Хмельницький національний університет
Факультет інформаційних технологій
Кафедра комп'ютерної інженерії та інформаційних систем

КВАЛІФІКАЦІЙНА РОБОТА

бакалавр
Освітній рівень

Інформаційна система організації повсякденної діяльності користувача
Назва теми

КвРІСТ 200193.20.01.05 ПЗ
Шифр

Галузь знань 12 «Інформаційні технології»
Шифр, назва

Спеціальність 126 «Інформаційні системи та технології»
Шифр, назва

Освітня програма «Інформаційні системи та технології»
Назва

Виконав: студент III курсу, група ІСТс-20-1 _____ Д. О. Коляда
Підпис Ініціали, прізвище

Керівник _____ Т. О. Говорущенко
Підпис, дата Ініціали, прізвище

Нормоконтролер _____ С.М. Лисенко
Підпис, дата Ініціали, прізвище

До захисту допускаю:
Зав. кафедри комп'ютерної інженерії та інформаційних систем _____ Т.О. Говорущенко
Підпис Ініціали, прізвище

«29» травня 2023 р.

Хмельницький 2023

ХМЕЛЬНИЦЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ

Факультет Інформаційних технологій

Кафедра Комп'ютерної інженерії та інформаційних систем

Освітній рівень Бакалавр

Галузь знань 12 Інформаційні технології

Спеціальність 126 Інформаційні системи та технології

Освітня програма Освітньо-професійна програма «Інформаційні системи та технології»

ЗАТВЕРДЖУЮ

Зав. кафедри Т.О.Говорущенко

" 10 " 01 2023 р.

ЗАВДАННЯ
НА КВАЛІФІКАЦІЙНУ РОБОТУ БАКАЛАВРА

Коляді Денису Олександровичу
Прізвище, ім'я, по батькові студента

1. Тема проекту (роботи) Інформаційна система організації повсякденної діяльності користувача

Керівник проекту (роботи) Говорущенко Т.О., д.т.н., проф.
Прізвище, ім'я, по батькові, науковий ступінь, вчене звання

Затверджена наказом ректора університету від 01.03.2022 р. № 5

2. Строк подання студентом проекту (роботи) на кафедру 01.06.2023 р.

3. Вихідні дані до проекту (роботи) Завдання на дипломне проектування

4. Зміст пояснювальної записки (перелік питань, які потрібно розробити)

Дослідження предметної області та постановка задачі

Архітектура інформаційної системи організації повсякденної діяльності користувача

Реалізація та функціонування інформаційної системи організації повсякденної діяльності користувача

5. Перелік графічного матеріалу (із зазначенням обов'язкових креслень)

Планування проектних робіт

Проектування архітектури інформаційної системи

Функціонування інформаційної системи

6. Консультанти розділів дипломного проекту (роботи)

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв
Нормоконтроль	Лисенко С.М., професор кафедри КІІС		
Антиплагіат	Нічепорук А.О., доцент кафедри КІІС		

7. Дата видачі завдання «10» 01 2023 р.

КАЛЕНДАРНИЙ ПЛАН

№з/п	Назва етапів (розділів) дипломного проекту (роботи)	Термін виконання етапів проекту (роботи)	Примітка
1	Вибір напряму дослідження та узгодження тематики кваліфікаційної роботи з керівником	10.01.2023	виконано
2	Ознайомлення з предметною областю; формулювання мети та задач дослідження; визначення об'єкта та предмета дослідження	01.02.2023	виконано
3	Робота над розділом 1 – дослідження предметної області та постановка задачі	01.03.2023	виконано
4	Робота над розділом 2 – архітектура інформаційної системи організації повсякденної діяльності користувача	01.04.2023	виконано
5	Робота над розділом 3 – реалізація та функціонування інформаційної системи організації повсякденної діяльності користувача	29.04.2023	виконано
6	Оформлення пояснювальної записки згідно вимог	25.05.2023	виконано
7	Попередній захист ВКР	26.05.2023	виконано
8	Захист ВКР на засіданні ЕК	Червень 2023 року	

Студент _____ Д. О. Коляда
Підпис Ініціали, прізвище

Керівник роботи _____ Т. О. Говорушенко
Підпис Ініціали, прізвище

№ рядка	формат	Позначення	Найменування	Кіл. листів	№ екз	Примітка
			Текстові документи			
1		КвРІСТ 200193.20.01.05 ПЗ	Пояснювальна записка	66		
			Графічні матеріали			
2		КвРІСТ 200193.20.01.05	Планування проектних робіт	1		
3		КвРІСТ 200193.20.01.05	Проектування архітектури інформаційної системи	1		
4		КвРІСТ 200193.20.01.05	Функціонування інформаційної системи	1		

Зм	Арк	№ докум	Підпис	Дата	КвРІСТ 200193.20.01.05 ВП		
Розробив		Коляда		29.05	Відомість проекту	Літера	Аркуш	Аркушів
Перевір.		Говорущенко		28.05		У	1	1
Н. контр.		Лисенко		23.05		ХНУ, ІСТс-20-1		
Затв.		Говорущенко		28.05				

АНОТАЦІЯ

Тема кваліфікаційної роботи: «Інформаційна система організації повсякденної діяльності користувача».

Автор роботи: Коляда Денис Олександрович.

Керівник роботи: Говорущенко Тетяна Олександрівна.

Пояснювальна записка: 66 с., 41 рис., 2 табл., 3 дод., 46 джерел.

Графічна частина: 3 креслення.

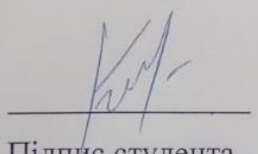
ІНФОРМАЦІЙНА СИСТЕМА, ЖИТТЄВИЙ ЦИКЛ, АРХІТЕКТУРА ІНФОРМАЦІЙНОЇ СИСТЕМИ, ФУНКЦІОНУВАННЯ ІНФОРМАЦІЙНОЇ СИСТЕМИ, ТЕСТУВАННЯ ІНФОРМАЦІЙНОЇ СИСТЕМИ.

Метою кваліфікаційної роботи є полегшення організації повсякденної діяльності користувача шляхом проєктування та розроблення інформаційної системи організації повсякденної діяльності користувача (органайзера).

Об'єктом дослідження є процес організації повсякденної діяльності користувача.

Предметом дослідження є інформаційна система організації повсякденної діяльності користувача (органайзер).

Практичне значення має спроєктована та реалізована інформаційна система організації повсякденної діяльності користувача (органайзер), яка відповідає за організацію повсякденної діяльності незалежно від її специфіки.



Підпис студента

30.05.2023

Дата

ЗМІСТ

ВСТУП 3
1 ДОСЛІДЖЕННЯ ПРЕДМЕТНОЇ ОБЛАСТІ ТА ПОСТАНОВКА ЗАДАЧІ .. 5
1.1 Аналіз задачі організації повсякденної діяльності користувача 5
1.2 Вибір моделі життєвого циклу, обгрунтування вибору апаратних та програмних ресурсів для реалізації поставленої задачі 17
1.3 Висновки. Постановка задачі 20
2 АРХІТЕКТУРА ІНФОРМАЦІЙНОЇ СИСТЕМИ ОРГАНІЗАЦІЇ ПОВСЯКДЕННОЇ ДІЯЛЬНОСТІ КОРИСТУВАЧА 21
2.1 Графік виконання проєктних робіт, аналіз ризиків 21
2.2 Формування вимог до інформаційної системи 25
2.3 Автоматна модель та UML-діаграми інформаційної системи 31
2.4 Висновки 46
3 РЕАЛІЗАЦІЯ ТА ФУНКЦІОНУВАННЯ ІНФОРМАЦІЙНОЇ СИСТЕМИ ОРГАНІЗАЦІЇ ПОВСЯКДЕННОЇ ДІЯЛЬНОСТІ КОРИСТУВАЧА 47
3.1 Реалізація інформаційної системи організації повсякденної діяльності користувача 47
3.2 Функціонування та тестування інформаційної системи організації повсякденної діяльності користувача 54
3.3 Висновки 59
ВИСНОВКИ 60
ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАНЬ 62
ДОДАТОК А Копія креслення «Планування проектних робіт» 67
ДОДАТОК Б Копія креслення «Проектування архітектури інформаційної системи» 68
ДОДАТОК В Копія креслення «Функціонування інформаційної системи» .. 69
ДОДАТОК Г Лістинг коду 70

ВСТУП

Попереднє планування справ допомагає підвищити ефективність будь-якої діяльності – як особистої, так і професійної.

Інформаційні системи організації повсякденної діяльності користувача (органайзери) дозволяють проводити планування особистого часу користувача, своєчасно нагадують про початок запланованих заходів та зустрічей, про необхідність виконання тих чи інших нагальних справ, ведуть персональні та інші картки з можливістю автоматичної вибірки інформації, ведуть персональні інформаційні записники для збереження різноманітної особистої інформації. Такі системи призначені для накопичення інформації користувача, а потім для організації справ і контролю за їх виконанням, відслідковування визначених користувачем подій [1-15].

Метою кваліфікаційної роботи є полегшення організації повсякденної діяльності користувача шляхом проєктування та розроблення інформаційної системи організації повсякденної діяльності користувача (органайзера).

Поставлена мета досягається розв’язанням такої основної задачі: розроблення інформаційної системи організації повсякденної діяльності користувача (органайзера), яка відповідатиме за організацію повсякденної діяльності незалежно від її специфіки. Проєктування та розроблення такого органайзера повинні базуватись на принципі розділення його функціональності на діяльності (проєкти) та задачі в цих діяльностях, стан яких користувач буде визначати і переглядати. Інтерфейс має бути простим в розумінні і не обтяжуватись специфічними можливостями, в той же час програма повинна бути різнобічна і мати клієнт-серверну архітектуру, щоб користуватись нею могли клієнти різного типу.

Об’єктом дослідження є процес організації повсякденної діяльності користувача.

Предметом дослідження є інформаційна система організації повсякденної діяльності користувача (органайзер).

Для досягнення поставленої мети використовуються такі методи дослідження, як методи синтезу, аналізу та моделювання процесів, принципи системного аналізу, теоретико-множинні підходи.

Практичне значення має спроєктована та реалізована інформаційна система організації повсякденної діяльності користувача (органайзер), яка відповідає за організацію повсякденної діяльності незалежно від її специфіки.

ДОСЛІДЖЕННЯ ПРЕДМЕТНОЇ ОБЛАСТІ ТА ПОСТАНОВКА ЗАДАЧІ

1.1 Аналіз задачі організації повсякденної діяльності користувача

Організація повсякденної діяльності користувача може бути полегшена за допомогою різноманітних інструментів та підходів. Ось деякі способи, які можуть допомогти [16-18]:

1) календар та планувальник для відстеження та організації зустрічей, завдань і важливих подій; використання нагадувань, щоб не пропустити важливі дати і події;

2) списки завдань для організації щоденного розкладу роботи; розподіл завдань на категорії, пріоритети та встановлення термінів виконання;

3) нотатки і блокноти для запису ідеї, списків покупок, нагадувань і важливої інформації з метою організації своїх думок і доступу до них зручним способом;

4) повідомлення і комунікація для зв'язку з іншими людьми; планування часу для перегляду та відповідей на повідомлення, щоб підтримувати ефективну комунікацію;

5) автоматизація та розумний дім, які можуть спростити повсякденну діяльність – наприклад, автоматичне вимкнення світла, контроль температури, підказки щодо розкладу роботи та багато іншого;

6) здоровий спосіб життя – планування часу для здорового харчування;

7) фітнес-трекери для вимірювання фізичної активності, кроків, сну та інших показників здоров'я, які допоможуть вести активний спосіб життя та контролювати свій фізичний стан;

8) дієта та харчування – використання додатків або онлайн-інструментів для ведення дієтного журналу, планування їжі та контролю харчування, які допоможуть вести здоровий спосіб харчування та досягати своїх харчувальних цілей;

9) медитація та релаксація – використання мобільних додатків або онлайн-ресурсів для медитації, релаксації та стрес-менеджменту, які можуть допомогти зосередитися, знизити рівень стресу і поліпшити загальний стан здоров'я;

10) розподіл часу з врахуванням пріоритетів та важливих завдань, наприклад, з використанням техніки управління часом "Помідора" (Pomodoro), щоб ефективно розподілити час між роботою, відпочинком і перервами.

Важливо знайти інструменти та підходи, які найкраще відповідають індивідуальним потребам та способу життя.

Існує багато різних інструментів, які можуть допомогти організувати повсякденну діяльність користувача. Ось деякі з них [19-22]:

1) календарі та планувальники – популярні інструменти цього типу включають Google Календар, Microsoft Outlook Календар, Todoist, Trello та Asana;

2) списки завдань – деякі популярні додатки для створення списків завдань включають Todoist, Any.do, Wunderlist та Microsoft To-Do;

3) додатки для нотаток – деякі популярні додатки для нотаток включають Evernote, OneNote, Google Keep та Apple Notes.

Розглянемо переваги та недоліки таких відомих інструментів.

Google Календар (рис. 1.1) є популярним інструментом для організації повсякденної діяльності.

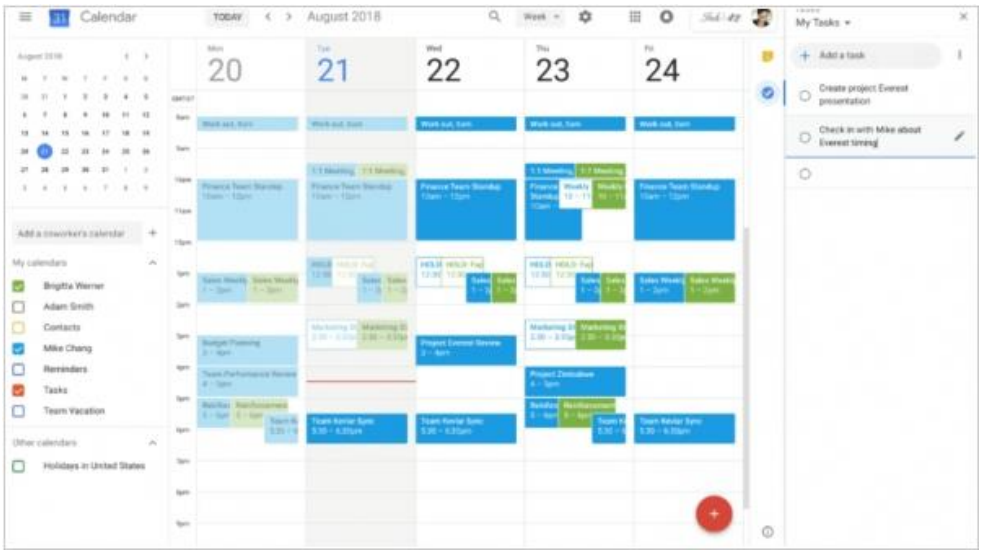


Рисунок 1.1 – Google Календар

Переваги використання Google Календаря [23-27]:

1) легке використання: Google Календар має простий і інтуїтивно зрозумілий інтерфейс, що дозволяє швидко створювати події, додавати нагадування та встановлювати повторювані події;

2) синхронізація на всіх пристроях: Календар автоматично синхронізується з іншими пристроями, такими як смартфони, планшети та комп'ютери, що дозволяє користувачу мати постійний доступ до свого розкладу незалежно від того, де він знаходиться;

3) можливість ділитися розкладом: користувач може ділитися своїм календарем з іншими користувачами, надавати їм доступ до перегляду або редагування розкладу; це корисно для спільного планування подій або координації зустрічей з колегами, друзями або сім'єю;

4) нагадування та сповіщення: користувач може встановлювати нагадування для важливих подій, завдань або зустрічей; Google Календар надсилатиме сповіщення на електронну пошту, мобільний пристрій або веб-повідомлення, щоб користувач не забув про них.

Недоліки використання Google Календаря [23-27]:

1) обмежені можливості налаштування: у порівнянні з іншими календарними додатками, Google Календар може мати обмежені можливості налаштування і персоналізації; деякі користувачі можуть прагнути більшої гнучкості в налаштуванні вигляду календаря або способу відображення подій;

2) необхідність підключення до Інтернету: для використання Google Календаря потрібне постійне підключення до Інтернету; це означає, що без доступу до Інтернету користувач не зможе отримувати оновлення, робити зміни у розкладі або отримувати сповіщення;

3) приватність і безпека: використання Google Календаря пов'язане з обробкою та збереженням особистої інформації в хмарних сервісах Google; для деяких користувачів це може бути проблемою з приватністю та безпекою даних;

4) залежність від екосистеми Google: якщо користувач не користується іншими сервісами Google, такими як Gmail або Google Drive, використання Google Календаря може бути менш зручним; деякі функції можуть бути більш інтегрованими з іншими продуктами Google, а не зі сторонніми додатками або сервісами.

Важливо враховувати ці переваги та недоліки при виборі календарного інструмента, оскільки вони можуть вплинути на досвід використання та відповідність індивідуальним потребам.

Microsoft Outlook Календар (рис. 1.2) є популярним інструментом для організації повсякденної діяльності, який включає функції календаря та планувальника.

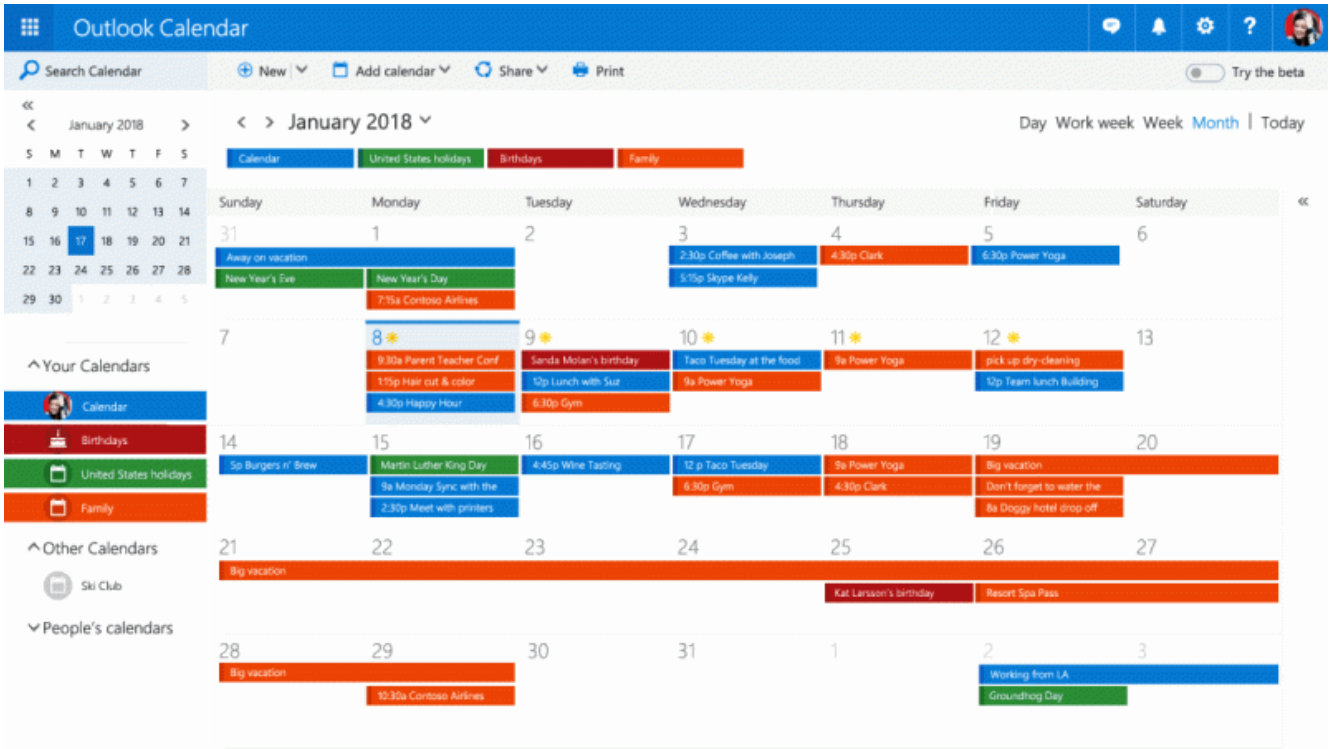


Рисунок 1.2 – Microsoft Outlook Календар

Переваги використання Microsoft Outlook Календар [28-32]:

1) інтеграція зі сховищами даних: Microsoft Outlook легко інтегрується зі сховищами даних, такими як Microsoft Exchange, Office 365, OneDrive та SharePoint;

це дозволяє зручно зберігати та синхронізувати календар та іншу інформацію з іншими сервісами Microsoft;

2) розширені можливості налаштування: Outlook надає велику кількість можливостей налаштування, що дозволяє вибрати вигляд календаря, налаштувати нагадування, налаштувати повторювані події та використовувати фільтри для перегляду подій за різними параметрами;

3) електронна пошта та комунікація: Outlook поєднує в собі функції електронної пошти, календаря та контактів, що дозволяє зручно спілкуватися з іншими користувачами, планувати зустрічі та обмінюватися інформацією;

4) інтеграція зі сторонніми додатками: Outlook підтримує інтеграцію з різними сторонніми додатками, такими як Evernote, Trello, PayPal та інші; це дозволяє розширити функціональність Outlook та підключити його до інших інструментів, які користувач вже використовує.

Недоліки використання Microsoft Outlook Календар [28-32]:

1) складність використання для новачків: Microsoft Outlook може бути трохи складним для новачків або тих, хто не має досвіду використання продуктів Microsoft; інтерфейс може здаватись перевантаженим і вимагати часу для його опанування;

2) залежність від екосистеми Microsoft: якщо користувач не використовує інші продукти Microsoft, такі як Exchange або Office 365, деякі функції Outlook можуть бути менш інтегрованими або обмеженими; для повноцінного використання всіх можливостей Outlook може знадобитися підписка на платний план або використання інших продуктів Microsoft;

3) обмежена мобільність: хоча Outlook має мобільні додатки для смартфонів і планшетів, деякі функції можуть бути менш доступними або обмеженими порівняно з повноцінною версією на комп'ютері; це може вплинути на зручність використання під час пересування або роботи з великим обсягом даних;

4) приватність та безпека даних: використання Outlook пов'язане з обробкою та збереженням особистих даних у хмарних сервісах Microsoft; для деяких

користувачів це може бути проблемою з приватністю та безпекою даних, особливо якщо вони стурбовані про збереження своєї інформації на серверах під контролем третіх сторін.

Враховуючи ці переваги та недоліки, можна визначити, чи підходить Microsoft Outlook для повсякденної діяльності та вимог користувача. Важливо враховувати потреби, звички та вподобання користувача щодо інструментів організації робочого часу перед прийняттям рішення.

Todoist (рис. 1.3) є популярним інструментом для управління завданнями та організації повсякденної діяльності.

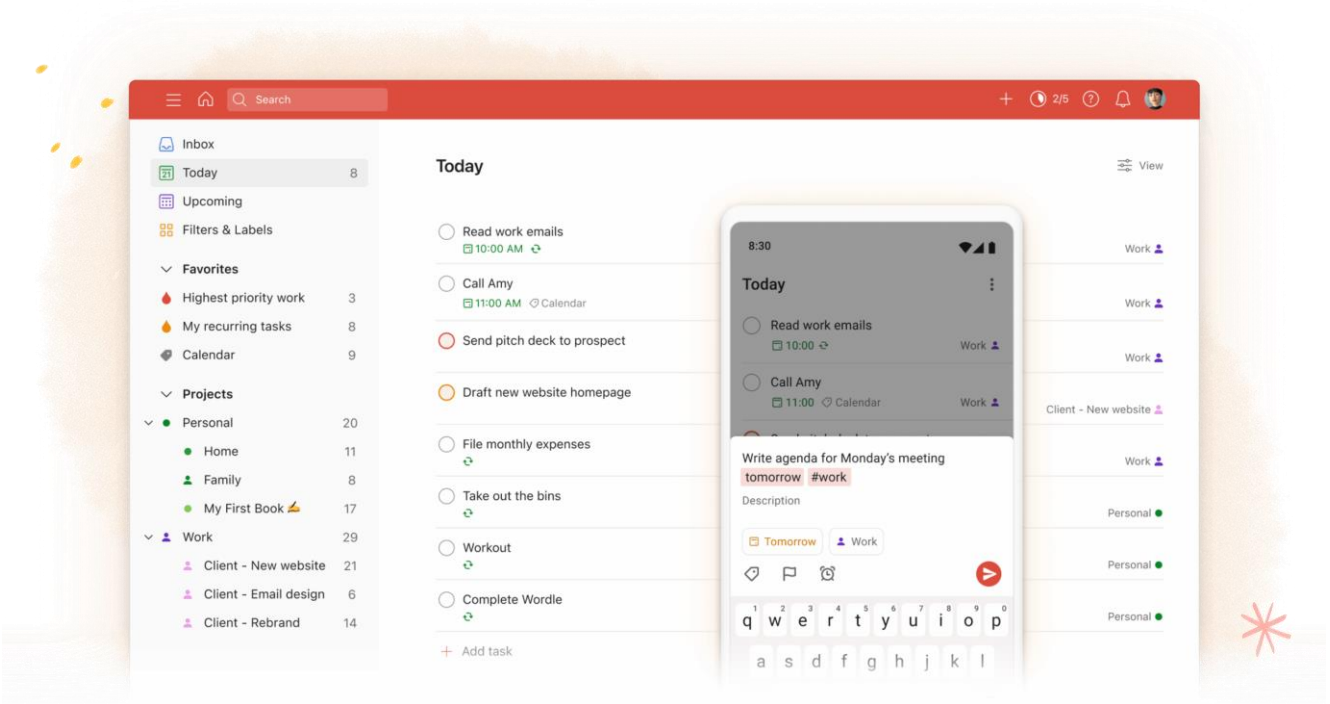


Рисунок 1.3 – Todoist

Переваги використання Todoist [33-37]:

1) простий та інтуїтивно зрозумілий інтерфейс: Todoist має простий та зручний інтерфейс, який дозволяє легко створювати, редагувати та організовувати завдання; він дозволяє швидко додавати нові завдання та надавати їм пріоритети, терміни виконання та мітки для категоризації;

2) кросплатформність та синхронізація: Todoist доступний на різних платформах, включаючи веб-версію, мобільні додатки для iOS та Android, а також настільні додатки для Windows та macOS; всі дані синхронізуються в режимі реального часу, що дозволяє користувачу отримувати доступ до своїх завдань з будь-якого пристрою;

3) розширені функції та інтеграції: Todoist пропонує ряд додаткових функцій, таких як множинні проекти, спільна робота над завданнями, нагадування про дедлайни та багато іншого; він також інтегрується з іншими популярними сервісами, такими як Gmail, Google Календар, Slack, Dropbox, і дозволяє користувачу зручно поєднувати свої завдання з іншими інструментами;

4) нагадування, планування та пріоритети: Todoist дозволяє встановлювати терміни виконання завдань, встановлювати пріоритети та надавати їм різні мітки; це допомагає користувачу зорієнтуватись у своїх завданнях та ефективно планувати свій час;

5) організація за допомогою проектів та міток: користувач може створювати проекти та призначати їм мітки для категоризації та організації завдань.

Недоліки використання Todoist [33-37]:

1) обмежена функціональність у безкоштовній версії: безкоштовна версія Todoist має обмежені можливості порівняно з платними планами; деякі функції, такі як нагадування про дедлайни або спільна робота над завданнями в команді, доступні лише в платних версіях;

2) обмеження на кількість проектів та завдань: безкоштовна версія Todoist має обмеження на кількість проектів та завдань, які можна створити; якщо користувач планує використовувати Todoist для складного управління завданнями або великою кількістю проектів, це обмеження може бути недостатнім;

3) відсутність деяких просунутих функцій: у порівнянні з деякими конкуруючими інструментами для управління завданнями, Todoist може бути менш функціональним у деяких аспектах; наприклад, він може не мати деяких

додаткових функцій, таких як гантограми або інтеграція з іншими складними системами управління проектами;

4) залежність від Інтернет-з'єднання: для використання Todoist необхідне постійне Інтернет-з'єднання; це означає, що користувач не зможе мати доступ до своїх завдань або робити зміни, якщо у користувача немає підключення до Інтернету;

5) інтерфейс може бути перевантаженим для деяких користувачів: для деяких користувачів інтерфейс Todoist може здаватися перевантаженим або складним для використання; він має багато функцій та налаштувань, які можуть змішуватися або бути заплутаними для новачків;

6) відсутність графічного інтерфейсу для гантограм: Todoist не надає вбудованого графічного інтерфейсу для гантограм, що може бути не зручним для деяких користувачів, особливо тих, хто працює з складними проектами.

Враховуючи ці переваги та недоліки, користувач може визначити, чи відповідає Todoist його потребам.

Trello (рис. 1.4) є популярним інструментом для керування проектами та завданнями.

Переваги використання Trello [38-42]:

1) візуальна організація завдань: Trello використовує систему дошок, списків та карток для візуального представлення завдань та їх стану; це дозволяє легко організовувати та відстежувати прогрес у реальному часі;

2) простота використання: ынтерфейс Trello є простим та інтуїтивно зрозумілим; користувач може легко створювати нові дошки, списки та картки, переміщувати їх, додавати коментарі та мітки;

3) колаборація та спільна робота: Trello дозволяє додавати учасників до дошок та карток, що сприяє спільній роботі над завданнями; користувач може призначати завдання, обговорювати їх та відстежувати прогрес кожного учасника;

4) розширені можливості: Trello має ряд додаткових функцій, таких як додаткові поля, дедлайни, сповіщення, чек-листи та інші, які допомагають більш детально налаштовувати та управляти завданнями;

5) інтеграція з іншими інструментами: Trello може інтегруватися з іншими популярними інструментами, такими як Slack, Google Drive, Dropbox та інші; це дозволяє зручно поєднувати Trello з іншими інструментами та джерелами даних.

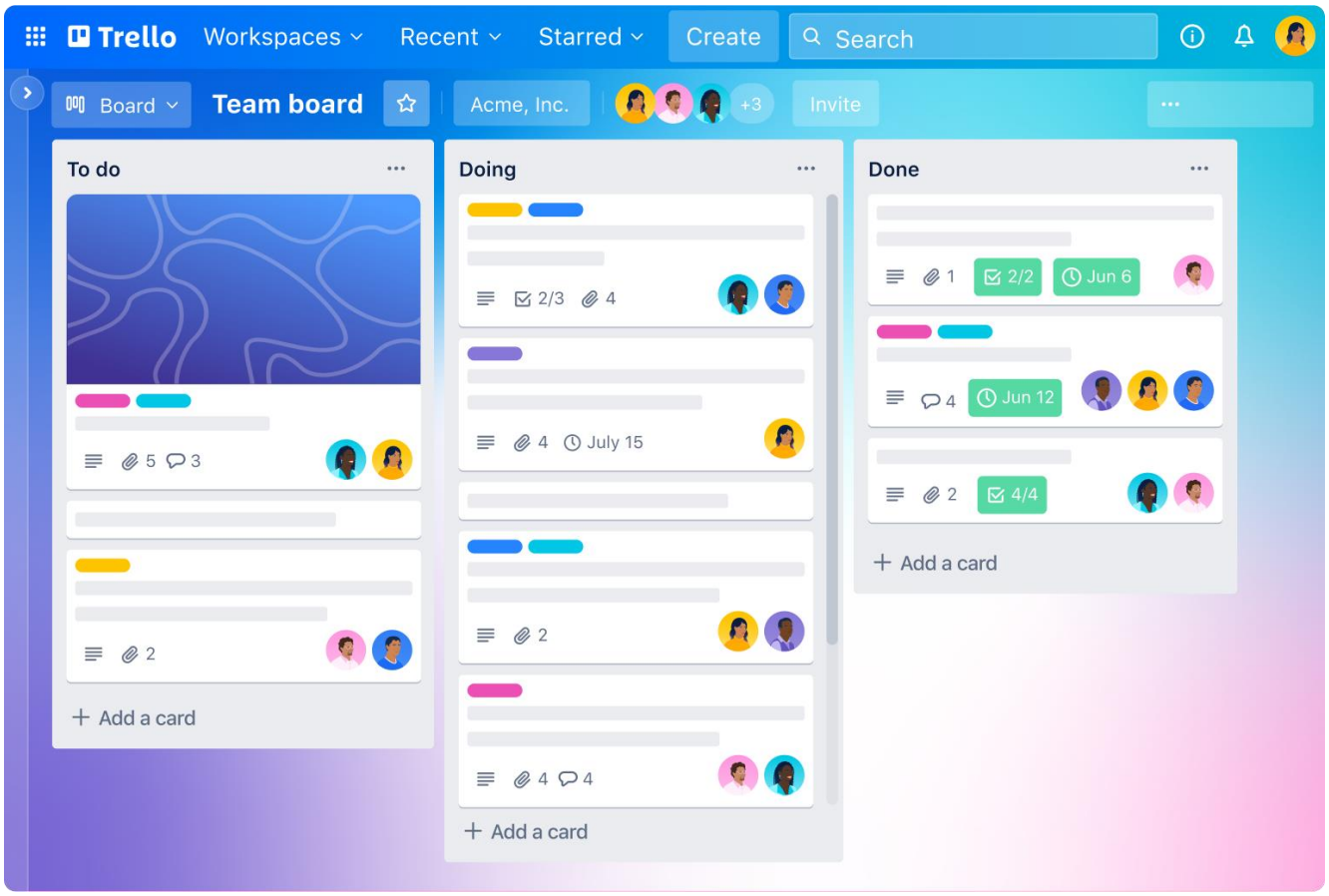


Рисунок 1.4 – Trello

Недоліки використання Trello [38-42]:

1) обмеження у безкоштовній версії: безкоштовна версія Trello має обмеження на кількість дошок, команд та вкладень; деякі додаткові функції, такі як обмежені можливості налаштування та адміністрування, доступні лише у плат них планах;

2) відсутність розширених функцій управління завданнями: у порівнянні з деякими іншими інструментами для управління завданнями, Trello може бути менш функціональним у деяких аспектах; наприклад, він не надає вбудованих можливостей для складних розкладів, автоматичного призначення завдань або відстеження ресурсів;

3) обмежена можливість сортування та фільтрації: Trello має базові функції сортування та фільтрації, але вони можуть бути обмеженими для великих проектів або завдань з великою кількістю даних;

4) залежність від Інтернет-з'єднання: Trello вимагає постійного Інтернет-з'єднання для роботи та синхронізації даних; це означає, що користувач не зможе отримувати доступ до своїх завдань або вносити зміни, якщо у нього немає підключення до Інтернету;

5) обмежена спеціалізація: Trello зосереджений переважно на керуванні проектами та завданнями, тому він може бути менш підходящим для комплексного управління процесами або спеціалізованих потреб певних галузей.

Враховуючи ці переваги та недоліки, користувач може визначити, чи відповідає Trello його потребам у керуванні завданнями та проектами.

Asana (рис. 1.5) є ще одним популярним інструментом для керування проектами та завданнями.

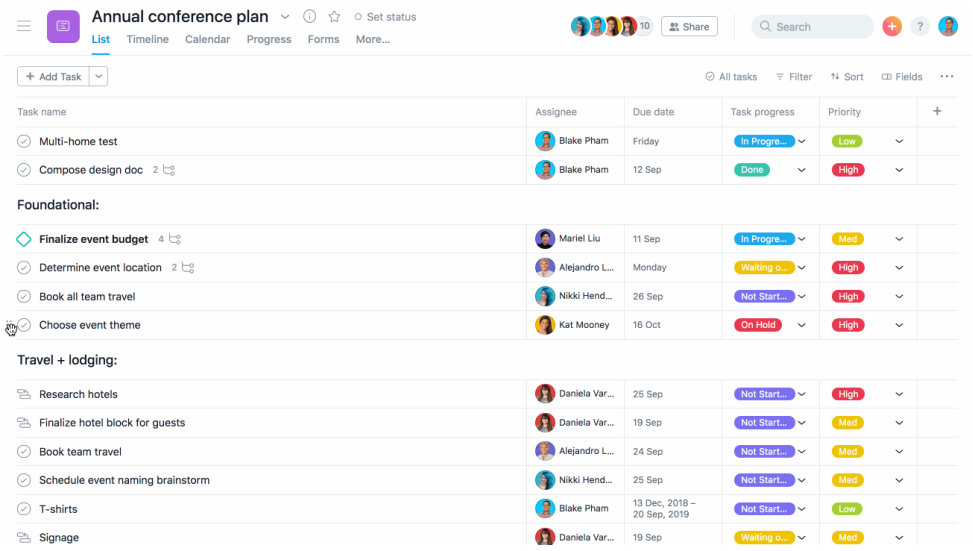


Рисунок 1.5 – Asana

Переваги використання Asana [43-46]:

1) потужність та функціональність: Asana надає широкі можливості для організації завдань, проектів, команд та термінів; користувач може створювати завдання, призначати їх учасникам, налаштовувати терміни, встановлювати пріоритети та використовувати різні функції для спільної роботи;

2) візуалізація та організація завдань: Asana надає гнучкість у візуалізації завдань, проектів та списків; користувач може використовувати дошки, список завдань, графіки Ганта та інші інструменти для кращого огляду та організації роботи;

3) колаборація та комунікація: Asana дозволяє користувачу спілкуватися та співпрацювати з колегами безпосередньо в контексті завдань та проектів; користувач може додавати коментарі, обговорювати питання, ділитися файлами та вести комунікацію в реальному часі;

4) інтеграція з іншими інструментами: Asana може інтегруватися з багатьма іншими популярними інструментами, такими як Slack, Google Drive, Dropbox, Microsoft Teams та інші; це дозволяє зручно поєднувати Asana з іншими робочими інструментами та автоматизувати робочі процеси.

Недоліки використання Asana [43-46]:

1) навчання та впровадження: оскільки Asana має багато функцій та можливостей, впровадження та навчання користувачів можуть зайняти певний час та зусилля; потрібно ознайомитися з інтерфейсом, нащо користувач повинен вивчити різні функції та налаштування для ефективного використання Asana;

2) обмежена безкоштовна версія: хоча Asana має безкоштовну версію, вона має обмежені функції та обмежену кількість користувачів; для повного доступу до всіх функцій та для великих команд може знадобитися платна підписка;

3) комплексність для простих проектів: якщо користувачу потрібно просто вести список завдань або керувати невеликими проектами, Asana може бути занадто складним; він більше підходить для складних та багатозадачних проектів;

4) залежність від Інтернет-з'єднання: Asana є хмарним сервісом, тому для його використання потрібне постійне Інтернет-з'єднання; це може бути обмеженням для користувачів, які працюють в офлайн-режимі або в умовах зі слабким Інтернет-з'єднанням;

5) обмежені можливості налаштування: деякі користувачі можуть відчувати обмеження у можливостях налаштування Asana; наприклад, обмежені можливості налаштування кольорів, макетів або вигляду списків завдань.

Оцінюючи ці переваги та недоліки, користувач може визначити, чи відповідає Asana його потребам у керуванні проектами та завданнями.

Існує також і багато інших інструментів для організації повсякденної діяльності користувача (програм-органайзерів), які допомагають керувати часом, завданнями, замітками та іншими аспектами повсякденного життя, наприклад:

1) Evernote: Evernote дозволяє створювати й організовувати замітки, списки завдань, зберігати фотографії та збирати ідеї; він має потужні функції пошуку та синхронізації між пристроями;

2) Any.do: Any.do є простим та ефективним інструментом для керування завданнями та списками; він дозволяє створювати завдання, нагадування, планувати події та спільно працювати з командою;

3) Wunderlist: Wunderlist дозволяє створювати списки завдань, встановлювати нагадування, додавати коментарі та використовувати шаблони; він має простий та інтуїтивно зрозумілий інтерфейс;

4) Google Keep: Google Keep є простим інструментом для створення заміток, списків та нагадувань; він синхронізується з обліковим записом Google, що дозволяє отримати доступ до них з будь-якого пристрою.

Кожна з цих програм-органайзерів має свої переваги та функціональність. Вибір залежить від особистого стилю роботи та потреб користувача.

Отже, інструменти для організації повсякденної діяльності користувача (програми-органайзери) – це додатки, які допомагають організувати роботу та керувати завданнями. Вони дають змогу створювати списки справ, розклади,

нагадування, нотатки тощо. Деякі з найпопулярніших програм-органайзерів включають Todoist, Trello, Asana, Google Keep, Evernote, Microsoft To Do і багато інших. Вони можуть бути корисні як для особистого використання, так і для роботи в команді, щоб спростити планування та координацію завдань.

1.2 Вибір моделі життєвого циклу, обгрунтування вибору апаратних та програмних ресурсів для реалізації поставленої задачі

Інформаційна система організації повсякденної діяльності користувача відноситься до прикладного програмного забезпечення і призначена для накопичення інформації користувача, а потім для організації справ і контролю за їх виконанням, відстеження певних подій користувачем. Функції типової інформаційної системи організації повсякденної діяльності користувача (органайзера) пов'язані із забезпеченням роботи наступних підрозділів: Календар; Менеджер контактів; Записна книжка і листки-замітки; Події, прив'язані до певної дати і часу; Планувальник завдань; Нагадувачі-будильники про певні користувачем події. Деякі системи-органайзери можуть як не мати будь-якого з перерахованих підрозділів, так і забезпечувати додаткову функціональність, наприклад, дозволяти працювати з електронною поштою, таким чином виконуючи функції поштового клієнта.

Для розроблення інформаційної системи організації повсякденної діяльності користувача була обрана каскадна модель життєвого циклу – рис. 1.6 [1-10]. Оскільки при проєктуванні та розробленні такої інформаційної системи встановлені чіткі цілі, тому саме така модель є найоптимальнішою. Перевагою цієї моделі є, те, що життєвий цикл розбивається на етапи, яких розробник повинен дотримуватися.

Каскадна модель чудово себе показує при проєктуванні та розробленні інформаційних систем, для яких потрібно сформувати всі необхідні етапи, умови і критерії. Але у такої моделі є свої недоліки, такі як: довге очікування результату,

високий шанс ризику помилок на кінцевих етапах розробки, важкість внесення змін, надлишкове проєктування. Вимоги до інформаційної системи, визначені на стадії формування вимог, документуються у вигляді технічного завдання і фіксуються на весь час розроблення. Критерієм якості розробки за такої моделі є точність виконання специфікацій технічного завдання.

Рисунок 1.6 – Схема каскадної моделі життєвого циклу

Завданням є створення інформаційної системи організації повсякденної діяльності користувача [11-15] у вигляді веб-базованої програми та мобільного додатку, що надають можливість користувачеві формувати декілька дошок, у яких будуть знаходитися окремі задачі, які можна буде переміщувати в залежності від статусу.

Споживачі системи: будь-яка людина, якій потрібна допомога в організації повсякденної діяльності.

Мета проєктування: збереження, обмін та поширення інформації, надання швидкого доступу користувачу до потрібної йому інформації.

Вимоги до апаратних ресурсів: проектоване програмне забезпечення для коректної своєї роботи не потребує найбільш сучасних і продуктивних апаратних ресурсів. Достатнім є наявність у користувача ПК з такими мінімальними вимогами:

1) процесор - Pentium-III 800 Мгц (рекомендовано 1200 Мгц);
2) оперативна пам'ять - 512 Мб (рекомендовано 1024 Мб);
3) монітор з роздільною здатністю не менше 800x600 пікселів;
4) операційна система – Linux, Microsoft® Windows 2000, XP, Vista, 7, 8, 10;
5) браузер:

a) Mozilla Firefox;
b) Opera;
c) Google Chrome;
d) Safari;
e) Internet Explorer (не нижче 7 версії).

Також користувач може скачати мобільний додаток на телефон. Для цього потрібно мати такі рекомендовані вимоги:

1) процесор - Qualcomm Snapdragon 675;
2) оперативна пам’ять – 2 ГБ;
3) діагональ – 6.3 дюйми;
4) платформа – Android.

Атрибути системи:

Мова реалізації = JS, HTML/CSS, JAVA.

Тип СУБД = mySQL.

Групи користувачів = користувач.

Стиль інтерфейсу = графічний, кольоровий.

Доступ до записів сайту = для користувача.

Захист = хороший.

1.3 Висновки. Постановка задачі

Попереднє планування справ допомагає підвищити ефективність будь-якої діяльності – як особистої, так і професійної.

Інформаційні системи організації повсякденної діяльності користувача (органайзери) дозволяють проводити планування особистого часу користувача, своєчасно нагадують про початок запланованих заходів та зустрічей, про необхідність виконання тих чи інших нагальних справ, ведуть персональні та інші картки з можливістю автоматичної вибірки інформації, ведуть персональні інформаційні записники для збереження різноманітної особистої інформації. Такі системи призначені для накопичення інформації користувача, а потім для організації справ і контролю за їх виконанням, відслідковування визначених користувачем подій.

В кваліфікаційній роботі слід розв’язати наступну задачу: розробити інформаційну систему організації повсякденної діяльності користувача (органайзер), яка відповідатиме за організацію повсякденної діяльності незалежно від її специфіки. Проєктування та розроблення такого органайзера повинні базуватись на принципі розділення його функціональності на діяльності (проєкти) та задачі в цих діяльностях, стан яких користувач буде визначати і переглядати. Інтерфейс має бути простим в розумінні і не обтяжуватись специфічними можливостями, в той же час програма повинна бути різнобічна і мати клієнт-серверну архітектуру, щоб користуватись нею могли клієнти різного типу.

АРХІТЕКТУРА ІНФОРМАЦІЙНОЇ СИСТЕМИ ОРГАНІЗАЦІЇ ПОВСЯКДЕННОЇ ДІЯЛЬНОСТІ КОРИСТУВАЧА

2.1 Графік виконання проєктних робіт, аналіз ризиків

Для графічного відображення етапів проекту використовують мережеву діаграму, на якій зображується взаємодія етапів, їх послідовність і час виконання. При побудові мережевого графіка робіт створюється граф (рис. 2.1), в якому вказуються:

1) початкова точка – подія або набір подій, які відбулися допочатку виконання даної фази процесу;

2) тривалість – інтервал часу, за який процес повинен успішно завершити своє виконання;

3) кінцева точка процесу – контрольна точка, в якій замовник перевіряє якість отриманих результатів процесу.

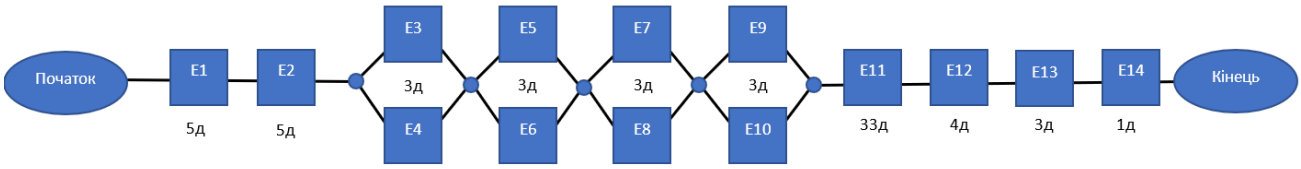


Рисунок 2.1 – Мережна діаграма етапів

На представленій мережній даіграмі:

E1 – аналіз предметної області

E2 – формування вимог до програми

E3 – побудова діаграми варіантів використання

E4 – побудова діаграми класів

E5 – побудова діаграми станів

E6 – побудова діаграми діяльності

E7 – побудова діаграми послідовності

E8 – побудова діаграми кооперацій

E9 – побудова діаграми компонентів

E10 – побудова діаграми розгортання

E11 – реалізація програми в кодах

E12 – відлагодження

E13 – тестування

E14 – впровадження і експлуатація.

Критичний шлях – найбільш довгий повний шлях в мережі; роботи, які лежать на цьому шляху, також називаються критичними. Саме тривалість критичного шляху визначає найменшу загальну тривалість робіт з проекту в цілому. Час виконання всього проекту в цілому може бути скорочений за рахунок скорочення часу виконання завдань, які лежать на критичному шляху. Відповідно будь-яка затримка виконання завдань критичного шляху приводить до збільшення часу виконання проекту.

Критичний шлях в графі указує максимальну тривалість робіт на графі (від початкової роботи до останньої). При виконанні проекту вибираються і виконуються роботи, які не впливають на час виконання інших (незалежних) робіт проекту або на їх тривалість. Роботи на критичному шляху можуть скорочуватися за рахунок зміни часу виконання.

Для підрахунку критичного шляху визначимо максимальний час реалізації проекту:

Критичний шлях = 5днів + 5днів + 4*3 дні + 33 дні + 4 дні + 3дні + 1день = 63дні.

Найчастіше визначення ролей виконавців проекту відповідає етапам розробки. Склад і кількість співробітників, що входять до групи проекту, залежить від масштабу робіт і досвіду співробітників. Співробітники повинні бути настільки кваліфікованими, щоб могли виявити помилки і неточності в проекті на самих ранніх етапах розробки.

Графік робіт представляється також у вигляді діаграм і графіків. Часові діаграми відображають час початку і закінчення етапу на відносній шкалі (рис. 2.2).

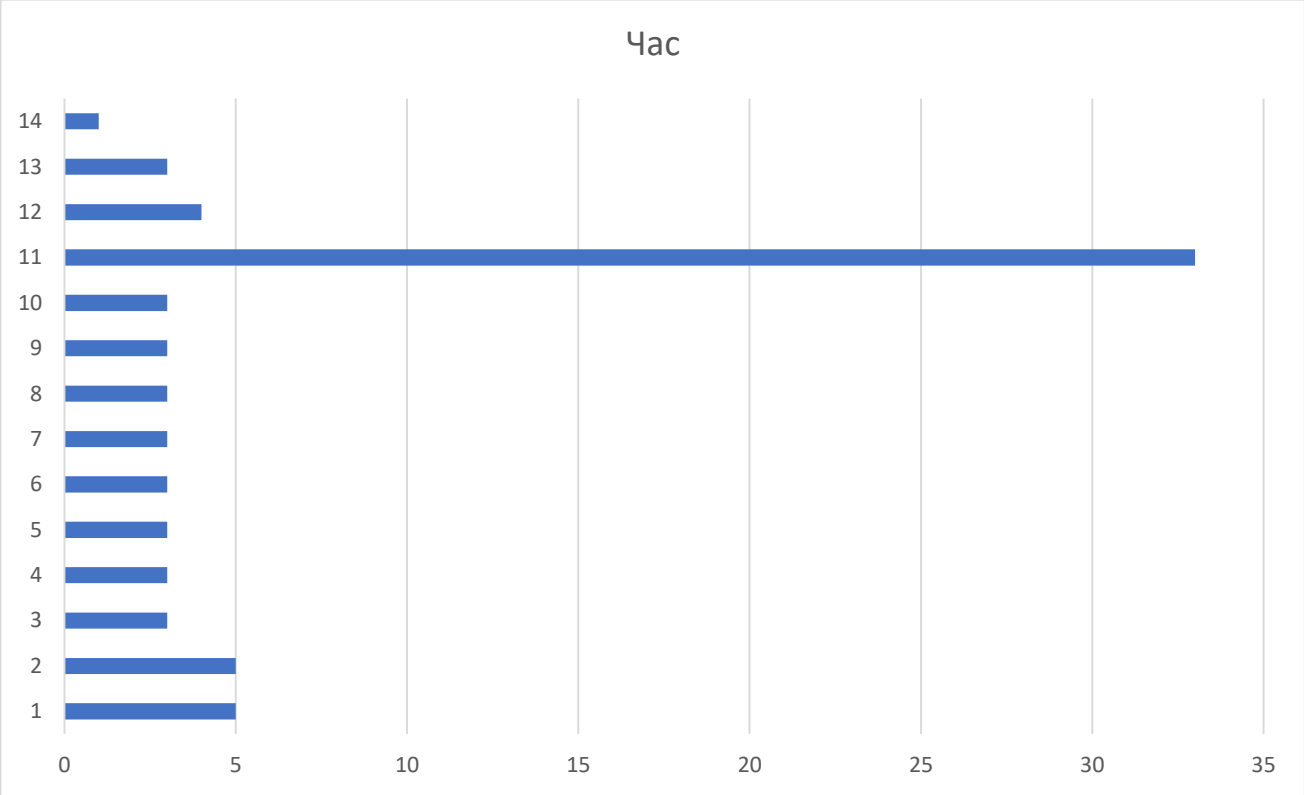


Рисунок 2.2 – Часова діаграма (відносна шкала)

Поняття «ризик проекту» пов’язується з можливістю понести втрати в ході виконання проекту. Ці втрати можуть проявлятися в зниженні якості кінцевого продукту, перевищенні вартості його розробки, затримці закінчення розробки або в зриві проекту (тобто, відмові замовника від проекту).

Величина ризику визначається добутком серйозності наслідків небажаної події в проекті (рівнем втрат) та ймовірності настання цієї події.

Серйозність наслідків оцінюється з позицій впливу небажаної події, з одного боку, на характеристики ПЗ і складність її подальшого супроводу, а, з іншого, - на ефективність, вартість і тривалість процесу розробки ПЗ.

Результати аналізу ризиків представляють у вигляді таблиці ризиків, впорядкованих по ступеню можливого збитку (табл. 2.1).

Таблиця 2.1 – Список ризиків після проведення їх аналізу

№	Ризик	Імовірність виникнення	Стратегія усунення	Імовірність після стратегії
1	Підбір досвідченої команди експертів	Висока	Найняти досвідченого працівника з найму персоналу	Низька
2	Організація працівників	Середня	Найняти людину як буде організовувати команду	Низька
3	Хвороби працівників	Висока	Введення в курс проекту всіх розробників	Низька
4	Технічні проблеми	Висока	Найняти працівника який буде відповідати за технічні моменти	Низька
5	Виникнення конфліктів	Середня	Чітко розподілити між учасниками проекту їхні обов’язки	Низька
6	Нехватка часу на виконання проект	Висока	Розробити план виконання проекту з часовими рамками	Низька

Аналіз ризиків поділяють на два види: кількісний та якісний. Кількісний аналіз ризику повинен дати можливість визначити число та розміри окремих ризиків та ризику проекту в цілому. Якісний аналіз визначає фактори, межі та види ризиків. Для аналізу ризику використовують метод аналогії, метод експертних оцінок, розрахунково-аналітичний метод та статистичний метод.

Метод аналогій передбачає використання даних по інших проектах, які вже виконані. Цей метод використовується страховими компаніями, які постійно публікують дані про найбільш важливі зони ризику та понесені витрати.

Експертний метод, який відомий як метод експертних оцінок, стосовно підприємницьких проектів може бути реалізований шляхом вивчення думок досвідчених керівників та спеціалістів. При цьому доцільно встановити показники найбільш допустимих, критичних та катастрофічних втрат, маючи на увазі як їх рівень, так і ймовірність.

Удосконаленням цього методу є застосування нечіткої логіки при оцінюванні експертами степені ризику.

Розрахунково-аналітичний метод базується на теоретичних уявленнях. Але прикладна теорія ризику добре розроблена лише для страхового та грального ризику.

Статистичний метод спочатку використовувався в системі PERT для визначення очікуваної тривалості кожної роботи та проекту в цілому. Останнім часом найбільш застосовуваним став метод статистичних випробувань (метод «Монте–Карло»). До переваг цього методу відносять можливість аналізувати та оцінювати різні шляхи реалізації проекту. Подальшим розвитком статистичних методів визначення ризиків є застосування байєсівських правил визначення вірогідності ризику на вершинах мережного графіку виконання проекту.

2.2 Формування вимог до інформаційної системи

Вимоги користувача:

1) реєстрація;
2) авторизація;
3) додавання проектів;
4) додавання задач до проектів;
5) редагування проектів;

6) редагування задач;

7) перегляд проектів;

8) перегляд задач окремих проектів;

9) перегляд всіх задач;

10) визначення статусу виконання задачі;

11) синхронізація даних із сервером;

12) робота програмного забезпечення у форматі веб-сторінки;

13) робота програмного забезпечення у форматі мобільного додатку;

14) можливість адміністрування на стороні сервера.

Функціональні вимоги – це вимоги до програмного забезпечення, які описують внутрішню роботу системи, її поведінку: калькулювання даних, маніпулювання даними, обробка даних та інші специфічні функції, які має виконувати система. На відміну від нефункціональних вимог, що визначають, якою система повинна бути, функціональні вимоги визначають, що система повинна робити.

Функціональні вимоги до даного проєкту:

1) реєстрація:

1.1) створити форму з двома полями для введення користувачем логіна та пароля для ідентифікації у системі; додати кнопку для виконання операції;

1.2) після натиснення кнопки дані відправляються на віддалений сервер, де виконується перевірка коректності даних;

1.3) далі відбувається аналіз повідомлення із серверу – якщо користувача зареєстровано, відбувається подальша робота, якщо помилка – відбувається її аналіз;

1.4) якщо у БД не знайдено такого логіна – відбувається перевірка коректності інших даних, а якщо у БД вже існує такий логін – система виводить повідомлення про існування користувача з таким логіном;

2) авторизація:

2.1) створити форму з двома полями для введення користувачем логіна та пароля для ідентифікації у системі; додамо кнопку для виконання операції;

2.2) після натискання кнопки входу в систему відбувається перевірка коректності даних;

2.3) після відправлення даних на сервер формується токен – ідентифікатор окремого користувача для доступу до його даних;

2.4) якщо операція пройшла успішно – завантажити дані із віддаленої бази даних та надати права на користування сервісом;

3) додавання проектів:

3.1) додати елемент інтерфейсу для додавання проекту;

3.2) при додаванні надати можливість вказати ім’я проекту;

3.3) після додавання проект повинен з’явитись у списку проектів користувача з можливістю проведення операцій вибору, редагування та видалення;

4) додавання задач до проектів:

4.1) створити елемент інтерфейсу для вибору активного проекту;

4.2) створити форму для додавання нової задачі, вона повинна містити поля для введення імені, встановлення дати, часу та статусу виконання;

4.3) після введення даних відправляється запит на сервер та повертається повідомлення із результатами виконання;

5) редагування проектів:

5.1) створимо форму для редагування проекту;

5.2) створимо елемент інтерфейсу для виклику форми;

5.3) після введення даних відправляється запит на сервер та повертається повідомлення із результатами виконання;

6) редагування задач:

6.1) створимо форму для редагування задачі; вона повинна містити поле для введення імені та елементи інтерфейсу для встановлення значень статусу, дати і часу;

6.2) створимо елемент інтерфейсу для виклику форми;

6.3) після введення даних відправляється запит на сервер та повертається повідомлення із результатами виконання;

7) перегляд проектів:

7.1) проекти повинні відображатись на головній формі програмної системи;

7.2) повинна бути забезпечена можливість вибору активного проекту для відображення задач, що відносяться до нього;

7.3) кількість проектів не повинна бути фіксованою;

8) перегляд задач окремих проектів:

8.1) створимо форму для перегляду задач обраного проекту; вона повинна містити всі задачі та елементи керування ними;

8.2) якщо користувач змінює активний проект, набір задач повинен бути замінений відповідним;

8.3) проекти та задачі повинні завантажуватись із віддаленого серверу та однаково відображатись на різних пристроях;

9) перегляд всіх задач:

9.1) створити елемент інтерфейсу для відображення усіх задач користувача;

9.2) задачі повинні відображати усі дані;

9.3) створимо елемент інтерфейсу для виклику форми;

9.4) задачі повинні завантажуватись із віддаленого серверу та однаково відображатись на різних пристроях;

10) визначення статусу виконання задачі:

10.1) додати елемент інтерфейсу для визначення статусу виконання задачі;

10.2) всі зміни повинні бути відправлені для збереження на віддаленому сервері;

10.3) повідомлення про статус виконання операції на сервері повинне бути опрацьованим;

11) синхронізація даних із сервером:

11.1) дані для користувача повинні завантажуватись із віддаленого сервера;

11.2) зміни потрібно синхронізувати із сервером для забезпечення цілісності даних;

11.3) повідомлення про статус виконання операції на сервері повинне бути опрацьованим;

12) робота програмного забезпечення у форматі веб-сторінки:

12.1) повинна бути можливість для використання сервісу за допомогою використання веб-сторінки у браузері;

12.2) дані повинні бути синхронізовані із віддаленим сервером;

12.3) інтерфейс повинен бути інтуїтивно зрозумілим та лаконічним;

13) робота програмного забезпечення у форматі мобільного додатку:

13.1) повинна бути можливість для використання сервісу за допомогою використання мобільного додатку;

13.2) дані повинні бути синхронізовані із віддаленим сервером;

13.3) інтерфейс повинен бути інтуїтивно зрозумілим та лаконічним;

14) можливість адміністрування на стороні сервера:

14.1) у разі необхідності повинна бути можливість доступу до бази даних на стороні віддаленого серверу;

14.2) доступ до даних повинен відбуватись із правами адміністратора.

Нефункціональні вимоги — це вимоги до програмного забезпечення, які задають критерії для оцінки якості його роботи. На відміну від функціональних вимог, які визначають, що система повинна робити, нефункціональні вимоги визначають якою система повинна бути.

Нефункціональні вимоги:

1) розподіленість системи – різні функції повинні виконуватись на різних програмних та апаратних комплексах;

2) надійність функціонування системи – стійкість функціонування, незалежно від дій користувача та значення вхідних даних;

3) висока швидкодія системи – програма повинна опрацьовувати запити за прийнятний час;

4) адаптивність системи – програма повинна працювати на різних апаратних конфігураціях;

5) універсальність – програма повинна бути розрахована на широкий діапазон вхідних даних;

6) простота у використанні – програма повинна бути зручною для використання широким колом користувачів;

7) естетичне задоволення – програма повинна мати лаконічний інтерфейс із оптимальними розмірами елементів інтерфейсу.

Верифікація вимог – процес пересвідчення, що програми та їх компоненти виконують запропоновані їм вимоги. Метою верифікації є пересвідчення в тому, що програмне забезпечення відповідає висунутим вимогам. Паралельно з цим фіксуються нові дефекти, додані в процесі розробки. Це метод аналізу, перевірки специфікацій і правильності виконання програм відповідно до заданих вимог і формального опису програми.

Мета процесу валідації – переконатися, що специфічні вимоги для програмного продукту виконано, і здійснюється це за допомогою:

1) розробленої стратегії і критеріїв перевірки всіх робочих продуктів;

2) обговорених дій з проведення валідації;

3) демонстрації відповідності розроблених програмних продуктів вимогам замовника і правилам їхнього використання;

4) узгодження із замовником отриманих результатів валідації продукту.

Процес верифікації полягає у визначенні критичних елементів, що повинні піддаватися верифікації, у виборі виконавця верифікації, інструментальних засобів підтримки процесу верифікації, у складанні плану верифікації і його затвердження. У процесі верифікації виконуються задачі перевірки умов: контракту, процесу,

вимог, інтеграції, коду і документації. Відповідно до плану і вимог замовника перевіряється правильність виконання функцій системи, інтерфейсів і взаємозв'язків компонентів, а також доступ до даних і до засобів захисту.

2.3 Автоматна модель та UML-діаграми інформаційної системи

Взаємодія з інформаційною системою відбувається через клієнтський додаток (Web або Android), тому автоматна модель буде будуватись з точки зору клієнтського додатку (рис. 2.3) в той час, як сама інформаційна система представлена у вигляді готового серверного рішення Back-end з базою даних, яка постійно перебуває в режимі очікування і обробляє запити клієнта.

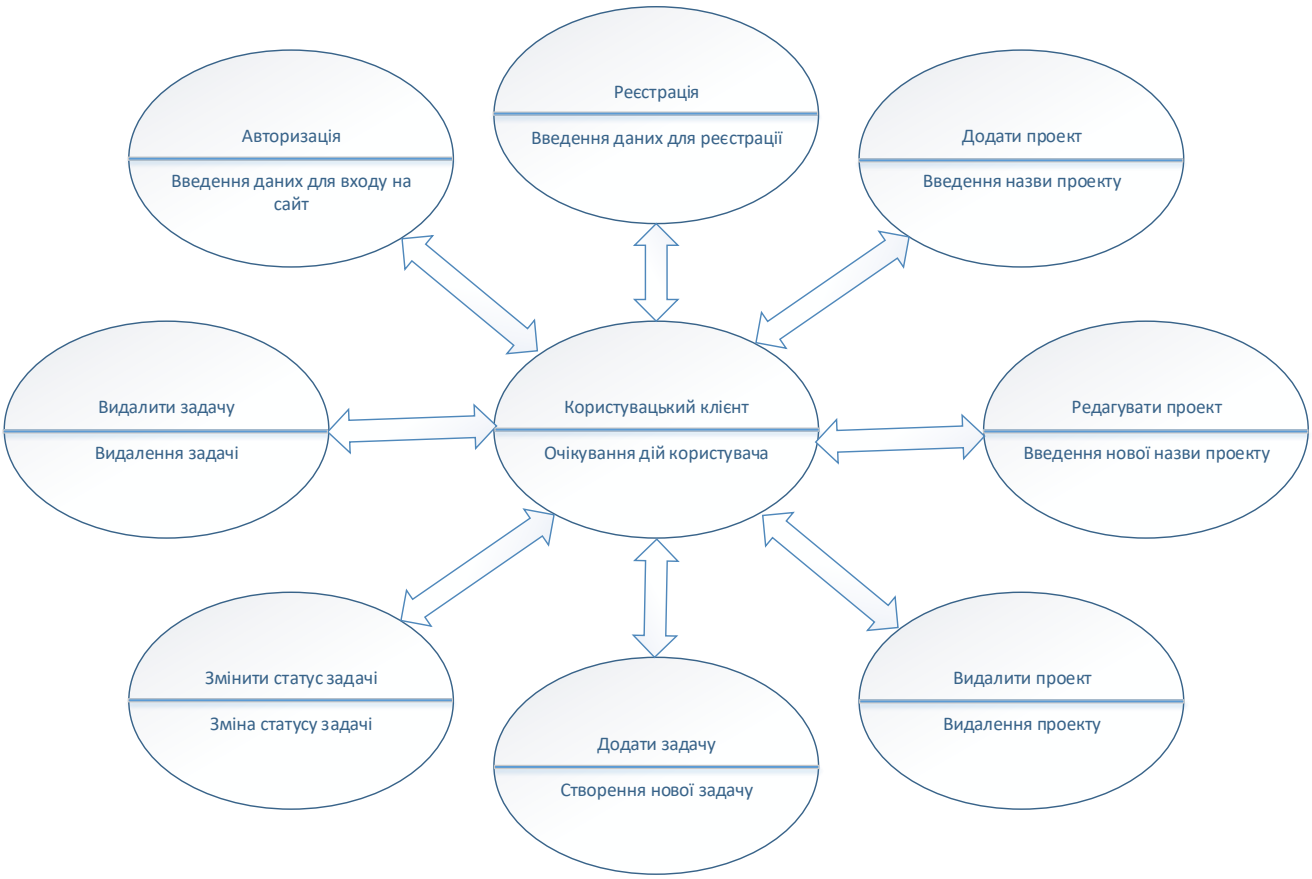


Рисунок 2.3 – Автоматна модель клієнтського додатку інформаційної системи

Коли користувач заходить в додаток, той у випадку незакешованих авторизаційних даних пропонує користувачу ввійти в систему і очікує введення

логіну і паролю та натискання кнопки «вхід». Або ж користувач може зареєструватись в системі, здійснивши ті самі дії і натиснувши кнопку «зареєструватись».

Щойно користувач авторизувався, додаток одержує з серверу дані про його проекти і відображає їх. Користувач може натиснути на «Додавання проекту» і створити новий проект, ввівши його назву в діалоговому вікні. Також користувач може змінити проект, натиснувши кнопку «Змінити проект» або видалити його, натиснувши кнопку «Видалити проект».

Також користувач може натиснути на проект зі списку і одержати задачі по цьому проекту. Відповідно, він може додати задачу, натиснувши відповідну кнопку. Також по перетягуванню задачі в відповідну колонку, або вибором відповідного елемента в випадаючому списку він може змінити її статус (з виконаної до запланованої або навпаки). Наостанок, користувач може видалити непотрібні задачі, натиснувши навпроти них відповідні кнопки.

Мовою UML (Unified Modeling Language) можна розробити моделі, які забезпечать усебічне розуміння концепції проекту, його функцій та моделей даних. За допомогою формалізованої графічної нотації та великої кількості діаграм можна наочно і зрозумілим чином дослідити та зобразити головні особливості проекту як для замовників, так і для учасників розробки. Більшості програмістів вона буде зрозуміла, незалежно від мови програмування та технологій реалізації, що робить мову UML ефективним засобом командної взаємодії та побудови архітектури програмних систем.

Реалізуємо вісім основних UML-діаграм:

1) діаграму варіантів використання (use case diagram);

2) діаграму класів (class diagram);

3) діаграму станів (statechart diagram);

4) діаграму діяльності (activity diagram);

5) діаграму послідовності (sequence diagram);

6) діаграму кооперації (collaboration diagram);

7) діаграму компонентів (component diagram);

8) діаграму розгортання (deployment diagram).

Діаграма варіантів використання необхідна, щоб продемонструвати основні можливості і сценарії використання інформаційної системи користувачем (рис. 2.4).

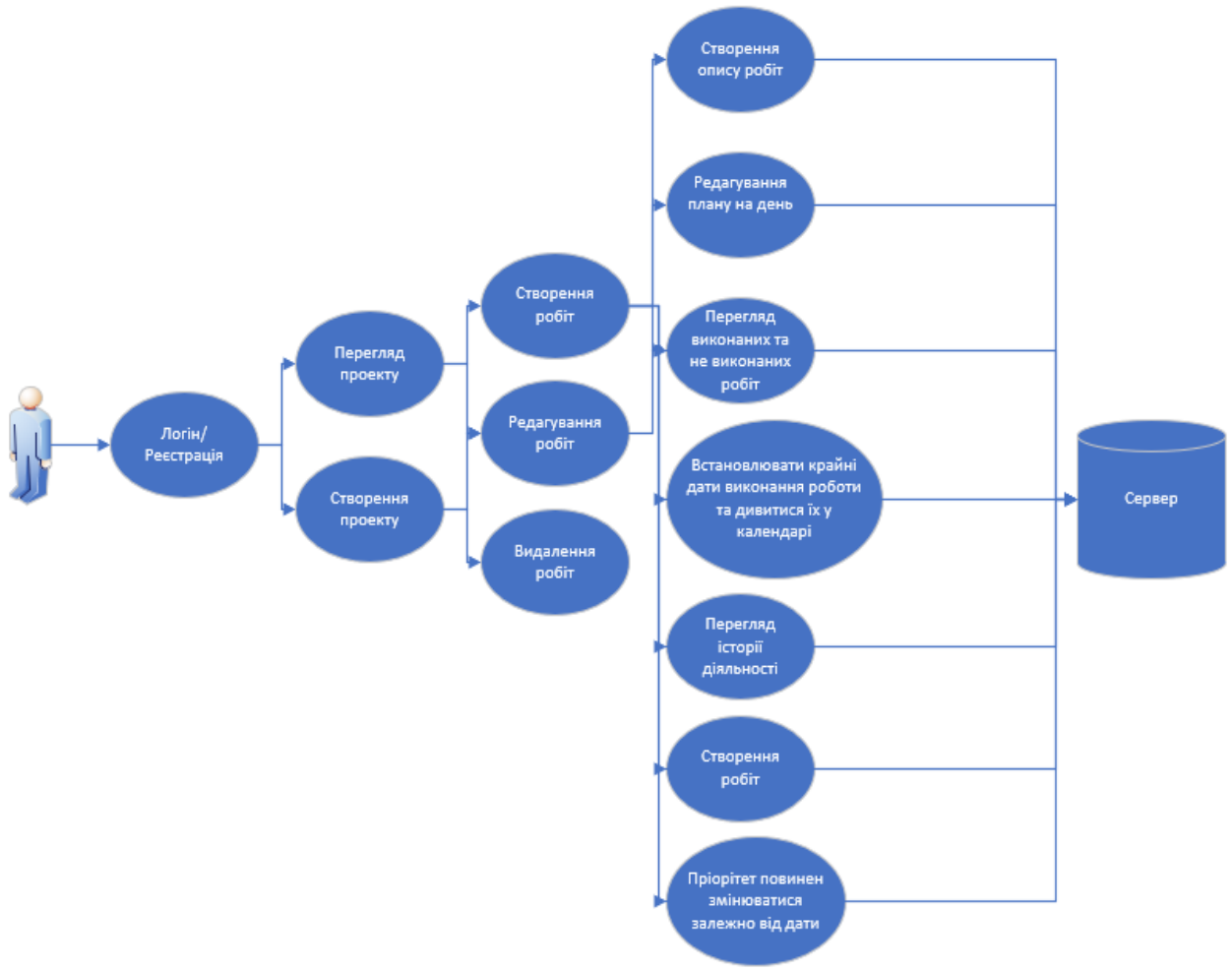


Рисунок 2.4 – UML-діаграма варіантів використання

Діаграма класів використовується для об’єктно-орієнтованого відображення інформаційної системи та бізнес-логіки, що може бути відображена в мові програмування, і є фактично представленням в графічній формі об’єктно-орієнтованих кодових структур та зв’язків між ними. Спробуємо охарактеризувати основні елементи бізнес-логіки та організації даних додатку:

- користувач (User) має проекти (Project);
- користувач (User) має записи журналу (LogMessage);
- проект (Project) має задачі (Task).

Опишемо використовувані класи (рис. 2.5-2.8):

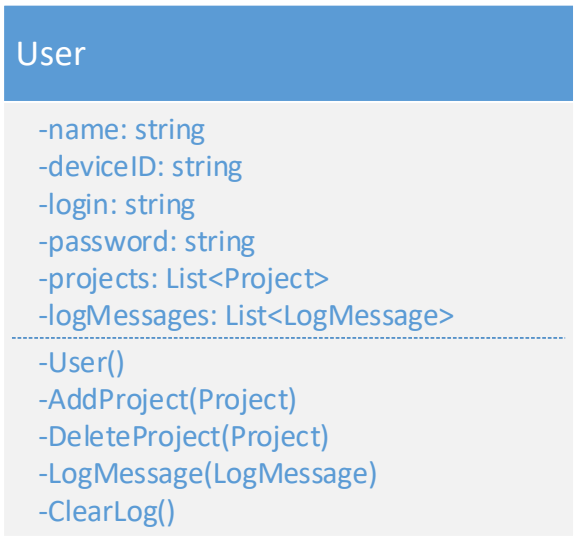


Рисунок 2.5 – Клас User

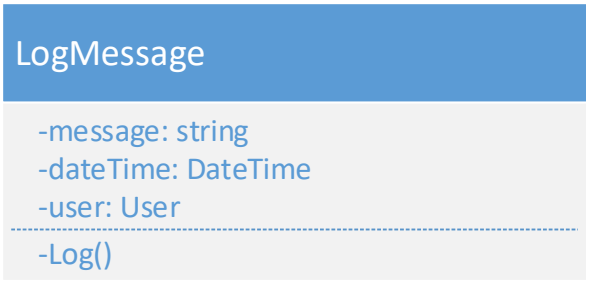


Рисунок 2.6 – Клас LogMessage

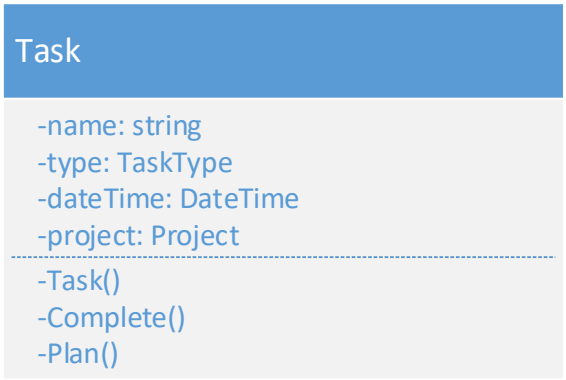


Рисунок 2.7 – Клас Task

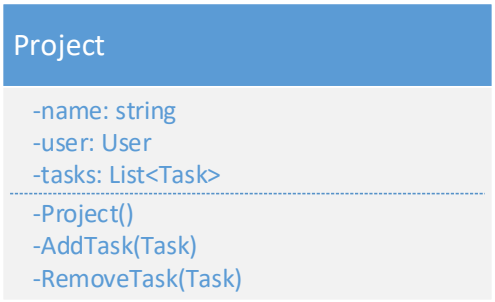


Рисунок 2.8 – Клас Project

Також опис передбачає агрегацію між всіма цими сутностями. Як бачимо, прослідковується агрегація між усіма сутностями. Відобразимо її на діаграмі класів (рис. 2.9):

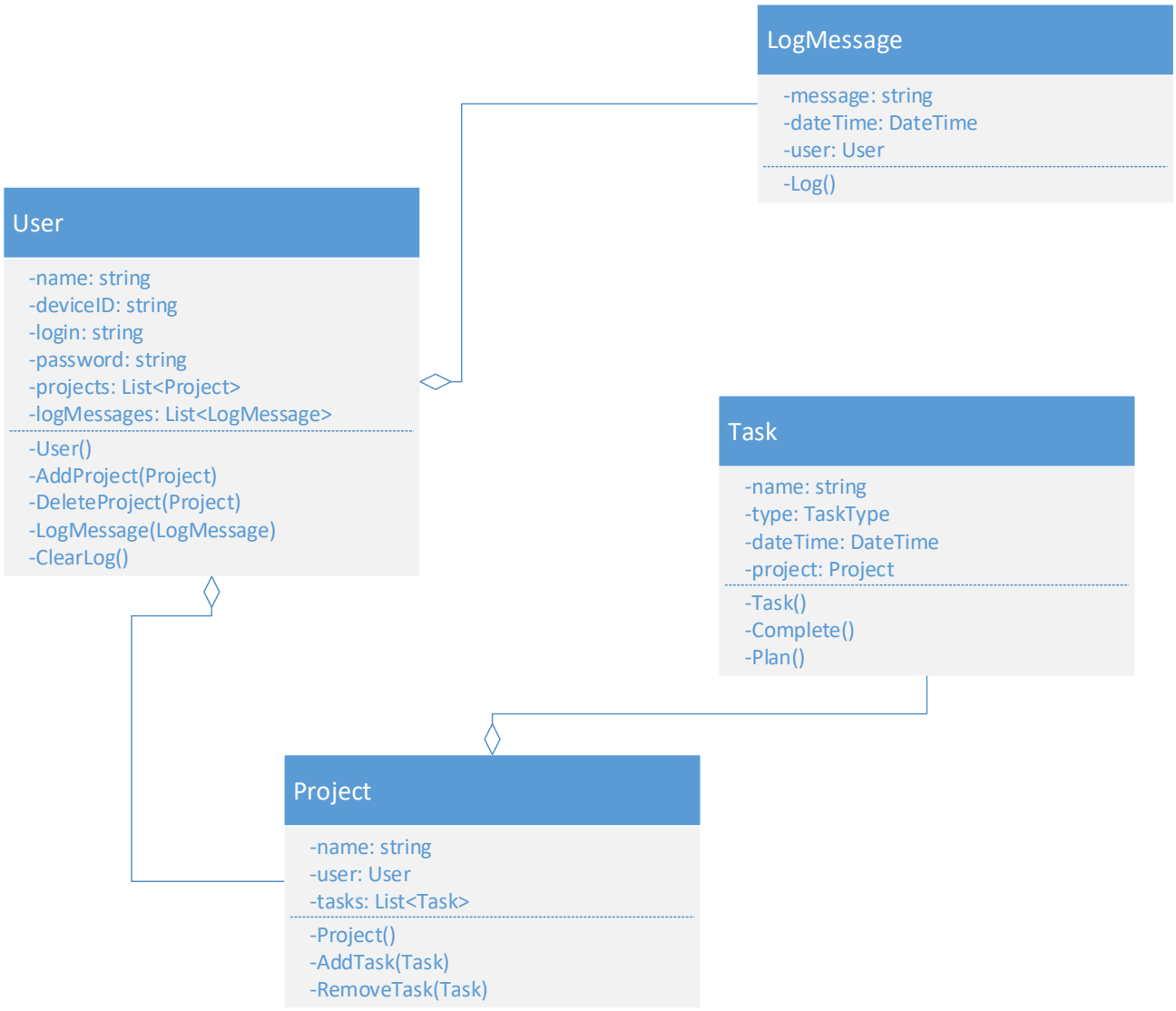


Рисунок 2.9 – UML-діаграма класів

Дана діаграма допоможе сформувати також ER-діаграму бази даних для цього проекта. В ній перелічено основні поля даних для кожної сутності, а також основна поведінка об’єктів та механізми зв’язку між ними. Наприклад, логічно, що користувач може додати або видалити проект, а з проекту може бути додана або ж видалена задача.

Визначимо основні концепти для діаграми станів. Користувач може ввійти або зареєструватися у систему. Після чого у нього будуть такі опції:

1) створення проекту – після того, як був створений проект, користувач може створити, редагувати і видаляти роботу, а також може додавати опис для роботи;

2) перегляд проекту – якщо проект був створений, його можна переглянути, а також в цьому режимі можна створити, редагувати і видаляти роботу і додавати опис для роботи;

3) перегляд виконаних та невиконаних робіт – після того, як користувач зайшов на створений проект або створив новий, він може перевірити, які роботи були виконані, а які – ні;

4) редагування плану на день – користувач може створити план на день, який буде оновлюватися до стану по замовчуванню кожні n годин;

5) перегляд історії діяльності – користувачу буде надана можливість перегляду діяльності по проекту;

6) змінення пріоритету в залежності від дати – користувач може налаштовувати пріоритети робіт в залежності від час закінчення робіт;

7) встановлення крайніх дат робіт та перегляд їх у календарі – окрім встановлення крайніх термінів закінчення робіт, їх можна буде переглядати у календарі, що зробить контроль задач більш простим.

На основі вказаних опцій можна побудувати діаграму станів (рис. 2.10).

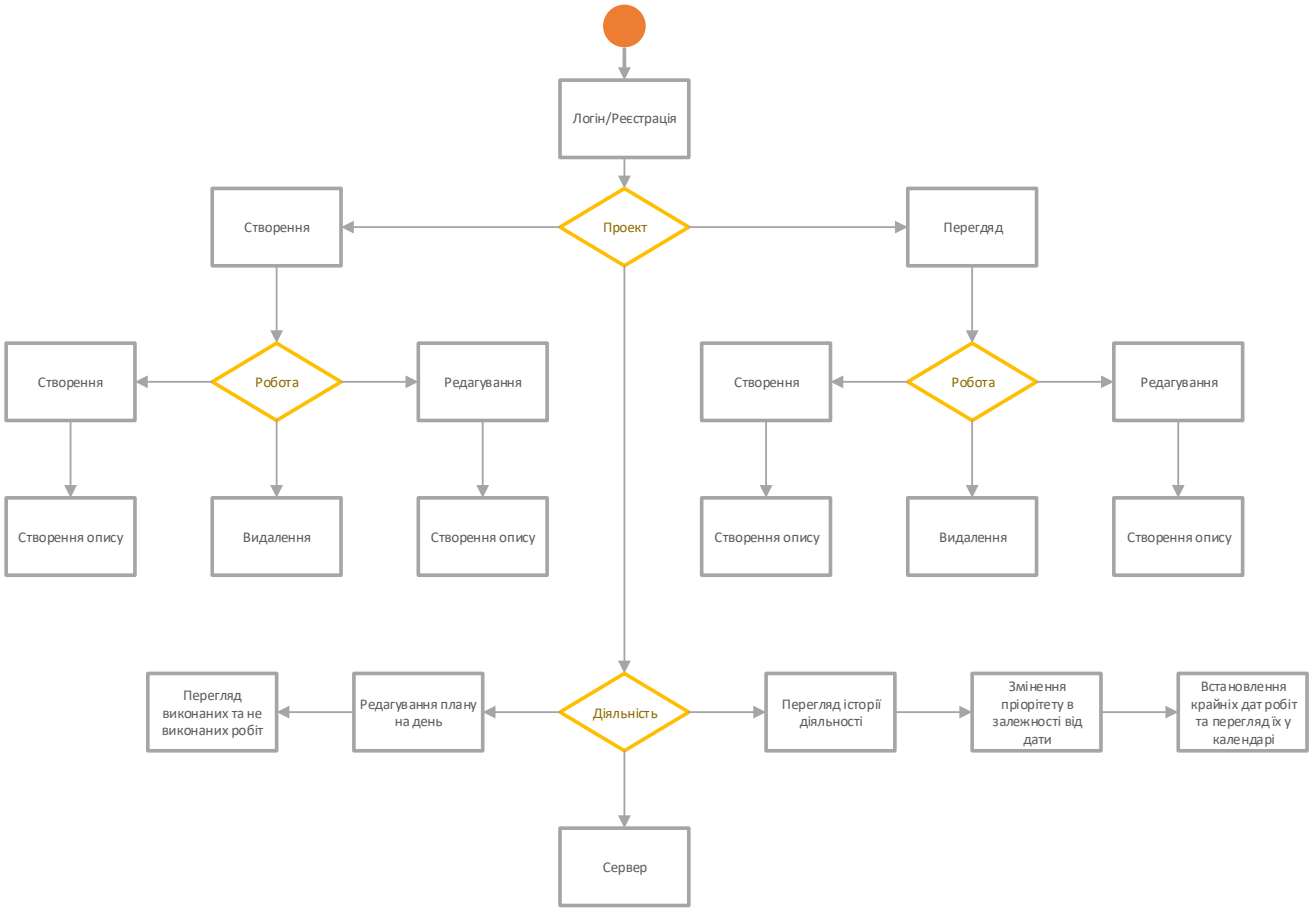


Рисунок 2.10 – UML-діаграма станів

Діаграми діяльності можна вважати частковим випадком діаграм станів. Відмінність полягає в семантиці станів та у відсутності на переходах назв подій з відповідними аргументами (сигнатур подій). Основним напрямком використання діаграм діяльності є візуалізація особливостей реалізації операцій класів, коли необхідно представити алгоритми їхнього виконання. Оскільки діаграма станів описує роботу програми найзагальнішим чином, діаграма діяльності буде більш конкретизовано описувати потік діяльностей при базовому сценарії використання програми користувачем. Після ініціалізації програми і виведення попередньо збережених даних користувач отримає можливість роботи з проектами – видаляти, відкривати та створювати нові. У разі створення нового проекту або відкриття уже існуючого, користувач може провести аналогічні операції із роботами всередині проекту – видалити, створити нову або відмітити виконання роботи – успішного

або невдалого. Після проведення даних операцій програма повинна зберегти зміни та завершити роботу. Зобразимо це на діаграмі діяльності (рис. 2.11).

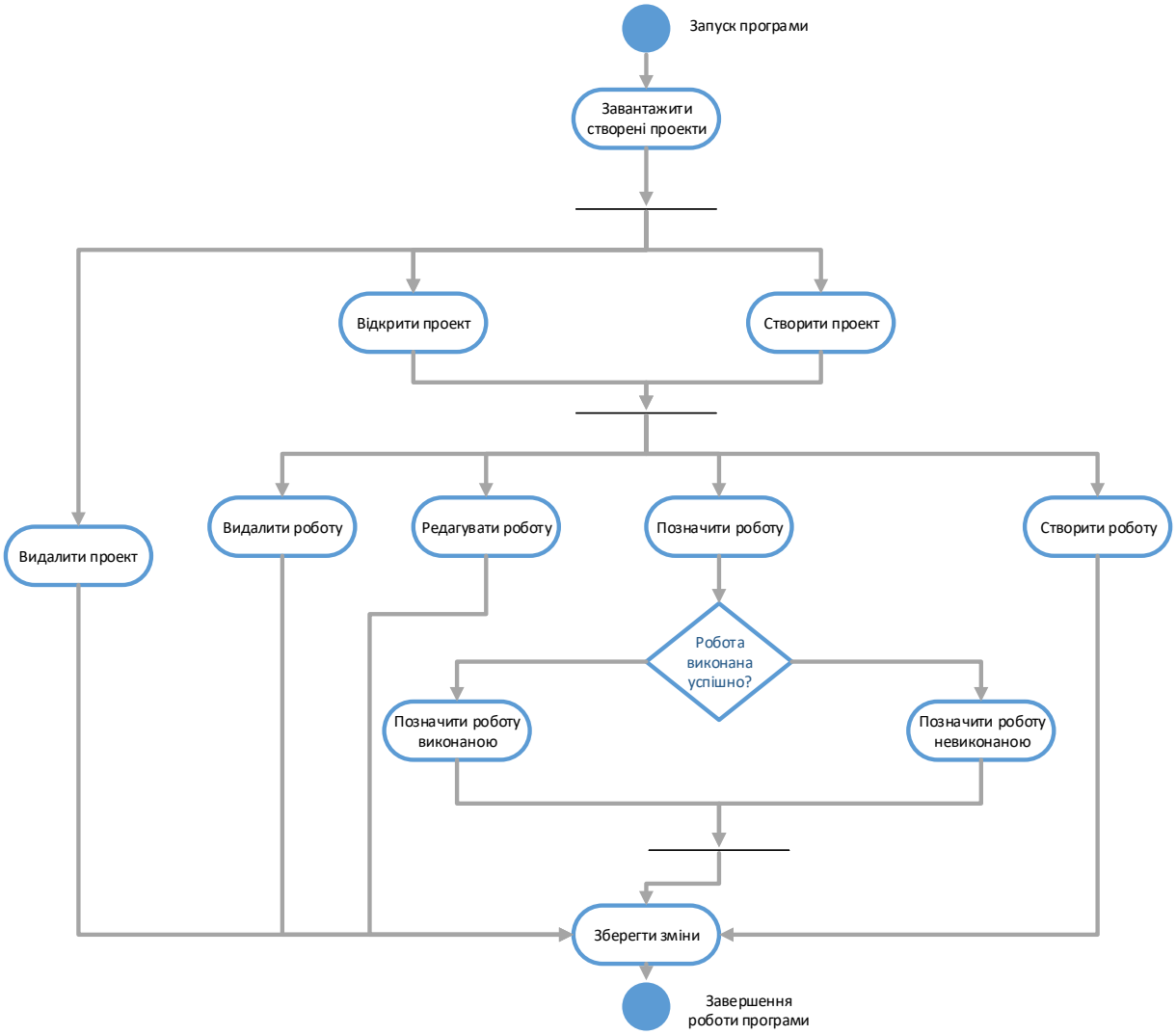


Рисунок 2.11 – UML-діаграма діяльності

Перейдемо до діаграми послідовності. Діаграма послідовності допоможе охарактеризувати процеси та обмін повідомлення між ними в часі. Виділимо основні лінії життя в проекті:

1) користувач (User);
2) клієнтський додаток (Client);
3) сервер (Server).

Спробуємо охарактеризувати за допомогою діаграми послідовності найпоширеніший варіант роботи з додатком-органайзером. А саме – процес обміну повідомленнями між сутностями, коли користувач хоче відкрити додаток та помітити якусь задачу як виконану.

Отже, в нас відбудеться така послідовність дій:

1) користувач відкриває додаток;
2) додаток авторизується на сервері;
3) сервер повертає дані по авторизації додатку;
4) додаток запитує інформацію про проекти;
5) сервер повертає інформацію про проекти додатку;
6) додаток відображає користувачу список проектів;
7) користувач обирає цільовий проект зі списку;
8) додаток запитує у сервера дані по задачам цього проекту;
9) сервер повертає додатку дані по задачам для запитуваного проекту;
10) додаток відображає список задач для проекту;
11) клієнт свайпом помічає задачу як виконану;
12) додаток відправляє серверу запит на зміну стану задачі;
13) сервер присилає відповідь про успішність запиту;
14) додаток відображає списки задач в оновленому вигляді;

Зобразимо це все на діаграмі послідовності (рис. 2.12):

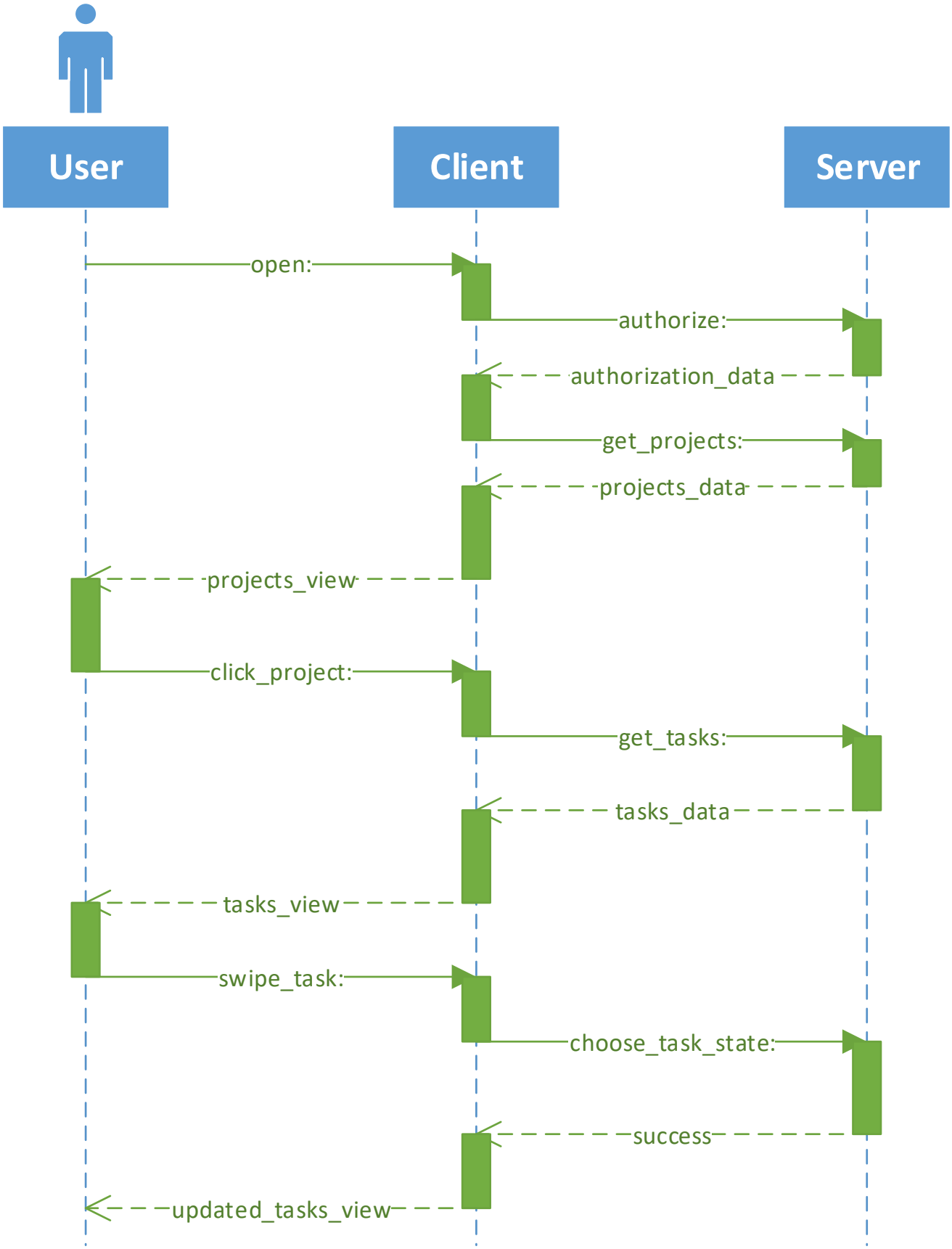


Рисунок 2.12 – UML-діаграма послідовності

Діаграма кооперації призначена для опису взаємодії різних складових частин системи в контексті виконання певних операцій. Дана діаграма описує взаємодію об’єктів програми при створенні плану робіт (рис. 2.13):

1) збереження змін даних при завершенні роботи;
2) фіксація змін у базі даних;
3) завантаження даних при ініціалізації;
4) результати процесу авторизації користувача;
5) користувач вводить дані авторизації;
6) користувач відкриває свої проекти;
7) відображається список робіт проекту;
8) користувач обирає проект – відображаються відповідні роботи;
9) користувач позначає роботу завершеною або редагує її.

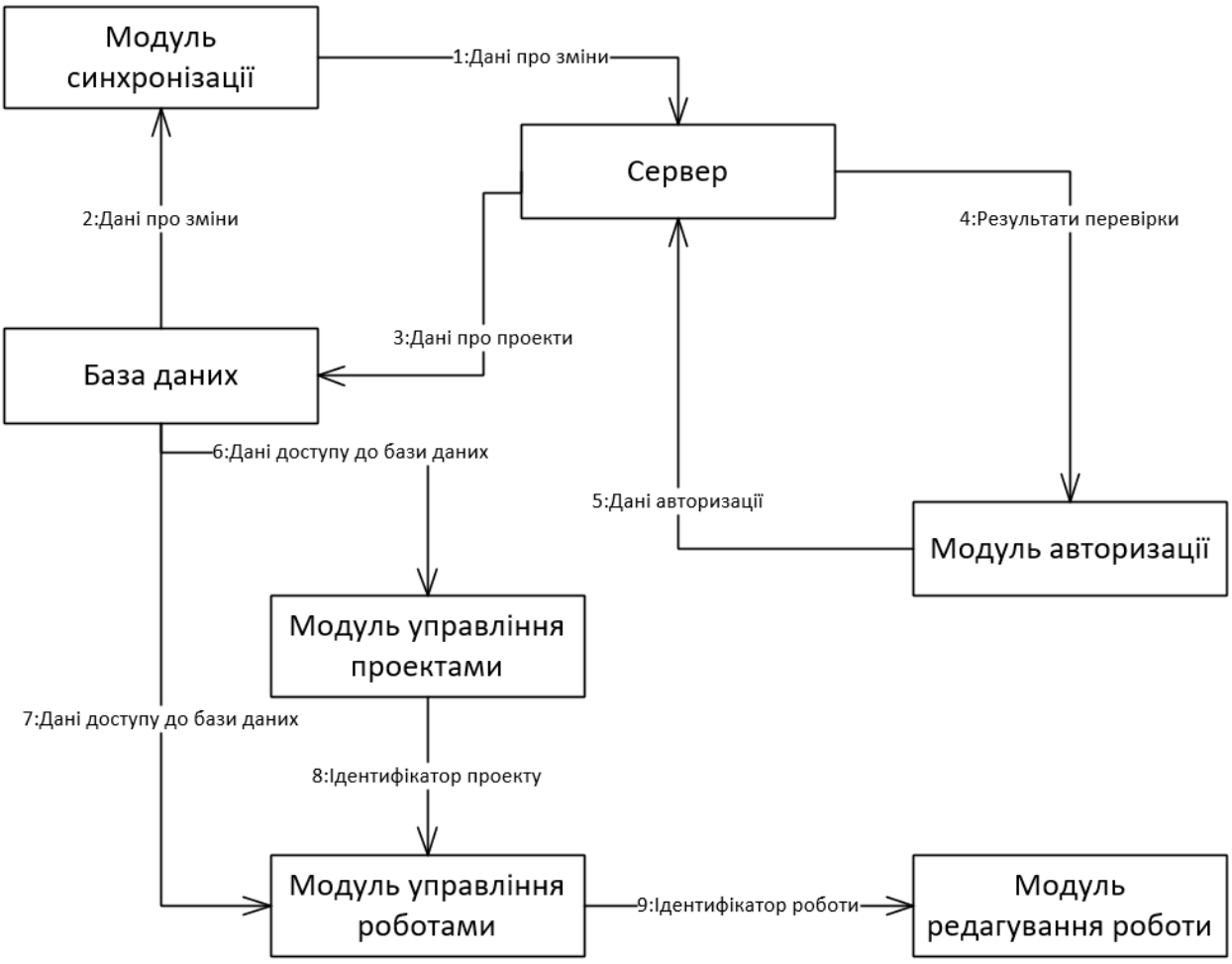


Рисунок 2.13 – UML-діаграма кооперації

Діаграма компонентів – візуалізація загальної структури сирцевого коду програмної системи. Оскільки дана система складатиметься із різних пристроїв для клієнтів та сервера, опишемо загальну структуру мобільного додатку (рис. 2.14).

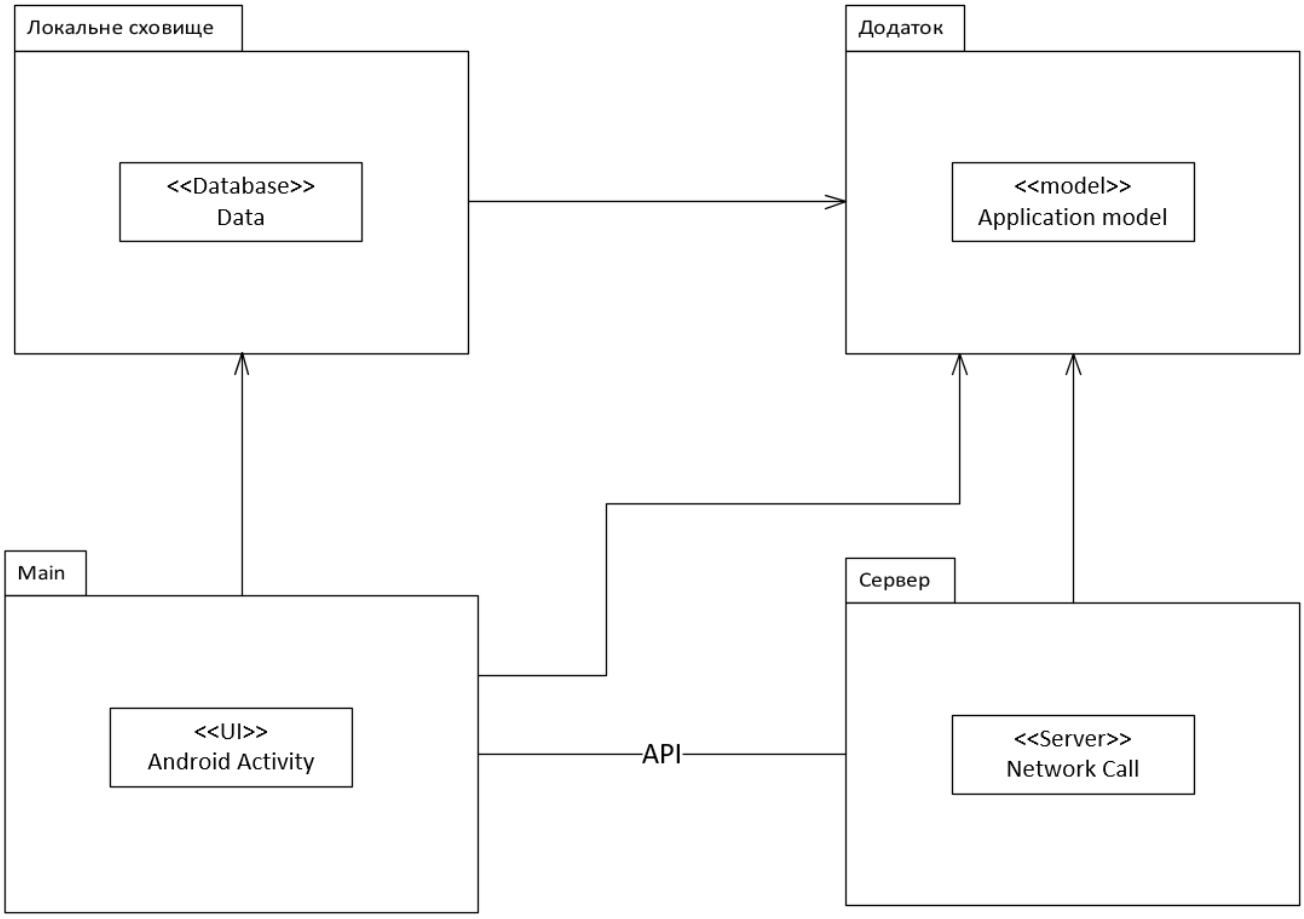


Рисунок 2.14 – UML-діаграма компонентів

Діаграма розгортання – діаграма, на якій відображаються обчислювальні вузли під час роботи програми, компоненти та об'єкти, що виконуються на цих вузлах. Компоненти відповідають представленню робочих екземплярів одиниць коду. Компоненти, що не мають представлення під час роботи програми на таких діаграмах не відображаються; натомість, їх можна відобразити на діаграмах компонентів. Діаграма розгортання відображає робочі екземпляри компонентів, а діаграма компонентів, натомість, відображає зв'язки між типами компонентів.

Діаграма розгортання в UML моделює фізичне розгортання артефактів на вузлах.

Вузли представляються, як прямокутні паралелепіпеди з артефактами, розташованими в них, зображеними у вигляді прямокутників. Вузли можуть мати підвузли, які представляються, як вкладені прямокутні паралелепіпеди. Один вузол діаграми розгортання може концептуально представляти безліч фізичних вузлів, таких, як кластер серверів баз даних.

Існує два типи вузлів:

1) вузол пристрою;

2) вузол середовища виконання.

Вузли пристроїв – це фізичні обчислювальні ресурси зі своєю пам'яттю і сервісами для виконання програмного забезпечення, такі як звичайні ПК, мобільні телефони.

Вузол середовища виконання – це програмний обчислювальний ресурс, який працює всередині зовнішнього вузла і який надає собою сервіс, що виконує інші виконувані програмні елементи.

Діаграму розгортання системи (рис. 2.15) можна описати наступним чином:

1) користувач взаємодіє із системою шляхом використання користувацького клієнта;

2) клієнт відправляє запити та отримує відповіді від web-сервера;

3) сервер надає користувацький інтерфейс, а також використовує інтерфейс бази даних для доступу до даних, їх зміни, видалення, модифікації.

Оскільки дані користувача додатку будуть зберігатись в базі реляційній базі даних, можна на основі вже описаної діаграми класів визначити сутності, які будуть зберігатись в цій базі даних. Як ми вже знаємо, у нас присутні сутності Project, Task, User. Тепер побудуємо інфологічну модель бази даних у вигляді ER-діаграми в нотації Crow’s Foot (рис. 2.16).

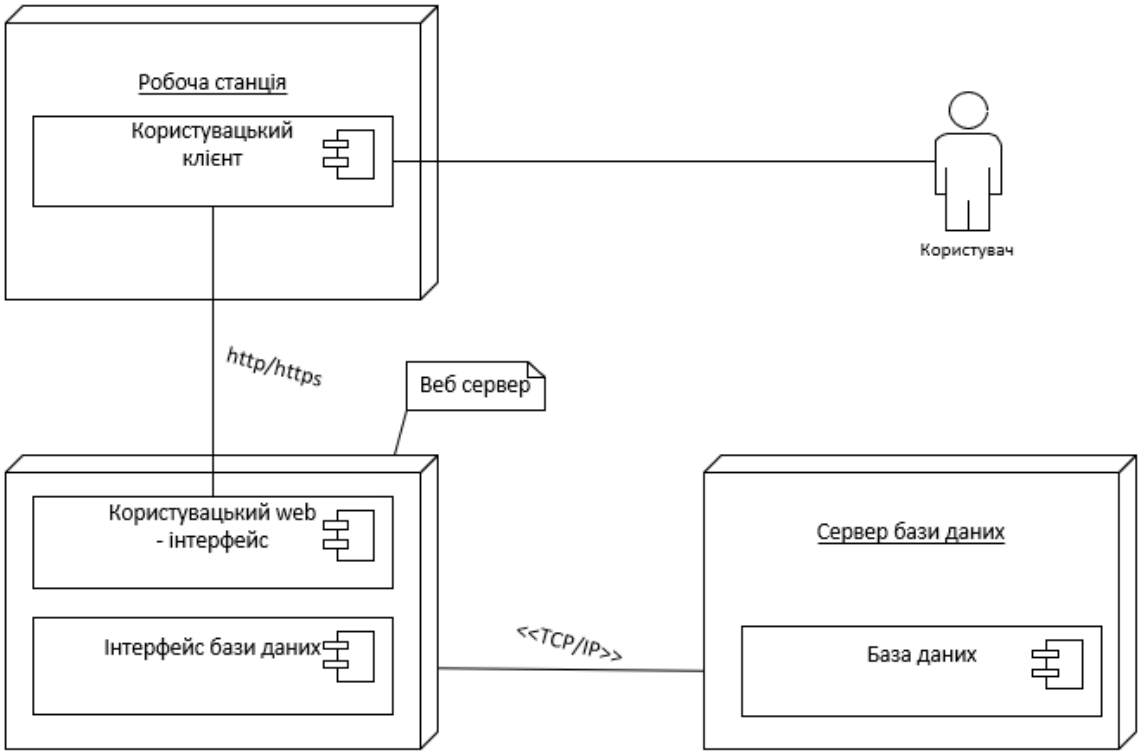


Рисунок 2.15 – UML-діаграма розгортання

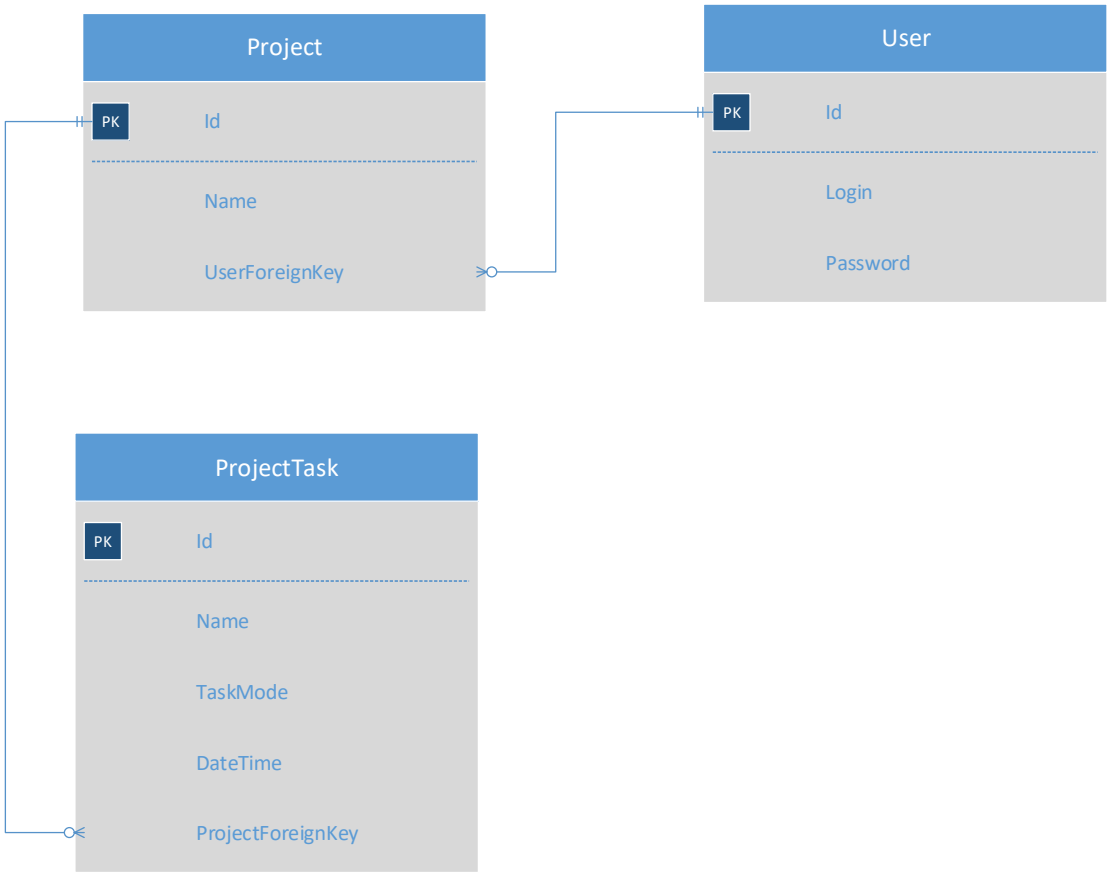


Рисунок 2.16 – Інфологічна модель бази даних

Доповнимо цю діаграму, вказавши типи полів даних для кожної сутності і таким чином одержимо фізичну модель бази даних (рис. 2.17). Побудова фізичної моделі значною мірою є залежною від технологій, які будуть використовуватись в ході розробки самої бази даних. Тож враховуємо, що база даних буде імовірно побудована на системі управління базами даних MS SQL Server, з використанням міграцій CodeFirst через Entity Framework з моделі на мові C#, тому, оскільки визначення типів даних лягає на плечі самого EntityFramework, то типи даних на фізичній моделі визначаються мовою C#.

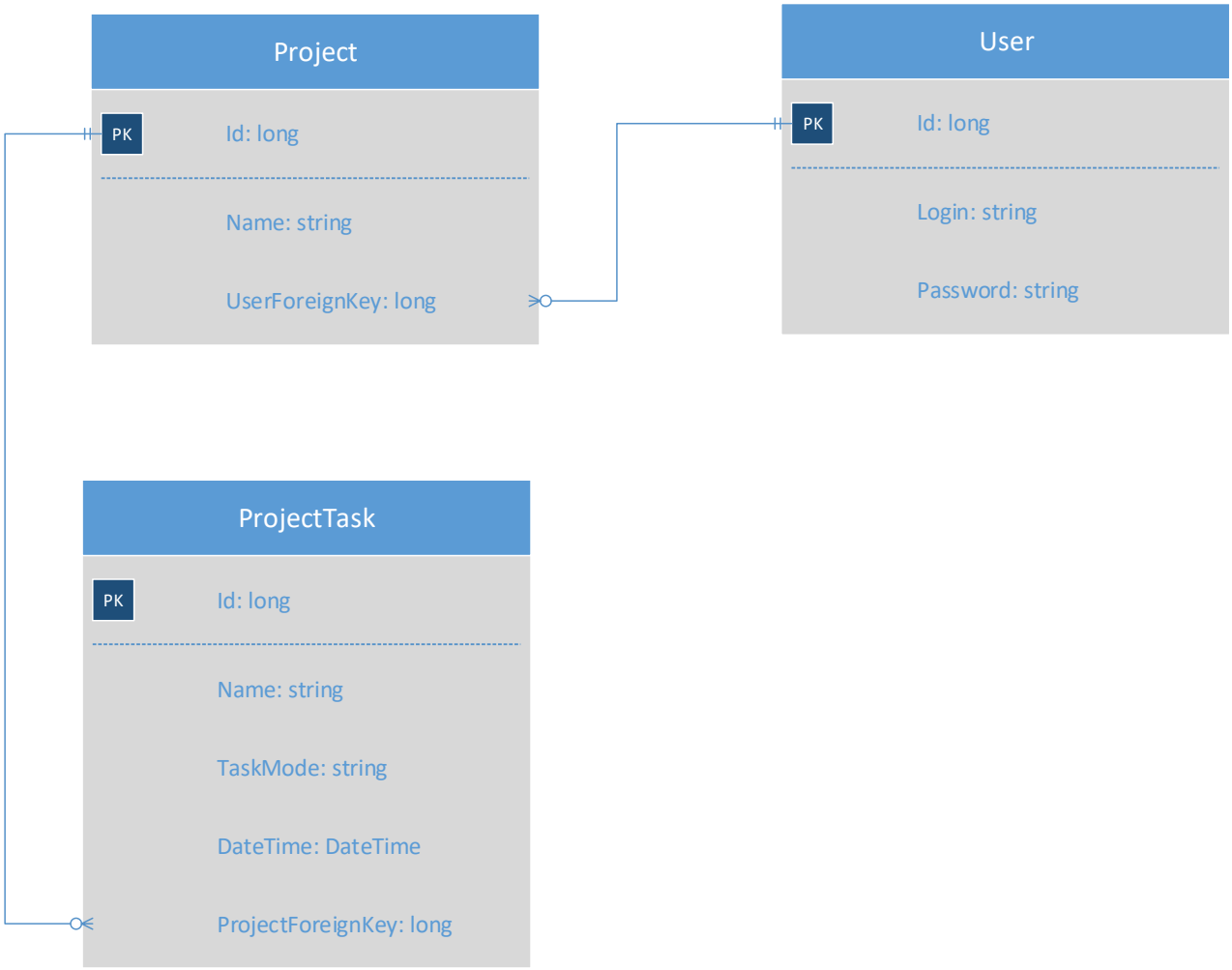


Рисунок 2.17 – Фізична модель бази даних

2.4 Висновки

В другому розділі кваліфікаційної роботи спроектовано архітектуру інформаційної системи організації повсякденної діяльності користувача. Спроектована інформаційна система відповідає за організацію повсякденної діяльності користувача незалежно від її специфіки. Проєктування такого органайзера базуються на принципі розділення його функціональності на діяльності (проєкти) та задачі в цих діяльностях, стан яких користувач може визначати і переглядати. Інтерфейс буде простим в розумінні і не буде обтяжуватись специфічними можливостями, в той же час програма повинна бути різнобічна і мати клієнт-серверну архітектуру, щоб користуватись нею могли клієнти різного типу.

Отже, у другому розділі кваліфікаційної роботи:

1) розроблено графік виконання проектних робіт;
2) проаналізовано ризики проекту;
3) сформовано вимоги до інформаційної системи;
4) розроблено автоматну модель інформаційної системи;
5) розроблено UML-діаграми інформаційної системи.

3 РЕАЛІЗАЦІЯ ТА ФУНКЦІОНУВАННЯ ІНФОРМАЦІЙНОЇ СИСТЕМИ ОРГАНІЗАЦІЇ ПОВСЯКДЕННОЇ ДІЯЛЬНОСТІ КОРИСТУВАЧА

3.1 Реалізація інформаційної системи організації повсякденної діяльності користувача

Створимо проект з розроблення програмного забезпечення (рис. 3.1-3.4).

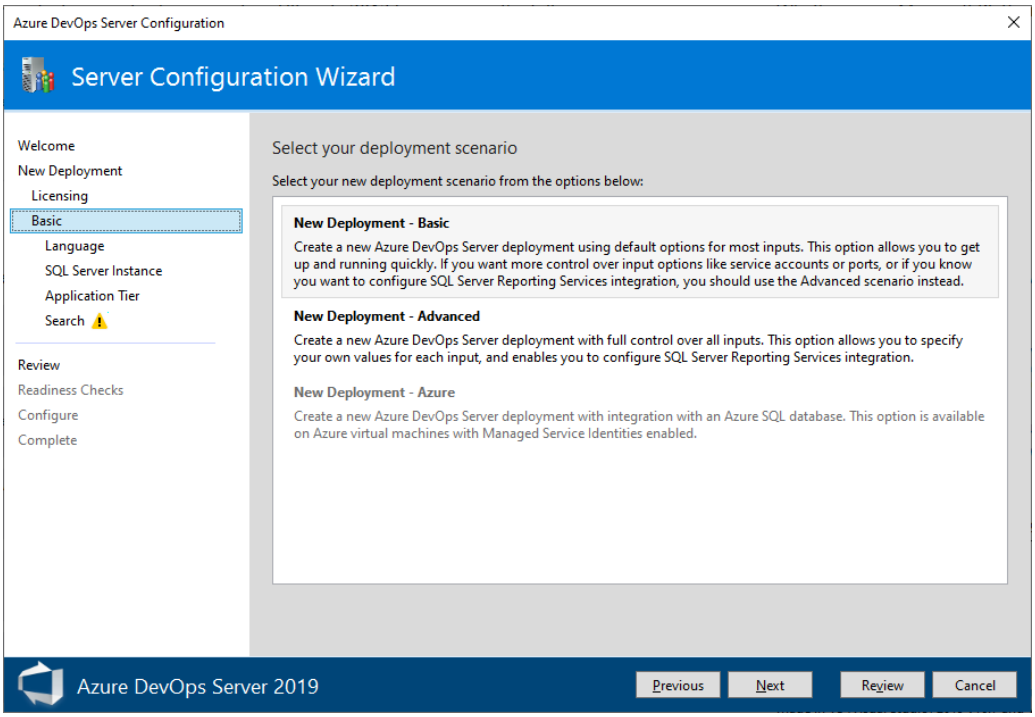


Рисунок 3.1 – Майстер налаштування сервера

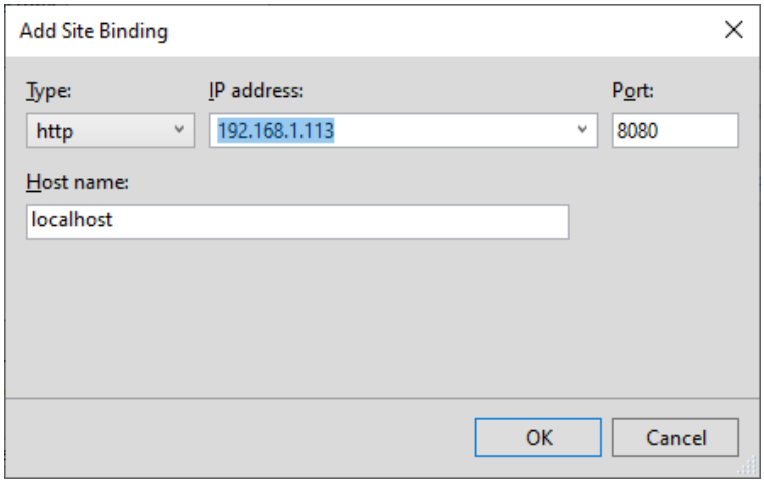


Рисунок 3.2 – Налаштування конфігурації сервера

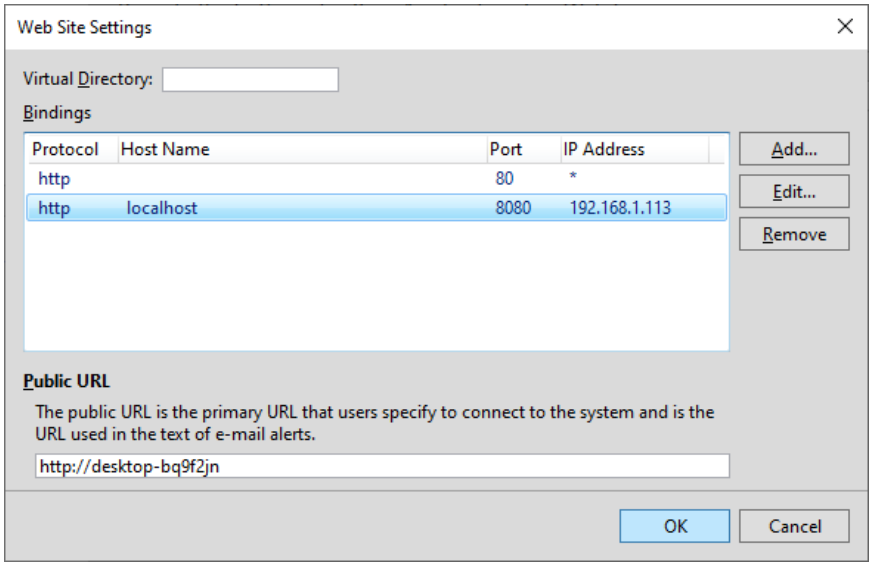


Рисунок 3.3 – Налаштування конфігурації додатку

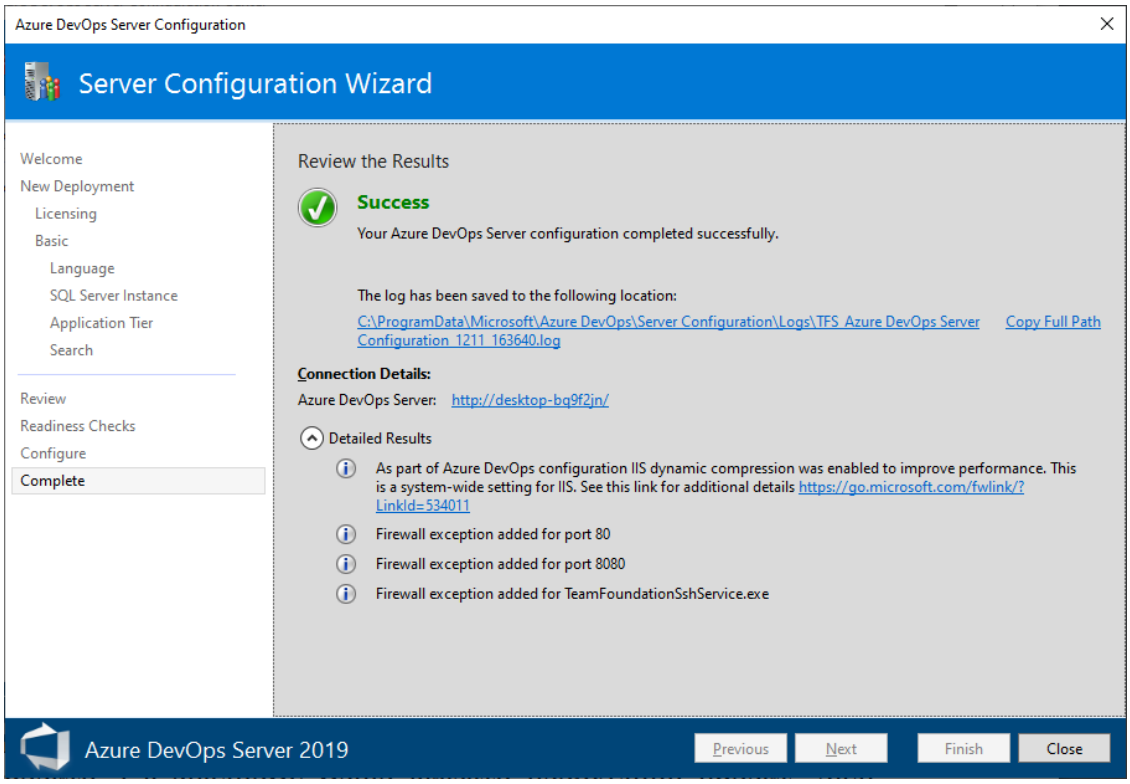


Рисунок 3.4 – Завершення налаштування сервера

Налаштуємо області ітерації, параметри команди, сповіщення та доступ до параметрів проекту.

Виділимо опції:

1) синхронізація з сервером;

2) реєстрація;
3) авторизація;
4) додавання проектів;
5) додавання задач;
6) свайп виконаних задач;
7) журнал дій;
8) редагування та видалення задач;
9) редагування та видалення проектів.

Пріоритет опцій:

1) синхронізація з сервером – 10;
2) реєстрація – 12;
3) авторизація – 13;
4) додавання проектів – 25;
5) додавання задач – 27;
6) свайп виконаних задач – 29;
7) журнал дій – 35;
8) редагування та видалення задач – 32;
9) редагування та видалення проектів – 30.

Ітерації:

1) каркас:
    a) синхронізація з сервером;
    b) реєстрація;
    c) авторизація;
2) базовий функціонал:
    d) додавання проектів;
    e) додавання задач;
    f) свайп виконаних задач;
3) розширений функціонал:
    g) журнал дій;

h) редагування та видалення задач;

i) редагування та видалення проектів.

Зареєструємо ім'я та опис проекту (рис. 3.5).

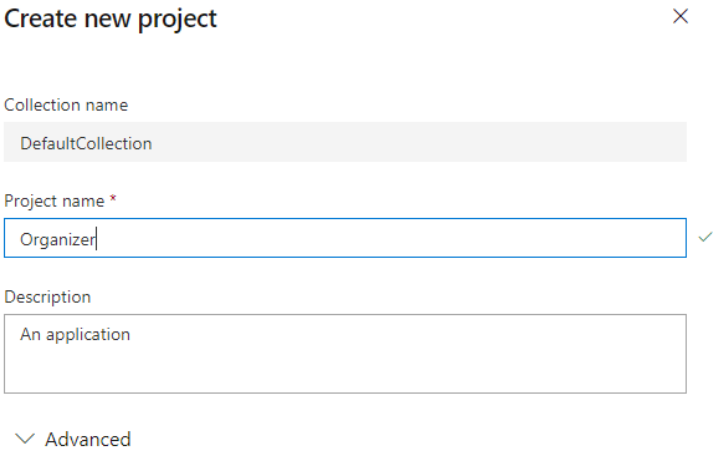


Рисунок 3.5 – Ім'я та опис проекту

Для успішної роботи клієнтських додатків потрібна серверна функціональність, яка і буде визначати бізнес-логіку додатку та принципи клієнт-серверної взаємодії. Для цієї задачі буде використано технологію розробки Web Api на фреймворку ASP.Net Core. Мова програмування – C#.

Тепер перейдемо до вибору паттерну проектування рішення. Принцип побудови клієнт-серверної архітектури буде обгрунтований паттерном MVC (Model, View, Controller). Причому роль Model та Controller будуть описані як елементи серверного рішення, а роль View лягатиме на клієнтські додатки.

Створимо область Model в Visual Studio з усіма необхідними класами (рис. 3.6).

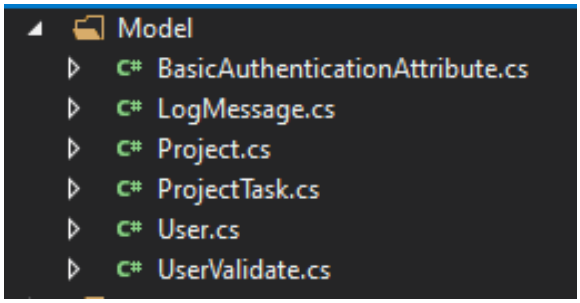


Рисунок 3.6 – Класи Model

Головне, на що треба звернути увагу, – це реалізація класів, власне, моделі, які являють собою дані, які будуть зберігатись в базі даних і якими сервер буде обмінюватись з клієнтами, а саме – Project, ProjectTask, User. Розглянемо їх реалізацію на прикладі Project (рис. 3.7). Цей клас має помічений атрибутом «[Key]» первинний ключ, а також помічений як «[ForeignKey(“UserForeignKey”)]» зовнішній ключ, що посилається на користувача, і відповідний екземпляр класу User. А функція CleanData дозволить класу створити копію себе в чистому вигляді і без циклів, спричинених використанням екземпляру класу User, щоб убезпечитись від помилок при серіалізації об’єкта під час його відправки в якості відповіді з серверу.

```
public class Project
{
    [Key]
    Ссылок: 4
    public long Id { get; set; }

    Ссылок: 9
    public string Name { get; set; }

    Ссылок: 13
    public long UserForeignKey { get; set; }

    [ForeignKey("UserForeignKey")]
    Ссылок: 0
    public User User { get; set; }

    Ссылок: 0
    public List<ProjectTask> Tasks { get; set; }


    Ссылок: 5
    public Project CleanData()
    {
        return new Project() { Id = this.Id, Name = this.Name, UserForeignKey = this.UserForeignKey };
    }
}
```

Рисунок 3.7 – Клас Project

Подібним чином розроблені і інші класи. Такими ж класами оперуватимуть і клієнтські додатки, що дозволить підтримувати двосторонню серіалізацію/десеріалізацію об’єктів даних в форматі JSON, які зможе зчитувати як клієнт, так і сервер, наприклад:

```
{
    "id": 83,
    "name": "Спорт",
    "userForeignKey": 2
}
```

Розроблена модель може бути використана для формування бази даних, і легкого доступу до неї без жодного запиту SQL. Це можливо завдяки ORM-технології Entity Framework. Варто лише вказати рядок підключення в конфігурації проекту, створити клас контексту (рис. 3.8), і провести міграцію моделі в базу даних командами Add-Migration та Update-Database.

```
public class EFOrganizerAPIDBContext:DbContext
{
    public static EFOrganizerAPIDBContext instance;
    Ссылок: 0
    public EFOrganizerAPIDBContext(DbContextOptions<EFOrganizerAPIDBContext> options) : base(options)
    {
        instance = this;
    }

    Ссылок: 6
    public DbSet<User> Users { get; set; }
    Ссылок: 12
    public DbSet<Project> Projects { get; set; }
    Ссылок: 0
    public DbSet<LogMessage> LogMessages { get; set; }
    Ссылок: 7
    public DbSet<ProjectTask> Tasks { get; set; }

}
```

Рисунок 3.8 – Клас контексту бази даних

Наступний елемент в паттерні MVC – контролери (рис. 3.9).

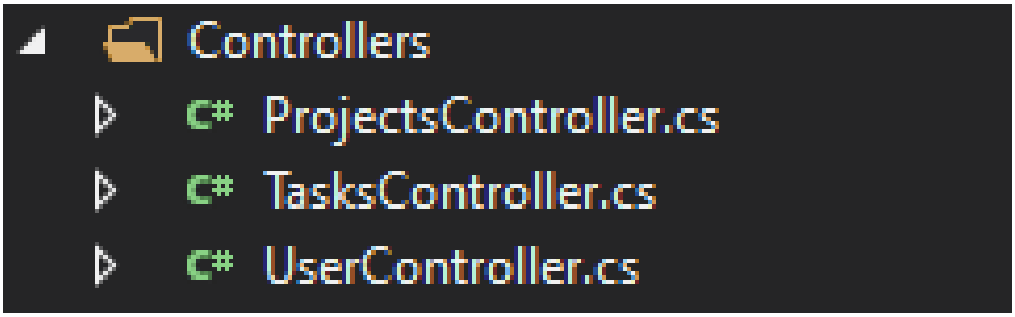


Рисунок 3.9 – Область контролерів

Для кожного елемента моделі був створений свій контролер. Контролер відповідає за бізнес-логіку та реалізовує запити до сервера, які робляться клієнтами. Кожен запит являє собою метод контролера з визначеним типом HTTP-запиту, до якого є свій URL-шлях, вхідні дані, та повернення певного результату (рис. 3.10).

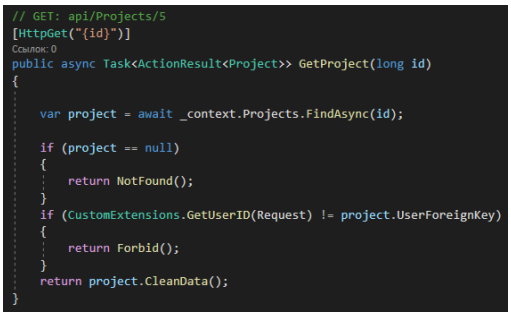


Рисунок 3.10 – Приклад реалізації метода контролера.

Наше програмне забезпечення має такі функції – табл. 3.1.

Таблиця 3.1 – Опис основних функцій програмного забезпечення

№ п/п	Назва функції	Опис функції
1	Get User	Отримання користувача
2	Register User	Зареєструвати користувача
3	Get Projects	Отримання проектів
4	Get Project	Отримання проекту
5	Get Project Tasks	Отримання задач
6	Create Project	Створення проекту
7	Change Project	Змінити налаштування проекту
8	Delete Project	Видалити проект
9	Get All Tasks	Отримання всіх задач
10	Create Task	Створення задачі
11	Change Task	Змінення налаштування задачі
12	Delete Task	Видалення задачі

Інтерфейсні вікна ПЗ: сторінка реєстрації, сторінка дошок, сторінка задач.

Переваги розробленої інформаційної системи організації повсякденної діяльності користувача:

1) система зручна у використані;
2) можливість зробити багато профілів під різні задачі;
3) можливість зробити багато проектів під різні задачі;
4) інтуїтивна зрозумілість.

Недоліки розробленої інформаційної системи організації повсякденної діяльності користувача:

1) складність розробки;
2) значні витрати на підтримку.

3.2 Функціонування та тестування інформаційної системи організації повсякденної діяльності користувача

Розглянемо функціонування розробленого мобільного додатка. Спробуємо створити новий проект за допомогою графічного інтерфейсу. Для цього натиснемо на кнопку «Новий проект» і отримаємо новостворений проект (рис. 3.11).

Як бачимо, після створення проект відобразився у списку проектів користувача в нижній частині екрану. Список робіт завантажується із віддаленого сервера, а отже проблем зі збереженням не було. Далі спробуємо створити задачу у даному проекті та вказати їй час виконання. Для цього оберемо необхідний проект та натиснемо кнопку «Нова задача» і отримаємо новостворену задачу (рис. 3.12).

Інтерфейс пропонує можливість редагування, додавання та видалення елементів. Також можна позначити задачу виконаною – протестуємо це натиснувши на відповідну кнопку і отримаємо виконану задачу (рис. 3.13).

На даному етапі функції розробленого мобільного додатку працюють адекватно.

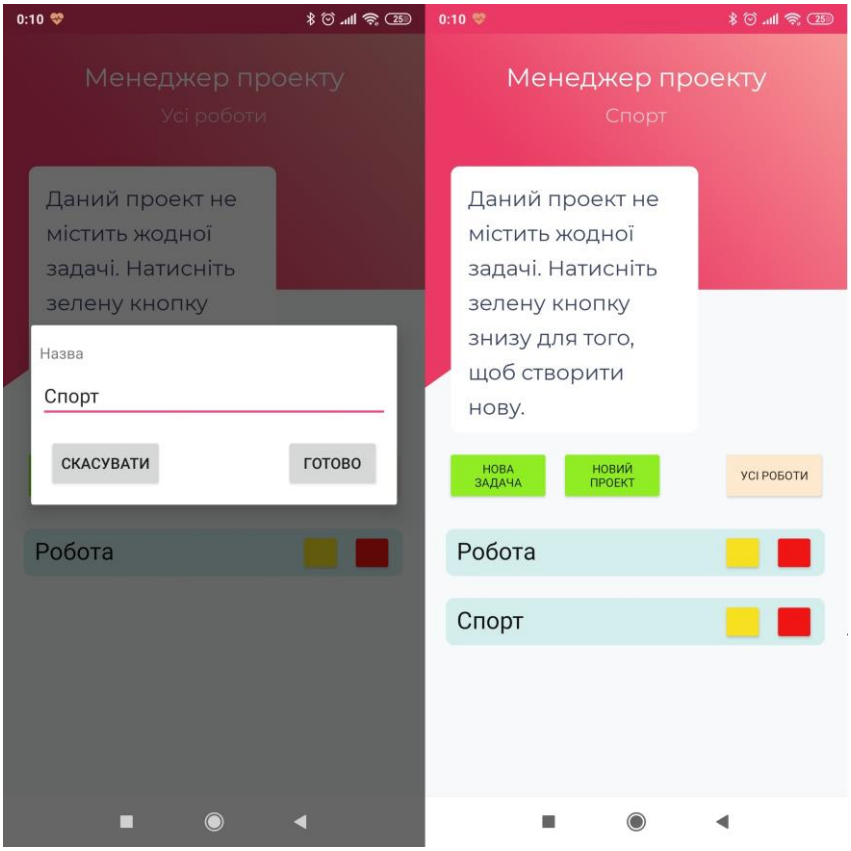


Рисунок 3.11 – Результат створення нового проекту

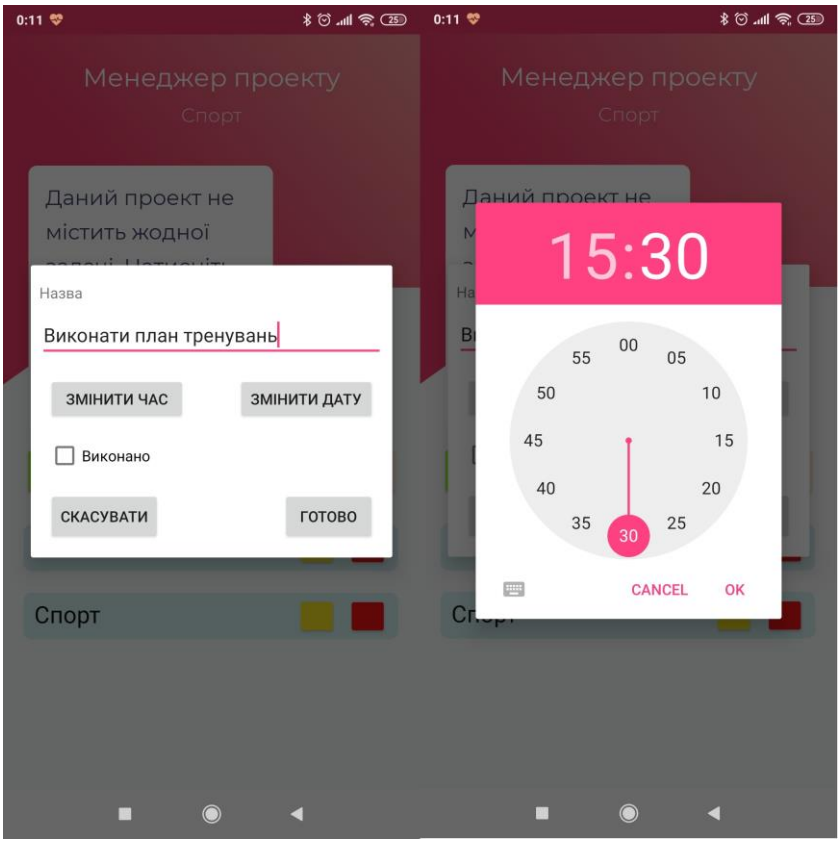


Рисунок 3.12 – Результат створення нової задачі

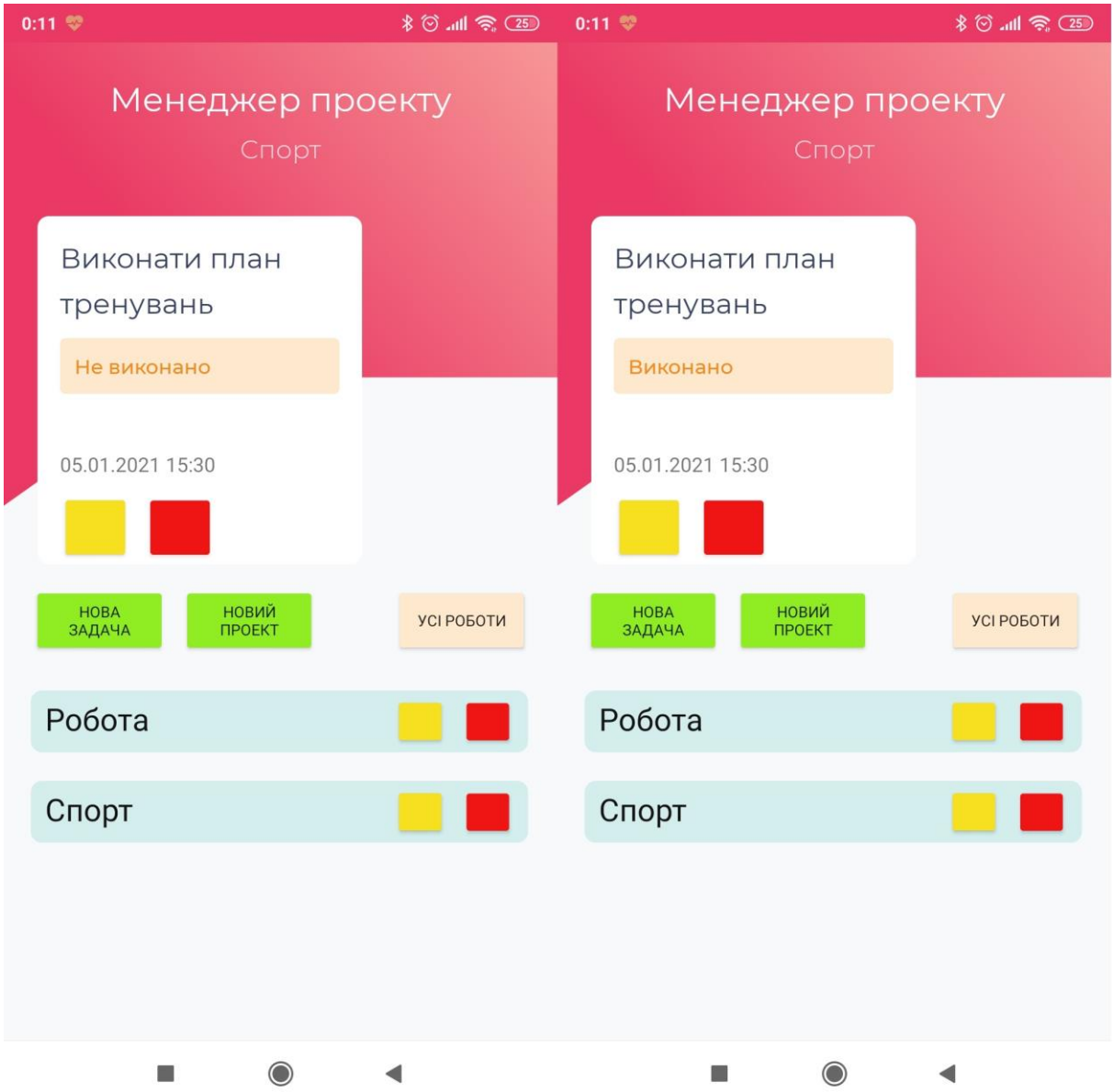


Рисунок 3.13 – Визначення статусу виконання

Найслабшим місцем розробленого програмного забезпечення є серверне рішення додатку, оскільки саме воно працює з бізнес-логікою та логікою зберігання даних в проекті. Оскільки серверне рішення розроблено в середовищі IDE Visual Studio, ми одержуємо потужні і вже вбудовані інструменти як для динамічного, так і для статичного аналізу коду та тестування. Окрім вбудованого .NET Compiler Platform (Roslyn) ми за допомогою Nuget встановимо також SonarAnalyzer.CSharp, який є чудовим інструментом статичного аналізу та пошуку помилок і вразливостей програмного коду. Усі повідомлення аналізаторів мають

різний формат та ступінь серйозності. Подвійний клік по них дозволяє одразу ж перейти на місце повідомлення. Наприклад, аналізатор може запропонувати зробити поле доступним тільки для читання, якщо це поле не змінюється поза конструктором, і це буде повідомлення низького ступеню серйозності, або ж може запропонувати об’єднати дві if-конструкції, якщо одна з них знаходиться всередині другої і не пропонує else-сценарію (рис. 3.14).

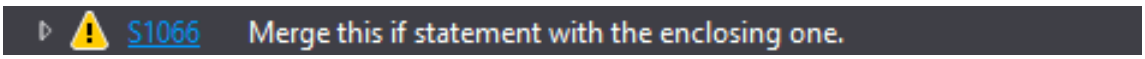


Рисунок 3.14 – Приклад повідомлення вищого ступеню серйозності

Загалом, Visual Studio дозволяє комбіноване використання великої кількості аналізаторів як стилю коду, так і його безпечності, але нам достатньо вбудованого Roslyn та SonarAnalyzer.CSharp.

Для тестування серверного додатку можна використати також вбудовані Unit-тести Visual Studio. Вони дозволять скористатись кодовою базою додатку, і проводити з нею дії, перевіряючи код на правильність виконання з допомогою коду. Для функціонального тестування вже самих запитів скористаємось програмою Postman, яка надає широкий функціонал для виконання будь-яких запитів.

Проведемо функціональне тестування серверного API додатка. Для цього скористаємось програмою Postman. Зареєструємо користувача (рис. 3.15).

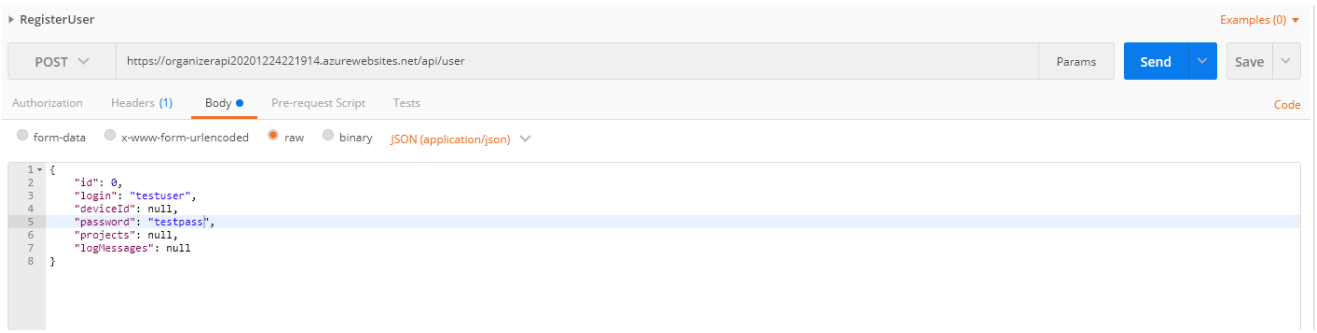


Рисунок 3.15 – Реєстрація користувача в запиті

Спробуємо додати проект – рис. 3.16.

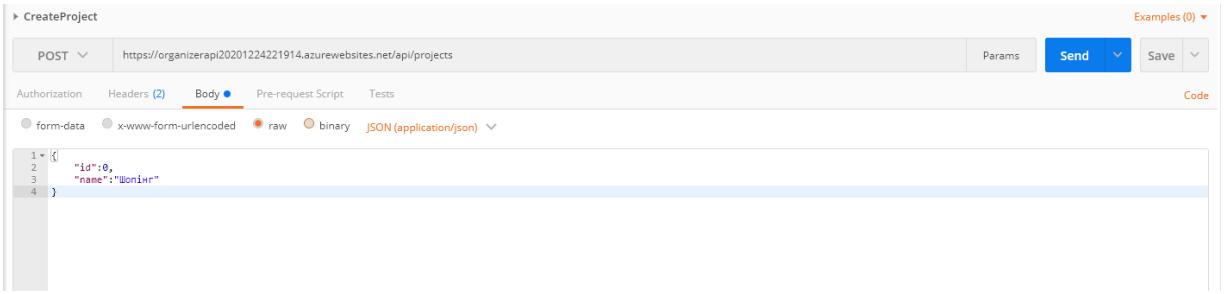


Рисунок 3.16 – Додавання проекту

Одержуємо успішну відповідь. Спробувавши знову додати цей проект, одержимо помилку з повідомленням про існування такого – рис. 3.17.

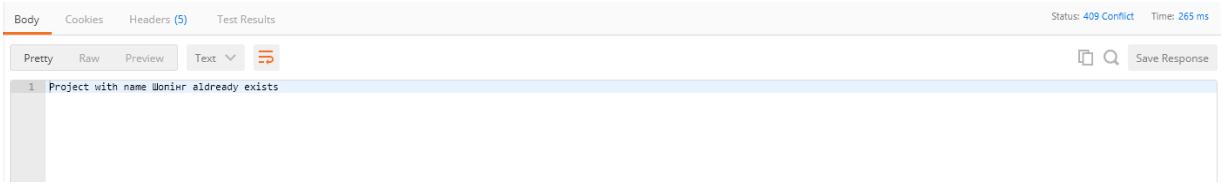


Рисунок 3.17 – Повідомлення про конфлікт назви проекту

Таким чином, ми змогли успішно протестувати функціональність додавання проекту.

Unit-тестування також проведено успішно (рис. 3.18).

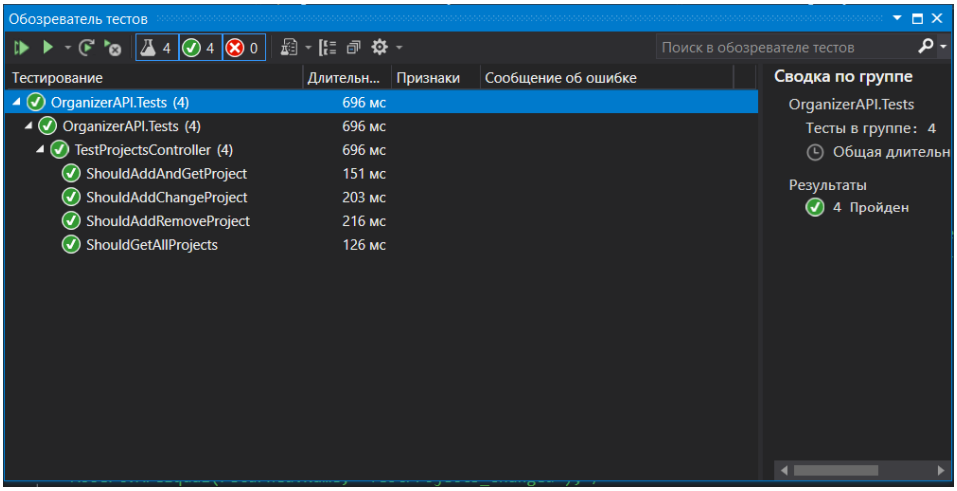


Рисунок 3.18 – Результати Unit-тестування

Ми використовуємо тут url для api, опублікованого в хмарі, і, оскільки ми одержуємо адекватні відповіді на запити замість 404, можна сказати, що і база даних, і всі запити до віддаленого API протестовані успішно. Системне ж тестування є турботою власне сервісу хмари, а точніше – її власника Microsoft.

3.3 Висновки

В третьому розділі кваліфікаційної роботи виконано реалізацію та тестування роботи інформаційної системи організації повсякденної діяльності користувача, а саме: описано реалізацію інформаційної системи організації повсякденної діяльності користувача; представлено приклади роботи мобільної версії інформаційної системи організації повсякденної діяльності користувача; проаналізовано результати тестування клієнтської та серверної частини інформаційної системи організації повсякденної діяльності користувача.

ВИСНОВКИ

Попереднє планування справ допомагає підвищити ефективність будь-якої діяльності – як особистої, так і професійної.

Інформаційні системи організації повсякденної діяльності користувача (органайзери) дозволяють проводити планування особистого часу користувача, своєчасно нагадують про початок запланованих заходів та зустрічей, про необхідність виконання тих чи інших нагальних справ, ведуть персональні та інші картки з можливістю автоматичної вибірки інформації, ведуть персональні інформаційні записники для збереження різноманітної особистої інформації. Такі системи призначені для накопичення інформації користувача, а потім для організації справ і контролю за їх виконанням, відслідковування визначених користувачем подій.

В результаті виконання кваліфікаційної роботи було розроблено інформаційну систему організації повсякденної діяльності користувача (органайзер), яка відповідає за організацію повсякденної діяльності незалежно від її специфіки. Проєктування та розроблення такого органайзера базуються на принципі розділення його функціональності на діяльності (проєкти) та задачі в цих діяльностях, стан яких користувач буде визначати і переглядати. Інтерфейс є простим в розумінні і не обтяжується специфічними можливостями, в той же час програма є різнобічною і має клієнт-серверну архітектуру.

Об’єктом дослідження є процес організації повсякденної діяльності користувача.

Предметом дослідження є інформаційна система організації повсякденної діяльності користувача (органайзер).

Для досягнення поставленої мети використовуються такі методи дослідження, як методи синтезу, аналізу та моделювання процесів, принципи системного аналізу, теоретико-множинні підходи.

В першому розділі кваліфікаційної роботи виконано дослідження предметної області. Крім цього, в першому розділі виконано постановку задачі подальшого дослідження.

В другому розділі кваліфікаційної роботи спроектовано архітектуру інформаційної системи організації повсякденної діяльності користувача: розроблено графік виконання проектних робіт; проаналізовано ризики проекту; сформовано вимоги до інформаційної системи; розроблено автоматну модель інформаційної системи; розроблено UML-діаграми інформаційної системи.

В третьому розділі кваліфікаційної роботи виконано реалізацію та тестування роботи інформаційної системи організації повсякденної діяльності користувача, а саме: описано реалізацію інформаційної системи організації повсякденної діяльності користувача; представлено приклади роботи мобільної версії інформаційної системи організації повсякденної діяльності користувача; проаналізовано результати тестування клієнтської та серверної частини інформаційної системи організації повсякденної діяльності користувача.

Практичне значення має спроєктована та реалізована інформаційна система організації повсякденної діяльності користувача (органайзер), яка відповідає за організацію повсякденної діяльності незалежно від її специфіки.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАНЬ

1. Говорущенко Т. О. Аналіз, дослідження та оцінювання програмних систем: навчальний посібник. Хмельницький: Хмельницький національний університет, 2019. 358 с.
2. Бевз О. М., Папінов В.М., Скидан Ю.А. Проєктування програмних засобів систем управління: навчальний посібник. Вінниця: ВНТУ, 2010. 125 с.
3. McConnell S. Software Estimation: Demystifying the Black Art (Developer Best Practices). Microsoft Press, 2016. 308 p.
4. Sommerville I. Software Engineering. London: Pearson, 2015. 816 p.
5. Munch J., Schmid K. Perspectives on the future of software engineering. Springer-Verlag Berlin Heidelberg, 2013. 365 p.
6. Diaz V. G., Lovelle J. M., Garcia-Bustelo B. C. Handbook of research on innovations in systems and software engineering. Hershey, 2015. 745 p.
7. Levenson N. G. Engineering a safer world: systems thinking applied to safety. MIT Press, 2012. 560 p.
8. Abran A. Software project estimation: the fundamentals for providing high quality information to decision makers. Wiley-IEEE Computer Society Press, 2015. 288 p.
9. Wiegers K., Beatty J. Software requirements: 3rd edition. Washington: MS Press, 2013. 640 p.
10. Chen A., Beatty J. Visual models for software requirements. Washington: MS Press, 2012. 444 p.
11. Грицунов О.В. Інформаційні системи та технології: навч. посіб. для студентів. Харків: Харківська національна академія міського господарства, 2010. 222 с.
12. Плескач В.Л., Затонацька Т.Г. Інформаційні системи і технології на підприємствах.: Підручник. Київ: Знання, 2011. 718 с.
13. Павлиш В. А., Гліненко Л. К., Шаховська Н. Б. Основи інформаційних технологій і систем: підручник. Львів : Видавництво Львівської політехніки, 2018. 620 с.

14. Галич О. А., Копішинська О. П., Уткін Ю. В. Управління інформаційними зв’язками та бізнес-процесами: навчальний посібник. Харків: Фінарт, 2016. 244 с.

15. Карімов Г. І., Карімов І. К. Інформаційні системи і технології в управлінні організаціями. Дніпродзержинськ: ДДТУ, 2014. 141 с.

16. Impacts of task interdependence and equivocality on ICT adoption in the construction industry: a task-technology fit view / Y. Hua et al. *Architectural Engineering and Design Management*. 2021. P. 1–18.

17. Lennon N. J. Balancing incremental and radical innovation through performance measurement and incentivization. *The Journal of High Technology Management Research*. 2022. Vol. 33, no. 2. P. 100439.

18. Saeed S. Digital Workplaces and Information Security Behavior of Business Employees: An Empirical Study of Saudi Arabia. *Sustainability*. 2023. Vol. 15, no. 7. P. 6019.

19. Lyons P. Workplace engagement interventions: empirically based alternatives for manager consideration. *Journal of Workplace Learning*. 2022.

20. Hanafizadeh P., Taherianfar A., Alami Neisi M. Individual Adaptation in the Face of Enterprise IT Changes in the Organization. *Journal of Telecommunications and the Digital Economy*. 2023. Vol. 11, no. 1. P. 57–90.

21. Pan L., Abdullah H. Multi-dimensional Data Perception and Intelligent Analysis Framework Design of Management Information System. *Lecture Notes on Data Engineering and Communications Technologies*. Cham, 2023. P. 36–44.

22. Falk M. T., Hagsten E. Reverse adoption of information and communication technology among organisers of academic conferences. *Scientometrics*. 2023.

23. Open Science Training in TRIPLE / L. Provost et al. *Open Research Europe*. 2023. Vol. 3. P. 39.

24. Seale D. E., LeRouge C. M., Kolotylo-Kulkarni M. Professional Organizers’ Description of Personal Health Information Management Work with a Spotlight on the Practices of Older Adults: A Qualitative e-Delphi Study (Preprint). *Journal of Medical Internet Research*. 2022.

25. Participant-centred planning Framework for effective gender balance activities in tech / E. Taylor-Smith et al. *UKICER2022: The United Kingdom and Ireland Computing Education Research Conference*, Dublin Ireland. New York, NY, USA, 2022.

26. Gray P. C. “The Same Tools Work Everywhere”. *Labour / Le Travail*. 2022. Vol. 90. P. 41–84.

27. Chatzis D., Papasalouros A., Kavallieratou E. Planning a robotic competition. *Computer Applications in Engineering Education*. 2022.

28. Iio J. EPOQAS: Development of an Event Program Organizer with a Question Answering System. *Advances in Networked-Based Information Systems*. Cham, 2021. P. 341–348.

29. Qi W., Jiang Y. Use of a Graphic Organiser as a Pedagogical Instrument for the Sustainable Development of EFL Learners’ English Reading Comprehension. *Sustainability*. 2021. Vol. 13, no. 24. P. 13748.

30. Ha Le V. Vietnamese Students' Perspectives towards Using Graphic Organizers on Reading Comprehension Skills. *ICAAI 2021: 2021 the 5th International Conference on Advances in Artificial Intelligence*, Virtual Event United Kingdom. New York, NY, USA, 2021.

31. Applying Motivational Techniques for User Adherence to adopt a Healthy Lifestyle in a Gamified Application / S. Fatima et al. *Entertainment Computing*. 2023. P. 100571.

32. Identification of everyday food-related activities with potential for direct and indirect energy savings: KTH Live–in–Lab explorative case study / E. Malakhatka et al. *Energy Policy*. 2022. Vol. 163. P. 112792.

33. Factors influencing sustainability of online platforms for professionals: a mixed-method study in OECD countries / K. M. H. Hubertus Bessems et al. *Health Promotion International*. 2021.

34. Harun S., Dorasamy M., Ahmad A. A. Effect of ERP Implementation on Organisational Performance: Manager's Dilemma. *International Journal of Technology*. 2022. Vol. 13, no. 5. P. 1064.

35. Farhood H., Saberi M., Najafi M. Human-in-the-Loop Optimization for Artificial Intelligence Algorithms. *Service-Oriented Computing – ICSOC 2021 Workshops*. Cham, 2022. P. 92–102.

36. Smartwatch-Based Face-Touch Prediction Using Deep Representational Learning / H. Rizk et al. *Lecture Notes of the Institute for Computer Sciences, Social Informatics and Telecommunications Engineering*. Cham, 2022. P. 493–499.

37. Smartphone-Based Lifelogging: Toward Realization of Personal Big Data / S. Ali et al. *Information and Knowledge in Internet of Things*. Cham, 2021. P. 249–309.

38. Abdullah H. Towards the development of an information privacy protection awareness inititiative for data subjects and organizations. *2021 National Computing Colleges Conference (NCCC)*, Taif, Saudi Arabia, 27–28 March 2021. 2021.

39. Pandey D., Maitrey S., Seth D. Artificially developed intelligent system using Python. *PROCEEDING OF INTERNATIONAL CONFERENCE ON FRONTIERS OF SCIENCE AND TECHNOLOGY 2021*, Ghaziabad, India. 2022.

40. Walter-Tscharf V. Multi-tenant Cloud SaaS Application for a meeting to task transition via deep learning models. *2022 IEEE Global Conference on Artificial Intelligence and Internet of Things (GCAIoT)*, Alamein New City, Egypt, 18–21 December 2022. 2022.

41. Mand S. K., Sharma N. Authentication tokens based smart backup and recovery system for modern android systems. *2022 International Interdisciplinary Humanitarian Conference for Sustainability (IIHC)*, Bengaluru, India, 18–19 November 2022. 2022.

42. A prototype of smart virtual assistant integrated with automation / H. Mauny et al. *2021 Third International Conference on Inventive Research in Computing Applications (ICIRCA)*, Coimbatore, India, 2–4 September 2021. 2021.

43. Ali I., Warraich N. F. The relationship between mobile self-efficacy and mobile-based personal information management practices. *Library Hi Tech*. 2020. Ahead-of-print, ahead-of-print.

44. Bhaskar L., Ranjith R. Robust Text Extraction in Images for Personal Event Planner. *2020 11th International Conference on Computing, Communication and Networking Technologies (ICCCNT)*, Kharagpur, India, 1–3 July 2020. 2020.

45. Scheduling of events through notifications in mobile devices / F. Z. Coto et al. *2019 IV Jornadas Costarricenses de Investigación en Computación e Informática (JoCICI)*, San Pedro, Costa Rica, 19–20 August 2019. 2019.

46. Atkinson S., Lee K. Design and Implementation of a Study Room Reservation System: Lessons from a Pilot Program Using Google Calendar. *College & Research Libraries*. 2018. Vol. 79, no. 7. P. 916–930.

Додаток А

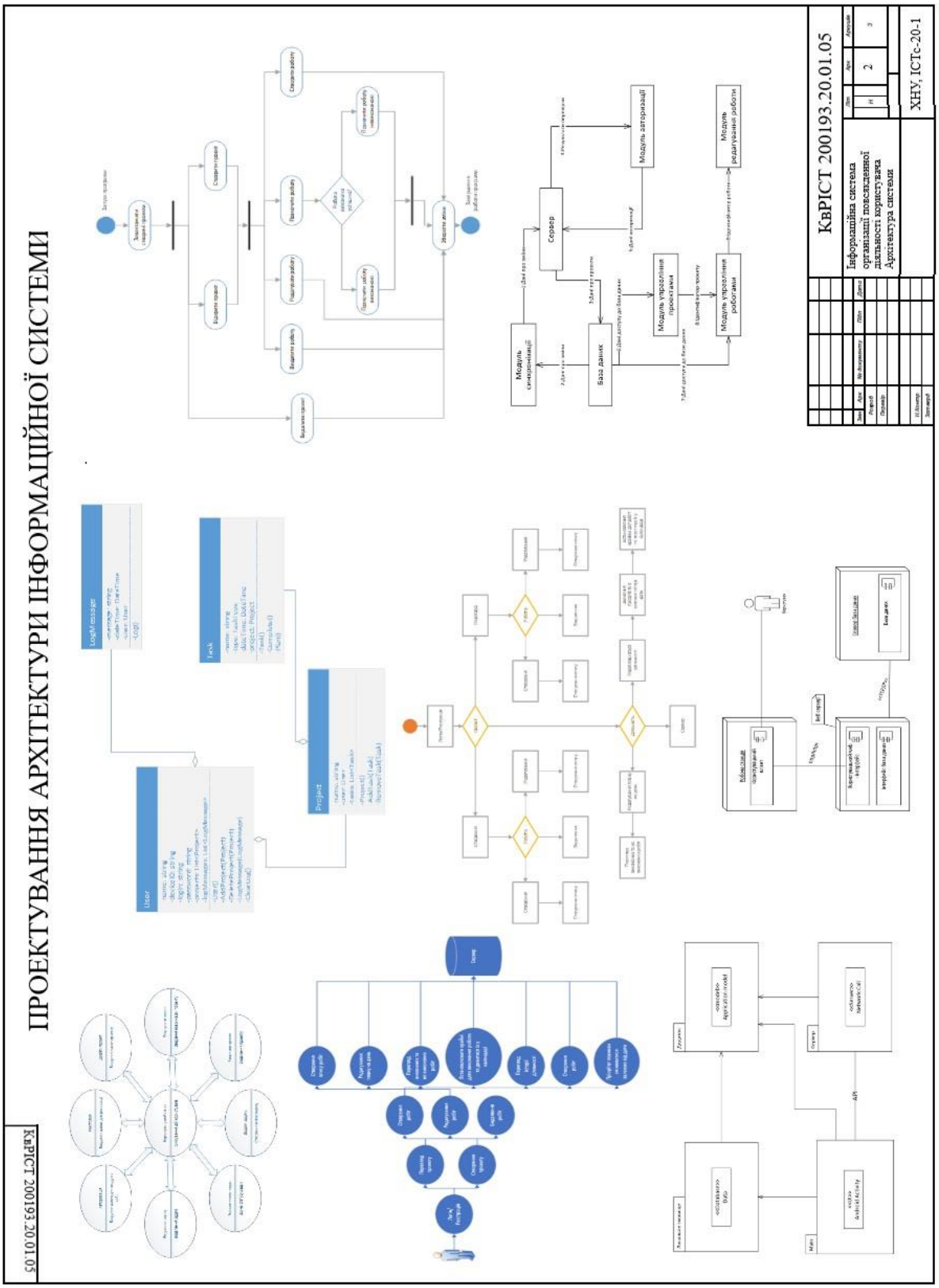
(обов’язковий)

Копія креслення «Планування проектних робіт»

Додаток Б

(обов’язковий)

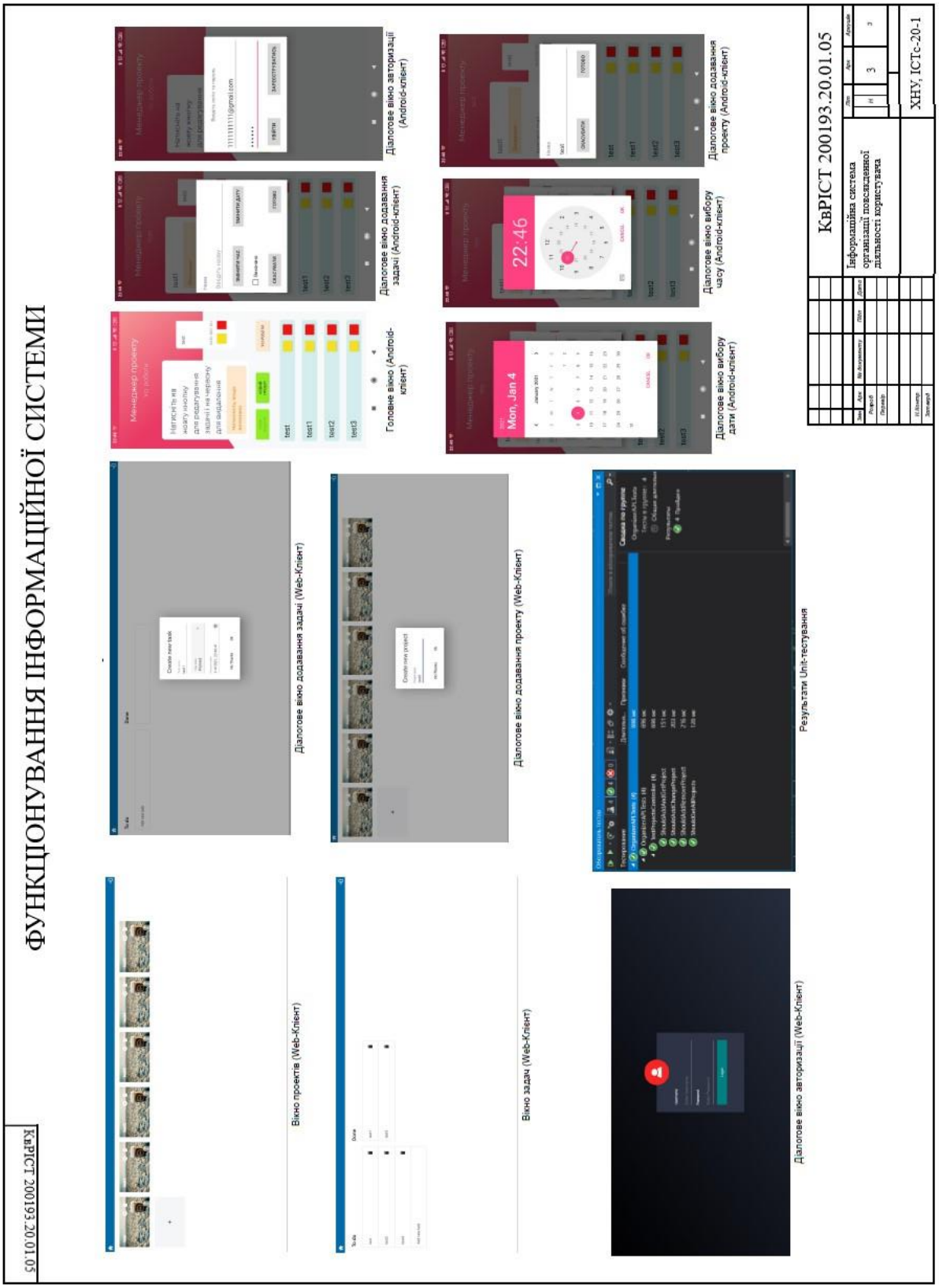
Копія креслення «Проектування архітектури інформаційної системи»



Додаток В

(обов’язковий)

Копія креслення «Функціонування інформаційної системи»



Додаток Г

Лістинг коду

Серверне рішення

Project.cs

```
using System;
using System.Collections.Generic;
using System.ComponentModel.DataAnnotations;
using System.ComponentModel.DataAnnotations.Schema;
using System.Linq;
using System.Threading.Tasks;

namespace OrganizerAPI.Model
{
    public class Project
    {
        [Key]
        public long Id { get; set; }

        public string Name { get; set; }

        public long UserForeignKey { get; set; }

        [ForeignKey("UserForeignKey")]
        public User User { get; set; }

        public List<ProjectTask> Tasks { get; set; }


        public Project CleanData()
        {
            return new Project() { Id = this.Id, Name = this.Name, UserForeignKey = this.UserForeignKey };
        }
    }

}
```

ProjectTask.cs

```
using System;
using System.Collections.Generic;
using System.ComponentModel.DataAnnotations;
using System.ComponentModel.DataAnnotations.Schema;
using System.Linq;
using System.Threading.Tasks;

namespace OrganizerAPI.Model
{
    public class ProjectTask
    {
        [Key]
        public long Id { get; set; }
```

```
        public string Name { get; set; }
        public string TaskMode { get; set; }

        public DateTime DateTime { get; set; }

        public long ProjectForeignKey { get; set; }

        [ForeignKey("ProjectForeignKey")]
        public Project Project { get; set; }

        public ProjectTask CleanData()
        {
            return new ProjectTask() { Id = this.Id, Name=this.Name, TaskMode = this.TaskMode,
DateTime = this.DateTime, ProjectForeignKey=this.ProjectForeignKey };
        }

    }
}
```

User.cs

```
using System;
using System.Collections.Generic;
using System.ComponentModel.DataAnnotations;
using System.Linq;
using System.Threading.Tasks;

namespace OrganizerAPI.Model
{
    public class User
    {
        [Key]
        public long Id { get; set; }


        public string Login { get; set; }

        public string DeviceId { get; set; }

        public string Password { get; set; }

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

        public List<LogMessage> LogMessages { get; set; }

        public User CreateWithoutPassword()
        {
            return new User() { Id = this.Id, Login = this.Login, DeviceId = this.DeviceId };
        }

    }
}
```

EFOrganizerAPIDBContext.cs

```
using Microsoft.EntityFrameworkCore;
using OrganizerAPI.Model;
using System;
using System.Collections.Generic;
using System.Linq;
```

```
using System.Threading.Tasks;

namespace OrganizerAPI
{
    public class EFOrganizerAPIDBContext:DbContext
    {
        public static EFOrganizerAPIDBContext instance;
        public  EFOrganizerAPIDBContext(DbContextOptions<EFOrganizerAPIDBContext>  options)  :
base(options)
        {
            instance = this;
        }

        public DbSet<User> Users { get; set; }
        public DbSet<Project> Projects { get; set; }
        public DbSet<LogMessage> LogMessages { get; set; }
        public DbSet<ProjectTask> Tasks { get; set; }

    }
}
```

ProjectsController.cs

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Threading.Tasks;
using Microsoft.AspNetCore.Http;
using Microsoft.AspNetCore.Mvc;
using Microsoft.EntityFrameworkCore;
using Microsoft.EntityFrameworkCore.Infrastructure;
using OrganizerAPI;
using OrganizerAPI.Extensions;
using OrganizerAPI.Model;

namespace OrganizerAPI.Controllers
{
    [Route("api/[controller]")]
    [ApiController]
    [BasicAuth]
    public class ProjectsController : ControllerBase
    {

        IOrganizerAPIRepository OrganizerAPIRepository;
        EFOrganizerAPIDBContext _context;
        public    ProjectsController(EFOrganizerAPIDBContext    context,IOrganizerAPIRepository
repository)
        {
            OrganizerAPIRepository = repository;
            _context = context;
        }

        // GET: api/Projects
        [HttpGet]
        public async Task<ActionResult<IEnumerable<Project>>> GetProjects()
        {
            List<Project> results = await _context.Projects.Where((x) => x.UserForeignKey ==
CustomExtensions.GetUserID(Request)).ToListAsync();
            List<Project> resultsCleaned = new List<Project>();
            results.ForEach(x => resultsCleaned.Add(x.CleanData()));
            return resultsCleaned;
        }
```

```
        // GET: api/Projects/5
        [HttpGet("{id}")]
        public async Task<ActionResult<Project>> GetProject(long id)
        {

            var project = await _context.Projects.FindAsync(id);

            if (project == null)
            {
                return NotFound();
            }
            if (CustomExtensions.GetUserID(Request) != project.UserForeignKey)
            {
                return Forbid();
            }
            return project.CleanData();
        }

        // PUT: api/Projects/5
        // To protect from overposting attacks, enable the specific properties you want to bind
to, for
        // more details, see https://go.microsoft.com/fwlink/?linkid=2123754.
        [HttpPut("{id}")]
        public async Task<IActionResult> PutProject(long id, Project project)
        {

            Project current = await _context.Projects.FindAsync(id);
            if (current==null)
            {
                return NotFound(project);
            }
            if (current.UserForeignKey!= CustomExtensions.GetUserID(Request))
            {
                return Forbid();
            }
            if (ProjectExists(project.Name)) return Conflict($"Project with name {project.Name}
aldready exists");
            current.Name = project.Name;
            _context.Entry(current).State = EntityState.Modified;


            try
            {
                await _context.SaveChangesAsync();
            }
            catch (DbUpdateConcurrencyException ex)
            {
                return new UnprocessableEntityObjectResult(ex);
            }

            return new ObjectResult(current.CleanData());
        }

        // POST: api/Projects
        // To protect from overposting attacks, enable the specific properties you want to bind
to, for
        // more details, see https://go.microsoft.com/fwlink/?linkid=2123754.
        [HttpPost]
        public async Task<ActionResult<Project>> PostProject([FromBody] Project project)
        {
            project.UserForeignKey = CustomExtensions.GetUserID(Request);
            if (ProjectExists(project.Name)) return Conflict($"Project with name {project.Name}
aldready exists");
            _context.Projects.Add(project);
```

```
            await _context.SaveChangesAsync();

            return new ObjectResult(project.CleanData());
        }

        // DELETE: api/Projects/5
        [HttpDelete("{id}")]
        public async Task<ActionResult<Project>> DeleteProject(long id)
        {
            var project = await _context.Projects.FindAsync(id);
            if (project == null)
            {
                return NotFound();
            }
            if (project.UserForeignKey!=CustomExtensions.GetUserID(Request))
            {
                return Forbid();
            }
            _context.Projects.Remove(project);
            await _context.SaveChangesAsync();

            return new ObjectResult(project.CleanData());
        }

        [HttpGet("{id}/Tasks")]
        public async Task<ActionResult<IEnumerable<ProjectTask>>> GetProjectTasks(long id)
        {
            Project project = await _context.Projects.FindAsync(id);
            if (project.UserForeignKey != CustomExtensions.GetUserID(Request))
            {
                return Forbid();
            }
            var tasks = _context.Tasks.Where((x) => x.ProjectForeignKey == project.Id);
            List<ProjectTask> taskResult = new List<ProjectTask>();
            await tasks.ForEachAsync((x) => taskResult.Add(x.CleanData()));
            return taskResult;
        }
        private bool ProjectExists(string name)
        {
            return _context.Projects.Any(e => e.Name == name);
        }


    }
}
```

TasksController.cs

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Threading.Tasks;
using Microsoft.AspNetCore.Http;
using Microsoft.AspNetCore.Mvc;
using Microsoft.EntityFrameworkCore;
using OrganizerAPI;
using OrganizerAPI.Extensions;
using OrganizerAPI.Model;

namespace OrganizerAPI.Controllers
{
    [Route("api/[controller]")]
```

```
    [ApiController]
    [BasicAuth]
    public class TasksController : ControllerBase
    {
        private readonly EFOrganizerAPIDBContext _context;

        public TasksController(EFOrganizerAPIDBContext context)
        {
            _context = context;
        }

        // GET: api/ProjectTasks
        [HttpGet]
        public async Task<ActionResult<IEnumerable<ProjectTask>>> GetTasks()
        {

            var tasks = _context.Tasks.Include((x) => x.Project).Where((y) =>
y.Project.UserForeignKey == CustomExtensions.GetUserID(Request));
            List<ProjectTask> result = new List<ProjectTask>();
            await tasks.ForEachAsync(x => result.Add(x.CleanData()));
            return result;
        }

        // GET: api/ProjectTasks/5
        [HttpGet("{id}")]
        public async Task<ActionResult<ProjectTask>> GetProjectTask(long id)
        {
            var projectTask = await _context.Tasks.FindAsync(id);
            if (projectTask == null)
            {
                return NotFound();
            }
            var project = await _context.Projects.FindAsync(projectTask.ProjectForeignKey);
            if (project.UserForeignKey!=CustomExtensions.GetUserID(Request))
            {
                return Forbid();
            }
            return projectTask.CleanData();
        }

        // PUT: api/ProjectTasks/5
        // To protect from overposting attacks, enable the specific properties you want to bind
to, for
        // more details, see https://go.microsoft.com/fwlink/?linkid=2123754.
        [HttpPut("{id}")]
        public async Task<ActionResult<ProjectTask>> PutProjectTask(long id, ProjectTask
projectTask)
        {
            if (id != projectTask.Id)
            {
                return BadRequest();
            }
            var project = await _context.Projects.FindAsync(projectTask.ProjectForeignKey);
            if (project==null)
            {
                return NotFound();
            }
            if (project.UserForeignKey!=CustomExtensions.GetUserID(Request))
            {
                return Forbid();
            }

            _context.Entry(projectTask).State = EntityState.Modified;
            projectTask.ProjectForeignKey = project.Id;
```

```
            try
            {
                await _context.SaveChangesAsync();
            }
            catch (DbUpdateConcurrencyException)
            {
                if (!ProjectTaskExists(id))
                {
                    return NotFound();
                }
                else
                {
                    throw;
                }
            }

            return projectTask.CleanData();
        }

        // POST: api/ProjectTasks
        // To protect from overposting attacks, enable the specific properties you want to bind
to, for
        // more details, see https://go.microsoft.com/fwlink/?linkid=2123754.
        [HttpPost]
        public async Task<ActionResult<ProjectTask>> PostProjectTask(ProjectTask projectTask)
        {
            Project project = await _context.Projects.FindAsync(projectTask.ProjectForeignKey);
            if (project==null)
            {
                return NotFound($"Project with id {projectTask.Id} doesn't exist");
            }
            if (project.UserForeignKey!=CustomExtensions.GetUserID(Request))
            {
                return Forbid();
            }

            _context.Tasks.Add(projectTask);
            await _context.SaveChangesAsync();

            return projectTask.CleanData();
        }

        // DELETE: api/ProjectTasks/5
        [HttpDelete("{id}")]
        public async Task<ActionResult<ProjectTask>> DeleteProjectTask(long id)
        {
            var projectTask = await _context.Tasks.FindAsync(id);
            if (projectTask == null)
            {
                return NotFound();
            }
            Project project = await _context.Projects.FindAsync(projectTask.ProjectForeignKey);
            if (project.UserForeignKey!=CustomExtensions.GetUserID(Request))
            {
                return Forbid();
            }
            _context.Tasks.Remove(projectTask);
            await _context.SaveChangesAsync();

            return projectTask.CleanData();
        }

        private bool ProjectTaskExists(long id)
        {
```

```
            return _context.Tasks.Any(e => e.Id == id);
        }
    }
}
```

UserController.cs

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Net;
using System.Net.Http;
using System.Security.Claims;
using System.Threading.Tasks;
using Microsoft.AspNetCore.Http;
using Microsoft.AspNetCore.Mvc;
using Microsoft.EntityFrameworkCore;
using OrganizerAPI;
using OrganizerAPI.Extensions;
using OrganizerAPI.Model;

namespace OrganizerAPI.Controllers
{
    [Route("api/[controller]")]

    //[ApiController]
    public class UserController : Controller
    {


        IOrganizerAPIRepository OrganizerAPIRepository;
        private readonly EFOrganizerAPIDBContext context;

        public      UserController(EFOrganizerAPIDBContext      context,IOrganizerAPIRepository
organizerAPIRepository)
        {
            OrganizerAPIRepository = organizerAPIRepository;
            this.context = context;
        }

        [BasicAuth]
        [Microsoft.AspNetCore.Mvc.HttpGet(Name="GetCurrent")]
        public async Task<IActionResult> Get()
        {
            try
            {
                User                        user                        =                        await
context.Users.FindAsync(CustomExtensions.GetUserID(Request));//await
OrganizerAPIRepository.GetUser(OrganizerAPIRepository.GetUserID(Request));

                if (user == null)
                {
                    return NotFound();
                }

                return new ObjectResult(user.CreateWithoutPassword());
            }
            catch (Exception ex)
            {
                return new BadRequestObjectResult(ex);
            }
        }
```

```
        [Microsoft.AspNetCore.Mvc.HttpPost(Name ="Register")]
        public async Task<IActionResult> Register([FromBody] User user)
        {
            try
            {
                if (await context.Users.CountAsync((x) => x.Login == user.Login)>0) return new
Microsoft.AspNetCore.Mvc.ConflictObjectResult($"Login \"{user.Login}\" is already used");
                user = (await context.Users.AddAsync(user)).Entity;
                await context.SaveChangesAsync();
                return new ObjectResult(user.CreateWithoutPassword());
            }
            catch(Exception ex)
            {
                return new BadRequestObjectResult(ex);
            }
        }


    }
}
```

Веб-клієнт

Sign-up.component.ts

```
import { Component, OnInit } from '@angular/core';
import { FormControl, FormGroup, Validators } from '@angular/forms';
import { Router } from '@angular/router';
import { AuthService } from '../auth.service';

@Component({
  selector: 'app-sign-up',
  templateUrl: './sign-up.component.html',
  styleUrls: ['./sign-up.component.scss']
})
export class SignUpComponent implements OnInit {
  public status;
  public emailInput: string;
  public hideTextLogin = true;
  public loaded = false;
  public isError;
  public spinner = false;

  loginForm: FormGroup;

  constructor(
    private authSrvice: AuthService,
    private router: Router
  ) { }
```

```
  ngOnInit(): void {
    this.checkUserExist();

    this.loginForm = new FormGroup({
      'email': new FormControl(null, [Validators.required, Validators.email]),
      'password': new FormControl(null, [Validators.required, Validators.minLength(6),
Validators.maxLength(30)])
    });
  }

  get email() {
    return this.loginForm.get('email');
  }

  get password() {
    return this.loginForm.get('password');
  }

  isTrueCharacter(event: any) {
    const pattern = '/|,?_&^%$#!*()+=-';
    if (pattern.includes(event.data)) {
      event.target.value = event.target.value.substr(0, event.target.value.length - 1);
    }
  }

  onSubmit() {
    this.hideTextLogin = false;
    this.spinner = true;
    localStorage.clear();

    const userData = {
      login: this.email.value,
      password: this.password.value
    }

    this.authSrvice.registerUser(userData).subscribe(data => {
      localStorage.setItem('login', data.login);
      localStorage.setItem('password', this.password.value);
      localStorage.setItem('id', data.id);

      this.router.navigate(['/dashboard']);
    },(err) => {
      this.spinner = false;
```

```
      if(err.status == 409) {
        localStorage.setItem('login', userData.login);
        localStorage.setItem('password', this.password.value);

        this.router.navigate(['/dashboard']);
      }
    });
  }

  private checkUserExist(): void {
    if(localStorage.getItem('login') && localStorage.getItem('password') && localStorag
e.getItem('id')) {
      this.router.navigate(['/dashboard']);
    } else {
      localStorage.clear();
    }
  }
}
```

Sign-up.component.html

```
<form [formGroup]="loginForm" (ngSubmit)="onSubmit()">
  <span class="form-help-text-error">
    {{status}}
  </span>
  <p>Username</p>
  <input type="text" placeholder="Enter Username" id="email" formControlName="em
ail" (input)="isTrueCharacter($event)">
  <span class="form-help-text" *ngIf="email.invalid && email.touched">
    <span *ngIf="email['errors']['required']">Email cannot be empty.</span>
    <span *ngIf="email['errors']['email']">Enter a valid email.</span>
  </span>
  <p class="password-title">Password</p>
  <input type="password" placeholder="Enter Password" id="password" formControlNa
me="password">
  <span class="form-help-text" *ngIf="password.invalid && password.touched">
    <span *ngIf="password['errors']['required']">Password cannot be empty.</span>
    <span *ngIf="password['errors']['minlength'] && password['errors']['minlength']['req
uiredLength']">Password
      should be more {{password['errors']['minlength']['requiredLength']}} characters. No
w
      {{password['errors']['minlength']['actualLength']}}.</span>
    <span *ngIf="password['errors']['maxlength'] && password['errors']['maxlength']['req
uiredLength']">
```

```
      Password should be less {{password['errors']['maxlength']['requiredLength']}} chara
cters.
      Now
      {{password['errors']['maxlength']['actualLength']}}.
    </span>
  </span>

  <button class="submitButton" type="submit" [disabled]="loginForm.invalid" [class.a
wesome]="isError">
    <span *ngIf="hideTextLogin">Login</span>
    <div class="circle-loader" [ngClass]="{'load-complete': loaded }" *ngIf="spinner">
      <div class="checkmark draw" *ngIf="loaded"></div>
    </div>
  </button>
</form>
```

Sign-up.component.scss

```
h1 {
    margin: 0;
    padding: 0 0 20px;
    text-align: center;
    font-size: 22px;
}

input {
    width: 100%;
}

input:-webkit-autofill,
input:-webkit-autofill:hover,
input:-webkit-autofill:focus,
input:-webkit-autofill:active {
    transition: background-color 5000s ease-in-out 0s;
    -webkit-text-fill-color: #fff !important;
}

button {
    outline: none;
}

input[type="text"],
input[type="password"] {
    border: none;
```

```
    border-bottom: 1px solid #fff;
    background: transparent;
    outline: none;
    height: 40px;
    color: #fff;
    font-size: 16px;
}

.form-help-text {
    color: #fb494d;
    font-size: 15px;
    display: block;
    margin-top: 15px;
}

.form-help-text-error {
    color: #fb494d;
    font-size: 15px;
    display: block;
}

.submitButton[disabled] {
    height: 40px;
    width: 260px;
    margin-top: 15px;
    border: none;
}

.submitButton[disabled]:hover {
    cursor: not-allowed;
    background-color: #e84118;
    border: none;
}

.submitButton {
    height: 40px;
    width: 260px;
    margin-top: 15px;
    transition: 0.5s;
    background-color: #008080;
    color: #fff;
    border: none;
}
```

```
.submitButton:hover {
    cursor: pointer;
    background-color: #4cd137;
    color: white;
    border: none;
}

.loginbox a {
    text-decoration: none;
    font-size: 12px;
    line-height: 20px;
    color: darkgrey;
}

.awesome {
    background-color: red;
}

$brand-success: #5cb85c;
$loader-size: 7em;
$check-height: $loader-size/2;
$check-width: $check-height/2;
$check-left: ($loader-size/6 + $loader-size/12);
$check-thickness: 3px;
$check-color: $brand-success;

.circle-loader {
    border: 1px solid rgba(0, 0, 0, 0.2);
    border-left-color: $check-color;
    animation: loader-spin 1.2s infinite linear;
    position: relative;
    display: inline-block;
    vertical-align: top;
    border-radius: 50%;
    width: 25px;
    height: 25px;
}

.load-complete {
    -webkit-animation: none;
    animation: none;
    border: none;
    transition: border 5s ease-out;
}
```

```
.checkmark {
  &.draw:after {
    animation-duration: 800ms;
    animation-timing-function: ease;
    animation-name: checkmark;
    transform: scaleX(-1) rotate(135deg);
  }

  &:after {
    opacity: 1;
    height: 20px;
    width: 10px;
    transform-origin: left top;
    border-right: $check-thickness solid $check-color;
    border-top: $check-thickness solid $check-color;
    content: "";
    left: 0px;
    top: 13px;
    position: absolute;
  }
}

.password-title {
  margin-top: 15px;
}

@keyframes loader-spin {
  0% {
    transform: rotate(0deg);
  }

  100% {
    transform: rotate(360deg);
  }
}

@keyframes checkmark {
  0% {
    height: 0;
    width: 0;
    opacity: 1;
  }
```

```
  20% {
    height: 0;
    width: 10px;
    opacity: 1;
  }

  40% {
    height: 20px;
    width: 10px;
    opacity: 1;
  }

  100% {
    height: 20px;
    width: 10px;
    opacity: 1;
  }
}
```

Boards.component.ts

```
import { Component, OnInit } from '@angular/core';
import { MatDialog } from '@angular/material/dialog';
import { Observable } from 'rxjs';
import { take } from 'rxjs/operators';
import { ChangeProjectComponent } from '../popups/change-project/change-
project.component';
import { CreateProjectComponent } from '../popups/create-project/create-
project.component';
import { DeleteProjectComponent } from '../popups/delete-project/delete-
project.component';
import { DashboardService } from '../services/dashboard.service';

@Component({
  selector: 'app-boards',
  templateUrl: './boards.component.html',
  styleUrls: ['./boards.component.scss']
})
export class BoardsComponent implements OnInit {

  userProjects: Observable<{id: number, name: string, userForeignKey: number}[]>;

  constructor(
```

```
  private dashboardService: DashboardService,
  private dialog: MatDialog
) { }

ngOnInit(): void {
  this.getUserProjects();
}

createProject(): void {
  const dialogRef = this.dialog.open(CreateProjectComponent);

  dialogRef.afterClosed().subscribe(() => {
    this.getUserProjects();
  });
}

boardSettings(boardId: number, projectName: string,  event) {
  event.stopPropagation();

  const dialogRef = this.dialog.open(ChangeProjectComponent, {
    data: {
      id: boardId,
      projectName
    }
  });

  dialogRef.afterClosed().subscribe(() => {
    this.getUserProjects();
  });
}

boardDelete(boardId: number, event) {
  event.stopPropagation();

  const dialogRef = this.dialog.open(DeleteProjectComponent, {
    data: {id: boardId}
  });

  dialogRef.afterClosed().subscribe(() => {
    this.getUserProjects();
  });
}

private getUserProjects(): void {
```

```
    this.userProjects = this.dashboardService.getCurrentUserProjects();
  }

}
```

Boards.component.html

```
<div class="container">
    <div class="container-
card" *ngFor="let project of userProjects | async" [routerLink]="['/dashboard',project.id
]">
        <span class="card-name">{{ project.name }}</span>

        <div class="card-actions">
            <span class="material-
icons" (click)="boardSettings(project.id, project.name, $event)">
                settings
            </span>
            <span class="material-icons" (click)="boardDelete(project.id, $event)">
                delete
            </span>
        </div>
    </div>
    <div class="container-card-add" (click)="createProject()">
        <span class="material-icons">
            add
        </span>
    </div>
</div>
```

Boards.component.scss

```
.container {
    display: flex;
    flex-wrap: wrap;
    height: calc(100% - 40px);
    box-sizing: border-box;
    padding: 20px;
    align-content: flex-start;

    .container-card-add,
    .container-card {
        height: 120px;
```

```
    width: 200px;
    cursor: pointer;
    position: relative;
}

.container-card-add {
    background-color: rgba(9,30,66,.04);

    &:hover {
        background-color: rgba(9,30,66,.08);
    }

    .material-icons {
        position: absolute;
        top: 50%;
        left: 50%;
        transform: translate(-50%, -50%);
    }
}

.container-card {
    background-image: url("../../../assets/images/pic1.jpg");
    background-size: cover;
    background-position: 50%;
    margin: 0 20px 20px 0;

    .card-name {
        position: absolute;
        top: 10px;
        left: 20px;
        color: white;
        font-size: 16px;
        font-weight: bold;
    }

    .card-actions {
        position: absolute;
        top: 10px;
        right: 20px;

        .material-icons {
            color: white;
            margin-left: 10px;
        }
```

```
      }
    }
}
```

User-board.component.ts

```
import { CdkDragDrop, moveItemInArray, transferArrayItem } from '@angular/cdk/dra
g-drop';
import { Component, OnInit } from '@angular/core';
import { MatDialog } from '@angular/material/dialog';
import { ActivatedRoute, Route, Router } from '@angular/router';
import { take } from 'rxjs/operators';
import { CreateTaskComponent } from '../popups/create-task/create-task.component';
import { DeleteTaskComponent } from '../popups/delete-task/delete-task.component';
import { DashboardService } from '../services/dashboard.service';

@Component({
  selector: 'app-user-board',
  templateUrl: './user-board.component.html',
  styleUrls: ['./user-board.component.scss']
})
export class UserBoardComponent implements OnInit {

  id: number;

  plannedTasks: { id: number, name: string, taskMode: string, dateTime: Date, projectFo
reignKey: number }[] = [];
  completedTasks: { id: number, name: string, taskMode: string, dateTime: Date, project
ForeignKey: number }[] = [];

  constructor(
    private route: ActivatedRoute,
    private dashboardService: DashboardService,
    private dialog: MatDialog
  ) { }

  ngOnInit(): void {
    this.route.params.subscribe(data => {
      this.id = Number(data.id);
      this.getProjectTasks(data.id);
    })
  }

  async drop(event: CdkDragDrop<any>, taskStatus: string) {
```

```
  if (event.previousContainer === event.container) {
    moveItemInArray(event.container.data, event.previousIndex, event.currentIndex);
  } else {
    const changeTask = {
      ...event.previousContainer.data[event.previousIndex],
      taskMode: taskStatus
    }

    transferArrayItem(event.previousContainer.data,
                      event.container.data,
                      event.previousIndex,
                      event.currentIndex);

    await this.dashboardService.changeTask(changeTask);
  }
}

createNewTask() {
  const dialogRef = this.dialog.open(CreateTaskComponent, {
    data: {id: this.id}
  });

  dialogRef.afterClosed().subscribe(() => {
    this.getProjectTasks(String(this.id));
  });
}

taskDelete(id: number, event) {
  event.stopPropagation();

  const dialogRef = this.dialog.open(DeleteTaskComponent, {
    data: {id: id}
  });

  dialogRef.afterClosed().subscribe(() => {
    this.getProjectTasks(String(this.id));
  });
}

private getProjectTasks(id: string) {
  this.dashboardService.getProjectTasks(id).pipe(take(1)).subscribe(data => {
    this.plannedTasks = [];
    this.completedTasks = [];
    data.forEach(item => {
```

```
      item.taskMode == "planned" ? this.plannedTasks.push(item) : this.completedTasks
.push(item)
    })
  });
 }

}
```

User-board.component.html

```
<div class="container">
 <div cdkDropListGroup>
  <div class="example-container">
   <h2>To do</h2>

   <div cdkDropList [cdkDropListData]="plannedTasks" class="example-list"
    (cdkDropListDropped)="drop($event, 'planned')">
    <div class="example-box" *ngFor="let item of plannedTasks" cdkDrag>
     {{item.name}}
     <span class="material-
icons" (click)="taskDelete(item.id, $event)" style="cursor: pointer;">
      delete
     </span>
    </div>
    <div class="example-box cdk-drag-
disabled " cdkDragDisabled (click)="createNewTask()">Add new task</div>
   </div>
  </div>

  <div class="example-container">
   <h2>Done</h2>

   <div cdkDropList [cdkDropListData]="completedTasks" class="example-list"
    (cdkDropListDropped)="drop($event, 'completed')">
    <div class="example-box" *ngFor="let item of completedTasks" cdkDrag>
     {{item.name}}

     <span class="material-
icons" (click)="taskDelete(item.id, $event)" style="cursor: pointer;">
      delete
     </span>
    </div>
   </div>
  </div>
```

```
  </div>
</div>
```

User-board.component.scss

```
.container {
    box-sizing: border-box;
    padding: 20px;
}

.example-container {
    width: 400px;
    max-width: 100%;
    margin: 0 25px 25px 0;
    display: inline-block;
    vertical-align: top;
}

.example-list {
    border: solid 1px #ccc;
    min-height: 60px;
    background: white;
    border-radius: 4px;
    overflow: hidden;
    display: block;
}

.example-box {
    padding: 20px 10px;
    border-bottom: solid 1px #ccc;
    color: rgba(0, 0, 0, 0.87);
    display: flex;
    flex-direction: row;
    align-items: center;
    justify-content: space-between;
    box-sizing: border-box;
    cursor: move;
    background: white;
    font-size: 14px;
}

.cdk-drag-preview {
    box-sizing: border-box;
    border-radius: 4px;
    box-shadow: 0 5px 5px -3px rgba(0, 0, 0, 0.2), 0 8px 10px 1px rgba(0, 0, 0, 0.14),
```

```
      0 3px 14px 2px rgba(0, 0, 0, 0.12);
}

.cdk-drag-placeholder {
  opacity: 0;
}

.cdk-drag-animating {
  transition: transform 250ms cubic-bezier(0, 0, 0.2, 1);
}

.example-box:last-child {
  border: none;
}

.example-list.cdk-drop-list-dragging .example-box:not(.cdk-drag-placeholder) {
  transition: transform 250ms cubic-bezier(0, 0, 0.2, 1);
}

.example-box.cdk-drag-disabled {
  cursor: pointer;
}
```

Мобільний додаток

MainActivity.java

```
package com.example.anggarisky.eventaround;

import android.app.DatePickerDialog;
import android.app.TimePickerDialog;
import android.content.ContentProviderClient;
import android.content.Context;
import android.content.Intent;
import android.os.Handler;
import android.support.annotation.NonNull;
import android.text.InputType;
import android.text.format.DateUtils;
import android.view.Gravity;
import android.view.LayoutInflater;
import android.view.View;
import android.view.animation.AccelerateInterpolator;
import android.widget.Button;
import android.widget.CheckBox;
import android.widget.DatePicker;
import android.widget.EditText;
import android.widget.LinearLayout;
import android.widget.TextView;
import android.widget.TimePicker;
import org.json.JSONArray;
```

```
import                                                  org.json.JSONException;
import                                                     org.json.JSONObject;

import                                                     java.io.IOException;
import                                                  java.time.LocalDateTime;
import                                               java.time.format.DateTimeFormatter;
import                                                     java.util.ArrayList;
import                                                        java.util.Base64;
import                                                     java.util.Calendar;
import                                                         java.util.Date;
import                                                         java.util.List;

import                                                           okhttp3.Call;
import                                                        okhttp3.Callback;
import                                                       okhttp3.MediaType;
import                                                    okhttp3.MultipartBody;
import                                                     okhttp3.OkHttpClient;
import                                                          okhttp3.Request;
import                                                       okhttp3.RequestBody;
import                                                          okhttp3.Response;

import                          static                          java.util.logging.Level.parse;

public          class          MainActivity          extends          AppCompatActivity          {
    static                                      String                                      resp;
    TextView                                                                          tempText;
    RecyclerView                                                                    eventsplace;
    RecyclerView                                                                   projectsplace;
    static                              ProjectAdapter                             projectAdapter;
    static                                EventAdapter                               eventAdapter;
    static                                List<Project>                               projectList;
    static                                  List<Event>                                 eventList;

    public     static     void     ChangeProject(int     id,     final     MainActivity     m)
    {
        for(int                          i                          =0;i<projectList.size();i++)
            if(projectList.get(i).id                          ==id)                          {
                textView4.setText(projectList.get(i).getProjecttitle());
                m.FetchTasksFromProject(id);
            }
        projectAdapter.notifyDataSetChanged();
    }

    public                  void                  CreateProject(String                  title)
    {
    CreateProjectRemote(title);
    }

    public                void                CreateProjectRemote(String                title)
    {
        OkHttpClient            client2            =            new            OkHttpClient();
        String url2 = "https://organizerapi20201224221914.azurewebsites.net/api/projects";
        RequestBody                                                                 requestBody;
        requestBody       =       RequestBody.create(MediaType.parse("application/json"),"{
\"name\":\""+title+"\"}");
        Request            request2            =            new            Request.Builder()
                .url(url2)
                .addHeader("Authorization",auth)
                .post(requestBody)
                .build();
        client2.newCall(request2).enqueue(new                  Callback()                  {
```

```
        @Override
        public void onFailure(Call call, IOException e) {
            e.printStackTrace();
        }
        @Override
        public void onResponse(Call call, Response response) throws IOException {
            if (response.isSuccessful()) {
                final String myResponse = response.body().string();
                MainActivity.this.runOnUiThread(new Runnable() {
                    @Override
                    public void run() {
                        try {
                            JSONArray obj = new JSONArray(myResponse);
                            FetchProjects();
                        } catch (JSONException e) {
                            e.printStackTrace();
                        }

                    }
                });
            }
        }
    });
}


public static void DeleteProject(int id, final MainActivity m)
{
    for(int i=0;i<projectList.size();i++)
        if(projectList.get(i).id==id)
            projectList.remove(i);

    m.DeleteProjectRemote(id);
    m.FetchTasks();
    projectAdapter.notifyDataSetChanged();
}

public String Calendar2Str(Calendar dateAndTime)
{
    String s = "";
    s+=String.valueOf(dateAndTime.get(Calendar.YEAR))+"-";

    if(dateAndTime.get(Calendar.MONTH)<10)
        s+="0"+String.valueOf(dateAndTime.get(Calendar.MONTH)+1)+"-";
    else
        s+=String.valueOf(dateAndTime.get(Calendar.MONTH)+1)+"-";

    if(dateAndTime.get(Calendar.DAY_OF_MONTH)<10)
        s+="0"+String.valueOf(dateAndTime.get(Calendar.DAY_OF_MONTH))+"T";
    else
        s+=String.valueOf(dateAndTime.get(Calendar.DAY_OF_MONTH))+"T";

    if(dateAndTime.get(Calendar.HOUR_OF_DAY)<10)
        s+="0"+String.valueOf(dateAndTime.get(Calendar.HOUR_OF_DAY))+":";
    else
        s+=String.valueOf(dateAndTime.get(Calendar.HOUR_OF_DAY))+":";

    if(dateAndTime.get(Calendar.MINUTE)<10)
        s+="0"+String.valueOf(dateAndTime.get(Calendar.MINUTE));
    else
        s+=String.valueOf(dateAndTime.get(Calendar.MINUTE));
```

```
        s+=":00";
        return                                                          s;
    }

    public  void  CreateTaskRemote(String  title,Boolean  status,  Calendar  dateAndTime)
    {
        OkHttpClient           client2          =          new          OkHttpClient();
        String  url2  =  "https://organizerapi20201224221914.azurewebsites.net/api/tasks";
        RequestBody                                                     requestBody;
        if(status)
           requestBody     =     RequestBody.create(MediaType.parse("application/json"),"{
\"name\":\""+title+"\",\"taskmode\":\"completed\",
\"datetime\":\""+Calendar2Str(dateAndTime)+"\",\"ProjectForeignKey\":"+String.valueOf(proj
ectId)+"}");
        else
            requestBody     =     RequestBody.create(MediaType.parse("application/json"),"{
\"name\":\""+title+"\",\"taskmode\":\"planned\",
\"datetime\":\""+Calendar2Str(dateAndTime)+"\",\"ProjectForeignKey\":"+String.valueOf(proj
ectId)+"}");

         Request           request2          =          new          Request.Builder()
                .url(url2)
                .addHeader("Authorization",auth)
                .post(requestBody)
                .build();
        client2.newCall(request2).enqueue(new                Callback()                {
            @Override
            public      void      onFailure(Call      call,      IOException      e)      {
                e.printStackTrace();
            }
            @Override
            public  void  onResponse(Call  call,  Response  response)  throws  IOException  {
                if                          (response.isSuccessful())                       {
                    final      String      myResponse      =      response.body().string();
                    MainActivity.this.runOnUiThread(new            Runnable()            {
                        @Override
                        public                void                run()                {
                            try                                                          {
                                JSONArray     obj     =     new     JSONArray(myResponse);
                                }         catch         (JSONException         e)         {
                                e.printStackTrace();
                            }

                        }
                    });
                }
            }
        });
        FetchTasksFromProject(projectId);
    }


    public                              void                              FetchTasksFirst()
    {
         OkHttpClient          client2          =          new          OkHttpClient();
        String  url2  =  "https://organizerapi20201224221914.azurewebsites.net/api/tasks";
        Request           request2          =          new          Request.Builder()
                .url(url2)
                .addHeader("Authorization",auth)
                .build();
        client2.newCall(request2).enqueue(new                Callback()                {
```

```
            @Override
            public      void      onFailure(Call      call,      IOException      e)      {
                e.printStackTrace();
            }
            @Override
            public  void  onResponse(Call  call,  Response  response)  throws  IOException  {
                if                          (response.isSuccessful())                          {
                    final      String      myResponse      =      response.body().string();
                    MainActivity.this.runOnUiThread(new            Runnable()            {
                        @Override
                        public                 void                 run()                 {
                            try                                                              {
                                JSONArray     obj     =     new     JSONArray(myResponse);

                                for(int         i         =0;i<obj.length();i++)         {
                                    JSONObject  object  =  new  JSONObject(obj.getString(i));
                                    address[Integer.parseInt(object.getString("id"))]    =
Integer.parseInt(object.getString("projectForeignKey"));

if(object.getString("taskMode").charAt(0)=="completed".charAt(0))
                                        eventList.add(new
Event(Integer.parseInt(object.getString("id")),                                                new
Date(Integer.parseInt(object.getString("dateTime").substring(0,4)),Integer.parseInt(object
.getString("dateTime").substring(5,7))-1,

Integer.parseInt(object.getString("dateTime").substring(8,10)),Integer.parseInt(object.get
String("dateTime").substring(11,13)),

Integer.parseInt(object.getString("dateTime").substring(14,16))),object.getString("name"),
"Виконано",R.drawable.picone));
                                    else
                                        eventList.add(new
Event(Integer.parseInt(object.getString("id")),                                                new
Date(Integer.parseInt(object.getString("dateTime").substring(0,4)),Integer.parseInt(object
.getString("dateTime").substring(5,7)),

Integer.parseInt(object.getString("dateTime").substring(8,10)),Integer.parseInt(object.get
String("dateTime").substring(11,13)),

Integer.parseInt(object.getString("dateTime").substring(14,16))),object.getString("name"),
"Не                                                          виконано",R.drawable.picone));

                                }
                                eventAdapter.notifyDataSetChanged();
                            }          catch          (JSONException          e)          {
                                e.printStackTrace();
                            }

                        }
                    });
                }
            }
        });



    }


    public                                  void                                  FetchProjects()
    {
```

```
        OkHttpClient client = new OkHttpClient();
        String url = "https://organizerapi20201224221914.azurewebsites.net/api/projects";
        Request request = new Request.Builder()
                .url(url)
                .addHeader("Authorization",auth)
                .build();
        client.newCall(request).enqueue(new Callback() {
            @Override
            public void onFailure(Call call, IOException e) {
                e.printStackTrace();
            }
            @Override
            public void onResponse(Call call, Response response) throws IOException {
                if (response.isSuccessful()) {
                    final String myResponse = response.body().string();
                    MainActivity.this.runOnUiThread(new Runnable() {
                        @Override
                        public void run() {
                            try {
                                projectList.clear();
                                JSONArray obj = new JSONArray(myResponse);
                                resp = obj.getString(0);
                                for(int i=0;i<obj.length();i++) {
                                    JSONObject object = new JSONObject(obj.getString(i));
                                    projectList.add(new
Project(Integer.parseInt(object.getString("id")), object.getString("name")));
                                }
                                projectAdapter.notifyDataSetChanged();
                            } catch (JSONException e) {
                                e.printStackTrace();
                            }
                            projectAdapter.notifyDataSetChanged();
                        }
                    });
                }
            }
        });
    }


    public static void EditProject(int id,String title,MainActivity m)
    {
        for(int i=0;i<projectList.size();i++)
            if(projectList.get(i).id==id) {
                projectList.get(i).projecttitle = title;
            }
        m.EditProjectRemote(id,title);
        projectAdapter.notifyDataSetChanged();
    }

    public static void EditProjectForm(final int editid, final MainActivity m)
    {
        final AlertDialog dialogBuilder = new AlertDialog.Builder(m).create();
        LayoutInflater inflater = m.getLayoutInflater();
        View dialogView = inflater.inflate(R.layout.project_dialog, null);
        final EditText editText = (EditText) dialogView.findViewById(R.id.edt_comment);
        Button button1 = (Button) dialogView.findViewById(R.id.buttonSubmit);
        Button button2 = (Button) dialogView.findViewById(R.id.buttonCancel);

        for(int i=0;i<projectList.size();i++)
            if(projectList.get(i).id==editid) {
```

```
                editText.setText(projectList.get(i).getProjecttitle());
            }

        button1.setOnClickListener(new              View.OnClickListener()              {
            @Override
            public            void            onClick(View            view)            {
                EditProject(editid,editText.getText().toString(),m);
                dialogBuilder.dismiss();
            }
        });

        button2.setOnClickListener(new              View.OnClickListener()              {
            @Override
            public            void            onClick(View            view)            {
                dialogBuilder.dismiss();
            }
        });
        dialogBuilder.setView(dialogView);
        dialogBuilder.show();
    }

    public        static        void        MakeToken(String        log,String        pass)
    {
        auth                        =                        "Basic                        ";
        if   (android.os.Build.VERSION.SDK_INT   >=   android.os.Build.VERSION_CODES.O)   {
            auth+=          Base64.getEncoder().encodeToString((log+":"+pass).getBytes());
        }
    }

    public        static        void        LoginForm(final        MainActivity        m)
    {
        final    AlertDialog    dialogBuilder    =    new    AlertDialog.Builder(m).create();
        dialogBuilder.setCanceledOnTouchOutside(false);
        LayoutInflater            inflater            =            m.getLayoutInflater();
        View      dialogView      =      inflater.inflate(R.layout.login_dialog,      null);
        final   EditText   pass   =   (EditText)   dialogView.findViewById(R.id.edt_comment);
        pass.setInputType(InputType.TYPE_CLASS_TEXT                                        |
InputType.TYPE_TEXT_VARIATION_PASSWORD);
        final   EditText   log   =   (EditText)   dialogView.findViewById(R.id.edt_comment2);
        Button     button1     =     (Button)     dialogView.findViewById(R.id.buttonSubmit);
        Button     button2     =     (Button)     dialogView.findViewById(R.id.buttonCancel);

        button2.setOnClickListener(new              View.OnClickListener()              {
            @Override
            public            void            onClick(View            view)            {
                m.Login(log.getText().toString(),pass.getText().toString());
                 Thread          thread          =          new          Thread()          {
                       @Override
                       public                void                run()                {
                           try                                                        {
                               sleep(1500);
                               if(logged)                                                {
                                   dialogBuilder.dismiss();
                                   m.FetchTasksFirst();
                                   m.FetchProjects();
                               }
                           }        catch        (InterruptedException        e)        {
                               e.printStackTrace();
                           }
                       }
                   };
```

```
                thread.start();
            }
        });

        button1.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View view) {

                Thread thread = new Thread() {
                    @Override
                    public void run() {
                        try {
                            sleep(1900);
                            if(logged) {
                                dialogBuilder.dismiss();
                                m.FetchTasksFirst();
                                m.FetchProjects();
                            }

                        } catch (InterruptedException e) {
                            e.printStackTrace();
                        }
                    }
                };

                m.Register(log.getText().toString(),pass.getText().toString());
                thread.start();
            }
        });

        dialogBuilder.setView(dialogView);
        dialogBuilder.show();
    }

    public void Login(String log, String pass)
    {
       MakeToken(log,pass);
      OkHttpClient client2 = new OkHttpClient();
        String url2 = "https://organizerapi20201224221914.azurewebsites.net/api/user";
        Request request2 = new Request.Builder()
                .url(url2)
                .addHeader("Authorization",auth)
                .build();
        client2.newCall(request2).enqueue(new Callback() {
            @Override
            public void onFailure(Call call, IOException e) {
                e.printStackTrace();
            }
            @Override
            public void onResponse(Call call, Response response) throws IOException {
              if (response.isSuccessful()) {
                    final String myResponse = response.body().string();
                    MainActivity.this.runOnUiThread(new Runnable() {
                        @Override
                        public void run() {
                            try {
                                logged = true;
                                JSONArray obj = new JSONArray(myResponse);

                            } catch (JSONException e) {
                                e.printStackTrace();
```

```
                        }
                    }
                });
            }
        }
    });
}


    TimePickerDialog.OnTimeSetListener t=new TimePickerDialog.OnTimeSetListener() {
        public void onTimeSet(TimePicker view, int hourOfDay, int minute) {
            dateAndTime.set(Calendar.HOUR_OF_DAY, hourOfDay);
            dateAndTime.set(Calendar.MINUTE, minute);

        }
    };

    DatePickerDialog.OnDateSetListener d=new DatePickerDialog.OnDateSetListener() {
        public void onDateSet(DatePicker view, int year, int monthOfYear, int dayOfMonth)
{
            dateAndTime.set(Calendar.YEAR, year);
            dateAndTime.set(Calendar.MONTH, monthOfYear);
            dateAndTime.set(Calendar.DAY_OF_MONTH, dayOfMonth);

        }
    };

    public void setDate(View v) {
        new DatePickerDialog(MainActivity.this, d,
                dateAndTime.get(Calendar.YEAR),
                dateAndTime.get(Calendar.MONTH),
                dateAndTime.get(Calendar.DAY_OF_MONTH))
                .show();
        }

    public void setTime(View v) {
        new TimePickerDialog(MainActivity.this, t,
                dateAndTime.get(Calendar.HOUR_OF_DAY),
                dateAndTime.get(Calendar.MINUTE), true)
                .show();
    }

@Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        textView4 = (TextView) findViewById(R.id.textView4);
        tempText = new TextView(this);
        dateAndTime=Calendar.getInstance();
        address = new int[1000];

       LoginForm(this);

        Thread thread = new Thread() {
            @Override
            public void run() {
                try {
                   while (true)
                    {
                        sleep(1300);
                        FetchProjects();
```

```
                }
            } catch (InterruptedException e) {
                e.printStackTrace();
            }
        }
    };

    thread.start();

    FetchProjects();
    FetchTasksFirst();

    button = (Button) findViewById(R.id.button);
    button5 = (Button) findViewById(R.id.button5);
    button6 = (Button) findViewById(R.id.button6);

    projectId = -1;
    button.setEnabled(false);

    button5.setOnClickListener(new View.OnClickListener() {
                                   @Override
                                   public void onClick(View v) {
                                   final AlertDialog dialogBuilder = new
AlertDialog.Builder(m).create();
                                   LayoutInflater inflater = m.getLayoutInflater();
                                   View dialogView =
inflater.inflate(R.layout.project_dialog, null);
                                   final EditText editText = (EditText)
dialogView.findViewById(R.id.edt_comment);
                                   Button button1 = (Button)
dialogView.findViewById(R.id.buttonSubmit);
                                   Button button2 = (Button)
dialogView.findViewById(R.id.buttonCancel);

                                       button1.setOnClickListener(new
View.OnClickListener()
                                       {
                                           @Override
                                           public void onClick(View view) {
                                               // DO SOMETHINGS

CreateProject(editText.getText().toString());
                                               dialogBuilder.dismiss();
                                           }
                                       });

                                       button2.setOnClickListener(new
View.OnClickListener()
                                       {
                                           @Override
                                           public void onClick(View view) {
                                               dialogBuilder.dismiss();
                                           }
                                       });
                                       dialogBuilder.setView(dialogView);
                                       dialogBuilder.show();
                                   }
                               }
    );

    button.setOnClickListener(new View.OnClickListener() {
                                  @Override
                                  public void onClick(View v) {
```

```
                                        final    AlertDialog    dialogBuilder    =    new
AlertDialog.Builder(m).create();
                                        LayoutInflater inflater = m.getLayoutInflater();
                                        View                dialogView                =
inflater.inflate(R.layout.event_dialog,                                                null);

                                        final    EditText    editText    =    (EditText)
dialogView.findViewById(R.id.edt_comment);
                                        final    CheckBox    checkBox    =    (CheckBox)
dialogView.findViewById(R.id.checkBox);
                                        Button         button1         =         (Button)
dialogView.findViewById(R.id.buttonSubmit);
                                        Button         button2         =         (Button)
dialogView.findViewById(R.id.buttonCancel);

                                        button1.setOnClickListener(new
View.OnClickListener()                                                                 {
                                            @Override
                                            public    void    onClick(View    view)    {
                                                //             DO             SOMETHINGS

CreateTask(editText.getText().toString(),checkBox.isChecked(),dateAndTime);
                                                dateAndTime=Calendar.getInstance();
                                                dialogBuilder.dismiss();
                                            }
                                        });

                                        button2.setOnClickListener(new
View.OnClickListener()                                                                 {
                                            @Override
                                            public    void    onClick(View    view)    {
                                                dialogBuilder.dismiss();
                                            }
                                        });
                                        dialogBuilder.setView(dialogView);
                                        dialogBuilder.show();
                                    }
                                }
        );

        button6.setOnClickListener(new              View.OnClickListener()              {
                                    @Override
                                    public       void       onClick(View       v)       {

                                     FetchTasks();
                                    }
                                }
        );

        projectsplace                =                findViewById(R.id.projectsplace);
        projectList                  =                new                  ArrayList<>();

        RecyclerView.LayoutManager             layoutManager             =             new
LinearLayoutManager(getApplicationContext());
        projectsplace.setLayoutManager(layoutManager);
        projectsplace.setHasFixedSize(true);

        projectAdapter=                new                ProjectAdapter(this,projectList);
        projectsplace.setAdapter(projectAdapter);
        projectsplace.setItemAnimator(new                          DefaultItemAnimator());
        eventsplace                  =                  findViewById(R.id.eventsplace);
```

```
        eventList                   =                   new                   ArrayList<>();

      eventList.add(
                new       Event(-1,               new       Date(2020,12,22,22,19)       ,
                        "Натисніть на жовту кнопку для редагування задачі і на червону для
видалення",
                        "Натисність,                    якщо                    виконано",
                        R.drawable.picone
                )
        );


        final   LinearLayoutManager   linearLayoutManager   =   new   LinearLayoutManager
                (this,               LinearLayoutManager.HORIZONTAL,                false);

        eventsplace.setLayoutManager(linearLayoutManager);
        eventsplace.setHasFixedSize(true);

        eventAdapter         =         new         EventAdapter(this,         eventList);
        eventsplace.setAdapter(eventAdapter);

        final    SnapHelper    snapHelper    =    new    GravitySnapHelper(Gravity.START);
        snapHelper.attachToRecyclerView(eventsplace);

        final         Handler         handler         =         new         Handler();
        handler.postDelayed(new                         Runnable()                         {
            @Override
            public                     void                     run()                     {
                RecyclerView.ViewHolder       viewHolderDefault       =       eventsplace.
                        findViewHolderForAdapterPosition(0);

                LinearLayout     eventparentDefault     =     viewHolderDefault.itemView.
                        findViewById(R.id.eventparent);
                eventparentDefault.animate().scaleY(1).scaleX(1).setDuration(350).
                        setInterpolator(new             AccelerateInterpolator()).start();

                LinearLayout     eventcategoryDefault     =     viewHolderDefault.itemView.
                        findViewById(R.id.eventbadge);
                eventcategoryDefault.animate().alpha(1).setDuration(300).start();

            }
        },                                                                          100);

        eventsplace.addOnScrollListener(new       RecyclerView.OnScrollListener()       {
            @Override
            public  void  onScrollStateChanged(@NonNull  RecyclerView  recyclerView,  int
newState)                                                                           {
                super.onScrollStateChanged(recyclerView,                        newState);

                if          (newState          ==          RecyclerView.SCROLL_STATE_IDLE){
                    View     view     =     snapHelper.findSnapView(linearLayoutManager);
                    int        pos        =        linearLayoutManager.getPosition(view);

                    RecyclerView.ViewHolder                  viewHolder                  =
                            eventsplace.findViewHolderForAdapterPosition(pos);

                    LinearLayout                       eventparent                       =
viewHolder.itemView.findViewById(R.id.eventparent);
                    eventparent.animate().scaleY(1).scaleX(1).setDuration(350).
                            setInterpolator(new        AccelerateInterpolator()).start();
```

```
                    LinearLayout        eventcategory        =        viewHolder.itemView.
                            findViewById(R.id.eventbadge);
                    eventcategory.animate().alpha(1).setDuration(300).start();

                }
                else                                                                    {
                    View     view     =     snapHelper.findSnapView(linearLayoutManager);
                    int        pos        =        linearLayoutManager.getPosition(view);

                    RecyclerView.ViewHolder                  viewHolder                  =
                            eventsplace.findViewHolderForAdapterPosition(pos);

                    LinearLayout                        eventparent                        =
viewHolder.itemView.findViewById(R.id.eventparent);
                    eventparent.animate().scaleY(0.7f).scaleX(0.7f).
                            setInterpolator(new
AccelerateInterpolator()).setDuration(350).start();

                    LinearLayout        eventcategory        =        viewHolder.itemView.
                            findViewById(R.id.eventbadge);
                    eventcategory.animate().alpha(0).setDuration(300).start();
                }
            }

            @Override
            public void onScrolled(@NonNull RecyclerView recyclerView, int dx, int dy) {
                super.onScrolled(recyclerView,                  dx,                  dy);
            }
        });
    }
}
```

EventAdapter.java

```
package                                                 com.example.anggarisky.eventaround;

import                                                          android.content.Context;
import                                                               android.os.Bundle;
import                                                  android.support.annotation.NonNull;
import                                              android.support.v7.widget.RecyclerView;
import                                                      android.view.LayoutInflater;
import                                                           android.view.View;
import                                                        android.view.ViewGroup;
import                                                         android.widget.Button;
import                                                       android.widget.ImageView;
import                                                     android.widget.LinearLayout;
import                                                         android.widget.TextView;

import                                                                  java.util.Date;
import                                                                  java.util.List;

public   class   EventAdapter   extends   RecyclerView.Adapter<EventAdapter.EventViewHolder>{

    Context                                                                      context;
    List<Event>                                                                eventList;
    private                                  Button                              button;

    public       EventAdapter(Context       context,       List<Event>       eventList){
        this.context                                =                                context;
```

```
        this.eventList = eventList;

    }

    @NonNull
    @Override
    public EventViewHolder onCreateViewHolder(@NonNull ViewGroup viewGroup, int i) {
        LayoutInflater layoutInflater = LayoutInflater.from(context);
        View view = layoutInflater.inflate(R.layout.item_event, null);

        return new EventViewHolder(view);
    }


    private String transformDate(Date d)
    {
        String s = "";
        if(d.getDate()<10)
            s+="0"+String.valueOf(d.getDate())+".";
        else
            s+=String.valueOf(d.getDate())+".";

        if(d.getMonth()<10)
            s+="0"+String.valueOf(d.getMonth())+".";
        else
            s+=String.valueOf(d.getMonth())+".";
        s+=String.valueOf(d.getYear())+" ";

        if(d.getHours()<10)
            s+="0"+String.valueOf(d.getHours())+":";
        else
            s+=String.valueOf(d.getHours())+":";

        if(d.getMinutes()<10)
            s+="0"+String.valueOf(d.getMinutes());
        else
            s+=String.valueOf(d.getMinutes());

        return s;
    }



    @Override
    public void onBindViewHolder(@NonNull EventViewHolder eventViewHolder, int i) {

        final Event event = eventList.get(i);
        eventViewHolder.b1.setOnClickListener(new View.OnClickListener() {
                                                  @Override
                                                  public void onClick(View v) {
                                                      MainActivity.EditTaskForm(event.id,
(MainActivity) context);
                                                  }
                                              }
        );
        eventViewHolder.b2.setOnClickListener(new View.OnClickListener() {
                                                  @Override
                                                  public void onClick(View v) {
                                                      MainActivity.DeleteTask(event.id,
(MainActivity) context);
                                                  }
```

```
                                                }
        );
        eventViewHolder.eventcategory.setOnClickListener(new    View.OnClickListener()    {
                                                    @Override
                                                    public    void    onClick(View    v)    {

MainActivity.CompleteTask(event.id,(MainActivity)                                          context);
                                                }
                                            }
        );
        eventViewHolder.badge.setOnClickListener(new        View.OnClickListener()        {
                                                     @Override
                                                     public void onClick(View v) {

MainActivity.CompleteTask(event.id,(MainActivity)                                          context);
                                                     }
                                                 }
        );

        eventViewHolder.eventtitle.setText(event.getEventitle());
        eventViewHolder.eventcategory.setText(event.getEventcategory());
        eventViewHolder.eventpicture.setImageDrawable(context.getResources().
                getDrawable(event.getEventpicture()));
        eventViewHolder.date.setText(transformDate(event.getDate()));
    }

    @Override
    public                    int                    getItemCount()                    {
        return                                                          eventList.size();
    }

    class          EventViewHolder          extends          RecyclerView.ViewHolder          {

        TextView                          eventtitle,                          eventcategory;
        ImageView                                                               eventpicture;
        Button                                                                         b1,b2;
        TextView                                                                     date;
        LinearLayout                                                               badge;
        public          EventViewHolder(@NonNull          View          itemView)          {
            super(itemView);

            badge                =                itemView.findViewById(R.id.eventbadge);
            eventtitle             =              itemView.findViewById(R.id.eventtitle);
            eventcategory          =           itemView.findViewById(R.id.eventcategory);
            eventpicture           =            itemView.findViewById(R.id.eventpicture);
            b1                   =                   itemView.findViewById(R.id.button3);
            b2                   =                   itemView.findViewById(R.id.button4);
            date                 =                 itemView.findViewById(R.id.textView2);


        }
    }

}
```

Event.java

```
package                                                com.example.anggarisky.eventaround;

import                                                                  android.os.Bundle;
```

```
import java.time.LocalDateTime;
import java.util.Date;

public class Event {
    int id;
    Date date;
    String eventitle;
    String eventcategory;
    Integer eventpicture;

    public Event() {
    }

    public Event(int id, Date date, String eventitle, String eventcategory, Integer
eventpicture) {
        this.id = id;
        this.date = date;
        this.eventitle = eventitle;
        this.eventcategory = eventcategory;
        this.eventpicture = eventpicture;
    }

    public String getEventitle() {
        return eventitle;
    }
    public Date getDate() {
        return date;
    }
    public void setEventitle(String eventitle) {
        this.eventitle = eventitle;
    }

    public String getEventcategory() {
        return eventcategory;
    }

    public void setEventcategory(String eventcategory) {
        this.eventcategory = eventcategory;
    }

    public Integer getEventpicture() {
        return eventpicture;
    }

    public void Delete() {

    }

    public void setEventpicture(Integer eventpicture) {
        this.eventpicture = eventpicture;
    }
}
```

activity_main.xml

```
<?xml version="1.0" encoding="utf-8"?>
<android.support.constraint.ConstraintLayout
xmlns:android="http://schemas.android.com/apk/res/android"
```

```
xmlns:app="http://schemas.android.com/apk/res-auto"
xmlns:tools="http://schemas.android.com/tools"
android:layout_width="match_parent"
android:layout_height="match_parent"
android:background="#F8F9FB"
tools:context=".MainActivity">

<Button
    android:id="@+id/button5"
    android:layout_width="96dp"
    android:layout_height="50dp"
    android:layout_marginStart="10dp"
    android:layout_marginTop="15dp"
    android:backgroundTint="#90ED24"
    android:fontFamily="sans-serif"
    android:text="Новий                                                    проект"
    android:textSize="11sp"
    app:layout_constraintStart_toEndOf="@+id/button"
    app:layout_constraintTop_toBottomOf="@+id/eventsplace"                    />

<Button
    android:id="@+id/button6"
    android:layout_width="96dp"
    android:layout_height="50dp"
    android:layout_marginTop="15dp"
    android:layout_marginEnd="20dp"
    android:backgroundTint="#FEE8CE"
    android:fontFamily="sans-serif"
    android:text="Усі                                                     роботи"
    android:textSize="11sp"
    app:layout_constraintEnd_toEndOf="parent"
    app:layout_constraintTop_toBottomOf="@+id/eventsplace"                    />

<View
    android:id="@+id/view"
    android:layout_width="match_parent"
    android:layout_height="330dp"
    android:background="@drawable/bgheader"
    app:layout_constraintEnd_toEndOf="parent"
    app:layout_constraintStart_toStartOf="parent"
    app:layout_constraintTop_toTopOf="parent"                    />

<TextView
    android:id="@+id/textView3"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_marginStart="8dp"
    android:layout_marginTop="24dp"
    android:layout_marginEnd="8dp"
    android:fontFamily="@font/mr"
    android:text="Менеджер                                                 проекту"
    android:textColor="#FFF"
    android:textSize="24sp"
    app:layout_constraintEnd_toEndOf="parent"
    app:layout_constraintStart_toStartOf="parent"
    app:layout_constraintTop_toTopOf="parent"                    />

<TextView
    android:id="@+id/textView4"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
```

```
        android:layout_marginStart="8dp"
        android:layout_marginTop="8dp"
        android:layout_marginEnd="8dp"
        android:fontFamily="@font/ml"
        android:text="Усі                                                    роботи"
        android:textColor="#c8ffffff"
        android:textSize="18sp"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toBottomOf="@+id/textView3"                />

    <android.support.v7.widget.RecyclerView
        android:id="@+id/eventsplace"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:layout_marginStart="24dp"
        android:layout_marginTop="36dp"
        android:layout_marginEnd="24dp"
        android:clipToPadding="false"

        android:paddingRight="250dp"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintHorizontal_bias="0.0"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toBottomOf="@+id/textView4">


    </android.support.v7.widget.RecyclerView>

    <Button
        android:id="@+id/button"
        android:layout_width="96dp"
        android:layout_height="50dp"
        android:layout_marginStart="20dp"
        android:layout_marginTop="15dp"
        android:backgroundTint="#90ED24"
        android:fontFamily="sans-serif"
        android:text="Нова                                                   задача"
        android:textSize="11sp"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toBottomOf="@+id/eventsplace"              />

    <android.support.v7.widget.RecyclerView
        android:id="@+id/projectsplace"
        android:layout_width="373dp"
        android:layout_height="267dp"
        android:layout_marginTop="15dp"
        app:layout_constraintBottom_toBottomOf="parent"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintHorizontal_bias="0.457"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toBottomOf="@+id/button"
        app:layout_constraintVertical_bias="0.046"                          />

</android.support.constraint.ConstraintLayout>
```

UNICHECK

Ім'я користувача:
Кафедра КІ

ID перевірки:
1015239038

Дата перевірки:
25.05.2023 06:30:24 EEST

Тип перевірки:
Doc vs Internet + Library

Дата звіту:
25.05.2023 06:34:14 EEST

ID користувача:
100005591

Назва документа: **Коляда_Інформаційна система транспортно-логістичного підприємства**

Кількість сторінок: **66** Кількість слів: **9600** Кількість символів: **75755** Розмір файлу: **2.91 MB** ID файлу: **1014915262**

Виявлено модифікації тексту (можуть впливати на відсоток схожості)

18.6%
Схожість

Найбільша схожість: **3.7%** з Інтернет-джерелом (https://ua-referat.com/uploaded/roboti-v4/index1.html)

18.1% Джерела з Інтернету	233	Сторінка 68
7.03% Джерела з Бібліотеки	71	Сторінка 69

0.36% Цитат

Цитати	3	Сторінка 70
Посилання	1	Сторінка 70

0%
Вилучень

Немає вилучених джерел

Модифікації

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

Підозріле форматування	11 сторінок

Thu May 25 05:36:08 EEST 2023, Медзатий Дмитро Миколайович, Хмельницький національний університет, ХНУ

Anti-Plagiarism v-15.257

Максимальне співпадіння з одним документом 4.0%

Словники перевірки: en_US, ru_RU, ua_UA. **Помилок в документах: 8%**

ID: 113972 Назва: БКР Інформаційна система організації повсякденної діяльності користувача Додано в БД: 2023-05-25 Автора: Д. О. Коляда Керівники: Т. О. Говорущенко Консультанти: Опоненти:	Документ		Сумарний збіг по Базі Даних	
	Символи	Лексеми	Символи	Лексеми
	63673	500	8473 (13%)	87 (17%)

Джерело плагіату

ID	Опис	Наявність плагіату в документі	
		Символи	Лексеми

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ХМЕЛЬНИЦЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ

РЕЦЕНЗІЯ НА КВАЛІФІКАЦІЙНУ РОБОТУ

Дипломник: Коляда Денис Олександрович

Тема: Інформаційна система організації повсякденної діяльності користувача

Спеціальність: 126 «Інформаційні системи та технології»

Обсяг кваліфікаційної роботи:

Кількість листів креслень ___3____ Кількість сторінок записки ___66___

1. Короткий зміст роботи та прийнятих рішень: Метою кваліфікаційної роботи є полегшення організації повсякденної діяльності користувача шляхом проєктування та розроблення інформаційної системи організації повсякденної діяльності користувача (органайзера).

2. Висновок про відповідність роботи дипломному завданню: Робота повністю відповідає поставленому завданню.

3. Характеристика виконання кожного розділу, ступінь використання останніх досягнень науки і техніки і передових методів роботи: В першому розділі кваліфікаційної роботи виконано дослідження предметної області. Крім цього, в першому розділі виконано постановку задачі подальшого дослідження. В другому розділі кваліфікаційної роботи спроектовано архітектуру інформаційної системи організації повсякденної діяльності користувача: розроблено графік виконання проектних робіт; проаналізовано ризики проекту; сформовано вимоги до інформаційної системи; розроблено автоматну модель інформаційної системи; розроблено UML-діаграми інформаційної системи. В третьому розділі кваліфікаційної роботи виконано реалізацію та тестування роботи інформаційної системи організації повсякденної діяльності користувача, а саме: описано реалізацію інформаційної системи організації повсякденної діяльності користувача; представлено приклади роботи мобільної версії інформаційної системи організації повсякденної діяльності користувача; проаналізовано результати тестування клієнтської та серверної частини інформаційної системи організації повсякденної діяльності користувача.

4. Позитивні сторони роботи: Висока практична цінність роботи. Практичне значення має спроєктована та реалізована інформаційна система організації повсякденної діяльності користувача (органайзер), яка відповідає за організацію повсякденної діяльності незалежно від її специфіки.

5. Негативні сторони роботи: недостатня увага перевагам та недолікам розробленої інформаційної системи.

6. Оцінка графічного оформлення та пояснювальної записки роботи: Пояснювальна записка оформлена коректно, згідно діючих стандартів оформлення документації.

7. Відгук про роботу в цілому: Робота виконана на належному інженерно-технічному рівні.

8. Інші зауваження: _____

9. Оцінка дипломної роботи: добре (3.75/C)

Рецензент (прізвище, ім'я, по батькові, посада, місце роботи) _____

Мартинюк Валерій Володимирович,
зав. каф. АКІТтаР

"__"_____2023 р. _____(підпис)

Завідувачу кафедри КІІС
д-р.техн.наук, проф. Говорущенко Т. О.

Коляди Дениса Олександровича
ПІБ здобувача вищої освіти

ФІТ, 3 курсу, групи ІСТс-20-1

ЗАЯВА

З правилами чинного Положення «Про систему забезпечення академічної доброчесності у Хмельницькому національному університеті» від 01.07.2022, згідно з яким виявлення плагіату є підставою для відмови в допуску кваліфікаційної роботи до захисту та застосування заходів дисциплінарної та академічної відповідальності, ознайомлений (а). Про використання програмно-технічних засобів для перевірки кваліфікаційних робіт здобувачів вищої освіти на плагіат оповіщений (а) та надаю свою згоду на обробку та збереження університетом моєї роботи в інституційному репозитарії університету.

Також надаю університету право на передачу моєї роботи для обробки та збереження в базах даних програмно-технічних засобів (Unicheck та Anti-Plagiarism) та використання роботи для виявлення плагіату в інших роботах, які перевіряються програмно-технічними засобами та користувачами, що мають доступ до цих програмно-технічних засобів, виключно в обмежених цілях для виявлення плагіату в текстах робіт.

Робота для перевірки університетом надається в друкованому та електронному варіанті. Електронна версія моєї роботи збігається (ідентична) з друкованою.

24 травня 2023 року

РІШЕННЯ ЕКСПЕРТНОЇ КОМІСІЇ

КАФЕДРИ КОМП'ЮТЕРНОЇ ІНЖЕНЕРІЇ ТА ІНФОРМАЦІЙНИХ СИСТЕМ

ПРО ДОПУСК КВАЛІФІКАЦІЙНОЇ РОБОТИ ДО ЗАХИСТУ

Підтверджуємо ознайомлення з результатом звіту подібності щодо роботи, генерованого системою виявлення текстових збігів/ідентичності/схожості:

Назва: Інформаційна система організації повсякденної діяльності користувача

Автор: Коляда Денис Олександрович

Спеціальність: 126 – Інформаційні системи та технології

Освітня програма: освітньо-професійна

Науковий керівник: Говорущенко Т.О., д.т.н, професор

Після аналізу звіту подібності зроблено такий висновок:

№	Висновок	Позначка про відповідність
1	Запозичення, виявлені в роботі, є законними і не є плагіатом. Робота приймається до захисту.	відповідає
2	Виявлені запозичення не є плагіатом, розміщені в розділах, які не описують безпосередньо авторське дослідження, але кількість цитат перевищує обсяг, виправданий поставленою метою роботи Робота приймається до захисту, але має бути відкоригована. Відкоригований варіант має бути поданий на кафедру за 2 дні до захисту, разом із заявою щодо самостійності виконання письмової роботи та ідентичності друкованої та електронної версії роботи	
3	Виявлені запозичення не є плагіатом, але частково розміщені в розділах, які описують безпосередньо авторське дослідження, а кількість цитат перевищує обсяг, виправданий поставленою метою роботи. В зв'язку з цим мета роботи та поставлені завдання не були досягнені. Робота може бути допущена до захисту (наступного року) після того як буде відкоригована та допрацьована і успішно пройде повторну перевірку на академічний плагіат.	
4	Робота містить навмисні текстові спотворення, передбачувані спроби укриття запозичень або інші прояви академічного плагіату. Робота містить фабрикацію або фальсифікацію даних. Робота не допускається до захисту.	

Підтвердження:

Запозичення, виявлені в роботі, є законними і не є плагіатом, оскільки:

1) запозичення розміщені в розділах аналізу існуючих аналогів та прототипів, які не описують безпосередньо авторське дослідження і не стосуються результатів роботи;
2) усі запозичення фрагментарні, або мають належним чином оформленні посилання;
3) найбільшу схожість встановлено з одним документом і становить вона 3.7% в частині загальноприйнятої термінології щодо автоматної моделі, UML-діаграм, життєвого циклу системи, тощо.

Сумарний обсяг всіх запозичень, визначений системою виявлення збігів/ ідентичності/схожості, складає 18.6% і адресується до 304 першоджерел, що, з урахуванням наведених обгрунтувань, відповідає характеру дослідження і свідчить на користь кваліфікаційної роботи.

Керівник роботи Т. О. Говорущенко

Гарант ОПП Є. Г. Гнатчук

Завідувач кафедри КІС Т. О. Говорущенко