Міністерство освіти і науки України

Державний університет «Одеська політехніка»

Інститут комп’ютерних систем  
Кафедра інформаційних систем

КУРСОВА РОБОТА

з дисципліни «Технології створення програмних продуктів»

за темою

«Gator»

Пояснювальна записка до етапів визначення вимог до програмного продукту та планування процесів розробки програмного продукту

Виконала:

студентка 3-го курсу

групи ЛАІ-195

Шакун Таїсія

Перевірив:

Блажко О. А.

Одеса-2021

# Анотація

В курсовій роботі розглядається процес створення програмного продукту «Gator» на етапах визначення вимог до програмного продукту та планування процесів розробки.

Робота виконувалась в команді з декількох учасників: Шакун Таїсія, Нікітін Павло

В робота пов`язана з такими матеріальними потребами споживача як Безпека . Аналіз вказаних потреб визначив інформаційну потребу - в данному додатку для агрегування варіантів шляху до місця призначення.

При визначені ступеня готовності існуючих програмних продуктів до вирішення інформаційної потреби проаналізовано наступні програмні продукти: Google Maps,  838, Transport Odessa

Поточну версію пояснювальної записки до результатів роботи розміщено на GitHub-репозиторії за адресою: <https://github.com/TaisiaShakun/Gator>

Та сам проект під назвою Gator v2

# Перелік скорочень

ОС – операційна система

ІС – інформаційна система

БД – база даних

СКБД – система керування базами даних

ПЗ – програмне забезпечення

ПП– програмний продукт

UML – уніфікована мова моделювання

Зміст

[Анотація 2](#_Toc89968902)

[Перелік скорочень 3](#_Toc89968903)

[1 Вимоги до програмного продукту 6](#_Toc89968904)

[1.1 Визначення потреб споживача 6](#_Toc89968905)

[1.1.1 Ієрархія потреб споживача 6](#_Toc89968906)

[1.1.2 Деталізація матеріальної потреби 7](#_Toc89968907)

[1.2 Бізнес-вимоги до програмного продукту 7](#_Toc89968908)

[1.2.1 Опис проблеми споживача 7](#_Toc89968909)

[1.2.2 Мета створення програмного продукту 8](#_Toc89968910)

[1.2.3 Назва програмного продукту 8](#_Toc89968911)

[1.3 Вимоги користувача до програмного продукту 8](#_Toc89968912)

[1.3.2 Історія користувача програмного продукту 8](#_Toc89968913)

[1.3.3 Діаграма прецедентів програмного продукту 9](#_Toc89968914)

[1.3.4 Сценаріїв використання прецедентів програмного продукту 9](#_Toc89968915)

[1.4 Функціональні вимоги до програмного продукту 10](#_Toc89968916)

[1.4.1. Багаторівнева класифікація функціональних вимог 10](#_Toc89968917)

[1.4.2 Функціональний аналіз існуючих програмних продуктів 12](#_Toc89968918)

[1.5 Нефункціональні вимоги до програмного продукту 13](#_Toc89968919)

[1.5.1 Опис зовнішніх інтерфейсів 13](#_Toc89968920)

[2 Планування процесу розробки програмного продукту 20](#_Toc89968921)

[2.1 Планування ітерацій розробки програмного продукту 20](#_Toc89968922)

[2.2 Концептуальний опис архітектури програмного продукту 21](#_Toc89968923)

[2.3 План розробки програмного продукту 22](#_Toc89968924)

[2.3.1 Оцінка трудомісткості розробки програмного продукту 22](#_Toc89968925)

[2.3.2 Визначення дерева робіт з розробки програмного продукту 24](#_Toc89968926)

[2.3.3 Графік робіт з розробки програмного продукту 25](#_Toc89968927)

[3 Проектування програмного продукту 27](#_Toc89968928)

[3.1 Концептуальне проектування на основі UML-діаграми концептуальни класів 27](#_Toc89968929)

[3.2 Проектування програмних класів 28](#_Toc89968930)

[3.3 Проектування алгоритмів роботи методів програмних класів 30](#_Toc89968931)

[3.4.1 Проектування тестових сценаріїв верифікації функціональних вимог 33](#_Toc89968932)

[3.4.2 Проектування тестових сценаріїв верифікації нефункціональних вимог 35](#_Toc89968933)

[3.4.3 Створення матриці відповідності вимог до програмного продукту 36](#_Toc89968934)

[4 Конструювання програмного продукту 37](#_Toc89968935)

[4.1 Особливості конструювання програмних модулів 37](#_Toc89968936)

[4.1.1 Конструювання програмної структури з урахуванням спеціалізованого Framework для FrontEnd-компонент архітектури (за наявністю) 37](#_Toc89968937)

[4.1.2 Конструювання програмних класів (за наявністю об`єктно-орієнтованого програмування) 38](#_Toc89968938)

[4.1.3 Конструювання алгоритмів методів програмних класів або процедур/функцій 39](#_Toc89968939)

[4.1.4 Особливості використання спеціалізованих програмних бібліотек та API (за наявністю) 40](#_Toc89968940)

[4.2 Модульне тестування програмних модулів 40](#_Toc89968941)

[5 Верифікація програмного продукту 42](#_Toc89968942)

[5.1 Тестування апаратно-програмних інтерфейсів програмного продукту 42](#_Toc89968943)

[5.2 Тестування інтерфейсу користувача програмного продукту 43](#_Toc89968944)

[5.3 Тестування часу реакції програмного продукту на дії користувача 43](#_Toc89968945)

[6 Розгортання та валідація програмного продукту 46](#_Toc89968946)

[6.1 Інструкція з встановлення системного програмного забезпечення 46](#_Toc89968947)

[6.2 Інструкція з використання програмного продукту 47](#_Toc89968948)

[6.3 Результати валідації програмного продукту 47](#_Toc89968949)

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

# 1 Вимоги до програмного продукту

## 1.1 Визначення потреб споживача

### 1.1.1 Ієрархія потреб споживача

Відомо, що в теорії маркетингу потреби людини можуть бути представлені у вигляді ієрархії потреб ідей американського психолога Абрахама Маслоу включають рівні:

* фізіологія (вода, їжа, житло, сон);
* безпека (особиста, здоров'я, стабільність),
* приналежність (спілкування, дружба, любов),
* визнання (повага оточуючих, самооцінка),
* самовираження (вдосконалення, персональний розвиток).

На рисунку 1.1 представлено одну ієрархію потреби споживача, яку хотілося б задовольнити, використовуючи майбутній програмний продукт.

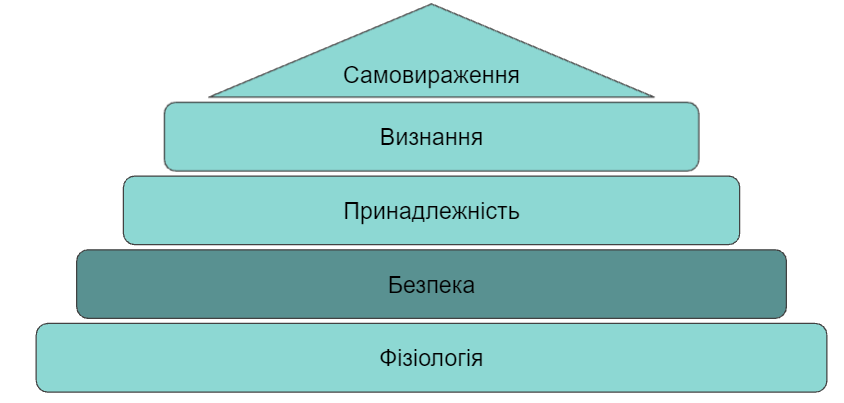


Рис. 1.1 – Приклад ієрархії потреби споживача

### 1.1.2 Деталізація матеріальної потреби



## 1.2 Бізнес-вимоги до програмного продукту

### 1.2.1 Опис проблеми споживача

#### 1.2.1.1 Концептуальний опис проблеми споживача

Загальна  проблема: Постійні запізнення на зустріч, пари, роботу в зв’язку з вибором не оптимального варіанту для досягнення місця призначення з показниками: часу, вартості, та комфорту.

#### 1.2.1.2 Опис цільової групи споживачів

Цільовою група споживача це люди що переміщуються по місту, і бажають це робити в найкоротший час

#### 1.2.1.3 Метричний опис проблеми споживача

Метричні показники: Відвідуваність пар, запізнень на роботу та зустріч.

Данний рівень показника проблем визначуєтся:

M-запізнення  
N-кількість переміщень з місця на місце  
R-result  
R  = M/N

### 1.2.2 Мета створення програмного продукту

#### 1.2.2.1 Проблемний аналіз існуючих програмних продуктів

#### 1.2.2.2 Мета створення програмного продукту

Створити додаток в якому можно буде преглянути усі варіанти  перемішення до місця призначення

### 1.2.3 Назва програмного продукту

Гасло програмного продукту:

Gator – зручний маршрут для вас швидше ніж ні де

Логотип програмного продукту

Изображение выглядит как текст, коллекция картинок

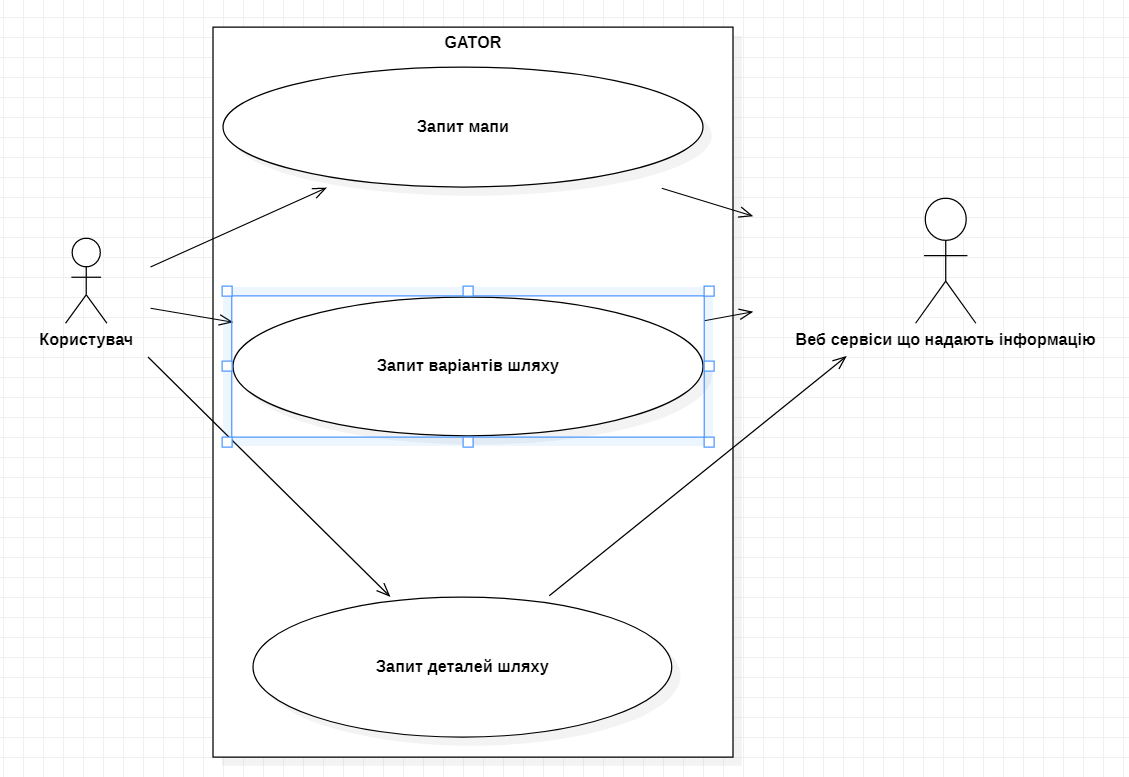
Автоматически созданное описание

## 1.3 Вимоги користувача до програмного продукту

### 1.3.2 Історія користувача програмного продукту

* Користувач отримує мапу міста
* Користувач вказує місця відправлення та прибуття
* Користувач запитує варіанти шляху
* Користувач ознайомлюється з варіантами шляху
* Користувач запитує деталі вибраного їм шляху

### 1.3.3 Діаграма прецедентів програмного продукту



### 1.3.4 Сценаріїв використання прецедентів програмного продукту

Назва прецеденту: Запит мапи

Передумови: Користувач відкриває додаток

Актори як зацікавлені особи у виконанні прецеденту: Користувач

Актор-основна зацікавлена особа як ініціатор початку прецеденту: Користувач

Гарантії успіху: Користувач отримує дійсну та робочу карту міста на даний момент

|  |  |
| --- | --- |
| основний успішний сценарій: | альтернативный сценарій: |
| 1. Користувач відкриває додаток 2. ПП пропонує мапу міста 3. Користувач розглядає мапу | 2. ПП не відкриває мапу з урахуванням можливої помилки(відсутність інтернету)  3. Користувач розглядає помилку. |
| 1. Користувач обирає місця відправки та прибуття на мапі 2. Користувач запитує варіанти шляху за обраним шляхом 3. Додаток надає інформацію веб сервісам 4. Веб сервіси надають інформацію по можливим шляхами 5. Додаток поєднує всі шляхи в одну таблицю 6. Додаток повертає таблицю користувачеві | 3.1 Веб сервіс відправив помилку.  4.1 Додаток не вносить варіант цього шляху до списку |

## 1.4 Функціональні вимоги до програмного продукту

### 1.4.1. Багаторівнева класифікація функціональних вимог

|  |  |
| --- | --- |
| Ідентифікатор функції  (назва) | Назва функції |
| FR1 | Запуск ПП |
| FR1.1 | Успішне завантаження ПП |
| FR1.2 | Відсутність інтернету |
| FR1.3 | Помилка сервера при завантаженні |
| FR2 | Отримання мапи |
| FR2.1 | Успішне отримання мапи |
| FR2.2 | Помилка сервісу, який надає мапу |

|  |  |
| --- | --- |
| Ідентифікатор функції  (назва) | Назва функції |
| FR3 | Введення координат |
| FR3.1 | Введення точки старту |
| FR3.1.1 | Успішне введення точки |
| FR3.1.2 | Точка відсутня на мапі |
| FR3.2 | Введення кінцевої точки |
| FR3.2.1 | Успішне введення точки |
| FR3.2.2 | Точка відсутня на мапі |
| FR4 | Отримання варіантів руху |
| FR4.1 | Успішне отримання ≥ 1 варіанту |
| FR4.2 | Помилка сервісів, які надають інформацію про транспортні засоби |

### 1.4.2 Функціональний аналіз існуючих програмних продуктів

|  |  |  |  |
| --- | --- | --- | --- |
| Ідентифікатор функції  (назва) | Google Maps | 838 | Транспорт Одеси |
| FR1.1 | + | + | - |
| FR1.2 | + | - | + |
| FR1.3 | + | - | + |
| FR2.1 | + | + | + |
| FR2.2 | + | + | + |
| FR3.1.1 | + | + | + |
| FR3.1.2 | + | + | + |
| FR3.2.1 | + | + | + |
| FR3.2.2 | + | + | + |
| FR4.1 | + | + | + |
| FR4.2 | + | + | + |

## 1.5 Нефункціональні вимоги до програмного продукту

### 1.5.1 Опис зовнішніх інтерфейсів

#### 1.5.1.1 Опис інтерфейсів користувача

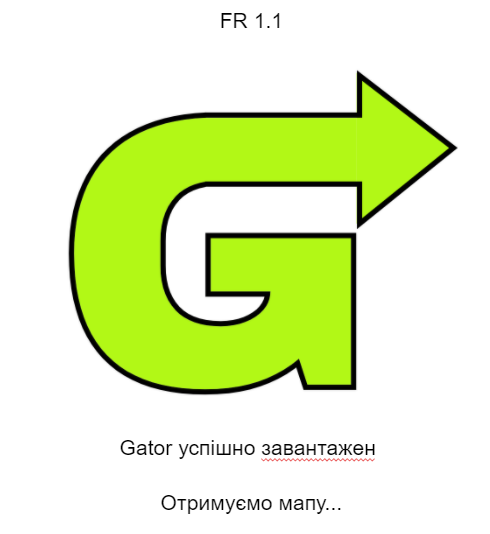
Опис INPUT-інтерфейсів користувача

|  |  |
| --- | --- |
| Ідентифікатор  функції  (назва) | Засіб INPUT-потоку |
| FR1.1 | * стандартна комп`ютерна клавіатура; * 2/3-кнопочний маніпулятор типу "миша"; * сенсорний екран (Touchscreen, Touchpad, Multi-touch); |
| FR3.1.1 | * стандартна комп`ютерна клавіатура; * 2/3-кнопочний маніпулятор типу "миша"; * сенсорний екран (Touchscreen, Touchpad, Multi-touch); |
| FR3.2.1 | * стандартна комп`ютерна клавіатура; * 2/3-кнопочний маніпулятор типу "миша"; * сенсорний екран (Touchscreen, Touchpad, Multi-touch); |

Опис OUTPUT-інтерфейсів користувача

|  |  |
| --- | --- |
| Ідентифікатор функції (назва) | Засіб OUTPUT - потоку |
| FR1.1 | графічний інтерфейс |
| FR1.2 | графічний інтерфейс |
| FR1.3 | графічний інтерфейс |
| FR2.1 | графічний інтерфейс |
| FR2.2 | графічний інтерфейс |
| FR3.1.1 | графічний інтерфейс |
| FR3.1.2 | графічний інтерфейс |
| FR3.2.1 | графічний інтерфейс |
| FR3.2.2 | графічний інтерфейс |
| FR4.1 | графічний інтерфейс |
| FR4.2 | графічний інтерфейс |

Приклади роботи деяких функцій



Изображение выглядит как текст

Автоматически созданное описание



Изображение выглядит как карта

Автоматически созданное описание

#### 1.5.1.2 Опис інтерфейсу із зовнішніми пристроями

|  |  |
| --- | --- |
| Ідентифікатор функції  (назва) | Зовнішній пристрій |
| FR1.1 | Смартфон / Notebook / Desktop-персональний комп`ютер |
| FR1.2 | Смартфон / Notebook / Desktop-персональний комп`ютер |
| FR1.3 | Смартфон / Notebook / Desktop-персональний комп`ютер |
| FR2.1 | Смартфон / Notebook / Desktop-персональний комп`ютер |
| FR2.2 | Смартфон / Notebook / Desktop-персональний комп`ютер |
| FR3.1.1 | Смартфон / Notebook / Desktop-персональний комп`ютер |
| FR3.1.2 | Смартфон / Notebook / Desktop-персональний комп`ютер |
| FR3.2.1 | Смартфон / Notebook / Desktop-персональний комп`ютер |
| FR3.2.2 | Смартфон / Notebook / Desktop-персональний комп`ютер |
| FR4.1 | Смартфон / Notebook / Desktop-персональний комп`ютер |
| FR4.2 | Смартфон / Notebook / Desktop-персональний комп`ютер |

#### 1.5.1.3 Опис програмних інтерфейсів

* ОС
  + ПП буде створенний як веб-додаток, тому ОС не важлива
* API
  + MapBox API
  + API сервісів, які надають інформацію про транспортні засоби

#### 1.5.1.4 Опис інтерфейсів передачі інформації

Для працездатності ПП потребує лише інтерфейси передачі даних за допомогою інтернету

* Ethernet / Wi-Fi

#### 1.5.1.5 Опис атрибутів продуктивності

|  |  |
| --- | --- |
| Ідентифікатор функції (назва) | максимальний час реакції ПП на дії користувачів, секунди |
| FR1.1 | 20 |
| FR1.2 | 5 |
| FR1.3 | 20 |
| FR2.1 | 10 |
| FR2.2 | 10 |
| FR3.1.1 | 3 |
| FR3.1.2 | 3 |
| FR3.2.1 | 3 |
| FR3.2.2 | 3 |
| FR4.1 | 30 |
| FR4.2 | 30 |

Максимальна кілкість одночасно обслуговуваних користувачів залежить від обраного фізичного серверу та реалізації серверу. ПП може обслуговувати мільйони одночасних користувачів (Google) та зберігати свою працездатність.

# 2 Планування процесу розробки програмного продукту

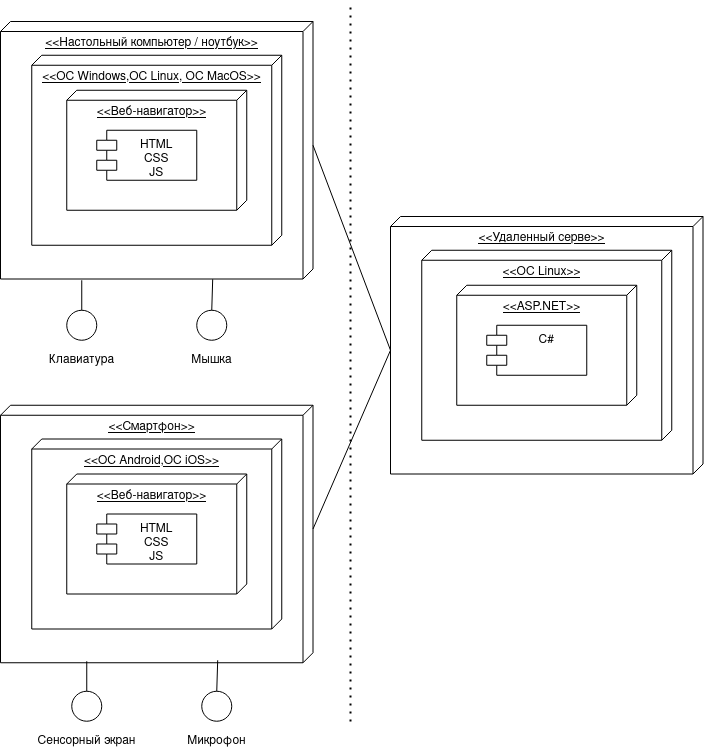
## 2.1 Планування ітерацій розробки програмного продукту

З метою забезпечення вимог таких рекомендацій IEEE-стандарту, як необхідність, корисність при експлуатації, здійсненність функціональних вимог до ПП, визначено функціональні пріоритети, які будуть використані при плануванні ітерацій розробки ПП. Результати представлено в таблиці 2.1

Таблиця 2.1 – приклад опису функцій з наданням унікальних ієрархічних ідентифікаторів

|  |  |
| --- | --- |
| Ідентифікатор  функції | Назва функції |
| FR1 | Запуск ПП |
| FR1.1 | Успішне завантаження ПП |
| FR1.2 | Відсутність інтернету |
| FR1.3 | Помилка сервера при завантаженні |
| FR2 | Отримання мапи |
| FR2.1 | Успішне отримання мапи |
| FR2.2 | Помилка сервісу, який надає мапу |
| FR3 | Введення координат |
| FR3.1 | Введення точки мишкою |
| FR3.1.1 | Успішне введення точки |
| FR3.1.2 | Точка відсутня на мапі |
| FR3.2 | Введення точки клавіатурою |
| FR4 | Отримання варіантів руху |
| FR4.1 | Успішне отримання ≥ 1 варіанту |
| FR4.2 | Помилка сервісів, які надають інформацію про транспортні засоби |

## 2.2 Концептуальний опис архітектури програмного продукту



## 2.3 План розробки програмного продукту

### 2.3.1 Оцінка трудомісткості розробки програмного продукту

Изображение выглядит как стол

Автоматически созданное описание

Изображение выглядит как стол

Автоматически созданное описание

Изображение выглядит как стол

Автоматически созданное описание

Изображение выглядит как текст

Автоматически созданное описание



EF – складність розробки додатку

UUCP = 32

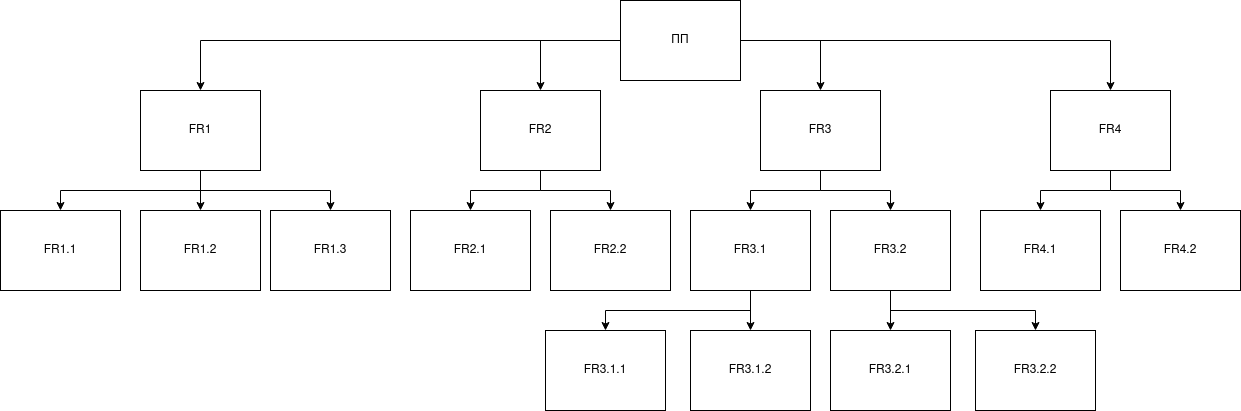
TCF = 8.035

EF = 10.555

UCP = 2713.9016

### 2.3.2 Визначення дерева робіт з розробки програмного продукту

|  |  |
| --- | --- |
| Ідентифікатор функції  (назва) | Назва функції |
| FR1 | Запуск ПП |
| FR1.1 | Успішне завантаження ПП |
| FR1.2 | Відсутність інтернету |
| FR1.3 | Помилка сервера при завантаженні |
| FR2 | Отримання мапи |
| FR2.1 | Успішне отримання мапи |
| FR2.2 | Помилка сервісу, який надає мапу |
| FR3 | Введення координат |
| FR3.1 | Введення точки старту |
| FR3.1.1 | Успішне введення точки |
| FR3.1.2 | Точка відсутня на мапі |
| FR3.2 | Введення кінцевої точки |
| FR3.2.1 | Успішне введення точки |
| FR3.2.2 | Точка відсутня на мапі |
| FR4 | Отримання варіантів руху |
| FR4.1 | Успішне отримання ≥ 1 варіанту |
| FR4.2 | Помилка сервісів, які надають інформацію про транспортні засоби |



### 2.3.3 Графік робіт з розробки програмного продукту

#### 2.3.3.1 Таблиця з графіком робіт

Изображение выглядит как текст, шкафчик, волейбол

Автоматически созданное описание

#### 2.3.3.2 Діаграма Ганта

Изображение выглядит как текст, шкафчик, волейбол

Автоматически созданное описание

|  |  |
| --- | --- |
| 3 Проектування програмного продукту |  |
|  |  |
| 3.1 Концептуальне проектування на основі UML-діаграми концептуальни класів   UML-діаграми концептуальни класів  На цій схемі можно побачити, як концептуально работає додаток, ми маємо маршрут який потребує наявність точок початку та кінця маршруту, та API котре прокладе йому маршрут та поверне значення по дістанції, часу і інших значеннях. Однак API потрібно знати яким типом пересування потрібно прокласти маршрут, бо там де не проїде машина може пройти людина і це може скоротити час пересування, чим наварачувати коло. |  |
|  |  |
| 3.2 Проектування програмних класів   UML-діаграма класів |  |
|  |  |
| Як можно побачити на схемі ми маємо класичний шаблон проектування MVC. В нас View що відповідає за відображення даних користувачеві та прийняття даних од нього. HomeController же виконує роботу обробку отриманих даних за допомогою Model котра виражена в цій роботі у двух класах Points та Point   |  |  |  |  | | --- | --- | --- | --- | | Ідентифікатор функції | Назва функції | Назва класу | Назва методу класу з параметрами | | F1.1.1 | Отримання мапи | Controller | GetMap() | | F2.1.1 | Отримання точок від користувача | View | SendPoint () | | F3.1.1 | Відправка даних до користувача | Controller | Send(StartPoint, EndPoint ) | | F4.1.1 | Відловлення помилок | Controller | CatchErrors() | | F5.1.1 | Функція отримання точок із мапи по кліку мищі | View | function\_Click(in e) | |  |
| 3.3 Проектування алгоритмів роботи методів програмних класів   Схема1 Алгоритму методу Send    Схема2 Алгоритму методу CatchErrors    Схема3 Алгоритму методу function click |  |
|  |  |
| Схема4 алгоритму методу SendPoint |  |
| На поданних схемах забраженні алгоритми основних методів/функцій додатку. Розглянемо їх  На схемі3 Показан алгоритм функції function click як показанно в алгоритмі ця функція починає діяти коли створюєтся дія від користувача клік по мапі. Після цього з місця натисканія ми отримуємо інформацію о позиції точки, та перевіряємо чи є в нас друга точка, якщо є значить це змінили маршрут і старий нам потрібно видалити, якщо ні, то потрібно навпаки додати точку. Якщо на виході усіх цих операцій ми маємо 2 точки, то потрібно передати дані до функції SendPoint що передасть іх до нашого контролеру на перевірку та обробку  На схемі4 як рз розглядаєтся функція SendPoint. Спочатку ми формуємо запит з отриманих до функції даних точки, та відпровляємо його до контролеру, після того як контролер все зробив він у цюж саму функцію повертає відповідь, коректні чи були дані. Якщо так то виводимо до консолі зі сторони серверу, що все добре, якщо ні, демонструємо помилку користувачеві з поясненням.  На схемі1 ми бачемо метод прийому даних в контролері, він відповідає за виклечення функції перевірка на помилки та створення відповіді до нашої функції SendPoint.  На схемі2 ми бачемо метод перевірки даних, нам відомо що довгота та широта можуть бути в деякому деапазоні з -180 – 180 та -90 – 90 відповідно |  |
| 3.4 Проектування тестових сценаріїв верифікації роботи програмних модулів |  |
| 3.4.1 Проектування тестових сценаріїв верифікації функціональних вимог Для методу map.on |  |
| |  |  |  |  | | --- | --- | --- | --- | | FR ID | Test  Case  ID | Опис значень вхідних даних  до методу (процедури, функції) | Опис очікуваних значень  результату виконання методу | | FR1 | TC1 | Нічого не передаєтся | Відображаєтся мапа міста | | FR1 | TC2 | point1 = point(82.93057,82.93057)  point2= point(102.93057,102.93057) | Відображаєтся маршрут з point1до point2 | | FR1 | TC3 | point1 = point(82.93057,82.93057)  point2 = null | Відображаєтся точка на мапі з координатами point1 | | FR1 | TC4 | point1 = null  point2 = point(82.93057,82.93057) | Відображаєтся точка на мапі з координатами point2 | | FR1 | TC5 | point1 = null  point2 = null | Відображаєтся мапа города | |  |
| Для функції SendPoint   |  |  |  |  | | --- | --- | --- | --- | | FR ID | Test  Case  ID | Опис значень вхідних даних  до методу (процедури, функції) | Опис очікуваних значень  результату виконання методу | | FR2 | TC1 | point1 = null  point2 = null | Отправляется пустой запрос на send() | | FR2 | TC2 | point1 = point(82.93057,82.93057)  point2= point(102.93057,102.93057) | Отправляется запрос с точками point1 и point2 на send () | | FR2 | TC3 | point1 = point(82.93057,82.93057)  point2 = null | Отправляется запрос с  точкой point1 на send () | | FR2 | TC4 | point1 = null  point2 = point(82.93057,82.93057) | Отправляется запрос с точкой point2 на send () |  3.4.2 Проектування тестових сценаріїв верифікації нефункціональних вимог  |  |  |  |  | | --- | --- | --- | --- | | NFR ID | Test  Case  ID | Опис значень вхідних даних  до методу (процедури, функції) | Опис очікуваних значень  результату виконання методу | | NFR1 | TC1 | Ввід точки в поле по координатам |  | | NFR1 | TC2 | Ввід точки в поле по тексту |  | |  |
| 3.4.3 Створення матриці відповідності вимог до програмного продукту  |  |  |  |  |  |  |  |  |  |  |  |  | | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | |  | TC1 | TC2 | TC3 | TC4 | TC5 | TC1 | TC2 | TC3 | TC4 | TC1 | TC2 | | FR1 | 1 |  |  |  |  |  |  |  |  |  |  | | FR2 | 1 |  |  |  |  |  |  |  |  |  |  | | FR3 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |  |  | | FR4 |  |  |  |  |  |  |  |  |  | 1 | 1 | |  |
|  | |
| 4 Конструювання програмного продукту | |
|  | |
| 4.1 Особливості конструювання програмних модулів | |
| 4.1.1 Конструювання програмної структури з урахуванням спеціалізованого Framework для FrontEnd-компонент архітектури (за наявністю) В цьому додатку як Framework викоритовувався JQuery з його допомогою відправляються запити до контролеру і його структура виглядає так  url: куди відправити  type: Якого типу выдправляємий запит, для отримання даних чи відправки  headers: Якого типу внутрішня частина із самими даними  data: Самі дані  success: Що робити при повернені позитивного відгуку нам запит  error: Що робити при повернені негативного відгуку на запит | |
| 4.1.2 Конструювання програмних класів (за наявністю об`єктно-орієнтованого програмування) В додатку використовувалось 3 класа це homeController, Points,Point де Point використовуєтся для зберегання даних точок і має лише поля для цього, Points зберігає початкову та кінцеву точку для роботи із маршрутом і також має лише два поля StartPoint – EndPoint  homeController головний клас котрий відповідає за правильну обробку даних на стороні BeckEnd та поверненя оброблених даних на FrontEnd, він має методи Index()Send()Error()CathErrors()котрі відповідають за підгрузку строниці, роботи із запитами та даними, перенаправлення у разі глобальної помилки на стороні FrontEnd до сторінки з повідомленням про це, та метод обробки помилок даних котрі будуть використовуватись для подальшого векористання та пришли від користувача | |
| 4.1.3 Конструювання алгоритмів методів програмних класів або процедур/функцій В даному додатку є 4 основні функції/методи це function\_click, SendPoints, Send,CatchErrors та вспоміжні, що використовуваються для стабільної роботи додатку Index(), Error()  Як мовилось раніше Index() відповідає за завантаження сторінки  Error() перенаправляє до вспоміжної сторінки при глобальній помилці, чи недоступності сторінки Index  function\_click отримує дані із натискання мищі по мапі за допомогою івентів, що має API та передачі цих даних до функції SendPoints попередньо первіряючи на обновлення точок та їх наявність  SendPoints() за допомогою раніше описаного фреймворку, відправляє запит і все, тоб то ця функція використовується лише для виклачення та формування запиту за допомогою Framework-у JQuery.  Send() отримує точки із SendPoints() та використовує функцію CatchErrors для перевірки даних, та після формування зворотньої відповіді в залежності від результату  CatchErrors() при отримані точок перевіряє їх на усі варіанти можливих похибок такі як виход за дозволені границі значень за допомогою оператору if…else та повертає відповідь про проходження тестування | |
| 4.1.4 Особливості використання спеціалізованих програмних бібліотек та API (за наявністю) В моєму додатку використовивались API сервісу MapBox котрі потребували підключення бібліотек роботи з даним сервісомтакі як mapbox-gl-js/v2.6.1/mapbox-gl.js та plugins/mapbox-gl-directions/v4.1.0/mapbox-gl-directions.js 4.2 Модульне тестування програмних модулів Для тестування Апартоно програмних інтерфейсів були розроблени Unit тести подібні один на одного і використовани для перевірки роботи методу Send котрий приймає і оброблює дані. Де ми перевіряли правільність відповіді методу при задані правильних точок і неправильних по різним параметрам. | |
| Изображение выглядит как текст  Автоматически созданное описание  Один із тестів як приклад | |
| Изображение выглядит как текст  Автоматически созданное описание  Результат роботи тестів | |

|  |
| --- |
| 5 Верифікація програмного продукту |
| 5.1 Тестування апаратно-програмних інтерфейсів програмного продукту |
| Щоб не повторюватись з минулим пунктом можу сказати, що тестування API зівпадає в моєму додатку співпадає с тестуванням модулів, бо робота з API і є головним модулем роботи додатка |
| 5.2 Тестування інтерфейсу користувача програмного продукту  |  |  |  |  | | --- | --- | --- | --- | | Test  Case  ID | Опис дій з ПП на  рівні інтерфейсу  користувача | Опис очікуваних результатів  взаємодії з ПП на рівні  інтерфейсу користувача | Результат тестування  (Passed/Failed/Blocked) | | 1 | Відкриття додатку |  | Passed | | 2 | Відправка точок |  | Passed | | 3 | Відправка точок |  | Passed | |
| 5.3 Тестування часу реакції програмного продукту на дії користувача |

За умови виконання умов, час реакцыъ продукту був:

|  |  |  |
| --- | --- | --- |
| Тип умови  (Hard/Soft) | Опис вимог | Опис реальних умов |
| Hard | CPU: Intel Core i3-10105 Comet Lake  3,700Ghz Box  Memory: HDD 256 | CPU: Intel Core i5-8600  3,1 GHz  Memory: SSD 256 |
| Soft | OS: Windows 7,8,10 | OS Windows 10 64bit |
| Soft | Web browser: Google Chrome,Microsoft Edge, Mozilla Firefox  Opera Mobile | Web browser: Google Chrome ver. 96 |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| FR  (назва) | Максимальний час  реакції ПП на дії  користувачів, секунди | Кількість  проведених  тестів | Результат  тестування,  секунди | Відносна  похибка  вимірювань, % |
| Виведення Мапи | 5 | 4 | 4 | 7 |
| Виведення маршруту на мапі | 0,1 | 5 | 0,05 | 8 |
| Виведення списку варіантів | 1 | 3 | 0,5 | 6 |

|  |
| --- |
| 6 Розгортання та валідація програмного продукту |
| 6.1 Інструкція з встановлення системного програмного забезпечення |
| 1) Скачати та розархівувати проект Gator  2) Також вам знадобиться ключ доступу від MapBox для взаємодії з картою, а отже і відправкою точок та демонстрації маршруту, для цього потрібно в полі mapboxgl.accessToken установити цей ключ  3) Запустити програму методами середи разробки, і у вас на комп'ютері буде розгорнутий локальний сервер готовий до роботи |
| 6.2 Інструкція з використання програмного продукту ПП надає можливість користувачу вести дані точок, як показано  на малюнку 1.  Изображение выглядит как карта  Автоматически созданное описание  Малюнок 1. Варіант введення точок  Для находження варіантів маршрута користувач потрібен ввести чи помітити на карті дві точки.  Слід пам’ятати що ці точки повинні існувати, тобто бути в діапазоні від -90 – 90 по широті, та в діапазоні від -180 – 180 по довготі |
| 6.3 Результати валідації програмного продукту |

Мета продукту – зменшити кількість кроків при отримані інформації варіантів маршруту та його швидкості

Раніше коли ми бажали побачити як дістатися від одного місця як найшвидше ми були повиннізайти на gooogle maps щоб переглянути як можно дістатися на машині, чи пішки, чи маршруткою та інші додатки котрі показували який час якщо їхати велосипедом, як іхати самокатом і т.п.

Зараз же ми можемо війти в додаток і побачити все що потрібно вже агреговане

Раніше:

Відкрити додаток GoogleMaps зареєструватись у ньому, відкрити додаток bolt зареєструватись у ньому, nextbike зареєструватись у ньому

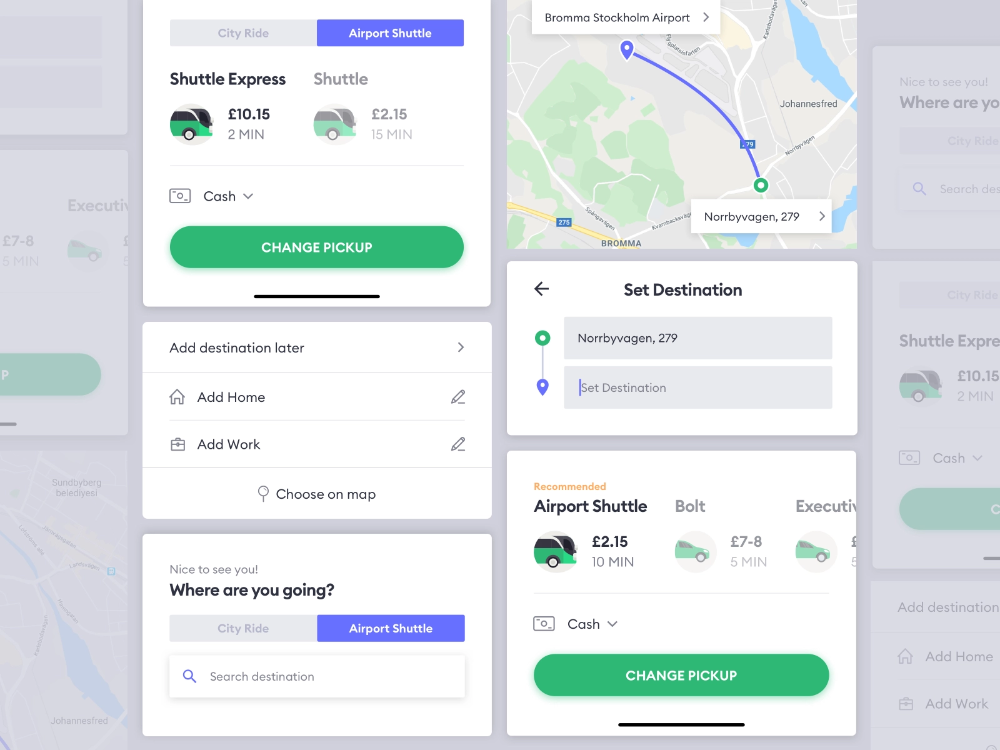
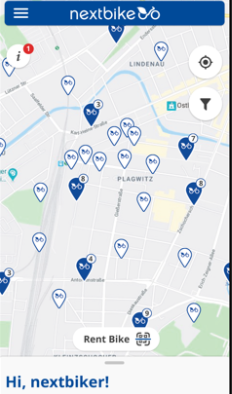
Кількість дій з додатком що в кожному додатку ми повині ввести кожен раз куди і звідки їхати

1. Зайти у додаток
2. Зареєструватись
3. Ввести точки

І так повторювати скільки додатков ти маєш

В нашому прикладі на 3 тому буде 9 кроків

Изображение выглядит как карта

Автоматически созданное описание 

Малюнок 2. Приклади роботи інних додатків

Зараз усе в одному додатку: Зайшов побачив усе разом. Кількість дій: 2

1. Зайти у додаток
2. Ввести точки

Изображение выглядит как текст

Автоматически созданное описание Изображение выглядит как текст

Автоматически созданное описание

Малюнок 3. Приклад роботи нашого додатку

Мета досягнена!