НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ імені ІГОРЯ СІКОРСЬКОГО»

ФАКУЛЬТЕТ ІНФОРМАТИКИ ТА ОБЧИСЛЮВАЛЬНОЇ ТЕХНІКИ

Кафедра автоматизованих систем обробки інформації та управління

До захисту допущено:
В.о. завідувача кафедри
_____ *Олександр ПАВЛОВ*
(підпис) (вл.ім'я, прізвище)
"___"_____2021 р.

Дипломний проєкт
на здобуття ступеня бакалавра

за освітньо-професійною програмою «Інформаційні управляючі системи та технології»
спеціальності 126 «Інформаційні системи та технології»

на тему: «Інформаційна система з розподілу завдань між виконавцями»

Виконав: студент *IV* курсу, групи *ІС-71*
Ноженко Ярослав Ігорович _____
(прізвище, ім'я, по батькові) (підпис)

Керівник *доц., к.т.н. Жураковська Оксана Сергіївна* _____
(посада, науковий ступінь, вчене звання, прізвище, ім'я, по батькові) (підпис)

Консультант з графічної документації *доц., к.т.н., доц. Жданова Олена Григорівна* _____
(посада, науковий ступінь, вчене звання, прізвище, ім'я, по батькові) (підпис)

Рецензент *Доц. каф. ТК ФІОТ, к.т.н., доц. Лісовиченко Олег Іванович* _____
(посада, науковий ступінь, вчене звання, прізвище, ім'я, по батькові) (підпис)

Засвідчую, що у цьому дипломному проєкті немає запозичень з праць інших авторів без відповідних посилань.

Студент _____
(підпис)

Київ – 2021 року

Національний технічний університет України
“Київський політехнічний інститут імені Ігоря Сікорського”

Факультет (інститут) інформатики та обчислювальної техніки
(повна назва)

Кафедра *автоматизованих систем обробки інформації та управління*
(повна назва)

Рівень вищої освіти – перший (бакалаврський)

Спеціальність – 126 «Інформаційні системи та технології»

Освітньо-професійна програма «Інформаційні управляючі системи та технології»

ЗАТВЕРДЖУЮ
В.о. завідувача кафедри
_____ *Олександр ПАВЛОВ*
(підпис) (вл.ім’я, прізвище)
“___”_____2021 р.

З А В Д А Н Н Я
на дипломний проєкт студенту

Ноженку Ярославу Ігоровичу
(прізвище, ім’я, по батькові)

1. Тема проєкту «Інформаційна система з розподілу завдань між виконавцями»
керівник проєкту *Жураковська Оксана Сергіївна, к.т.н.*
(прізвище, ім’я, по батькові, науковий ступінь, вчене звання)
затверджені наказом по університету від “*11*” *05* 2021 р. № 1139-с

2. Термін подання студентом проєкту “*04*”*червня* 2021 року

3. Вихідні дані до проєкту
Технічне завдання

4. Зміст пояснювальної записки
1. Загальні положення: основні визначення та терміни, опис предметного середовища, огляд ринку програмних продуктів, постановка задачі
2. Інформаційне забезпечення: вхідні та вихідні дані, опис структури бази даних
3. Математичне забезпечення: змістовна та математична постановки задачі, обґрунтування та опис методу розв’язання
4. Програмне та технічне забезпечення: засоби розробки, вимоги до технічного забезпечення, архітектура програмного забезпечення, побудова звітів
5. Технологічний розділ: керівництво користувача, методика випробувань програмного продукту

5. Перелік графічного матеріалу
1. Схема структурна варіантів використань
2. Схема бази даних

3. Схема архітектури програмного забезпечення

4. Схема структурна послідовності

5. Схема структурна класів програмного забезпечення

6. Схема структурна компонентів

6. Консультанти розділів проєкту

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

7. Дата видачі завдання *«7» квітня 2021 року*

Календарний план

№ з/п	Назва етапів виконання дипломного проєкту	Термін виконання етапів проєкту	Примітка
1.	*Пошук матеріалів, опис предметного середовища*	*16.04.2021*	
2.	*Виявлення вимог до програмного продукту*	*21.04.2021*	
3.	*Аналіз існуючих рішень*	*23.04.2021*	
4.	*Формування змістовної та математичної постановки задачі*	*30.04.2021*	
5.	*Вибір алгоритму та алгоритмізація задачі*	*03.05.2021*	
6.	*Розробка програмного забезпечення*	*11.05.2021*	
7.	*Тестування та налагодження програмного забезпечення*	*12.05.2021*	
8.	*Виконання графічних документів*	*13.05.2021*	
9.	*Оформлення пояснювальної записки*	*14.05.2021*	
10.	*Подання ДП на попередній захист*	*14.05.2021*	
11.	*Подання ДП на основний захист*	*04.06.2021*	
12.	*Подання ДП рецензенту*	*07.06.2021*	

Студент Ярослав НОЖЕНКО

Керівник Оксана ЖУРАКОВСЬКА

№ з/п	Формат	Позначення	Найменування	Кількість листів	Примітка
1	А4		Завдання на дипломний проєкт	2	
2	А4	ДП 7119.00.000 ПЗ	Пояснювальна записка	75	
3	А4	ДП 7119.01.000 ТЗ	Технічне завдання	9	
4	А3	ДП 7119.02.000 ССВ	Схема структурна варіантів використання	1	
5	А3	ДП 7119.03.000 СБД	Схема бази даних	1	
6	А3	ДП 7119.04.000 СА	Схема архітектури програмного забезпечення	1	
7	А3	ДП 7119.05.000 ССП	Схема структурна послідовності	1	
8	А3	ДП 7119.06.000 ССК	Схема структурна класів програмного забезпечення	1	
9	А3	ДП 7119.07.000 СК	Схема структурна компонентів	1	

Зм.	Арк.	ПІБ	Підп.	Дата	ДП 7119 00.000.00		
Розробн.		Ноженко Я.І.			Відомість дипломного проєкту	Літ.	Лист 1 / Листів 1
Керівн.		Жураковська О.С.				КПІ ім. Ігоря Сікорського Каф. АСОІУ Гр. ІС-71	
Консульт.							
Н/контр.		Жданова О.Г..					
В.о.зав.каф.		Павлов О.А.					

Пояснювальна записка
до дипломного проєкту

на тему: Інформаційна система з розподілу завдань між виконавцями

Київ – 2021 року

АНОТАЦІЯ

Структура та обсяг роботи. Пояснювальна записка дипломного проєкту складається з п’яти розділів, містить 57 рисунків, 4 таблиці, 6 додатків, 11 джерел.

В першому розділі проаналізовано предметне середовище ІТ-проєктів, визначена проблема розподілу завдань між виконавцями. Проведено огляд наявних аналогів та сформульовано мету і задачу дослідження процесу розподілу завдань.

В другому розділі описано вхідні та вихідні дані, які мають місце в ІТ-проєктах, наведено схему бази даних.

В третьому розділі базуючись на попередніх даних наведено змістовну та математичні постановку задачі. Виявлено метод вирішення для алгоритмізації побудови розподілу завдань між виконавцями базуючись на вхідних критеріях.

В четвертому розділі наведено програмне та технічне забезпечення, яке було використане під час розробки та яке необхідне для роботи з системою.

П’ятий розділ містить керівництво користувача, опис проведених випробувань функціональних можливостей системи та їх результати.

Дипломний проєкт присвячений розробці інформаційної системи з розподілу завдань між виконавцями.

РОЗПОДІЛ ЗАВДАНЬ, ЗАДАЧА БАГАТОКРИТЕРІАЛЬНОГО ПРИЙНЯТТЯ РІШЕНЬ, МЕТОД VIKOR

ABSTRACT

Structure and scope of work. The explanatory note of the diploma project consists of five sections, contains 60 figures, 4 tables, 2 appendices, 2 sources.

In the first section the subject environment of IT-projects is analyzed, the problem of distribution of tasks between executors is defined. The review of available analogues is carried out and the purpose and task of research of process of distribution of tasks is formulated.

The second section describes the input and output data that take place in IT-projects, provides a diagram of the database.

The third section, based on previous data, provides a meaningful and mathematical formulation of the problem. The method of the decision for algorithmization of construction of distribution of tasks between executors based on input criteria is revealed.

The fourth section lists the software and hardware that was used during development and that is required to work with the system.

The fifth section contains a user guide, a description of the system functionality tests and their results.

The diploma project is devoted to the development of an information system for the distribution of tasks between performers.

DISTRIBUTION OF TASKS, THE TASK OF MULTICRITERIAL DECISION-MAKING, VIKOR METHOD

ЗМІСТ

ВСТУП

Наразі, IT-сфера поповнилася різними методологіями розробки. Однією з найпопулярніших методологій є Scrum [1] методологія.

Суть методологія полягає в розділені процесу розробки продукту на ітерації, що тривають невеликий період та по закінченню надають частину готового функціоналу продукту. Після кожної ітерації проводиться її аналіз, щоб виявити сильні та слабкі сторони команди та розробників. Далі, ця інформація буде використана для планування наступної ітерації. Планування ітерації включає процес призначення необхідних завдань, які необхідно виконати за ітерацію та призначення виконавців. Далі, виконується розподіл завдань між виконавцями базуючись на аналізах попередніх ітерацій. Для розподілу завдань менеджер базується на параметрах завдання, що характеризують його (наприклад, тип завдання, примітки від автора завдання, щодо сфери пов’язаної з завданням, складність завдання) та досвіду виконавців з завданнями зі схожими параметрами.

Цей процес не є складною процедурою, але є досить часозатратною, так як на практиці менеджеру необхідно ознайомитися з параметрами кожного завдання ітерації, а потім, виконати вербальну взаємодію з виконавцями для вияснення наявного досвіду роботи з завданнями зі схожими параметрами. Якщо метою є виконання завдання якнайшвидше, то виконавець з найбільшим досвідом зі схожими завданнями є першим кандидатом на отримання його для виконання під час ітерації.

Ці дії не є складними, не залучають творчої діяльності людини, параметри завдань є скінченними та інформація про досвід виконавця може бути отримана шлях аналізу історії змін завдань, що виконував виконавець, тож вони можуть бути ефективно автоматизовані.

В цій роботі описано рішення, що було розроблене для автоматизації цих дії, яке дозволяє в загальних випадках планування ітерацій полегшити роботу менеджера, шляхом надання йому рекомендованого розподілу завдань та

звільнити час, що використовувався для взаємодії менеджера та виконавців під час розподілу завдань, що дозволить використати його для реальної роботи виконавців.

Практичне значення одержаних результатів.

Розроблене програмне забезпечення може бути застосовано для підтримки процесу розподілу завдань між виконавцями ІТ-проєктів.

Публікації.

Результати роботи були опубліковані в тезах доповідей на науково-технічній конференції [2].

1 ЗАГАЛЬНІ ПОЛОЖЕННЯ

1.1 Опис предметного середовища

Досить популярною методологією розробки в IT-сфері є методологія Scrum [1]. Життєвий цикл Scrum-проєктів складаються з ітерацій, що виконуються одна за одною і кожна передбачає виконання деякої частини роботи, що в сумі приводить до готового кінцевого продукту. Перед кожною ітерацією проводиться її планування. Створюється список завдань ітерації, куди відбираються завдання, що мають буть виконані в межах цієї ітерації та призначаються виконавці, які будуть виконувати ці завдання. Далі, менеджер розподіляє завдання між виконавцями. Цей процес залучає всіх виконавців, так як він передбачає вербальну взаємодію і займає деякий час.

1.1.1 Опис процесу діяльності

В більшості випадків процес розподілу задач супроводжуються запитаннями менеджера до виконавців, з метою вияснення того хто найбільш ефективніше (швидше) зможе зробити відповідну задачу, базуючись на їх досвіді роботи з задачами такого типу чи досвіду роботи зі схожими задачами. Тож цей процес є повністю вербальним та займає деякий час.

При використанні інформаційної системи з розподілу задач між виконавцями, процес буде набагато спрощено. Менеджер проекту зможе запросити у системи варіанти розподілу задач за заданими критерія які можливо гнучко налаштувати та вибрати для кожної задачі виконавця із списку найкращих варіантів. Це дозволить позбутися обов’язкової взаємодії менеджера з виконавцями, тим самим звільнив час для реальної роботи виконавців.

1.1.2 Опис функціональної моделі

З системою взаємодіють два актора: менеджер і виконавець. Менеджер відповідальний за формування завдань та їх розподіл між виконавцями, виконавець повинен змінювати стани задач відповідно до процесу роботи над ними.

Вимоги до варіантів використання наведені у таблиці 1.1.

Таблиця 1.1 - Вимоги до варіантів використання

Актор	Варіант використання	Опис вимог варіанта використання
Виконавець/Менеджер	Реєстрація	Реєстрація за допомогою поштової адреси та паролю
	Авторизація	Авторизація за допомогою поштової адреси та паролю
Виконавець	Отримання переліку задач	Отримання переліку задач, що призначені виконавцю
	Зміна стану задач	Зміна стану призначеної задачі
Менеджер	Ведення виконавців	Додавання виконавців на проект
	Ведення задач	Управління задачами (додавання, зміна)
	Формування автоматичного розподілу задач	Формування розподілу задач за заданими критеріями

Схема структурна варіантів використання наведена в графічних матеріалах.

1.2 Огляд наявних аналогів

Існує багато наявних рішень для керування проектами. Розглянемо деякі з них.

Atlassian Jira [3] – одне за найпопулярніших рішень. Має повну підтримку Agile процесів, просунуте управління проектами та існує можливість автоматизації деяких процесів. Недоліками є те, що сервіс є платним та вбудовані можливості автоматизації процесів дозволяє автоматизувати лише найпримітивніший розподіл завдань.

Microsoft Azure DevOps [4] – ще одне популярне рішення, яке має підтримку Agile. Сервіс є комплексним рішенням, що включає можливість керування всіма етапами розробки продукту, але не має ніяких можливостей для автоматичного розподілу завдань.

JetBrains YouTrack [5] – рішення для управління проектами, яке дає можливість написання власних скриптів для автоматизації деяких процесів, в тому числі розподілу завдань, але це повинна бути власна реалізація алгоритму і не є вбудованою можливістю.

1.3 Постановка задачі

Розробити інформаційну систему, що дозволяє ефективно розподілити завдання між виконавцями базуючись на заданих критеріях.

1.3.1 Призначення розробки

Підтримка процесу розподілу завдань між виконавцями з урахуванням заданих критеріїв.

1.3.2 Цілі та задачі розробки

Підвищення ефективності процесу розподілу завдань між виконавцями за критеріями, що враховують параметри завдань та показники ефективності виконавців.

Задачі розробки:

- ведення задач проекту, формування параметрів задач;
- ведення виконавців;
- формування показників ефективності виконавців;
- формування розподілу задач між виконавцями із врахуванням параметрів задач і показників ефективності виконавців;
- розробка бази даних;
- розробка інтерфейсу.

Висновок до розділу

В даному розділі наведений опис предметної області. Наведений опис процесу діяльності до та після впровадження системи. Зроблено огляд існуючих аналогів. Сформульовані функціональні вимоги до системи, наведено діаграму варіантів використання. Здійснено постановку задачі, сформульовано призначення розробки, мету та перелік задач для досягнення поставленої мети.

2 ІНФОРМАЦІЙНЕ ЗАБЕЗПЕЧЕННЯ

2.1 Вхідні дані та вихідні дані

Вхідні дані програмного забезпечення представленні у вигляді параметрів запитів до API, та тіла запита, що може містити моделі описані вище. Вхідні дані показано в таблиці 2.1 у колонці «Кінцева точка API» у вигляді параметрів запиту виду *{назва параметру}*, та у колонці «Тіло запиту», що може містити модель. Вихідні дані описані у колонці «Опис та результат».

Взаємодія з системою виконується за допомогою HTTP-запитів до API. Вхідні та вихідні дані, що використовуються в API наведено на рисунках 2.1-2.15.

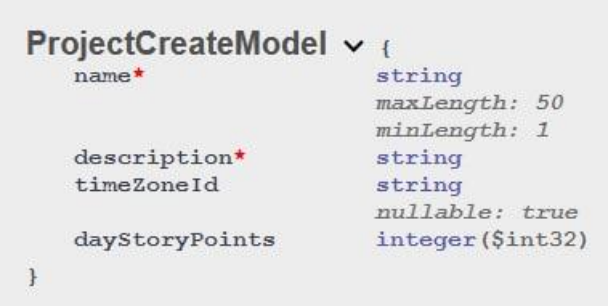


Рисунок 2.1 – Модель створення проєкту

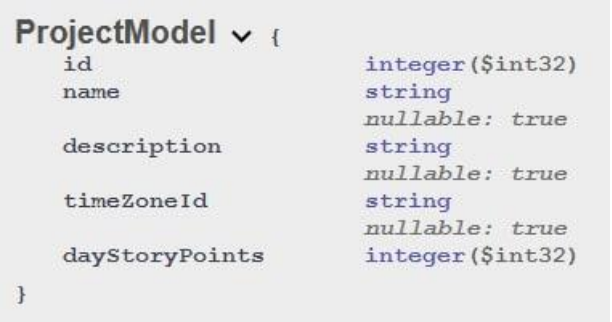


Рисунок 2.2 – Модель проєкту

```
ProjectUserCreateModel ▾ {
    userId*          string
    kind*            UserKind  integer($int32)
                     Enum:
                      ▾ [ 0, 1, 2 ]
}
```

Рисунок 2.3 – Модель створення проєктного користувача (виконавця)

```
ProjectUserModel ▾ {
    id               integer($int32)
    userId           string
                     nullable: true
    projectId        integer($int32)
    kind             UserKind  integer($int32)
                     Enum:
                      ▸ Array [ 3 ]
}
```

Рисунок 2.4 – Модель проєктного користувача (виконавця)

```
RuleCreateModel ▾ {
    rule*            string
}
```

Рисунок 2.5 – Модель створення правила побудови критерія

```
RuleModel ▾ {
    projectId        integer($int32)
    rule             string
                     nullable: true
}
```

Рисунок 2.6 – Модель правила побудови критерія

```
SprintCreateModel ▾ {
    name*            string
                     maxLength: 50
                     minLength: 1
    startDateTime*   string($date-time)
    endDateTime*     string($date-time)
}
```

Рисунок 2.7 – Модель створення ітерації

```
SprintModel {
    id                  integer($int32)
    projectId           integer($int32)
    name                string
                        nullable: true
    startDateTime       string($date-time)
    endDateTime         string($date-time)
}
```

Рисунок 2.8 – Модель ітерації

```
StartWorkDistributionModel {
    rulesIds*           [
                        minItems: 1
                        integer($int32)]
    weights*            [
                        minItems: 1
                        number($double)]
    vCoef*              number($double)
}
```

Рисунок 2.9 – Модель параметрів розподілу завдань

```
UserModel {
    id                  string
                        nullable: true
    userName            string
                        nullable: true
    email               string
                        nullable: true
    kind                UserKind  integer($int32)
                        Enum:
                        [ 0, 1, 2 ]
}
```

Рисунок 2.10 – Модель користувача системи

```
UserRegisterModel {
    userName*           string
    email*              string($email)
    password*           string
}
```

Рисунок 2.11 – Модель даних реєстрації користувача системи

```
UserSignModel ▾ {
    email*          string($email)
    password*       string
}
```

Рисунок 2.12 – Модель даних авторизації користувача системи

```
WorkItemChangeModel ▾ {
    id                          integer($int32)
    workItemId                  integer($int32)
    oldName                     string
                                nullable: true
    newName                     string
                                nullable: true
    oldContent                  string
                                nullable: true
    newContent                  string
                                nullable: true
    oldRemainingStoryPoints     integer($int32)
    newRemainingStoryPoints     integer($int32)
    oldPriority                 integer($int32)
    newPriority                 integer($int32)
    oldKind                     WorkItemKind  integer($int32)
                                Enum:
                                ▾ [ 0, 1, 2 ]
    newKind                     WorkItemKind  integer($int32)
                                Enum:
                                ▾ [ 0, 1, 2 ]
    oldStatus                   WorkItemStatus  integer($int32)
                                Enum:
                                ▾ [ 0, 1, 2, 3, 4, 5, 6 ]
    newStatus                   WorkItemStatus  integer($int32)
                                Enum:
                                ▾ [ 0, 1, 2, 3, 4, 5, 6 ]
    oldAssignedProjectUserId    integer($int32)
    newAssignedProjectUserId    integer($int32)
    estimatedStoryPoints        integer($int32)
    dateTime                    string($date-time)
}
```

Рисунок 2.13 – Модель даних про зміну властивостей завдання

```
WorkItemCreateModel ▾ {
  name*                 string
                        maxLength: 50
                        minLength: 1
  content               string
                        nullable: true
  estimatedStoryPoints  integer($int32)
  priority              integer($int32)
  kind                  WorkItemKind  integer($int32)
                        Enum:
                         ▾ [ 0, 1, 2 ]
  tags                  string
                        nullable: true
}
```

Рисунок 2.14 – Модель створення завдання

```
WorkItemPutModel ▾ {
  name                  string
                        nullable: true
  content               string
                        nullable: true
  estimatedStoryPoints  integer($int32)
  remainingStoryPoints  integer($int32)
  priority              integer($int32)
  kind                  WorkItemKind  integer($int32)
                        Enum:
                         ▾ [ 0, 1, 2 ]
  status                WorkItemStatus  integer($int32)
                        Enum:
                         ▾ [ 0, 1, 2, 3, 4, 5, 6 ]
  assignedProjectUserId integer($int32)
  tags                  string
                        nullable: true
}
```

Рисунок 2.15 – Модель зміни завдання

Таблиця 2.1 – Опис кінцевих точок API та їх вхідних і вихідних даних

Кінцева точка API	Тіло запиту	Опис та результат
GET /api/projects	-	Отримання моделей проєктів, де авторизований користувач системи є проєктним користувачем
POST /api/projects	Модель створення проєкту	Створення нового проєкту
GET /api/projects/{projectId}	-	Отримання моделі проєкту з ідентифікатором {projectId}
DELETE /api/projects/{projectId}	-	Видалення проєкту з ідентифікатором {projectId}
GET /api/projects/{projectId}/project-users	-	Отримання моделей проєктних користувачів на проєкті з ідентифікатором {projectId}
POST /api/projects/{projectId}/project-users	Модель проєктного користувача	Додання проєктного користувача на проєкт з ідентифікатором {projectId}
DELETE /api/projects/{projectId}/project-users/{projectUserId}	-	Видалення проєктного користувача з ідентифікатором {projectUserId} з проєкту з ідентифікатором {projectId}

Продовження таблиці 2.1

GET /api/projects/{projectId}/project-users/calculate-statistics	-	Ручний виклик процесу обчислення показників ефективності проєктого користувача на проєкті з ідентифікатором{projectId}
GET /api/projects/{projectId}/rules	-	Отримання моделей правил побудови критеріїв, що збережені на проєкті з ідентифікатором{projectId}
POST /api/projects/{projectId}/rules	Модель створення правила побудови критерія	Збереження правила побудови критеріїв на проєкт з ідентифікатором{projectId}
GET /api/projects/{projectId}/rules/{ruleId}	-	Отримання моделі правила побудови критерія з ідентифікатором {ruleId}, що збережене на проєкті з ідентифікатором{projectId}
DELETE /api/projects/{projectId}/rules/{ruleId}	-	Видалення правила побудови критерія з ідентифікатором {ruleId}, що збережене на проєкті з ідентифікатором{projectId}

Продовження таблиці 2.1

GET /api/projects/{projectId}/sprints	-	Отримання моделей ітерацій на проєкті з ідентифікатором{projectId}
POST /api/projects/{projectId}/sprints	Модель створення ітерації	Додання ітерації на проєкт з ідентифікатором{projectId}
GET /api/projects/{projectId}/sprints /{sprintId}	-	Отримання моделі ітерації з ідентифікатором {sprintId} на проєкті з ідентифікатором{projectId}
GET /api/projects/{projectId}/sprints /{sprintId}/work-items	-	Отримання моделей завдань, що призначені на ітерацію з ідентифікатором {sprintId} на проєкті з ідентифікатором{projectId}
GET /api/projects/{projectId}/sprints /{sprintId}/project-users	-	Отримання моделей проєктних користувачів, що призначені на ітерацію з ідентифікатором {sprintId} на проєкті з ідентифікатором{projectId}
POST /api/projects/{projectId}/sprints /{sprintId}/work-items/{workItemId}	-	Призначення завдання з ідентифікатором {workItemId} на ітерацію з ідентифікатором {sprintId} на проєкті з ідентифікатором{projectId}

Продовження таблиці 2.1

DELETE /api/projects/{projectId}/sprints/{sprintId}/work-items/{workItemId}	-	Видалення завдання з ідентифікатором {workItemId}, що призначене на ітерацію з ідентифікатором {sprintId} на проєкті з ідентифікатором{projectId}
POST /api/projects/{projectId}/sprints/{sprintId}/project-users/{projectUserId}	-	Призначення проєктного користувача з ідентифікатором {projectUserId} на ітерацію з ідентифікатором {sprintId} на проєкті з ідентифікатором{projectId}
DELETE /api/projects/{projectId}/sprints/{sprintId}/project-users/{projectUserId}	-	Видалення проєктного користувача з ідентифікатором {projectUserId}, що призначений на ітерацію з ідентифікатором {sprintId} на проєкті з ідентифікатором{projectId}
POST /api/projects/{projectId}/sprints/{sprintId}/make-work-distributions	Модель параметрів розподілу завдань	Отримання моделі розподілу завдань між проєктними користувачами, де завдання і користивачі призначені на ітерацію з ідентифікатором {sprintId} на проєкті з ідентифікатором{projectId} з заданими параметрами

Продовження таблиці 2.1

GET /api/users	-	Отримання моделей користувачів системи
POST /api/users/register	Модель даних реєстрації користувача системи	Реєстрація користувача системи
POST /api/users/login	Модель даних авторизації користувача системи	Отримання токену авторизації користувача системи
GET /api/projects/{projectId}/work-items/{workItemId}/work-item-changes	-	Отримання моделей даних про зміну властивостей завдання з ідентифікатором {workItemId} на проєкті з ідентифікатором {projectId}
GET /api/projects/{projectId}/work-items/all	-	Отримання моделей завдань на проєкті з ідентифікатором {projectId}
GET /api/projects/{projectId}/work-items	-	Отримання моделей призначених завдань авторизованого користувача системи на проєкті з ідентифікатором {projectId}

Продовження таблиці 2.1

POST /api/projects/{projectId}/work-items	Модель створення завдання	Додання завдання на проєкт з ідентифікатором {projectId}
GET /api/projects/{projectId}/work-items/{workItemId}	-	Отримання завдання з ідентифікатором {workItemId} на проєкті з ідентифікатором {projectId}
PUT /api/projects/{projectId}/work-items/{workItemId}	Модель завдання	Зміна завдання з ідентифікатором {workItemId} на проєкті з ідентифікатором {projectId}
DELETE /api/projects/{projectId}/work-items/{workItemId}	-	Видалення завдання з ідентифікатором {workItemId} на проєкті з ідентифікатором {projectId}

2.2 Опис структури бази даних

Діаграма зв’язків між сутностями бази даних наведено в графічних матеріалах. Опис таблиць бази даних представлено в таблиці 2.2.

Таблиця 2.2 – Описи таблиць бази даних

Назва таблиці	Опис
UserEntity	Містить користувачів системи
ProjectEntity	Містить проєкти
ProjectUserEntity	Містить проєктних користувачів (виконавців). Кожен користувач системи може бути проєктним користувачем на декількох проєктах

Продовження таблиці 2.2

WorkItemEntity	Містить завдання на проєкті
WorkItemChangeEntity	Містить дані про зміну завдання на проєкті
SprintEntity	Містить ітерації проєктів
SprintEntityWorkItemEntity	Допоміжна таблиця для зв’язку типу «багато до багатьох» між ітерацією та завданнями проєкту
ProjectUserEntitySprintEntity	Допоміжна таблиця для зв’язку типу «багато до багатьох» між ітерацією та проєктними користувачами
RuleEntity	Містить правила побудови критеріїв
WorkItemDayProjectUserStatisticEntity	Містить денну статистику проєктного користувача
WorkItemKindProjectUserStatisticEntity	Містить статистику завдань проєктного користувача базуючись на типу завдання
WorkItemTagProjectUserStatisticEntity	Містить статистику завдань проєктного користувача базуючись на тегу завдання

Висновок до розділу

В даному розділі були описані методи взаємодії з програмою, їх вхідні та вихідні дані та наведено опис схеми бази даних.

3 МАТЕМАТИЧНЕ ЗАБЕЗПЕЧЕННЯ

3.1 Змістовна постановка задачі

Існує множина виконавців, кожен з яких характеризується певними властивостями та інформацією про вирішувані ним раніше задачі. Існує множина задач, які характеризуються множиною властивостей.

Необхідно сформувати розподіл задач за виконавцями, так щоб він був найбільш ефективний за визначеною множиною критеріїв.

3.2 Математична постановка задачі

Існує множина задач $T = \{t_i\}, i = 1..n$, кожна задача t_i характеризується набором параметрами $(y_{i1}, .., y_{ik})$, які визначають її тип та рівень складності, де y_{ij} – значення параметру j для задачі t_i.

Існує множина виконавців $U = \{u_i\}, i = 1..m$.

Задано множину показників ефективності виконання задач кожним виконавцем

$S = \{s_{ij}\}, i = 1..m, j = 1..n,$

s_{ij} – показник ефективності виконання задач з параметрами $(y_{j1}, .., y_{jk})$ виконавцем u_i.

Задано множину критеріїв, яка визначається особою, що приймає рішення. Кожен критерій формується у вигляді функції від множини параметрів задач та показників ефективности виконання задач виконавцем

$C = c_p((y_{i1}, .., y_{ik}), s_{ji}), p = 1..l.$

Задано множину вагових коефіцієнтів критеріїв $w_j, j = 1..l$.

Необхідно для кожної задачі t_i сформувати множину виконавців, які можуть вважатися найбільш ефективними за сукупністю критеріїв множини С.

3.3 Обґрунтування методу розв’язання

Сформульована задача може бути розв'язана за допомогою багатокритеріальних методів прийняття рішень.

Розглянемо три багатокритеріальних метода прийняття рішень: ELECTRE, TOPSIS, VIKOR [6].

ELECTRE [7] – в цьому методі використовуються чіткі бінарні відношення між заданими альтернативами. Етапи метода:

1) розрахунок нормалізованої зваженої матриці рішень;
2) побудова множин узгодженості та неузгодженості;
3) розрахунок матриць узгодженості та неузгодженості;
4) побудова матриць переваг узгодженості та неузгодженості;
5) побудова агрегованої матриці переваг;
6) видалення гірших альтернатив.

TOPSIS [8] – основна ідея цього методу є знаходження альтернативи, що є найближчою до «ідеальної» точки та найвіддаленішою від «ідеально-негативної» точки. Етапи метода:

1) розрахунок нормалізованої зваженої матриці рішень;
2) визначення «ідеальної» та «ідеально-негативної» точок;
3) розрахунок наближення до точок;
4) ранжування критеріїв.

VIKOR [9] – ідея цього методу в знаходженні оптимальної множини альтернатив використовуючи стратегію, що передбачає мінімізацію індивідуальних витрат та максимізацію групової корисності.

Найбільш прийнятним методом для рішення даної проблеми є VIKOR. Так, як він на відміну від ELECTRE базується на значеннях критеріїв, а не на їх відношеннях та має можливість вибору стратегії в залежності від необхідних вимог до результату [10]-[11].

3.4 Опис методу розв'язання

Розв'язання складається з двох етапів: попередній та основний, які виконуються для всіх задач з множини *T*.

Попередній етап.

Для кожної задачі k формується матриця $A = a_{ij}, i = 1..m, j = 1..p$. Кожен елемент матриці $a_{ij=}c_j((y_{k1},..,y_{kr}),s_{ik})$ – це значення функціонального критерій j з параметрами $(y_{k1},..,y_{kr})$ – параметри задачі k, та s_{ik} – показник ефективності виконавця i для задач з параметрами k.

Основний етап.

Використовується метод VIKOR [9].

Крок 1. Задається число $\gamma \in [0,1]$, що визначає стратегію алгоритму.

Крок 2. Виконується нормалізацію вагових коефіцієнтів $w_j = \frac{w_j}{\sum w}$.

Крок 3. Знаходиться значення a_j^* - максимальне значення елемента, що знаходиться в j стовпці матриці A, та значення a_j^- - мінімальне значення елемента, що знаходиться в j стовпці матриці A.

Крок 4. Обчислюються значення

$$S_v = \sum_j w_j \left|a_j^* - a_{vj}\right| / |a_j^* - a_j^-|,$$

$$R_v = \max_j \{w_j \left|a_j^* - a_{vj}\right| / |a_j^* - a_j^-|\},$$

Крок 5. Обчислюються значення

$$Q_v = \gamma(S_v - S^*)/(S^- - S^*) + (1-\gamma)(R_v - R^*)/(R^- - R^*),$$

де $S^* = \min_v S_v$, $S^- = \max_v S_v$, $R^* = \min R_j$, $R^- = \max R_j$

Крок 6. Будуються три ранжування виконавців u_i, за спаданням значень Q, S, R.

Вводяться дві умови.

Вводиться позначення a' - виконавець, що в ранжуванні за Q, знаходиться на останній позиції, a''- на передостанній.

С1. Прийнятна перевага. $Q(a'') - Q(a') \geq DQ$, a'' наступна альтернатива в ранжуванні Q після a'. $DQ = 1/(m-1)$.

С2. Прийнятна стабільність. a' має найркаще значення в ранжуванні Q, та найкращі значення в ранжуваннях S та (або) R.

Якщо виконується С1 та С2, то a' є найкращою альтернативою.

Якщо не виконується лише С2, то найкращими альтернативами є a' і a''.

Якщо не виконується С1, то накращими альтернативами є ті, що задовольняють умові $Q\left(a^{(k)}\right) - Q(a') < DQ, a^{(k)}$- виконавець, що є k-тий з кінця ранжування за Q.

Приклад розв'язання.

Маємо проект з тривалістю робочого дня 6 годин. Після завершення робочого дня є множина задач, що показана на рисунках 3.1, 3.2 і 3.2.

```
  {
    "id": 1,
    "projectId": 1,
    "name": "Rule Builder",
    "content": "string",
    "estimatedStoryPoints": 10,
    "remainingStoryPoints": 1,
    "priority": 0,
    "kind": 0,
    "status": 2,
    "assignedProjectUserId": 1,
    "tags": "[\"Builder\"]"
  },
  {
    "id": 4,
    "projectId": 1,
    "name": "New panel",
    "content": "string",
    "estimatedStoryPoints": 10,
    "remainingStoryPoints": 2,
    "priority": 0,
    "kind": 0,
    "status": 2,
    "assignedProjectUserId": 3,
    "tags": "[\"Panel\"]"
  },
```


Рисунок 3.1 – Завдання та їх параметри на кінець робочого дня

```
  "id": 5,
  "projectId": 1,
  "name": "Homepage bug",
  "content": "string",
  "estimatedStoryPoints": 10,
  "remainingStoryPoints": 1,
  "priority": 0,
  "kind": 2,
  "status": 2,
  "assignedProjectUserId": 2,
  "tags": "[\"Homepage\"]"
},
{
  "id": 6,
  "projectId": 1,
  "name": "House builder",
  "content": "string",
  "estimatedStoryPoints": 10,
  "remainingStoryPoints": 10,
  "priority": 0,
  "kind": 0,
  "status": 0,
  "assignedProjectUserId": 1,
  "tags": "[\"Builder\"]"
},
{
```


Рисунок 3.2 – Завдання та їх параметри на кінець робочого дня

```
    "id": 7,
    "projectId": 1,
    "name": "Bread bug",
    "content": "string",
    "estimatedStoryPoints": 10,
    "remainingStoryPoints": 10,
    "priority": 0,
    "kind": 2,
    "status": 0,
    "assignedProjectUserId": 1,
    "tags": "[\"Bread\"]"
  },
  {
    "id": 8,
    "projectId": 1,
    "name": "Panel bug",
    "content": "string",
    "estimatedStoryPoints": 10,
    "remainingStoryPoints": 10,
    "priority": 0,
    "kind": 2,
    "status": 0,
    "assignedProjectUserId": 1,
    "tags": "[\"Panel\"]"
  }
]
```


Рисунок 3.3– Завдання та їх параметри на кінець робочого дня

Задачі з ідентифікаторами (id) 6, 7 та 8 потрібно розподілити між виконавцями з ідентифікаторами 1, 2 та 3.

Зробимо запит на розподіл з такими параметрами (рисунок 3.4).

Рисунок 3.4– Параметри запиту на створення розподілу завдань

Сформовано два критерії з ідентифікаторами вказаними в списку *«rulesIds».*

Критерій 1 – визначає наскільки виконавець ефективно робить задачі базуючись на типу задачі.

Критерій 2 – визначає наскільки виконавець ефективно робить задачі базуючись на тегу задачі.

Після застосування методу до вирішення поставленої задачі отримано такий результат (рисунки 3.5 та 3.6)

```
  "6": [
    {
      "projectUserId": 1,
      "value": 0
    },
    {
      "projectUserId": 3,
      "value": 0.8666666666666667
    },
    {
      "projectUserId": 2,
      "value": 1
    }
  ],
  "7": [
    {
      "projectUserId": 2,
      "value": 0
    },
    {
      "projectUserId": 1,
      "value": 1
    },
    {
      "projectUserId": 3,
      "value": 1
```


Рисунок 3.5 – Створений розподіл завдань

```
      "value": 0
    },
    {
      "projectUserId": 1,
      "value": 1
    },
    {
      "projectUserId": 3,
      "value": 1
    }
  ],
  "8": [
    {
      "projectUserId": 3,
      "value": 0
    },
    {
      "projectUserId": 2,
      "value": 0.6666666666666666
    },
    {
      "projectUserId": 1,
      "value": 1
    }
  ]
}
```


Рисунок 3.6 – Створений розподіл завдань

Аналіз результатів.

Проведено аналіз вхідних даних. Критерій 1 має менший ваговий коефіцієнт ніж критерій 2, отже показники ефективності, що базуються на тегу задачу мають більший вплив на результат.

Виконавець 1 має таки показники ефективності: для задач з типом 0 значення 3; для задач з тегом «Builder» значення 3.

Виконавець 2 має таки показники ефективності: для задач з типом 2 значення 3; для задач з тегом «Homepage» значення 3.

Виконавець 3 має таки показники ефективності: для задач з типом 0 значення 2; для задач з тегом «Panel» значення 2.

В результаті сформований розподіл показує ранжування виконавців, де перший виконавець повинен найефективніше виконати призначене завдання базуючись на сформованих критеріях.

Висновок до розділу

Задачу розподілу завдань між виконавцями сформульовано як задачу багатокритеріального методу прийняття рішень, до вирішення якої застосовано метод VIKOR, це дозволило при розподілі врахувати особливості виконавців і множин параметрів завдань і сформувати такий розподіл який є найбільш ефективним за сформованими критерія. Наведено схему алгоритму та приклад розв’язання задачі.

4 ПРОГРАМНЕ ТА ТЕХНІЧНЕ ЗАБЕЗПЕЧЕННЯ

4.1 Засоби розробки

Для розробки рішення було використане таке програмне забезпечення:

а) операційна систем Windows 10;

б) середовище розробки Visual Studio 2019 Community;

в) веб-браузер Mozilla Firefox;

г) платформа розробки .NET 5;

д) мова програмування C#;

е) фреймворк для розробки веб-додатків ASP.NET Core.

4.2 Вимоги до технічного забезпечення

4.2.1 Загальні вимоги

Для роботи з системою на комп’ютері повинна бути встановлена одна з операційних систем з переліку:

а) Windows 7+;

б) MacOS 10.13+;

в) Debian 9+;

г) Fedora 32+;

д) Ubuntu 16.04+.

Додаткове програмне забезпечення, необхідне для роботи системи – платформа для запуску веб додатків на базі фреймворку .NET – ASP.NET Core Runtime 5+.

Програмне забезпечення, необхідне для роботи з програмою може бути будь-яке , що дозволяє робити HTTP-запити. Наприклад, веб-браузери Mozilla Firefox, Google Chrome, чи спеціалізоване ПЗ, наприклад Postman. При використанні вбудованого інтерфейсу, достатньо тільки веб-браузера.

4.3 Архітектура програмного забезпечення

Програмне забезпечення зроблене у вигляді Web API, що має використовує сервіси для обробки запитів. Сервіси мають доступ до бази даних. Схему архітектури програмного забезпечення наведено в графічних матеріалах.

4.3.1 Діаграма класів

Діаграми класів наведено в графічних матеріалах.

4.3.2 Діаграма послідовності

Діаграму послідовності роботи з програмним забезпеченням наведено в графічних матеріалах.

4.3.3 Діаграма компонентів

Схему структурну компонентів програмного забезпечення наведено в графічних матеріалах.

4.3.4 Специфікація функцій

Опис функцій основних класів програмного забезпечення наведено в таблиці 4.1.

Таблиця 4.1 – Опис функцій основних класів програмного забезпечення

Назва функції	Опис
Клас ProjectsController	
GetAllProjects()	Отримання моделей проєктів, де авторизований користувач системи є проєктним користувачем
GetProjectById(int projectId)	Отримання моделі проєкту з ідентифікатором {projectId}

Продовження таблиці 4.1

CreateProject(ProjectCreateModel projectCreateModel)	Створення нового проєкту
DeleteProjectById(int projectId)	Видалення проєкту з ідентифікатором {projectId}
Клас ProjectUsersController	
GetProjectUsers(int projectId)	Отримання моделей проєктних користувачів на проєкті з ідентифікатором {projectId}
CreateProjectUser(int projectId, ProjectUserCreateModel projectUserCreateModel)	Додання проєктного користувача на проєкт з ідентифікатором {projectId}
DeleteProjectUserById(int projectId, int projectUserId)	Видалення проєктного користувача з ідентифікатором {projectUserId} з проєкту з ідентифікатором {projectId}
CalculateStatistics(int projectId)	Ручний виклик процесу обчислення показників ефективності проєктого користувача на проєкті з ідентифікатором{projectId}
Клас RulesController	
GetAllRules(int projectId)	Отримання моделей правил побудови критеріїв, що збережені на проєкті з ідентифікатором{projectId}
GetRuleById(int projectId, int ruleId)	Отримання моделі правила побудови критерія з ідентифікатором {ruleId}, що збережене на проєкті з ідентифікатором{projectId}

Продовження таблиці 4.1

CreateRule(int projectId, RuleCreateModel ruleCreateModel)	Збереження правила побудови критеріїв на проєкт з ідентифікатором{projectId}
DeleteRuleById(int projectId, int ruleId)	Видалення правила побудови критерія з ідентифікатором {ruleId}, що збережене на проєкті з ідентифікатором{projectId}
Клас SprintsController	
GetAllSprints(int projectId)	Отримання моделей ітерацій на проєкті з ідентифікатором{projectId}
GetSprintById(int projectId, int sprintId)	Отримання моделі ітерації з ідентифікатором {sprintId} на проєкті з ідентифікатором{projectId}
CreateSprint(int projectId, SprintCreateModel sprintCreateModel)	Додання ітерації на проєкт з ідентифікатором{projectId}
GetSprintWorkItems(int projectId, int sprintId)	Отримання моделей завдань, що призначені на ітерацію з ідентифікатором {sprintId} на проєкті з ідентифікатором{projectId}
GetSprintProjectUsers(int projectId, int sprintId)	Отримання моделей проєктних користувачів, що призначені на ітерацію з ідентифікатором {sprintId} на проєкті з ідентифікатором{projectId}
AddWorkItemToSprint(int projectId, int sprintId, int workItemId)	Призначення завдання з ідентифікатором {workItemId} на ітерацію з ідентифікатором {sprintId} на проєкті з ідентифікатором{projectId}

Продовження таблиці 4.1

AddProjectUserToSprint(int projectId, int sprintId, int projectUserId)	Призначення проєктного користувача з ідентифікатором {projectUserId} на ітерацію з ідентифікатором {sprintId} на проєкті з ідентифікатором{projectId}
RemoveWorkItemFromSprint(int projectId, int sprintId, int workItemId)	Видалення завдання з ідентифікатором {workItemId}, що призначене на ітерацію з ідентифікатором {sprintId} на проєкті з ідентифікатором{projectId}
RemoveProjectUserFromSprint(int projectId, int sprintId, int projectUserId)	Видалення проєктного користувача з ідентифікатором {projectUserId}, що призначений на ітерацію з ідентифікатором {sprintId} на проєкті з ідентифікатором{projectId}
MakeWorkDistribution(int projectId, int sprintId, StartWorkDistributionModel startWorkDistributionModel)	Отримання моделі розподілу завдань між проєктними користувачами, де завдання і користивачі призначені на ітерацію з ідентифікатором {sprintId} на проєкті з ідентифікатором{projectId} з заданими параметрами
Клас UsersController	
GetAllUsers()	Отримання моделей користувачів системи
RegisterUser(UserRegisterModel userRegisterModel)	Реєстрація користувача системи

Продовження таблиці 4.1

LoginUser(UserSignModel userSignModel)	Отримання токену авторизації користувача системи
Клас WorkItemChangesController	
GetWorkItemChanges(int projectId, int workItemId)	Отримання моделей даних про зміну властивостей завдання з ідентифікатором {workItemId} на проєкті з ідентифікатором {projectId}
Клас WorkItemsController	
GetAllProjectWorkItems(int projectId)	Отримання моделей завдань на проєкті з ідентифікатором {projectId}
GetAllWorkItems(int projectId)	Отримання моделей призначених завдань авторизованого користувача системи на проєкті з ідентифікатором {projectId}
GetWorkItemById(int projectId, int workItemId)	Отримання завдання з ідентифікатором {workItemId} на проєкті з ідентифікатором {projectId}
CreateWorkItem(int projectId, WorkItemCreateModel workItemCreateModel)	Додання завдання на проєкт з ідентифікатором {projectId}
UpdateWorkItem(int projectId, int workItemId, WorkItemPutModel workItemPutModel)	Зміна завдання з ідентифікатором {workItemId} на проєкті з ідентифікатором {projectId}
DeleteWorkItemById(int projectId, int workItemId)	Видалення завдання з ідентифікатором {workItemId} на проєкті з ідентифікатором {projectId}

Продовження таблиці 4.1

Клас ProjectUserStatisticsCalculationService	
Calculate(int projectUserId, TimeSpan workItemsChangesTimeRange)	Обчислення показників ефективності проєктного користувача з ідентифікатором {projectUserId} базуючись змінах його завдань в періоді {workItemsChangesTimeRange}
Клас SprintWorkDistributionsCalculationService	
Calculate(int sprintId, IReadOnlyList<int> rulesIds, IReadOnlyList<double> weights, double vCoef)	Отримання розподілу завдань між проєктними користувачами, де завдання і користивачі призначені на ітерацію з ідентифікатором {sprintId} на проєкті з ідентифікатором{projectId} з заданими параметрами
Клас VIKOR	
WorkOut(IMatrix<double> alternativesCriteriaMatrix, IReadOnlyList<TargetFuncKind> targetFuncKinds, IReadOnlyList<double> weights, double vCoef)	Обчислення ранжування альтернатив за допомогою метода VIKOR

Висновок до розділу

В даному розділі було описано архітектуру програмного забезпечення, наведені діаграми класів, діяльності та компонентів, наведено специфікацію функцій. Сформульовано вимоги до програмного та технічного забезпечення.

5 ТЕХНОЛОГІЧНИЙ РОЗДІЛ

5.1 Керівництво користувача

Для користування розробленим програмним забезпеченням можна використовувати графічний інтерфейс API за шляхом «*swagger/index.html*».

Відповідно до технічного завдання реалізовані наступні функціональні можливості.

Надано можливість реєстрації та авторизації системних користувачів (рисунок 5.1).

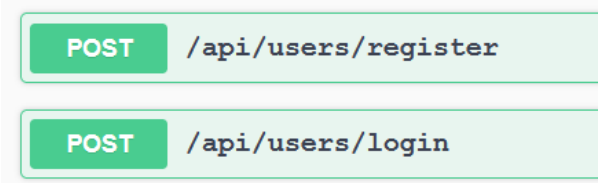


Рисунок 5.1 – Кінцеві точки API для реєстрації та авторизації

Надано можливість базового керування проєктами, що включає можливості створення, перегляд та видалення проєктів. (рисунки 5.2 та 5.3).

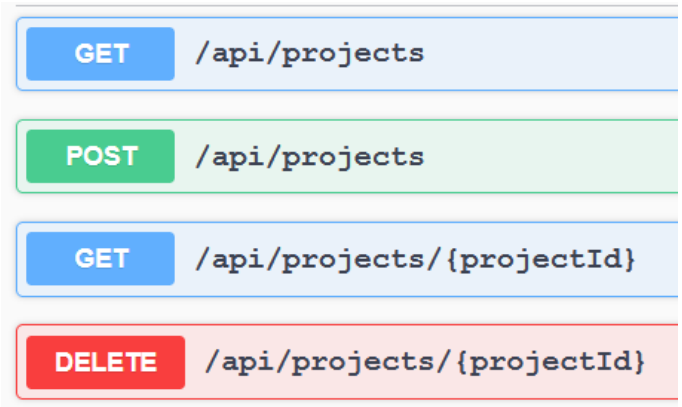


Рисунок 5.2 – Кінцеві точки API для керування проєктами

Рисунок 5.3 – Кінцеві точки API для керування проєктними користувачами (виконавцями)

Надана можливість базового керування завданнями, що включає можливості створення, перегляд, зміну та видалення завдань (рисунок 5.4).

Рисунок 5.4 – Кінцеві точки API для керування завданнями

Надана можливість базового керування ітераціями, що включає можливості створення, перегляд, видалення ітерацій, призначення та видалення з ітерацій завдань та виконавців (рисунок 5.5).

GET /api/projects/{projectId}/sprints
POST /api/projects/{projectId}/sprints
GET /api/projects/{projectId}/sprints/{sprintId}
GET /api/projects/{projectId}/sprints/{sprintId}/work-items
GET /api/projects/{projectId}/sprints/{sprintId}/project-users
POST /api/projects/{projectId}/sprints/{sprintId}/work-items/{workItemId}
DELETE /api/projects/{projectId}/sprints/{sprintId}/work-items/{workItemId}
POST /api/projects/{projectId}/sprints/{sprintId}/project-users/{projectUserId}
DELETE /api/projects/{projectId}/sprints/{sprintId}/project-users/{projectUserId}

Рисунок 5.5 – Кінцеві точки API для керування ітераціями

Нада можливість керування правилами побудови критеріїв, що включає можливості створення, перегляду та видалення правил (рисунок 5.6).

GET /api/projects/{projectId}/rules
POST /api/projects/{projectId}/rules
GET /api/projects/{projectId}/rules/{ruleId}
DELETE /api/projects/{projectId}/rules/{ruleId}

Рисунок 5.6 – Кінцеві точки API для керування правилами побудови критеріїв

Надано можливість ручного запуску аналізу показників ефективності для виконавця. (рисунок 5.7).

GET /api/projects/{projectId}/project-users/calculate-statistics

Рисунок 5.7 – Кінцеві точка API для запуску аналізу показників ефективності виконавця

Надано можливість створення розподілу завдань між виконавцями з заданими параметрами (рисунок 5.8).

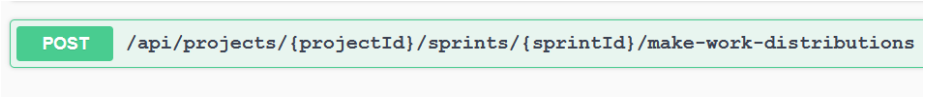


Рисунок 5.8 – Кінцеві точка API для створення розподілу завдань між виконавцями

5.2 Випробування програмного продукту

Для тестування програмного забезпечення проведено кожна кінцева точка API, що відповідає за функціональну можливість, була викликана з тестовими даними, для перевірки коректності результату виклику.

Проведені тести:

а) реєстрація користувача;

б) авторизація користувача;

в) створення проєкту;

г) призначення виконавця на проєкт;

д) створення завдання на проєкт;

е) зміна завдання;

ж) створення ітерації;

з) призначення виконавців на ітерацію;

и) призначення завдань на ітерацію;

к) створення розподілу завдань між виконавцями.

5.2.1 Мета випробувань

Метою випробувань являється перевірка відповідності функцій інформаційної системи з розподілу завдань між виконавцями вимогам технічного завдання.

5.2.2 Загальні положення

Випробування проводяться на основі наступних документів:

– ГОСТ 34.603–92. Інформаційна технологія. Види випробувань автоматизованих систем;
– ГОСТ РД 50-34.698-90. Автоматизовані системи вимог до змісту документів.

5.2.3 Результати випробувань

Приклад реєстрації користувача (рисунки 5.9 та 5.10).

Рисунок 5.9 – Запит реєстрації користувача

Рисунок 5.10 – Результат запиту реєстрації користувача

Приклад отримання токену авторизації користувачем (рисунки 5.11 та 5.12).

Рисунок 5.11 – Запит авторизації користувача

Рисунок 5.12 – Результат запиту авторизації користувача

Приклад створення проєкту (рисунки 5.13 та 5.14).

Рисунок 5.13 – Запит створення проєкту

```
{
  "id": 1,
  "name": "Project1",
  "description": "My first project",
  "timeZoneId": null,
  "dayStoryPoints": 6
}
```


Рисунок 5.14 – Результат запиту створення проєкту

Приклад призначення виконавця на проєкт (рисунки 5.15-5.20).

Рисунок 5.15 – Запит отримання користувачів системи


```
[
  {
    "id": "21d4a2f9-6988-48ff-bf7d-b457ccbe0945",
    "userName": "User1",
    "email": "user1@example.com",
    "kind": 0
  },
  {
    "id": "6cc6c2ce-6338-4f3e-b268-6d84fa5c414f",
    "userName": "User2",
    "email": "user2@example.com",
    "kind": 0
  }
]
```


Рисунок 5.16 – Результат запиту отримання користувачів системи

Рисунок 5.17 – Запит призначення виконавця на проєкт

Рисунок 5.18 – Результат запиту призначення виконавця на проєкт

Рисунок 5.19 – Запит отримання виконавців на проєкті

```
[
  {
    "id": 1,
    "userId": "21d4a2f9-6988-48ff-bf7d-b457ccbe0945",
    "projectId": 1,
    "kind": 2
  },
  {
    "id": 2,
    "userId": "6cc6c2ce-6338-4f3e-b268-6d84fa5c414f",
    "projectId": 1,
    "kind": 0
  }
]
```


Рисунок 5.20 – Результат запиту отримання виконавців на проєкті

Приклад створення завдання на проєкт (рисунки 5.21 та 5.22).

Рисунок 5.21 – Запит створення завдання на проєкт

```
{
  "id": 1,
  "projectId": 1,
  "name": "Image bug",
  "content": "The image on homepage is cut off",
  "estimatedStoryPoints": 10,
  "remainingStoryPoints": 10,
  "priority": 0,
  "kind": 2,
  "status": 0,
  "assignedProjectUserId": 1,
  "tags": "[\"Image\"]"
}
```


Рисунок 5.22 – Результат запиту створення завдання на проєкт

Приклад зміни завдання (рисунки 5.23-5.25).

Рисунок 5.23 – Запит зміни завдання

Рисунок 5.24 – Запит отримання призначених завдань

Рисунок 5.25 – Результат запиту отримання призначених завдань

Приклад створення ітерації (рисунки 5.26 та 5.27).

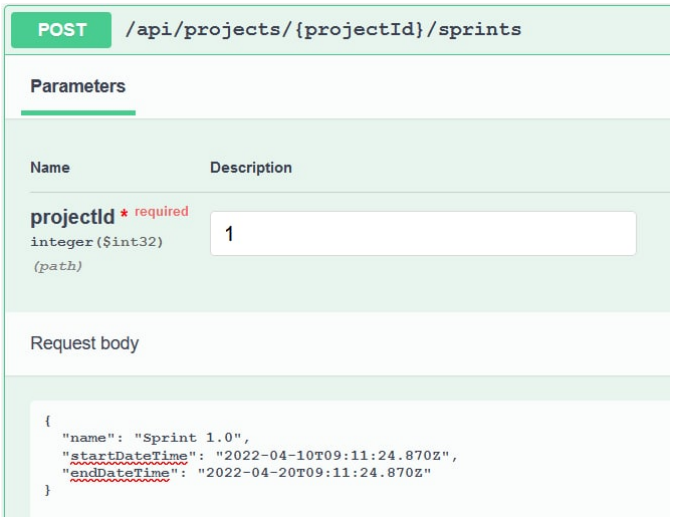


Рисунок 5.26 – Запит створення ітерації

```
{
  "id": 1,
  "projectId": 1,
  "name": "Sprint 1.0",
  "startDateTime": "2022-04-10T09:11:24.87Z",
  "endDateTime": "2022-04-20T09:11:24.87Z"
}
```


Рисунок 5.27 – Результат запиту створення ітерації

Приклад призначення виконавців на ітерацію (рисунки 5.28-5.31).

Рисунок 5.28 – Запит призначення першого виконавця на ітерацію

Рисунок 5.29 – Запит призначення другого виконавця на ітерацію

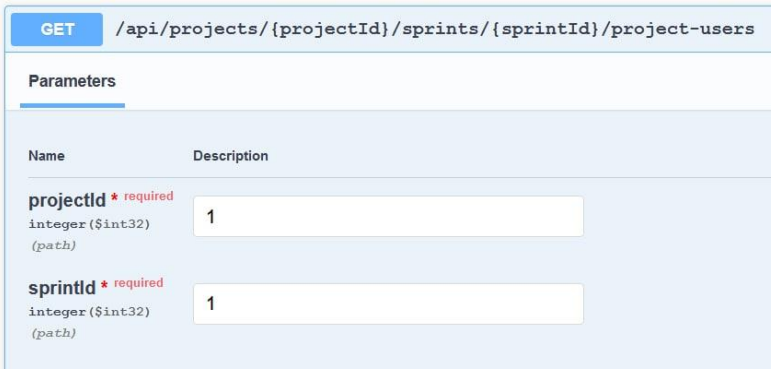


Рисунок 5.30 – Запит отримання призначених виконавців на ітерацію

Рисунок 5.31 – Результат запиту отримання призначених виконавців на ітерацію

Приклад призначення завдань на ітерацію (рисунки 5.32-5.34).

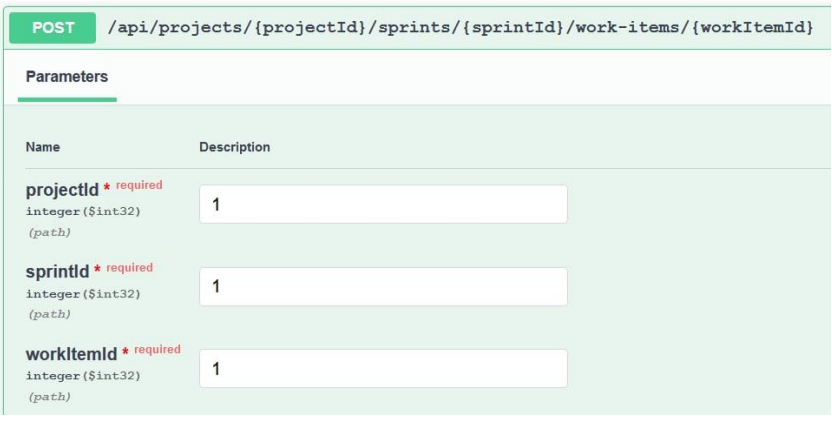


Рисунок 5.32 – Запит призначення завдання на ітерацію

Рисунок 5.33 – Запит отримання призначених завдань на ітерацію


```
[
  {
    "id": 1,
    "projectId": 1,
    "name": "Image bug",
    "content": "The image on homepage is cut off",
    "estimatedStoryPoints": 10,
    "remainingStoryPoints": 2,
    "priority": 0,
    "kind": 2,
    "status": 2,
    "assignedProjectUserId": 1,
    "tags": "[\"Image\"]"
  }
]
```


Рисунок 5.34 – Результат запиту отримання призначених завдань на ітерацію

Приклад створення розподілу завдань між виконавцями (рисунки 5.35 та 5.36).

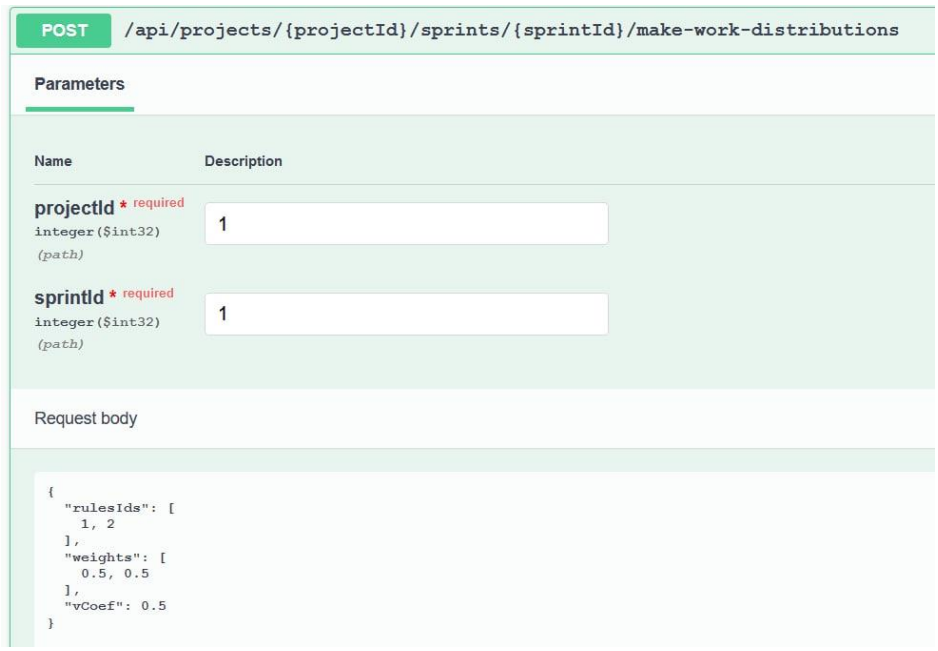


Рисунок 5.35 – Запит створення розподілу завдань між виконавцями

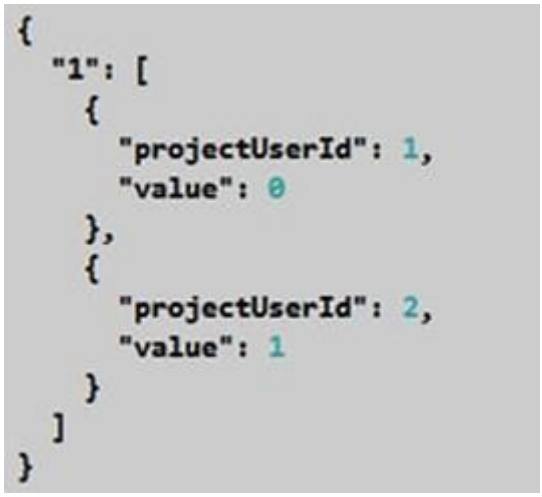


Рисунок 5.36 – Результат запиту створення розподілу завдань між виконавцями

Висновок до розділу

В даному розділі розроблено керівництво користувача в якому описано функціональні можливості, що доступні для користувача, описані випробування, що були проведені, наведено результати випробувань.

ЗАГАЛЬНІ ВИСНОВКИ

В даній роботі, було проведено дослідження сфери ІТ-проєктів та визначено актуальну проблему розподілу завдань між виконавцями. Для вирішення поставленої проблеми було зроблено постановку задачі. Проведено аналіз існуючих аналогів та виявлено відсутність у них необхідного функціоналу. Визначено вхідні та вихідні дані, що необхідні для вирішення проблеми, а саме дані про проєкт, завдання на проєкті, призначених виконавців та статистика роботи виконавців над завданнями. Базуючись на цих даних було наведено змістовну та математичну постановки задачі. Показано що дана задача є задачею багатокритеріального вибору, обгрунтовано вибір методу її розв’язання. Наведено схему метода VIKOR та показано приклади розв’язання задачі. Розроблено відповідне програмне забезпечення, що використовує вибраний метод для вирішення задачі підтримки процесу розподілу завдань між виконавцями. Наведено програмне та технічне забезпечення. Проведено випробування функціоналу програмного забезпечення. Результати випробувань показали, що функціональність продукту є коректною.

Розроблене програмне забезпечення рекомендовано використовувати при плануванні розподілу завдань між виконавцями ІТ-проєктів.

ПЕРЕЛІК ПОСИЛАНЬ

1. Scrum [Електронний ресурс] – Режим доступу до ресурсу: https://www.scrum.org/resources/scrum-guide

2. Ноженко Я.І. Інформаційна система з розподілу завдань між виконавцями / Я.І. Ноженко // Матеріали VI всеукраїнської науково-практичної конференції молодих вчених та студентів «Інформаційні системи та технології управління» (ІСТУ-2021) – м. Київ.: НТУУ «КПІ ім. Ігоря Сікорського», 22-23 квітня 2021 р.

3. Atlassian Jira [Електронний ресурс] – Режим доступу до ресурсу: https://www.atlassian.com/software/jira

4. Microsoft Azure DevOps [Електронний ресурс] – Режим доступу до ресурсу: https://azure.microsoft.com/en-us/services/devops

5. JetBrains YouTrack [Електронний ресурс] – Режим доступу до ресурсу: https://www.jetbrains.com/youtrack

6. Gwo-Hshiung Tzeng, Jih-Jeng Huang. Multiple Attribute Decision Making Methods and applications. Taylor & Francis Group, LCC, 2011. 335 p.

7. Блюмин С.Л., Шуйкова И.А. Модели и методы принятия решений вусловиях неопределенности.—Липецк: ЛЭГИ, 2000.—139 с.

8. Нечеткие множества в моделях управления и искусственного интеллекта/Под ред. Д.А.Поспелова.–М.: Наука. Гл. ред. физ.-мат. лит., 1986.–312с.

9. Орловский С.А. Проблемы принятия решений при нечеткой исходнойинформации.—М.: Наука.—1981.—194 с.

10. Теорія прийняття рішень [Електронний ресурс]: навч. посіб. для студ. спеціальності 126 «Інформаційні системи та технології» та спеціальності 121 «Інженерія програмного забезпечення» / КПІ ім. Ігоря Сікорського; уклад.: О.С.Жураковська. –Електронні текстові дані (1 файл: 2.7 Мбайт). –Київ: КПІ ім. Ігоря Сікорського, 2020. –99 с.

11. В. Демидовский. Сравнительный анализ методов многокритериального принятия решений: ELECTRE, TOPSIS и ML-LDMA. Национальный Исследовательский Университет Высшая Школа Экономики Нижний Новгород, Россия, 2020, 237 с.

Додаток А

Тексти програмного коду

Інформаційна система з розподілу завдань між виконавцями

(Найменування програми (документа))

DVD-R

(Вид носія даних)

15 арк, 32 Кб

(Обсяг програми (документа) , арк., Кб)

Київ – 2021 року

```
namespace Diplom.Api.Services
{
   public class ProjectUserStatisticsCalculationService
   {
      private readonly ILogger<ProjectUserStatisticsCalculationService> _logger;
      private readonly ApplicationDbContext _db;

      public
ProjectUserStatisticsCalculationService(ILogger<ProjectUserStatisticsCalculationService>
logger, ApplicationDbContext db)
      {
         _logger = logger;
         _db = db;
      }

      public async Task Calculate(int projectUserId, TimeSpan workItemsChangesTimeRange)
      {
         var calculationGuid = Guid.NewGuid();

         _logger.LogInformation($"{DateTime.UtcNow:dd/MM/yyyy}                Calculation
({calculationGuid}) is started on projectUserId {{{projectUserId}}}");

         var workItemsChangesDateTime = DateTime.UtcNow - workItemsChangesTimeRange;

         var projectUser =
            await _db.ProjectUsers
               .Include(x => x.Project)
               .FirstOrDefaultAsync(x => x.Id == projectUserId);

         if (projectUser is null)
         {
            _logger.LogWarning($"{DateTime.UtcNow:dd/MM/yyyy}                Calculation
({calculationGuid}) is aborted (project user is not found)");

            return;
         }

         if (projectUser.IsStatisticsCalculationRunning)
         {
            _logger.LogWarning($"{DateTime.UtcNow:dd/MM/yyyy}                Calculation
({calculationGuid}) is aborted (calculation for the project user is running already)");

            return;
         }

         projectUser.IsStatisticsCalculationRunning = true;

         await _db.SaveChangesAsync();
```

```
        var timeZone =
            projectUser.Project.TimeZoneId is null ?
                TimeZoneInfo.Utc :
                TimeZoneInfo.FindSystemTimeZoneById(projectUser.Project.TimeZoneId);

        var storyPointsPerDay = projectUser.Project.DayStoryPoints;

        var workItemsChanges =
            await _db.WorkItemChanges
                .Where(x => x.NewAssignedProjectUserId == projectUserId) //&& x.DateTime >
workItemsChangesDateTime)
                .ToListAsync();

        if (workItemsChanges.Count == 0)
        {
            return;
        }

        var orderedWorkItemsChanges =
            workItemsChanges
                .Select
                (
                    x =>
                    {
                        x.DateTime = timeZone.ConvertFromUtc(x.DateTime);
                        return x;
                    }
                )
                .OrderBy(x => x.DateTime)
                .ToList();

        var workItemDayProjectUserStatisticsGroupedByWorkItem =
            orderedWorkItemsChanges
                .GroupBy(x => x.WorkItemId)
                .Select
                (
                    x =>
                        new
                        {
                            WorkItem = _db.WorkItems.First(y => y.Id == x.Key),
                            Statistics =
                                x
                                    .GroupBy(y => y.DateTime.Date)
                                    .Aggregate
                                    (
                                        new Dictionary<StatisticKind, double>(),
                                        (statistics, dayWorkItemChanges) =>
                                        {
                                            var (first, last) = dayWorkItemChanges.GetFirstAndLast();

                                            if (first is null)
```

```
                {
                    return statistics;
                }

                var workItemDayUnderEffortDeviationValue =
CalculateWorkItemDayUnderEffortDeviation(first, last, storyPointsPerDay);
                var workItemDayOverEffortDeviationValue =
CalculateWorkItemDayOverEffortDeviation(first, last, storyPointsPerDay);
                var workItemDayEffortDeviationValue =
workItemDayUnderEffortDeviationValue - workItemDayOverEffortDeviationValue;

                statistics.ApplyOrSet
                (
                    StatisticKind.UnderEffortDeviation,
                    workItemDayUnderEffortDeviationValue, (value, foundValue)
=> value + foundValue
                );

                statistics.ApplyOrSet
                (
                    StatisticKind.OverEffortDeviation,
                    workItemDayOverEffortDeviationValue, (value, foundValue)
=> value + foundValue
                );

                statistics.ApplyOrSet
                (
                    StatisticKind.EffortDeviation,
                    workItemDayEffortDeviationValue, (value, foundValue) =>
value + foundValue
                );

                return statistics;
            }
        )
    }
)
.ToList();

var workItemDayProjectUserStatisticEntities = new
List<WorkItemDayProjectUserStatisticEntity>();

try
{
    workItemDayProjectUserStatisticEntities =
        workItemDayProjectUserStatisticsGroupedByWorkItem
            .SelectMany
            (
                x =>
                    x.Statistics
                        .Select
```

```
                    (
                        y =>
                            new WorkItemDayProjectUserStatisticEntity
                            {
                                ProjectUserId = projectUserId,
                                WorkItemId = x.WorkItem.Id,
                                StatisticKind = y.Key,
                                Value = y.Value
                            }
                    )
                )
                .ToList();
        }
        catch (Exception ex)
        {
            _logger.LogError(ex, "workItemDayProjectUserStatisticEntities calculation");
        }

        var workItemKindProjectUserStatisticEntities = new
List<WorkItemKindProjectUserStatisticEntity>();

        try
        {
            workItemKindProjectUserStatisticEntities =
                workItemDayProjectUserStatisticsGroupedByWorkItem
                    .GroupBy(x => x.WorkItem.Kind, x => x.Statistics)
                    .SelectMany
                    (
                        x =>
                            x
                                .Aggregate
                                (
                                    new Dictionary<StatisticKind, double>(),
                                    (statistics, itemStatistics) =>
                                    {
                                        foreach (var (statisticKind, statisticValue) in itemStatistics)
                                        {
                                            statistics.ApplyOrSet(statisticKind, statisticValue, (value,
foundValue) => value + foundValue);
                                        }

                                        return statistics;
                                    }
                                )
                                .Select
                                (
                                    y =>
                                        new WorkItemKindProjectUserStatisticEntity
                                        {
                                            ProjectUserId = projectUserId,
                                            WorkItemKind = x.Key,
```

```
                        StatisticKind = y.Key,
                        Value = y.Value
                    }
                )
            )
            .ToList();
        }
        catch (Exception ex)
        {
            _logger.LogError(ex, "workItemKindProjectUserStatisticEntities calculation");
        }

        var workItemTagProjectUserStatisticEntities = new
List<WorkItemTagProjectUserStatisticEntity>();

        try
        {
            workItemTagProjectUserStatisticEntities =
                workItemDayProjectUserStatisticsGroupedByWorkItem
                .Where(x => !string.IsNullOrEmpty(x.WorkItem.Tags))
                .SelectMany
                (
                    x =>
                        x.WorkItem.Tags.Deserialize<string[]>()
                        .Select
                        (
                            y =>
                                new
                                {
                                    Tag = y,
                                    Item = x
                                }
                        )
                )
                .GroupBy(x => x.Tag, x => x.Item.Statistics)
                .SelectMany
                (
                    x =>
                        x
                        .Aggregate
                        (
                            new Dictionary<StatisticKind, double>(),
                            (statistics, itemStatistics) =>
                            {
                                foreach (var (statisticKind, statisticValue) in itemStatistics)
                                {
                                    statistics.ApplyOrSet(statisticKind, statisticValue, (value,
foundValue) => value + foundValue);
                                }

                                return statistics;
```

```
                    }
                )
                .Select
                (
                    y =>
                        new WorkItemTagProjectUserStatisticEntity
                        {
                            ProjectUserId = projectUserId,
                            Tag = x.Key,
                            StatisticKind = y.Key,
                            Value = y.Value
                        }
                )
            )
            .ToList();
        }
        catch (Exception ex)
        {
            _logger.LogError(ex, "workItemTagProjectUserStatisticEntities calculation");
        }

        await
_db.ProjectUserWorkItemDayStatistics.AddRangeAsync(workItemDayProjectUserStatisticEntit
ies);
        await
_db.ProjectUserWorkItemKindStatistics.AddRangeAsync(workItemKindProjectUserStatisticEnt
ities);
        await
_db.ProjectUserWorkItemTagStatistics.AddRangeAsync(workItemTagProjectUserStatisticEntiti
es);

        await _db.SaveChangesAsync();

        projectUser.IsStatisticsCalculationRunning = false;

        await _db.SaveChangesAsync();

        _logger.LogInformation($"{DateTime.UtcNow:dd/MM/yyyy}                Calculation
({calculationGuid}) is finished");
    }

    private static double CalculateWorkItemDayUnderEffortDeviation(WorkItemChangeEntity
firstWorkItemChange, WorkItemChangeEntity lastWorkItemChange, int storyPointsPerDay)
    {
        var storyPointsDiff = firstWorkItemChange.OldRemainingStoryPoints -
lastWorkItemChange.NewRemainingStoryPoints;

        var dayStoryPointsQuota =
            firstWorkItemChange.OldRemainingStoryPoints > storyPointsPerDay ?
                storyPointsPerDay :
                firstWorkItemChange.OldRemainingStoryPoints;
```

```
            var diff = storyPointsDiff - dayStoryPointsQuota;

            return diff < 0 ? 0 : diff;
        }

        private static double CalculateWorkItemDayOverEffortDeviation(WorkItemChangeEntity
firstWorkItemChange, WorkItemChangeEntity lastWorkItemChange, int storyPointsPerDay)
        {
            var storyPointsDiff = firstWorkItemChange.OldRemainingStoryPoints -
lastWorkItemChange.NewRemainingStoryPoints;

            var dayStoryPointsQuota =
                firstWorkItemChange.OldRemainingStoryPoints > storyPointsPerDay ?
                    storyPointsPerDay :
                    firstWorkItemChange.OldRemainingStoryPoints;

            var diff = dayStoryPointsQuota - storyPointsDiff;

            return diff < 0 ? 0 : diff;
        }
    }
}

namespace Diplom.Api.Services
{
    public class SprintWorkDistributionsCalculationService
    {
        private readonly ILogger<SprintWorkDistributionsCalculationService> _logger;
        private readonly ApplicationDbContext _db;

        public
SprintWorkDistributionsCalculationService(ILogger<SprintWorkDistributionsCalculationServic
e> logger, ApplicationDbContext db)
        {
            _logger = logger;
            _db = db;
        }

        public async Task<Dictionary<int, List<Alternative>>> Calculate(int sprintId,
IReadOnlyList<int> rulesIds, IReadOnlyList<double> weights, double vCoef)
        {
            var calculationGuid = Guid.NewGuid();

            _logger.LogInformation($"{DateTime.UtcNow:dd/MM/yyyy} Calculation
({calculationGuid}) is started on sprintId {{{sprintId}}} with rulesIds {{{string.Join(',',
rulesIds)}}} and weights {{{string.Join(',', weights)}}}");

            if (vCoef is < 0 or > 1)
            {
```

```
            _logger.LogWarning($"{DateTime.UtcNow:dd/MM/yyyy}                Calculation
({calculationGuid}) is aborted (invalid vCoef)");

            return null;
        }

        if (rulesIds.Count == 0 || weights.Count == 0)
        {
            _logger.LogWarning($"{DateTime.UtcNow:dd/MM/yyyy}                Calculation
({calculationGuid}) is aborted (rulesIds count or weights count equals 0)");

            return null;
        }

        if (rulesIds.Count != weights.Count)
        {
            _logger.LogWarning($"{DateTime.UtcNow:dd/MM/yyyy}                Calculation
({calculationGuid}) is aborted (rulesIds count not equals weights count)");

            return null;
        }

        var sprint =
            await _db.Sprints
                .Include(x => x.ProjectUsers)
                .Include(x => x.WorkItems)
                .FirstOrDefaultAsync(x => x.Id == sprintId);

        if (sprint is null)
        {
            _logger.LogWarning($"{DateTime.UtcNow:dd/MM/yyyy}                Calculation
({calculationGuid}) is aborted (sprint is not found)");

            return null;
        }

        var rules = new List<IRule>();

        foreach (var ruleId in rulesIds)
        {
            var ruleEntity = await _db.Rules.FirstOrDefaultAsync(x => x.Id == ruleId);

            // ReSharper disable once InvertIf
            if (ruleEntity is null)
            {
                _logger.LogWarning($"{DateTime.UtcNow:dd/MM/yyyy}                Calculation
({calculationGuid}) is aborted (rule {{{ruleId}}} is not found)");

                return null;
            }
```

```
            rules.Add(RuleHelper.DeserializeRule(ruleEntity.Rule));
        }

        var rulesFuncs =
            Enumerable.Range(0, rulesIds.Count)
                .Select(i => rules[i].Build())
                .ToList();

        var projectUsers = sprint.ProjectUsers.ToList();
        var workItems = sprint.WorkItems.ToList();

        foreach (var projectUser in projectUsers)
        {
            await _db.Entry(projectUser).Collection(x => x.WorkItemDayStatistics).LoadAsync();
            await _db.Entry(projectUser).Collection(x => x.WorkItemKindStatistics).LoadAsync();
            await _db.Entry(projectUser).Collection(x => x.WorkItemTagStatistics).LoadAsync();
        }

        var targetFuncsKinds = Enumerable.Repeat(TargetFuncKind.Max, rulesIds.Count).ToList();

        var results = new Dictionary<int, List<Alternative>>();

        foreach (var workItem in workItems)
        {
            var matrix = new CalculateIndexMatrix<double>(projectUsers.Count, rulesFuncs.Count);

            for (var rowIndex = 0; rowIndex < projectUsers.Count; ++rowIndex)
            {
                var projectUser = projectUsers[rowIndex];

                var ruleInput =
                    new RuleInput
                    {
                        WorkItem = workItem,
                        ProjectUser = projectUser,
                        ProjectUserWorkItemDayStatistics =
                            projectUser.WorkItemDayStatistics
                                .GroupBy(x => x.Kind)
                                .Select(x => x.Last())
                                .ToDictionary
                                (
                                    x => x.StatisticKind,
                                    x => x.Value
                                ),
                        ProjectUserWorkItemKindStatistics =
                            projectUser.WorkItemKindStatistics
                                .GroupBy(x => x.Kind)
                                .Select(x => x.Last())
```

```
                    .GroupBy(x => x.WorkItemKind)
                    .ToDictionary
                    (
                        x => x.Key,
                        x => (IReadOnlyDictionary<StatisticKind, double>)
                            x.ToDictionary
                            (
                                y => y.StatisticKind,
                                y => y.Value
                            )
                    ),
                ProjectUserWorkItemTagStatistics =
                    projectUser.WorkItemTagStatistics
                        .GroupBy(x => x.Kind)
                        .Select(x => x.Last())
                        .GroupBy(x => x.Tag)
                        .ToDictionary
                        (
                            x => x.Key,
                            x => (IReadOnlyDictionary<StatisticKind, double>)
                                x.ToDictionary
                                (
                                    y => y.StatisticKind,
                                    y => y.Value
                                )
                        )
                };

            for (var columnIndex = 0; columnIndex < rulesFuncs.Count; ++columnIndex)
            {
                var ruleFunc = rulesFuncs[columnIndex];

                matrix[rowIndex, columnIndex] = ruleFunc(ruleInput);
            }
        }

        var (bestAlternatives, otherAlternatives) = VIKOR.WorkOut(matrix, targetFuncsKinds,
weights, vCoef);

        var alternatives =
            bestAlternatives.Union(otherAlternatives)
                .Select
                (
                    x =>
                        new Alternative
                        {
                            ProjectUserId = projectUsers[x.AlternativeIndex].Id,
                            Value = x.Value
                        }
                )
                .OrderBy(x => x.Value)
```

```
                .ToList();

            results.Add(workItem.Id, alternatives);
        }

        _logger.LogInformation($"{DateTime.UtcNow:dd/MM/yyyy} Calculation ({calculationGuid}) is finished");

        return results;
    }

    public class Alternative
    {
        public int ProjectUserId { get; set; }

        public double Value { get; set; }
    }
  }
}

namespace YaMath.Algorithms.DecisionTheory
{
  // ReSharper disable once InconsistentNaming
  public static class VIKOR
  {
    private static readonly Dictionary<TargetFuncKind, ITargetFunc> TargetFuncs;

    static VIKOR()
    {
      TargetFuncs = new Dictionary<TargetFuncKind, ITargetFunc>
      {
        {TargetFuncKind.Max, new MaxTargetFunc()},
        {TargetFuncKind.Min, new MinTargetFunc()}
      };
    }

    public static (List<AlternativeValue> BestAlternatives, List<AlternativeValue> OtherAlternatives) WorkOut
    (
      IMatrix<double> alternativesCriteriaMatrix,
      IReadOnlyList<TargetFuncKind> targetFuncKinds,
      IReadOnlyList<double> weights,
      double vCoef = 0.5
    )
    {
      var normalizedWeights = Normalizations.NormalizeValues(weights).ToArray();
      var normalizedMatrix = NormalizeAlternativesCriteriaMatrix(alternativesCriteriaMatrix, targetFuncKinds, normalizedWeights);
      var (sDistances, rDistances, qValues) = GetSRQ(normalizedMatrix, vCoef);

      var dq = 1d / (alternativesCriteriaMatrix.ColumnsCount - 1);
```

```
        var best = qValues[0];
        var candidate = qValues[1];

        var isAcceptableAdvantageSatisfied = CheckAcceptableAdvantage(dq, best.Value, candidate.Value);

        var isAcceptableStabilitySatisfied =
            sDistances[0].AlternativeIndex == best.AlternativeIndex ||
            rDistances[0].AlternativeIndex == best.AlternativeIndex;

        var bestAlternative = new List<AlternativeValue> { best };
        var otherAlternatives = new List<AlternativeValue>();

        if (isAcceptableAdvantageSatisfied && isAcceptableStabilitySatisfied)
        {
            otherAlternatives.AddRange(qValues[1..]);

            return (bestAlternative, otherAlternatives);
        }

        bestAlternative.Add(candidate);

        if (!isAcceptableStabilitySatisfied)
        {
            if (qValues.Length > 2)
            {
                otherAlternatives.AddRange(qValues[2..]);
            }

            return (bestAlternative, otherAlternatives);
        }

        for (var i = 2; i < qValues.Length; ++i)
        {
            candidate = qValues[i];

            if (!CheckAcceptableAdvantage(dq, best.Value, candidate.Value))
            {
                bestAlternative.Add(candidate);
            }
            else
            {
                otherAlternatives.AddRange(qValues[i..]);

                break;
            }
        }

        return (bestAlternative, otherAlternatives);
    }
```

```
        private static double GetAlternativesBestCriteriaValue(IEnumerable<double>
alternativesCriteriaValues, TargetFuncKind targetFuncKind) =>
            TargetFuncs[targetFuncKind].GetBestValue(alternativesCriteriaValues);

        private static double GetAlternativesWorstCriteriaValue(IEnumerable<double>
alternativesCriteriaValues, TargetFuncKind targetFuncKind) =>
            TargetFuncs[targetFuncKind].GetWorstValue(alternativesCriteriaValues);

        // ReSharper disable once SuggestBaseTypeForParameter
        private static IMatrix<double> NormalizeAlternativesCriteriaMatrix(IMatrix<double>
matrix, IReadOnlyList<TargetFuncKind> targetFuncKinds, IReadOnlyList<double>
normalizedWeights)
        {
            var resultMatrix = (IMatrix<double>) matrix.Clone();

            for (var criteriaIndex = 0; criteriaIndex < resultMatrix.ColumnsCount; ++criteriaIndex)
            {
                var criteriaAlternatives = resultMatrix.GetColumnRows(criteriaIndex);
                var bestValue = GetAlternativesBestCriteriaValue(criteriaAlternatives,
targetFuncKinds[criteriaIndex]);
                var worstValue = GetAlternativesWorstCriteriaValue(criteriaAlternatives,
targetFuncKinds[criteriaIndex]);
                var delta = bestValue - worstValue;

                if (delta == 0) delta = 1;

                for (var alternativeIndex = 0; alternativeIndex < resultMatrix.RowsCount;
++alternativeIndex)
                {
                    var value = resultMatrix[alternativeIndex, criteriaIndex];

                    resultMatrix[alternativeIndex, criteriaIndex] =
                        normalizedWeights[criteriaIndex] * Math.Abs(bestValue - value) /
Math.Abs(delta);
                }
            }

            return resultMatrix;
        }

        // ReSharper disable once ReturnTypeCanBeEnumerable.Local
        private static (AlternativeValue[] SDistances, AlternativeValue[] RDistances,
AlternativeValue[] QValues) GetSRQ(IMatrix<double> alternativesCriteriaMatrix, double
vCoef)
        {
            var sDistances = new AlternativeValue[alternativesCriteriaMatrix.RowsCount];
            var rDistances = new AlternativeValue[alternativesCriteriaMatrix.RowsCount];

            var sDistanceMax = 0d;
            var sDistanceMin = double.MaxValue;
```

```
        var rDistanceMax = 0d;
        var rDistanceMin = double.MaxValue;

        for (var alternativeIndex = 0; alternativeIndex < alternativesCriteriaMatrix.RowsCount;
++alternativeIndex)
        {
           var alternativeCriteria = alternativesCriteriaMatrix.GetRowColumns(alternativeIndex);

           var sDistance = 0d;
           var rDistance = 0d;

           foreach (var value in alternativeCriteria)
           {
              sDistance += value;

              if (value > rDistance)
              {
                 rDistance = value;
              }
           }

           sDistances[alternativeIndex] = new AlternativeValue(sDistance, alternativeIndex);
           rDistances[alternativeIndex] = new AlternativeValue(rDistance, alternativeIndex);

           if (sDistance > sDistanceMax)
           {
              sDistanceMax = sDistance;
           }
           else if (sDistance < sDistanceMin)
           {
              sDistanceMin = sDistance;
           }

           if (rDistance > rDistanceMax)
           {
              rDistanceMax = rDistance;
           }
           else if (rDistance < rDistanceMin)
           {
              rDistanceMin = rDistance;
           }
        }

        var sDistanceDelta = sDistanceMax - sDistanceMin;

        if (sDistanceDelta == 0) sDistanceDelta = 1;

        var rDistanceDelta = rDistanceMax - rDistanceMin;

        if (rDistanceDelta == 0) rDistanceDelta = 1;
```

```
            var vCoefInverse = 1 - vCoef;

            var qValues = new AlternativeValue[alternativesCriteriaMatrix.RowsCount];

            for (var alternativeIndex = 0; alternativeIndex < alternativesCriteriaMatrix.RowsCount;
++alternativeIndex)
            {
                var value = vCoef * (sDistances[alternativeIndex].Value - sDistanceMin) /
sDistanceDelta +
                    vCoefInverse * (rDistances[alternativeIndex].Value - rDistanceMin) /
rDistanceDelta;

                qValues[alternativeIndex] = new AlternativeValue(value, alternativeIndex);
            }

            return (sDistances.OrderBy(x => x.Value).ToArray(), rDistances.OrderBy(x =>
x.Value).ToArray(), qValues.OrderBy(x => x.Value).ToArray());
        }

        private static bool CheckAcceptableAdvantage(double dq, double bestValue, double
candidateValue) => candidateValue - bestValue >= dq;

        public class AlternativeValue
        {
            public double Value { get; }

            public int AlternativeIndex { get; }

            public AlternativeValue(double value, int alternativeIndex)
            {
                Value = value;
                AlternativeIndex = alternativeIndex;
            }
        }
    }
}
```

НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
“КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ імені ІГОРЯ СІКОРСЬКОГО”
Кафедра автоматизованих систем обробки інформації та управління

УЗГОДЖЕНО

Керівник проєкту

_____ *Оксана ЖУРАКОВСЬКА*
(підпис) (вл. ім’я, прізвище)

“5” квітня 2021 р.

ЗАТВЕРДЖУЮ

В.о. завідувача кафедри

_____ *Олександр ПАВЛОВ*
(підпис) (вл. ім’я, прізвище)

“6” квітня 2021 р.

Інформаційна система з розподілу завдань між виконавцями

ТЕХНІЧНЕ ЗАВДАННЯ

Шифр *ДП 7119.01.000 ТЗ*

на 9 сторінках

Київ 2021

ЗМІСТ

1 ЗАГАЛЬНІ ПОЛОЖЕННЯ

1.1 Повне найменування системи та її умовне позначення

Повне найменування системи**:** Інформаційна система з розподілу завдань між виконавцями.

Умовне позначення**:** Система.

1.2 Найменування організації-замовника та організацій-учасників робіт

Замовником виступає кафедра автоматизованих систем обробки інформації та управління Національного технічного університету України "Київський політехнічний інститут ім. Ігоря Сікорського" (далі за текстом — Замовник). Адреса замовника: м. Київ, п. Перемоги 37, 18 корпус ФІОТ.

Розробник системи – студент групи ІС-71 кафедри автоматизованих систем обробки інформації та управління факультету ФІОТ Національного технічного університету України "Київський політехнічний інститут ім. Ігоря Сікорського" Ноженко Ярослав Ігорович.

1.3 Перелік документів, на підставі яких створюється система

Підставою для розробки інформаційної системи з розподілу завдань між виконавцями є завдання на дипломне проектування, затверджене кафедрою автоматизованих систем обробки інформації та управління факультету ФІОТ Національного технічного університету України «Київський політехнічний інститут імені Ігоря Сікорського».

При розробці системи і створення проектно-експлуатаційної документації Виконавець повинен керуватися вимогами наступних нормативних документів:

а) ДСТУ 19.201-78. Технічне завдання. Вимоги до змісту і оформлення;

б) ДСТУ 34.601-90. Комплекс стандартів на автоматизовані системи. Автоматизовані системи.

в) ДСТУ 34.201-89. Інформаційні технології. Комплекс стандартів на автоматизовані системи. Види, комплексність і позначення документів при створенні автоматизованих систем.

1.4 Планові терміни початку і закінчення роботи зі створення системи

Плановим терміном початку роботи над розробкою системи є 12 квітня 2021 року. Плановим терміном по закінченню роботи над розробкою системи є 12 травня 2021 року.

2 ПРИЗНАЧЕННЯ І ЦІЛІ СТВОРЕННЯ СИСТЕМИ

2.1 Призначення системи

Підтримка процесу розподілу завдань між виконавцями.

2.2 Цілі створення системи

Підвищення ефективності процесу розподілу завдань між виконавцями за критеріями, що враховують параметри завдань та показники ефективності виконавців.

3 ХАРАКТЕРИСТИКА ОБ'ЄКТА АВТОМАТИЗАЦІЇ

Об’єктом автоматизації є процес розподілу завдань між виконавцями.

Для того, щоб автоматизувати цей процес нам потрібно:

а) провести аналіз показників ефективності виконавців, шляхом аналізу процесу виконання ними завдань, що призначені на проєкті;

б) використовуючи правила побудови критеріїв сформувати розподіл завдань між виконавцями, що призначені на ітерацію.

Після впровадження системи процес розподілу завдань має такий вигляд:

а) авторизуватись як менеджер проєкту;

б) створити ітерацію та призначити їх виконавців та завдання, що потрібно виконати;

в) створити чи використати збережені правила побудови критерії та запросити у системи розподіл завдань між виконавцями;

г) розподілити завдання між виконавців вибираючи до кожного завдання виконавця із наданої множини виконавців до цього завдання.

4 ВИМОГИ ДО СИСТЕМИ

4.1 Вимоги до системи в цілому

Вимоги до структури і функціонування системи: система передбачає можливість покращення та доповнення існуючого функціоналу в залежності від вимог користувача.

Вимоги до надійності, безпеки, експлуатації, технічного обслуговування, ремонту, захисту інформації від несанкціонованого доступу, захисту від впливу зовнішніх дій: реалізація системи захищена від несанкціонованого доступу, має механізм авторизації.

Вимоги до персоналу: 1 користувач зі знаннями основ HTTP запитів.

4.2 Вимоги до функціональних характеристик

Інформаційна система з розподілу задач між виконавцями повинна надавати такі функціональні можливості:

а) реєстрація та авторизація системних користувачів;

б) базове керування проєктами;

1) створення проєкту;
2) перегляд списку проєктів;
3) видалення проєкту;
4) призначення виконавців на проєкт;
5) видалення виконавців з проєкту;
6) перегляд списку виконавців на проєкті;

в) базове керування завданнями;

1) створення завдання;
2) видалення завдання;
3) зміна завдання;

г) базове керування ітераціями;

1) створення ітерації;
2) призначення виконавців на ітерацію;
3) призначення завдань на ітерацію;

д) створення правил побудови критеріїв;

е) аналіз показників ефективності виконавців;

ж) створення розподілу завдань між виконавцями з заданими параметрами.

4.3 Вимоги до видів забезпечення

Вимоги до технічних засобів:

Комп’ютер з процесором тактовою частотою не менше 1 ГГц, оперативною пам’яттю не менше 4 Гб, постійною пам’яттю не менше 1 Гб та операційною системою Windows 7+ або MacOS 10.13+ або Debian 9+ або Fedora 32+ або Ubuntu 16.04+ зі встановленим .NET SDK 5+. Також для роботи з програмою потрібно встановити ПЗ, яке дозволяє робити HTTP запити. Наприклад, веб-браузери Mozilla Firefox, Google Chrome, чи спеціалізоване ПЗ, наприклад Postman.

5 СТАДІЇ ТА ЕТАПИ РОЗРОБКИ

Стадії виконання наведені в таблиці 5.1

Таблиця 5.1

№ з/п	Назва етапу виконання дипломного проєкту	Термін виконання етапу
1	Пошук матеріалів, опис предметного середовища	16.04.2021
2	Виявлення вимог до програмного продукту	21.04.2021
3	Аналіз існуючих рішень	23.04.2021
4	Формування змістовної та математичної постановки задачі	30.04.2021
5	Вибір алгоритму та алгоритмізація задачі	03.05.2021
6	Розробка програмного забезпечення	11.05.2021
7	Тестування та налагодження програмного забезпечення	12.05.2021
8	Виконання графічних документів	13.05.2021
9	Оформлення пояснювальної записки	14.05.2021
10	Подання ДП на попередній захист	14.05.2021
11	Подання ДП на основний захист	04.06.2021
12	Подання ДП рецензенту	07.06.2021

6 ПОРЯДОК КОНТРОЛЮ ТА ПРИЙМАННЯ

6.1 Види випробувань

Для перевірки правильності функціонування програмного забезпечення буде проведено тестування функціоналу системи та перевірка відгуку системи на правильність. Тестування граничних значень вхідних даних.

UNICHECK

Ім'я користувача:
Попенко Володимир Дмитрович

ID перевірки:
1008082024

Дата перевірки:
30.05.2021 01:32:55 EEST

Тип перевірки:
Doc vs Internet + Library

Дата звіту:
30.05.2021 23:37:39 EEST

ID користувача:
77149

Назва документа: **Nojenko_bachelor_is71**

Кількість сторінок: **54** Кількість слів: **4698** Кількість символів: **37665** Розмір файлу: **4.41 MB** ID файлу: **1008167663**

Виявлено модифікації тексту (можуть впливати на відсоток схожості)

6.6%
Схожість

Найбільша схожість: **1.94%** з джерелом з Бібліотеки (ID файлу: **1008167646**)

2.87% Джерела з Інтернету	44	Сторінка 56
5.98% Джерела з Бібліотеки	243	Сторінка 56

0% Цитат

Не знайдено жодних цитат

Вилучення списку бібліографічних посилань вимкнене

0%
Вилучень

Немає вилучених джерел

Модифікації

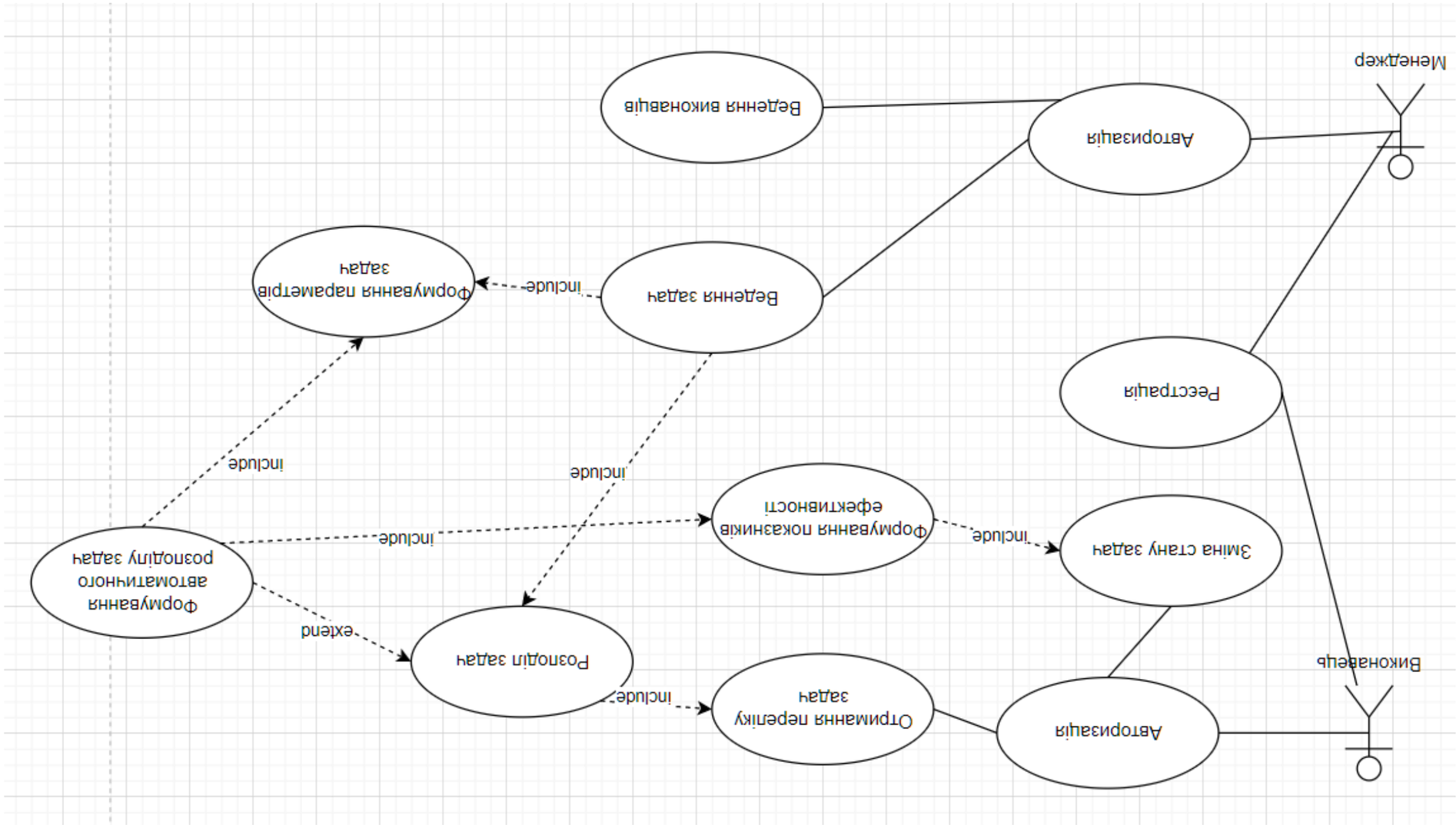
Виявлено модифікації тексту. Детальна інформація доступна в онлайн-звіті.

Замінені символи	2
Підозріле форматування	23 сторінки

Графічний матеріал
до дипломного проєкту

на тему: Інформаційна система з розподілу завдань між виконавцями

Київ – 2021 року



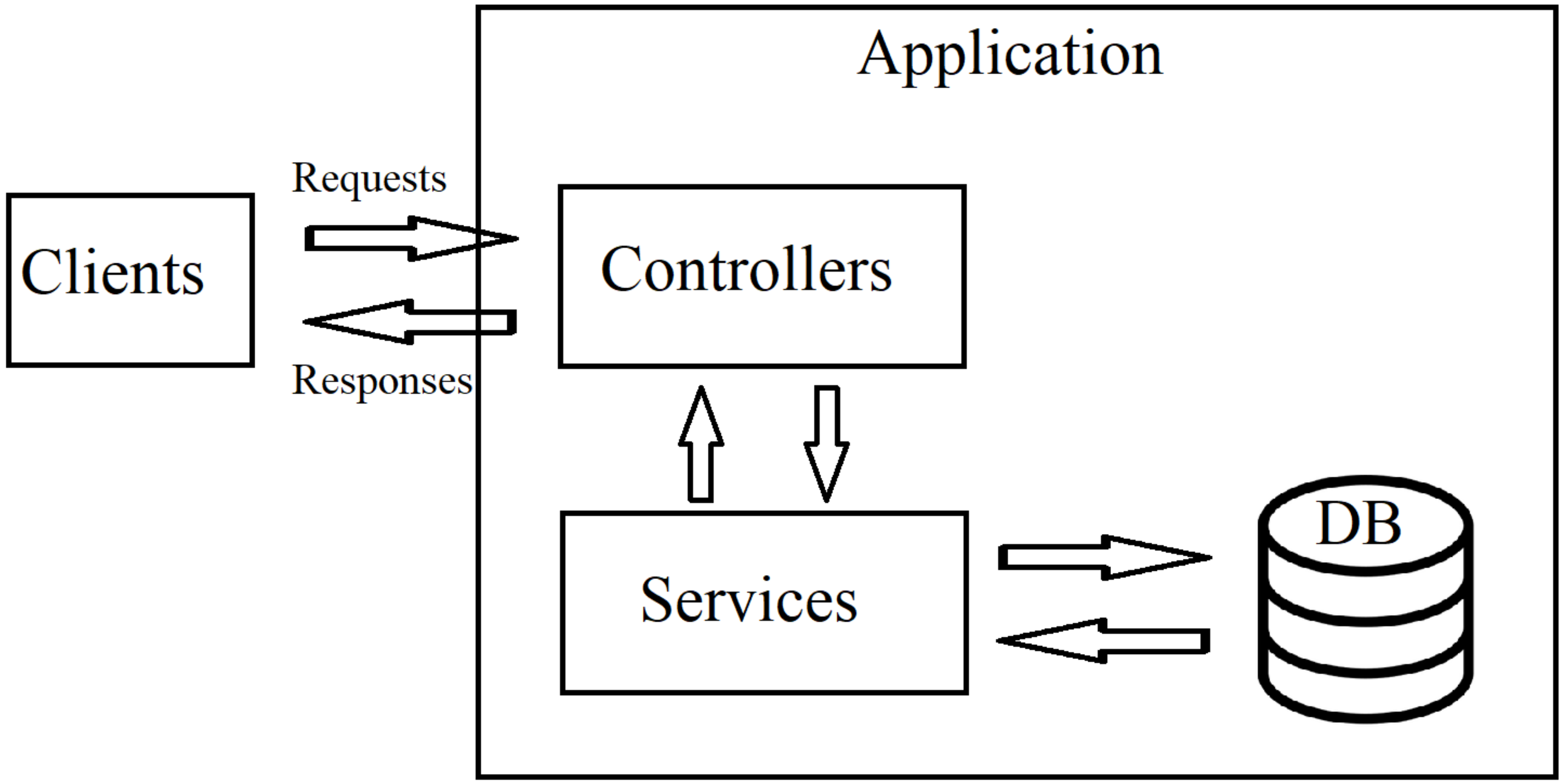
ДП 7119.02.000 ССВ

Зм.	Арк.	№ документа	Підпис	Дата	ДП 7119.02.000 ССВ			
Розробив		Ноженко Я.І.			Схема структура варіантів використання	Літера	Маса	Масштаб
Перевірив		Журавковська О.С.						
Консульт.						Аркуш 1	Аркушів 1	
Н. кон.		Жданова О.Г.			Інформаційна система з розподілу завдань між виконавцями	КПІ ім. Ігоря Сікорського Каф. АСОІУ Гр. ІС-71		
Затвердив		Журавковська О.С.						

ДП 7119.03.000 СБД

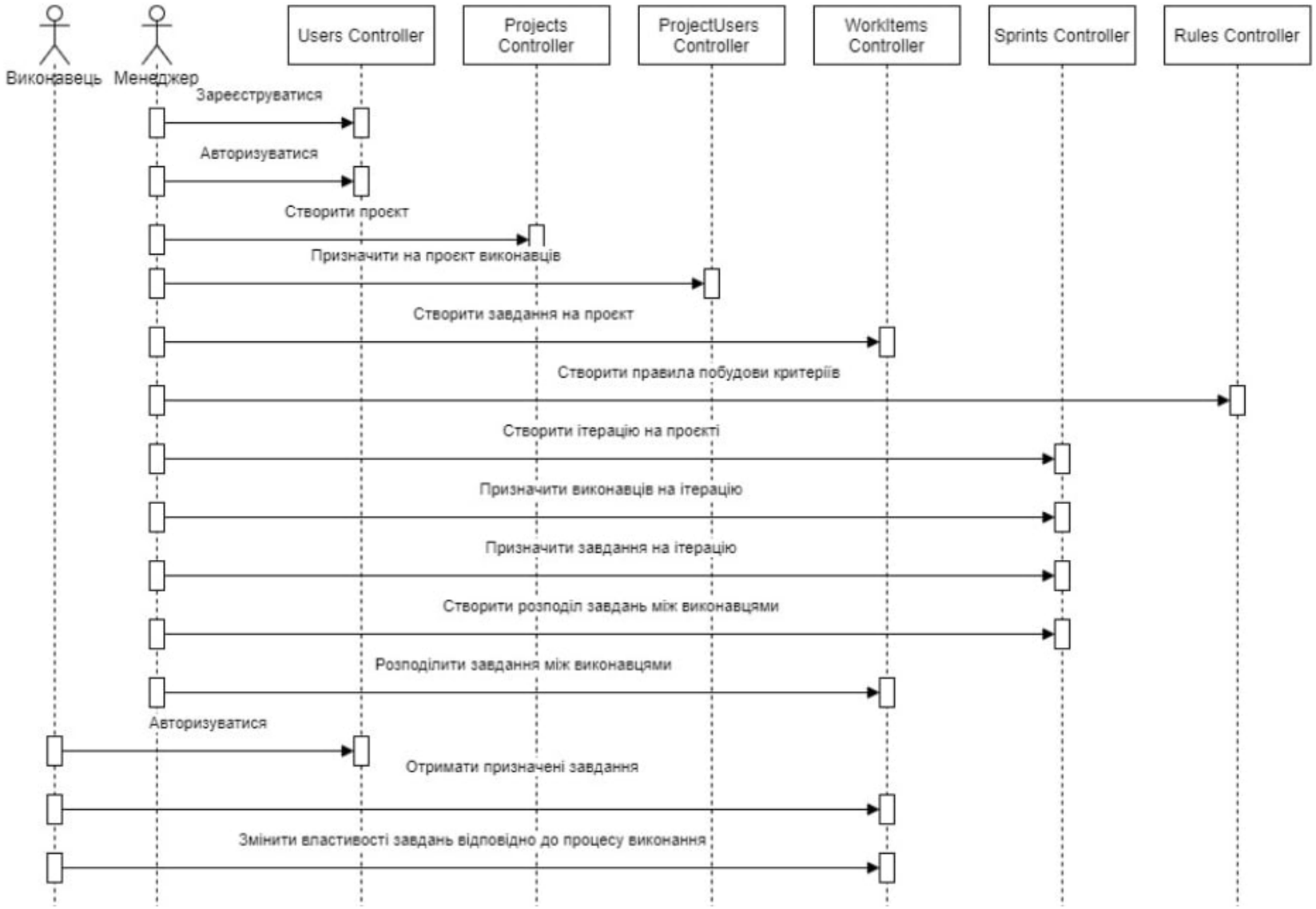
					ДП 7119.03.000 СБД	
Зм.	Арк.	№ документа	Підпис	Дата	Схема бази даних	Літера / Маса / Масштаб
Розробив		Ноженко Я.І.				
Перевірив		Жураковська О.С.				Аркуш 1 / Аркушів 1
Консульт.					Інформаційна система з розподілу завдань між виконавцями	КПІ ім. Ігоря Сікорського Каф. АСОІУ Гр. ІС-71
Н. кон.		Жданова О.Г.				
Затвердив		Жураковська О.С.				

ДП 7119.04.000 СА



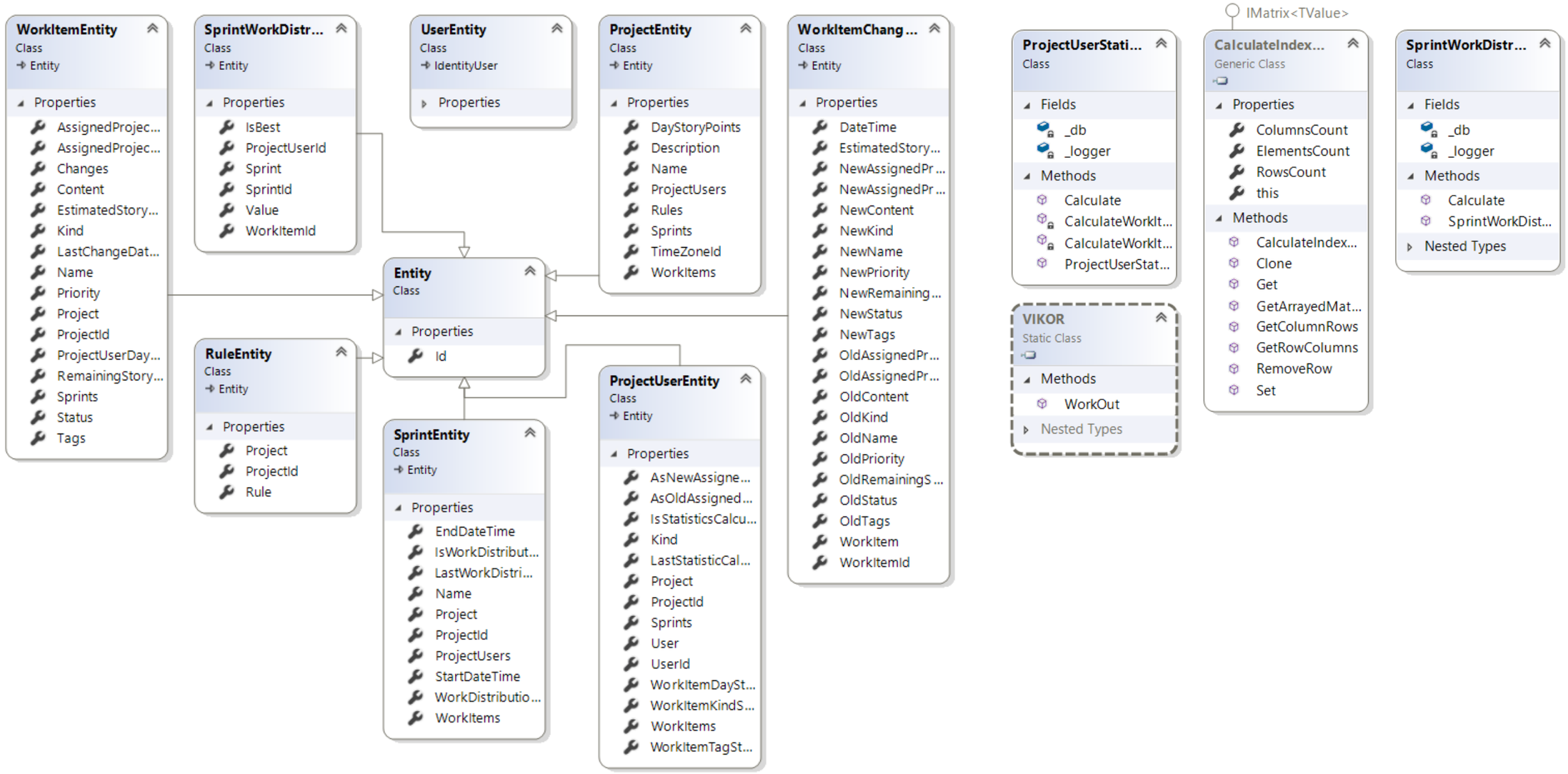
					ДП 7119.04.000 СА			
					Схема архітектури програмного забезпечення	Літера	Маса	Масштаб
Зм.	Арк.	№ документа	Підпис	Дата				
Розробив		Ноженко Я.І.				Аркуш 1	Аркушів 1	
Перевірив		Жураковська О.С.						
Консульт.					Інформаційна система з розподілу завдань між виконавцями	КПІ ім. Ігоря Сікорського Каф. АСОІУ Гр. ІС-71		
Н. кон.		Жданова О.Г.						
Затвердив		Жураковська О.С.						

ДП 7119.05.000 ССП

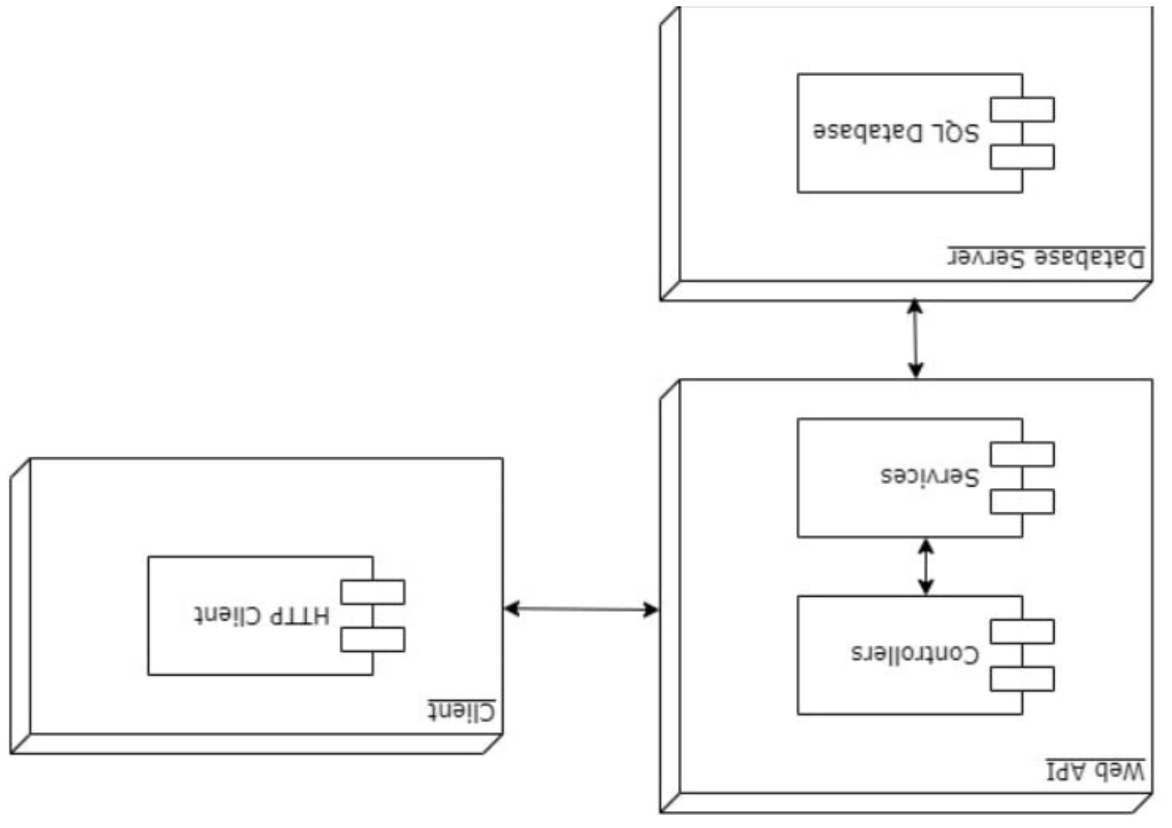


					ДП 7119.05.000 ССП			
Зм.	Арк.	№ документа	Підпис	Дата	Схема структурна послідовності	Літера	Маса	Масштаб
Розробив		Ноженко Я.І.						
Перевірив		Жураковська О.С.				Аркуш 1	Аркушів 1	
Консульт.					Інформаційна система з розподілу завдань між виконавцями	КПІ ім. Ігоря Сікорського Каф. АСОІУ Гр. ІС-71		
Н. кон.		Жданова О.Г.						
Затвердив		Жураковська О.С.						

ДП 7119.06.000 ССК



					ДП 7119.06.000 ССК	
Зм.	Арк.	№ документа	Підпис	Дата	Схема структурна класів програмного забезпечення	Літера / Маса / Масштаб
Розробив		Ноженко Я.І.				
Перевірив		Жураковська О.С.				Аркуш 1 / Аркушів 1
Консульт.					Інформаційна система з розподілу завдань між виконавцями	КПІ ім. Ігоря Сікорського Каф. АСОІУ Гр. ІС-71
Н. кон.		Жданова О.Г.				
Затвердив		Жураковська О.С.				



Зм.	Арк.	№ документа	Підпис	Дата						
Розробив		Ноженко Я.І.			Схема структурна компонентів			Літера	Маса	Масштаб
Перевірив		Журавковська О.С.								
Консульт.								Аркуш 1	Аркушів 1	
Н. кон.		Жданова О.Г.			Інформаційна система з розподілу завдань між виконавцями			КПІ ім. Ігоря Сікорського Каф. АСОІУ Гр. ІС-71		
Затвердив		Журавковська О.С.								

ДП 7119.07.000 СК

ДП 7119.07.000 СК