МАРІУПОЛЬСЬКИЙ ДЕРЖАВНИЙ УНІВЕРСИТЕТ
ЕКОНОМІКО-ПРАВОВИЙ ФАКУЛЬТЕТ
КАФЕДРА СИСТЕМНОГО АНАЛІЗУ ТА ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ

До захисту допустити:
Завідувач кафедри

_____ Шабельник Т.В.
(підпис) *(ПІБ завідувача кафедри)*
«__ » _____ 2021р.

«УПРАВЛІННЯ ПРОЄКТОМ З РОЗРОБКИ ІТ-ДОДАТКУ СКЛАДСЬКОЇ ДІЯЛЬНОСТІ»

Кваліфікаційна робота
Здобувача вищої освіти другого (магістерського) рівня вищої освіти
Освітньо-професійної програми
«Системний аналіз»
Миронця Владислава Олександровича
Науковий керівник:
Шабельник Т.В., д.е.н., професор
Рецензент:
Левицька Т.О., к.т.н., доцент, доцент кафедри комп'ютерних наук ДВНЗ "ПДТУ"

Кваліфікаційна робота захищена
з оцінкою _____
Секретар ЕК _____
« ___»_____2021 р.

Маріуполь – 2021

МАРІУПОЛЬСЬКИЙ ДЕРЖАВНИЙ УНІВЕРСИТЕТ
ЕКОНОМІКО – ПРАВОВИЙ ФАКУЛЬТЕТ
КАФЕДРА СИСТЕМНОГО АНАЛІЗУ ТА ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ

Рівень вищої освіти «Магістр»
Шифр та назва спеціальності 124 «Системний аналіз»
Освітньо-професійна програма «Системний аналіз»

ЗАТВЕРДЖУЮ
Завідувач кафедри системного аналізу та інформаційних технологій, д.е.н., професор,
_____ Шабельник Т.В.
(підпис) *(ПІБ завідувача кафедри)*
«__ » _____ 2021р.

ПЛАН ВИКОНАННЯ КВАЛІФІКАЦІЙНОЇ РОБОТИ

_____ Миронець Владисла Олександровичь_____
(прізвище, ім'я, по батькові)

1. Тема роботи: Управління проєктом з розробки it-додатку складської діяльності.
керівник роботи Шабельник Тетяна Володимирівна, д.е.н., професор
(прізвище, ім'я, по батькові, науковий ступінь, вчене звання)
затверджені наказом Маріупольського державного університету
від «26» лютого 2021 року №195
2. Строк подання студентом роботи_____
3. Вихідні дані до роботи (мета, об'єкт, предмет)

Метою роботи є удосконалення та розробка системи для управління проєктом з розробки складської діяльності.
Об'єктом дослідження є процеси методолгії для управління розробкою ІТ-додатку.
Предметом дослідження є моделі та методи управліня проєктами з розробки ІТ-додаків.
4. Зміст роботи (перелік питань, які потрібно розробити)

Розділ 1. Теоретичні аспекти укравління проєктами з розробки it додатків.
1.1. Основні поняття теорії управління проєктами.
1.2. Класифікація методів і моделей управління проєктами з «Scrum».
1.3. Аналіз сучасних додатків для управління «Scrum» командою.
Розділ 2. Доцільність розробки додатку для реалізації scrum методології.
2.1. Обгрунтування доцільності розробки додатку.

2.2. Опис функціоналу додатку.
2.3. Опис інформаційної моделі додатку.
Розділ 3 Технічна реалізація.
3.1. Загальна структура автоматизованої системи управління задачами.
3.2. Опис технологій клієнтської частини.
3.3. Вибір фреймворку для створення інтерфейсу користувача.
3.4. Технології для сервера.

5. Консультанти розділів роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		Завдання видав	Завдання прийняв

6. Дата видачі завдання_______

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів кваліфікаційної роботи	Строк виконання етапів роботи	Примітка
1	Аналіз літературних джерел з теми: «Управління проєктом з розробки it-додатку складської діяльності»	05.02 – 11.04.2021	
2	Робота та формування матеріалів параграфу 1.1. Основні поняття теорії управління проєктами	12.04 – 12.05.2021	
3	Робота та формування матеріалів параграфу 1.2. Класифікація методів і моделей управління проєктами з «Scrum»	12.05 – 11.06.2021	
4	Робота та формування матеріалів параграфу 1.3. Аналіз сучасних додатків для управління «Scrum» командою	12.06 – 11.07.2021	
5	Робота та формування матеріалів параграфу 2.1. Обгрунтування доцільності розробки додатку	11.07 – 30.07.2021	
6	Робота та формування матеріалів параграфу 2.2. Опис функціоналу додатку	01.08 – 15.08.2021	
7	Робота та формування матеріалів параграфу 2.3. Опис інформаційної моделі додатку	16.08 – 30.08.2021	

8	Робота та формування матеріалів параграфу 3.1. Загальна структура автоматизованої системи управління задачами	01.09 – 25.09.2021	
9	Робота та формування матеріалів параграфу 3.2. Опис технологій клієнтської частини	24.10 – 10.10.2021	
10	Робота та формування матеріалів параграфу 3.3. Вибір фреймворку для створення інтерфейсу користувача	10.09 - 18.09.2021	
11	Робота та формування матеріалів параграфу 3.4. Технології для сервера	01.11 – 04.11.2021	
12	Формування висновків кваліфікаційної роботи	05.11 – 10.11.2021	
13	Оформлення кваліфікаційної роботи	10.11 – 15.11.2021	
14	Підготовка доповіді та презентації	15.11 – 25.11.2021	

Здобувач_____ __Миронець В.О.___
(підпис) (прізвище та ініціали)

Науковий керівник роботи_____ __Шаьельник Т.В.__
(підпис) (прізвище та ініціали)

ЗМІСТ

ВСТУП

Актуальність теми. Неефективний розподіл товару та його обліку може призвести до щомісячних втрат компаній продавців чи складських посередників. Це не критично, коли йде партійний облік, у якому кожна нова партія зберігається окремо від вже існуючого на складі товару. На нову партію заводиться картка обліку із зазначенням товару, його кількості, дати надходження. Але цей метод використовується у випадках, коли на склад надходить один вид продукції, а у сучасному світі інтернет торгівлі потреба в таких складах незначна. Тому використовують сортовий метод складського обліку, де весь товар зберігається разом та контролюється унікальним кодом, штрих кодом або Qr-кодом, або обома одночасно. У такому підході організації дуже великий ризик фактору людської помилки, тому на зміну людям приходить програмне забезпечення, яке повністю контролює оборот, умови зберігання, розташування товару на складі.

Такі компанії як ID Logistics, X5 Retail Group, Amazon, і такі Українські компанії як Rozetka, Olx, мають програмне забезпечення для повного контролю товаром, людина потрібна тільки щоб відсканувати товар та розташувати його згідно інструкції, які пропонує алгоритм програми. І перехід багатьох підприємств на такі методи організації складського обліку свідчить про його ефективність.

Необхідність розробки додатку для управління складською діяльністю є дуже важливим у сучасному світі так як керування і ведення обліку товару без автоматизованої системи є повільним і неефективним. Хмарні сервіси надають можливість інтегрувати необмежену кількість частин системи і використовувати їх в любому куточку світу, де є інтернет, що дозволяє вмить бачити і керувати логістикою підприємства. Такі компанії як X5 Retail Group, Amazon повністю перейшли на автоматизовану систему управління діяльності складу, що підвищили їх ефективність обробки товару від отримання на склад та відправки кінцевому отримувачу на 70%.

Питаннями управління проєктами займалися такі закордонні та вітчизняні вчені як: Кен Швабер, Джефф Сазерленд, Корі Когон, Сьюзетт Блейкмор, Джеймс Вуд, Скотт Беркун, Кулініч Т.В., Болібрух Л.І.

Питання розробки проєктів ІТ-сфери з використанням Scrum досліджувались у роботах Scrum in Hardware, Scrum Guides Sutherland, застосування підходу Scrum на великих промислових підприємствах.

Виходячи з підтвердженої актуальності теми роботи, можна сформулювати мету.

Метою роботи є управління проєктом з розробки ІТ-додатку складської діяльності з відкритим кодом для адаптації «Scrum» методології під потреби компанії.

Для досягнення поставленої мети в роботі було поставлено та вирішено наступні завдання:

- визначено основні поняття теорії управління проєктами;
- виділено переваги та недоліки існуючих підходів моделювання системи розробки додатку та середовищ для моделювання проєктів;
- обґрунтовано вибір методу розробки проєкту з підходом Agile&Scrum;
- обґрунтовано вибір технологій для розробки ІТ-додатку складської діяльності;
- систематизовано вимоги від власників бізнес-процесів складської діяльності;
- обґрунтовано вибір технологій для розробки концепту ІТ-додатку складської діяльності для адаптації методології;
- розроблено та описано архітектуру ІТ-додатку складської діяльності;
- розроблено структуру документів для підтримки і розробки ІТ-додатку складської діяльності.

Об’єктом дослідження є процеси управління проєктами з розробки ІТ-додатків складської діяльності.

Предметом дослідження є методи і моделі управління проєктами з розробки IT-додатків складської діяльності, програмний інструментарій.

Методи дослідження. Методологічну основу складають методи системного аналізу, прийняття рішень, управління проєктами, логістики та адаптивного управління, методи побудови діаграми Гантта та Pert-діаграми, методи побудови та аналізу критичного шляху робіт, «Scrum» методологія, функціональне програмування з використанням ReactJs, TypeScript, Reduxjs та хмарні сервіси.

Інформаційну базу дослідження складають законодавчі та нормативні документи, що регулюють діяльність підприємств в Україні, дані окремих підприємств (ID Logistics, X5 Retail Group, Amazon, Rozetka, Olx), монографічна література (наукові статті вчених, монографії, матеріали науково-практичних конференцій).

Наукова новизна одержаних результатів роботи полягає в розвитку методів управління проєктами з використанням сучасних інформаційних технологій та програмного інструментарію.

Публікації. За результатами наукової роботи опубліковано 2 тези доповідей у збірниках матеріалів наукових конференцій та декади студнауки Маріупольського державного університету.

Структура та обсяг кваліфікаційної роботи. Робота складається зі вступу, трьох розділів, висновків, списку використаних джерел, який налічує 50 найменуваннь, 1 таблиця та 14 рисунків. Загальний обсяг роботи становить 70 сторінок, з яких основний текст – 61 сторінок.

РОЗДІЛ 1 ТЕОРЕТИЧНІ АСПЕКТИ УПРАВЛІННЯ ПРОЄКТАМИ З РОЗРОБКИ ІТ-ДОДАТКІВ

1.1. Основні поняття теорії управління проєктами

Управління проєктами – галузь яка базується на планувані, управлінні енергоресурсами та іншими видами ресурсів з цілью благополучного виконання завдань і завершення надзавдань проєкту. Часом ототожнюється з веденням програм, але телепрограма - це фактично може бути вищий рівень та як підгрупа зв’язаних і взаємозалежних проєктів [1].

Проєкт – це механізм з обмеженими часовими рамками, який має певний початок і фінал, традиційно обмежений числом, але, наприклад, може обмеження можуть встановлюватись спонсорством або набуттям показників, який виконується для розробки ексклюзивних цілей і завдань, як правило, щоб привести до виграшних поліпшень та створення додаткової вартості [3]. Сутність тимчасових проєктів дисонує з бізнес процесами, які характеризуються моделю з повтореннями, постійним або частково постійним видом діяльності по виготовленню товарів та надання послуг. У реаліях риночної економіки , управління вищезазначеними системами часто залежить від ринку який стрімко розвивається і таким чином вимагає розвитку спеціальних навичок в технічній галузі і застосування розподіленого ведення ними.

Основною метою ведення проєкту є досягнення усіх цілей і виконання задач проєкту, також виконання обов'язків по заздалегідь визначеним цілям проєкту [5]. Характерними обмеженнями маються на увазі кордони і такі фактори як зміст проєкту, часові рамки, бюджет. Другорядним цілями управління , але найбільш прибутковими, є оптимізація, розподіл і адаптація завдач, потрібних для здійснення заздалегідь конкретних цілей [8].

Існує декілька підходів для управління проєктними, гнучкі, інтерактивні, послідовні, розподілу на етапи [5,6,8].

Незважаючи на обраний підхід, необхідно дуже уважно спланувати та проаналзувати основні цілі проєкту, календарний план, витрати, та встановити ролі та відповідальність усіх виконавців та зацікавлених сторін.

Отже, розглянемо методології управління детально.

Модель водоспаду (англ. Waterfall Model).

Модель водоспаду є першою моделлю процесу, яка була представлена. Модель також називають лінійно-послідовною моделью життєвого циклу. Він дуже простий для розуміння та використання. Модель Waterfall - це найдавніший підхід використовувався у розробці програмного забезпечення. Модель показує процес розробки у лінійній послідовності. Це означає, що будь -яка фаза процесу розвитку починається лише за умови, що попередня фаза завершена. У цій моделі водоспаду фази не перекриваються [7].

Переваги освоєння водоспаду полягають у тому, що він дозволяє проводити департаменти та контроль. Можна встановити графік із зазначенням термінів виконання кожної стадії розробки та продукту. Можуть проходити по етапах модельного процесу розробки. Розробка рухається від концепції до проектування, впровадження, тестування, встановлення, усунення несправностей і закінчується експлуатацією та обслуговуванням. Кожен етап розвитку протікає в строгому порядку.

Деякі з основних сильних сторін моделі водоспаду наступні:

1. Чітко визначені етапи та цілі.
2. Добре зрозумілі етапи процесів.
3. Легко організовувати завдання.
4. Процес та основні задачі добре задокументовані.
5. Простий і легкий для розуміння та використання.
6. Легкий в управлінні завдяки жорсткості моделі.
7. Кожен етап має свою специфіку, результати та процес перегляду.
8. Фази обробляються та завершуються одна за одною.

9. Добре працює для маленьких проектів, вимоги дуже чіткі і зрозумілі.

Але поряд з перевагами існують і певні недоліки. Недоліком освоєння водоспаду є те, що він не допускає великого відображення або перегляд. Після того, як заявка перебуває на стадії тестування, дуже важко повернутися назад і змінити те, що не було добре задокументовано або продумано на стадії концепції.

Основні недоліки моделі водоспаду наступні:

- Робочий продукт не виробляється до кінця життєвого циклу.
- Великі ризики та невизначеність.
- Не найкраща модель для складних та об’єктно-орієнтованих проектів.
- Погана модель для довгих та поточних проектів.
- Не підходить для проектів, якщо вимоги мають середній та високий ризик змінюватись. Отже, ризик та невизначеність є високими у цій моделі процесу.
- Важко оцінювати прогрес на етапах.
- Не може відповідати зміненим вимогам.
- Регулювання масштабу протягом життєвого циклу може завершити проект.
- Інтеграція здійснюється як "великий вибух" в самому кінці, що не дозволяє ідентифікувати будь -які технологічні чи бізнес-вузькі місця або проблеми на ранніх стадіях.

Ітеративна модель (англ. Iterative Model).

Ітераційний процес починається з простої реалізації підмножини програмного забезпечення. Вимоги і ітерації покращують розвиток версії, поки вся система не буде реалізована. На кожній ітерації вносяться зміни в дизайн і додається новий функціонал. Основна ідея цього методу - розробити систему через повторювані цикли (ітераційні) і меншими порціями за раз (інкрементальні). Ітеративна і інкрементна розробка - це комбінація ітеративного проектування або ітераційний метод і модель інкрементальної

збірки для розробки. Під час розробки програмного забезпечення більше однієї ітерації циклу може бути декілька процесів в той же час. Цей процес можна описати як «еволюційне придбання» або підхід «інкрементальної збірки» [7].

У цій інкрементальній моделі всі вимоги розділені на різні збірки. Протягом на кожній ітерації модуль розробки перевіряє вимоги, дизайн, фази впровадження і тестування. Кожен наступний випуск модуля додає функцію до попереднього випуску. Процес триває до тих пір, поки вся система не буде готова відповідно до вимог.

Ключ до успішного використання ітераційного життєвого циклу розробки програмного забезпечення - це строгість. перевірка вимог, а також перевірка і тестування кожної версії програмного забезпечення проти цих вимог в рамках кожного циклу моделі. У міру розвитку програмного забезпечення через послідовні цикли тести повинні повторюватися і розширюватися для перевірки кожної версії програмного забезпечення. Перевага цієї моделі в тому, що на дуже ранній стадії з'являється працююча модель системи, стадія розробки, що спрощує пошук функціональних або конструктивних недоліків. Знахідка- проблеми на ранній стадії розробки дозволяє вживати заходів в обмеженому бюджеті.

Недоліком цієї моделі є те, що вона може бути застосована тільки для великих і громіздких проектів розробки програмного забезпечення. Це тому, що невелику програмну систему складно зламати на подальші невеликі справні збільшені модулі.

Переваги:

- Деякі робочі функції можна розробити швидко і на ранніх етапах життєвого циклу.
- Результати виходять завчасно і періодично.
- Паралельний розвиток може бути заплановано.
- Прогрес можна виміряти.
- Менш затратна зміна обсягу та вимог.

- Легке тестування і налагодження під час невеликої ітерації.
- Ризики виявляються і усуваються під час ітерації; і кожна ітерація легко керована віха.

Недоліки:

- Можуть знадобитися додаткові ресурси.
- Хоча вартість зміни менша, але він не дуже підходить для зміни
- вимоги.
- Потрібно більше уваги з боку керівництва.
- Можуть виникнути проблеми з архітектурою чи дизайном системи, тому що не всі вимоги збираються на початку всього життєвого циклу.
- Визначення збільшень може зажадати визначення всієї системи.
- Не підходить для невеликих проектів.
- Складність управління більша.
- Кінець проекту може бути невідомий, що є ризиком.
- Для аналізу ризиків потрібні висококваліфіковані ресурси.
- Прогрес проектів багато в чому залежить від фази аналізу ризиків.

«Agile&Scrum» модель.

Agile заснований на адаптивних методах розробки програмного забезпечення, тоді як традиційні моделі, такі як модель водоспаду, засновані на прогнозному підході. Групи прогнозування в традиційних моделях SDLC зазвичай працюють з докладним плануванням і мають повний прогноз конкретних завдань і функцій, які будуть реалізовані в найближчі кілька місяців або протягом життєвого циклу продукту. Методи прогнозування повністю залежать від аналізу вимог і планування, виконаного на початку циклу. Будь-які зміни, що вносяться, проходять строгий контроль зміну правління і розстановки пріоритетів [7].

Agile використовує адаптивний підхід, при якому немає докладного планування і є ясність на майбутні завдання тільки щодо того, які функції необхідно розробити. Особливість Scrum це - керована розробка, і команда адаптується до мінливих вимог до продукта динамічно. Продукт тестується

дуже часто через ітерації випуску, мінімізацію ризику серйозних збоїв в майбутньому.

Взаємодія з клієнтами - основа цієї гнучкої методології та відкрите спілкування з мінімумом документації - типові особливості Agile середовища розробки. Agile-команди працюють в тісній співпраці і найчастіше розташовані в одному і тому ж географічному місці.

Останнім часом в світі програмного забезпечення широко застосовуються гнучкі методи. Однак цей метод не завжди може підходити для всіх продуктів. Ось деякі плюси і мінуси гнучкої моделі.

Переваги:

- Це дуже реалістичний підхід до розробки програмного забезпечення.
- Сприяє спільній роботі і перехресному навчанню.
- Функціональність може бути швидко розвинена і продемонстрована.
- Вимоги до ресурсів мінімальні.
- Підходить для фіксованих вимог, або вимог які змінюються.
- Надає ранні часткові робочі рішення.
- Хороша модель для постійно мінливого середовища.
- Мінімальні правила, зручна документація.
- Забезпечує паралельну розробку і поставку в рамках загального запланованого контексту.
- Планування практично не потрібно.
- Легко керувати.
- Надає розробникам гнучкість**.**

Недоліки:

- Не підходить для обробки складних залежностей.
- Підвищений ризик стійкості, ремонтопридатності і розширюваності.

- Загальний план, гнучкий лідер і гнучка практика управління проектами - необхідність, без якої все не працює.
- Суворе управління доставкою диктує обсяг, що надають функціональні можливості і коригування для дотримання термінів.
- Сильно залежить від взаємодії з клієнтом, тому, якщо клієнт не зрозумілий, команда може бути загнана в неправильному напрямку.
- Існує дуже висока індивідуальна залежність, оскільки мінімум документації згенеровано.
- Передача технологій новим членам команди може бути досить складним завданням через відсутність документації.

Управління проєктом впродовж останнього десятиріччя стало сильною технологію управлінням змінами в висококонкурентному світі. Оптимальне використання засобів управління проєктами дозволяє успішно втілювати будь-який проєкт відповідно критеріям якості, досягти в час поставлених завдань, заощадивши кошти і скоротивши ризики. З глобалізацією економіки проєктно - орієнтоване управління стає одним з значущих аспектів перемоги компанії в конкурентній боротьбі і в підкоренні нових ринків.

Визначення проєкта і ведення проєкту пов'язані з потребою управління змінами. Управління проєктами характеризується нероздільною частиною повсякденної активності керівників різного рівня. Чимало керівників потребує використання формалізованих способів управління проєктами, все ще пов'язують з великими проєктами, такими як запуск міжпланетної станції, реалізацією новітніх видів обладнань або будівництвом атомної електростанції. Використання формалізованих принципів управління проєктами дозволяє більш конкретно визначати мету інвестицій і максимально планувати інвестиційну діяльність, найбільш повно враховувати проєктні витрати, оптимізувати застосування наявних резервів і уникати проблемних ситуацій, відстежувати виконання плану, аналізувати документальні показники і привносити своєчасну корекцію в хід роботи, накопичувати, аналізувати і задіяти в подальшому досвід реалізованих проєктів. Таким

чином, система управління проєктами є однією з найважливіших компонентів всієї системи управління організацією [8].

1.2. Класифікація методів і моделей управління проєктами з «Scrum»

Використання класичних моделей управління проєктами не дає гарного результату при створенні нових невідомих цифрових продуктів. Так як серед тривіальних завдань команда розробник буде вирішувати нові підходи реалізації продукту і тому Scrum алгоритми становлять дуже велику частку успіху кінцевого результату. Scrum не має чітких законів, є тільки рекомендації гарних моделей. Нижче я привів гарний алгоритм з використанням таблиць та карток за якими я працюю сам в ролі члена команди розробників.

Scrum-команда складається з декількох ключових ролей, для яких суворо визначені обов'язки і відповідальність. Будь-яка Scrum-команда повинна включати наступні ролі:

Власник продукту - людина, яка розуміє, як продукт повинен виглядати, представляє інтереси замовника, працює з командою, розставляє пріоритети завдань. Обов'язки: описує елементи продукту, управляє пріоритетністю завдань, забезпечує ясність завдань для команди, ставить завдання команді, але не виконавцям [6].

Scrum-майстер - «службовець лідер», сполучна ланка між менеджментом і командою. Обов'язки: веде зустрічі, усуває перешкоди, робить проблеми видимими, відповідає за дотримання правил, допомагає максимізувати ефективність команди [6].

Команда є самоорганізованою і самокерованою. Команда бере на себе зобов'язання щодо виконання обсягу робіт перед власником продукту. Розмір

команди обмежується групою людей, здатних ефективно взаємодіяти як одне ціле. Оптимальна чисельність команди складає 7 плюс-мінус 2 людини.

Розглянемо більш детально алгоритм роботи Scrum.

Алгоритм роботи Scrum побудований на ряді подій і артефактів. Для того, щоб Scrum почав працювати, його необхідно впроваджувати комплексно і враховувати послідовність і періодичність реалізації подій, а також правильно спроєктувати артефакти з урахуванням специфіки проєкту, який ви реалізуєте [6].

Беклог продукту - це список вимог клієнта, які необхідно реалізувати в рамках проєкту. Завдання повинні бути впорядковані, відсортовані за пріоритетністю і складністю. Беклог продукту - це єдине джерело вимог до продукта, яким керуються учасники проєкту. Відповідальність за беклог продукту, включаючи його зміст, доступність і упорядкування елементів, несе власник продукту [6].

Приклад:

- Потрібно мати графічний інтерфейс.
- Додаток повинен працювати у всіх сучасних браузерах.
- Додаток повинен бути адаптивним для сучасних смартфонів.
- Додаток має сканувати Qr коди та вносити їх до бази даних.
- База даних повинна бути SQL(MySql).
- Додаток повинен мати зв’язок з камерами на складі для регулювання розташування продукції.
- Додаток має враховувати товари на складі та розподіляти по магазинам власника із врахуванням вимог кожного магазина.
- Треба відправляти статистику про працю складу кожні 48 годин.
- Час на виконання 2 місяці.

Беклог спринту - елементи беклога продукту, вибрані для виконання в поточному спринті. Завдання розподіляються членами проєктної групи по часу і виконавцям виходячи з їх пріоритетності та складності. Основне правило - якщо команда домовилася про певну кількість завдань, які потрібно

виконати за один спринт, то додавати нові вже не можна. Беклог спринту належить виключно команді і служить наочним поданням реального обсягу робіт, який планує виконати команда протягом спринту [6].

Приклад:

- Створення дизайну макету з використанням графічного редактору Figma.
- Реалізація дизайн-макету на технологіях HTML&CSS.
- Розробка інтерфейс логіки з використанням ReactJs.
- Розробка серверу для зв'язку бази даних та інтерфейсу з використанням NodeJs.

Спринт - ядро Scrum. Спринт - це часовий період, протягом якого команда створює готову до використання частина продукту. Тривалість спринту складає від однієї до чотирьох тижнів. Мета спринту - це встановлений орієнтир, який досягається за допомогою виконання частини беклога продукту [6].

Щоденний Scrum - «Stand up» нарада тривалістю не більше 15 хвилин, спрямована на створення і синхронізацію плану робіт на найближчі 24 години [6].

Учасники відповідають на 3 питання:

- Які завдання виконані командою з моменту минулої зустрічі для досягнення мети спринту?
- Які завдання необхідно виконати сьогодні, щоб допомогти команді досягти мети спринту?
- Які перешкоди і проблеми уповільнюють досягнення проєктної командної мети спринту?

Огляд і ретроспектива спринту. Огляд спринту - зустріч, на якій команда розповідає, що зроблено за спринт і демонструє готові частини продукту. Ретроспектива спринту - це оцінка ефективності роботи команди, створення плану поліпшень для наступного спринту [6].

Учасники відповідають на 3 питання:

- Яке поліпшення команда може запровадити в процес негайно?
- Що пройшло добре під час реалізації останнього спринту?
- Що можна зробити краще при реалізації наступного спринту?

Інструменти Scrum наступні.

Scrum-дошка ділиться на три колонки: «Необхідно виконати», «В процесі виконання» і «Виконано». Використовуються паперові картки, на яких пишуться завдання. Всі картки поміщаються в колонку «Необхідно виконати» відповідно до пріоритетів. Кожен день, коли хтось говорить «я почав працювати над ...» картка із завданням переміщається в колонку «В процесі». На наступному Daily Scrum, кожен, хто каже «я закінчив роботу над ...» переміщує відповідну картку в колонку «Виконано» [6].

За правильністю використання дошки стежить Scrum-майстер. В його обов'язки входить вирішення суперечок при переміщенні карток, коли у членів команди розходяться думки про ступінь і якість виконання завдання з беклога спринту. Scrum-майстер стежить, за тим, щоб дошка регулярно оновлювалася виходячи із виконаних завдань».

Діаграма вигорання задач - це графічне представлення того, як швидко команда працює над задачами, гнучкий інструмент, який використовується для збору опису функції з точки зору кінцевого користувача. Діаграма вигорання показує загальні зусилля в порівнянні з обсягом роботи для кожної ітерації [4].

Основні поняття діаграми (див. рис. 1.1):

- Діаграма вигорання завдань - діаграма, що показує кількість зробленої роботи і скільки залишилося. Оновлюється щодня з тим, щоб в простій формі показати зрушення в роботі над спринтом. Графік повинен бути загальнодоступний.
- Принцип оцінки - Обраний принцип оцінки складності зазначений на вертикальній осі.
- Кількість роботи, що залишилася. Червона лінія показує кількість роботи, що залишилася, за оцінками команди.

- Лінія управління. Сіра лінія являє собою приблизне положення, яке команда зайняла б, якби робота виконувалася лінійно, без затримок або достроково. Якщо червона лінія знаходиться нижче сірої, вітаємо - все йде до того, що команда завершить всю роботу до кінця спринту. Але це не означає, що успіх гарантований. Відстежуючи прогрес команди, ви повинні вміти правильно використовувати наявну інформацію.

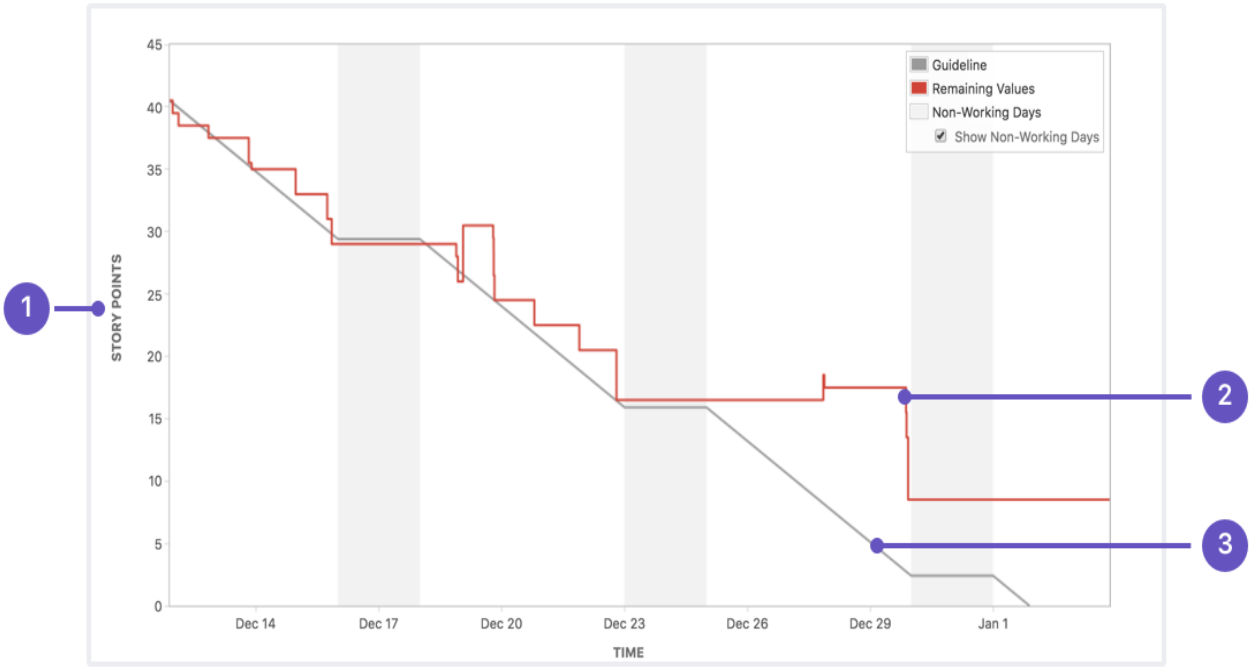

Рис. 1.1. Діаграма вигорання задач

Методологія Scrum є ефективним способом управління проєктами в тих випадках, коли середовище діяльності нестабільне, змінюється бюджет, вимоги до результатів роботи, інструменти, масштаби. Найбільш корисною і ефективною методологія є при виробництві нового, не типового продукту - проєктування і будівництві споруд, створення програмного забезпечення, підготовка та реалізація заходів, маркетингові кампанії та багато іншого. При правильному застосуванні методики і чіткому дотриманні ключовим принципам, команди здатні багаторазово збільшити ефективність роботи, що підтверджується рядом живих прикладів.

1.3. Аналіз сучасних додатків для управління «Scrum» командою

Інструменти управління проєктами - це відповідь керівника проєкту на управління проєктом. Для простих проєктів не потрібно нічого, крім контрольного списку, в той час як для інших складних потрібне правильне планування, розподіл завдань, установка термінів, забезпечення їх дотримання і відстеження витраченого часу.

Необхідність кількісної оцінки, об'єктивізації, поділу та делегування завдань належним чином і пропорційно має велике значення, і саме тут на допомогу приходять сучасні інструменти. Існує безліч програмних інструментів для управління проєктами. Нижче наведено список сучасних веб-додатків для управління проєктом з використанням Scrum методології.

Kissflow Project - багатофункціональна потужна програма, одна з найпопулярніших програм для управління проєктами. Kissflow Project - ідеальний варіант інструменту управління проєктами для функціональних менеджерів проєктів і людей, які погано знайомі з керуванням проєктами [25]. Kissflow Project перевіряє всі можливості за допомогою вичерпного набору функцій управління проєктами. Це простий у використанні і естетичний графічний інтерфейс користувача, який робить це універсальне рішення відмінним вибором для всіх типів організацій.

Переваги:

- Відстеження. Якщо вам потрібно знати, де що відбувається, той факт, що ця програма призначена для робочого процесу, дозволяє неймовірно легко визначити, де люди перебувають в потоці роботи.
- Інтерфейс: Зручний інтерфейс.

Недоліки:

- Навчання. Якщо у вас 100% паперовий процес, і у вас є співробітники, які більше нічого не знають, то безумовно є крива навчання. Вам потрібно буде запросити кого-небудь, щоб допомогли вам навчити працівників основної концепції зміни (питання «Чому ми це робимо?»).

- Дизайн робочого процесу: при створенні робочого процесу немає функції копіювання, вставки, і ви не можете переміщати завдання робочого процесу між паралельними гілками. Це стає проблемою, коли в елементі міститься 98% того, що вам потрібно, але ви не можете скопіювати його і помістити в іншу гілку, щоб внести останні зміни. Замість цього його потрібно повністю відтворити.
- Ціна: Дуже велика ціна для маленьких команд, 25$ за кожного члена команди в місяць.

Trello - це провідне програмне забезпечення для управління завданнями проєкту з вбудованими інструментами спільної роботи для роботи з вашими командами. Він легкий, простий і зрозумілий у використанні. Він використовує дошки, списки і картки для створення завдань і підтримки порядку. Користувачі можуть співпрацювати в картках, обмінюватися файлами і залишати коментарі [26].

Переваги:

- Обмін файлами: Можливість ділитися файлами з членами вашої команди.
- Адаптивність: Підходить для мобільних пристроїв.
- Ціна: Менш складна структура ціноутворення.

Недоліки:

- Керування: Відсутні функції звітності та обліку робочого часу.
- Інтерфейс: Ви не можете переглядати ітерації.
- Інтерфейс: Можна дати лише простий опис.
- Керування: Занадто багато дощок ускладнює швидке виявлення завдань.

Asana - це гнучкий інструмент управління Saas-проєктами, орієнтований на спільну роботу. Існують робочі процеси, що автоматизують повторювані завдання. Він інтуїтивно зрозумілий з різними дисплеями завдань і підтримує налаштування полів і форм. Також є можливості для створення власних правил і робочих процесів затвердження. Asana пропонує багато

можливостей для продуктивності і спільної роботи, в той час як багато основних функцій управління проєктами реалізуються за допомогою інтеграції, яка може бути не ідеальною для всіх користувачів. Це добре для великих підприємств, але не ідеально для маленьких [28].

Переваги:

- Керування: розподіл завдань і відстеження. Управління віртуальною нарадою, що включає порядок денний, відгуки та подальші дії.
- Обмін файлами: спільна робота над завданнями з широким набором функцій для зберігання всіх видів інформації про завдання в одному місці.

Недоліки:

- Ціна: дуже велика ціна для маленьких команд, 25$ за кожного члена команди в місяць.
- Інтерфейс: дуже складні шаблони інтерфейсу.

Zoho Projects - популярна пропозиція від Zoho, яка допомагає підприємствам відслідковувати завдання, співпрацювати, відстежувати помилки і створювати інтуїтивно зрозумілі звіти. Інструмент пропонує такі функції, як діаграми Ганта, і можливість виставлення рахунків за години, одночасно за кількома проєктами. Zoho Projects також може похвалитися дуже вражаючим набором комунікаційних інструментів, включаючи чат в реальному часі і сторінки форумів [27].

Переваги:

- Керування: відмінні функції управління проєктами за ціною, яку вони пропонують. Любі функції таймерів для задач, коли запускається нова задача, можна реєструвати годинник для цього, що є відмінною функцією.
- Ціна: дуже низька ціна, 10$ в місяць без ліміту на користувачів.

Недоліки:

- Адаптивність: неможливо використовувати на мобільному пристрої.

- Налаштування: проєкти Zoho складно налаштувати. Його масштабованість дуже обмежена певними завданнями. Це добре для управління проєктами, але не забезпечує канал зв'язку між людьми. Вікна чатів сильно запізнюються між спілкуванням, а на оновлення форумів в реальному часі потрібно багато часу. Zoho вимагає високої пропускної здатності інтернету, повільний інтернет створює проблеми з оновленнями, і Zoho починає відставати. Розрахований на багато користувачів портал також має багато проблем. Ця функція створює проблеми зі спільним доступом до файлів. На передачу файлів йде багато часу.

JIRA - популярне програмне забезпечення для управління проєктами, призначене в основному для гнучких команд розробників програмного забезпечення. В рамках програмного забезпечення ці команди можуть створювати власні дошки, дошки Канбан і використовувати гнучку звітність (в режимі реального часу). Користувачі також можуть відслідковувати помилки, переглядати невирішені проблеми і відстежувати час, витрачений на виконання завдання. JIRA поставляється з надійним набором API-інтерфейсів, які дозволяють користувачам підключати його до багатьох сторонніх програм [26].

Переваги:

- Інтерфейс: зручний інтерфейс.
- Підтримка: управління заявками для користувачів поза системою за допомогою функції служби підтримки.

Недоліки:

- Ціна: дуже велика ціна для маленьких команд, 140$ підписка за продукт кожного місяця, а також 5$ за кожного користувача.
- Адміністрування: дуже багато функцій які не потрібні для ведення дошки проєкту.

Висновки до розділу 1

Розглянуто особливості застосування підходу до управління проектами та створення продуктів Scrum, який колись був розроблений з метою керування IT проектами (при розробці програмного забезпечення) Базовими позиціями, які доводять ефективність застосування даного підходу є: підвищення продуктивності, мотивації до праці, скорочення непродуктивних втрат часу і ресурсів, продумана інтенсифікація виробництва.

В роботі досліджено основні моделі та їх реалізації Scrum та наведено приклади застосування даного підходу у практиці та додатків, які допомагають в реалізації цієї моделі. При виборі додатку для реалізації Scrum треба розрахувати бюджет який компанія може витрати і можливості в наймі додаткового персоналу для підтримки процесу.

На основі аналізу переваг і недоліків додатків для реалізації методології можно сказати що найкращим вибором для середньої та великої компанії є Jira, Zoho Projects, Asana, Kissflow Project. Вони мають повний спектр інструментів та абстракції для реалізації моделі Scrum для подальших їх модифікацій. Але це потребує повної оплати сервісів та потреба в Scrum майстрах які будуть налаштовувати процес та підтримувати його. Для маленьких організацій гарним вибором є Jira або Asana але прийдеться сплачувати підписку за повний набір інструментарію який може навіть і не знадобитись. Альтернативним та безкоштовним вибором є Trello, але він не має основних важливих інструментів для менеджменту та часу звітності.

РОЗДІЛ 2. ДОЦІЛЬНІСТЬ РОЗРОБКИ ДОДАТКУ ДЛЯ РЕАЛІЗАЦІЇ SCRUM МЕТОДОЛОГІЇ

2.1. Обгрунтування доцільності розробки додатку

Відмінна риса IT-проектів полягає в високих вимогах до актуальності продукту, динамічно мінливого ринку. Тому актуальним стає вибір такої методології управління проектами, яка б відповідала всім вимогам до сучасних IT-проектів. У своєму щорічному звіті, State of Aglie 2020 року, Digital.ai, де брало участь більше 40 000 респондентів, повідомляється, що 95% респондентів практикують Agile метод в своїй організації, з них 58% практикують Скрам і більше 27% практикують гібриди Скрам [32]. Згідно з даними опитування очевидно, що, Скрам є більш затребуваним проектним підходом на сьогоднішній день. Скрам дозволяє підвищити прозорість і прогнозованість робіт над проектами, знизити ризики і стабілізувати швидкість розробки. Скрам, як один з методів гнучкої методології, показує себе дуже ефективно, але має ряд недоліків при роботі з ним. До них можна віднести: відсутність належної кваліфікації членів команди; низьку увагу до тестів на різних стадіях розробки програмного проекту; вузька спеціалізація методології Скрам вимагає від команди знання основних правил Скрам. У великому дослідженні гнучких методів по всьому світу було виявлено, що досліджувані компанії на додаток до Скрам створювали свої спеціальні і специфічні гнучкі методи. Метод, який часто зустрічається, коли мова йде про коригування гнучкого методу, є адаптованою версією Скрам, тобто, деякі частини Скрам не враховуються або коригуються. Однак такий підхід не відповідає визначенням Скрам, а скоріше виступає прикладом того як Скрам адаптують під потреби компанії. До того, як був створений Agile-маніфест, в книзі Вільяма Брауна описуються найбільш поширені проблеми при розробці програмного забезпечення, і визнають їх антипатерн. Антипаттерн, це набір

правил для вирішення проблем які часто зустрічаються при розробці програмного коду. Елоранта і співавтори у своїй роботі вивчили дане питання і надали список які антипатерни Скрам найчастіше зустрічаються.

Один з них часто називають «ScrumBut». Це означає, що деякі елементи Скрам відсутні або змінені. Це відправна точка синдрому ScrumBut. Кен Швабер дав дуже гарне визначення ScrumBut: «ScrumBut має певний синтаксис: (Причина) (Тимчасове рішення). Приклад: «(Ми використовуємо Scrum, але) (щоденні збори Scrum занадто трудомісткі) (тому ми проводимо їх тільки раз в тиждень, якщо вони не потрібні нам частіше)». Такого роду анти-патерни з деякими відхиленнями від Скрам, можуть призвести до виключення важливих частин Скрам. Анти-патерни - це шаблони помилок, які виникають при вирішенні різних завдань [17]. Скрам-анти-патерни звичайні практики, які здаються зручними на початку робіт, але зрештою шкідливі для проекту. Скрам анти-патерни часто можуть бути пов'язані з альтернативним способом роботи, який є рекомендованою практикою і більш підходить в більшості випадків. Скрам анти- патерни служать двом важливим цілям: допомагати ідентифікувати проблеми і допомагати впроваджувати рішення. Коли деякі частини Скрам пропускаються, це називається ScrumBut. Швабер наводить приклад: «Ми не проводимо щоденні зустрічі, тому що це може призвести до негативного результату при використанні Скрам ». Елоранта і співавтори описують ScrumBut як анти-патерни Скрам. Їх дослідження показало найбільш часті анти-патерни Скрам. Всього вони виявили 14 патернів Скрам, які можуть бути неповноцінні для компаній. Для прикладу, «Клієнт - Власник продукту в одній особі, призвело до того, що власник продукту намагався диктувати, як Скрам-команда повинна реалізувати функціонал ПО і, таким чином, порушував роботу Скрам-команди». «Неупорядкований Беклог продукту призвів до того, що команда Скрам може не побачити потенційні ризики або цінні елементи продуктів». Нижче наведена таблиця, яка містить список з 14 анти-патернів Скрам [20].

Таблиця 2.1

Анти-патерни використання методології Скрам

№	Виявлені Скрам анти шаблони (анти патерни)	Наслідки
1	Проведення тестування в наступному спринті	Коли тестування відбувається в наступному спринті, новий код може бути написаний поверх не протестованого коду.
2	Занадто довгий Спринт	Завдання, як правило, залишаються незавершеними в кінці спринту, можливо, через те, що перші тижні використовуються недостатньо ефективно.
3	Великий документ з вимогами	Вимоги системи в цілому не зовсім зрозумілі команді, так як їх важко знайти у великих документах.
4	Невпорядкований беклог продукту	Оскільки беклог продукту не впорядкований, команді може не вистачати бачення і розуміння про ризиковані або цінні елементи (завдання) продукту.
5	Завдання оцінені кимось і передані команді на виконання	Результатом анти-патерну є команди, яким не вистачає прихильності (відданості): якщо в спринті є вільний час, Команди не переносять завдання з Журналу Продукту в Спринт.
6	Власник продукту є клієнтом	Клієнт може бути надто зайнятий, щоб готувати всі завдання журналу продукту для Команди. Таким чином, вимоги і завдання можуть бути неясними для Команди. Організаційний бар'єр може зробити спілкування ще складнішим. Власник продукту (клієнт) часто намагається диктувати, як команда повинна реалізувати функцію і таким чином порушувати роботу команд. Клієнт проводить оптимізацію бюджету.
7	Невидимий прогрес	Невидимий прогрес призводить до недостатнього розуміння прогресу Команди.
8	Власник продукту без повноважень. (Член команди розробників призначений власником продукту)	Власник товару не має повноважень приймати рішення, які елементи реалізувати і які відкидати, розставляти пріоритети. Над власником продукту можуть перебувати інші менеджери, які приймають рішення щодо продукту. Власник товару не обов'язково знаходиться в прямому контакті з клієнтом, і тому власник продукту не знає, які функції мають цінність для клієнта.

Як вказувалося, Скрам - це метод, який легко можна адаптувати до різних проектів. Тому компанії часто адаптують Скрам метод під потреби

своєї компанії випускаючи свої варіанти Скрам. Одним із часто вживаних методів розробки, який використовує підхід гнучке мислення, є Скрам. Було досліджено основні анти-патерни при роботі з методологією в проекті і проаналізовано ряд існуючих проєктів на ринку їх співвідношення функціоналу та ціни а так само користь для команд і організація різних розмірів. Основними критеріями за якими було проведено аналіз це: можливість дати оцінку складності завдання, статуси виконанню завдання, опис вимог завдання, маркери пріоритетності завдання, можливість пріоритизації списку завдань, звітність за часом а також звітності у вигляді діаграм вигорання завдань і Ганта. Кожен продукт має тільки частини необхідних функциональностей і не захищає від використання антипатернів командою що знижує ефективність використання методології. Тому було прийнято рішення розробити концепт веб-додатку який мав би тільки потрібний функціонал, відкритий код і простий у використанні.

Основний функціонал додатку та поради по інтеграції методології в ньому. Робочі елементи, які призначені для користувача історії, завдання і помилки, «проблеми». Можливість створити кілька користувальницьких історій з опцією швидкого створення в беклозі. Якщо у вас немає користувальницьких історій, просто створіть зразки історій, щоб почати роботу і подивитися, як працює процес.

Створення спринту в беклозі, щоб приступити до планування спринту раніше. На початку спринту ви повинні провести нараду з планування спринту з іншою частиною вашої команди. Зустріч з планування спринту - це церемонія, яка налаштовує всю команду на успіх протягом усього спринту. На цій зустрічі вся команда обговорює мету спринту і історії в пріоритетному беклозі продукту. Команда розробників створює докладні завдання та оцінки для пріоритетних історій. Потім команда розробників зобов'язується завершити певну кількість історій в спринті. Ці історії і план їх завершення стають так званим беклогом спринту.

Можливість додавати оцінки в балах до своїх історій, додавши число в поле оцінки балів в історії. Ви також можете додати додаткові відомості в історії або клацнути значок створення підзадачі, щоб докладніше розбити роботу над історією.

Під час спринту рекомендується перевіряти діаграму згоряння та діаграму Ганта. У діаграмі згоряння показується фактичний і передбачуваний обсяг роботи, яку необхідно виконати за спринт. Діаграма вигорання автоматично оновлюється в міру виконання робочих елементів.

Огляд спринту або демонстрація спринту - це збори, на яких команда показує, що вони зробили в цьому спринті. Кожен спринт зазвичай виробляє робочу частину продукту, звану инкрементом. Це зібрання з великою кількістю відгуків про проект і включає в себе мозковий штурм, щоб допомогти вирішити, що робити далі. Можливо, вашій команді доведеться працювати над функціями, які становлять більшу частину роботи. Коли це відбувається, вам слід подумати про використання епіків при плануванні роботи та усунення відставання. Епіки - це, по суті, великі призначені для користувача історії, які можна розбити на більш дрібні призначені для користувача історії. Більш того, епіки можуть охоплювати кілька проектів. Отже, якщо ваша команда працює над одними історіями, які створюють більшу призначену для користувача історію, яка, в свою чергу, впливає на кілька проектів, продовжуйте і використовуйте епіки - вони можуть спростити відстеження проєктів в цьому сценарії.

Вибір правильних робочих елементів для спринту - це співпраця між власником продукту, майстром і командою розробників.

Вибір правильних робочих елементів для спринту - це співпраця між власником продукту, майстром і командою розробників. Власник продукту обговорює мету, яка повинна бути досягнута за допомогою спринту, і елементи затримки продукту, які досягають мети спринту по завершенні. Потім команда створює план того, як вони будуть збирати елементи накопичення і виконувати їх до кінця спринту. Вибрані робочі завдання і план

їх виконання називаються задачі. До кінця планування спринту команда готова почати роботу над затримкою спринту, сортуванням елементів боргу до «В процесі» і «Виконано». Під час спринту команда перевіряє, як просувається робота під час щоденного сбору. Мета цієї зустрічі - виявити всі перешкоди і проблеми, які впливають на здатність команд виконати мету спринту. Після спринту команда демонструє, що вони виконали під час спринту. Це можливість для вашої команди продемонструвати свою роботу зацікавленим сторонам і товаришам по команді до того, як вона з'явиться у виробництві. Завершіть свій цикл спринту ретроспективою спринту. Це можливість для вашої команди визначити області, які потребують поліпшення для наступного спринту. Таким чином, ви готові почати наступний цикл спринту [20, 19, 16].

2.2 Опис функціоналу додатку

В даному розділі кваліфікаційної роботи буде розглянуто основний функціонал додатку для управління проєктом з розробки IT-додатку.

Основне меню додатку(див. рис. 2.1). Додаток має 3 основних частини:

1. Шапка додатку та глобальний пошук завдань(див. рис. 2.1).
2. Меню навігації(див. рис. 2.2).
3. Дошка завдань і пошук для фільтрації (див. рис. 2.3).

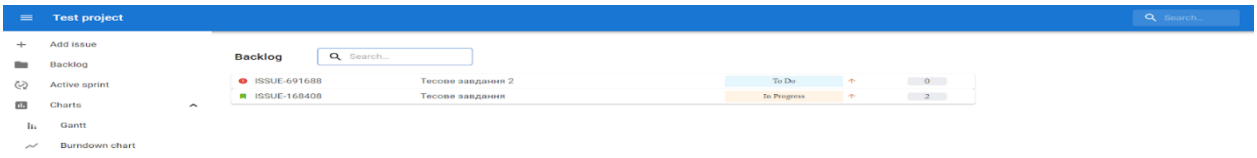

Рис. 2.1. Основне меню додатку

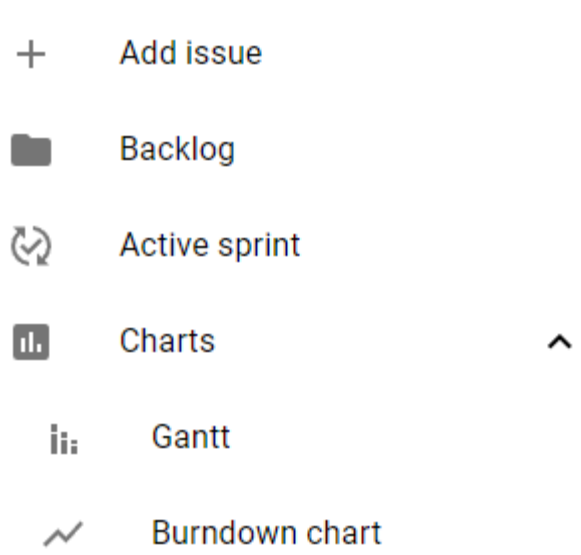

Рис. 2.2. Меню навігації

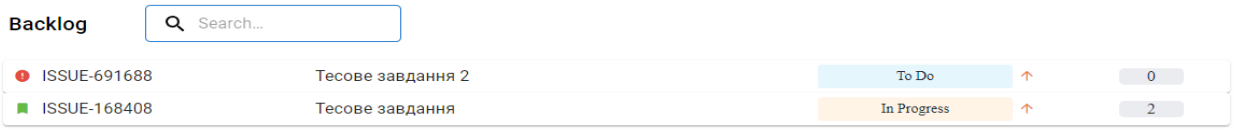

Рис. 2.3. Дошка завдань

Зробимо огляд основних інструментів лівої панелі(див. рис. 2.2).

Add issue – викликає вікно створення задач(див. рис. 2.4).

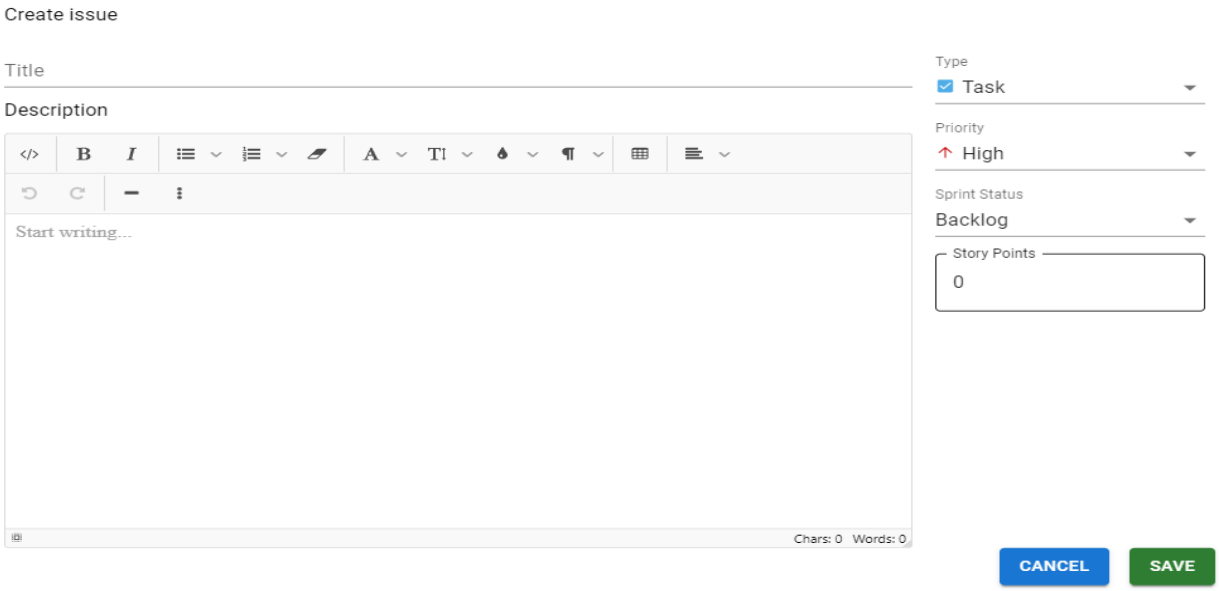

Рис. 2.4. Вікно створення задач

Центральні поля дозволяють задавати назву і опис задачі та можливість вставляти картинки та редагувати вид тексту(див. рис. 2.5).

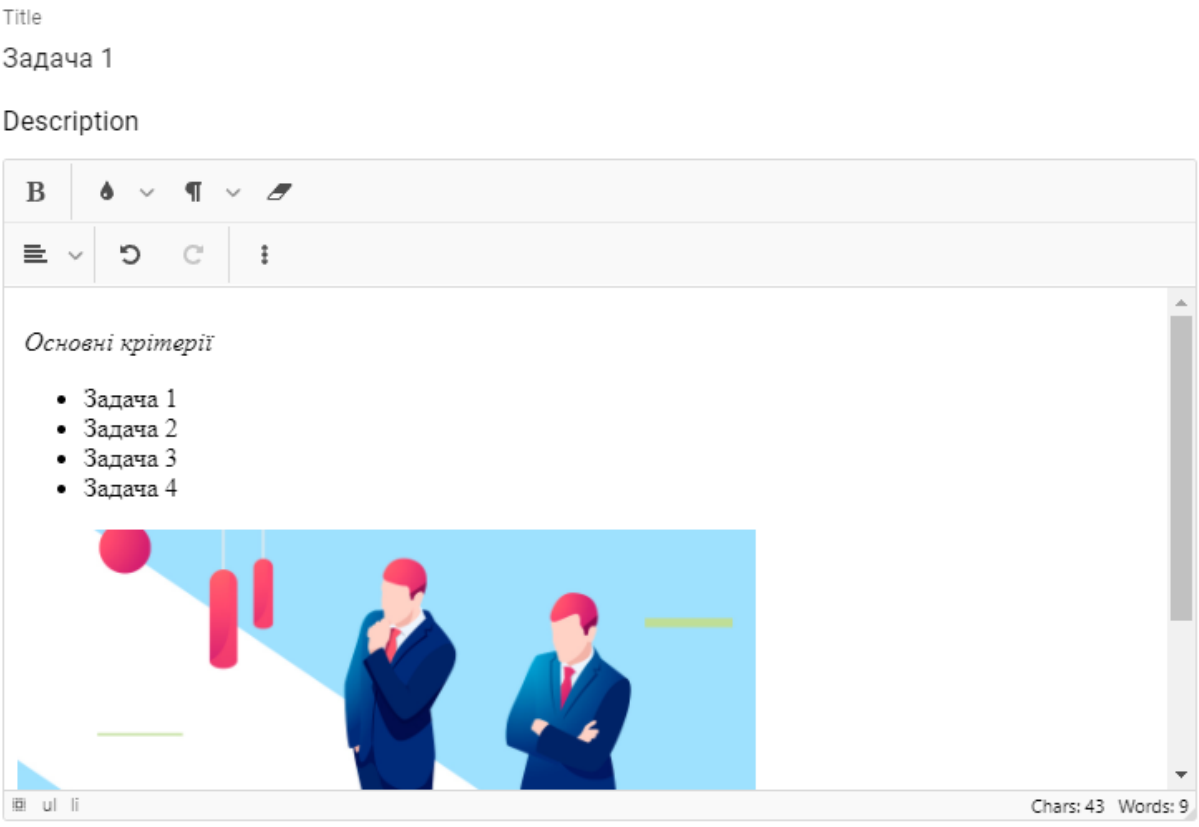

Рис. 2.5. Редактор опису задачі

Права панель опцій дозволяє встановити основні показники для задачі. Такі як тип, пріоритетність, зв'язок із подією, бали складності задачі(див. рис. 2.6).

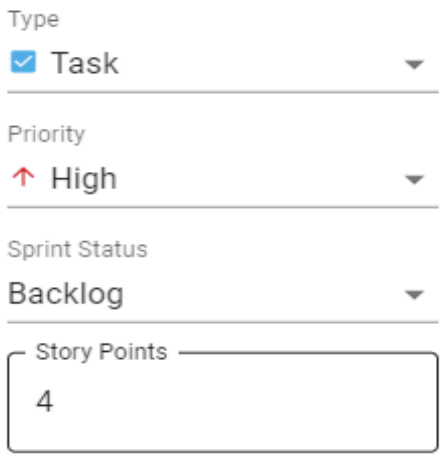

Рис. 2.6. Панель опцій

Backlog – основна дошка з усіма подіями які не були приєднані до спринта(див. рис. 2.7).

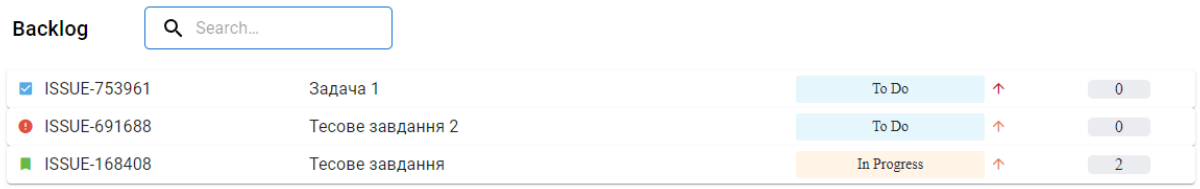

Рис. 2.7. Дошка задач

Active sprint – дошка яка показує задачі які були додані в спринт(див. рис. 2.8).

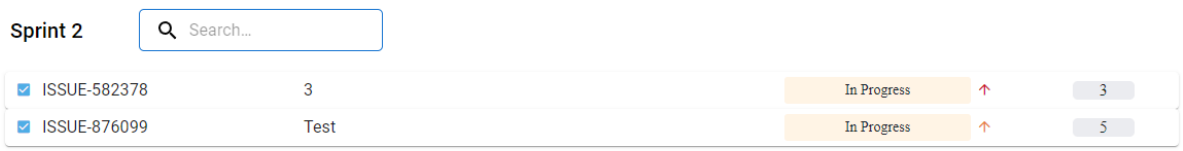

Рис. 2.8. Дошка спринта

Charts(Gantt) – генерує діаграму Ганта беручи у фільтр задачі з активного спрінта(див. рис. 2.9).

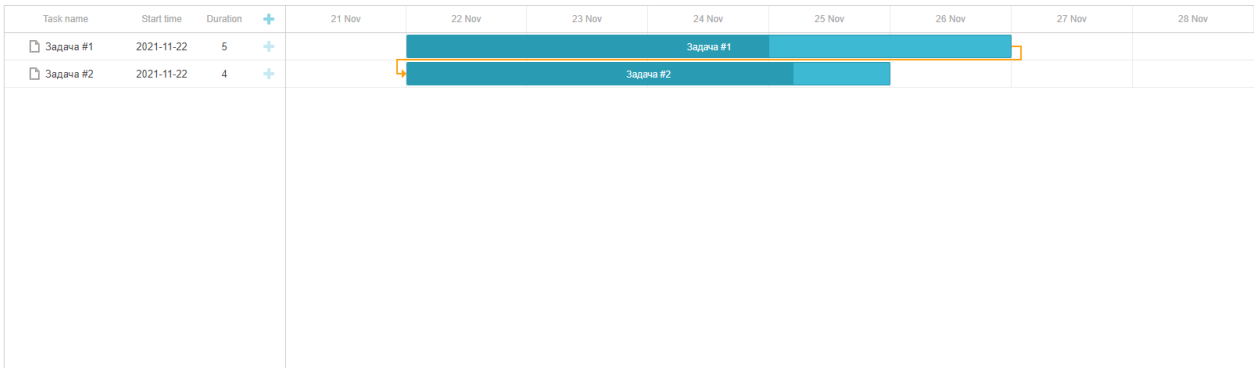

Рис. 2.9. Діаграма Ганта

Burdown chart – генерує графік вигорання задач відносно часу спринту та виконаних задач в ньому(див. рис. 2.10).

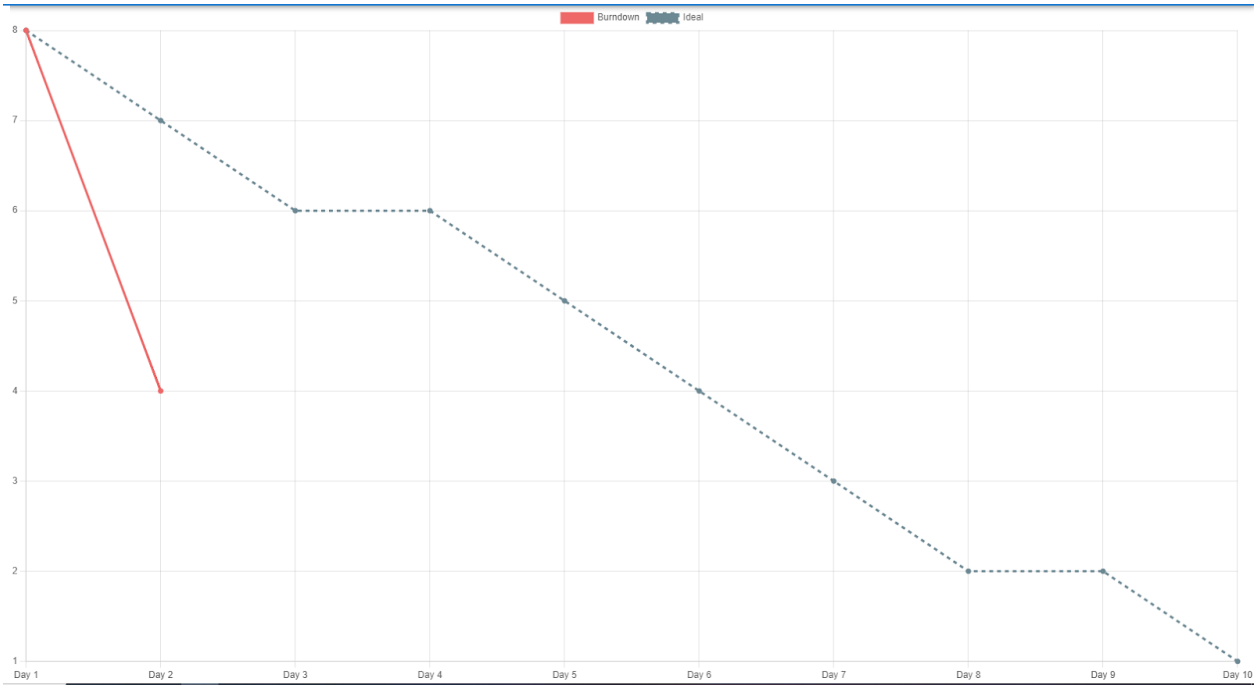

Рис. 2.10. Діаграма вигорання задач

Основна структура задачі на дошці(див. рис. 2.11):

- Тип задачі.
- Унікальний номер.
- Заголовок задачі.

- Прогрес.
- Маркер пріоритетності.
- Складність задачі.

Рис. 2.11. Лейбл задачі

Вікно розгорнутого опису задачі(див. рис. 2.12).

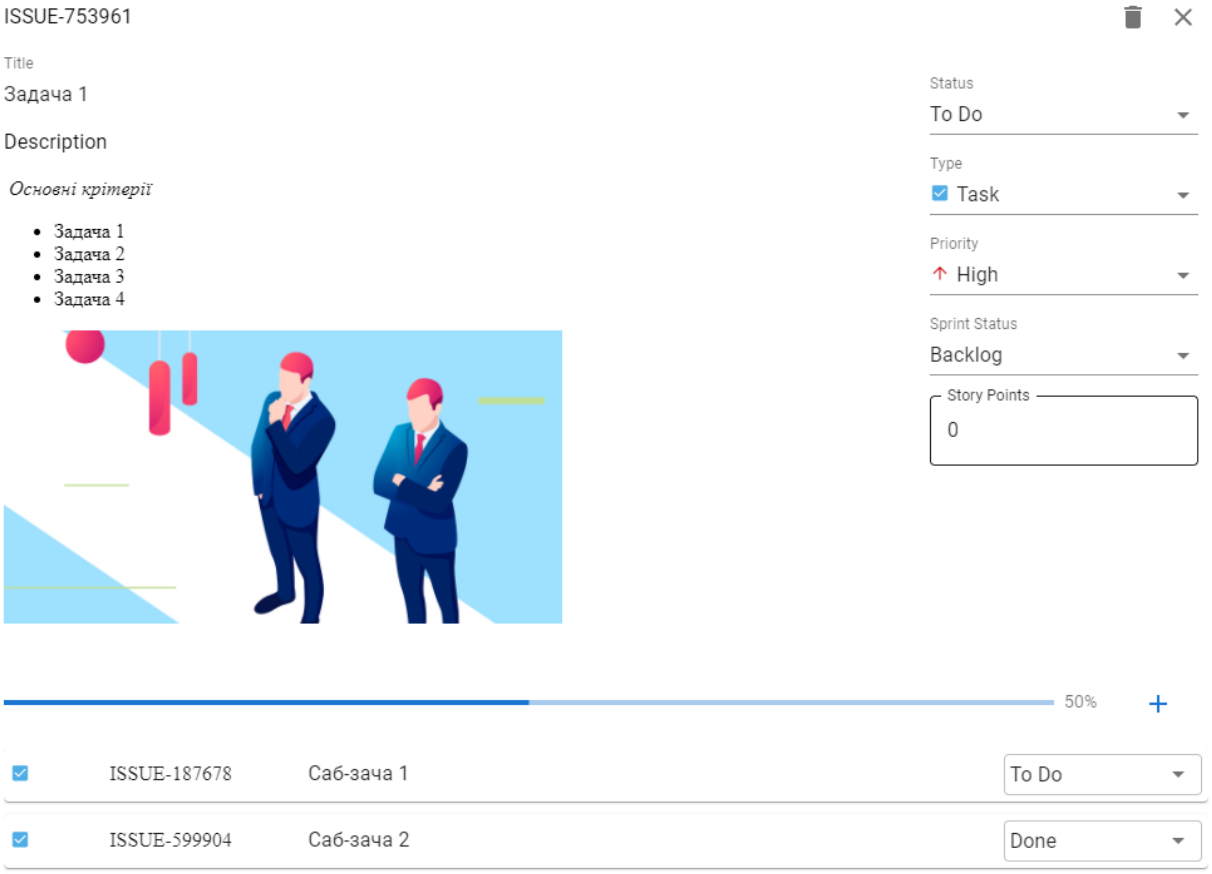

Рис. 2.13. Повний опис задачі

Підзадачі для більш детального опису кожної операції необхідної для виконання основної задачі і також полоса прогресу виконання підзадач(див. рис. 2.13).

У даному розділі описані моделі данних та основні функції веб–додатку, безпечність, інтерфейс. Виконана постановка задачі та встановлені вимоги до якості розподілення ресурсів на розробку веб–додатку для організації та контролю робочого процесу команди з розробки IT-додатку складської діяльності.

2.3. Опис інформаційної моделі додатку

Проблема складської алгоритмізації стає дуже важливою частиною торгівлі, так як класичні пакеті такі як Microsoft Excel, Kanbana,Microsoft Access, не дуже добре інтегруються до храмових сховищ та не підходять для гнучких рішень зв’язаних с зберіганням даних через систему управління базами даних такими як MySql, PostSql, Oracle. Тому сучасний web-додаток на основі мові програмування JavaScrip є надійним та швидким рішенням. Нижче описані потреби замовника , та моделювання проєкту.

Модель задач які покривають основні цілі для розробки застосунку наведені нижче:

- Потрібно мати графічний інтерфейс.
- Додаток повинен працювати у всіх сучасних браузерах.
- Додаток повинен бути адаптивним для сучасних смартфонів.
- Додаток має сканувати Qr коди та вносити їх до бази даних.
- База даних повинна бути SQL(MySql).
- Додаток повинен мати зв’язок з камерами на складі для регулювання розташування продукції.

- Додаток має враховувати товари на складі та розподіляти по магазинам власника із врахуванням вимог кожного магазина.
- Треба відправляти статистику про працю складу кожні 48 годин.

Визначення команди для розробки.

5 - Front-end розробників.

3 - Back-end розробників

1 - Дизайнер

1 - Верстальщик

Розглянемо основні функції Front-end та Back-end розробників які будуть користуватись додатком для керування задачами по методології Scrum.

Отже, у програмній інженерії терміни "front end" і "back end" розрізняють за критерієм розмежування обов'язків між рівнем уявлення і рівнем доступу до даних відповідно. Front end - це функціонал для взаємодії між користувачем і back end. Front end і back end можуть бути поділені між однією або двома структурами. У програмній архітектурі може бути чимало рівнів між обчислювальним забезпеченням і кінцевим користувачем. Кожен з цих рівнів може мати як front end, так і back end. Front - це абстракція, спрощення базисного елемента через надання користувач комфортного (user - friendly) інтерфейсу. У конструюванні програмного забезпечення, наприклад, архітектура Model - View-Controller надає front end і back end для бази даних, елементів обробки користувачів і даних. Поділ програмних систем на front end і back end полегшує розробку і розмежовує підтримкою. Емпіричне правило полягає в тому, що front (або "клієнтська") сторона - це будь-який елемент, яким керує користувач. Код серверної сторони (або "back end" -у) розташовується на сервері. Плутанина виникає, коли хтось повинен зробити front-end зміни файлів на стороні сервера. Більшість HTML - дизайнерів, наприклад, не повинно бути на сервері під час розробки HTML, і навпаки, серверні інженери, за визначенням, не повинні бути ніде крім сервера. Таким

чином, потрібні обидва розробника, щоб в кінцевому результаті створити функціональний, інтерактивний веб-сайт.

На основі беклогу продукту розробимо спринти які можно внести у додаток для керування задач.

Беклог спринту наступний:

- Створення дизайну макету з використанням графічного редактору Figma.
- Реалізація дизайн-макету на технологіях HTML&CSS.
- Розробка інтерфейс логіки з використанням ReactJs.
- Розбробка серверу для зв'язку бази данних та інтерфейсу з використанням NodeJs.

Використовуючи середовище для розробки календарного проєкту створимо план всього проєкту , а також плани для кожного спринту, щоб оптимізувати використання ресурсів та грошей. Використовуючи такі інструменти як:

1. Беклог задача.
2. Діаграма Гантта (Gantt Chart).
3. Діаграма задач.
4. Виконання робіт (Task Usage).
5. Використання ресурсів таких як час (Resource Usage).

Основні форми подання відомостей про проєкт:

1. Беклог задача.
2. Діаграма задач.
3. Діаграма ресурсів, яка може бути представлена в графічній формі.
4. Розподіл робіт, що визначає навантаженість фахівців у проміжку часу.

Для формування загальних даних у проэкті про роботи широко застосовуються ієрархічні структури організації даних. Найбільш важливою з них є ієрархічна структура праці, призначена для того, щоб забезпечити агрегацію необхідних для реалізації проєкту пакетів та плану робіт, розподіл бюджету, розподіл відповідальності фахівців.

Найважливішими видами ресурсів є: час та бюджет.

У якості базової методики обчислення головних показників графіка проєкту використовується добре зарекомендована діаграма вигорання задач яка демонструє кількість виконаних задач в залежності від запланованого часу. Маючи звіт діаграми вигорання задачо можно використовувати метод критичного шляху та сукупність методів і формул мережевого планування і управління, що забезпечують автоматичне обчислення для всіх робіт графіка моментів початку та закінчення, а також резервів часу. Роботи, які мають негативний або нульовий резерв часу, вважають що знаходяться на критичному шляху. Часто до складу критичного шляху включають роботи, що мають досить малий резерв часу, що не перевищує деякої заздалегідь заданої малої позитивної величини (порядок такої величини може бути порівнянний з точністю визначення часових показників робіт і проекту в цілому). Критичні роботи визначають термін завершення всього проекту і вимагають до себе підвищеної уваги з боку керівництва. У деяких системах управління проектами на додаток до методу критичного шляху можуть використовуватися і інші методи мережевого планування і управління (наприклад, методи статистичного моделювання тривалості робіт PERT або Monte-Carlo).

У цьому розділі було наведено основі элементами проєкту такі як, роботи, зв'язки між роботами, ресурси і їх призначення (ресурсів робіт), що буде формуватися з урахуванням істоти проєкту. Модель та описання методів та принців реалізації проєкту та основних підходів у відображені результатів, як графік , діаграма які формуються так, що всі роботи в проєкті відображають

технологічну послідовність їх виконання з урахуванням ієрархічної структури робіт проекту.

Висновки до розділу 2

В даному розділі проведено дослідження доцільності розробки програмного продукту, огляд основного функціоналу продукту, інформаційна модель з прикладом даних. Здійснено порівняння з існуючими аналогами, і цим показано, що таке проектне рішення має переваги в порівнянні з аналогами, зокрема: надійність, простота використання, гнучкість, зручність. Виконана постановка задачі та встановлені вимоги до якості розподілення ресурсів на розробку веб–додатку для організації та контролю робочого процесу компанії. Огляд основних помилок які вирішує додаток для управління та моніторінгу виконання задач.

Серед них можна виділити 2 основних:

- Дуже великий документ з задачами і їх описом.
- Невидимий прогрес.

Було зроблено вибір на користь нормалізованих та гнучких інструментів для управління та моніторінгу, так як вони є більш сучасними та точнішими у визначені стану виконання задачі. Було проведено аналіз основних методів роботи з данними, а саме розбиття та збереження данних корестувача на об’єкти. Основноб перевагою розбиття на об’єкти є те, що в ньому також виконується сегментація та класифікація розбитих об’єктів.

Виконаний опис системи та разроблених рішеннь з організаційного забезпечення.

РОЗДІЛ 3. ТЕХНІЧНА РЕАЛІЗАЦІЯ

3.1 Загальна структура автоматизованої системи управління задачами

Автоматизована система – це набір або сукупність засобів для обробки,зберігання, передачі інформації згідно потреб користувача, в якій частину функцій коригує користувач, а інша частина виконується автоматично [50, 42]. Автоматизовані системні операції - це поєднання програмного та апаратного забезпечення, яке розроблено та запрограмовано для автоматичної роботи без необхідності надання оператором людини вводу даних та інструкцій для кожної операції. Автоматизовані системні операції застосовуються в широкому діапазоні застосувань, таких як системи управління та моніторингу, програми захисту даних, системи автоматизації на заводі, автоматизовані системи відповіді на повідомлення, тощо [14].

Однією з переваг автоматизованих системних операцій є [48]:

• Усуває ризик людських помилок.

• Підвищує продуктивність користувачів.

• Забезпечує стандартизовані операції.

• Забезпечує краще управління операціями та ведення проєкту.

Використання автоматизованих системних операцій економить працю, час та витрати, підвищуючи при цьому точність виконуваної роботи. Це підвищує доступність, ефективність та надійність наданих послуг.

Дана автоматизована система моніторингу та стану управління задачами для розробки проєкту з урахуванням найважливіших проблем ведення проєктів. Правильно проведений розподіл задач для цього комплексу може мінімізувати і прискорити оцінки часу для реалізації ідеї. Архітектура системи(див. рис. 3.1.).

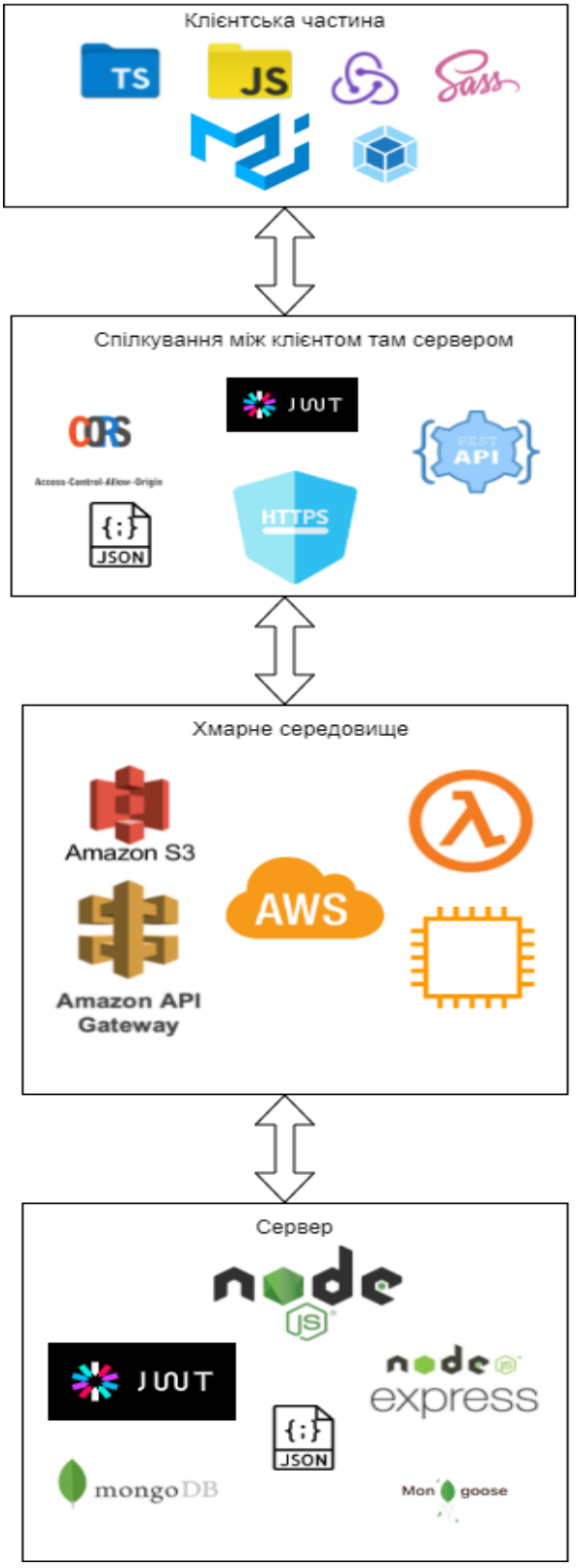

Рис. 3.1 Архітектура додатку.

Клієнтська частина представлена у вигляді сайту, до якої мають доступ користувачі, яким було надано відповідний дозвіл і налаштовано їх профіль. На сайті відображається інформація про проєкт та задачі.

Клієнтська частина:

- JavaScript – основа мови яку розуміють браузери.
- TypeScript – абстракція над JavaScript для використання типізації даних
- Redux – бібліотека для динамічного зв’язку даних з клієнтською частиною.
- Sass – бібліотека для написання стилів для інтерфейсу.
- Material Ui – бібліотека для інтегрування стилів у JavaScript.
- ReactJs – бібліотека для представлення частин інтерфейсу у вигляді функцій.
- Webpack – сборщик для JavaScript та додаткових бібліотек у декілька мінімізованих файлів.

Серверна частина в свою чергу служить для обробки інформації, а також передачі обробленої інформації через мережу інтернет на клієнтську частину. Серверна частина складається з декількох модулів.

AWS:

- S3 bucket – зберігає клієнтську частину.
- Lambda – функція – виконує верифікацію запросів між сервісами.
- Ec2 – віртуальна машина для запуску сервера.
- Amazon API Gateway – абстракція для безпечних запитів між сервером для користувачів.

Сервер:

- NodeJs – платформа для написанная серверу на JavaScript.
- ExpressJs – бібліотека для зручного користування Node Js API.
- JWT – технологія кодування запитів.
- MongoDB - база даних.

- Mongoose – бібліотека для написання запитів до бази даних.

В даному розділі було запроповано загальну архітектуру автоматизованої системи моніторингу та керування з використанням сучасних інструментів програмування та преведенно назви інструментів.

3.2 Опис технологій клієнтської частини

JavaScript - це мова програмування, яка використовується переважно веб-браузерами для створення динамічної та інтерактивної взаємодії з користувачем. Більшість функцій та програм, які роблять Інтернет незамінним у сучасному житті, написані в тій чи іншій формі на JavaScript [37].

Найбільш ранні втілення JavaScript були розроблені наприкінці 1990-х для веб-браузера Netscape Navigator. В той час веб-сторінки були статичними, пропонуючи мало взаємодії з користувачем, крім клацання посилань і завантаження нових сторінок. Вперше JavaScript увімкнув анімацію, адаптивний контент та перевірку форми на сторінці.

Протягом багатьох років JavaScript працював лише в обмеженій кількості браузерів. Internet Explorer від Microsoft, найбільша база браузерів, не підтримував JavaScript набагато пізніше. Натомість Microsoft створила власний пропрієтарний клієнтський скрипт під назвою JScript. На зорі веб-розробки програмісти, які бажали створювати динамічні веб-сайти, часто змушені були вибирати одне сімейство браузерів замість іншого. Це було далеко не ідеально, тому що через це Інтернет став менш доступним.

JavaScript не став стандартизованим і широко застосовувався до 1999 року. Навіть після стандартизації сумісність браузерів залишалася проблемою понад десять років.

JavaScript – це так званий клієнтський скрипт. Більшість веб-програм, таких як пошукова система, працюють завдяки взаємодії між пристроєм

користувача (наприклад, комп'ютером, телефоном або планшетом) і віддаленим сервером. Програмне забезпечення на віддаленому сервері надсилає інформацію клієнту (тобто машині користувача), а програмне забезпечення на стороні клієнта зчитує інформацію та відображає веб-сторінку на екрані [24, 31].

Клієнтський сценарій - це мова програмування, яка повністю виконує свої завдання на клієнтській машині і не потребує взаємодії з сервером для роботи [34, 35]. Наприклад, якщо на вашому комп'ютері завантажена веб-сторінка, а ваш інтернет-провайдер не працює, ви все одно можете взаємодіяти з веб-сторінками, які вже завантажені у вашого браузера. Однак ви не зможете переходити на нові веб-сторінки або отримувати доступ до будь-яких даних, розміщених віддалено [24].

Деякі з покращень динамічного веб-сайту, виконані за допомогою JavaScript:

- Автозаповнення.
- Завантаження нового контенту або даних на сторінку без перезавантаження сторінки.
- Ефекти стилів, меню.
- Анімація елементів сторінки, таких як згасання, зміна розміру або переміщення.
- Відтворення аудіо та відео.
- Перевірка введення з веб-форм.
- Усунення проблем сумісності з браузером.

Хоча JavaScript є клієнтською мовою, деякі з найбільш потужних функцій включають асинхронну взаємодію з віддаленим сервером. Асинхронний просто означає, що JavaScript може взаємодіяти із сервером у фоновому режимі, не перериваючи взаємодії користувача, що відбувається на передньому плані.

Візьмемо, наприклад, пошукову систему. Сьогодні багато пошукових систем мають функцію автозаповнення. Користувач починає вводити слово у

полі пошуку, і нижче з'являється список можливих умов пошуку чи фраз. Досвід бездоганний. Пропоновані умови пошуку відображаються без перезавантаження сторінки.

У фоновому режимі JavaScript зчитує літери при введенні користувачем, відправляє ці листи на віддалений сервер, а сервер відправляє пропозиції назад.

Програмне забезпечення на стороні сервера аналізує слова та запускає алгоритми, щоб передбачити пошуковий запит користувача. Такі програми диявольськи великі та складні. JavaScript на клієнтській машині максимально простий і малий, щоб не уповільнювати взаємодію користувача. Зв'язок між JavaScript та серверною програмою обмежений пропускною здатністю користувача. Ось чому розробники ставлять в основу ефективність функцій JavaScript і роблять обсяг даних, що передаються між програмами, якнайменше.

Тільки після того, як користувач вибирає пошуковий запит, вся сторінка перезавантажується та видає результати пошуку. Такі двигуни, як Google, зменшили або усунули необхідність перезавантаження навіть на цьому етапі. Вони просто роблять результати, використовуючи той самий асинхронний процес [17].

TypeScript - це надмножина JavaScript, що означає, що він містить усі функції JavaScript, а потім деякі. Отже, будь-яка програма, написана на допустимому JavaScript, також працюватиме належним чином у TypeScript. Фактично, TypeScript просто компілюється у простий JavaScript. Отже, у чому різниця? TypeScript пропонує нам більше контролю над нашим кодом за допомогою анотацій типів, інтерфейсів та класів. JavaScript має динамічну типізацію. Отже, програми, написані на JavaScript, не знають тип даних змінної, поки цій змінній не буде присвоєно значення під час виконання. Змінною можна перепризначити або примусово надати значення іншого типу без проблем або попереджень. Це може призвести до помилок, які часто не беруться до уваги, особливо у великих додатках.

TypeScript, з іншого боку, використовує статичну типізацію. Змінним можна присвоїти тип при їх оголошенні. TypeScript перевірить типи під час компіляції та видасть помилку, якщо змінній будь-коли буде присвоєно значення іншого типу. Однак помилка не перешкоджає виконанню коду. Код, як і раніше, буде скомпільований у простий JavaScript і працюватиме нормально. Таким чином, TypeScript є свого роду "перевіркою орфографії" для вашого коду. Він повідомить, коли щось виглядає не так, але не змінить спосіб роботи вашого коду.

Статична типізація TypeScript не обов'язкова. Змінним можна присвоїти тип any, що дозволить його значенням бути будь-яким типом. Якщо тип не вказано, за промовчанням буде встановлено будь-який тип [15].

Redux - це контейнер з передбачуваним станом, призначений для допомоги в написанні програм JavaScript, які однаково працюють у клієнтському, серверному та нативному середовищах і легко тестуються.

Хоча він в основному використовується як інструмент керування станом з ReactJs, ви можете використовувати його з будь-якою іншою платформою або бібліотекою JavaScript, як-от Angular, Vue.js і…. Він легкий - 2 КБ (включно з залежністю), тому вам не потрібно турбуватися про те, що він збільшить розмір активу вашої програми.

З Redux стан вашої програми зберігається в сховищі, і кожен компонент може отримати доступ до будь-якого стану, який йому потрібний, з цього сховища.

Redux допомагає вам впоратися з керуванням загальним станом, але, як і будь-який інший інструмент, він має компроміси. Він не призначений для використання як найшвидший або найкоротший спосіб написання коду. Він має допомогти відповісти на запитання «Коли певна частина стану змінилася і звідки прийшли дані?» З передбачуваною поведінкою. Потрібно написати більше коду та вивчити більше концепцій. Він також додає деяку непрямість до вашого коду і просить вас дотримуватися певних обмежень. Це компроміс між короткостроковою та довгостроковою продуктивністю [24, 19].

Переваги Redux:

Передбачуваність результату - завжди є одне джерело істини, сховище без плутанини в тому, як синхронізувати поточний стан з діями та іншими частинами програми.

Ремонтопридатність - наявність передбачуваного результату та суворої структури спрощує супровід коду.

Організація - redux більш строгий щодо того, як повинен бути організований код, що робить код більш узгодженим та легшим для команди.

Серверний рендеринг - це дуже корисно, особливо для початкового рендеринга, для кращої взаємодії з користувачем або пошукової оптимізації. Просто передайте сховище, створене на сервері клієнтській стороні.

Інструменти розробника -розробники можуть відстежувати все, що відбувається в додатку в режимі реального часу, від дій до змін стану.

Спільнота та екосистема - це величезний плюс, коли ви вивчаєте чи використовуєте якусь бібліотеку чи фреймворк. Наявність спільноти, що стоїть за Redux, робить її ще більш привабливою у використанні.

Легкість тестування - перше правило написання тестованого коду – писати невеликі функції, які роблять лише одне та які незалежні. Код Redux в основному складається з простих, невеликих та ізольованих функцій.

Scss - це каскадні таблиці стилів. Scss можна розділяти крапкою з комою та запускати в одному рядку.

SCSS - це препроцесор, який дозволяє використовувати функції, які ще не є частиною ширшого стандарту CSS, і забезпечує більш ефективні робочі процеси для підтримки ваших таблиць стилів. З препроцесором SCSS ви можете скоротити кількість повторень і гарантувати, що ви пишете чистий код, що підтримується на майбутнє. Scss може приймати css код і працювати. SCSS повністю сумісний із синтаксисом CSS, але при цьому підтримує всі можливості Sass.

Scss – це розширення синтаксису CSS. Це означає, що кожна допустима таблиця стилів CSS є допустимим файлом SCSS з тим самим значенням. Крім

того, SCSS розуміє більшість хаків CSS та синтаксис, що залежить від постачальника, наприклад, старий синтаксис фільтрів IE. Цей синтаксис розширено функціями Sass, наведеними нижче. Файли, які використовують синтаксис, мають розширення .scss [24].

Material UI - це інтерфейсний фреймворк з відкритим вихідним кодом для компонентів React, який має понад 60500 зірок на github. Він побудований за допомогою Less. Less (розшифровується як Leaner Style Sheets) - це зворотне сумісне мовне розширення для CSS. Інтерфейс користувача Material заснований на Material Design від Google, щоб забезпечити високу якість цифрового інтерфейсу при розробці інтерфейсної графіки. Material Design фокусується на створенні сміливих та чітких дизайнів – він створює текстури, фокусуючись на тому, як компоненти відкидають тіні та відбивають світло.

Є кілька ключових переваг проєктування з використанням Material UI:

Деякі інтерфейсні фреймворки не дуже добре документовані, що ускладнює розробку з ними. Однак у Material UI є докладна документація, яка полегшує навігацію по фреймворку.

Користувальницький інтерфейс матеріалів постійно оновлюється. На момент написання цієї статті останнім оновленням була версія 4.11.0 (від 1 липня 2020 р.).

Всі компоненти єдині за дизайном і кольорами, що дозволяє розробленому додатку / веб-сторінці виглядати естетично [24].

Webpack – це збирач статичних модулів для додатків JavaScript – він бере весь код з вашої програми та робить його придатним для використання у веб-браузері. Модулі - це фрагменти коду, що багаторазово використовуються, створені з JavaScript вашої програми, node_modules, зображень і стилів CSS, які упаковані для зручного використання на вашому веб-сайті. Webpack розділяє код залежно від того, як він використовується у вашому додатку, і з цим модульним поділом обов'язків стає набагато простіше керувати, налагоджувати, перевіряти та тестувати ваш код. Приклад збирання коду(див. рис. 3.2) [24].

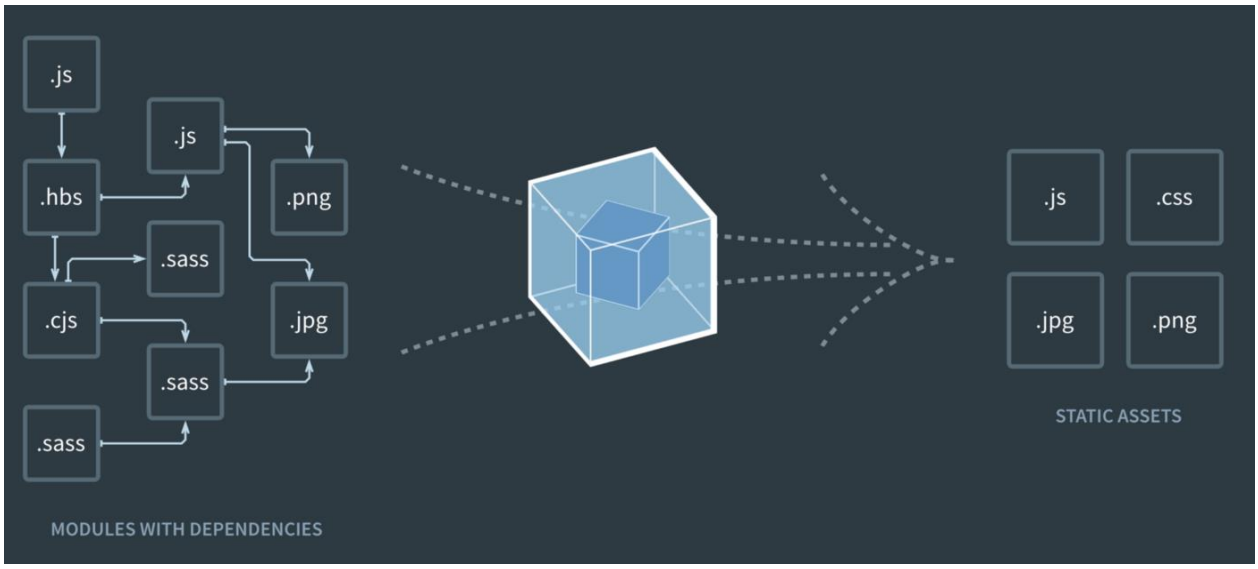

Рис. 3.2 Діаграма збирання залежностей

Коли Webpack обробляє вашу програму, він будує граф залежностей, який відображає модулі, необхідні вашому проекту, і генерує один або кілька пакетів. Пакет - це окрема група зв'язаного коду, яка була скомпільована та перетворена для браузера.

Якщо один файл залежить від іншого (він використовує код окремого файлу), Webpack розглядає це як залежність. Webpack також приймає ваші активи, не пов'язані з кодом (зображення, шрифти, стилі і т.д.), і перетворює їх залежно для вашої програми.

3.3 Вибір фреймворку для створення інтерфейсу користувача

React - це каркас який приходить на допомогу багатьом розробникам інтерфейсу користувача. Співтовариство React налічує понад 56 000 розробників, і щонайменше 8 787 лідерів галузі використовують популярну бібліотеку JavaScript[47].

React був розроблений Facebook, зокрема, інженером-програмістом Джорданом Уолке. Це була ініціатива, спрямована на те, щоб зробити розробку користувальницького інтерфейсу на багато проще, чим це було

раніше. В результаті Джордан Уолк розробив компонентну бібліотеку JavaScript з відкритим кодом - React. React знімає навантаження з розробниками, дозволяючи створювати компоненти з відстежуваним станом, які відображаються відповідно до даних часу виконання.

React працює з віртуальним DOM. Немає необхідності оновлювати всі компоненти на сторінці всякий раз, коли користувач взаємодіє з нею. Це дає React перевагу для динамічних веб-сайтів, продуктивність яких падає при звичайному оновленні DOM[37]. З React також немає кошмара кодування. Він створений з урахуванням можливостей повторного використання, а розробники можуть мати нові функції, не переписуючи існуючого функцыоналу. Компоненти інтерфейсу користувача, створені за допомогою React легко відлажувати. Завдяки всім цим перевагам React швидко зміцнився в якості переважного інструменту розробки інтерфейсу користувача серед розробки JavaScript. Популярна бібліотека завантажувалась 7 мільйонів разів за тиждень. Розробка з React також є в мобільній сфері. У 2015 році Facebook випустив React Native, призначений для фронтенд-розробки на iOS, Android та UWP. Facebook оновив алгоритм бібліотеки React у 2017 році, представив React Fiber. Не дивно, що розробники зробили React своїм кращим інструментом для створення потужних веб-візуальних елементів. З моменту свого заснування React привернув увагу багатьох компаній. Візуальні матеріали мають вирішальне значення для роботи користувачів, але вони мають бути збалансовані з ефективною розробкою та програмуванням. React довів, що він ідеально підходить для веб-розробників і розробників інтерфейсу користувача, щоб розробляти дуже потужні та красиві компоненти. React дуже люблять за його неглибоке навчання, економію часу та компоненти, що реагують. На сьогоднішній день понад 2 мільйони веб-сайтів створено за допомогою React, і немає жодних ознак зупинки. Серед них деякі належать до впізнаваних імен у різних галузях.

Це популярні компанії, які зробили React основною частиною свого інтерфейсу:

- Netflix
- Facebook
- Instagram
- Airbnb
- Cloudflare
- Dropbox
- BBC
- Flipboard
- Imgur
- Postmates
- Reddit

React, безсумнівно, є універсальним інструментом для розробки інтерфейсу користувача, який дозволить створити візуально привабливий веб-сайт за короткий час. Це позбавляє вас необхідності занурюватися в складні коди JavaScript, що забирає багато часу.

Netflix - чудовий приклад того, як створення багатофункціональних інтерфейсів може призвести до високої залученості користувачів. Це дозволяє користувачам переглядати прев'ю трейлерів з основного каналу та за бажанням переглядати додаткову інформацію. Багаті інтерфейси - це чуйність і взаємодія.

До React веб-сайти були сумно відомі затримками та ресурсами під час відображення складних динамічних сайтів. Взаємодія з користувачем - це біль, оскільки кожне клацання призводить до нестерпно довгого оновлення інтерфейсу користувача. Вина, звісно, полягає у тому, як браузер оновлює DOM через HTML. DOM означає об'єктну модель документа, яка є ієрархічною структурою, що показує складні відносини різних компонентів інтерфейсу користувача. Невелика зміна одного елемента призведе до того, що

браузер оновить всі компоненти інтерфейсу користувача. Коли ви використовуєте React, браузер використовує віртуальний DOM замість реального DOM. У віртуальній моделі DOM оновлюються лише ті об'єкти, які були змінені, інші залишаються без змін. Це набагато ефективніший метод, що дозволяє прискорити рендеринг. Оскільки рендеринг значною мірою автоматизований, розробники інтерфейсу користувача можуть створювати більш компактні і зрозумілі функції. Розробники можуть зосередитися на створенні багатофункціональних інтерфейсів, не турбуючись про налагодження складних зв'язків. Коли кажуть про односторінкові додатки або про ізоморфний JavaScript, мається на увазі, що можна використовувати один і той же код як в середовищі де виконується серверний код, так і в браузері кіонцевого користувача. Коли користувач відкриває сайт у своєму браузері, вміст сторінки може бути згенерований на сервері та відправлений у вигляді HTML сторінки чи повний пакет JavaScript функцій який почне виконуватись після повної загрузки даних. У випадку з SPA-додатками (Single Page Application) це підхід коли додаток генерується на сервері як наприклад у PHP то це може зайняти деякий час. Під час повного завантаження сторінки користувачі бачать або порожню сторінку, або анімацію завантаження. З огляду на сучасні тренди який диктує Google, що за очікування протягом більш ніж двох секунд може бути досить помітним та не зручним для користувача і для цього додатку буде низький рівень показу в пошуковій системі, скорочення часу завантаження може виявитися вкрай важливим для сучасних бізнес додатків. Вагомою проблемою є то що пошукові машини не індексують такі сторінки так добре як звичайні HTML сторінки тому виконання JavaScript коду в серверному середовищі допомагає виправити проблеми з індексацією вашого додатку.

Якщо створюється додаток з допомогою бібліотеки React, можна отримати помітну вигоду у вигляді інструментів які генерують його на стороні вашого серверу. Після завантаження HTML сторінки з часткою данних ви все ще можете вказати ті частини які потрібно показати пізніше шо значного

прискорить рендеринг компонентів. Така можливість рендеринга сторінок як на сервері так і на клієнті називають ланцюгом де загальна сборка розбиваються на маленьки частини і потрішки відправляються кліенту, таким чином є можливість досягнути кращого індексування сторінок пошуковими машинами і поліпшення користувацького досвіду. Більш того, такий підхід дозволяє знизити час, що витрачається на розробку.

При використанні інших сучасних фреймворків шо мають ізоморфні можливості, то ви повинні створювати окремий функціонал для серверной генерації та клієнтської частина, які повинні рендерити на стороні сервера, а також шаблони для клієнтської сторони додатка. React дозволяє фахівцям створювати функціональні компоненти, які працюють як і на сервері так і на клієнтський частині. При цьому підході не здійснюється регулярне оновлення DOM. Завдяки такій філософії можуть створювути високо продуктивніс React додатки. Мобільні додатки одного і того ж самого проєкту можуть мати деякі переваги в порівнянні з сайтами. Наприклад більш юзерів дуже обережно відноситься до оплати підписки вашего сервісу через сайт на телефоні, але коли це веріфікований додаток який знаходиться в маркеті популярних платформ, це рішення дається їм набагато простіше також додаток можна використовувати без підключення до Інтернету. Вони мають доступ до багатьох функцій пристрою, такі як спливаючі повідомлення, пам'ять телефону, камера і тд. React Native - це платформа яка дуже схожа на React своїм підходом але компілюєтсья в мобільний пакет який працює на всіх відомих мобільних платформах. Бізнес логіка додатку створюється на мові програмування JavaScript, таким чином, фахівцю з веб розробки не потрібно відмовлятися від звичних прийомів веб-розробника та вчити нову мову.

Крім швикдого процессу створення додатку та можливості перевикористання коду, React дозволяє створювати якісні додатки за не дуже великий часовий період. Якщо ви створюєте добре описані функціональні частини інтерфейсу, які потім можно перевикористати, завдяки цьому вам прийдеться писати менше коду і в майбутньому легко підтримувати такий

продукт. Чим менше нового коду вам потрібно, тим більша стабільність застосунку та шанс виникнення некоректної поведінки програми нижче. Компонентно-орієнтований підхід у сучасній веб розробці дає можливість з легкістю змінювати компоненти і перевикористати код, перетворюють React в потужний інструмент. Добре написані компоненти які були створені під час роботи не мають додаткових залежностей. Таким чином, ніщо не заважає використовувати їх знову і знову в проектах різного типу.

3.4 Технології для сервера

AWS – це сервіс надання обчислень в «хмарі». На сьогоднішній день це найпопулярніший сервіс для надання послуг у сфері хмарних обчислень. Ця система дозволяє користувачам, а також великим компаніям отримувати доступ до своїх сервісів через мережу інтернет[19]. Сервіс заснований на великій кількості серверів(комп’ютерів), які надають свої послуги по всьому світу. Вони включають в себе віртуальні машини, бази даних, сховища, віртуальні приватні мережі. Сервери або так звані географічні зони є по всьому світу і їх кількість зростає, завдяки все більшим потребам користувачів. AWS гарантує надійність, безпеку і захищеність приватних даних зі свого боку. Також до окремих плюсів слід віднести гнучку політику платежів. Можна здійснювати як платежі за використання погодинно якогось сервісу, або ж за певний об’єм роботи або зайнятості пам’яті. Існує також можливість створення безкоштовного акаунту з обмеженням до певних сервісів. Користувачі мають у своєму використанні цілий кластер віртуальних машин включаючи апаратні пристрої через мережу інтернет [21].

Amazon Simple Storage Service (Amazon S3) - це сервіс зберігання об'єктів, що пропонує кращі в галузі показники продуктивності, масштабованості, доступності та безпеки даних.[19] Це означає, що клієнтами

можуть бути компанії будь-яких розмірів і з будь-яких областей діяльності. Вони можуть використовувати наш сервіс для зберігання і захисту будь-яких обсягів даних в різних ситуаціях, наприклад для забезпечення роботи сайтів, мобільних додатків, для резервного копіювання та відновлення, архівації, корпоративних додатків, пристроїв IoT і аналізу великих даних. Amazon S3 пропонує прості у використанні інструменти адміністрування, які дозволяють організувати дані і точно налаштувати обмеження доступу відповідно потребам вашого бізнесу або законодавчими вимогами. Amazon S3 забезпечує надійність +99,999999999% (тут 11 дев'яток) і зберігає дані мільйонів додатків в інтересах компаній з усього світу. Amazon S3 пропонує різні інструменти, які дозволяють організувати і контролювати для підтримки певних сценаріїв використання, скорочення витрат, забезпечення безпеки і дотримання 32 законодавчих вимог. Дані зберігаються як об'єкти в ресурсах, які називають кошиками, при цьому розмір одного об'єкта може становити до 5 ТБ.

Сховище S3 дозволяє додавати теги метаданих в об'єкти, переміщати і зберігати дані в класах сховища S3, налаштовувати і застосовувати елементи управління доступом до даних, захищати дані від несанкціонованого використання, застосовувати аналітику великих даних і відстежувати дані на рівні об'єкта і кошики. Для захисту даних в Amazon S3 за замовчуванням користувачам надається тільки доступ до ресурсів S3, які створили вони. Доступ іншим користувачам можна надати за допомогою одного або декількох з наступних можливостей доступу в приміщення: AWS Identity and Access Management (IAM) для створення користувачів і управління правами доступу, списки управління доступом (ACL) для надання доступу до окремих об'єктів авторизованим користувачам, політики кошика для настройки дозволів для всіх об'єктів в одному кошику S3 і аутентифікація рядка запиту для надання тимчасового доступу іншим користувачам за допомогою короткострокових URL-адрес. Amazon S3 також підтримує журнали аудиту, які містять запити до ресурсів S3 для забезпечення повної візуалізації дій користувачів і даних, які вони запитують [21].

AWS Lambda - це подієво-орієнтований сервіс безсерверних обчислень, який дозволяє виконувати код без виділення і адміністрування серверів і доповнювати інші сервіси AWS на основі користувальницької логіки. Lambda автоматично реагує на різні події (так звані тригери), наприклад на HTTP запити через Amazon API Gateway, зміна даних в кошиках Amazon S3 або таблицях Amazon DynamoDB; або можна запустити свій код через виклики API, використовуючи AWS SDK і переходи між станами в AWS Step Functions. 33 Lambda виконує код на обчислювальній інфраструктурі і повністю відповідає за адміністрування нижче лежачої платформи, включаючи обслуговування серверів і операційної системи, виділення ресурсів, автоматичне масштабування, моніторинг коду і ведення журналів. Тобто вам досить завантажити свій код і налаштувати, як і коли він повинен виконуватися. У свою чергу, сервіс подбає про його запуск і забезпечує високу доступність вашого застосування.

AWS Lambda - це зручна обчислювальна платформа, що підходить для безлічі сценаріїв застосування, зрозуміло, якщо мова і середовище виконання вашого коду підтримуються сервісом. Якщо ви хочете зосередитися на коді і бізнес-логікою, доручивши обслуговування серверів, виділення ресурсів і масштабування сторонньому постачальнику за розумні гроші, вам точно варто перейти на AWS Lambda. Lambda ідеально підходить для створення програмних інтерфейсів, а якщо використовувати сервіс разом з API Gateway, можна значно скоротити витрати і швидше вийти на ринок.

Є різні способи використання функцій Lambda і варіанти організації без серверної архітектури - кожен може вибрати щось підходяще з урахуванням поставленої мети. Lambda дозволяє виконувати широкий спектр завдань. Так, завдяки підтримці CloudWatch можна створювати відкладені завдання і автоматизувати окремі процеси. Немає ніяких обмежень за характером і інтенсивністю використання сервісу (враховуються витрати пам'яті і час), і вам ніщо не заважає планомірно працювати над повноцінним мікросервісом на основі Lambda. Тут можна створювати сервіс-орієнтовані дії, які не

виконуються постійно. Типовий приклад - масштабування зображень. Навіть в разі розподілених систем функції Lambda не втрачають своєї актуальності.

Отже, якщо ви не хочете займатися виділенням і адмініструванням обчислювальних ресурсів - спробуйте AWS Lambda; якщо вам не потрібні важкі, ресурсомісткі обчислення - також спробуйте AWS Lambda; якщо ваш код виконується періодично - все правильно, вам варто спробувати AWS Lambda. Основна перевага Lambda полягає в тому, що, виконуючи функцію від вашого імені, сервіс сам виділяє необхідні ресурси. Ви можете не витрачати час і сили на 34 адміністрування систем і зосередитися на бізнес-логіці і написанні коду.

Сервіс Lambda розділений на дві площини. Перша - площина управління. Згідно Вікіпедії, площина управління (control plane) - це частина мережі, що відповідає за транспортування сигнального трафіку і маршрутизацію. Вона є головним компонентом, які приймають глобальні рішення про виділення, обслуговування і розподіл робочих навантажень. Крім того, площина управління виступає в ролі мережевої топології постачальника рішення, що відповідає за маршрутизацію трафіку і управління ним. Друга площина - площина даних. У неї, як і у площині управління, свої завдання.

Площина управління надає API для управління функціями (CreateFunction, UpdateFunctionCode) і контролює взаємодію Lambda з іншими сервісами AWS. Площина даних управляє API викликів (Invoke API), який запускає функції Lambda. Після виклику функції площину управління виділяє або вибирає існуючу, заздалегідь підготовлену для цієї функції середу виконання, а потім виконує в ній код. AWS Lambda підтримує безліч мов програмування, включаючи Java 8, Python 3.7, Go, NodeJS 8, .NET Core 2 і інші, через відповідні середовища виконання. AWS регулярно їх оновлює, поширює виправлення безпеки і виконує інші операції з обслуговування цих середовищ.

Lambda дозволяє використовувати і інші мови за умови, що ви самі впровадите відповідне середовище виконання. І тоді вже вам доведеться займатися її обслуговуванням, в тому числі стежити за безпекою. Поки що, до

безпеки немає нарікань. З іншого боку, оскільки від користувача керованого середовища виконання AWS Lambda приховані багато внутрішніх процесів і особливості реалізації цієї моделі, деякі загальноприйняті правила хмарної безпеки втрачають актуальність. Як і більшість сервісів AWS, Lambda надається за принципом спільної відповідальності AWS і клієнта по частині безпеки і дотримання нормативних вимог. Цей принцип знижує операційну навантаження на клієнта, оскільки AWS бере на себе завдання обслуговування, адміністрування та контролю компонентів сервісу - від 35 операційної системи хоста і рівня віртуалізації до фізичної безпеки об'єктів інфраструктури. Якщо говорити конкретно про AWS Lambda, то AWS відповідає за управління нижчележачою інфраструктурою, пов'язаними базовими сервісами, операційною системою і платформою додатків. У той час як клієнт несе відповідальність за безпеку свого коду, зберігання конфіденційних даних, контроль доступу до них, а також до сервісу і ресурсів Lambda (Identity and Access Management, IAM), в тому числі в межах використовуваних функцій [21].

Ec2 - означає Amazon Elastic Compute Cloud. Amazon Ec2 це базова віртуальна машина з апаратними компонентами, що настроюються, і ОС. Система дозволяє запускати різні віртуальні машини та керувати ними за допомогою одного обладнання [21].

Elastic Compute Cloud - це широко використовувана та основна сервісна система в потужній екосистемі AWS. Хмарна система надає безліч функцій, наприклад, вона спрощує обчислення на запит і масштабує обчислювальні потужності в хмарній системі Amazon [21, 22].

Інстанси Amazon звільняють вас від додаткових авансових вкладень у обладнання. Також немає зайвого багажу, пов'язаного із обслуговуванням орендованого обладнання. Універсальне віртуальне обладнання просте у використанні та дозволяє створювати і запускати програми з вищою швидкістю. Адаптація хмари еластичних обчислень AWS дозволяє запускати

декілька віртуальних серверів. Він також забезпечує керування збільшенням або зменшенням масштабу відповідно до швидкості трафіку сайту.

Крім того, система працює з багатотонними робочими навантаженнями і здатна виділяти та ініціалізувати ресурси відповідно до поточного попиту. Ця поведінка системи, що адаптується, складає слово «еластичний» в Elastic Computing Cloud.

Примірники Ec2 – це віртуальні середовища, відключені від базової служби на запит. Тут користувач ec2 може орендувати віртуальний сервер або так званий екземпляр відповідно до вимог та ефективно переміщати/підключати до нього програми. Зазначене вище, використовуючи інстанси ec2, можна легко збільшувати або зменшувати в залежності від динаміки відвідуваності веб-сайту. Примірники AWS Ec2 також усувають вашу залежність від додаткових інвестицій в апаратне та програмне забезпечення та управління.

Amazon API Gateway - це повністю керований сервіс, який дозволяє розробникам легко створювати, публікувати, підтримувати, відстежувати та захищати API у будь-якому масштабі. API-інтерфейси виступають як «вхідні двері» для додатків для доступу до даних, бізнес-логіки або функцій ваших серверних служб. Використовуючи API Gateway, можна створювати RESTful API і WebSocket API, які дозволяють використовувати додатки двостороннього зв'язку в реальному часі. API Gateway підтримує контейнерні та безсерверні робочі навантаження, а також веб-додатки [21].

API Gateway виконує всі завдання, пов'язані з прийомом та обробкою до сотень тисяч одночасних викликів API, включаючи керування трафіком, підтримку CORS, авторизацію та контроль доступу, регулювання, моніторинг та керування версіями API. API Gateway не має мінімальних зборів чи початкових витрат. Ви сплачуєте за отримані дзвінки API та обсяг даних, що передаються, а за допомогою багаторівневої моделі ціноутворення API Gateway ви можете знизити свої витрати в міру збільшення масштабів використання API [21, 39].

Node.js – це серверна платформа, побудована на движку Google Chrome JavaScript Engine (V8 Engine). Node.js був розроблений Райаном Далем у 2009 році, а його остання версія – v16.13.0.

Node.js - це платформа, побудована на середовищі виконання JavaScript Chrome для простого створення швидких та масштабованих мережевих додатків. Node.js використовує керовану подіями неблокуючу модель введення-виводу, яка робить його легким і ефективним, ідеально підходить для додатків з інтенсивним використанням даних у реальному часі, які працюють на розподільчих пристроях. Node.js - це кросплатформне середовище виконання з відкритим вихідним кодом для розробки серверних та мережевих програм. Програми Node.js написані на JavaScript і можуть запускатися в середовищі виконання Node.js у OS X, Microsoft Windows і Linux. Node.js також надає багату бібліотеку різних модулів JavaScript, яка значно спрощує розробку веб-додатків з використанням Node.js.

Нижче наведено деякі з важливих функцій, які роблять Node.js найкращим вибором для архітекторів програмного забезпечення.

Асинхронний та керований подіями - всі API бібліотеки Node.js є асинхронними, тобто не блокуючими. По суті це означає, що сервер на основі Node.js ніколи не чекає, поки API поверне дані. Сервер переходить до наступного API після його виклику, і механізм сповіщення подій Node.js допомагає серверу отримати відповідь від попереднього виклику API.

Бібліотека Node.js, створена на движку Google Chrome V8 JavaScript Engine, дуже швидко виконує код.

Однопотоковий, але добре масштабований - Node.js використовує однопоточну модель із циклом подій. Механізм подій допомагає серверу реагувати неблокуючим чином і забезпечує високу масштабованість сервера на відміну традиційних серверів, які створюють обмежені потоки обробки запитів. Node.js використовує однопоточну програму, і та сама програма може обслуговувати набагато більше запитів, ніж традиційні сервери, такі як HTTP-сервер Apache [24].

Без буферизації - програми Node.js ніколи не буферизуються дані. Ці програми просто виводять дані частинами.

Ліцензія – Node.js випускається під ліцензією MIT.

Express.js - це безкоштовна платформа веб-додатків з відкритим кодом для Node.js. Він використовується для швидкого та простого проєктування та створення веб-додатків. Веб-програми - це програми, які можна запускати у веб-браузері. Оскільки Express.js вимагає лише JavaScript, програмістам і розробникам стає простіше створювати веб-додатки та API без будь-яких зусиль.

Express.js - це фреймворк Node.js, що означає, що більшість коду вже написана для роботи програмістам. Ви можете створювати односторінкові, багатосторінкові або гібридні веб-застосунки за допомогою Express.js. Express.js є легковажним і допомагає організувати веб-програми на стороні сервера в більш організовану архітектуру MVC. Щоб використовувати Express.js, важливо вивчити JavaScript та HTML. Express.js спрощує керування веб-застосунками. Він є частиною технології на основі JavaScript, званої програмним стеком MEAN, що означає MongoDB, ExpressJS, AngularJS і Node.js. Express.js є серверною частиною MEAN і керує маршрутизацією, сеансами, HTTP-запитами, обробкою помилок тощо. Бібліотека JavaScript Express.js допомагає програмістам створювати ефективні та швидкі веб-програми. Express.js розширює функціональність node.js. Фактично, якщо ви не використовуєте Express.js вам доведеться виконати багато складного програмування, щоб створити ефективний API. Він спростив програмування на node.js та надав безліч додаткових функцій [24, 46].

MongoDB - документ-орієнтована NoSQL база даних, що використовується для високих розмірів даних. Установи з використанням JSON структури документа, MongoDB створює зручні колекції даних.

Чому MongoDB ?

Орієнтована на документи - оскільки MongoDB є базою даних типу NoSQL, замість того, щоб мати дані у форматі реляційного типу, вона зберігає

дані у документах. Це робить MongoDB дуже гнучким та адаптованим до реальної ситуації та вимог ділового світу.

Спеціальні запити - MongoDB підтримує пошук за полями, діапазонами та пошук за регулярними виразами. Можуть бути зроблені запити повернення певних полів у документах.

Індексування - можна створювати індекси для підвищення продуктивності пошуку MongoDB. Будь-яке поле в документі MongoDB можна проіндексувати.

Реплікація - MongoDB може забезпечити високу доступність за допомогою наборів реплік. Набір реплік складається із двох або більше екземплярів mongo DB. Кожен член набору реплік може бути у ролі первинної чи вторинної репліки. Первинна репліка – це головний сервер, який взаємодіє з клієнтом та виконує всі операції читання/запису. Повторні репліки підтримують копію даних первинної репліки за допомогою вбудованої реплікації. При виході з ладу первинної репліки набір реплік автоматично перемикається на вторинний, потім стає первинним сервером.

Балансування навантаження - MongoDB використовує концепцію сегментування горизонтального масштабування шляхом поділу даних між декількома примірниками MongoDB. MongoDB може працювати на декількох серверах, балансуючи навантаження та/або дублюючі дані, щоб система залишалася працездатною у разі відмови обладнання [24, 37, 38].

В даному розділі було проаналізовані та описані технології для серверної частини додатку. Серверна частина системи моніторингу відповідає за збереження, та обробку отриманої інформації, а також її передачі на клієнтську чатсину. Для обробки інформації було обрано мову програмування JavaScript та платформа NodeJs, так як вона має найбільшу кількість бібліотек необхідних для обробки та сегментації данних, а також легко імпортується на різні хмарні сервіси і є легкою в освоєні.

Висновки до розділу 3

В даному розділі було запропоновано загальну архітектуру автоматизованої системи управління. Клієнтська частина додатку дозволяє відстежувати прогресс виконання задач в реальному часі з любого девайсу. Серверна частина системи моніторингу відповідає за збереження, та обробку отриманої інформації, а також її передачі на клієнтську частину. Клієнтська частина системи представлена інтерфейсом користувача у вигляді сервісу та можливими функціями перегляду минулих та теперішніх станів проєкту, а також можливість зручного керування. Для обробки інформації було обрано мову програмування JavaScript, так як вона має найбільшу кількість бібліотек необхідних для обробки даних у браузері, а також легко імпортується на різні хмарні сервіси і є легкою в освоєнні. “Хмарні” сервіси AWS в даній роботі використовуються у якості серверної частини. Зокрема, сервіс S3 використовується для збереження статичних даних сервісу. Lambda-функція в свою чергу є обчислювальним ресурсом, який здійснює обрахунки отриманої інформації в системі AWS. MongoDB – не релятивна база даних для збереження даних, які були передані з Lambda функції. Клієнтська частина написана на базі фреймворку React. На сьогоднішній день це один з найпопулярніших фреймворків, який постійно оновлюється, а також підтримується компанією Facebook.

ВИСНОВКИ

В роботі проведено дослідження та розробки додатку для управління проєкту. Проаналізовано та оглянуто основні технології для проведення моніторингу та відображення інформації для зручного користування, а також основних технологій обробку та збереження даних. Було розроблено архітектуру автоматизованої системи управління.

Серверна частина системи відповідає за збереження, та обробку отриманої інформації, а також її передачі на клієнтську частину.

Клієнтська частина представлена інтерфейсом користувача у вигляді веб-сайту та можливими функціями перегляду, редагування, генерування звітів, а також можливість завантаження свої фото.

Дана автоматизована система реалізована на мові програмування JS. В якості бібліотеки для сегментації зображення було використано бібліотеку React.js, Express.Js. Уся архітектура реалізована за допомогою хмарних сервісів Amazon Web Services.

Проведено опис та характеристику технологій за допомогою яких була розроблена дана автоматизована система. Було представлено блок-схем алгоритму роботи серверної частини, діаграму взаємодій та діаграму послідовностей.

Розроблена система є простою і логічною у використанні. Приведено порівняння з існуючими аналогами, і цим показано, що таке проектне рішення має переваги в порівнянні з аналогами, зокрема: надійність, простота використання, гнучкість, зручність. Згідно проведеного економічного обґрунтування дане проектне рішення є конкурентноздатним.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Анісімов С.Н., Анісімов Є.В. Управління проєктами. Український досвід: збірник наукових праць. 2015. Вип. 3 (16). С. 37-38.

2. Грашіна М.В. Дункан.В. Основи управління проєктами. Посібник: 2009 . - 632 с.

3. Фесенко Т.Г. Управління проєктами: теорія та практика виконання проєктних дій: навч. Посібник: Харьков: Біла К. О., 2019. – Розд. 6. – С. 410–417.

4. Пастернак І.В. Класифікація засобів модульної взаємодії між клієнтом і сервером. Посібник: 2019. С. 46-97.

5. Пастернак, Ю.В. Морозов. Вісник «Комп'ютерні системи та мережі»: НУ «Львівська політехніка», Львів , 2011.- №717.- С. 108-113.

6. Богданюк В.Є., Березовський К.В., Пашін В.П. Методичні вказівки до виконання організаційно-економічного розділу дипломних проектів. Київ : Таксон, 2016р.

7. Джефф Сазерленд. Scrum. Навчись робити вдвічі більше за менший час. Революционный метод управления: 2018р.

8. Корі Когон., Джеймс Вудс., Керування проєктами для «неофіційних» проект-менеджерів. Посібник: 2016. – Розд. 6. – С. 410–417.

9. Тарасенко Ф.П. Прикладний системний аналіз.

10. Тімоті Д. Джонсон. Microsoft Project 2016 Step by Step. 2018 Посібник: 2016. – Розд. 6. – С. 127–315.

11. Оліфіров О. В. Інформаційні системи в менеджменті / О.В. Оліфіров, Н.М. Спіцина, Т.В. Шабельник. – Донецьк: ДонНУЕТ, 2013. - 240с.

12. Джефф Сазерленд. Революційний метод управління проєктами. Посібник: 2012. – Розд. 2. – С. 12–19.

13. Расмуссон Д. Гнучке управління ІТ-проєктами. Керівництво для справжніх самураїв. 2015. Вип. 2 (11). С. 36-38.

14. Майк Кон. Оцінка та планування проєктами. Посібник: 2016. – Розд. 1. – С. 86–115.

15. Клепікова О. А. Сучасні технології моделювання бізнес-процесів підприємства / О. А. Клепікова // Наукові праці Донецького національного технічного університету. Сер.: Економічна . - 2014. - № 4. - С. 257-263.

16. Кен Швабер. Agile Project Management with Scrum. 2017р Посібник: 2017. С. 22-63.

17. Документація «Microsoft Project Tutorial for Beginners Smartshee» / [Електронний ресурс]. - Режим доступу: https://www.smartsheet.com/microsoft-project-2016-tutorial-newbies

18. Посібник «Project Management Tutorials for Beginners» / [Електронний ресурс]. Режим доступу: https://www.guru99.com/project-managementtutorial.html

19. Документація «Scrum Guide» / [Електронний ресурс]. Режим доступу: https://www.scrum.org/resources/scrum-guide

20. Аналіз цін на ринку проєктів. [Електронний ресурс]. Режим доступу: https://dou.ua/

21. Аналіз цін праці розробників. [Електронний ресурс]. Режим доступу: https://www.upwork.com/

22. Вольфсон Б.Л., Гнучке управління проєктами та продуктами. Посібник: 2013. 85–117.

23. Скотт Беркун. Проєктний менеджмент на практиці. Посібник: 2016. – Розд. 2. – С. 55–83.

24. Документація хмарного сервісу. [Електронний ресурс]. Режим доступу: https://aws.console.com

25. Документація бібліотеки. [Електронний ресурс]. Режим доступу: https://reactjs.org/

26. Документація додатку. «Kasandra» / [Електронний ресурс]. Режим доступу: https://www.process.st

27. Документація додатку. «Trello&Jira» / [Електронний ресурс]. Режим доступу: https://www.atlassian.com

28. Документація додатку. «Zoho» / [Електронний ресурс]. Режим доступу: https://www.zoho.com

29. Документація додатку. «Notion» / [Електронний ресурс]. Режим доступу: https://www.notion.so

30. Биков В.Ю., Лапінський В.В., Методологічні та методичні основи створення і використовування електронних засобів. Київ: 2012, 3-6с.

31. Алексейчук І.С. Про технологію створення системи тестування. Київ: 2013. 43-92с.

32. Автоматизація та комп'ютерно-інтегровані технології. [Електронний ресурс]. Режим доступу: http://conference.ikto.net

33. Як захистити веб-додаток: основні поради, інструменти. [Електронний ресурс]. Режим доступу: https://tproger.ru/translations/webapp-security/

34. Управління контентом в організації. [Електронний ресурс]. Режим доступу: https://soferblog.ru/uk/remont/upravlenie-kontentom-vorganizacii-vybor-sistemy-upravleniya-kontentom.html

35. Дейв Крейн., Бер Бибо., Джордон Сонневельд. «Ajax in Practice» Посібник: 2019., 25–55.

36. Посібник. «Recruiting & Staffing Software and Tracking System» / [Електронний ресурс]. Режим доступу: http://www.pcrecruiter.net

37. Ліза Гарднер, Джейсон Григсби, Розробка веб-сайтов. Вільямс: 2014., 136-140с.

38. Дэвид Сойер Макфарланд. Розробка сайтов на JavaScript и jQuery. 2012. 116-127с.

39. Эрик Фримен, Элизабет Робсон, «Head First JavaScript Programming» / 2016. – Розд. 1. – С. 86–115.

40. Бретт Маклафлин. Объектно - оріентований аналіз. Посібник : 2011., 32–46.

41. Бер Бибо., Иегуда Кац. «jQuery» / 2011, 217-218с.

42. Дэвид Макфарланд. Велика книга CSS3 - CSS3. 2016, 46-80с.

43. Кейт Джонс. «DOM Scripting: Web Design with JavaScript and the Document Object Model» 2005, 368-386 с.

44. Посібник. «What is REST?.» / [Електронний ресурс]. Режим доступу: http://www.restapitutorial.com/lessons/whatisrest.html

45. Посібник. «Архітектура клієнт-сервер» / [Електронний ресурс]. Режим доступу: http://inter.ptngu.com/

46. Посібник. «CRM Recruiting Software Bullhorn» / [Електронний ресурс]. Режим доступу: http://www.bullhorn.com/home-crm-june-alt/?utm_expid=93119204-7.xALLIBT1T52GzQcjL6HjBA.1&utm_referrer=https%3A%2F%2Fwww.google.com.ua%2F

47. Посібник. «Anthony Gore» / [Електронний ресурс]. Режим доступу: https://www.greycampus.com/blog/programming/full-stack-developer-guide

48. Посібник. «Applicant Tracking system» / [Електронний ресурс]. Режим доступу: http://www.icims

49. Rita Mulcahy. PMP Exam Prep: Rita's Course in a Book for Passing the PMP Exam. 2008. – 469

50. Reg Austin., Unmanned aircraft systems : UAVS design, development and deployment. 2008. – 469.