МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Тернопільський національний економічний університет
Факультет комп'ютерних інформаційних технологій
Кафедра комп'ютерних наук

ВОЗНЮК Олександр Вікторович

Web-орієнтована програмна система для управління стартапами/ Web-oriented software system for managing startups

напрям підготовки: 6.050103 - Програмна інженерія
фахове спрямування - Програмне забезпечення систем

Бакалаврська дипломна робота

Виконав студент групи ПЗС-41
О. В. Вознюк

_____

Науковий керівник:
викладач КРЕПИЧ С.Я.

_____

Бакалаврську дипломну роботу
допущено до захисту:

"___" _____ 20___ р.

Завідувач кафедри
_____ **А. В. Пукас**

ТЕРНОПІЛЬ - 2016

Резюме

Дипломна робота містить 101 сторінка, 16 таблиць, 44 рисунки, список використаних джерел із 20 найменувань та 2 додатка.

Метою дипломної роботи є розробка веб-орієнтованої системи управління стартапами.

Об'єктом дослідження є процес управління стартапами.

Предмет досліджень – застосування сучасних технологій створення веб-орієнтованих додатків для розробки інтернет-сервісу управління стартапами. Методи розробки базуються на технології PHP, сервер бази даних MySQL і веб-сервер Open Server.

Одержані результати полягають в розробці web-ресурсу для управління стартапами.

Ключові слова: web-додаток, управління стартапами.

Summary

Theis containts 101 pages, 16 tables, 44 figures, list of source with 20 titles, 2 application.

The aim of the theis is develop web – oriented software system managing startups.

Object of research is the process managing sturtups.

The subject of research is application of modern technologies create Web-oriented applications for the development of Internet services management startup. The methods of making based on technology PHP, MySQL database server and Web server Open Server.

The resulting is creating a web-service for managing sturtups.

Keywords web-aplication, managing sturtups.

Зміст

ВСТУП

Актуальність дипломної роботи полягає в тому, що, як і на сьогоднішній день, так і надалі, створення нових стартапів є актуальним. Розробка інтернет-сервісу, який дозволяє не виходячи з дому, чи робочого місця створювати, управляти власними стартапами та командою розробників є дуже зручним сервісом, тому що, де б ми не були, ми завжди зможемо працювати над нашим проектом. Для цього нам необхідний лише доступ до мережі інтернет. Однак вагомим мінусом таких систем є те, що спілкування між учасниками команди, відбувається завжди на відстані і в різний час, тому в них немає можливості особисто при зустрічі обговорити всі деталі роботи над проектом. Тому виникає проблема організації роботи команди над проектом.

На організацію роботи потрібно багато часу , щоб зібрати команду в один час, для обговорення всіх питань щодо проекту . Тому розробка веб-сайту, який би надавав можливість швидко організувати роботу усієї команди, є актуальною задачею.

Мета роботи полягає в дослідженні та детальному аналізі уже існуючих інтернет-сервісів управління стартапами (якщо такі є наявні) та сайтів, які дозволяють розробляти власні проекти, і розробка власного інтернет-сервісу, який дозволив би створювати та управляти проектами та командою.

Отож, спираючись на мету, можна виділи такі основні задачі, які ставляться до розроблюваного продукту:

1. Дослідити існуючі аналоги управління стартапами.

2. На основі аналізу розробити функціональні та не функціональні вимоги до розроблюваного продукту.

3. Розробити модель бази даних розроблюваного продукту.

4. Обрати необхідні технології для реалізації обраного функціоналу.

5. Розробка веб-орієнтованої системи управління стартапами згідно з виконаних задач, описаних в пунктах 1-4.

Об’єкт дослідження – процес управління стартапами.

Предмет досліджень – застосування сучасних технологій створення веб-орієнтованих додатків для розробки інтернет-сервісу управління стартапами.

Під час створення системи використовуватимемо наступні технології необхідні для розробки інтернет-сервісів:

Php – скриптова мова програмування, була створена для генерації HTML -сторінок на стороні веб-сервера. PHP є однією з найпоширеніших мов, що використовуються у сфері веб-розробок

JavaScript – скриптова мова, призначена для створення інтерактивних веб-сторінок, не потребує компіляції.

Sql – мова, яка дає можливість створювати та працювати з базами даних, котрі є набором зв’язаної інформації, що зберігається в таблицях

Практична цінність розроблюваного інтернет-сервісу полягає в швидкому та зручному управлінні стартапами.

РОЗДІЛ 1 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ

1.1 Коротка характеристика об’єкту управління

Основним напрямом діяльності об’єкту управління стартапами є створення, пошук та реалізація проектів різного типу. За допомогою даної системи користувачі мають можливість створювати проекти, команди, власні компанії.

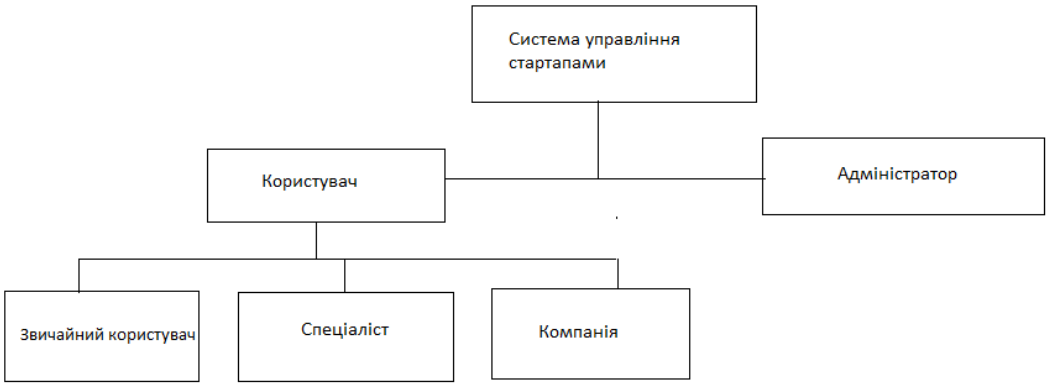


Рисунок 1.1 - Схема організаційної структури управління підприємством

В наш час є дуже багато сайтів, де ми можемо придбати певну продукцію (сайт, програму) за відповідну суму грошей. Однак багато людей не можуть собі дозволити витрачати великі суми на реалізацію цих проектів.Тому доцільно створити систему, яка надасть нам можливість реалізувати свої ідеї без фінансових вливань з нашої сторони, а навпаки зібрати кошти на реалізацію та подальший розвиток даного проекту.

Людина, яка створила проект, автоматично стає адміністратором проекту.

Адміністратор – це людина, яка створила проект, займається пошуком команди, фінансування. Також ця людина може назначати на по саду адміністратора учасників проекту.

Існує три типи користувачів:

- Звичайний користувач
- Спеціаліст
- Компанія

Якщо користувач має цікаву ідею, але не має достатньо знань в певній сфері для її реалізації, він може зареєструватися як звичайний користувач, створити проект, описати свою ідею і всі зацікавлені користувачі зможуть зробити свій внесок у це проект, якщо це спеціаліст, та допомогти з розробкою, якщо компанія то фінансова допомога проекту, або викупити проект повністю на певних умовах, про які домовляються компанія і користувач, що створив даний проект.

Є тип користувачів “спеціалісти”, які мають знання в конкретній області, а саме в програмуванні, але вони не можуть себе реалізувати. Більшість з яких працює на не цікавій для них роботі, або взагалі фрілансерами і давно вже хотіли створити свій проект, але не мали власної ідеї і тому для такого типу користувачів є “звичайні користувачі”. Спеціаліст підбирає для себе цікавий проект і пропонує свої послуги, якщо “звичайний користувач” не проти його допомоги, вони домовляються між собою на яких умовах вони будуть співпрацювати і розпочинають розробку.

Компанія – юридична особа, що ставить собі за мету знайти спеціаліста, або проект для подальшої роботи з ними. Головною метою компанії є отримання прибутку.

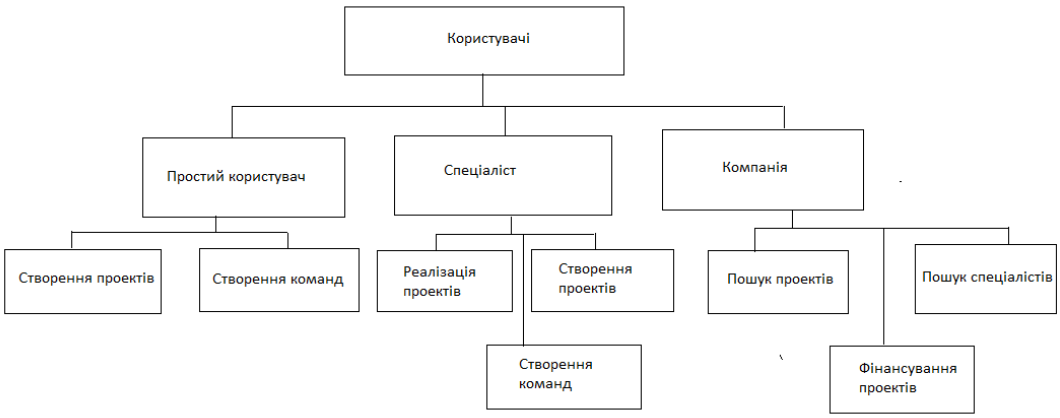


Рисунок 1.2 - Схема організаційної структури управління підприємств

1.2 Опис предметної області

Для організації роботи системи управління стартапами необхідне виконання таких основних бізнес-процесів (рисунок 1.3):

- процесу формування календарного плану;
- процесу управління командою;
- процесу управління стартапом;
- процесу фінансування стартапу;
- процесу управління зустічами.

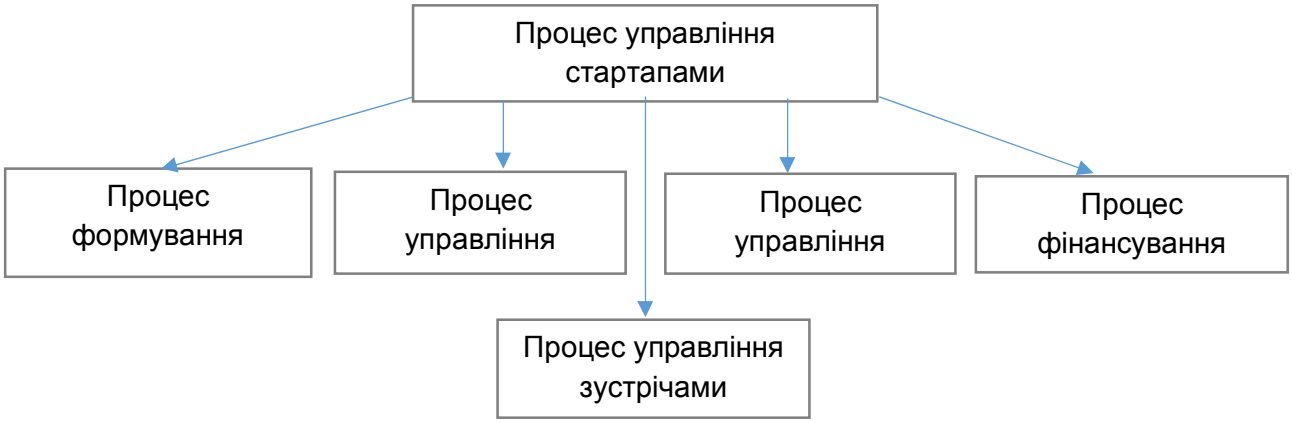


Рисунок 1.3 - Діаграма дерево функцій

Розглянемо детальніше кожен з вище представлених бізнес-процесів. На рисунку 1.4 бачимо діаграму функцій процесу формування календарного плану.

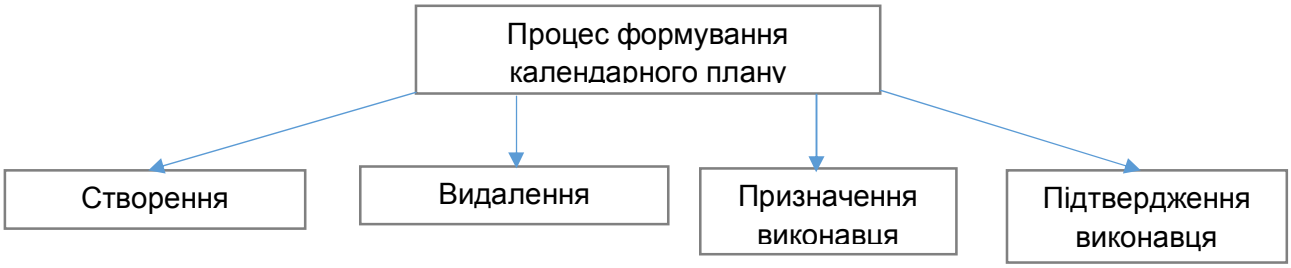


Рисунок 1.4 – Діаграма функцій процесу формування календарного плану

У кожному проекті є завдання, терміни виконання цих завдань та відповідальні за їх виконання, тому для зручності створення нових проектів буде використовуватися процес формування календарного плану. Цей процес буде поділятися на такі етапи:

- створення завдання (вводиться назва завдання і його короткий опис);
- видалення завдання (видалення уже створених завдань);
- призначення виконавця (вибирається зі списку учасників проекту).
- підтвердження виконавця (керівник проекту затверджує виконання завдання за конкретним користувачем після його згоди).

Характеристику бізнес-процесу формування календарного плану наведено в таблиці 1.1

Таблиця 1.1

Характеристика бізнес-процесу формування календарного плану

Назва характеристики	Значення характеристики
Ім'я бізнес-процесу	Формування календарного плану
Основні учасники	Керівник
Вхідна подія	Запит на створення завдання
Вхідні документи	-
Вихідна подія	Повідомлення про статус завдання (помилка, завдання створене)
Вихідні документи	-
Клієнт бізнес-процесу	Процес управління командою, процес управління стартапом, процес управління зустрічами

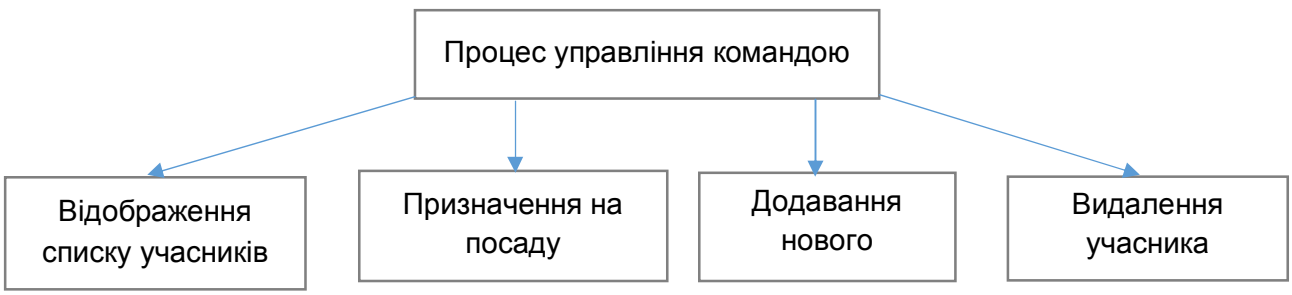


Рисунок 1.5 – Діаграма функцій процесу управління командою

В більшості випадків для створення проекту потрібні люди зі спеціальними навиками в різних областях (дизайнери, програмісти, тестери тощо). Разом вони будуть утворювати команду, яка розроблятиме конкретний продукт, тому необхідно організувати процес управління командою . Цей процес буде поділятися на такі етапи:

- відображення списку учасників (всі користувачі, що приймають участь у розробці проекту);

- призначення на посаду (призначати на посаду може лише керівник проекту);

- додавання нового учасника (додаванням нового учасника займається керівник проекту);

- видалення учасника (видаляти учасників може лише керівник проекту).

Характеристику бізнес-процесу формування календарного плану наведено в таблиці 1.2

Таблиця 1.2.

Характеристика бізнес-процесу управління командою

Назва характеристики	Значення характеристики
Ім'я бізнес-процесу	Управління командою
Основні учасники	Керівник, учасник команди
Вхідна подія	Запит на управління учасником
Вхідні документи	-
Вихідна подія	Повідомлення про статус запиту (помилка)
Вихідні документи	-
Клієнт бізнес-процесу	процесу формування календарного плану, процесу управління стартапом, процесу управління зустічами.

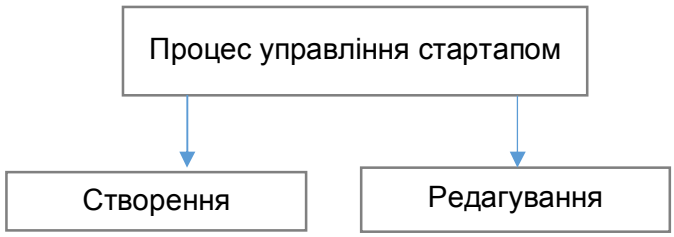


Рисунок 1.6 – Діаграма функцій процесу управління стартапом

Для створення нового проекту необхідно його описати, вказати учасників, які необхідні для його реалізації. Ці дані можна змінювати під час розробки проекту, для цього був реалізований процес управління стартапом. Цей процес буде поділятися на такі етапи:

- створення стартапу (користувач що створює новий проект, автоматично стає керівником, при створенні він вказує назву проекту, короткий опис);

- редагування (редагуванням займається керівник проекту).

Характеристику бізнес-процесу управління стартапом наведено в таблиці 1.3

Таблиця 1.3

Характеристика бізнес-процесу управління стартапом

Назва характеристики	Значення характеристики
Ім'я бізнес-процесу	Управління стартапом
Основні учасники	Керівник
Вхідна подія	Запит на створення стартапа
Вхідні документи	-
Вихідна подія	Повідомлення про статус стартапа (помилка, проект створений)
Вихідні документи	-
Клієнт бізнес-процесу	Процес формування календарного плану,процесу управління командою, процес управління зустічами.

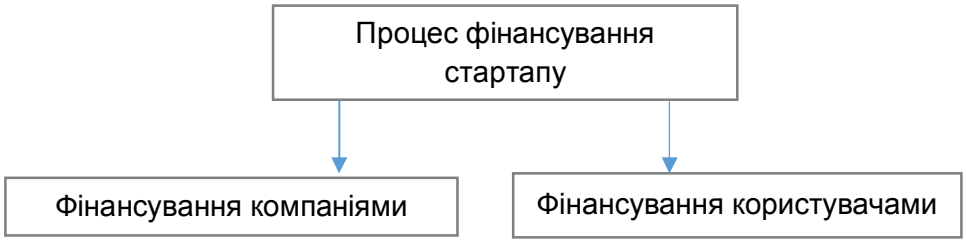


Рисунок 1.7 – Діаграма функцій процесу фінансування стартапу

Кожен проект потребує фінансування. Фінансування може відбуватися з двох сторін: компаніями та звичайними користувачами. Для зручності був розроблений процес фінансування стартапу. Цей процес буде поділятися на такі етапи:

- фінансування компаніями;
- фінансування користувачами (звичайні користувачі фінансують проект).

Характеристику бізнес-процесу формування календарного плану наведено в таблиці 1.4

Таблиця 1.4.

Характеристика бізнес-процесу фінансування стартапа

Назва характеристики	Значення характеристики
Ім'я бізнес-процесу	Фінансування стартапа
Основні учасники	Керівник, простий користувач, компанія
Вхідна подія	Запит на можливість фінансування
Вхідні документи	-
Вихідна подія	Повідомлення про статус фінансування (помилка, дозволено)
Вихідні документи	-
Клієнт бізнес-процесу	Процес управління стартапом

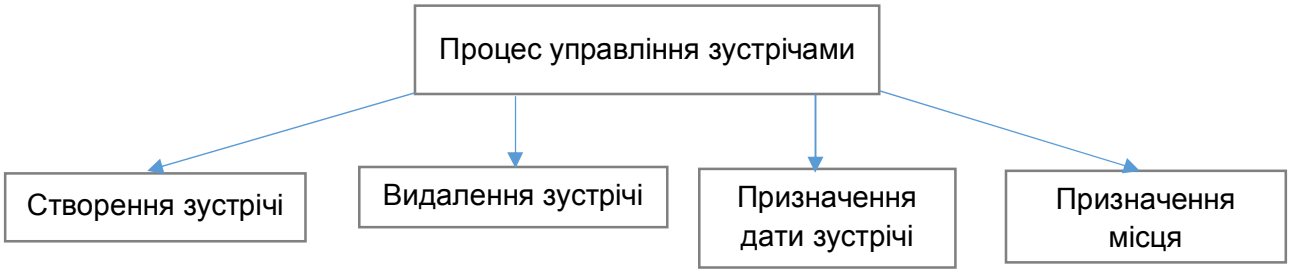


Рисунок 1.8 – Діаграма функцій процесу управління зустрічами

Спілкування через інтернет в наш час є дуже зручним, але все таки він не може замінити спілкування “в живу”. Тому за допомогою процесу управління зустрічами, учасники команди можуть домовитися про зустріч, де вони зможуть обговорити всі питання стосовно проекту, який вони розроблюють. Цей процес буде поділятися на такі етапи:

- створення зустрічі (створювати зустрічі може лише керівник);
- видалення зустрічі (видаляти зустрічі може лише керівник проекту);
- призначення дати зустрічі (встановленням дати займається керівник проекту);
- призначення місця проведення (місце проведення вирішує керівник проекту).

Характеристику бізнес-процесу управління зустрічами наведено в таблиці 1.5

Таблиця 1.5

Характеристика бізнес-процесу управління зустрічами

Назва характеристики	Значення характеристики
Ім'я бізнес-процесу	Управління зустрічами
Основні учасники	Керівник
Вхідна подія	Запит на створення зустрічі

Вхідні документи	-
Вихідна подія	Повідомлення про статус зустрічі (помилка, зустріч створена)

Продовження таблиці 1.5

Вихідні документи	-
Клієнт бізнес-процесу	Процес управління командою;

1.3 Огляд і аналіз існуючих аналогів

Indiegogo — це сайт для фінансування проектів за схемою краудфандингу. Сервіс заснований в 2008 році. Головний офіс знаходиться в Сан-Франциско, Каліфорнія. Близько дев'яти мільйонів людей з усього світу відвідують сайт щомісяця.

Indiegogo є одним з найпопулярніших веб-сайтів краудфандингу в Америці. Сайт дозволяє кожному зібрати гроші для власної ідеї.

Даний сайт дозволяє користувачам створити сторінку для фінансування свого проекту, налаштувати обліковий запис PayPal, скласти список "пільг" для різних рівнів вкладень. Користувачі мають можливість публікувати свої проекти через соціальні мережі, такі як Facebook, Twitter та аналогічні платформи.

Сайт отримує 4% комісії з компаній що досягли своєї цілі. Для кампаній, що не змогли зібрати вказану суму, користувачі мають можливість безкоштовного відшкодування всіх грошей їх вкладникам, або можуть зберегти всі зібрані гроші, але з комісією у 9%.

Indiegogo виплачує кошти негайно, як тільки внески збираються на PayPal рахунку користувача. Станом на січень 2014 року, було проведено більше 200 тисяч рекламних компаній та зібрано «мільйони доларів»

Основним функціоналом на Indiegogo є створення проекту, який потім зможе фінансувати велика аудиторія людей, якщо цей проект їх зацікавить. При створенні проекту, користувач повинен вказати наступні пункт:

- суму, яку необхідно зібрати для створення даної ідей;
- назву проекту;
- зображення проекту;

- вибрати категорію, до якої буде відноситися даний проект:

а) Тварини,

б) Святкування,

в) Навчання,

г) Медицина,

д) Спорт,

е) Волонтерство;

- вказати місто проживання

- описати даний проект (Розкати, чому потрібні кошти, на що саме. Що спонукало вас , щоб розпочати збір коштів? Як люди можуть допомогти?);

- добавити фото;

- вказати одержувача коштів. (Можна вказати що це ви, або вказати іншу особу).

Після заповнення всіх пунктів, потрібно натиснути на кнопку створити, після чого створиться сторінка з вашим проектом і буде доступна всім користувачам для фінансування.

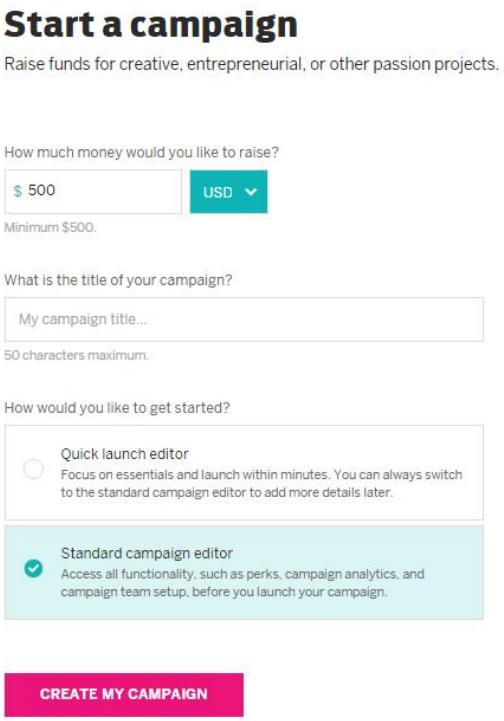


Рисунок 1.9 - Створення проекту

Під час створення нового проекту на Indiegogo потрібно вказати скільки грошей необхідно для створення даного проекту, назву проекту .

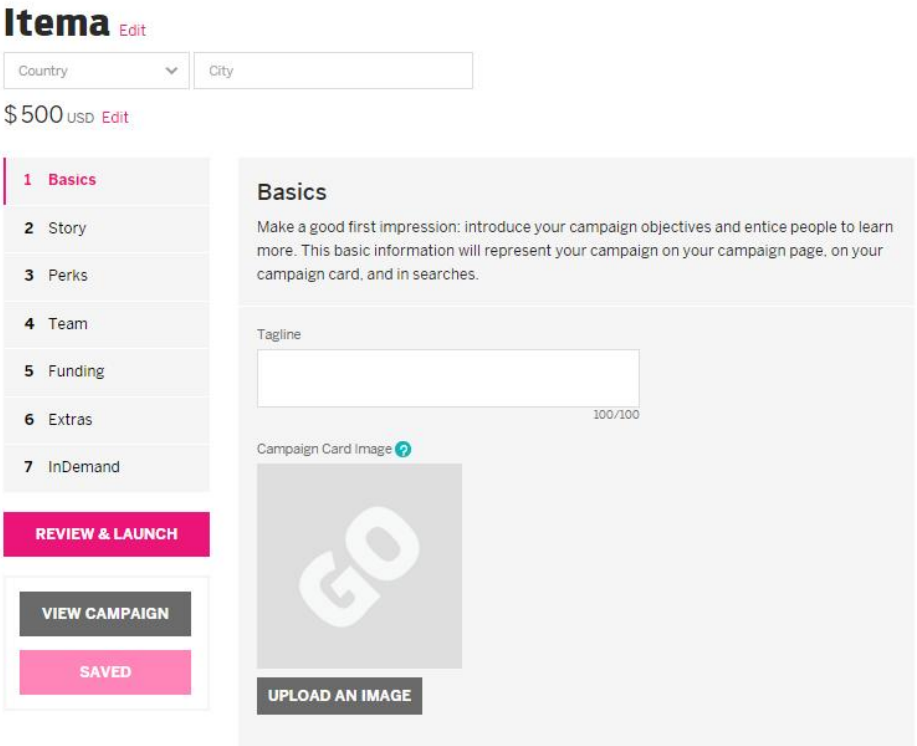


Рисунок 1.10 - Інформація про проект

В формі інформація про проект необхідно заповнити всі поля, описати проект так, щоб зацікавити користувачів, щоб вони захотіли вкласти гроші у ваш проект.

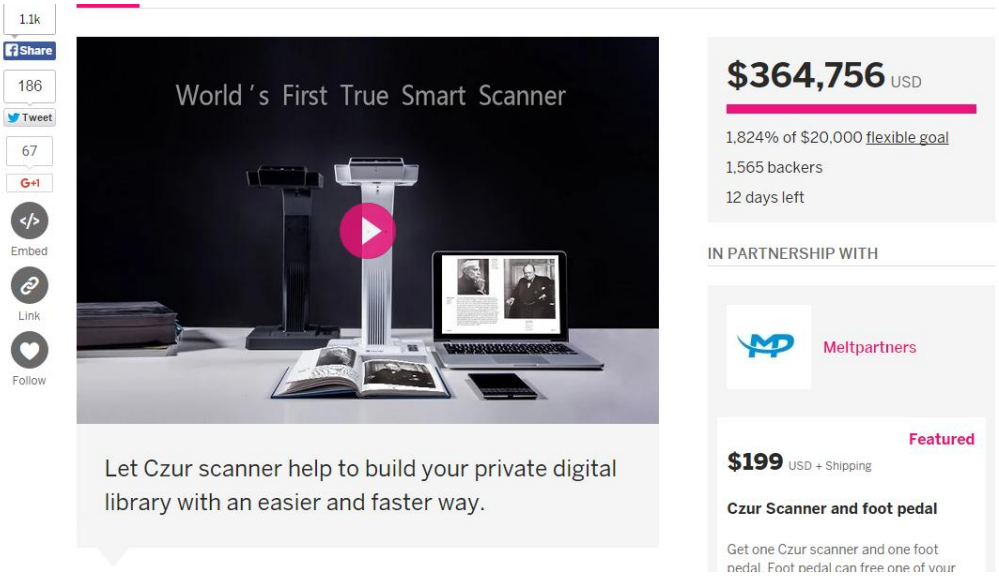


Рисунок 1.11 - Вигляд проекту після створення

Після створення проекту ми можемо бачити, як бачать наш проект всі користувачі і скільки грошей проект отримав.

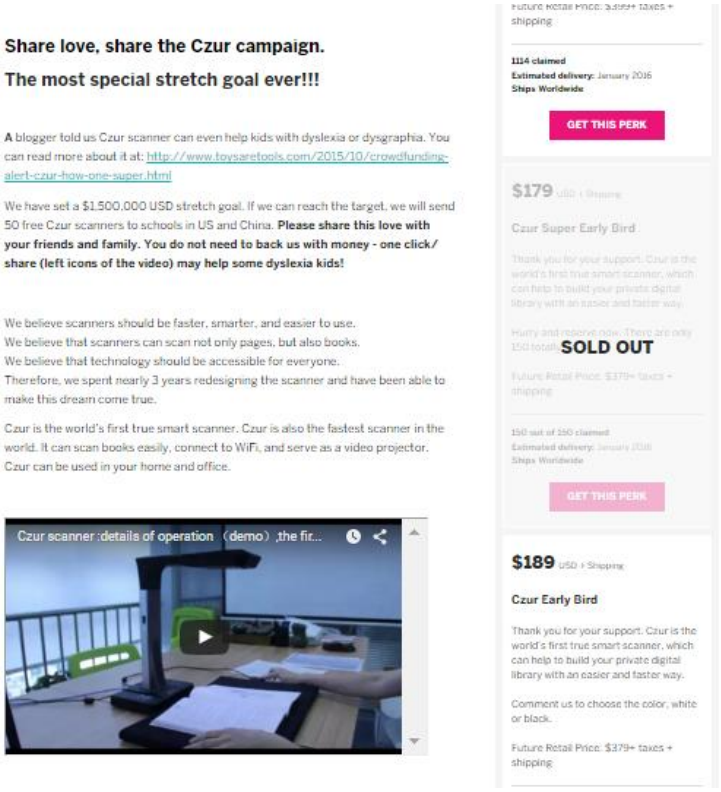


Рисунок 1.12 - Опис проекту

В описі проекту відображається вся інформація, яку ми вводили під час створення проекту, в любий момент ми можемо її змінити.

Teamfinding - це “добровільний” соціальний сервіс, орієнтований на тих користувачів, які хочуть свої ідеї і тих, хто хоче приймати участь в реалізації таких ідей.

Teamfinding вирішує дві проблеми**.**

Перша**:** проблема людей, які хочуть прийняти участь в реалізації цікавих проектів, приймати участь на зустрічах на різну тематику,і в цілому, десь застосувати свої навики та досвід *(крім своєї постійної роботи)* – тобто самореалізуватися, але пошук таких проектів і однодумців важкий і не цікавий процес.

Teamfinding вирішує дану проблему,і рішення заключається в можливості швидкого пошуку цікавих проектів, груп, зустрічей, в які можна подати заявку на участь, познайомитися з новими людьми і отримати безцінний досвід.

Друга**:** проблема тих, хто хоче реалізувати дані ідеї, але зустрічаєтьтся з тим, що в певній області вони не достатньо кваліфіковані, мають малий досвід або просто не вистачає часу реалізувати самостійно.

Рішення заключається в можливості створити і стати координатором проекту / зустрічі /спільноти і знайти людей, яким буде цікаво приймати участь у проекті. Можливість знайти і скоординувати навики і досвід спеціалістів для реалізації ідей.

Участь в проектах добровільна, про компенсацію в тому чи іншому проекті можна домовлятися лише з координатором проекту.

Teamfinding – для створення нової ідеї і пошуку людей для її реалізації необхідно вказати наступні пункти:

- назва ідеї;
- опис ідеї;
- веб-сайт ;

- тип ідеї;
- проект;
- стартап;
- знімальна група;
- об’єднання;
- спільнота;
- зустріч;
- стадію виконання;
- ідея;
- задокументована ідея;
- проектування;
- розробка;
- тестування;
- реалізована;
- пошук інвесторів;
- розширення команди;
- пошук команли;
- роль людини, що створила ідею.

Після того ,як будуть вказані всі пункти, буде створена ідея, до якої можна буде запрошувати людей та приймати зацікавлених осіб до себе в команду.

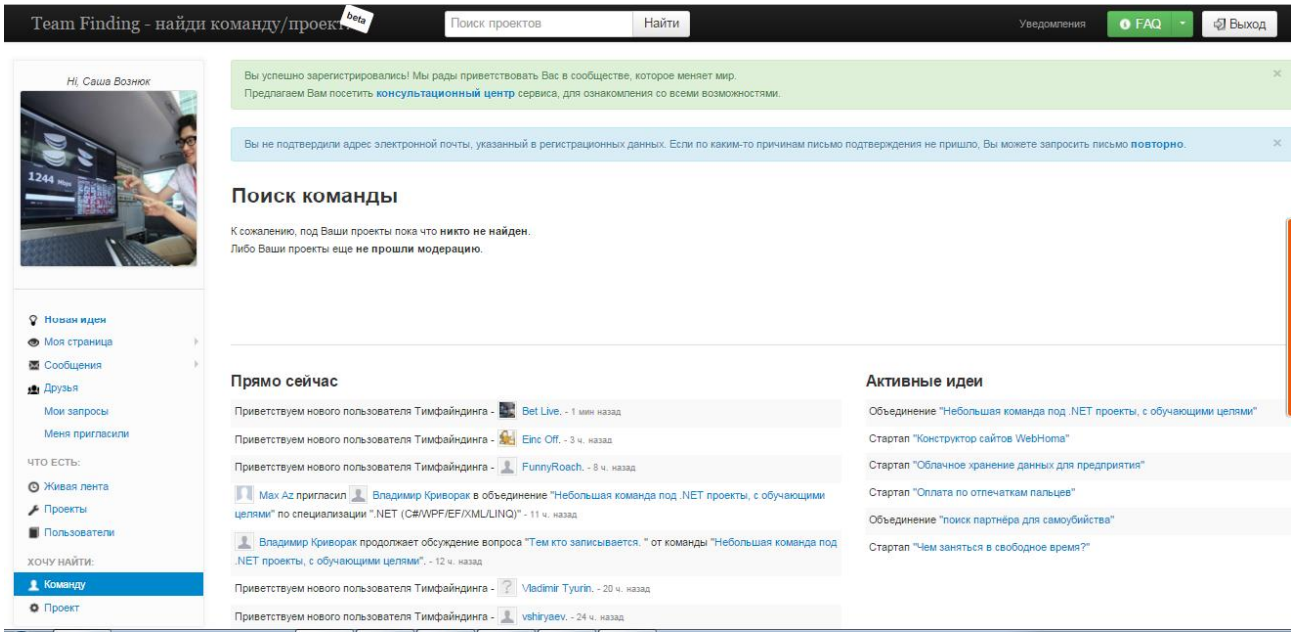


Рисунок 1.13 - Головна сторінка користувача

На головній сторінці користувача ми бачимо всі новини сайту (нових зареєстрованих користувачів, створені проекти, нових учасників проекту тощо), активні дії (як розроблюються проекти).

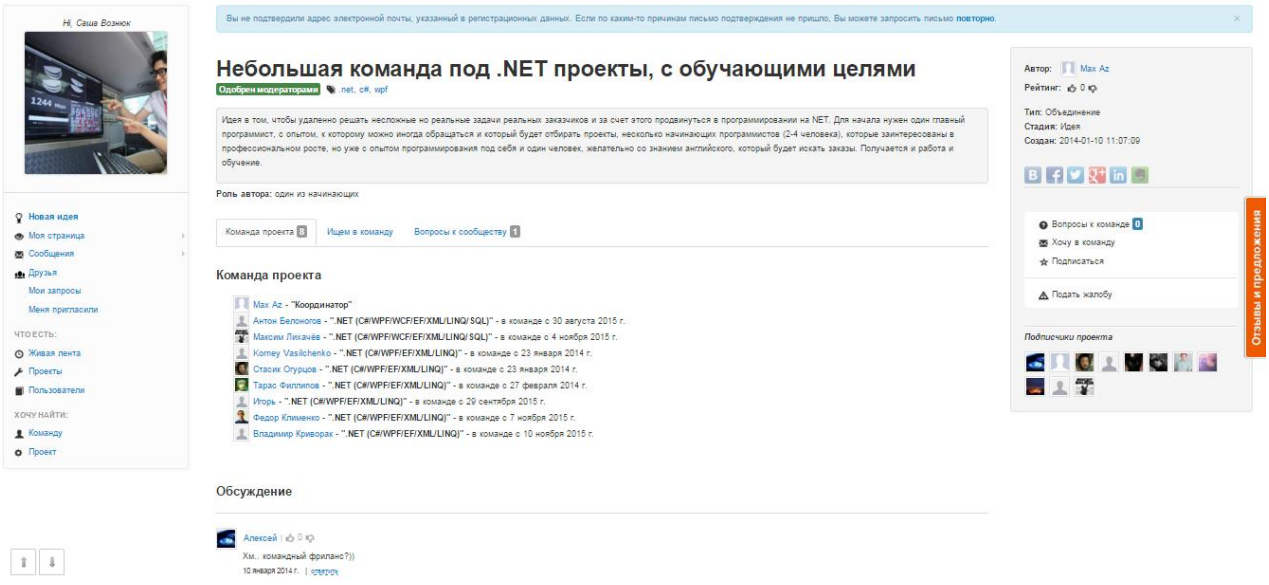


Рисунок 1.14 - Вигляд створеного проекту

Після створеного проекту ми бачимо його сторінку, де відображається інформація про проект, короткий опис, хто створив даний проект, список учасників та інформація про проект.

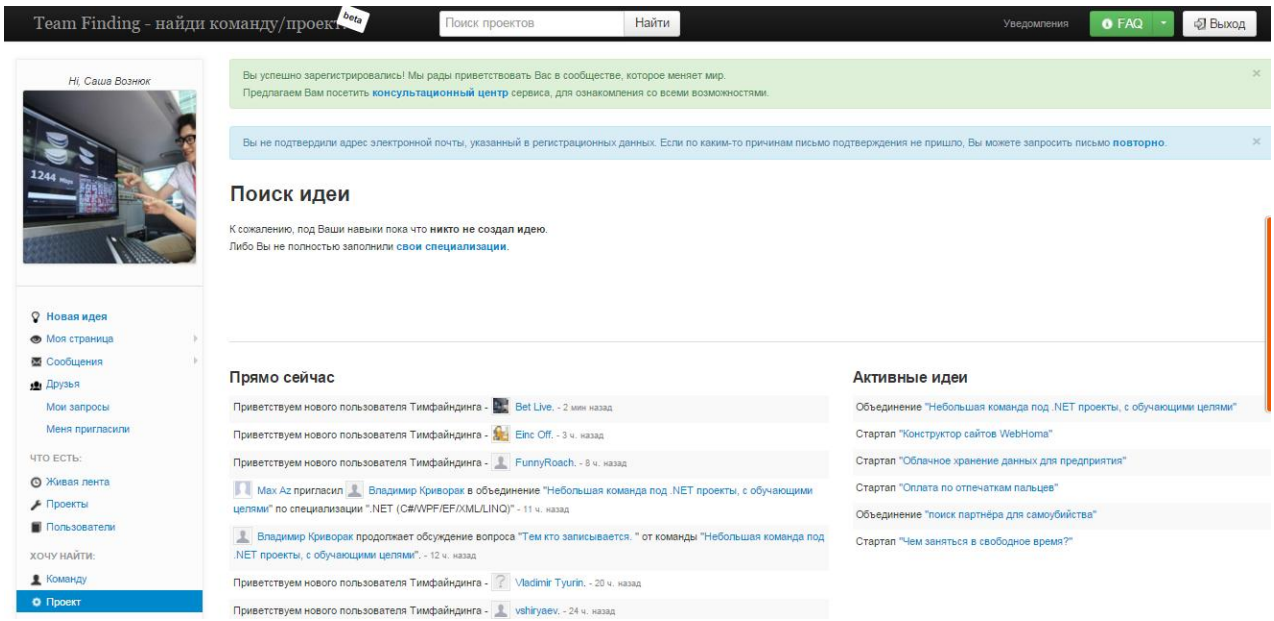


Рисунок 1.15 - Пошук ідей

Пошук ідеї. Якщо новий користувач хоче створити новий проект, але він не впевнений, що такий проект вже існує, то він може скористатися пошуком ідей, де користувач вводить назву ідеї і якщо така ідея існує ,то вона буде відображена.

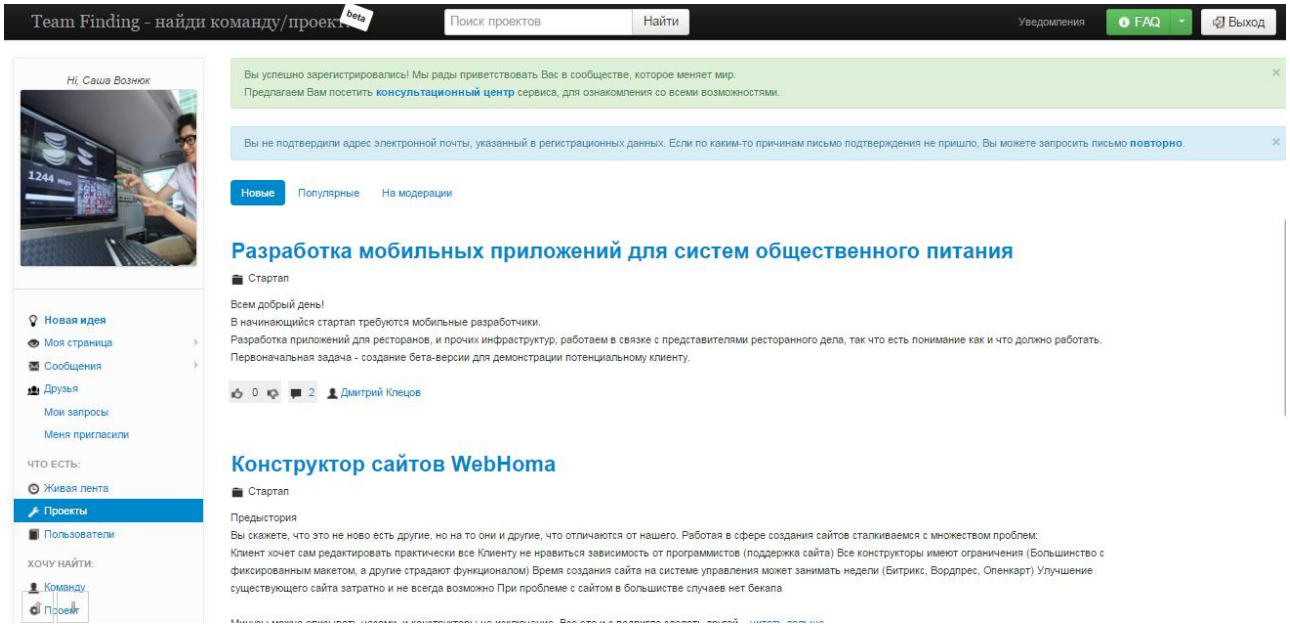


Рисунок 1.16 - Список проектів

Список проектів. На даній сторінці користувач може переглянути всі проекти і вибрати той проект в якому він би хотів приймати участь.

Sturtup fellows – сайт призначений для пошуку стартапів та людей, які можуть їх реалізувати.

Для створення нового стартапу потрбно виконати наступні кроки:

- натиснути на кнопку “новая запись” після чого заповнити наступні поля:

а)Тип запису:

- стартап шукає людей;
- шукаю стартап.

б)Заголовок.

в)Короткий опис.

г)Опис;

- натиснути кнопку опублікувати.

Для того, щоб знайти стартап, потрібно вибрати в меню “Стартап ищет людей”, після чого зі списку стартапів вибрати той який вам подобається і вибрати “Написати автору”.

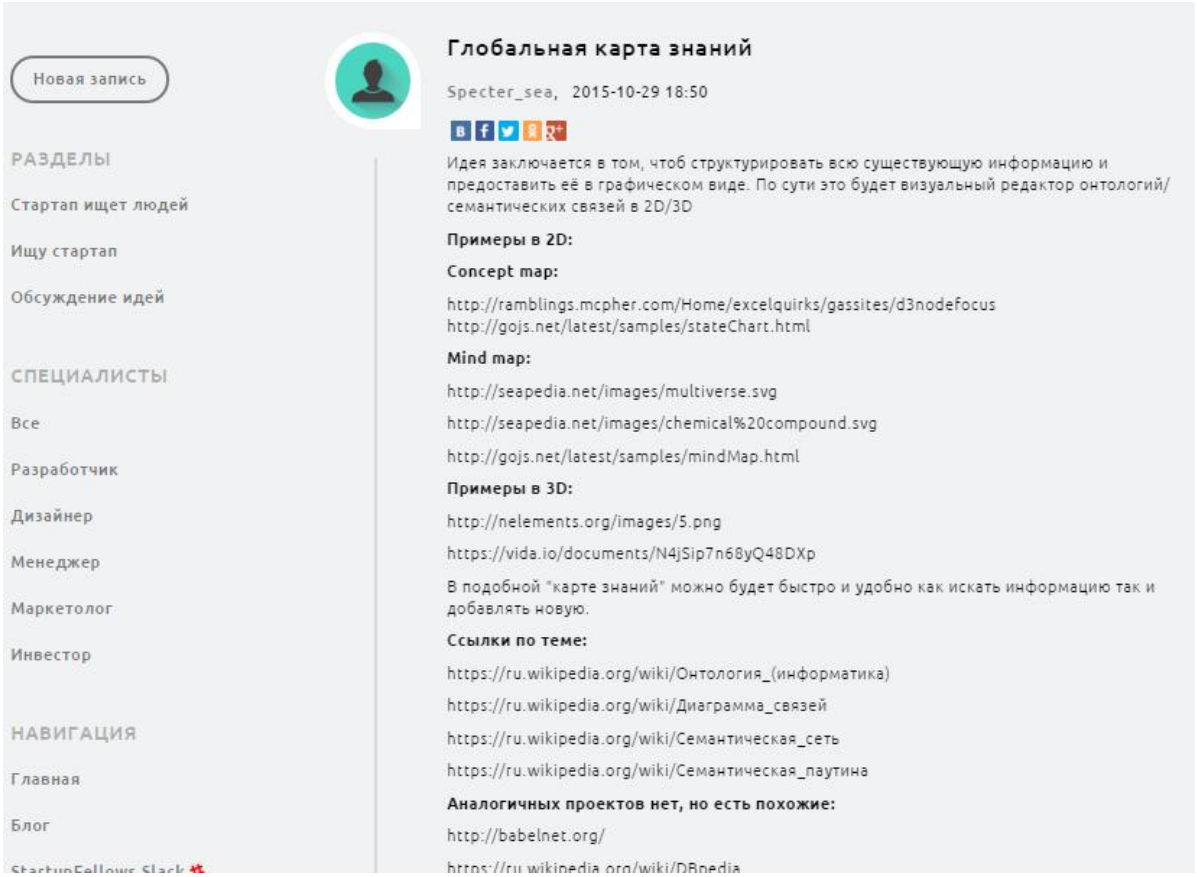


Рисунок 1.17 - Створений проект

В створеному стартапі можна переглянути короткий опис про проект, посилання в соціальних мережах та посилання на:

- приклади;
- посилання по темі;
- аналоги.

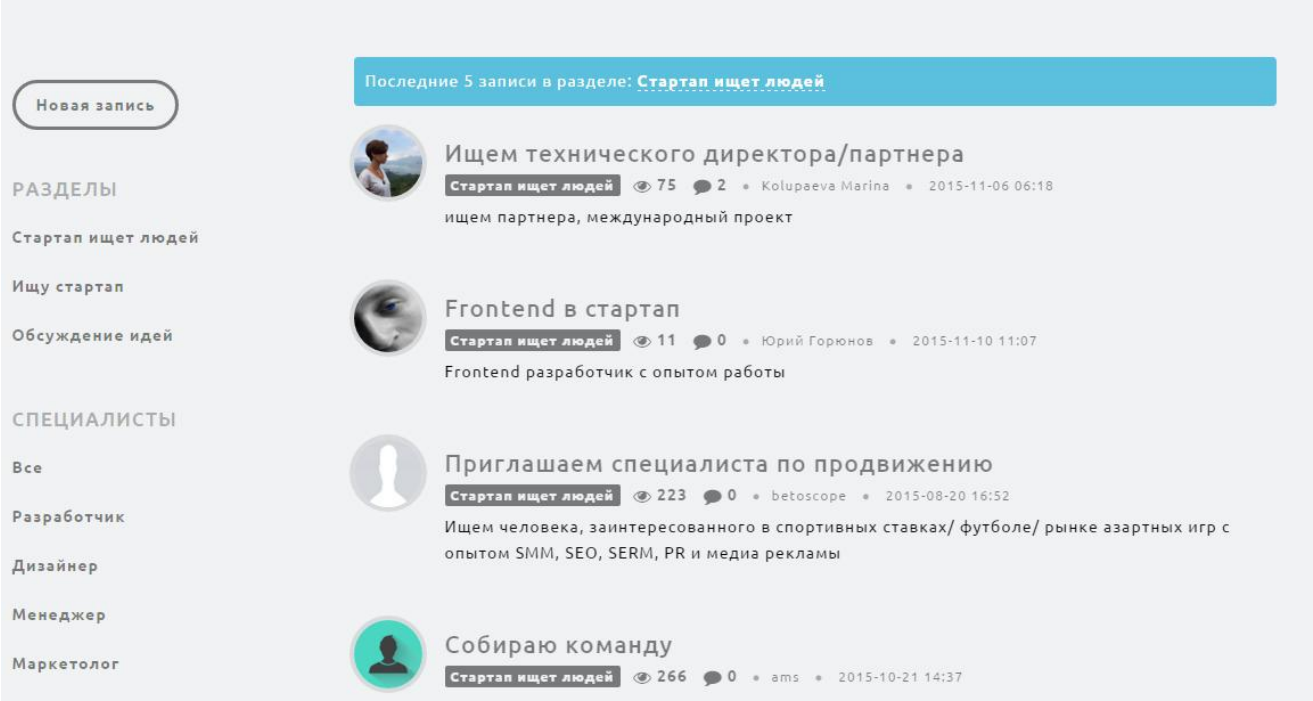


Рисунок 1.18 - Список проектів

В списку проектів ми бачимо усі створені проекти та інформацію про них:

- назва;
- чи шукає стартап людей;
- короткий опис.

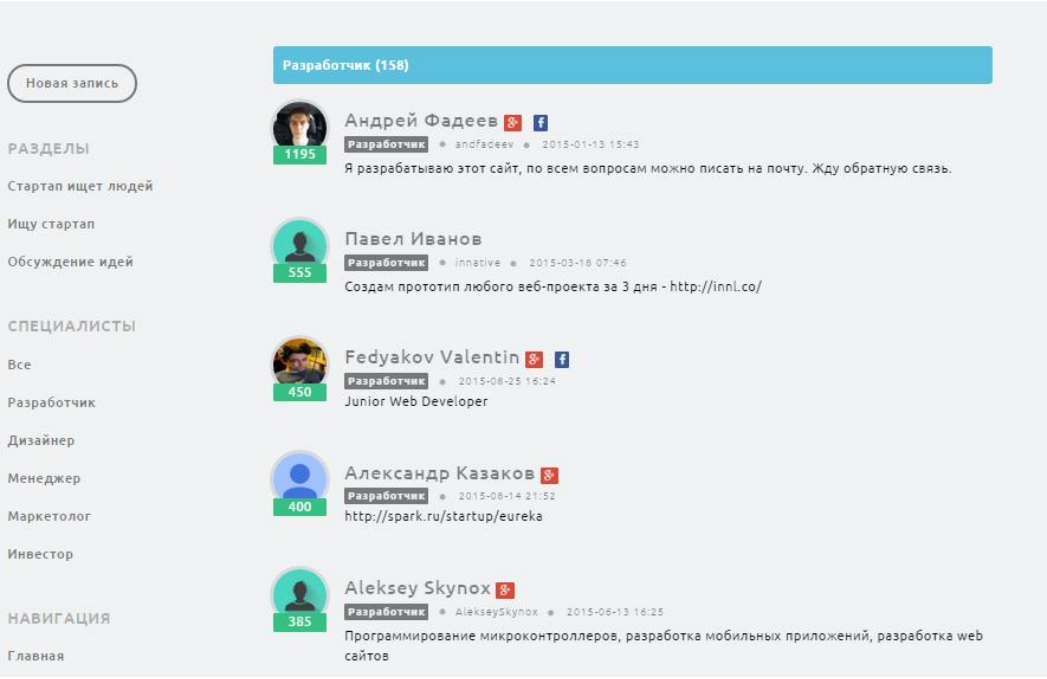


Рисунок 1.19 - Список користувачів

В списку користувачів ми можемо переглянути всіх, хто зареєстрований на сайті і деяку інформацію про них, а саме:

- ім’я та прізвище;
- статус користувача (розробник, звичайний користувач);
- коротка інформація про користувач;
- посилання на сторінки в соціальних мережах.

Таблиця 1.6

Порівняльна характеристика програмних продуктів

Фірма-розробник	indiegogo	teamfinding	startup fellows
Тип продукту	Краудфандінг	Управління стартапами	Пошук стартапів та людей для стартапів
Дата заснування	19-04-2007	21-08-2012	25-12-2014
Місце знаходження	США, Каліфорнія, Сан-Франциско	Україна, Київ	США, Ашберн
Назва програмного продукту	Indiegogo.com	teamfinding.com	startupfellows.ru
Версії продукту	2015	2015	2015
Функціональність	Створення проектів для їх фінансування	Створення проектів для спільної розробки з користувачами	Створення проектів для спільної розробки з користувачами
Інтерфейс користувача	Приємний	Помірний	Не зручний
Допомога користувачу	Присутня	Відсутня	Відсутня

Порівнявши дані програмні продукти можна сказати, що startup fells і

teamfinding виконують одні і ті ж функції, дають можливість користувачам створювати проекти, команди. Недоліком цих систем є їхній інтерфейс, який не є зручним, тому користувачу на перший погляд може здатись, що створення або пошук проекту є важким процесом. До переваг можна віднести створену систему коментарів під проектами, де користувачі можуть обговорити даний проект.

Indiegogo - є потужною системою по збору коштів для реалізації проектів. До недоліком можна віднести назвати недостатню кількість систем оплати.

На даний момент існує дуже багато сервісів для оплати в мережі інтернет і кожен користувач вибирає той сервіс, який йому найбільше до вподоби, тому я вважаю, що в системі, яка працює з сервісами оплати, необхідно по максимуму підключити ці сервіси для зручності користувачів. До переваг можна віднести інтерфейс, який дуже зручний і зрозумілий для користувача. Також однією з переваг є те, що вже за декілька хвилин ми можемо створити власний проект і розпочати збір коштів.

Після аналізу програмних продуктів, можна прийняти наступні рішення для розробки модуля системи:

- здатність поширювати проекти за допомогою соціальних мереж, що надасть можливість збільшити кількістю користувачів та збільшити фінансування проектів;

- створити систему для підтвердження особи, щоб зменшити кількість людей, які хочуть “нажитися”, створивши проект, який неможливо реалізувати.

1.4Специфікація вимог до модуля (системи)

У таблиці 1.7 представлений короткий словник термінів розробленої системи.

Таблиця 1.7

Глосарій

Термін	Опис Терміну
Основні поняття та критерії предметної області та проекту	
Краудфандінг	Об'єднання людей, які добровільно залучають свої гроші або інші ресурси, як правило через Інтернет, щоб підтримати розробки інших людей. Краудфандингове фінансування виконує різні функції — допомога постраждалим, підтримка з боку вболівальників чи фанатів, підтримка політичних кампаній, фінансування стартап-компаній та малого бізнесу, створення програмного забезпечення і багато ін. Для стартування збору коштів обов'язково повинна бути мета, ціна її досягнення, а обрахунок усіх витрат і процес збору мають бути відкриті для публіки у вільному доступі.
Стартап	Стартáп (англ. *startup*), стартап-компанія — нещодавно створена компанія (можливо, ще не зареєстрована офіційно, але серйозно планує стати офіційною), що будує свій бізнес на основі інновацій або інноваційних технологій, не вийшла на ринок або ледве почала на нього виходити і що володіє обмеженим набором ресурсів. Часто стартап-компанії називають «гаражними». Особливо часто термін «стартап» застосовується відносно інтернет-компаній і інших фірм, що працюють в сфері IT, проте, це поняття розповсюджується і на інші сфери діяльності.
Користувачі системи	
Простий користувач	Створює проекти, навчається програмуванню, створює команду
Програміст	Спеціаліст в певній області, це може бути (UX дизайнер, UI дизайнер, php програміст і т.д). Шукає цікаві проекти, приймає участь у реалізації проектів, може створювати проекти, команди.
Компанія	Шукає цікаві проекти, фінансує, купує них. Також наймає

	спеціалістів у свою компанію

У таблиці 1.8 наведена специфікація вимог до системи.

Таблиця 1.8

Специфікація функціональних вимог

Ідентифікатор вимог	Назва (варіанту вимоги)	Атрибути вимог		
		Пріоритет	Складність	Контакт
1	Створення стартапу	Обов’язкове	середня	Користувач, Спеціаліст
2	Пошук стартапу	Обов’язкове	середня	Користувач, Компанія, Спеціаліст
3	Реєстрація	Обов’язкове	низька	Користувач, Компанія, Спеціаліст
4	Вхід	Обов’язкове	низька	Користувач, Компанія, Спеціаліст
5	Обмін повідомленнями	Обов’язкове	висока	Користувач, Компанія, Спеціаліст

У таблиці 1.9 наведена специфікація нефункціональних вимог системи.

Таблиця 1.9

Специфікація нефункціональних вимог

Ідентифікатор вимоги	Назва вимоги	Атрибути вимог		
		Пріоритет	Складність	Контакт
Застосовність				
1	Час необхідний для навчання звичайних і	Середній	Середня	Адміністратор

	досвідчених користувачів			

Продовження таблиці 1.9

Надійність				
1	доступність	Високий	Середня	
2	середній час безвідмовної роботи	Високий	Висока	
Робочі характеристики				
1	швидкодія для транзакції	Високий	Середня	
2	продуктивність	Високий	Висока	
Вимоги до документації, призначеної для користувача, і до системи				
1	Зрозумілість	Високий	Низька	Адміністратор
Інтерфейси				
1	Зручність	Високий	Середня	
2	Зрозумілість	Високий	Середня	

Діаграма варіантів використання представлена рисунком 1.20

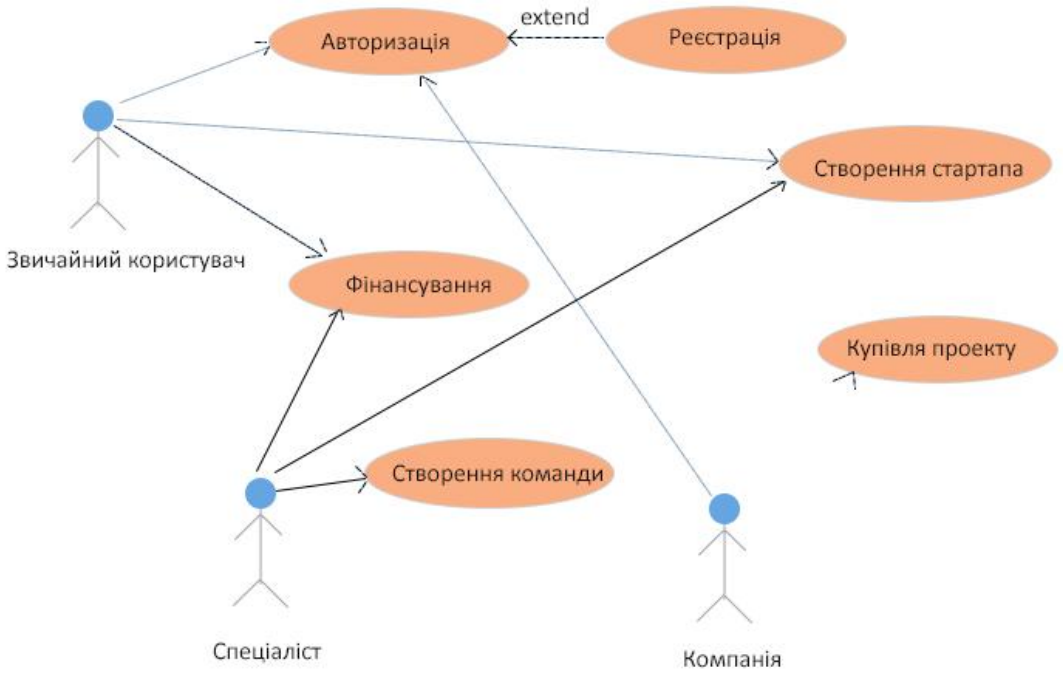


Рисунок 1.20 - Діаграма варіантів використання

Таблиця 1.10

Варіант використання авторизації

Контекст використання	Перенаправлення на головну сторінку користувача що авторизувався
Дійові особи	Користувач, його інтересом є вхід у систему та подальше використання її в своїх цілях, а саме (створенні стартапа, пошуку та ін.)
Передумова	Має бути сторінка авторизації з формою для введення даних, при коректному вводі вхід у систему
Тригер	-
Сценарій	-
Постумова	Після реєстрації відкривається сторінка користувача

Розкадровка вхід в систему

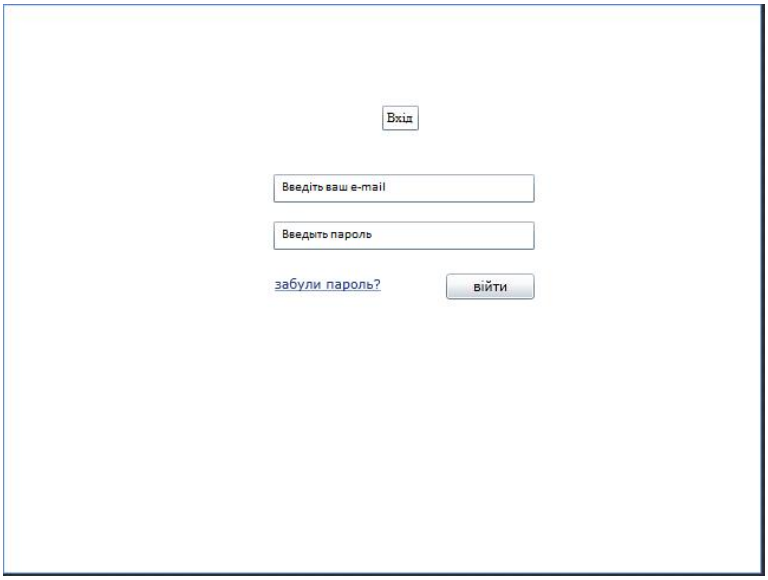


Рисунок 1.21 - Розкадровка вхід в систему

Для використання системи управління стартапами користувач повинен зареєструватися. Для реєстрації необхідно заповнити наступні поля:

- поле email – потрібно ввести e-mail, який був введений при реєстрацій;
- поле пароль - потрібно ввести пароль, який був введенй при реєстрацій;
- кнопка війти – після нажаття на кнопку, якщо введені дані були вірні, нас перенаправляє на сторінку сайту, якщо ні, то або не правильно були введені дані, або користувач з такими даними не існує.

Таблиця 1.11

Варіант використання “створення стартапу”

Контекст використання	Перенаправлення на головну сторінку користувача що авторизувався
Дійові особи	Користувач, спеціаліст – можуть створюють проект.
Передумова	Має бути сторінка авторизації з формою для введення даних, при коректному вводі вхід у систему
Тригер	
Сценарій	
Постумова	Перенаправлення на сторінку створеного проекту

Розкадровка створення стартапу

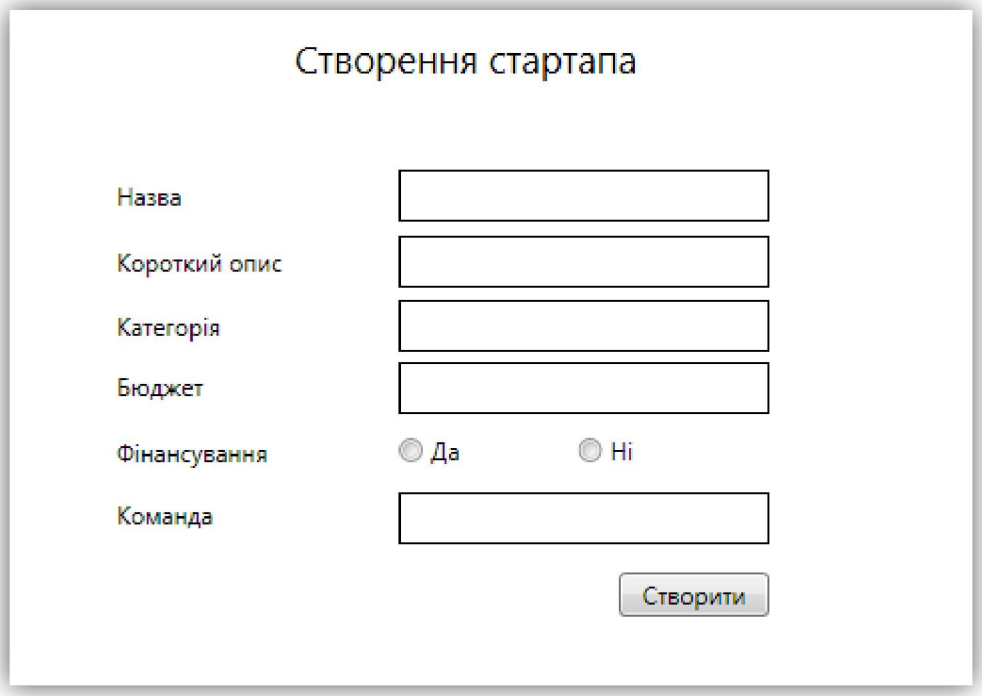


Рисунок 1.22 - Розкадровка створення стартапу

При створенні нового проекту користувач повинен заповнити всі необхідні поля. Після створення проекту, користувач може розпочати розробку та розвиток даного проекту.

Для створення нового проекту потрібно заповнити наступні поля:

- поле “Назва” – назва проекту, має бути унікальною;
- короткий опис - в цьому полі описують основну діяльність проекту і головну його мету;
- категорія – в даному полі потрібно вказати до якої категорії буде відноситися проект (веб – додаток, додаток на android тощо) ;
- бюджет – в даному полі користувач вводить доступний бюджет на розробку даного проекту;
- фінансування – в цьому полі є два варіанта вибору, так або ні. Коли користувач вибрав так, то даний проект зможе фінансуватися користувачами та компаніями, коли користувач вибрав ні, тоді дана можливість пропадає;

- команда – в даному полі користувач вказує, які користувачі йому потрібні (php програмісти, дизайнери тощо) ;
- кнопка війти – після нажаття на кнопку, якщо введені дані були вірні, користувача перенаправляє на нашу сторінку, якщо ні, то або не правильно були введені дані, або користувач з такими даними не існує.

Висновок.

1. Основна діяльність об’єкту управління стартапами – створення, пошук та реалізація проектів різного типу.
2. Основноми бізнес – процесами є:

- процес формування календарного плану;
- процес управління стартапом;
- процес фінансування стартапу;
- прроцес управління зустрічами;

3. Аналогами програмної системи є:

- Indiegogo
- Teamfinding
- Sturtup fellows

РОЗДІЛ 2 ПРОЕКТУВАННЯ

2.1 Розробка архітектури програмної системи.

Веб додаток буде використовувати 3 основні компоненти: веб браузер, веб сервер та базу даних. На рисунку 23 зображено діаграму компонентів, яка відображає залежності між компонентами програмного забезпечення.

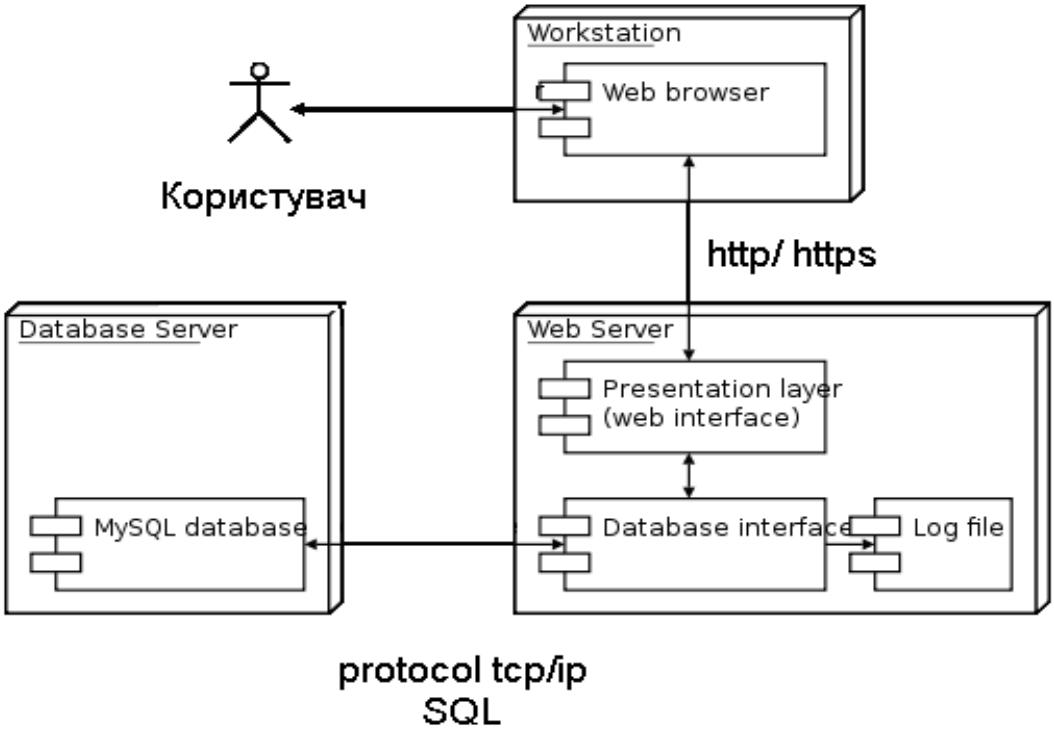


Рисунок 2.1 – Діаграма компонентів розроблюваної програмної системи

Статичне представлення структурної моделі зображають у вигляді діаграми класів. Структурна модель додатку є досить складною. Система матиме велику кількість класів, що забезпечуватимуть можливість її функціонування. Більшість модулів системи відповідатимуть за інтерфейс або стосуватимуться особливостей налагодження роботи. Тому їх відображення на даному етапі є не потрібним, оскільки воно зробить діаграму класів занадто громіздкою та важкою для читання і сприйняття. Отже, на рисунку 2.2 діаграма відображена лише тими класами, які забезпечуватимуть виконання логіки програми.

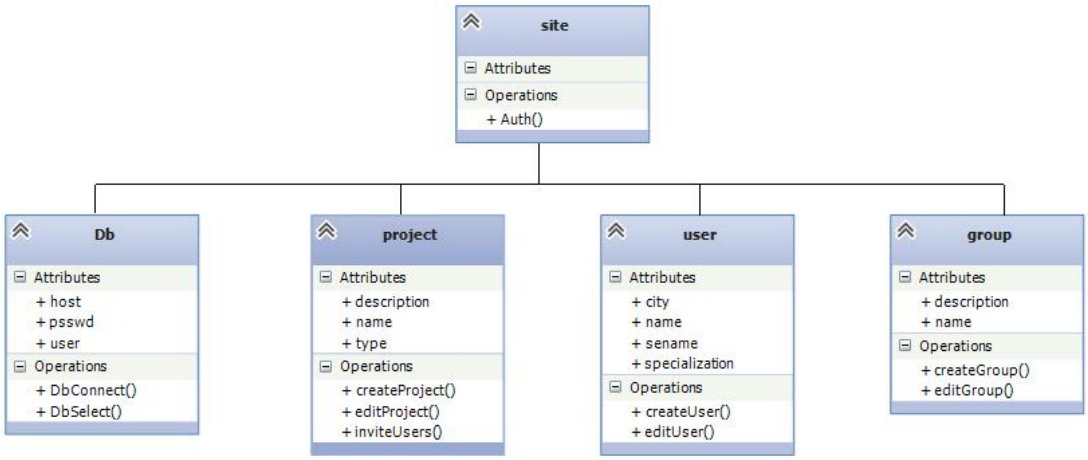


Рисунок 2.2 - Діаграма класів

На рисунку 2.3 зображена діаграма послідовності для варіанту використання «Редагування користувачів», яка відображає взаємодію об'єктів впорядкованих за часом. Можливість маніпулювання даними користувачів буде доступна лише адміністратору.

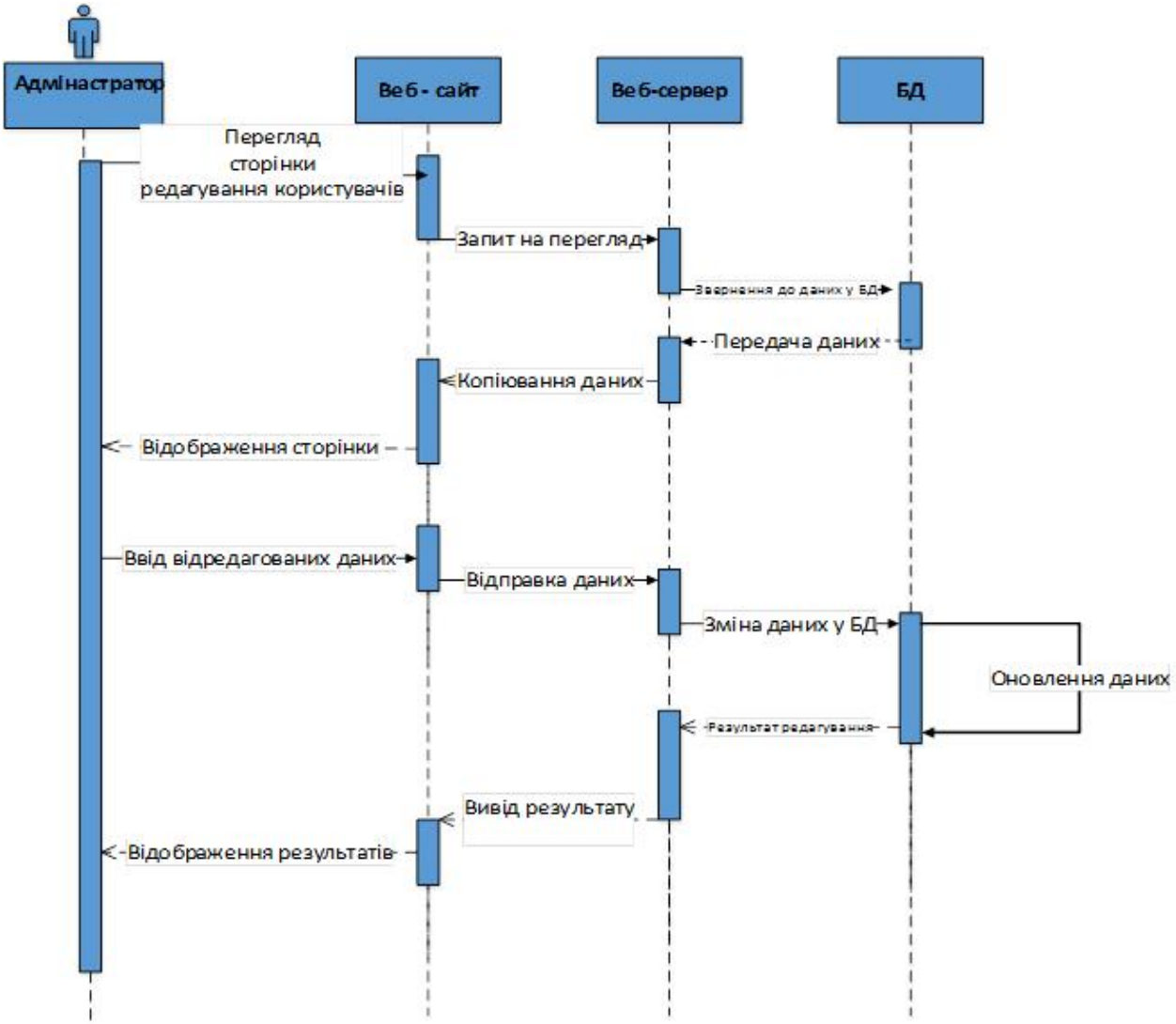


Рисунок 2.3 - Діаграма послідовності (процесу редагування користувачів)

На рисунку 2.4 зображена діаграма послідовності для варіанту використання «Додавання проекту», яка відображає взаємодію об'єктів впорядкованих за часом.

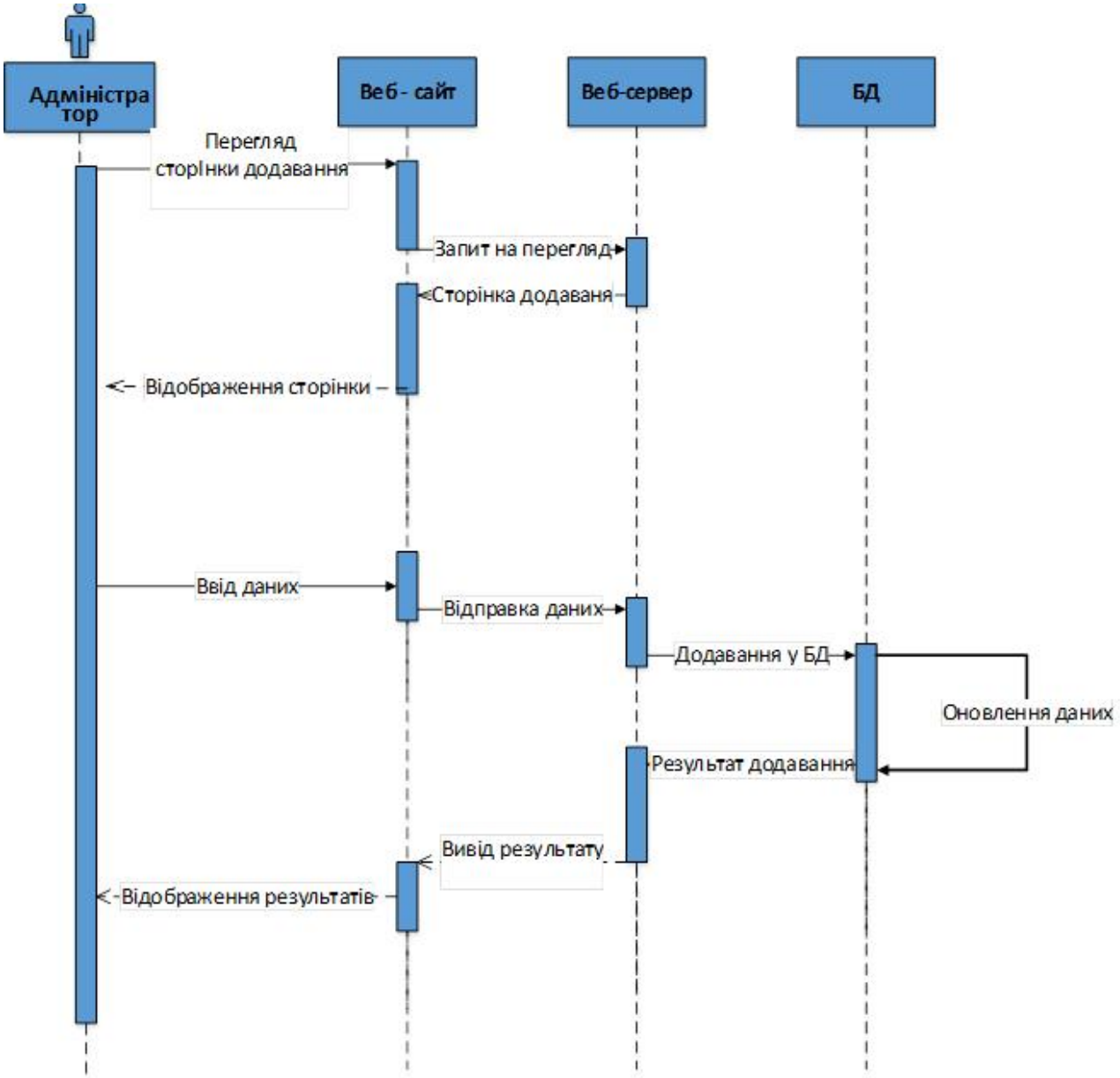


Рисунок 2.4 - Діаграма послідовності “Процесу додавання нового проекту”

Як видно з рисунку, найдовшим процесом буде відображення результатів після внесення змін, оскільки він містить найбільше етапів.

Для моделювання поведінки об'єктів системи при переході з одного стану в інший використовують діаграми станів. Діаграма станів для функції «Редагування» зображено на рисунку 2.5

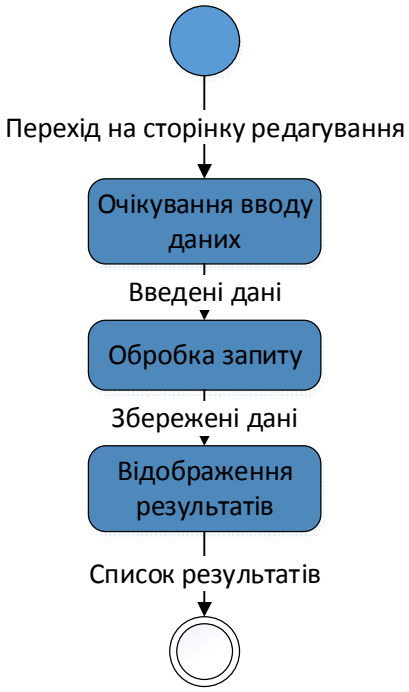


Рисунок 2.5 - Діаграма станів процесу “Редагування”

Як видно з рисунку, після переходу на сторінку редагування, система очікує вводу даних. Коли користувач здійснить дану операцію, введені дані обробляються, після чого зберігаються у БД. В подальшому дані оновлюються і відображаються користувачу.

На рисунку 2.6 зображено діаграму станів процесу “Видалення”

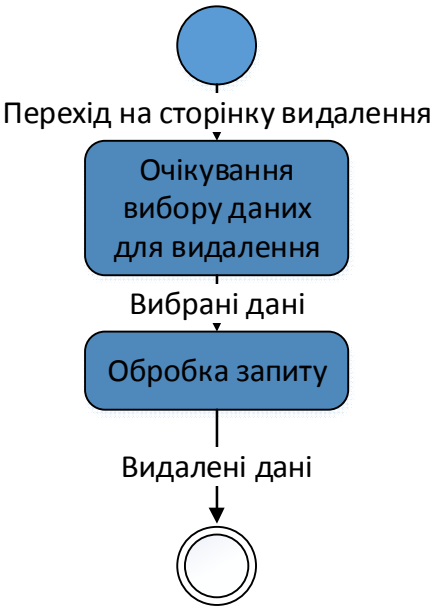


Рисунок 2.6- Діаграма станів процесу “Видалення”

Як видно з рисунку, після переходу на сторінку видалення, обраний кортеж обробляється системою. Під час виконання запиту запис видаляється.

На рисунку 2.7 зображено діаграму станів процесу “Додавання студента”

Рисунок 2.7 - Діаграма станів процесу “Додавання студента”

Як видно з рисунку, після переходу на сторінку додавання, система очікує вводу даних. Якщо введено назву проекту, ідентичний тому, який вже є у БД, система повідомить про те, що запис про проект уже присутній і запропонує перевірити правильність вводу. Якщо ж всі поля заповнені коректно дані успішно будуть внесені у БД та відобразиться повідомлення, про успішно доданий запис.

2.2 Проектування структури бази даних

Для нормальної роботи системи, потрібно забезпечити збереження великого обсягу даних, доступ до яких буде можливий лише користувачам цього програмного продукту. Це стало можливим за допомогою баз даних та

засобами їх управління, зокрема системою управління базами даних (СУБД)MySQL, яка використовується в системі.

Для реалізації програмного продукту було обрано клієнт-серверну архітектуру. У цій архітектурі запити розподілені між постачальниками послуг (серверами) та споживачами (користувачами). Фізично клієнт і сервер – це програмне забезпечення. Зазвичай вони між собою взаємодіють, за допомогою мережі, через мережеві протоколи і знаходяться на різних обчислювальних машинах. Програми – сервера, очікують від клієнтських програм запитів і надають їм свої ресурси, у вигляді даних або ж сервісних функцій.

На рисунку 2.8 наведено елементів та зв’язки між ними, які будуть використовуватись на етапі експлуатації системи.

Рисунок 2.8 - Діаграма елементів і зв’язків

При високорівневому (концептуальному) проектуванні баз даних використовують ERD-модель. Під час проектування БД відбувається перетворення ERD-моделі в конкретну схему бази даних на основі обраної моделі даних. На рисунку 2.9 зображено модель ERD для розроблюваної

системи.

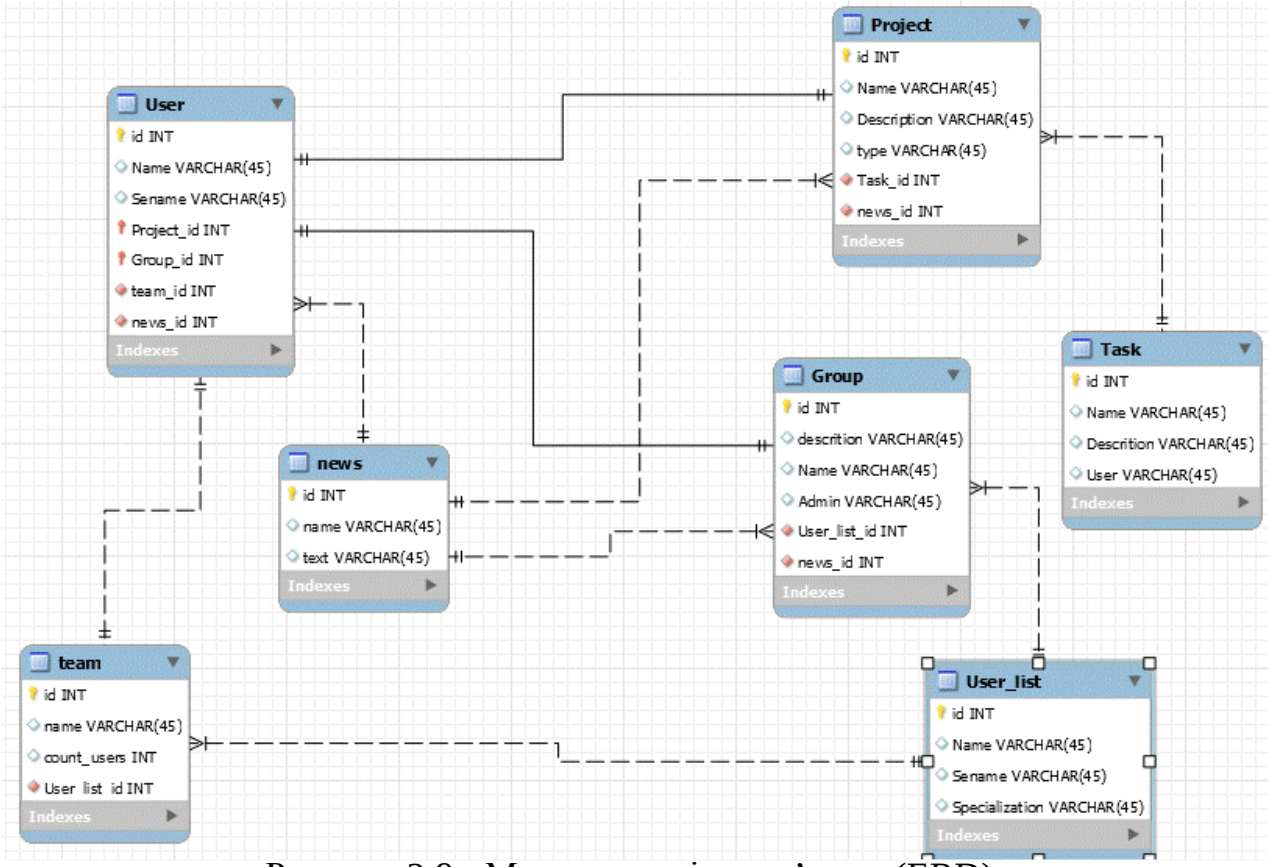


Рисунок 2.9 - Модель сутність-зв’ язок (ERD)

Усі відношення автоматично індексуються, по первинному ключі. Завдяки чому значно пришвидшується процес пошуку даних у БД.

DDL опис бази даних наведено нижче

```
SET @OLD_UNIQUE_CHECKS=@@UNIQUE_CHECKS, UNIQUE_CHECKS=0;
SET @OLD_FOREIGN_KEY_CHECKS=@@FOREIGN_KEY_CHECKS,
FOREIGN_KEY_CHECKS=0;
SET @OLD_SQL_MODE=@@SQL_MODE,
SQL_MODE='TRADITIONAL,ALLOW_INVALID_DATES';

-- Schema mydb
-- Schema mydb
CREATE SCHEMA IF NOT EXISTS `mydb` DEFAULT CHARACTER SET utf8 ;
USE `mydb` ;
```

```
-- Table `mydb`.`Task`
CREATE TABLE IF NOT EXISTS `mydb`.`Task` (
  `id` INT NOT NULL,
  `Name` VARCHAR(45) NULL,
  `Descrition` VARCHAR(45) NULL,
  `User` VARCHAR(45) NULL,
  PRIMARY KEY (`id`))
ENGINE = InnoDB;

-- Table `mydb`.`news`
CREATE TABLE IF NOT EXISTS `mydb`.`news` (
  `id` INT NOT NULL,
  `name` VARCHAR(45) NULL,
  `text` VARCHAR(45) NULL,
  PRIMARY KEY (`id`))
ENGINE = InnoDB;

-- Table `mydb`.`Project`
CREATE TABLE IF NOT EXISTS `mydb`.`Project` (
  `id` INT NOT NULL,
  `Name` VARCHAR(45) NULL,
  `Description` VARCHAR(45) NULL,
  `type` VARCHAR(45) NULL,
  `Task_id` INT NOT NULL,
  `news_id` INT NOT NULL,
  PRIMARY KEY (`id`),
  INDEX `fk_Project_Task1_idx` (`Task_id` ASC),
  INDEX `fk_Project_news1_idx` (`news_id` ASC),
  CONSTRAINT `fk_Project_Task1`
    FOREIGN KEY (`Task_id`)
    REFERENCES `mydb`.`Task` (`id`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION,
  CONSTRAINT `fk_Project_news1`
    FOREIGN KEY (`news_id`)
    REFERENCES `mydb`.`news` (`id`)
    ON DELETE NO ACTION
```

```
    ON UPDATE NO ACTION)
ENGINE = InnoDB;


-- Table `mydb`.`User_list`
CREATE TABLE IF NOT EXISTS `mydb`.`User_list` (
  `id` INT NOT NULL,
  `Name` VARCHAR(45) NULL,
  `Sename` VARCHAR(45) NULL,
  `Specialization` VARCHAR(45) NULL,
  PRIMARY KEY (`id`))
ENGINE = InnoDB;



-- Table `mydb`.`Group`
CREATE TABLE IF NOT EXISTS `mydb`.`Group` (
  `id` INT NOT NULL,
  `descrition` VARCHAR(45) NULL,
  `Name` VARCHAR(45) NULL,
  `Admin` VARCHAR(45) NULL,
  `User_list_id` INT NOT NULL,
  `news_id` INT NOT NULL,
  PRIMARY KEY (`id`),
  INDEX `fk_Group_User_list1_idx` (`User_list_id` ASC),
  INDEX `fk_Group_news1_idx` (`news_id` ASC),
  CONSTRAINT `fk_Group_User_list1`
    FOREIGN KEY (`User_list_id`)
    REFERENCES `mydb`.`User_list` (`id`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION,
  CONSTRAINT `fk_Group_news1`
    FOREIGN KEY (`news_id`)
    REFERENCES `mydb`.`news` (`id`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION)
ENGINE = InnoDB;
```

```
-- Table `mydb`.`team`
CREATE TABLE IF NOT EXISTS `mydb`.`team` (
  `id` INT NOT NULL,
  `name` VARCHAR(45) NULL,
  `count_users` INT NULL,
  `User_list_id` INT NOT NULL,
  PRIMARY KEY (`id`),
  INDEX `fk_team_User_list1_idx` (`User_list_id` ASC),
  CONSTRAINT `fk_team_User_list1`
    FOREIGN KEY (`User_list_id`)
    REFERENCES `mydb`.`User_list` (`id`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION)
ENGINE = InnoDB;

-- Table `mydb`.`User`
CREATE TABLE IF NOT EXISTS `mydb`.`User` (
  `id` INT NOT NULL,
  `Name` VARCHAR(45) NULL,
  `Sename` VARCHAR(45) NULL,
  `Project_id` INT NOT NULL,
  `Group_id` INT NOT NULL,
  `team_id` INT NOT NULL,
  `news_id` INT NOT NULL,
  PRIMARY KEY (`id`, `Project_id`, `Group_id`),
  INDEX `fk_User_Project_idx` (`Project_id` ASC),
  INDEX `fk_User_Group1_idx` (`Group_id` ASC),
  INDEX `fk_User_team1_idx` (`team_id` ASC),
  INDEX `fk_User_news1_idx` (`news_id` ASC),
  CONSTRAINT `fk_User_Project`
    FOREIGN KEY (`Project_id`)
    REFERENCES `mydb`.`Project` (`id`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION,
  CONSTRAINT `fk_User_Group1`
    FOREIGN KEY (`Group_id`)
    REFERENCES `mydb`.`Group` (`id`)
```

```
    ON DELETE NO ACTION
    ON UPDATE NO ACTION,
  CONSTRAINT `fk_User_team1`
    FOREIGN KEY (`team_id`)
    REFERENCES `mydb`.`team` (`id`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION,
  CONSTRAINT `fk_User_news1`
    FOREIGN KEY (`news_id`)
    REFERENCES `mydb`.`news` (`id`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION)
ENGINE = InnoDB;


-- Table `mydb`.`table1`
CREATE TABLE IF NOT EXISTS `mydb`.`table1` (
)
ENGINE = InnoDB;


SET SQL_MODE=@OLD_SQL_MODE;
SET FOREIGN_KEY_CHECKS=@OLD_FOREIGN_KEY_CHECKS;
SET UNIQUE_CHECKS=@OLD_UNIQUE_CHECKS;
```

Висновок.

В ході проектування програмної системи:

1. Було спроектовано архітектуру програмного продукту, розроблена діаграма класів, послідовності, діаграма станів.
2. Вибрана СУБД, спроектована діаграма елементів і зв’язків, ERD діаграма.

РОЗДІЛ 3. ПРОГРАМНА РЕАЛІЗАЦІЯ

3.1 Обгрунтування вибору мови програмування

Для розробки web-орієнтованої системи управління стартапами, була використана мова php.

PHP (**Hypertext PreProcessor**) - мова програмування, виконувана на стороні веб-сервера для розробки динамічних сайтів. Дана мова програмування є дуже гнучкою і потужною, тому використовується в проектах різного масштабу і є дуже популярним інструментом для розробки web-додатків.

Основну роль у виборі мови програмування зіграли такі аспекти:

- вільне програмне забезпечення;
- простий у використанні;
- підтримується великою кількістю користувачів і розробників;
- розвинена підтримка баз даних:
- велика кількість бібліотек і розширень;
- може використовуватися в ізольованому середовищі;
- може бути розгорнутий майже на любому сервері;
- легко працювати на різних платформах та ОС.

3.1.1 Організація інтерфейсу з користувачем

При організації інтерфейсу з користувачем головна увага приділялася простоті і зручності, щоб людина, яка вперше зайшла на сайт, могла відразу розібратися і розпочала працювати з програмною системою.

Інтерфейс головної сторінки користувача містить в собі:

- “шапку” сайту:

а) назва сайту;

б) меню;

в) форма пошуку;

г) іконку користувача;

д) кнопку “Створити”

- блок з головною інформацією про користувача:

а) фото користувача;

б) інформація (Ім’я, прізвище, країна, місто, рік народження, мови, де навчався, де працює чи працював) ;

в) статус (коротка інформація, яку користувач хоче донести до інших, які будуть переглядати його сторінку) ;

г) дві кнопки (“Написати” – для обміну повідомленнями між користувачами, “Запросити” – для запрошення користувача у проект) ;

- блок з власними проектами (проекти, які створив користувач) ;
- блок з проектами у яких приймає участь;
- моя команда (учасники команди, в які перебуває сам користувач) ;
- контактна інформація (e-mail, телефон, скайп, сторінка в фейсбуці, сторінка вконтакті).
- інтерфейс головної сторінки користувача має два різних вигляди:
- як бачить свою сторінку сам користувач;
- як бачать її інші користувачі.

На сторінці, яку бачать інші користувачі знаходиться основна інформація про користувача, контактні дані, проекти в яких користувач приймає участь і власні проекти, які створив він. Також, на сторінці користувача знаходиться дві кнопки: “Написати”, “Запросити”. Скориставшись кнопкою “Написати”, ми можемо зв'язатися з користувачем, кнопка “Запросити” для того щоб ви могли запропонувати користувачеві прийняти участь у вашому проекті.

Інтерфейс користувача, як бачить його сам користувач містить таку саму інформацію, але відрізняється тим що заміть кнопок: “Написати ”, “Запросити”, кнопки: Проект “створити новий проект”, ”Команда” (створити

команду).

Інтерфейс проекту, має містити в собі дані, які будуть зрозумілі всім хто зайде на сторінку проекту, щоб користувач відразу міг зрозуміти чому і для чого був створений цей проект, хто бере участь в розробці, кого шукає даний стартап, можливо саме такого спеціаліста не вистачає проекту, яким є переглядач проекту.

Інтерфейс сторінки проекту містить в собі:

- “шапку” сайту:
  а) назва сайту;
  б) меню;
  в) форма пошуку;
  г) іконку користувача;
  д) кнопку “Створити”
- блок з інформацією про проект:
  а) Опис головної ідеї проекту;
  б) чи потребує проект фінансування ;
  в) які спеціалісти потрібні для розробки проекту;
- блок зустрічей (якщо зустріч створена) ;
- блок з календарним планом;
- блок зі списком учасників, які приймають участь у розробці проекту.

Інтерфейс сторінки проекту, містить в собі список всіх створених стартапів користувачами, це зроблено для зручного перегляду проектів і щоб кожен спеціаліст міг вибрати зі списку проект, який йому найбільш до вподоби.

Всі проекти динамічно завантажуються при прокрутці сторінки користувачем, що є дуже зручно для швидкого перегляду списку.

Інтерфейс списку проектів містить:

- список всіх проектів:
  а) зображення проекту;

б) назву.

- кнопку “Завантажити ще” (на випадок, якщо сторінка динамічно не загрузилася).

Інтерфейс сторінки списку команд, містить в собі список команд. Кожен користувач може перейти на сторінку команди і подивитися на її учасників, проекти які вона розробила.

Всі команди динамічно завантажуються при прокрутці сторінки користувачем, що є дуже зручно для швидкого перегляду.

Інтерфейс списку проектів містить:

- список всіх команд:

а) зображення команди;

б) назву.

в) список двох проектів команди.

- кнопку “Завантажити ще” (на випадок, якщо сторінка динамічно не загрузилася).

3.1.2 Програмна реалізація проекта

При створенні системи управління стартапами використовувалась мова програмування PHP та об'єктно-орієнтований підхід.

Об’єктно – орієнтований підхід - це метод програмування, в якому програми представляються у вигляді сукупності об’єктів, які взаємодіють між собою.

Об’єкт є екземпляром класу, а клас – це спосіб опису сутності, який визначає стан і поведінку, а також правила для взаємодії з цією сутністю. Клас складається з властивостей та методів.

В даній програмній системі було розроблено такі класи:

- Register;
- User;

- Db;
- Project.

За допомогою класу Register відбувається реєстрація нових користувачів, в даному класі основну роль виконують наступні методи:

- GetId().

```
public function getId($email){
    $db = new Db();
    $db->connect();
    $sql = "SELECT id,email,password FROM users WHERE email='$email' limit 1";
    $res = mysql_query($sql);
    $val = mysql_fetch_assoc($res);
    return $val['id'];
  }
```

- ChekData().

```
public function checkData($user_name,$last_name,$email,$password,$repeat_password){
    if(empty($user_name) || empty($last_name) || empty($email) || empty($password) || empty($repeat_password) ){
        echo("Не всі дані заповнено");
    }else{
        if($password === $repeat_password){
            $password = md5($password);
            $db = new Db();
            $db->connect();
            $sql = 'INSERT INTO `users` (`user_name`, `last_name`, `email`, `password`) VALUES ('".$user_name."', '".$last_name."', '".$email."','".$password."');';
            $db->write_data($sql);
            header('Location: http://dev.pro/reg_info.php');
        }else{
            echo('Паролі не співпадають');
        }

    }
  }
```

- Login().

```
public function login($email,$password){
    if(empty($email) || empty($password)){
```

```
        echo("Не всі поля заповнені");
    }else{
        $db = new Db();
        $db->connect();
        $user = new Register();
        $email = $user->getEmail();
        $sql = "SELECT id,email,password FROM users WHERE email='$email' limit 1";
        $res = mysql_query($sql);
        $val = mysql_fetch_assoc($res);
        if($val['password'] === md5($password)){
            $_SESSION['id'] = $val['id'];
            header('Location: http://dev.pro');
        }else{
            echo('Не віроно введені дані');
        }
    }
}
```

- WriteInfoUser();

```
public function writeInfoUser($country,$city,$language,$school,$job){
        if(empty($country) || empty($city) || empty($language) || empty($school) || empty($job)){
            echo("Не всі поля заповнені");
        }else{
            $db = new Db();
            $email = $_SESSION['email'];
            $db->connect();
            $sql = "SELECT id FROM users WHERE email='$email'";
            $res = mysql_query($sql);
            $val = mysql_fetch_assoc($res);
            $id = $val['id'];
            $sql = "UPDATE users SET country = '$country', city = '$city', language = '$language',
school = '$school',job = '$job' WHERE id='$id'";
            $db->write_data($sql);
        }
    }
```

Ці методи виконують перевірку, запис, та збереження даних, які вводить користувач а форму при реєстрації.

Клас User несе в собі методи за допомогою яких створюється

персональна сторінка користувачів усіх трьох типів, а саме спеціаліст, компанія, звичайний користувач.

Клас DB розроблений для роботи з базою даних, а саме підключенням та вибором потрібної бази даних і маніпуляціями з даними різного роду, таких як запис, видалення, редагування.

Клас Project виконує функцію створення нового проекту користувачем. Містить такі методи:

- CreateProject();

```
public function CtreateProject($title,$description,$id_creater ,$ava,$id){
      if(empty($title) || empty($description) || empty($id_creater) || empty($ava) ){
         echo("Не всі дані заповнено");
      }else{
            $db = new Db();
            $db->connect();
            $sql = 'INSERT INTO `projects` (`title`, `description`, `id_creater`, `ava`) VALUES ('".$title."', '".$description."', '".$id_creater."','".$ava."');';
            $db->write_data($sql);
            header('Location: http://dev.pro/project.php?id=$id');
      }
   }
```

- UpdateProject();

```
public function UpdateProject($title,$description,$id_creater ,$ava,$id){
       if(empty($title) || empty($description) || empty($id_creater) || empty($ava) ){
         echo("Не всі дані заповнено");
      }else{
         $db = new Db();
         $email = $_SESSION['email'];
         $db->connect();
         $sql = "SELECT id FROM users WHERE email='$email'";
         $res = mysql_query($sql);
         $val = mysql_fetch_assoc($res);
         $id = $val['id'];
         $sql = "UPDATE projects SET title = '$title, description= '$description, ava = '$ava, WHERE id='$id'";
         $db->write_data($sql);}
```

Ці методи відповідають за створення нового проекту, редагування даних, запрошення нових учасників, створення календарного плану та зустрічей.

3.2 Програмна реалізація бази даних

Створення бази даних програмної системи відбувається в СУБД MySQL.

Phpmyadmin – веб-додаток написаний на php і є інтерфейсом для адміністрування СУБД MySQL, дозволяє через браузер здійснювати адміністрування сервера MySQL, запускати SQL команд і перегладати вміст таблиць і баз даних!

Для створення таблиць ми будемо використовувати SQL запити.

Створення таблиці відбувається в phpmyadmin. Створити нову таблицю можна 2 способами:

- за допомогою SQL запита (див. рис. 31)

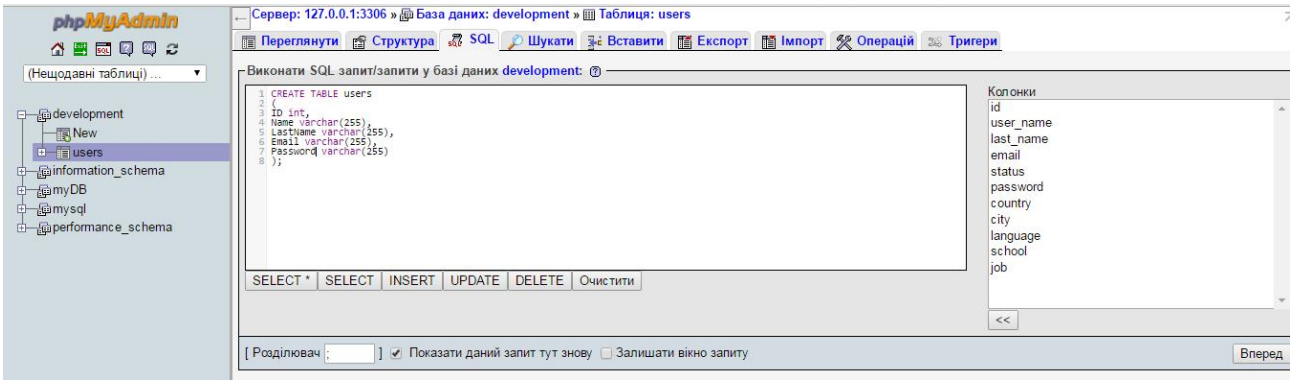


Рисунок 3.1 - Створення таблиці за допомогою SQL запиту

- за допомогою форми (див. рис. 3.2)

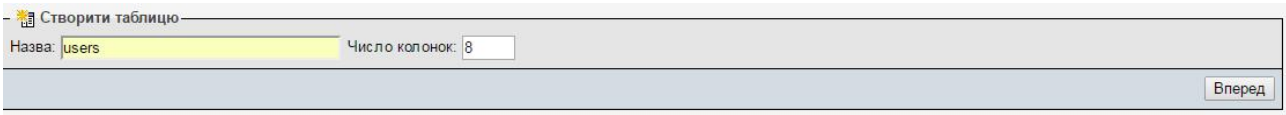
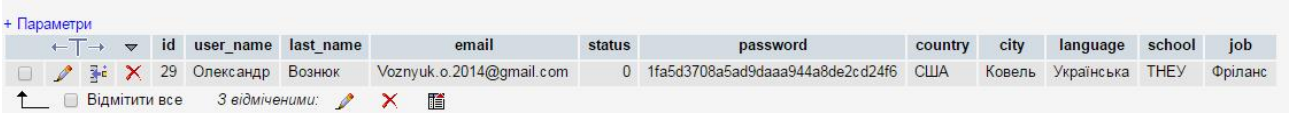


Рисунок 3.2 Створення таблиці за допомогою форми

В результаті ми отримаємо таблицю з заданими полями (дивитися рис. 3.3).



+ Параметри

	id	user_name	last_name	email	status	password	country	city	language	school	job
	29	Олександр	Вознюк	Voznyuk.o.2014@gmail.com	0	1fa5d3708a5ad9daaa944a8de2cd24f6	США	Ковель	Українська	ТНЕУ	Фріланс

Відмітити все З відміченими:

Рисунок 3.3 Створена таблиця

Після створення нової таблиці ми можемо робити різні операції над нею, такі як додавання записів, видалення, редагування, видалення таблиць, редагування таблиці.

Є чотири основні SQL запити, без яких була б неможлива робота з базами даних:

- додавання нового запису;
- видалення нового запису;
- редагування запису;
- вибір даних з бази даних.

Для додавання нового запису виконують такий SQL запит - $sql = 'INSERT INTO `users` (`user_name`, `last_name`, `email`, `password`) VALUES ('".$user_name."', '".$last_name."', '".$email."','".$password."');

Видалення даних відбувається за допомогою команди – “DELETE FROM users WHERE name=$name;”;

Редагування вже існуючих даних в базі даних відбувається за допомогою запиту - $sql = "UPDATE users SET country = '$country', city = '$city', language = '$language', school = '$school',job = '$job' WHERE id='$id'";

Вибірка виконується за домогою команди - $sql = SELECT *`users WHERE name = $name;

Висновок

В ході програмної реалізації було:

1. Організовано інтерфейс користувача
2. Розроблена програмна реалізація проекту, а саме опис класів, методів.
3. Програмна реалізація баз даних, вибрана система управління базами даних, варіанти SQL запитів

РОЗДІЛ 4. ТЕСТУВАННЯ ТА ДОСЛІДНА ЕКСПЛУАТАЦІЯ

4.1 Тестування програмного продукту

Тестування – це процес дослідження, що використовується для виміру якості програмного забезпечення і має на меті:

- показати розробникам і замовникам відповідність програми до вимог;
- визначити моменти, в яких поведінка програми не відповідає специфікації.

Як показує практика, програм без помилок не буває і найчастіше за все винуватцями помилок є самі програмісти. Тому програміст, повинен писати не лише ефективні програми, а знаходити помилки в цих програмах. Тому тестування веб-орієнтованої системи управління стартапами є не лише доцільним, а й необхідним аспектом при розробці даної системи.

Під час проведення тестування програмної системи були використані такі види тестування:

- функціональне тестування;
- тестування сумісності;
- тестування продуктивності;
- тестування GUI;
- тестування безпеки;

Також було застосовано як автоматизоване тестування так і ручне. Для автоматизованого тестування використовувався інструмент Selenium.

Selenium – це інструмент для автоматичного тестування Web – додатків. На рисунку 4.1 зображено тестування системи реєстрації на сайті за допомогою Selenium.

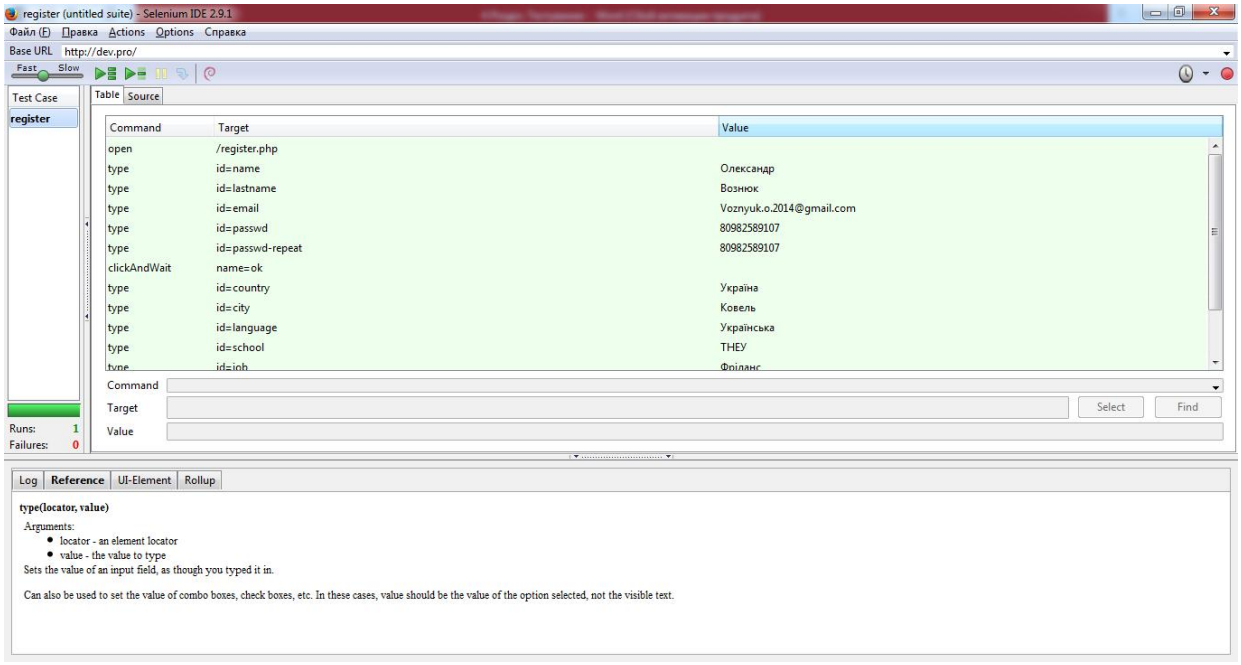


Рисунок 4.1- Тестування реєстрації в Selenium

4.2 Розробка тестів

Під час розробки тестових випадків було спроектовано тести для функціонального тестування, тестування безпеки. Кожен тестовий випадок містить детальні кроки, тестові дані і очікувані результати.

4.2.1 Функціональні тестові випадки

Під час розробки тестів було спроектовано 11 функціональних тестових випадків.

Таблиця 4.1

Функціональні тестові випадки

id	Модуль	Назва	Кроки	Очікуваний результат	Статус

Продовження таблиці 4.1

1	Register	Реєстрація	1. Перейти на сторінку реєстрації 2.Заповнити форму 3.Натиснути кнопку “Реєстрація”	Перенаправлення на сторінку заповнення інформації про користувача	Виконано
2	Register	Заповнення даних про користувача	1. Заповнити форму 2. Натистнути на кнопку “Завершити реєстрацію”	Перенаправлення на сторінку користувача	Виконано
3	Project	Створення нового проекту	1.Натиснути на кнопку “Створити” в шапці сторінки 2.Заповнити форму 3. Натиснути на кнопку “Створити”	Створення нового проекту та добавлення на сторінку користувача	Виконано
4	Project	Запрошення в проект	1.Натиснути на сторінці проекту на кнопку “Запросити” 2.Вибрати зі списку тих, кого потрібно запросити	Відправитися запрошення на участь у проекті	Виконано
5	Project	Запрошення в проект через сторінку користувача	1.Зайти на сторінку користувача, якого хочемо запросити 2.Натиснути на кнопку “Запросити” 3.Вибрати зі списку проект, в який хочемо запросити	Відправитися запрошення на участь у проекті	Виконано

Продовження таблиці 4.1

6	Project	Вигнання з проекту	1.Зайти на сторінку проекту 2.Відкрити список учасників 3.Натиснути на кнопку “Вигнати з проекту ” на проти того користувача, яий нам потрібен	Учасник пропаде зі списку учасників проекту	Виконано
7	Project	Розприділення ролей в проекті	1.Відкрити список учасників 2. Натиснути на кнопку “Посада” 3.Вибрати зі списку посаду 4. Натиснути “Призначити”	Учасник отримає посаду в команді	Виконано
8	Project	Звільнення з посади	1.Відкрити список учасників 2. Натиснути на кнопку “посада” 3. Натиснути на кнопку “Звільнити”	Учасника буде звільнено з посади	Виконано
9	Project	Зміна посади	1.Відкрити список учасників 2. Натиснути на кнопку “Посада” 3.Вибрати зі списку іншу посаду 4. Натиснути “Призначити”	Змінена посада	Виконано

Продовження таблиці 4.1

10	Profile	Контактна інформація	1.Натиснути на посилання “Змінити контактні дані” 2.Заповнити дані 3.Натиснути “Зберегти”	Дані будуть змінені	Виконано
11	Profile	Основна інформація	1.Натиснути в головному блоці на іконку “налаштування” 2. Змінити дані на потрібні 3. натиснути “зберегти”	Інформація буде змінена	Виконано

4.2.2 Тестові випадки тестування сумісності

Під час розробки тестів сумісності було спроектовано 8 тестових випадків.

Таблиця 4.2

Тестові випадки тестування сумісності

id	Назва	Очікуваний результат	Результат	Статус
1	Реєстрація	Вигляд, у всіх браузерах однаковий	Вигляд, у всіх браузерах однаковий	Виконано
2	Сторінка користувача	Вигляд, у всіх браузерах однаковий	В IE 8 версії поїхав блок з контактими	Помилка
3	Сторінка проекту	Вигляд, у всіх браузерах однаковий	Вигляд, у всіх браузерах однаковий	Виконано

Продовження таблиці 4.3

4	Створення проекту	Вигляд, у всіх браузерах однаковий	Вигляд, у всіх браузерах однаковий	Виконано
5	Сторінка користувачів	Вигляд, у всіх браузерах однаковий	Вигляд, у всіх браузерах однаковий	Виконано
6	Сторінка команд	Вигляд, у всіх браузерах однаковий	Вигляд, у всіх браузерах однаковий	Виконано
7	Сторінка компаній	Вигляд, у всіх браузерах однаковий розробки	Вигляд, у всіх браузерах однаковий розробки	Виконано
8	Сторінка Учасників проекту	Вигляд, у всіх браузерах однаковий	Вигляд, у всіх браузерах однаковий	Виконано

4.2.3 Тестові випадки тестування продуктивності

Під час розробки тестів продуктивності було спроектовано 8 тестових випадків.

Таблиця 4.4

Тестові випадки тестування продуктивності

id	Назва	Очікуваний результат	Результат	Статус
1	Завантаження сторінки користувача	Швидкість завантаження не більше 1 сек	0,542	Виконано
2	Завантаження сторінки проекту	Швидкість завантаження не більше 1 сек	0,443	Виконано

Продовження таблиці 4.5

3	Завантаження сторінки команд	Швидкість завантаження не більше 1 сек	0,574	Виконано
4	Завантаження сторінки компаній	Швидкість завантаження не більше 1 сек	0,612	Виконано
5	Завантаження сторінки користувачів	Швидкість завантаження не більше 1 сек	0,489	Виконано
6	Завантаження сторінки реєстрацій	Швидкість завантаження не більше 1 сек	0,375	Виконано
7	Створення нового проекту	Швидкість завантаження не більше 1 сек	0,468	Виконано
8	Завантаження сторінки учасників проекту	Швидкість завантаження не більше 1 сек	0,398	Виконано

4.2.4 Тестові випадки тестування GUI

Під час розробки тестів GUI було спроектовано 9 тестових випадків.

Таблиця 4.6

Функціональні тестові випадки

id	Назва	Кроки	Очікуваний результат	Статус
1	Форма реєстрації	Вигляд, спроектований на початку розробки	Вигляд, спроектований на початку розробки	Виконано

Продовження таблиці 4.6

2	Сторінка користувача	Вигляд, спроектований на початку розробки	Вигляд, спроектований на початку розробки	Виконано
3	Сторінка проекту	Вигляд, спроектований на початку розробки	Вигляд, спроектований на початку розробки	Виконано
4	Форма створення проекту	Вигляд, спроектований на початку розробки	Вигляд, спроектований на початку розробки	Виконано
5	Сторінка управління учасниками проекту	Вигляд, спроектований на початку розробки	Вигляд, спроектований на початку розробки	Виконано
6	Сторінка команд	Вигляд, спроектований на початку розробки	Вигляд, спроектований на початку розробки	Виконано
7	Сторінка компаній	Вигляд, спроектований на початку розробки	Вигляд, спроектований на початку розробки	Виконано
8	Сторінка команд	Вигляд, спроектований на початку розробки	Вигляд, спроектований на початку розробки	Виконано
9	Сторінка редагування даних	Вигляд, спроектований на початку розробки	Вигляд, спроектований на початку розробки	Виконано
10	Форми	Вигляд, спроектований на початку розробки	Вигляд, спроектований на початку розробки	Виконано
11	Кнопки	Вигляд, спроектований на початку розробки	В IE 7.0 в кнопок відсутні заокруглення	Помилка

4.2.5 Тестові випадки тестування безпеки

Під час розробки тестів безпеки було спроектовано 6 тестових випадків.

Таблиця 4.7

Тестові випадки тестування безпеки

id	Назва	Очікуваний результат	Результат	Статус
1	Перевірка на запис в формі SQL запиту	Всі SQL запити будуть видалені з форми і не записані в базу даних	Всі SQL запити видалені з форми і не записані в базу даних	Виконано
2	Перевірка на запис в формі проекту SQL запиту	Всі SQL запити будуть видалені з форми і не записані в базу даних	Всі SQL запити видалені з форми і не записані в базу даних	Виконано
3	Перевірка на цілісність інформації користувача	Доступ до інформації користувача лише за його дозволом	Доступ до інформації користувача лише за його дозволом	Виконано
4	Перевірка на шифрування паролів користувачів	Паролі зашифровані за допомогою функції md5 і записані в базу даних у зашифрованому вигляді.	Паролі зашифровані і записані в базу даних	Виконано

Продовження таблиці 4.7

5	Перевірка на дешифрування паролів	При авторизації введений пароль користувачем шифрується і порівнюється з записаним шифром в базі даних	При авторизації введений пароль користувачем шифрується і порівнюється з записаним шифром в базі даних	Виконано
6	Первірка на доступ до закритих розділів	При переході на закриті сторінки, користувача має перенаправляти на його сторінку.	При переході на закриті сторінки, користувача направляє на його сторінку.	Виконано

4.3 Функціональне тестування

Функціональне тестування – це найбільш трудомісткий етап тестування ПЗ. Суть даного тестування полягає у перевірці всього функціоналу, в даному випадку ми будемо перевіряти наступний функціонал:

- перевірка всіх обов’язкових функцій
- тестування всіх користувацьких форм (реєстрації, авторизації, створення нового стартапу);
- перевірка роботи пошуку;
- перевірка всіх гіперпосилань, та пошук не робочих посилань;
- перевірка завантаження файлів на сервер;

Результат виконання функціонального тестування.

Підсумок тестування:

10 тестів з 11 функціональних тестів наведених в таблиці 4.1 Пройшли успішно, отже функціональне тестування розглядається як частково успішне – 97% тестових випадків пройшли.

Відомі дефекти:

Дефект №1: Коли користувач вводить дані в форму реєстрації, то може ввести декілька разів один і той самий логін.

Опис: Відповідно, до тестового випадку “Перевірка форми”, система повинна при повторному введені одного і того ж логіна інформувати, що користувач з таким логіном вже є.

Подолання дефекту: Даний дефект можна виправити, добавивши в class Register умову, яка буде перевіряти введений користувачем логін на існування ідентичного у базі даних.

4.4 Тестування сумісності

Тестування сумісності – це вид тестування, метою якого є перевірка роботи програми в певному оточенні. Суть даного тестування полягає у перевірці роботи програми в різних середовищах, в даному випадку у різних браузерах. Це дасть змогу з’ясувати необхідні зміни, щоб в подальшому система працювала коректно у всіх браузерах.

Перевірку на сумісність будемо проводити у таких браузерах:

- Google Chrome;
- Opera;
- Mozilla Firefox;
- Internet Explorer (не нижче 7 версії);
- Safari;
- UCBrowser (для Android);
- Safari (для IOS);
- UCBrowser (для Windows phone).

Результат виконання функціонального тестування.

Підсумок тестування:

7 тести з 8 функціональних тестів наведених в таблиці 4.2 Пройшли

успішно, отже тестування на сумісність розглядається як частково успішне – 97% тестових випадків пройшли.

Відомі дефекти:

Дефект №1: В Internet Explorer 7.0 при перегляді головної сторінки користувача, блок з контактною інформацією з'їхав в бік.

Опис: Система повинна при перегляді сторінок в різних браузерах виглядати однаково з мінімальним відхиленням у показниках.

Подолання дефекту: Даний дефект можна виправити, змінивши у файлі стилів позиціонування блоку з fixed на absolute.

4.5 Тестування продуктивності

Тестування продуктивності – це тестування призначене для перевірки на скільки швидко працює програма, або її частина під навантаженням.

Головним критерієм продуктивності веб-додатку є:

- час відповіді сервера;
- час відображення.

При перевірці продуктивності програмного продукту ми будемо використовувати такі види тестування продуктивності:

- навантажувальне тестування;
- стресове тестування;
- тестування стабільності;
- конфігураційне тестування.

Результат виконання тестування продуктивності.

Підсумок тестування:

10 тестів з 10 тестів продуктивності наведених в таблиці 1. Пройшли успішно. Час відповіді сервера – 0,35 сек. , час з'єднання з сервером – 0,14 сек, отже спираючись на результати, можна сказати, що тестування продуктивності пройшло успішно.

4.6 Тестування GUI

Тестування GUI – це перевірка графічного інтерфейсу користувача на відповідність до програмного додатку, як він виглядає, чи виконаний в єдиному стилі. Під час тестування GUI було проведено інтерфейс користувача на:

- відповідність стандартам графічних інтерфейсів;
- тестування з різними розширеннями екранів;
- тестування в обмежених умовах.

Підсумок тестування:

7 тестів з 8 тестів GUI пройшли успішно , отже тестування розглядається як частково успішне – 97% тестових випадків пройшли.

Відомі дефекти:

Дефект №1: В Internet Explorer заокруглення кнопок та блоків не відображається.

Опис: Заокруглення елементів інтерфейсу повинне виглядати скрізь однаково.

Подолання дефекту: Даний дефект можна виправити, добавивши в стилі строчу, що містить наступний код – behavior: url(border-radiuse.htc).

4.7 Тестування безпеки

Тестування безпеки – це тестування в якому перевіряється вразливість програмного додатку, та проводиться аналіз ризиків, пов’язаних з вірусами та атаками хакерів. Під час даного тестування було перевірено програмну систему на:

- конфіденційність;
- цілісність;
- ідентифікація;

- доступність;
- безвідмовність.

Результат виконання функціонального тестування.

Підсумок тестування:

10 тестів з 11 тестів пройшли успішно, отже функціональне тестування розглядається як успішне – 100% тестів пройшли

Умови тестування, які визнавалися успішними були наступні:

- розробка тестів:

1. Всі заплановані тестові випадки розроблено;
2. Покриття тестами програмних вимог досягає 100%;
3. Покриття тестами варіантів використання досягає 100%;
4. тестування:
5. Всі розроблені тестові випадки виконано;
6. Виконано тестування продуктивності, вимоги продуктивності задоволено;
7. Всі внутрішні дефекти виправлені і виправлення підтверджено.
8. Всі наведені умови задоволено, проект вважається успішним.

4.8 Інструкція користувача

Для того що розпочати роботу з програмною системою нам необхідно зареєструватися, для цього потрібно перейти в браузері по посиланні http://dev.pro/register.php . В результаті ми побачимо форму реєстрації (див. рис. 4.2), в якій потрібно заповнити всі поля.

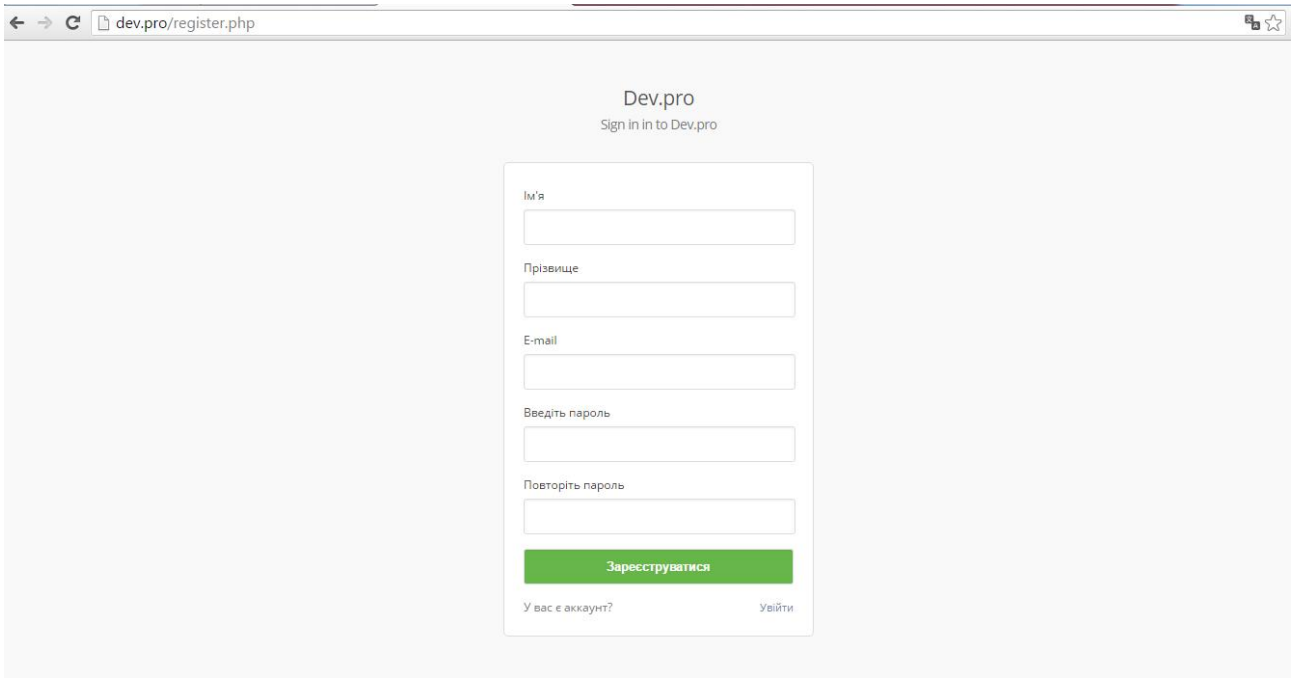


Рисунок 4.2 – Форма реєстрації

Після заповнення форми потрібно натиснути на кнопку “Зареєструватися”. Якщо всі дані були вірно заповнені, то нас направить на сторінку заповнення даних про користувача (див. рис. 4.3).

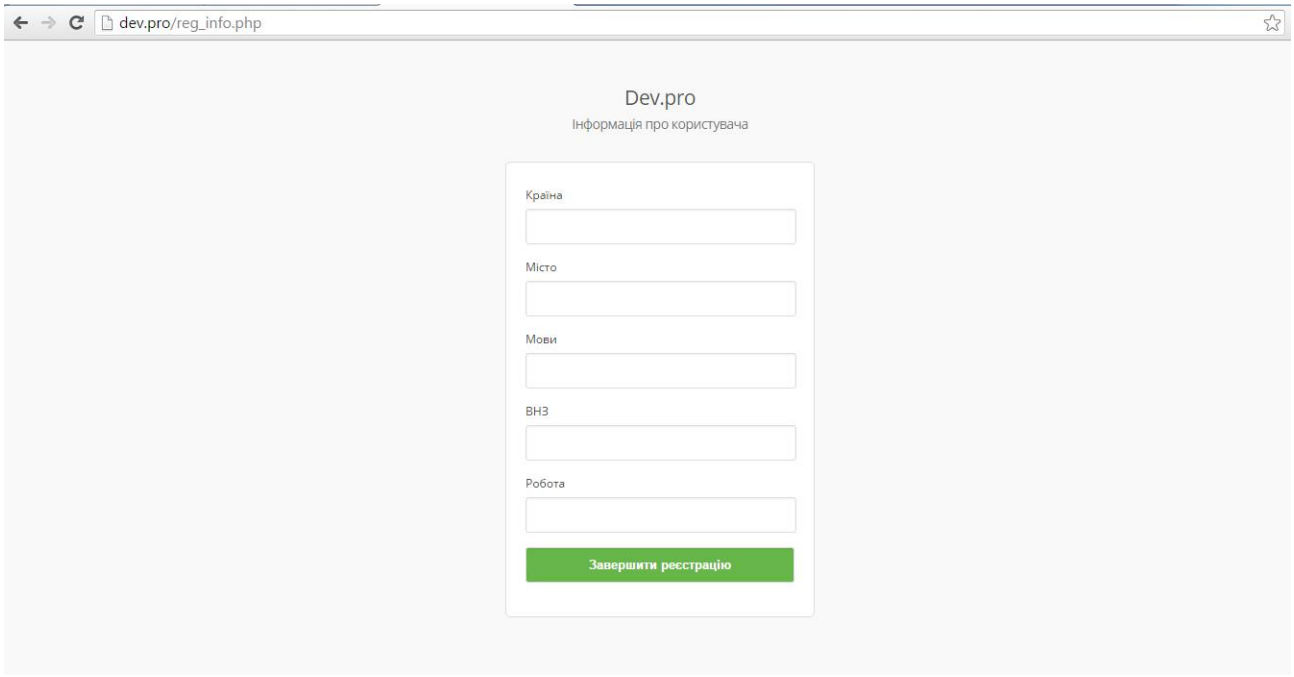


Рисунок 4.3 – Інформація про користувача

Після завершення реєстрації, система направить нас на сторінку

користувача (див. рисунок 4.4).

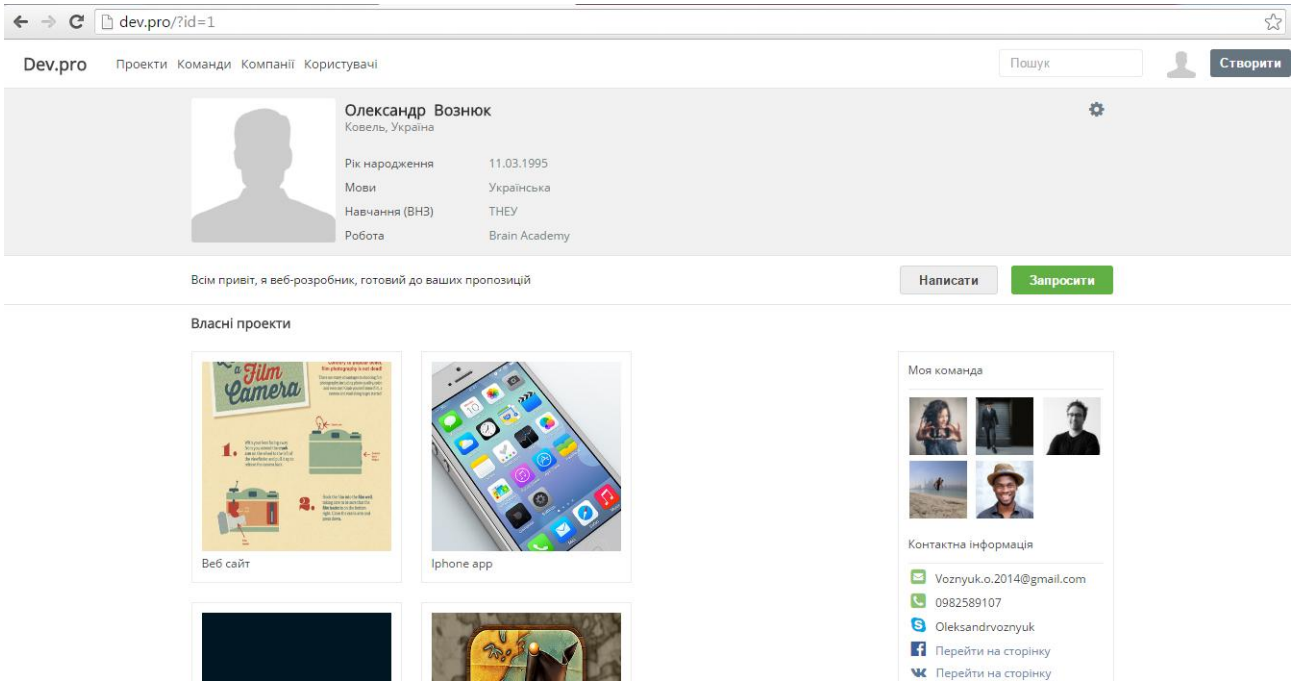


Рисунок 4.4 – Сторінка користувача

Для того щоб редагувати дані користувача потрібно натиснути на іконку “налаштування” з правої сторони в блоці про користувача. Після чого нас направить на сторінку редагування даних (див. ри. 4.5).

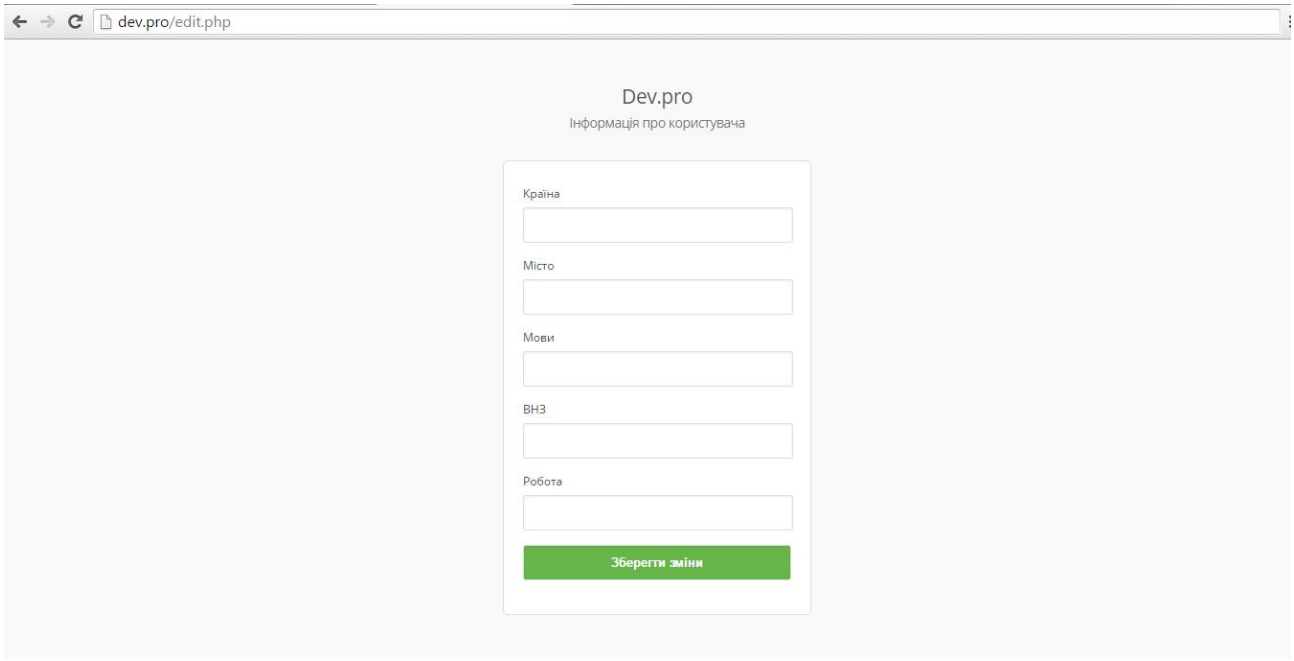


Рисунок 4.5 – Редагування даних

Для того щоб створити новий проект, потрібно в “шапці” сторінки натиснути на кнопку “Створити”. Після чого на направить на сторінку створення нового проекту (див. рис. 4.6)

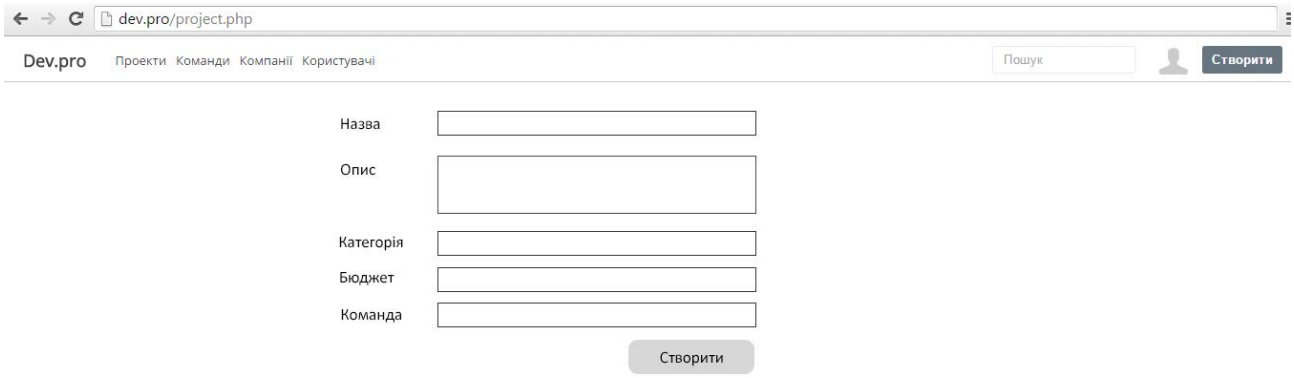


Рисунок 4.6 – Створення нового проекту

Після коректного заповнення форми користувача направить на сторінку створеного проекту (див. рис. 4.7).

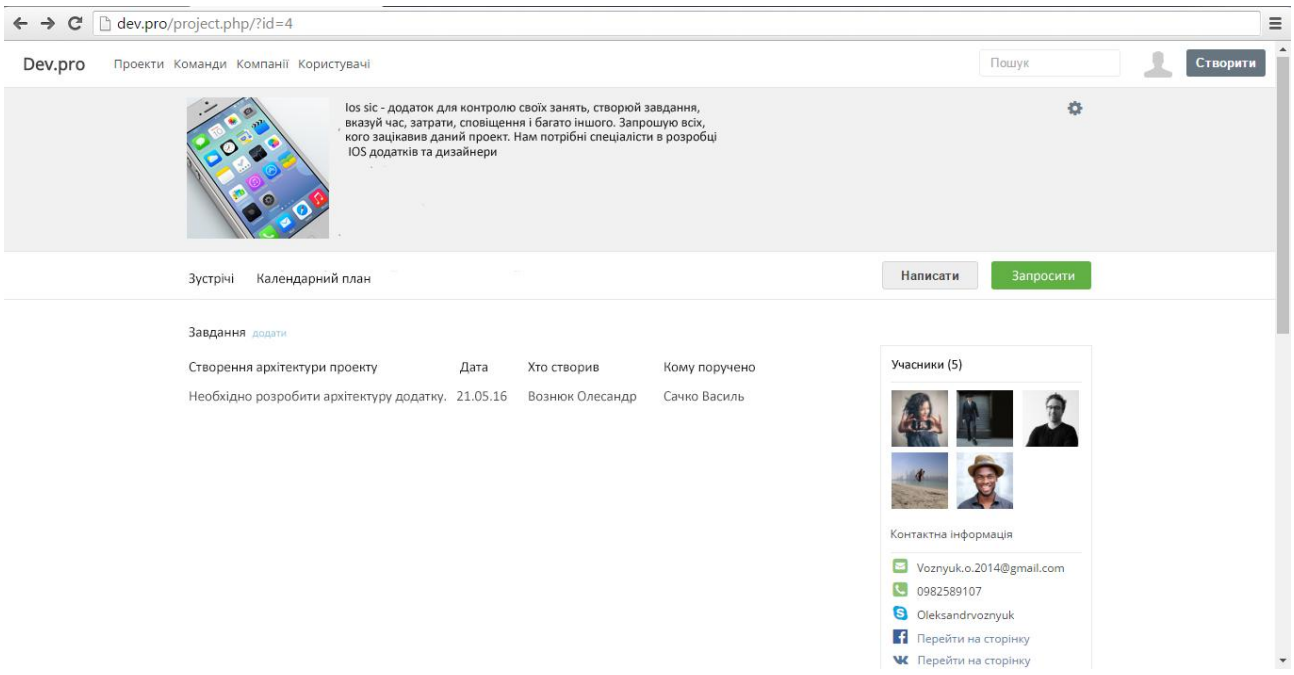


Рисунок 4.7 – Проект

В кінці ми отримаємо створений проект з яким можна працювати, запрошувати людей, створювати задачі, зустрічі.

Для перегляду всіх зареєстрованих користувачів необхідне перейти за посиланням http://dev.pro/users.php (див. рис. 4.8).

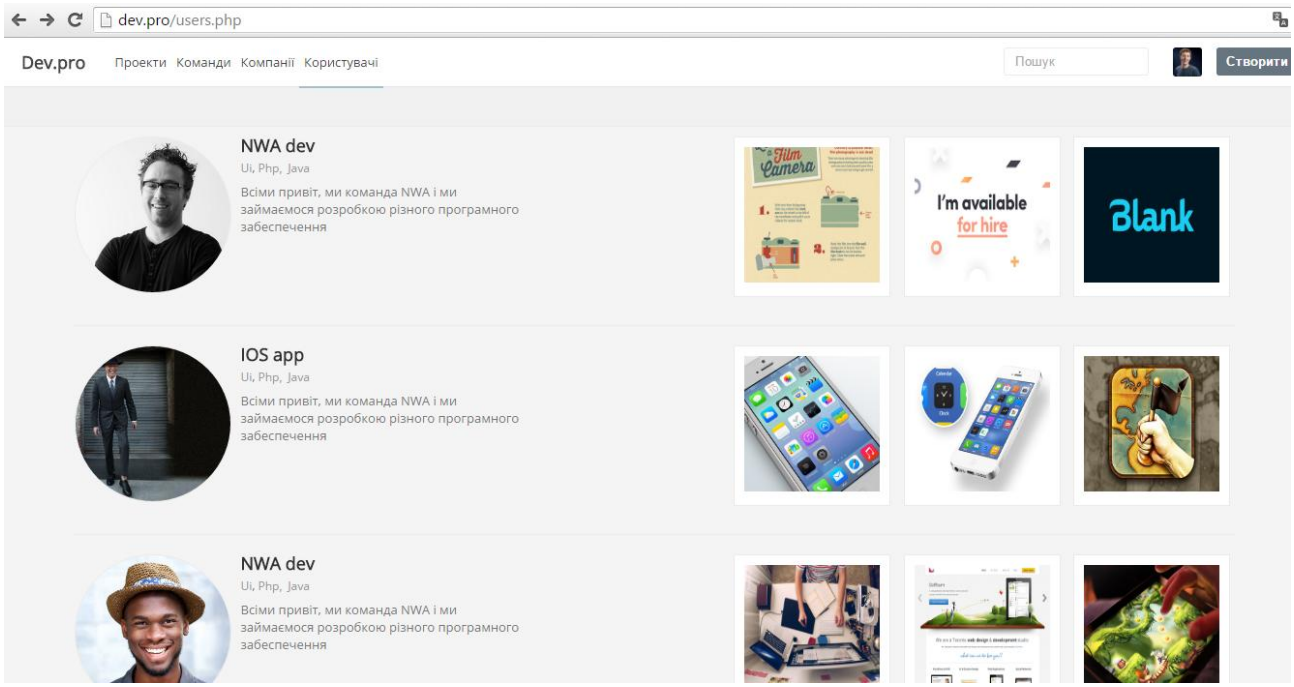


Рисунок 4.8 – Список зареєстрованих користувачів

Щоб переглянути всі створені користувачами проекти необхідно перейти за посиланням http://dev.pro/projects.php (див. рис. 4.9).

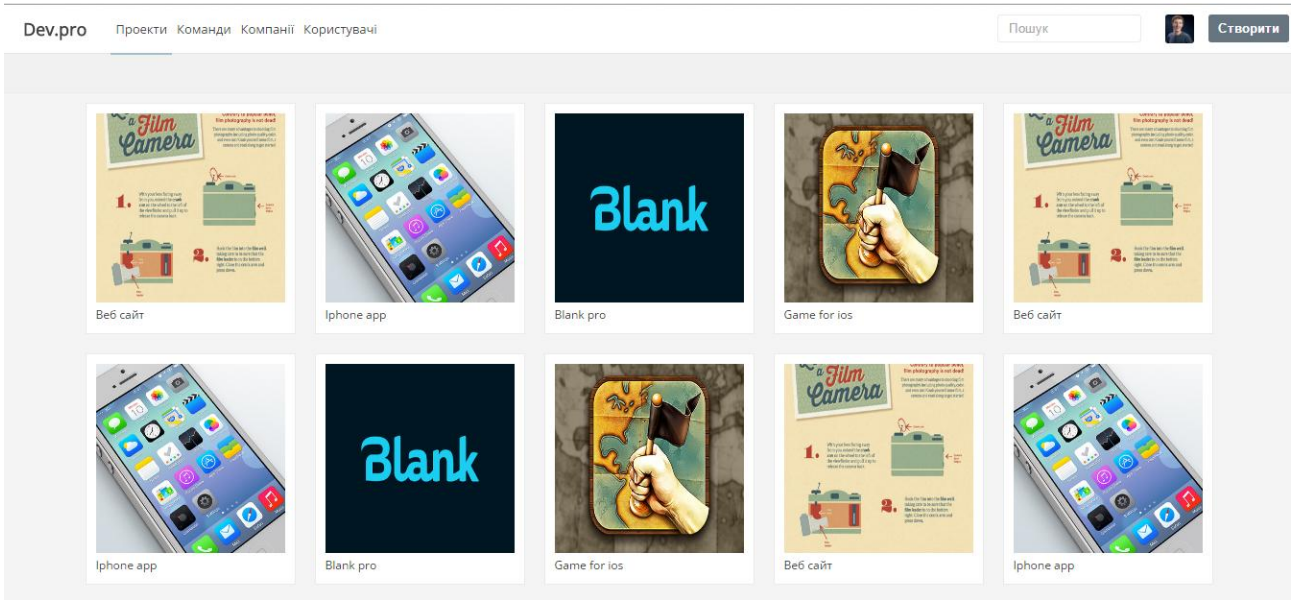


Рисунок 4.9 – Список проектів

Висновок

Провівши тестування програмної системи можна сказати, що це є невід’ємною частиною при розробці програмних продуктів. Було проведено функціональне тестування, де ми перевірили всі функції, що виконує програм. Тестування сумісності – перевірили на сумісність з усіма браузерами. Тестування продуктивності – перевірили як швидко і ефективно працює наша програма. Тестування GUI – перевірили зручність користувацького інтерфейсу. Тестування безпеки – перевірили чи є вразливим наш програмний продукт.

В результаті всіх тестів було виявлені і ліквідовані дефекти в роботі програмної системи

ВИСНОВОК

Робота присвячена дослідженню та розробці веб-орієнтованої системи управління стартапами.

Значний інтерес зосереджений на створенні цікавих проектів, пошуку команд та пошуку фінансування для свого стартапу.

Були отримані наступні результати:

1. Проведений аналіз системи управління стартапами, де ми розглянули актуальність програми. Також були розглянуті аналоги, які є найбільш схожими на дану систему, проведений їхній детальний аналіз і порівняння.

2. Спроектовані класи, архітектрута та структура бази даних додатку.

3. Вибрана мова програмування для реалізації програмного додатку. Організували інтерфейс користувача, розробили класи та методи, які виконували всю логіку програми та створили базу даних для збереження даних користувачів.

4. Проведене тестування програмної системи на функціональність, безпеку, навантаження, GUI.

В результаті виконання дипломної роботи , були вдосконалені знання у проектуванні, тестуванні та розробці програмних систем.

Під час розробки було застосовані на практиці знання в:

- Мові програмування PHP;
- мові програмуванні Javascript;
- СУБД MySQL;
- локальному сервері Open Server;

В кінцевому результаті була отримана веб-орієнтована система управління стартапами, яка дасть змогу усім бажаючим створити власний стартап, розвивати його, залучати в його нових людей, фінансування, ділитися власними розробками, ідеями та надихати інших користувачів на створення цікавих та корисних сервісів.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Поздняев А.С., Власов А.И. Развитие информационно-телекоммуникационного сектора
экономики при гармонизации мировой финансовой системы им. Н.Э. Баумана. серия: Приборостроение. 2010. № 1. - С. 121-125.

2. Корсаков С.Н. Начертание нового способа исследования при помощи машин, сравнивающих идеи / Пер. с франц. под ред. А.С. Михайлова. – М.: МИФИ, 2009, 44 с.

3. Novak J. D., Canas A.J. The Theory Underlying Concept Maps and How to Construct Them. //Technical Report IHMC CmapTools 2006-01 Rev 01-2008, Florida Institute for Human and Machine Cognition, 2008.

4. Буч Г., Рамбо Дж., Джекобсон А. Язык UML. Руководство пользователя.-М.:ДМК ПРЕСС; СПб.:Питер,2004.-429 с.

5. Герман О.В. Введение в теорию экспертных систем и обработку знаний. Учебное посо-бие. Мн.: ДизайнПро, 1995. - 456 с.

6. Сотник С. Л. Конспект лекций по курсу «Основы проектирования систем искусственного интеллекта», 1997-1998.

7. Зандстра М., PHP: объекты, шаблоны и методики программирования, 3-е издание = PHP Objects, Patterns and Practice, Third Edition — М.: «Вильямс», 2010. — С. 560. — ISBN 978-5-8459-1689-1.

8. Суэринг С., Конверс Т., Джойс П. PHP и MySQL. Библия программиста, 2-е издание = PHP 6 and MySQL 6 Bible — М.: «Диалектика», 2010. — 912 с. — ISBN 978-5-8459-1640-2.

9. Кормен Т., Лейзерсон И. Ч., Ривест Р. Л., Штайн К. Алгоритмы: построение и анализ = INTRODUCTION TO ALGORITHMS — 2-е изд. — М.: «Вильямс», 2006. — С. 1296. — ISBN 0-07-013151-1.

10. Кнут Д. Искусство программирования, том 1. Основные алгоритмы = The Art of Computer Programming, vol.1. Fundamental Algorithms — 3-е изд. — М.: «Вильямс», 2006. — С. 720. — ISBN 0-201-89683-4

11. Нильсен Я., Перниче К. Веб-дизайн: анализ удобства использования веб-сайтов по движению глаз = Eyetracking Web Usability — М.: «Вильямс», 2010. — С. 480. — ISBN 978-5-8459-1652-5.

12. Титтел Э., Ноубл Дж. HTML, XHTML и CSS для чайников, 7-е издание = HTML, XHTML & CSS For Dummies, 7th Edition — М.: «Диалектика», 2011. — 400 с. — ISBN 978-5-8459-1752-2.

13. Zakas N., McPeak J., Fawcett J. Professional Ajax — 2nd ed. — Wrox, 2007. — 624 p. — (Programmer to Programmer). — ISBN 0470109491.

14. Lindley C. jQuery Cookbook. Solutions & Examples for jQuery Developers. O'Reilly Media, 2009. – 478 с.

15. Джон К. ВанДик , Мэт Вестгейт. Pro Drupal 7 Development: Third Edition / Todd Tomlinson . John K. VanDyk - Apress, 2010 .

16. Стивен Хольцнер . PHP в примерах. / Стивен Хольцнер . М.: 000 «Бином-Пресс», 2007 г. Пер. с англ. 352 с

17. Ларри Ульман. Ульман Л. Основы программирования на PHP:/Ларри Ульман. Пер. с англ. -М.: ДМК Пресс, 2001. -288 с.: ил. (Самоучитель).

18. Александр Мазуркевич. MB PHP: настольная книга программиста /Александр Мазуркевич, Дмитрий Еловой. — Мн.: Новое знание, 2003. — 480 с.: ил

19. Томсон Лаура. Разработка Web-приложений на PHP и MySQL: Пер. с англ. /Лаура Томсон, Люк Вел.

20. JavaScript. Подробное руководство. Дэвид Флэнаган.

ДОДАТОК А

```
mysql_connect("localhost", "myhost", "myhost");

mysql_select_db("testtable");

if(isset($_POST['submit']))

{

   $err = array();


   if(!preg_match("/^[a-zA-Z0-9]+$/",$_POST['login']))

   {

      $err[] = "Логін з англійських букв";

   }


   if(strlen($_POST['login']) < 3 or strlen($_POST['login']) > 30)

   {

      $err[] = "логін не менше 3 символів і не більше 30";

   }


   $query = mysql_query("SELECT COUNT(user_id) FROM users WHERE user_login='".mysql_real_es
cape_string($_POST['login'])."'");

   if(mysql_result($query, 0) > 0)

   {

      $err[] = "Такий логін вже є";

   }

   if(count($err) == 0)

   {

      $login = $_POST['login'];
```

```
        $password = md5(md5(trim($_POST['password'])));

        mysql_query("INSERT INTO users SET user_login='".$login."', user_password='".$password."'");

        header("Location: login.php"); exit();

    }

    else
    {

        print "<b>Виникла помилка:</b><br>";

        foreach($err AS $error)

        {

            print $error."<br>";

        }

    }

}

?>

session_start();
if (isset($_POST['login'])) {
 $login = $_POST['login'];
 if ($login == '') {
 unset($login);
}
}
    if (isset($_POST['password'])) {
$password=$_POST['password'];
if ($password =='') {
 unset($password);
}
}

if (empty($login) or empty($password))    {
    exit ("Введена не вся інформація!");
    }
    $login = stripslashes($login);
    $login = htmlspecialchars($login);
$password = stripslashes($password);
    $password = htmlspecialchars($password);
```

```
    $login = trim($login);
    $password = trim($password);
    include ("bd.php");/

$result = mysql_query("SELECT * FROM users WHERE login='$login'",$db);    $myrow =
mysql_fetch_array($result);
    if (empty($myrow['password']))
    {
    exit ("Введений логін невірний.");
    }
    else {
    if ($myrow['password']==$password) {
    $_SESSION['login']=$myrow['login'];
    $_SESSION['id']=$myrow['id'];
    echo "Ви увійшли на сайт! <a href='index.php'>Головна</a>";
    }
 else {

    exit ("Пароль не віриний.");
    }
    }
```

login.php

```
<?

function generateCode($length=6) {

    $chars = "abcdefghijklmnopqrstuvwxyzABCDEFGHI JKLMNOPRQSTUVWXYZ0123456789";

    $code = "";

    $clen = strlen($chars) - 1;
    while (strlen($code) < $length) {

            $code .= $chars[mt_rand(0,$clen)];
    }

    return $code;

}


mysql_connect("localhost", "myhost", "myhost");

mysql_select_db("testtable");


if(isset($_POST['submit']))
```

```
{

   $query = mysql_query("SELECT user_id, user_password FROM users WHERE user_login='".mysql_real_escape_string($_POST['login'])."' LIMIT 1");

   $data = mysql_fetch_assoc($query);

   if($data['user_password'] === md5(md5($_POST['password'])))

   {

      $hash = md5(generateCode(10));



      if(!@$_POST['not_attach_ip'])

      {

         $insip = ", user_ip=INET_ATON('".$_SERVER['REMOTE_ADDR']."')";

      }

      mysql_query("UPDATE users SET user_hash='".$hash."' ".$insip." WHERE user_id='".$data['user_id']."'");

      setcookie("id", $data['user_id'], time()+60*60*24*30);

      setcookie("hash", $hash, time()+60*60*24*30);

      header("Location: check.php"); exit();

   }

   else

   {

      print "Не правильний логін чи пароль";

   }

}

?>
```

check.php

```
<?
```

```
mysql_connect("localhost", "myhost", "myhost");

mysql_select_db("testtable");


if (isset($_COOKIE['id']) and isset($_COOKIE['hash']))

{

   $query = mysql_query("SELECT *,INET_NTOA(user_ip) FROM users WHERE user_id = '".intval($_
COOKIE['id'])."' LIMIT 1");

   $userdata = mysql_fetch_assoc($query);


   if(($userdata['user_hash'] !== $_COOKIE['hash']) or ($userdata['user_id'] !== $_COOKIE['id'])<br> or (
($userdata['user_ip'] !== $_SERVER['REMOTE_ADDR'])  and ($userdata['user_ip'] !== "0")))

   {

      setcookie("id", "", time() - 3600*24*30*12, "/");

      setcookie("hash", "", time() - 3600*24*30*12, "/");

      print "Хм, что-то не получилось";

   }

   else

   {

      print "Привет, ".$userdata['user_login'].". Всё работает!";

   }

}

else

{

   print "Включите куки";

}

?>
```

Клас DB

```
<?php
class Db {
    public $host = 'localhost';
    public $host_name = 'development';
    public $host_password = 1111;

    public function connect(){
        mysql_connect($this->host, $this->host_name, $this->host_password)
            or die ("Помилка підключення");
        mysql_select_db("development");
    }

    public  function  write_data($sql){
        $rez = mysql_query($sql);
        if(!$rez) {
            die('Could not enter data: ' . mysql_error());
        }
    }
}
```

Клас Register

```
<?php
require_once "Db.php";

class Register {
    private $name;
    private $last_name;
    private $email;
    private $password;
    private $repeatPassword;

    public function getName(){
        return $this->name;
    }

    public function setName($name){
        $this->name = $name;
    }

    public function getLastName(){
        return $this->last_name;
    }

    public function setLastName($last_name){
        $this->last_name = $last_name;
    }

    public function getEmail(){
```

```
        return $this->email;
    }

    public function setEmail($email){
        $this->email = $email;
    }

    public function getPassword(){
        return $this->password;
    }

    public function setPassword($password){
        $this->password = $password;
    }

    public function getRepeatPassword(){
        return $this->repeatPassword;
    }

    public function setRepeatPassword($repeat_password){
        $this->repeatPassword = $repeat_password;
    }

    public function getId($email){
        $db = new Db();
        $db->connect();
        $sql = "SELECT id,email,password FROM users WHERE email='$email' limit 1";
        $res = mysql_query($sql);
        $val = mysql_fetch_assoc($res);
        return $val['id'];
    }

    public function checkData($user_name,$last_name,$email,$password,$repeat_password){
        if(empty($user_name) || empty($last_name) || empty($email) || empty($password) ||
empty($repeat_password) ){
            echo("Не всі дані заповнено");
        }else{
            if($password === $repeat_password){
                $password = md5($password);
                $db = new Db();
                $db->connect();
                $sql = 'INSERT INTO `users` (`user_name`, `last_name`, `email`, `password`) VALUES
('".$user_name."', '".$last_name."', '".$email."','".$password."');';
                $db->write_data($sql);
                header('Location: http://dev.pro/reg_info.php');
            }else{
                echo('Паролі не співпадають');
            }
```

```
        }
    }

    public function login($email,$password){
        if(empty($email) || empty($password)){
            echo("Не всі поля заповнені");
        }else{
            $db = new Db();
            $db->connect();
            $user = new Register();
            $email = $user->getEmail();
            $sql = "SELECT id,email,password FROM users WHERE email='$email' limit 1";
            $res = mysql_query($sql);
            $val = mysql_fetch_assoc($res);
            if($val['password'] === md5($password)){
                $_SESSION['id'] = $val['id'];
                header('Location: http://dev.pro');
            }else{
                echo('Не віроно введені дані');
            }
        }
    }

    public function writeInfoUser($country,$city,$language,$school,$job){
        if(empty($country) || empty($city) || empty($language) || empty($school) || empty($job)){
            echo("Не всі поля заповнені");
        }else{
            $db = new Db();
            $email = $_SESSION['email'];
            $db->connect();
            $sql = "SELECT id FROM users WHERE email='$email'";
            $res = mysql_query($sql);
            $val = mysql_fetch_assoc($res);
            $id = $val['id'];
            $sql = "UPDATE users SET country = '$country', city = '$city', language = '$language', school =
  '$school',job = '$job' WHERE id='$id'";
            $db->write_data($sql);
        }
    }


<?php

use Pubnub\Pubnub;
use Pubnub\PubnubPAM;

class PAMChannelLevelIntegrationTest extends PAMTestCase
{

    public function setUp()
```

```
{
    parent::setUp();

    $this->pubnub_secret->revoke();

    sleep(5);
}

public function testGrantNoReadNoWrite()
{
    $this->pubnub_secret->grant(0, 0, $this->channel);

    $response = $this->pubnub_secret->audit($this->channel);

    $this->assertEquals('0', $response['payload']['channels'][$this->channel]['r']);
    $this->assertEquals('0', $response['payload']['channels'][$this->channel]['w']);
    $this->assertEquals('channel', $response['payload']['level']);
}

public function testGrantReadNoWrite()
{
    $this->pubnub_secret->grant(1, 0, $this->channel);

    $response = $this->pubnub_secret->audit($this->channel);

    $this->assertEquals('1', $response['payload']['channels'][$this->channel]['r']);
    $this->assertEquals('0', $response['payload']['channels'][$this->channel]['w']);
    $this->assertEquals('channel', $response['payload']['level']);
}

public function testGrantNoReadWrite()
{
    $this->pubnub_secret->grant(0, 1, $this->channel);

    $response = $this->pubnub_secret->audit($this->channel);

    $this->assertEquals('0', $response['payload']['channels'][$this->channel]['r']);
    $this->assertEquals('1', $response['payload']['channels'][$this->channel]['w']);
    $this->assertEquals('channel', $response['payload']['level']);
}

public function testGrantReadWrite()
{
    $this->pubnub_secret->grant(1, 1, $this->channel);

    $response = $this->pubnub_secret->audit($this->channel);

    $this->assertEquals('1', $response['payload']['channels'][$this->channel]['r']);
    $this->assertEquals('1', $response['payload']['channels'][$this->channel]['w']);
    $this->assertEquals('channel', $response['payload']['level']);
}

public function testRevoke()
{
    $this->pubnub_secret->grant(1, 1, $this->channel, 10);
```

```
        $response = $this->pubnub_secret->revoke($this->channel);

        $this->assertEquals('0', $response['payload']['channels'][$this->channel]['w']);
        $this->assertEquals('0', $response['payload']['channels'][$this->channel]['r']);
    }
}

class PAMUserIntegrationTest extends PAMTestCase
{
    public function setUp()
    {
        parent::setUp();

        $this->pubnub_secret->revoke();

        sleep(5);
    }

    public function testGrantNoReadNoWrite()
    {
        $this->pubnub_secret->grant(0, 0, $this->channel, self::$access_key);

        $response = $this->pubnub_secret->audit($this->channel, self::$access_key);

        $this->assertEquals('0', $response['payload']['auths'][self::$access_key]['r']);
        $this->assertEquals('0', $response['payload']['auths'][self::$access_key]['w']);
        $this->assertEquals('user', $response['payload']['level']);
        $this->assertEquals($this->channel, $response['payload']['channel']);
    }

    public function testGrantReadNoWrite()
    {
        $this->pubnub_secret->grant(1, 0, $this->channel, self::$access_key);

        $response = $this->pubnub_secret->audit($this->channel, self::$access_key);

        $this->assertEquals('1', $response['payload']['auths'][self::$access_key]['r']);
        $this->assertEquals('0', $response['payload']['auths'][self::$access_key]['w']);
        $this->assertEquals('user', $response['payload']['level']);
    }

    public function testGrantNoReadWrite()
    {
        $this->pubnub_secret->grant(0, 1, $this->channel, self::$access_key);

        $response = $this->pubnub_secret->audit($this->channel, self::$access_key);

        $this->assertEquals('0', $response['payload']['auths'][self::$access_key]['r']);
        $this->assertEquals('1', $response['payload']['auths'][self::$access_key]['w']);
        $this->assertEquals('user', $response['payload']['level']);
    }

    public function testGrantReadWrite()
    {
        $this->pubnub_secret->grant(1, 1, $this->channel, self::$access_key);
```

```
    $response = $this->pubnub_secret->audit($this->channel, self::$access_key);

    $this->assertEquals('1', $response['payload']['auths'][self::$access_key]['r']);
    $this->assertEquals('1', $response['payload']['auths'][self::$access_key]['w']);
    $this->assertEquals('user', $response['payload']['level']);
}

public function testAuditNoAuth()
{
    // granting rw access to admin, r to user, non-avail to users w\o auth key
    $this->pubnub_secret->grant(1, 1, $this->channel, 'admin_key', 10);
    $this->pubnub_secret->grant(1, 0, $this->channel, 'user_key', 10);
    $this->pubnub_secret->grant(0, 0, $this->channel);

    $response = $this->pubnub_secret->audit($this->channel);

    $this->assertEquals('1', $response['payload']['channels'][$this->channel]['auths']['admin_key']['w']);
    $this->assertEquals('0', $response['payload']['channels'][$this->channel]['auths']['user_key']['w']);
    $this->assertEquals('0', $response['payload']['channels'][$this->channel]['w']);
    $this->assertEquals('0', $response['payload']['channels'][$this->channel]['r']);
}

/**
 * @group pam
 * @group pam-user
 */
public function testNewInstancesWithAuthKey()
{
    $this->pubnub_secret->grant(1, 1, $this->channel, 'admin_key', 10);
    $this->pubnub_secret->grant(1, 0, $this->channel, 'user_key', 10);
    $this->pubnub_secret->grant(0, 0, $this->channel, null, 10);

    $nonAuthorizedClient = new Pubnub(array(
        'subscribe_key' => self::$subscribe,
        'publish_key' => self::$publish
    ));

    $authorizedClient = new Pubnub(array(
        'subscribe_key' => self::$subscribe,
        'publish_key' => self::$publish,
        'auth_key' => 'admin_key'
    ));

    $authorizedResponse = $authorizedClient->publish($this->channel, 'hi');
    $nonAuthorizedResponse = $nonAuthorizedClient->publish($this->channel, 'hi');

    $this->assertEquals(1, $authorizedResponse[0]);
    $this->assertEquals(403, $nonAuthorizedResponse['status']);

    $nonAuthorizedClient->setAuthKey('admin_key');
    $nonAuthorizedResponse = $nonAuthorizedClient->publish($this->channel, 'hi');

    $this->assertEquals(1, $nonAuthorizedResponse[0]);
}

public function testRevoke()
```

```
    {
        $this->pubnub_secret->grant(1, 1, $this->channel, 'admin_key');
        $this->pubnub_secret->revoke($this->channel, 'admin_key');

        $response = $this->pubnub_secret->audit($this->channel, 'admin_key');

        $this->assertEquals('0', $response['payload']['auths']['admin_key']['w']);
        $this->assertEquals('0', $response['payload']['auths']['admin_key']['r']);
    }
}

class PAMChannelGroupTest extends PAMTestCase
{
    protected $channelGroup;
    protected $namespace;
    protected $authKey;

    public function setUp()
    {
        parent::setUp();

        $this->channelGroup = "ptest-" . rand();
        $this->namespace = 'ptest-namespace';
        $this->authKey = "user-ptest";
    }

    public function testGrantAllChannelGroup()
    {
        $this->pubnub_secret->pamGrantChannelGroup(true, true, $this->channelGroup);
        $response = $this->pubnub_secret->pamAuditChannelGroup($this->channelGroup);
        $auths = $response["payload"]["channel-groups"][$this->channelGroup];

        $this->assertEquals("channel-group", $response["payload"]["level"]);
        $this->assertEquals("0", $auths["w"]);
        $this->assertEquals("1", $auths["r"]);
        $this->assertEquals("1", $auths["m"]);
    }

    public function testGrantUserChannelGroup()
    {
        $this->pubnub_secret->pamGrantChannelGroup(true, true, $this->channelGroup, $this->authKey);
        $response = $this->pubnub_secret->pamAuditChannelGroup($this->channelGroup, $this->authKey);
        $auths = $response["payload"]["auths"][$this->authKey];

        $this->assertEquals("channel-group+auth", $response["payload"]["level"]);
        $this->assertEquals("0", $auths["w"]);
        $this->assertEquals("1", $auths["r"]);
        $this->assertEquals("1", $auths["m"]);
    }
}

class PAMSubKeyLevelIntegrationTest extends PAMTestCase
{
    public function setUp()
    {
        parent::setUp();
```

```
    $this->pubnub_secret->revoke();

    sleep(5);
}

public function testGrantNoReadNoWrite()
{
    $this->pubnub_secret->grant(0, 0);

    $response = $this->pubnub_secret->audit();

    $this->assertEquals('0', $response['payload']['r']);
    $this->assertEquals('0', $response['payload']['w']);
    $this->assertEquals('subkey', $response['payload']['level']);
}

public function testGrantReadNoWrite()
{
    $this->pubnub_secret->grant(1, 0);

    $response = $this->pubnub_secret->audit();

    $this->assertEquals('1', $response['payload']['r']);
    $this->assertEquals('0', $response['payload']['w']);
    $this->assertEquals('subkey', $response['payload']['level']);
}

public function testGrantNoReadWrite()
{
    $this->pubnub_secret->grant(0, 1);

    $response = $this->pubnub_secret->audit();

    $this->assertEquals('0', $response['payload']['r']);
    $this->assertEquals('1', $response['payload']['w']);
    $this->assertEquals('subkey', $response['payload']['level']);
}


public function testGrantReadWrite()
{
    $this->pubnub_secret->grant(1, 1);

    $response = $this->pubnub_secret->audit();

    $this->assertEquals('1', $response['payload']['r']);
    $this->assertEquals('1', $response['payload']['w']);
    $this->assertEquals('subkey', $response['payload']['level']);
}


public function testRevoke()
{
    $this->pubnub_secret->grant(1, 1);
```

```
        $response = $this->pubnub_secret->revoke();

        $this->assertEquals('0', $response['payload']['w']);
        $this->assertEquals('0', $response['payload']['r']);
        $this->assertEquals('subkey', $response['payload']['level']);
    }
}
```

```
  var getMaxHeight = function ($elms) {
    var maxHeight = 0;
    $elms.each(function () {

      var height = $(this).height();
      if (height > maxHeight) {
        maxHeight = height;
      }
    });
    return maxHeight;
  };

  function isValidDate(value, userFormat) {

    userFormat = userFormat || 'mm/dd/yyyy';

    var delimiter = /[^mdy]/.exec(userFormat)[0];

    var theFormat = userFormat.split(delimiter);

    var theDate = value.split(delimiter);

    function isDate(date, format) {
      var m, d, y, i = 0, len = format.length, f;
      for (i; i < len; i++) {
        f = format[i];
        if (/m/.test(f)) m = date[i];
        if (/d/.test(f)) d = date[i];
        if (/y/.test(f)) y = date[i];
      }
      return (
        m > 0 && m < 13 &&
        y && y.length === 4 &&
        d > 0 &&
        d <= (new Date(y, m, 0)).getDate()
      );
    }

    return isDate(theDate, theFormat);
  }

  $.fn.fadeAll = function (ops) {
    var o = $.extend({
      delay: 500,
      speed: 500,
```

```
    ease: 'swing')
  }, ops);
  var $el = this;
  for (var i=0, d=0, l=$el.length; i<l; i++, d+=o.delay) {
    $el.eq(i).delay(d).fadeIn(o.speed, o.ease);
  }
  return $el;
}

function checkmail()
{
txt=document.mail.address.value;
if (txt == "") {
    alert("");
    return false
    }
if (txt.indexOf(".") == -1) {
    alert();
    return false
    }
if((txt.indexOf(",")>=0)||(txt.indexOf(";")>=0)||(txt.indexOf(" ")>=0)){
    alert('');
    return false
    }
dog = txt.indexOf("@");
    if (dog == -1) {
    alert(""@\".");
    return false
    }
if ((dog < 1) || (dog > txt.length - 5)) {
    alert("");
    return false
    }
if ((txt.charAt(dog - 1) == '.') || (txt.charAt(dog + 1) == '.')) {
alert();
    return false
    }
alert('');
```

ДОДАТОК Б

```
CREATE TABLE IF NOT EXISTS `my_friends` (
  `id` int(12) NOT NULL AUTO_INCREMENT,
  `username` varchar(200) NOT NULL,
  `friend` varchar(200) NOT NULL,
  PRIMARY KEY (`id`)
) ENGINE=MyISAM DEFAULT CHARSET=latin1 AUTO_INCREMENT=1 ;


CREATE TABLE IF NOT EXISTS `friend_request` (
  `id` int(12) NOT NULL AUTO_INCREMENT,
  `username` varchar(200) NOT NULL,
  `friend` varchar(200) NOT NULL,
  PRIMARY KEY (`id`)
) ENGINE=MyISAM DEFAULT CHARSET=latin1 AUTO_INCREMENT=1 ;


CREATE TABLE IF NOT EXISTS `friendship_system_users_table` (
  `id` int(100) NOT NULL AUTO_INCREMENT,
  `fullname` varchar(200) NOT NULL,
  `username` varchar(200) NOT NULL,
  `email` varchar(200) NOT NULL,
  `password` varchar(200) NOT NULL,
  PRIMARY KEY (`id`)
) ENGINE=MyISAM  DEFAULT CHARSET=latin1 AUTO_INCREMENT=1 ;

	SET @OLD_UNIQUE_CHECKS=@@UNIQUE_CHECKS, UNIQUE_CHECKS=0;
	SET @OLD_FOREIGN_KEY_CHECKS=@@FOREIGN_KEY_CHECKS,
FOREIGN_KEY_CHECKS=0;
	SET @OLD_SQL_MODE=@@SQL_MODE,
SQL_MODE='TRADITIONAL,ALLOW_INVALID_DATES';


	-- Schema mydb
	-- Schema mydb
	CREATE SCHEMA IF NOT EXISTS `mydb` DEFAULT CHARACTER SET utf8 ;
	USE `mydb` ;


	-- Table `mydb`.`Task`
	CREATE TABLE IF NOT EXISTS `mydb`.`Task` (
	  `id` INT NOT NULL,
	  `Name` VARCHAR(45) NULL,
	  `Descrition` VARCHAR(45) NULL,
```

```
  `User` VARCHAR(45) NULL,
  PRIMARY KEY (`id`))
ENGINE = InnoDB;

-- Table `mydb`.`news`
CREATE TABLE IF NOT EXISTS `mydb`.`news` (
  `id` INT NOT NULL,
  `name` VARCHAR(45) NULL,
  `text` VARCHAR(45) NULL,
  PRIMARY KEY (`id`))
ENGINE = InnoDB;

-- Table `mydb`.`Project`
CREATE TABLE IF NOT EXISTS `mydb`.`Project` (
  `id` INT NOT NULL,
  `Name` VARCHAR(45) NULL,
  `Description` VARCHAR(45) NULL,
  `type` VARCHAR(45) NULL,
  `Task_id` INT NOT NULL,
  `news_id` INT NOT NULL,
  PRIMARY KEY (`id`),
  INDEX `fk_Project_Task1_idx` (`Task_id` ASC),
  INDEX `fk_Project_news1_idx` (`news_id` ASC),
  CONSTRAINT `fk_Project_Task1`
    FOREIGN KEY (`Task_id`)
    REFERENCES `mydb`.`Task` (`id`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION,
  CONSTRAINT `fk_Project_news1`
    FOREIGN KEY (`news_id`)
    REFERENCES `mydb`.`news` (`id`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION)
ENGINE = InnoDB;


-- Table `mydb`.`User_list`
CREATE TABLE IF NOT EXISTS `mydb`.`User_list` (
```

```
  `id` INT NOT NULL,
  `Name` VARCHAR(45) NULL,
  `Sename` VARCHAR(45) NULL,
  `Specialization` VARCHAR(45) NULL,
  PRIMARY KEY (`id`))
ENGINE = InnoDB;



-- Table `mydb`.`Group`
CREATE TABLE IF NOT EXISTS `mydb`.`Group` (
  `id` INT NOT NULL,
  `descrition` VARCHAR(45) NULL,
  `Name` VARCHAR(45) NULL,
  `Admin` VARCHAR(45) NULL,
  `User_list_id` INT NOT NULL,
  `news_id` INT NOT NULL,
  PRIMARY KEY (`id`),
  INDEX `fk_Group_User_list1_idx` (`User_list_id` ASC),
  INDEX `fk_Group_news1_idx` (`news_id` ASC),
  CONSTRAINT `fk_Group_User_list1`
    FOREIGN KEY (`User_list_id`)
    REFERENCES `mydb`.`User_list` (`id`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION,
  CONSTRAINT `fk_Group_news1`
    FOREIGN KEY (`news_id`)
    REFERENCES `mydb`.`news` (`id`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION)
ENGINE = InnoDB;



-- Table `mydb`.`team`
CREATE TABLE IF NOT EXISTS `mydb`.`team` (
  `id` INT NOT NULL,
  `name` VARCHAR(45) NULL,
  `count_users` INT NULL,
```

```
  `User_list_id` INT NOT NULL,
  PRIMARY KEY (`id`),
  INDEX `fk_team_User_list1_idx` (`User_list_id` ASC),
  CONSTRAINT `fk_team_User_list1`
    FOREIGN KEY (`User_list_id`)
    REFERENCES `mydb`.`User_list` (`id`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION)
ENGINE = InnoDB;

-- Table `mydb`.`User`
CREATE TABLE IF NOT EXISTS `mydb`.`User` (
  `id` INT NOT NULL,
  `Name` VARCHAR(45) NULL,
  `Sename` VARCHAR(45) NULL,
  `Project_id` INT NOT NULL,
  `Group_id` INT NOT NULL,
  `team_id` INT NOT NULL,
  `news_id` INT NOT NULL,
  PRIMARY KEY (`id`, `Project_id`, `Group_id`),
  INDEX `fk_User_Project_idx` (`Project_id` ASC),
  INDEX `fk_User_Group1_idx` (`Group_id` ASC),
  INDEX `fk_User_team1_idx` (`team_id` ASC),
  INDEX `fk_User_news1_idx` (`news_id` ASC),
  CONSTRAINT `fk_User_Project`
    FOREIGN KEY (`Project_id`)
    REFERENCES `mydb`.`Project` (`id`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION,
  CONSTRAINT `fk_User_Group1`
    FOREIGN KEY (`Group_id`)
    REFERENCES `mydb`.`Group` (`id`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION,
  CONSTRAINT `fk_User_team1`
    FOREIGN KEY (`team_id`)
    REFERENCES `mydb`.`team` (`id`)
    ON DELETE NO ACTION
```

```
    ON UPDATE NO ACTION,
  CONSTRAINT `fk_User_news1`
    FOREIGN KEY (`news_id`)
    REFERENCES `mydb`.`news` (`id`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION)
ENGINE = InnoDB;


-- Table `mydb`.`table1`
CREATE TABLE IF NOT EXISTS `mydb`.`table1` (
)
ENGINE = InnoDB;


SET SQL_MODE=@OLD_SQL_MODE;
SET FOREIGN_KEY_CHECKS=@OLD_FOREIGN_KEY_CHECKS;
SET UNIQUE_CHECKS=@OLD_UNIQUE_CHECKS;
```