МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ХЕРСОНСЬКИЙ ДЕРЖАВНИЙ УНІВЕРСИТЕТ
Факультет комп'ютерних наук, фізики та математики
Кафедра інформатики, програмної інженерії
та економічної кібернетики

ПРОЄКТУВАННЯ ТА РОЗРОБЛЕННЯ СЕРВІСУ ПОВІДОМЛЕНЬ НА ОСНОВІ МІКРОСЕРВІСНОЇ АРХІТЕКТУРИ

Кваліфікаційна робота (проєкт)

на здобуття ступеня вищої освіти “бакалавр”

Виконав: студент 431 групи
Спеціальності 122 Комп'ютерні науки
Освітньо-професійної програми:
Комп'ютерні науки
Лапшій Олександр Миколайович
Керівники: кандидатка технічних наук,
доцентка Шишко Людмила
Станіславівна,
кандидат фізико-математичних наук,
доцент Єрмолаєв Вадим Анатолійович
Рецензент:
к.ф.м.н., доцент Кузьмич В.І.

Херсон – 2021

ЗМІСТ

ВСТУП

Актуальність теми: Сучасна людина почала більше свого вільного часу проводити в інтернет-мережі. Значна частина часу припадає на спілкування в соціальних мережах та месенджерах. Рекламні компанії, інтернет-маркетинг кардинально змінились та підстроїлись під сучасного користувача. Більшість людей користується інтернетом через телефон — тому не дивно, що месенджери стали найпопулярнішим способом взаємодії із користувачем. Сервіси повідомлень дають швидку і персональну відповідь на запит — те, чого очікує сучасний клієнт. Боти, як сервіс для месенджерів, стали потужним інструментом в бізнесі: дуже зручним, багатофункціональним та не викликає дискомфорту у користувачів.

Для забезпечення стабільної та комфортної роботи розробники використовують різні підходи до організації архітектури програми. Мікросервісна архітектура за допомогою незалежності розробки частин додатку та легкою зміною компонентів є найбільш зручною для розробки сервісів повідомлень. Зараз це актуально для ринка України та закордоном, а отже попит на таких спеціалістів з розробки стрімко зростає.

Мета дослідження: на основі мікросервісної архітектури спроєктувати та розробити сервіс повідомлень.

Завдання дослідження: дослідити сервіси повідомлень та основні принципи роботи мікросервісів, спроєктувати універсальну структуру додатку на базі мікросервісної архітектури і за допомогою неї розробити сервіс повідомлень.

Об'єкт дослідження: взаємодія мікросервісів для створення додатка.

Предмет дослідження: сервіс повідомлень на основі мікросервісної архітектури, як засіб взаємодії із користувачем.

РОЗДІЛ 1
МІКРОСЕРВІСНА АРХІТЕКТУРА

1.1 Що таке мікросервіси?

Теперішні підприємства наштовхуються на значні проблеми, що орієнтовані на клієнта в розподільному порядку. Парадигма мікросервісів постала «наступною великою річчю» задля реалізації IT-результатів для сприяння сучасного виробництва, з великою кількістю постачальників технологій та послуг.

Не дивлячись на те, що поняття «мікросервіси» постало з середини 2000-х років, масове появлення терміну відбулося в 2011 рік. Зокрема у 2012 році мікросервіси згадувались на конференції 33d Degree в Кракові.

Архітектура мікросервісу або просто мікросервісів – це особливий метод розробки програмних систем, який старається зосередитись на будові однофункціональних модулів з конкретними інтерфейсами та операціями. Ця тенденція набула популярності протягом останніх років, позаяк підприємства прагнуть стати більш спритними та рухатись до DevOps та постійно тестуватись.

Мікропослуги також відомі як архітектура мікропослуг – це архітектурний стиль, який структурує додаток як сукупність послуг, які є:

- дуже ремонтопридатними та перевіряються;
- слабко зчеплені;
- незалежно розгортаються;
- організовано навколо ділових можливостей;
- належить невеликій команді.

Архітектура мікросервісу дозволяє швидко, часто і надійно доставляти великі, складні програми. Це також дозволяє організації розвивати свій стек технологій.

Мікросервіси мають багато переваг для команд Agile та DevOps, як зазначає Мартін Фаулер (інженер-програміст), Netflix, eBay, Amazon, Twitter, PayPal та інші технічні зірки еволюціонували від монолітної до архітектури мікросервісів. На відміну від мікросервісів, монолітний додаток побудований як єдиний, автономний блок. Це вносить зміни в додаток повільно, оскільки це впливає на всю систему. Зміни, внесені до невеликого розділу коду, можуть вимагати створення та розгортання абсолютно нової версії програмного забезпечення. Масштабування конкретних функцій програми також означає, що потрібно масштабувати всю програму.

Мікросервіси вирішують ці виклики монолітних систем, будучи максимально модульними. У найпростішій формі вони допомагають створити додаток як набір невеликих служб, будь-яка орудує у своєму особистому процесі і також є можливість розгорнутись незалежно. Ці послуги можуть бути написані різними мовами програмування та можуть використовувати різні методи зберігання даних. Хоча це призводить до розробки масштабованих та гнучких систем, вона потребує динамічного перетворення. Мікросервіси часто підключаються за допомогою API, і можуть використовувати багато однакових інструментів та рішень, які виросли в екосистемі RESTful та веб-сервісу. Тестування цих API може допомогти перевірити потік даних та інформації протягом усього розгортання мікросервісу.

1.2 Шість характеристик мікропослуг

1) Кілька компонентів.

Програмне забезпечення, сформоване як мікросервіс, за визначенням може бути розбито на багатокомпонентні служби. Чому? Тому, що кожна з даних служб уміє бути розгорнута, тобто цілісно налаштована, а потім передислокована незалежно, не порушуючи цілісність програми. Як результат, може пригодитися тільки трансформувати одну або кілька окремих служб, замість того, щоб

перерозподіляти цілі програми. Варто зазначити, що даний підхід включає в себе ряд мінусів, включаючи дорогі віддалені дзвінки (замість викликів у процесі), більш грубі віддалені API та підвищену складність під час перерозподілу відповідальності посеред ключовими компонентами.

2) Створений для бізнесу.

Стиль мікропослуг зазвичай організований навколо ділових можливостей та пріоритетів. На відміну від традиційного монолітного підходу до розроблення – де будь-яка команда має особливу увагу, скажімо, на інтерфейси користувача, технологічні рівні або логіку на стороні сервера – архітектура мікросервісу користується міжфункціональними групами. Повинності кожної команди поникають у виробництві визначених продуктів на засадах однієї або декількох суб'єктивних служб, які спілкуються через програми звісток. У сфері мікросервісів команда володіє матеріалом протягом усього життя, як у часто цитованій в Amazon фразі «Ви будуєте його, ви запускаєте його».

3) Проста маршрутизація.

Мікросервіси діють трохи схоже на класичну систему UNIX: вони одержують запити, розглядають їх і адекватно генерують відповідь. Це протилежне тому, як працює багато інших продуктів, таких як ESB (Enterprise Service Buses), де використовуються високотехнологічні системи для маршрутизації повідомлень і застосуванні бізнес-правил. Варто сказати, що мікросервіси включають в собі толкові кінцеві точки, що досліджують інформацію та використовують закономірність, і німі канали, по яких інформація протікає.

4) Децентралізований.

Оскільки мікросервіси залучають різноманітні технології та платформи, адже методики центрального керівництва старих шкіл не є оптимальними. Децентралізоване управління сприяє спільноті мікросервісів, оскільки його розробники прагнуть виробляти вигідні

інструменти, які в майбутньому можуть споживати інші, для вирішення тих самих проблем. Подібно до децентралізованого управління, архітектура мікросервісів також сприяє децентралізованому управлінню даними. Об'єднані системи користуються єдиною логічною основою даних для різноманітних додатків. У програмі мікросервісу кожна служба зазвичай управляє особистою винятковою базою даних.

5) Стійкість до відмов.

Мікросервіси призначені для того, щоб справлятися з невдачами. Позаяк декілька унікальних і багатоманітних служб взаємодіють одна з одною, цілком імовірно, що послуга може зійти з ладу з ряду причин (зокрема, коли постачальник недоступний). В таких випадках клієнт зобов'язаний попустити сусіднім службам функціонувати, поки він не функціонує належним чином. Однак моніторинг мікросервісів може допомогти запобігти ризику збою. Зі зрозумілих мотивів така вимога додає мікросервісам більшої складності порівняно з архітектурою об'єднаних систем.

6) Еволюційна.

Архітектура мікросервісів є еволюційним дизайном, чудово годиться для еволюційних систем, де не можна цілком спрогнозувати типи пристроїв, що колись можуть отримати доступ до додатку. Багато додатків постають на основі монолітної архітектури, але оскільки з’явилося кілька непередбачених вимог, можна повільно оновити до мікросервісів, які взаємодіють через стару монолітну архітектуру за допомогою API.

1.3 Приклади мікропослуг

Значна кількість масштабних веб – сайтів, зокрема такі як: Netflix, Amazon і eBay розвивались та трансформувались з монолітної архітектури і перейшли до архітектури мікросервісів.

Netflix має широко розповсюджену архітектуру, яка трансформувалася з монолітної на SOA. Щодня він одержує понад один

мільярд дзвінків з понад 800 різноманітних видів пристроїв до особистого API потокового відео. Потім кожен виклик API запрошує біля п’яти побічних запитів до серверної служби.

Amazon також перейшов на мікросервіси. Вони отримують незліченну кількість дзвінків з різних програм – зокрема програми, що розпоряджаються API веб-служби, і сам веб-сайт, це було б неможливо для старішої дворівневої архітектури.

Сайт аукціону eBay – ще один приклад, який пройшов таку ж трансформацію. Зазначу, що провідний додаток формується з кількох окремих додатків, кожен з них реалізовує бізнес-логіку для розбіжних видів функцій.

1.4 Переваги та недоліки

Шлюзи API у мікросервісах можуть значно скоротити час та зусилля для збірки та контролю якості.

Однією із поширених проблем є спільне використання схеми, логіки перевірки між службами. Те, що потрібно А, щоб вважати деякі дані дійсними, не завжди стосується В, якщо В має інші потреби. Найкраща рекомендація – застосовувати схему керування версіями та розповсюджувати у спільних бібліотеках. Зміни в бібліотеках стають обговореннями між командами. Крім того, із сильним керуванням версіями виникають залежності, які можуть спричинити більші накладні витрати. Найкращою практикою для подолання цього є планування зворотної сумісності та прийняття регресійних тестів від зовнішніх служб. Вони спонукають до розмови до того, як порушити чужий бізнес-процес, а не після.

Як і в усьому іншому, чи підходить мікросервісна архітектура, залежить від ряду вимог, оскільки всі вони мають свої плюси і мінуси.

Ось короткий опис хорошого і поганого.

Плюси:

- архітектура мікросервісу надає розробникам свободу самостійної розробки та розгортання служб;
- мікросервіс може бути розроблений досить невеликою командою;
- код для різних служб може бути написаний різними мовами (хоча багато практиків не рекомендують це робити);
- проста інтеграція та автоматичне розгортання (за допомогою інструментів безперервної інтеграції з відкритим кодом, таких як Дженкінс, Хадсон тощо);
- легко зрозуміти та змінити для розробників, отже, може допомогти новому члену команди швидко стати продуктивним;
- розробники можуть використовувати новітні технології;
- код організований навколо ділових можливостей;
- запуск веб-контейнера швидше, тому розгортання також відбувається швидше;
- коли в певній частині програми потрібні зміни, лише відповідна служба може бути змінена та передислокована - немає потреби змінювати та передислокувати всю програму;
- краща ізоляція несправностей: якщо одна мікрослужба вийде з ладу, інша буде продовжувати працювати (хоча одна проблемна область монолітного додатку може поставити під загрозу всю систему);
- легко масштабувати та інтегрувати зі сторонніми послугами;
- відсутність довгострокової відданості технологічному стеку.

Мінуси:

- через розподілене розгортання тестування може стати складним та нудним;
- збільшення кількості послуг може призвести до виникнення інформаційних бар'єрів;

- архітектура додає додаткової складності, оскільки розробникам доводиться пом'якшувати відмовостійкість, затримку мережі та мати діло з різними форматами повідомлень, а також балансуванням навантаження;
- будучи розподіленою системою, це може призвести до дублювання зусиль;
- коли кількість послуг зростає, інтеграція та управління цілими продуктами може ускладнитися;
- на додаток до декількох складностей монолітної архітектури, розробникам доводиться мати роботу з побічною складністю розподіленої системи;
- розробники повинні докласти додаткових зусиль для впровадження механізму зв'язку між службами;
- обробка випадків використання, які охоплюють більше однієї послуги без використання розподілених транзакцій, є не лише жорсткою, але також вимагає спілкування та співпраці між різними командами.

1.5 Перехід від моноліту до мікросервісу

Мартін Фаулер, славний автор і програміст, пропонує слідувати домінанті «спершу моноліт». Таке бачення прораховано тим, що уживання архітектури мікросервісу з почину розроблювання є ризикованим, позаяк в більшості подій воно годиться тільки для премудрих систем та чималих команд розробників.

Основним критерієм, котрий зобов'язаний підштовхувати до переходу до нової архітектури, є величина команди. Невеликі групи не повинні цього робити.

Враховуючи зростання компанії та розросту команд розроблювачів може пригодитися краща координація. Програмісти неодноразово починають перешкоджати роботі між собою. Збагнути призначення точного фрагмента коду постає важчим. В подібних подіях перехід до мікропослуг має беззаперечну підставу, оскільки допоможе поділити обов'язки та зможе внести ясність у суцільну картину системи. Будь-яка

ланка повинна дотримуватися поставлених цілей і працювати більш самостійно, приділяючи увагу та висвітлюючи інтерфейс. Проте зазначу, задля того, щоб перехід мав виправдання, розробників повинна бути значна кількість.

1.6 Причини використання

Одним з мотивуючих чинників вживання мікросервісів є те, що підприємства прагнуть раптово внести зміни, задля швидкого реагування на переміни вимог бізнесу, тим самим мати можливість опередити конкурентів. Мікросервіси сприяють роботі скоріше, та якісніше провадити зміни, схвалювати хід розроблювання продукту, аж коли він стає неосяжним. Врешті-решт, не щільно пов'язані служби допускають добавляти зміни з значно короткою періодичністю ітерацій, тим самими поглинаючи вплив перемін на останки системи.

Проте не варто забувати, даний підхід добавляє і важкості проєкту в цілому. Адже необхідні DevOps задля моніторингу і керування, і посеред них та розробників мають бути зв’язки та взаємодія. Робота з мікросервісами покладає більше, система моніторингу стає важчою, імовірність збоїв суттєво збільшується. Таким чином, сильна культура DevOps є досить важливою для виробництва.

1.7 Майбутнє архітектури мікросервісів

Незважаючи на те, чи стане архітектура мікросервісів улюбленим стилем розробників у майбутньому, це явно потужна ідея, яка пропонує серйозні переваги при розробці та впровадженні корпоративних програм. Багато розробників та організацій, ніколи не використовуючи назви або навіть не позначаючи свою практику як SOA, застосовують підхід до використання API, які можна класифікувати як мікросервіси.

Можна спостерігати, як низка існуючих технологій намагається вирішити частини проблем сегментації та зв'язку, які мікросервіси мають на меті вирішити. SOAP добре описує операції, доступні для даної кінцевої точки, і де їх можна знайти за допомогою WSDL. Теоретично

UDDI є гарним кроком на шляху до реклами того, що може зробити послуга і де її можна знайти. Але ці технології були скомпрометовані порівняно складним впровадженням, і, як правило, не застосовуються в нових проєктах. Послуги REST, що базуються, стикаються з тими ж проблемами, і хоча можна використовувати WSDL з REST, це масово не робиться.

Припускаючи, що виявлення є вирішеною проблемою, схема спільного використання та значення між не пов'язаними програмами все ще залишається складною пропозицією для будь-чого іншого, крім мікросервісів та інших систем SOA. Такі технології, як RDFS, OWL та RIF, існують і стандартизовані, але не використовуються зазвичай. JSON-LD та Schema.org пропонують уявлення про те, як виглядає ціла відкрита мережа, що ділиться визначеннями, але це ще не прийнято на великих приватних підприємства.

РОЗДІЛ 2
СЕРВІСИ ПОВІДОМЛЕНЬ ЯК ЗАСІБ ВЗАЄМОДІЇ ІЗ КОРИСТУВАЧЕМ

2.1 Що таке сервіс повідомлень, його можливості та переваги

Сервіс повідомлень – це віртуальний співрозмовник і помічник, іншими словами програма, яка допомагає користувачам з їх потребами. Сервіси повідомлень також називають чат-ботами. Крім того, чат-бот дозволяє керівникам онлайн-проєктів залучати нових клієнтів. Завдяки даній програмі робота на сайтах стає простіша, як творцям проєктів, так і клієнтам. Вони можуть отримувати швидку відповідь на своє питання, дізнаватися про акції та знижки. Завдяки, так званих, скриптів-помічників можна заробляти.

Щоб почати роботу з чат-ботом не обов'язково мати освіту програміста. У цьому розділі ви дізнаєтесь як і де можна застосувати та зустріти чат-бота, нижче будуть наведені приклади сервісів. Також розглянемо як цю програму можна застосувати в онлайн-школах.

Швидше за все, в повсякденному житті Ви вже зустрічалися з сервісами повідомлень: при перегляді соціальних мереж, користуванні месенджерами, пошуку товарів в інтернет-магазині та при перегляді інших веб-сторінок.

Коли Вам приходить повідомлення з пропозицію товару або послуг – це і є віртуальний помічник. Його програмування направлено на допомогу потенційним клієнтам і користувачам сайту. Його допомога полягає в консультуванні, рішенням купити чи ні конкретний товар, оформити замовлення або іноді, просто розважити клієнта.

На цей час сервіси повідомлень стали сучасними і ефективними механізмами, які допомагають розвивати та просувати власний бізнес. Цей віртуальний помічник і співрозмовник, здатний виконувати

величезний асортимент завдань і обов'язків. На сьогодні в інтернеті існує безліч видів чат-ботів:

- Бот-консультант;
- Бот-психолог;
- Бот-агент;
- Бот-юрист;
- Бот-співрозмовник;
- Бот-маркетолог.

Спектр завдань чат-бота розвивається кожен день з величезною швидкістю. Крім вищезазначених функцій, віртуальний помічник, може виконувати розважальні функції, наприклад, на ваше прохання надіслати добірку фільмів або серіалів, надсилати гумористичні відео, розповідати анекдоти.

У своїй роботі даними віртуальними механізмами користуються керівники онлайн-проєктів і онлайн-шкіл. Використовують вони дану програму-робота, тому що вона допомагає залучати до проєкту нових учасників і потенційних клієнтів, надає їм інформацію про переваги компанії або конкретного продукту, відповідає на питання, що цікавлять, допомагає з оформленням замовлення, здійснює платіж. Внаслідок цього, продажи фірми ростуть і клієнтів стає більше.

Але всупереч перевагам, не всі власники онлайн-бізнесу користуються цим сучасним і універсальним інструментом. Попросту, деякі виробники звикли працювати по-старому, тому утримувати операторів кол-центру або наймають менеджерів. Безумовно, це обходиться дорожче, ніж впровадити в систему віртуального помічника. Зрозуміло, що сервіси повідомлень неспроможні повністю замінити людей, але виконати одноманітні та прості завдання в їх силах.

Є прості сервіси повідомлень та більш досконалі. Робота їх полягає на основі вже готового алгоритму. Прості сервіси повідомлень відповідають на питання, які зберігаються в його базі даних. У разі, якщо

заданого питання, немає в його пам'яті, і він не може на нього відповісти. Тому віртуальний помічник перенаправляє клієнта на співробітника кампанії, який допоможе розібратися і відповість на всі запитання.

Високий рівень сервіса повідомлень є штучним інтелектом, даний робот здатний розвиватися самостійно, і його внутрішні механізми влаштовані складніше – від цього і назва сервіс повідомлень. Дані віртуальні помічники зустрічаються дуже рідко, але з усім тим, вони існують.

Інтегрувати роботу чат-бота можна в сайт. Але виходячи з того, що більшість потенційних клієнтів – це користувачі месенджерів та соціальних мереж, ефективніше інтегрувати його роботу саме на подібних платформах.

Сама робота бота, по суті це розмова між потенційним клієнтом і запрограмованим скриптом. Клієнту приходить повідомлення з привітанням віртуального помічника і короткий курс того, чим він може допомогти. Якщо клієнт зацікавився, то може продовжити діалог двома способами:

- Відповідь повідомленням.

  В такому випадку, віртуальний помічник задасть ще ряд навідних запитань або запропонує виконати команди, щоб, безпосередньо. приступити до суті діалогу;

- Здійснення дій на сайті.

  Тобто, подати заявку або записатися на послугу.

2.2 Статистика

Чат-боти викликають все більше інтересу у людей, але з усім тим, дана тенденція і його напрямки тільки розвиваються в нашій країні. За кордоном активно користуються віртуальними помічниками та ведуться більш масштабні дослідження даної області.

Іноземна статистика:

- 40% власників кампаній застосовують кожен день роботу чат-ботів;

- 35% потенційних клієнтів просять деякі кампанії внести у свою програму чат-ботів (мова йде про тих, хто ще цього не зробив);
- 55% потенційних клієнтів або користувачів сайту охочіше ведуть діалог з віртуальним помічником, ніж з реальним консультантом. Адже якщо у клієнта просте питання, чат-бот відповість запросто на нього, що скоротить час очікування відповіді.
- 21% потенційних клієнтів знаходять, що спілкування з віртуальним помічником більш зручне, ніж спілкування з представником будь-якої кампанії.

2.3 Переваги чат-бота

Як і згадувалося раніше, власники бізнесу, які використовують універсального віртуального співробітника підвищуючи можливість просування їхньої фірми, оптимізації та підвищення продажів.

Завдяки розвитку технологій впровадження чат-ботів в роботу стало більш ніж реальним в наш час. Розробкою віртуальних помічників займаються фахівці, їхня професія називається «архітектор чат-ботів», якій якраз, можна навчитися онлайн. При цьому високі знання і навички програмування зовсім не потрібні, навчитися даній професії можна з нуля.

Статистика показує, що більшу частину часу клієнти проводять саме в соціальних мережах і месенджерах, тому більш резонно, запускати скрипт саме там. На цей час, найчастіше можна зустріти бота в Telegram і Viber, рідше в WhatsApp. Саме тут клієнти швидше реагують на пропозиції, ніж онлайн.

2.4 Плюси сервісів повідомлень

Плюси сервісу повідомлень:

1. Для просування віртуального помічника можна використовувати абсолютно будь-який пристрій;
2. Швидка обробка запитів клієнтів;

3. Цілодобовий зв'язок – це є їх перевагою перед звичайними менеджерами і консультантами, які зазвичай працюють за графіком;
4. Обробка повідомлень автоматизована, що зводить допуск помилок, буквально, до мінімуму. Якщо враховувати людський фактор – помилки будуть виникати частіше, внаслідок цього інформація буде спотворюватися;
5. Економія бюджету і витрат власника бізнесу. Для впровадження в систему віртуального помічника потрібно разова витрата. А ось, реальним співробітникам доведеться оплачувати роботу щомісяця, що підвищує витрати фірми;
6. Економія часу внаслідок автоматизації відповідей. Скрипт працює на вхідні заявки набагато швидше, ніж на ручній обробці інформації;
7. Мінімальне навантаження на персонал. Односкладову і монотонну роботу можна перемикнути на віртуального помічника. Це, природно, полегшить роботу співробітників, та й підвищить якість роботи фірми в цілому.

2.5 Як користуватися чат-ботом?

Розглянемо можливі і найпопулярніші варіанти, як можна використовувати чат-бот.

Серед кампаній цей варіант є найпоширенішим, тому він стоїть на першому місці. Наприклад, такою моделлю користується велика мережа «Київстарт».

На ділі робота бота виглядає досить просто: віртуальний помічник надсилає повідомлення з привітанням клієнту, після чого просити оцінити якість товару або послуги, а також, роботу персоналу. У відповідь клієнт може поставити питання щодо акцій і появи нових опцій або товарів. Чат-бот надає потрібну клієнту інформацію і дякує йому. Таким чином фірма робить аналітику і покращує якість кампанії.

Багато фірм привертають увагу проводячи різні конкурси, тим самим, збільшуючи потік клієнтів. І ті люди, які хочуть випробувати свою удачу беруть участь в даному заході. Наприклад, «PepsiCo» організовувала багаторівневі квести, щоб перегнати трафік клієнтів на новий продукт «Хрусteam». І віртуальний помічник спілкувався з учасниками квесту від імені Шерлока, ставив їм цікаві питання і ділився таємницею рецептури. Клієнтам було цікаво брати участь в подібному конкурсі, адже вони дізнавалися більше про фірму та її продуктах. А тим самим у кампанії піднялися продажу. Це було ідеальне поєднання конкурсу та впровадження віртуального помічника, в якого були запрограмовані цікаві сценарії.

Віртуальний помічник проводить листування з клієнтами від початку знайомства до кінця надання послуги або замовлення товару. Механізм роботи чат-бота підігріває інтерес покупця і підштовхує його на потрібне рішення. У тому випадку, якщо клієнт починає сумніватися у виборі або придбанні товару, то чат-бот допоможе розвіяти сумніви та допоможе з нестачею інформації. Адже, чат-бот універсальний консультант, який буде описувати всі переваги товару своєї фірми. І замовити товар з яким визначився клієнт буде простіше, адже зробити це можна в пару кліків, завдяки віртуальному помічнику.

Для опису даної моделі, можна взяти в приклад основу роботи віртуального помічника «Яндекс». Їх чат-бот надає користувачам безліч опцій, він дає розважальну програму, але на тлі з цим надає важливі новини, і розповідає про нові продукти даної кампанії. Крім цього, віртуальний помічник надсилає своїм користувачам курс валют, прогноз погоди та багато іншого, що необхідно для комфортної роботи з ним.

Але не варто думати, що тільки IT-сфера використовує віртуальних помічників. Зараз їх впроваджують і торгові великі мережі, банки, онлайн-проєкти, таксі та ресторани.

Чат-бот створений не тільки щоб розважати людей, хоча ця функція немало важлива, сама по собі розважальна функція має бути присутня в будь-якому проєкту. Бо часом користувачеві хочеться просто відірватися від проблем і повсякденної рутини, і розслабитися у вільну хвилинку. І саме в цьому випадку, клієнту допоможе віртуальний помічник з розважальним алгоритмом. В такому випадку у клієнта пропадає скептичний настрій, а піднімається настрій, внаслідок чого він буде глибше залучений в процес і швидше оформить замовлення.

2.6 Чат-бот в онлайн-школах

На цей час багато керівників впроваджують в процес навчання чат-ботів, це допомагає знаходити нових учнів і в майбутньому далі взаємодіяти з ними, в їх навчанні.

Віртуального помічника можна під'єднати до аудиторії на самому початковому етапі в процесі знайомства. Чат-бот буде вести знайомство з майбутніми підручниками, розповідати про переваги їх курсу і відповідати на їхні запитання. На цьому етапі чат-боти більш ніж ефективні, тут вони грають роль візитки.

Новини. У процесі навчання за допомогою чат-бота учні будуть дізнаватися про зміну графіка, часу початку ефірів і в загальному, про останні новинах онлайн-школи.

Віртуальні помічники можуть просувати окремі товари школи своїм учня, наприклад, пропонувати купити книгу експерта, інтелектуальні картки, і інші матеріалу курсу.

Платежі. Віртуальний помічник допоможе здійснити оплату за навчання, через нього це навіть буде швидше.

В онлайн-школі є можливість запустити конкурс або квест з різними рівнями. І в якості призу надати знижку на курс.

Чат-бот допомагає власникам даних шкіл заявити людям про себе, і допомогти знайти нових учнів, прихильників, збільшити продажі та утримати вже наявну аудиторію. Тому багато хто радить використовувати

їх в таких проєкт. Віртуальні помічники зроблять процес роботи простіше.

2.7 Як створити власний чат-бот?

Для цього фахівці використовують різні сервіси для консультування спамерських пошукових роботів. Чат-боту потрібно задати потрібний алгоритм роботи, за яким він буде далі існувати. Як і говорилося раніше, алгоритми є прості та складні. Тому для їх створення використовуються абсолютно різні конструктори та механізми створення. Виконавець починає створювати ланцюг майбутнього скрипта, після додає в неї всю потрібну для роботи інформацію, яка буде донесена до користувачів в строгому порядку. Іншими словами, розробник заздалегідь продумує весь діалог для потенційних клієнтів. І, наприклад, якщо клієнт на повідомлення бота відповість «Так» – бот надішле один готовий текст з відповіддю, якщо «Ні» – то інший. Приклад більш ніж перебільшений, але суть діалогу з ботом і людиною зрозуміла.

2.8 Сервіси які часто використовують власники онлайн-шкіл

Senler – сервіс, який активно застосовується для розсилки окремих повідомлень для цільової аудиторії в «Facebook». У даного сервісу є пробний період, який дозволить зрозуміти, чи підходить вам дана програма чи ні.

Автопілот – сервіс дозволяє автоматизувати навчання, і зібрати ігрових ботів в онлайн-школах. Основною перевагою є автоматизація та інтеграція найпопулярніших сервісів, можливість створення і користування на їх основах чат-бота. Є можливість створити чат-бот в соціальній мережі та месенджері, наприклад, у Facebook і Telegram.

SaleBot – в даному конструкторі зібрані найпопулярніші та доступні чат-боти, які з легкістю можна впроваджувати в соціальні мережі. На ньому можна формувати як складні, так і прості продажі, автоматизувати спілкування з клієнтами.

ActiveUsers – цей конструктор дозволяє збирати розважальних чат-ботів, а також, придумувати інтелектуальні конкурси та завдання. Сервіс досить простий, тому багато проєктів можна на ньому реалізувати. Активним користувачам нараховуються бали, які можна надалі перевести в знижку на даному курсі. На сервісі можна підключати не тільки гейміфікацію, але і розсилку клієнтам, стежити за продажами та оплатою. Чат-бот здатний сам підтверджувати та розглядати заявки клієнтів.

Сервісів і проєктів для створення віртуальних помічників велика кількість, але вище представлені найпопулярніші, якими користується переважна більшість онлайн-шкіл.

2.9 Поради для користування і створення чат-бота

Завдяки просуванню в маси чат-ботів вони змогли витіснити багатьох консультантів, операторів і менеджерів. Цілі у віртуальних помічників можуть бути різними, але ось загальні рекомендації на цей рахунок – чітко і грамотно позначте мети та завдання. На першому етапі конструктора, потрібно відповідально підійти до формулювання майбутніх обов'язків віртуального помічника. Чат-бот повинен з точністю вирішити поставлені перед ним завдання, тобто:

- вивчити приблизно потенційну базу з клієнтами, а так само її запити;
- залучити нових користувачів і клієнтів, і утримати вже наявні аудиторії
- намагатися підвищити рівень продажів
- високоякісно консультувати;
- допомагати при виборі товару або послуги, а після допомагати в оплаті;
- робити розсилку і повідомляти про новини;
- здійснити розважальний зміст.

Не забувайте, що є люди, які не часто користуються соціальними мережами, і не звертають увагу на розсилку, навіть якщо вони у них і

встановлені. Такі люди вважають за краще отримувати інформацію через телефон від операторів і менеджерів через технічну підтримку. Тому поставтесь серйозно до першого повідомленням, яке чат-бот буде надсилати клієнту. Перше повідомлення повинно бути максимально інформативним і містити максимум інформації про фірму і їх товар. Тобто, клієнт не повинен бути в замішанні «Що робити далі, куди натискати?». Все повинно бути більш ніж зрозуміло і доступно.

Хороший бот зобов'язаний відповідати на питання, що цікавлять клієнтів. Тому треба детально проєктувати чат-бот.

Є критерії, яким бот зобов'язаний відповідати для ідеальної роботи:

- діалог повинен починатися зі слів «хто», «де», «що», «чому» і «коли»;
- давати клієнту відповідь на вибір;
- формулювати питання так, щоб кінцевою відповіддю було «Так» або «Ні»;
- при використанні опитувань і діалогу, розробнику не варто брати за основу складні речення, адже в такому випадку, у віртуального помічника відбудеться збій у програмі. Тобто, клієнт може відповісти на питання так, що бот просто буде неспроможний обробити цю інформацію;
- уточняйте всі дані. Такий запобіжний захід допоможе дотримуватися потрібної стратегії діалогу і тоді, візуальний помічник не зіб'ється з курсу і буде слідувати сценарію з його пам'яті. Але якщо ж чат-бот не зміг відповісти на питання клієнта, він повинен буде перенаправити людини з питанням в техпідтримку для зв'язку з оператором або менеджером;
- Використовуйте активні клавіші. В такому випадку обробка відповіді буде знижена до мінімуму.

Атмосфера в діалозі повинна бути максимально ввічливою і доброзичливою. Безумовно відвідувачі знатимуть, що по той бік нікого

немає, і з ними розмовляє віртуальний помічник, але все ж, їм буде приємно отримувати позитивні відповіді.

Не навантажуйте чат-бота. Не варто навантажувати віртуального помічника складними та багатоповерховими завданнями. Все повинно бути легко і просто, щоб відвідувач розумів що від нього хочуть з півслова. Не робіть помилку покладаючи всю роботу на віртуального співробітника вашої кампанії. Він всього лише видає вже запрограмовані відповіді в його системі. Якщо хочете отримати складного бота, краще звернутися до фахівця.

РОЗДІЛ 3
ТЕХНОЛОГІЇ ТА РОЗРОБКА СЕРВІСУ ПОВІДОМЛЕНЬ

3.1 Технології та інструментарій

Для розробки сервісу повідомлень необхідно спершу визначити технології та інструментарій:

1. Linux Ubuntu версії 18.04 lts
2. Docker
3. Process Manager 2
4. JavaScript (Node.js)
5. MySQL
6. Telegram Bot API

Для розробки було обрано операційну систему Linux Ubuntu версії 18.04 lts. Lts означає, що версія підтримується оновленнями, стабільна і готова для розробки. Ubuntu – це безкоштовна операційна система з дружнім інтерфейсом, заснована на базі Debian і використовує можливості ядра Linux. Операційна система надає різноманітні можливості користувачу: розваги, навчання, робота, інтернет-серфінг. З плюсів можна виділити: мінімальну вірусну загрозу та безпеку, дуже зручний інтерфейс, безкоштовність, підтримка будь-якого обладнання, якісне резервне копіювання.

Docker — інструментарій для створення та управління ізольованими Linux-контейнерами. Він керує контейнерами з можливістю ізоляції окремих процесів, запускає довільні процеси в режимі ізоляції, що дозволяє переносити та дублювати створенні для цих процесів контейнери на інші сервери, виконуючи всю роботу зі створення та підтримки контейнерів. З плюсів можна виділити: швидкий процес розробки, зручна інкапсуляція, однакова поведінка на локальній машині та серверах, простий та зрозумілий моніторинг, зручне масштабування.

Process Manager 2 – це менеджер процесів із відкритим вихідним кодом Node.js, який допомагає розробникам керувати програмами Node.js. Ключові особливості менеджера процесів є автоматичне збалансування навантаження додатка, декларативна конфігурація додатка, зручна система розгортання та моніторингу.

JavaScript – це мова програмування, основними архітектурними рисами якої є: самостійний розподіл пам'яті, типізація, прототипне програмування, функції як об'єкти класу.

MySQL – це найпопулярніша безкоштовна система управління базами даних, яка дозволяє отримувати, додавати та обробляти дані. Система підтримується майже на всіх платформах, встановлюється як на сервері, працюючим під операційною системою Windows, так і на сервері Linux.

Telegram Bot API — це інструментарій на основі HTTP, створений для роботи з ботами. Одна з найцікавіших можливостей Telegram Bot API - кастомізація клавіатури. При передачі сервером відповіді є можливість передати команду на відображення спеціальної клавіатури з запрограмованими варіантами відповіді. Клієнт Telegram, отримавши повідомлення, відобразить користувачеві вашу клавіатуру. Натискання на клавішу відразу ж відправить на сервер відповідну команду. Таким чином можна дуже спростити взаємодію робота з користувачем.

3.2 Архітектура та взаємодія між компонентами

Схематично моя програма має такий вигляд:

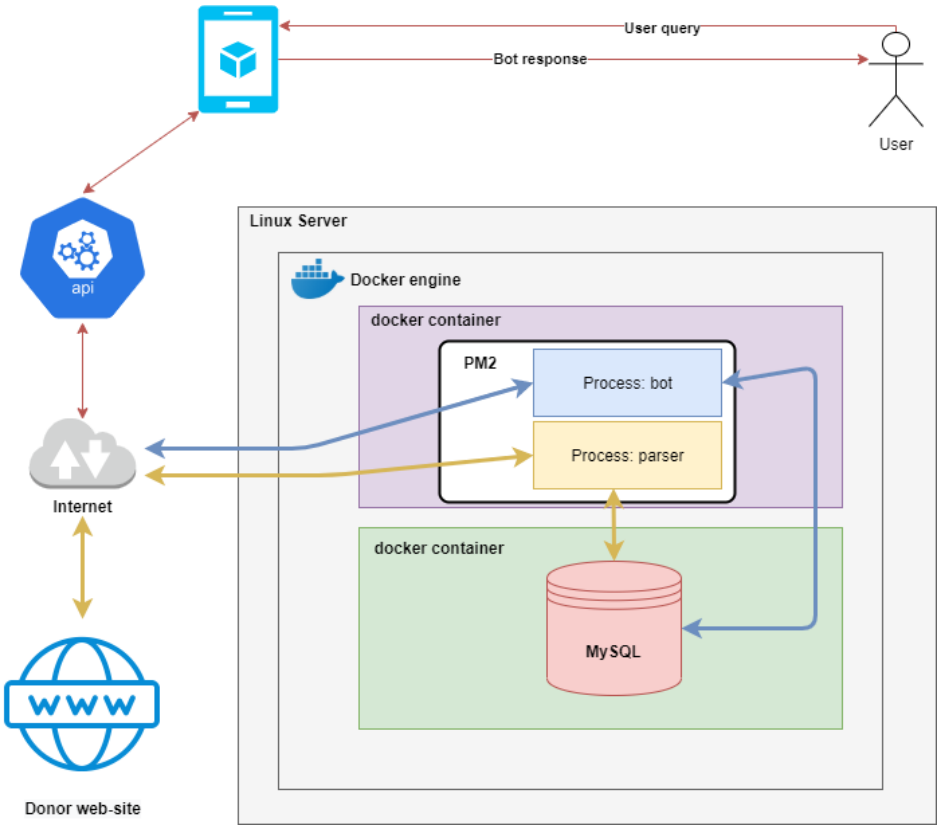

Рисунок. 3.1 - Схематичне представлення програми.

На операційній системі Linux Ubuntu версії 18.04 lts розгортається Docker і створює декілька ізольованих контейнерів. В одному із контейнерів буде міститись база даних MySQL, в іншому - Process Manager 2. Process Manager 2 запустить два процеси, які буде постійно підтримувати. Перший процес буде парсить данні з сайта-донора з відкритими даними в базу даних MySQL та придавати їм потрібний вигляд. Другий процес – сам бот, який буде брати данні з бази даних, зв’язуватись через API, передавати інформацію користувачу, реагувати на запрограмовані команди.

3.3 Сценарії для сервісу повідомлень.

Перший сценарій сервісу повідомлень:

- Збирати дані з цінами комунальних послуг.
- Використовуючи статистичні функції розраховувати зміни за місяць або рік.
- За потребою користувача показувати ціну.

- Відсилати самостійно повідомлення з ціною, якщо ціна значно змінилась .
- Використовуючи ціни та показники користувача, порахувати ціну за місяць.

Другий сценарій сервісу повідомлень:

- Збирати дані про погоду.
- Оновлювати та записувати дані кожні декілька годин.
- За потребою користувача вивести погоду.
- Дати можливість вивести погоду з різних областей.
- Відсилати самостійно повідомлення з попередженням при небезпечних фіксаціях погоди.

Розроблена мікросервісна архітектура є універсальною, тому однаково підходить для втілення цих або інших сценаріїв. Щоб наглядно це показати був розроблений сервіс повідомлень за другим сценарієм.

3.4 Розробка сервісу

Розроблюваний погодній сервіс для платформи Телеграм буде містити та оперувати інформацією про погодні умови на різних областях території України.

За допомогою PM2 запускається два процеса з скриптами. Скрипти написані на JavaScript і запускаються на бекенді під NodeJs. Перший скрипт збирає дані на сайті та зберігає в базу даних. PM2 запускає скрипт кожні 3 години.

```
const getPageHtml = function (url) {
    return new Promise( executor: resolve => {
        request( url: {
            url: url,
            method: 'GET',
            headers: {'Cache-Control' : 'no-cache'},
        }, options: function(error, response, data) {
            resolve(data);
        });
    });
};

const parseHtmlNow = function (html) {
    return new Promise( executor: resolve => {
        let $ = cheerio.load(html);
        let $page = $('.forecast_wrap');
        let result = [];
        result.text = $('.tooltip').data('text');
        result.temperature = $($('.tooltip').get(0))
            .find('.unit_temperature_c .tab-weather__value_l')
            .text()
            .replace(/[\n\s]/g, '');
        result.windSpeed = $($('.forecast_wrap .now__info .nowinfo__item_wind .nowinfo__value').get(0))
            .text()
            .replace(/[\n\s]/g, '');
        result.pressure = $($('.forecast_wrap .now__info .nowinfo__item_pressure .nowinfo__value').get(0)).text();
        result.humidity = $($('.forecast_wrap .now__info .nowinfo__item_humidity .nowinfo__value').get(0)).text();
        result.water = $($('.forecast_wrap .now__info .nowinfo__item_water .nowinfo__value').get(0)).text();
        resolve(result);
    });
};
```

Рисунок 3.2 - Деякі функції.

GetPageHtml – за допомогою цієї функції ми можемо отримати html код із сайта донора для наступного парсингу.

ParseHtmlNow – функція парсує html код отриманий попередньой функцією. В результаті ми отримуємо об'єкт з даними, перетворюємо об'єкт приємливий вигляд і відправляєм користувачеві.

Другий скрипт – це сам бот, написаний на Telegram Bot API. Тут буде описаний інтерфейс, з‘єднання з платформою Telegram та функції для виклику повідомлень.

Моя база даних складається з двох таблиць. Перша таблиця – Users.

```
CREATE TABLE `users` (
    `id` int(11) NOT NULL AUTO_INCREMENT,
    `tid` int(11) DEFAULT NULL,
    `is_bot` tinyint(1) DEFAULT NULL,
    `is_ban` tinyint(1) DEFAULT NULL,
    `is_admin` tinyint(1) DEFAULT NULL,
    `username` varchar(255) COLLATE utf8mb4_unicode_ci DEFAULT NULL,
    `first_name` varchar(255) COLLATE utf8mb4_unicode_ci DEFAULT NULL,
    `language_code` varchar(255) COLLATE utf8mb4_unicode_ci DEFAULT NULL,
    `createdAt` datetime NOT NULL,
    `updatedAt` datetime NOT NULL,
    PRIMARY KEY (`id`)
) ENGINE=InnoDB AUTO_INCREMENT=4 DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_unicode_ci
```

Рисунок 3.3 - Перша таблиця.

В цю таблицю буде записуватись інформація про клієнтів. Розділи в таблиці включають в себе логін, дата початку використання сервісу, унікальний код в базі Телеграма. Унікальний код в базі Телеграма

використовується для розпізнавання користувача після зміни імені. Також створений розділ «Бан», змінивши його значення на 1, бот перестає відповідати клієнту.

Друга таблиця – Data_gismeteo_ru_weathers.

```
CREATE TABLE `data_gismeteo_ru_weathers` (
    `id` int(11) NOT NULL AUTO_INCREMENT,
    `date` date DEFAULT NULL,
    `iso` varchar(5) COLLATE utf8mb4_unicode_ci DEFAULT NULL,
    `day_temperature` int(3) DEFAULT NULL,
    `day_pressure` int(4) DEFAULT NULL,
    `day_cloudiness` enum('clear','partly_cloudy','cloudy','overcast') COLLATE utf8mb4_unicode_ci DEFAULT NULL,
    `day_weather_condition` enum('rain','snow','storm') COLLATE utf8mb4_unicode_ci DEFAULT NULL,
    `day_wind_direction` enum('n','s','e','w','nw','ne','sw','se','calm') COLLATE utf8mb4_unicode_ci DEFAULT NULL,
    `day_wind_speed` int(3) DEFAULT NULL,
    `evening_temperature` int(3) DEFAULT NULL,
    `evening_pressure` int(4) DEFAULT NULL,
    `evening_cloudiness` enum('clear','partly_cloudy','cloudy','overcast') COLLATE utf8mb4_unicode_ci DEFAULT NULL,
    `evening_weather_condition` enum('rain','snow','storm') COLLATE utf8mb4_unicode_ci DEFAULT NULL,
    `evening_wind_direction` enum('n','s','e','w','nw','ne','sw','se','calm') COLLATE utf8mb4_unicode_ci DEFAULT NULL,
    `evening_wind_speed` int(3) DEFAULT NULL,
    PRIMARY KEY (`id`),
    UNIQUE KEY `unique_date_iso` (`date`,`iso`)
) ENGINE=InnoDB AUTO_INCREMENT=116474 DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_unicode_ci
```

Рисунок 3.4 - Друга таблиця.

В цій таблиці буде міститись інформація про погоду. Наприклад, температура, опади, тиск тощо. Дані в цій таблиці оновлюються кожні 3 години.

ВИСНОВКИ

Таким чином, використовуючи ряд джерел інформації, та власні набутті знання, висвітлено питання мікросервісної архітектури. Підсумовуючи матеріал, зазначу, що архітектура мікросервісів – це такий підхід, коли один додаток формується як цілісність маленьких, які є самодостатніми, незалежними, мало пов'язані сервіси, як можуть комунікувати один з одним з підтримкою пластичних механізмів (HTTP, gRPC, AMQP). Такі сервіси найчастіше використовуються для бізнес-потреб (де будь-який відповідальний за визначений хід) та розвиваються самостійно з уживанням цілком налаштованого автоматичного оточення. Також зазначу, що сервіси включають в себе різні мови і можуть користуватися різними технологіями зберігання даних.

Було підтверджено, що за останні роки популярність систем обміну повідомленнями (месенджерів) почала стрімко зростати. Месенджери перетворилися із засобів для спілкування в засоби для отримання інформації та потужний бізнес інструмент.

У цій роботі було описано ключові принципи створення сервісу повідомлень на основі мікросервісної архітектури. На основі описаних принципів розроблено сервіс повідомлень для платформи Телеграм.

Сервіси повідомлень дуже корисні інструменти для власників бізнесу тому, що економлять бюджет фірми, не навантажують інших співробітників і скорочують час очікування клієнтів.

Зараз віртуального помічника впроваджують всюди, завдяки йому можна викликати таксі, замовити додому їжу, отримати консультацію у психолога або юриста, записатися на онлайн-курс. На відміну від простих працівників, чат-бот працює цілодобово і швидко обробляє інформацію.

Також було проаналізовано ряд переваг та мінусів технології мікросервісної архітектури. Висвітлений перехід від моноліту до мікросервісу.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Фаулер М. Архітектура корпоративних програмних додатків : навч. посіб. К.: Вільямс. 544 с.
2. Ньюмен С. Создание микросервисов. – СПб.: Питер, 2016. 304 с.
3. Hohpe G. , Woolf B. Enterprise integration patterns: Designing, building, and deploying messaging solutions. Boston: Addison-Wesley, 2003. 683 p.
4. Richardson C. From Design to Deployment. B.: Floyd Earl Smith, 2016. 80 p.
5. Evans E. Domain-Driven design: Tackling Complexity in the Heart of Software. B.: Addison-Wesley Professional, 2003. 560 p.
6. Lewis J., Fowler M. Microservices. *MartinFowler.com*. 2014. URL: https://martinfowler.com/articles/microservices.html (дата звернення: 02.03.2021).
7. Chris Richardson of Eventuate. Introduction to Microservices. *NGINX*. 2015. URL: https://www.nginx.com/blog/introduction-to-microservices/ (дата звернення: 26.02.2021).
8. Zmerzlyi I. Мікросервісна архітектура. *Medium*. 2019. URL: https://medium.com/@IvanZmerzlyi/microservices-architecture-461687045b3d (дата звернення: 05.03.2021).
9. Павленко А. От простого к сложному: путь от монолита к микросервисам. *DOU*. 2021. URL: https://dou.ua/lenta/columns/from-monolithic-to-microservice-architecture/ (дата звернення: 06.03.2021).
10. Зеленін В. Побудова мікросервісів за допомогою різних програмних платформ. URL: https://cad.kpi.ua/attachments/093_2017p_Зеленін.pdf (дата звернення: 28.02.2021).

11. Sean K. Microservices – Please, don't. *RIAK BLOG*. 2016. URL: https://riak.com/posts/technical/microservices-please-dont/ (дата звернення: 05.03.2021).
12. Корисні боти для Telegram . *ONet*. Бердянськ. 2020. URL: https://onet.zp.ua/корисні-боти-для-telegram/ (дата звернення: 19.03.2021).
13. Сервісно-орієнтована архітектура (SOA). *СтудопедіЯ*. URL: https://studopedia.com.ua/1_151441_servisno-oriientovana-arhitektura-SOA.html (дата звернення: 19.03.2021).
14. An Overview of Service-Oriented Architecture in Retail. *WaybackMachine*. 2007. URL: https://web.archive.org/web/20070701160401/http://msdn2.microsoft.com/en-us/library/bb264584.aspx (дата звернення: 18.03.2021).
15. Syed B. Beginning Node.js. B.: Apress, 2014. 326 p.
16. Limonczenko M. Top 5 Node and Express Books for Beginners in 2021. *BOOKS ON CODE*. URL: https://booksoncode.com/articles/best-node-and-express-books (дата звернення: 17.03.2021).
17. Express web framework (Node.js/JavaScript). *MDN Web Docs*. 2021. URL: https://developer.mozilla.org/uk/docs/Learn/Server-side/Express_Nodejs (дата звернення: 19.03.2021).
18. Фленов М. Linux глазами хакера. БХВ-Петербург, 2016. 432 с.
19. Unix и Linux. Руководство системного администратора . Э. Немет, Г. Снайдер, Т. Хейн, Б. Уэйли ; пер. з англ. Д. Клюшина, Н. Ручко. Вильямс, 2012. 1312 с.
20. Кочер П. С. Микросервисы и контейнери Docker. ДМК Пресс, 2019. 240 с.
21. Goasguen S. Docker Cookbook: Solutions and Examples for Building Distributed Applications. O'Reilly Media, 2015. 366 p.
22. MySQL. URL: https://www.mysql.com (дата звернення: 12.03.2021).

23. MySQL – Свободная реляционная СУБД / Хабр. *Хабр*. URL: https://habr.com/ru/hub/mysql/ (дата звернення: 20.03.2021).
24. Partitioning Problems in Parallel, Pipelined, and Distributed Computing. Bokhari_S. IEEE Transactions Computers, 1988. 57 с.
25. Осипенко А. А. Переход от монолита к микросервисам. Все публикации подряд / Хабр. URL: https://habr.com/ru/post/305826/ (дата звернення: 12.03.2021).
26. Vladimir. Микросервисы (Microservices). Все публикации подряд / Хабр. URL: https://habr.com/ru/post/249183/ (дата звернення: 04.03.2021).
27. Учасники проектів Вікімедіа. Мікросервіси – Вікіпедія. Вікіпедія. URL: https://uk.wikipedia.org/wiki/Мікросервіси (дата звернення: 03.04.2021).
28. Преимущества и недостатки микросервисной архитектуры | Записки программиста. Записки программиста. URL: https://eax.me/micro-service-architecture/ (дата звернення: 12.03.2021).
29. Архитектура микросервисов. Все публикации Хабрахабр, Гиктаймс, Мегамозг... URL: http://itnan.ru/post.php?c=1&p=320962 (дата звернення: 11.03.2021).
30. Давлеткалиев Р. Микросервисы: пожалуйста, не нужно. Все публикации подряд / Хабр. URL: https://habr.com/ru/post/311208/ (дата звернення: 13.03.2021).