Міністерство освіти і науки України
Харківський національний університет радіоелектроніки
Факультет _____Комп'ютерних наук_____
(повна назва)

Кафедра _____Програмної інженерії_____
(повна назва)

АТЕСТАЦІЙНА РОБОТА
Пояснювальна записка

_____другий (магістерський)_____

Дослідження методів організації спільних поїздок

Виконав: студент 2 курсу, групи ПЗСм-17-3___
спеціальності 121- Інженерія програмного забезпечення

Освітньо-наукової програми
Інженерія програмного забезпечення_____

_____Журавок М.О._____

Керівник _____проф. Бондарев В.М._____

Допускається до захисту
Зав. кафедри, проф. _____ З.В.Дудар
2019 р.

Харківський національний університет радіоелектроніки

Факультет_Комп'ютерних наук_____

Кафедра_Програмної інженерії_____

Рівень вищої освіти_____другий (магістерський)_____

Спеціальність _121-Інженерія програмного забезпечення_____
освітньо-наукова програма _Інженерія програмного забезпечення_____

ЗАТВЕРДЖУЮ:
Зав. кафедри _____

«____»_____ 20 ___ р.

ЗАВДАННЯ

НА АТЕСТАЦІЙНУ РОБОТУ

студентові _Журавку Михайлу Олексійовичу_____

1. Тема роботи _Дослідження методів організації спільних поїздок _____ _____
затверджена наказом по університету від "___" _____20 ___ р № _____
заповнюється вручну після отримання наказу
2. Термін подання студентом роботи до екзаменаційної комісії
20 червня 2019 р.
3. Вихідні дані до роботи *статистика публічна системи blablacar, open source репозиторій github Blablacar'.*
4. Перелік питань, що потрібно опрацювати в роботі *система роботи програмних ресурсів організації спільних поїздок, переваги та недоліки кожної з систем, регуляція питань безпеки у системах, в першу чергу Blablacar, проектування та моделювання власного програмного продукту, аналіз відкритого репозиторію на знаходження особливостей для власної системи.*

5. Перелік графічного матеріалу із зазначенням креслеників, схем, плакатів, комп'ютерних ілюстрацій (слайдів) *Презентація для захисту – скриншоти з програм існуючих сервісів спільних поїздок – Blablacar, OnTaxi; скриншоти репозиторію Блаблакар та аналітичні графікі того ж репозиторію.*

6 Консультанти розділів роботи

Найменування розділу	Консультант (посада, прізвище, ім'я, по батькові)	Позначка консультанта про виконання розділу	
		підпис	дата
Снецчастина	проф. Бондарев В.М.		

КАЛЕНДАРНИЙ ПЛАН

№	Назва етапів роботи	Терміни виконання етапів роботи	Примітка *
1.	Аналіз предметної галузі	19 квітня 2019р.	
2.	Огляд існуючих методів	27 квітня 2019р.	
3.	Методи швидкого детектування відрізків ліній		
4.	Підготовка пояснювальної записки	25 травня 2019р.	
5.	Спецчастина	26 травня 2019р.	
6.	Підготовка презентації та доповіді	28 травня 2019р.	
7.	Попередній захист	30 травня 2019р.	
8.	Нормоконтроль, рецензування	02 червня 2019р.	
9.	Занесення диплома в електронний архів	03 червня 2019р.	
10.	Допуск до захисту у зав. кафедри	04 червня 2019р.	
* заповнюється вручну після виконання чергового пункту			

Дата видачі завдання _____ _____2019 р.

Студент _____
(підпис)

Керівник роботи _____ __проф. Бондарев В.М._____
(підпис) (посада, прізвище, ініціали)

РЕФЕРАТ / ABSTRACT

Звіт з науково-дослідної практики магістрів: 57 с., 33 рис., 11 джерел, 1 додаток.

СПІЛЬНІ ПОЇЗДКИ, КЛІЄНТ-СЕРВЕР, СЕРВІС, БЕЗПЕКА, ДЕЦЕНТРАЛІЗАЦІЯ

Об’єкт – методи організації спільних поїздок.

Мета роботи – дослідження роботи сервісів, надаючих послуги спільних подорожей чи допомогаючих в об’єднанні людей для спільного переміщення.

В роботі проведений аналіз цільових програмних продуктів, зрівнено важливі аспекти різних тематичних програмних продуктів (безпека, зручність, підтримка різних країн та мов), виокремлення найкращих програмниї продуктів, створення інновацій та проектування свого програмного продукту, реалізуючого спільні поїздки.

JOINT TRIPS, CUSTOMER SERVICE, SERVICE, SAFETY, DECENTRALIZATION Object - methods of organization of joint trips.

The purpose of the work is to investigate the work of services that provide services for joint travel or help in bringing people together for joint movement. In this work, the analysis of target software products, the comparative analysis of aspects of PP (safety, convenience, support of different countries and languages), the selection of the best PPs, creation of upgrading innovations and designing of their software product, implementing joint trips.

ЗМІСТ

ВСТУП

Проблема організації спільних поїздок – є далеко не останнью проблемою сучасного світу. З розвитком стільникових телефонів, а точніше смартфонів, люди почали все більш і більш довіряти информації з програмних продуктів внутрішніх маркетів (app store/ android market). Та насправді, модерація усіх ПП проходить кваліфікованою групою адміністраторів, але дійсно неможливо розібрати усі можливі сценарії взаємодії користувача з даною програмою.

Найпоширеніші засоби для організації спільних поїздок – blablacar, beepcar (прототип блаблакару, який згодом був викуплений та приєднаний до найбільшого ПП спільних поїздок), також у Росії використовується аналог «Яндекс.Поездки»; до спільних поїздок за деякими параметрами можно добавити також системи онлайн таксі, такі як onTaxi чи uber. У всіх цих систем, процес регістрації та верифікації має мінімальні вимоги до користувача – реальні випадки в новинах підтверджують можливість найстрашніших ісходів внаслідок немаксимальної перевірки психологічного стану кліентів подібних служб.

Об’ект дослідження – реакція на події, представлені у ЗМІ, що побудили небайдужість до проблем безпеки подобних сервісів. Тому наша ціль також прийти висновку – чи можно прийти до балансу в зручному використанні та більш надійної верифікації, бо навіть найбільша автоматизація повинна мати деякі захисні механізми. Ми повинні знайти баланс між зручністю та безпечністю, звичайно з перевагою на безпеку, бо використовується даний сервіс кожний день незліченною кількістю людей. (400 співробітників з усього світу, 15 млн користувачів їздять з BlaBlaCar щокварталу, 5 млн щомісяця, понад 3 мільярди кілометрів опублікованих маршрутів (можна зробити 80 535 кругосвітніх подорожей)

Саме тому наша ціль змоделювати утопічно-ідеальний сервіс спільних поїздок, спроектувати його на заміщенні нереального в моделі на актуальні та можливі засоби безпеки, та підготовити дійсно готовий скелет продукту, якому до повної роботи лише не буде хватати всесвітнього покриття, технічної команди підтримки та деяких мажорних аспектів, не достатніх для реалізації одному.

1 АНАЛІЗ ПРОБЛЕМНОЇ ГАЛУЗІ ТА ПОСТАНОВКА ЗАДАЧІ

1.1 Актуальність обраної теми

Напевне кожній людині доводилося планувати поїздку до іншого міста чи країни та зовсім незавжди це просто зробити – немає прямого маршруту чи потрібні зайві документи для міжнародних перельотів, чи просто неіснує такого авіа чи автошляху. Також спільні поїздки у багатьох випадках надають вам гнучкості – по рейсу Харків – Київ відправлення є кожні 5-10 хвилин кожен день року без виключень. В деяких випадках доцільно використання спільних поїздок для економії грошей. Ця низка та деякі «екзотичні» причини роблять системи організації спільних поїздок невід'ємною частиною Apple Store чи Google Market'у.

Ваша подорож з попутниками розпочинається не в дорозі, а під час планування маршруту і спілкування з учасниками спільноти BlaBlaCar. Водії і пасажири, які користуються сервісом — це зареєстровані користувачі, які піклуються про власну репутацію та рейтинг та формують довіру. Подорожуйте з попутниками, які заповнили профіль, приєднали облікові записи Facebook і LinkedIn, а також мають хороші відгуки від інших.

Райдшарінг (ride sharing) стає все більш популярним в Україні, і кожному варто знати, що очікувати від цього своєрідного, не позбавленого романтики і зручності сервісу пересування. Варто чітко окреслити, ніж цей формат відрізняється від звичайного таксі, а також показати, чого варто чекати від середньостатистичної поїздки.

Основним моментом дискусій та щоденних головних болей таких служб – є безпека. Безпека водія, безпека пасажира, безпека даних (без цього вочевидь не може працювати жодний програмний продукт в XXI віці).

Наприклад найвідоміший сервіс BlaBlaCar, в основних цифрах:

- більше 60 мільйонів користувачів;

- 22 країни: Бельгія, Великобританія, Голландія, Індія, Іспанія, Італія, Люксембург, Мексика, Німеччина, Польща, Португалія, Росія, Румунія, Сербія, Туреччина, Угорщина, Україна, Франція, Хорватія, Бразилія, Чехія, Словаччина;
- 400 співробітників з усього світу;
- 15 млн користувачів їздять з BlaBlaCar щокварталу, 5 млн щомісяця;
- понад 3 мільярди кілометрів опублікованих маршрутів (можна зробити 80 535 кругосвітніх подорожей);
- близько 275 мільйонів євро заощаджено нашими водіями;
- понад 30 мільйонів завантажень мобільного додатку (iOS та Android).

Актуальність питання не має ніяких питань, це дійсно важливий програмний продукт – наша ціль спроектувати максимально безпечну систему та винести дійсно невирішуюмі питання на публічний дискус, бо глобальна комуна зможе знайти рішення чи деякі уразливості системи винести в прийняття прав користувача, щоб кліенти сервісу завжди знали небезпечність деяких аспектів та брали ці питання на себе.

1.2 Аналіз проблем та рішень у предметній галузі

BlaBlaCar (БлаБлаКар) — це найбільша у світі спільнота попутників (сайт і мобільні додатки для iOS та Android), що об’єднує водіїв і пасажирів, яким по дорозі, з простою метою — подорожувати вигідно і зручно. Ефективна команда служби підтримки користувачів, сучасні веб та мобільні рішення, швидко зростаюча спільнота користувачів — все це робить BlaBlaCar унікальною платформою для ефективних та економних подорожей для мільйонів користувачів по всьому світу. Ідея створення сервісу для спільних поїздок вперше з’явилася у француза Фреда Мазелли у 2004 році, а через рік Франція побачила BlaBlaCar. На ринок України BlaBlaCar вийшов

на початку 2014 року, купивши успішний український стартап «Подорожники», і на сьогоднішній день стрімко розвивається. Змоделюємо очевидні ситуації, які сильно загрожують репутації сервісу, але з ними всеодно кожен раз зіткується користувач системи (спочатку – мізерні мінуси, порушення котрих стосується більшості користувіачів, бо є мінорними відхиленнями, до злочинних намірів та найгріших огріхів системи):

а) відсутність жорсткої модерації анкет – погані звички, музика в авто, можливість транспортування живності. Ці дані не є основопологаючими в системі та контроль над правильністю цих «опцій» мінімальний – висновок про можливість паління та зловживання цим під час поїздки, зазвичай ви зможете зробити вже у подорожі. Тим паче водіям завжди корисніше зазначити ідеальний перелік цих даних, для підвищення конвертаційності профіля, навіть, коли водій не займається перевозами на постійній основі, він має деякий коммерційний інтерес, бо розмістив заявку на сервісі;

б) оцінка за рівень вміння водити автомобіль – дуже суб’єктивна річь, а тим паче підхід системи – перший раз оцінка йде від водія, потім же в хід йде користувацька оцінка. Вочевидь, що оцінка користувачів може штучним чином проставлятися, та поставив n раз найвищу оцінку, навіть найжлахливіший водій, може довгий час їздити с задовільною оцінкою. Цей параметр вже є більш ключевішим, хоч і на перший погляд, дуже суб’єктивна цифра – бо за сер’озні порушення за жахливі відгуки аккаунти забанять, але взимку вночі, саме я їхав би лише з водієм з максимальною оцінкою – наприклад, Шумахером чи Феттелем;

в) можливість мультиаккаунтингу – боротьба з поганими відгуками чи дійсно тяжкими порушеннями. Все просто – зробив погану поїздку, аккаунт втратив у кредиті довіри – просто створи новий та почни все з чистого аркуша, потім обхідними шляхами декілька гарних оцінок і відгуків, і знову недобросовісний водій має дуже приємний аккаунт;

г) ніякої реальної прив’язанності водія до аккаунту та пасажира до аккаунту. Невирішуєма проблема, яка ставить під сомнів можливість

створення систем спільних поїздок, бо кількість перевірок зі всіх аспектів, сягає неможливих обсягів. Це дійсно реалії, але за одним аккаунтом Блаблакару за сутки між містами зазвичай їздять 2-3 водії, та не завжди на одній автівці. Це війшло вже в звичаї та не непокоїть громадянинов, але в дійсності, існує величезна кількість проблем та навіть прецедентів, у випадках коли люди, йдуть на контакт, з повною довірою до будь якої людини, яка за певних обставин може заполучити доступ до клієнтської частини програмного продукта водія з гарним (за нормами сервісу) профілем;

Блаблакар затверджує роботу в сторону модерації саме так: Водії і пасажири, які користуються сервісом, розповідають про себе у профілі. Ми уважно модеруємо профілі водіїв і пасажирів, які реєструються на BlaBlaCar. Ми перевіряємо профілі користувачів, верифікуємо фотографії, переглядаємо відгуки та пропозиції поїздок, щоб гарантувати взаємну довіру та повагу у спільноті.

1.3 Огляд існуючих систем організації спільних поїздок

1.3.1 Огляд системи BlaBlaCar

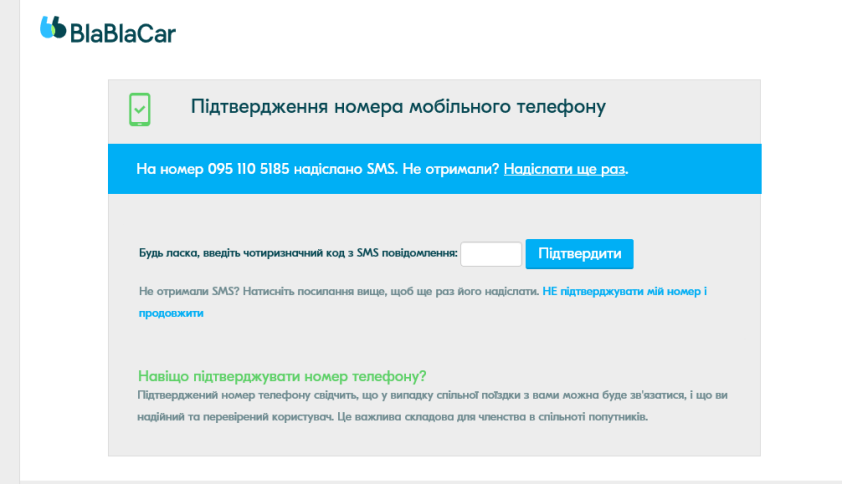

Рисунок 1.1 – Регістрація у сервісі BlaBlaCar

На рисунку 1.1 зображено останній етап первинної регістрації у системі BlaBlaCar. До цього у простих формах у користувача дізнаються його електронну скриньку, ім’я та прізвище. На цьому етапі ми маємо найпростішу авторизацію, що дозволяє нам користуватися сервісом – проглядувати список поїздок, подавати заявки на бронювання.

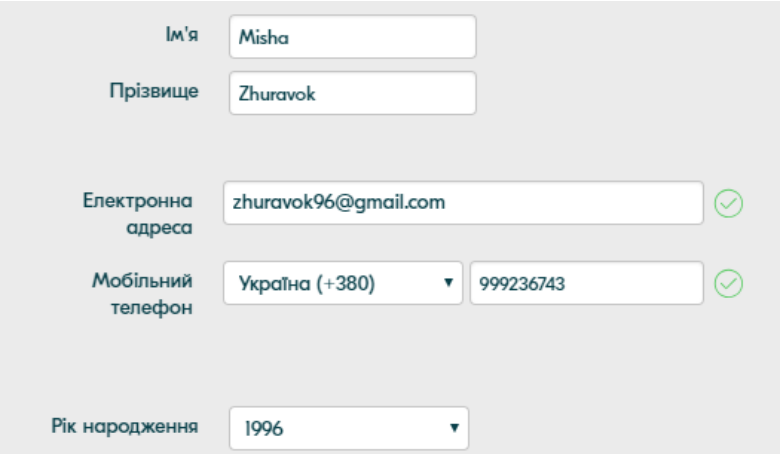

Рисунок 1.2 – Профіль після регістрації

На рисунку 1.2 усі дані, які має Блаблакар про користувача, а саме – мобільний телефон, електронна адреса, рік народження, ім’я та прізвище.

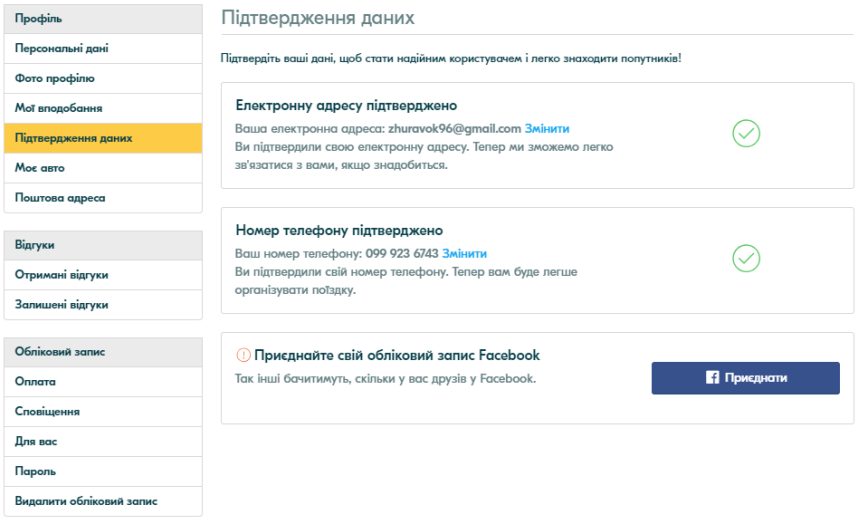

Рисунок 1.3 – Меню підтвердження даних

На рисунку 1.3 зображена сторінка з підтвердженням даних користувача – якщо відокремити електронну адресу та номер мобільного телефону, які зовсім не важко за 15 хвилин отримати чи просто придбати в інтернеті за мінімальну суму, залишається обліковий запис Facebook.

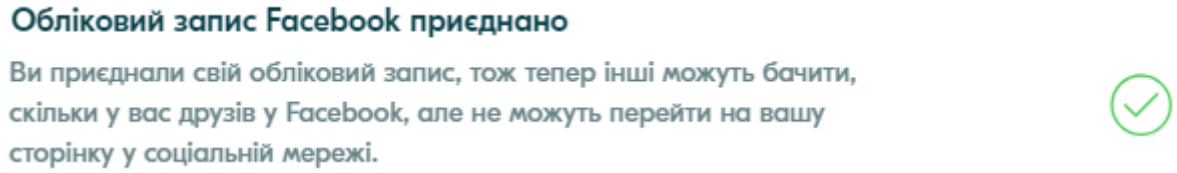

Рисунок 1.4 – Підїєднання облікового запису Facebook

Після під’єднання аккаунта Фейсбука (рисунок 1.4), також не проходить ніякої верифікації. Усе, що змінюється, для клієнтів системи – ви бачите кількість друзів у соціальній мережі. Навіть не здогадуюсь, яким чином творці Блаблакару розробили даний функціонал, але це не є метрикою якоїсь фізичної чи соціальної характеристики людини, якщо навіть не приймати за увагу факт легкої підробки цього числа – взяти у оренду сторінку з «достатньою» кількістю друзів, купити собі тих самих «друзів» - нульові сторінки, які скриптом добавляють у друзі будь-кого, хто заплатить гроші.

Подивившись на розділ – підтвердження даних, здається що це просто муляж будь-якої верифікації, бо дійсно пройшовши усі три перевірки (результати яких підробити не коштує великих грошей), підтверджені дані – не верифікують ніяку людину та не створюють жодних обмежень, щодо клієнтів служби Блаблакар.

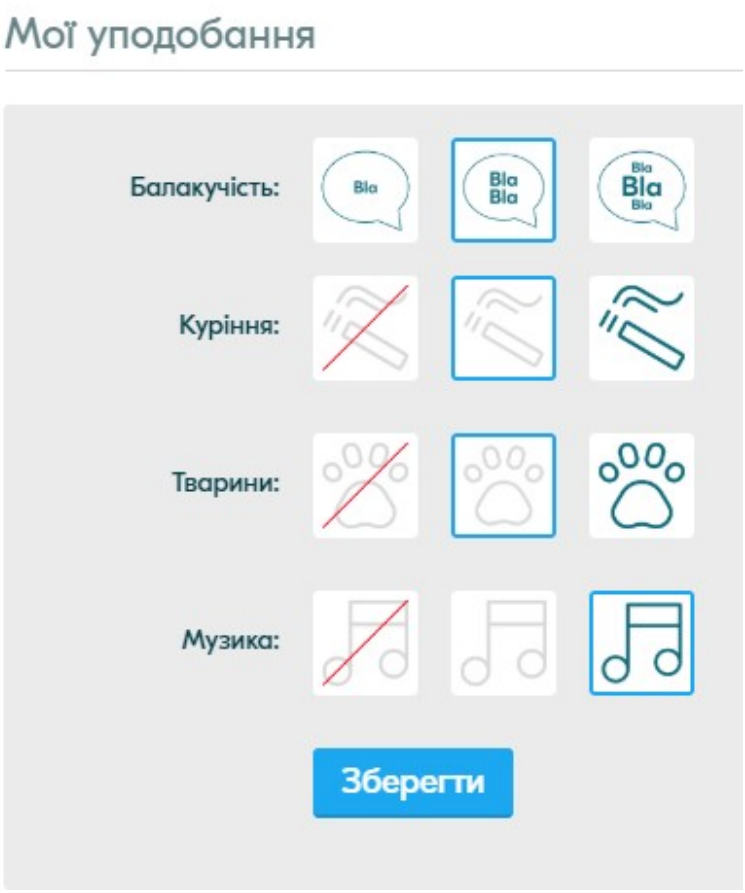

Рисунок 1.5 – Сторінка «мої уподобання

На рисунку 1.5 зображено сторінку ресурсу на якій ти можеш вибрати уподобання щодо розмов в машині, паління, находження в салоні тварин та гучності/репертуару музики.

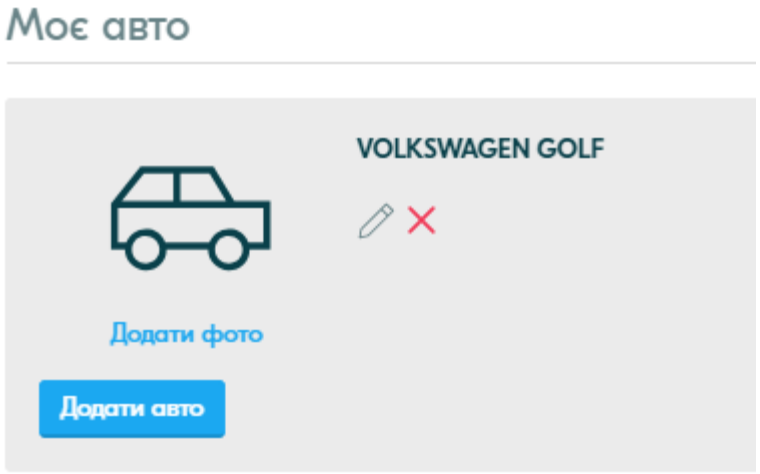

Рисунок 1.6 – Панель «Моє авто»

На рисунку 1.6 відображена панель, у якій користувачі-водії, зберігають дані об автівках, які використовують для організацій спільних поїздок. Процес додавання автівки – також не переслідує жодної верифікації. Першим полем для вводу у анкеті додати авто – є додати державний номерний знак (який не відображается у системі для користувачів за умовами конфіденційності). Перша моя думка була – що с витягу з державноъ бази, за кузовним номером автоматично підтягнеться усі дані про мій автомобіль, але ні – потім я послідовно надавав інформацію про машину, та її прийняли. Я спеційно ввів неіснуючий номер автомобілю (точніше сказати – вигаданий, мабуть він і існує) та його також прийняли до мого профілю без якихось проблем. Саме до таких моментів у системі направлені мої притензії.

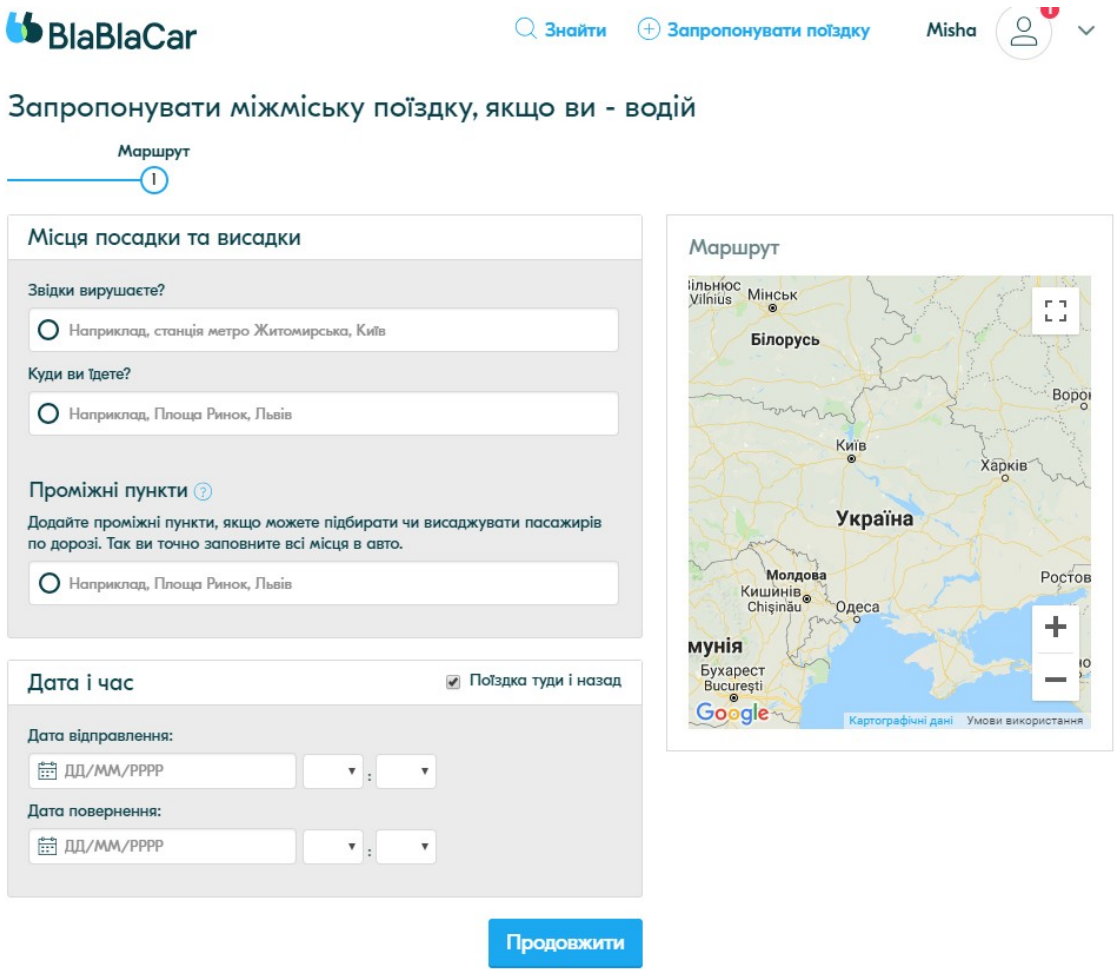

Рисунок 1.7 – Шаблон «запропонувати поїздку»

Зручний та оптимальний шаблон для створення поїздки у систему блаблакар – з інтерактивними календарями, поміччю при написанні міста (для відправки та прибуття). Також у наступному меню ви вибираете машину та деталі поїздки, що зображено на рисунку 1.8.

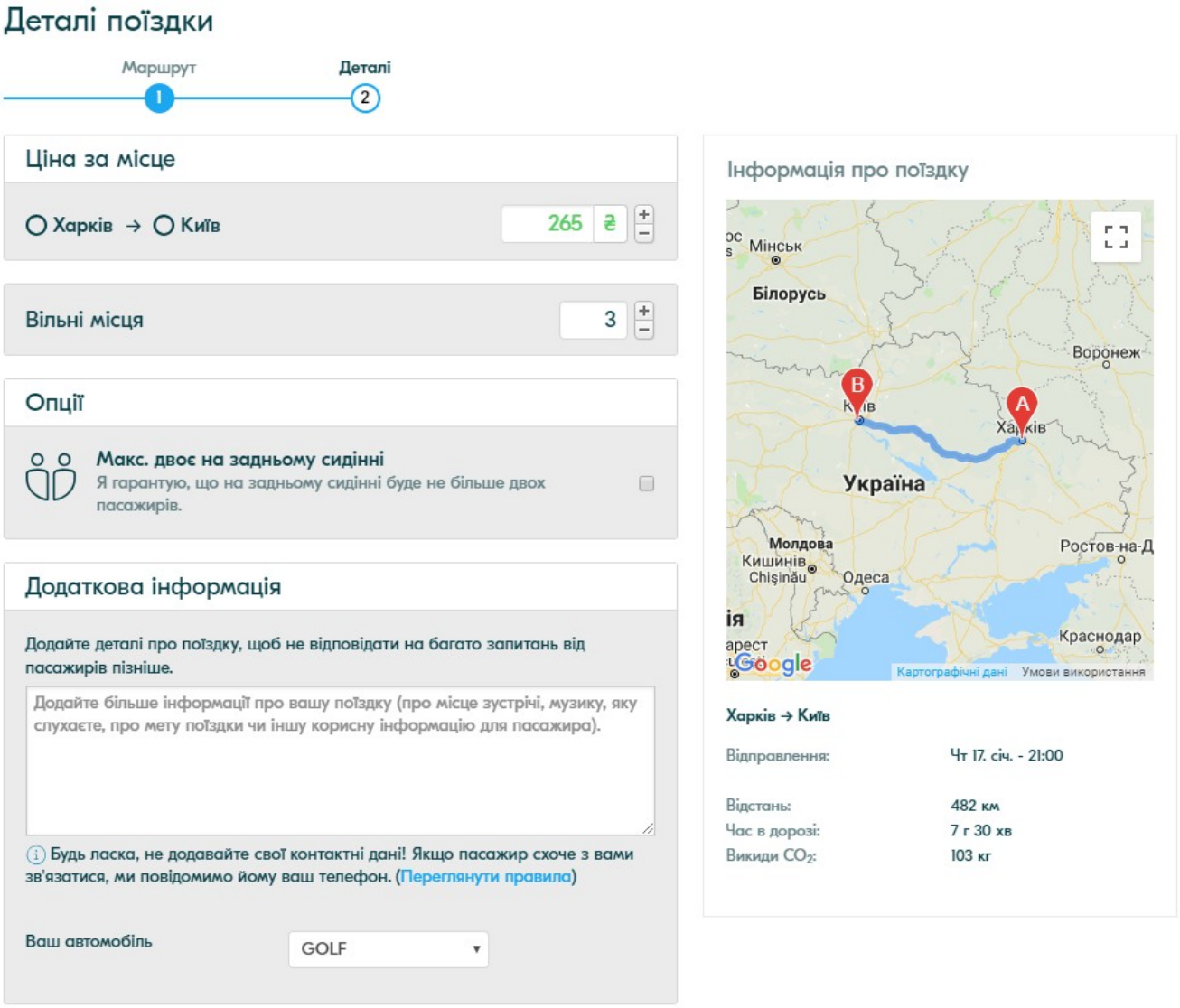

Рисунок 1.8 – Продовження організації спільної поїздки

Тут водій може встановити потрібну йому ціну, кількість вільних місць в автомобілі та можливі опції (список опцій змінюється від типу автомобілю – вантажівка, автобус, мікроавтобус, легкова машина). Також ви можете залишити додаткову інформацію, щоб спростити дальнішу комунікацію з пасажирами.

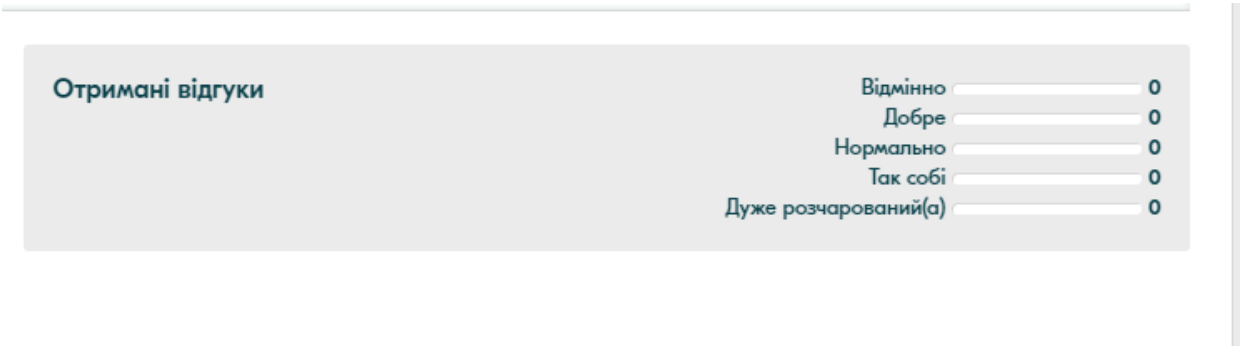

Рисунок 1.9 – сторінка залишених відгуків

На рисунку 1.9 показана сторінка відгуків – я, від'їздивши більше десяти поїздок типу місто-місто, не отримав жодного відгуку від водія. Таким чином, нема прямого бажання залишати відгуки в Блаблакарі (нашого регіону) та сподіватися на відгуки також не завжди придеться – якщо поїздка тобі не сподобалась, легше видалити програму з телефона та користуватися іншим транспортним засобом.

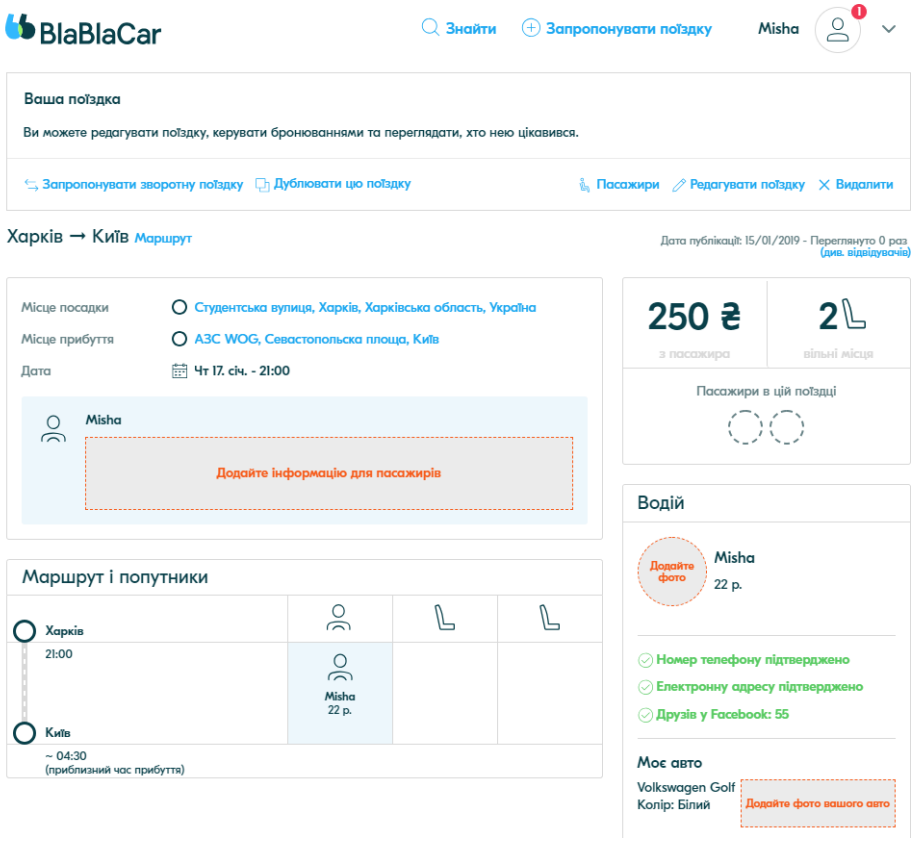

Рисунок 1.10 – Залишена мною «недійсна» заявка на Блаблакарі

На рисунку 1.10 останній крок мого використання сервісу – я залишив недійсні дані поїздки, з неіснуючою машиною, з великою кількістю друзів в Фейсбук – з низькою ціною, та я певен, що не відміни я заявку, знайшлись би пасажири на таку дешеву поїздку на гарній машині.

1.3.2 Сервіс спільних поїздок OnTaxi

Даний сервіс не виконує усіх функцій повноцінного сервісу спільних поїздок, але часто групи людей використовують їх для пересування ланцюгами (декілька адрес, але в межах міста); виконуються ті ж ролі – водія та пасажира, та за поїздку кліенту прийдется заплатити. Саме з цієї системи, я підчерпну моменти безпеки – але спочатку проаналізуємо структуру програмного продукта.

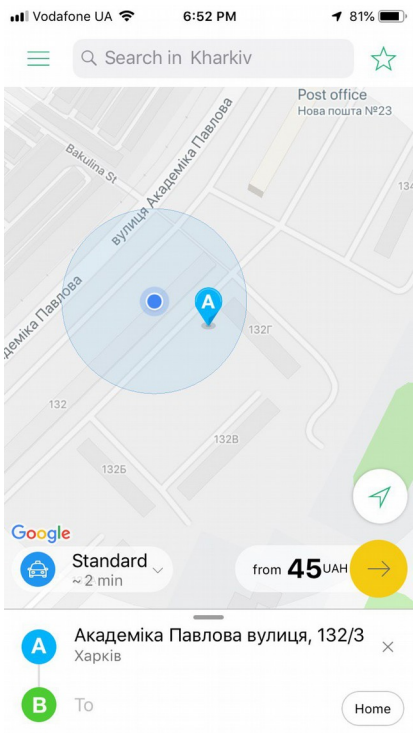

Рисунок 1.11 – Головне меню сервісу OnTaxi

На рисунку 1.11 зображено головне меню системи – тут є два поля вводи для пункту відправки та для кінцевого пункту. Також система автоматично знаходить вашу геолокацію і допомагає вам, завдяки GPS, спостерігати за тим, як ваше таксі їде до вас по карті. Після поїздки ви теж зможете залишити відгук і ваші враження про водія, оцінити роботу шофера. Якщо ваша геолокація значно відрізняється від пункту заказу таксі – диспетчер дозвониться на ваш мобільний номер та обов'язково дізнається причину «незвичного» заказу.

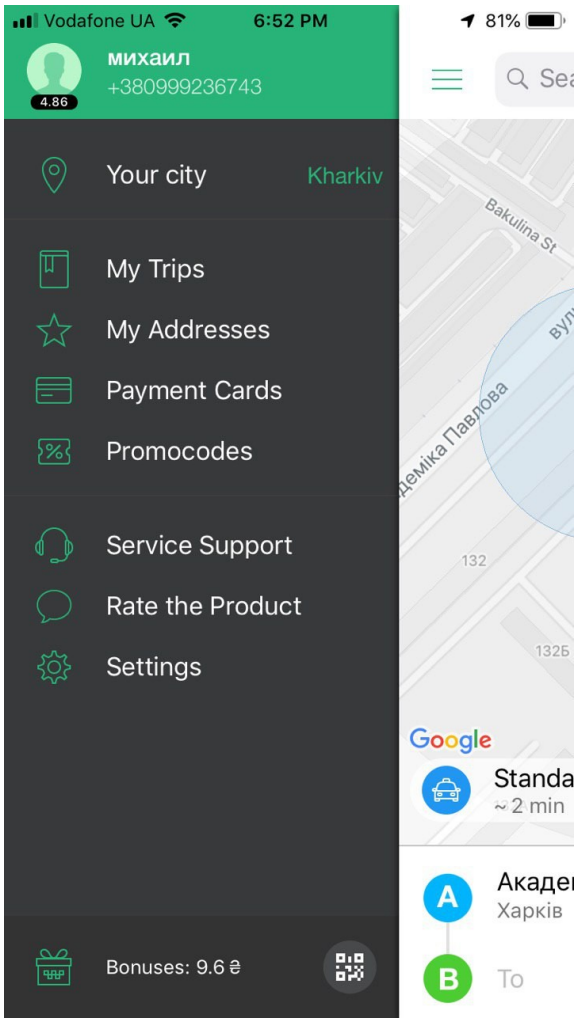

Рисунок 1.12 – Усі можливі налаштування у программі сервісу

На рисунку 1.12 представлені всі можливості ОнТаксі – сервісна пітдримка, оцінити продукт та глобальні налаштування. Також ви можете поставити закладку з домашньою адресою, продивитись ваші попередні поїздки, прив’язати кредитну картку чи ввести промокод. Також ви можете змінити деякі дані профілю – мобільний номер, електронна скринька та місто вашого знаходження (є п’ять філіалів служби по Україні). Саме цим я добавляю сервіс до дипломної роботи – диспетчер контролює усілякі сценарії, що можуть відбуватися під час використання служби.

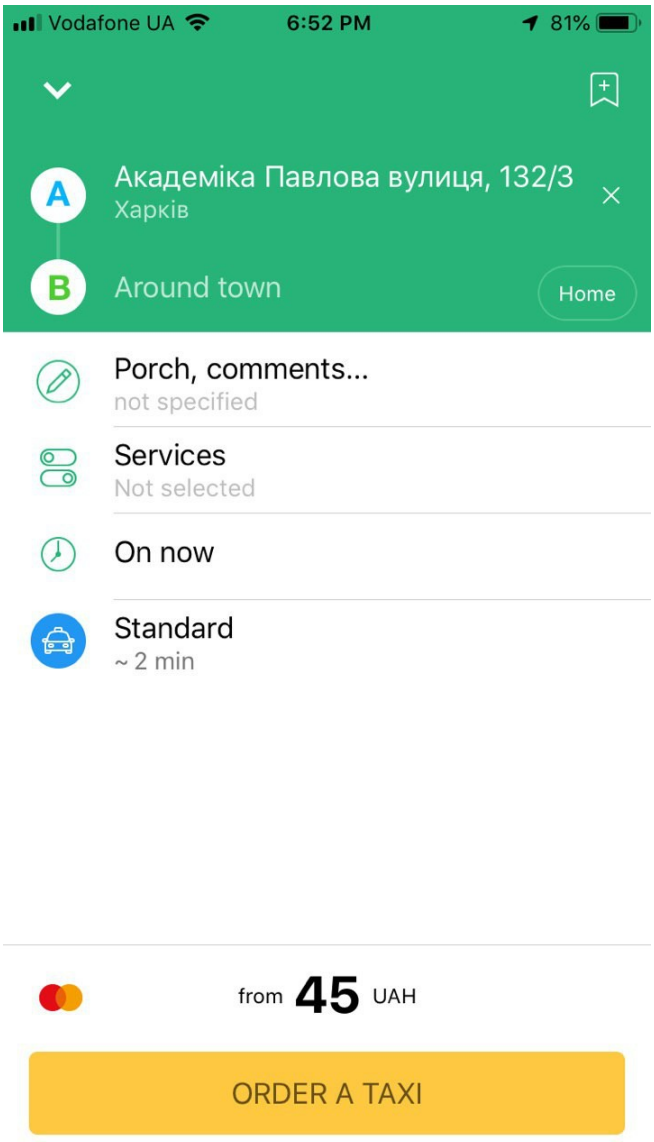

Рисунок 1.13 – Вікно замовлення таксі. Роль пасажир

На рисунку 1.13 – останнє вікно, яке міститься у сервісі, це вікно замовлення таксі. Тут кліент вибирає деталі місцезнаходження – під'їзд чи установа, в якій він знаходиться, ставить примітки (хвороби, потребує в курінні, є вантаж і т.д.) та, в кінці кінців, вибирає спосіб оплати поїздки.

1.4 Переваги сервісів

Деякий аналіз аспектів різних сервісів каршерінгу чи спільних поїздок ми залишимо під капотом, в дану роботу винесу накращі, за моїм висновком, моменти з кожного сервісу.

Blablacar. Звичайно, розділ з перевагами цього сервісу буде найбільшим, тож почнем:

- зручність. Через сайт і його мобільний додаток можна швидко знайти водія, який буде їхати в потрібному напрямку, в відповідний день і навіть час. Доступні і вузькі настройки: наприклад, реально вибрати некурящого водія, який не включатиме «шансон» або розмовляти всю дорогу про політику;
- виокремлення найважніших характеристик водія з транспортним засобом. Вибираючи водія, користувач бачить його вік, марку і клас автомобіля, особисті уподобання і пропоновану вартість поїздки. Наявність відгуків і гарного рейтингу - додатковий фактор вибору;
- на основі існування характеристик – можливість сортування, фільтрації актуального образу водія для користувача;
- популярність сервісу та наявність, без сумніву, найбільшої пропозиції на ринку. Це, як і всюди, діє як цепна залежність – в найпопулярнішому сервісі найбільша цінність для нових користувачів для регистрації.

«Метнись Кабанчиком»:

- передбачена можливість знайти водія для спільної поїздки і власника автомобіля для вантажних перевезень;

- з сайтом сервісу не виникає проблем: якщо попередній сервіс дає можливість шукати водіїв серед наявних варіантів, «Метн кабанчика» дозволяє стати замовником самостійно. Для пошуку водія потрібно оформити завдання, вказати умови, дати та ціни, яку планується заплатити. Цей процес в середньому займає 2-3 хвилини. Після досить почекати: зацікавлені водії самі побачать завдання і подадуть заявки;

- адміністрація сервісу строго стежить за об'єктивністю зазначеної в профілі інформації, перевіряє паспортні дані виконавців, не дає можливості зробити «накрутку» рейтингу або відгуків. Стоїть відмітити, що саме з цих моментів з’явилась деяка важкість у використанні сервісу – для регістрації та організації будь-якої спільної поїздки потрібно пройти багато перевірок, саме тому саме цей сервіс використовується більш у коммерційних цілях та при повторному використанні.

«Везёт Всем»:

- система пропонує додаткові можливості - наприклад, страхування вантажу і відстеження маршруту. В цілому, більшість клієнтів задоволені сервісом, але є і недоліки: наприклад, обраний водій на місці може спробувати змінити ціну перевезення або диктувати власні умови.

«ВПути»:

- сервіс забезпечує відстеження кожної поїздки, і при виникненні будь-яких скарг на порушення правил з боку водія або пасажира блокує акаунт порушника. Така «чистка рядів» дозволяє значно знизити ризики неякісних поїздок.

1.5 Недоліки сервісів

Сервіс колективних поїздок BlaBlaCar в Україні запустився давно, але особливо актуальним стало з настанням економічної кризи і, як наслідок, космічних цін на паливо і подорожчання перевезень в цілому. І водії, і пасажири раді заощадити на транспортних витратах. Однак заради цього не варто нехтувати безпекою, адже ніколи не знаєш, наскільки чесні наміри твоїх попутників.

У деяких країнах BlaBlaCar верифікує користувачів по паспорту, однак в Україні такої функції все ще немає, і користувачам необхідно самим убезпечити себе від контактів зі злочинцями. Питання особливо актуальне в світлі останніх подій.

Водії і пасажири, які здійснюють спільні поїздки за допомогою сервісу BlaBlaCar - невипадкові люди. Це зареєстровані користувачі, які:

- заповнили свій профіль;
- додали фотографію;
- інформацію про себе і про транспортний засіб;
- пройшли верифікацію номера телефону та електронної пошти.

Більшість цих даних перевіряються вручну адміністраторами сервісу. У разі, якщо потрібно уточнити дані, співробітники служби підтримки можуть навіть передзвонювати користувачеві. Крім того, користувачі мають можливість зателефонувати зі своїми попутниками перед поїздкою, щоб уточнити деталі і задати хвилюючі питання.

Також користувачі усвідомлюють, що всі дані про комунікації між людьми зберігаються в базі даних сервісу, що є додатковою гарантією безпеки.

На сайті BlaBlaCar є опція «Поскаржитися». Досить клікнути на прапорець на сторінці користувача і вибрати відповідну проблему, щоб служба

підтримки отримала автоматичне повідомлення і вжила заходів. Наприклад, якщо ви знаєте, що водій займається комерцією або сумніваєтеся в надійності пасажира, скористайтеся даною опцією і відправте повідомлення адміністраторам. Ця функція допоможе спільноті ще більш ефективно самостійно формувати рівень довіри в суспільстві. Одним з ключових моментів, які формують довіру в суспільстві, є відгуки попутників. Перед тим, як домовитися про поїздку, користувачі можуть прочитати відгуки від інших попутників, з якими їм доводилося здійснювати поїздки. Відгуки і кількість поїздок, здійснених за допомогою сервісу, допомагають користувачам підвищувати персональний рейтинг на сайті, адже від цього залежить, як швидко вони знайдуть попутників для наступної поїздки. Якщо ви сумніваєтеся у водії або пасажира - просто знайдіть перевіреного спільнотою користувача з відгуками і досконалими поїздками. Сьогодні в деяких країнах blablacar тестує і впроваджує страхові продукти, які, на додаток до існуючого обов'язковому страховим полісом, зможуть допомогти водієві в дорозі і навіть покривають ризики пасажирів щодо забутих в автомобілі речей.

У Туреччині нещодавно почала працювати система ідентифікації користувачів через урядове API, а в Індії - через скан документів, що засвідчують особу. Природно, надані копії документів не видно користувачам, але в профілі з'являється позначка, що людина надав свої ID, і довіру до такого користувача, відповідно, вище. Blablacar вивчає можливості, які можуть бути впроваджені в Україні. Але навіть зараз деякі користувачі перед поїздкою просять попутників показати паспорт або водійські права.

Найближчим часом в Україні почне працювати система повноцінного бронювання поїздок, а через деякий час буде впроваджена система оплати послуг сервісу, що зі свого боку має на увазі ідентифікацію користувача через платіжні інструменти. Такий принцип вже працює практично у всіх західноєвропейських країнах.

При бронюванні місць в автомобілі попутники зможуть бачити не тільки профіль водія, але і інших попутників, які поїдуть з ними, а значить, зможуть заздалегідь переглянути їх профілі, почитати відгуки.

BlaBlaCar - історія виключно соціальна. Тут немає замовлення послуги, немає комерційної складової, немає заробітку, а є тільки спільні витрати приватних користувачів. На жаль, не всі водії це розуміють. Комерційні водії намагаються використовувати сервіс, займаючись регулярними перевезеннями. Це суперечить принципам сервісу і тягне за собою блокування профілю. Випадки використання сервісу в комерційних цілях зустрічаються і в інших країнах, де працює BlaBlaCar, але, на жаль, в Україні вони поки що проявляються найсильніше.

Поведінка комерційних водіїв має характерні ознаки. По-перше, комерційні водії часто заявляють вартість поїздки, яка істотно нижче або навпаки - вище середньої. Також, якщо під час листування чи телефонної розмови з пасажиром водій каже, що:

- за кермом буде хтось інший;
- поїде тільки тоді, коли збереться достатня кількість пасажирів;
- ціна буде набагато вище заявленої на сайті.

Пасажиру слід повідомити про це адміністраторам сервісу і просто знайти іншого водія.

Усі мажорні та мінорні проблеми системи э адаптованими під нашу країну, система blablacar використовує інші методи ідентифікації у інших країнах, у нас же насамперед усим займається компанія «Подорожник» під грифом роботи Блаблакару, які є гарантом якості, але більшість проблем, які я розглянув у попередніх пунктах, є вимушеними та прив’язаними до нашого регіону. Наша ж ціль виокремити дійсно-вимушені мінуси та невимушені упущення вирішити чи спроектувати іх рішення через офлайн заходи.

1.6 Постановка задачі

Для вирішення задач проектування існує безліч ефективних підходів і дуже важливо максимально уважно та ретельно підійти до вибору архітектури системи, оскільки це визначить подальші перспективи всього проекту. Побудова складних нетривіальних програмних системи є задачею не з простих. Застосування монолітної архітектури має сенс тільки при створенні відносно невеликих та легких систем, якщо ж все таки застосовувати цей підхід для створення великих систем, в кінцевому підсумку з’явиться багато проблем, які буде неможливо вирішити.

В першу чергу, спланує основні процесси нашої кліент-кліент системи з модерацією від адміністратора. Найпростіший перелік функцій, які ми повинні реалізувати в системі:

- регістрація в системі користувача. Приєднання до нього атрибуту водія чи пасажира;
- можливість створення заявки про організацію поїздки. Наявність максимальної кількості упрощень (створення характеристик поїздки);
- можливість залишання відгуків від користувачів. Чітка модерація відгуків – на реальність та реагування на контекст – позитивно чи негативно;
- організація переліку поїздок за вибраними параметрами.

Сервіс повинен складатися наступних компонентів, з відповідними їм вимогами:

а) серверна частина, яка повинна складатися з шару бізнес-логіки, шару доступу до даних і класів моделей;

б) база даних, яка повинна бути реалізовано за допомогою технології, яка дозволяє обробляти великий обсяг даних, при цьому вона також повинна бути гнучкою при зміні моделей даних;

в) webAPI, яке буде надавати функціонал за допомогою http і https запитів;

д) веб-клієнт, який дозволить користувачам мати повний доступ до всіх функцій програми;

е) мобільний клієнт, який буде виконувати набір усіх функцій веб-версії з зручним мобільним інтерфейсом.

UML − мова графічного опису для об'єктного моделювання в області розробки програмного забезпечення. UML є мовою широкого профілю, це відкритий стандарт, який використовує графічні позначення для створення абстрактної моделі системи, яку називають UML-моделлю.

UML дозволяє розробникам програмного забезпечення досягти угоди в графічних позначеннях для представленнязагальних понять і більше сконцентруватися на проектуванні та архітектурі.

Для повного уявлення про програмний продукт було створено декілька UML діаграм. Use Case, Sequence діаграми та Data Flow діаграма. Use Case діаграма − діаграма, що відображає відносини між акторами та прецедентами і є складовою частиною моделі прецедентів, що дозволяє описати систему на концептуальному рівні (рис. 1.14).

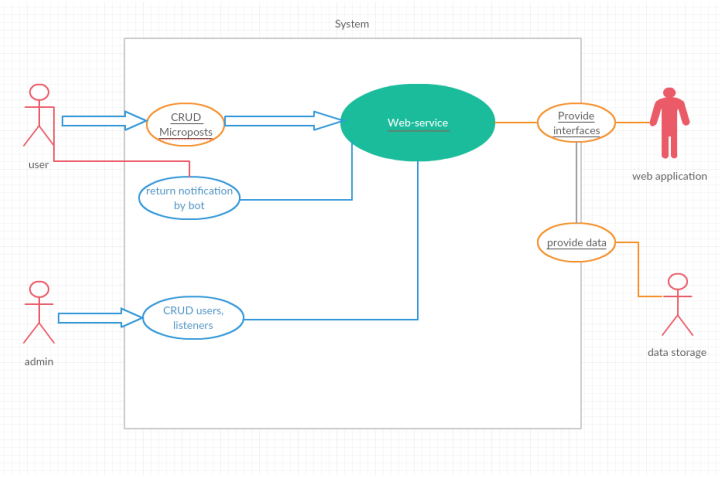

Рисунок 1.14 – Діаграма варіантів використання

З цієї діаграмі видно, що система має 4 ролі. Працівник, веб-додаток,

сховище даних та адміністратор. Користувач клієнта має основні можливості користування додатком. Сервер обробляє запити та створює рекомендації за критеріями. Сховище дозволяє записувати та зчитувати дані для серверу. Адміністратор може лише редагувати деякі атрибути користувачів та свої мікропости.

Sequence діаграма − діаграма, на якій показано взаємодію об'єктів (обмін між ними сигналами і повідомленнями), впорядковане за часом, з відображенням тривалості обробки і послідовності їх прояви. На рисунку 1.15 зображена діаграма послідовності головного алгоритму з рекомендації записів користувачу.

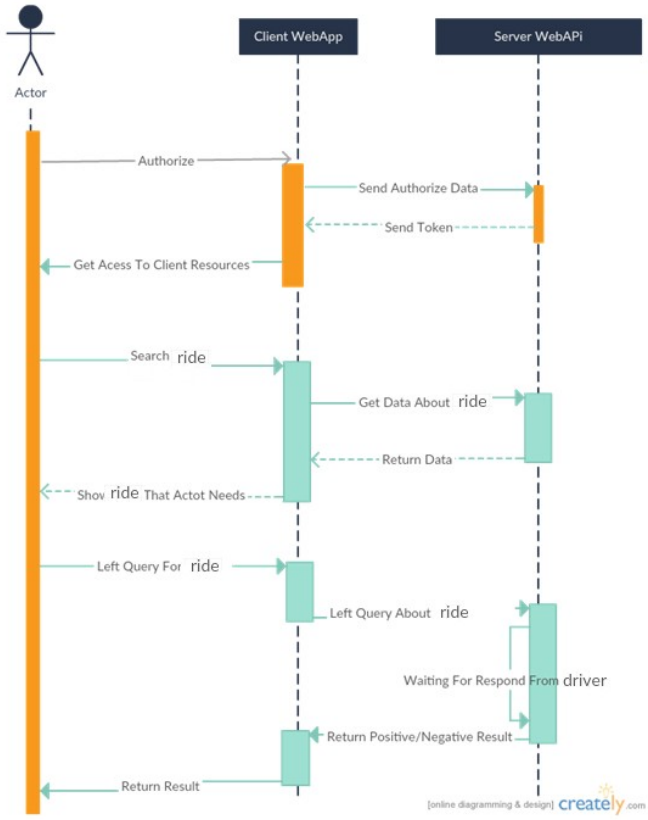

Рисунок 1.15 – Діаграма послідновності

Саме тому ми можемо побудувати діаграму послідовності для найпростіших функцій нашої системи. Користувач отримує дані про існуючі

поїздки, та може залишити заявку про участь у поїздці. До основних об'єктів системи, які беруть участь у взаємодії користувача з системою є:

а) actor (User) - користувач, в цій діаграмі ми розглядаємо користувача, який шукає спільну поїздку чи організовує її;

б) web-Client / Mobile Client - клієнт, за допомогою якого буде відбуватися взаємодія з функціоналом сервісу;

в) API - проміжний рівень між бізнес-логікою і клієнтами програмної системи, що надає можливість відправляти запити в форматі http request і отримувати відповідь. Містить в собі бізнес-логіку, шар доступу до даних і базу даних;

Data flow діаграми − діаграми потоків даних. Так називається методологія графічного структурного аналізу, що описує зовнішні по відношенню до системи джерела і адресати даних, логічні функції, потоки даних і сховища даних, до яких здійснюється доступ.

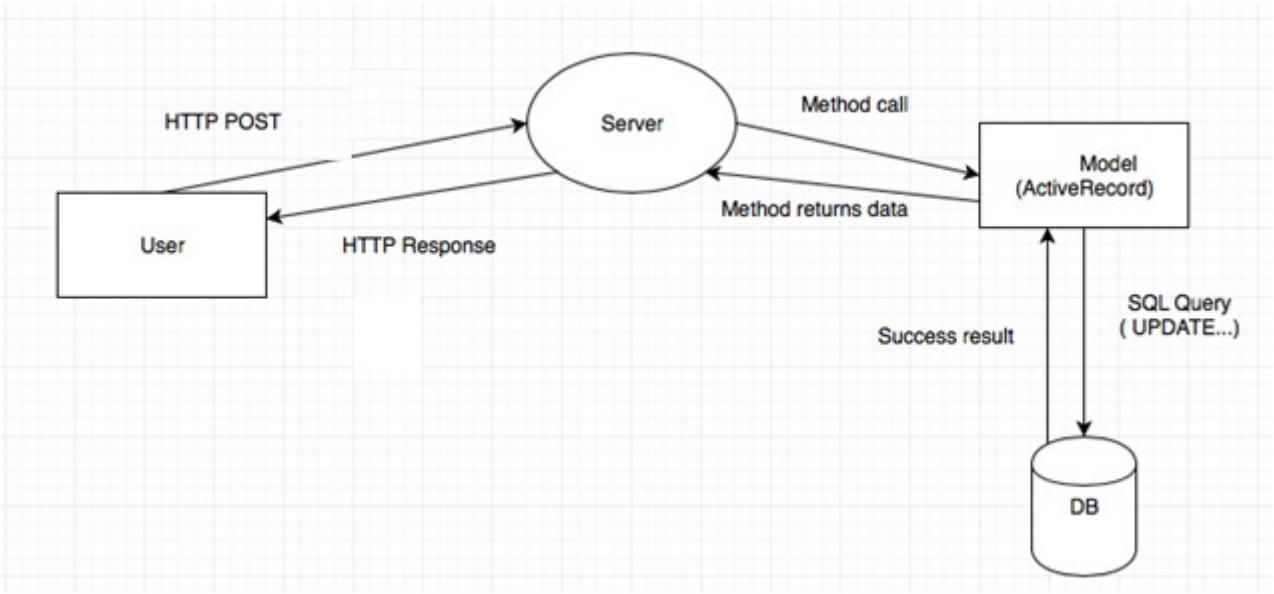

Рисунок 1.16 – Діаграма потоків даних

На рисунку 1.16 зображена діаграма потоків даних - з сесії вилучається унікальний ідентифікатор користувача та за ним знаходяться жанри які він полюбляє, які прослуховував за весь час та записи з тільки улюбленими

жанрами. Сервіс обробляє ці дані та повертає користувачу найбільш цікави для нього записи та плейлісти.

Виходячи з означених вимог до кожного з компонентів можна виділити технології, за допомогою яких необхідно реалізувати кожен з них:

а) серверна частина повинна бути реалізована на платформі .NET (мова програмування C #);

б) в якості бази даних необхідно вибрати технологію, яка сумісна з платформою .NET, і має високі показники швидкості взаємодії - MySQL;

в) webAPI має бути реалізовано за допомогою технології ASP.NET WebAPI на мові програмування C #;

д) веб-клієнт повинен бути створений із застосуванням HTML 5, CSS, JavaScipt, jQuery;

е) мобільний клієнт повинен бути реалізований за допомогою фреймворка Xamarin на мові C #;

ж) розгортання сервісу має відбуватися на сервісі Microsoft Azure.

При створенні дизайну системи рекомендується використання адаптивного веб-дизайну (responsive web design) – дизайн веб-сторінок, що забезпечує оптимальне відображення та взаємодію сайту з користувачем незалежно від роздільної здатності та формату пристрою, з якого здійснюється перегляд сторінки.

Адаптивний веб дизайн будувався за принципом desktop first – розробка починалася з інтерфейсу для стаціонарних ПК і далі адаптувалася до інтерфейсу мобільних телефонів.

В якості шаблону використовувався рухомий макет (layout). Даний шаблон є найбільш адаптивним, оскільки в ньому передбачено наявність декількох контрольних точок для екранів різної ширини. Основною відмінністю цього макета є те, що замість розміщення стовпців один під одним рухається сам контент. Через значні відмінності між основними

контрольними точками доводиться змінювати не тільки загальний макет контенту, але і його елементи.

При розробці веб-сайту використовувався не використовувалося клієнтського CSS-фреймворку, все було зроблено своїми силами за допомогою SASS. Також було використано Javascript-фреймворк jQuery.

1.7 Рекомендації до тестування програмного продукту

Об’єкт тестування: засіб організації спільних поїздок. Вид тестування мобільного додатку: ручне, функціональне, модульне, TDD.

Послідовність проведення робіт: підготовка (Test Preparation), тестування (Testing), аналіз результатів (Test Result Analisys) в розрізі запланованих фаз розробки.

Розробка через тестування (англ тест розробки на основі, TDD.) - техніка розробки програмного забезпечення, яка ґрунтується на повторенні дуже коротких циклів розробки: спочатку пишеться тест, що покриває бажану зміну, потім пишеться код, який дозволить пройти тест, і під кінець проводиться рефакторинг новий код до відповідних стандартів.

Критерієм початку тестування була закінченість розробки необхідного функціоналу модуля.

В процесі тестування використовувалися наступні сценарії функціональних тестів:

а) перевірити коректність роботи обов'язкових полів:

б) переконатися, що робота програми під час запуску / виходу задовольняє основним вимогам;

в) перевірити роботу кожного окремого контроллера, модулю та сценарію;

г) перевірити, чи присутня належна навігація між важливими модулями програми;

д) перевірити наявність повідомлень про помилки в разі некоректної роботи мережі;

е) перевірити, чи здатний додаток повернутися в той стан, в якому він знаходився перед припиненням (наприклад, жорстке перезавантаження або системний збій);

ж) перевірити, що встановлення програми проходить без значних помилок за умови, що пристрій відповідає системним вимогам;

з) переконатися, що автоматичний запуск програми працює коректно;

и) переконатися, що працює керівництво користувача.

2 МОДЕЛЮВАННЯ ПРОГРАМНОГО ПРОДУКТУ

Після достатнього розбору проблем системи та можливостей їх рішення можем перейти до моделювання вже нашого програмного продукту. Ми маємо достатню інформацію для створення об’єктної моделі.

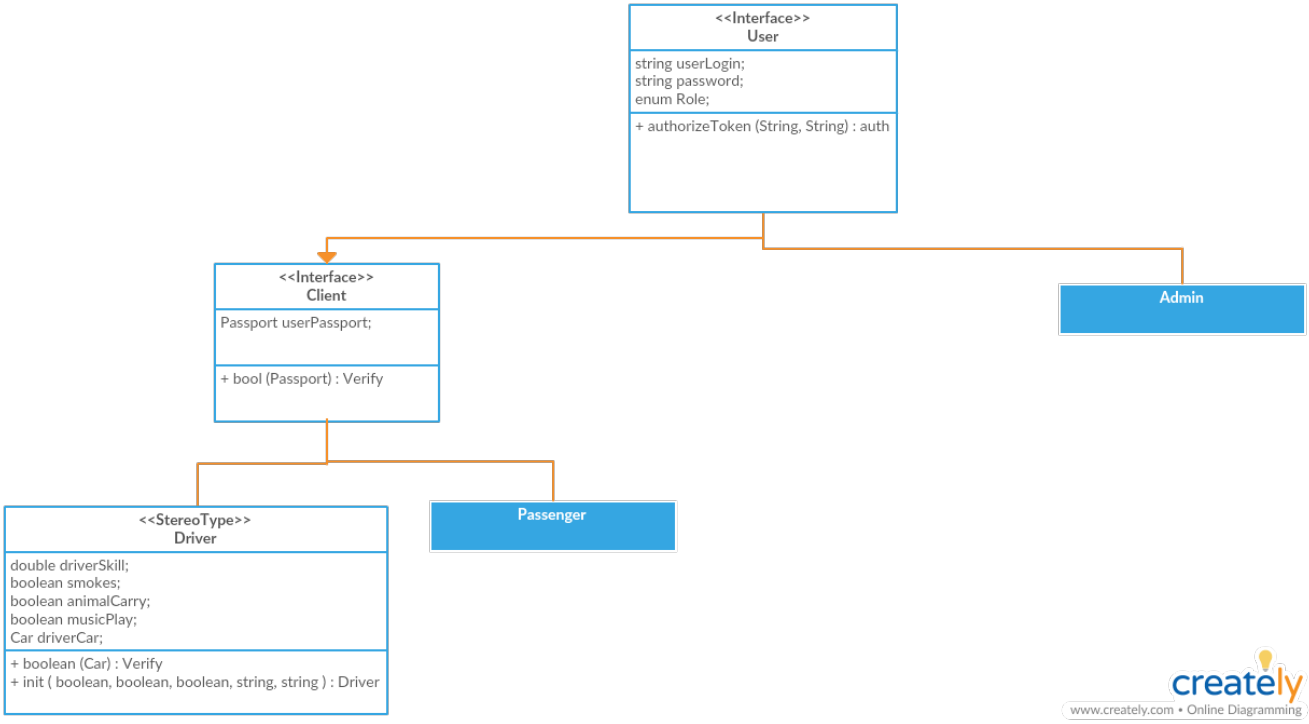

Рисунок 2.1 – Діаграма об’єктів, а саме користувачів нашої системи

На рисунку 2.1 зображена достатня діаграма користувачів нашого сервісу. А саме в нас є три конечні типи користувачів:

- адміністратор. Цей обліковий запис містить в собі дані для авторизації (пароль та логін) та роль, яка надасть йому максимально можливий доступ до адміністративних функцій програми, а саме редагування профілей, підтвердження верифікацій людей та транспортних засобів;
- кліент – пасажир. Найпоширеніша роль системи, та найпростіша в плані збереження даних на сервері. Він похожий на Адміністратора, але в залежності від програмної реалізації ми все одно винесли пасажира до

іншого класу. Пасажиру, як і Водієві потрібно буде підтверджувати свою особистість – за допомогою верифікації паспорта;

- водій. Наймасивніша, в плані даних, сущність системи. Він має характеристики – повний список буде доповнюватись і модернізуватись, але ми вибрали: паління, можливість транспортування тварин та прослуховування музики. Це нестерпні моменти для пасажирів і нерідко саме ці показники впливають на вибір того чи іншого водія та поїздки. Також дуже важливим, саме для України, пунктом є верифікація автівки.

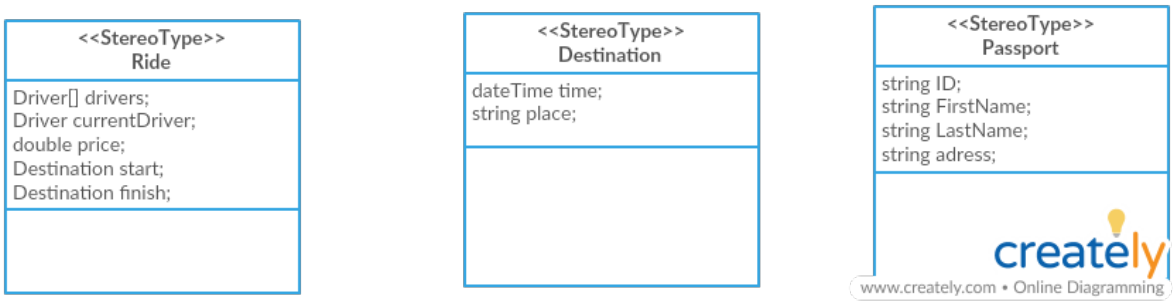

Рисунок 2.2 – Діаграма об’єктів. Вспоміжні класи

На рисунку 2.2 зображені 3 вспоміжних класи, необхідні нашій системі. Класи Ride та Destination ми ще не використовували для моделювання, в них будуть зберігатися дані про поїздки, клас Паспорт – використовували для регістрації користувачів:

- ride – поїздка. По-перше містить в собі поле Водії, на випадок, якщо це повторна поїздка (маршрутне таксі, автобус, тощо) та, звісно, водій для саме цього старту та фінішу. Старт та фініш містять в собі приблизний час відправлення та прибуття, також будуть використовуватись для удобної фільтрації та сортування (щодо геолокації нашого кліента). Залишилось лише поле Ціна – за усі спільні поїздки пасажири повинні платити;
- destination. Клас, розроблений за правилами ООП. А саме для інкапсуляції. Містить в собі час та місце. Місце саме строкою, бо в

залежності від програмної реалізації це може бути адреса, геолокація, координати чи будь який стандартний реалізований тип;

- pasport. Це наш з вами ідентифікаційний документ. На діаграммі зображена мінімальна кількість полів, справді їх буде більше, але ми записали лише найголовнішу інформацію, інша завжди доступна за серію та номером паспорта.

2.1 Функціонал програмного продукту для ролі «Пасажир»

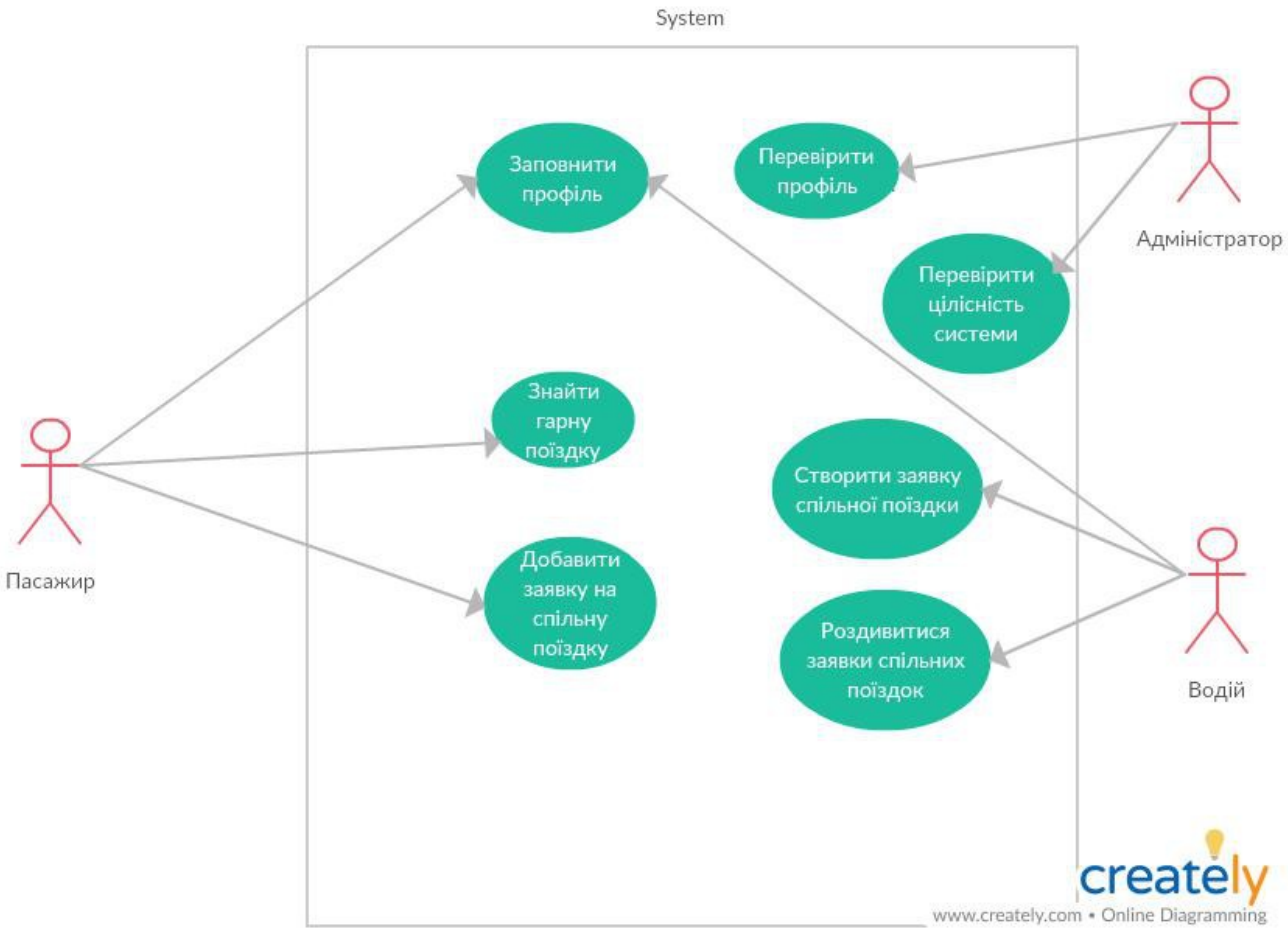

Рисунок 2.3 – Use-Case діаграма нашої системи.

На діаграмі (рисунок 2.3) зображені усі актори нашої системи, а саме Водій, Пасажип, а також десь під капотом є адміністративний розділ, який максимально перевіряє актуальність усіх даних і намагається дотримувати цілістність системи – бо будь які хибні дані, збивають повністю сортування чи фільтрування, і частково чи повністю вбивають friendly-частину та юзабіліті.

2.2 Проектування архітектури програмного продукту

Першим кроком роботи з сервісом є реєстрація користувача або ж його авторизація (в усі наступні рази). Після цього користувач отримає токен - секретний ключ, що ідентифікує будь-якого учасника цього додатка в системі. Алгоритм автоматично виділить його права, відповідно до ролі цього облікового запису, наприклад, не всі можуть додавати об'єкти до бази даних. Після пошуку поїздок веб сервіс допоможе знайти потрібну, і надасть до неї доступ користувачеві негайно. На рисунку 2.4 зображені всі компоненти, з яких складається цільової сервіс. Насамперед варто почати описувати найнижчі рівні програмної системи для зручності сприйняття архітектурного рішення. Компонент DAL (Data Access Layer) - це шар доступу до даних, який описує процес звернення до бази даних: отримання, додавання і редагування зберігається там інформації. Як можна помітити компонент доступу до даний складається з чотирьох модулів:

а) DAL - забезпечує отримання та обробку всіх даних в системі;

б) mobileApp - компонент мобільного додатка, який спілкується з АПІ і знає тільки про сутності, інші відносини - розірвані;

в) webClient - більш наповнений модуль, але суть та ж, що і в мобільному додатку;

г) business layer - відповідає за всю бізнес логіку програми. Тобто шар, що відповідає за виконання будь-яких ланцюжків дій - після видалення, додавання і т.д. Шар, що відповідає за реальне уявлення системи.

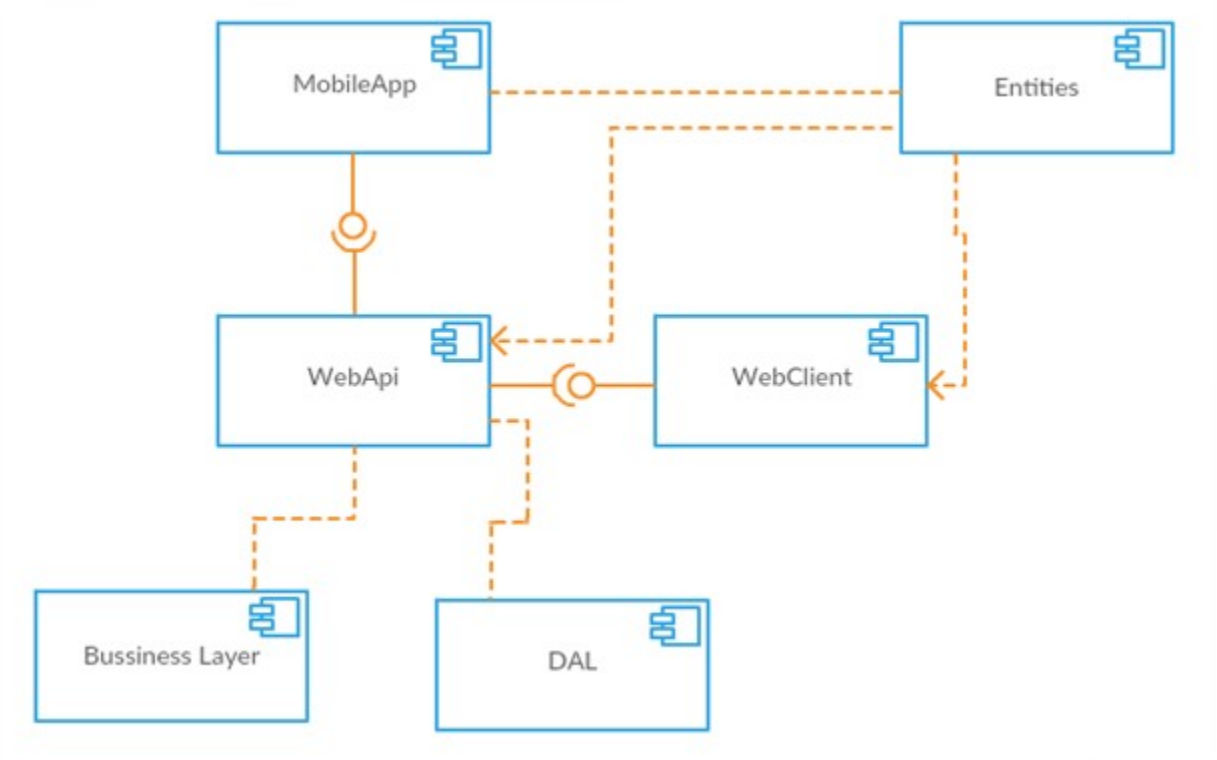

Рисунок 2.4 – Діграма компонентів

Даний компонент базується на понятті аналізу даних. області математики та інформатики, що займається побудовою і дослідженням найбільш загальних математичних методів і обчислювальних алгоритмів вилучення знань з експериментальних (в широкому сенсі) даних процес дослідження, фільтрації, перетворення і моделювання даних з метою отримання корисної інформації та прийняття рішень. Аналіз даних має безліч аспектів і підходів, охоплює різні методи в різних областях науки і діяльності.

У свою чергу, інтелектуальний аналіз даних є особливий метод аналізу даних, який фокусується на моделюванні і відкритті даних, а не на їх описі. Бізнес-аналітика охоплює аналіз даних, який покладається на агрегацію. У

статистичному сенсі деякі поділяють аналіз даних на описову статистику, дослідницький аналіз даних і перевірку статистичних гіпотез. Дослідницький аналіз даних займається відкриттям нових характеристик даних, а перевірка статистичних гіпотез на підтвердження або спростування існуючих гіпотез. Прогнозний аналіз фокусується на застосуванні статистичних або структурних моделей для передбачення або класифікації, а аналіз тексту застосовує статистичні, лінгвістичні і структурні методи для вилучення і класифікації інформації з текстових джерел, що належать до неструктурованих даними. Все це різновиди аналізу даних.

Тепер розглянемо діаграму кооперації, зображену на малюнку 2.5. Діаграма кооперації надає можливість побачити роботу системи у вигляді передачі потоків управління і повідомлень між об'єктами системи.

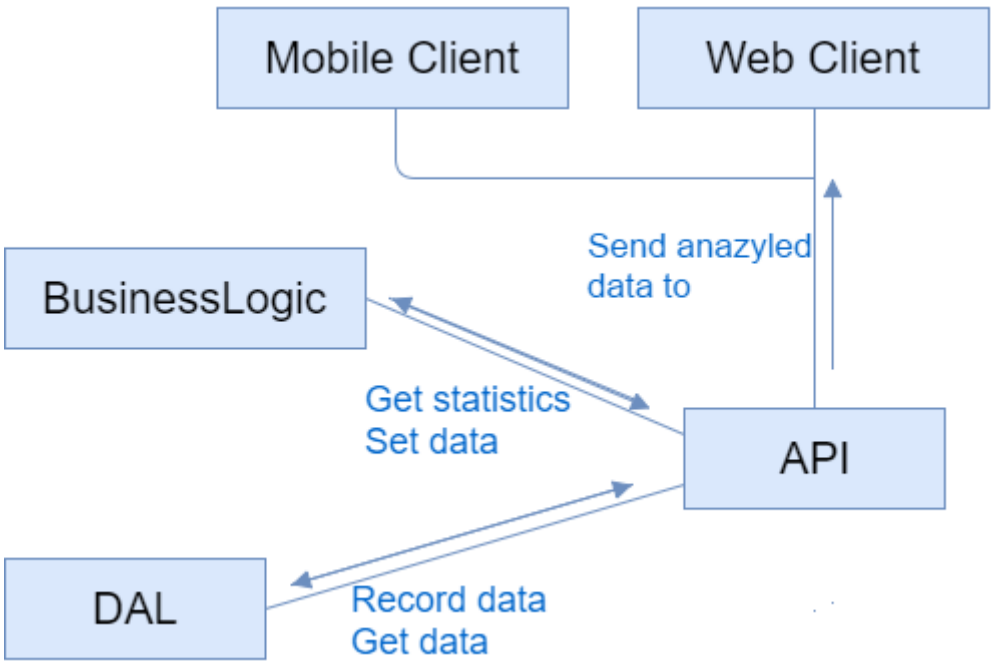

Рисунок 2.5 – Діаграма кооперації

API отримує інформацію з DAL і передає її всім клієнтам в JSON, як і відбувається все кооперація в системі, після того як будь-якої клієнт отримає сирі від користувача в браузері і відправить запит на сервер на отримання даних або на збереження / зміна існуючих сутностей.

Найпростіший спосіб розгортання веб-сайту передбачає копіювання файлів з робочої станції на сервер. Незважаючи на простоту, цей підхід вимагає наявності безпосереднього доступу до сервера. Саме з цієї причини деякі IT-підрозділу і компанії, що надають послуги хостингу, не підтримують цю опцію. Але вона може бути найпростішої при управлінні власним сервером або наявності особливих домовленостей з компанією хостингу.

Вузол пристрої «Веб клієнт» це так само веб ресурс, який ми розмістимо на окремому хостингу. Він потребує основний бібліотеці класів, а про інших учасників ланцюга - йому не відомо, на зразок великого розриву зв'язності. Сервер повинен ховати від клієнта якомога більше деталей своєї реалізації. Клієнту не слід знати про те, яка СУБД використовується на сервері або скільки серверів в даний момент обробляють запити та інші подібні речі. Організація правильного розподілу функцій важлива для масштабування якщо проект почне швидко набирати популярність.

Останній вузол пристрою це клієнтський пристрій - додаток на мобільній операційній системі iOs. Воно буде розміщено на окремих мобільних телефонах, буде так само використовувати основні бібліотеки для роботи, але мати інші функції і, очевидно, іншу структуру.

Таким чином, наш проектуемий проект виглядає наступним чином:

а) серверна частина, яка повинна складатися з шару бізнес-логіки, шару доступу до даних і класів моделей;

б) база даних, яка повинна бути реалізовано за допомогою технології, яка дозволяє обробляти великий обсяг даних, при цьому вона також повинна бути гнучкою при зміні моделей даних;

в) webAPI, яке буде надавати функціонал за допомогою http запитів;

г) бібліотека FlatRent, яка буде спільною для всіх компонентів нашої системи;

д) веб-клієнт, який дозволить користувачам виконувати всі функції, описані раніше в пунктах, і залишатися працездатною при збільшенні

навантаження;

е) мобільний клієнт, який буде виконувати ті ж функції, що і веб-клієнт.

3 АНАЛІЗ РЕПОЗИТОРІЮ БЛАБЛАКАР

3.1 Найвикористованіші мови програмування

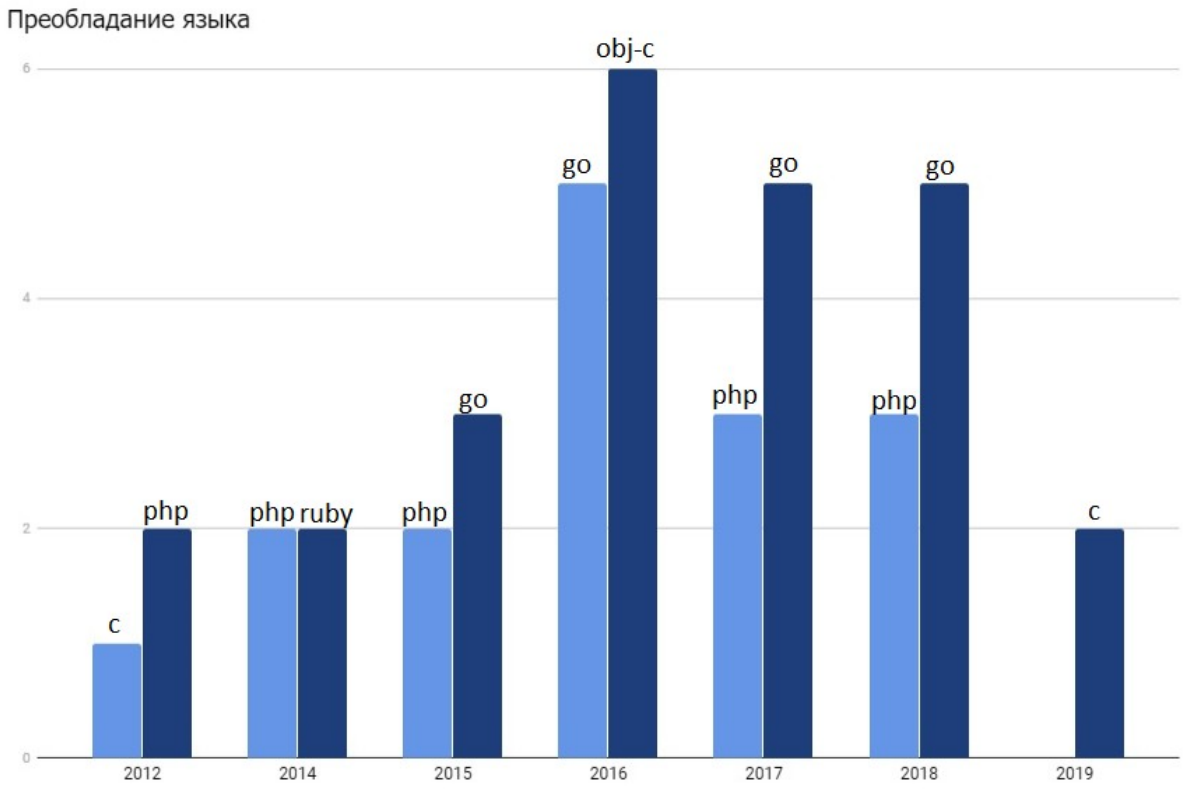

Рисунок 3.1 – Використання мов в репозиторії Блаблакар

5 найбільш використовуваних мов програмування, згідно сховища Blablablacar github:

PHP - 18 репозиторіїв

Перший репозиторій - 17.09.2012

Останній репозиторій - 6.03.2019

Google GO - 15 репозиторіїв

Перший репозиторій - 29.09.2015

Останній репозиторій - 21.02.2019

Java - 6 репозиторіїв

Перший репозиторій - 13.02.2015

Останній репозиторій - 30.11.2018

Objective-C - 5 репозиторіїв

Перший репозиторій - 09.03.2016

Останній репозиторій - 03.08.2017

C - 4 сховища

Перший репозиторій - 08.03.2019

Останній репозиторій - 12.10.2015

Судячи з динаміки переважання мов, репозиторій БлаБлаКара, команда БлаБлаКара і їх же розробка розвивається разом з розвитком мов і сучасними трендами. Незважаючи ні на що, в основі репозиторію закладена велика кількість компонентів на php, що є базисом, основною мовою ресурсу. Після 2015-го року в систему впроваджується Google Go, який приблизно в той час визнається світовим співтовариством і знаходить надійність і стабільність.

Розберемо по черзі всі використовувані мови програмування - виходячи з історії створення та застосування кожного з мов, стає зрозумілим використання тієї чи іншої мови в репозиторії.

3.2 C в репозиторії Блаблакар

Сі придумали інженери. Якщо Паскаль придумав учений, то Сі придумали Керниган і Рітчі - інженери в Bell. У той час на Fortran, COBOL, Algol нічого системного написати було не можна (операційну систему, драйвера). Ці мови призначалися для математичних розрахунків, для бізнес-розрахунків, для всього такого. А все інше писали на Асемблері.

Вони переписали Асемблер, звичайно, навіть розробили фічи, щоб грати на ньому в найпростіші гри. Але це навело їх на думку, що переписувати під нову архітектуру кожен раз - це не дуже розумно.

І вони вирішили написати таку мову високого рівня, який буде підходити для системного програмування, тобто, в якому можна буде управляти пам'яттю, в якому можна буде розуміти де-що лежить і як звертатися до цих шматочках пам'яті. І так з'явилася мова Сі, який зробив величезний вплив на все подальше.

Відповідно, це був основний мову в Unix - операційній системі, яка в той час була ще популярнішим, ніж зараз. При цьому деякі архітектурні речі там робляться досить складно - знову ж таки, як і в Асемблері, нам доводиться весь час стежити, де ми, чого і яку пам'ять виділили; весь час «тече» кудись ця пам'ять - тобто ми виділили, забули видалити, видалили не те, вилізли за межі пам'яті.

3.3 PHP в Blablacar

PHP з'явився "випадково". - програміст написав набір макросів для Перла, які були схожі на Сі. І назвав це Personal HomePage.

Загалом, в результаті цей PHP став жити і став згодом набагато популярніше, ніж Perl. Сам набір макросів для Перла вийшов дивний. Тобто він розвивався сам по собі, його ніхто не проектував, ніхто не администрировал процес розвитку (ні компанія, ні якийсь чоловік), а було багато груп, кожна з яких дороблялі те, що їм подобається. В результаті там функції називаються по-різному, навіть стилю немає, все через підкреслення, досить хаотично; настройки лежать тут і там, і як все це буде працювати не дуже зрозуміло.

3.4 Golang в Blablacar

Официально язык был представлен в ноябре 2009 года. На данный момент поддержка официального компилятора, разрабатываемого создателями языка, осуществляется для всех популярных операционных систем. Также Go поддерживается набором компиляторов gcc, существует несколько независимых реализаций. Ведётся разработка второй версии языка.

Язык Go разрабатывался как язык программирования для создания высокоэффективных программ, работающих на современных распределённых системах и многоядерных процессорах. Он может рассматриваться как попытка создать замену языкам C и C++. По словам Роба Пайка, «Go был разработан для решения реальных проблем, возникающих при разработке программного обеспечения в Google». В якості основних таких проблем він називає:

- повільну збірку програм;
- неконтрольовані залежності;
- використання різними програмістами різних підмножин мови;
- труднощі з розумінням програм;
- дублювання розробок;
- високу вартість оновлень;
- несинхронні поновлення при дублюванні коду;
- складність розробки інструментарію;
- проблеми міжмовної взаємодії.

Go створювався в розрахунку на те, що програми на ньому будуть транслюватися в об'єктний код цільової апаратної і програмної платформи і надалі виконуватися безпосередньо, не вимагаючи віртуальної машини, тому одним з критеріїв вибору архітектурних рішень була можливість забезпечити

швидку компіляцію в ефективний об'єктний код і відсутність надмірних вимог до динамічної підтримки.

3.5 Внесок кожної мови в розробку

Всі мови дуже органічно інтегровані до складу сховища та кожен модуль відповідає за свою специфіку.

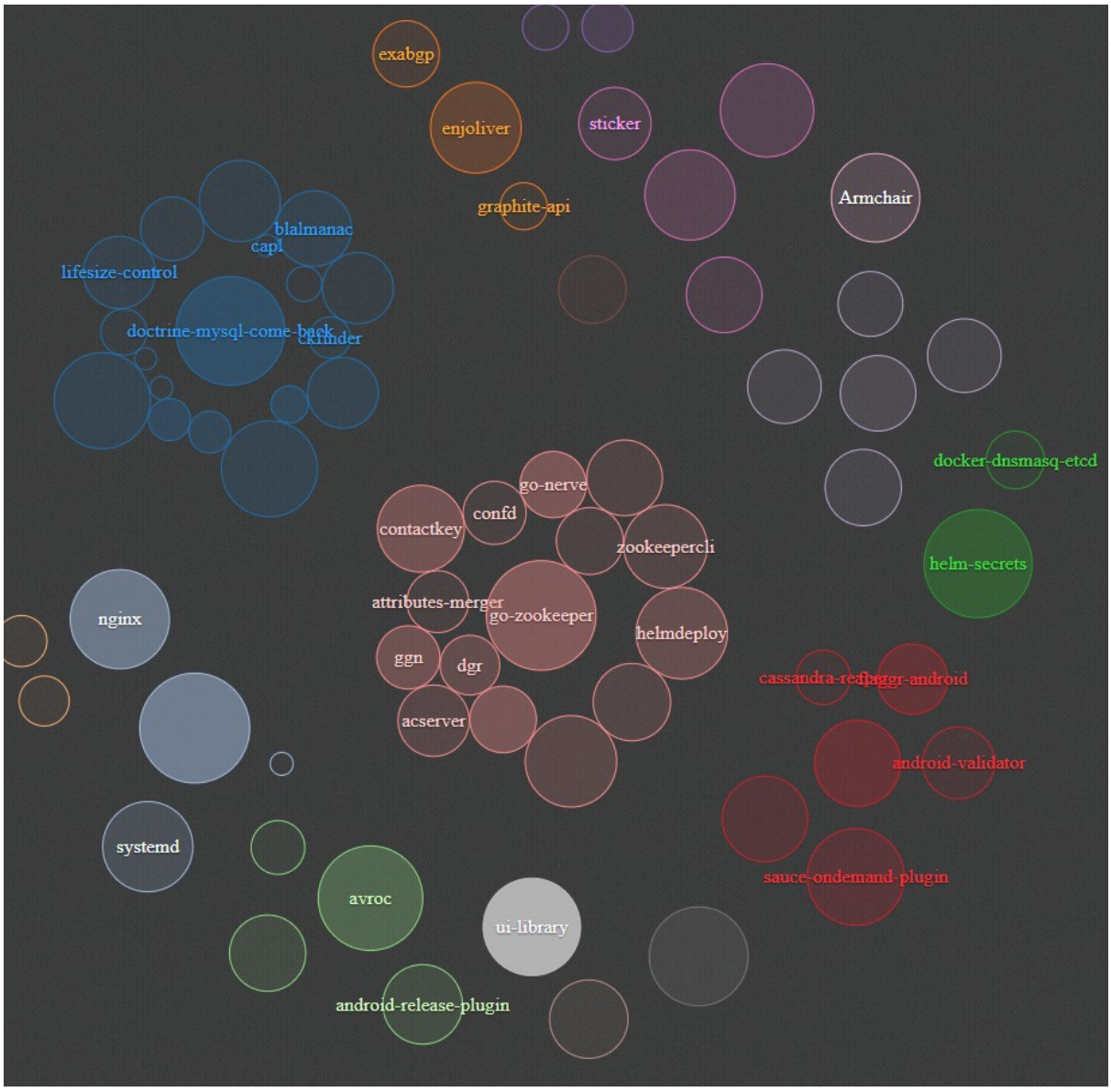

Рисунок 3.2 – Мовний розподіл у репозиторії

Так само візуалізація модулів сховища показує які мови реалізують один глобальний модуль, які - реалізують безліч допоміжних компонентів для системи - на кшталт enjoliver, android-release plugin, avroc.

3.6 Метрики якості коду в Blablacar

З блогу блаблакар - https://medium.com/blablacar-tech/data-metrology-at-blablacar-8c068c022489, команда впровадила просту метрику якості коду - простий запит з динамічним параметром, що дозволяє обчислити значення в одному циклі Python.

metrics_name	metrics_day	day_of_week	value	average_same_day	delta
eventlogs_push_notification_interact	2018-06-13	Wednesday	247 634	206 620,89	1,2
eventlogs_push_notification_interact	2018-06-12	Tuesday	244 259	199 879,11	1,22
eventlogs_push_notification_interact	2018-06-11	Monday	238 628	202 980,11	1,18
eventlogs_push_notification_interact	2018-06-10	Sunday	248 517	233 825,75	1,06
eventlogs_push_notification_interact	2018-06-09	Saturday	251 856	209 868,5	1,2
eventlogs_push_notification_interact	2018-06-08	Friday	278 619	258 257,5	1,08
eventlogs_push_notification_interact	2018-06-07	Thursday	244 779	232 465,88	1,05
eventlogs_push_notification_interact	2018-06-06	Wednesday	201 580	206 620,89	0,98
eventlogs_push_notification_interact	2018-06-05	Tuesday	191 101	199 879,11	0,96
eventlogs_push_notification_interact	2018-06-04	Monday	191 308	202 980,11	0,94
eventlogs_push_notification_interact	2018-06-03	Sunday	218 970	233 825,75	0,94
eventlogs_push_notification_interact	2018-06-02	Saturday	186 328	209 868,5	0,89
eventlogs_push_notification_interact	2018-06-01	Friday	232 095	258 257,5	0,9
eventlogs_push_notification_interact	2018-05-31	Thursday	214 103	232 465,88	0,92
eventlogs_push_notification_interact	2018-05-30	Wednesday	198 499	206 620,89	0,96
eventlogs_push_notification_interact	2018-05-29	Tuesday	183 115	199 879,11	0,92
eventlogs_push_notification_interact	2018-05-28	Monday	191 330	202 980,11	0,94
eventlogs_push_notification_interact	2018-05-27	Sunday	223 511	233 825,75	0,96
eventlogs_push_notification_interact	2018-05-26	Saturday	182 321	209 868,5	0,87

Рисунок 3.3 – Результати метричних обчислювань у репозиторії Блаблакар

Підсумкове значення, як на рисунку 3.3, не вказує точну або детальну оцінку якості коду, але простим значенням поверхнево визначає якість певного коммітов github за той чи інший робочий день.

```
INSERT INTO metrology_history
SELECT
    '${metrics_name}' as metrics_name,
    vm.metrics_day,
    vm.metrics_value,
    avg(vm.metrics_value) over(partition by dayOfWeek(vm.metrics_day) as average_same_day,
    vm.metrics_value / avg(vm.metrics_value) over(partition by dayOfWeek(vm.metrics_day) as delta
FROM
    v_metrics_'${metrics_name}' vm
WHERE
    vm.metrics_day >= getutcdate() - interval '2 months'
;
```

Рисунок 3.4 – Той самий код, вичислюючий єфективність роботи

Згідно з результатами метрики, співробітник компанії Thomas Tygreat зауважив підвищення динаміки якості написанні коду за відрізок в кілька тижнів. Перший період значення метрики збільшувалася з кожним днем, але було в середньому 200 000, через тиждень це значення перевалило за 215 000 в середньому - хоча мова йде про команду дійсно кваліфікованих професіоналів і поліпшення якості коду на 7.5% це величезний результат, якого BlaBlaCar зміг добітсья за допомогою простого циклу на Python.

ВИСНОВКИ

В даній роботі було змоделювано утопічно-ідеальний сервіс спільних поїздок, спроектували його на заміщенні нереального в моделі на актуальні та можливі засоби безпеки.

Також було розглянуто ключові компоненти для побудови мікросервісних систем, які утворюють інфраструктурний рівень та надають необхідну гнучкість всій системі. До даних компонентів відносяться: сервіс єдиного входу, сервіс відкриття, балансувальних навантаження, автоматичний вимикач.

Одну з головних недоліків даної системи БлаБлаКару ми підмітили недостатньу верифікацію даних та мультиаккаунтинг. Ми дішли висновку, що відслідження IP-адрес чи фізичних адресів телефону, зможе викоринити такі проблеми, але звичайно підіб’є популярність нашого програмного продукту.

Підсумовуючи, можна стверджувати, що в нас дійсно є можливість покращити існуючі сервіси спільних поїздок. Безумовно, реалізація проекту за нашим планом на короткій дистанції буде мати куди меншу монетизацію та необхідність відкриття багатьох технічних та верифікаційних центрів, але ми зменьшимо відсоток поганих ситуацій та завоюємо кредит довіри у зовсім нових пластів аудиторії, ті які раніше цуралися використання сервісу BlaBlaCar та інших менш популярних аналогів. Ми зробили достатньу роботу для переходу до проектування та реалізації нашого програмного продукту.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. М. Фаулер. Архитектура корпоративных программных приложений. – Издательский дом Вильямс, 2006 – 544 с.

2. Ньюмен С. Создание микросервисов. – СПб.: Питер, 2016 – 304 с.

3. Chris Richardson. From Design to Deployment. - Floyd Smith, 2016. – 74 с.

4. Gregor Hohpe, Bobby Woolf. Enterprise Integration Patterns: Designing, Building, and Deploying Messaging Solutions. – Addison-Wesley, 2004 – 736 с.

5. E. Evans. Domain-Driven Design: Tackling Complexity in the Heart of Software. – Addison-Wesley, 2003 – 560 с.

6. Martin Fowler – Microservices. URL: http://martinfowler.com/ articles/microservices.html (дата звернення: 05.11.2018).

7. I. Nadareishvili. Microservice Architecture: Aligning Principles, Practices, and Culture. – O'Reilly Media, 2016 – 146 с.

8. Introduction to microservices. URL: https://nginx.com/blog/introduction-to-microservices (дата звернення: 29.10.2018).

9. Using an API Gateway. URL: https://nginx.com/blog/buildingmicroservices-using-an-api-gateway (дата звернення: 27.10.2018).

10. Service Discovery. URL: https://nginx.com/blog/service-discovery-ina-microservices-architecture (дата звернення: 05.11.2018).

11. Офіційний сайт Docker. URL: https://docker.com (дата звернення: 15.10.2018).