МІНІСТЕРСТВО ОСВІТИ ТА НАУКИ УКРАЇНИ
СУМСЬКИЙ ДЕРЖАВНИЙ УНІВЕРСИТЕТ
ФАКУЛЬТЕТ ЕЛЕКТРОНІКИ ТА ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ
КАФЕДРА КОМП'ЮТЕРНИХ НАУК
СЕКЦІЯ ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ ПРОЕКТУВАННЯ

КВАЛІФІКАЦІЙНА РОБОТА МАГІСТРА

на тему: «Інформаційна система аналізу та розподілу задач для компанії "Apptimized Operations"»

за освітньою програмою 8.122.00.02 «Інформаційні технології проектування»

Виконавець роботи: студент групи ІТ.м.п-71 Будник Олександр Сергійович

Кваліфікаційну роботу захищено на засіданні ЕК з оцінкою _____ «___» грудня 2018 р.

Науковий керівник _____ к.т.н., доц., Гайдабрус Б.В.
(підпис)

Голова комісії _____ к.т.н. Дорошенко С. О.
(підпис)

Засвідчую, що у цій дипломній роботі немає запозичень з праць інших авторів без відповідних посилань.

Студент _____
(підпис)

Суми-2018

Сумський державний університет
Факультет електроніки та інформаційних технологій
Кафедра комп'ютерних наук
Секція інформаційних технологій проектування
Спеціальністьго 122 «Комп'ютерні науки»
Освітньо-професійна програма «Інформаційні технології проектування»

ЗАТВЕРДЖУЮ
Зав. секцією ІТП

В. В. Шендрик
«_____» _____ 2018 р.

ЗАВДАННЯ

на кваліфікаційну роботу магістра студентові

Будник Олександр Сергійович
(прізвище, ім'я, по батькові)

1 Тема проекту Інформаційна система аналізу та розподілу робочих задач для компанії "Apptimized Operations"

затверджена наказом по університету від «20» листопада 2018 р. №2458-III

2 Термін здачі студентом закінченого проекту «_12__» ___грудня___ 2018 р.

3 Вхідні дані до проекту Для розробки продукту даного проекту – інформаційної системи аналізу та розподілу задач – необхідною є інформація про веб-системи, що використовуються у компанії «Apptimized Operations» для виконання задач із пакування ПЗ (зовнішня веб-система «JIRA»), зберігання даних про витрачений на процес пакування час (внутрішня веб-система «TimeTracker») та зберігання даних про спеціалістів компанії (внутрішня веб-система «Apptimized Intranet»). Для розроблюваної інформаційної системи ці веб-системи слугують джерелами даних про пакети ПЗ (назва пакету, рівень складності, пріоритетність), залогований пакувальниками ПЗ час на створення пакетів для підрахунку навантаження та аналізу нових замовлень на пакування.

4 Зміст розрахунково-пояснювальної записки (перелік питань, що їх належить розробити) Завдання, Реферат, Вступ, Аналіз предметної області, Постановка задачі та методи дослідження, Моделювання інформаційної системи, Реалізація інформаційної системи, Висновки.

5 Перелік графічного матеріалу (з точним зазначенням обов'язкових креслень) Презентація (18 слайдів)

6. Консультанти випускної роботи із зазначенням розділів, що їх стосуються:

Розділ	Консультант	Підпис, дата	
		Завдання видав	Завдання прийняв

Дата видачі завдання _____.

Керівник _____
(підпис)

Завдання прийняв до виконання _____
(підпис)

КАЛЕНДАРНИЙ ПЛАН

№ п/п	Назва етапів випускної проекту	Термін виконання етапів проекту	Примітка
1	Аналіз структури та бізнес-процесів компанії «Apptimized Operations»	До 19.03.18	
2	Порівняльний аналіз аналогів ІС	До 26.03.18	
3	Аналіз вимог	До 01.04.18	
4	Планування проекту	До 14.05.18	
5	Моделювання інформаційної системи	До 09.08.18	
6	Проектування веб-інтерфейсу продукту	До 18.08.18	
7	Розробка ІС	До 20.10.18	
8	Тестування продукту	До 02.12.18	
9	Оформлення документації по проекту	До 12.12.18	

Магістрант _____ Будник О.С.

Керівник роботи _____ к.т.н., доц. Гайдабрус Б.В.

РЕФЕРАТ

Тема кваліфікаційної роботи магістра «Інформаційна система аналізу та розподілу задач для компанії “Apptimized Operations”».

Пояснювальна записка складається зі вступу, 4 розділів, висновків, списку використаних джерел із 26 найменувань, додатків. Загальний обсяг роботи − 108 сторінок, у тому числі 62 сторінок основного тексту, 2 сторінки списку використаних джерел, 37 сторінка додатків.

Кваліфікаційну роботу магістра присвячено розробці інформаційної системи для аналізу новостворених задач та підбору виконавців для кожної задачі для керівництва компанії «Apptimized Operations».

В роботі проведено порівняльний аналіз поширених різновидів машинного навчання, алгоритмів порівняння послідовностей, а також дослідження щодо доцільності розробки інформаційної системи для аналізу робочих задач, і огляд технологій для реалізації інформаційної системи. У роботі виконано аналіз структури компанії «Apptimized Operations», моделювання процесу аналізу та розподілу задач у межах компанії за нотацією IDEF0, створення моделі прецедентів системи і моделі основних модулів (аналізу та розподілу), розробка інформаційної системи, а також реалізація програмного прошарку періодичної синхронізації даних різних систем. Результатом проведеної роботи є реалізована інформаційна система. Практичне значення роботи полягає у реалізації функціоналу перегляду проаналізованої задачі, її розрахованого часу на виконанння, а також перегляд повної інформації про навантаження підібраних виконавців для задачі, а також реалізоване виведення графіку відношення навантаження працівника до середнього значення навантаження команди за певний період часу.

Ключові слова: інформаційна система, аналіз, розподіл, пакування ПЗ, пакет ПЗ, пакувальник ПЗ, веб-система, навантаження.

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СИМВОЛІВ, ОДИНИЦЬ, СКОРОЧЕНЬ І ТЕРМІНІВ

ІС	сукупність організаційних і технічних засобів, як організаційних так і технічних, які дозволяють зберігати та обробляти інформацію з метою забезпечення потреб користувачів.
БД	База даних.
СУБД	Система управління (керування) базою даних.
ПЗ	Програмне забезпечення.
Пакування ПЗ	Процес перетворення оригінальних інсталяційних файлів ПЗ в один із пакетних форматів (MSI, App-V, MSIX, AppX) для розповсюдження на необмежену кількість комп’ютерів у тихому режимі.
Пакет ПЗ	Результат процесу пакування деякого ПЗ у межах компанії «Apptimized Operations».
Пакувальник ПЗ	Спеціаліст із пакування програмного забезпечення, співробітник компанії «Apptimized Operations».
IDEF0	Фундаментальна нотація моделювання бізнес-процесів.
MVC	Паттерн проектування програмного забезпечення «Model-View-Controller» (з англ. «Модель-Представлення-Контролер»).

ВСТУП

На сучасному етапі розвитку інформаційних технологій все частіше підприємства починають використовувати системи автоматизації для пришвидшення роботи підприємства, а також мінімізації виникнення помилок через людський фактор.

Сучасні підприємства часто у своїй діяльності використовують інформаційні технології, які можуть вирішувати різний спектр задач: від управління самим підприємством до допомоги в прийнятті управлінських або інших рішень.

«Apptimized Operations» є однією з IT-компаній Сумщини. Дана компанія спеціалізується на пакуванні програмного забезпечення (ПЗ). Пакування програмного забезпечення (Software packaging) – це процес перетворення інсталяційних файлів деякого ПЗ в один із форматів інсталяційних пакетів: MSI, App-V, AppX або MSIX, із налаштуванням програми.

Даний процес слугує для подальшого легкого встановлення «запакованого» інсталяційного пакету ПЗ на безмежну кількість корпоративних комп’ютерів через системи розповсюдження ПЗ, такі як System Center Configuration Manager (SCCM). Перелічені вище формати інсталяційних пакетів легко налаштовуються для розповсюдження через подібні системи, тоді як розповсюдження оригінальних інсталяційних файлів ПЗ можуть бути складним завданням для систем розповсюдження. Також проходження налаштувань інтерфейсу інсталяції таких оригінальних файлів додатку для кожного комп’ютера буде займати багато часу, тоді як інсталяційний пакет слугую для «тихого» встановлення без будь-якого інтерфейсу (повне налаштування програми у даному випадку здійснюють спеціалісти «Apptimized Operations»).

На сьогоднішній день процес отримання нових замовлень, виконання завдань з пакування ПЗ та здачі результатів роботи замовникам проводиться через веб-систему виконання та відстеження задач JIRA. Це зручна система для виконання поставлених завдань, відстеження їх статусів та комунікації із замовниками.

Кожен пакет, який надходить має свій певний життєвий цикл – починаючи від надходження запити від замовника, закінчуючи відправленням готового продукту.

Першим етапом цього життєвого циклу є переведення завдання на робітника компанії. Він проводить збір інформації про пакет, його складність, та формує комплексну матрицю. В ній визначається складність пакета.

Проблема полягає у етапах життєвого циклу пакета. На даний момент складність визначається після того, як робітник його почав виконувати. Такий підхід має декілька недоліків, таких як – не можливість розрахувати завантаженість команди, як тільки запит був створений.

Для вирішення проблеми та забезпечення більш комфортного процесу може бути аналіз складності задачі та автоматизація її життєвого циклу .

Об’єкт дослідження: діяльність менеджменту компанії та процесу розподілу задач.

Предмет дослідження: виконані та завершені завдання для проведення подальшого аналізу; повний життєвий цикл робочої задачі.

Метою даного проекту є розробка інформаційної системи аналізу та розподілу робочих задач для компанії «Apptimized Operations».

Для досягнення мети роботи необхідно виконати наступні задачі:

- Аналіз предметної області:
  - аналіз структури компанії;
  - аналіз бізнес-процесів компанії;
  - аналіз процесу розподілу задач:
- Збір необхідної інформації із сторонніх систем;
- Аналіз існуючих підходів розробки;
- Моделювання інформаційної системи розподілу робочих задач;
- Створення прототипу ІС;
- Тестування системи;
- Введення в експлуатацію;

Задля досягнення мети були задіяні такі методи дослідження як: аналіз, аналогія, моделювання та експеримент.

Практичне значення даної роботи полягає у реалізації інформаційної системи аналізу та розподілу робочих задач, яка дозволить переглядати навантаження робітників та розподіляти задачі між командою зважаючи на поточне навантаження кожного робітника. Також до результатів даної роботи входить наступне:

- вперше розроблена інформаційна модель бізнес-процесів розподілу робочих задач IT-компанії «Apptimized Operations»;
- вперше розроблене інформаційна система для аналізу навантаження робітників в компанії «Apptimized Operations»;

1 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ

1.1 Аналіз структури та бізнес-процесів компанії «Apptimized Operations»

Компанія «Apptimized» є одною з найбільших компаній в світі є з постачання послуг упаковки додатків.

Компанія надає послуги з пакування ПЗ спеціалістами Apptimized, надає можливість самостійного пакування ПЗ клієнтом із використанням програмних інструментів компанії Apptimized у власній системі, та вибір готового пакету програмного забезпечення; окремим напрямком діяльності є послуги моніторингу оновлень програмних продуктів.

Для організації діяльності підрозділу існує команда менеджерів, до обов’язків якої входить:

- розподіл задач, які з певною періодичністю надходять від замовників;
- організація роботи команди робітників;
- аналіз навантаження команди;
- організаційні обов’язки;

До складу менеджерів входять Office Director та Head of Operations.

Ефективне управління - цінний ресурс організації, такий як фінансовий, матеріальний, людський тощо. Автоматизація – найбільш дієвий спосіб підвищення ефективності протікання процесів.

Розвиток інформаційних технологій, покращення технічної платформи і створення нових класів програмних продуктів призвів до зміни підходів в автоматизації управління підприємством та виробництвом.

Динамічний розвиток світового ринку ІТ здійснює значний вплив на розвиток світового підприємництва, розробка та впровадження нових технологій оптимізує процеси керівництва та виробництва, дозволяє більш ефективно використовувати ресурси, сприяє прискоренню обміну інформацією.

Діяльність роботи компанії, в першу чергу, пов’язана з пакуванням програмного забезпечення у відповідні програмні пакунки. Як тільки компанія отримує нове замовлення на пакування ПЗ – відповідальна особа, в даному випадку Head of Operations, мусить проаналізувати задачу, визначити для неї її можливий час на виконання, оцінити навантаження команди пакувальників, визначити людей, які можуть виконати дану задачу для даного проекту тощо.

Тож, процес розподілу задач між командою є досить специфічним завданням, оскільки на першому етапі – як тільки задача прийшла від замовника дуже важко оцінити її в точки зору часу, який буде необхідний на виконання. Відповідальній особі необхідно кожен раз враховувати безліч параметрів та нюансів для кожного нового замовлення та для кожного пакувальника ПЗ. Даний процес займає досить багато часу, оскільки все дані, які слід проаналізувати знаходяться в різних веб-системах, їх потрібно об’єднати, скомпонувати для отримання точного представлення навантаження команди тощо.

Таким чином, даний підхід спричиняє такі наслідки:

- Можливе порушення дедлайну;
- Нерівномірний розподіл навантаження між командою;

При такому підході витрачається велика кількість часу хоча б на поверхневий аналіз задачі, щоб мінімізувати наслідки.

«Apptimized Operations» використовує продукт компанії «Atlassian» – Jira. Даний продукт дозволяє робітникам та керівництву відслідковувати виконані та поточні задачі, час на виконання, аналізувати ресурси тощо. Кожне завдання в Jira має інформацію, яка є необхідною для виконання завдання пакувальником.

Дана система дозволяє покрити лише частину проблеми – моніторинг задач, та їх життєвого циклу. Але не дозволяє оцінити можливий рівень зусиль (level of effort) на самому початку життєвого циклу задачі, що має негативний вплив на планування навантаження команди, розподіл задач тощо.

Керівництво компанії хоче мати можливість отримувати аналіз кожного замовлення, яке надходить від замовників, для прийняття рішень керування компанією, а саме:

- Контроль за навантаженням команд пакувальників ПЗ для окремих проектів;
- Прогнозування можливої завантаженості команди та окремо працівників;

Підбивши підсумок вищесказаного, можна сказати, що компанія потребує інструментарій, який дозволив би:

- мати миттєвий аналіз нової задачі на складність;
- отримувати список можливих виконавців нових задач;

Так, актуальність розробки та впровадження даної інформаційної системи зумовлена неможливістю оцінити та розрахувати можливий рівень зусиль (LoE), який буде необхідний на створення пакунку.

Метою даного проекту є розроблення більш вузько-спрямованого продукту для конкретних цілей – аналізу задачі, визначення приблизного рівня зусиль, необхідного на вирішення задачі, та визначення можливих виконавців даної задачі базуючись на поточному навантаження команди та робітників.

Використання такого інформаційної системи забезпечить більш швидкий розподіл задач, який буде базуватися на критеріях, які були визначені як основні:

- Визначена можливих часових витрат на задачу;
- Навантаження робітника;

1.1.1 Аналіз замовлення від замовника

Процес пакування у компанії «Apptimized Operations» починається із створення клієнтом замовлення на пакування ПЗ. Це замовлення, створене у хмарній веб-системі «Apptimized Dashboard», автоматично створюється у веб-системі виконання та моніторингу завдань «JIRA» у вигляді задачі.

Задача із пакування ПЗ призначається одному із пакувальників людиною, відповідальною за розподіл завдань.

Дана задача має в собі всю необхідну інформацію для пакувальника. У ході роботи над пакетом всі дані заносяться до задачі у системі «JIRA» у формалізованому виді.

Кожна задача має наступні ключові поля, які дозволяють визначати прогрес по виконанню задачі, складність, терміновість тощо. Так, поле SLA – визначає кількість часу, який вже був витрачений на виконання пакування, загальний час на виконання завдання, та дату дедлайну. Статус – поле, яке визначає поточний статус замовлення: в роботі, чекає на тест тощо. Пріоритет – поле, яке визначає терміновість задачі: нормальний, терміновий. Технологія пакування – дане поле відповідає за технологію пакування ПЗ: MSI, App-V тощо.

Також, замовлення має поля, які відображають людину, яка на даний момент займається замовленням, поле, яке відображає пакувальника, а також поле, яке відображає QA інженера.

Кожна задача має власний ідентифікатор в системі «Jira», який дозволяє однозначно ідентифікувати ту чи іншу задачу. Даний ідентифікатор використовується як системою «TimeTracker» так і буде використовуватися інформаційною системою розподілу задач для синхронізацію даних між всіма взаємодіючими системами. Тож, даний ідентифікатор використовується для того щоб час, який спеціалісти витрачають на пакет – пакування та тестування, заносилися до внутрішньої веб-системи «TimeTracker» у формалізованому вигляді.

Кожен тікет із системи «Jira» і «TimeTracker» може бути описаний в інформаційній системі використовуючи числові представлення основних критеріїв, які описують ту чи іншу задачу. Зіставивши всі дані із різних систем, ми зможемомо однозначно описати поточні задачі пакувальників, нові задачі, а також вже відправлені замовникам.

Використовуючи даний опис, інформаційна система буде аналізувати поточне навантаження робітників та визначати пожливий час на виконання нової задачі.

Приклад задачі представлений на рисунку 1.1.

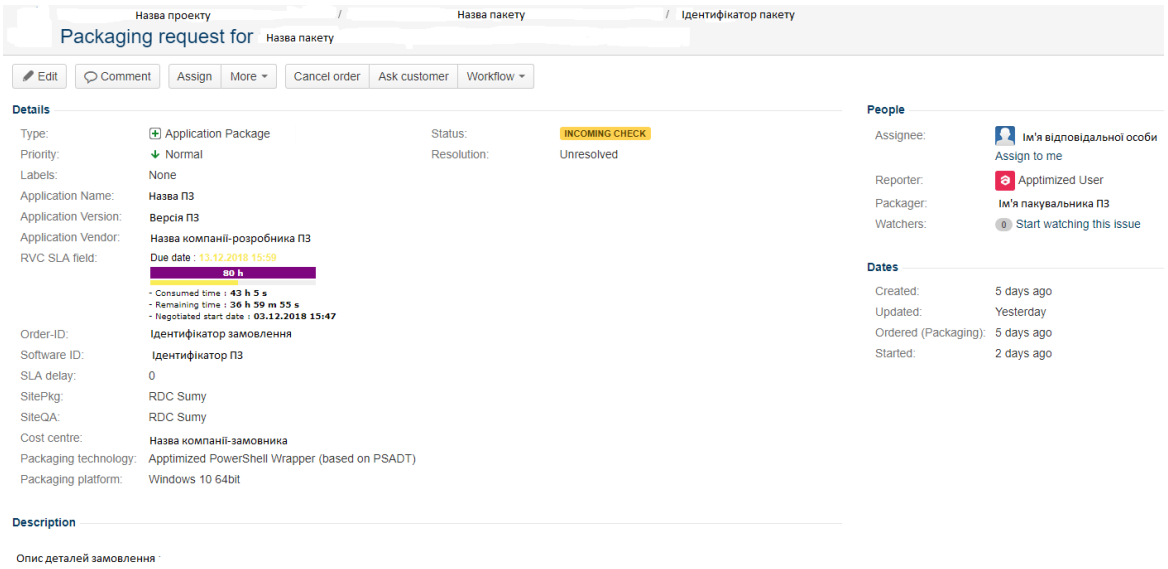

Рисунок 1.1 – Приклад сторінки замовлення на пакування ПЗ

1.2 Порівняльний аналіз поширених систем розподілу задач

У даному підрозділі будуть розглядатися особливості, переваги та недоліки таких різновидів та підходів до аналізу та розподіленн робочих задач.

Передусім слід зазначити основні вимоги замовника проекту (керівництва компанії «Apptimized Operations») до розроблюваної інформаційної системи аналізу та розподілу задач.

На основі доступних даних уже з працюючих веб-систем компанії, інформаційна система повинна аналізувати нові задачі та отримувати оцінку задачі у відповідності з вибраними параметрами – серед яких:

- Оцінка складності запиту на пакування відносно проектної складності (ProjectComplexityRatio);
- Оцінка можливої складності запиту на пакування (ComplexityRatio);
- Оцінка терміновості запиту (PriorityRatio);
- Оцінка часу (SLARatio).

На основі даних показників повинна аналізуватися нова задача, та визначатися її можлива складність, що в с воб чергу дозволить покращити планування.

Також, основною вимогою до системи є – можливість назначення робочої задачі на робітника, який найкраще підходить для її виконання.

Розглянувши дані мивоги, слід переглянути основні програмні продукти, які теоритично мужуть виконувати схожі функції.

Microsoft Excel

Аналогом розроблюваної системи розрахунків та визначення результатів може слугувати програмне забезпечення Excel, яке дозволяю за допомогую вбудованих функцій, а також програмних розширень функціоналу виконувати поставлені задачі перед інформаційною системою. Але даний продукт все ж таки більше підходить для зберігання інформації в табличному виді та її обробки, але формулізувати задачі на пакування даний продукт не зможе, оскільки він не має такого функціоналу – доступ та взаємодія з API інших веб-систем компанії.

Трекінгова система Jira

Atlassian JIRA — система відстеження задач, помилок, розроблена для організації спілкування між командою а також з користувачами, а також для управління проектами. Розроблена компанією Atlassian в 2002 році. Наразі існує дві версії: хмарна і сервера.

Головні елементи JIRA - проблема і робочий процес (англ. workflow).

Проблема описує задачу яка має бути виконана, і мусить мати назву і детальний опис, також проблема може мати пріоритет, автор проблеми і хто має над нею працювати. В залежності від її виду (задача, баг, інцидент) атрибути даної проблеми варіюються від її виду.

Важливим атрибутом кожної задачі є її статус (англ. status) який показує поточний етап роботи над задачею. Статус міняється згідно робочого процесу створеного для цієї проблеми при роботі над нею.

Робочий процес – це набір дій, які виконуються над проблемою для її вирішення.

Найпростішим робочим процесом є «заплановано» — «в роботі» — «зроблено», проте для кожного виду проблеми є свій робочий процес. Будь-які зміни в проблемі записуються в журнал активності.

В JIRA як проблема, так і робочий процес можуть бути сконфігуровані адміністратором, він може створити користивуцькі елементи, або модифікувати стандартні елементи.

Jira наразі використовується в компанії для відслідковування задач – її прямих функцій. Так, за допомогою вбудованих плагінів, користувач може оцінити кількість завдань, які наразі має пакувальник, але весь процес збору інформації займає доволі багато часу – всі ті проблеми, які були описані в розділі 1.1.

Наразі даний підхід використовується в компанії – аналіз навантаження по інформації з цієї веб-системи. Але, як бачимо, компанія зацікавлена в більш вузько-направленій системі, для вирішення конкретних задач, які не дозволяє вирішити дана веб-система.

Trello

Trello — безкоштовна веб-система управління проектами, яка використовує канбан як парадигму керування проектами. Проекти зображуються дошками, які містять списки, які в свою чергу містять картки, якими, в системі, зображуються задачі.

Картки переходять з одного списку до наступного (за допомогою перетягування) – цей рух відображає проходження задачі по процесу картки. Картці може бути присвоєно відповідальних за неї користувачів. Веб-ситема має організації – об’єднання користувачів та дошок.

Дана система, як і Jira дає можливість відслідковування задач в команді, відслідковувати її статуси та часові показники, але не дозволяє визначити поточну навантаженість команди та окремого працівника на основі визначених критеріїв.

Wrike

Можливості Wrike включають планування і управління бізнес-процесами, обмін інформацією між співробітниками, аналіз даних і формування звітів.

Робочі процеси в Wrike представлені у вигляді категорій, проектів, завдань і підзадач. До особливостей сервісу також можна віднести прив'язку обговорень до завдань, щоб забезпечити швидке занурення в контекст роботи і доступ до останніх файлів і дискусій по завданню.

Функції створення звітів дозволяють економити час на виконанні таких рутинних завдань, як збір даних по поточному прогресу проекту [19].

Основні функції:

- Регулювання прав доступу і редагування на рівні проектів, окремих завдань і користувачів.
- Інструменти для спільної роботи.
- Панелі задач з актуальних статусом важливих доручень.
- Настроюються візуальні звіти за ключовими показниками, статусам проектів і продуктивності команди.
- Настроювані поля і статуси завдань, що дозволяють адаптувати сервіс під унікальні робочі процеси.
- Інтерактивні діаграми Ганта для планування проектів та розподілу завантаження.
- Редагування проектів у вигляді таблиць.
- Календарі для планування графіка співробітників, обліку відпусток, лікарняних і святкових днів.
- Відстеження часу, витраченого співробітником на кожну задачу.
- Додатки для Outlook і Apple Mail, що дозволяють створювати і редагувати завдання з поштового клієнта.
- Миттєві повідомлень в браузері і email про важливі оновлення проектів і завдань.
- Підтримка технології єдиного входу, що дозволяє використовувати єдиний логін і пароль для корпоративних систем.

ProofHub. Програмне забезпечення для управління проектами, яке вирішує всі проблеми, що постають перед зростаючою командою.

Робочі процеси та канаби канбанів у ProofHub спрощують процеси керування завданнями, дозволяючи визначити індивідуальний робочий процес для команди, який ви можете покращити, щоб забезпечити більшу гнучкість у вашій роботі.

ProofHub дозволяє логувати час, який робітники витрачають на виконання завдання. Даний модуль дозволить компанії вдмовитись від внутрішньої додаткової системи, до якої необхідно вносити витрачений час на виконання задачі. Це полегшить роботу керівництва та робітників, оскільки необхідно використовувати одну систему.

Програмне забезпечення звітування проекту ProofHub надає чітке уявлення про те, як використовуються ресурси, про те, як реалізуються проекти, щоб ви могли позбутися неефективності проектів. Даний елемент дає можливість за допомогою плагінів виводити дані по проектам та робітникам, але не дозволяє виводити список працівників, які наразі менш навантажені ніж інші. Даний модуль є одним із функціональних вимог до системи – тому даний програмний продукт не задовільняє всі вимоги до функціоналу.

Проаналізувавши аналогічні ПП, слід зазначити, що жоден ПП не може в повній мірі покрити вимоги до інформаційної системи, особливо слід взяти до уваги, специфіку роботи компанії, а як наслідок – і її робочі процеси.

Тож, актуальність розробки системи зумовлена неможливістю задоволення вимог замовника уже відомими програмними продуктами – аналізувати робочі задачі.

А також, підвищити швидкість процесів, та зменшити кількість ризиків пов’язаних з плануванням навантаження як на окремого робітника так і на всю команду.

Так, метою даного дослідження є інформаційної системи аналізу та розподілу робочих задач, яка буде мати інтеграції із системами «JIRA» і «TimeTracker» за допомогою програмної частини даного проекту, яка буде також слугувати для

керування системою та перегляду результатів виконання аналізу та розподілу задач відповідно.

2 ПОСТАНОВКА ЗАДАЧІ ТА МЕТОДИ ДОСЛІДЖЕННЯ

2.1 Мета та задачі дослідження

Метою даного проекту є розробка інформаційної системи для збору інформації про виконані задачі, аналізу їх складності, формування складності нових задач, визначення можливих виконавців.

Для досягнення мети в роботі розв’язувались такі задачі:

- збір інформації та аналіз предметної області (структури та діяльності компанії) – проаналізувати бізнес-процеси та робочі процеси компанії. Визначити основні етапи, які можуть бути автоматизовані за допомогою інформаційної системи;
- дослідити підходи, методів та технологій розробки інформаційної системи, а також стеку технологій для створення веб-інтерфейсу;
- змоделювати бізнес-процесів компанії «Apptimized Operations», процес розподілу робочих задач, а також розробка моделей процесів, що будуть проходити під час використання веб-системи;
- розробити функціонал для збору та аналізу даних стосовно нових надісланих задач на пакування ПЗ;
- розробити функціонал для збору даних стосовно поточних задач робітників, їх відпусток тощо, що дозволить розподіляти задачі серед команди пакувальників на основі сформуваного числового опису задачі та навантаження пакувальника;

Інформаційна система буде використовуватись для повехневого аналізу нових задач на пакування ПЗ, визначення менш навантажених робітників, які можуть виконати нове завдання на пакування. Дана інформаційна система дозволить легко визначати, які параметри в даний момент часу є важливішими для визначення робітників, в свою чергу від цього буде залежати результат виконання.

Інтерфейс програми дозолить людині, яка займається роподілом задач отримати всю необхідну інформацію стосовно робітника та нового замовлення, отримати дані по навантаженню робітника відносно команди пакувальників, а також перевести замовлення на робітника не використовуючи систему «Jira».

2.2 Методи дослідження

На даний момент існує багато різних методів дослідження. Але всі вони мають різну популярність. Базуючись на спостереженні буда створена проста схема проходження задач, яка представлена на рис.2.1.

Рисунок 2.1 – Базовий життєвий цикл задачі

Розглядаючи даний процес, виявлено, що він не є ідеальним для сфери діяльності компанії, оскільки два процеси: визначення складності та розрахунку дедлайну створюють низку ризиків та можуть викликати проблеми в роботі.

Експериментально визначено, що переміщення підпроцесів вище по життєву циклу спрощує робочий процес для робітників, а також дає можливість скорити час на виконання завдання, оскільки час який робітник витрачає на аналіз задачі та визначення складності може бути автоматизований.

На основі вимог замовника ІС були проаналізовані доступні інструментарії та підходи до реалізації.

2.2.1 Вибір підходу реалізації аналізу задач

Задача аналізу системи полягає у визначення можливої складності нового запиту на пакування ПЗ. Щоб отримати числове представлення даної складності визначаємо основні ключові показники, які впливають на складність пакетного замовлення:

- складність проекту;
- час на виконання;
- терміновість;
- складності все виконаних схожих замовлень;
- середня складність всіх пакетів для даного проекту;
- технологія виконання замовлення;

Всі ці показники впливають на результуючу складність пакета (рис.2.2).

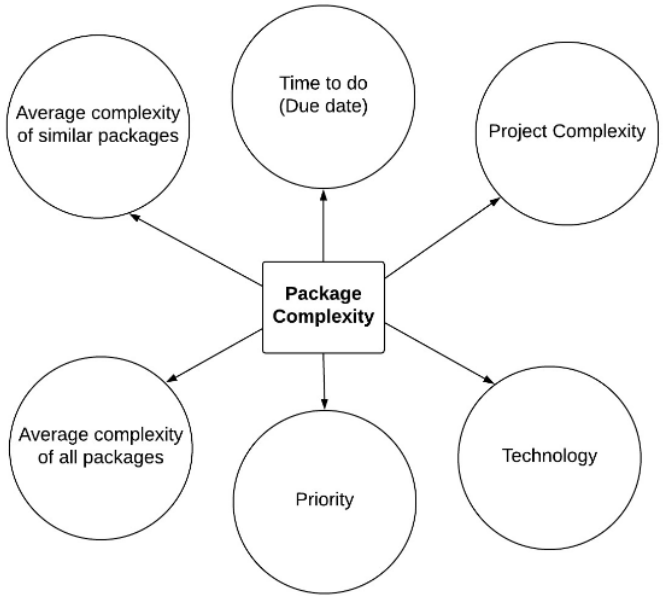

Рисунок 2.2 – Схема показників складності пакета

Для вирішення даної задачі найбільш підходящим методом є регресійний аналіз.

Регресійний аналіз — розділ статистики, який мітить в собі методи аналізу залежності однієї величини змінної величини від статичної.

На відміну від кореляційного аналізу, в регресії не з'ясовує істотність зв'язоку, а шукається моделі цього зв'язку, вираженої у функції регресії.

Регресійний аналіз використовується коли відношення між змінними можуть бути виражені кількісно у виді комбінації змінних. Отримана комбінація використовується для передбачення значення, що може приймати цільова (залежна) змінна, яка обчислюється на заданому наборі значень вхідних (незалежних) змінних.

У найпростішому випадку для вирішення такої задачі використовуються стандартна лінійна регресія.

Але, більшість реальних моделей не можуть вирішитися за допомогою лінійної регресії.

Наприклад, фондові ціни, розміри продаж тощо, дуже складні для передбачення, оскільки залежать від комплексу зв'язків множин змінних. Таким чином, необхідні комплексні методи для передбачення майбутніх значень.

Рівняння регресії – це функція, яка описує залежність середнього значення

ознаки від заданих значень аргументів.

Термін "регресія" (лат. - "regression" - відступ, повернення до чого-небудь) введений англійським психологом і антропологом Ф. Гальтпном і пов'язаний тільки зі специфікою одного з перших конкретних прикладів, у якому це поняття було використано.

Тож, в цьому випадку регресійний аналіз буде використовуватися для визначення можливої складності пакета. Отриманий реультат буде використовуватися при розподілі задачі серед команди пакувальників.

Мета лінійної регресійної задачі полягає в тому, щоб передбачити значення числової змінної, що базується на значеннях одного чи кількох чисельних змінних предиктора. Наприклад, ви можете передбачити щорічний дохід людини на підставі його рівня освіти, років досвіду роботи та статі (чоловік = 0, жінка = 1).

Змінна для передбачення зазвичай називається залежною змінною. Прогнозовані змінні зазвичай називають незалежними змінними. Коли існує лише одна змінна предиктора, техніку іноді називають простим лінійним регресом. Коли існують два або більше змінних для прогнозування, ця методика зазвичай називається множинною або багатоваріантною лінійною регресією. [18]

2.2.2 Вибір методології розподілу задач

Зважаючи на швидкий та динамічний розвиток штучного інтелекту, як інструменту вирішення задач та аналізу даних – саме машинне навчання повинно дозволити вирішити поставлені задачі.

Провівши поверхневий аналіз доступних підходів реалізації штучного інтелекту були вибрані реалізації, які задовольняють поставлені потреби:

- Розподіл задач на групи;
- визначення необхідної групи об’єктів.

Серед всіх видів машинного навчання були виділені основні для даної задачі:

- кластерний аналіз;
- класифікація;
- нечітка логіка.

Кластерний аналіз. При використанні кластерного аналізу важливо визначити, скільки в результаті цієї процедури повинно бути побудовано кластерів.

Передбачається, що кластерний аналіз повинен виявити локальні згущення об’єктів. Тому, кількість кластерів є параметром, який може ускладнити алгоритм, якщо визначено, що даний параметр невідомий і може суттєво вплинути на якість результату.

Алгоритми кластеризації зазвичай будуються як спосіб перебору кількості кластерів і визначення його оптимального значення в процесі перебору та включають 5 основних кроків:

1. Відбір вибірки для кластерного аналізу.

2. Визначення ознак, для оцінки об’єктів вибірки.

3. Обчислення значень тієї або іншої міри подібності між об’єктами.

4. Застосування кластерного аналізу для створення груп з подібних об’єктів.

5. Перевірка результатів кластерного аналізу.

На сьогоднішній день існує велика кількість методів для розбиття вхідної групи елементів на групу об’єктів – кластери.

Методи кластерного аналізу дозволяють розв’язувати такі задачі:

– проведення класифікаційного аналізу об’єктів разом з урахуванням ознак, що відображають сутність та природу об’єктів. Розв’язок таких задач веде до поглиблення знань про сукупность об’єктів, які піддаються класифікації;

– перевірка припущень про наявність певної структурної організації в досліджуваній сукупності об’єктів – відбувається пошук структури та залежностей;

– побудова нових класифікацій для слабовивчених явищ, коли необхідно визначити наявність зв’язків усередині сукупності і спробувати визначити її структуру.

Для вирішення поставленої задачі був вибраний неієрархічний метод кластерного аналізу, оскільки мають за основу предвизначену кількість кластерів (k-means, PAM кластеризація) або використовують інші більш складні алгоритми

знаходження кількості кластерів (CLOPE, карти Кохонена).

Ієрархічна кластеризація виконується при послідовному об’єднанні малих кластерів до більших або навпаки – розбиття від більших кластерів до менших (дивизивна). На відміну від неієрархічних, такі алгоритми кластерного аналізу будують розбиття для всіх варіантів.

Ітеративні методи заслуговують найбільшої уваги еред неієрархічних методів кластеризації оскільки такі медоти працюють за наступним алгоритмом:

1. вихідні дані розбиваються на певну кількість кластерів та обчислюються центри тяжіння для кожного з цих кластерів;
2. кожен об’єкт з початкових даних поміщується в кластер з найближчим центром тяжіння;
3. обчислюються нові центри тяжіння кластерів; вони не замінюються на нові, поки не будуть повністю переглянуті всі вхідні дані;
4. кроки 2 і 3 повторюються, поки не перестануть змінюватись кластери.

Загальну схему та процес роботи ітеративних методів представлено на рисунку 2.3.

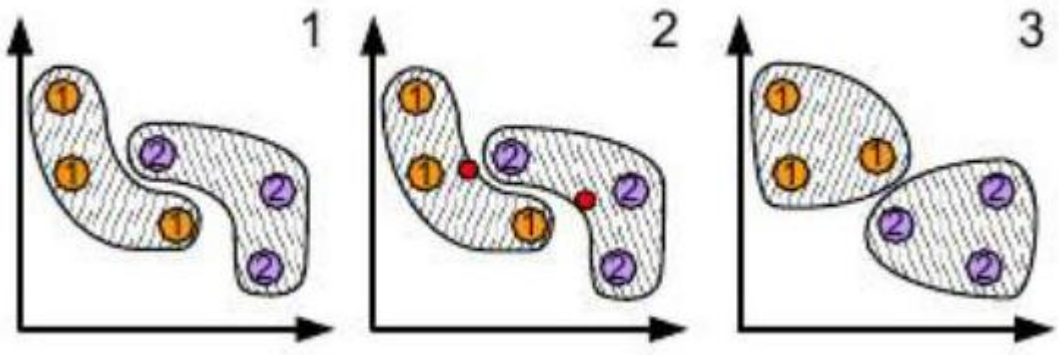

Рисунок 2.3 - Загальна схема роботи ітеративних методів

На відміну від ієрархічних методів кластерного аналізу, які потребують обчислення і збереження матриці схожості між об’єктами, ітеративні методи працюють безпосередньо з початковими, вхідними даними.

Тому, за допомогою датих методів можна обробляти великі обсяги даних. Більш того, ітеративні методи виконують декілька переглядів, ітераій проходження

початкових даних і, за рахунок цього, нівелюють наслідки помилкової вихідної розбивки даних.

Ці методи породжують кластери одного рангу, які не є вкладеними, і тому не можуть бути частиною ієрархії. Зазвичай, ітеративні методи не допускають перекриття кластерів. Властивості таких методів групування можуть бути описані за допомогою таких трьох основних чинників: вибір вихідної розбивки, тип ітерації і статистичного критерію.

Ці параметри можуть поєднуватись, утворюючи алгоритми відбору даних при визначенні оптимальної розбивки. Різні комбінації цих параметрів ведуть до розробки методів, породжують різні результати при роботі з одними й тими ж даними [13].

Оскільки кількість кластерів вже визначена то, неієрархічні методи виявляють високу стійкість відносно шумів і викидів, некоректного вибору метрики, введення незначущих змінних у набір, що бере участь в кластерному аналізі.

Метод k-найближчих сусідів. Непараметричний класифікаційний метод, де для класифікації об'єктів у рамках простору властивостей використовуються відстані (зазвичай евклідові), пораховані до усіх інших об'єктів. Вибираються об'єкти, до яких відстань найменша, і вони виділяються в окремий клас.

Алгоритм методу найближчих сусідів ґрунтується на порівнянні відомих статистичних даних з новими елементами. Для нового запису, продовження якого необхідно спрогнозувати, знаходять найбільш подібні записи у минулому та ідентифікують їх як найближчих сусідів. Метод найближчих сусідів спирається на так звану “гіпотезу компактності” [16]. Ця гіпотеза стверджує, що різні відображення одного і того ж образу у просторі ознак породжують геометрично близькі точки, утворюючи компактні “згустки”. Умовою ефективного застосування методу НС є вдалий вибір наступних параметрів: вид метрики, яка служить для оцінки близькості між об’єктами; розмірність навчальних зразків (паттернів), кількість К найближчих сусідів, які беруться до уваги при побудові прогнозу. Вибір малого значення параметра К приведе до сильного розкиду значень прогнозу; велике значення К може спричинити сильну зміщеність моделі.

Отже, значення К повинно бути достатньо великим, щоб мінімізувати ймовірність помилкової класифікації, і досить малим, щоб К близьких сусідів розташовувалися досить близько до точки запиту.

Прийняття рішень у проблемно-орієнтованих інформаційних системах та системах керування здійснюється в умовах невизначеності, яка зумовлена неточністю або неповнотою вхідних даних, природою зовнішніх впливів, відсутністю адекватно сформованої математичної моделі функціонування, нечіткістю мети, людським фактором тощо [17].

Нечітка логіка. Невизначеність системи призводить до зростання ризиків від прийняття неефективних рішень, до того, що результатом можуть бути негативні економічні, технічні та соціальні та інші наслідки. Невизначеність у системах прийняття рішень компенсують за допомогою затосування різних методів машинного навчання та штучного інтелекту.

Для ефективного прийняття рішень в умовах невизначеності – для коректного функціонування системи використовують методи які базуються на основних правилах нечіткої логіки.

Методи нечітких множин корисні коли відсутні точні математичні моделі, які описують функціонування системи.

Теорія нечітких множин дає можливість застосувати для прийняття рішень не формалізовані суб’єктивні експертні знання про предметну область [15].

З використанням методів нечітких множин можна вирішуюти питання узгодження суперечливих критеріїв прийняття рішень, створення логічних регуляторів систем.

Нечіткі множини застосовують лінгвістичний опис складних процесів, за допомогою встановлення нечітких відношеннь між поняттями,також допомогають прогнозувати поведінку системи та формувати множину дій, і, нарешті, формально описати правил прийняття рішень.

Методи нечітких множин є гарним засобом дл проектування інтерфейсів у людиномашинних системах. На основі нечіткого логічного виведення можна побудувати системи керування, підтримки прийняття рішень, подання знань,

структурної та параметричної ідентифікації, розпізнавання образів.

Нечіткі експертні системи знаходять широке застосування у військовій справі, медицині, економіці для підтримки прийняття рішень. З їх допомогою здійснюють бізнес-прогнозування, оцінювання ризиків та прибутковості проектів [14,15].

Важливим застосуванням нечітких множин є контролери нечіткої логіки, які використовують у системах керування. Замість математичної моделі для опису системи – контролери використовують знання експертів, які описуються за допомогою лінгвістичних нечітких множин.

Перспектива застосування нечіткої логіки в даній предметній області полягає у розробленні гібридних методів, до яких можна віднести нечіткі штучні нейронні мережі, постійне поповнення баз нечітких правил, що дозволить гнучко реагувати на зміни в робочому процесі компанії та постійно додаючи нові правила – постійно покращувати точність отриманого результату. [13 – 15].

2.2.3 Вибір методології реалізації пошуку схожості замовлень

Основною проблемою пошуку схожих пакетів – є те, що інфраструктура компанії не дозволяє отримувати список батьківських пакетів, чи, взагалі, отримати залежності між пакетами. Оскільки підтримка даного функціоналу неможлива за допомогою уже наявних інструметаріїв компанії – слід підібрати підхід для знаходження схожих пакетів, у списку тікетів, які були уже відправлені замовнику і у нас є уся необхідна інформація для знаходження таких тікетів.

Кожен тікет має назву компанії виробника програмного забезпечення, а також саму назву програми. Маючи ці вхідні дані, використовуючи алгоритми порівняння рядків, визначити схожість того чи іншого замовлення на пакування програмного забезпечення.

Відстань Левенштейна. Це міра відмінності двох рядків – двох символьних послідовностей. Даний алгоритм обчислює мінімальну кількість операція встановлення, видалення чи заміни для того щоб змінити одну послідовність символів до іншої.

Для розрахунку даної відстані застосовується простий алгоритм, в якому використовується матриця, яка має розмір:

$$(n + 1) * (m + 1), \qquad (2.1)$$

де n, m – довжини першого та другого рядка.

В даному алгоритмі, операції (вилучення, заміни, встановлення) мають однакову вартість.

Для відстані Левенштейна існують верхня і нижня межі:

- Дистанція Левенштейна не менша ніж різницю довжини рядків, які порівнюються;
- Вона не більша довжини самого довгого рядка;
- Вона дорівнює 0 тоді і тільки тоді, коли рядки однакові (однакові символи на однакових позиціях)

Між відстанню Левенштайна та відстанню Гемінга існують такі взаємозв'язки:

- Для рядків однакової довжини відстань Левенштайна рівна відстані Гемінга;
- Якщо рядки різної довжини, то верхньою межею є відстань Гемінга плюс різниця довжини рядків;

Відстань Дамерау – Левенштейна. Даний алгоритм є модифікацією попереднього, але з врахуванням доткової операції – транспозиції – перестановки двох сусідніх символів в послідовності.

У приблизній відповідності рядків мета полягає в тому, щоб знайти збіги для коротких рядків у багатьох довгих текстів, в ситуаціях, коли очікується невелика кількість відмінностей. Наприклад, короткі рядки можуть надходити з словника. Тут одна з рядків зазвичай коротка, а інша довільна. Це має широкий спектр додатків, наприклад, перевірки орфографії, системи виправлення оптичного розпізнавання символів та програмне забезпечення для сприяння перекладу природного мови на основі пам'яті перекладу.

Відстань Левенштейна також може бути обчислена між двома довшими рядами. Але вартість його обчислення, яка приблизно пропорційна добутку двох

довжин струни, робить це непрактичним. Таким чином, коли вони використовуються для пошуку нечітких рядків у таких програмах, як записування, порівняні рядки зазвичай короткі, щоб допомогти покращити швидкість порівняння.

Алгоритм Джаро-Вінклера. У комп'ютерних науках та статистиці відстань Джаро-Вінклера - це метрика рядків, яка вимірює відстань між двома послідовностями.

Відстань Джаро-Вінклера використовує масштаб префікса p, який дає більш сприятливі рейтинги для строк, які співпадають з самого початку для встановленої довжини префікса.

Чим нижче розташована відстань Джаро-Вінклера для двох рядків, тим більше схожих рядків. Оцінка нормується таким чином, що 1 прирівнює до подібності і 0 - точне збіг. Схожості Джаро -Вінклера дає 1-відстань Джаро-Вінклер.

Яро-Вінклер вираховує подібність між двома рядками, а повернене значення лежить в інтервалі [0.0, 1.0]. Це (грубо) варіація Дамерау – Левенштейна, де заміщення двох близьких символів вважається менш важливим, ніж заміщення 2 символів далеко один від одного.

Хоча часто називають метрикою відстані, відстань Джаро-Вінклера не є метрикою в математичному значенні цього терміну, оскільки вона не підкоряється нерівності трикутника.

Даний алгоритм використовують при переписі населення.

Найдовша спільна підпослідовність. Проблема пошуку найдовшої загальної підпослідовності (LCS) полягає у знаходженні найдовшої підпослідовності, загальної для двох (або більше) послідовностей. Якщо видалити з деякої кінцевої послідовності

деяку безліч її елементів (можливо порожній), то таким чином можна отримати підпослідовність.

Наприклад, BCDB є підпослідовність послідовності ABCDBAB. Вважаємо, що послідовність Z є спільною підпослідовність послідовностей X і Y, якщо Z є підпослідовність як X, так і Y. Потрібно для двох послідовностей X і Y знайти загальну підпослідовність найбільшої довжини.

Даний алгоритм використовується утилітою diff, за допомогою Git для узгодження кількох змін.

Відстань LCS еквівалентна дистанції Левенштейна, коли допускається лише вставка та видалення (без заміни) або коли вартість заміщення є подвійною вартістю вставки або видалення.

Висновок. Після перегляду алгоритмів, було вирішено що алгоритм Джаро-Вінклера буде використаний для пошуку схожих задач, оскільки він дозволить однозначно визначити відношення двох рядків (назв програмного забезпечення) при низьких затратах серверних ресурсів на визначення даних залежностей.

2.2.4 Вибір технологій реалізації інформаційної системи

На сьогоднішній день у розробників веб-додатків є великий вибір щодо того, які методи, мови програмування та технології використовувати для створення сайту. Існує багато варіантів: JSP, PHP, ASP.NET, Python, Node.js та багато інших. Найпоширенішими технологіями у наш час є PHP і ASP.NET, Python. Звачаючи, що дана ІС буде використовуватися компанією «Apptimized Operations» інструментарій підбираємо у відповідності з вимогами замовника та використовуваними інструментами розробки та використання в компанії.

Для реалізації інформаційної системи виберемо ASP.NET Core з архітектурою системи MVC.

Паттерн MVC, вбудований у функціонал серверної технології ASP.NET, був обраний на основі вимоги замовника та представляє собою схему розділення основної логіки додатку, графічного представлення та даних, під назвою «Model – View – Controller». Технологія ASP.NET MVC являється фреймворком, призваним полегшити застосування основних принципів паттерну MVC для розділення відповідальності за графічний інтерфейс, бази даних та обробки даних серед трьох основних компонентів: Моделей, Представлень (відповідає за окремі частини

графічного інтерфейсу веб-додатку) та Контролерів (відповідає за логіку обробки та управління даними веб-додатку).

Принцип роботи веб-додатку за патерном MVC представлений на рис. 2.4.

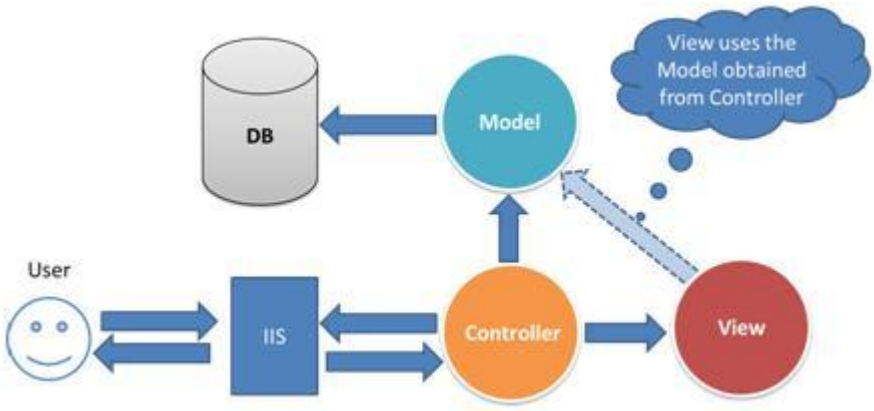

Рисунок 2.4 – Схема розділення веб-додатку за патерном MVC

Використання фреймворку дозволить вбудувати дану інформаційну систему в існуючий інструментарій керування.

ASP.NET Core - це нова веб-структура від Microsoft. Він був перероблений з нуля, щоб бути швидким, гнучким, сучасним і працювати на різних платформах. Переміщення вперед ASP.NET Core - це система, яка може бути використана для розробки веб-сторінок з .NET.

ASP.NET Core - це веб-структура, оптимізована для роботи з відкритим кодом та хмаркою, для розробки сучасних веб-додатків, які можуть бути розроблені та працювати на ОС Windows, Linux та Mac. Вона включає в себе структуру MVC, яка тепер поєднує в собі функції MVC і Web API в єдину систему веб-програмування.

Основні додатки ASP.NET можуть запускатися на .NET Core або на повній платформі .NET Framework.

Він був розроблений для забезпечення оптимізованої структури розробки для додатків, які розгортаються в хмарі або запускаються на місцевому рівні.

Він складається з модульних компонентів з мінімальними накладними витратами, тому ви зберігаєте гнучкість при побудові ваших рішень.

Додатки ASP.NET Core можна запускати на різних платформах на Windows,

Mac та Linux.

ASP.NET Core має наступні переваги:

- ASP.NET Core має ряд архітектурних змін, що призводять до створення набагато меншої та модульної структури.
- ASP.NET Core більше не базується на System.Web.dll. Він заснований на наборі пакетів NuGet.
- Переваги меншої площі поверхні додатків включають жорстку безпеку, зменшення обслуговування, підвищення продуктивності та зниження витрат.
- Створення та запуск крос-платформних додатків ASP.NET на Windows, Mac та Linux.
- Новий інструмент, що спрощує розробку Web.
- Веб-стек з єдиним виділеним інтерфейсом для веб-інтерфейсу.
- Обладнане середовище конфігурації.
- Вбудована підтримка ін'єкцій залежності.
- Tag Helpers, що робить розмітку Razor більш натуральною за допомогою HTML.
- Можливість хостингу в IIS або самостійному хості у вашому власному процесі.

Для реалізації алгоритму визначення виконавців, на даному етапі, існує безліч мов програмування, IDE для розробки та різних фреймворків. Наприклад: Python, Matlab, R, Java, LISP, Prolog тощо. Оскільки обрана мова програмування основної частини є C#, то слід відштовхуватися від цього. Існує безліч інтеграцій модулів між мовами програмування, але така система потребує більш обширних знань в мовах програмування і знань інтеграції таких «підпрограм». Таку систему важко підтримувати та модернізувати.

Середовище для розробки інформаційної системи – Microsoft Visual Studio - середовище розробки програмного забезпечення та безліч додаткових додатків, які розширяють функціонал середовища. Вони дозволяють розробляти: консольні програми, програми з графічним інтерфейсом (з підтримкою технології

Windows Forms), а також веб-сайти та вею-системи як в рідному, так і в керованому кодах для всіх платформ, які підтримуються Microsoft Windows.

2.2.5 Вибір технологій збереження інформації

Вибір методу є досить важливим, оскільки доступ до даних повинен бути швидким, та надійним. На сьогоднішній день, існує багато СУБД та методів зберігання та організації даних.

MySql. MySQL - це система управління реляційною базою даних з відкритим кодом, захищена Oracle (RDBMS) на основі мови Structured Query Language (SQL). MySQL працює практично на всіх платформах, включаючи Linux, UNIX та Windows. Хоч, вона може звикористовуватися в широкому діапазоні застосувань, MySQL найчастіше асоціюється з веб-додатками та онлайн-публікаціями.

MySQL є важливим компонентом стекового середовища з відкритим кодом під назвою LAMP. LAMP - це платформа веб-розробки, яка використовує Linux як операційну систему, Apache як веб-сервер, MySQL як систему управління реляційною базою даних, а також PHP як об'єктно-орієнтовану мову сценаріїв. (Іноді замість PHP використовується Perl або Python.)

Сьогодні MySQL є СУБД на багатьох провідних веб-сайтах у світі та незліченних корпоративних та споживчих веб-додатків, включаючи Facebook, Twitter та YouTube.

MySQL базується на моделі клієнт-сервер. Ядро MySQL - це сервер MySQL, який обробляє всі інструкції (або команди) бази даних. Сервер MySQL доступний як окрема програма для використання в мережевому середовищі клієнт-сервер та як бібліотека, яку можна вбудувати (або зв'язати) в окремі додатки.

MySQL працює поряд з кількома утилітними програмами, які підтримують адміністрування баз даних MySQL. Команди надсилаються до MySQLServer через клієнт MySQL, який встановлюється на комп'ютері.

MySQL був спочатку розроблений для обробки великих баз даних швидко. Хоча MySQL зазвичай встановлюється лише на одній машині, він може надсилати

базу даних у декілька розташувань, оскільки користувачі можуть отримати доступ до нього за допомогою різних клієнтських інтерфейсів MySQL. Ці інтерфейси надсилають SQL-заяви на сервер, а потім відображають результати.

SQL Server. Microsoft SQL Server - це система управління реляційною базою даних, або RDBMS, яка підтримує широке коло обробки транзакцій, бізнес-аналітики та аналітичних додатків в корпоративних середовищах IT. Це одна з трьох провідних технологій баз даних, а також Oracle Database та IBM DB2.

Як і інше програмне забезпечення RDBMS, Microsoft SQL Server побудований на вершині SQL, стандартизованої мови програмування, яку адміністратори баз даних (DBA) та інші спеціалісти з інформаційних технологій використовують для управління базами даних та запиту даних, які вони містять. SQL Server прив'язаний до Transact-SQL (T-SQL), реалізація SQL з Microsoft, яка додає набір власних програмних розширень на стандартну мову.

Microsoft SQL Server 2016, який став загальнодоступним у червні 2016 року, був розроблений як частина "технології першої технології першої технології", яка була прийнята корпорацією Майкрософт два роки тому. Серед іншого, SQL Server 2016 додав нові функції для налаштування продуктивності, оперативної аналітики в режимі реального часу та візуалізації даних і звітування на мобільних пристроях, а також підтримку гібридної підтримки хмар, що дозволяє базам даних DBA запускати комбінацію локальних систем і загальнодоступних хмарних служб зменшити витрати на IT. Наприклад, технологія SQL Server Stretch Database переміщує нечасті доступні дані з локальних пристроїв зберігання до хмарного пакета Microsoft Azure, зберігаючи при необхідності дані для запитів.

Sqlite. SQLite – рантайм-бібліотека, яка реалізує автономну, безсерверну, конфігурацію баз даних SQL.

Код SQLite є загальнодоступним і, таким чином, безкоштовним для використання для будь-яких цілей, комерційних або приватних. SQLite - це найпоширеніша в світі база даних з великою кількістю додатків.

SQLite - вбудований двигун бази даних SQL. На відміну від більшості інших баз даних SQL, SQLite не має окремих серверних процесів. SQLite читає та пише

безпосередньо до звичайних файлів. Повна база даних SQL з декількома таблицями, індексами, активаторами та представленнями міститься в одному файлі диска. Формат файлу бази даних є крос-платформою - ви можете вільно копіювати базу даних між 32-бітними та 64-бітними системами або між великими і менш-кінцевими архітектурами. Ці особливості роблять SQLite популярним вибором як формат файлу додатків. Файли бази даних SQLite є рекомендованим форматом зберігання бібліотеки Конгресу США.

SQLite - це компактна бібліотека. Якщо увімкнено всі функції, розмір бібліотеки може бути меншим за 600 Кб, залежно від цільової платформи та параметрів оптимізації компілятора. (64-розрядний код більший. І деякі оптимізаційні комбінатори, такі як агресивна функція inlining та розгортання циклу, можуть призвести до значного збільшення об'єктного коду.)

SQLite використовує динамічні типи для таблиць. Це означає, що можна зберігати будь-яке значення в будь-якому стовпці, незалежно від типу даних.

SQLite дозволяє єдиному підключенню бази даних одночасно отримувати доступ до декількох файлів бази даних. Це приносить багато приємних функцій, таких як приєднання таблиць до різних баз даних або копіювання даних між базами даних в одній команді. SQLite здатний створювати бази даних у пам'яті, які дуже швидко працюють.

SQLite дуже ретельно тестується перед кожним релізом і має репутацію дуже надійного додатку. Автоматизований набір тестів містить мільйони і мільйони тестів, що включають сотні мільйонів окремих SQL-повідомлень, і забезпечує 100% охоплення тесту. SQLite чудово реагує на помилки виділення пам'яті та помилки введення / виведення диска.

Рисунок 2.5 – Приклад роботи Sqlite БД

SQLite - це автономна система, що вимагає мінімальної підтримки операційної системи або зовнішньої бібліотеки. Це робить SQLite доступним у будь-яких середовищах, особливо в вбудованих пристроях, таких як iPhone, Android телефони, ігрові приставки, портативні медіапрогравачі тощо.

Висновок. Проаналізувавши наявні СУБД та підходи, оцінивши всі переваги та недоліки кожного підходу – SQLite є найбільш підходящою БД, оскільки дозволяє зберігати дані у окремому файлі без встановдення додаткових серверів підтримки бази даних. Це дозволить працювати системі значно швидше, враховуючи невеликий об’єм даних який буде зберігати спроектована БД.

3 МОДЕЛЮВАННЯ ІНФОРМАЦІЙНОЇ СИСТЕМИ

Інформаційна система — сукупність організаційних і технічних засобів, як організаційних так і технічних, які дозволяють зберігати та обробляти інформацію з метою забезпечення потреб користувачів.

Моделювання інформаційної системи аналізу та розподілу робочих задач для компанії «Apptimized Operations» проводилось на основі виконаного аналізу діяльності компанії в цілому, а також процесу розподілу задач між командою пакувальників в часності, які були описані в розділі 1.

3.1 Побудова контекстної діаграми в нотації IDEF0

Методологія IDEF0 застосовується в IT галузях як ефективний засіб аналізу, проектування та подання бізнес процесів. Основною одиницею IDEF0-моделі є діаграма - графічний опис моделі предметної області або її частини. Головними компонентами IDEF0-діаграми є блоки, які відображають роботи, процеси, завдання, різні функції тощо. Вони, зазвичай, виконуються протягом певного часу та мають, відповідно, резудьтат. У кожної сторони структурного блоку є своє призначення, яке так чи інакше впливає на процеси :

- ліва сторона – вхід;
- права сторона – вихід, отриманий результат;
- верхня – управління процесу, елементи, які впливають на вихідний результат;
- нижня – межанізми процесу;

Основною задачею інформаційної системи є – визначення декількох виконавців для відповідної нової задачі, які задовольняють параметри, через які йде опис завантаженості робітника.

Відповідно до методу IDEF0 для будь-якої роботи необхідно визначити вхідні дані, вихідні дані, управління і механізм, які зображуються на діаграмі стрілками. Тож для даної ІС та процесу розподілу задач визначаємо наступні параметри:

- Вхідні дані: необхідність автоматизації документообігу деканата.
- Вихідні дані: документація, створена система
- Управління: нормативні документи та ТЗ підсистеми.
- Механізми: технічне та програмне забезпечення.

Таким чином, контекстна діаграма містить блок «Моделювання процесу пакування програмного забезпечення у межах компанії “Apptimized Operations”». Вона має рівень А0. Це найвищий рівень абстракції для даного завдання.

Основною точкою зору була взята точка зору голови відділу пакування компанії «Apptimized Operations», а також було визначено мету – аналіз роботи відділу пакування.

Контекстна діаграма представлена на рис. 3.1.

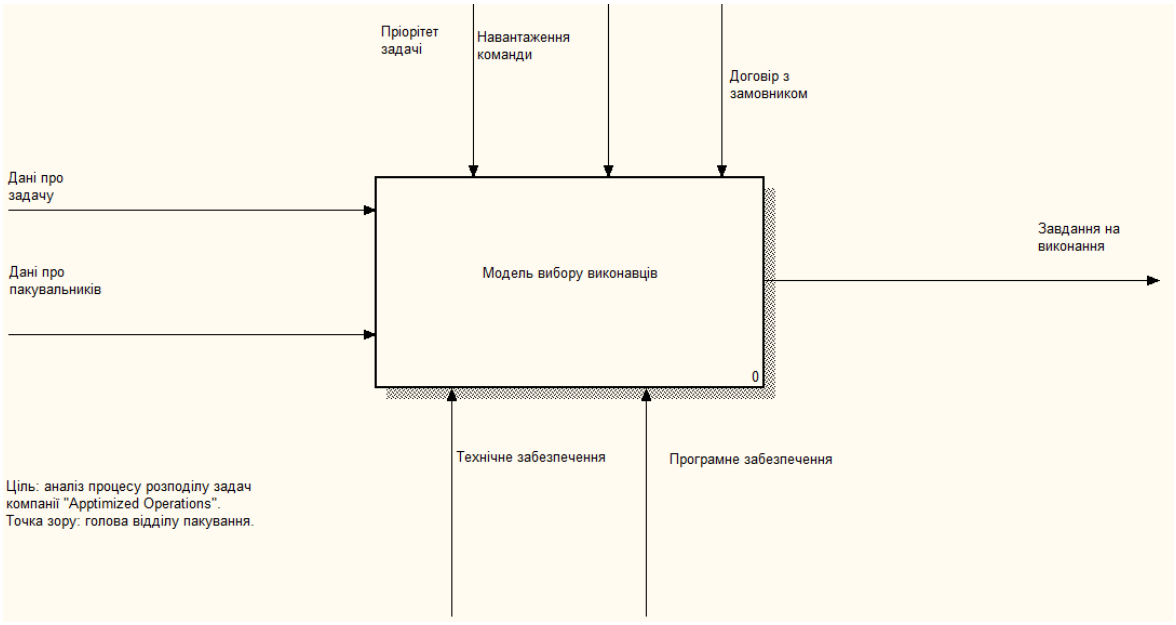

Рисунок 3.2 – Контекстна діаграма процесу вибору виконавця у нотації IDEF0

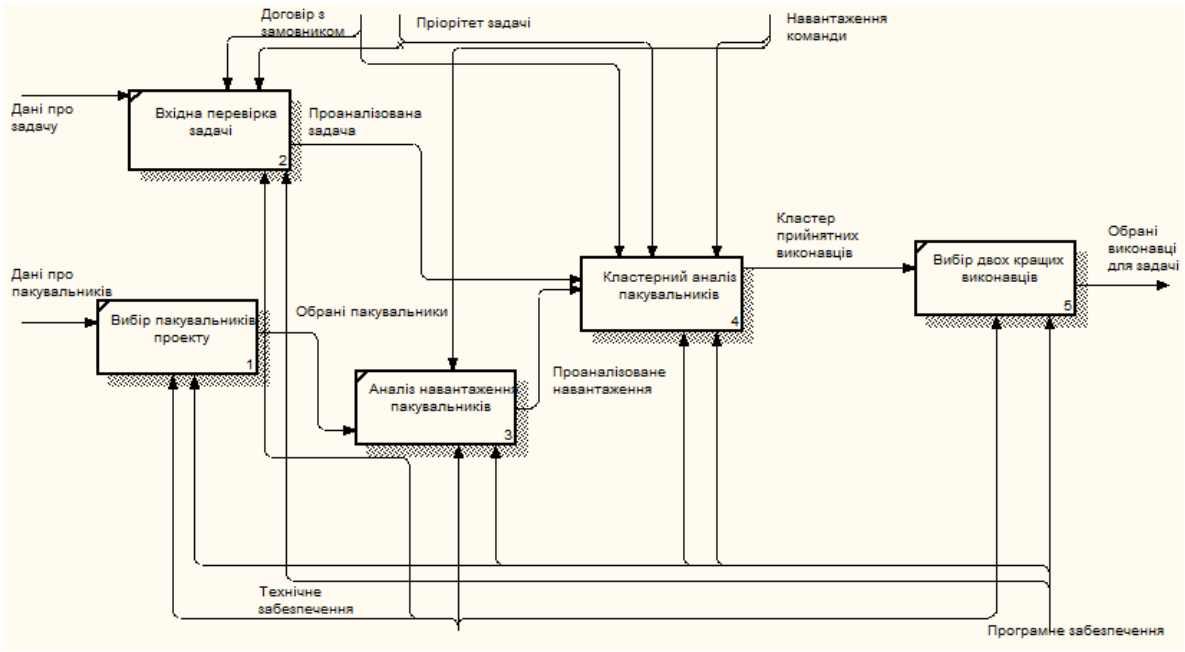

Рисунок 3.2 – Діаграма декомпозиції процесу вибору виконавця у нотації IDEF0

3.2 Побудова моделі прецедентів

За методологію створення ІС обрано метод використання UML-діаграм, який дозволяє стисло та легко визначати особливості проектування та розробки інформаційної системи у візуальному вигляді.

В той час, як вимоги можуть бути представлені у вигляді списків, таблиці або інших стандартних засобів проектної документації, то інформаційна система аналізу та розподілу робочих задач, а також її основний функціонал та використання може бути представлена за допомогою UML-діаграми.

Основна мета даної діаграми – система представляється у вигляді безлічі акторів та сутностей, які взаємодіють із системою за допомогою визначених варіантів використання.

Варіант використання (англ. use case) – це опис функцій чи послуг, які система надає актору. Кожен варіант використання має на увазі деякий набір дій, який виконує система при взаємодії з актором. При цьому реалізація цих дій не описується.

У ролі основного користувача системи виступають представники керівництва компанії «Apptimized Operations», об’єднані в одного актора під назвою «Керівництво компанії». Він може використовувати користувацький інтерфейс як систему доступу до ІС, а саме:

- переглядати список проектів компанії з визначеними робітниками для певного проекту;
- переглядати поточне навантаження робітника;
- переглядати інформацію та можливу оцінити можливі затрати часу для певного замовлення на пакуванння ПЗ, базуючись уже на існуючих даних по схожим замовленням;
- обирати на назначати замовлення на пакування ПЗ на обраного робітника.

Також до моделі прецедентів входять такі актори, як «Адміністратор системи» та «Інженер технічної підтримки». Обов’язками адміністратора є підтримка системи. Інженер технічної підтримки займається удосконаленням веб-інтерфейсу за вимогами користувачів системи.

Основні варіанти використання інформаційної системи представлені на рис. 3.3.

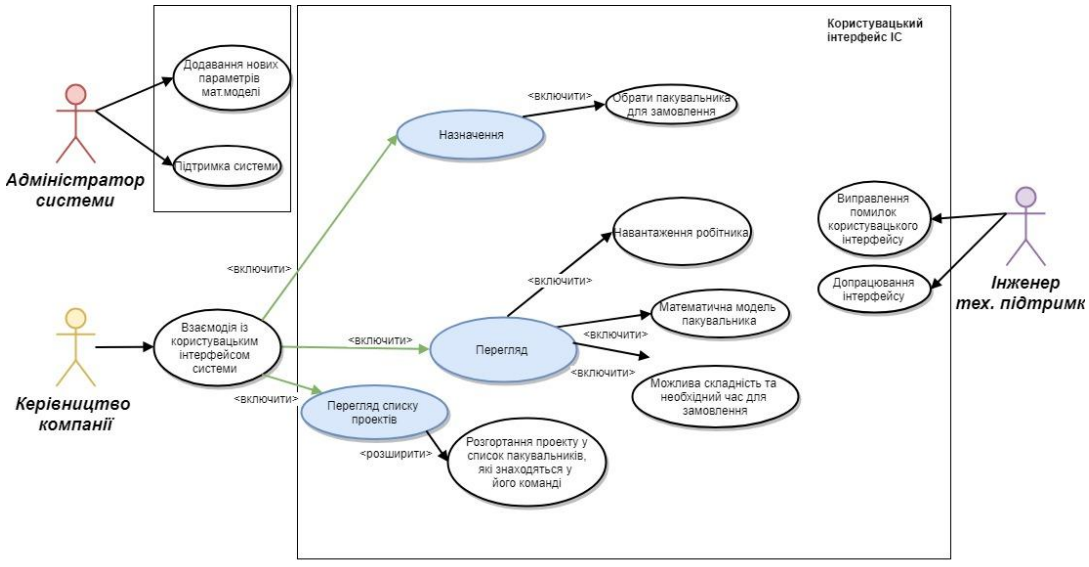

Рисунок 3.3 – Модель прецедентів проекту створення інформаційної системи

3.2 Моделювання системи

Виходячи з функціональних вимог від курівництва компанії, проведеного аналізу діяльності компанії, а також процесу розподілу задач між командою пакувальників ПЗ – модель системи набуває специфічного характеру, котрий повинен задовільнити потреби.

Збір метаданих для роботи системи. Модель даних – це абстрактний опис об’єктів реального світу, які відносяться до обраної предметної області розроблюваної системи [24].

У випадку із розробкою ІС для ІТ-компанії «Apptimized Operations», джерелами інформації слугують 3 системи: «JIRA, та 2 внутрішні продукти – «TimeTracker» для введення й моніторингу витраченого на пакування ПЗ часу, база даних веб-системи «Apptimized Intranet».

«JIRA» включає дані про проекти та пакети ПЗ, що перебувають у різних статусах – від “замовлення пакету” (заявка на який тільки поступила у систему, та процес пакування ще не розпочатий) до “затверджено замовником” (позначає, що пакет ПЗ пройшов внутрішнє та зовнішнє тестування і вже затверджений клієнтом).

Дані зберігаються в окремих “тікетах” (сторінках замовлень з пакування ПЗ) проектів та пакетів, та будуть заноситися до бази даних за допомогою додаткового програмного модулю синхронізації даних.

«TimeTracker» через окремий API надає можливість отримувати дані про занесений час пакування окремого замовлення пакету ПЗ, та дізнатися дані із усього часу, занесеного на пакування ПЗ одним спеціалістом за місяць. Таким чином можливо зберегти дані про кількість часу, який пакувальник ПЗ витратив на виконання того чи іншого замовлення на пакування, що буде використане для визначення для нових замовленнь можливі витрати часу базуючись на вже відправлених схожих замовленнях на пакування.

3.2.1 Критерії оцінювання навантаження працівника

Кожен робітник (пакувальник) весь час має певну кількість поточних робочих задач, які можна описати за допомогою певних критеріїв, які, в свою чергу, впливають на її складність, час до дедлайну тощо. Дані критерії задач були визначені в першому розділі.

Але, крім критеріїв задачі є критерії самого робітника, які теж повинні бути враховані під час процесу розподілу робочих задач. Серед них можна виділити наступні критерії:

- доступність робітника (чи знаходить він/вона у відпустці);
- попередній виконавець (чи виконував уже схожу задачу).

Зважаючи на специфіку діяльності, критерій попереднього виконавця може зіграти значну роль у виконанні задачі, оскільки робітник вже має досвід пакування того чи іншого софту, та повинен впоратись, як показує практика, набагато швидше.

Визначивши дані критерії, слід привести їх у відповідний вигляд, для чіткого розуміння як людиною так і програмою. Опис критеріїв представлений у таблиці 3.1.

Таблиця 3.1 – Критерії оцінки підбирання робітників

Група критерію	**Критерій**	**Тип даних**	**Діапазон**
Критерії задачі	Час	double	[0,10]
	Пріорітет	double	[0,10]
	Статус	double	[0,10]
	К-сть задач	double	[0,10]
	Категорія	double	[0,10]
Критерії робітника	Днів до відпустки	int	[0,10]
	Попередній виконавець	boolean	[0,10]
	Навантаження	double	[0,10]

Описані критерії мають ваги важливості, що дозволяє, змінювавши їх, визначать більш важливий критерій в той чи інший момент часу.

Дані критерії приведені до однакових одиниць виміру, оскільки використання різних діапазонів можливих значень може призвести до некоректної роботи системи та, як наслідок отримання викидів в результатах під час обробки невалідних об’єктів.

Тому, приведення діапазонів можливих значень до однакового маштабу дозволить максимально мінімізувати можливість виникнення таких ситуацій.

Для отримання числового представлення критеріїв необхідно проводити математичні оперіції на данними. Тож для визначення часового критерію була використана наступна формула:

$$SlaRatio = \sum \frac{SlaConsumedTime}{SlaTotalTime}, \qquad (3.1)$$

Даний критерій розраховується для пакувальника наступним чином – для кожного поточного завдання робітника визначається відношення витраченого часу до повного часу на задачу. В результаті отримуємо суму таких відношень для кожної задачі.

Наступним критерієм є пріорітет. Формула обчислення для пакувальника представлена нижче.

$$PriorRatio = \sum PriorityRate, \qquad (3.2)$$

В даній формулі – PriorityRate – це статичне число, яке визначається для кожного пріорітету, який вже є в системі «Jira». Для робітника – сума таких чисел для кожного із його поточних завданнь.

Наступним критерієм є статус. Формула обчислення для пакувальника представлена нижче.

$$StatusRatio = \sum StatusRate, \qquad (3.3)$$

В даній формулі – StatusRate – це статичне число, яке визначається для кожного стутус тікета, який вже є в системі «Jira». Для робітника – сума таких чисел для кожного із його поточних завданнь.

Наступним критерієм є кількість задач. Формула обчислення для пакувальника представлена нижче.

$$TicketRatio = \sum \frac{NumberOfTickets}{TotalTickets/TeamMembersCount}, \qquad (3.4)$$

В даній формулі – $TicketRatio$ – це відношення кількості поточних задач робітника до середньої кількості задач, що має пакувальник.

Наступним критерієм є статус. Формула обчислення для пакувальника представлена нижче.

$$CategoryRatio = \sum CategoryRate, \qquad (3.5)$$

В даній формулі – CategoryRate - це статичне число, яке визначається для кожної категорії тікета, який вже є в системі «Jira». Для робітника – сума таких чисел для кожного із його поточних завданнь.

Дні до відпустки – це критерій, який визначається для кожного робітника, на основі даних які були отримані із зовнішньої системи трекінга відпусток робітників. Завдяки такій інтергації, можна чітко знати інтервали відпусток робітників, та уникати випадків, коли система буде рекомендувати людей, які фізично знаходяться поза офісом.

Якщо, у пакувальника залишається до відпустки днів менше чим 4, то такий пакувальник не буде рекомендуватися системою, оскільки, зважаючи на досвід роботи компанії, малоймовірно, що робітник встигне завершити нові задачі за такий короткий період часу.

Попередній виконавець – це критерій, який визначається на основі вже виконаних завдань на пакування робітником. Даний критерій є булевим, так як робітник або виконував схожі завдання або ні.

Даний параметр визначається наступними формулами – порівнючи схожість рядків:

$$d_j = \begin{cases} 0 \\ \frac{1}{3}\left(\frac{m}{|s_1|} + \frac{m}{|s_2|} + \frac{m-t}{m}\right), \end{cases} \qquad (3.6)$$

В даній формулі $|S_1|$ – довжина рядка S_1, m – число співпавших символів.

$$d_w = d_j + \left(lp\left(1 - d_j\right)\right), \qquad (3.7)$$

Даний підхід відомий як алгоритм Джаро-Вінклера, який дозволяє визначити схожість двох вхідних рядків. Неформально, відстань Джаро між двома словами - це мінімальна кількість односимвольних перетворень, яка необхідна для того, щоб змінити одне слово в іншому.

Навантаження робітника – це відношення кількості витраченого часу робітником за певний період часу до середнього даного показника по команді пакувальників. Даний параметр дозволяє визначити, чи був робітник перенавантажений протягом певного періоду, чи він був навантажений менше ніж команда.

$$TicketRatio = \sum \frac{EmployeeWorkload}{TotalWorkload / TeamMembersCount}, \qquad (3.8)$$

3.2.2 Моделювання процесу підбору виконавців

На основі проведеного аналізу та порівняння алгоритмів вибір був зроблений на користь стандартного кластерного аналізу на основі K-means. Даний метод машинного навчання дозволяє визначити групу робітників, які можуть виконувати ту чи іншу задачу базуючись на критеріях описаних вище.

В основі кластерного аналізу важливу роль відіграє алгоритм визначення принадлежності елементу до того чи іншого кластеру. Зазвичай, у класичному кластерному аналізу K-Means використовують визаченн відстані від центроїда до елемента за допомогою Евклідової відстані.

$$d = \sqrt{\sum_{i=1}^{n} (q_i - p_i)^2}, \qquad (3.9)$$

Де, q та p критерії об’єктів описані у відповідному розділу. Зважачи, на специфіку задачі та кількість параметрів опису працівника – формула має бути змінена. Додамо вагу елементу, як множник до результату.

$$d = w * \sqrt{\sum_{i=1}^{n} (q_i - p_i)^2}, \qquad (3.10)$$

Додавши, даний множник, ми даємо знати системі, який критерій є більш важливішим на даний момент часу.

Після формування кластерів, використавши дану формулу – визначаємо «найближчий» кластер до «ідеального» елементу (зважаючи на критерії).

3.2.3 Моделювання аналізу задачі

Якщо розглядати випадок, коли інформаційна система знайшла с базі даних схожі пакунки, які вже були зроблені та відправлені замовнику – проводиться аналіз нової задачі на основі минулих задач.

Цей процес повинен бути доволі простим, але повинен давати поверхневий аналіз задачі. Очікується, що система може вивести можливу складність пакунку – цей параметр є доволі суб’єктивним, оскільки кожне замовлення має безліч конфігурацій, які впливають на заключну складність пакету.

Для визначення можливої складності використаємо формулу:

$$Complexity = {\sum ComplexityRate * ProjectRate}/{TicketsAmount}, \qquad (3.11)$$

За допомогою даної формули – середнього значення складності пакунків, будемо визначати можливі витрати часу на даний пакунок.

Для реалізації даного функціоналу буде використаний простий підхід – лінійний регресійний аналіз, який дозволяє прогнозувати невідомий параметр на основі відомого (значення середньої складності).

Загалом лінійна регресійна модель визначається у виді:

$$y = \beta_0 + \beta_1 x_1 + \cdots + \beta_k x_k + u, \qquad (3.12)$$

де y – залежна пояснювана змінна, x - незалежні змінні, u – випадкова похибка.

Тож, відповідно до моделі, описаної вище – математичне сподівання для залежної змінної - це лінійна функція незалежної змінної і розраховується наступною формулою:

$$y_i = \beta_0 + \beta_1 x_1 + \cdots + \beta_k x_k, \qquad (3.13)$$

Для отримання оптимальних результатів від регресійного аналізу – використовуємо поширений метод найменших квадратів. Даний метод дозволяє оцінити параметр значення, який мінімізує суму квадратів залишків по всій послідовності вхідних даних.

Аналітичний запис даного методу виглядає так:

$$\hat{\beta} = (X' * X)^{-1} * X'y, \qquad (3.14)$$

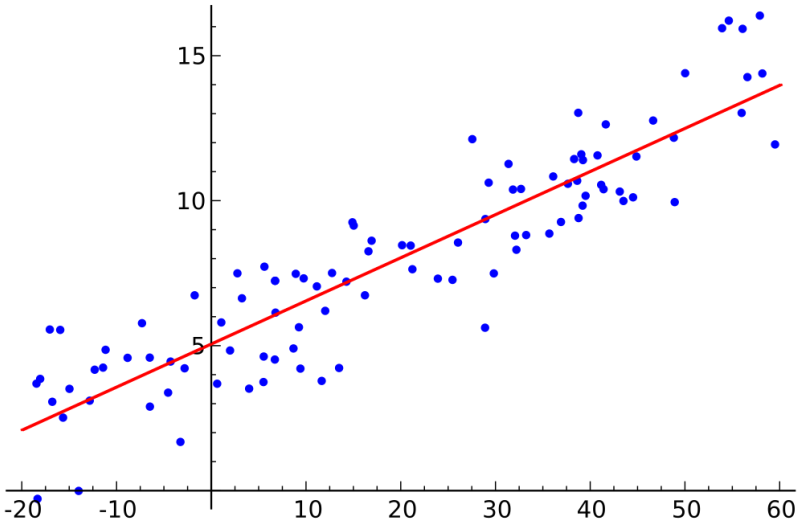

Рисунок 3.4 – Приклад лінійної регресії

Даний метод дозволить оцінити можливі витрати часу на ту чи іншу задачу, як тільки вона надходить до внутрішньої системи «Jira» і дозволить керівництву компанії, керуючись результатами аналізу покращити планування навантаження як команди в цілому, так і пакувальника окремо.

3.2.4 Моделювання бази даних

База даних (скорочено – БД) – це сукупність інформації і взаємозалежних даних, які використовуються для задоволення потреб користувачів.

Ці дані організовані і структуровані за правилами, які передбачені загальними принципами опису, зберігання і маніпулювання, незалежно від прикладних програм [14].

Звертання до баз даних відбувається за допомогою систем керування базами, які забезпечують підтримку, керування тощо.

Основна мета проектування бази даних - це скорочення надмірності збережених даних, а отже, економія об'єму використовуваної пам'яті, пришвидшення взаємодії з даними і обробки запитів. Результатом проектування БД є перетворення опису предметної області у внутрішню схему БД [14].

У ході проектування БД вирішуються наступні задачі:

- виявлення і представлення даних, а також їх зв’язків, необхідних для точного опису всих основних областей застосування даного додатка;
- створення моделі даних;
- розробка попереднього варіанта проекту, структура якого дозволяє задовольнити всі основні вимоги з боку продуктивності системи [14].

Задачі й етапи проектування БД випливають із трирівневої схеми (так називаної моделі ANSI/SPARC) даних (зовнішня – концептуальна – внутрішня).

Для функціонування інформаційної системи було виділено основні сутності предметної області, проведене концептуальне та логічне моделювання.

Високорівнева структура бази даних представлена на рисунку 3.5.

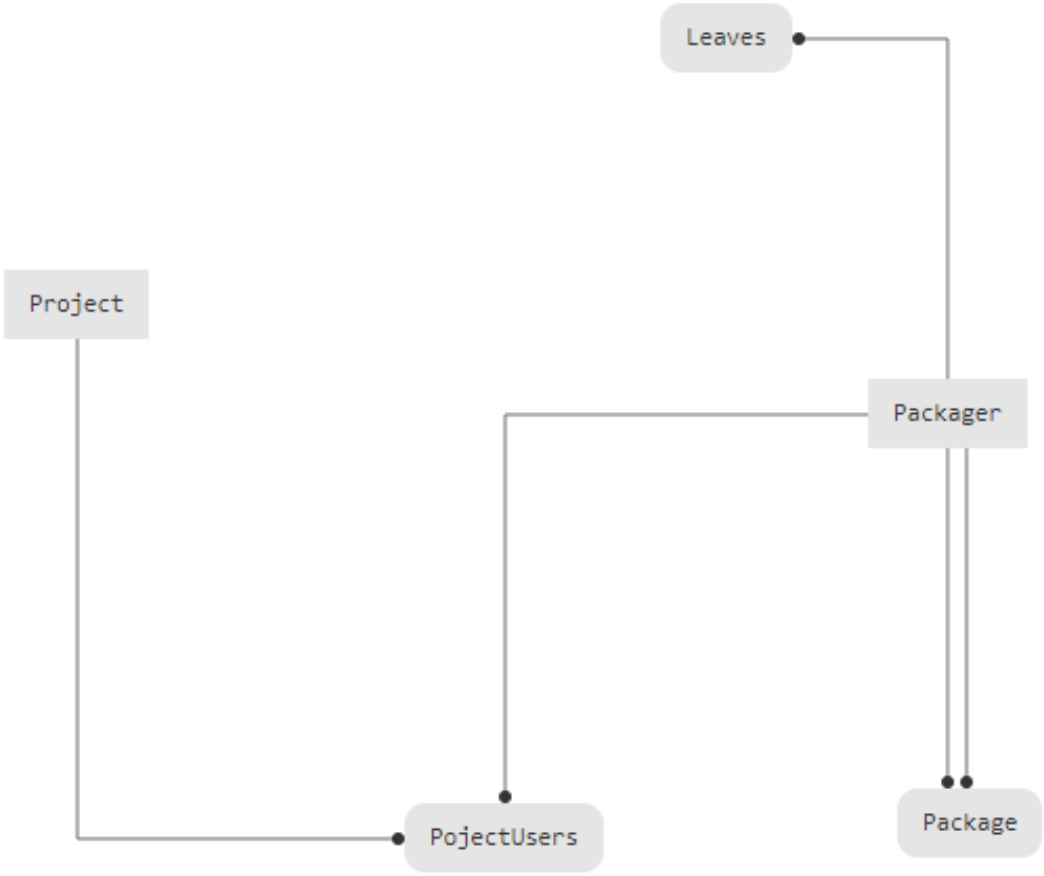

Рисунок 3.5 – Високорівнева структура бази данних

Кожна виділена сутність предметної області має певне призначення. Так в табл. 3.2 описана характеристика сутностей бази даних системи.

Таблиця 3.2 – Опис сутностей бази даних

Сутність	Характеристика
Packager	Дана сутність буде містити всіх користувачів системи та даних про них
Project	Дана сутність буде містити всі проекти та даних про них
Leaves	Відпустки використовуються для визначення доступності користувачів
Package	Таблиця бази даних яка буде містити всю інформацію про пакунки та пов’язану з ними

	інформацію
ProjectUsers	Проміжна таблиця для організації зв'язку «багато-до-багатьох»

Кожна таблиця має спеціально визначені поля для зберігання відповідної інформації.

Таблиця 3.3 – Опис атрибутів таблиці Packager

Назва	**Опис**
Id	Первинний ключ таблиці Packager, який однозначно ідентифікує запис; зв’язаний з зовнішнім ключем в таблиці Package, Leaves, ProjectUsers
Full Name	Повне ім’я робітника
UserName	Ім’я, яке використовується в системах компанії

Таблиця 3.4 – Опис атрибутів таблиці Project

Назва	**Опис**
Id	Первинний ключ Project, який однозначно ідентифікує запис; зовнішній ключ в таблиці Package, ProjectUsers
Name	Ім’я робітника
Weight	Коефіцієнт важкості проекту

Таблиця 3.5 – Опис атрибутів таблиці Leaves

Назва	**Опис**
Id	Первинний ключ таблиці Leaves, який однозначно ідентифікує запис
UserId	Містить Id запису з таблиці Packager; зовнішній ключ
Till	Дата завершення відпустки
From	Дата початку відпустки

Таблиця 3.6 – Опис атрибутів таблиці Package

Назва	**Опис**
Id	Первинний ключ таблиці Package, який однозначно ідентифікує запис
Packager	Містить Id запису з таблиці Packager; зовнішній ключ
QAEngineer	Містить Id запису з таблиці Packager; зовнішній ключ
ApplicationVendor	Назва виробника програми
ApplicationName	Назва програми
Key	Внутрішній ключ в системі “Jira”
Priority	Пріорітет задачі
Category	Категорія задачі

Таблиця 3.7 – Опис атрибутів таблиці ProjectUsers

Назва	**Опис**
Id	Первинний ключ таблиці ProjectUsers, який однозначно ідентифікує запис
UserId	Містить Id запису з таблиці Packager; зовнішній ключ
ProjectId	Містить Id запису з таблиці Project; зовнішній ключ

Повна структура бази даних представлена на рисунку 3.6.

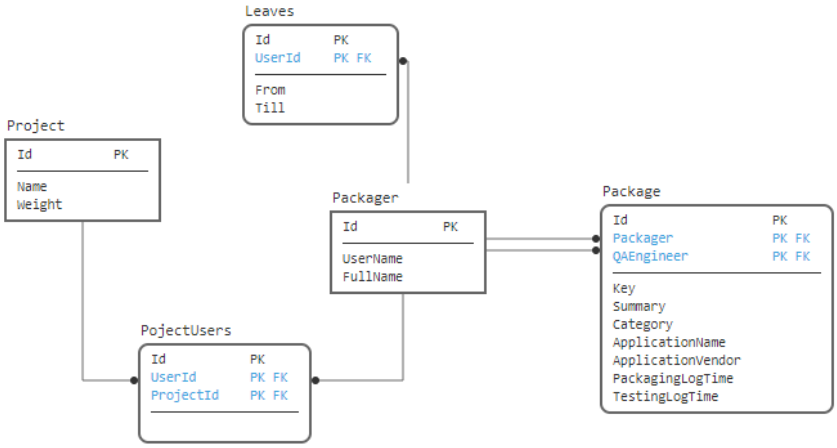

Рисунок 3.6 – Повна структура бази даних

3.3 Робота інформаційної системи

Описавши критерії оцінки пакувальників, обраний алгоритм обробки об’єктів можна описати стандартний алгоритм роботи системи – end2end кейс, який дозволить наявно побачити які функції та операції виконуються протягом простої роботи системи для отримання списку людей, які можуть виконати задачу базуючить на критеріях. Високорівневу процес можна продемонструвати на рисунку 3.7.

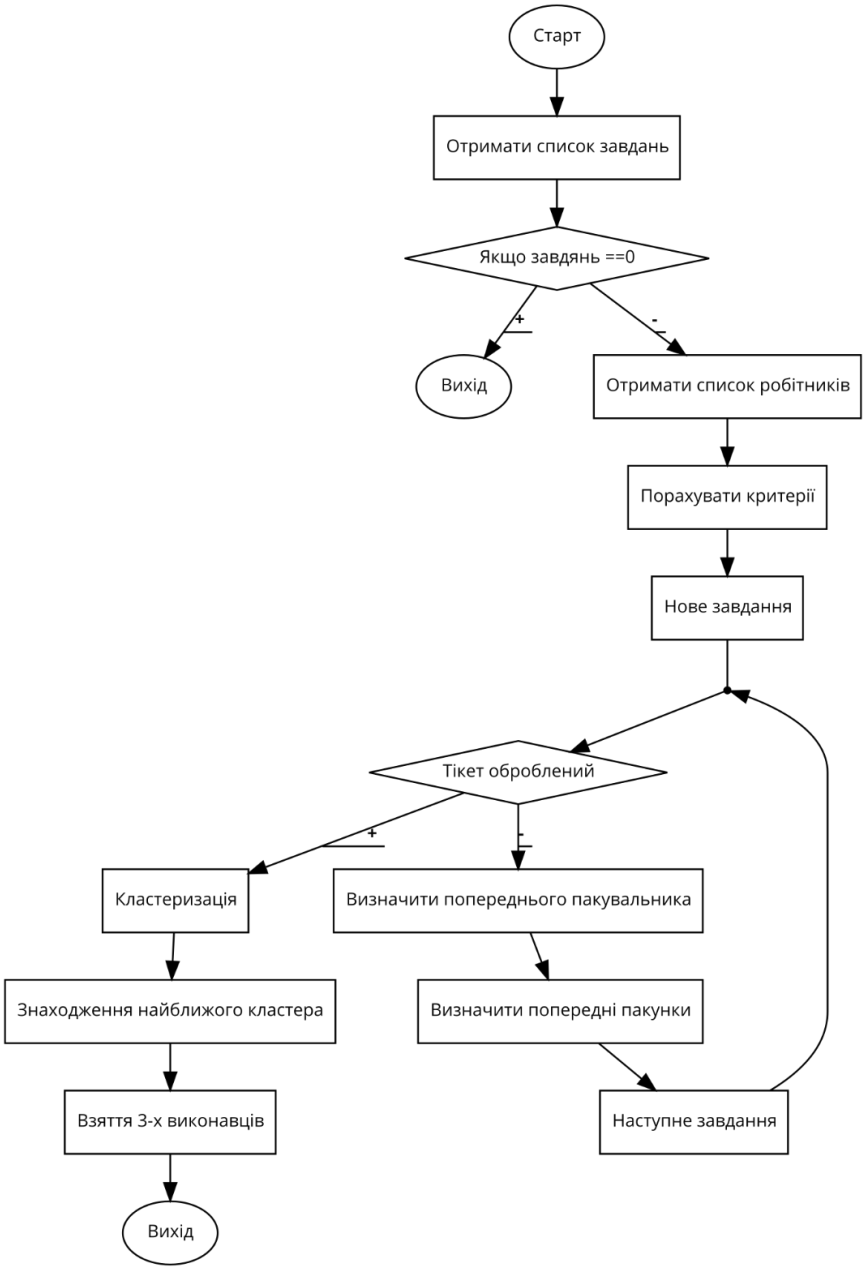

Рисунок 3.7 – Високорівневий алгоритм визначення виконавців

Дане представлення є дуже схематичним, але дозволяє зрозуміти кроки та основні етапи визначення виконавців для кожного нового завдання.

Зважаючи на дане представлення можна сказати, що процес обробки інформації є доволі часозатратним, оскільки даний алгоритм виконує дволі багато процесів. Все починається з отримання списку тікетів із зовнішньої системи «Jira». Після того, для кожного з робітників визначаються критерії, які описують його навантаження. Після цього, відбувається прохід через всі тікети, для яких визначаються попередні пакувальники, та попередні тікети.

Цей процес супроводжується отриманням даних із системи «TimeTracker», яка повертає дані про використаний час стосовно того чи іншого пакунка. Отримані дані структуруються для подальшого використання. Після даної обробки – виконується кластерний аналіз об’єктів, які були сформовані на попередніх етапах.

Після проведення кластеризації визначається необхідний кластер та виводяться на екран відібрані виконавці. На цьому етапі даний алгоритм завершує своє виконання.

4 РЕАЛІЗАЦІЯ ІНФОРМАЦІЙНОЇ СИСТЕМИ

4.1 Реалізація інформаційної системи

База даних – це один з ключових елементів будь-якого програмного продукту. Правильно сформована база даних дозволяє повністю розробити необхідний функціонал програмного продукту, а також, за необхідності, підтримувати розширення на нарощення функціоналу системи. На основі аналізу, проведеного в підрозділі 3.2.3, були розроблені моделі відповідно до патерну MVC. Для збереження даних була обрана база даних Sqlite. Вона дозволяє без встановлення додаткових СУБД та програм взаємодіяти з базою за допомогою бібліотек, які вбудовуються в основний додаток під час компіляції.

Даний підхід дозволяє легко взаємодіяти з базою даних та зменшити навантаження на сервер, де знаходиться додаток та полегшити роботу самого додатку.

Перелік таблиць приведена на рисунку 4.1.

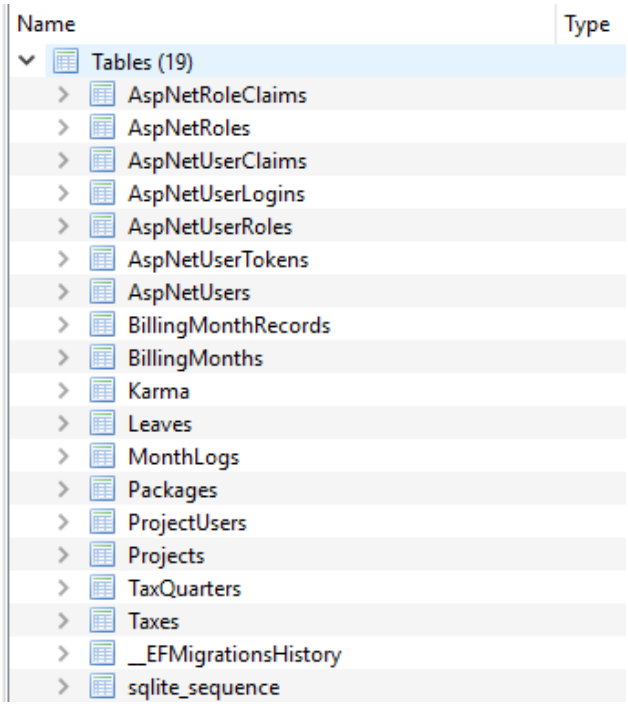

Рисунок 4.1 – Перелік таблиць бази даних

Дані таблиці були сформовані під час виконання проекту, відповідно до розроблених моделей (відповідно до патерну NVC).

Були розроблені такі моделі:

- Package.cs
- ApptimizedUser.cs
- Leaves.cs
- Project.cs
- ProjectUsers.cs

Оскільки в проект побудований на технологіях ASP.NET MVC, тому для покращення та пришвидшення роботи з базою даних був використаний EntityFramework – об’єктно-орієнтована технологія доступу до даних, яка дозволяє взаємодіяти з об’єктами використовуючи LINQ.

Щоб підключитися до бази даних через Entity Framework, бувреалізований контекст даних. Контекст даних являє собою клас, похідний від класу DbContext. Контекст даних містить одне або кілька властивостей типу DbSet <T>, де T представляє тип об'єкта, що зберігається в базі даних.

Так був розроблений контекст ApptimizedDbContext.cs який саме виконує ці функції (рис. 4.2)

```
public ApptimizedDbContext(
    DbContextOptions<ApptimizedDbContext> options) : base(options)
{
}

public DbSet<Leave> Leaves { get; set; }

public DbSet<ApptimizedUser> ApptimizedUser { get; set; }

public DbSet<Project> Projects { get; set; }

public DbSet<Package> Packages { get; set; }

public DbSet<ProjectUsers> ProjectUsers { get; set; }
```

Рисунок 4.2 – Контекст даних системи

Оскільки дана інформаційна система буде частиною внутрішньої системи компанії, для того щоб уникнути вайпу даних були створені міграції, які під час

компіляції створять додаткові таблиці та поля в уже існуючій базі із збереженням вже записаної в неї інформації.

Список міграцій приведений на рисунку 4.3.

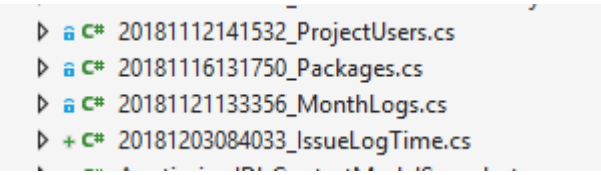

Рисунок 4.3 – Список створених міграцій

Після розгорнтання проекту, система стає доступною для користувачів. Для того щоб увійти в систему користувачу необхідно ввести логін та пароль. Ці дані валідуються та шукається відповідний юзер в базі даних.

Для управління користувачами використовується не контекст даних, а спеціальний клас - UserManager <T> з простору імен Microsoft.AspNetCore.Identity. Основні з його методів і властивостей:

- ChangePasswordAsync (user, old, new): змінює пароль користувача
- CreateAsync (user): створює нового користувача
- DeleteAsync (user): видаляє користувача
- FindByIdAsync (id): шукає користувача по id
- FindByEmailAsync (email): шукає користувача по email
- FindByNameAsync (name): шукає користувача по ніку
- UpdateAsync (user): оновлює користувача
- Users: повертає всіх користувачів
- AddToRoleAsync (user, role): додає для користувача user роль role
- GetRolesAsync (user): повертає список ролей, до яких належить користувач user
- IsInRoleAsync (user, name): повертає true, якщо користувач user належить ролі name

- RemoveFromRoleAsync (user, name): видаляє роль name у користувача user

За допомогою даного класу, .net core дозволяє легко та швидко здійснювати операції з користувачами та уникнути скоплення функцій для контексту даних.

Приклад використання UserManager представлений на рисунку 4.4.

```
var signInResult = await _signInManager.PasswordSignInAsync(user, vm.Password, vm.RememberMe, true);

if (signInResult.IsLockedOut)
    Flasher.Flash("danger",
        "You account is locked for the next 30 minutes because of wrong password attempts");
```

Рисунок 4.4 – Використання UserManager

Авторизація користувача здійснюється через відповідну форму в системі. Приклад авторизації представлений на рисунку 4.5.

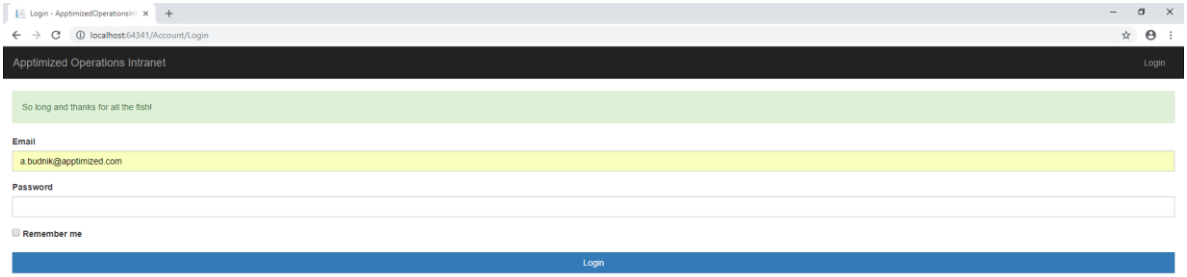

Рисунок 4.5 – Приклад сторінки авторизації в систему

Після авторизацію користувач потрапляє на основну сторінку внутрішньої системи копманії.

Доступ до використання інформаційної системи аналізу та розподілу робочих задач дозволений лише для користувачів групи адміністраторів.

Розмежування по ролям та правам здійснюється через авторизайні атрибути, які дозволяють підвищити якість та читаємість коду.

Приклад використання атибутів представлений на рисунку 4.6.

```
[Authorize(Roles = "Administrators,HR")]
public class ProjectsController : BaseController
{
    public ProjectsController(ApptimizedDbContext context, IFlasher flasher, IMapper mapper) : base(context, flasher, mapper)
    {
    }

    public ActionResult Index()
    {
```

Рисунок 4.6 – Приклад використання авторизаційних атрибутів

Після того, як пройшла перевірка на ролі, користувач або має доступ до певного функціоналу або ні.

Для того, щоб інформаційна система коректно функціонувала необхідно створити залежності користувач-проект, для того щоб визначити команди робітників для кожного з проектів.

Для реалізації даного функціоналу був створений контролер ProjectController, який дозволяє створювати нові записи проектів, визначати їх проектну вагу – для визначення числового представлення складності проекту. А також визначати для кожного проекту команду, яка буде виконувати завдання для даного проекту. Даний список пакувальників, які закріплені за даним проектом будуть використовуватися для коректної роботи розподілу.

Приклад створення проекту приведений на рисунку 4.7.

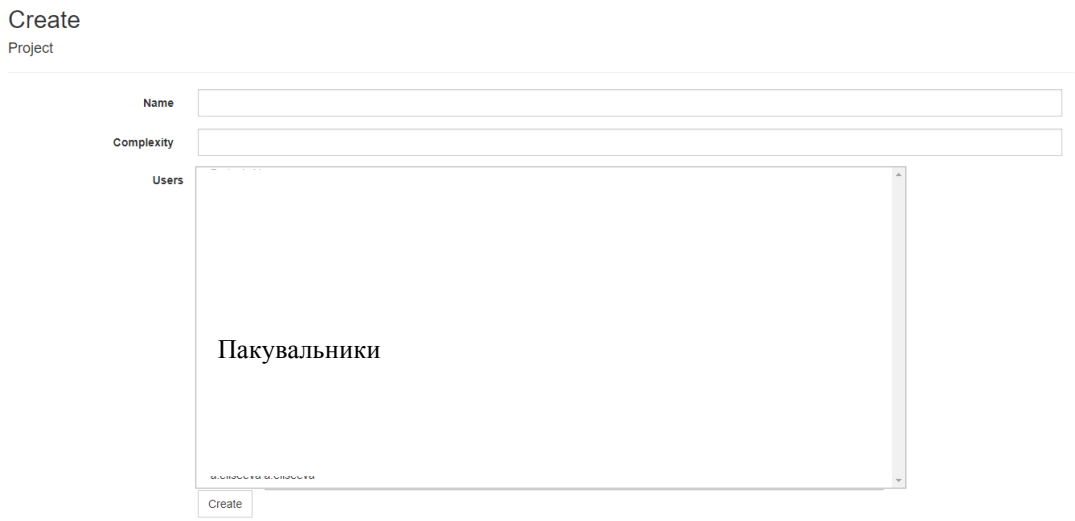

Рисунок 4.8 – Приклад створення нового проекту

Основна сторінка проектів дозволяє переглянути весь список поточних проектів, вагу проекту, а також команду того чи іншого проекту.

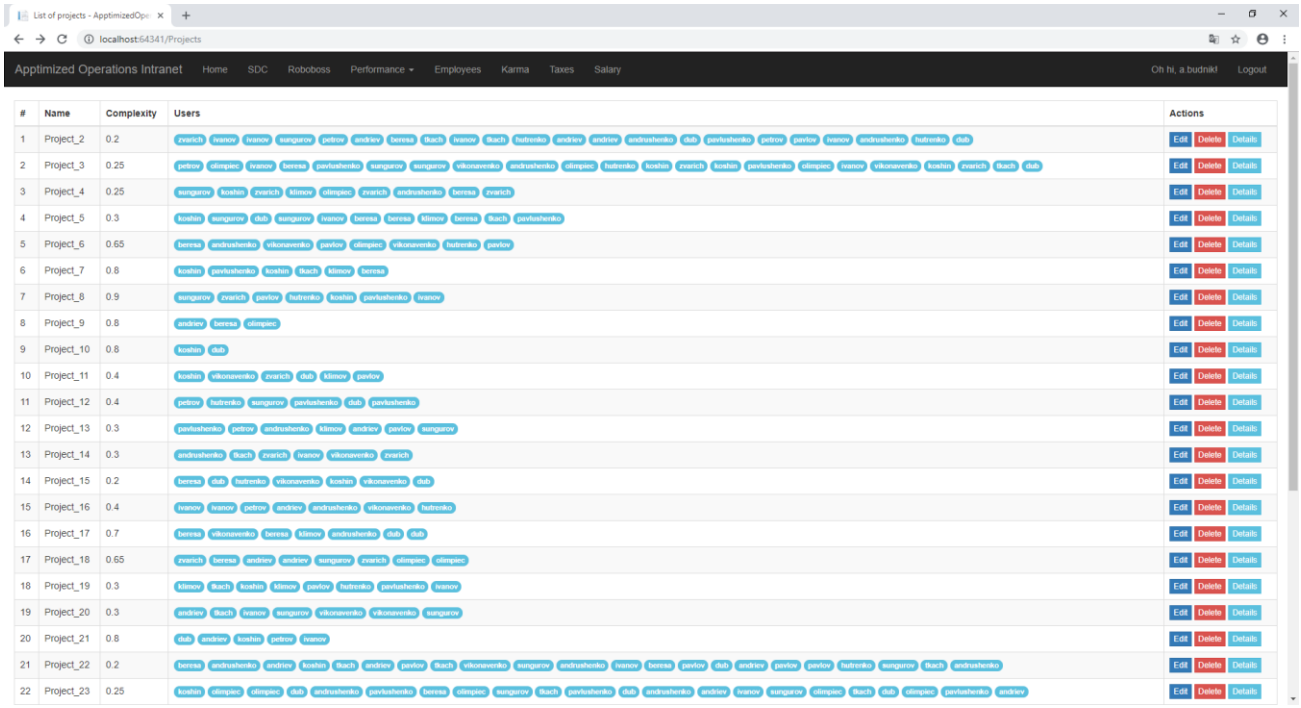

Рисунок 4.8 – Приклад списку проектів

Робота та результат роботи інформаційної системи залежить від багатьох параметрів, які використовуються в розрахунках, і впливають на роботу та розрахунки під час виконання алгоритму програми.

Конфігураційні параметри були розділені на певні групи, які відносяться до відповідної області конфігурації. Так критерії, які були визначені протягом аналізу предметної області та моделювання системи, статичні параметри з тікетів на пакування, та даних отриманих з інших систем були визначені як числові представлення.

Наступні групи були визначені:

- Ваги;
- Статуси;
- Пріорітети;
- Категорії;

Кожна група має список рядків, які отримуються із зовнішніх систем, а також число – яку відображає, так сказати, важливість того чи іншого варіанту групи від 0 до 1.

Приклад конфігураційної сторінки представлений на рисунку 4.9.

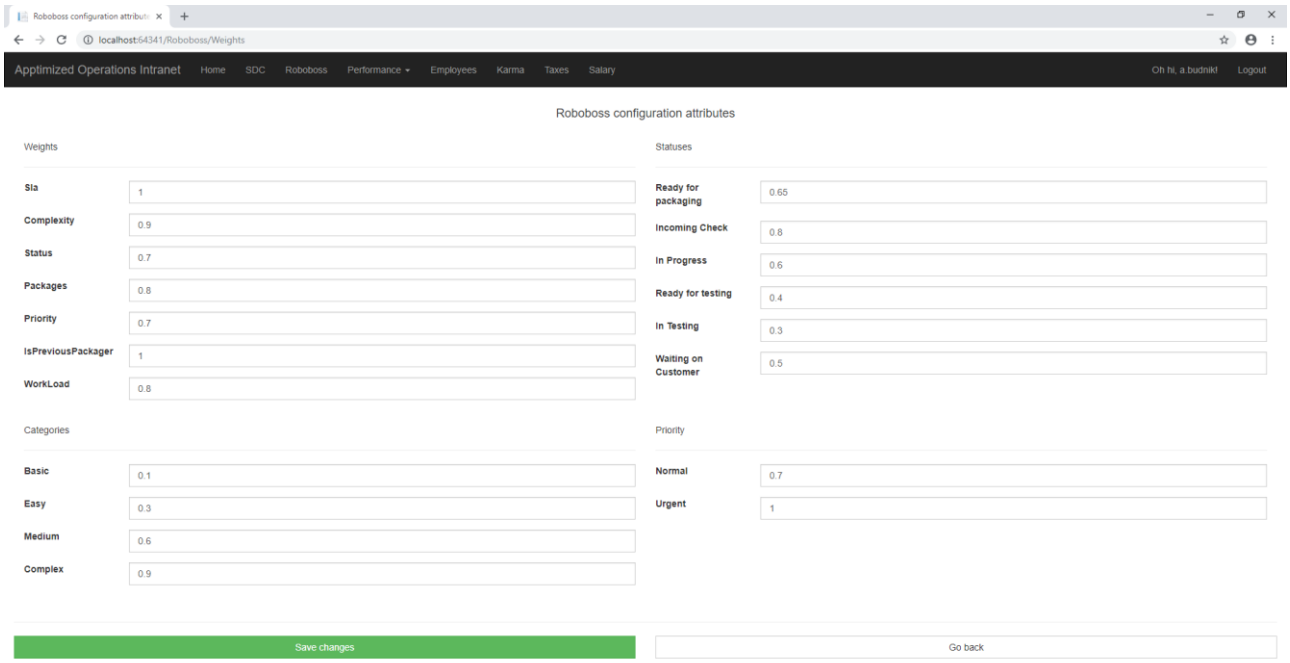

Рисунок 4.9 – Приклад конфіграційної сторінки

Після проведення конфінурації на налаштування всіх необхідних модулів можна працювати з інформаційною системою.

Натиснувши на кнопку «Roboboss» на головному меню системи користувач переходить на сторінку інформаційнох системи, де відразу запускається алгоритм підбору виконавіців під нові задачі.

Після деякого часу очікування, відображається талиця, яка містить список всіх нових замовленнь на пакування ПЗ, а також 2-х виконавців які можуть виконати дану задачу.

Підбір виконавців здійснюється на основі описаних вище критеріїв. Кожна картка виконаця містить всю інформацію по його описаним критеріям, які числові значення були визначені для кожного з критеріїв в момент виконання програми. Приклад головної сторінки ІС представлений на рисунку 4.10.

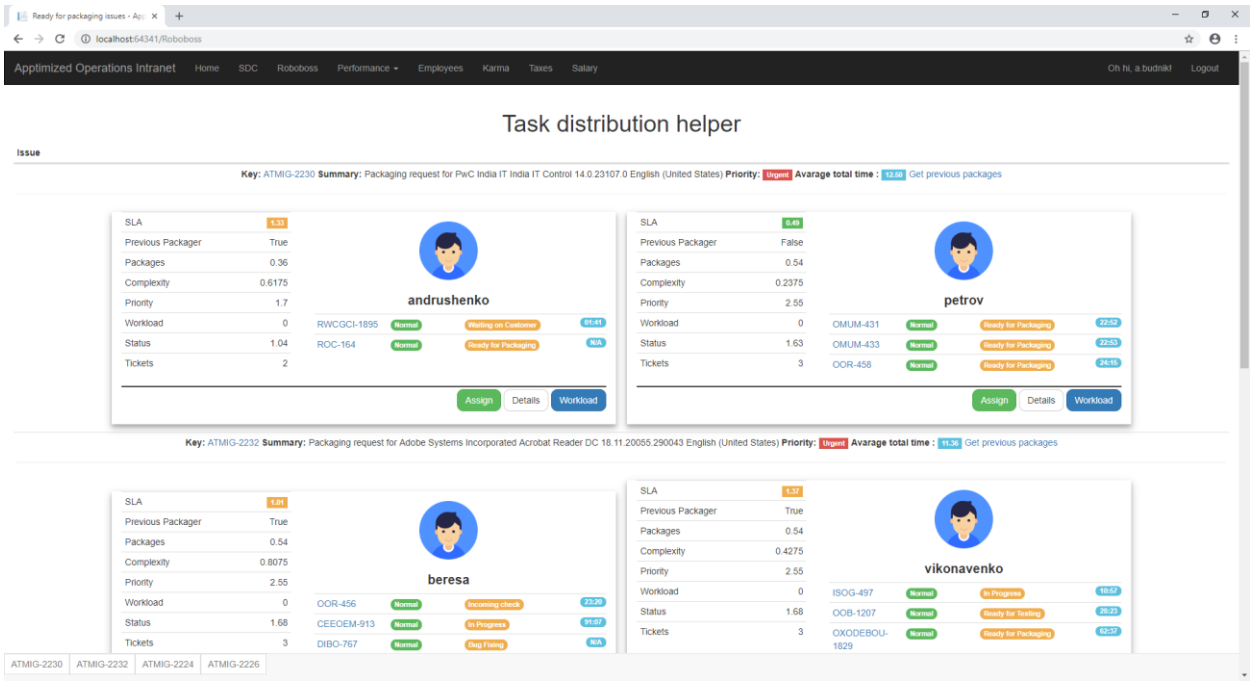

Рисунок 4.10 – Основна сторінка ІС

Кожен тікет має інформацію із зовнішньої системи «Jira», а також дані із інших систем. ІС надає інформацію користувачеві стосовно витраченого часу на попередні замовлення, детально скільки часу було витрачено на пакування ПЗ, скільки часу було витрачено на тестування програмного забезпечення. Ці дані дозволять оцінити керівництву копманії можливі витрати часу на даний пакунок.

Після натискання на посилання «Get previous packages», користувач побаче модальне вікно, яке містить інформацію стосовно попередніх схожих завдань. Приклад представлений на рисунку 4.11.

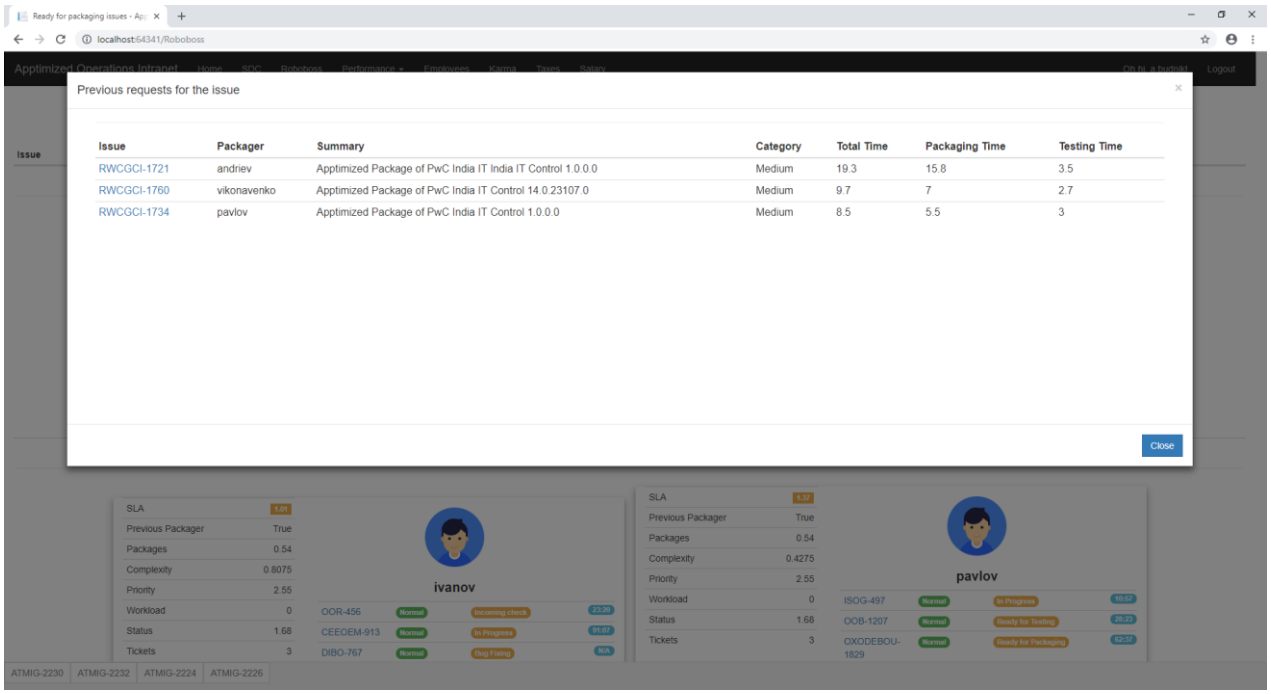

Рисунок 4.11 – Попередні завдання

Дане модальне вікно відображає всі схожі попередні завдання, хто був виконацем задачі, дані про час – залоговано часу на пакування та тестування, а також категорію складності попередніх задач.

Оцінивши можливий час, необхідний на виконання задачі, відповідальна особа починає аналіз наданих інформаційною системою виконавців.

Після аналізу критеріїв, користувач може переглянути детальну інформацію стосовно його поточних задач. Для цього необхідно натиснути кнопку «Details», після чого з’явиться модальне вікно з інформацією по поточним задачам у вигляді таблиці.

Дане вікно відображає наступну інформацію:

- Номер завдання;
- Людей, які так чи інакше працюють над ним;
- Часові параметри (використаний час, час, який залишився, дата дедлайну);
- Пріорітет задачі;
- Поточний статус задачі.

Приклад модального вікна предаставлений на рисунку 4.12.

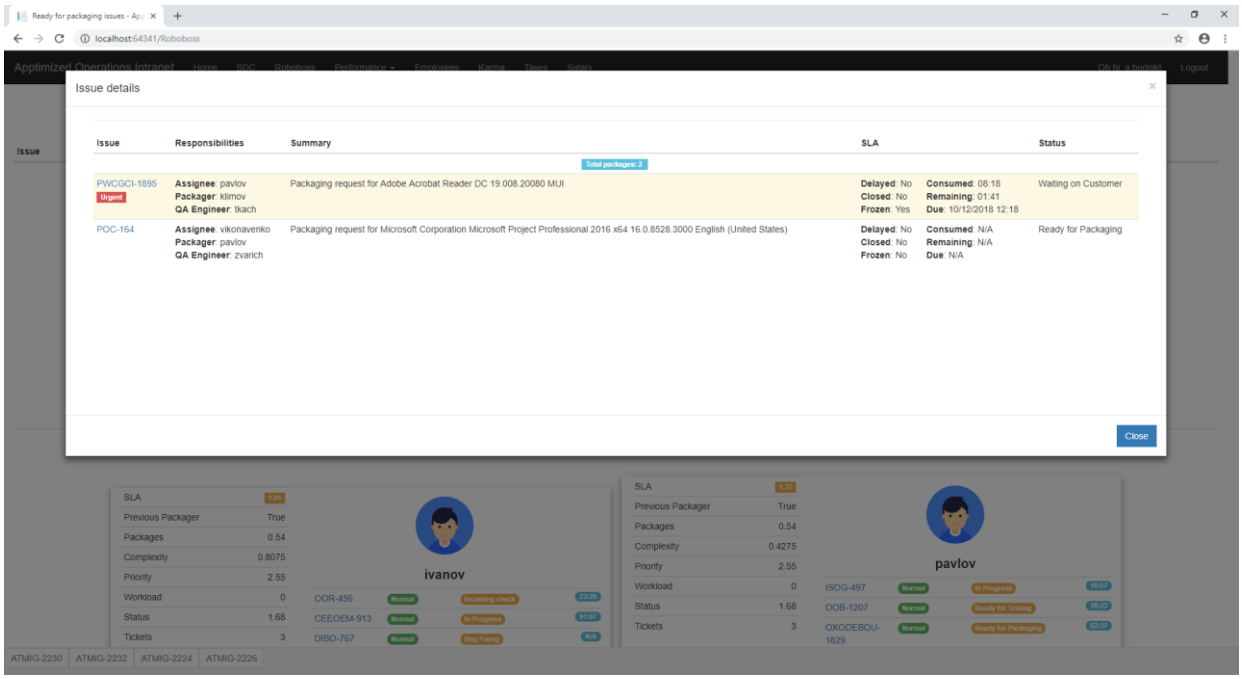

Рисунок 4.12 – Детальна інформація по поточним задачам пакувальника

Після цього, людина, яка розподіляє задачі може переглянути навантаженість працівника відносно команди. Даний інструмент дозволяє побачити, чи був робітник перенавантажений протягом деякого періоду часу відносно навантаженості команди в середньому, чи навпаки – був менше навантажений ніж команда.

Це дозволяє визначити, слід зараз давати тому чи іншому пакувальнику нову задачу, чи можна дати йому відпочити – ц дозволить збалансувати навантаження команди, що в свою чергу краще позначится на продуктивності робітників.

На даний момент інформація по завантаженості береться із системи трекінгу часу за останні 7 тижнів з даного моменту – моменту виконання програми. На основі отриманих даних будується графік, який містить 2 криві. Перша крива – навантаження робітника за цей період часу, інша крива – середні показники навантаження команди за такий самий період часу.

Графічне представлення дозволяє легко побачити точки екстремуми, або точки перетину двох кривих.

Для побудови графіків була використана бібліотека Chart.js.

Chart.js ідеально підходить для таких проектів - коли потрібні пласкі, чисті, елегантні графіки. Ця бібліотека з відкритим вихідним кодом. Вона включає в себе 6 основних типів діаграм (лінія, стовпчата діаграма, радіальна, полярна, кругова діаграма). Для отримання необхідних графіків була використана звичайний лінійний графік.

Бібліотека використовує HTML5 canvas елемент для надання графіків та діаграм.

Приклад графіку представлений на рисунку 4.13.

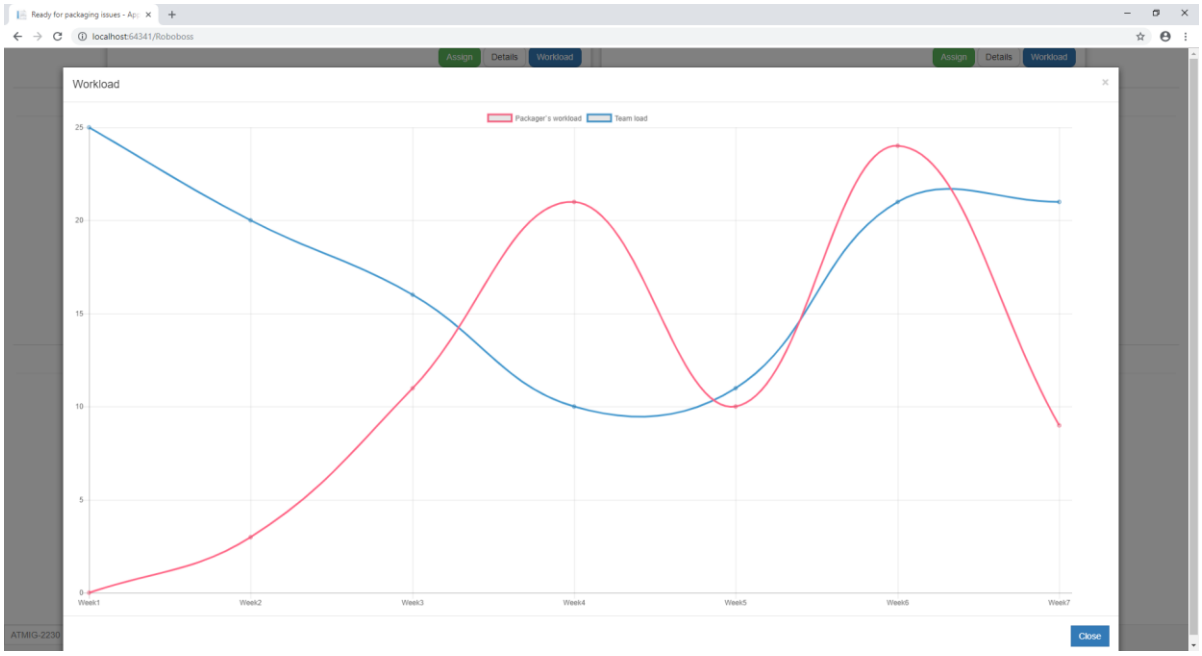

Рисунок 4.13 – Графік навантаження команди/робітника

Зважаючи на дані, які представлені на рис.4.13, можна зробити висновок, що за четвертий тиждень робітник працював більше ніж в середньому команда пакувальників. На сьомому тижні робітник має низькі показники, оскільки у даної людини на наступний тиждень запланована відпустка – таким чином керівництво штучно знижує навантаження на працівника в останні дні перед відпусткою.

4.1 Система контролю версіями

Для взаємодії між працівниками розробки та контролю версіями була використана система від «Microsoft» Dev Azure система, яка має Git-воркфлов, що дозволяє розробнику контролювати процес розробки, та коректно відслідковувати зміни в проекті.

Дана система дозволяє провести інтеграцію, підвищити якість і швидкість розгортання програмного забезпечення. При використанні Azure DevOps або Jenkins для створення додатків в хмарі і їх розгортання в Azure, код автоматично проходить збірку і перевірку, що дозволяє швидше виявляти помилки.

Дозволяє відстежувати працездатність інфраструктури за допомогою служб Azure Log Analytics і Azure Monitor і виконуйте інтеграцію з існуючими панелями моніторингу, наприклад Grafana або Kibana. Служба Azure Application Insights надає практичну інформацію за допомогою засобів для управління продуктивністю додатків і миттєвої аналітики. За допомогою даних засобів можно легко відслідковувати витрати ресурсів сервера (пам’ять та процесор), логувати всі дії в системі тощо.

Приклад потоку даних в Azure DevOps представлений на рисунку 4.14.

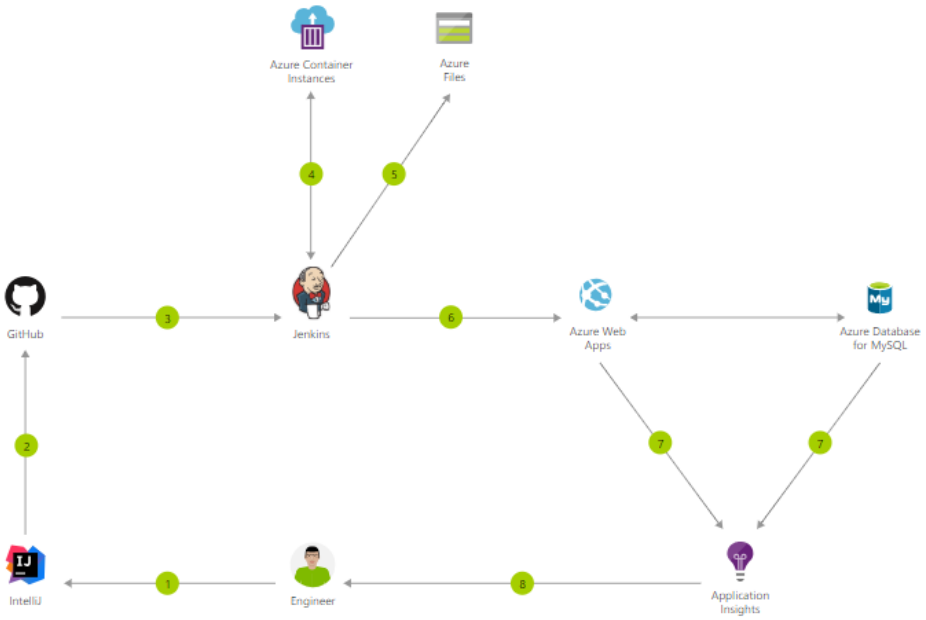

Рисунок 4.14 – Приклад роботи Azure DevOps

ВИСНОВКИ

У ході виконання даної роботи був проведений збір інформації, необхідної для подальшої роботи над реалізацією інформаційної системи, детальний аналіз предметної області пакування ПЗ та розподілу задач у межах IT-компанії «Apptimized Operations» та структури компанії, аналіз можливих аналогів розроблюваної інформаційної системи на основі аналізу джерел інформації, а також були розглянуті та визначені методи дослідження, підходи до проектування системи, технології аналізу та розподілу робочих задач.

Формалізована мета проекту полягає у створенні інформаційної системи аналізу та розподілу робочих задач, що буде оцінена експертами компанії «Apptimized Operations» для пришвидшення робочих процесів та підвищення ефективності планування, і дозволить рівномірно розподілити навантаження команди.

У ході роботи над проектом був змодельований процес розподілу задач в компанії «Apptimized Operations» за нотацією IDEF0.

В ході моделювання були визначені основні проблеми даного процесу та можливі наслідки виникнення таких проблем. Моделювання інформаційної системи допомогло досконально проаналізувати предметну область та область діяльності компанії, що в свою чергу дозволило однозначно точно створити систему відповідну до функціональних вимог замовника.

У ході порівняльного аналізу відомих підходів для вирішення схожих задач були вибрані наступні підходи машинного навчання для вирішення конкретних задач: аналіз задачі – лінійний регресійний аналіз; підбір виконавців – кластерний аналіз k-середніх. В ході тестування системи дані підходи показали очікуваний результат і повністю задовільняють вимоги замовників.

Інформаційна система була розроблена на основі розробленої моделі системи та описаними основними компонентами системи, за допомогою технолоогії .NET Core із використанням методів технології «Entity Framework».

Виконана робота доводить, що автоматизація процесів, навіть таких як розподіл задач між працівниками компанії, значно пришвидшує проходження процесу та полегшує роботу відповідальної особи, яка займається розподілом задач.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Artificial Intelligence [Електронний ресурс] – Режим доступу: https://hub.packtpub.com/welcome-to-machine-learning-using-the-net-framework/

2. Кластерний аналіз [Електронний ресурс] – Режим доступу: https://studfiles.net/preview/6440954/page:20/

3. ASP.NET [Електронний ресурс] – Режим доступу: https://uk.wikipedia.org/wiki/ASP.NET_MVC_Framework

4. Кластерний аналіз [Електронний ресурс] – Режим доступу: https://en.wikipedia.org/wiki/Cluster_analysis

5. Visual Studio и Xamarin [Електронний ресурс] – Режим доступу: http://msdn.microsoft.com/ru-ru/library/mt299001.aspx

6. IronPython [Електронний ресурс] – Режим доступу: https://uk.wikipedia.org/wiki/IronPython

7. Рівень зусиль [Електронний ресурс] – Ржим доступа: https://en.wikipedia.org/wiki/Level_of_effort

8. Visual Studio [Електронний ресурс] – Режим доступу: https://uk.wikipedia.org/wiki/Microsoft_Visual_Studio

9. Анализ и оценка способов формирования целей предприятия. // научное сообщество студентов: междисциплинарные ИССЛЕДОВАНИЯ: сб. ст. по мат. II междунар. студ. науч.-практ. конф. – 2012. – №3.

10. SMART [Електронний ресурс] // Вікіпедія – Режим доступу до ресурсу: https://uk.wikipedia.org/wiki/SMART#cite_note-1.

11. Организационная структура компании [Електронний ресурс]: – Режим доступу до ресурсу: http://project.dovidnyk.info/index.php/programnye-proekty/upravlyeniyeproyektamisprimavera/551-organizacionnaya_struktura_kompanii – Назва з титул. екрану.

12. Навіщо потрібна матриця відповідальності? [Електронний ресурс]: – Режим доступу до ресурсу: http://svitohlyad.com.ua/novyny-i-suspilstvo/navischo-potribna-matrytsya-vidpovidalnosti/ – Назва з титул. екрану.

13. Діаграма Ганта – ваш помічник у плануванні. Що таке діаграма Ганта і як її скласти? [Електронний ресурс]: – Режим доступу до ресурсу: http://faqukr.ru/biznes/103951-diagrama-ganta-vash-pomichnik-u-planuvanni-shho.html – Назва з титул. екрану.

14. Тренды и статистика: Тенденции развития рынка облачных технологий 2015 [Електронний ресурс]. – 2016. – Режим доступу до ресурсу: https://habrahabr.ru/company/it-grad/blog/271635/.

15. Дронов С.В. Многомерный статистический анализ. – Барнаул: Изд-во П.Р. К.П Р.А К.В М.П О.В Алтайского гос. ун-та, 2003. – 213 с.

16. Воронин, А.В. Использование кластерного анализа для выбора локальных стратегий [Текст] / А.В. Воронин // Проблемы и перспективы управления экономикой и маркетингом в организации. – №1. – 2001

17. Рідкокаша А.А., Голдер К.К. Основи систем штучного інтелекту. Навчальний посібник. Черкаси, "ВІДЛУННЯ – ПЛЮС", 2002. – 240 с.

18. Прикладные нечеткие системы / Асаи К., Ватада Д., Иваи С. и др./Под ред. Т. Тэрано, К. Асаи, М. Сугено. – М.: Мир, 1993. – 368 с.

19. Мелихов А.Н., Берштейн Л.С., Коровин С.Я. Ситуационные советующие системы с нечеткой логикой. – М.: Наука, 1990. – 272 с.

20. Discovering Knowledge in Data: An Introduction to Data Mining, Jan 2005 – 190-201 с.

21. Chiang, C.L, (2003) Statistical methods of analysis, World Scientific. ISBN 981-238-310-7 – 274 с.

ДОДАТОК А
ПЛАНУВАННЯ РОБІТ

Планування проекту - це послідовність виконання наступних етапів:

- структуризація проекту (ієрархічна структура робіт – WBS (Work Breakdown Structure), організаційна структура проекту – OBS (Organizational Breakdown Structure), матриця відповідальності – RAM (Responsibility Assignment Matrix);
- розробка плану-графіка реалізації проекту (мережний графік, діаграма Ганта);

Структуризація проекту може виконуватися за будь-якою ознакою. Розрізняють структуризацію проекту за функціональними видами робіт, за виконавцями, ресурсами, відповідальними тощо, а також залежно від складності проекту, його специфіки, вимог керівництва, інших учасників або інвесторів.

Окремо слід виділити проектні структури, які застосовуються в усьому світі під час планування проекту:

- ієрархічна структура робіт – WBS;
- організаційна структура проекту – OBS;
- матриця відповідальності – RAM.

Ідентифікація мети IT-проекту методом SMART. На відміну від традиційних методів до формулювання цілей, у багатьох сферах людського життя та конкретно у проектному менеджменті існує SMART-аналіз якості постановки цілей. Переводячи з англійської мови, «smart» має значення «інтелектуальний» або «розумний». Критерії методу SMART являються більш конкретизованими і, як вважають автори [15], більш вичерпними для формулювання завдань проекту або стратегічних цілей підприємства, ніж традиційні методи, до яких входять такі не конкретизовані критерії до цілі, як «реальна», «перевіряєма», «гнучка», «та, що люди признають в якості власних особистих цілей».

Абревіатура SMART складена із англійських слів «specific», «measurable», «attainable», «relevant», «time-bound», – тобто конкретна, вимірювана, досяжна, актуальна та обмежена в рамках часу. Автори даного методу та спеціалісти, які придержуються його, вважають, що відповідність поставлених задач критеріям даного методу підвищує ймовірність їх виконання у поставлені терміни та досягнення загальної мети. [16]

Базуючись на методі SMART, формалізована мета даного проекту полягає у наступному: створити робоче сховище даних та користувацький веб-інтерфейс до нього, що буде оцінена експертами компанії «Apptimized Operations» для визначення економічної ефективності та якості створеної системи для побудови та перегляду фінансових звітів компанії, і дозволить відображати ефективність роботи спеціалістів компанії та кореляцію між собівартісю і фіксованою ціною компанії у вигляді зручних таблиць та графіків, необхідних керівництву компанії, що підтверджується інтересом компанії до даного проекту. Проект буде виконано вчасно, що підтверджується календарним планом дипломного проекту.

Планування змісту структури робіт IT-проекту. Найкращий шлях для встановлення завдань, необхідних для досягнення мети проекту, – декомпозиція проекту на доступні для контролю та управління частини. Для цього використовується структура робіт (work breakdown structure) – ієрархічна структура декомпозиції робіт проекту, доки весь проект не буде представлений як мережа окремих робіт. Структура робіт допоможе вам визначити завдання та суб-завдання, термін виконання кожної роботи найбільш зручним чином, через побудову графічної схеми. Повний обсяг робіт за проектом розташований на вершині схеми і потім підрозділяється на підсистеми нижчих рівнів. На найнижчому рівні — так званий пакет робіт – сукупність найпростіших операцій. Нижче наведена структура робіт для видання та розповсюдження буклету.

Після того, як ви підготуєте схему структури робіт, на її основі буде легко визначити терміни виконання завдань, необхідні для цього ресурси та кошти.

Структура розподілу (декомпозиція) робіт (WorkBreakdownStructure – WBS) – ієрархічна структура розподілу проекту на підпроекти, пакети робіт різного рівня,

пакети детальних робіт. WBS є засобом для створення системи управління проектом, тому що дозволяє вирішувати проблеми організації робіт, розподілу відповідальності, оцінки варотості, створення системи звітності тощо.

Основні етапи побудови СРР та можливості її використання:

- розподіл та класифікація робіт проекту на підставі заданих критеріїв.

- для кожної роботи визначаються постачальники, виконавці, тривалість робіт, обсяги, бюджет, витрати, обладнання, матеріали тощо;

- побудова матриці відповідальності;

WBS-діаграма проекту представлена на рис. А.1.

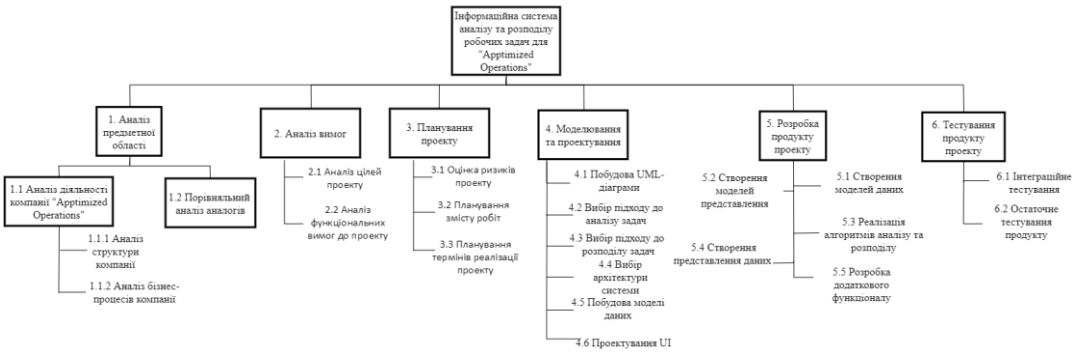

Рисунок А.1 – WBS-діаграма проекту

Наступним кроком розробки структури проекту є визначення організаційної структури (OBS) проекту.

Організаційна структура проекту (OBS) – є графічним відображенням учасників проекту (фізичних та юридичних осіб) та їхніх відповідальних осіб, залучених до реалізації проекту.

Елементами OBS можуть бути:

- окремі виконавці (керівники, фахівці, службовці);
- організації, структурні підрозділи і служби, у яких зайнята та або інша кількість фахівців, що виконують певні функціональні обов'язки;
- інші організації.

Типи зв'язків між елементами OBS:

- вертикальні, або зв'язки підпорядкування;

- горизонтальні, або відносини під час співпраці і узгодження.

Правила побудови OBS:

- на верхньому рівні OBS проекту знаходиться керівник та команда управління проектом;
- на наступному рівні – виконавці: організації, відділи, підрозділи тощо. Виконавцями виступають окремі організаційні структури, які володіють технологією виконання пакетів робіт нижчого рівня у WBS-структурі;
- останнім рівнем OBS-структури є відповідальні особи виконавців. Це не обов’язково повинні бути керівники, а ті співробітники, яким доручено безпосередньо організовувати і відповідати перед виконавцем за виконання конкретного елемента WBS-структури.

В ході планування та підготовки проекту була визначена відповідна команда:

1. Смірнов В.В. – консультант із розробки та проектування системи;
10. Гайдабрус Б.В. – консультант із управління ІТ-проектами;
11. Будник О.С. – керівник проекту, розробник системи;
12. Нестеров А.Д. – бізнес-аналітик проекту, спеціаліст із тестування ПЗ;
13. Антипенко Б.А. – спеціаліст із моделювання моделей даних;

OBS-діаграма проекту представлена на рис. А.2.

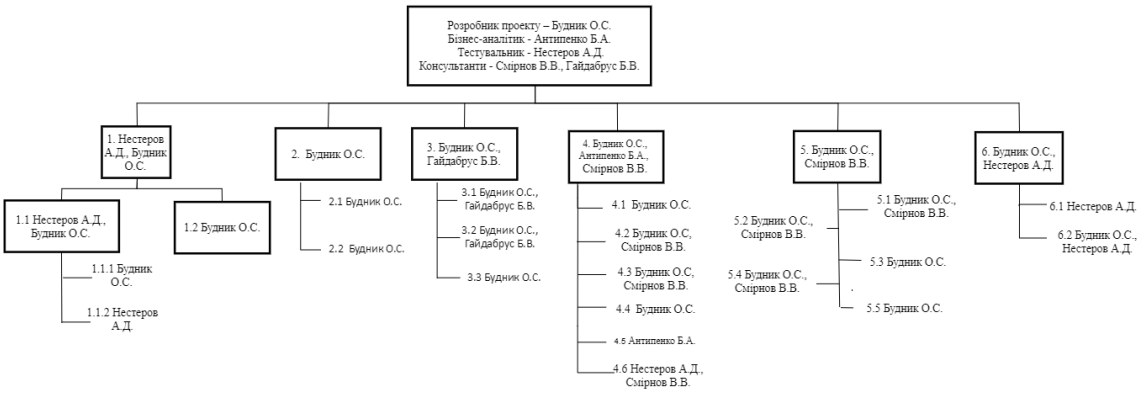

Рисунок А.2 – OBS-діаграма проекту

Суттєвого полегшення управління сприяє такий інструмент, як матриця відповідальності (або лінійний графік). Її будують на підставі попередньо розробленої WBS-структури (ієрархічної основи для виконуваних робіт) і OBS-

структури (організаційної конструкції по виконавцям). Вона застосовується для здійснення керівних впливів шляхом введення принципу відповідальності (Principle Responsibility). Для проекту створюється матриця відповідальності - це відображення спланованих зв`язків (найчастіше - графічне). Її зображують у вигляді таблиці, що базується на базисі співвідношень WBS- і OBS-структур. Тоді, першу представляють в якості вертикальних рядків матриці, а другу - горизонтальних стовпців. Для кожного WBS-елемента визначається співробітник, який буде координувати його виконання. На місці перетину двох компонентів проставляють відповідний знак.

RAM - так коротко позначається матриця відповідальності проекту (Responsibility assignment matrix) - розподіл звітності та обов`язків у вигляді таблиці. Для добре організованою програми повинне дотримуватися правило, щоб виконання будь-якої мети контролював певний керуючий орган.

Функціональні обов`язки всіх осіб, задіяних в заплановані роботи, визначає матриця відповідальності. Вона служить для конкретизації набору завдань, а за їх виконання люди звітують персонально.

Матриця відповідальності проекту представлена у таблиці А.1.

Таблиця А.1 – Матриця відповідальності

		Аналіз предметної області		Аналіз вимог	Планування проекту		Моделювання та проектування			Розробка продукту проекту		Тестування продукту проекту	
		Нестеров А.Д.	Будник О.С.	Будник О.С.	Гайдабрус Б.В.	Будник О.С.	Будник О.С.	Антипенко Б.А.	Смірнов В.В.	Смірнов В.В.	Будник О.С.	Будник О.С.	Нестеров А.Д.
1	1.1	X	X										
	1.1.1		X										
	1.1.2	X											
	1.2	X											
2	2.1			X									
	2.2			X									
3	3.1				K	X							
	3.2				K	X							

	3.3					X							

Продовження таблицы А.1 – Матриця відповідальності

4	4.1						X						
	4.2						X		К				
	4.3						X		К				
	4.4						X						
	4.5							X					
	4.6						X		К				
5	5.1									К	X		
	5.2									К	X		
	5.3										X		
	5.4									К	X		
	5.5										X		
6	6.1												X
	6.2											X	X

Побудова календарного графіку виконання ІТ-проекту. Календарне планування проекту, яке полягає у визначенні календарних дат виконання всіх робіт, ставить за мету координацію діяльності залучених до проекту виконавців для забезпечення його успішного завершення.

Календарний план як перелік тільки планових параметрів проектних робіт втрачає свій сенс без порівняння з фактичними термінами їх виконання, тому

частіше ведуть мову про календарні графіки.

Календарний графік відбиває планові й фактичні дані про початок, кінець і тривалість кожного робочого елементу WBS. У ньому також відмічається можлива гнучкість у даті початку роботи без ускладнення виконання усього проекту (тобто запас часу по некритичних роботах). Для найскладнішого календарного графіка записується чотири версії для дат початку, кінця, тривалості та запасу: рання, пізня, запланована календарна, фактична.

Цілі календарного графіка:

- забезпечити вчасне надходження фінансування;
- координувати надходження ресурсів;
- вчасно забезпечити потрібні ресурси;
- передбачити у різні моменти рівень потрібних фінансових витрат і ресурсів та раціональний розподіл їх між проектами;
- забезпечити вчасне виконання проекту.

Діаграма Ганта є достатньо зручним інструментом для використання. Її будують наступним чином: на горизонталі фіксують календар у тих одиницях часу, які обрані для проекту (години, дні). Ліворуч на вертикалі розташовують найменування всіх робіт. На полі, що утворилось, поставляють у вигляді прямокутників роботи, довжина яких по горизонталі відповідає їхній тривалості.

Фрагмент діаграми Ганта проекту представлений на рис. 2.5. Повна діаграма Ганта представлена у додатку А (рис. А.3-А.6).

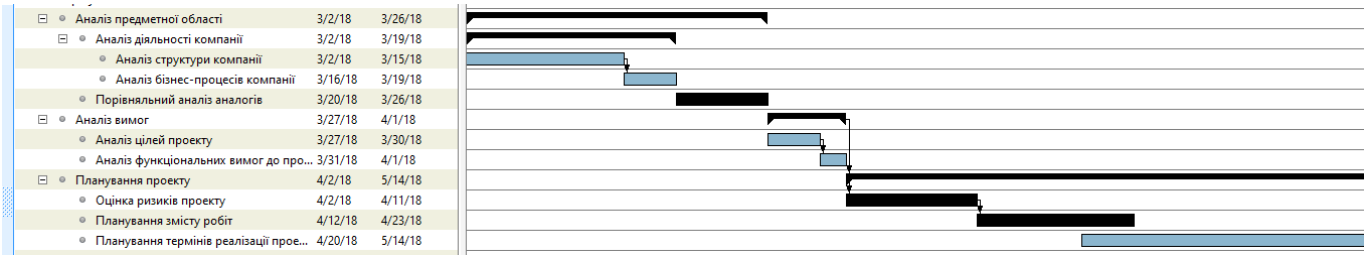

Рисунок А.3 – Перша частина діаграми Ганта

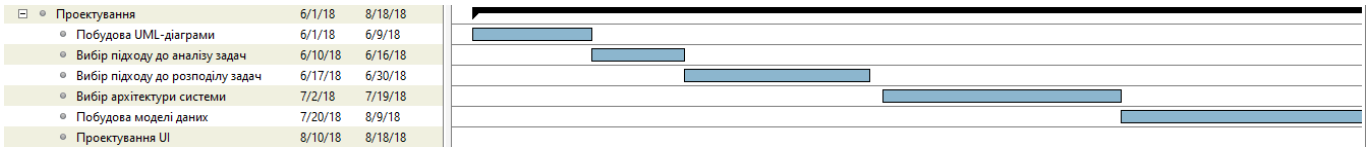

Рисунок А.4 – Друга частина діаграми Ганта

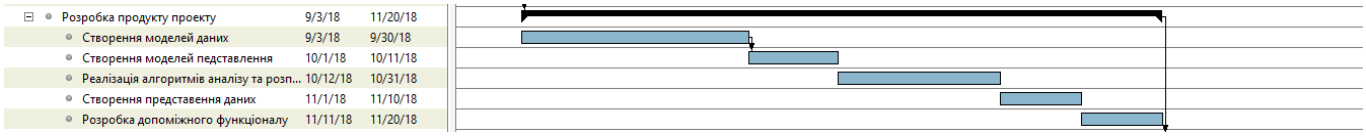

Рисунок А.5 – Третя частина діаграми Ганта

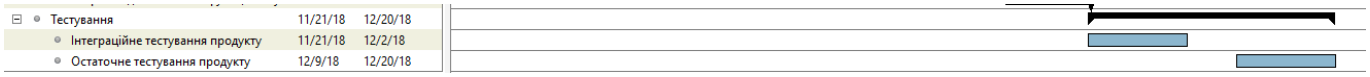

Рисунок А.6 – Четверта частина діаграми Ганта

Планування ризиків проекту. Оскільки розробка програмного забезпечення – це складний процес, до якого залучені люди з різним рівнем підготовки та знань, у ньому використовується безліч технологічних елементів і складних систем, то для кожної команди важливо вміти правильно реагувати на виникаючі і можливі проблеми в майбутньому. Така реакція називається управлінням ризиками.

Проектів без ризиків не буває. Збільшення складності проекту призводить до збільшення числа і масштабів супутніх ризиків. При управлінні ризиками у проекті перш за все слід подумати, як розробити такий план реагування, щоб домогтися зниження ризикованості.

Ризики з'являються з невизначеностей, а невизначеності, як відомо, існують завжди і всюди. Всі ризики можна умовно розділити на дві категорії. До першої категорії відносяться ті ризики, про які ми знаємо, які можливо знайти і оцінити. Для таких ризиків можливо планування. Але бувають ризики, які ми не можемо передбачити, які з'являються несподівано. Такі ризики відносяться до другої категорії. Найчастіше зустрічаються ризики першої категорії, які можна визначити, це відомо з минулого досвіду.

Управління ризиками являє собою велику кількість різних процесів, пов'язаних з пошуком і аналізом ризиків. Ці процеси, спрямовані на те, щоб максимально зменшити негативні наслідки ризиків і максимально збільшити кількість позитивних наслідків.

Основними процедурами даного виду управління є:

- ідентифікація;
- оцінка;
- планування реагування;
- моніторинг і контроль.

Виконаємо якісну і кількісну оцінку ризиків проекту. Про якісній оцінці ризиків визначимо ризику, що потребують швидкого реагування. Така оцінка ризиків визначить ступінь важливості ризику і дозволить вибрати спосіб реагування. Кількісна оцінка ризиків буде виконана для більш повної ідентифікації ризиків та ступеня їхнього впливу на виконання проекту. Кількісна і якісна оцінка ризиків можуть використовуватися окремо або разом, залежно від наявного часу і бюджету, необхідності в кількісній або якісній оцінці ризиків. У таблиці А.1, що знаходиться у додатку А знаходиться класифікація ризиків за показниками ймовірності виникнення ризику та величині втрат.

Далі виконаємо планування реагування на ризики – це розробка методів і технологій зниження негативного впливу ризиків на проект. Визначимо ефективність розробки реагування на проект, визначимо чи будуть наслідки впливу ризику на проект позитивними або негативним.

Таблиця А.2 – Оцінка ймовірності виникнення, величини витрат та індексу ризику

ID	Статус ризику	Опис ризику	Ймовірність виникнення	Вплив ризику	Ранг ризику	План А	Тип стратегії реагування	План Б
RS_1	Відкритий	Непорозуміння між розробником та замовником	Низька	Середній	3	1. Налагодити гарні відносини між розробником та керівником. 2. Дотримуватися ділового етикету спілкування. 3. Створити комфортні умови для співпраці	Попередження	При виявлення непорозуміння потрібно вияснити, що саме стало причиною непорозуміння обсудити її та створити здорову атмосферу в колективі.
RS_2	Відкритий	Поява альтернативного продукту	Низька	Високий	3	1. Провести попереднє дослідження альтернативних продуктів. 2. Вибрати унікальну стратегію створення проекту.	Прийняття	
RS_3	Відкритий	Нечітке завдання на розробку	Середня	Високий	6	1. Ясно і однозначно скласти ТЗ разом із замовником, обговоривши усі види вимог. 2. Скласти глосарій для запобігання розбіжностей у розумінні слів та термінів. 3. Періодичний контроль замовником етапів роботи.	Попередження	При виявленні невідповідностей деяких характеристик продукту заявленим вимовам потрібно уважно та чітко окреслити те, що було виконано невірно (після розмови із замовником) та зробити правки.

Продовження таблиці А.2 – Оцінка ймовірності виникнення, величини витрат та індексу ризику

ID	Статус ризику	Опис ризику	Ймовірність виникнення	Вплив ризику	Ранг ризику	4. План А	Тип стратегії реагування	План Б
RS_4	Відкритий	Низька кваліфікація розробників проекту	Середня	Середній	4	1. Підвищити кваліфікацію персоналу. 2. Переглянути онлайн-ресурси для підвищення рівня знань.	Пом'якшення	Врахувати час на підготовку працівників. Видати літературу, переглянути онлайн-уроки.
RS_5	Відкритий	Неоптимальний розподіл часу	Висока	Високий	9	Провести аналіз актуальності найважливіших процесів та робіт. Звернути особливу увагу на правильність розподілу часу. Правильно визначити пріоритети виконання робіт. Чітко дотримуватися календарного плану.	Пом'якшення	Змінити порядок пріоритетів робіт. Знайти способи оптимізації роботи із вже існуючою розстановкою. Обговорити варіанти внесення правок до термінів реалізації із замовником.
RS_6	Відкритий	Часте внесення змін у ТЗ	Середня	Середній	4	1. Виділити всі необхідні параметри проекту. 2. Чітко описати вимоги до проекту. 3. Обговорити всі технічні засоби виконання проекту та умови реалізації.	Перенос	Узгодити всі положення з замовником, у разі потреби внести необхідні зміни та поправки.
RS_7	Відкритий	Вибір неефективної технології розробки	Середня	Високий	6	1. Проаналізувати методи та засоби, для виконання проекту.	Пом'якшення	Виділити час та ресурсі на пошуки покращення обраної технології.

Продовження таблиці А.2 – Оцінка ймовірності виникнення, величини витрат та індексу ризику

ID	Статус ризику	Опис ризику	Ймовірність виникнення	Вплив ризику	Ранг ризику	План А	Тип стратегії реагування	План Б
RS_8	Відкритий	Не вірна оцінка масштабів проекту	Низька	Середній	2	Провести детальний аналіз проекту. Визначити основні етапу проекту, розподілити час на їх виконання. Проаналізувати масштаби проекту на основі додаткових джерел.	Пом’якшення	Переоцінка масштабів проекту. Перебудова стратегії реалізації проекту.
RS_9	Відкритий	Помилки проектування	Висока	Високий	9	На етапі проектування тісно співпрацювати із замовником та на певних етапах демонструвати поточні результати.	Пом’якшення	Здійснювати проміжний контроль результатів в ході виконання проекту.
RS_10	Відкритий	Збої в роботі програмного забезпечення	Низька	Високий	3	1. Підготувати резерв програмних засобів. 2. Залучити спеціаліста для усунення збоїв.	Попередження	Замінити програмне забезпечення.
RS_11	Відкритий	Відсутність резервних копій даних	Низька	Середній	2	1. Налаштувати автоматичне збереження даних. 2. Зберігати дані на різних носіях інформації.	Попередження	Робити копію даних після кожного виконаного етапу.
RS_12	Відкритий	Реалізація непотрібної функціональності	Низька	Низький	1	Попередити замовника про можливість додаткового функціоналу.	Використання	Обговорити вигоди і збитки від можливих змін проекту.

Продовження таблиці А.2 – Оцінка ймовірності виникнення, величини витрат та індексу ризику

ID	Статус ризику	Опис ризику	Ймовірність виникнення	Вплив ризику	Ранг ризику	План А	Тип стратегії реагування	План Б
RS_13	Відкритий	Невиконання моніторингу проекту	Середня	Низький	2	Здійснювати проміжний контроль результатів в ході виконання проекту. Здійснювати моніторинг проекту працівниками.	Перенос	Здійснювати моніторинг проекту замовником. Надання проміжних результатів виконання проекту після кожного етапу.
RS_15	Відкритий	Виникнення проблем із програмним забезпеченням користувачів	Середня	Високий	6	Розробка проекту з врахуванням вимог до програмного забезпечення користувачів проекту. Модифікація проекту з врахуванням різних версій програмного забезпечення, яке буде застосовуватися.	Прийняття	
RS_15	Відкритий	Зміна вимог замовника в процесі розробки проекту	Низька	Високий	3	Узгодити всі питання на початкових етапах, щоб мінімізувати кількість змін під час розробки.	Пом’якшення	Переоцінка проекту, кожного разу, коли вимоги змінюються.

ДОДАТОК Б
КОД РОЗРОБЛЕНОЇ СИСТЕМИ

Б.1 Класи моделей та контексту системи

ApptimizedDbContext.cs

```
namespace ApptimizedOperationsIntranet.Models.DataModels
{
    public sealed class ApptimizedDbContext : IdentityDbContext<ApptimizedUser>
    {
        public ApptimizedDbContext(
            DbContextOptions<ApptimizedDbContext> options) : base(options)
        {
        }

        public DbSet<Leave> Leaves { get; set; }
        public DbSet<ApptimizedUser> ApptimizedUser { get; set; }
        public DbSet<Project> Projects { get; set; }
        public DbSet<Package> Packages { get; set; }
        public DbSet<ProjectUsers> ProjectUsers { get; set; }
        public DbSet<MonthLog> MonthLogs { get; set; }

        protected override void OnModelCreating(ModelBuilder builder)
        {
            base.OnModelCreating(builder);

            builder.Entity<ApptimizedUser>().HasKey(u => u.Id);

            builder.Entity<ApptimizedUser>()
                .HasMany(x => x.Packages)
                .WithOne(z => z.Packager);

            builder.Entity<Project>()
                .HasMany(x => x.Packages)
                .WithOne(z => z.Project);

            builder.Entity<ProjectUsers>()
                        .HasKey(t => new { t.ProjectId, t.UserId });

            builder.Entity<ProjectUsers>()
                .HasOne(p => p.Project)
                .WithMany(u => u.Users)
                .HasForeignKey(p => p.ProjectId);

            builder.Entity<ProjectUsers>()
                .HasOne(u => u.User)
                .WithMany(p => p.Projects)
                .HasForeignKey(u => u.UserId);

            builder.Entity<ApptimizedUser>()
                .HasMany(c => c.Leaves)
                .WithOne(e => e.User);

            builder.Entity<ApptimizedUser>()
                .HasMany(c => c.MonthLogs)
                .WithOne(e => e.Packager);
        }
    }}
```

Package.cs

```
namespace ApptimizedOperationsIntranet.Models.DataModels
{
    public class Package
    {
        public int Id { get; set; }

        public string Key { get; set; }

        public ApptimizedUser Packager { get; set; }

        public ApptimizedUser QAEngineer { get; set; }

        public Project Project { get; set; }

        public string SoftwareName { get; set; }

        public string SoftwareVendor { get; set; }

        public string Category { get; set; }

        public string Summary { get; set; }

        public double PackagingTime { get; set; }

        public double TestingTime { get; set; }

    }
}
```

Project.cs

```
namespace ApptimizedOperationsIntranet.Models.DataModels
{
    public class Project
    {
        [Key]
        public int Id { get; set; }

        public string Name { get; set; }

        public double Complexity { get; set; }

        public List<ProjectUsers> Users { get; set; } = new List<ProjectUsers>();

        public List<Package> Packages { get; set; } = new List<Package>();

        [NotMapped]
        public ICollection<string> SelectedUsers { get; set; } = new List<string>();

    }
}
```

ProjectUsers.cs

```
namespace ApptimizedOperationsIntranet.Models.DataModels
{
    public class ProjectUsers
    {
        public int ProjectId { get; set; }
        public Project Project { get; set; }

        public string UserId { get; set; }
        public ApptimizedUser User { get; set; }
    }
}
```

Leave.cs

```
namespace ApptimizedOperationsIntranet.Models.DataModels
{
    public enum LeaveType : byte
    {
        Unknown,
        Vacation,
        Sick,
        University,
    }

    public class Leave
    {
        [Key]
        public int Id { get; set; }

        [Required]
        public virtual ApptimizedUser User { get; set; }

        public LeaveType Type { get; set; }

        [Required]
        public DateTime From { get; set; } = DateTime.Now;

        public DateTime? Till { get; set; }

        public DateTime? ClosedByDocumentFrom { get; set; }

        public DateTime? ClosedByDocumentTill { get; set; }
    }
}
```

MonthLog.cs

```
namespace ApptimizedOperationsIntranet.Models.DataModels
{
    public class MonthLog
    {
        [Key]
        public int Id { get; set; }

        public int Month { get; set; }

        public int Year { get; set; }

        public double LoggedTime { get; set; }

        public ApptimizedUser Packager { get; set; }
    }
}
```

Б.2 Класи фонового сервісу та обробки сторонньої інформації

TimeTrackerService.cs

```
namespace ApptimizedOperationsIntranet.Services
{
    public class TimeTrackerService:IDisposable
    {
        private readonly ApptimizedDbContext _context;

        public TimeTrackerService(ApptimizedDbContext context)
        {
            _context = context;
        }

        public List<MonthlyIssueViewModel> GetMonthPackagerReport(string url, string username, int
month)
        {
            var soapEnvelopeXml = CreateMonthPackagerReportSoapEnvelope(username, month);
            var webRequest = CreateWebRequest(url);

            InsertSoapEnvelopeIntoWebRequest(soapEnvelopeXml, webRequest);

            var asyncResult = webRequest.BeginGetResponse(null, null);
            asyncResult.AsyncWaitHandle.WaitOne();

            string soapResult;
            using (var webResponse = webRequest.EndGetResponse(asyncResult))
            {
                using (var rd = new StreamReader(webResponse.GetResponseStream()))
                {
                    soapResult = rd.ReadToEnd();
                }
                Console.Write(soapResult);
            }
            var xDoc = new XmlDocument();
            xDoc.LoadXml(soapResult);
            var jsonText = xDoc.GetElementsByTagName("IssueMonthlyReviewResult")[0].InnerText;
            var result = "{\"list\":[" + jsonText.Substring(jsonText.IndexOf("{\"jira_key\"")) + "}";
            return JsonConvert.DeserializeObject<List<MonthlyIssueViewModel>>(result);
        }

        public MonthlyIssueViewModel InvokeService(string url, string username, int month, int year)
        {
            var soapEnvelopeXml = CreateMonthPackagerReportSoapEnvelope(username, month);

            var webRequest = CreateWebRequest(url);

            InsertSoapEnvelopeIntoWebRequest(soapEnvelopeXml, webRequest);

            var asyncResult = webRequest.BeginGetResponse(null, null);
            asyncResult.AsyncWaitHandle.WaitOne();

            string soapResult;
            using (var webResponse = webRequest.EndGetResponse(asyncResult))
            {
                using (var rd = new StreamReader(webResponse.GetResponseStream()))
                {
                    soapResult = rd.ReadToEnd();
                }
            }

            var xmlResponse = new XmlDocument();
            xmlResponse.LoadXml(soapResult);
```

```
            var totalJsonResult = xmlResponse
                .GetElementsByTagName("IssueMonthlyReviewResult")[0].InnerText;

            if (totalJsonResult.Contains("No report for this month!"))
                return null;

            var result = "[" + totalJsonResult.Substring(totalJsonResult.IndexOf("{\"jira_key\""));

            var responseResult = JsonConvert.DeserializeObject
                <List<MonthlyIssueViewModel>>(result);

            var totalTimeLog = responseResult.FirstOrDefault(l => l.IssueId == "ALL");

            totalTimeLog.PackagerLoggedHours =
totalTimeLog.ConvertLoggedHours(totalTimeLog.PackagerLoggedHoursString);
            totalTimeLog.TotalLoggedHours =
totalTimeLog.ConvertLoggedHours(totalTimeLog.TotalLoggedHoursString);

            var atopsTimeLog = responseResult.FirstOrDefault(l => l.IssueId == "ATOPS");

            if (atopsTimeLog != null)
            {
                atopsTimeLog.PackagerLoggedHours =
totalTimeLog.ConvertLoggedHours(atopsTimeLog.PackagerLoggedHoursString);
                atopsTimeLog.TotalLoggedHours =
totalTimeLog.ConvertLoggedHours(atopsTimeLog.TotalLoggedHoursString);

                totalTimeLog.TotalLoggedHours -= atopsTimeLog.TotalLoggedHours;
                totalTimeLog.PackagerLoggedHours -= atopsTimeLog.PackagerLoggedHours;
            }

            return totalTimeLog;
        }

        public void SaveMonthLogs()
        {
            var users = _context.Users?.Where(a=>a.Department==ApptimizedUserDepartment.Packaging)
                .OrderBy(p => p.UserName).ToList();

            if (users == null) return;

            foreach (var user in users.AsParallel())
            {

                for (var month = 4; month < DateTime.Now.Month; month++)
                {
                    if (packagerMonthTimeLogs == null) continue;

                    var totalMonthTimeLog = packagerMonthTimeLogs;

                var monthLog = new MonthLog()
                {
                    Month = month,
                    Year = 2018,
                    LoggedTime = totalMonthTimeLog.PackagerLoggedHours,
                    Packager = user
                };
                    if(!_context.MonthLogs.Any(x=>x.Packager==user && x.Month==month && x.Year ==
2018))
                        _context.MonthLogs.Add(monthLog);
                }
                _context.SaveChanges();
            }
        }

        public void SaveLoggedTime()
        {
```

```
            foreach (var package in _context.Packages.Where(x=>x.PackagingTime==0).AsParallel())
            {
                var soapEnvelopeXml = GetJiraKeyInfo(package.Key);
                InsertSoapEnvelopeIntoWebRequest(soapEnvelopeXml, webRequest);

                var asyncResult = webRequest.BeginGetResponse(null, null);
                asyncResult.AsyncWaitHandle.WaitOne();

                string soapResult;
                using (var webResponse = webRequest.EndGetResponse(asyncResult))
                {
                    using (var rd = new StreamReader(webResponse.GetResponseStream()))
                    {
                        soapResult = rd.ReadToEnd();
                    }
                }

                var xdoc = new XmlDocument();

                xdoc.LoadXml(soapResult);

                var incomingText = xdoc.GetElementsByTagName("IssueDetailResult")[0].InnerText;

                var index = incomingText.IndexOf("{\"result\"");
                var length = incomingText.IndexOf(",{\"date\":") - 1;

                if (index == -1 || length < 1)
                    continue;

                var trimmedText = incomingText.Substring(index, incomingText.IndexOf(",{\"date\":")-1);

                var result = JsonConvert.DeserializeObject<IssueDetailViewModel>(trimmedText);

                if(result!=null)
                {
                    package.PackagingTime = double.Parse(result.PackagingTime.Substring(0,
result.PackagingTime.IndexOf("hour")).Trim());
                    package.TestingTime = double.Parse(result.TestingTime.Substring(0,
result.TestingTime.IndexOf("hour")).Trim());
                }
            }
            _context.SaveChanges();
        }

        public void CreateLeaveEntries(int month, int year)
        {
            foreach (var user in _context.Users.Where(x => x.Department ==
ApptimizedUserDepartment.Packaging).AsParallel())
            {
                var soapEnvelopeXml = GetUserVocationsSoapEnvelope(user.UserName, month, year);

                InsertSoapEnvelopeIntoWebRequest(soapEnvelopeXml, webRequest);

                var asyncResult = webRequest.BeginGetResponse(null, null);
                asyncResult.AsyncWaitHandle.WaitOne();

                string soapResult;
                using (var webResponse = webRequest.EndGetResponse(asyncResult))
                {
                    using (var rd = new StreamReader(webResponse.GetResponseStream()))
                    {
                        soapResult = rd.ReadToEnd();
                    }
                }

                var xdoc = new XmlDocument();

                xdoc.LoadXml(soapResult);
```

```
                var incomingText = xdoc.GetElementsByTagName("GetEmployeeVacationResult")[0].InnerText;

                var result =
JsonConvert.DeserializeObject<List<TimeTrackerVacationViewModel>>(incomingText).First();

                result.User = user;

                if (!result.result)
                    continue;

                var leaves = result.ToLeaveEntity();

                foreach(var leave in leaves)
                {
                    if (_context.Leaves.Any(x=>x.From == leave.From && x.Till == leave.Till && x.Type
== leave.Type && x.User.Id == leave.User.Id))
                        continue;
                    _context.Add(leave);
                }
            }
            _context.SaveChanges();
        }

        private XmlDocument GetUserVocationsSoapEnvelope(string username, int month, int year)
        {
            var soapEnvelopeDocument = new XmlDocument();
            soapEnvelopeDocument.LoadXml("<?xml version=\"1.0\" encoding=\"utf-8\"?>" +
                                         "<soap:Envelope xmlns:xsi=\"http://www.w3.org/2001/XMLSchema-
instance\" xmlns:xsd=\"http://www.w3.org/2001/XMLSchema\"
xmlns:soap=\"http://schemas.xmlsoap.org/soap/envelope/\">" +
                                         "<soap:Body>" +
                                         "<GetEmployeeVacation xmlns = \"http://api.revacom.com/\">" +
                                         $"<username>{username}</username>" +
                                         $"<month>{month}</month>"+
                                         $"<year>{year}</year>" +
                                         "</GetEmployeeVacation>" +
                                         "</soap:Body>" +
                                         "</soap:Envelope>");

            return soapEnvelopeDocument;
        }

        private XmlDocument GetJiraKeyInfo(string jiraKey)
        {
            var soapEnvelopeDocument = new XmlDocument();
            soapEnvelopeDocument.LoadXml("<?xml version=\"1.0\" encoding=\"utf-8\"?>" +
                                         "<soap:Envelope xmlns:xsi=\"http://www.w3.org/2001/XMLSchema-
instance\" xmlns:xsd=\"http://www.w3.org/2001/XMLSchema\"
xmlns:soap=\"http://schemas.xmlsoap.org/soap/envelope/\">" +
                                         "<soap:Body>" +
                                         "<IssueDetail  xmlns = \"http://api.revacom.com/\">" +
                                         $"<jiraKey>{jiraKey}</jiraKey>" +
                                         "</IssueDetail>" +
                                         "</soap:Body>" +
                                         "</soap:Envelope>");

            return soapEnvelopeDocument;
        }

        private HttpWebRequest CreateWebRequest(string url)
        {
            var webRequest = (HttpWebRequest)WebRequest.Create(url);
            webRequest.ContentType = "text/xml;charset=\"utf-8\"";
            webRequest.Accept = "text/xml";
            webRequest.Method = "POST";
            webRequest.Host = _host;
            return webRequest;
```

```
        }

        private XmlDocument CreateMonthPackagerReportSoapEnvelope(string username, int month)
        {
            var soapEnvelopeDocument = new XmlDocument();
            soapEnvelopeDocument.LoadXml("<?xml version=\"1.0\" encoding=\"utf-8\"?>" +
                                         "<soap:Envelope xmlns:xsi=\"http://www.w3.org/2001/XMLSchema-
instance\" xmlns:xsd=\"http://www.w3.org/2001/XMLSchema\"
xmlns:soap=\"http://schemas.xmlsoap.org/soap/envelope/\">" +
                                         "<soap:Body>" +
                                         "<IssueMonthlyReview xmlns = \"http://api.revacom.com/\">" +
                                         $"<username>{username}</username>" +
                                         $"<month>{month}</month>" +
                                         $"<year>{DateTime.Now.Year}</year>" +
                                         "</IssueMonthlyReview>" +
                                         "</soap:Body>" +
                                         "</soap:Envelope>");

            return soapEnvelopeDocument;
        }

        private void InsertSoapEnvelopeIntoWebRequest(XmlDocument soapEnvelopeXml, HttpWebRequest
webRequest)
        {
            using (var stream = webRequest.GetRequestStream())
            {
                soapEnvelopeXml.Save(stream);
            }
        }
        public void Dispose()
        {
            GC.Collect();
            GC.SuppressFinalize(this);
        }
    }
}
```

ScheduleTask.cs

```
namespace ApptimizedOperationsIntranet.Jobs
{
    public class ScheduleTask : ScheduledProcessor
    {

        public ScheduleTask(IServiceScopeFactory serviceScopeFactory) : base(serviceScopeFactory)
        {
        }

        protected override string Schedule => "*/1 * * * *";

        public override async Task ProcessInScopeAsync(IServiceProvider serviceProvider)
        {
            using (var sdc = serviceProvider.GetService<SdcJiraService>())
            {
                await sdc.GetApprovedPackages();
            }
            using (var service = serviceProvider.GetService<TimeTrackerService>())
            {
                service.CreateLeaveEntries(DateTime.Now.Month, DateTime.Now.Year);
                service.SaveMonthLogs();
                service.SaveLoggedTime();
            }
            await Task.CompletedTask;
        }
    }
}
```

ScopedProcessor.cs

```
namespace ApptimizedOperationsIntranet.Jobs
{
    public abstract class ScopedProcessor : BackgroundService
    {
        private readonly IServiceScopeFactory _serviceScopeFactory;

        public ScopedProcessor(IServiceScopeFactory serviceScopeFactory) : base()
        {
            _serviceScopeFactory = serviceScopeFactory;
        }

        protected override async Task Process()
        {
            using (var scope = _serviceScopeFactory.CreateScope())
            {
                await ProcessInScopeAsync(scope.ServiceProvider);
            }
        }
        public abstract Task ProcessInScopeAsync(IServiceProvider serviceProvider);
    }
}
```

LoadCalculationHelper.cs

```
namespace ApptimizedOperationsIntranet.Helpers
{
    public static class LoadCalculationHelper
    {
        /// <summary>
        /// Static method to get Packager description as Math Model
        /// </summary>
        /// <param name="user">User entity</param>
        /// <param name="model">SdcViewModel view model</param>
        /// <returns>Object of PackagerViewModel type</returns>
        public static PackagerViewModel ToPackagerMathModel(this ApptimizedUser user, SdcViewModel
model, ConfigurationViewModel configuration)
        {
            var requests = model.Issues.Where(x => x.Assignee == user.UserName);

            var packager = new PackagerViewModel()
            {
                Name = user.UserName,
                Sla = 0,
                Complexity = 0,
                DaysToVacation = user.DaysToVacation(),
                IsAvailable = user.IsAvailable(),
                Priority = 0,
                Status = 0,
                Packages = user.CalculatePackagesRatio(model),
                WorkLoad = 0
            };

            foreach (var item in requests)
            {
                packager.Tickets.Add(item.ToPkgComplexity(configuration));
            }

            foreach (var item in packager.Tickets)
            {
                packager.Sla += item.SlaRatio;
                packager.Status += item.StatusRatio;
                packager.Priority += item.PriorityRatio;
                packager.Complexity += item.ComplexityRatio;
            }
            return packager;
```

```
        }

        /// <summary>
        /// Numeric representation of SDC issue
        /// </summary>
        /// <param name="sdcIssue">SdcIssueViewModel</param>
        /// <returns>Returns object of PackageComplexityViewModel that has numeric presentation of
parameters</returns>
        public static PackageComplexityViewModel ToPkgComplexity(this SdcIssueViewModel sdcIssue,
ConfigurationViewModel configuration)
        {
            var package = new PackageComplexityViewModel()
            {
                Key = sdcIssue.Key,
                Assignee = sdcIssue.Assignee,
                Summary = sdcIssue.Title,
                Status = sdcIssue.Status,
                DueDate = sdcIssue.SlaDueDate ?? DateTime.Now,
                ComplexityRatio = sdcIssue.CalculateComplexityRatio(configuration),
                ProjectComplexity = sdcIssue.Project.Complexity,
                SlaRatio = sdcIssue.CalculateSlaRatio(),
                StatusRatio = sdcIssue.CalculateAttribute(configuration, "Status"),
                PriorityRatio = sdcIssue.CalculateAttribute(configuration, "Priority"),
                Issue = sdcIssue
            };
            package.OveralMark = package.CalculateOveralMark();
            return package;
        }

        public static double GetCategoryValue(this Package package, ConfigurationViewModel
configuration)
        {
            var section = configuration.Configuration.FirstOrDefault(x =>
x.Section.Contains("Categories"))?.Items;
            if (section != null && section.TryGetValue(package.Category, out var element))

                return element * package.Project.Complexity;

            return section.Values.Average() * package.Project.Complexity;
        }

        #region helpers
        private static double CalculateAttribute(this SdcIssueViewModel sdcIssue,
ConfigurationViewModel configuration, string attr)
        {
            var section = configuration.Configuration.FirstOrDefault(x =>
x.Section.Contains(attr))?.Items;

            if (section != null && section.TryGetValue(sdcIssue.Status, out var element))
                return element;

            return section.Values.Average();
        }
        private static double CalculateSlaRatio(this SdcIssueViewModel sdcIssue)
        {
            if (sdcIssue.SlaConsumed == null || sdcIssue.SlaRemaining == null)
            {
                return 0.5;
            }

            if (sdcIssue.SlaConsumed.Value.TotalMinutes > sdcIssue.SlaConsumed.Value.TotalMinutes &&
sdcIssue.Status == "Incoming Check")
                return 0.25;

            return (sdcIssue.SlaConsumed.Value.TotalMinutes / (sdcIssue.SlaConsumed.Value.TotalMinutes
+ sdcIssue.SlaRemaining.Value.TotalMinutes));
        }
```

```
        private static double CalculatePackagesRatio(this ApptimizedUser user, SdcViewModel model)
        {
            var usersTickets = model.Issues.Where(x => x.Assignee == user.UserName);
            var count = (double)model.Issues.Count();
            return usersTickets.Count() / (count / model.Packagers.Count);
        }

        private static double CalculateStatusRatio(this SdcIssueViewModel sdcIssue,
ConfigurationViewModel configuration)
        {
            var c = configuration.Configuration.FirstOrDefault(x => x.Section.Contains("Status")).Items
as Dictionary<string, double>;

            if (c.TryGetValue(sdcIssue.Status, out var element))
                return element;

            return c.Values.Average();
        }

        private static double CalculatePriorityRatio(this SdcIssueViewModel sdcIssue,
ConfigurationViewModel configuration)
        {
            var c = configuration.Configuration.FirstOrDefault(x =>
x.Section.Contains("Priority")).Items as Dictionary<string, double>;

            if (c.TryGetValue(sdcIssue.Priority, out var element))
                return element;

            return c.Values.Average();
        }

        private static double CalculateComplexityRatio(this SdcIssueViewModel sdcIssue,
ConfigurationViewModel configuration)
        {
            var section = configuration.Configuration.FirstOrDefault(x =>
x.Section.Contains("Categories")).Items;

            if(sdcIssue.Complexity == null)
                return section.Values.Average() * sdcIssue.Project.Complexity;

            if (section.TryGetValue(sdcIssue.Complexity, out var element))
                return element * sdcIssue.Project.Complexity;

            return section.Values.Average() * sdcIssue.Project.Complexity;
        }

        private static double CalculateOveralMark(this PackageComplexityViewModel viewModel)
        {
            return viewModel.SlaRatio * viewModel.StatusRatio * viewModel.PriorityRatio *
viewModel.ComplexityRatio;
        }

    }
    #endregion
}
```

RegressionAnalisys.cs

```
namespace ApptimizedOperationsIntranet.Helpers
{
    public static class RegressionAnalisys
    {
        public static double GetPredictedValue(this List<Package> packages,
            ConfigurationViewModel configurationViewModel)
        {
            double sumCategories = 0;
```

```
            double sumTotalTime = 0;
            double sumCategoriesSq = 0;
            double sumTotalTimeSq = 0;
            double sumCodeviates = 0;

            foreach (var package in packages)
            {
                var x = package.GetCategoryValue(configurationViewModel);
                var y = package.PackagingTime + package.TestingTime;
                sumCategories += x;
                sumTotalTime += y;
                sumCodeviates += x * y;
                sumCategoriesSq += x * x;
                sumTotalTimeSq += y * y;
            }

            var count = packages.Count;
            var ssX = sumCategoriesSq - ((sumCategories * sumCategories) / count);
            var ssY = sumTotalTimeSq - ((sumTotalTime * sumTotalTime) / count);

            var rNumerator = (count * sumCodeviates) - (sumCategories * sumTotalTime);
            var rDenom = (count * sumCategoriesSq - (sumCategories * sumCategories))
                         * (count * sumTotalTimeSq - (sumTotalTime * sumTotalTime));
            var sCo = sumCodeviates - ((sumCategories * sumTotalTime) / count);

            var meanX = sumCategories / count;
            var meanY = sumTotalTime / count;
            var dblR = rNumerator / Math.Sqrt(rDenom);

            var yIntercept = meanY - ((sCo / ssX) * meanX);
            var slope = sCo / ssX;

            return (slope * sumCategories/count) + yIntercept;
        }

        public static string GetPredictedCategory(this List<Package> packages,
            ConfigurationViewModel configurationViewModel)
        {
            var projectComplexity = packages.First().Project.Complexity;
            double sumCategories = 0;
            foreach (var package in packages)
            {
                var x = package.GetCategoryValue(configurationViewModel);
                sumCategories += x;
            }

            var value = sumCategories / packages.Count();
            var      section      =      configurationViewModel.Configuration.FirstOrDefault(x      =>
x.Section.Contains("Categories"))?.Items;
            var category = string.Empty;
            var min = 100d;
            foreach (var item in section)
            {
                var abs = Math.Abs(item.Value * projectComplexity - value);
                if (!(abs < min)) continue;
                min = abs;
                category = item.Key;
            }

            return category;
        }
    }
}
```

В.3 Класи представлень, моделей представлень та контролеру

AssignViewModel.cs

```
namespace ApptimizedOperationsIntranet.Models.ViewModels.Roboboss
{
    public class AssignViewModel
    {
        public SdcIssueViewModel SdcIssue { get; set; }
        public List<PackagerViewModel> Packagers { get; set; }

        public override string ToString()
        {
            return $"{SdcIssue.Key} {SdcIssue.Title}";
        }
    }
}
```

ConfigurationViewModel.cs

```
namespace ApptimizedOperationsIntranet.Models.ViewModels.Roboboss
{
    [Serializable]
    public class ConfigurationViewModel
    {
        public IEnumerable<ConfigurationItemViewModel> Configuration { get; set; }
    }

    [Serializable]
    public class ConfigurationItemViewModel
    {
        public string Section { get; set; }
        public Dictionary<string,double> Items { get; set; }
    }
}
```

PackageComplexityViewModel.cs

```
namespace ApptimizedOperationsIntranet.Models.ViewModels.Roboboss
{
    public class PackageComplexityViewModel
    {
        public string Key { get; set; }

        public string Summary { get; set; }

        public string Status { get; set; }

        public DateTime DueDate { get; set; }

        public string Assignee { get; set; }

        public double SlaRatio { get; set; }

        public double ComplexityRatio { get; set; }

        public double ProjectComplexity { get; set; }

        public double StatusRatio { get; set; }

        public double PriorityRatio { get; set; }
```

```
        public double OveralMark { get; set; }

        public override string ToString()
        {
            return $"{Key} {Summary}";
        }

        public SdcIssueViewModel Issue { get; set; }
    }
}
```

PackagerViewModel.cs

```
namespace ApptimizedOperationsIntranet.Models.ViewModels.Roboboss
{
    public class PackagerViewModel
    {
        /// <summary>
        /// Name of packager
        /// </summary>
        public string Name { get; set; }

        /// <summary>
        /// Sum of SLA coefs for each package
        /// </summary>
        [KMeansValue]
        public double Sla { get; set; }

        /// <summary>
        /// Sum of Complexity coefs for each package
        /// </summary>
        [KMeansValue]
        public double Complexity { get; set; }

        /// <summary>
        /// Sum of Status coefs for each package
        /// </summary>
        [KMeansValue]
        public double Status { get; set; }

        /// <summary>
        /// The ratio of packager`s packages to avarage of all team packages
        /// </summary>
        [KMeansValue]
        public double Packages { get; set; }

        /// <summary>
        /// Sum of Priority coefs for each package
        /// </summary>
        [KMeansValue]
        public double Priority { get; set; }

        /// <summary>
        /// The ratio of days to vacation
        /// </summary>
        public int DaysToVacation { get; set; }

        /// <summary>
        /// Is in vacation now
        /// </summary>
        public bool IsAvailable { get; set; }

        /// <summary>
        /// Is packager - previous packager of request
        /// </summary>
        [KMeansValue]
        public double IsPreviousPackager { get; set; }
```

```
        /// <summary>
        /// The ratio of packager workload against team avarage workload
        /// </summary>
        [KMeansValue]
        public double WorkLoad { get; set; }

        /// <summary>
        /// Collection of current packager`s requests
        /// </summary>
        public ICollection<PackageComplexityViewModel> Tickets { get; set; } = new
List<PackageComplexityViewModel>();

        public string WorkloadArray { get; set; }

        public override string ToString()
        {
            return $"{Name}";
        }
    }
}
```

RobobossController.cs

```
namespace ApptimizedOperationsIntranet.Controllers
{
    public class RobobossController : BaseController
    {
        private readonly SdcJiraService _jiraService;
        private readonly ClasteringService _clustering;
        private List<Package> _packages;
        private PackagerViewModel _common;
        private ConfigurationViewModel _configuration;

        public RobobossController(ApptimizedDbContext context, IFlasher flasher, IMapper mapper,
            SdcJiraService jiraService, ClasteringService clusteringService) : base(context, flasher,
mapper)
        {
            _jiraService = jiraService;
            _clustering = clusteringService;
            _packages = Context.Packages.ToList();
            _common = JsonConvert.DeserializeObject<PackagerViewModel>(
                                System.IO.File.ReadAllText("Configuration/Packager.json"));

            _configuration = JsonConvert.DeserializeObject<ConfigurationViewModel>(

System.IO.File.ReadAllText("Configuration/RobobossConfiguration.json"));
        }

        public IActionResult Index()
        {
            return View();
        }

        public async Task<IActionResult> Details(string username)
        {
            var issues = await _jiraService.GetCachedSdcViewModel();

            return PartialView("_Details", issues.Issues.Where(a => a.Assignee == username).ToList());
        }
        [HttpGet]
        public IActionResult Configuration()
        {
            return View(_common);
        }

        [HttpPost]
        public IActionResult Configuration(PackagerViewModel model)
        {
```

```
            if (!ModelState.IsValid || model.Equals(_common))
                return View(model);

            model.Name = "SuperHero";
            model.IsAvailable = true;
            _common = model;

            var output = JsonConvert.SerializeObject(model, Formatting.Indented);
            System.IO.File.WriteAllText("Configuration/Packager.json", output);

            return RedirectToAction("Configuration");

        }

        [HttpGet]
        public IActionResult Weights()
        {
            return View(_configuration.Configuration);
        }

        [HttpGet]
        public IActionResult PreviousPackages(string key)
        {
            var keys = key.Split(',');
            var packages = new List<Package>();
            foreach(var item in keys)
            {
                if (_packages.Any(x => x.Key == item))
                    packages.Add(_packages.First(x => x.Key==item));
            }
            return PartialView("_PreviousPackages", packages);
        }

        [HttpPost]
        public IActionResult Weights(ConfigurationItemViewModel[] result)
        {
            if (!ModelState.IsValid)
                return View(model);

            var output = JsonConvert.SerializeObject(model, Formatting.Indented);
            System.IO.File.WriteAllText("Configuration/RobobossConfiguration.json", output);

            return RedirectToAction("Weights");
        }

        [HttpPost]
        public async Task<IActionResult> Result()
        {
            var currentIssues = await _jiraService.GetSdcATMIG();

            var allIssues = await _jiraService.GetCachedSdcViewModel();

            var issues = currentIssues.Issues.Where(a => a.Status == "Ready for Packaging" &&
a.CreatedDate.Value.Date > new DateTime(DateTime.Now.Year, DateTime.Now.Month, 1).Date).ToList();

            var users = Context.Users.Where(a => a.Department ==
ApptimizedUserDepartment.Packaging).ToList();

            var model = new List<AssignViewModel>();

            var elements = new List<PackagerViewModel>
            {
                _common
            };
            foreach (var itemRequest in issues)
            {
                if (itemRequest.Project.Name.Equals("ATUM"))
                    continue;
```

```
                itemRequest.PreviousPackages = PreviousPackages(_packages, itemRequest);
                if (itemRequest.PreviousPackages.Count > 0)
                {
                    itemRequest.PredictedComplexity =
                        itemRequest.PreviousPackages.GetPredictedCategory(_configuration);

                    itemRequest.PredicetedTime =
itemRequest.PreviousPackages.GetPredictedValue(_configuration);
                }

                var modelItem = new AssignViewModel()
                {
                    SdcIssue = itemRequest
                };
                var projectUsers = GetProjectUsers(itemRequest.Project);

                var usersList = GetUsers(projectUsers, allIssues);

                var userModels = usersList.Where(a => projectUsers.Select(x => x.UserName).ToList()
                                          .Contains(a.Name)).ToArray();

                foreach (var item in userModels)
                {
                    item.IsPreviousPackager = PreviousPackager(item, _packages.Where(x =>
x.Packager.UserName == item.Name).ToList(),
                                               itemRequest);
                }

                if (userModels?.Count() > 2)
                {
                   var clusters = _clustering.Cluster(userModels.ToArray(), 4, 200,
                       _configuration.Configuration.FirstOrDefault(x=>x.Section== "Weights")?.Items);

                    modelItem.Packagers = GetNearestCluster(clusters, elements).OrderBy(x =>
x.Sla).Take(2).ToList();
                }
                else
                {
                    modelItem.Packagers = userModels.ToList();
                }
                var random = new Random();

                foreach (var item in modelItem.Packagers)
                {
                    item.WorkloadArray = $"[{random.Next(0, 25)}, {random.Next(0, 25)}, {random.Next(0,
25)}, " +
                                         $"{random.Next(0, 25)}, {random.Next(0, 25)}, " +
                                         $"{random.Next(0, 25)}, {random.Next(0, 25)}]";
                }
                model.Add(modelItem);
            }
            return View(model);
        }

        protected override void Dispose(bool disposing)
        {
            if (disposing)
            {
                Context.Dispose();
            }
            base.Dispose(disposing);
        }

        #region Helpers
        private SdcIssueViewModel[] GetUsersRequests(SdcViewModel model, string username)
        {
            return model.Issues.Where(x => x.Assignee == username).ToArray();
```

```
        }

        private List<ApptimizedUser> GetProjectUsers(Project project)
        {
            return Context.Projects
                        .Include(x => x.Users)
                        .ThenInclude(x => x.User)
                        .ThenInclude(x => x.Projects).First(a => a.Equals(project))
                        .Users.Select(a => a.User).ToList();
        }

        private ApptimizedUser GetUserWithLeaves(ApptimizedUser user)
        {
            return Context.Users
                        .Include(x => x.Leaves)
                        .ThenInclude(e => e.User)
                        .First(a => a.UserName == user.UserName);
        }

        private PackagerViewModel[] GetUsers(IEnumerable<ApptimizedUser> users, SdcViewModel
currentIssues)
        {
            return users.Select(GetUserWithLeaves).Select(userLeaves => userLeaves
                                    .ToPackagerMathModel(currentIssues, _configuration))
                                    .Where(userModel => userModel.DaysToVacation != 0).ToArray();
        }

        private IEnumerable<PackagerViewModel> GetNearestCluster(KMeansResults<PackagerViewModel>
meansResults,
                                                                 List<PackagerViewModel> elements)
        {
            var cluster = new PackagerViewModel[] { };
            var min = 1000d;
            for (var i = 0; i < meansResults.Means.Length; i++)
            {
                var elementsArray = _clustering.ConvertEntities(elements.ToArray(),
                    _configuration.Configuration.FirstOrDefault(x => x.Section ==
"Weights")?.Items)[0];

                var res = _clustering.CalculateDistance(elementsArray, meansResults.Means[i]);
                if (!meansResults.Clusters[i].Any())
                    continue;
                if (!(res < min))
                    continue;

                min = res;
                cluster = meansResults.Clusters[i];
            }
            return cluster.ToList();
        }

        private Dictionary<PackagerViewModel, double> GetNearestCluster(List<PackagerViewModel> all,
                                                        List<PackagerViewModel> etalon)
        {
            var cluster = new Dictionary<PackagerViewModel, double>();
            var min = 1000d;
            var etalonArray = _clustering.ConvertEntities(etalon.ToArray(),
                    _configuration.Configuration.FirstOrDefault(x => x.Section ==
"Weights")?.Items)[0];
            foreach (var item in all)
            {
                var aa = new List<PackagerViewModel>();
                aa.Add(item);
                var elemArray = _clustering.ConvertEntities(aa.ToArray(),
_configuration.Configuration.FirstOrDefault(x => x.Section == "Weights")?.Items)[0];
                var res = _clustering.CalculateDistance(etalonArray, elemArray);

                cluster.Add(item, res);
```

```
            }
            return cluster;
        }

        public double PreviousPackager(PackagerViewModel user, ICollection<Package> packages,
SdcIssueViewModel sdcIssue)
        {
            var jaro = new F23.StringSimilarity.JaroWinkler();
            var max = 0d;
            foreach (var item in packages)
            {
                if (string.IsNullOrEmpty(sdcIssue.ApplicationName) ||
string.IsNullOrEmpty(item.SoftwareName))
                    continue;

                var result = jaro.Similarity(sdcIssue.ApplicationName, item.SoftwareName);

                if (!(result > max)) continue;
                max = result;
            }
            return max > 0.9 ? 2 : 0;
        }

        public List<Package> PreviousPackages(ICollection<Package> packages, SdcIssueViewModel
sdcIssue)
        {
            var resultList = new List<Package>();
            var jaro = new F23.StringSimilarity.JaroWinkler();
            foreach (var item in packages)
            {
                if (string.IsNullOrEmpty(sdcIssue.ApplicationName) ||
string.IsNullOrEmpty(item.SoftwareName))
                    continue;

                var result = jaro.Similarity(sdcIssue.ApplicationName, item.SoftwareName);

                if (result > 0.87)
                    resultList.Add(item);
            }
            return resultList;
        }
        #endregion
    }
}
```

Result.cshtml

```
@using ApptimizedOperationsIntranet.Models.ViewModels.Roboboss
@using System.Globalization
@using ApptimizedOperationsIntranet.Helpers
@model List<AssignViewModel>

@{
    Layout = null;
}
@functions
{
    public List<PackageComplexityViewModel> List = new List<PackageComplexityViewModel>();

    private static string GetTimeSpanInSdc(TimeSpan? ts)
    {
        if (ts == null)
            return "N/A";

        var result = string.Empty;

        if (ts.Value.TotalSeconds < 0)
            result += "-";
```

```
        result += string.Format(
            "{0:00}:{1:00}",       Math.Abs(ts.Value.Days       *       24       +       ts.Value.Hours),
Math.Abs(ts.Value.Minutes));

        return result;
    }

    public static string GenerateName()
    {
        Random r = new Random();
        string[]  consonants  =  {  "ivanov",  "petrov",  "pavlov",  "vikonavenko",  "andriev",  "dub",
"beresa",   "klimov",   "pavlushenko",   "tkach",   "olimpiec",   "sungurov",   "hutrenko",   "koshin",
"andrushenko", "zvarich" };
        return consonants[r.Next(0, consonants.Length)];

    }
}
```

```
<h1 class="text-center">Task distribution helper</h1>
<div class="table-container" id="container" style="display:block; min-width:400px; margin-bottom:80px
!important" data-spy="scroll" data-target="#navbar" data-offset="50">
    <table class="table table-condensed">
        <thead>
            <tr>
                <th>Issue</th>
            </tr>
        </thead>
        <tbody>
            @foreach (var item in Model)
            {
                if (item.SdcIssue.Key.Contains("ATUM"))
                {
                    continue;
                }
                <tr>
                    <td class="center-block" align="center" id="@item.SdcIssue.Key">
                        <p class="title">
                            <strong>Key:                                                                </strong><a
href="https://sdc.revacom.com/browse/@item.SdcIssue.Key" target="_blank">@item.SdcIssue.Key</a>
                            <strong> Summary: </strong>@item.SdcIssue.Title
                            <strong> Priority: </strong>
                            @if (item.SdcIssue.Priority == "Normal")
                            {
                                <span class="label label-success">@item.SdcIssue.Priority</span>
                            }
                            else
                            {
                                <span class="label label-danger">@item.SdcIssue.Priority</span>
                            }

                            @if (item.SdcIssue.PreviousPackages.Count > 0)
                            {
                                <strong>Predicted category :</strong>
                                <span                        class="label                        label-
info">@item.SdcIssue.PredictedComplexity</span>

                                <strong>Predicted time :</strong>
                                <span                        class="label                        label-
info">@item.SdcIssue.PredicetedTime.ToString("0.00", CultureInfo.InvariantCulture)</span>
                                <a      data-toggle="modal"      data-target="#packagesModal"      data-
content="@string.Join(',',item.SdcIssue.PreviousPackages.Select(x=>x.Key).ToArray())">Get      previous
packages</a>
                            }


                        </p>
                        <hr />
```

```
                    <div class="row center-block">
                        @foreach (var packager in item.Packagers)
                        {
                            <div class="col-md-5 packager">
                                <div class="packager-main">
                                    <div style="width:35%; float:left; text-align:left">
                                        <table class="table table-condensed">
                                            <tr>
                                                <td>SLA</td>
                                                <td style="text-align:right;">
                                                    @if (packager.Sla < 0.8)
                                                    {
                                                        <span     class="label     label-success">
@packager.Sla.ToString("0.00", CultureInfo.InvariantCulture)</span>
                                                    }
                                                    else
                                                    {
                                                        <span          class="label          label-
warning">@packager.Sla.ToString("0.00", CultureInfo.InvariantCulture)</span>
                                                    }
                                                </td>
                                            </tr>
                                            <tr>
                                                <td>Previous Packager</td>
                                                @if (packager.IsPreviousPackager > 0)
                                                {
                                                    <td style="text-align:right;">True</td>
                                                }
                                                else
                                                {
                                                    <td style="text-align:right;">False</td>
                                                }
                                            </tr>
                                            <tr>
                                                <td>Packages</td>
                                                <td                                        style="text-
align:right;">@packager.Packages.ToString("0.00", CultureInfo.InvariantCulture)</td>
                                            </tr>
                                            <tr>
                                                <td>Complexity</td>
                                                <td                                        style="text-
align:right;">@packager.Complexity</td>
                                            </tr>
                                            <tr>
                                                <td>Priority</td>
                                                <td                                        style="text-
align:right;">@packager.Priority</td>
                                            </tr>
                                            <tr>
                                                <td>Workload</td>
                                                <td                                        style="text-
align:right;">@packager.WorkLoad</td>
                                            </tr>
                                            <tr>
                                                <td>Status</td>
                                                <td                                        style="text-
align:right;">@packager.Status.ToString("0.00", CultureInfo.InvariantCulture)</td>
                                            </tr>
                                            <tr>
                                                <td>Tickets</td>
                                                <td                                        style="text-
align:right;">@packager.Tickets.Count</td>
                                            </tr>
                                        </table>
                                    </div>
                                    <div class="packager-front" style="float:right; width:65%;">
```

```
                                            <img  src="https://i1.wp.com/www.winhelponline.com/blog/wp-
content/uploads/2017/12/user.png?resize=256%2C256&quality=100&ssl=1" class="img-rounded" />
                                            <h3>@GenerateName()</h3>


                                            @if (packager.Tickets.Count > 0)
                                            {
                                                foreach (var tickets in packager.Tickets)
                                                {
                                                    <div class="pull-right">
                                                        <table      class="table      table-condensed"
style="margin-left:20px; text-align:left; table-layout:fixed; margin-bottom:0px !important">
                                                            <tr>
                                                                <td>
                                                                    <a
href="https://sdc.revacom.com/browse/@tickets.Key" target="_blank">

@tickets.Key.Replace("P","R").Replace("A","O").Replace("T","M").Replace("L","E")
                                                                    </a>
                                                                </td>
                                                                <td>
                                                                    @if (tickets.PriorityRatio == 1)
                                                                    {
                                                                        <span class="label label-danger
element-rounded">Urgent</span>
                                                                    }
                                                                    else
                                                                    {
                                                                        <span    class="label    label-
success element-rounded">Normal</span>
                                                                    }
                                                                </td>
                                                                <td>
                                                                    <span   class="label   label-warning
element-rounded">@tickets.Status</span>
                                                                </td>
                                                                <td><span    class="label    label-info
element-rounded pull-right">@GetTimeSpanInSdc(tickets.Issue.SlaRemaining)</span></td>
                                                            </tr>
                                                        </table>

                                                    </div>
                                                }
                                            }
                                            else
                                            {
                                                <strong>There are no issues assigned</strong>
                                            }
                                        </div>
                                    </div>
                                    <div class="footer title pull-right">
                                        <button class="btn btn-success element-rounded">Assign</button>
                                        <button    class="btn    btn-default    element-rounded"    data-
toggle="modal" data-target="#detailsModal" data-content="@packager.Name">Details</button>
                                        <button    class="btn    btn-primary    element-rounded"    data-
toggle="modal"       data-target="#chartModal"       data-whatever="@packager.WorkloadArray"       data-
content="@packager.Name">Workload</button>
                                    </div>
                                </div>
                            }
                        </div>
                    </td>
                </tr>
            }
        </tbody>
    </table>
</div>
```

```
<footer class="navbar navbar-default navbar-fixed-bottom footer" style="position:fixed; min-height:20px
!important" id="navbar">
    <ul class="nav navbar-nav">
        @foreach (var item in Model)
        {
            if (item.SdcIssue.Key.Contains("ATUM"))
            {
                continue;
            }
            <li><a                          class="list-group-item                          bottom-menu"
href="#@item.SdcIssue.Key">@item.SdcIssue.Key</a></li>
        }
    </ul>
</footer>

<div class="modal fade" id="chartModal" tabindex="-1" role="dialog" aria-labelledby="exampleModalLabel"
aria-hidden="true">
    <div class="modal-dialog" style="width:90%; height:750px" role="document">
        <div class="modal-content">
            <div class="modal-header">
                <button      type="button"      class="close"      data-dismiss="modal"><span      aria-
hidden="true">×</span><span class="sr-only">Close</span></button>
                <h4 class="modal-title"></h4>
            </div>
            <div class="responsive-frame">
                <canvas class="framed" id="myChart" data-content=""></canvas>
            </div>
            <div class="modal-footer">
                <button     type="button"     id="closeChart"     class="btn     btn-primary"     data-
dismiss="modal">Close</button>
            </div>
        </div>
    </div>
</div>

<div  class="modal"  id="detailsModal"  tabindex="-1"  role="dialog"  aria-labelledby="exampleModalLabel"
aria-hidden="true">
    <div class="modal-dialog" style="width:90%; height:750px" role="document">
        <div class="modal-content" style="height:80%">
            <div class="modal-header">
                <button      type="button"      class="close"      data-dismiss="modal"><span      aria-
hidden="true">×</span><span class="sr-only">Close</span></button>
                <h4 class="modal-title" id="lineModalLabel"></h4>
            </div>
            <div class="responsive-frame">
                <iframe     id="frameDetails"     class="framed-details"     frameborder="0"     src=""
allow="autoplay; encrypted-media" allowfullscreen></iframe>
            </div>
            <div class="modal-footer">
                <button type="button" class="btn btn-primary" data-dismiss="modal">Close</button>
            </div>
        </div>
    </div>
</div>

<div  class="modal"  id="packagesModal"  tabindex="-1"  role="dialog"  aria-labelledby="exampleModalLabel"
aria-hidden="true">
    <div class="modal-dialog" style="width:90%; height:750px" role="document">
        <div class="modal-content" style="height:80%">
            <div class="modal-header">
                <button      type="button"      class="close"      data-dismiss="modal"><span      aria-
hidden="true">×</span><span class="sr-only">Close</span></button>
                <h4 class="modal-title" id="lineModalLabel"></h4>
            </div>
            <div class="responsive-frame">
                <iframe     id="framePackages"     class="framed-details"     frameborder="0"     src=""
allow="autoplay; encrypted-media" allowfullscreen></iframe>
            </div>
```

```
            <div class="modal-footer">
                <button type="button" class="btn btn-primary" data-dismiss="modal">Close</button>
            </div>
        </div>
    </div>
</div>

<script>
    $('#chartModal').on('show.bs.modal', function (event) {
        var button = $(event.relatedTarget);

        var dataArray = button.data('whatever');
        var packager = button.data('content');

        var modal = $(this);
        modal.find('.modal-title').text('Workload');
        modal.find('#myChart').attr('data-content', dataArray);

        var element = document.getElementById("myChart");
        var ctx = document.getElementById("myChart").getContext('2d');
        this.chart = new Chart(ctx, {
            type: 'line',
            data: {
                labels: ["Week1", "Week2", "Week3", "Week4", "Week5", "Week6", "Week7"],
                datasets: [{
                    label: "Packager`s workload",
                    borderColor: 'rgb(255, 99, 132)',
                    data: element.getAttribute("data-content").split(","),
                    fill: false
                },
                {
                    data: [25, 20, 16, 10, 11, 21, 21],
                    label: "Team load",
                    fill: false,
                    borderColor: "#3e95cd"
                }]
            }
        });
    });
    $('#detailsModal').on('show.bs.modal', function (event) {

        var button = $(event.relatedTarget);
        var packager = button.data('content');
        var str = 'Roboboss/Details?username=' + packager;
        var modal = $(this);

        modal.find('.modal-title').text('Issue details for ' + packager);
        modal.find('#frameDetails').attr('src', str);
    });

    $('#packagesModal').on('show.bs.modal', function (event) {

        var button = $(event.relatedTarget);
        var keys = button.data('content');
        var str = 'Roboboss/PreviousPackages?key=' + keys;
        var modal = $(this);

        modal.find('.modal-title').text('Previous requests for the issue');
        modal.find('#framePackages').attr('src', str);
    });

    $('#exampleModal').on('hidden.bs.modal', function () {
        this.chart.destroy();
    });

    $(document).ready(function () {
        $('#container').scrollspy({
            target: ".scrollspy",
```

```
            offset: 50
        });
    });
    //$(".navbar a").on('click', function (event) {
    //    event.preventDefault();
    //    var hash = this.hash;
    //    $('#container').animate({
    //        scrollTop: $(hash).offset().top - 50
    //    }, 800, function () {
    //        window.location.hash = hash;
    //    });
    //});
</script>
```