Міністерство освіти і науки України

Харківський національний університет радіоелектроніки

Факультет комп’ютерних наук

Кафедра програмної інженерії

КУРСОВА РОБОТА  
пояснювальна записка

«Програмна система для догляду за кiмнатними рослинами»

Студент гр. ПІ-13-3 Гаврилов В.С.

Керівник роботи: доц. каф. ПІ Лещинський В.О.

Роботу захищено «\_\_»\_\_\_\_\_\_2016 р.

з оцінкою \_\_\_\_\_\_\_\_\_\_\_\_\_

Комісія:

проф. каф. ПІ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_Дудар З.В.

доц. каф. ПІ \_\_\_\_\_\_\_\_\_\_\_\_\_\_Лещинський В.О.

ст. викл. каф. ПІ \_ \_ \_\_\_\_\_\_\_\_\_\_\_\_ Лещинська І.О.

Харків

2016 р.

Харківський національний університет радіоелектроніки

Факультет комп’ютерних наук

Кафедра програмної інженерії

Напрям підготовки Програмна інженерія

Курс 3 . Семестр 6 .

Навчальна дисципліна Аналіз та рефакторинг коду програмного забезпечення .

ЗАВДАННЯ  
НА КУРСОВУ РОБОТУ СТУДЕНТОВІ

*Гаврилову Володимиру Сергiйовичу .*

1. Тема роботи: *«Сервіс для догляду за кiмнатними рослинами»*

*.*

2. Термін узгодження завдання курсової роботи «*04*»\_\_*лютого*\_\_\_\_\_\_\_\_\_ 2016 р.

3. Термін здачі студентом закінченої роботи «*13*»\_\_\_*червня\_\_\_\_\_\_ \_\_* 2016р.

4. Вихідні дані до проекту (роботи): *Використовувати ОС Windows, середовище об'єктно-орієнтованого проектування Microsoft Visual Studio 2015 Community*

*.*

*.*

*.*

*.*

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

*.*

*.*

6. Перелік графічного матеріалу (з точним зазначенням обов’язкових креслень)   
*діаграма класів, діаграма прецедентів, діаграма послідовностей, діаграма станів, діаграма компонентів, діаграма кооперації.*

*.*

*.*

*.*

*.*

КАЛЕНДАРНИЙ ПЛАН

|  |  |  |  |
| --- | --- | --- | --- |
| № | Назва етапів курсової роботи | Термін виконання етапів проекту (роботи) | Примітка |
| 1 | Функціональна специфікація програми | 05-02-2013 | виконано |
| 2 | Проектування програми | 20-02-2015 | виконано |
| 3 | Кодування програми | 20-03-2015 | виконано |
| 4 | Оформлення пояснювальної записки | 09-05-2015 | виконано |
| 5 | Захист курсової роботи | 13-06-2015 | виконано |

Дата видачі завдання «*04*»\_\_*лютого*\_\_\_\_\_\_\_\_2015 р.

Керівник роботи \_\_\_\_\_\_\_\_\_\_\_ доц. каф. ПІ Лещинський В.О.

Завдання прийняла до виконання

студент гр. ПІ-13-3 \_\_\_\_\_\_\_\_\_\_\_ Петренко Д.О.

РЕФЕРАТ / ABSTRACT

Пояснювальна записка до курсової роботи: 25 с., 14 рис.,   
6 джер.

Метою роботи є проектування та розробка програмної системи для відстеження змін стану автомобілю.

Методи розробки базуються на технології Java, PostgreSQL базі даних, середовищі розробки Spring Tool Suite та Intellij IDEA Community Edition.

Результатом роботи є програмна система, що складається з серверу, веб-клієнта, мобільного додатку та емулятору пристрою, що відстежує зміну стану автомобіля, яка дозволить визначеній групі людей отримувати сповіщення про зміну стану автомобіля, наприклад аварію чи злом.

JAVA, АВТОМОБІЛЬ, СПОВІЩЕННЯ, МОБІЛЬНИЙ ДОДАТОК, ANDROID, SPRINGMVC, ПРОГРАМНА СИСТЕМА.

The aim is to design and develop a software system for tracking vehicle status changing.

The methods of development technologies are based on Java, PostgreSQL database, development environment Spring Tool Suite and Intellij IDEA Community Edition.

The result of work is a software system that consisting server, web-client, mobile application and device emulator that tracks status changing of vehicle, which will allow to get notification to some people group abou status changing of vehicle, for example crash or hacking.

JAVA, VEHICLE, NOTIFICATION, MOBILE APPLICATION, ANDROID, SPRINGMVC, SOFTWARE SYSTEM.

ЗМІСТ

Вступ….………………………………………………………….……….….….……......6

1 Аналіз предметної області…..……………………….…………….……...…….….…8

2 Постановка задачі….………………………………………………..….……….……13

3 Моделювання програмного продукту……..…..…………………………...….…….14

4 Реалізація програмного продукту…………….…...………………..……..…….…...18

5 Інтерфейс і функціонал……………………..………………………....….……….…20

5.1 Сервер…………………..………………….………………………..……….20

5.2 Веб-клієнт…………………………………….……...……....….…………..21

5.3 Додаток для Windows Phone……...………………………………………..23

Висновки……….……………………………...…………………………………...…....24

Перелік посилань………...…………………….………………..……...……….……...25

ВСТУП

Автомобіль невід’ємна частина життя багатьох людей в світі. У розвинутих країнах рівень автомобілізації вагається від 600 до 900 автомобілів на 1000 людей. Автомобілі використовуються як для повсякденних цілей, так і для перевезення вантажу або пасажирів. Для багатьох людей автомобіль це не просто засіб пересування, а місце роботи.

Але так само з автомобілями пов’язано дуже багато труднощів, цей вид транспорту важко назвати надійним, тому що багато залежить не від водія а від самого автомобіля, а також від інших автомобілів та їх господарів. Близько 5 мільйонів ДТП трапляється у світі за рік. А скільки інших пригод трапляється з автомобілями? Зломи, серйозні технічні несправності та інше. Бувають ситуації, коли про змін стану автомобіля треба знати не тільки водію, але і його родині, або компанії, що володіє цим авто.

В основі проекту лежить «розумний» пристрій, що дозволить відстежувати стан автомобілю. Він буде пов’язаний з сервером та відправлятиме повідомлення про зміни. Цей пристрій працює за концепцією Internet of Things.

Інтернет речей (Internet of Things) − єдина мережа, що об’єднує техніку, якою ми користуємося щодня, та віртуальний світ. Технологія не лише дозволяє віддалено керувати різними приладами, а й пов’язує їх між собою. Обмінюючись даними, речі починають «спілкуватися» один з одним.

Ця ситуація може бути втілена в життя в тому випадку, якщо побутові прилади та інші електронні пристрої будуть підключені до Інтернету і навчаться обмінюватися між собою інформацією, виконувати команди, відправлені з інших пристроїв – компонентів цієї глобальної мережі. Реалізація цієї концепції отримала назву Інтернету речей – «Internet of Things». Вперше термін був запропонований в 1999-му році. А кількісний перехід від «Інтернету людей» до «Інтернету речей» відбувся в 2008-2009 році: саме в той період кількість пристроїв, підключених до Інтернету, перевищила кількість інтернет-користувачів.

В наш час ця технологія стає дедалі популярнішою, наприклад такі гіганти, як Apple чи Google виробляють системи для керування «розумним будинком».

Підключити інтернет зараз можна не тільки для дому. Вже зараз до мережі підключають автомобілі, які можуть у режимі онлайн отримувати мультимедійний контент, наприклад, відтворювати музику з iTunes, а також оновлювати інформацію про дорожню ситуацію і погодних умовах.

Футуристи і фахівці в технологіях називають Інтернет речей проміжним етапом на шляху до нового витка розвитку технологій – до мережі чергового покоління, названого «Інтернетом всього». У цій мережі просто не буде існувати пристроїв, що працюють в режимі «офлайн», а Інтернет буде те саме що електрика – без нього просто не зможе функціонувати жодний електронний пристрій. Сьогодні складно собі уявити, яким буде наше життя в «Інтернеті всього», проте вже зараз можна припустити, що сховатися від всевидющого ока такої воістину глобальної мережі буде практично неможливо.

Отже тема та засоби для виконання курсової роботи є актуальними. Метою курсової роботи є проектування та створення програмної системи, яка дозволить відстежувати зміну стану автомобіля та відправляти повідомлення про зміни особам, які закріплені за даним автомобілем.

2 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ

Як було зазначено вище обрана тема є актуальною на сьогодні і на близьке майбутнє. Щоб проаналізувати дану предметну область треба сказати про засоби, що все використовуються, їх плюси та мінуси.

Якщо казати про конкуренцію на ринку в даній сфері, то можна сміливо заявити, що наразі немає жодного аналогу, який міг би робити щось подібне. Єдиний засіб, що надає функціонал, щодо зміну стану автомобілю є сигналізація. Це також можна назвати системою, що сповіщає про зміну стану та повідомляє користувачу.

Якщо розглядати плюси та мінуси «конкурента» треба сказати, що єдиним плюсом є відсутність прив'язки до Інтернету. Так, це може зіграти свою роль при спробі викрадення. Але це єдиний плюс при порівнянні з даним програмним продуктом. Якщо розглядувати мінуси, то по-перше це реакція тільки на спроби викрадення або зовнішні пошкодження. Також дуже важливим є те, що користувач сповіщається лише за допомогою звукової сирени, яку він не почує на великій відстані.

Отже, ця програмна система має функціонал, якому немає аналогів. Якщо казати про те, кому саме може знадобитись додаток, то можна впевнено сказати – будь-якому автомобілісту.

Розглянемо звичайного автомобіліста, який використовує автомобіль у повсякденних цілях, з ним може трапитись що завгодно у дорозі. Ситуації трапляються різні, і в деяких з них потрібно оперативно зреагувати. Наприклад, коли трапляється ДТП буває, що потрібно швидко викликати допомогу і від цього може залежати життя людини, це може зробити людина якій прийде повідомлення. Також у випадку якихось несправностей в майбутньому програма зможе визначати свої координати, відсилати її людям, за якими закріплений автомобіль та пропонувати найближчі авто-сервіси та СТО.

Можна розглядувати й іншу сторону користувачів. Програмний продукт може бути корисним великим компаніям, пов’язаним з перевезенням, наприклад сервіси таксі, або які займаються перевезенням вантажів.

Насправді можна говорити навіть про те, що цей продукт може використовуватися державними службами, наприклад поліцією, що в деяких ситуаціях допоможе скоріше зрозуміти, що трапилось аніж чекати на відповідь.

Також існує проблема з підтримкою автомобілів в багатьох країнах світу. Деякі фірми не надають можливість обслуговування своїх автомобілів в усіх містах. Тоді у разі несправності автомобіліст не знає, що йому робити, чи відвозити автомобіль в інше місто, чи шукати представників ближче, які можуть займатися цим. В цій ситуації можна представити, що у представників компаній є доступ до інформації про стан усіх автомобілів цієї компанії. І у випадку несправності представник може сам надіслати інформацію про себе, а не чекати коли його знайде власник автомобіля, для нього е буде як реклама, але при цьому це буде дуже зручно і для власника авто.

Отже, в цьому розділі було розглянуто предметну область. Можна підсумувати, що була обрана проблемна предметна область, яка потребує саме цього продукту, адже наразі немає жодного аналогу. Розглянуті приклади підтвердили, що ринок потребує цей продукт і він буде корисним дуже багатьом людям і навіть може використовуватися у великих масштабах. Враховуючи, що продукт буде розширюваний, його можна змінювати у залежності від потреб ринку та галузі використання.

2 ПОСТАНОВКА ЗАДАЧІ

Метою поданої курсової роботи є створення сервісу для керування веломаршрутами та поїздками за ними. З поставленої мети виходять мінімальні вимоги до користувачів.

По-перше для функціонування мобільного додатку необхідна наявність у користувача мобільного пристрою з операційної системою Android 4.0.x Ice Cream Sandwich або вищою, та наявність на мобільному пристрої GPS-датчика.

Дані користувачів передаються на сервер, який розміщено у хмарному середовищі, на серверах якого повинен бути встановлений та налаштований Tomcat Server. Потужності хмарного середовища повинні витримувати одночасну обробку кількох тисяч запитів. На сервері баз даних зберігаються дані користувачів у базі даних MongoDB [2], що також розміщується у хмарному середовищі. Дані, оброблені на серверах, передаються власне клієнтам. Мобільний додаток реалізований для подальшого розміщення у Play Market.

Веб частина повинна мати наступний функціонал: будь-який користувач може зайти в сервіс після авторизації. Якщо він ще не був зареєстрований, йому буде запропонована форма для реєстрації через соціальну мережу. Авторизований користувач має змогу переглядати свої маршрути то публічні маршрути інших користувачів. Також він може створювати та видаляти свої маршрути. Користувач може зробити свій маршрут публічним.

Мобільна частина повинна мати наступний функціонал: авторизований користувач може переглядати свої маршрути на карті та здійснювати поїздки за обраним з них. Якщо gps ввімкнено, то під час поїздки на карті повинно відображатись поточне положення.

Серверна частина повинна зберігати усіх користувачів системи та їхні маршрути. Вона має надавати REST сервіси для використання веб і мобільною частинами для реєстрації, логіну, отримання маршрутів з бази.

У подальшому планується розширити соціальну та функціональну складову розробленої системи шляхом додавання наступних можливостей:

а) у мобільному додатку під час поїздки додати можливість зробити фотографію та додати її на карту з коментарем або без нього;

б) розширити статистику по проведеним тренуванням, а саме додати можливість отримати максимальну, мінімальну та середню швидкість по маршруту;

в) додати досягнення, новинами про які користувач може поділитися з друзями;

г) підраховувати калорії згідно з інформацією про швидкість та пройдену відстань.

Можлива також розробка мобільного додатку для платформ iOS та Windows Phone.

3 МОДЕЛЮВАННЯ ПРОГРАМНОГО ПРОДУКТУ

Для проектування програмної системи було використано засоби проектування Microsoft Visio 2013 та створено наступні діаграми UML:

а) Діаграма прецедентів (див. рис. 3.1) має одного актора – Користувача. Користу-вач – людина, яка увійшла у сервіс під своїм Facebook-акаунтом, або встановила на пристрій додаток з Google Play Market. Після авторизації він має можливість створити маршрут, редагувати та видалити його. Він може переглядати свої маршрути та публічні маршрути інших користувачів. Також є можливість об’єднуватися у групи для спільних поїздок. Користувач може відстежувати своє положення, якщо ввімкнений GPS-датчик;

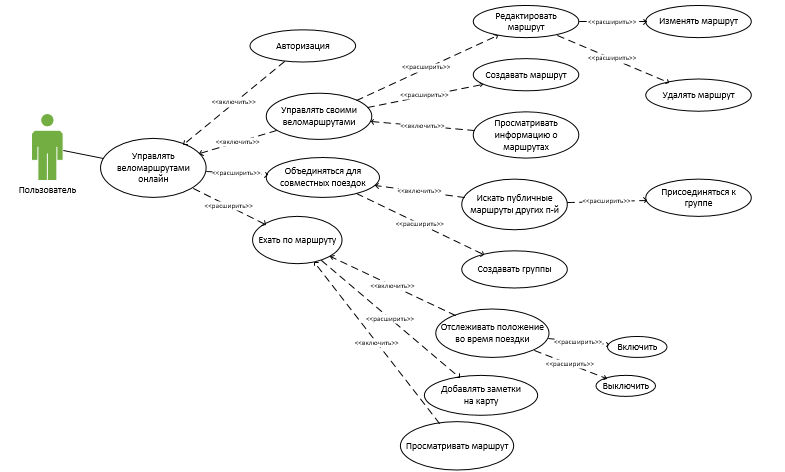


Рисунок 3.1 – Діаграма прецедентів

б) Діаграма класів (див. рис. 3.2) включає в себе серверні класи Point, як модель географічної точки; Route, як модель маршруту, до якого входить список точок, загальна дистанція та номер користувача, який створив цей маршрут; User, як модель користувача, до якого входить номер, ім’я та фамілія користувача; DBManager – інтерфейс, що визначає всі операції, що повинні бути реалізовані для роботи з базою даних,за допомогою цього інтерфейсу забезпечується розширяємість системи, тому що є можливість використовувати іншу базу даних; MongoDBManager, який є реалізацією інтерфейсу DBManager для MongoDB СУБД;

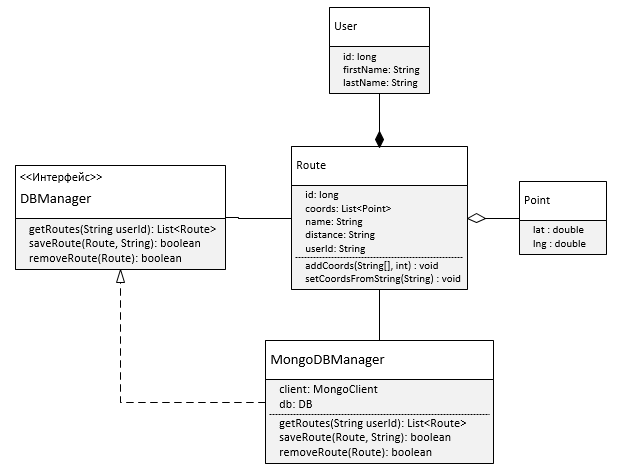


Рисунок 3.2 – Діаграма класів

в) На діаграмі послідовностей (див. рис. 3.3) зображена система в цілому на вищому рівні абстракції та взаємодія основних компонентів – клієнту, сервісу та менеджера бази даних;

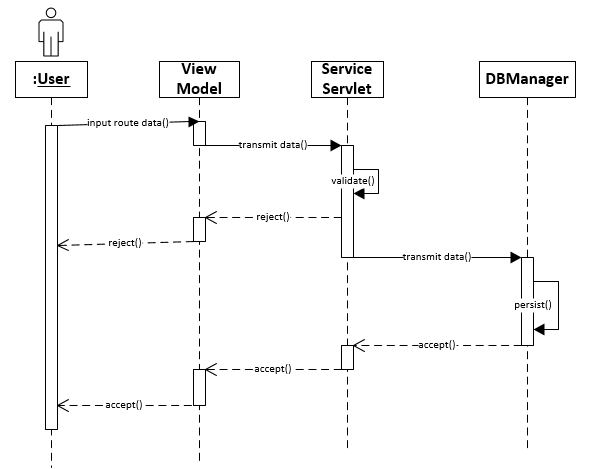


Рисунок 3.3 – Діаграма послідовностей

г) На діаграмі станів (див. рис. 3.4) зображений процес створення, редагування та просмотру маршруту користувачем. Для того, щоб користувачу було доступне створення маршруту, він повинен авторизуватися або зареєструватися у системі;

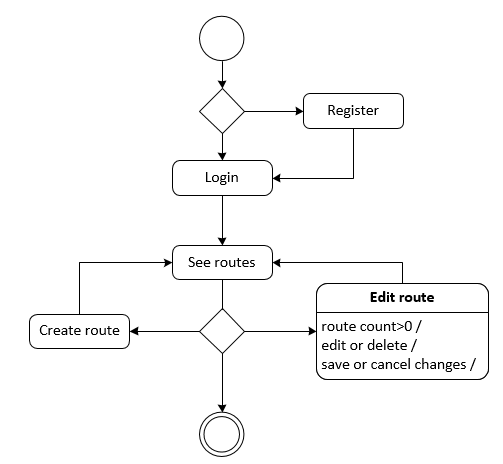


Рисунок 3.4 – Діаграма станів

д) На діаграмі компонентів (див. рис. 3.5) зображені основні частини системи веб-клієнт, мобільний клієнт, основний сервер системи, на якому розмішений сервіс та окремо класи даних, та сервер баз даних;

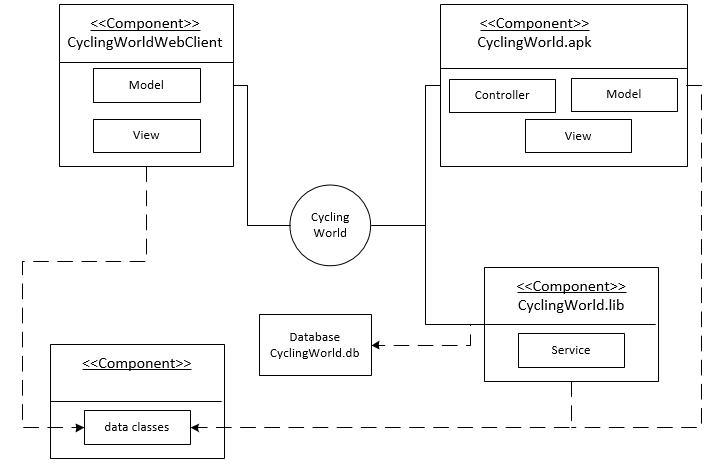


Рисунок 3.5 – Діаграма компонентів

е) На діаграмі кооперації (див. рис. 3.6) зображені стосунки між користувачем, що створив маршрут, та користувачем, що шукає цей створений маршрут.

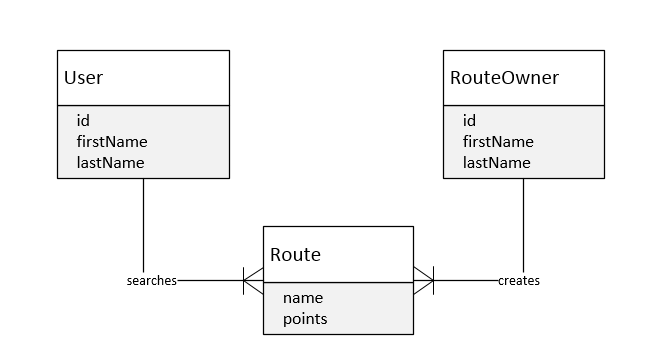


Рисунок 3.6 – Діаграма кооперації

4 РЕАЛІЗАЦІЯ ПРОГРАМНОГО ПРОДУКТУ

Під час планування та моделювання програмного продукту було визначено, що проект має складатися з трьох частин [4]: сервер, на якому будуть зберігатися дані, до серверу підключаються клієнти за допомогою REST API. На першому етапі реалізується веб-клієнт та мобільний додаток на платформі Android.

Для розробки серверу використовується технологія створення Web-застосувань мовою Java [5] з використанням SpringMVC фреймворку. об'єктно-орієнтована мова програмування, випущена компанією Sun Microsystems у 1995 році як основний компонент платформи Java. Java є платформо-незалежною мовою, тому вона є досить поширеною та популярною мовою програмування.

Для розробки використовується середовище Spring Tool Suite та платформа Java. Для виклику методів сервера використовуються REST[6] контролери з фреймворку Spring, що дозволяє отримувати та відправляти різноформатні дані по протоколу http.

На сьогоднішній день прийнято використовувати REST - (скор. від англ. Representational State Transfer - «передача репрезентативного стану») - метод взаємодії компонентів розподіленого додатка в мережі Інтернет, при якому виклик віддаленої процедури являє собою звичайний HTTP-запит, а необхідні дані передаються як параметри запиту.

У свою чергу HTTP — протокол передачі даних, що використовується в комп'ютерних мережах. Назва скорочена від Hyper Text Transfer Protocol, протокол передачі гіпер-текстових документів.

HTTP — протокол прикладного рівня, схожими на нього є FTP і SMTP. Обмін повідомленнями йде за звичайною схемою «запит-відповідь». Для ідентифікації ресурсів HTTP використовує глобальні URI. На відміну від багатьох інших протоколів, HTTP не зберігає свого стану. Це означає відсутність збереження проміжного стану між парами «запит-відповідь». Компоненти, що використовують HTTP, можуть самостійно здійснювати збереження інформації про стан, пов'язаний з останніми запитами та відповідями. Браузер, котрий посилає запити, може відстежувати затримки відповідей. Сервер може зберігати IP-адреси та заголовки запитів останніх клієнтів. Проте, згідно з протоколом, клієнт та сервер не мають бути обізнаними з попередніми запитами та відповідями, у протоколі не передбачена внутрішня підтримка стану й він не ставить таких вимог до клієнта та сервера

Дані зберігаються у базі даних MongoDB, доступ до якої здійснюється за допомогою JDBC (від англ. Java DataBase Connectivity – з’єднання з базою даних) -драйвера для Java. MongoDB — документо-орієнтована система керування базами даних (СКБД) з відкритим сирцевим кодом, яка не потребує опису схеми таблиць. MongoDB підтримує зберігання документів в JSON-подібному форматі, має досить гнучку мову для формування запитів, може створювати індекси для різних збережених атрибутів, ефективно забезпечує зберігання великих бінарних об'єктів, підтримує журналювання операцій зі зміни і додавання даних в БД, може працювати відповідно до парадигми Map/Reduce, підтримує реплікацію і побудову відмовостійких конфігурацій. У MongoDB є вбудовані засоби із забезпечення шардінга (розподіл набору даних по серверах на основі певного ключа), комбінуючи який реплікацією даних можна побудувати горизонтально масштабований кластер зберігання, в якому відсутня єдина точка відмови (збій будь-якого вузла не позначається на роботі БД), підтримується автоматичне відновлення після збою і перенесення навантаження з вузла, який вийшов з ладу. Розширення кластера або перетворення одного сервера в кластер проводиться без зупинки роботи БД простим додаванням нових машин.

В якості системи контролю версій використовується Git.

Для розробки Front end веб-клієнту використовується Java Script, CSS та html, а для розробки серверу використовується фреймворк SpringMVC.

Для розробки мобільного додатку на платформі Android використовується Eclipse Luna з Android SDK та мова Java. Додаток не має бази даних, усі дані для зберігання відправляються на сервер POST-запитом, а для відображання - GET-запитом.

Якщо описувати модель взаємодії усіх компонентів системи, то можна зрозуміти, що головною ланкою між усіма клієнтськими додатками є сервер з REST-сервісом. В усіх трьох компонентах системи присутні одні й ті ж класи-моделі для зручності передачі даних.

5 ІНТЕРФЕЙС ТА ФУНКЦІОНАЛ

Як вже було неодноразово сказано, програмна система складається з трьох компонентів: серверу, веб-клієнту та мобільного додатку. Під час розробки програмної системи однією з цілей було створити найбільш простий інтерфейс у мобільному додатку, шляхом перенесення більшості функціоналу у веб-клієнт. Тому було вирішено залишити можливість переглядати список своїх маршрутів і визначати своє положення під час поїздки у мобільному додатку, а створення маршрутів, пошук по публічних маршрутах реалізувати у веб-клієнті.

5.1 Сервер

На сервері реалізований REST-сервіс, за допомогою якого можна скористатися функціями системи. Методи, що вертають дані, вертають їх у форматі JSON. Наведемо список доступних методів для API:

а) GET /getPolyLines/{id} –вертає всі маршрути для користувача з ідентифікатором id;

б) GET /getRoute/{id}/{routeName} – вертає маршрут з назвою routeName користувача з ідентифікатором id;

в) GET /getRouteNames/{id} – вертає список назв маршрутів користувача з ідентифікатором id;

г) GET /getRouteCount/{id} – вертає кількість публічних маршрутів користувача з ідентифікатором id;

д) GET /getPublicRoutes/{id} – вертає всі публічні маршрути користувача з ідентифікатором id;

е) POST /removeRoute/{id}/{routeName} – видаляє маршрут з назвою routeName користувача з ідентифікатором id;

є) POST /savePolyLine/{id}/{routeName} – зберігає маршрут з назвою routeName користувача з ідентифікатором id.

5.2 Веб-клієнт

Веб-клієнт є найбільш насиченим з точки зору інтерфейсу. Головна сторінка клієнту являє собою форму, за допомогою якої можна авторизуватися використовуючи Facebook-акаунт. Після авторизації користувач потрапляє на сторінку профілю (див. рис. 5.1), де він може побачити список всіх своїх маршрутів. Маршрути виводяться списком, у якому можна отримати назву маршруту та його дистанцію. Справа від назви знаходяться дві кнопки «Show» та «Х». Після натискання на кнопку «Show» справа завантажується мапа з обраним маршрутом. Маршрут відображається зеленою лінією. Кнопка «X» потрібна для видалення маршруту. Зліва знаходиться навігаційне меню, з якого можна потрапити на сторінку профілю, на сторінку створення маршруту, а також на сторінку, де можна переглянути друзів.

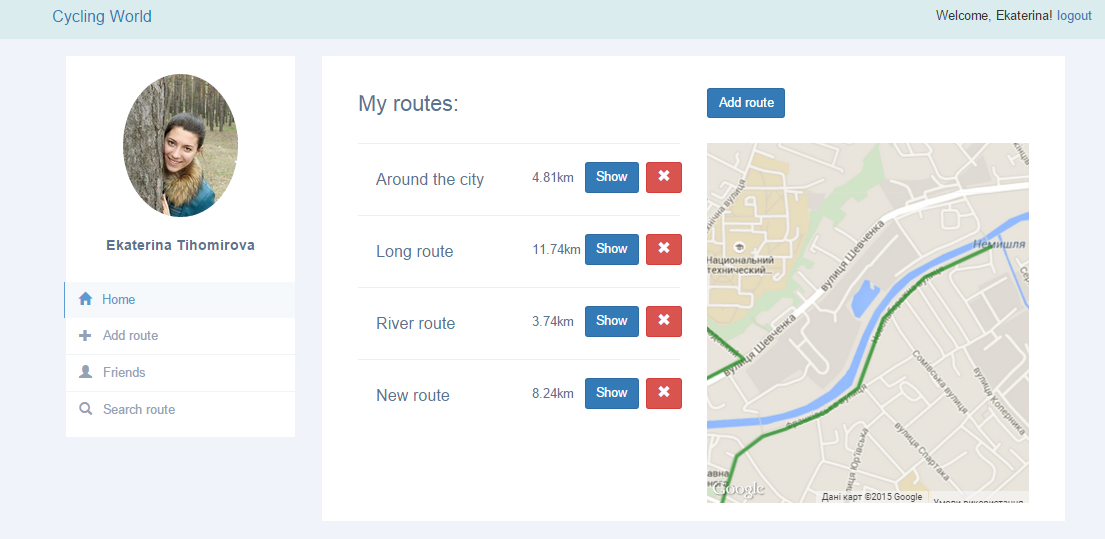


Рисунок 5.1 – Сторінка профілю

На сторінці створення маршруту (див. рис. 5.2) є карта, на якій можна створити маршрут. При натисканні на карті створюється маркер, який можна переміщувати. При повторному натисканні на маркер від видаляється. Після додавання другої точки, на середині між першою та другою точками з’являється ще один маркер, за допомогою якого можна скорегувати маршрут. Під час будь-яких дій з маркерами постійно підраховується загальна довжина маршруту.

Над картою є поле з назвою нового маршруту, яке потрібно обов’язково заповнити для створення маршруту. Користувач також може вказати, маршрут є публічним чи приватним. Для зберігання маршруту треба натиснути на кнопку «Save».

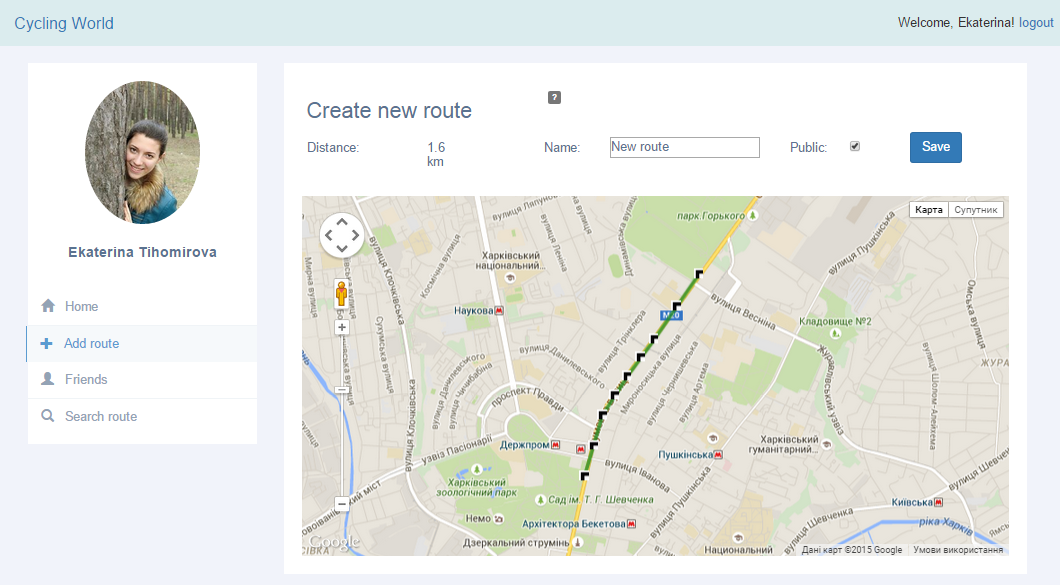


Рисунок 5.2 – Додавання нового маршруту

На сторінці з друзями (див. рис. 5.3), можна отримати список друзів, які користуються додатком та отримати скільки в якого друга публічних маршрутів. Також можна запросити друга скористуватися додатком натиснувши на кнопку «Invite friend to CyclingWorld». Після натискання з’явиться вікно, де можна вибрати кому відіслати посилання на CyclingWorld, та додати текст повідомлення.

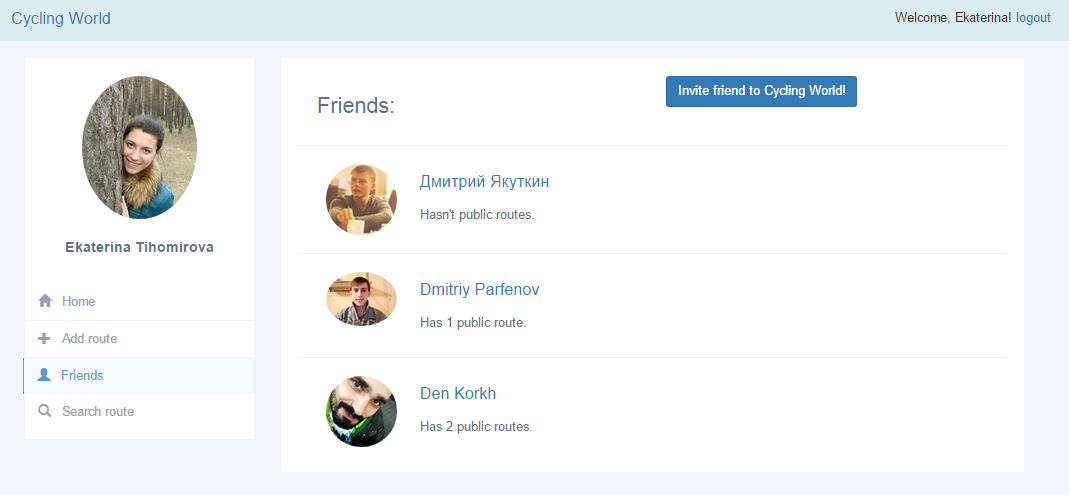


Рисунок 5.3 – Сторінка з друзями

5.3 Додаток для Android

Мобільний клієнт має досить простий інтерфейс, що повністю відповідає усім функціям, що він повинен виконувати. Після відкриття додатку користувач має авторизуватися через свій Facebook-акаунт, натиснувши кнопку «Log in with Facebook». Після авторизації він потрапляє на вікно, у якому є список всіх його маршрутів. Для переглядання будь-якого маршруту користувач потрібен натиснути на кнопку «Show» поряд з назвою маршруту. Після натискання користувач потрапляє на сторінку з картою, на якій вже відображений обраний маршрут. Якщо GPS увімкнений, користувач може визначити своє положення на карті.

ВИСНОВКИ

У результаті виконання курсової роботи був розроблений програмний продукт, що дозволяє керувати веломаршрутами та поїздками за ними. Система базується на сервісно-орієнтованній архітектурі, тому складається з окремих частин: сервера, веб-клієнта та мобільного додатку.

Сервер та клієнт для платформи Android було розроблено з використанням мови програмування Java. Сервер було розроблено у середовищі Spring Tool Suite, мобільний додаток у Eclipse Luna з Android SDK. Сервер розроблявся з використанням фреймворка SpringMVC та був розгорнутий у хмарному середовищі Pivotal Cloud Founry. База даних розміщується у сервісі MongoLab.

В результаті розробки поставлену задачу було цілком виконано. Програмний продукт має зрозумілий інтерфейс для користувачів, не викликає труднощів з відправкою даних на сервер.

Як вже зауважувалося вище, програмний продукт навіть після реалізації усіх поставлених на курсову роботу задач, для комерційного запуску має дороблюватися. Перш за все, необхідно доробити додатковий функціонал та створити комерційну складову проекту. Другорядним завданням стоїть реалізація клієнтів для різних мобільних платформ (iOS, OS Android, Web), це дозволить покрити більшість сучасних смартфонів та планшетів. Також, необхідно доробити локалізацію.

ПЕРЕЛІК ПОСИЛАНЬ

1. Вікіпедія [Електронний ресурс]/ Режим доступу: www/URL: http://uk.wikipedia.org/wiki/GPS - GPS.
2. MongoDB docs [Електронний ресурс]/ Режим доступу: www/URL: http://docs.mongodb.org/manual/ - The MongoDB 3.0 Manual.
3. Фаулер, М. UML. Основы [Текст] : пер. с англ. А.: Петухов/ М. Фаулер, К. Скотт. - СПб.: Символ, 2006. - 184 с.
4. Мартин, Р. Чистый код. Создание, анализ и рефакторинг [Текст] : пер. с англ. Е.: Матвеев/ Р. Мартин. - Питер, 2010. - 464 с.
5. Еккель, Б. Философия Java [Текст] : пер. с англ. Е.: Матвеев/ Б. Еккель. - Спб: Питер, 2001.- 880с.
6. Смирнов, Н. И. JAVA 2 Enterprise. Основы практической разработки распределенных корпоративных приложений [Текст]/ Н.И. Смирнов. – М.: КУДИЦ-ОБРАЗ, 2002. – 240 с.