НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ імені ІГОРЯ СІКОРСЬКОГО»
ФАКУЛЬТЕТ ІНФОРМАТИКИ І ОБЧИСЛЮВАЛЬНОЇ ТЕХНІКИ
Кафедра автоматизованих систем обробки інформації та управління

До захисту допущено:

В.о. завідувача кафедри

_____ *Олександр ПАВЛОВ*

(підпис) (вл.ім'я, прізвище)

Дипломний проєкт

на здобуття ступеня бакалавра

за освітньо-професійною програмою «Інформаційні управляючі системи та технології»
спеціальності 122 «Комп'ютерні науки та інформаційні технології»

на тему: « Інформаційно-аналітична система адміністрування бюджетних програм місцевої влади на прикладі дорожнього будівництва»

Виконала: Студентка *IV* курсу, групи *ІС-61*

Шолудько Анна Анатоліївна _____
(прізвище, ім'я, по батькові) (підпис)

Керівник *доц., к.т.н. Попенко Володимир Дмитрович* _____
(посада, науковий ступінь, вчене звання, прізвище, ім'я, по батькові) (підпис)

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

Рецензент *доц., к.т.н. Ткаченко Валентина Василівна* _____
(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.
Студентка _____
(підпис)

Київ – 2020 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 – 122 «Комп'ютерні науки та інформаційні технології»

Освітньо-професійна програма «Інформаційні управляючі системи та технології»

ЗАТВЕРДЖУЮ

В.о. завідувача кафедри

_____ *Олександр ПАВЛОВ*

(підпис) (вл.ім’я, прізвище)

З А В Д А Н Н Я

на дипломний проєкт студенту

Шолудько Анні Анатоліївні
(прізвище, ім’я, по батькові)

1. Тема проєкту « Інформаційно-аналітична система адміністрування бюджетних програм місцевої влади на прикладі дорожнього будівництва»

керівник проєкту *Попенко Володимир Дмитрович, к.т.н., доцент*
(прізвище, ім’я, по батькові, науковий ступінь, вчене звання)

затверджені наказом по університету від “*7*”*травня* 2020 р. №1081-с

2. Термін подання студентом проєкту “*01*”*червня* 2020 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 дані, 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 діяльності

2. Схема бази даних

3. Схема структурна класів програмного забезпечення

4. Схема структурна послідовності

5. Схема структурна компонентів програмного забезпечення

6. Схема структурна варіантів використань

6. Консультанти розділів проєкту

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

7. Дата видачі завдання *«13» квітня 2020 року*

Календарний план

№ з/п	Назва етапів виконання дипломного проєкту	Термін виконання етапів проєкту	Примітка
1.	*Вивчення рекомендованої літератури*	*15.03.2020*	1
2.	*Аналіз існуючих методів розв'язання задачі*	*01.04.2020*	2
3.	*Постановка та формалізація задачі*	*10.04.2020*	3
4.	*Розробка інформаційного забезпечення*	*20.04.2020*	4
5.	*Алгоритмізація задачі*	*25.04.2020*	5
6.	*Обґрунтування використовуваних технічних засобів*	*04.05.2020*	6
7.	*Розробка програмного забезпечення*	*05.05.2020-20.05.2020*	7
8.	*Налагодження програми*	*21.05.2020-25.05.2020*	8
9.	*Виконання графічних документів*	*26.05.2020*	9
10.	*Оформлення пояснювальної записки*	*26.05.2020-31.05.2020*	10
11.	*Подання ДП на попередній захист*	*15.05.2020*	11
12.	*Подання ДП на основний захист*	*01.06.2020*	12
13.	*Подання ДП рецензенту*	*02.06.2020*	13

Студентка Анна ШОЛУДЬКО

Керівник Володимир ПОПЕНКО

№ з/п	Формат	Позначення	Найменування	Кількість листів	Примітка
1	А4		Завдання на дипломний проєкт	2	
2	А4	ДП 6129.00.000 ПЗ	Пояснювальна записка	55	
3	А4	ДП 6129.01.000 ТЗ	Технічне завдання	8	
4	А3	ДП 6129.02.000 ССД	Схема структурна діяльності	3	
5	А3	ДП 6129.03.000 СБД	Схема бази даних	1	
6	А3	ДП 6129.04.000 ССК	Схема структурна класів програмного	1	
			забезпечення		
7	А3	ДП 6129.05.000 ССП	Схема структурна послідовності	2	
8	А3	ДП 6129.06.000 ССК	Схема структурна компонентів	1	
			програмного забезпечення		
9	А3	ДП 6129.07.000 ССВ	Схема структурна варіантів використань	3	

Зм.	Арк.	ПІБ	Підп.	Дата	ДП 6129.00.1081-с		
Розробн.		Шолудько А. А.			Відомість дипломного проєкту	Літ.	Лист 1 / Листів 1
Керівн.		Попенко В. Д.					
Консульт.						КПІ ім. Ігоря Сікорського Каф. АСОІУ Гр. ІС-61	
Н/контр.		Тєлишева Т. О.					
В.о.зав.каф.		Павлов О.А.					

Пояснювальна записка
до дипломного проєкту

на тему: Інформаційно-аналітична система адміністрування бюджетних програм місцевої влади на прикладі дорожнього будівництва

Київ – 2020 року

АНОТАЦІЯ

Структура та обсяг роботи. Пояснювальна записка дипломного проекту складається з п’яти розділів, до її складу входять 17 рисунків, 17 таблиць, 1 додаток та 8 джерел.

У дипломному проєкті була створена інформаційно-аналітична система адміністрування бюджетних програм місцевої влади на прикладі дорожнього будівництва.

У розділі «Загальні положення» були визначені актори та функції, які вони виконують, сформульовані цілі та задачі розробки, визначене її призначення.

У розділі «Інформаційне забезпечення» описані вхідні та вихідні дані, спроектована база даних.

У розділі «Математичне забезпечення» була розглянута задача про розподіл капіталовкладень на ремонт ділянок дороги, сформульована її математична модель, визначені та описані методи для розв’язання цієї задачі.

У розділі «Програмне та технічне забезпечення» були визначені та описані засоби для розробки програмного забезпечення даного дипломного проєкту, розроблені діаграми класів, послідовності та компонентів для опису архітектури інформаційно-аналітичної системи, описані специфікації функцій, детально описані звіти, які користувачі можуть бачити під час користування даною системою.

У розділі «Технологічний розділ» було розроблене детальне керівництво для користувача, описані тестові випробування для програмного забезпечення.

АДМІНІСТРУВАННЯ, РОЗПОДІЛ КОШТІВ, ЗАВДАННЯ, ПОВІДОМЛЕННЯ, КЕРІВНИК, ПІДЛЕГЛІ, ЗАДАЧА ПРО РОЗПОДІЛ КАПІТАЛОВКЛАДЕНЬ НА РЕМОНТ ДІЛЯНОК ДОРОГИ

ABSTRACT

The structure and scope of work. Explanatory node consists of five sections, it includes 17 pictures, 17 tables, 1 annex and 8 sources.

In this diploma work Information Analysis System for Local Government Budget Administration on the Example of Road Construction was created.

In the section «Terms» actors and their functions were identified, goals and tasks of development were formulated.

In the section «Information support» input and output data was described, the database was designed.

In the section «Mathematical software» the task of distribution of capital investments for repair of roads was considered, mathematical model was formulated, methods for solving this task were defined and described.

In the section «Software and hardware» tools for developing software were defined and described, class diagram, sequence diagram, component were designed to describe the architecture of information analysis system, function specifications and reports that users can see while using this system were described.

In the section «Technological section» a detailed user guide and tests for software were developed.

ADMINISTRATION, DISTRIBUTION OF FUNDS, TASKS, MESSAGES, HEAD, SUBORDINATE EMPLOYEES, THE TASK OF DISTRIBUTION OF CAPITAL INVESTMENTS FOR REPAIR OF ROADS

ЗМІСТ

ВСТУП

В умовах 21-го століття здійснюється швидкий розвиток людства, швидко росте виробництво, а разом з ним і фірми, компанії, підприємства. Розмір компаній вже давно не обмежується парою десятків, а то й сотень, людей. Кількість працівників часто досягає тисяч та десятків тисяч людей.

У таких підприємствах дуже важливо правильно організувати процес управління, адже керувати тисячами підлеглих справа непроста.

Щодо розподілу коштів у межах держави, то тут теж повинен бути чітко розроблений механізм. Адже кошти повинні пройти всі рівні влади та при цьому їх треба правильно розподілити, на найбільш нагальні потреби.

Але технології теж не стоять на місці. Саме вони і можуть вирішити проблему адміністрування та розподілу коштів.

Для полегшення адміністрування, зокрема полегшення слідкування за роботою своїх підлеглих, було вирішено створити інформаційно-аналітичну систему.

Дипломний проєкт присвячений розробці інформаційно-аналітичної системи адміністрування бюджетних програм місцевої влади на прикладі дорожнього будівництва.

Практичне значення одержаних результатів. Дана система дозволятиме швидко призначати завдання, переглядати уже призначені завдання, контролювати їх стан, обмінюватися із співробітниками повідомленнями, не відходячи від свого комп’ютера.

У рамках даної системи можна буде розв’язати задачу про розподіл капіталовкладень на ремонт ділянок дороги. Система допоможе визначити, як можна використати виділені кошти з максимальною користю, тобто які ділянки треба відремонтувати на виділені кошти, щоб мінімізувати втрати часу від поїздки по відремонтованій дорозі.

Публікації. Результати роботи були опубліковані у 1 тезах доповідей на науково-технічній конференції [1].

1 ЗАГАЛЬНІ ПОЛОЖЕННЯ

1.1 Опис предметного середовища

Адміністрування – це управлінська діяльність посадових осіб, що має чітко регламентовані функції. Іншими словами адміністрування означає професійну діяльність державних посадовців, яка спрямована на виконання рішень керівництва.

В основі адміністрування лежить цикл OODA – самовідтворюваний і саморегулюючий цикл в структуру якого входять спостереження, орієнтація, рішення та дія [2].

На всіх рівнях процеси адміністрування такі:

- керівник або отримує свої задачі від свого керівника, або сам формулює їх, крім того він отримує інформацію щодо поточного стану справ від підлеглих;
- керівник ділить свої задачі на підзадачі (бізнес-процеси) і ставить їх підлеглим;
- підлеглі керівники за допомогою виділених їм персоналу і ресурсів виконують задачі. Бізнес-процеси можуть тривати кілька циклів контролю;
- підлеглі звітують перед керівником, інформуючи про виконані та невиконані завдання.

Бюджетна програма – це сукупність заходів, які спрямовані на досягнення єдиної мети, завдань, очікуваного результату.

При розподілі бюджетних коштів перш за все бюджет ухвалюється Верховною Радою. Загальнодержавні програми фінансуються безпосередньо з державного бюджету , місцеві – з місцевих бюджетів. Завдання органів місцевої влади полягає в тому, щоб розподілити кошти, які надійшли від вищих органів влади, так, щоб максимально задовольнити потреби регіону та

при цьому не порушити чинне законодавство.

Бюджетна програма складається з таких елементів:

- завдання;
- напрямки діяльності;
- виконавці програми;
- показники результативності: затрат, продукту, ефективності, якості;
- аналіз та оцінка бюджетної програми.

Мета бюджетної програми відображає кінцевий результат та повинна узгоджуватися з місією головного розпорядника.

Завдання бюджетної програми – це цілі чи результати, які будуть досягнуті після виконання бюджетної програми.

Характерні риси завдання:

- чітке формулювання завдання та орієнтованість на результат;
- результат виражається показниками у кількісних вимірах;
- конкретна дата виконання завдання;
- конкретність та реалістичність завдання.

Розподіл бюджетних коштів здійснюють розпорядники, а саме бюджетні установи в особі їх керівників, які уповноважені на отримання бюджетних коштів та здійснення видатків. Для того, щоб розподілити кошти, розпорядник повинен виконати такі дії: отримати бюджетні призначення через їх затвердження у Законі про Державний бюджет України, затвердити кошториси розпорядників бюджетних коштів нижчого рівня, розробити проекти порядку використання коштів державного бюджету за бюджетними програмами, розробити та затвердити паспорти бюджетних програм і скласти звіти про їх виконання, здійснити управління бюджетними коштами в межах установлених йому бюджетних повноважень. Також розпорядник оцінює ефективність бюджетних програм, здійснює внутрішній контроль за повнотою надходжень, забезпечує організацію та веде бухгалтерський облік,

складає та подає фінансову і бюджетну звітність в порядку, забезпечити доступність інформації про бюджет [3].

1.1.1 Опис процесу діяльності

Діаграма діяльності загального процесу адміністрування наведена на аркуші 3 графічних матеріалів із схемою структурною діяльності.

Опишемо дану діаграму діяльності: спочатку керівник отримує розпорядження від свого керівника, аналітичні звіти від маркетолога або економіста. Після цього він вивчає звіт по бізнес-процесах. По кожному новому або існуючому бізнес-процесу виконавцям розподіляються задачі, які призначаються співробітникам. Співробітники виконують свої завдання.

Діаграма діяльності для користувача, який користується системою для перегляду та розподілу завдань між співробітниками наведена на аркуші 1 графічних матеріалів із схемою структурною діяльності.

Опишемо дану діаграму діяльності: на початку програми співробітник заходить в свій аккаунт. Після цього він може зайти на сторінку із завданнями або на сторінку з повідомленнями. Зайшовши на сторінку із завданнями, співробітник може переглянути завдання, а потім відсортувати або відфільтрувати їх. Зайшовши на сторінку з повідомленнями, він може написати повідомлення або переглянути вхідні.

Керівник може виконувати ті ж самі дії, але на додачу також може призначити завдання своїм підлеглим.

Діаграма діяльності для користувача, який розв’язує задачу про розподіл капіталовкладень на ремонт ділянок дороги наведена на аркуші 2 графічних матеріалів із схемою структурною діяльності.

Опишемо дану діаграму діяльності: спочатку розпорядник коштів в межах бюджету виділяє громадам кошти на дорожнє будівництво, потім на рівні місцевих рад ці кошти розподіляються між дорогами, інженер вводить інформацію про стан доріг у систему. Після цього користувач розв’язує

задачу про розподіл капіталовкладень на ремонт ділянок дороги, і вкінці він переглядає результати розв'язку, тобто ділянки дороги, які треба відремонтувати, та втрати часу від їзди по відремонтованій дорозі.

1.1.2 Опис функціональної моделі

У межах даного програмного продукту будуть працювати такі актори:

- керівник компанії (фірми) або керівник виконавчого органу місцевої влади;
- підлеглі;
- користувач.

Керівник може призначати завдання підлеглим, підлеглі відповідно будуть отримувати завдання. Відфільтрувати, сортувати, надіслати повідомлення зможуть і кервник, і підлеглі.

Щодо користувача, то він зможе розв'язувати задачу про розподіл вкладень, тобто визначити, які саме ділянки дороги необхідно відремонтувати, щоб мінімізувати втрати часу на поїздку по відремонтованій дорозі.

Модель варіантів використання загального процесу адміністрування наведена на аркуші 3 графічних матеріалів із моделлю варіантів використання.

Модель варіантів використання користувача, який користується системою наведена на аркуші 1 графічних матеріалів із моделлю варіантів використання.

Модель варіантів використання користувача, який розв'язує задачу про ремонт доріг наведена на аркуші 2 графічних матеріалів із моделлю варіантів використання.

Функціональні вимоги системи наведені в таблиці 1.1.

Таблиця 1.1 – Функціональні вимоги системи

№ п/п	Зміст вимоги
1	Система надає можливість зайти в свій аккаунт.
2	Система надає можливість призначити завдання співробітнику.
3	Система надає можливість перегляну всі завдання.
4	Система надає можливість переглянути завдання, які призначені поточному користувачу.
5	Система надає можливість відсортувати завдання.
6	Система надає можливість відфільтрувати завдання.
7	Система надає можливість надіслати повідомлення.
8	Система надає можливість переглянути вхідні повідомлення.
9	Система надає можливість розв’язати задачу про розподіл капіталовкладень на ремонт ділянок доріг.

1.2 Огляд наявних аналогів

Прямих аналогів даної інформаційної системи не було виявлено. Але на даний момент існує схожа система під назвою «Бітрікс24».

Вона дозволяє полегшити робочий процес співробітникам, налаштувавши автоматичний розподіл відповідальних. Відбувається все таким чином: у компанію надходить заявка на розгляд. Далі для налаштування автоматизації співробітникам надається відповідний статус. Залежно від нього система автоматично випадковим чином відправляє заявку тому, кому призначений потрібний статус.

Щодо задачі про ремонт доріг, то вже існує задача про капіталовкладення, яка визначає оптимальний розподіл коштів між філіалами виробничої системи для максимізації прибутку. Тобто задача про ремонт доріг подібна до даної, але відрізняється тим, що її не можна віднести до

задач динамічного програмування, оскільки в ній не виконується принцип оптимальності Беллмана.

Але саме програм, які виконують точно ті ж функції, що і інформаційна система, яка буде реалізована в рамках дипломного проекту, знайдено не було.

1.3 Постановка задачі

1.3.1 Призначення розробки

Дана інформаційна система призначена для призначення завдань керуючим своїм підлеглим а також дає можливість здійснювати наступні функції:

- сортувати завдання по різних критеріях, а саме дата призначення завдання, співробітник, якому призначене завдання, виконане завдання чи ні;
- фільтрувати завдання по тих самих критеріях;
- обмінюватися повідомленнями.

Також інформаційна система дозволяє розв’язати задачу про дороги, а саме визначити, які ділянки дороги необхідно відремонтувати, щоб вкластися у виділений бюджет та мінімізувати кількість незадоволених людей.

1.3.2 Цілі та задачі розробки

Цілі розробки:

- полегшити процес розподілу завдань між працівниками компанії;
- створити засіб контролю стану бізнес-процесів в організації;
- надати можливість обмінюватися повідомленнями;
- надати можливість швидко переглядати завдання конкретного працівника, їх виконання;

- підвищити ефективність капіталовкладень у ремонт доріг.

Задачі розробки:

- створення системи для розподілу завдань між працівниками та перегляду поточної ситуації їх виконання;
- визначення ділянок дороги, які необхідно відремонтувати для мінімізації кількості незадоволених людей.

Висновок до розділу

У розділі описане предметне середовище даної інформаційно-аналітичної системи, її функціональна модель та процес діяльності, наведені існуючі аналоги, описане призначення розробки, а саме: цілі та задачі.

2 ІНФОРМАЦІЙНЕ ЗАБЕЗПЕЧЕННЯ

2.1 Вхідні дані

Початкові дані вводяться в систему вручну користувачем в текстові поля.

Для роботи системи необхідні такі вхідні дані:

- логін та пароль користувача;
- текст завдання та логін користувача, якому призначається це завдання;
- текст повідомлення, яке буде надіслане іншому користувачу;
- текст для фільтрації;
- максимальна сума бюджету, яка може бути витрачена на ремонт доріг.

2.2 Вихідні дані

Результатом роботи програми є додані завдання, надіслані повідомлення та розв’язання задачі про розподіл капіталовкладень на ремонт ділянок доріг.

Тобто вихідними даними є:

- список завдань співробітників;
- вхідні повідомлення користувачів;
- номери ділянок доріг, які необхідно відремонтувати;
- втрати часу від поїздки.

2.3 Опис структури бази даних

Для зберігання даних була спроектована база даних, схема якої наведена на аркуші 1 графічних матеріалів схеми бази даних.

У таблиці Person зберігаються дані про користувачів. Її опис наведений в таблиці 2.1.

Таблиця 2.1 – Опис таблиці «Користувач»

Назва поля	Опис
id	Id користувача
логін	Логін користувача
пароль	Пароль користувача
ім’я	Ім’я користувача
прізвище	Прізвище користувача
роль	Роль користувача в рамках фірми

У таблиці Task зберігаються дані про завдання, які призначають співробітникам фірми. Її опис наведений в таблиці 2.2.

Таблиця 2.2 – Опис таблиці «Завдання»

Назва поля	Опис
id	Id користувача
текст	Текст завдання
дата_призначення	Дата призначення
дата_виконання	Дата виконання
Id_користувача	Id користувача, якому призначене завдання
id_процесу	Id процесу, до якого належить завдання

У таблиці Message зберігаються дані про повідомлення, які надсилають користувачі. Її опис наведений в таблиці 2.3.

Таблиця 2.3 – Опис таблиці «Повідомлення»

Назва поля	Опис
id	Id користувача

Продовження таблиці 2.3

Назва поля	Опис
текст	Текст повідомлення
дата	Дата відправки
id_отримувача	Id користувача, якому прийшло повідомлення
id_відправника	Id користувача, який надіслав повідомлення

У таблиці Activity зберігаються дані про вид діяльності, який фінансується. Її опис наведений в таблиці 2.4.

Таблиця 2.4 – Опис таблиці «Вид діяльності»

Назва поля	Опис
id	Id виду діяльності
назва	Назва виду діяльності

У таблиці Process зберігаються дані про бізнес-процес. Її опис наведений в таблиці 2.5.

Таблиця 2.5 – Опис таблиці «Процес»

Назва поля	Опис
id	Id бізнес-процесу
назва	Назва бізнес-процесу
id_діяльності	Id виду діяльності, до якого належить бізнес-процес

Висновок до розділу

Під час написання даного розділу була спроектована база даних, яка буде необхідна для даної інформаційно-аналітичної системи, було визначено, які вхідні дані необхідні для її роботи, а також вихідні дані, які будуть отримані в результаті роботи програми.

3 МАТЕМАТИЧНЕ ЗАБЕЗПЕЧЕННЯ

3.1 Змістовна постановка задачі

Дорога ділиться на однакові за розміром ділянки.

$\{1, \dots, n\}$ – скінченна множина номерів ділянок, впорядкованих вздовж дороги.

Ділянку або потрібно ремонтувати, або ні. Кожна ділянка має свою вартість, тобто розмір коштів, які потрібно витратити на її ремонт.

$\{l_1, \dots, l_n\}$ – скінченна множина вартості ділянок. Якщо ділянка не потребує ремонту, вартість її ремонту дорівнює нулю.

Кошти, виділені на ремонт дороги, обмежені. Тому з ділянок, які потребують ремонту, ремонтуються лише деякі. Відремонтовані ділянки складають відремонтовану частину дороги. Чим довша неперервна частина дороги, яка є відремонтованою, тим більшу швидкість може розвити водій. Чим більша швидкість розвинена, тим менше часу автомобіль знаходиться в дорозі. Швидкість, яку може розвивати автомобіль, обмежена.

Необхідно визначити, які ділянки дороги необхідно відремонтувати, щоб мінімізувати час поїздки.

3.2 Математична постановка задачі

Сформулюємо цю задачу у вигляді задачі цілочислового лінійного програмування. Введемо булеві змінні:

$$x_j = \begin{cases} 1, \text{якщо } j - \text{ту ділянку дороги ремонтуватимуть} \\ 0, \text{інакше} \end{cases} \quad (3.1)$$

і невід'ємні цілочисельні змінні:

u – максимальний обсяг бюджету, який виділений на ремонт доріг.

a – прискорення автомобіля.

$s_{\text{поч ділянки}}$ – початкова довжина ділянки.

s_j – довжина j-ої ділянки дороги. Оскільки при розташуванні декількох відремонтованих чи невідремонтованих ділянок доріг підряд вони об'єднуються в одну ділянку, то довжина ділянки може змінюватися.

$s_g = 10$ м – відстань, протягом якої здійснюється гальмування.

$V_{max} = 90$ км/год – максимальна швидкість, яку може розвинути автомобіль, з урахуванням Правил дорожнього руху.

$V_{min} = 10$ км/год – мінімальна швидкість, яку може розвинути автомобіль на дорозі, яка потребує ремонту.

t_g – втрати часу від гальмування.

t_r – втрати часу від розгону.

$t_{p\ j}$ – втрати часу від повільної їзди на j-й ділянці дороги.

$t_{\text{розгону}}$ – час розгону.

$t_{\text{гальмування}}$ – час гальмування.

t_j – загальні втрати часу від поїздки на j-й ділянці дороги.

Тоді в цільової функції мінімізується втрати часу від поїздки:

$$z = \sum_{j=1}^{n} t_j \rightarrow min \tag{3.2}$$

При умовах:

Бюджет, виділений на ремонт доріг, обмежений:

$$\sum_{j=1}^{n} x_j l_j \leq u \tag{3.3}$$

Швидкість, яку може розвивати автомобіль, обмежена:

$$V_j \le V_{max}, j \in \{1, \dots, n\}. \quad (3.4)$$

Втрати часу від їзди на j-й ділянці дороги:

$$t_j = t_g + t_{p\,j} + t_r, j \in \{1, \dots, n\} \quad (3.5)$$

Якщо попередня ділянка j-1 нормальна, а j – не відремонтована, додаємо час гальмування. Якщо попередня ділянка j-1 не відремонтована, а j – нормальна, додаємо час розгону. Якщо ділянка j не відремонтована, додаємо втрати часу від повільної їзди.

Якщо $j \ne 1$, $l_{j-1}>0$ та $l_j = 0, t_j = t_j + t_g$. (3.6)

Якщо $j \ne 1$, $l_{j-1} =0$ та $l_j > 0, t_j = t_j + t_r$. (3.7)

Якщо $l_j > 0, t_j = t_j + t_{pj}$. (3.8)

Час розгону [4]:

$$t_{\text{розгону}} = \frac{V_{max} - V_{min}}{a}. \quad (3.9)$$

Втрати часу від розгону:

$$t_r = \frac{s}{V_{max}} - t_{\text{розгону}} \quad (3.10)$$

Час гальмування автомобіля дороги знаходимо з наступної формули [4]:

$$s_g = V_{max} t_{\text{гальмування}} - \frac{a t_{\text{гальмування}}^2}{2} \quad (3.11)$$

Втрати часу від гальмування [4]:

$$t_g = \frac{s_{\text{поч.ділянки}}}{V_{max}} - t_{\text{гальмування}}. \quad (3.12)$$

Втрати часу від повільного руху:

$$t_{p\,j} = \frac{s_j}{V_{min}} - \frac{s_j}{V_{max}}, j \in \{1, \dots, n\} \quad (3.13)$$

3.3 Обґрунтування методу розв’язання

Принцип Беллмана говорить, що оптимальна стратегія визначається лише станом системи в даний момент і не залежить від того, як ця система прийшла в дану точку.

Розглядаючи дану задачу, ми можемо бачити, що час поїздки автомобіля залежить від початкового розташування автомобіля та розміщення відремонтованих ділянок дороги, оскільки від цього залежить, яку середню швидкість може розвинути автомобіль, час гальмування та розгону. Зокрема, початкова швидкість на ділянці залежить від стану попередньої ділянки.

З цього можемо зробити висновок, що для даної задачі принцип оптимальності Беллмана не виконується. Отже, її не можна розв’язати методами динамічного програмування.

Можемо зробити висновок, що для розв’язання цієї задачі найкраще підходять наближені алгоритми, а саме жадібний та генетичний.

3.4 Опис методів розв’язання

Жадібний алгоритм – алгоритм, що полягає в прийнятті локально оптимальних рішень на кожному етапі, при цьому допускаючи, що кінцеве рішення також виявиться оптимальним [5].

Реалізуючи жадібний алгоритм, ми будемо обирати ті ділянки, які мають найменшу вартість, до тих пір, доки не закінчаться кошти, виділені на ремонт дороги. Після цього треба обчислити втрати часу від поїздки по відремонтованій дорозі, враховуючи розташування відремонтованих ділянок дороги.

Послідовність дій при розв’язку задачі про розподіл капіталовкладень на ремонт ділянок дороги жадібним алгоритмом:

- записати ділянку з найменшою вартістю ремонту до розв’язку;

- виконувати пункт 1 до тих пір, доки сумарна вартість ремонту буде менша або дорівнюватиме сумі виділеного бюджету;
- обчислення значення цільової функції, тобто втрат часу від поїздки, для кожного нащадку (розв'язку).

Генетичний алгоритм – це еволюційний алгоритм, який використовують для вирішення задач оптимізації і моделювання шляхом послідовного підбору, комбінування та варіації шуканих параметрів. Генетичні алгоритми обробляють закодовану форму параметрів задачі, здійснюють пошук рішення, виходячи з деякої популяції, використовують тільки цільову функцію, застосовують ймовірнісні правила відбору [6].

Реалізуючи генетичний алгоритм, ми генеруємо популяцію, тобто розв'язки задачі (вектор x), випадковим чином. Після цього виконується попарне схрещування особин з популяції. Потім виконується мутація кожного нащадка: ген, який обирається випадковим чином, змінює своє значення на протилежне. Потім обчислюється вартість ремонту дороги для кожного розв'язку (особи) і відкидаються ті, вартість, яких перевищує вартість виділеного бюджету. Серед нащадків, які залишилися, обирається найкращий розв'язок, тобто той, де втрати часу від поїздки найменші.

Послідовність дій при розв'язку задачі про розподіл капіталовкладень на ремонт ділянок дороги генетичним алгоритмом:

- генерація популяції. (0 – якщо ділянку не будуть ремонтувати, 1 – якщо інакше);
- попарне схрещування особин з популяції;
- мутація кожного нащадка;
- обчислення вартості ремонту дороги для кожного нащадка;
- відкидання тих нащадків, для яких вартість ремонту дороги перевищує вартість виділеного бюджету;
- обчислення значення цільової функції, тобто втрат часу від поїздки, для кожного нащадку (розв'язку);

- вибір нащадка, для якого обчислене значення цільової функції найменше, як кінцевого розв’язку задачі. [1]

Висновок до розділу

У даному розділі розглянута задача про розподіл капіталовкладень на ремонт ділянок доріг. Визначено, які наближені алгоритми можна використати для її розв’язання та яким чином їх можна застосувати в рамках даної задачі.

Планується, що дана задача допоможе визначити, яким чином потрібно ремонтувати дороги, щоб зменшити час поїздки та мінімізувати втрати часу на поїздку.

4 ПРОГРАМНЕ ТА ТЕХНІЧНЕ ЗАБЕЗПЕЧЕННЯ

4.1 Засоби розробки

Для розробки серверної частини програмного забезпечення була обрана мова програмування Java, зокрема Spring framework.

Переваги Java:

- об’єктно-орієнтована;
- платформонезалежна;
- проста;
- безпечна;
- портативна;
- багатопоточна;
- високопродуктивна;
- динамічна.

Spring framework – це універсальний фреймворк з відкритим вихідним кодом для Java-платформи. Він був обраний через те, що [7]:

- в застосунку на основі Spring всі об’єкти слабозв’язані за рахунок використання впровадження залежностей;
- не потрібно створювати об’єкти вручну;
- немає необхідності зв’язувати об’єкти;
- налаштування знаходяться окремо від програмного коду, що значно спрощує його написання та читабельність.

Angular був використаний для розробки частини клієнта програмного забезпечення. Angular – це відкрита платформа для написання веб-застосунків, написана на мові TypeScript [8].

Переваги Angular:

- декларативний стиль коду;

- висока швидкість розробки;
- використання директив;
- використання схеми MVC;
- модульність;
- наявність готових розв’язків;
- двостороннє зв’язування даних.

Для зберігання даних використана база даних Oracle. Вона має такі переваги:

- добре опрацьовує транзакції;
- здатна підтримувати роботу будь-якої кількості користувачів, які виконують операції одночасно;
- легко переноситься з однієї операційної системи на іншу;
- має локальну керованість.

Середовищем розробки була обрана Intellij Idea.

Intellij Idea – інтегроване середовище розробки програмного забезпечення для багатьох мов програмування, зокрема Java, JavaScript, Python, розроблена компанією JetBrains.

Це Java IDE із широким набором інтегрованих інструментів для рефакторингу, що дозволяє програмістам швидко реорганізовувати код програм. Дизайн середовища орієнтовано на продуктивність праці програмістів, дозволяючи їм сконцентруватися на розробці функціональності, тоді як IntelliJ Idea бере на себе виконання рутинних операцій.

Беручи до уваги всі переваги Java, вона найкраще підходить для розробки даної курсової роботи, а IntelliJ Idea – зручне середовище для її використання.

4.2 Вимоги до технічного забезпечення

4.2.1 Загальні вимоги

Для нормальної роботи програми повинні бути наявні технічні засоби:

1. Комп’ютер або ноутбук з такими характеристиками:
   - 2 ГБ і більше оперативної пам’яті;
   - тактова частота процесору – 1 ГГЦ і більше.
2. Повинні бути інстальоване таке програмне забезпечення:
   - MS WINDOWS XP або новіша версія Windows;
   - база даних Oracle 11g і вище;

4.3 Архітектура програмного забезпечення

4.3.1 Діаграма класів

Опис діаграми класів, яка наведена на аркуші 1 графічних матеріалів діаграми структурної класів програмного забезпечення наведений в таблиці 4.1.

Таблиця 4.1 – Опис діаграми класів

Клас	Поля	Методи
Message (сутність «Повідомлення»)	id, текст, дата відправки, ім’я та прізвище відправника	-
Task (сутність «Завдання»)	id, ім’я та прізвище особи, якій призначене завдання, текст завдання, дата призначення та виконання	-
Person (сутність «Особа»)	id, логін, ім’я та прізвище, роль	

Продовження таблиці 4.1

Клас	Поля	Методи
DataLoader	-	Методи для отримання даних з бази даних та запис даних в базу даних
RoadsTask	Максимальна та мінімальна швидкість, яку може розвивати автомобіль, початкова довжина ділянки дороги, сума бюджету, який був виділений на ремонт доріг, дані про початковий стан доріг та їх вартість ремонту, втрати часу під час поїздки по відремонтованій дорозі, список ділянок дороги, які треба відремонтувати	Методи, які розв’язують задачу жадібним та генетичним алгоритмами, знаходять втрати часу на гальмування та розгони, обчислюють загальні втрати часу, об’єднують невідремонтовані чи відремонтовані ділянки дороги, які розташовані поряд, виводять результати
MessageService	Екземпляр класу DataLoader	Методи для отримання повідомлень та запису нового повідомлення в базу даних

Продовження таблиці 4.1

Клас	Поля	Методи
TaskService	Екземпляр класу DataLoader	Методи для отримання всіх завдань, завдань, які призначені конкретному користувачу, відсортованих та відфільтрованих завдань, отримання інформації про користувача, позначення завдання як виконаного, призначення нового завдання
MessageService	Екземпляр класу DataLoader	Методи для отримання повідомлень та запису нового повідомлення в базу даних
TaskService	Екземпляр класу DataLoader	Методи для отримання всіх завдань, завдань, які призначені конкретному користувачу, відсортованих та відфільтрованих завдань, отримання інформації про користувача, позначення завдання як виконаного, призначення нового завдання
MessageController	-	Методи для передачі повідомлень з сервера на клієнт та передачі нового повідомлення з клієнта на сервер

Продовження таблиці 4.1

Клас	Поля	Методи
TaskController	-	Методи для передачі всіх завдань, завдань, які призначені конкретному користувачу, відсортованих та відфільтрованих завдань, інформації про користувача з сервера на клієнт, передачі помітки виконання завдання, нового завдання з клієнта на сервер, передачі інформації про користувача та розв’язку задачі про розподіл капіталовкладень з сервера на клієнт
message.service	-	Методи для отримання повідомлень з контроллера та передачі нового повідомлення на контроллер
task.service	id користувача, його роль	Методи для отримання всіх завдань, завдань, які призначені конкретному користувачу, відсортованих та відфільтрованих завдань, інформації про користувача з контроллера, передачі позначення завдання як виконаного, нового завдання на контроллер, отримання інформації про користувача та розв’язку задачі про розподіл капіталовкладень

Продовження таблиці 4.1

Клас	Поля	Методи
message.service	-	Методи для отримання повідомлень з контроллера та передачі нового повідомлення на контроллер
task.service	id користувача, його роль	Методи для отримання всіх завдань, завдань, які призначені конкретному користувачу, відсортованих та відфільтрованих завдань, інформації про користувача з контроллера, передачі позначення завдання як виконаного, нового завдання на контроллер, отримання інформації про користувача та розв’язку задачі про розподіл капіталовкладень з контроллера
view-message.component	id користувача, його роль, масив повідомлень	Методи для перегляду повідомлень та відправлення нового повідомлення
assign-task.component	Екземпляр класу «Task»	Метод призначення нового завдання
account.component	id користувача, його роль	Метод входу в свій аккаунт
view-taskRoads.component	Масив для запису розв’язку задачі про розподіл капіталовкладень	Метод перегляду розв’язку задачі про розподіл капіталовкладень на ремонт ділянок доріг

Продовження таблиці 4.1

view-task.component	id користувача, його роль, масив завдань	Методи для перегляду всіх завдань, завдань, які призначені поточному користувачу, відсортованих та відфільтрованих завдань та для помітки завдання як виконаного

4.3.2 Діаграма послідовності

4.3.2.1 Діаграма послідовності для користувача, який користується системою для розподілу та моніторингу завдань

Для відображення взаємодії об’єктів для випадку, коли користувач користується системою для розподілу та моніторингу завдань була створена діаграма послідовності, яка наведена на аркуші 1 графічних матеріалів схеми структурної послідовності.

До складу діаграми послідовності входять такі об’єкти: користувач, аккаунт, сторінка із завданнями, завдання, сторінка з повідомленнями, повідомлення.

Користувач заходить на свій аккаунт. Потім здійснюється перехід на сторінку із завданнями, де можна переглянути всі завдання, відсортовані завдання, відфільтровані завдання, свої завдання, позначити завдання як виконане, призначити завдання. Після цього можна перейти на сторінку з повідомленнями, яка надає можливість переглянути свої повідомлення та надіслати повідомлення.

4.3.2.2 Діаграма послідовності для розв’язку задачі про ремонт доріг

Для відображення взаємодії об’єктів для випадку розв’язання задачі про розподіл капіталовкладень на ремонт ділянок доріг була створена діаграма послідовності, яка наведена на аркуші 2 графічних матеріалів схеми структурної послідовності.

До складу діаграми послідовності входять такі об’єкти: бюджетні кошти, інформація про стан доріг, задача про дороги.

Спочатку розпорядник коштів в межах бюджету виділяє громадам кошти на дорожнє будівництво, потім на рівні місцевих рад ці кошти розподіляються між дорогами, інженер вводить інформацію про стан доріг у систему. Після цього користувач розв’язує задачу про розподіл капіталовкладень на ремонт ділянок дороги, і вкінці він переглядає результати розв’язку, тобто ділянки дороги, які треба відремонтувати, та втрати часу від їзди по відремонтованій дорозі.

4.3.3 Діаграма компонентів

Для опису компонентів, залежностей та зв’язків між ними була створена діаграма компонентів, яка наведена на аркуші 1 графічних матеріалів схеми структурної компонентів програмного забезпечення.

Дане програмне забезпечення представляє собою веб-застосунок, який складається із html сторінок для входу у свій аккаунт, призначення завдання, перегляду завдань, перегляду та надсилання повідомлень. Вся бізнес-логіка описується на бекенд-сервері, який зв’язується з базою даних Oracle для отримання даних та їх запису в неї.

4.3.4 Специфікація функцій

У таблиці 4.2 описані функції класів програмного забезпечення:

Таблиця 4.2 – Специфікація функцій

Клас	Назва функції	Специфікація функції
Функції на стороні сервера		
MessageService	public List<Text> getMyMessage(String userId)	Отримати всі повідомлення поточного користувача з бази даних

Продовження таблиці 4.2

Клас	Назва функції	Специфікація функції
	public void sendMessage(String idTo, String idFrom, String text)	Записати нове повідомлення користувачу в базу даних
TaskService	public Person getUser(String login, String password)	Отримати інформацію про користувача з бази даних
	public void markAsCompleted(String id)	Позначити завдання як виконане
	private List<Task> getTasks(String sql)	Отримати завдання з бази даних
	public List<Task> getAllTasks()	Отримати всі завдання з бази даних
	public List<Task> getSortedTasks(String column)	Отримати відсортовані завдання з бази даних
	public List<Task> getFilteredTasks(String filterText)	Отримати відфільтровані завдання з бази даних
	public List<Task> getMyTasks(String id)	Отримати завдання, які призначені поточному користувачу з бази даних
	public void addTask(String id, String text)	Призначити завдання користувачу

Продовження таблиці 4.2

Клас	Назва функції	Специфікація функції
DataLoader	public ResultSet getInformation (String sql) throws SQLException	Функція для отримання даних з бази даних
	public ResultSet getInformationWithParameters(String sql, String id) throws SQLException	Функція для отримання даних з бази даних з підстановкою параметрів
	public void insertData(String sql) throws SQLException	Функція для запису інформації в базу даних
RoadsTask	private void greedy()	Розв’язок задачі про розподіл капіталовкладень на ремонт ділянок дороги жадібним алгоритмом
	private void geneticAlgorithm()	Розв’язок задачі про розподіл капіталовкладень на ремонт ділянок дороги генетичним алгоритмом
	private List<ArrayList<Integer> > mergePartsOfRoad()	Об’єднати декілька відремонтованих (невідремонтованих) ділянок дороги в одну
	private double getTimeHalmuvan(int vMax, double a, int sHalmuv)	Обчислення часу гальмувань та розгонів

Продовження таблиці 4.2

Клас	Назва функції	Специфікація функції
	private double calculateWasteOfTime()	Обчислення втрат часу
	public List<Double> showAnswer()	Відобразити результат
MessageController	public List<Text> getMessages(@PathVariable String id)	Передати список повідомлень з сервера на клієнт
	public void sendMessage(@PathVariable String idTo, @PathVariable String idFrom, @RequestBody String message)	Передати повідомлення з клієнта на сервер
TaskController	public List<Task> getTasks()	Передати список завдань з сервера на клієнт
	public List<Task> getSortedTasks(@RequestParam(value = "column") String column)	Передати список відсортованих завдань з сервера на клієнт
	public List<Task> getFilteredTasks(@RequestParam(value = "filterText") String filterText)	Передати список відфільтрованих завдань з сервера на клієнт

Продовження таблиці 4.2

Клас	Назва функції	Специфікація функції
	public void addTask(@RequestBody String[] text)	Передати нове призначене завдання з клієнта на сервер
	public void markAsCompleted(@RequestBody String id)	Передати відмітку завдання як виконаного з клієнта на сервер
	public List<Double> getSolutionRoads()	Передати розв’язки задачі про розподіл капіталовкладень на ремонт ділянок доріг з сервера на клієнт
	public Person getUser (@RequestParam(value = "login") String login, @RequestParam(value = "password") String password)	Передати інформацію про поточного користувача з сервера на клієнт
Функції на стороні клієнта		
message.service	getMessages(id: string): Observable<Text[]>	Отримати список повідомлень з контроллера
	sendMessage(text: string, idTo: string, idFrom: string)	Відправити нове повідомлень на контроллер
task.service	getTasks(): Observable<Task[]>	Отримати список завдань з контроллера
	getSortedTasks(column: string): Observable<Task[]>	Отримати список відсортованих завдань з контроллера

Продовження таблиці 4.2

Клас	Назва функції	Специфікація функції
	getFilteredTasks(filterText: string): Observable<Task[]>	Отримати список відфільтрованих завдань з контроллера
	getUser(login: string, password: string): Observable<Person>	Отримати інформацію про користувача з контроллера
	getMyTasks(id: string): Observable<Task[]>	Отримати список завдань конкретного користувача з контроллера
	addTask(text: string[])	Відправити інформацію про нове призначене завдання на контроллер
	markAsCompleted(id: string)	Відправити інформацію про виконане завдання на контроллер
	getSolutionRoads(): Observable<number[]>	Отримати розв’язок задачі про розподіл капіталовкладень на ремонт ділянок дороги з контроллера
	private handleError<T>(result?: T)	Опрацювати помилку
	setUserId(userId: number)	Встановити значення id користувача
	getUserId(): number	Отримати id користувача
	setRole(role: string)	Встановити значення ролі користувача
	getRole(): string	Отримати роль користувача

Продовження таблиці 4.2

Клас	Назва функції	Специфікація функції
account.component	loginInAccount(login: string, password: string): void	Увійти в свій аккаунт
assign-task.component	assignTask(id: string, text:string)	Призначити завдання
view-message.component	getMessages(): void	Переглянути повідомлення поточного користувача
	sendMessage(idTo: string, text: string): void	Відправити повідомлення
view-taskRoads.component	showResults()	Переглянути розв’язок задачі про розподіл капіталовкладень на ремонт ділянок дороги
view-task.component	getTasks(): void	Переглянути всі завдання
	getSortedTasks()	Переглянути відсортовані завдання
	getFilteredTasks(filterText: string)	Переглянути відфільтровані завдання
	getMyTasks()	Переглянути завдання поточного користувача
	markAsCompleted(id: string)	Відмітити завдання як виконане

4.4 Опис звітів

№	Ім'я	Прізвище	Текст завдання	Дата призначення	Дата виконання
1	Petro	Petrenko	task1	2020-05-05	2020-05-14
2	Halyna	Kovalenko	task2	2020-04-09	2020-04-19
3	Petro	Petrenko	new task	2020-05-05	2020-05-07

Рисунок 4.1 – Перегляд завдань користувачів

На рисунку 4.1 зображені завдання користувачів. Спочатку вказується номер завдання по порядку, потім ім’я та прізвище особи, якій воно призначене, текст завдання, дата призначення та дата виконання.

Мої повідомлення

Ivan Ivanenko	message with time	June 10, 2020 12:00AM
Ivan Ivanenko	test message	May 8, 2020 12:00AM
Ivan Ivanenko	message	May 5, 2020 11:22PM
Ivan Ivanenko	message1	May 5, 2020 12:00AM

Рисунок 4.2 – Перегляд повідомлень користувача

На рисунку 4.2 зображені повідомлення користувача. Спочатку вказується від кого прийшло повідомлення, потім його текст і дата відправки.

Рисунок 4.3 – Перегляд завдань користувачів розв’язку задачі про розподіл капіталовкладень на ремонт ділянок дороги

На рисунку 4.3 зображений вивід розв’язку задачі про розподіл капіталовкладень на ремонт ділянок дороги жадібним та генетичним

алгоритмами. Червоним кольором зображені ті ділянки дороги, які залишилися невідремонтованими, зеленим – відремонтовані. Нижче можна побачити втрати часу від поїздки по дорозі в секундах. Тобто 115, 485 с – це втрати часу при розв’язку задачі жадібним алгоритмом і 118,485 с – генетичним.

Висновок до розділу

У цьому розділі були описані засоби розробки програмного забезпечення, наведені їх переваги. Також були спроектовані діаграми класів, послідовності та компонентів для опису структури програми, наведені специфікації функцій, описані звіти, які виводить дане програмне забезпечення.

5 ТЕХНОЛОГІЧНИЙ РОЗДІЛ

5.1 Керівництво користувача

Для запуску програми необхідно запустити сервер та відкрити localhost:8081 в браузері. Після цього буде відкрита сторінка для входу користувача у свій аккаунт, який зображений на рисунку 5.1.

Користувачу необхідно ввести логін та пароль у відповідні поля вводу так, як це зображено на рисунку 5.1. Після цього натиснути кнопку «Увійти».

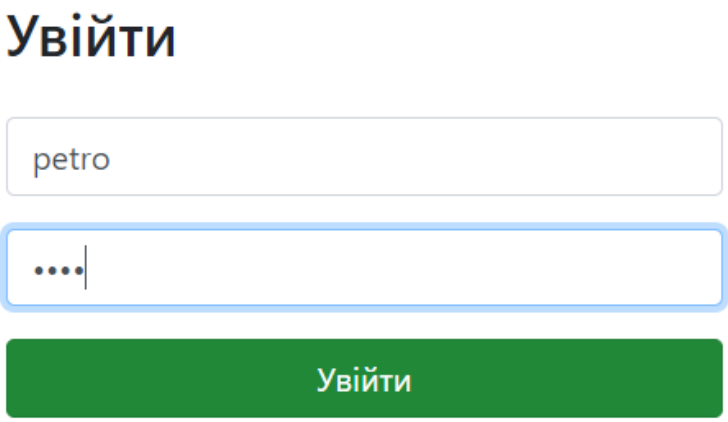


Рисунок 5.1 – Форма входу в систему

Якщо користувач з даним логіном та паролем зареєстрований в системі, то буде відкрита сторінка із завданнями, яка зображена на рисунку 5.2.

На цій сторінці користувач має можливість переглянути всі завдання, а конкретніше номер завдання, ім’я та прізвище людини, якій воно призначене, текст завдання,

Рисунок 5.2 – Сторінка з завданнями

Користувач має змогу відсортувати завдання. Для цього треба обрати назву колонки, по якій треба виконати сортування у випадаючому списку, який зображений на рисунку 5.3.

Рисунок 5.3 – Список для вибору поля, по якому необхідно відсортувати завдання

На рисунку 5.4 зображений результат сортування по даті призначення.

Рисунок 5.4 – Сторінка з відсортованими по даті призначення завданнями

Для фільтрації завдань треба ввести текст у відповідне поле для вводу так, як це зображено на рисунку 5.5 та натиснути кнопку «Відфільтрувати».

Рисунок 5.5 – Форма для фільтрації завдань

На рисунку 5.6 зображені результати фільтрації завдань по слову «May».

Рисунок 5.6 – Сторінка з результатами фільтрації завдань по слову «May»

Для того, щоб переглянути завдання, які призначені тому користувачу, який зайшов в систему треба натиснути кнопку «Мої завдання». На рисунку 5.7 зображений вигляд сторінки після натискання кнопки «Мої завдання».

Рисунок 5.7 – Сторінка з результатами відображення завдань поточного користувача

Для того, щоб відмітити завдання як виконане користувач повинен ввести номер завдання, яке він хоче відмітити як виконане так, як це зображено на рисунку 5.8 та натиснути кнопку «Відмітити як виконане».

Рисунок 5.8 – Форма для помітки завдання як виконаного

Якщо користувач є керівником, то він може також призначити завдання своїм співробітникам. Для цього він повинен ввести логін користувача, якому він хоче призначити завдання, ввести текст завдання у відповідні поля вводу так, як це зображено на рисунку 5.9, та натиснути кнопку «Призначити завдання».

Рисунок 5.9 – Форма для призначення завдань

Для того, щоб перейти на сторінку з повідомленнями треба натиснути на посилання «Повідомлення» бічної панелі (рисунок 5.10).

Рисунок 5.10 – Сторінка з панеллю для переходу на сторінку «Повідомлення»

На сторінці «Повідомлення» користувач має можливість написати повідомлення. Для цього потрібно ввести текст повідомлення у відповідне поле так, як це зображено на рисунку 5.11, потім обрати особу, якій необхідно надіслати повідомлення із випадаючого списку. Після цього повідомлення буде надіслане.

Рисунок 5.11 – Сторінка надсилання повідомлення

На сторінці «Повідомлення» користувач може переглянути свої вхідні повідомлення. Вони зображені на рисунку 5.12.

Рисунок 5.12 – Сторінка «Повідомлення» з відображеними вхідними повідомленнями

Для того, щоб перейти на сторінку, де можна розв’язати задачу про розподіл капіталовкладень на ремонт ділянок дороги треба натиснути на посилання «Розв’язати задачу про ремонт доріг» боковій панелі (рисунок 5.13).

Рисунок 5.13 – Сторінка з панеллю для переходу на сторінку «Розв’язати задачу про ремонт доріг»

Для того, щоб переглянути розв’язок задачі треба натиснути кнопку «Розв’язати». Будуть виведені розв’язки задачі жадібним та генетичним алгоритмом. Результати розв’язання задачі зображені на рисунку 5.14.

Рисунок 5.14 – Сторінка, на якій відображені результати розв’язку задачі про ремонт доріг

5.2 Випробування програмного продукту

У цьому підрозділі наведено опис тестів і порядок їх виконання для перевірки відповідності програмного забезпечення комплексу задач функціональним вимогам, представленим у технічному завданні на створення інформаційно-аналітичної системи адміністрування бюджетних програм місцевої влади на прикладі дорожнього будівництва.

5.2.1 Мета випробувань

Метою випробувань являється перевірка відповідності функцій інформаційно-аналітичної системи адміністрування бюджетних програм місцевої влади на прикладі дорожнього будівництва.

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

У таблицях 5.1 - 5.9 наведені випробування, які були проведені та їх результати.

Таблиця 5.1 – Перевірка функції «Увійти до свого аккаунту»

Мета тесту	Перевірка функції «Увійти до свого аккаунту»
Початковий стан системи	Відкрита сторінка «Логін».
Вхідні дані	Логін та пароль.
Схема проведення тесту	Ввести логін та пароль, натиснути кнопку «Увійти».
Очікуваний результат	Відкрита сторінка «Переглянути завдання», якщо користувач з даним логіном та паролем зареєстрований у системі, з’явилося повідомлення про те, що неможливо увійти, якщо незареєстрований.
Стан системи після випробування	Якщо користувач із введеним логіном та паролем зареєстрований – відкрита сторінка із завданнями, якщо незареєстрований – сторінка «Логін».

Таблиця 5.2 – Перевірка функції «Переглянути відсортовані завдання»

Мета тесту	Перевірка функції «Переглянути відсортовані завдання»
Початковий стан системи	Відкрита сторінка «Переглянути завдання».
Вхідні дані	Назва колонки, по якій необхідно відсортувати завдання.
Схема проведення тесту	Обрати назву колонки, по якій треба відсортувати завдання.

Продовження таблиці 5.2

Мета тесту	Перевірка функції «Переглянути відсортовані завдання»
Очікуваний результат	Відсортовані завдання по обраній колонці.
Стан системи після випробування	Відкрита сторінка «Переглянути завдання», на якій відображені відсортовані завдання.

Таблиця 5.3 – Перевірка функції «Переглянути відфільтровані завдання»

Мета тесту	Перевірка функції «Переглянути відфільтровані завдання»
Початковий стан системи	Відкрита сторінка «Переглянути завдання».
Вхідні дані	Текст для фільтрації.
Схема проведення тесту	Ввести текст, по якому треба відфільтрувати завдання. Натиснути кнопку «Відфільтрувати».
Очікуваний результат	Відображені відфільтровані завдання.
Стан системи після випробування	Відкрита сторінка «Переглянути завдання», на якій відображені відфільтровані завдання.

Таблиця 5.4 – Перевірка функції «Переглянути свої завдання»

Мета тесту	Перевірка функції «Переглянути свої завдання»
Початковий стан системи	Відкрита сторінка «Переглянути завдання».
Вхідні дані	-
Схема проведення тесту	Натиснути кнопку «Переглянути мої завдання».
Очікуваний результат	Відображені завдання, які призначені поточному користувачу.
Стан системи після випробування	Відкрита сторінка «Переглянути завдання», на якій відображені завдання, які призначені поточному користувачу.

Таблиця 5.5 – Перевірка функції «Призначити завдання»

Мета тесту	Перевірка функції «Призначити завдання»
Початковий стан системи	Відкрита сторінка «Призначити завдання».
Вхідні дані	Номер працівника та текст завдання.
Схема проведення тесту	Ввести номер працівника, якому буде призначене завдання та ввести текст завдання. Натиснути кнопку «Призначити завдання».
Очікуваний результат	Завдання призначене.
Стан системи після випробування	Відкрита сторінка «Призначити завдання», відображене повідомлення про те, що завдання призначене успішно.

Таблиця 5.6 – Перевірка функції «Відмітити завдання як виконане»

Мета тесту	Перевірка функції «Відмітити завдання як виконане»
Початковий стан системи	Відкрита сторінка «Переглянути завдання».
Вхідні дані	Номер завдання.
Схема проведення тесту	Ввести номер завдання та натиснути кнопку «Відмітити як виконане».
Очікуваний результат	Завдання відмічене як виконане.
Стан системи після випробування	Відкрита сторінка «Переглянути завдання».

Таблиця 5.7 – Перевірка функції «Переглянути свої повідомлення»

Мета тесту	Перевірка функції «Переглянути свої повідомлення»
Початковий стан системи	Відкрита сторінка «Повідомлення».

Продовження таблиці 5.7

Мета тесту	Перевірка функції «Переглянути відсортовані завдання»
Вхідні дані	-
Схема проведення тесту	Відкрити сторінку з повідомленнями.
Очікуваний результат	Відображені повідомлення поточного користувача.
Стан системи після випробування	Відкрита сторінка «Повідомлення», на якій відображені повідомлення користувача.

Таблиця 5.8 – Перевірка функції «Надіслати повідомлення»

Мета тесту	Перевірка функції «Надіслати повідомлення»
Початковий стан системи	Відкрита сторінка «Повідомлення».
Вхідні дані	Прізвище та ім’я користувача, якому необхідно надіслати повідомлення, текст повідомлення.
Схема проведення тесту	Ввести текст повідомлення та обрати користувача, якому необхідно надіслати повідомлення.
Очікуваний результат	Повідомлення надіслане.
Стан системи після випробування	Відкрита сторінка «Повідомлення».

Таблиця 5.9 – Перевірка функції «Переглянути розв’язок задачі про дороги»

Мета тесту	Перевірка функції «Переглянути розв’язок задачі про дороги»
Початковий стан системи	Відкрита сторінка «Розв’язати задачу».
Вхідні дані	Вартість ремонту кожної ділянки дороги.
Схема проведення тесту	Натиснути кнопку «Розв’язати задачу».

Продовження таблиці 5.9

Мета тесту	Перевірка функції «Переглянути відсортовані завдання»
Очікуваний результат	Відображений розв’язок задачі після розв’язання жадібним та генетичним алгоритмом.
Стан системи після випробування	Відкрита сторінка «Розв’язати задачу».

Висновок до розділу

У даному розділі була описана інструкція користувача для користування даною інформаційно-аналітичною системою, була визначена мета її випробувань та сформовані конкретні тести випробувань.

ЗАГАЛЬНІ ВИСНОВКИ

У розділі «Загальні положення» було описане предметне середовище даної інформаційно-аналітичної системи, її функціональна модель та процес діяльності, наведені існуючі аналоги, описане призначення розробки, а саме: цілі та задачі.

Під час написання розділу «Інформаційне забезпечення» була спроектована база даних, яка буде необхідна для даної інформаційно-аналітичної системи, було визначено, які вхідні дані необхідні для її роботи, а також вихідні дані, які будуть отримані в результаті роботи програми.

У розділі «Математичне забезпечення» розглянута задача про розподіл капіталовкладень на ремонт ділянок доріг. Визначено, які наближені алгоритми можна використати для її розв’язання та яким чином їх можна застосувати в рамках даної задачі.

Планується, що дана задача допоможе визначити, яким чином потрібно ремонтувати дороги, щоб зменшити час поїздки та мінімізувати втрати часу на поїздку.

У розділі «Програмне та технічне забезпечення» були описані засоби розробки програмного забезпечення, наведені їх переваги. Також були спроектовані діаграми класів, послідовності та компонентів для опису структури програми, наведені специфікації функцій, описані звіти, які виводить дане програмне забезпечення.

У розділі «Технологічний розділ» була описана інструкція користувача для користування даною інформаційно-аналітичною системою, була визначена мета її випробувань та сформовані конкретні тести випробувань.

ПЕРЕЛІК ПОСИЛАНЬ

1. Шолудько А. А. Інформаційна система адміністрування бюджетних програм місцевої влади на прикладі дорожнього будівництва/ Шолудько А. А., Попенко В. Д. // Матеріали IV всеукраїнської науково-практичної конференції молодих вчених та студентів «Інформаційні системи та технології управління» (ІСТУ-2020) – м. Київ.: НТУУ «КПІ ім. Ігоря Сікорського», 30 квітня 2020 р. – С. 91-94;
2. Цикл решений OODA [Електронний ресурс]. – 2016. – Режим доступу до ресурсу: https://4brain.ru/blog/%D1%86%D0%B8%D0%BA%D0%BB-%D1%80%D0%B5%D1%88%D0%B5%D0%BD%D0%B8%D0%B9-ooda/.
3. Бюджетний кодекс України [Електронний ресурс]. – 2010. – Режим доступу до ресурсу: https://www.gioc.kiev.ua/files/File/budjetniy_2010.htm.
4. Рівноприскорений рух. Прискорення. Швидкість тіла і пройдений шлях під час рівноприскореного прямолінійного руху. [Електронний ресурс]. – 2019. – Режим доступу до ресурсу: https://vseosvita.ua/library/rivnopriskorenij-ruh-priskorenna-svidkist-tila-i-projdenij-slah-pid-cas-rivnopriskorenogo-pramolinijnogo-ruhu-118370.html
5. Жадібні алгоритми. Переваги та недоліки [Електронний ресурс]. – 2016. – Режим доступу до ресурсу: https://lektsii.org/8-43572.html.
6. Генетичний алгоритм [Електронний ресурс]. – 2020. – Режим доступу до ресурсу: https://uk.wikipedia.org/wiki/%D0%93%D0%B5%D0%BD%D0%B5%D1%82%D0%B8%D1%87%D0%BD%D0%B8%D0%B9_%D0%B0%D0%BB%D0%B3%D0%BE%D1%80%D0%B8%D1%82%D0%BCl
7. Spring Boot tutorial [Електронний ресурс]. – 2020. – Режим доступу до ресурсу: https://www.journaldev.com/7969/spring-boot-tutorial

8. The good and the bad of Angular development [Електронний ресурс]. – 2020. – Режим доступу до ресурсу: https://www.altexsoft.com/blog/engineering/the-good-and-the-bad-of-angular-development/

Додаток А

Тексти програмного коду

Інформаційно-аналітична система адміністрування бюджетних програм місцевої влади на прикладі дорожнього будівництва

(Найменування програми (документа))

DVD-R

(Вид носія даних)

55 арк

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

Київ – 2020 року

```
package com.qucat.quiz.services;

@Slf4j
@Service
public class DataLoader {
    public ResultSet getInformation(String sql) throws SQLException {
        ResultSet rs = null;
        Connection c = null;
        try {
            Class.forName("oracle.jdbc.driver.OracleDriver");
            c = DriverManager.getConnection(
                    "jdbc:oracle:thin:@localhost:1521:xe", "system", "PASSWORD");
            Statement s = c.createStatement();
            c.setAutoCommit(false);
            rs = s.executeQuery(sql);
        } catch (ClassNotFoundException ex) {
            System.out.println(ex);
        } catch (SQLException ex) {
            if (c != null) {
                c.rollback();
                c.setAutoCommit(true);
            }
            while (ex != null) {
                System.out.println("Message : " + ex.getMessage());
                System.out.println("SQLState : " + ex.getSQLState());
                System.out.println("ErrorCode : " + ex.getErrorCode());
                ex = ex.getNextException();
            }
        }
        return rs;
    }

    public ResultSet getInformationWithParameters(String sql, String id) throws
SQLException {
        ResultSet rs = null;
        Connection c = null;
        try {
            Class.forName("oracle.jdbc.driver.OracleDriver");
            c = DriverManager.getConnection(
                    "jdbc:oracle:thin:@localhost:1521:xe", "system", "PASSWORD");
            PreparedStatement stmt = c.prepareStatement(sql);

            stmt.setString(1, id);

            stmt.executeUpdate();
            c.setAutoCommit(false);
            rs = stmt.executeQuery(sql);
        } catch (ClassNotFoundException ex) {
            System.out.println(ex);
        } catch (SQLException ex) {
            if (c != null) {
                c.rollback();
                c.setAutoCommit(true);
            }
            while (ex != null) {
                System.out.println("Message : " + ex.getMessage());
                System.out.println("SQLState : " + ex.getSQLState());
```

```
                System.out.println("ErrorCode : " + ex.getErrorCode());
                ex = ex.getNextException();
            }
        }
        return rs;
    }

    public void insertData(String sql) throws SQLException{
        Connection c = null;
        try {
            Class.forName("oracle.jdbc.driver.OracleDriver");
            c = DriverManager.getConnection(
                    "jdbc:oracle:thin:@localhost:1521:xe", "system", "PASSWORD");
            Statement s = c.createStatement();
            c.setAutoCommit(false);
            s.executeUpdate(sql);
            c.commit();

        } catch (ClassNotFoundException ex) {
            System.out.println(ex);
        } catch (SQLException ex) {
            if (c != null) {
                c.rollback();
                c.setAutoCommit(true);
            }
            while (ex != null) {
                System.out.println("Message : " + ex.getMessage());
                System.out.println("SQLState : " + ex.getSQLState());
                System.out.println("ErrorCode : " + ex.getErrorCode());
                ex = ex.getNextException();
            }
        }
    }
}
```

```
package com.qucat.quiz.services;

import lombok.extern.slf4j.Slf4j;
import org.springframework.stereotype.Service;

import java.io.File;
import java.io.FileNotFoundException;
import java.io.FileReader;
import java.io.IOException;
import java.util.*;

@Slf4j
@Service
public class RoadsTask {
    double[] totalLoss = new double[2];
    private final int s = 1000;
    private final int budget = 1500;
    private int[][] beginCost;//початкові дані
    private int[][] beginCostGenetic;//початкові дані
    private int[][] cost;//вартість

    final int POPULATION_SIZE = 6; //к-сть осіб
    final int PIECE_COUNT = 10;//початкова к-сть ділянок дороги
    private int PIECE_COUNT;//початкова к-сть ділянок дороги
```

```
    private void readData(String fileName) {
        try {
            File myObj = new File(fileName);
            Scanner myReader = new Scanner(myObj);
            while (myReader.hasNextLine()) {
                String data = myReader.nextLine();
                System.out.println(data);
                String[] readData = data.split(" ");
                PIECE_COUNT = readData.length;
                cost = new int[readData.length][2];
                beginCost = new int[readData.length][2];
                beginCostGenetic = new int[readData.length][2];
                for (int i = 0; i < readData.length; i++) {
                    System.out.println(readData[i]);
                    cost[i][0] = Integer.parseInt(readData[i]);
                    cost[i][1] = i;
                    beginCost[i][0] = Integer.parseInt(readData[i]);
                    beginCost[i][1] = i;
                    beginCostGenetic[i][0] = Integer.parseInt(readData[i]);
                    beginCostGenetic[i][1] = i;
                }
            }
            myReader.close();
        } catch (FileNotFoundException e) {
            System.out.println("An error occurred.");
            e.printStackTrace();
        }
    }

    private double getTimeHalmuvan(double vMax, double a, int sHalmuv) {
        double tHalmuv = 0, discriminant = 0;
        discriminant = vMax * vMax - 4 * (a / 2) * sHalmuv;
        tHalmuv = (-vMax + Math.sqrt(discriminant)) / (2 * (-a / 2));
        return tHalmuv;
    }

    private double calculateWasteOfTime(List<Integer> answer, int[][] beginCost) {
        double result;
        List<ArrayList<Integer>> afterRepair = mergePartsOfRoad(answer, beginCost);
        double perehodyCount = afterRepair.size() - 1;
        int halmCount = 0, rozhCount = 0;

        double tRozhony = 0, tHalmuv = 0, tPovRuhu = 0;
        double vtrRozhony = 0, vtrHalmuv = 0, totalVtrPovRuhu = 0;
        double vMax = 30.56; //м/с
        double vMin = 8.33;
        double a = 1.5;
        tRozhony = (vMax - vMin) / a;
        vtrRozhony = (s / vMax) - tRozhony;
        int sG = 10;
        tHalmuv = getTimeHalmuvan(vMax, a, sG);
        vtrHalmuv = (s / vMax) - tHalmuv;


        for (int j = 0; j < afterRepair.size(); j++) {
            if (afterRepair.get(j).get(2) == 1) {
                totalVtrPovRuhu += (afterRepair.get(j).get(0) / vMin) -
(afterRepair.get(j).get(0) / vMax);// для кожної ділянки своя втрата
            }
        }
```

```
        result = (perehodyCount / 2) * vtrRozhony + (perehodyCount / 2) * vtrHalmuv +
totalVtrPovRuhu;
        return result;

    }
    private synchronized List<Integer> greedy(int[][] cost) {
        List<Integer> answer = new ArrayList<>();
        int sum = 0;

        boolean isSorted = false;
        int buf, buf2;
        while (!isSorted) {
            isSorted = true;
            for (int i = 0; i < cost.length - 1; i++) {
                if (cost[i][0] > cost[i + 1][0]) {
                    isSorted = false;
                    buf = cost[i][0];
                    buf2 = cost[i][1];
                    cost[i][0] = cost[i + 1][0];
                    cost[i][1] = cost[i + 1][1];
                    cost[i + 1][0] = buf;
                    cost[i + 1][1] = buf2;
                }
            }
        }
        int ind = 0;
        while (sum <= (budget)) {
            if (ind < cost.length) {
                if (cost[ind][0] != 0) {
                    sum += cost[ind][0];
                    if (sum > budget) sum -= cost[ind][0];
                    else {
                        System.out.print(cost[ind][1] + " ");//номери ділянок, які
треба відремонтувати
                        answer.add(cost[ind][1]);
                    }
                }
                ind++;
            } else break;
        }
        return answer;
    }

    private synchronized List<ArrayList<Integer>> mergePartsOfRoad(List<Integer>
answer, int[][] beginCost) {
        List<ArrayList<Integer>> mergedParts = new ArrayList();//номер ділянки і
довжина і вартість (відремонтована чи ні)
        for (int i = 0; i < beginCost.length; i++) {
            for (int j = 0; j < answer.size(); j++) {
                if (beginCost[i][1] == answer.get(j)) {
                    beginCost[i][0] = 0;
                }
            }
        }
        int len = s, j = 0;
        for (int i = 0; i < beginCost.length - 1; i++) {
            if (beginCost[i][0] == 0 && beginCost[i + 1][0] == 0) {
                len += s;
            } else if (beginCost[i][0] != 0 && beginCost[i + 1][0] != 0) {
                len += s;
```

```
            } else {
                ArrayList<Integer> arr1 = new ArrayList<>();
                arr1.add(len);
                arr1.add(j);
                if (beginCost[i][0] == 0) {
                    arr1.add(0);
                } else {
                    arr1.add(1);
                }
                mergedParts.add(arr1);
                j++;
                len = s;
            }
        }
        ArrayList<Integer> arr1 = new ArrayList<>();
        arr1.add(len);
        arr1.add(j);
        if (beginCost[beginCost.length - 1][0] == 0) {
            arr1.add(0);
        } else {
            arr1.add(1);
        }
        mergedParts.add(arr1);
        return mergedParts;
    }

    private int[][] generatePopulation() {
        Random rnd = new Random();
        int[][] population = new int[POPULATION_SIZE][PIECE_COUNT];
        for (int i = 0; i < POPULATION_SIZE; i++) {
            for (int j = 0; j < PIECE_COUNT; j++) {
                population[i][j] = rnd.nextInt(2);
            }
        }
        return population;
    }

    public List<List<Integer>> geneticAlgorithm() {
        List<List<Integer>> answerGeneticParts = new ArrayList<>();
        List<List<Integer>> answerGenetic = new ArrayList<>();
        Random rnd = new Random();
        int[][] population = generatePopulation();
        Integer[][] children = new Integer[POPULATION_SIZE][PIECE_COUNT];
        //створюємо нащадків
        for (int i = 0; i < POPULATION_SIZE; i += 2) {
            for (int j = 0; j < PIECE_COUNT; j++) {
                if (j < (PIECE_COUNT / 2)) {
                    children[i][j] = population[i][j];
                    children[i + 1][j] = population[i + 1][j];
                } else {
                    children[i][j] = population[i + 1][j];
                    children[i + 1][j] = population[i][j];
                }
            }
        }
        //мутація рандомний індекс
        for (int i = 0; i < POPULATION_SIZE; i += 2) {
            int index = rnd.nextInt(PIECE_COUNT);
            if (children[i][index] == 0) {
                children[i][index] = 1;
```

```
            } else {
                children[i][index] = 0;
            }
        }
        //обчислення вартості ремонту
        int[] sum = new int[POPULATION_SIZE];
        for (int i = 0; i < POPULATION_SIZE; i++) {
            sum[i] = 0;
            for (int j = 0; j < PIECE_COUNT; j++) {
                sum[i] += children[i][j] * cost[j][0];
            }
        }
        //відкидання тих, вартість яких перевищує бюджет
        for (int i = 0; i < POPULATION_SIZE; i++) {
            if (sum[i] <= budget) {
                answerGenetic.add(Arrays.asList(children[i]));
            }
        }

        //відбираємо номери ділянок
        for (int i = 0; i < answerGenetic.size(); i++) {
            List<Integer> parts = new ArrayList<>();
            for (int j = 0; j < answerGenetic.get(i).size(); j++) {
                if (answerGenetic.get(i).get(j) == 1 && beginCost[j][0] != 0) {
                    parts.add(j);
                }
            }
            if (!parts.isEmpty()) {
                answerGeneticParts.add(parts);
            }
        }
        if (answerGeneticParts.isEmpty()) {
            geneticAlgorithm();
        }
        return answerGeneticParts;
    }

    public synchronized List<Integer> chooseAnswer() {
        double min = 100000000;
        int indexAnswer = 0;
        List<List<Integer>> answerGenetic = geneticAlgorithm();

        double[] getTotalVtraty = new double[answerGenetic.size()];
        for (int i = 0; i < answerGenetic.size(); i++) {
            readData("road.txt");
            getTotalVtraty[i] = calculateWasteOfTime(answerGenetic.get(i),
beginCostGenetic);
            System.out.println(getTotalVtraty[i]);
            if (getTotalVtraty[i] < min) {
                min = getTotalVtraty[i];
                indexAnswer = i;
            }
        }
        totalLoss[1] = min;
        return answerGenetic.get(indexAnswer);
    }

    private synchronized List<Solution> getAnswerWithMark(List<Integer> answer,
int[][] cost, int[][] beginCost) {
        List<Solution> answerWithMark = new ArrayList<>();
```

```
            readData("road.txt");
            for (int i = 0; i < cost.length; i++) {
                answerWithMark.add(new Solution(i, false));
                if (beginCost[i][0] == 0) {
                    answerWithMark.set(i, new Solution(i, true));
                }
            }

            for (int i = 0; i < answerWithMark.size(); i++) {
                for (double d : answer) {
                    if (answerWithMark.get(i).getId() == d) {
                        answerWithMark.set(i, new Solution(i, true));
                    }
                }
            }

            for (Solution solution : answerWithMark) {
                System.out.println(solution.getId() + " " + solution.isRepair());
            }
            return answerWithMark;
        }


        public List<Solution> showAnswer() {
            readData("road.txt");
            List<Integer> answer = greedy(cost);
            double totalVtraty = calculateWasteOfTime(answer, beginCost);
            totalLoss[0] = totalVtraty;
            System.out.println("totalLoss greedy" + totalLoss[0]);
            return getAnswerWithMark(answer, cost, beginCost);
        }

        public synchronized List<Solution> showGeneticAnswer() {
            readData("road.txt");
            List<Integer> gen = chooseAnswer();
            return getAnswerWithMark(gen, cost, beginCost);
        }

        public double[] getTotalLoss() {
            return totalLoss;
        }

    }
    package com.qucat.quiz.services;

    @Slf4j
    @Service
    public class TaskService {

        @Autowired
        private DataLoader dataLoader;


        public void markAsCompleted(String id) {
            try {
                DateTimeFormatter dtf = DateTimeFormatter.ofPattern("dd-MM-yyyy");
                LocalDateTime date = LocalDateTime.now();
                String currentDate = dtf.format(date);
                dataLoader.insertData("update task set completion_date=TO_DATE('" +
    currentDate + "', 'dd-mm-yyyy') where id=" + id);
```

```
            System.out.println("marked");
        } catch (SQLException e) {
            e.printStackTrace();
        }
    }

    private List<Task> getTasks(String sql) {
        List<Task> tasks = new ArrayList<Task>();
        DataLoader dataLoader=new DataLoader();
        try {
            ResultSet rset = dataLoader.getInformation(sql);
            while (rset.next()) {
                Task task = new Task(rset.getInt("id"),
                        rset.getString("first_name"),
                        rset.getString("last_name"),
                        rset.getString("text"),
                        rset.getDate("assignment_date"),
                        rset.getDate("completion_date"),
                        rset.getString("role")
                );
                tasks.add(task);

            }
        } catch (SQLException ex) {
        }
        return tasks;
    }

    public List<Task> getAllTasks() {
        List<Task> tasks = getTasks("select t.id, p.first_name,p.last_name, p.role,
t.text,t.assignment_date,t.completion_date from person p inner join task t on
t.person_id=p.id");
        return tasks;
    }

    public List<Task> getSortedTasks(String column) {
        List<Task> tasks = getTasks("select t.id, p.first_name,p.last_name, p.role,
t.text,t.assignment_date,t.completion_date from person p inner join task t on
t.person_id=p.id order by " + column);

        return tasks;
    }

    public List<Task> getFilteredTasks(String filterText) {
        List<Task> tasksFilterId = getTasks("select t.id, p.first_name,p.last_name,
p.role, t.text,t.assignment_date,t.completion_date from person p inner join task t on
t.person_id=p.id where t.id like '%" + filterText + "%'");
        List<Task> tasksFilterFirstName = getTasks("select t.id,
p.first_name,p.last_name, p.role, t.text,t.assignment_date,t.completion_date from
person p inner join task t on t.person_id=p.id where p.first_name like '%" +
filterText + "%'");
        List<Task> tasksFilterLastName = getTasks("select t.id,
p.first_name,p.last_name, p.role, t.text,t.assignment_date,t.completion_date from
person p inner join task t on t.person_id=p.id where p.last_name like '%" +
filterText + "%'");
        List<Task> tasksFilterText = getTasks("select t.id, p.first_name,p.last_name,
p.role, t.text,t.assignment_date,t.completion_date from person p inner join task t on
t.person_id=p.id where t.text like '%" + filterText + "%'");
        List<Task> tasksFilterCompletionDate = getTasks("select t.id,
p.first_name,p.last_name, p.role, t.text,t.assignment_date,t.completion_date from
```

```
person p inner join task t on t.person_id=p.id where t.completion_date like '%" +
filterText + "%'");
        List<Task> tasksFilterAssignmentDate = getTasks("select t.id,
p.first_name,p.last_name, p.role, t.text,t.assignment_date,t.completion_date from
person p inner join task t on t.person_id=p.id where t.assignment_date like '%" +
filterText + "%'");
        List<Task> filteredTasks = new ArrayList<>();
        filteredTasks.addAll(tasksFilterId);
        filteredTasks.addAll(tasksFilterFirstName);
        filteredTasks.addAll(tasksFilterLastName);
        filteredTasks.addAll(tasksFilterText);
        filteredTasks.addAll(tasksFilterAssignmentDate);
        filteredTasks.addAll(tasksFilterCompletionDate);
        return filteredTasks;
    }

    public List<Task> getMyTasks(String id) {
        List<Task> tasks = new ArrayList<Task>();
        DataLoader dataLoader = new DataLoader();
        try {
            ResultSet rset = dataLoader.getInformationWithParameters("select t.id,
p.first_name,p.last_name, p.role, t.text,t.assignment_date,t.completion_date from
person p inner join task t on t.person_id=p.id where p.id=?", id);

            while (rset.next()) {
                Task task = new Task(rset.getInt("id"),
                        rset.getString("first_name"),
                        rset.getString("last_name"),
                        rset.getString("text"),
                        rset.getDate("assignment_date"),
                        rset.getDate("completion_date"),
                        rset.getString("role")
                );
                System.out.println(task.getFirstName() + " " + task.getLastName());
                tasks.add(task);
            }
        } catch (SQLException ex) {
        }
        return tasks;
    }

    public void addTask(String id, String text) {
        DateTimeFormatter dtf = DateTimeFormatter.ofPattern("dd-MM-yyyy");
        LocalDateTime date = LocalDateTime.now();
        String currentDate = dtf.format(date);
        System.out.println(currentDate);
        String sql = "insert into
task(id,text,assignment_date,completion_date,person_id)\n" +
                "values (1,'" + text + "',TO_DATE('" + currentDate + "', 'dd-mm-
yyyy'),null," + id + ")";
        try {
            dataLoader.insertData(sql);
        } catch (SQLException e) {
            e.printStackTrace();
        }
    }

}
package com.qucat.quiz.services;
```

```
@Slf4j
@Service
public class TextService {
    @Autowired
    private DataLoader dataLoader;

    public List<Text> getMyMessage(String userId) {
    List<Text> messages = new ArrayList<Text>();
    DataLoader dataLoader=new DataLoader();
    try {
        String sql = "select p.id as id, m.text as text, TO_TIMESTAMP(m.date_send) as
date_send," +
                "p.first_name first_name, p.last_name as last_name from message m
inner join person p on p.id=m.person_id_from where m.person_id_to=" + userId
                + " order by m.date_send desc"; //to_char(m.date_send,'DD/MM/YYYY
HH24:MI:SS')
        ResultSet rset = dataLoader.getInformation(sql);

        while (rset.next()) {
            Text message = new Text(
                    rset.getInt("id"),
                    rset.getString("text"),
                    rset.getTimestamp("date_send"),
                    rset.getString("first_name"),
                    rset.getString("last_name")
            );
            messages.add(message);
        }
    } catch (SQLException ex) {
    }
    return messages;
}

public void sendMessage(String idTo, String idFrom, String text) {
    String currentDate = new SimpleDateFormat("dd/MM/yyyy HH:mm:ss").format(new
Date());
    try {
        String sqlSend = "insert into message
(id,text,date_send,person_id_to,person_id_from)\n" +
                "values (1,'" + text + "',TO_DATE('" + currentDate + "', 'dd-mm-yyyy
HH24:MI:SS')," + idTo + "," + idFrom + ")";
        dataLoader.insertData(sqlSend);
    } catch (SQLException e) {
        e.printStackTrace();
    }
    System.out.println("sent");
}

}
package com.qucat.quiz.services;

@Slf4j
@Service
public class PersonService {

    @Autowired
    private DataLoader dataLoader;

    public Person getUser(String login, String password) {
        DataLoader dataLoader=new DataLoader();
```

```
        Person user = null;
        ResultSet rset = null;
        try {
            rset = dataLoader.getInformation("select * from person where login='" +
login + "'");

            if (!rset.isBeforeFirst()) {
                System.out.print("incorrect login");
            }
            while (rset.next()) {
                user = new Person(rset.getInt("id"),
                        rset.getString("login"),
                        rset.getString("password"),
                        rset.getString("first_name"),
                        rset.getString("last_name"),
                        rset.getString("role")
                );

                System.out.println(user.getId() + "  " + user.getLogin() + " " +
user.getFirstName() + " " + user.getLastName() + " " + user.getRole());
                if (password.equals(user.getPassword())) {
                    System.out.println("You have successfully logged");
                    return user;
                } else {
                    System.out.println("Password is incorrect");
                    return null;
                }
            }
        } catch (SQLException e) {
            e.printStackTrace();
        }
        return user;
    }

    public List<Person> getAllUsers() {
        DataLoader dataLoader=new DataLoader();
        String sql = "select * from person";
        List<Person> users = new ArrayList<Person>();
        try {
            ResultSet rset = dataLoader.getInformation(sql);
            while (rset.next()) {
                Person user = new Person(rset.getInt("id"),
                        rset.getString("login"),
                        rset.getString("password"),
                        rset.getString("first_name"),
                        rset.getString("last_name"),
                        rset.getString("role")
                );
                users.add(user);

            }
        } catch (SQLException ex) {
        }
        return users;
    }
}
package com.qucat.quiz.controllers;

@RestController
@RequestMapping("api/v1")
```

```
public class TaskController {
    @Autowired
    private TaskService taskService;

    @Autowired
    private RoadsTask roadsTask;


    @GetMapping("/task")
    public List<Task> getTasks() {
        return taskService.getAllTasks();
    }

    @GetMapping("/task/sorted")
    public List<Task> getSortedTasks(@RequestParam(value = "column") String column) {
        return taskService.getSortedTasks(column);
    }

    @GetMapping("/task/filtered")
    public List<Task> getFilteredTasks(@RequestParam(value = "filterText") String
filterText) {
        return taskService.getFilteredTasks(filterText);
    }

    @GetMapping("/task/my/{id}")
    public List<Task> getMyTasks(@PathVariable String id) {
        return taskService.getMyTasks(id);
    }

    @PostMapping("task/add")
    public void addTask(@RequestBody String[] text) {
        taskService.addTask(text[0], text[1]);
    }

    @PutMapping("task/complete")
    public void markAsCompleted(@RequestBody String id) {
        taskService.markAsCompleted(id);
    }

    @GetMapping("task/roads")
    public List<Solution> getSolutionRoads() {
        return roads = roadsTask.showAnswer();
    }

    @GetMapping("task/roads/totalLoss")
    public double[] getTotalLoss() {
        return roads = roadsTask.getTotalLoss();
    }
}
package com.qucat.quiz.controllers;

@RestController
@RequestMapping("api/v1")
public class MessageController {

    @Autowired
    private TextService textService;

    @GetMapping("/message/{id}")
    public List<Text> getMessages(@PathVariable String id) {
```

```
            return textService.getMyMessage(id);
        }

        @PostMapping("/message/send/{idTo}/{idFrom}")
        public void updateUser(@PathVariable String idTo,
                               @PathVariable String idFrom,
                               @RequestBody String message) {
            textService.sendMessage(idTo, idFrom, message);
        }
    }
    package com.qucat.quiz.controllers;

    @RestController
    @RequestMapping("api/v1")
    public class PersonController {

        @Autowired
        private PersonService personService;

        @GetMapping("/task/login")
        public Person getUserByLogin(@RequestParam(value = "login") String login,
                                     @RequestParam(value = "password") String password) {
            return personService.getUser(login, password);
        }

        @GetMapping("/users")
        public List<Person > getAllUsers() {
            return personService.getAllUsers();
        }

    }

    import {Injectable} from '@angular/core';
    @Injectable({
      providedIn: 'root'
    })

    export class TaskService {
      private userId: number;
      private role: string;
      httpOptions = {
        headers: new HttpHeaders({'Content-Type': 'application/json'})
      };

      constructor(private http: HttpClient) {
      }

      getTasks(): Observable<Task[]> {
        return this.http.get<Task[]>(`${url}/task`)
          .pipe(
            catchError(this.handleError<Task[]>([]))
          );
      }


      getSortedTasks(column: string): Observable<Task[]> {
        return this.http.get<Task[]>(`${url}/task/sorted?column=${column}`)
          .pipe(
            catchError(this.handleError<Task[]>([]))
          );
```

```
  }

  getFilteredTasks(filterText: string): Observable<Task[]> {
    return this.http.get<Task[]>(`${url}/task/filtered?filterText=${filterText}`)
      .pipe(
        catchError(this.handleError<Task[]>([]))
      );
  }

  getMyTasks(id: string): Observable<Task[]> {
    return this.http.get<Task[]>(`${url}/task/my/${id}`)
      .pipe(
        catchError(this.handleError<Task[]>([]))
      );
  }
  addTask(text: string[]) {
    return this.http.post<string[]>(`${url}/task/add/`, text, this.httpOptions);
  }

  markAsCompleted(id: string) {
    return this.http.put<string>(`${url}/task/complete/`, id, this.httpOptions);
  }

  getSolutionRoads(): Observable<Solution[]> {
    return this.http.get<Solution[]>(`${url}/task/roads`);
  }

  getTotalLoss(): Observable<number[]> {
    return this.http.get<number[]>(`${url}/task/roads/totalLoss`);
  }

  private handleError<T>(result?: T) {
    return (error: any): Observable<T> => {

      console.error(error);

      return of(result as T);
    };
  }
}

import {Injectable} from '@angular/core';
@Injectable({
  providedIn: 'root'
})

export class MessageService {
  private userId: number;
  private role: string;
  httpOptions = {
    headers: new HttpHeaders({'Content-Type': 'application/json'})
  };

  constructor(private http: HttpClient) {
  }


  getMessages(id: string): Observable<Text[]> {
    return this.http.get<Text[]>(`${url}/message/${id}`)
      .pipe(
```

```
        catchError(this.handleError<Text[]>([]))
      );
  }
  sendMessage(text: string, idTo: string, idFrom: string) {
    return this.http.post<string>(`${url}/message/send/${idTo}/${idFrom}`, text,
this.httpOptions);
  }

  private handleError<T>(result?: T) {
    return (error: any): Observable<T> => {
      console.error(error);
      return of(result as T);
    };
  }
}
import {Injectable} from '@angular/core';
@Injectable({
  providedIn: 'root'
})

export class PersonService {
  private userId: number;
  private role: string;
  httpOptions = {
    headers: new HttpHeaders({'Content-Type': 'application/json'})
  };

  constructor(private http: HttpClient) {
  }
  setUserId(userId: number) {
    this.userId = userId;
  }

  getUserId(): number {
    return this.userId;
  }

  setRole(role: string) {
    this.role = role;
  }

  getRole(): string {
    return this.role;
  }

  getUser(login: string, password: string): Observable<Person> {

    return
this.http.get<Person>(`${url}/task/login?login=${login}&password=${password}`)
      .pipe(
        catchError(this.handleError<Person>(null))
      );
  }

  getAllUsers(): Observable<Person[]> {

    return this.http.get<Person[]>(`${url}/users`)
      .pipe(
        catchError(this.handleError<Person[]>(null))
      );
```

```
  }

  private handleError<T>(result?: T) {
    return (error: any): Observable<T> => {
      console.error(error);
      return of(result as T);
    };
  }
}

import {Component, HostListener, Input, OnInit} from '@angular/core';
@Component({
  selector: 'app-account',
  templateUrl: './account.component.html',
})
export class AccountComponent implements OnInit {

  messages: Text[] = [];
  user: Person;
  userId: number;

  constructor(private personService: PersonService,
              private router: Router
  ) {}
  loginInAccount(login: string, password: string): void {
    this.personService.getUser(login, password).subscribe(data => {
      if (data != null) {
        console.log("data " + data);
        this.user = data;
        this.personService.setUserId(this.user.id);
        this.personService.setRole(this.user.role);
        this.router.navigate(['task']).then();
      } else {
        console.log("can`t login");
      }
    });
  }

  ngOnInit(): void {}

}
import {Component, HostListener, Input, OnInit} from '@angular/core';

@Component({
  selector: 'app-assign',
  templateUrl: './assign-task.component.html',
})
export class AssignTaskComponent implements OnInit {

  taskNew: string[]=[];
  tasks: Task[] = [];
  users: User[] = [];
  userId: number;


  constructor(private taskService: TaskService
  ) {}
  assignTask(id: string, text:string) {

    this.taskNew[0] = id;
```

```
    this.taskNew[1] = text;
    this.taskService.addTask(this.taskNew).subscribe(
      get => {
      },
      error => {
        console.log(error);
      });
  }
  ngOnInit(): void {}

}
import {Component, HostListener, Input, OnInit} from '@angular/core';
@Component({
  selector: 'app-view-activities',
  templateUrl: './view-message.component.html',
})
export class ViewMessageComponent implements OnInit {

  messages: Text[] = [];
  users: Person[] = [];
  userId: number;
  role: string;
  constructor(private messageService: MessageService,
              private taskService: TaskService,
              private personService: PersonService
  ) {

  }

  getAllUsers(): void {
    this.personService.getAllUsers().subscribe(activities => {
        if (activities.length == 0) {
          return;
        }
        this.users = this.users.concat(activities);
      },
      err => {
        console.log(err);
      });
  }


  getMessages(): void {
    this.messageService.getMessages(this.userId.toString())
      .subscribe(
        activities => {
          console.log(activities.length);
          if (activities.length == 0) {

            return;
          }
          this.messages = this.messages.concat(activities);
        },
        err => {
          console.log(err);
        });

  }

  sendMessage(idTo: string, text: string): void {
```

```
      this.messageService.sendMessage(text, idTo,
    this.userId.toString()).subscribe(data => {
      });

    }

    ngOnInit(): void {
      this.userId = this.personService.getUserId();
      this.role = this.personService.getRole();
      this.getMessages();
      this.getAllUsers();
    }
  }
  import {Component, HostListener, Input, OnInit} from '@angular/core';
  @Component({
    selector: 'app-view-activities',
    templateUrl: './view-task.component.html',
  })
  export class ViewTasksComponent implements OnInit {

    tasks: Task[] = [];
    users: User[] = [];
    userId: number;
    role:string;
    columnsForSorting = [
      {
        text:"Ім'я",
        columnName:"p.first_name"
      },
      {
        text:"Прізвище",
        columnName:"p.last_name"
      },
      {
        text:"Текст завдання",
        columnName:"t.text"
      },

      {
        text:"Дата призначення",
        columnName:"t.assignment_date"
      },

      {
        text:"Дата виконання",
        columnName:"t.completion_date"
      }
    ];
    constructor(private taskService: TaskService,
                private personService: PersonService
    ) {

    }

    getTasks(): void {

      this.taskService.getTasks()
        .subscribe(
          activities => {
            if (activities.length == 0) {
```

```
            return;
          }

          this.tasks = this.tasks.concat(activities);
        },
        err => {
          console.log(err);
        })

    }
    public getSortedTasks(columnName:string) {
      this.tasks = [];

      this.taskService.getSortedTasks(columnName)
        .subscribe(
          activities => {
            if (activities.length == 0) {
              return;
            }

            this.tasks = this.tasks.concat(activities);
          },
          err => {
            console.log(err);
          })
    }
    public getFilteredTasks(filterText: string) {
      this.tasks = [];

      this.taskService.getFilteredTasks(filterText)
        .subscribe(
          activities => {
            if (activities.length == 0) {
              return;
            }

            this.tasks = this.tasks.concat(activities);
          },
          err => {
            console.log(err);
          })
    }
    public getMyTasks() {
      this.tasks = [];

      this.taskService.getMyTasks(this.userId.toString())
        .subscribe(
          activities => {
            if (activities.length == 0) {
              return;
            }

            this.tasks = this.tasks.concat(activities);
            console.log(this.tasks[0].firstName);
          },
          err => {
            console.log(err);
          })
    }
    markAsCompleted(id: string) {
```

```
    this.taskService.markAsCompleted(id).subscribe(
      get => {
      },
      error => {
        console.log(error);
      });
  }

  ngOnInit(): void {
    this.userId=this.personService.getUserId();
    this.role=this.personService.getRole();
    this.getTasks();
  }
}
import {Component, HostListener, Input, OnInit} from '@angular/core';
@Component({
  selector: 'app-assign',
  templateUrl: './view-taskRoad.component.html',
})
export class ViewTaskRoadComponent implements OnInit {

taskNew: string[] = [];
tasks: Task[] = [];
users: User[] = [];
userId: number;
solution: Solution[] = [];
solutionGenetic: Solution[] = [];
color: boolean[] = [];
timeLoss: number[] = [];

constructor(private taskService: TaskService
) {

}
ngOnInit(): void {
  this.solution=[];
  this.solutionGenetic=[];
  this.timeLoss=[];
  this.taskService.getSolutionRoads().subscribe(data => {
    console.log(data);
    this.solution = data;
  });
  setTimeout(() => {
    this.taskService.getSolutionRoadsByGeneticAlgorithm().subscribe(data => {
      console.log(data);
      this.solutionGenetic = data;

      setTimeout(() => {
        this.taskService.getTotalLoss().subscribe(data => {
          console.log(data);
          this.timeLoss = data;
        });
      }, 500);
    });
    }, 500);

}
```

НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ

“КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ ІМЕНІ ІГОРЯ СІКОРСЬКОГО”

Кафедра автоматизованих систем обробки інформації і управління

УЗГОДЖЕНО

Керівник проекту

_____ *В. Д. Попенко*

(підпис) (ініціали, прізвище)

“10” травня 2020 р.

ЗАТВЕРДЖУЮ

В.о. завідувача кафедри

_____ *О.А. Павлов*

(підпис) (ініціали, прізвище)

“11” травня 2020 р.

Інформаційно-аналітична система адміністрування бюджетних програм місцевої влади на прикладі дорожнього будівництва

ТЕХНІЧНЕ ЗАВДАННЯ

Шифр *ДП ІС-6129.000 ТЗ*

на 8 сторінках

Київ – 2020 року

ЗМІСТ

					ДП ІС-6129.01.000 ТЗ		
Зм.	Арк.	Прізвище	Підпис	Дата			
Розроб.		Шолудько А. А.			Інформаційно-аналітична система адміністрування бюджетних програм місцевої влади на прикладі дорожнього будівництва	Літ.	Лист 2 / Листів 8
Перевірив.		Попенко В. Д.					
Н. кон.		Тєлишева Т.О.				КПІ ім. ІгоряСікорського кафедра АСОІУ гр. ІС-61	
Затв.		Павлов О.А.					

1 ЗАГАЛЬНІ ПОЛОЖЕННЯ

1.1 Повне найменування системи та її умовне позначення.

Повна назва системи: Інформаційно-аналітична система адміністрування бюджетних програм місцевої влади на прикладі дорожнього будівництва.

Умовне позначення: ІАС АБПНПДБ.

1.2 Найменування організації-замовника та організацій-учасників робіт

Генеральним замовником системи є кафедра Автоматизованих систем обробки інформації та управління НТУУ "КПІ" імені Ігоря Сікорського. Попенко Володимир Дмитрович – представник замовника

Розробником системи є студента групи ІС-61 факультету інформатики та обчислювальної техніки НТУУ "КПІ" імені Ігоря Сікорського Шолудько Анна.

1.3 Перелік документів, на підставі яких створюється система

Система створюється на основі таких документів:

- ДСТУ 19.201-78. Технічне завдання, вимоги до змісту та оформлення.

1.4 Планові терміни початку і закінчення роботи зі створення системи

Плановий термін початку роботи зі створення системи – 15 квітня 2020 року.

Плановий термін закінчення роботи зі створення системи – 12 червня 2020 року.

2 ПРИЗНАЧЕННЯ І ЦІЛІ СТВОРЕННЯ СИСТЕМИ

2.1 Призначення системи

Дана інформаційна система призначена для призначення завдань керуючим своїм підлеглим а також дає можливість здійснювати наступні функції:

- сортувати завдання по різних критеріях, а саме дата призначення завдання, співробітник, якому призначене завдання, виконане завдання чи ні;
- фільтрувати завдання по тих самих критеріях;
- обмінюватися повідомленнями.

Також інформаційна система дозволяє розв’язати задачу про дороги, а саме визначити, які ділянки дороги необхідно відремонтувати, щоб вкластися у виділений бюджет та мінімізувати кількість незадоволених людей.

2.2 Цілі створення системи

Цілі створення системи:

- полегшити процес розподілу завдань між працівниками компанії;
- створити засіб контролю стану бізнес-процесів в організації;
- надати можливість швидко формувати звіт по виконаній роботі;
- надати можливість швидко переглядати завдання конкретного працівника, їх виконання;
- підвищити ефективність капіталовкладень у ремонт доріг.

3 ХАРАКТЕРИСТИКА ОБ'ЄКТА АВТОМАТИЗАЦІЇ

Об'єктом автоматизації є підтримка розподілу завдань між співробітниками в компанії, а також розв'язання задачі про розподіл капіталовкладень на ремонт ділянок доріг.

У рамках великої фірми чи компанії, в якій працює велика кількість співробітників, інколи важко без проблем розподілити завдання між ними, вести контроль над їх виконанням. Тому необхідно автоматизувати цей процес для полегшення та пришвидшення цих процесів.

Доцільно буде надати керівнику компанії систему, яка дозволить вводити текст завдання та призначати його конкретному співробітнику. Співробітник в свою чергу зможе зразу ж переглянути всі завдання та призначені конкретно йому, а також відмітити його, як виконане.

Щодо задачі про розподіл капіталовкладень на ремонт ділянок доріг, то після введення максимальної суми коштів, яка може бути витрачена на ремонт дороги, здійснюється розв'язання цієї задачі та виведення номерів ділянок доріг, які треба відремонтувати.

4 ВИМОГИ ДО ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

4.1 Вимоги до функціональних характеристик

Програмне забезпечення складається з таких частин:

- модуль, призначений для розв'язання задачі про розподіл капіталовкладень на ремонт ділянок доріг;
- модуль з реалізацією системи, яка буде призначена для розподілу завдань та моніторингом їх поточного стану.

Робота користувача з програмою повинна здійснюватися з допомогою екранних форм. Інтерфейс програми повинен бути зрозумілим для користувача, повинні бути присутні написи, які допоможуть йому зрозуміти, як правильно користуватися програмою. Але, разом з тим, інтерфейс має бути лаконічним і простим.

Введення даних повинно здійснюватися з допомогою мишки та клавіатури.

4.2 Вимоги до надійності

При різних позаштатних ситуаціях система повинна зберігати свою працездатність:

- при помилках, які пов'язані з програмним забезпеченням, дані повинні зберігатися - це повинно відбуватися з допомогою бази даних;
- при збоях в апаратних засобах – це повинно відбуватися з допомогою операційної системи.

4.3 Вимоги до складу і параметрів технічних засобів

Для нормальної роботи програми повинні бути наявні технічні засоби:

1. Комп'ютер або ноутбук з такими характеристиками:
   - 2 ГБ і більше оперативної пам'яті;
   - тактова частота процесору – 1 ГГЦ і більше.
2. Повинні бути інстальоване таке програмне забезпечення:
   - MS WINDOWS XP або новіша версія Windows;
   - база даних Oracle 11g і вище;

5 СТАДІЇ ТА ЕТАПИ РОЗРОБКИ

Основні стадії та етапи розробки інформаційно-аналітичної система адміністрування бюджетних програм місцевої влади на прикладі дорожнього будівництва наведено в таблиці нижче.

№ п/п	Назва етапу	Термін виконання	Результат виконання
1	Узгодження теми диплому разом із дипломним керівником	01.11.2019	Тема узгоджена
2	Обговорення інтерфейсу програми разом із дипломним керівником	01.12.2019	Інтерфейс узгоджений
3	Розробка технічного завдання	15.03.2020	Готове ТЗ
4	Розробка програмного забезпечення	01.05.2020 – 06.05.2020	Розроблене програмне забезпечення
5	Розробка пояснювальної записки	07.05.2020 – 25.05.2020	Готова пояснювальна записка
6	Тестування програмного забезпечення	26.05.2020 – 01.06.2020	Здійснене налагодження програмного забезпечення
7	Здача готового програмного продукту	15.06.2020	Готове програмне забезпечення та документація

6. ПОРЯДОК КОНТРОЛЮ ТА ПРИЙМАННЯ

6.1.Види випробувань

Для контролю роботи програми буде виконане тестування всіх модулів програми.

Тестування модуля реалізацією системи, яка буде призначена для розподілу завдань та моніторингом їх поточного стану буде успішним, якщо користувач зможе успішно зайти в систему із своїм логіном та паролем, переглянути всі завдання, переглянути лише свої завдання, відсортувати та відфільтрувати їх, надіслати повідомлення іншому користувачеві, а також переглянути свої вхідні повідомлення. Якщо користувач є керівником, то також призначити завдання своєму підлеглому.

Тестування модуля, який призначений для розв’язання задачі про розподіл капіталовкладень на ремонт ділянок доріг буде успішним, якщо буде повністю виконане тестування функціональності модуля, в тому числі і на типових некоректних даних.

UNICHECK

Власник документу:
Попенко Володимир Дмитрович

Дата перевірки:
08.06.2020 18:22:36 EEST

Дата звіту:
08.06.2020 18:36:19 EEST

ID перевірки:
1003881981

Тип перевірки:
Doc vs Internet + Library

ID користувача:
77149

Назва документу: Sholudko_bachelor_is61

ID файлу: 1003896820 Кількість сторінок: 79 Кількість слів: 10778 Кількість символів: 90454 Розмір файлу: 2.78 MB

10.6% Схожість

Найбільша схожість: 3.55% з джерело бібліотеки. ID файлу: 5839560

6.47% Схожість з Інтернет джерелами	133	Page 81
9.37% Текстові збіги по Бібліотеці акаунту	670	Page 83

1.05% Цитат

Цитати	3	Page 84

Вилучення переліку посилань вимкнено

0% Вилучень

Вилучений текст відсутній

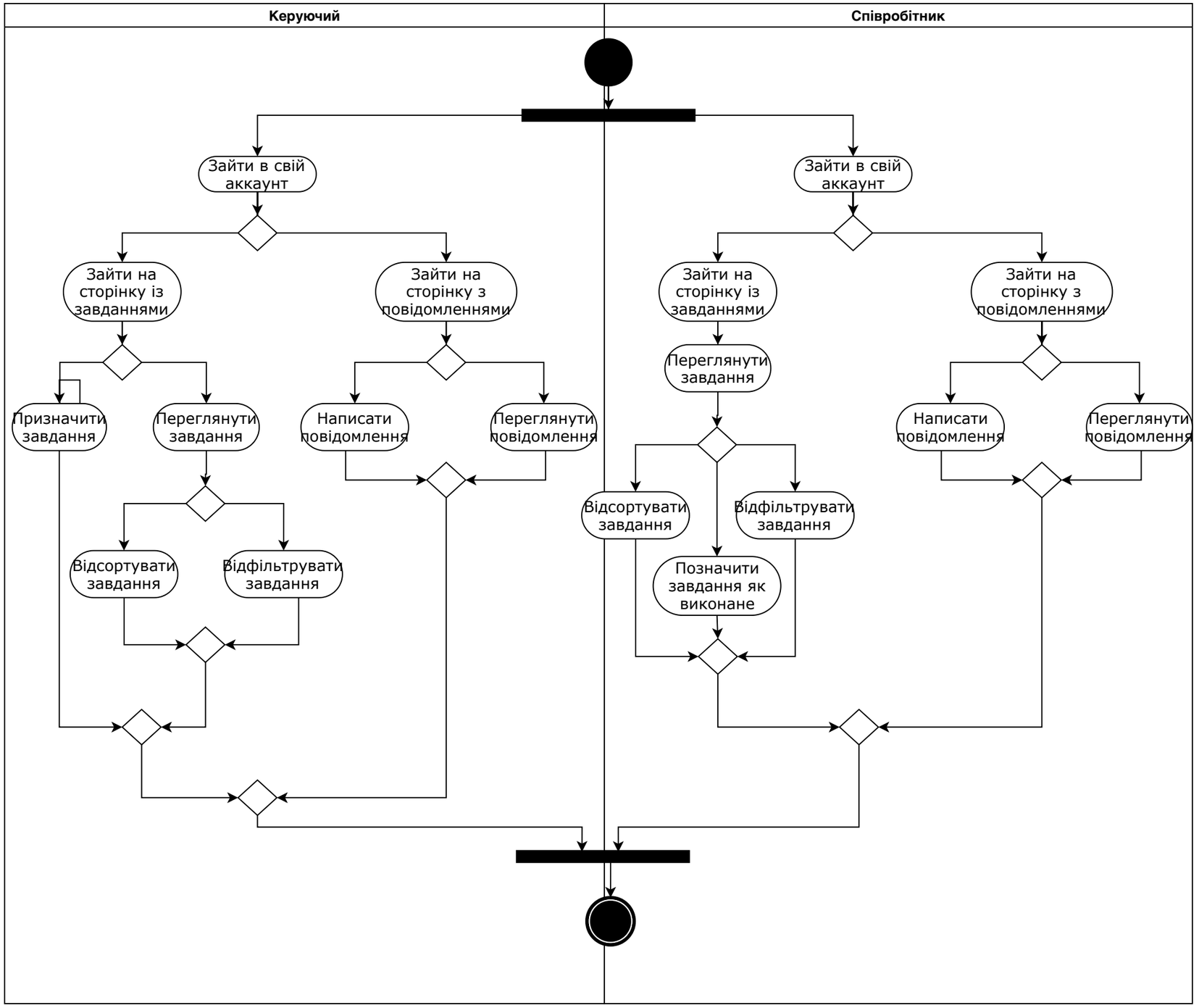
Підміна символів

Заміна символів	1

Графічний матеріал
до дипломного проєкту

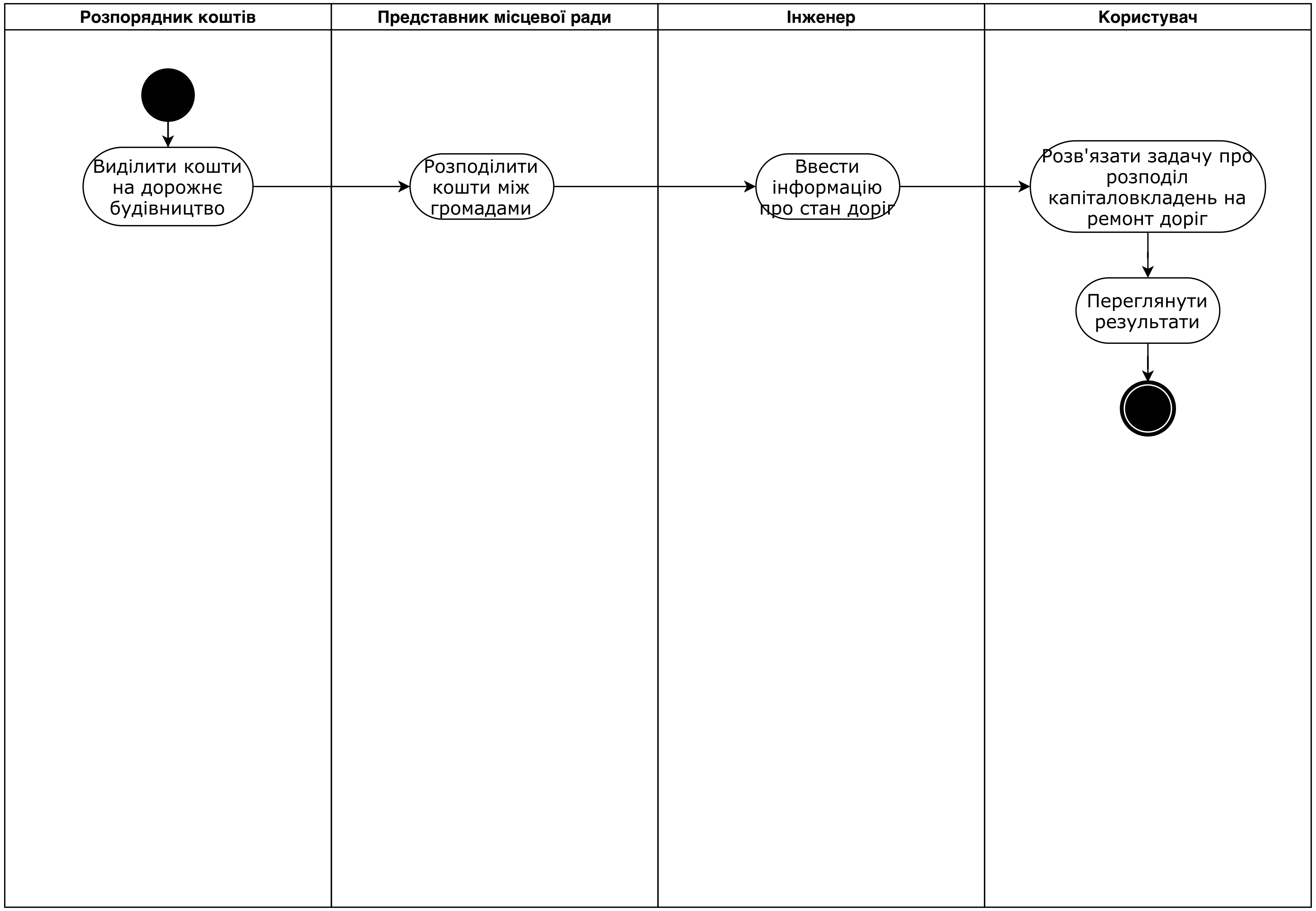
на тему: Інформаційно-аналітична система адміністрування бюджетних програм місцевої влади на прикладі дорожнього будівництва

Київ – 2020 року

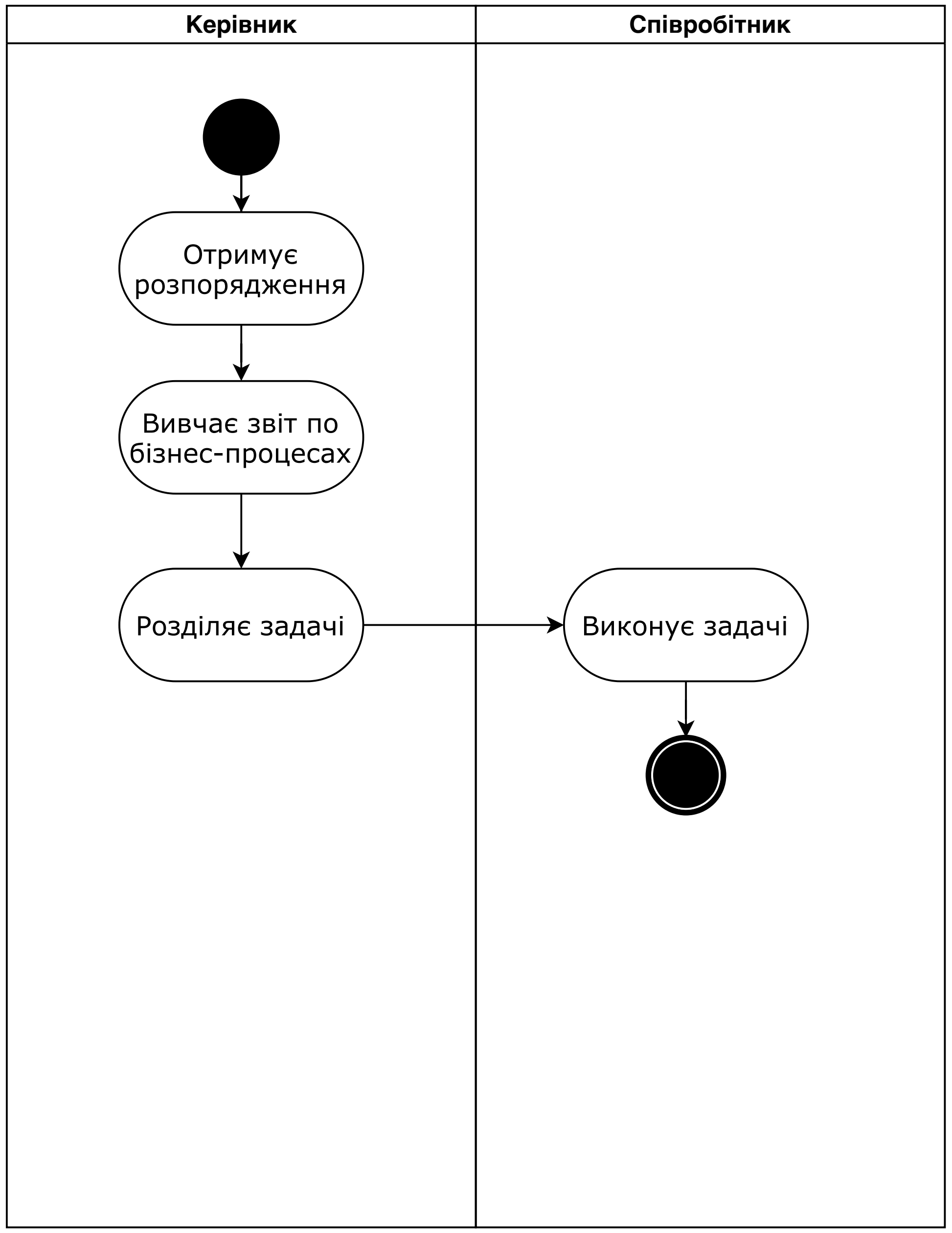


					ДП 6129.02.000 ССД	
Зм.	Арк.	№ документа	Підпис	Дата	*Схема структурна діяльності*	Літера / Маса / Масштаб
Розробив		Шолудько А. А.				
Перевірив		Попенко В. Д.				Аркуш 1 / Аркушів 3
Т. кон.						
Н. кон.		Тєлишева Т. О.			*Інформаційно-аналітична система адміністрування бюджетних програм місцевої влади на прикладі дорожнього будівництва*	*КПІ ім. Ігоря Сікорського кафедра АСОІУ гр. ІС-61*
Затвердив		Попенко В. Д.				

ДП 6129.02.000 ССД

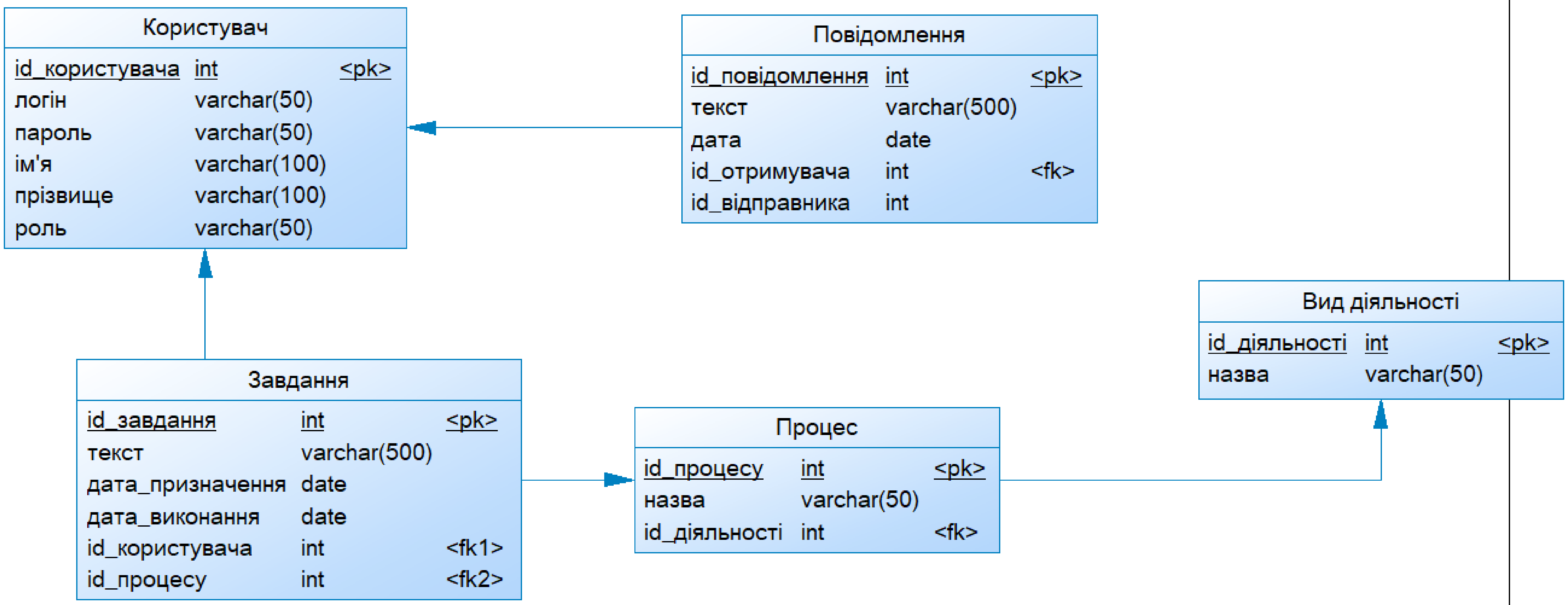


					ДП 6129.02.000 ССД		
Зм.	Арк.	№ документа	Підпис	Дата	Схема структурна діяльності	Літера	Маса	Масштаб
Розробив		Шолудько А. А.					
Перевірив		Попенко В. Д.					
Т. кон.						Аркуш 2	Аркушів 3
					Інформаційно-аналітична система адміністрування бюджетних програм місцевої влади на прикладі дорожнього будівництва	КПІ ім. Ігоря Сікорського кафедра АСОІУ гр. ІС-61	
Н. кон.		Тєлишева Т. О.					
Затвердив		Попенко В. Д.					



					ДП 6129.02.000 ССД		
Зм.	Арк.	№ документа	Підпис	Дата			
Розробив		Шолудько А. А.			Схема структурна діяльності	Літера	Маса	Масштаб
Перевірив		Попенко В. Д.					
Т. кон.						Аркуш 3	Аркушів 3
Н. кон.		Тєлишева Т. О.			Інформаційно-аналітична система адміністрування бюджетних програм місцевої влади на прикладі дорожнього будівництва	КПІ ім. Ігоря Сікорського кафедра АСОІУ гр. ІС-61	
Затвердив		Попенко В. Д.					

ДП 6129.03.000 СБД



Зм.	Арк.	№ документа	Підпис	Дата
Розробив		Шолудько А. А.		
Перевірив		Попенко В. Д.		
Т. кон.				
Н. кон.		Тєлишева Т. О.		
Затвердив		Попенко В. Д.		

ДП 6129.03.000 СБД

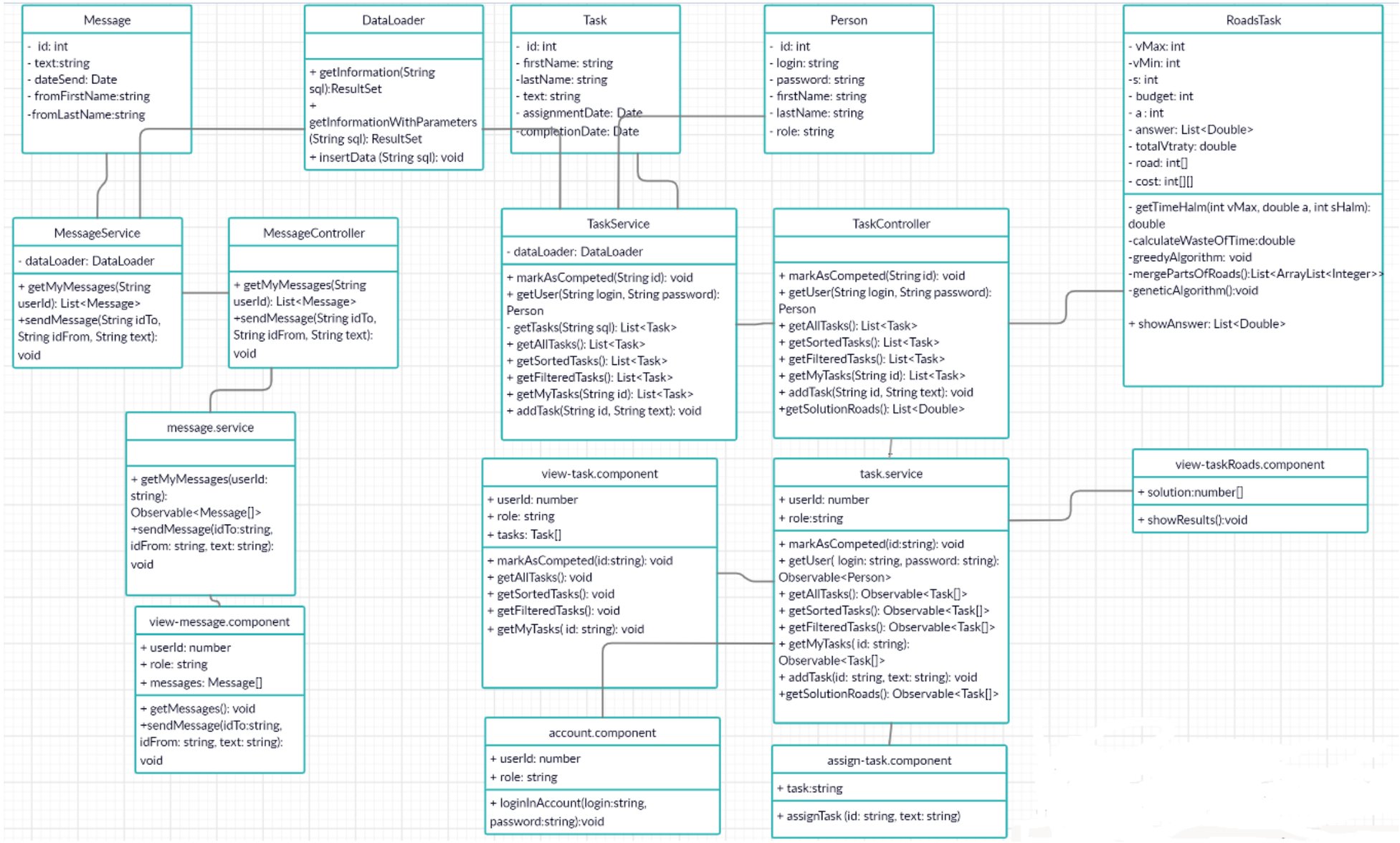
Схема бази даних

Літера	Маса	Масштаб

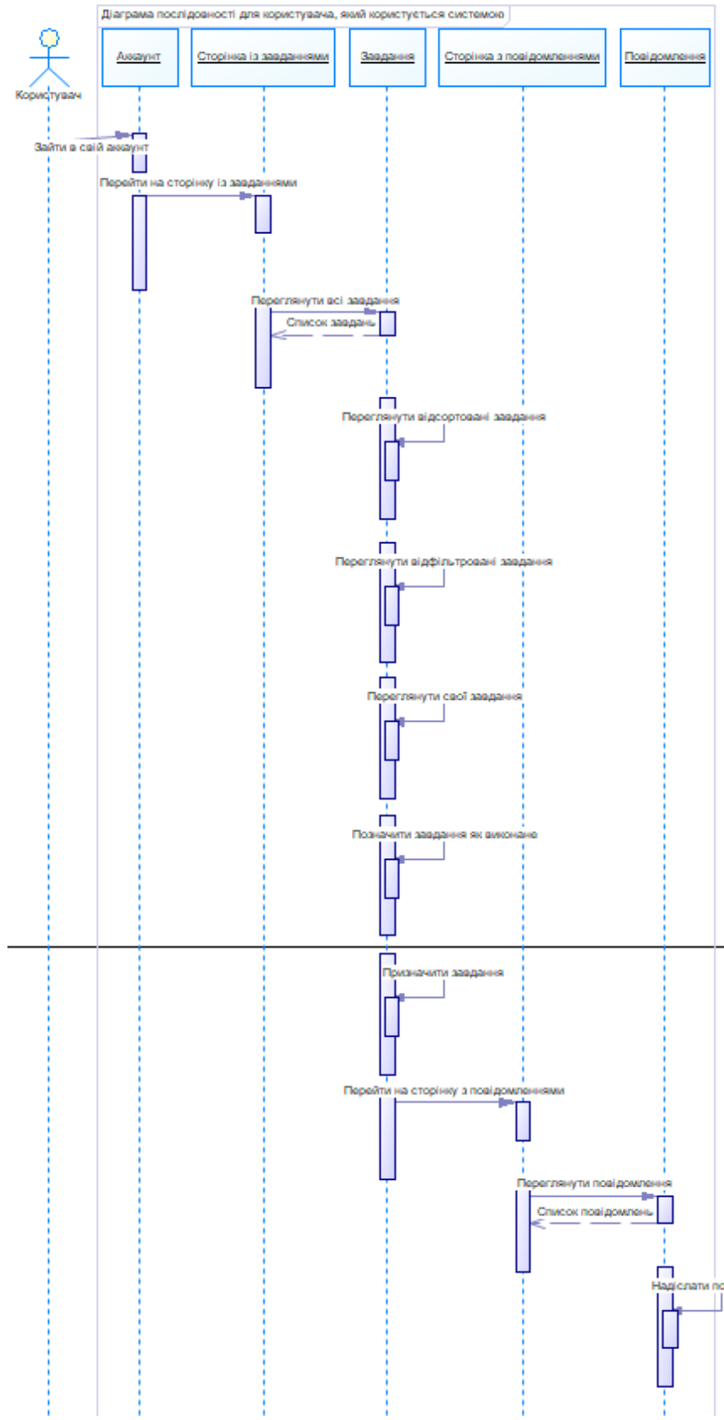
Аркуш 1 | Аркушів 1

Інформаційно-аналітична система адміністрування бюджетних програм місцевої влади на прикладі дорожнього будівництва

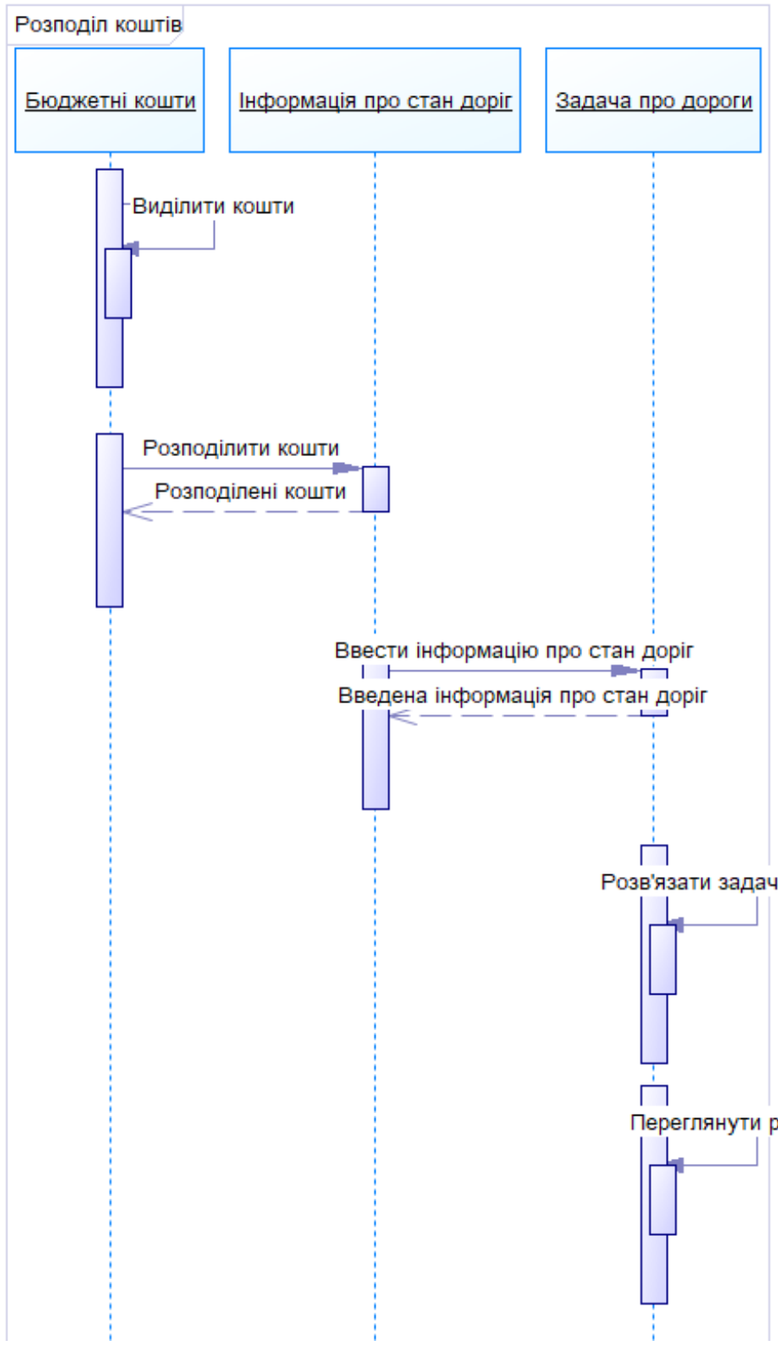
КПІ ім. Ігоря Сікорського кафедра АСОІУ гр. ІС-61



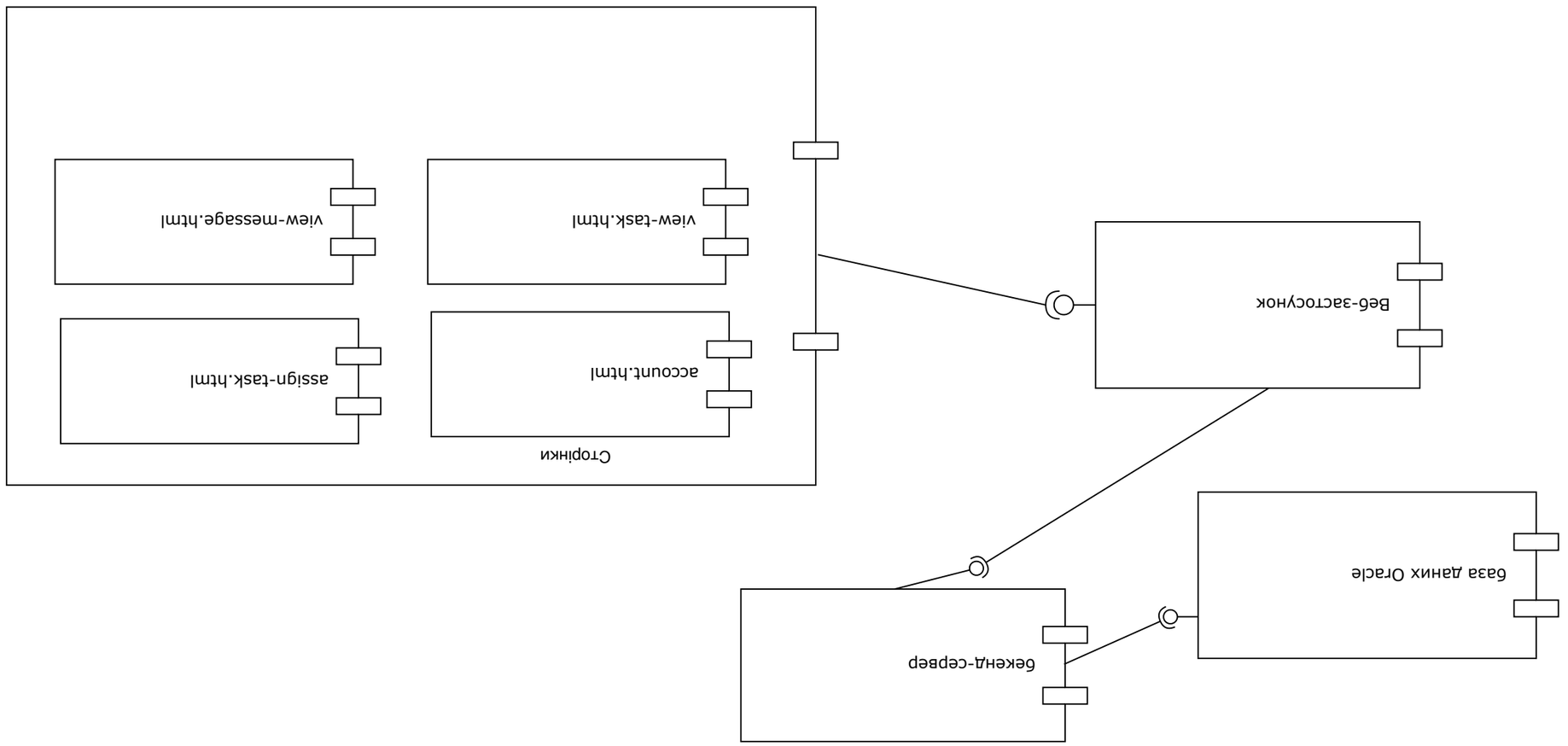
					ДП 6129.04.000 ССК	
					Схема структурна класів програмного забезпечення	Літера / Маса / Масштаб
Зм.	Арк.	№ документа	Підпис	Дата		
Розробив		Шолудько А. А.				Аркуш 1 / Аркушів 1
Перевірив		Попенко В. Д.				
Т. кон.					*Інформаційно-аналітична система адміністрування бюджетних програм місцевої влади на прикладі дорожнього будівництва*	*КПІ ім. Ігоря Сікорського кафедра АСОІУ гр. ІС-61*
Н. кон.		Тєлишева Т. О.				
Затвердив		Попенко В. Д.				



					ДП 6129.05.000 ССП	
					Схема структурна послідовності	Літера: — Маса: — Масштаб: —
Зм.	Арк.	№ документа	Підпис	Дата		Аркуш 1 / Аркушів 2
Розробив		Шолудько А. А.				
Перевірив		Попенко В. Д.				
Т. кон.						
Н. кон.		Тєлишева Т. О.			*Інформаційно-аналітична система адміністрування бюджетних програм місцевої влади на прикладі дорожнього будівництва*	*КПІ ім. Ігоря Сікорського кафедра АСОІУ гр. ІС-61*
Затвердив		Попенко В. Д.				



					ДП 6129.05.000 ССП	
					Схема структурна послідовності	Літера: — / Маса: — / Масштаб: —
Зм.	Арк.	№ документа	Підпис	Дата		Аркуш 2 / Аркушів 2
Розробив		Шолудько А. А.				
Перевірив		Попенко В. Д.				
Т. кон.						
Н. кон.		Тєлишева Т. О.			*Інформаційно-аналітична система адміністрування бюджетних програм місцевої влади на прикладі дорожнього будівництва*	*КПІ ім. Ігоря Сікорського кафедра АСОІУ гр. ІС-61*
Затвердив		Попенко В. Д.				



Зм.	Арк.	№ документа	Підпис	Дата					
Розробив		Шолудько А. А.							
Перевірив		Поленко В. Д.							
Т. контр.									
Н. контр.		Тепшева Т. О.							
Затвердив		Поленко В. Д.							

ДП 6129.06.000 ССК

Літера | Маса | Масштаб

Схема структура компоненті програмного забезпечення

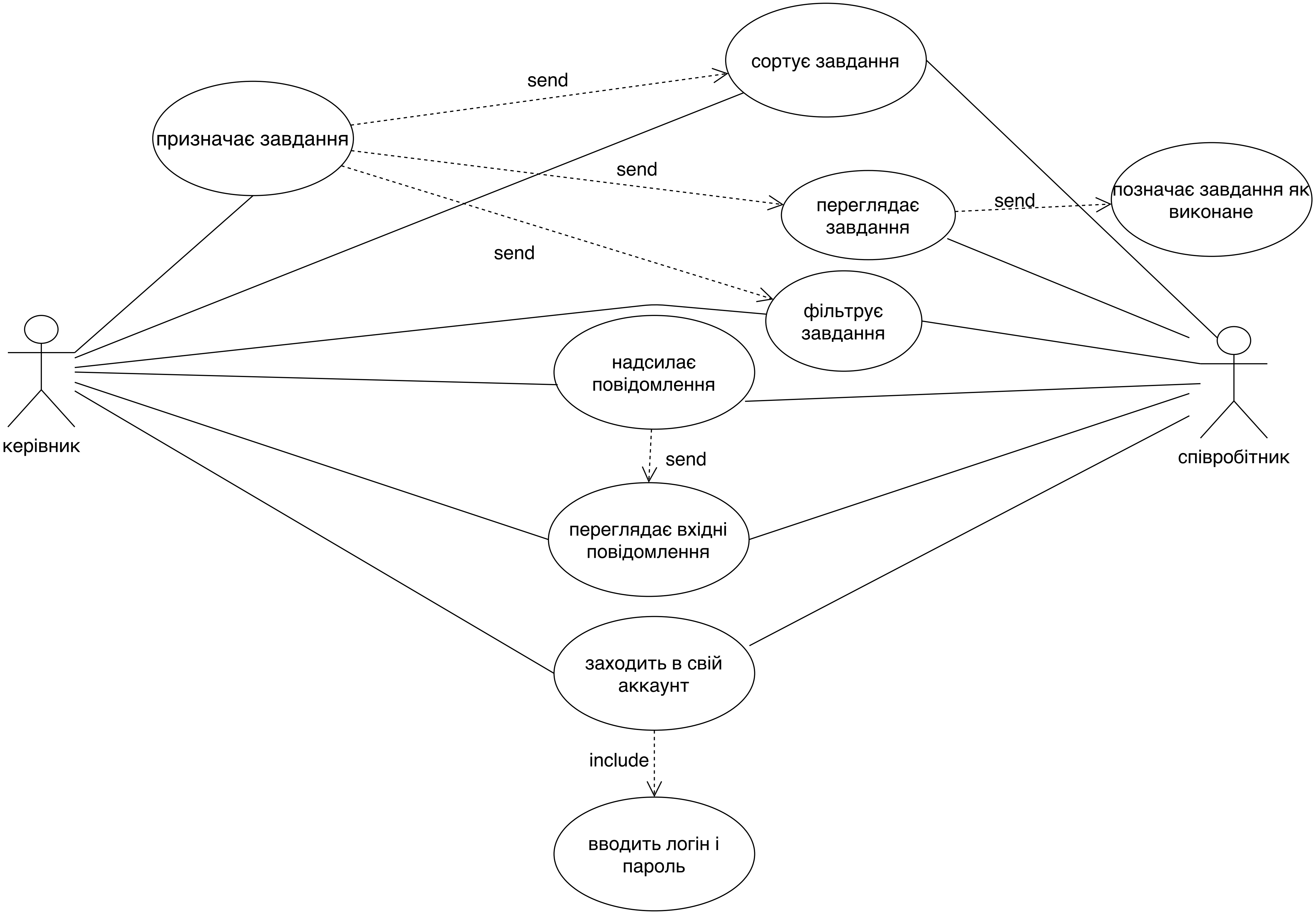
Аркуш 1 | Аркушів 1

Інформаційно-аналітична система адміністрування бюджетних програм місцевої влади на прикладі дорожнього будівництва

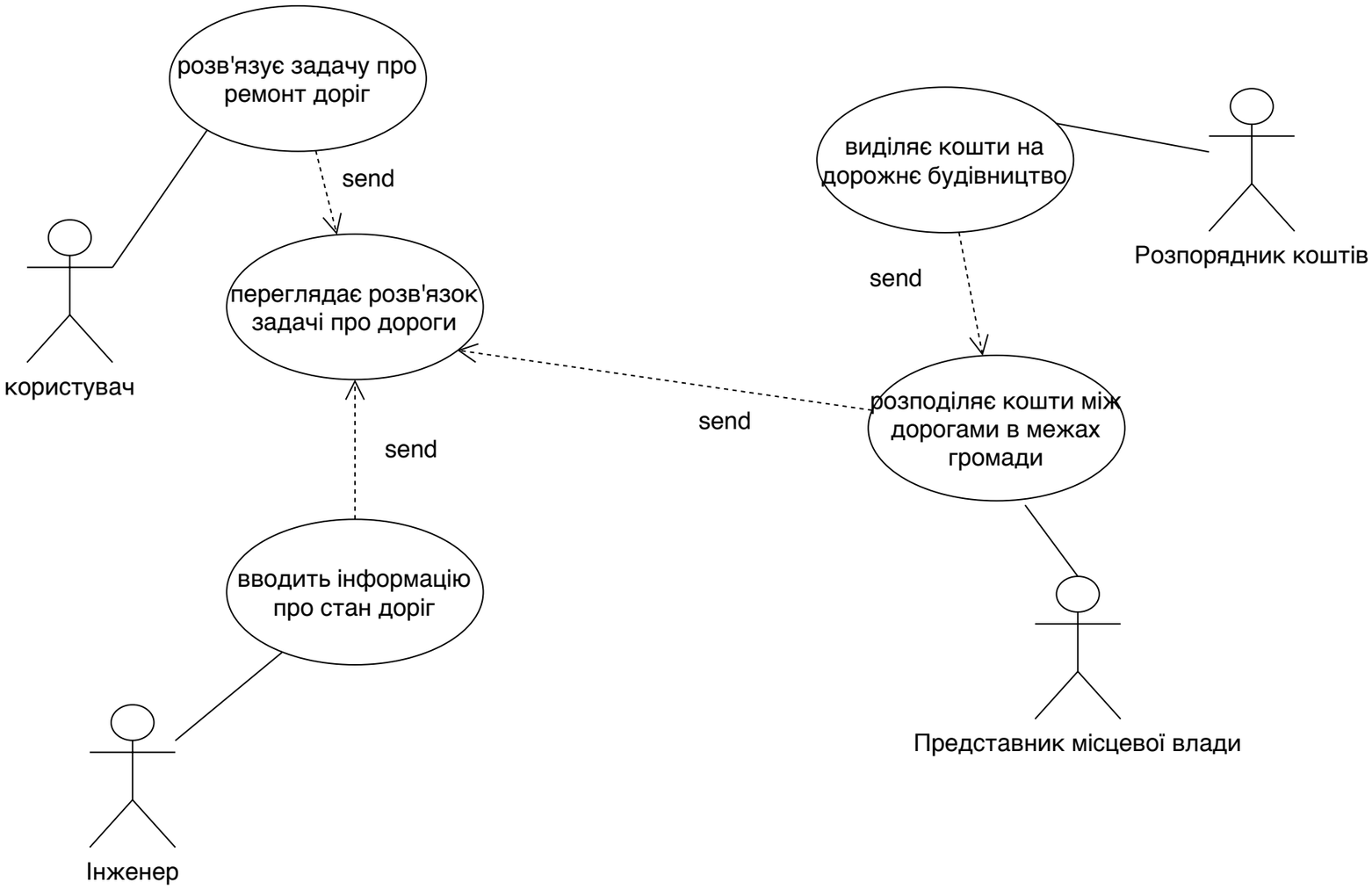
КПІ ім. Ігоря Сікорського
кафедра АСОІУ гр. ІС-61

ДП 6129.06.000 ССК

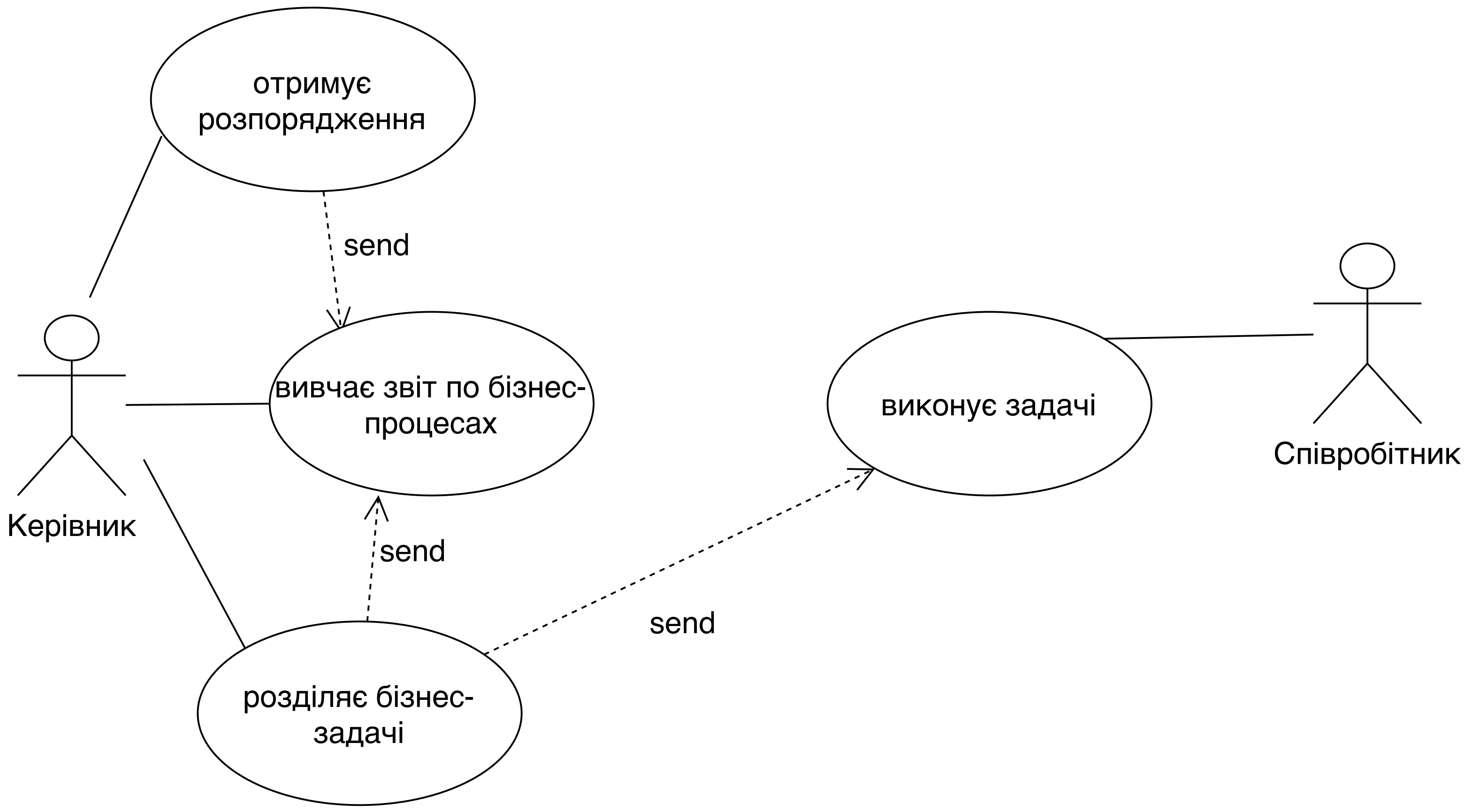
ДП 6129.07.000 ССВ



					ДП 6129.07.000 ССВ			
Зм.	Арк.	№ документа	Підпис	Дата				
Розробив		Шолудько А. А.			*Схема структурна варіантів використань*	Літера	Маса	Масштаб
Перевірив		Попенко В. Д.						
Т. кон.						Аркуш 1		Аркушів 3
Н. кон.		Тєлишева Т. О.			*Інформаційно-аналітична система адміністрування бюджетних програм місцевої влади на прикладі дорожнього будівництва*	*КПІ ім. Ігоря Сікорського кафедра АСОІУ гр. ІС-61*		
Затвердив		Попенко В. Д.						



					ДП 6129.07.000 ССВ	
					Схема структурна варіантів використань	Літера: — Маса: — Масштаб: —
Зм.	Арк.	№ документа	Підпис	Дата		Аркуш 2 / Аркушів 3
Розробив		Шолудько А. А.				
Перевірив		Попенко В. Д.				
Т. кон.						
Н. кон.		Тєлишева Т. О.			*Інформаційно-аналітична система адміністрування бюджетних програм місцевої влади на прикладі дорожнього будівництва*	*КПІ ім. Ігоря Сікорського кафедра АСОІУ гр. ІС-61*
Затвердив		Попенко В. Д.				



					ДП 6129.07.000 ССВ			
Зм.	Арк.	№ документа	Підпис	Дата	*Схема структурна варіантів використань*	Літера	Маса	Масштаб
Розробив		Шолудько А. А.						
Перевірив		Попенко В. Д.						
Т. кон.						Аркуш 3		Аркушів 3
					Інформаційно-аналітична система адміністрування бюджетних програм місцевої влади на прикладі дорожнього будівництва	*КПІ ім. Ігоря Сікорського кафедра АСОІУ гр. ІС-61*		
Н. кон.		Тєлишева Т. О.						
Затвердив		Попенко В. Д.						