МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

ДЕРЖАВНИЙ УНІВЕРСИТЕТ «ЖИТОМИРСЬКА ПОЛІТЕХНІКА»

Кафедра інженерії програмного забезпечення

**КУРСОВА РОБОТА**

(ПОЯСНЮВАЛЬНА ЗАПИСКА)

з дисципліни: «Моделювання та аналіз програмного забезпечення»:

**«Моделювання сайту »**

студента IV курсу групи ІПЗ-19-1

спеціальності 121 «Інженерія програмного забезпечення»

Євдокимова Владислава Вікторовича

(прізвище, ім’я та по-батькові)

Керівник: зав. каф. КН к.т.н. доцент Сугоняк І.І.

Дата захисту: " \_\_\_ " \_\_\_\_\_\_\_\_\_\_\_\_ 20\_\_ р.

Національна шкала \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Кількість балів: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Оцінка: ECTS \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Члени комісії \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Сугоняк І. І.

(підпис) (прізвище та ініціали)

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Власенко О.В

(підпис) (прізвище та ініціали)

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Кравченко С.М.

(підпис) (прізвище та ініціали)

Житомир – 2022

**АНОТАЦІЯ**

Пояснювальна записка до курсової роботи на тему «Розв’язання задачі лінійного програмування» складається з вступу, трьох розділів, висновків, джерел інформації та додатків.

Текстова частина викладена на 30 сторінках друкованого тексту.

Пояснювальна записка має 9 сторінок додатків. Джерела інформації містять 10 найменувань і займають 1 сторінку. В роботі наведено 11 рисунків. Загальний обсяг роботи – 39 сторінок.

У першому розділі було проаналізовано завдання та на основі цього було побудовано математичну модель задачі.

У другому розділі було проаналізовано методи розв’язання задач лінійного програмування та обрано відповідний для розв’язання поставленої задачі.

У третьому розділі проаналізовано методи реалізації програмного продукту, проаналізовані різни мови програмування, розроблено веб додаток на основі обраних алгоритмів, обчислені різні задачі та продемонстровано роботу додатку.

Висновок складається з результатів виконаної роботи вирішення задач лінійного програмування та створення програми, що обчислює відповідні задачі на основі даних введених користувачем.

У додатках частково представлений вихідний код веб додатку.

**ЗМІСТ**

[**ВСТУП** 5](#_Toc121675953)

[**1 АНАЛІЗ ВИМОГ КОРИСТУВАЧА ТА КОНЦЕПТУАЛЬНЕ МОДЕЛЮВАННЯ** 6](#_Toc121675954)

[**1.1 Технічне завдання на розробку системи** 6](#_Toc121675955)

[**1.2 Обґрунтування вибору засобів моделювання** 6](#_Toc121675956)

[**1.3 Аналіз вимог до програмного продукту** 6](#_Toc121675957)

[**Висновки до розділу 1** 6](#_Toc121675958)

[**2 РОЗРОБКА МОДЕЛІ ПРОГРАМНОГО КОМПЛЕКСУ НА ЛОГІЧНОМУ РІВНІ** 6](#_Toc121675959)

[**2.1 Алгоритм роботи та стани програмної системи** 6](#_Toc121675960)

[**2.2 Взаємодія об’єктів системи** 6](#_Toc121675961)

[**Висновки до розділу 2** 6](#_Toc121675962)

[**3**  **ФІЗИЧНА МОДЕЛЬ ТА ПРОТОТИП ПРОГРАМНОГО КОМПЛЕКСУ** 6](#_Toc121675963)

[**3.1 Взаємодія компонентів системи** 6](#_Toc121675964)

[**3.2 Генерування програмного коду для прототипу програми** 6](#_Toc121675965)

[**Висновки до розділу 3** 6](#_Toc121675966)

[**ВИСНОВКИ** 7](#_Toc121675967)

[**ДЖЕРЕЛА ІНФОРМАЦІЇ** 8](#_Toc121675968)

[**ДОДАТКИ** 9](#_Toc121675969)

# **ВСТУП**

Unified Modeling Language (UML) — уніфікована мова моделювання. Розшифруємо: modeling передбачає створення моделі, що описує об'єкт. Unified (універсальний, єдиний) — підходить для широкого класу проектованих програмних систем, різних областей додатків, типів організацій, рівнів компетентності, розмірів проектів. UML описує об'єкт в єдиному заданому синтаксисі, тому де б ви не намалювали діаграму, її правила будуть зрозумілими для всіх, хто знайомий з цією графічною мовою — навіть в іншій країні.

UML може бути застосовано на всіх етапах життєвого циклу аналізу бізнес-систем і розробки прикладних програм. Різні види діаграм які підтримуються UML, і найбагатший набір можливостей представлення певних аспектів системи робить UML універсальним засобом опису як програмних, так і ділових систем.

Основною причиною використання мови UML є спілкування розробників між собою. Крім того, UML спеціально створювалася для оптимізації процесу розробки програмних систем, що дозволяє збільшити ефективність їх реалізації у кілька разів і помітно поліпшити якість кінцевого продукту. Діаграми підвищують супровід проекту і полегшують розробку документації. та уніфікація процесу проектування.

Мета курсової роботи полягає у аналізі вимог та проблем, які можуть виникнути при розробці платформи краудфандингу.

Об‘єктом дослідження є методи та засоби проектування програмного забезпечення.

Предметом дослідження є можливості застосування CASE-засобів проектування програмного забезпечення.

**1 АНАЛІЗ ВИМОГ КОРИСТУВАЧА ТА КОНЦЕПТУАЛЬНЕ МОДЕЛЮВАННЯ**

* 1. **Технічне завдання на розробку системи**

Розробка веб-додатку краудфандингу:

1. Розробка дизайну для веб-додатку
   1. Створити графічне представлення екранів різних розмірів веб-додатку за допомогою програмного забезпечення Figma.
   2. Продемонструвати замовнику відповідний результат і внести правки, якщо це потрібно.
2. Розробка архітектури веб-додатку
   1. Провести бесіду із замовником, обговорити відповідні задачі та визначити архітектуру та відповідний підхід розробки веб-додатку.
3. Розробка backend для додатку
   1. Розробка бази даних для додатку.
   2. Створення таблиць коментарів, зборів, власників зборів, адміністраторів.
   3. Реалізація логіки за допомогою мови програмування NodeJS для відповідних запитів до бази даних.
   4. Реалізація реєстрації та авторизації користувачів.
4. Розробка frontend для додатку
   1. Створити веб-додаток за допомогою сучасного фреймворка NextJS.
   2. Реалізувати flux архітектуру для відстеження станів веб-додатку.
   3. Реалізувати підключення ендпойнтів та апі до веб-додатку.
   4. Створити web3.0 додаток.
   5. Написати смартконтракти за допомогою мови програмування Solidity, для обміну даних з віртуальною машиную Etherium.
   6. Відповідно до макету розробити основні компоненти веб-додатку та реалізувати бізнес логіку у відповідних компонентах.
5. Провести розгортання веб-додатку на хостингу.
   1. **Обґрунтування вибору засобів моделювання**

UML є мовою широкого профілю, це - відкритий стандарт , який використовує графічні позначення для створення абстрактної моделі системи , званої UML-моделлю . UML був створений для визначення, візуалізації, проектування та документування, в основному, програмних систем. UML не є мовою програмування, але на підставі UML-моделей можлива генерація коду .

UML призначена для документування архітектури системи й усіх її деталей. Крім того, це мова для вираження вимог до системи й опису тестів. Вона підходить для моделювання робіт на етапі проектування й керування версіями. Галузі використання UML як засобу моделювання й розроблення програмних систем у різних галузях (корпоративні інформаційні системи, телекомунікації, транспорт, авіація і космонавтика, наука, освіта, розподілені Web-сервіси тощо). Також Застосування UML не обмежене моделюванням ПЗ, а може застосовуватись і до не програмних систем.

Одне із завдань UML — служити засобом комунікації всередині команди та при спілкуванні з замовником. Розглянемо основні переваги UML діаграм.

Проектування. UML-діаграми стануть у пригоді при моделюванні архітектури великих проектів, в якій можна зібрати як великі, так і дрібніші деталі і намалювати каркас (схему) програми. По ньому пізніше буде будуватись код.

Реверс-інжиніринг — створення UML-моделі з існуючого коду додатку, зворотна побудова. Может застосовуватися, наприклад, на проектах підтримки, де є написаний код, але документація неповна або відсутня.

З моделей можна витягувати текстову інформацію і генерувати відносто читабельні тексти — документувати. Текст і графіка будуть доповнювати один одного.

Існує безліч додатків для створення UML діаграм, розглянемо деякі з них.

Diagrams.net — зручний сервіс для створення блок-схем, UML-діаграм, моделей бізнес-процесів онлайн. Сумісний з більшістю популярних інструментів, включаючи Google Docs, Git, Dropbox, OneDrive та інші.

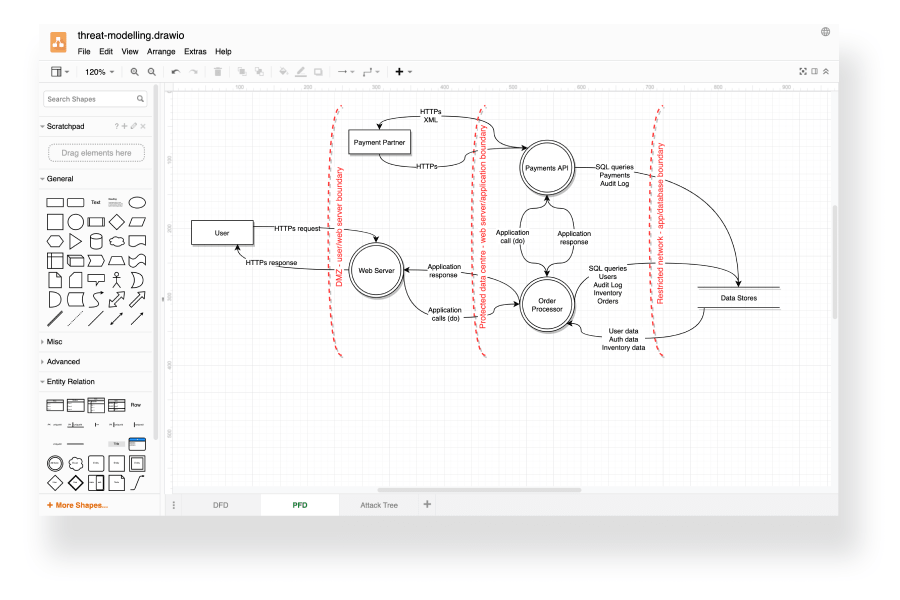


Рис. 1.1 - Інтерфейс Diagrams.net – редактора

Dbdiagram.io — додаток для побудови діаграм зв'язків для баз даних. Хороший інструмент для розробників і аналітиків.

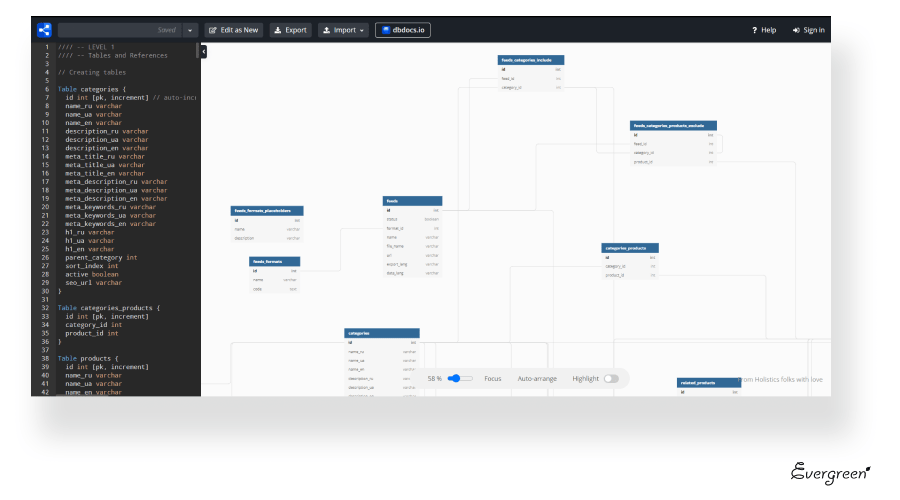


Рис. 1.2 - Інтерфейс Dbdiagram.io – редактора

StarUML – це інструмент розробки програмного забезпечення для моделювання систем з використанням уніфікованої мови моделювання, а також мови моделювання систем та класичних нотацій моделювання.

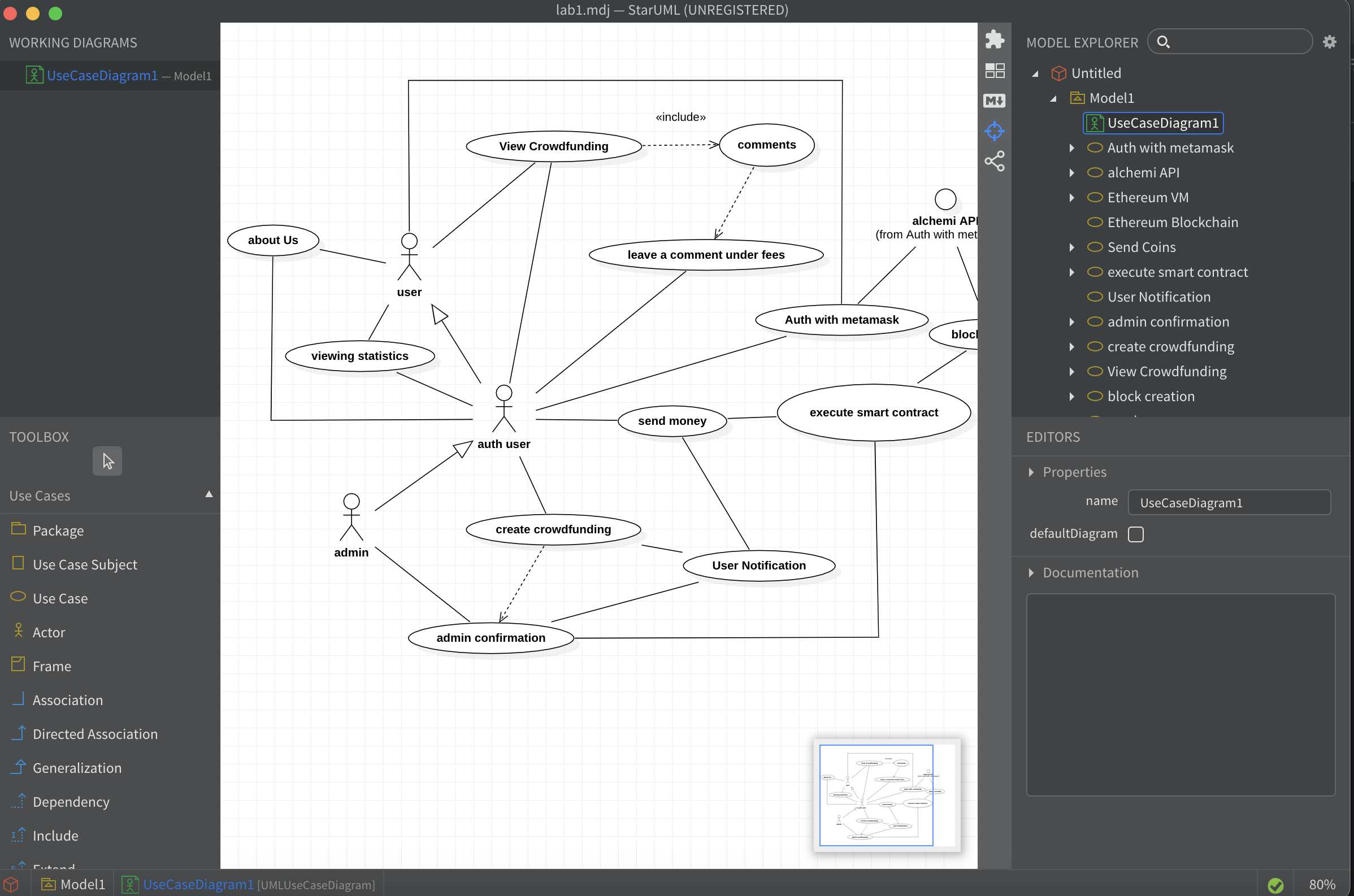


Рис. 1.3 - Інтерфейс Dbdiagram.io – редактора

StarUML – це професійний додаток, який використовують безліч великих компаній для полегшення розробки власних проектів. Це зручний мультиплатформенний додаток, який може використовувати кожен. У даного додатку інтуїтивно зрозумілий інтерфейс, що є великою перевагою перед іншими подібними додатками.

Простота використання є найбільш важливою характеристикою при розробці додатків. Безкоштовна платформа StarUML вигідно відрізняється від своїх аналогів, в тому числі вона володіє великою кількістю особливостей такими як швидкий діалог, управління за допомогою клавіатури, огляд діаграм і багато іншого.

Оскільки StarUML адаптується до користувача**,** надає максимальну ступінь адаптації середовища розробки користувача, пропонуючи налаштування параметрів, саме тому її було обрано для побудови діаграм.

* 1. **Аналіз вимог до програмного продукту**

В документі проводиться аналіз і обґрунтовується доцільність створення веб-додатку краудфандингу.

Веб-додаток краудфандингу призначений для створення зборів, щоб будь який користувач міг анонімно допомогти тій чи іншій людині, яка постраждала від війни, стихійного лиха або просто потрапила у біду.

Цілю веб-додатку є створити систему, яка буде допомагати людям. Кожен може створити збір на який користувачі, що мають активи в криптовалюті можуть переказати певну кількість коштів.

Основними перспективами даної інформаційної системи є актуальність в даний час, технології майбутнього, анонімність, можливість перегляду статистики для власників зборів.

Характеристика платформи:

Адміністратор через веб-інтерфейс буде мати можливість адмініструвати збори, а саме: перевіряти доброчесність зборів та спілкуватися з власниками зборів. Такий не великий обсяг роботи пояснюється тим, що система буде побудована на сучасних технологіях, що дозволить автоматизувати систему.

Загальний опис платформи:

* Можливість перегляду зборів.
* Можливість створення зборів.
* Збір та перегляд статистики для зборів.
* Відстеження процесу зборів.
* Можливість задати питання власнику зборів.
* Просте користування платформою.
* Забезпечення анонімності.
* Спрощення процесу домоги людям.

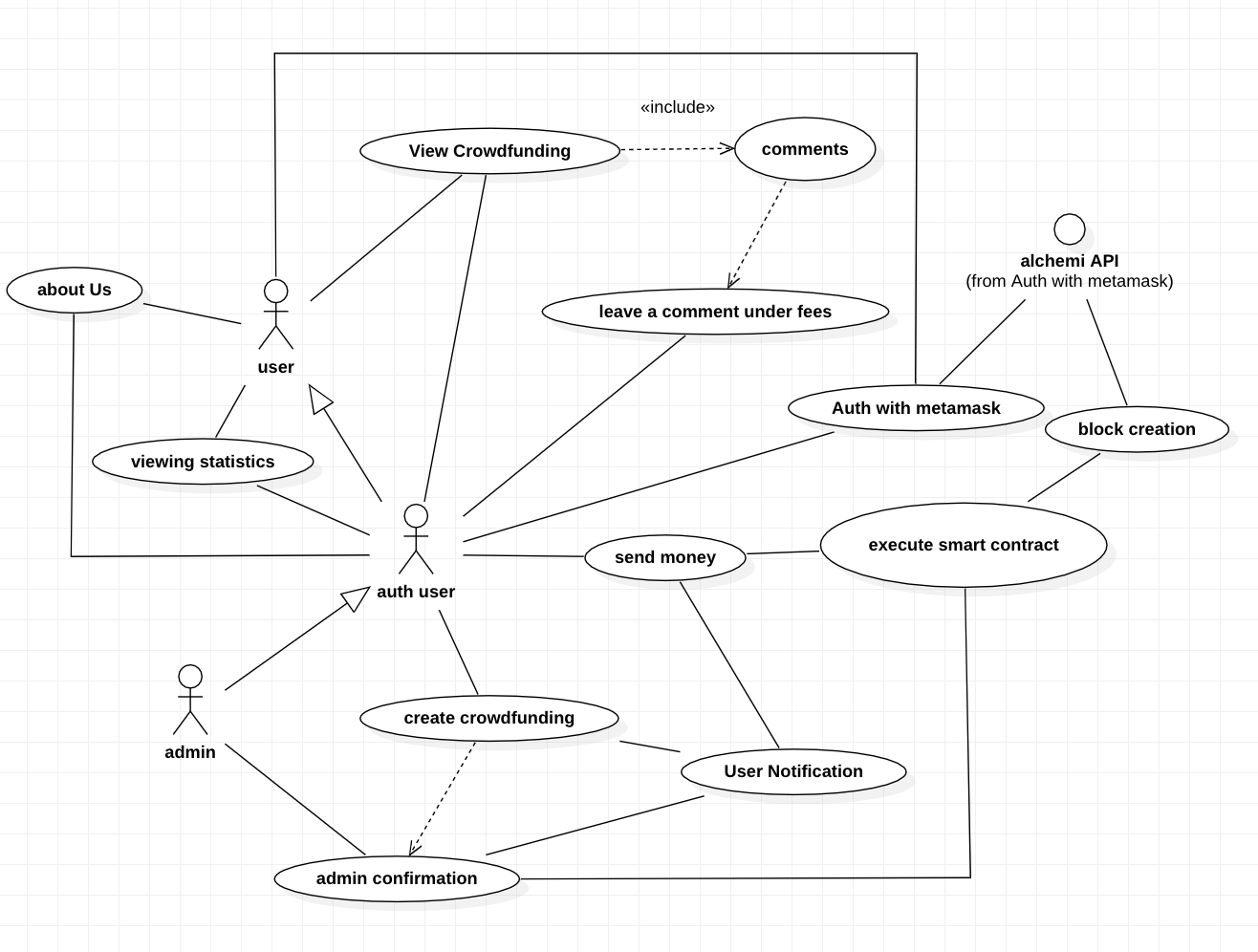


Рис. 1.4 – Діаграма варіантів використання

Вимоги користувачів:

Внутрішній користувач – адміністратор:

1. Спілкування з авторами зборів.
2. Підтвердження створення зборів.
3. Перегляд заявок на надання послуги.

**Внутрішній користувач – клієнт:**

1. Авторизація.
2. Перегляд власних зборів.
3. Перегляд статусу сотворених зборів.
4. Створення нової заявки на надання послуг.
5. Перегляд списку зборів.
6. Перегляд статистики.

**Внутрішній користувач – гість:**

1. Перегляд списку зборів.
2. Анонімна авторизація.
3. Спілкування з власниками зборів.

**Функціональні вимоги:**

1. В системі повинна бути представлення можливість авторизації користувачів, двома способами, за допомогою технологій Web 3.0 та звичайна авторизація.
2. Забезпечення анонімності звичайним користувачам.
3. Забезпечення спілкування користувачів з власниками зборів та власників зборів з адміністрацією.
4. Відображення результатів зборів.
5. Можливість збереження інформації про збори.
6. Забезпечити власникам зборів переглядати статистику.

**Системні вимоги:**

1. Вимоги до середовища виконання: платформа повинна задовільнять вимогам на пристроях, що знаходиться в наступній мінімальній комплектації:

* 2 Гб оперативної пам’яті.
* Процесор з тактовою частотою не нижче 1.4 GHz.
* Вихід в мережу інтернет.
* Будь який браузер.
* Мультиплатформенність.

1. Вимоги до серверу інформаційної системи:

• У ядрі системи повинна бути документаційно орієнтована база даних у зв’язці з Web 3.0 орієнтованим веб-додатком для забезпечення анонімної авторизації користувачів та виконання смарт контрактів.

* Підримка сервером технологій NodeJS.

**Не функціональні вимоги:**

1. Сприйняття:

* час, потрібний для освоєння веб-сайту – від 3-х до 5-ти хвилин, для досвідчених менше 2-х хвилини;
* час відповіді веб-сайту на дії користувача не повинні перевищувати 3 секунд, а для складних виконання смарт контрактів не більше – 30-ти секунд;
* інтерфейс веб-сайту повинен бути інтуїтивно зрозумілим та зручним для користувача, та не вимагати від нього додаткової підготовки.

1. Надійність:

* веб-сайт має бути доступний на протязі усієї доби.

1. Продуктивність:

* велика кількість користувачів мають змогу переглядати веб-ресурс одночасно.

## **Висновки до розділу 1**

В першому розділі було проаналізовано основні переваги використання CASE-засобів проектування програмного забезпечення, розглянуто основні додатки за допомогою яких можна створювати діаграми, та обрано найбільш зручний та багатофункціональний для розробки. Також визначено основні вимоги до розробки веб-додатку та побудовано діаграму варіантів використання.

**2 РОЗРОБКА МОДЕЛІ ПРОГРАМНОГО КОМПЛЕКСУ НА ЛОГІЧНОМУ РІВНІ**

* 1. **Алгоритм роботи та стани програмної системи**

Діаграма діяльності (англ. activity diagram) — в UML, візуальне представлення графу діяльностей. Граф діяльностей є різновидом графу станів скінченного автомату, вершинами якого є певні дії, а переходи відбуваються по завершенню дій.

Дія (англ. action) є фундаментальною одиницею визначення поведінки в специфікації. Дія отримує множину вхідних сигналів, та перетворює їх на множину вихідних сигналів. Одна із цих множин, або обидві водночас, можуть бути порожніми. Виконання дії відповідає виконанню окремої дії. Подібно до цього, виконання діяльності є виконанням окремої діяльності, буквально, включно із виконанням тих дій, що містяться в діяльності. Кожна дія в діяльності може виконуватись один, два, або більше разів під час одного виконання діяльності. Щонайменше, дії мають отримувати дані, перетворювати їх та тестувати, деякі дії можуть вимагати певної послідовності. Специфікація діяльності (на вищих рівнях сумісності) може дозволяти виконання декількох (логічних) потоків, та існування механізмів синхронізації для гарантування виконання дій у правильному порядку.

Діаграми активностей будуються з обмеженої кількості фігур, з’єднаних стрілочками.

Найважливіші типи фігур:

* скруглені прямокутники позначають дії;
* ромби позначають рішення;
* риски позначають початок (розподіл) чи кінець (об’єднання) паралельних активностей;
* чорний кружок позначає старт (початковий стан) процесу;
* чорний кружок в колі позначає кінець (кінцевий стан).

Стрілки ведуть від старту до кінця і позначають порядок в якому відбуваються активності.

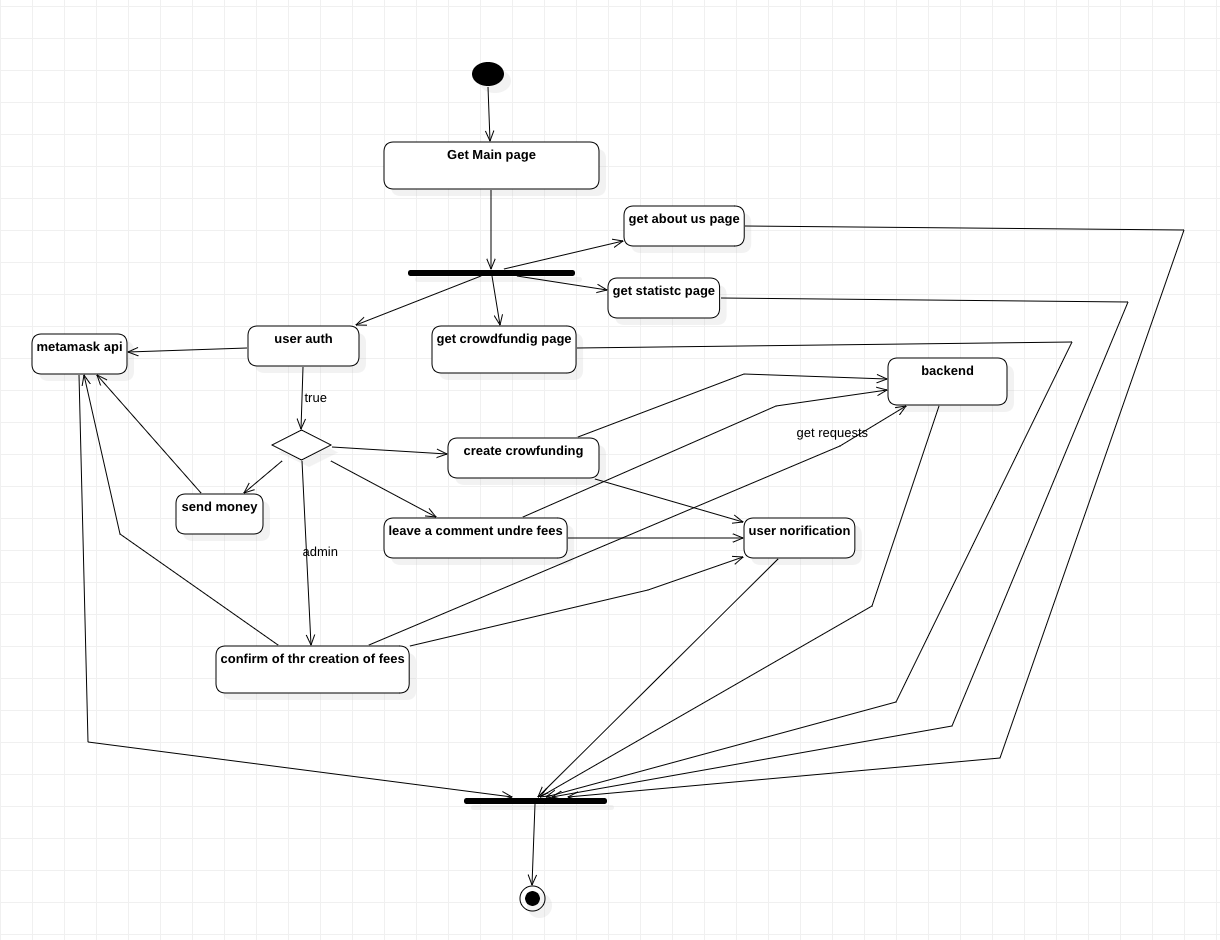


Рис. 2.1 - Діаграма активності

**Якщо авторизувався адміністратор:**

* після авторизації менеджер потрапляє в адмін панель;
* в панелі адміністратор може переглянути всі заявки і підтвердити їх, якщо вони задовольняють умови користування платформою;
* якщо потрібно адміністратор може провести бесіду із замовником для уточнення деталей.

**Якщо авторизувався клієнт:**

* клієнт має можливість переглянути список активних зборів;
* клієнт має можливість переглянути статистику по окремим зборам;
* клієнт має можливість створити нову заявку на надання послуг;
* клієнт має можливість зв’язатися з адміністрацією платформи;

**Якщо авторизувався користувач:**

* користувач має можливість переглянути активні збори;
* користувач має можливість анонімно переказати певну кількість коштів;користувач має можливість звернутися із запитанням власника збору;
  1. **Взаємодія об’єктів системи**

Оскільки в основі додатку буде лежати бібліотека ReactJS важливо зберігати ту саму реактивність.

React спрощує створення інтерактивних інтерфейсів. Потрібно лише описати, як різні частини інтерфейсу виглядають у кожному стані нашого додатку і React ефективно оновить та відрендерить лише потрібні компоненти, коли ваші дані зміняться. Це все відбувається за допомогою VirtualDOM, спочатку всі зміни вносяться у віртуальне дом дерево, після цього справжнє і віртуальне дерева порівнюються і вносяться зміни.

Декларативні інтерфейси роблять код більш передбачуваним і його набагато легше налагоджувати.

ReactJS заснований на компонентах. Можна створювати інкапсульовані компоненти, які керують власним станом, а з них будувати складні інтерфейси.

Оскільки логіка компонентів написана на JavaScript, замість шаблонів, ми з легкістю можемо передавати складні дані у додатку і зберігати стан окремо від DOM.

Щоб полегшити процес розробки та відстеження станів додатку потрібно використовувати flux аархітектуру.

React відповідає за V або View у MVC. Flux, шаблон проектування, відповідає M у MVC.

Це архітектура, відповідальна за створення шару даних у додатках JavaScript і розробку серверної сторони в веб-додатках. Flux доповнює складові компоненти виду View React, використовуючи односпрямований потік даних.

Також можна сказати, що Flux більше ніж шаблон, більше ніж фреймворк і має 4 головні компоненти: диспетчер (Dispatcher), сховище (Stores), подання (Views) (React компонент), дія (Action)

Flux не схожий на звичний MVС, який ми звикли бачити в інших фреймворках. Там є Контролер, але здебільшого це контролер, який відповідає за Views. Views знаходяться вгорі ієрархії, і вони передають функціонал та дані елементам-нащадкам.

Flux слідує концепції односпрямованого потоку даних, що робить його простим для пошуку помилок. Дані проходять через прямий потік програми. React і Flux — на даний момент два найпопулярніші фреймворки, які використовують принцип односпрямованого потоку даних.

У той час, як React використовує віртуальний DOM для відображення змін, Flux робить це трохи інакше. У Flux, взаємодія з інтерфейсом користувача викличе ряд дій, які можуть змінити дані програми.

Flux допомагає зробити код більш передбачуваним порівняно з МVC фреймворками. Розробники можуть створювати програми, не турбуючись про складні взаємодії між джерелами даних.

Flux вигідно відрізняється більш організованим потоком даних – односпрямованим. Те, що він односпрямований, головна особливість Flux. Ці дії поширюються на нову систему щодо взаємодії з користувачем.

Однонаправлений потік у Flux - у той час як MVC двонаправлений у своєму потоці, у Flux всі зміни проходять через один напрямок, через Dispatcher даних. Store не може бути змінено само по собі, і той самий принцип працює для інших Actions. Зміни, які потрібно внести, повинні пройти через Dispatcher, через Actions.

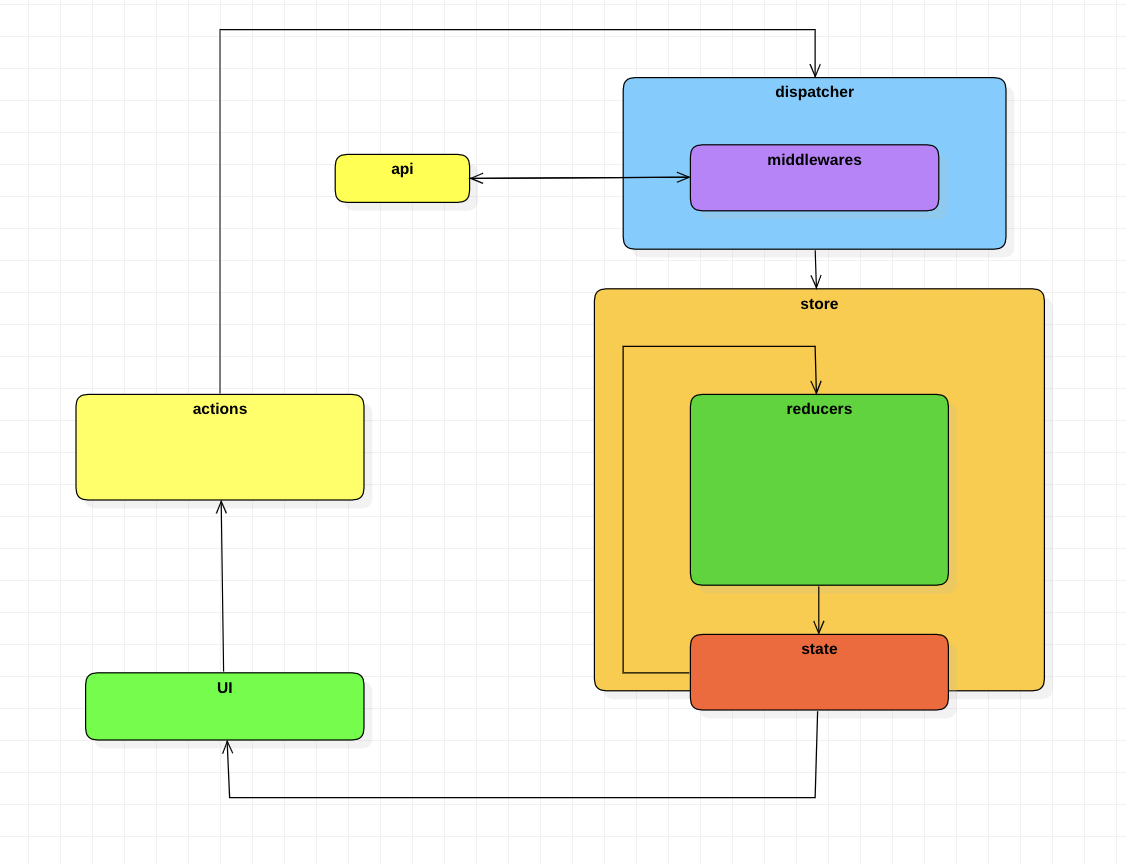


Рис. 2.2 – Діаграма роботи flux архітектури

* 1. **Комунікації і послідовність взаємодії об’єктів системи**

Діаграма послідовності — відображає взаємодії об'єктів впорядкованих за часом. Зокрема, такі діаграми відображають задіяні об'єкти та послідовність відправлених повідомлень.

Іншими словами, діаграма послідовностей відображає часові особливості передачі і прийому повідомлень об'єктами.

Діаграми послідовностей можна використовувати для уточнення діаграм прецедентів, більш детального опису логіки сценаріїв використання. Це відмінний засіб документування проекту з точки зору сценаріїв використання. Діаграми послідовностей зазвичай містять об'єкти, які взаємодіють у рамках сценарію, повідомлення, якими вони обмінюються, і які повертаються результати, які пов'язані з повідомленнями.

На діаграмі послідовності зображаються тільки ті об'єкти, які безпосередньо беруть участь у взаємодії.

Лінія життя об'єкта зображується пунктирною вертикальною лінією, асоційованою з єдиним об'єктом на діаграмі послідовності. Лінія життя служить для позначення періоду часу, протягом якого об'єкт існує в системі і, отже, може потенційно брати участь у всіх її взаємодіях. Якщо об'єкт існує в системі постійно, то і його лінія життя повинна продовжуватися по всій площині діаграми послідовності від самої верхньої її частини до самої нижньої.

У процесі функціонування об'єктно-орієнтованих систем одні об'єкти можуть перебувати в активному стані, безпосередньо виконуючи певні дії, або стані пасивного очікування повідомлень від інших об'єктів. Щоб явно виділити подібну активність об'єктів, в мові UML застосовується спеціальне поняття, що отримало назву фокуса управління . Фокус управління зображується у формі витягнутого вузького прямокутника, верхня сторона якого позначає початок отримання фокусу управління об'єкта, а його нижня сторона - закінчення фокусу управління. Прямокутник розташовується нижче позначення відповідного об'єкта і може замінювати його лінію життя, якщо на всьому її протязі він є активним.

В UML кожна взаємодія описується сукупністю повідомлень, якими ті об’єкти, що беруть участь у ньому обмінюються між собою. Повідомлення є закінченим фрагментом інформації, який відправляється одним об'єктом іншому. Прийом повідомлення ініціює виконання певних дій, спрямованих на вирішення окремого завдання тим об'єктом, якому це повідомлення відправлено.

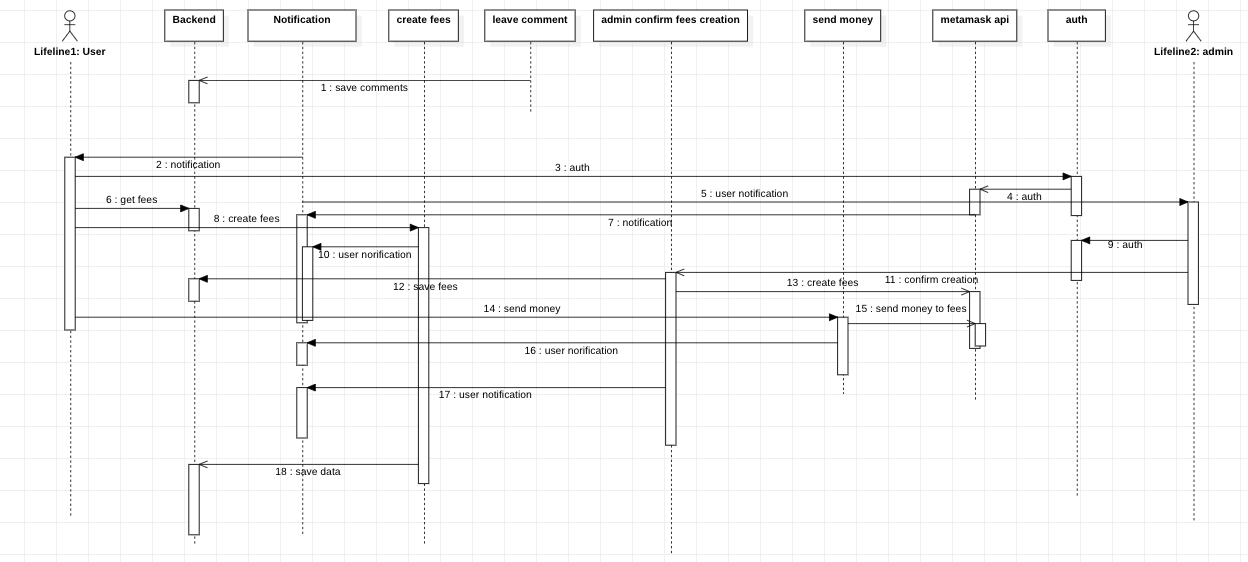


Рис. 2.3. Діаграмапослідовності

## **Висновки до розділу 2**

В другому розділі було розглянуто способи побудови алгоритму роботи програми та розроблено діаграму активності, також було визначено можливості користувачів залежно від ролі. Після цього було розглянуто патерн розробки веб-додатку, який дозволяє відслідковувати стан додатку весь час, що дозволяє зробити код більш передбачуваним. Також було розглянуто діаграму послідовності, яка відображає взаємодію об'єктів впорядкованих за часом, та було побудовано відповідну діаграму для додатку.

# **3 ФІЗИЧНА МОДЕЛЬ ТА ПРОТОТИП ПРОГРАМНОГО КОМПЛЕКСУ**

# **3.1 Взаємодія компонентів системи**

# **3.2 Генерування програмного коду для прототипу програми**

# **Висновки до розділу 3**

# **ВИСНОВКИ**

# **ДЖЕРЕЛА ІНФОРМАЦІЇ**

# **ДОДАТКИ**

Додаток А