МІНІСТЕРСТВО ОСВІТИ ТА НАУКИ УКРАЇНИ

ХАРКІВСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ РАДІОЕЛЕКТРОНІКИ

Факультет комп’ютерних наук

Кафедра програмної інженерії

КУРСОВА РОБОТА

пояснювальна записка

з дисципліни «Аналіз та рефакторінг коду програмного забезпечення»

тема роботи: «Система SmartWater»

Студент гр. ПІ-14-5 Бабікян С.В.

Керівник роботи к.т.н., доц. каф. ПІ Лещинський В.О.

Роботу захищено «\_\_»\_\_\_\_\_\_2017 р.

з оцінкою \_\_\_\_\_\_\_\_\_\_\_\_\_

Комісія:

к.т.н., доц. каф. ПІ Лещинський В.О.\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

к.т.н., доц. каф. ПІ Лещинська І.О. \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

к.т.н., ст.викл. каф. ПІ Русакова Н.Є. \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Харків 2017 р.

Харківський національний університет радіоелектроніки .

Факультет комп’ютерних наук Кафедра програмної інженерії .

Спеціальність Програмна інженерія .

Курс 3 Семестр 6 .

Навчальна дисципліна Аналіз та рефакторінг коду програмного забезпечення .

ЗАВДАННЯ

НА КУРСОВУ РОБОТУ СТУДЕНТОВІ

Бабікяну Симону Ваграмовичу .

1. Тема роботи:    *«SmartWater»* .

2. Термін узгодження завдання курсової роботи «\_\_\_»\_\_\_\_\_\_\_\_\_\_\_ 2017 р.

3. Термін здачі студентом закінченої роботи «\_\_\_»\_\_\_\_\_\_\_\_\_\_\_ 2017 р.

4. Вихідні дані до проекту (роботи): *методичні вказівки до виконання курсової роботи,\_вимоги до інформаційної системи, постановка задачі, предметна область, використовувати ОС Android, середовище об'єктно-орієнтованого проектування Android Studio . »*

5. Зміст пояснювальної записки *(перелік питань, що належить розробити)* *вступ, аналіз предметної області і постановка задачі, моделювання програмного продукту, опис реалізації програмного продукту, опис інтерфейсу та функціоналу* .

6. Перелік графічного матеріалу (з точним зазначенням обов’язкових креслень) *Діаграма варіантів використання, діаграма   класів,   діаграма  станів діаграма компонентів, діаграма  розгортання,  діаграма послідовностей, діаграма кооперацій, діаграма активності, діаграма об’єктів, скріншоти програми, скріншоти інтерфейсу програмної системи, результати тестування програмної системи. »* .

КАЛЕНДАРНИЙ ПЛАН

|  |  |  |  |
| --- | --- | --- | --- |
| № | Назва етапів курсової роботи | Термін виконання етапів проекту (роботи) | Примітки |
| 1 | Функціональна специфікація програми | 19.02.2017 | Виконано |
| 2 | Проектування програми | 16.03.2017 | Виконано |
| 3 | Кодування програми | 27.03.2017 | Виконано |
| 4 | Тестування програми | 21.04.2017 | Виконано |
| 5 | Оформлення пояснювальної записки | 25.04.2017 | Виконано |
| 6 | Захист курсової роботи | 19.05.2017 | Виконано |

Дата видачі завдання «\_\_\_\_\_\_»\_\_\_\_\_\_\_\_\_\_\_\_\_\_20\_\_\_ р.

Керівник \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ к.т.н., доц. каф. ПІ Лещинський В.О.

Завдання прийняв до виконання

Студент гр. ПІ-14-5 Бабікян С.В.

РЕФЕРАТ

Пояснювальна записка до курсової роботи: 32 с., 4 роз., 19 рис., 10 джерел.

Метою роботи є проектування та розробка системи слідкування стану автоматів з водоу.

Методи розробки базуються на орієнтованому підході до розробки програмного забезпечення, технології̈ Android, MS MySQL Server базі даних та середовищі розробки Android Studio.

Методом вирішення є створення веб серверу за допомогою технології Java, розробка клієнтського додатка за допомогою мови програмування Java, Web клієнта за допомогою мови программування JavaScript(HTML, CSS).

Результатом курсового проекту є повноцінна система, що складається з Web – клієнту, мобільного клієнту та серверу, яка дозволить слідкувати за станом автоматів з водою.

ДОДАТОК, Android, Java, MS MySQL Server, API, JavaScript.

ЗМІСТ

[Вступ 6](#_Toc452657151)

[1 Аналіз та постановка задачі 8](#_Toc452657152)

[2 Моделювання програмного продукту 13](#_Toc452657153)

[3 Реалізація програмного продукту 22](#_Toc452657154)

[3.1 Загальні відомості про систему 22](#_Toc452657155)

[3.2 Серверна частина 23](#_Toc452657156)

[3.3 Мобільний клієнт 24](#_Toc452657157)

[3.4 Веб сайт 24](#_Toc452657158)

[4 Інтерфейс і функціонал 25](#_Toc452657159)

[4.1 Веб сайт 25](#_Toc452657160)

[4.2 Мобільний клієнт 29](#_Toc452657161)

[Висновки 31](#_Toc452657162)

[Перелік використаних джерел 32](#_Toc452657163)

# ВСТУП

Вода-це життя. Вона є найважливішим розчинником, бере активну участь в обмінних процесах. Але водопровідна вода потребує додаткової очистки. Це пояснює необхідність використання спеціального обладнання. Сьогодні автомати води поширені повсюдно. Вони дозволяють отримувати покупцям воду, очищену від домішок. В її хімічний склад входять життєво важливі мінеральні компоненти, кількість яких збалансовано з урахуванням потреб людського організму.

Важливим моментом є своєчасне «оновлення» води в автоматах. Розумна система «SMARTWATER» стежитиме за кількістю води в автоматі, сповіщаючи про те, що кількість води в автоматі досягло мінімуму. Після оповіщення, машини що розвозять воду, отримують інформацію про те, що автомат має не достатню кількість води, отримує місце розташування автомата і його статистику (кількість людей купуютчих воду, скільки літрів купує певний покупець і т.д.). інформація про автомати з водою буде розміщуватися на карті, в режимі онлайн можна подивитися всю статистику бажаного автомата.

1 АНАЛІЗ ТА ПОСТАНОВКА ЗАДАЧІ

Створення системи для «трекінгу» інформації автоматів з водою досить важка задача яка містить у собі:

* розробку апаратного рішення (розробка прототипу та перевірка даних, що він передає);
* розробку програмного рішення (веб-додаток, мобільний додаток, серверна частина);
* розробку інтерфейсу їх взаємодії (часто розподілений між першими двома пунктами).

Основною задачею, що постане перед данною системою, буде організація та отримання данних про автомат з водою, отримання статистики використання та висилання повідомлень якщо є несправності.

До основного функціоналу, що надаватиме дане рішення, входить:

* отримання інформації про автомат з водою;
* отримання повідомлень;
* можливість перегляду місцеположення автоматів з водою;
* моєливість авторизації;
* відправлення інформації про автомат з водою.

Такий набір параметрів є достатнім для використання як зі сторони користувача, так і зі сторони організатора кампанії.

Для більш зручного використання даної системи слід створити додаток, що матиме сервер, веб-клієнт та мобільний клієнт. Це дозволить швидко та зручно переглядати інформацію про автомати з водою. Система повинна мати швидкий доступ до бази, надавати можливість зручного доступу до інформації, та надавати можливість користуватися системою та моніторити данні по автоматам з водою.

Програмний продукт має включати базу даних, яка зберігає данні про автомати та користувачів, сервер, що обробляє запит та віддає відгук.

Сервер повинен реалізовувати наступні функції:

* надавати інформацію про автомати з водою;
* створювати автомати;
* обробляти запит від клієнта;
* повертати результати до клієнта.

Він повинен бути реалізований за допомогою наступних технологій:

* технологія Spring;
* драйвер до бази данних SQL Driver for Java.

В якості веб-сервера для віддачі статичних даних використовуватиметься веб-сервер на базі Java, характерною рисою якого є стійкість при високих завантаженості, що дозволить заздалегідь підготуватись до можливих ситуацій. Він працюватиме у зв'язці з фреймворком TomCat. Таке рішення дозволяє нам додати реалізацію підходу MVC і також додатково створити велику кількість модулів:

* + модуль юзера;
  + модуль автомату;
  + модуль бази данних.

Загалом, система зберігатиме наступні дані:

* основні дані користувача (логін, пароль);
* інформацію про автомати з водою(місцеположення, ИД, рівень води, останнє оновлення води, кількість користувачів)

На веб-сайт має реалізувати наступний функціонал:

* авторизація користувачів;
* відображення автоматів з водою та їх детальна інформація.

Мобільний додаток на Android має реалізувати наступний функціонал:

* авторизація користувачів:
* отримання списку автоматів з водою;
* отримання повідомлень.

# 2 МОДЕЛЮВАННЯ ПРОГРАМНОГО ПРОДУКТУ

Під час моделювання системи була проведена побудова декількох діаграм, що цілком відображаюсь роботу та функціонал майбутньої системи. Кожна діаграма відображує систему з конкретної сторони, і у сукупності це дає змогу отримати потрібну інформацію для розробки системи.

На діаграмі комунікацій (див. рис. 2.1) зображено послідовність використання системи. Гість може зареєструватися або увійти до системи чи переглянути інформацію. Авторизований користувач може також переглянути інформацію.

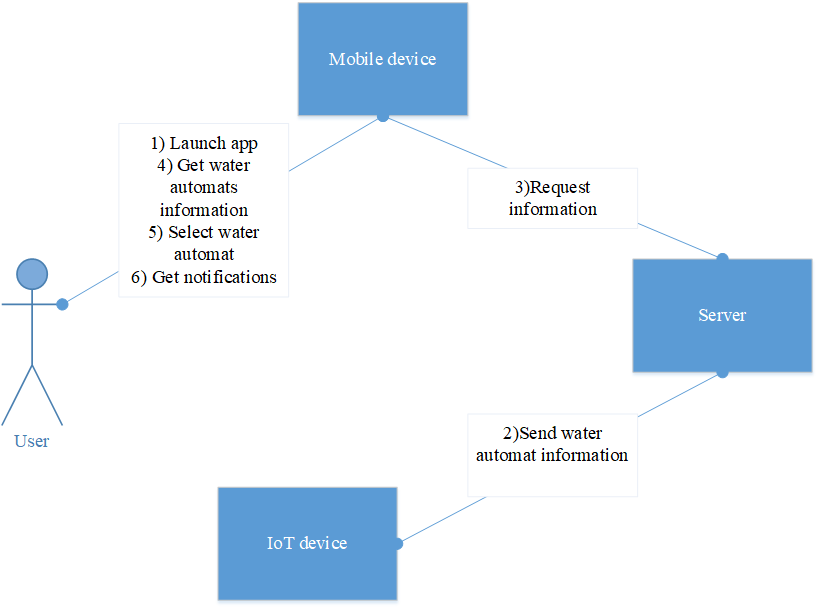


Рисунок 2.1 – Діаграма комунікацій

Діаграми використання допомагають описати варіанти використання системи. На рисунку 2.2 зображена діаграма використання з можливими діями з системою для користувача, який може отримувати інформацію про автомати з водою та їх отримувати повідомлення.

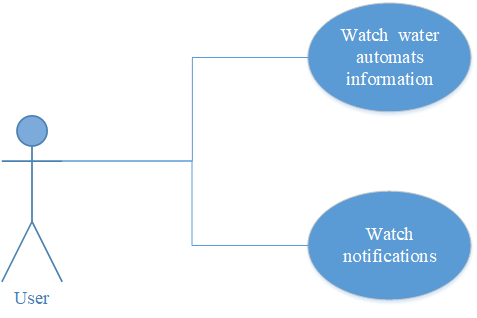


Рисунок 2.2 – Діаграма використання

На рисунку 2.3 зображена діаграма компонентів. Діаграма компонентів відображає залежності між компонентами програмного забезпечення, включаючи компоненти вихідних кодів, бінарні компоненти, та компоненти, що можуть виконуватись. Компоненти об'єднуються, разом використовуючи структурні зв'язки, щоб об'єднати інтерфейси трьох компонентів. Це ілюструє зв'язок типу «клієнт-iot-сервер» .

З неї видно, що в системі є компонент бази даних, серверна частина додатку, API, клієнтська частина додатку, IOT та мобільний клієнт.

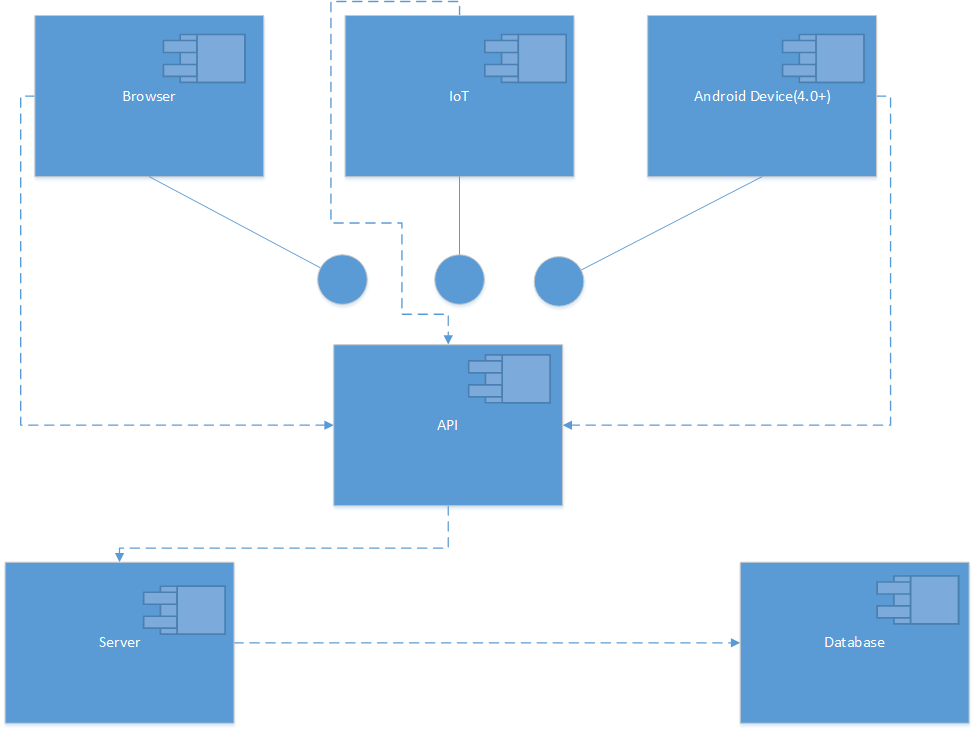


Рисунок 2.3 – Діаграма компонентів

Треба відзначити, що на діаграмі показані усі компоненти системі та зв’язки між ними. Клієнтська частина додатку, мобільний клієнт та IOT взаємодіють з API, які обробляють дані та відправляють дані до сервера, а він в свою чергу взаємодіє з базою даних.

Відповідно до діаграми компонентів побудована діаграма розгортання (див. рис. 2.4). На діаграмі розгортання відображаються обчислювальні вузли під час роботи програми, компоненти, та об'єкти, що виконуються на цих вузлах. Компоненти відповідають представленню робочих екземплярів одиниць коду. Компоненти, що не мають представлення під час роботи програми на таких діаграмах не відображаються. Діаграма розгортання відображає робочі екземпляри компонент.

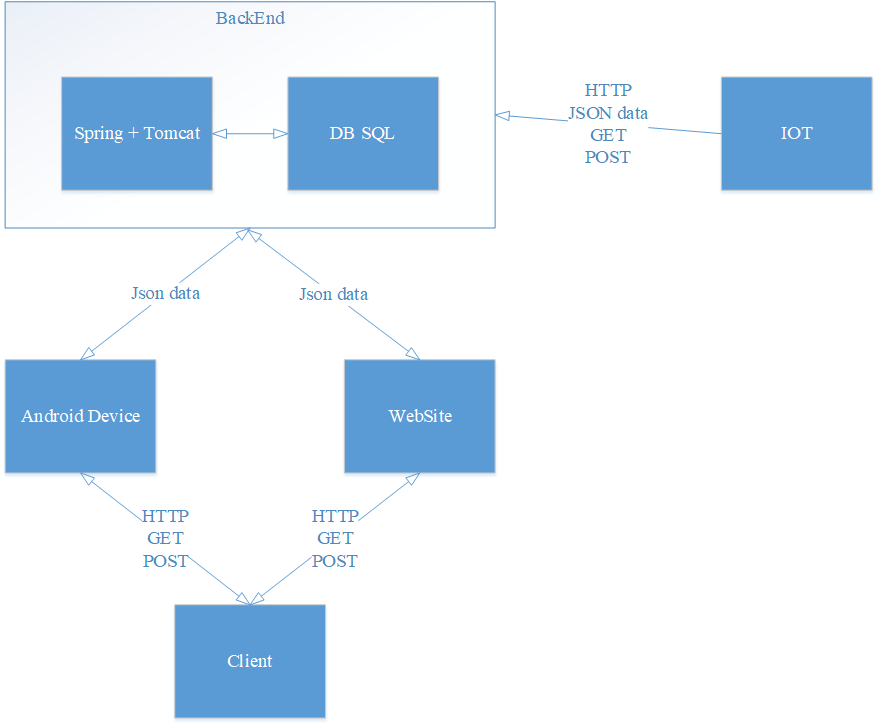


Рисунок 2.4 – Діаграма розгортання

На діаграмі розгортання зображено місцезнаходження окремих компонентів системи: на сервері додатку, сервері бази даних та на мобільному додатку.

Діаграма послідовностей представляє собою систему в цілому та те, як відбувається взаємодія між об’єктами на різних рівнях. Відображає взаємодії об'єктів впорядкованих за часом. Зокрема, такі діаграми відображають задіяні об'єкти та послідовність відправлених повідомлень На даній діаграмі послідовності (див. рис. 2.5) зображено послідовність дій при реєстрації нового користувача.

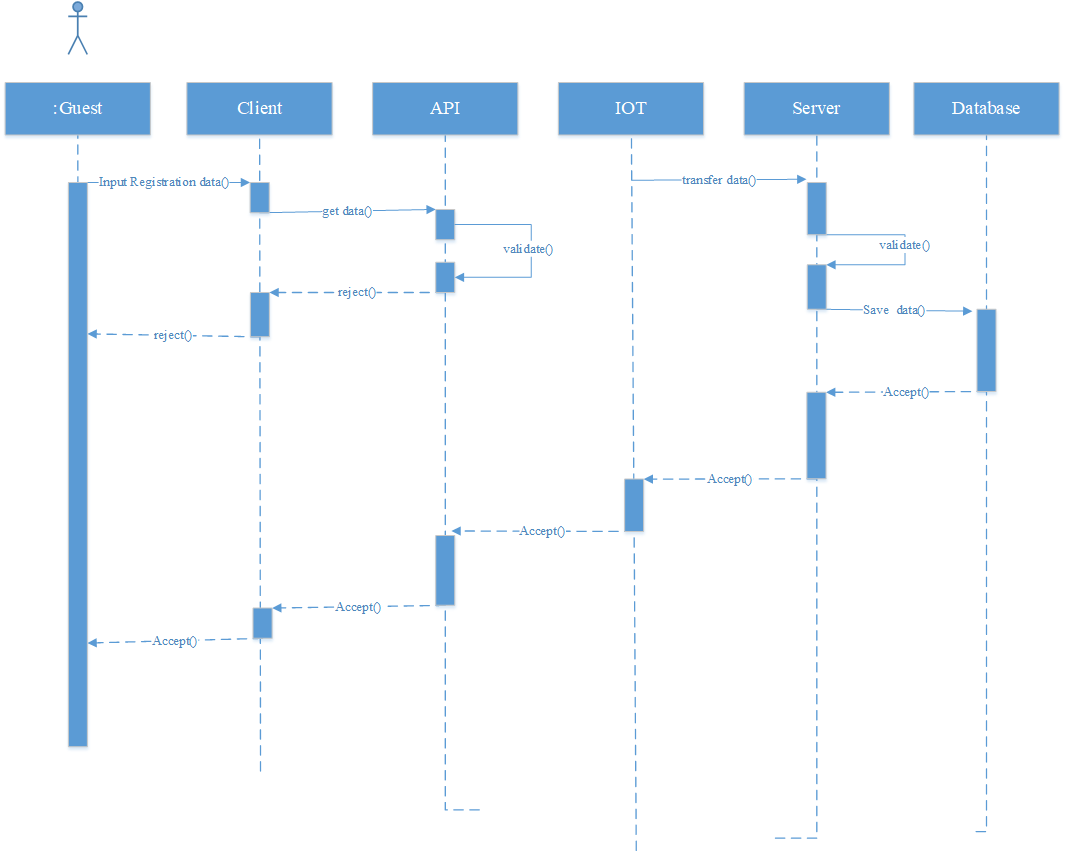


Рисунок 2.5 – Діаграма послідовностей

Користувач вводить дані для реєстрації чи входу, клієнт їх обробляє та передає на сервер, який у свою чергу після обробки передає дані до бази даних, яка їх зберігає, IOT передає отримані данні з автомату на сервер, який оброблює данні та передає їх до бази данних.

На рисунку 2.6 зображена діаграма кооперації, що відображає те, як користувач впливає на систему та взаємодію об’єктів. На даній діаграмі зображено тільки відносини між об'єктами, що грають певні ролі у взаємодії. Головна особливість діаграми кооперації полягає в можливості графічно представити не тільки послідовність взаємодії, але і всі структурні відносини між об'єктами, які беруть участь у цій взаємодії.

На рисунку 2.6 зображена діаграма активності системи. На даній діаграмі відображено процес логіну користувача до системи.

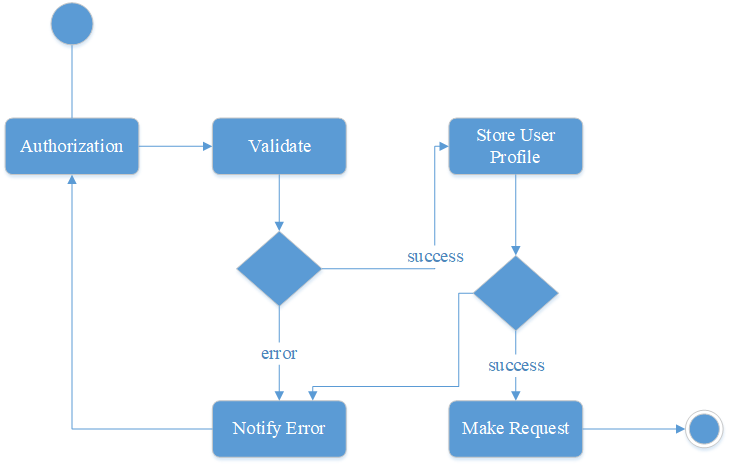


Рисунок 2.6 – Діаграма активності

В залежності від даних, які валідуються на сервері, клієнт залогіниться в систему, чи отримає помилку авторизації. Після авторизації клієнт може створити виклик інформації про автомати з водою. Якщо валідація даних пройде не успішно клієнт повернеться до кроку авторизації.

# 3 РЕАЛІЗАЦІЯ ПРОГРАМНОГО ПРОДУКТУ

Під час планування та моделювання програмного продукту було визначено, що проект має складатися з п’яти частин:

* веб-сайт для перегляду інформації про автомат з водою, після авторизації;
* IOT для відправлення інформації, отриманої з автоматів з водою;
* сервер з базою даних, на якому будуть оброблятися запити ззовні і зберігатися вся інформація про систему, її користувачів, та автоматів з водою;
* API-сервіси, що будуть звертатися до серверу. Клієнти відсилають запити до API через веб сайт;
* мобільний клієнт, розроблений під Android, для надання механізму управління системою. Cлід зауважити, що в наступних версіях також планується розробити мобільний додаток для інших популярних мобільних операційних систем, як, наприклад, iOS і Windows).

Інформація про кожну з них буде приведена далі в відповідному розділі.

## 3.1 Загальні відомості про систему

Під час моделювання і планування програмного продукту було встановлено, що проект повинен складатися з таких частин:

* веб сайт;
* IOT;
* сервер та база даних;
* API-сервіси;
* мобільний додаток.

Для розробки веб-сайту були використані кілька технологій, щоб компенсувати деякий недолік в функціональності сайту в той час, коли проект знаходиться на першій стадії розробки. Для розробки використовувався Angular, Node.js, HTML, CSS, JS, JQuery. Для взаємодії клієнта з сервером використовується HTTPS з’єднання. Для роботи з базою даних був використани MongoDBDriver. У відповідності з технологіями, використовується мови програмування Swift, JavaScript. Для розгортання серверу використовується Heroku.

Під час розробки клієнту для iOS використовується iOS SDK, та мова програмування Swift. Дані з мобільного додатку передаються на сервер та оновлюються у основній базі даних. Додаток має розповсюджуватися та інсталюватися на мобільних девайсах та планшетах локально за допомогою ipa-файлу.

Якщо описувати модель взаємодії усіх компонентів системи, то можна зрозуміти, що головною ланкою між усіма клієнтськими додатками та базою даних є сервер, API. Але за рахунок того, що всі головні елементи системи розділені, то це дозволяє гарно масштабувати систему. Ця архітектура добре підходить для хмарних обчислень (Microsoft Azure).

## 3.2 Серверна частина

Серверна частина - головний компонент системи GoMoon. Система не буде повністю працездатною, якщо в ній буде відсутній хоча б один з перерахованих вище основних компонентів, однак без серверної частини система перестане бути працездатною зовсім. Як тільки буде виключений з системи сервер, сама система перестане існувати: замість неї буде кілька окремо взятих компонентів, ніяк один з одним не взаємодіючі додаток для управління, веб-сайт.

Сервер написаний на мові Swift, використовує Vapor framework. Для розробки серверу використовується технологія створення Web-застосувань Vapor. Всі дані зберігаються з використанням СУБД MongoDB. Для роботи з базою даних використовується MongoDBDriver, а сама БД створена за допомогою підходу Сode First.

Сервер реалізує всю бізнес логіку та зберігає дані. Він написаний згідно шаблону MVC.

На рівні моделі знаходяться усі класи, що описують бізнес модель, та сервісний клас, у якому представлена уся бізнес-логіка. Зазначимо, що в усіх чотирьох компонентах системи присутні одні й ті ж класи-моделі для зручності передачі даних. На рис. 3.1 можна побачити моделі сервера.

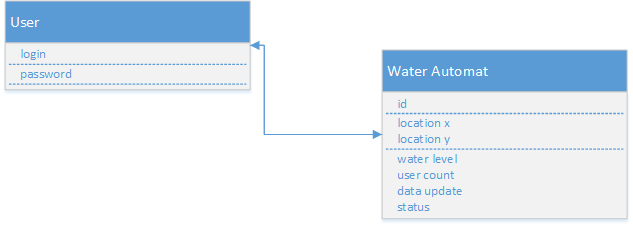


Рисунок 3.1 – Схема бази данних сервера

Для виклику методів на сервері використовується WebAPI, що дозволяє отримувати та відправляти різноформатні дані по протоколу https.

Сервер є багаторівневим, має кілька окремих рівнів, пов'язаних один з одним: рівень загальнодоступного API, бізнес-рівень, рівень доступу до бази даних.

У наступних версіях поліпшення на серверної частини будуть зводитися до поліпшення продуктивності і всебічної оптимізації.

Протягом роботи над сервером створена API документація:

* User:
  1. GET Api/users/id: - отримання інформації о користувачі по токену і унікальному ідентіфикатору.
  2. GET Api/users/payments/id: - отримання платежів користувача по токену і унікальному ідентифікатору.
  3. POST Api/users/ - регістрація користувача.
  4. POST Api/users/login – авторизація користувача.
* Payments:
  1. GET Api/payments/id: - отримання платежів по ідентифікатору користувача і токену.
  2. POST Api/payments/ - створення платежу.
  3. DELETE Api/payments/ - видалення платежу.
* Campaigns:
  1. GET Api/campaigns/ - отримання кампаній.
  2. GET Api/campaigns/payments/id: - отримання списку платежів для кампании по унґікальному идентифікатору.
  3. POST Api/campaigns/ - створення кампанії.
  4. DELETE Api/campaigns/ - видалення кампанії.

|  |  |
| --- | --- |
|  |  |

Усі данні в цьому API повертаються у спеціальному зручному форматі  JSON, які передаються між рішеннями системи.

## 3.3 Мобільний клієнт

Останній з перерахованих вище компонентів системи і один з найбільш важливих її складових – це мобільний клієнт.

Всі дані від користувача приходять на сервер саме від мобільного додатку, тому що саме з ним безпосередньо може взаємодіяти користувач. Це є основною причиною для того, щоб поставити юзабіліті мобільного клієнта на друге місце за важливістю критеріїв якості.

Мобільний додаток було реалізовано з використанням iOS SDK під ОС iOS. Він отримує дані про кампанії та платежі клієнта за допомогою використання https протоколу, та передає ці дані API-сервісу, який обробляє їх та перенапправлює на сервер.

У наступній версії будуть розроблений мобільний клієнти для користувачів Android.

## 3.4 Веб сайт

Для першої версії проекту цілком достатньо мати простий ознайомлювальний веб-сайт, який дозволить відвідувачам дізнатися більше. Сайт дає відвідувачам змогу ознайомитися з певною інформацією.

Для розробки веб-сайту були використані кілька технологій, щоб компенсувати деякий недолік в функціональності в той час, коли проект знаходиться на першій версії. Сам сайт буде написаний з використанням Angular, Node.js, HTML, CSS, JS і JQuery. Розгортатися сайт буде завдяки Heroku і доступ до нього буде відкритий для будь-якого користувача мережі Інтернет.

У наступній версії проекту, що розробляється веб-сайт буде доповнений новою функціональністю. Проста інформаційно-ознайомча сторінка доповниться новими функціями, наприклад, відображення статистики, як для конкретного користувача, так і загальної статистики – для всіх користувачів системи.

Кожен актор гість може зареєструватися в системі через веб-сайт, а не мобільний додаток, також гість матиме можливість залогінитися під уже існуючим ім'ям користувача і паролем, тоді його роль зміниться на активного користувача.

# 4 ІНТЕРФЕЙС І ФУНКЦІОНАЛ

Як вже неодноразово було вімічено, система складається з чотирьох компонентів: серверу, API, веб сайту та клієнту для Android. Сервер та API не мають власного інтерфейсу користувача адже вони не повинні напряму контактувати з користувачами.

## 4.1 Веб сайт

Для того, щоб розпочати роботу з системою, потрібно зайти на головну сторінку сайту. Для неавторизованих користувачів потрібно залогінитись (див. рис. 4.1).

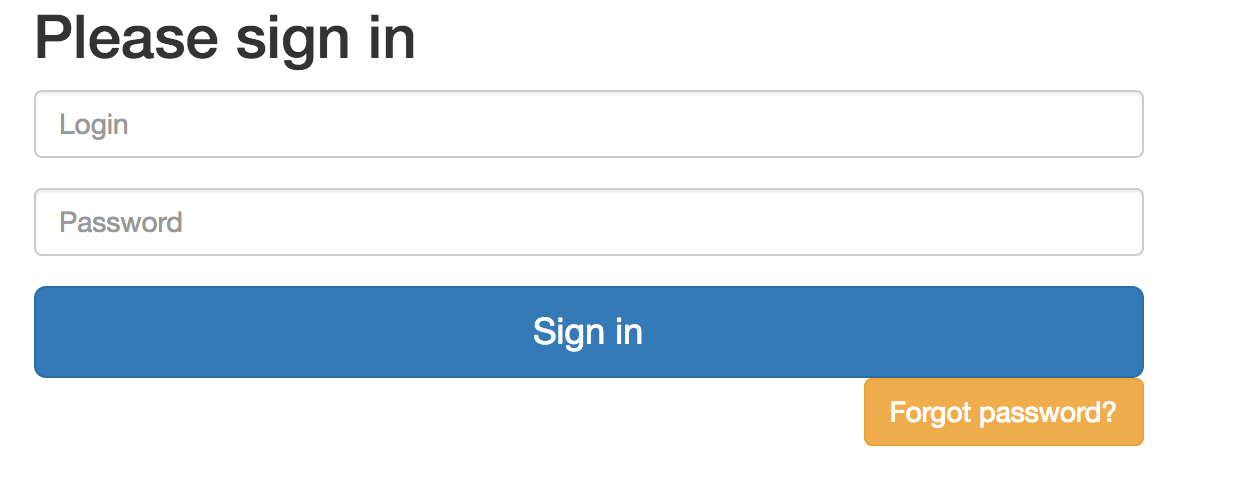


Рисунок 4.1 – Сторінка логіна

Неавторизований користувач може переглядати інформацію про додаток, ознайомитися з інструкцією. Для того щоб увійти у систему, користувач повинен ввести правильні логін та пароль, після чого його переадресує на головну сторінку, де він може переглянути список доступних кампаній (рис. 4.2).

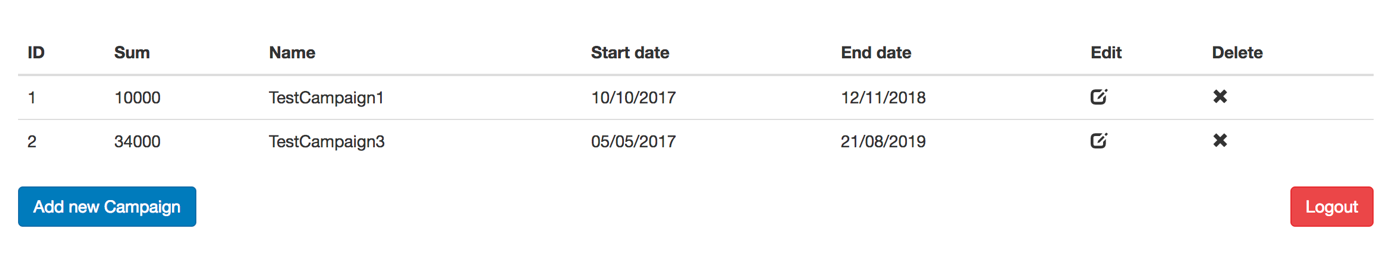


Рисунок 4.2 – Dashboard веб додатка

Користувач має можливість додати нову кампанію, нажавши кнопку «Add new Campaign» (рис. 4.3).

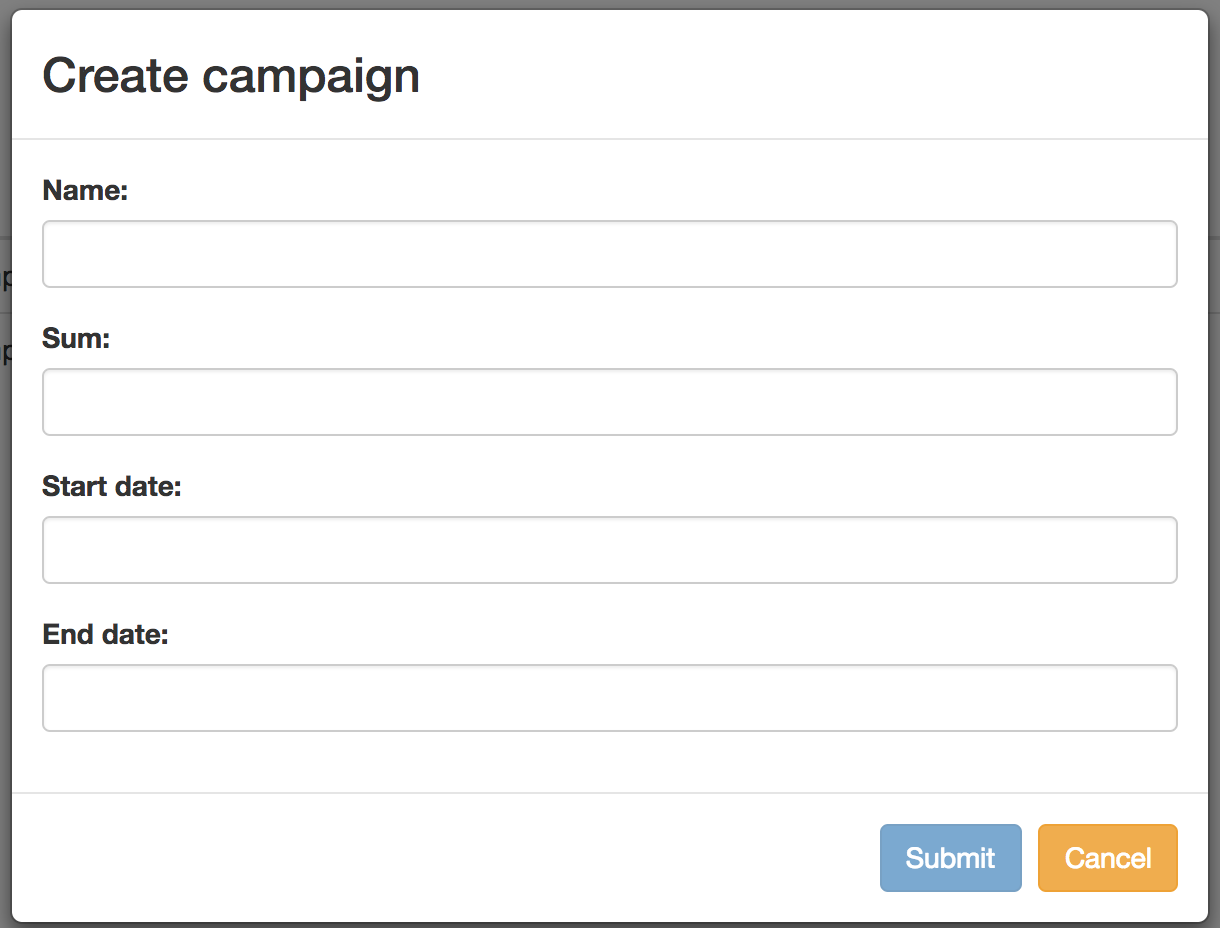


Рисунок 4.3 – Окно створення нової кампанії

Якщо виникає необхідність у зміні інформації, користувач може натиснути кнопку «Edit» та внести необхідні зміни (див. рис. 4.4). Перевіривши чи всі дані введені вірно можна натиснути кнопку «Submit» та зберегти змінену інформацію.

Також користувач має можливість видалити кампанію або переглянути платежі (у наступних версіях програми).

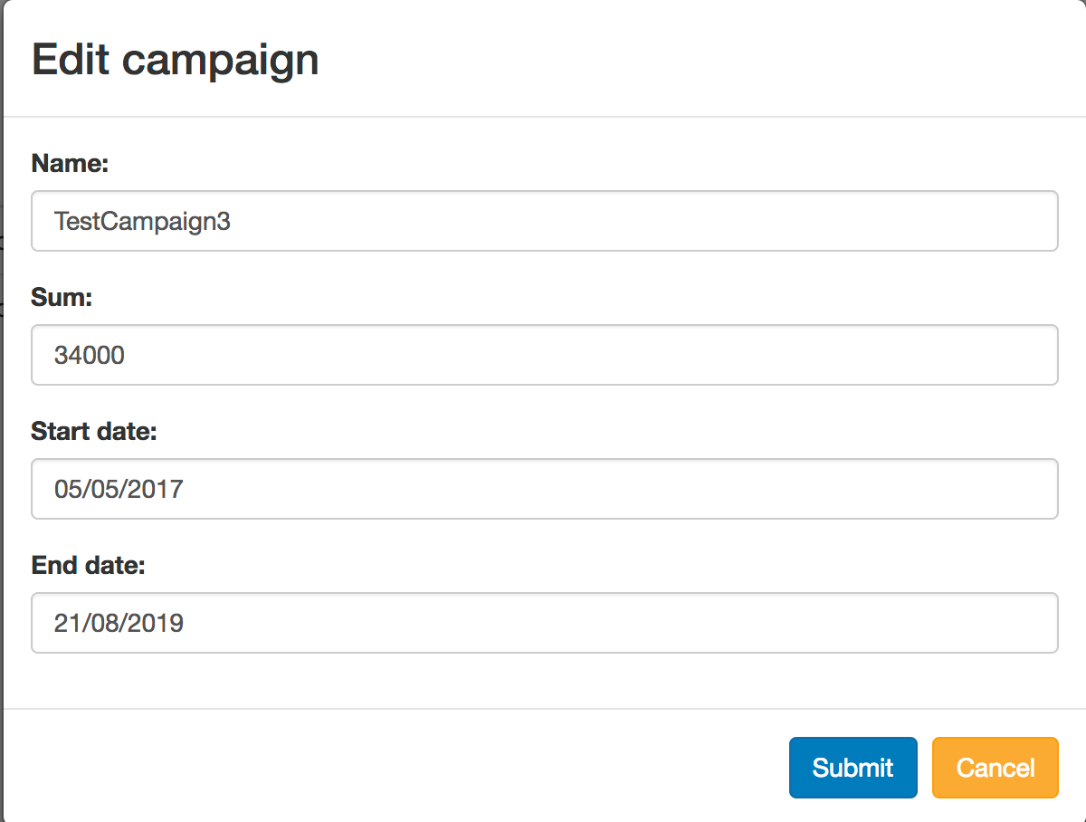


Рисунок 4.4 – Редагування кампанії користувачем

## 4.2 Мобільний клієнт

При відкритті iOS додатку користувачеві пропонується ввести його логін та пароль для того, аби мати змогу користуватися надалі мобільним клієнтом та його функціоналом (див. рис. 4.5).

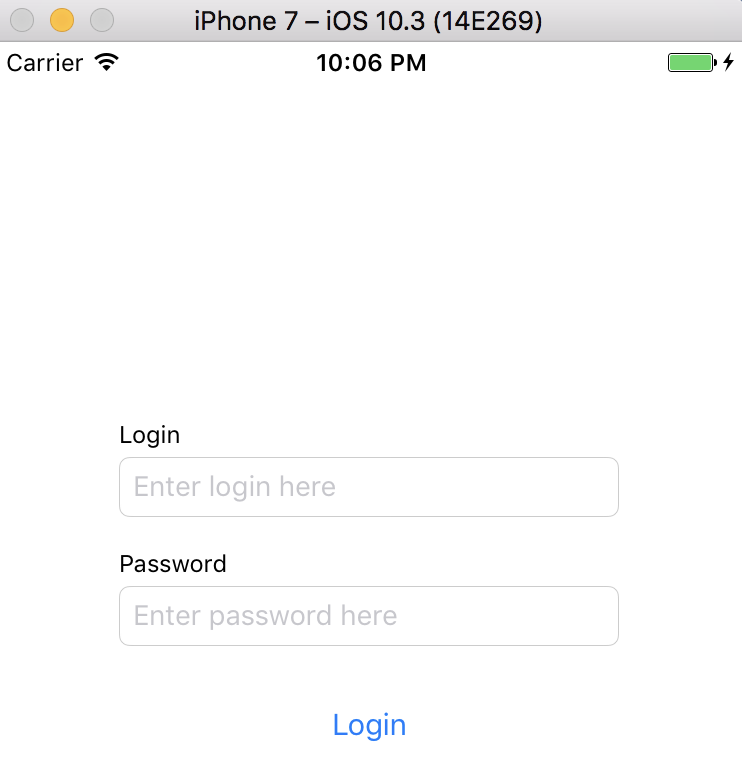


Рисунок 4.4 – Авторизація користувача через мобільних додаток

Після авторизації користувач має можливість переглянути список доступних кампаній у даний період часу (рис 4.5).

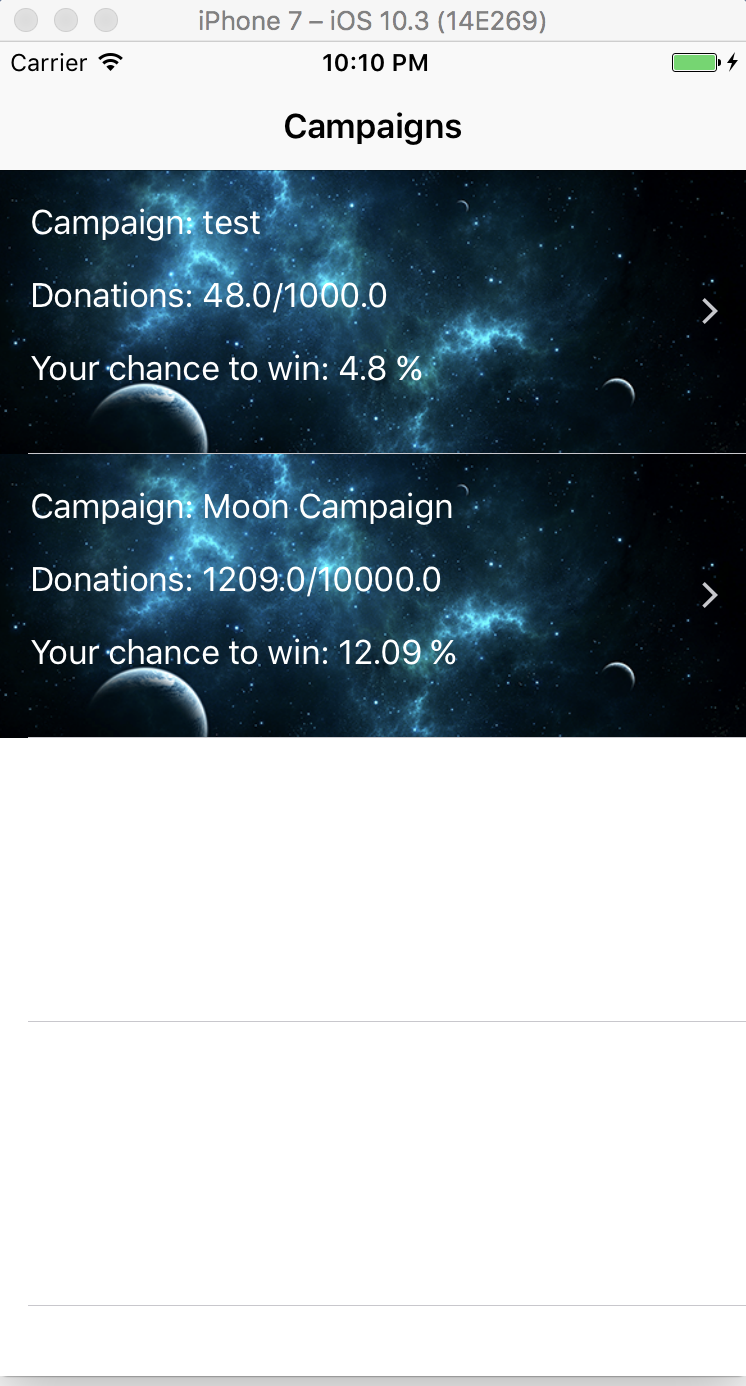


Рисунок 4.5 – Список доступних кампаній

Якщо користувач бажає прийняти участь у кампанії, він повинен перейти до конкретної кампанії та нажати кнопку «Make a donation» (рис. 4.6), після чого юзера буде переадресовано на сторінку платежу.

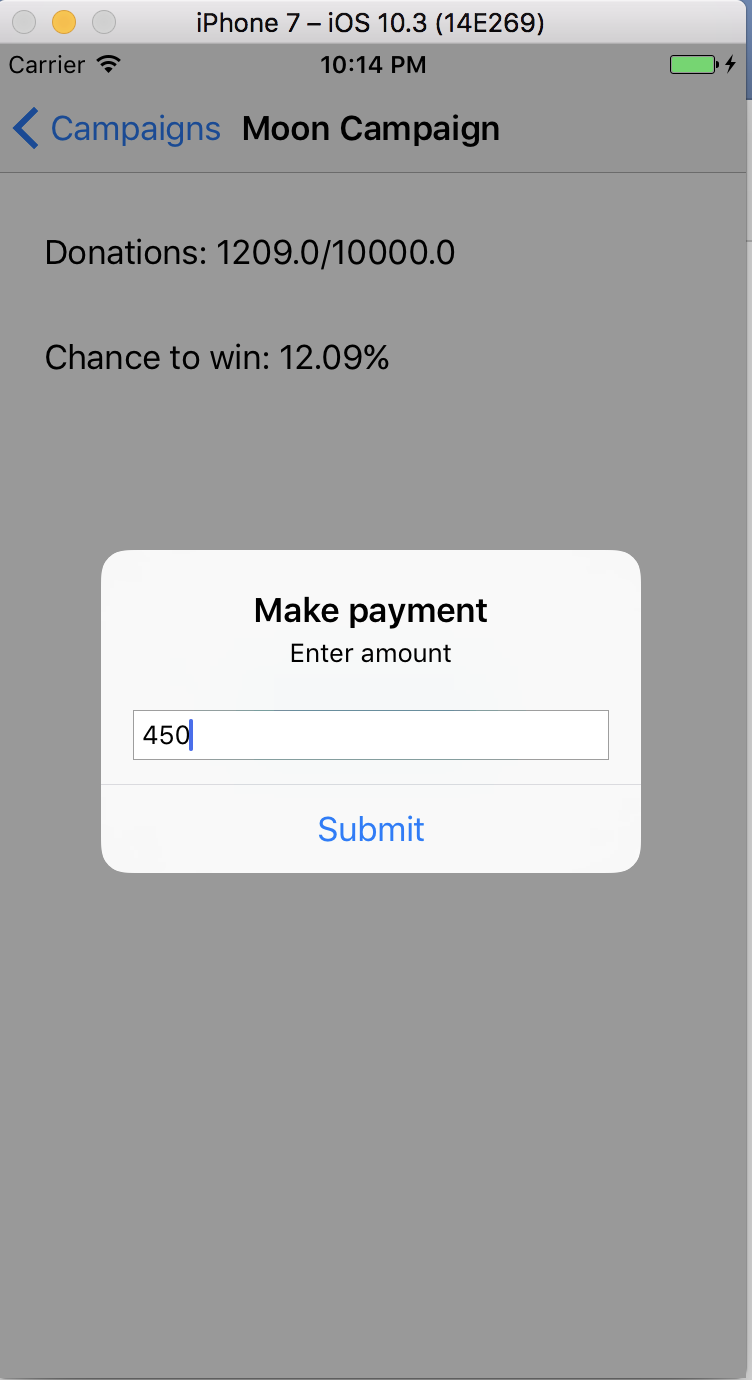


Рисунок 4.6 – Виконання платежу

# ВИСНОВКИ

В результаті виконання курсової роботи було створено систему «GoMoon» за допомогою технології Swift, розробка клієнтського додатка з елементами адміністрування за допомогою мови програмування Angular, Node.js.

Результатом курсового проекту є повноцінна система, що складається з клієнту та серверу, яка дозволить проводити та приймати участь у спонсоруванні компаній, які займаються космічним туризмом та дослідженням космосу, що значно прискорить розвиток космічних технологій.

Для реалізації всього необхідного функціоналу була проаналізована предметна область, виявлені взаємовідносини між основними об'єктами системи. Для зберігання інформації була спроектована і нормалізована база даних, яка відображає всі зв'язки між сутностями даної предметної області. Проаналізувавши створену систему на відповідність потребам, які необхідні для зберігання інформації про кампанії, прийшли до висновку, що система повністю задовольняє поставленим завданням.

Дана інформаційна система є багатофункціональною у використанні, має доступний інтерфейс, що дозволяє працювати з програмою користувачам з різним рівнем комп'ютерної грамотності, має середні вимоги до апаратного та програмного забезпечення, забезпечує досить високу швидкість роботи, безпечна у використанні, що є явними плюсами цього додатка.

Як уже згадувалося GoMoon дозволяє користувачам дати можливість прийняти участь у розвитку космічного туризму, та мати можливість отримати шанс на отримання путівки на космічну екскурсію. Зараз на ринку не має аналогів цього програмного продукту.

Цей продукт виділяється через свою унікальну пропозицію – подорож у космос, допомога у розвитку космічних науково – дослідницьких центрів.

Отримана система задовольняє заявленим до неї вимогам, а саме:

* Можливість збереження кампанії.
* Можливість збереження користувача та його платежів.
* Відображення кампаній та їх поточного статусу.
* Можливість створення платежу до конкретної кампанії.
* Підрахунок шансу на перемогу конкретного користувача.

# ПЕРЕЛІК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Андрєєв, В. В. Как организовать делопроизводство на предприятии [Текст] / В.В.Андрєєв. - М. : Наука, 2015. - 274 с.
2. Грабер М. SQL. [Текст] / М.Грабер.- 2-е изд. - К: Изд-во «ЛОРИ», 2003. –  644 с.
3. Дейт К.Дж. Введение в системы баз данных [Текст] / К.Дж. Дейт. - 8-е изд. – М.: Издательский дом «Вильямс», 2005.- 1328с.
4. Кренке Д. Теория и практика построения баз данных. [Текст] / Д.Кренке - 8-е изд. – СПб.: Питер, 2003. – 800 с.
5. Молинаро Э. SQL. Сборник рецептов. [Текст] / Молинаро Э. пер. с англ. – СПб.: Изд.: Символ-Плюс, 2009 г. - 672 с.

Додаток 1- вихідний код

* 1. Клас, який забезпечує асинхронну відправку запросів на сервер

class RequestManager {

static let shared = RequestManager()

private init() {}

func process(\_ request: RequestConvertible, completion: @escaping (\_ data: Data?, \_ error: Error?) -> Void) {

let task = URLSession.shared.dataTask(with: request.request()) { (data, response, error) in

debugPrint(response ?? "")

DispatchQueue.main.async {

completion(data, error)

}

}

DispatchQueue.global(qos: .userInitiated).async {

task.resume()

}

}

}

* 1. Метод для створення кампаній

func create(\_ req: Request) throws -> ResponseRepresentable {

guard let summ = req.data["summ"]?.double,

let name = req.data["name"]?.string,

let startDate = req.data["startDate"]?.string,

let endDate = req.data["endDate"]?.string else {

throw Abort.custom(status: .badRequest, message: "Wrong parameters")

}

do {

var newCampaign = Campaign(name: name,

summ: summ,

startDate: startDate,

endDate: endDate)

try newCampaign.save()

return try JSON(node: [["error": false,

"message": "Created"],

try newCampaign.makeNode()] )

} catch {

throw Abort.custom(status: .badRequest, message: error.localizedDescription)

}

}

* 1. Метод для отримання платежів вибраної кампанії

func payments(\_ req: Request) throws -> ResponseRepresentable {

let campaign: Campaign = try CampaignsController.model(with: req, for: "id")

let paymentNodes = try campaign.payments().all().map { try $0.makeNode() }

return try JSON(node: paymentNodes)

}

* 1. Метод авторизації користувача

function logInUser(req, res, user) {

req.logIn(user, (internalError) => {

if (internalError) {

res.send({ success: false, error: internalError });

}

const userForToken = user;

userForToken.accessToken = jwt.sign({ userId: userForToken.\_id }, config.secret, {

expiresIn: 86400, // expires in 24 hours

});

userForToken.save((queryError, userWithToken) => {

if (queryError) {

res.send({ success: false, error: queryError });

}

return res.json({

success: true,

message: 'Enjoy!',

token: userWithToken.accessToken,

user: userWithToken,

});

});

});

}

* 1. Метод відкриття модальних вікон у веб клієнті

angular.module('GoMoon').service('Modal', function ($uibModal, Models) {

const self = this;

self.openEditCampaignModal = function (campaign) {

return $uibModal.open({

templateUrl: './views/editCampaignModal.html',

controller: ($scope, $uibModalInstance, campaign) => {

const scope = $scope;

scope.campaign = campaign;

scope.ok = () => {

$uibModalInstance.close($scope.campaign);

};

scope.cancel = () => {

$uibModalInstance.dismiss();

};

},

resolve: {

campaign,

},

});

};

self.openAddCampaignModal = function () {

return $uibModal.open({

templateUrl: './views/addCampaignModal.html',

controller: ($scope, $uibModalInstance) => {

const scope = $scope;

scope.campaign = {};

scope.ok = () => {

$uibModalInstance.close($scope.campaign);

};

scope.cancel = () => {

$uibModalInstance.dismiss();

};

},

});

};

});