Під монолітом ми маємо на увазі, що вся система побудована з урахуванням конкретного сценарію розгортання та експлуатації; більшу частину часу вони розгортаються як єдиний процес або, як максимум, розподіляються слідом за трирівневої чи n-рівневої архітектури. Але на цьому гнучкість розгортання. Не менш важливим є процес, який необхідно слідувати при отриманні такої системи. побудована у багатокомандній чи розподіленій командній організації. У цьому розділі ви розглянете кілька вибраних проблем, із якими стикаються підприємства під час будівництва. сучасні програмні програми.

2.1.1 Мікросервісна архітектура

Архітектура мікросервісів призначена для того, щоб допомогти розробникам не допустити, щоб їхні додатки виросли громіздкими, монолітними та негнучкими. Замість того, щоб створювати одну велику програму, ціль полягає в тому, щоб створити кілька різних крихітних програм,

г

а потім створювати нову маленьку програму щоразу, коли хтось хоче додати нову функцію.

"Якщо ви зайдете на свій iPad і подивіться на інтерфейс користувача Netflix, кожен окремий об'єкт в цьому інтерфейсі відключить окремий сервіс", - Іран Річардс. Список ваших обраних, оцінки, які ви даєте частотним характеристикам, та бухгалтерська інформація – все це доставляється окремими пакетами специфічних служб. Як Netflix це набір з безлічі веб-сайтів, які просто користуються одним сервісом.

Цей підхід схожий на підходи, що керуються подіями, і підходи мікроядра, але він використовується в основному, коли різні завдання легко поділити. У багатьох випадках використовуються різні завдання, пов'язані з різною ретельністю обробки та підбором збірок у зборі. Сервери, що доставляють контент Netflix, завантажуються дуже інтенсивно вечорами у п'ятницю та суботу, тому вони мають бути готові до масштабування. З іншого боку, сервери, які відстежують повернення DVD, виконують більшу частину своєї роботи протягом тижня, одразу після того, як пошта доставляє денну пошту. Реалізуючи їх як певні послуги, хмара Netflix може масштабувати їх незалежно один від одного в міру зміни споживання.

Застереження:

- Сервіси мають бути в умовах обмеженої свободи, інакше взаємодія може призвести до дисбалансу хмар.
- Не всі програми мають завдання, які не можна легко розділити на незалежні блоки. Продуктивність може постраждати, якщо вони виявлені серед частинок мікросервісами. Витрати на зв'язок можуть бути великими.
- Занадто багато мікросервісів може збити користувачів з пантелику, тому що частини веб-сторінок тисячі пізніше за інших.

Підходить для:

- Сайти з корисними компонентами

ґ

- Корпоративні центри обробки даних із чітким відображенням кордонів
- Нові підприємства та веб-додатки, що швидко розвиваються
- Команди розвиваються розосереджені, часто по всьому світу.

2.1.2 Багатошарова (n-рівнева) архітектура

Цей підхід, ймовірно, є найбільш поширеним, оскільки він зазвичай будується навколо бази даних, а багато бізнес-додатків природно піддаються зберіганню інформації в таблицях. Це щось на зразок самовиконуваного пророцтва. Багато з найбільших і кращих програмних середовищ, таких як Java EE, Drupal і Express, були створені з урахуванням цієї структури, тому багато програм, створених з їх допомогою, природно мають багаторівневу архітектуру. Код влаштований так, що дані надходять на верхній рівень і просуваються вниз по кожному рівню, доки досягнуть нижнього, яким зазвичай є база даних. Принагідно кожен шар має певне завдання, наприклад, перевірка даних на узгодженість або переформатування значень для забезпечення їх узгодженості. Зазвичай, різні програмісти працюють незалежно один від одного на різних шарах.

Структура Model-View-Controller (MVC), яка є стандартним підходом до розробки програмного забезпечення, що пропонує більшість популярних веб-фреймворків, явно є багаторівневою архітектурою. Безпосередньо над базою даних знаходиться рівень моделі, який часто містить бізнес-логіку та інформацію про типи даних бази даних. Вгорі знаходиться шар уявлення, який часто є CSS, JavaScript і HTML з динамічним вбудованим кодом. У середині у вас є контролер, який має різні правила та методи для перетворення даних, що переміщуються між поданням та моделлю.

ґ

Перевага багаторівневої архітектури полягає у розподілі завдань, що означає, що кожен рівень може зосередитися виключно на своїй ролі. Це робить ці рівні:

- Легкими для підтримки
- Легкими для тестування
- Легко призначати окремі «ролі»
- Легко оновлювати та покращувати шари окремо

Належні багаторівневі архітектури будуть мати ізольовані шари, на які не впливають певні зміни в інших шарах, що спрощує рефакторинг. Ця архітектура також може містити додаткові відкриті рівні, такі як рівень обслуговування, які можна використовувати для доступу до загальних служб тільки на бізнес-рівні, але які також можна обходити для підвищення швидкості. Поділ завдань та визначення окремих шарів – найбільша проблема для архітектора. Коли вимоги добре відповідають шаблону, шари легко розділити і призначити різним програмістам.

Застереження:

- Вихідний код може перетворитися на «велику грудку бруду», якщо він неорганізований, а модулі не мають чітких ролей або взаємозв'язків.
- Код може виявитися повільним через те, що деякі розробники називають антишаблоном вирви. Більшість коду може бути присвячена передачі даних через шари без використання будь-якої логіки.
- Ізоляція рівнів, яка є важливою метою архітектури, також може ускладнити розуміння архітектури без розуміння кожного модуля. Кодувальники можуть пропустити минулі рівні, щоб створити тісний зв'язок та створити логічну плутанину, повну складних взаємозалежностей.
- Монолітне розгортання часто неминуче, а це означає, що невеликі зміни можуть вимагати повного повторного розгортання програми.

Підходить для:

ґ

3. Нові програми, які потрібно створювати швидко

4. Корпоративні або бізнес-програми, які повинні відображати традиційні IT-відділи та процеси.

5. Команди з недосвідченими розробниками, які ще не знаються на інших архітектурах

6. Програми, що вимагають строгих стандартів ремонтопридатності та тестування

2.1.3 Архітектура на основі простору

Багато веб-сайтів побудовані навколо бази даних, і вони добре функціонують, поки база даних здатна справлятися з навантаженням. Але коли використання досягає піку, і база даних не справляється із постійним завданням ведення журналу транзакцій, весь веб-сайт виходить з ладу. Космічна архітектура призначена для запобігання функціональному колапсу при високому навантаженні за рахунок поділу обробки та зберігання даних між кількома серверами. Дані розподіляються на вузлах так само, як і відповідальність за обслуговування викликів. Деякі архітектори використовують більш розпливчастий термін "хмарна архітектура". Назва "на основі простору" відноситься до "простору кортежів" користувачів, який розділений для поділу роботи між вузлами. «Це всі об'єкти у пам'яті, – каже Річардс. «Архітектура на основі простору підтримує речі, які мають непередбачувані сплески за рахунок усунення бази даних». Зберігання інформації у ОЗУ значно прискорює виконання багатьох завдань, а розподіл пам'яті з обробкою може спростити багато основних завдань. Але розподілена архітектура може зробити деякі види аналізу складнішими. Обчислення, які мають бути розподілені по всьому набору даних, наприклад, пошук середнього значення або виконання статистичного аналізу, повинні бути

г

поділені на підзадачі, розподілені по всіх вузлах, а потім об'єднані, коли це буде зроблено.

Застереження:

10. Транзакційна підтримка складніша з базами даних ОЗУ.

11. Створення достатнього навантаження для тестування системи може бути складним завданням, але окремі вузли можна тестувати незалежно.

12. Розвиток досвіду кешування даних для прискорення без пошкодження кількох копій є складним завданням.

Підходить для:

- Великі обсяги даних, такі як потоки кліків та журнали користувачів.
- Недоцільні дані, які можуть іноді губитися без серйозних наслідків.
- Соціальні мережії

2.1.4 Подійно-орієнтована архітектура

Подійно-орієнтована архітектура Багато програм проводять більшу частину свого часу в очікуванні, поки щось станеться. Це особливо характерно для комп'ютерів, які працюють безпосередньо з людьми, але також часто зустрічаються в таких областях, як мережі. Іноді є дані, які потрібно обробити, інколи ж їх немає.

Подійно-керована архітектура допомагає впоратися з цим шляхом створення центрального блоку, який приймає всі дані, а потім делегує їх окремим модулям, які обробляють дані певного типу. Ця передача, як то кажуть, генерує "подію", і вона делегується коду, призначеному для цього типу.

ґ

Програмування веб-сторінки за допомогою JavaScript (наприклад) включає написання невеликих модулів, що реагують на такі події, як клацання миші або натискання клавіш. Браузер сам організує все введення і слідкує за тим, щоб тільки потрібний код бачив потрібні події. У браузері відбувається безліч різних типів подій, але модулі взаємодіють лише з тими подіями, що їх стосуються. Це дуже відрізняється від багаторівневої архітектури, де всі дані зазвичай проходять через усі рівні. Загалом архітектури, керовані подіями:

- Легко адаптуються до складних, часто хаотичних середовищ легко масштабуються
- Легко масштбувати
- Легко доповнювати появою нових типів подій

Застереження:

- Тестування може бути складним, якщо модулі можуть впливати один на одного. Хоча окремі модулі можна тестувати незалежно, взаємодія між ними може бути перевірена тільки в системі, що повністю функціонує.
- Обробка помилок може бути важко структурована, особливо коли кілька модулів повинні обробляти ті самі події.
- Коли модулі виходять із ладу, центральний блок повинен мати план резервного копіювання.
- Накладні витрати на обмін повідомленнями можуть уповільнити швидкість обробки, особливо коли центральний блок повинен буферизувати повідомлення серії.
- Розробка загальносистемної структури даних для подій може бути складною, якщо події мають різні потреби.
- Підтримувати механізм узгодженості на основі транзакцій складно, оскільки модулі настільки роз'єднані та незалежні.

Найкраще підходить для:

ґ

- Асинхронні системи з асинхронним потоком даних Додатки, в яких окремі блоки даних взаємодіють лише з деякими з багатьох модулів
- Інтерфейси користувача

2.2 Вибір мови програмування для розробки на стороні клієнт

Розробка на стороні клієнта - це тип розробки, в якому присутні програми, що працюють на клієнті або враження користувачів. Розробники розробили частини програми, зосереджені на створенні частини веб-сайту, яка може взаємодіяти з користувачем. Іноді складання компонентів для клієнта також має на увазі появу переднього плану, оскільки вона фокусується на «лицьовій» частині програм, які можуть бути змінені користувачами. Розробники на боці супротивника грають безліч завдань, у тому числі:

- Створення макетів сайту
- Розробка миттєвих інтерфейсів
- Додавання форми перевірки
- Додавання елементів, таких як кольори та візуальні шрифти

Люди, які займаються розробкою професій, як веб-дизайн та дизайн взаємодії з користувачем, зосереджуються на розробці за клієнта. Розробники виходячи з закону пред'являють скарги на мову програмування. Деякі поширені клієнтські мови включають:

- HTML: HTML, що означає мову гіпертекстової розмітки, є мовою розмітки, яка є предметом вивчення для веб-розробок. HTML створює структуру веб-сайту та відображає веб-сайт у браузері. CSS: CSS, який включає каскадні таблиці стилів, є мовою дизайну, яку розробники створили для додавання елементів реалізації дизайну на веб-сайт, закодований в HTML. Розробники можуть використовувати CSS, щоб їх веб-сайти привертали увагу

г

користувачів. JavaScript: JavaScript — це мова сценаріїв, яку розробники використовують для веб-розробок, веб-застосунків та інших цілей. Розробники можуть використовувати JavaScript, щоб зробити веб-сайти динамічними та інтерактивними. VBScript: VBScript — це клієнтська мова сценаріїв загального призначення, яку підтримують браузери. Розробники можуть використовувати VBScript для додавання інтерактивних елементів на веб-сайти.

Також, для ефективного застосування CSS та маніпулювання DOM деревом застосовують різні фреймворки, написані на javascript. Далі буде наведено порівняння фреймворків для користувацької частини та їх порівняння.

2.2.1 Lit

Lit – це середовище Javascript, побудоване на основі стандартів веб-компонентів. Подібно до Vue та Mithril, Lit прагне стати легкою структурою з невеликими накладними витратами та простою функціональною сумісністю. Google рекомендує використовувати Lit замість Polymer, який зараз у режимі обслуговування.

Найкращі функції:

- Компоненти Lit є також нативними веб-компонентами. Ви можете використовувати їх будь-де, навіть з іншими фреймворками.
- При розмірі 5 КБ Lit є найменшим фреймворком у списку.
- Lit не використовує DOM для оновлення компонентів, тому оновлення сторінок відбуваються дуже швидко.

Що слід враховувати:

- Lit легкий, але його основна увага надається компонентам. Вам знадобиться додатковий код для повної програми.

г

2.2.2 Mithril

Mithril – це легке середовище JavaScript, яке може покращити продуктивність файлу, швидкий рендеринг та використання API. Офісні веб-сайти мають значення і сильне документування, яке ви будете робити і бігати протягом часу.

Найкращі функції :

- Є розміром файлу >10kb (стислий), слід Mithril однією з частин своєї штрихової сторінки завантаження та часу рендерингу.
- Mithril має кілька одних найкращих прикладів із документації для початківців для швидкого старту.

Що слід враховувати:

- Одним із недоліків Mithril є реалізація таких функцій, як JSX, ESX, завантаження файлів або автентифікація. Ви потребуєте додаткових бібліотек або ж самі пишете код.

2.2.3 Aurelia

Aurelia – це платформа програм, заснована на стандартах. Він дотримується узгодження за шаблоном проектування. Більшість програм можуть використовувати згенерований код Aurelia без будь-яких змін, але якщо вам потрібно перевизначити значення за замовчуванням, Aurelia більш ніж рада «піти з вашого шляху».

Найкращі функції:

- Підхід Aurelia до стандартизації означає, що вона має неглибоку криву навчання. Ви виявите, що дуже легко налаштувати та запустити програму.

ґ

- Ви можете написати свої компоненти універсальною мовою JavaScript або Typescript та підключити їх до Aurelia.
- Екосистема Aurelia пропонує прототипи для всіх основних платформ веб-розробки.

Що слід враховувати:

- Aurelia до посилення та її ненав'язливого характеру полегшують інтеграцію з існуючим кодом.

2.2.4 Backbone

Backbone існує вже давно, але він все ще перебуває на стадії стабільної та регулярної розробки. Це хороший вибір, якщо вам потрібне гнучке середовище JavaScript з простою моделлю для представлення даних та отримання їх в уявленнях.

Найкращі функції:

- Backbone — один із найзріліших фреймворків. Він стабільний і надійний.
- Це ще один легкий фреймворк із мінімальним розміром файлу. Ви можете підключити свій механізм шаблонів або використовувати Underscore, одну із залежностей Backbone.

Що слід враховувати:

- З фреймворком легко розібратись. Це фреймворк, який дозволяє створювати програми так, як ви хочете, за допомогою основного набору інструментів для роботи.
- Модель Backbone для інтеграції з RESTful API надзвичайно проста у роботі.

2.2.5 React

г

Розробники React називають це бібліотекою. Хоча інші наполягають на тому, що технічно це фреймворк, назва дійсно відображає естетику дизайну React. Автори створили його, щоб дати вам більше свободи в тому, як ви структуруєте свою програму порівняно з його конкурентами.

Найкращі функції:

- React Native забезпечує простий спосіб перенесення вашої веб-програми на мобільні та настільні платформи.
- JSX React спрощує визначення та структурування ваших HTML-елементів. Широке використання та широке впровадження React полегшує зростання вашої команди.

Що слід враховувати:

- Meta підтримує React, і деякі користувачі стурбовані їхніми умовами використання або подальшою підтримкою бібліотеки.
- Сильне співтовариство розробників означає, що легко знайти такі компоненти, як Material-UI, React Bootstrap та React Router, які допомагають створювати найкращу програму.

Враховуючи всі особливості вищезгаданий фреймоврків, було обрано React, через вилику кількість бібліотек а тевлику спільноту, що безумовно сприяє розробці.

2.3 Вибір мови програмування для серверної частини

З міркування простоти розробки мова програмування, яка має підтримувати парадигму ООП та бути достатньо гнучкою для відносно

ґ

простого розширення функіоналу. Також, основним критерієм для вибору має бути зручність у взаємодії з клієнтом та веб-розробці вцілому.

2.3.1 Go

Go - статично типізована компілювана мова програмування. Його доменне ім'я – golang.org, тому деякі люди називають його Golang.

Мова вперше з'явилася у 2009 році і була розроблена Google. Роберт Гріземр, Роб Пайк і Кен Томпсон об'єдналися у своїй ворожості до C++ і спробували розробити мову, яка включає корисні властивості інших мов, але не має їх недоліків.

Go поєднує у собі переваги таких мов, як C (статична типізація та час виконання), Python та JavaScript (нескладний синтаксис). За даними опитування розробників StackOverflow, за 12 років Go перевершив за популярністю Ruby та Kotlin і став однією з найулюбленіших і найпопулярніших мов у спільноті розробників.

Синтаксис Go дуже простий, особливо якщо у вас є досвід роботи з C або Java. Розробники мови з Google, де багатьом розробникам доводилося працювати над одним і тим самим кодом. Тому вони вирішили створити якомога більш простий і легкий синтаксис, щоб інші розробники теж могли зрозуміти код. Фрагменти коду також досить незалежні, а це означає, що якщо є зміни в одному розділі, інший розділ не торкнеться.

Golang - швидка мова, тому що він компілюється у свій код і має статичну типізацію. Процес компіляції також швидкий. Розробники стверджують, що він швидше за Python і так само швидкий, як C++.

Багато хто з тих, хто стверджує, що Go не підходить для веб-розробки, кажуть, що мова не спирається на будь-які великі фреймворки. Насправді, це не проблема. У Go чудова стандартна бібліотека: чиста, послідовна та проста

ґ

у використанні. Більше того, вам не потрібно проводити великі дослідження, щоб знайти потрібний фреймворк.

Пакет Net/http є важливим для веб-розробки, і ви можете отримати його зі стандартної бібліотеки Go. Go пропонує 3 http-запити: get, post та head. Створивши http. Client та передаючи значення Timeout, ви можете створювати додаткові параметри та запити.

Go ділить великі програми на маленькі завдання і дозволяє цим завданням виконуватись одночасно через горутині та канали. Отже, комп'ютер виконуватиме завдання набагато швидше і менше часу простоювати. Таким чином, Golang/Go легко вивчати та читати, він є паралельним завдяки горутинам і каналам, він швидше, ніж багато інших популярних мов, має приголомшливу стандартну бібліотеку та пропонує пакет net/http для веб-розробки.

2.3.2 PHP

PHP - дуже популярна мова програмування, призначена для розробки веб-сайтів. Більше 20 мільйонів веб-сайтів створено за допомогою PHP. Є багато мов, які використовуються для веб-розробки або програмування. Але серед усіх них PHP є найбільш вивченою веб-скриптів. Він призначений насамперед для веб-розробки, але також використовується як мова програмування загального призначення.

Код PHP може бути вбудований у розмітку HTML або HTML5. Ця технологія дозволяє використовувати набагато більше функцій, ніж HTML, наприклад таблицю DIV, функції входу в систему та графічні відображення. Спочатку він був відомий як персональна домашня сторінка, яка використовує цільову мову. PHP простий у освоєнні і дозволяє користувачам зробити так, щоб веб-сторінка виглядала і поводився саме так, як вони хочуть.

г

Він має як мову процедурного програмування, і функції мови ООП (об'єктно-орієнтоване програмування). Це означає, що програмісти різних мов програмують цю мову протягом короткого періоду часу. Найбільш схожий на синтаксис мов C та C++.

Відкритий код: він розробив великий планшет. Це допоможе у створенні спільноти підтримки та розповсюдження розширеної бібліотеки. Швидкість: Щодо швидко, тому що не використовує багато системних ресурсів. Простота збору: він використовує синтаксис, перевірку C, тому для тих, хто знайомиться з C, їм дуже легко підібрати і легко створити сценарії веб-сайту. Стабільний: оскільки він чудовий для багатьох розробників, помилки виявляються і виправляються досить швидко, що робить його стабільним програмним забезпеченням. Потужна підтримка бібліотеки: ви можете легко знайти потрібні вам функціональні модулі, такі як PDF, графіки і т.д.

Вбудовані модулі підключення до бази даних: ви можете легко підключитися до бази даних за допомогою PHP, оскільки багато веб-сайтів керуються даними/контентом, тому ми часто використовуватимемо базу даних, що значно скоротить час розробки веб-додатків.

Його можна використовувати на багатьох платформах, включаючи Windows, Linux та Mac. тому користувачі легко знаходять послуги хостингу. MySQL використовується з PHP як внутрішній інструмент.

Популярна онлайн-база даних дуже добре взаємодіє із PHP. Так що це був чудовий вибір для веб-майстрів. Він має потужну буферизацію виводу. Він може внутрішньо перебудувати буфер так, щоб заголовки розташовувалися перед вмістом. Він динамічний і працює в акустиці HTML для відображення елементів на сторінках. Він може працювати на всіх веб-браузерах (наприклад, Apache, веб-сервер, Microsoft IIS, Netscape, iPlanet) і всіх базах даних (наприклад, MySQL, dBase, IBM DB2, ODBC, PostgreSQL, Inter Base, Front Base, SQLite). PHP5 - це повністю об'єктно-орієнтована мова, яку можна

г

використовувати практично скрізь. Його документація чудова. PHP має вибір відповідних CMS, таких як Drupal, Expression Engine і WordPress.

PHP працює в замкнених процесах в Apache, тому процес дуже складно вивести з побудови всього веб-браузера. Якщо щось піде не так, ефект спостерігатиметься, тому що стан PHP повністю скидається на початку кожного запиту. Це відбувається більш часто, ніж системи, що використовують живі процеси, що обробляють безліч джерел. Це абсолютно безкоштовно і не потрібно жодних комісій. Він дуже гнучкий і використовує вільний простір пам'яті. Підтримка спільноти: використання технологій є її спільнотою. Якщо ви шукаєте скрипт, швидше за все інший користувач вже створив щось подібне. Далі про недоліки.

- **Безпека**: оскільки вихідний код є вихідним, всі люди бачать вихідний код. Якщо вихідний код має помилки, люди використовують його для вивчення слабких місць.
- **Слабка типізація:** неявне перетворення може здивувати необережних програмістів і здатися несподіваною помилкою. Плутанина між масивами та хеш-таблицями. Це повільно і може бути швидше. Часто є кілька підключень до комп'ютерів. Він є суворо типизированным. Він інтерпретується та використовує фігурні дужки.
- **Поганий метод обробки помилок:** у фреймворку поганий метод обробки помилок. Це неправильне рішення для спільноти. Тому, як кваліфікованого PHP-розробника, може знадобитися це знищення.
- **PHP не може працювати з великою кількістю програм:** технологія безсила для підтримки великої кількості програм. Їм дуже складно управляти, тому що він не є модульним. Він імітує особливості мови Java.

Це не дасть продуктивності, наприклад, для мов C або C++. це мова відповідей і він інтерпретується, він буде трохи повільнішим, ніж оптимізовані програми C++.

ґ

PHP – дуже популярна мова програмування, тому вона використовується серед західних людей для створення різних типів програм. Він в основному використовується як мова реакцій на серверах веб-сайтів. Деякі люди також використовували цю мову для створення програм для Mac, Linux та Windows. Широке використання мови програмування зрозуміло, що «переваг значно більше, ніж недоліків». Так що в цілому це дешево, безпечно та швидко для розробки веб-додатків.

2.3.3 Python

Python широко хвалять. 2019 року його популярність зросла на 4,2%. Згідно з опитуванням Stack Overflow 2020, Python займає 3-е місце серед найпопулярніших мов після Rust і TypeScript.

Переваги використання:

- **Простий синтаксис:** Завдяки меншій кількості розділових знаків і символів у Python код коротший, чіткіший і цілком зрозумілий навіть для новачків у технології. Простий синтаксис також робить мову легкою для вивчення. Якщо ви порівняєте Python з Java, Python це динамічна мова. На відміну від статичної Java, інженерам не потрібно вказувати тип при оголошенні масиву і вони можуть помістити в нього все, що захочуть.
- **Асинхронна розробка:** Асинхронна розробка може бути складнішою, ніж лінійне програмування, але вона підвищує продуктивність додатків і підвищує швидкість відгуку. Асинхронне означає паралельне програмування, у якому кілька завдань можуть оброблятися одночасно. Багатопроцесорність Python дозволяє вирішувати завдання, пов'язані з процесором. Якщо є необхідність змусити систему виконувати кілька завдань одночасно, Python чудово з цим впорається. Уявіть, що програма повинна підключитися до 6 баз даних і виконати матричні перетворення.

ґ

Багатопроцесорність допоможе кожному завданню працювати на власному процесорі і в той же час скоротити час виконання, зробивши його ефективним.

- **Велика кількість бібліотек:** Python пропонує велику стандартну бібліотеку, яка є набором з більш ніж 200 основних модулів. Там розробники Python можуть знаходити та керувати документацією, базами даних, веб-браузерами, модульним тестуванням. Він дійсно величезний і надає інженерам безліч можливостей для повторного використання коду та включення його до своїх проектів. Інженери також можуть інсталювати корисні пакети з індексу пакетів Python (PyPI).

Список додаткових бібліотек Python величезний. Загалом, є 137 тисяч готових бібліотек, які дуже допомагають інженерам та позбавляють їх написання коду з нуля. Більшість бібліотек у Python мають справу з аналітикою даних, інтелектуальним аналізом даних, автоматизацією та дизайнерськими рішеннями. Найбільш популярні:

- Pandas
- Matplotlib
- NumPy
- SciPy

- **Великий список сторонніх інтеграцій:** Якщо ви хочете інтегрувати свою програму з такими рішеннями, як Twilio для обміну повідомленнями та голосовими сервісами або Stripe для платежів. Для реалізації необхідної функціональності API Python надає фреймворки API:

- Django Rest Framework
- API Star
- Starlette
- Aiohttp

- **Це чудово підходить для додатків з науки про дані:** Існують великі можливості використання Python для машинного навчання та штучного

г

інтелекту. Python багатий на готові функціональні бібліотеки та стійкі фреймворки — найпопулярніші — Flask, Django. Завдяки широким спискам можливостей і ясному коду Python дозволяє створювати великі веб-програми для різних сфер застосування. Ця технологія також відмінно підходить для створення програм AI і ML. Наприклад, штучний інтелект має вирішальне значення в автомобілебудуванні та впровадженні круїз-контролю. Це також важливо для прогнозування продажів та аналітики, фінтех-додатків із системою голосових платежів. Більшість компаній, які постачають свій продукт із інтегрованими системами штучного інтелекту, покладаються на Python. Фахівці з машинного навчання використовують Python для систем аналізу настроїв та NLP (інструментів обробки природної мови). Ви можете додати в проект аналітику зображень або тексту, визначення мови та повністю покластися на можливості Python. Одними з найпопулярніших бібліотек на Python є Seaborn, Pytorch, TensorFlow, Scikit Learn. Python - це добре масштабована і швидка мова, яка пропонує велику гнучкість у вирішенні проблем з даними. Першим кроком аналізу даних є структурування даних, з яким Python справляється дуже добре.

Виходячи з того, що система керування чергами буде тісно пов’язана з наукою про дані, а саме, розпізнаванням людей на фото, вибір падає на мову програмування Python.

2.4 Вибір веб-фреймворку для Python

2.4.1 Django

Django – це повнофункціональний фреймворк Python. Це найвідоміший і найулюбленіший фреймворк для розробки насичених веб-додатків. З часом

ґ

він набирає популярності. У 2021 році він увійшов до десятки найкращих фреймворків для веб-розробки. Він орієнтований на принцип DRY (Don't Repeat Yourself).

Django пропонує кілька вбудованих бібліотек та відмінних функцій, які доступні безкоштовно. Django використовує ORM для порівняння об'єктів із таблицями бази даних.

Він пропонує підтримку баз даних, а також забезпечує легку міграцію з однієї бази даних до іншої. Однак у ньому вбудована підтримка баз даних MySQL, PostgreSQL, SQLite та Oracle. Ми також можемо використовувати інші бази даних за допомогою драйверів сторонніх розробників.

Ключові особливості Django:

- У порівнянні з іншими веб-фреймворками, він дуже безпечний.
- Маршрутизація URL-адреси.
- Механізм шаблонів.
- Міграція схем бази даних.
- Підтримка автентифікації.
- ORM (Object Relation Model).
- Безліч готових до використання бібліотек.

2.4.2 Aiohttp

Це асинхронний фреймворк, який залежить від функціоналу Python 3.5+, такого як async і await. Бібліотека Python asyncio відіграє важливу роль у функціонуванні цього фреймворку. Будучи серверним веб-фреймворком, AIOHTTP може також служити клієнтським фреймворком. Ми можемо перенаправляти запити, використовуючи об'єкт запиту та маршрутизатор.

Ключові особливості AIOHTTP:

- Гнучка маршрутизація.

ґ

- Забезпечує можливість ефективної побудови представлень.
- Система сигналів.
- Підтримуються серверні WebSockets та клієнтські WebSockets без залежності від зворотніх викликів функцій
- Підтримується проміжне програмне забезпечення.

2.4.3 Flask

Flask – це ще один популярний мікрофреймворк Python, що випускається під ліцензією BSD. Надихає його фреймворк Sinatra Ruby. Цей фреймворк вимагає шаблонів Jinja2 та інструментарію Werzeug WSGI.

Він легкий та має модульну конструкцію. Flask легко адаптується.

Використовуючи Flask, розробники можуть побудувати міцний фундамент веб-застосунку, на основі якого можна використовувати будь-яке необхідне розширення. Він також сумісний із Google App Engine.

Ключові особливості:

- Flask Забезпечує вбудовану підтримку.
- Підтримує шаблонизацію jinja2.
- Підтримує Unicode.
- Обробка HTTP запитів.
- Вбудований швидкий наладчик.
- Допомагає підключити будь-який ORM.
- Підтримуються безпечні куки для встановлення сеансів на стороні клієнта.

2.4 Вибір брокеру повідомлень

ґ

Через вибір сервісо-орієнтованої архітектури та можливе високе навантаження сервісів з’являється необхідність у їх взаємодії один з одним. Для такої комунікації можна використовувати синхронний підхід, а саме — Http запити, або ж асинхронний — з використанням брокерів.

2.4.1 Apache Kafka

- **Масштабування:** може надсилати до мільйона повідомлень за секунду.
- **Постійність:** так.
- **Отримувачі:** один-до-одного або один-до-багатьох: тільки один-до-багатьом.

Kafka була створена компанією Linkedin у 2011 році для обробки даних з високою пропускною здатністю та низькою затримкою. Як розподілена потокова платформа, Kafka реплікує службу публікації-підписки. Вона забезпечує сталість даних та зберігає потоки записів, що робить її здатною обмінюватися якісними повідомленнями. Kafka керує SaaS на Azure, AWS та Confluent. Усі вони є творцями та основними співавторами проекту Kafka. Kafka підтримує більшість сучасних мов програмування таких як: Python, Java

2.4.2 Redis Streams

- **Масштабування:** може надсилати до мільйона повідомлень за секунду.
- **Постійність:** в принципі, ні – це сховище даних in-memory.
- **Отримувачі:** "один до одного" та "один до багатьох": обидва варіанти.

г

Redis трохи відрізняється від інших брокерів повідомлень. По суті Redis - це сховище даних у пам'яті, яке може використовуватися як високопродуктивне сховище ключових значень або як брокер повідомлень. Ще одна відмінність у тому, що Redis немає персистентності, а скидає свою пам'ять на диск/БД. Він також ідеально підходить для обробки даних у реальному часі.

Спочатку Redis був не one-to-one і many-to-one. Однак після того, як у Redis 5.0 з'явився pub-sub, можливості розширилися, і may-to-one став реальною можливістю.

Оскільки Redis забезпечує надзвичайно швидке обслуговування та можливості in-memory, він є ідеальним кандидатом для короткоживучих повідомлень, де збереження не таке важливе, і ви можете миритися з деякими втратами.

ґ

ВИСНОВОк до Розділу 2

В розділі були розглянуті та порівняні основні інструменти для розробки системи управляння чергами, а саме: архітектурний підхід — суміш моноліту і мікросервісів, що ще часто називають сервісо-орієнтованим підходом, для уникнення проблем розподілених систем, як властиві мікросервісній архітектурі, фреймворк для клієнтської частини — React з його бібліотекою React Bootstrap, через велику кількість готових рішень та простоту розробки користувацьких інтерфейсів, та мова і фреймворк для серверної частини — Python та aiohttp, так як, при розвитку проекту будеде збільшена кількісьть IOT камер та треба буде оброблять більшу кількість запитів у секунду, отже вибір асинхронного підходу є доцільним.

ґ

РОЗДІЛ 3. РОЗРОБКА ПРОГРАМНОГО ПРОДУКТУ

3.1 Постанова вимог і задач

Потрібно розробити веб-додаток, для моніторингу черг в супермаркетах в реальному часі, та при виникненні черг, будеінформуватися персонал та відкриватимуться нові каси.

Повинен бути реалізований наступний функціонал:

7. Авторизація
8. Можливість оглядати стаистику надходження фото з камер та кількість інцидентів
9. Розсилка повідомлень внаслідок виникнення інциденту
10. Можливість додавати магазини, локації, камери та інші ключові сутності

3.2. Вибір мови програмування

Для розробки веб додатку була вибрана мова програмування Python. Вибір

зумовлений наступними факторами:

- Python – об’єктно-орієнтовна мова, це дає можливість гнучкої розробки та застосування багатьох паттернів програмування;
- Більшість нейронних мереж написано на цій мові, тож не доведеться використовувати одночасно дві;
- Мова має колосальну стандартну бібліотеку з більш ніж 200 модулями та велику кількість фрейморків

г

3.3. Вибір супутніх програм та технологій

- Як ORM до фреймворку Aiohttp буде використаний асинхронний аналог Django orm — Tortoise
- Для спрощення роботи клієнта з базою даних та розвантаження основного бекенду затосовано Hasura — що надає можливість розбити прості запити одрзу з клієту до бази даних використовуючи QraphQL
- Для обміну повідомленнями між сервісами застосовану технологію Redis Streams та бібліотеку aioredis
- Для розробки нейронної мережі обрано бібліотеку PyTorch

3.4. Проєктування

Для реалізації програми розроблена діаграма класів зерен, яка подана на рисунку 3.1.

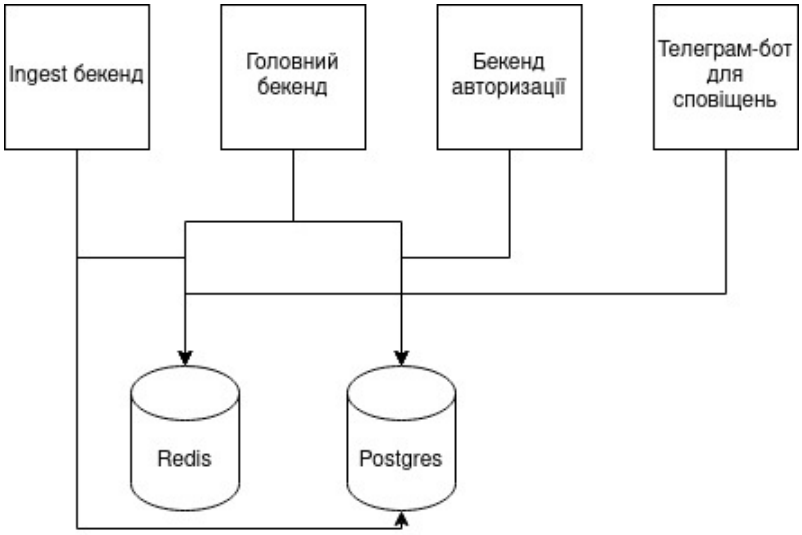


Рисунок 3.1. Схема сервісів

При проєктуванні було створено 4 сервіси та підключено їх до 2 сховищ даних.

г

3.4.1 Ingest

В даному сервісі реалізована обробка офотографій, що надіслані камерами по HTTP.

Алгоритм:

- При надходженні фото одразу перевіряється приналежність фото до камери
- Відправляться запит до класу TimeoutManager для перевірки таймауту камери, при довгому “простоюванні” камери створюється timeout інцидентв
- Дані про фото записуються в кеш
- Фото зберігається в сховище (AWS S3)
- Фото відправляється на подальшу обробку через брокер

3.4.2 Головний бекенд

В цьому сервісі надані основні API ендпоінти для взаємодією з базою даних, та модулі для обробки фотографій та розпізнавання людей на них.

Реалізовані CRUD операції для додавання мереж магазинів, магазинів, локацій в магазинах, камер а також читання фотографій (QueueReading).

Також описана логіка обробки фотографій для зручного читання нейронною мережею. Реалізований модуль для виділення регіонів при обробці фотографії.

3.4.3 Бекенд авторизації

В цьому сервісі реалізований функціонал для авторизації користувачів, розподілу їх по ролям. Також в модулі містяться декоратори на обов’язкову аторизацію при запитах до бекенду та на певні права доступу.

г

3.5 Деталі реалізації

3.5.1 Брокер

За допомогою бібліотеки aioredis підходу зі зворотнім викликом функцій було реалізовано брокер для відправки та отримання повідомлень. Реалізація брокеру з використанням асинхронного підходу зображено на рисунку 3.2.

Основна ідея полягає у швидкому доступі до даних через кеш методами set та get для встанвлення та вибірки даних відповідно. Маніпуляція даними відбувається всередині нескінченного циклу що дозволяє передавати інформацію для обробки в реальному часі.

Для отримання даних переданих методом xadd в один із redis streams в іншому сервісі в redis_litener передається інформація через глобальну змінну redis_cb яка наявна у кожному модулі з консьюмером і є словником. Приклад використання при створенні фонової задачі з косьюмерами для модулів зображено на рисунку 3.3 на прикладі запуску консьюмерів для сервісу з телеграм-ботом.

г

```python
async def redis_listener(redis, callbacks, name='listener'):
    streams = [k for k, v in callbacks.items()]
    logging.info(f'{streams=}')
    last_id_keys = [f'{name}_last_id_{k}' for k in streams]

    try:
        last_ids = await redis.mget(last_id_keys)
        for i, (stream, last_id) in enumerate(zip(streams, last_ids)):
            if last_id is None:
                latest = await redis.xrevrange(stream, count=1)
                if latest:
                    nlast_id, _ = latest[0]
                else:
                    nlast_id = 0
                last_ids[i] = nlast_id
                await redis.set(f'{name}_last_id_{stream}', nlast_id)
        logging.info(f'Listening for {streams}: {last_ids}')
    except Exception as e:
        await log_exc(e, redis)
        logging.exception(f'Something went wrong {e}')

    while True:
        last_ids = await redis.mget(last_id_keys)
        logging.info(f'Listening for {streams}: {last_ids}')
        events = await redis.xread(dict(zip(streams, last_ids)), block=0)
        logging.info(f'Got events: {events}')
        for stream, items in events:
            for last_id, fields in items:
                try:
                    logging.info(f'Got item from {stream}')
                    await callbacks[stream](fields)
                    logging.info(f'{stream=}, {last_id=}')
                except Exception as e:
                    await log_exc(e, redis)
                    logging.exception(f'Failed to send {fields} with error {e}')
                finally:
                    await redis.set(f'{name}_last_id_{stream}', last_id)
```

Рисунок 3.2. Redis брокер

```python
async def alert_setup(_):
    global redis, listener_task
    await db_init()
    redis = await get_redis()
    listener_task = asyncio.get_running_loop().create_task(
        redis_listener(redis, redis_cbs, 'alertbot')
    )
```

Рисунок 3.3. Налаштування брокеру

3.5.2 Менеджер інцидентів

Менеджер інцидентів — це клас, що створений за принципом паттерну

г

Singletone, тобто цей клас може мати лише один екземпляр на весь сервіс.

Екземпляр цього класу надає інтерфейс для старту, зупинки чи оновлення статусу інциденту. Якщо камера не надсилає фото впродовж встановленого часу — виникає інцидент таймауту. Всі існуючі та майбутні типи інцидентів використовують саме цей підхід

3.6 Нейронна мережа

При обробці фотографій зробленими камерами, що закріплені над касами, для досягнення поставлених цілей виникає дві задачі:

- Підрахунок окремо людей та касирів (необхідно для коректної роботи алгоритму)
- Розділення зон розпізнавання на фотографії (для людей та касирів)

Для розпізнавання людей на фото було обрано бібліотеку PyTorch, взято з неї готову нейронну мережу YoloV5 та перенавчено її на реальних фото черг з магазинів.

3.7 Розгортання

Для комфортної розробки та відносно легкого розгортання необхідно використовувати системи “оркестрації”, якщо ми маємо справу з великою кількістю сервісів.

Розробка ПЗ ведеться виключно в ізольованих контейнерах Docker для відсутності конфліктів оточень. Для зручного управління декількома сервісами зручно використовувати утиліту docker-compose разом з Traefik. Це дозволяє розгортати проєкт у дуже короткий час та легко встановлювати ssl-сертифікацію використовуючи letsencrypt.

г

3.7 Алгоритм механізму черги

Нижче приведено алгоритм виникнення інцидентів черги за приведеними в налаштуваннях критеріями:

- Якщо в локації до якої прbв’язана камера не встановлена максимальна кількість людей — береться стандартне значення з налаштувань
- Іде перевірка на вже створений інцидент для даної локації
- Якщо інцидент вже створений:

1. Інцидент перевіряється на критичність (критичним вважається інцидент, якщо при заданому часі реагування на інцидент персонал не встиг відкрити нову касу

2. Якщо кількість людей на фотографії перевищує задану — інцидент вважається діючим і відправляється запит до моделі QueueOverLimitIncident яка наслідується від менеджеру інцидентів

3. Якщо кількість людей на фотографії не перевищує задану , то проводиться перевірка критеріїв для закриття інциденту — йде запит до бази даних та повертається інформація про останнє фото. Якщо в останньому запису з камери клькість людей менша за задану — інцидент рахується неактивним.

- Якщо інцидент не знайдено:

1. Якщо на фото людей менше заданої кількості — вихід з функції

2. З бази даних повертаються останні фотографії відсортовані по даті від найновіших у заданій кількості та іде перевірка чи у всіх фото кількість людей перевищує норму, якщо ні — вихід з функції, якщо так то дістається ще один попередній допис та перевіряється на кількість людей - якщо більше норми, то йде запит до менеджеру інцидентів для створення інциденту черги, у іншому випадку — вихід з функції.

ґ

Висновки до розділу 3

В даному розділі, базуючись на даних та технологіях, що описані в розділі 2, була описана структура проєкту та його функціонал. Також було описано у вибірковому порядку основні технічні характеристики, підходи вдо проєктування та алгоритми щодо побудови складної системи такого типу.

г

РОЗДІЛ 4. ТЕСТУВАННЯ ПРОГРАМНОГО ПРОДУКТУ

4.1 Вимоги до технічного забезпечення

1)Основні вимоги до устаткування:

- Ноутбук або ПК з процесором починаючи з Intel Core i5-12400, з тактовою частотою 2.4 гігагерц, рекомендовано — Intel Core i5 з тактовою частотою 3.3 гігагерц та вище;
- оперативна пам’ять об’ємом — 4 Гб, рекомендований об’єм — 8 Гб та більше;
- мінімальний об’єм пам’яті на диску — 50 Г,, реомендований об’єм - 100Гб;

2)Вимоги до програмного забезпечення:

- операційна система Linux, рекомендований дистрибутив Fedora;
- з’єднання з мережею Інтернет.

4.2 Інструкція користувача

Програмний продукт запускає систему “оркестрації” за допомогою команди make, яка має бути зазделегіть встановлена на ПК. Попередньо потрібно запустити команду make init для ініціалізації бази даних і створення користувача за замовчуванням. Логін за замовчуванням — admin, а пароль можна переглянути у файлі ADMIN_PASSWORD в корені проєкту.

Рисунок 4.1. Скриншот запущеного бекенду

г

Після переходу на https://localhost необхідно авторизуватися. Далі користувач потрапля до вкладки з панеллю управлінні (рисунки 4.2., 4.3., 4.4).

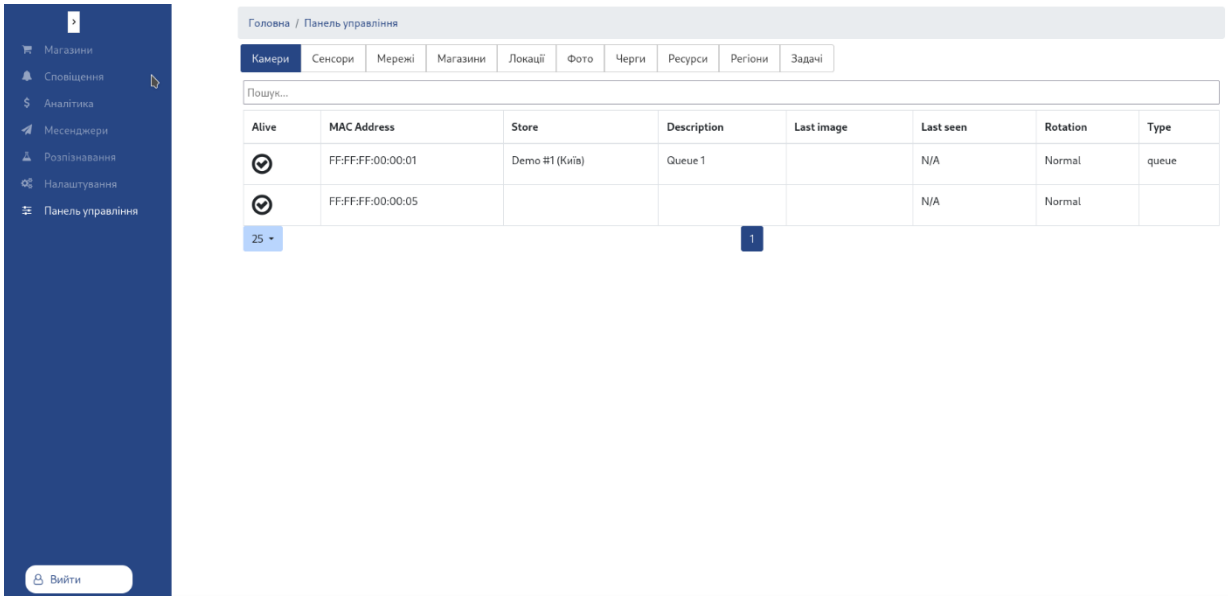


Рисунок 4.2 Панель управління камерами

В даній вкладці відображається вся актуальна інформація по встановленим камера, їх дані та чи продовжують вони надсилати фотографії до додатку.

На рисунку 4.4. показана панель управління магазином. Наявний фукціонал додавання мапи магазину, локацій та камер.

г

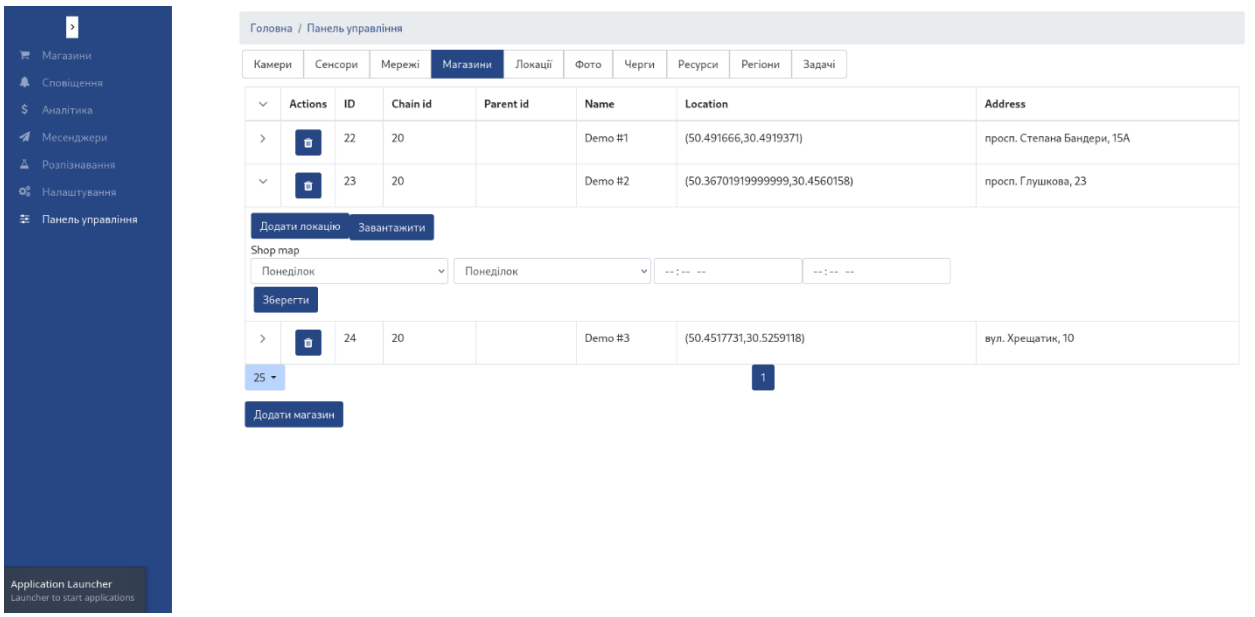


Рисунок 4.4. Панель управління магазинами

Також є вкладка для корегування налаштувань, залежно від потреб бізнесу, прикклад наведений на рисунку 4.5.

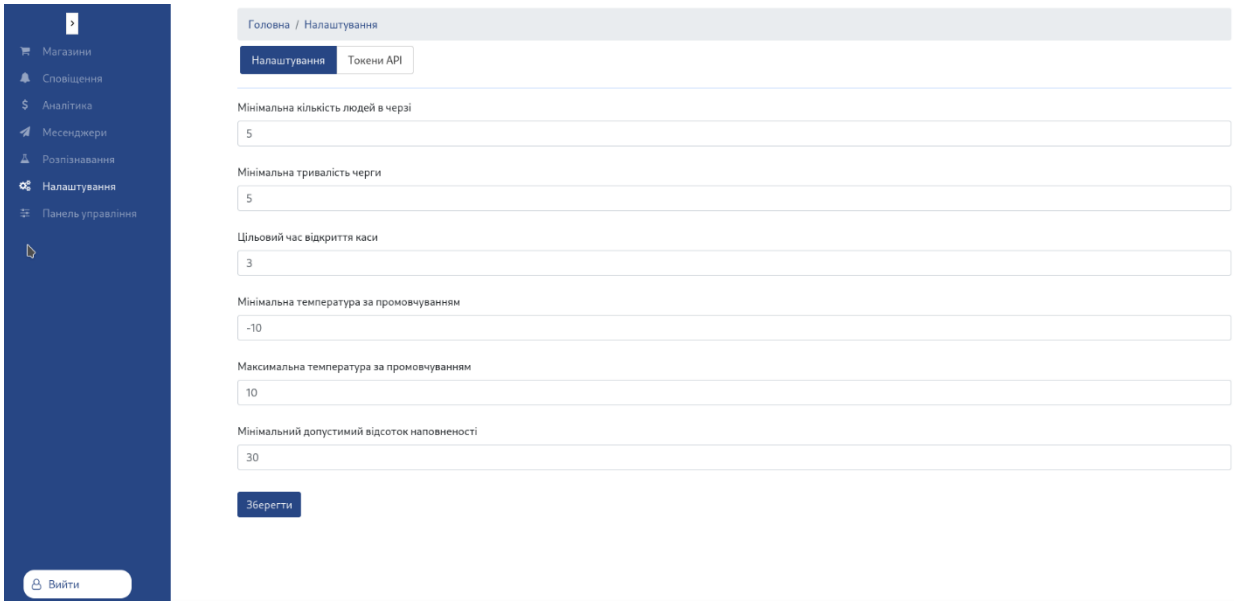


Рисунок 4.5. Панель налаштувань

Наступна вкладка містить всю необхідну інформацію про виникнення інцидентів черги, їх тривалість, час іквідації і тому подібне (рисунок 4.6.)

г

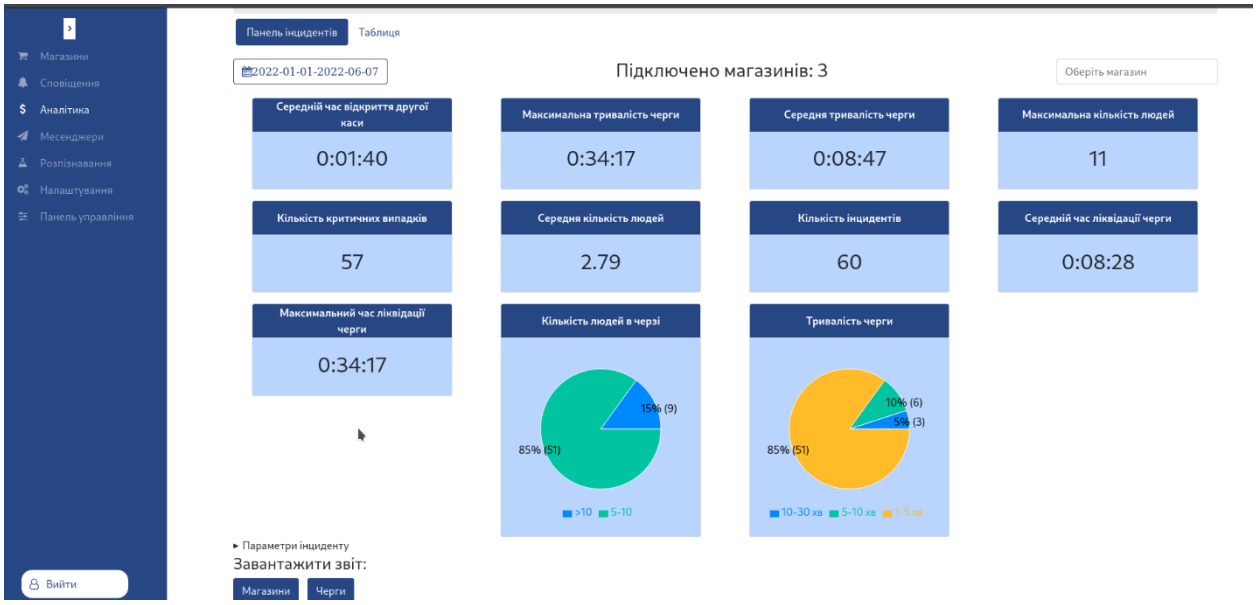


Рисунок 4.6 Панель аналітики

Також є можливість відображення аналітичної іформації в тибличному вигляді (рисунок 4.7)

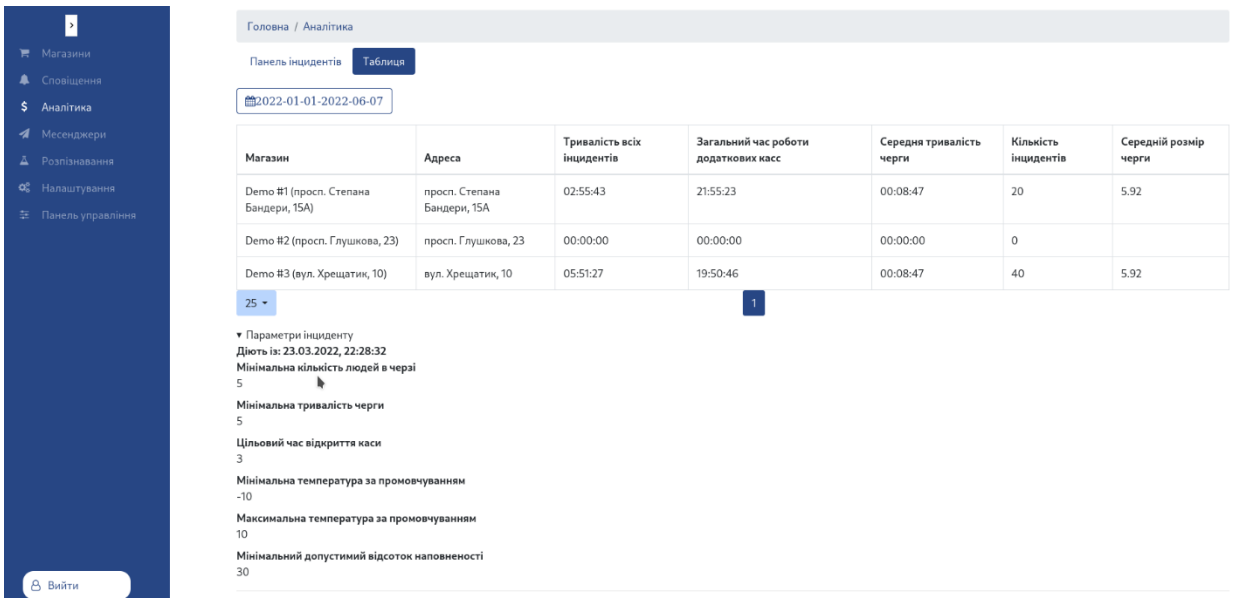


Рисунок 4.7. Панель аналітики (таблиця)

Всі сповіщення про виникнення будь-яких інцидентів які надходять до месенджерів логуються також і до сховища даних. Приклад сповіщень зображено на рисунку 4.8.

г

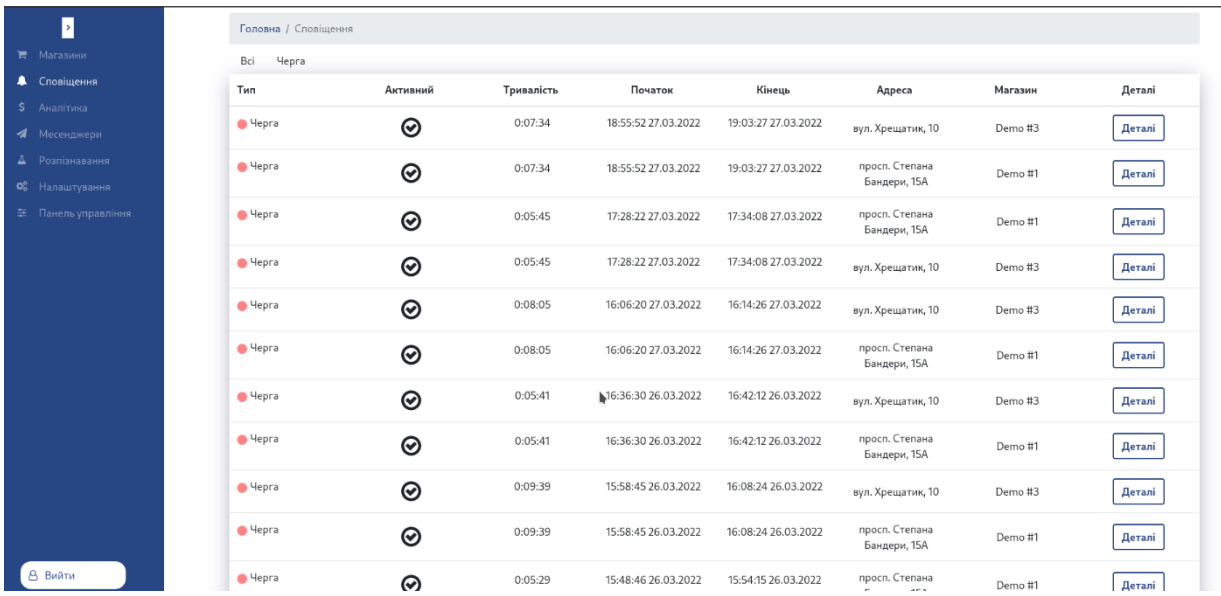


Рисунок 4.8. Панель сповіщень

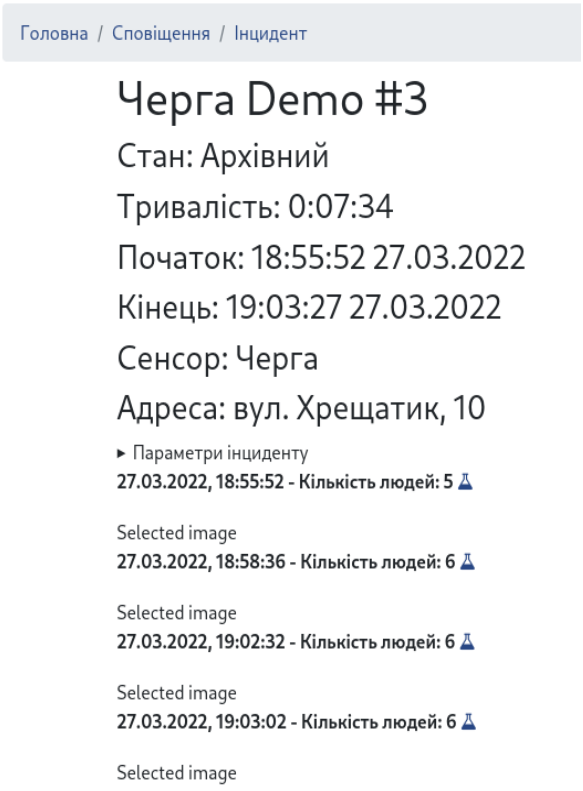


Рисунок 4.9. Деталі інциденту

ґ

Висновки до розділу 4

У цьому розділі були надані основні вимоги до технічного забезпечення.

Також оформлена розгорнута «Інструкція користувача» та приклади інтерфейсу з детальним оглядом наявних можливостей функціоналу. Дана документація має на меті полегшити знайомство з програмним продуктом для користувачів.

ґ

ВИСНОВКИ

У межах дипломного проєкту були досягнуті перераховані нижче цілі:

1. Проведено аналіз існуючих рішень та ринку систем управління чергами.
2. Було розглянуто актуальні технології для ефективного напсання сучасних додатків та зроблено оптимальний вибір
3. Було розроблено веб-додаток, який допомагає боротися зі скупченнями людей в супермаркетах.
4. Програмний продукт було протестовано та ретельно задокументовано.

г

СПИСОК ВИКОРИСТАНОЇ ЛІТЕРАТУРИ

1. William Athas and Nanette Boden Cantor: An Actor Programming System for Scientific Computing in Proceedings of the NSF Workshop on Object-Based Concurrent Programming. 1988. Special Issue of SIGPLAN Notices.
2. Кнут, Д. Искусство программирования, том 1. Основные алгоритмы [Текст] / Д. Кнут. — М. : Вильяме. 2006. — 720 с.
3. Кнут, Д. Искусство программирования, том 2. Получисленные методы [Текст] / Д. Кнут. - М. : Вильяме. 2007. - 720 с.
4. Филипп Крюштан: Архитектурные Проекты - 4+1 Модель Представления Архитектуры программного обеспечения. В: программное обеспечение IEEE. 12 (6) ноября 1995, стр 42-50
5. 0ртега, Дж. Введение в параллельные и векторные методы решения линейных систем [Текст] / Дж. Ортега. — М. : Мир, 1991.
6. Пол Клементс, Феликс Бахман, Лен Басс, Дэвид Гэрлан, Джеймс Иверс, Тростник Мало, Паулу Мерсон, Роберт Норд, Джудит Стэффорд: Документирование Архитектуры программного обеспечения: Взгляды и Вне, Второй Выпуск. Аддисон-Уэсли, 2010, ISBN 0-321-55268-7
7. Лен Басс, Пол Клементс, Рик Кэзмен: Архитектура программного обеспечения на практике, Третий Выпуск. Аддисон Уэсли, 2012, ISBN 0-321-81573-4
8. Танненбаум,Э. Распределенные системы. Принципы и парадигмы [Текст] / Э. Танненбаум, М. Ван Стеен. - СПб. : Питер. 2003. - 877 с.
9. Тербер,К. Дж. Архитектура высокопроизводительных вычислительных систем [Текст] / К. Дж. Тербер. - М.: Наука. 1985.
10. Python [Електронний ресурс] - https://www.python.org/
11. Len Bass, Paul Clements, Rick Kazman. Software Architecture in Practice (3rd Edition) (SEI Series in Software Engineering). — 3 edition (October 5, 2012). — 2012. — 640 с. — ISBN 978-0321815736.

г

12. . Э. Гамма, Р. Хелм, Р. Джонсон, Дж. Влиссидес. Приемы объектно-ориентированного проектирования. Паттерны проектирования = Design Patterns: Elements of Reusable Object-Oriented Software. — СПб: «Питер», 2007. — С. 366. — ISBN 978-5-469-01136-1. (также ISBN 5-272-00355-1)

ДОДАТОК 1

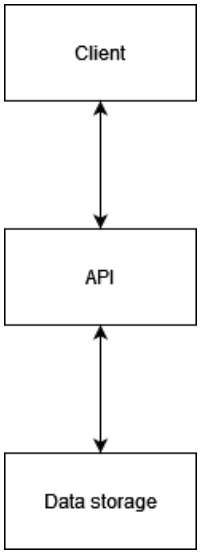
Система керування чергами в супермаркетах

Структурна схема системи

ІАЛЦ.467200.004 Д1

Аркушів 1

Київ 2022 р



					ІАЛЦ.467200.005 Д2	
Зм.	Арк.	№ докум.	Підпис	Дата		
Розробив		Савічев Д.А.			**Система керування чергами в супермаркетах** Структурна схема	Літ. / Арк: 1 / Аркушів: 1
Перевірив		Ткаченко В.В.				
Реценз.						НТУУ КПІ ім. Ігоря Сікорського, ФІОТ ІВ-81
Н. Контр.		Сімоненко В. П.				

ДОДАТОК 2

Система керування чергами в супермаркетах

Функціональна схема (схема сховища даних)

ІАЛЦ.467200.005 Д2

Аркушів 1

Київ 2022 р

					ІАЛЦ.467200.005 Д2	
Зм.	Арк.	№ докум.	Підпис	Дата		
Розробив		Савічев Д.А.			**Система керування чергами в супермаркетах** Функціональна схема (схема сховища даних)	Літ. / Арк / Аркушів: 1 / 1
Перевірив		Ткаченко В.В.				
Реценз.						НТУУ КПІ ім. Ігоря Сікорського, ФІОТ
Н. Контр.		Сімоненко В. П.				

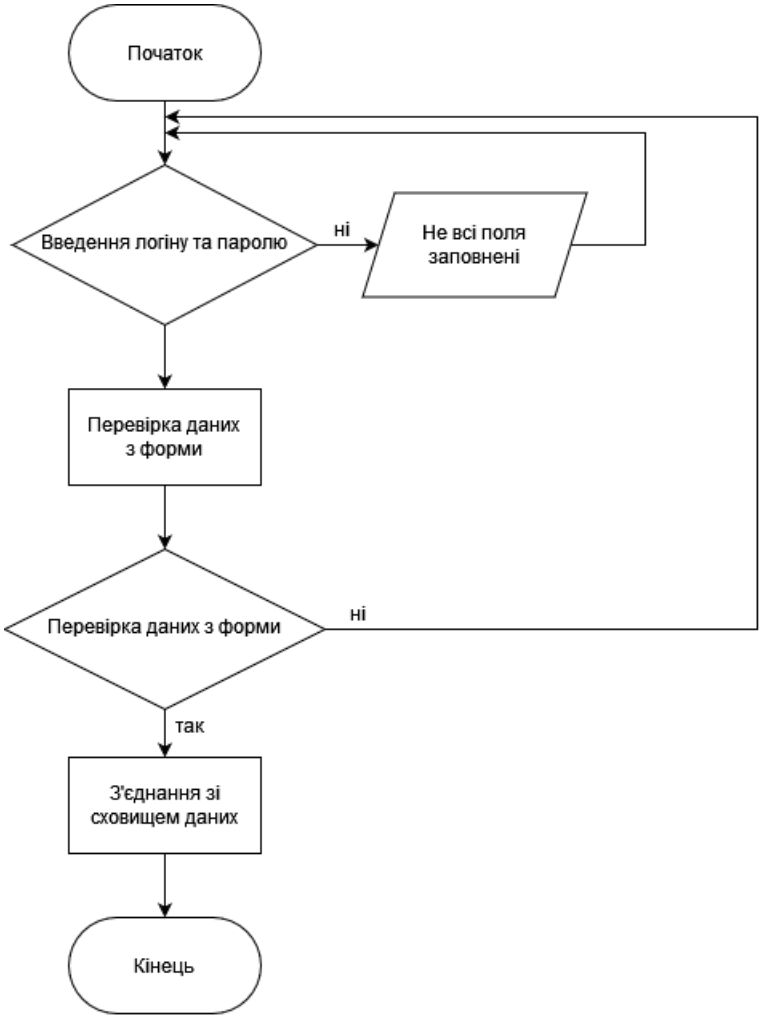
ДОДАТОК 3

Система керування чергами в супермаркетах

Алгоритм авторизації

ІАЛЦ.467200.006 Д3

Аркушів 1



					ІАЛЦ.467200.005 Д2	
Зм.	Арк.	№ докум.	Підпис	Дата		
Розробив		Савічев Д.А.			**Система керування чергами в супермаркетах** Принципова схема (схема авторизації)	Літ. / Арк: 1 / Аркушів: 1
Перевірив		Ткаченко В.В..				
Реценз.						НТУУ КПІ ім. Ігоря Сікорського, ФІОТ ІВ-81
Н. Контр.		Сімоненко В. П.				