МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
КИЇВСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ
ІМЕНІ ТАРАСА ШЕВЧЕНКА
Факультет інформаційних технологій
Кафедра інтелектуальних технологій

ВИПУСКНА КВАЛІФІКАЦІЙНА РОБОТА БАКАЛАВРА

НА ТЕМУ

Програмний модуль прогнозування потреб техобслуговування картінгів

Галузь знань **12 «Інформаційні технології»**

Спеціальність **122 «Комп'ютерні науки»**

Освітня програма «**Комп'ютерні науки»**

Освітній рівень: бакалавр

Виконав: студент 4 курсу, групи КН- 41
Пашковський Павло Володимирович

Керівник: кандидат технічних наук, доцент Іларіонов О. Є.

Випускна кваліфікаційна робота бакалавра допущена до захисту
рішенням кафедри *інтелектуальних технологій*
Протокол № 11 від 06.06.2022 р.
зав. кафедри_____ доц. Іларіонов О.Є.

Київ – 2022

КИЇВСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ ІМЕНІ ТАРАСА ШЕВЧЕНКА

Факультет інформаційних технологій
Кафедра інтелектуальних технологій
Спеціальність 122 «Комп'ютерні науки»

ЗАТВЕРДЖУЮ
Завідувач кафедри інтелектуальних технологій
Іларіонов О.Є.
_____
"23" грудня 2021 р.

ЗАВДАННЯ
НА ВИПУСКНУ КВАЛІФІКАЦІЙНУ РОБОТУ СТУДЕНТОВІ
Пашковському Павлу Володимировичу

1. Тема проекту (роботи)
Програмний модуль прогнозування потреб техобслуговування картінгів затверджена протоколом засідання кафедри від «23» грудня 2021 р. № 4
2. Термін здачі студентом закінченого проекту (роботи) 31 травня 2022 року
3. Вихідні дані до проекту (роботи)
Таймінг картинг центру «Жага Швидкості» у місті Києві
4. Зміст розрахунково-пояснювальної записки (перелік питань, що їх належить розробити)
    1. Аналіз перегонів на картах
    2. Розробка архітектури модулю прогнозування потреб техобслуговування картінгів
    3. Програмне забезпечення модулю прогнозування потреб техобслуговування картінгів
5. Перелік презентаційного матеріалу (з точним зазначенням обов'язкових презентацій)
Об'єкт, предмет, мета та завдання дослідження (1 слайд), порівняльний аналіз існуючих рішень (1 слайд), проектування інформаційної системи (3 слайди), інформаційне забезпечення (1 слайд), структура програмного забезпечення (1 слайд), огляд процесу тестування (1 слайд), висновок (1 слайд)
6. Дата видачі завдання 15 лютого 2022 року

Керівник _____ / Іларіонов О. Є. /
(підпис) (ПІБ)

Завдання прийняв до виконання _____/ Пашковський П. В. /
(підпис) (ПІБ)

КАЛЕНДАРНИЙ ПЛАН

Пор. №	Назва етапів випускної кваліфікаційної роботи	Термін виконання етапів випускної кваліфікаційної роботи	Примітка
1.	Аналіз постановки задачі, формалізація задачі, вибір методів та засобів реалізації поставленої задачі, аналіз літературних джерел. Постановка задачі	25.01-04.02	
2.	Функціональний, бізнес-аналіз	05.02-14.02	Виконано
3.	Огляд обраних засобів/технологій	15.02-25.02	Виконано
4.	Проектування архітектури	26.02-07.03	Виконано
5.	Розробка та тестування програмного продукту.	08.03-25.04	Виконано
6.	Демонстрація базового варіанту програмного продукту	26.04-07.05	
7.	Оформлення пояснювальної записки, підготовка презентації		

Студент-дипломник _____ / Пашковський П. В. /
(підпис)

Керівник випускної
кваліфікаційної роботи _____ / Іларіонов О. Є. /
(підпис)

ЗМІСТ

АНОТАЦІЯ

Пашковський Павло Володимирович виконав випускну кваліфікаційну роботу на тему «Програмний модуль прогнозування потреб техобслуговування картінгів» за спеціальністю 122 – «Комп’ютерні науки».

У випускній кваліфікаційній роботі проведено аналіз сучасних перегонів на картах, розглянуто архітектуру системи. Спроектована архітектура інформаційної системи аналізу статистики перегонів на картах. Розроблена програмне забезпечення для інформаційної системи аналізу статистики перегонів на картах.

Ключові слова: перегони, картинг, web-додаток.

ANNOTATION

Pashkovskyi Pavlo Volodymyrovych graduated from the graduation qualification work on the topic "Software module for the kart service forecast" for the specialty 122 - "Computer Science".

At graduation qualification work, an " Software module for the kart service forecast " was carried out, the architecture of the system was examined. The architecture of the basic information system for kart race data was designed. The software for the initial information system in kart race data has been decomposed.

Key words: race, kart, web application.

ВСТУП

Діджиталізація процесів, цифрова трансформація, цифрова освіта, діджитал-маркетинг - слово "діджитал" вже декілька років у всіх на вустах. Нинішня криза з її ефектом бомби, що вибухнула, довела, що не лише майбутнє залежить від переходу на цифрові технології, а й теперішнє поставлене на карту, якщо ми не діятимемо швидко. [6]

Діджиталізація - рушійна сила. Її основні переваги:

- економія часу і підвищення продуктивності - автоматизація внутрішніх процесів;
- оптимізація та покращення комунікацій - як внутрішніх, так і зовнішніх;
- конкурентні можливості за рахунок поліпшення клієнтського досвіду і загальної оптимізації робочого процесу.

Мета моєї дипломної роботи є розробка нової системи аналізу гоночної статистики для допомоги командам чи окремим пілотам в прийняті рішень про піт-стоп.

Об’єктом дослідження є перегони на картах.

Предметом дослідження є аналіз статистики перегонів на картах для подальшого прийняття рішення про піт-стоп.

Карт — найпростіший гоночний автомобіль без кузова. Швидкість карта (клас Суперкарт) може досягати 260 км/год. Гонки на картах називають картингом.

Картинг – вид моторного спорту, у якому застосовуються карти – невеликі гоночні автомобілі з відкритими колесами. Карт може розвивати швидкість від 24 км/год (любительські моделі) до 260 км/год (спортивні суперкарти). Часто картинг розглядається як початковий етап у підготовці до дорожчих і престижних моторних видів спорту.

РОЗДІЛ 1. АНАЛІЗ ПРОЦЕСУ ПЕРЕГОНІВ

1.1 Аналітичний огляд літератури за темою досліджень процесу перегонів на картах.

На сьогоднішній день майже відсутні автоматичні системи для запису аналізу статистики картів, всі системи записуються людьми по результатам після завершення самого змагання.

В даній системі мають бути відсутні люди які записуються результати, є лише гонщик та система для аналізу яка приймає результати за час всієї гонки від датчиків які розташовані на карті.

Для корисливих цілей аналізу кожного водія дозволяє переглянути можливі варіанти кожного водія та мати інформацію про події та їх результати. На основі мого дослідження можливо в майбутньому передбачати можливі результати гонок, або рейтинг пілотів.

Так як кожній з команд необхідно виконати регламентовану кількість замін, існує проблема «вчасного» заїзду для технічного обслуговування. Піт-стоп проводиться в режимі черги, а отже команда має змогу обрати наступний карт. Через це крайньою необхідністю є аналіз картів для того щоб знати яких карт наразі «найшвидший» [5]

Основні модулі системи гоночної статистики для картів(рис. 1):

1. Гонка (подія);
2. Водії та карти які приймають участь;
3. Поле для гонки - траса;
4. Датчики для збору інформації;

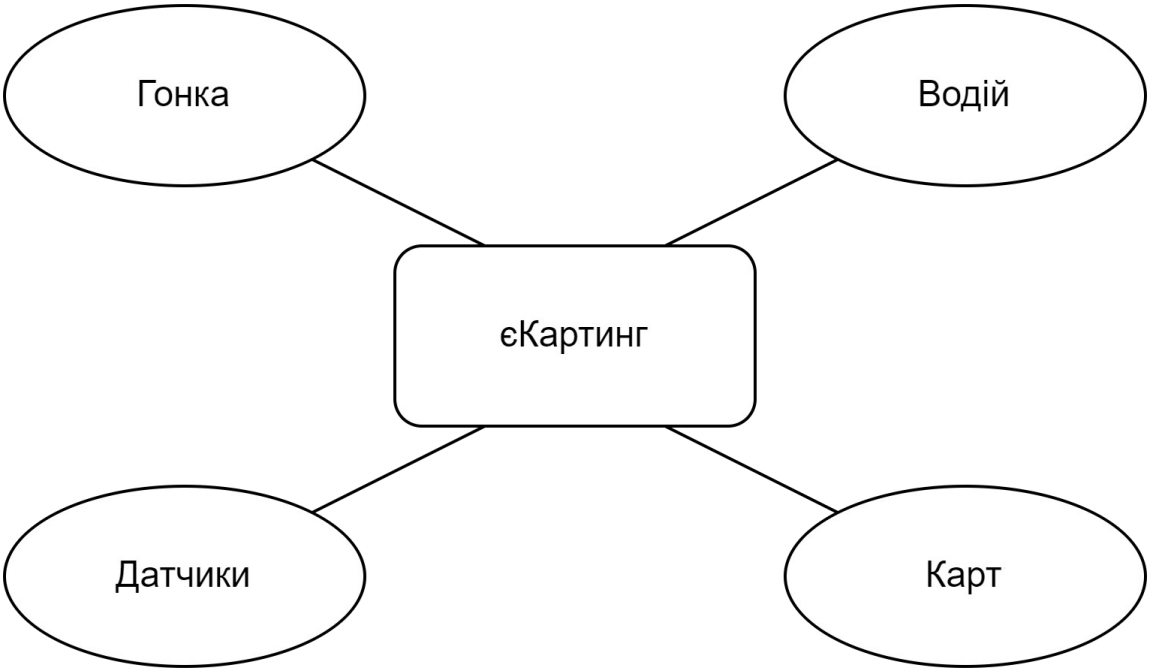


Рис. 1 Основні модулі системи гоночної статистики для картів.

1.2 Аналіз існуючих видів перегонів

Перегони поділяються на два види: спринтерські та стаєрські в кожному з яких стартові позиції вирішуються за допомогою кваліфікації – попереднього заїзду на певний період часу з метою встановлення найкращого часу кола.

Спринтерські дистанції обмежуються кількістю кіл та зазвичай мають декілька заїздів за результатами яких нараховуються бали та вирішується стартова позиція на наступний.

На відміну від спринтерських, стаєрські перегони обмежуються за часом – пілоту необхідно проїхати найбільшу кількість кіл за певний відрізок часу. Зазвичай у марафонах приймають учать команди з пілотів, тому що знаходитись всю дистанцію за кермом дуже важко. А це означає що команда має зробити регламентовану кількість піт-стопів для заміни пілотів.

У моделі, розглянутою мною, перегони відбуваються на прокатному картингу де карти надає організатор змагань, тому кожен карт відрізняється від інших, тому один і той самий пілот може показати різний час на різних картах. Для більш рівних умов організатором регламентовано зміна картів під час перегонів. Враховуючи це аналіз статистики грає важливу роль для оцінки швидкості пілотів та картів для отримання кращих результатів.

З аналізу перегонів я виявив необхідні критерії для розрахунку рейтингу пілотів та картів:

1. Інформація про кожного водія:
    1.1. Темп
    1.2. Стабільність
2. Інформація про кожен карт:
    2.1. Темп
    2.2. Стабільність
3. Інформація про перегони
    3.1. Час перегонів
    3.2. Тип перегонів

Всі перераховані критерії можна отримати за допомогою датчиків та які знаходяться на картах та треку картинг центру. Всю інформацію можна отримати за допомогою арі певного картинг центру. Тому ця система має буди уніфікована для кожного треку та перелаштовуватись за короткий проміжок часу.

А отже для обробки та відображення інформації про кожного пілота та карта необхідні:

1. Модуль отримання інформації
2. Модуль аналізу інформації
3.
4. Модуль збереження інформації
5. Модуль відображення інформації

Враховуючи швидкоплинність подій під час змагань неможливо повністю покладатись на програмний модуль, тому він має допомагати оцінювати пілотів та карти та надавати їм певний рейтинг.

Додаток має містити таку інформацію:

1. Інформація про кожного водія
2. Інформація про кожен карт
3. Рейтинг водія
4. Рейтинг карту

1.3 Аналіз основних процесів предметного середовища.

Створення інформаційної системи аналізу гоночної статистики для картів.

Користувач переходить за посиланням на сторінку додатку та має інформацію про рейтинг кожного водія та картів для отримання чіткої статистики.

Система в свою чергу аналізує рейтинг кожного водія та карту за інформацією отриманої під час перегонів.

Основні цілі системи:

- Зручний перегляд статистики;
- Рейтинг кожного водія;
- Рейтинг кожного карта;

Опис профілів зацікавлених сторін:

1. Внутрішні зацікавлені сторони:
    - Розробник сайту
2. Зовнішні зацікавлені сторони:
    - Стратеги

- Пілоти

Функціонал з точки зору стратега представлено на рис 2 [10]

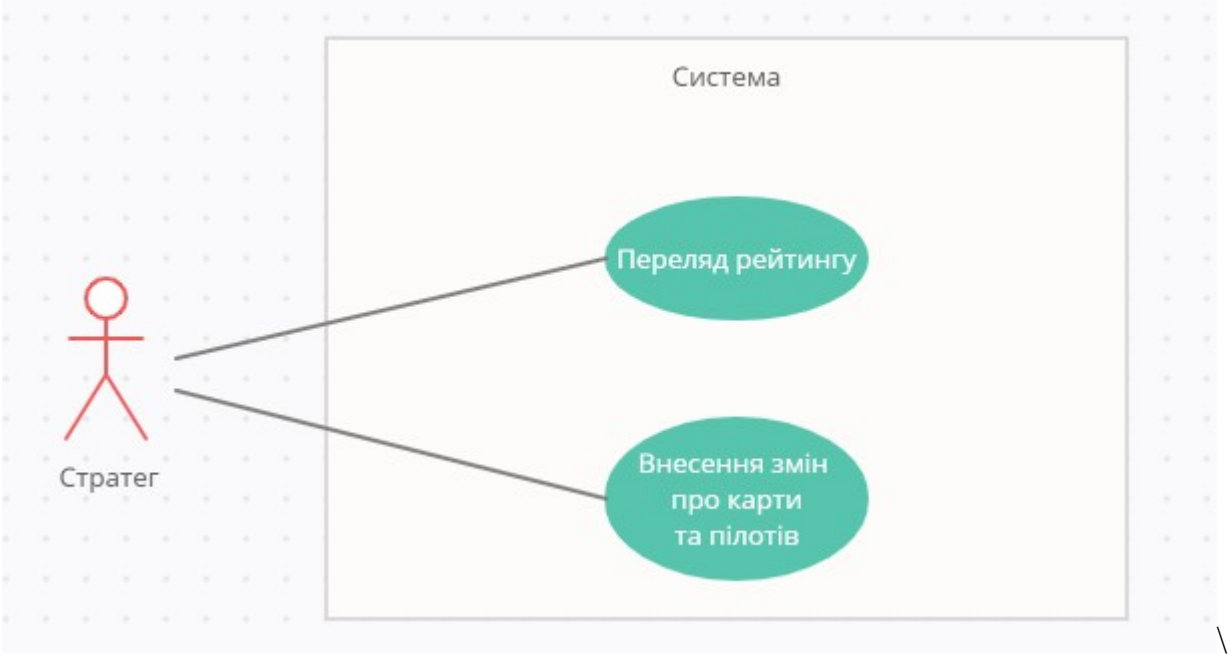


Рис. 2 UML – діаграма прецедентів.

Першим етапом користування ІС аналізу гонок є отримання інформації про користувача який бажає отримати рейтинг для конкретного водія та карту. Персонажі завдяки яким працює система є ІС аналізу водіїв яка в свою чергу перевіряє всю інформацію переданої через датчики на картингу та обробляє запити користувача для його перегляду рейтингу водіїв та картів.

1.4 Постановка задачі на розробку програмного модуля для аналізу гоночної статистики для картів

Основна задача для програмного модуля - розробка рейтингової системи. Система має відображувати оброблені данні на адаптивному та крос-браузерному сайту Головні критерії якого є минулі результати водіїв, картів та інформація про перегони.

Функціональні вимоги:

1. Отримання даних - Система має автоматично діставати дані для формування статистики та подальшого аналізу
2. Збереження статистики водія та карта - Для чіткого аналізу рейтингу водія необхідно записувати всю інформацію минулих гонок водів та картів для подальшого використання.
3. Аналіз статистики водія та карта - Завдяки системі аналізувати результати гонки, та розраховувати рейтинг для окремого карта та пілота.

Нефункціональні вимоги:

1. Відображення статистики водія та карта - Для зручності перегляду системи аналізу необхідна інформація кожного пілота та карта минулим гонкам.
2. Зручний інтерфейс для відображення - Модуль має містити веб-інтерфейс для відображення інформації для перегляду рейтингу водіїв, картів. Сайт повинен бути зручним для ока користувача та мати інформативні таблиці.

1.5 Висновки до першого розділу

Завдяки аналізу подій гонки можливо переглядати рейтинг кожного водія та картів для аналізу майбутніх подій та передбачити результати кожної гонки завдяки датчикам з картингу які фіксують увесь процес гонки для запису всього процесу. Мій проект має майбутнє для всіх можливих результатів інших подій в яких є машина та водій. Для стратегів які можуть більш детально та чітко виявити можливі варіанти кожної події гонки та більш чітко визначити швидкість кожного пілота та карту. Моя система опрацьовує без людини всю інформацію та запобігає можливого людського фактору помилки події, та отримувати, записувати всю інформація з датчиків картингу.

РОЗДІЛ 2. РОЗРОБКА АРХІТЕКТУРИ ГОНОЧНОЇ СТАТИСТИКИ ДЛЯ КАРТІВ

2.1 Розробка архітектури

2.1.1 Функціональний аналіз

Бізнес-процеси (рис. 3) допомагають отримати повну картину початку розробки системи. Вершина дерева функцій є функція модулю аналітики є функціональний модуль картингу який включає в собі клієнтські функції та адміністративні функції. [11]

До клієнтських функцій належать три основних модулі: ведення акаунту, реєстрація та отримання статистики. Реєстрація допомагає системі аналізувати та відрізняти нових користувачів та старих, звідки і виходить другий модуль - ведення акаунту, для редагування особистого кабінету та можливість змінювати особисті дані. До модулю отримання статистики належать три під модулі. Перший модуль - отримання рекомендація, завдяки системі можливо проаналізувати можливі варіанти гонки від аналізу даних про саму гонку та всі причасних водіїв. Другий модуль - статистика минулих матчів включає в собі можливість переглядати старі записані дані від минулих матчів та переглянути детальну інформацію наживу не через комп'ютерну аналітику. Та останній третій модулю - статистика водіїв, а саме їх рейтинг завдяки якому можливо самостійно проаналізувати досягнення кожного водія по його можливостям.

До адміністративних функцій належать два модулі - робота з базами даних та робота з клієнтами. До роботи з клієнтами є можливість переглядати кожного користувача та розуміти адекватність користувачів и можливий віковий інтерес до нашого додатку. До роботи з базами даних входить редагування всі можливих таблиць, це редагувати водіїв, траків або записувати старі матчі які зараз можливо актуальні із-за старої гонки водія. Та додавати як було прописано раніше, додавати стару інформацію, додавати можливі

майбутні події та додавати події в реальному часі - гонки які відбуваються зараз.

Рисунок 3 – Дерево функцій

Змоделюємо функціонування сайту з отримання аналітики водіїв. Початком запуску є відкриття самого додатку за посиланням користувачем та можлива реєстрація нового користувача або увійти в систему як вже старий користувач, який не перший раз відвідує сторінку аналітики. Наступним кроком є перехід до основної сторінки всієї інформації по всім матчам, водіям, майбутнім матчам та іншої аналітики від системи яка орієнтується від датчиків на карті. Основною функцією сайту є перегляд сторінки водіїв та вибір необхідного водія для перегляду ‘ого минулих гонок та рейтингу водія, можливий аналіз водія якщо в нього були раніше гонки і система змогла отримати дані від датчиків та проаналізувати водія.

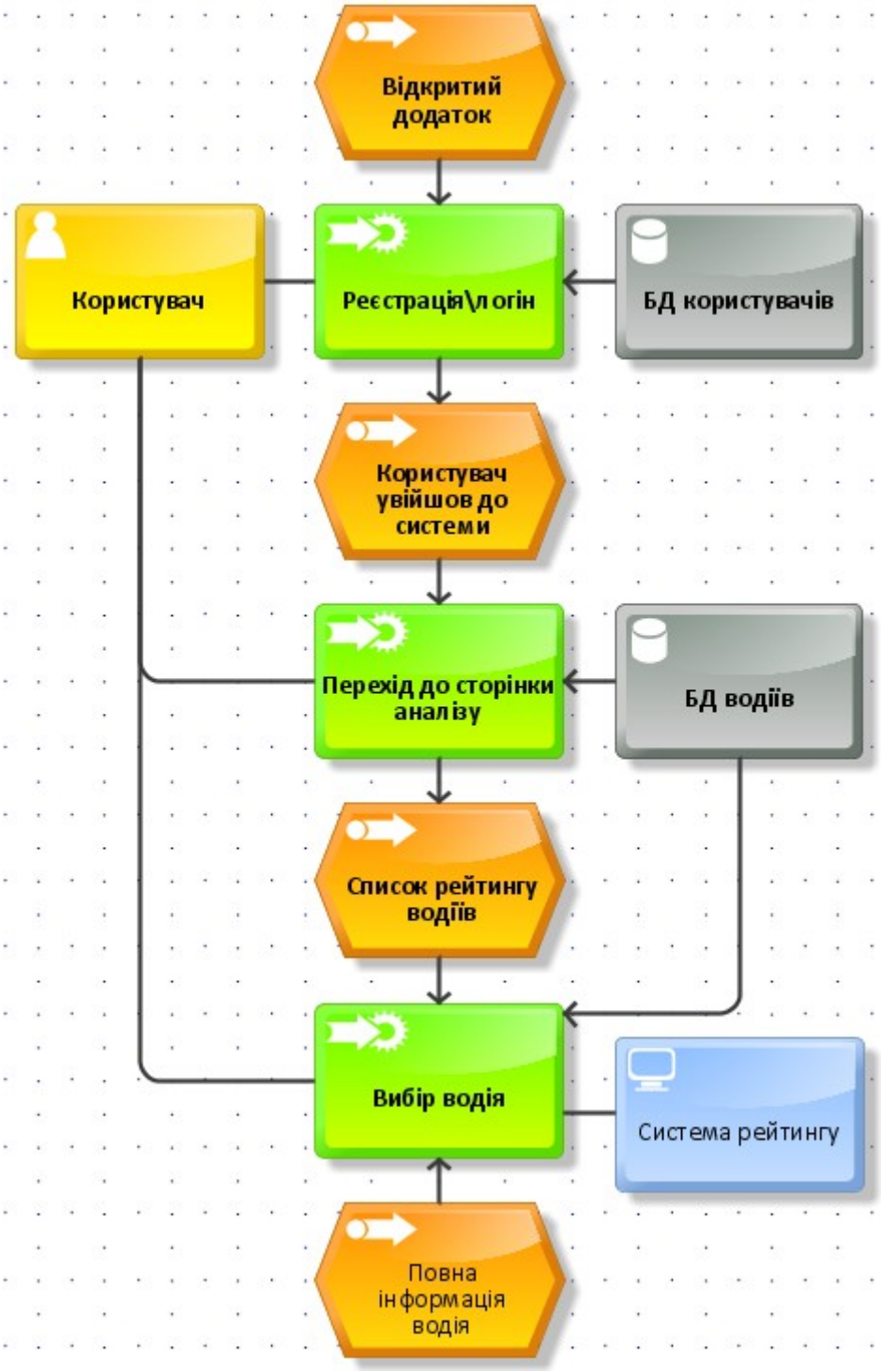


Рисунок 4 – Діаграма Event-Driven Process Chain для функціонування сайту

Наступним кроком моделювання сайту аналізу гонок на траках є ланцюжок процесів - що відбуваються під час функціонування сайту. На

найвищому рівні знаходиться процес “Робота з сайтом” з деякими під процесами, а саме: “Реєстрація на сайті” та “Користування сайтом з аналізу”. Процес реєстрації включає послідовні процеси: введення особистих даних: “Введення логіну” та “Введення паролю” які підтримує процес “Підтвердження профілю” через можливі пристрої та додатки, наприклад через телефон на мобільний номер телефону або через комп'ютер на електронну пошту, як буде зручно користувачу.

Під процес “Користування сайтом аналізу” є найголовнішим процесом для користувача який перейшов за посиланням на нашу сторінку - перегляд інформації через процеси “Пошуку водіїв”, “Перегляд майбутніх гонок” та “Перегляд минулих гонок”. Для перегляду минулих та майбутніх гонок достатньо інформації для перегляду результату події та перегляд інформації хто приймав участь в гонці. Найголовніший процес - “Пошук водіїв” який включає в собі під процеси “Перехід на сторінку рейтинга водія” та “інформація за минулі гонки. Процес перегляду рейтингу допомагає користувачу зрозуміти майстерність водія та переглянути минулі гонки водія для особистого аналізу водія та його навичок.

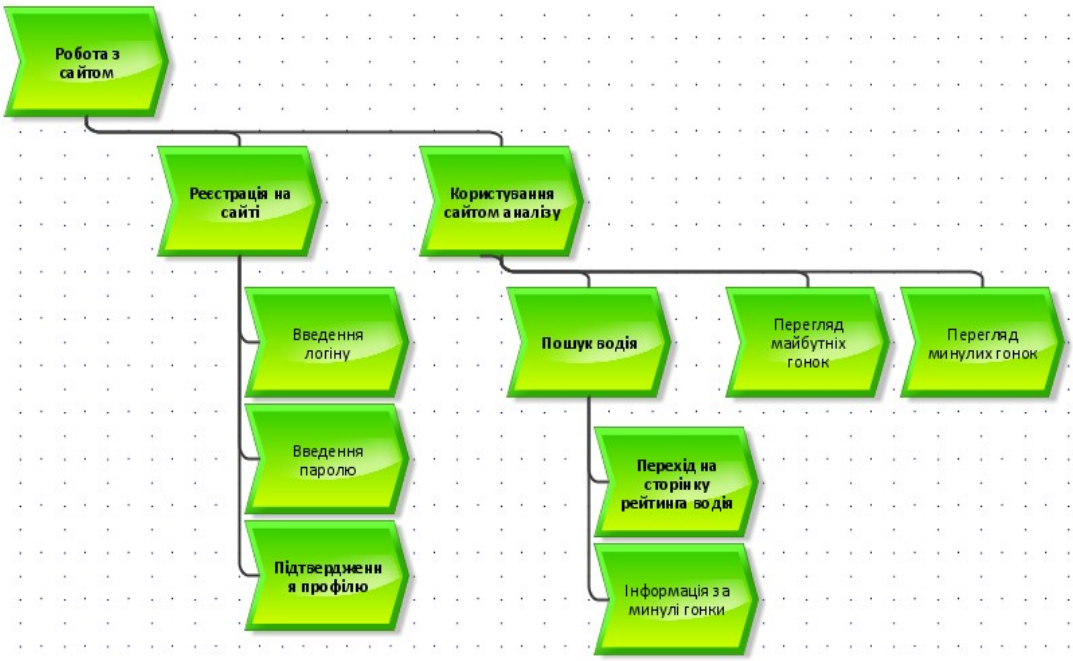


Рисунок 5 – Діаграма Value Added Chain Diagram для функціонування мобільного додатку

2.1.2 IDEF0 процесу видачі рейтингу за допомогою аналізу інформаційної системи

Представимо виконувані процеси у предметному середовищі за допомогою діаграм «ЯК БУДЕ» у нотації IDEF0, тобто вже з використанням сайту аналізу рейтингу водія.

Першим етапом користування ІС аналізу гонок є отримання інформації про користувача який бажає отримати рейтинг для конкретного водія. Персонажі завдяки яким працює система є ІС аналізу водіїв яка в свою чергу перевіряє всю інформацію переданої через датчики на картингу та обробляє запити користувача для його перегляду рейтингу водіїв.

Елементом керування є закон України про «Захист персональних даних», тому що на кожному етапі зберігаються особисті дані користувача.

Рисунок 6 - Контекстна діаграма ЯК БУДЕ

На наступному етапі проведемо декомпозицію основних процесів діаграми ЯК БУДЕ (рис.8).

Перший процес для запуску системи є передача через авторизацію даних користувача - завдяки якій система зможе відсортувати нових користувачів та старих для введення інформації про відвідування додатку аналізу рейтингу водіїв гонок на картах. Система обробляє інформацію користувача та переходить до етапу очікування вибору користувача, а саме вибір користувачем необхідного водія та передачі даних водія для аналізу водія, його минулі гонки та результати і наступним етапом системи є створення інформації рейтингу водія для виконання запиту користувача на отримання рейтингу водіїв.

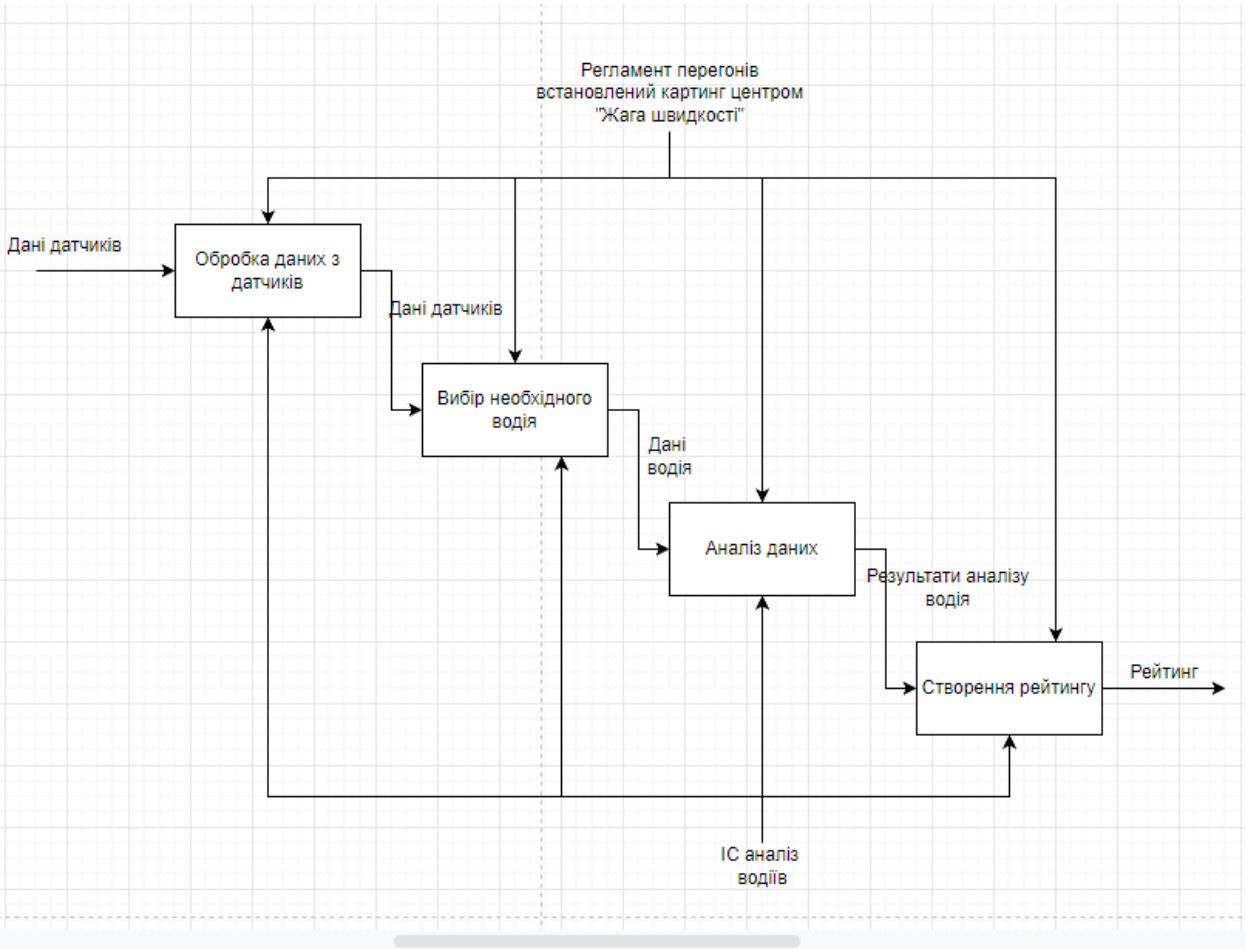


Рисунок 7 – Декомпозиція основних процесів діаграми ЯК БУДЕ

2.2 Інформаційне забезпечення аналітичної рейтингової системи

2.2.1 Розробка модулю отримання даних

Актуальну інформація під час перегонів можливо отримати за допомогою API картинг центру «Жага Швидокості» за посиланням: http://nfs.playwar.com:3333/getmaininfo.json. В результаті обробки json об’єкту модуль має отримати такі вхідні дані:

- Найкраще коло
- Стабільність пілота

2.2.2 Розробка модулю аналізу даних

Логіка розрахунку рейтингу будується на персептроні який має такий вигляд: Два нейрони на вході, ваги які визначаються шляхом навчання на даних кваліфікації, функція активації – сигмоід та один вихід - рейтинг. (рис. 8). [3] [13]

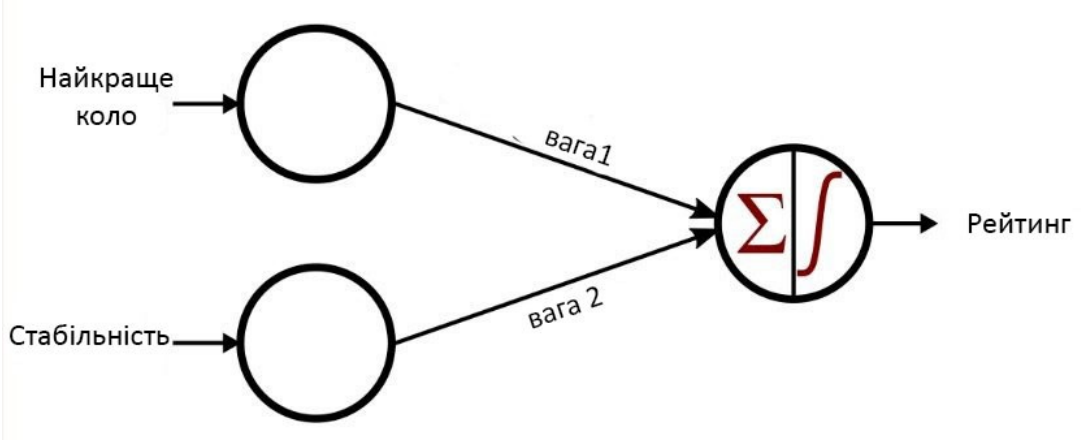


Рисунок 8 – Персептрон.

Персептрон навчаєтеся на результатах кваліфікації де результатом є позиція на стартовій решітці. Рейтинг пілота має значення від 0 до 1 тому він і є результатом отриманим під час тестування персептрону.

2.2.3 Розробка модулю збереження інформації

Як вже було зазначено раніше, дана інформаційна система зберігає та використовує певні дані. Для збереження цих даних буде розроблена реляційна база даних PostgresSQL, яка зберігатиме та видаватиме потрібну інформацію за допомогою бібліотеки sqlalchemy.

Спочатку розробимо концептуальну модель бази даних (рис.9). На ній для заданої предметної області відображено наступні сутності:

- Водій
- Гонка
- Карт

Кожна сутність має зв'язок між іншими (таб. 1) для отримання інформації

на якому полі, який водій та дату події - вся інформація необхідна для аналізу рейтингу кожного водія.

Таблиця 1 - Зв’язки між сутностями

№	Сутності, що утворюють зв’язок	Тип зв’язку	Пояснення
1	Водій - Гонка	Багато до одного	Один водій може відвідувати одночасно тільки одну гонку, а на одній гонці можливо мати участь кілька водіїв.
2	Гонка- карт	Один до одного	Для однієї гонки тільки один карт.

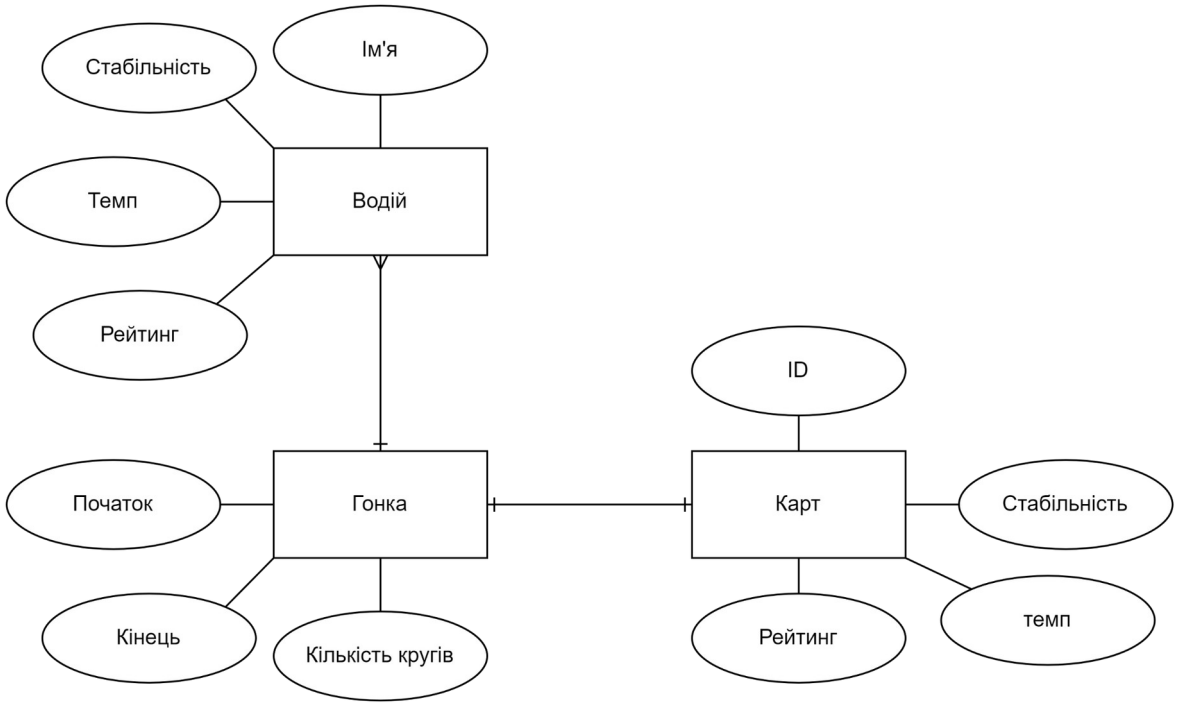


Рисунок 9 – Концептуальна модель бази даних

Наступною є логічна модель (рис. 10). Вона також, як і концептуальна, описує дані на досить абстрактному рівні, але все ж більше наближеному до

реальних баз даних. Також, на відміну від попередньої моделі, ми вже вказуємо первинні ключі, якими в даному випадку є поля «id», і для читача («User») – login, та зовнішні ключі, які пов’язують таблиці між собою. [15]

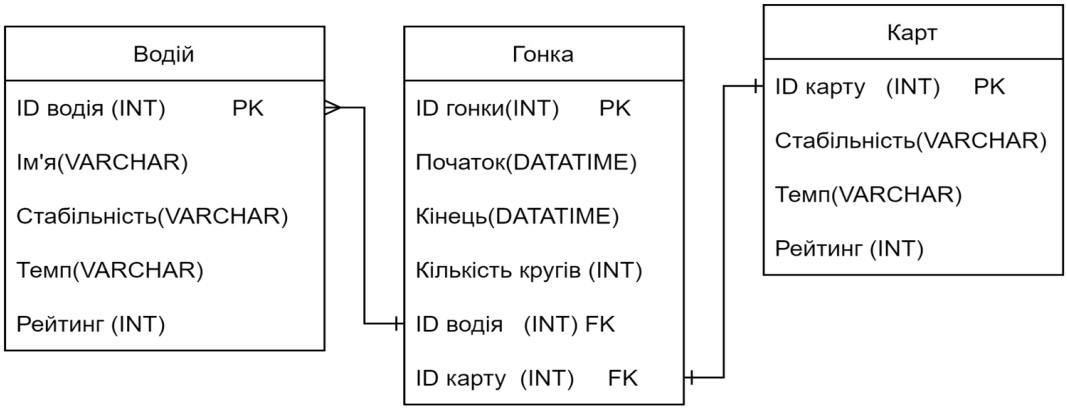


Рисунок 10 – Логічна модель бази даних

Останньою є фізична модель бази даних (таб. 2), розроблена на основі даталогічної. Вона вже відображає, як дані насправді зберігаються в базі даних. На ній вже можна побачити детальніші дані про атрибути, а саме – тип даних та обмеження для деяких з них. Опишемо їх у таблиці.

Таблиця 2 - Склад та характеристика атрибутів таблиць

№ п/п	Назва елемента даних	Тип елемента даних	Обов’язкове значення	Обмеження	Ключ
Таблиця «Водій»					
1	ID водія	Ціле число	так	30	ПК
2	Імя	Символьний	так		
3	Стабільність	Символьний	так		
4	Темп	Символьний	так		
5	Рейтинг	Ціле число			
Таблиця «Гонка»					

1	ID гонки	Ціле число		30	ПК
4	Найкраще коло	Символьний			
5	ID водія	Ціле число		30	ЗК
6	ID карту	Ціле число		30	ЗК
Таблиця «Карт»					
1	ID карту	Ціле число	так	30	ПК
2	Номер	Ціле число			
3	Стабільність	Символьний	так		
4	Темп	Символьний	так		
5	Рейтинг	Ціле число	так		

2.2.4 Розробка модулю відображення інформації

Новий етап - побудова архітектури розроблюваної інформаційної системи. Дана схема відображає модель, структуру, виконувані функції та взаємозв'язок компонентів. До архітектури розробленої інформаційної системи належать два модулі: клієнтський модуль та адміністративний модуль. (рис. 11)

Клієнтський модуль вміщує в собі основні чотири модулі для всіх користувачів сайту. Перший модуль - робота з профілем для користувачів які бажають редагувати свій профіль та змінювати пароль для додатку сайту. Другий модуль - реєстрація, яка допомагає новим користувачам заходити до системи через авторизовані акаунти для ведення статистики кількості нових користувачів та старих. Третій модуль - перегляд всіх подій, які все відбулися та які ще необхідно очікувати. Останній четвертий модуль - модуль пошуку водіїв та перегляд його минулих гонок та отримання інформації від системи рейтингу

Адміністративний модуль вміщує майже два однакових модулі: модуль роботи з водіями та подіями. Модулі роботи з водіями та подіями об'єднує

однакові модулі: редагування, додавання та видалення подій або водіїв для системи аналізу рейтингу водіїв та картів на сайті.[16]

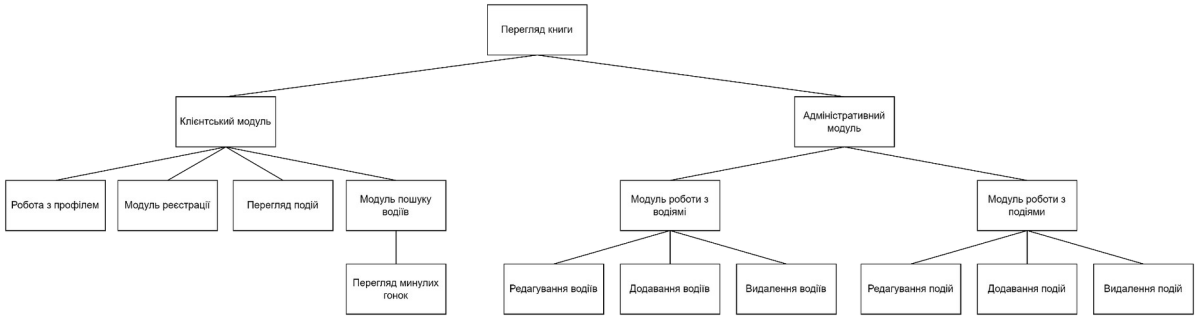


Рисунок 11 – Архітектура системи

2.3 Висновки до другого розділу

В результаті проведення етапу розробки сайту аналітики водіїв було спроектовано систему для подальшої її реалізації.

Було проведено функціональний аналіз та побудовано дерево функцій, яке відображає бізнес-процеси інформаційної системи. На даних схемах було зображено та описано логіку роботи програми. Створена база даних бібліотеки, фізична модель та логічна модель БД.

РОЗДІЛ 3. ПРОГРАМНЕ ЗАБЕЗПЕЧЕННЯ ІНФОРМАЦІЙНОЇ СИСТЕМИ ПРОГНОЗУВАННЯ ПРОТРЕБ ТЕХОБСЛУГОВУВАННЯ КАРТИНГІВ

3.1 Обґрунтування вибору програмних засобів

Для розробки застосунку було використано ряд інструментів та методів, за допомогою яких створено інформаційну систему з відповідною базою даних та користувацьким інтерфейсом.

Основна мова програмування Python [1][7] з використанням фреймворку TensorFlow для побудови нейронної мережі, а також FastApi, HTML5, CSS та JavaScript для розробки веб-застосунку.

TensorFlow - це комплексна платформа для машинного навчання з відкритим вихідним кодом. Вона була розроблена командою Google Brain як продовження закритої системи машинного навчання DistBelief. [9]

FastAPI – це фреймворк для створення лаконічних та досить швидких HTTP API-серверів із вбудованою валідацією, серіалізацією та асинхронністю, що називається, із коробки. Він збудований на базі двух інших фреймворків: роботою з web у FastAPI займається Starlette, а за валідацію відповідає Pydantic. [4][8]

HTML-документ - це звичайний текстовий документ, може бути створений як у звичайному текстовому редакторі (Блокнот), так і в спеціалізованому з підсвічуванням коду (Notepad++, Visual Studio Code і т.п.). HTML-документ має розширення “.html”.

HTML-документ складається з дерева HTML-елементів та тексту. Кожен елемент позначається у вихідному документі початковим (відкриває) і кінцевим (закриває) тегом (за рідкісним винятком).

CSS (Cascading Style Sheets) — мова таблиць стилів, яка дозволяє прикріплювати стиль (наприклад, шрифти та колір) до структурованих документів (наприклад, документів HTML та додатків XML). [2]

Зазвичай CSS-стилі використовуються для створення та зміни стилю елементів веб-сторінок та інтерфейсів користувача, написаних на мовах HTML і XHTML, але також можуть бути застосовані до будь-якого виду XML-документа, у тому числі XML, SVG і XUL.

JavaScript створювався як скриптова мова для браузерів Netscape. Компанія Microsoft також визнала його потенціал і включила під ім'ям JScript в Internet Explorer 3, забезпечивши часткову підтримку стандартів мови, що призвело до безладу зі стандартами і версіями JavaScript. Тому Netscape, Microsoft та інші зацікавлені компанії звернулися до організації ECMA (Європейська асоціація виробників комп'ютерів), де було схвалено першу специфікацію мови ECMA-262. У зв'язку з тим, що назва «JavaScript» була зареєстрованим товарним знаком, для нового стандарту було вирішено використовувати ECMAScript (або скорочено ES). ECMAScript спочатку був розроблений для використання як мова сценаріїв, але пізніше став широко використовуватися як мова програмування загального призначення.

3.2 Структура програмного забезпечення

Сайт будує себе самостійно завдяки фреймворку FastApi. Існує сторінки .html “Головна сторінка” яка відображає таблицю з рейтингом, “Налаштування” яка відображає сторінку з налаштуваннями, “Вхід” яка відображає форму для логіну, Розмітка включає в собі три файли на мові таблиць стилів CSS: reset.css, fonts.css, style.css та скрипт для відправлення запитів JavaScript: script.js.

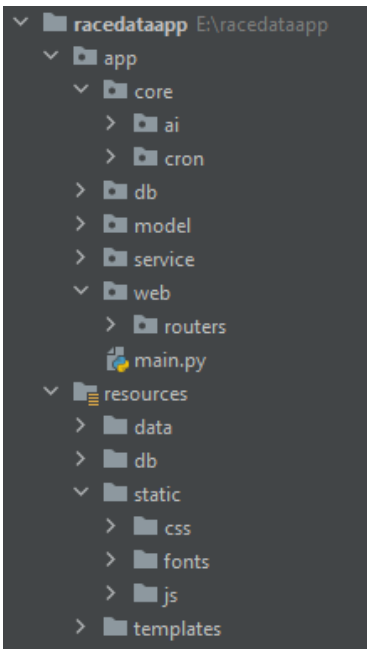


Рисунок 12 – Файлова структура проекту

Таблиця 4 - Специфікація програмних модулів користувача

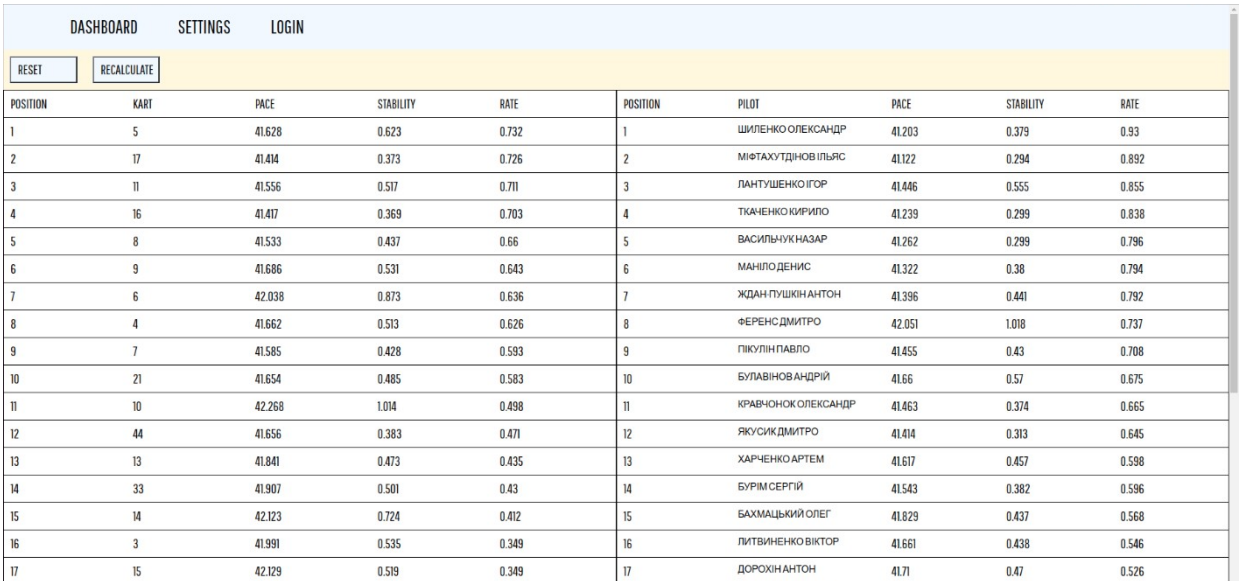
Модуль	Опис
Головна сторінка	Початковий модуль, на якому знаходиться весь основний контент сайту, який відображає таблицю з інформацією про пілотів та картів. Вхідна інформація: данні про пілотів та картів.
Налаштування	Модуль, на якому знаходиться контент для зміни інформації про пілотів та картів, який відображає форму для зміни інформації про пілотів або картів. Вихідна інформація: форма для зміни інформації про пілотів або картів.
Логін	Модуль для логіну на сайт. Вихідна інформація: форма для логіну на сайт.
script.js	Модуль для відправлення запитів на пошук у базі даних.

Таблиця 5 - Специфікація програмних модулів системи

Модуль	Опис
core	Модуль, який відповідає за аналіз статистики способом створення нейронної мережі та оновлення актуальної інформації.
db	Модуль, на якому знаходиться функціонал зв'язку з базою даних та моделі сутностей які зберігаються в базу даних
model	Модуль для в якому описані моделі сутностей.
service	Модуль який виконує маніпуляції з сутностями.
web	Модуль для відправлення відповідей на запити клієнта.
main.py	Основний модуль програми який збирає та керує попередніми модулями

3.3 Керівництво користувача

Головна сторінка - відповідає за основний контент сайту, який відображає таблицю з інформацією про пілотів та картів. (рис. 13).



POSITION	KART	PACE	STABILITY	RATE
1	5	41.628	0.623	0.732
2	17	41.414	0.373	0.726
3	11	41.556	0.517	0.711
4	16	41.417	0.369	0.703
5	8	41.533	0.437	0.66
6	9	41.686	0.531	0.643
7	6	42.038	0.873	0.636
8	4	41.662	0.513	0.626
9	7	41.585	0.428	0.593
10	21	41.654	0.485	0.583
11	10	42.268	1.014	0.498
12	44	41.656	0.383	0.471
13	13	41.841	0.473	0.435
14	33	41.907	0.501	0.43
15	14	42.123	0.724	0.412
16	3	41.991	0.535	0.349
17	15	42.129	0.519	0.349

POSITION	PILOT	PACE	STABILITY	RATE
1	ШИЛЕНКО ОЛЕКСАНДР	41.203	0.379	0.93
2	МІФТАХУТДІНОВ ІЛЬЯС	41.122	0.294	0.892
3	ЛАНТУШЕНКО ІГОР	41.446	0.555	0.855
4	ТКАЧЕНКО КИРИЛО	41.239	0.299	0.838
5	ВАСИЛЬЧУК НАЗАР	41.262	0.299	0.796
6	МАНІЛО ДЕНИС	41.322	0.38	0.794
7	ЖДАН-ПУШКІН АНТОН	41.396	0.441	0.792
8	ФЕРЕНС ДМИТРО	42.051	1.018	0.737
9	ПІКУЛІН ПАВЛО	41.455	0.43	0.708
10	БУЛАВІНОВ АНДРІЙ	41.66	0.57	0.675
11	КРАВЧОНОК ОЛЕКСАНДР	41.463	0.374	0.665
12	ЯКУСИК ДМИТРО	41.414	0.313	0.645
13	ХАРЧЕНКО АРТЕМ	41.617	0.457	0.598
14	БУРІМ СЕРГІЙ	41.543	0.382	0.596
15	БАХМАЦЬКИЙ ОЛЕГ	41.829	0.437	0.568
16	ЛИТВИНЕНКО ВІКТОР	41.661	0.438	0.546
17	ДОРОХІН АНТОН	41.71	0.47	0.526

Рисунок 13 – Головна сторінка.

Налаштування - знаходиться контент для зміни інформації про пілотів та картів, який відображає форму для зміни інформації про пілотів або картів. (рис. 14).

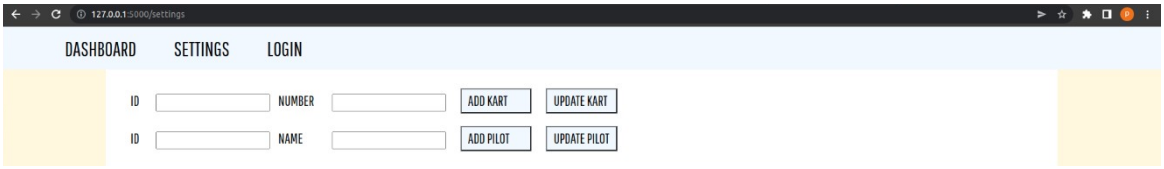


Рисунок 14 – Налаштування.

Логін - основний файл який відповідає форми для входу на сайт. (рис. 15).

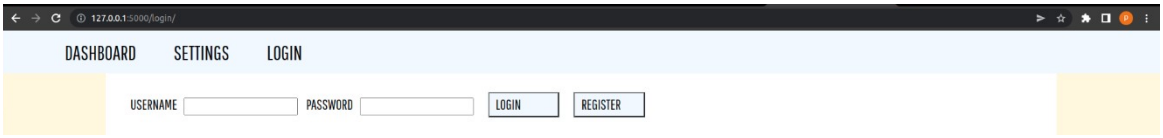


Рисунок 15 – Логін.

3.4 Висновки до третього розділу

Для безпосередньої реалізації додатку для аналізу статистики перегонів на картах.

Було визначено структуру програмного застосунку, побудовано її у вигляді схеми та описано специфікацію програмних модулів у вигляді таблиці. Також було представлено навігацію по застосунку та основний принцип його роботи.

ВИСНОВОК

Збір та аналіз статистики є однією з найважливіших складових сучасних перегонів. Враховуючи вплив змагань на галузь в якій вони проходять я вважаю що продовжуючи цю справу та покращуючи свої навички я згодом зможу вносити зміни в сучасну галузь автомобілебудування як в світі так і в Україні.

В ході роботи було проведено аналіз перегонів та розробка веб-застосунку за допомогою всіх навичок отриманих в університеті, а саме:

Для розробки даного застосунку було спроектовано базу даних, дерево функцій, архітектуру додатку, його структуру.

В результаті реалізації було створено навчальну інформаційну систему для хімії з використанням таких інструментів, як Python, TensorFlow, FastApi, HTML5, CSS та JavaScript.

Розроблений додаток можна покращувати та доповнювати новими функціями у майбутньому.

ВИКОРИСТАНІ ДЖЕРЕЛА

1. Python Introduction [Електронний ресурс] / Режим доступу: https://www.w3schools.com/python/python_intro.asp

2. Diana Bondrenko HtmlSchool [Електронний ресурс] / Режим доступу: https://github.com/DianaBond/course-work/blob/main/pos-page1.html

3. Нейронні мережі [Електронний ресурс] / Режим доступу: https://habr.com/ru/post/144881/

4. Документація FastApi [Електронний ресурс] / Режим доступу: https://fastapi.tiangolo.com/

5. Картинг Центр «ЖАГА ШВИДКОСТІ» [Електронний ресурс] / Режим доступу: https://www.karting.ua/

6. Що таке діджиталізація? [Електронний ресурс] / Режим доступу: https://evergreens.com.ua/ua/articles/business-digitalization.html

7. What is Python? Executive Summary [Електронний ресурс] / Режим доступу: https://www.python.org/doc/essays/blurb/

8. Документація Jinja2 [Електронний ресурс] / Режим доступу: https://jinja.palletsprojects.com/en/3.1.x/

9. Документація Keras [Електронний ресурс] / Режим доступу: https://keras.io/guides/

10. Леоненков А.В. Объектно-ориентированный анализ и проектирование с использованием UML и IBM Rational Rose: учебное пособие. – М.: БИНОМ. Лаборатория знаний, 2006. – 320 с

11. Ф.И.Андон и др., Основы инженерии качества программных систем.- Издательский Дом «Академпериодика»,2007. – 673 с

12. Басс, Л. Архитектура программного обеспечения на практике / Л. Басс, П. Клементс, П. Кацман. – 2-е изд. – СПб. : Питер, 2006. – 575 с.

13. Уоссермен. Нейрокомпьютерная техника: Теория и практика Перевод на русский язык, Хайкин С. Нейронные сети: полный курс, 2-е изд., испр.: Пер. с англ. – М.: ООО «И.Д. Вильямс», 2006. – 1104 с.

14. Томас Кормен, Чарльз Лейзерсон, Рональд Ривест, Клиффорд Штайн. Introduction to Algorithms. — 2nd Edition. — MIT Press, McGraw-Hill, 2001

15. Гайна Г.А. Основи проектування баз даних: Навчальний посібник. К.:КНУБА, 2005. – 204 с

16. Маклаков С.В. – Создание информационных систем с ALLFusionModelingSuite». – М. : Диалог – МИФИ, 2005 – 427 с.

ДОДАТОК 1 - ЛІСТИНГ КОДУ

/core/ai/perceptron.py

```
import keras
import numpy as np
from keras.layers import Dense, Input
from numpy import array



def train_perceptron(db):
    file = open("../resources/data/data.csv")
    numpy_array = np.loadtxt(file, delimiter=",")
    training_inputs = []
    training_outputs = []
    for element in numpy_array:
        training_inputs.append([element[0], element[1]])
        training_outputs.append(element[2])
    training_inputs = array(training_inputs)
    training_outputs = array(training_outputs)


    model = keras.Sequential()


    model.add(Input(shape=(2,)))
    model.add(Dense(units=1, activation='sigmoid'))
    model.compile(loss='mean_squared_error', optimizer="adam")


    print(model.layers)
    history = model.fit(training_inputs, training_outputs, epochs=20, verbose=0)


    weights = history.get_weights()
    db.add(weights)
    db.commit()
    db.refresh(weights)
```

/core/cron/timing_cron.py

```
import requests
from fastapi import Depends
from fastapi_utils.tasks import repeat_every
from sqlalchemy.orm import Session

from app.db.connection_pool import get_session
from app.service import data_service



@repeat_every(seconds=1)
async def get_json():
    json = requests.get("http://nfs.playwar.com:3333/getmaininfo.json").json()



@repeat_every(seconds=60)
async def recalculate_data(db: Session = Depends(get_session)):
    data_service.recalculate(db)
```

/db/connection_pool.py

```
from sqlalchemy import create_engine
from sqlalchemy.orm import sessionmaker

engine = create_engine(
    "postgresql://postgres:postgres@localhost:5432/rda"
)

Session = sessionmaker(
    engine,
    autocommit=False,
    autoflush=False
)



def get_session():
    session = Session()
    try:
        yield session
    finally:
        session.close()
```

/db/models.py

```
from sqlalchemy import Column, Integer, Float, String
from sqlalchemy.ext.declarative import declarative_base

Base = declarative_base()



class Pilot(Base):
    __tablename__ = "pilots"
    id = Column(Integer, primary_key=True)
    name = Column(String, unique=True)
    stability = Column(Float)
    pace = Column(Float)
    rate = Column(Float)



class Kart(Base):
    __tablename__ = "karts"
    id = Column(Integer, primary_key=True)
    number = Column(Integer, unique=True)
    stability = Column(Float)
    pace = Column(Float)
    rate = Column(Float)



class Lap(Base):
    __tablename__ = "laps"
    id = Column(Integer, primary_key=True)
    number = Column(Integer, unique=True)
    time = Column(Float)
    pilot_id = Column(Integer)
    kart_id = Column(Integer)



class Stint(Base):
```

```
    __tablename__ = "stints"
    id = Column(Integer, primary_key=True)
    best_lap = Column(Float)
    pilot_id = Column(Integer)
    kart_id = Column(Integer)
```

/model/kart.py

```
from typing import List

from pydantic import BaseModel

from app.model.lap import Lap



class Kart(BaseModel):
    id: int
    number: str
    stability: float = 0
    pace: float = 0
    rate: float = 0.5
    laps: List[Lap]
    points: int
```

/model/pilot.py

```
from typing import List

from pydantic import BaseModel

from app.model.lap import Lap



class Pilot(BaseModel):
    id: int
    name: str
    stability: float = 0
    pace: float = 0
    rate: float = 0
    laps: List[Lap]
```

/model/stint.py

```
from pydantic import BaseModel

from app.model.kart import Kart
from app.model.lap import Pilot



class Stint(BaseModel):
    id: int
    best_lap: float
    pilot: Pilot
    kart: Kart
```

/service/kart_service.py

```
from app.db import models
```

```
def get_karts(db):
    return db.query(models.Kart).order_by(models.Kart.rate).all()



def create_kart(form, db):
    db_kart = models.Kart(
        number=form.number,
        stability=form.stability,
        pace=form.pace,
        rate=form.rate)

    db.add(db_kart)
    db.commit()
    db.refresh(db_kart)



def update_kart(form, db):
    db_kart = db.query(models.Kart).filter(models.Kart.id == form.id).first()
    db_kart.number = form.number
    db_kart.stability = form.stability
    db_kart.pace = form.pace
    db_kart.rate = form.rate
    db.commit()
    db.refresh(db_kart)
```

/service/pilot_service.py

```
from app.db import models



def get_pilots(db):
    return db.query(models.Pilot).order_by(models.Pilot.rate).all()



def create_pilot(form, db):
    db_pilot = models.Pilot(
        name=form.name,
        stability=form.stability,
        pace=form.pace,
        rate=form.rate)

    db.add(db_pilot)
    db.commit()
    db.refresh(db_pilot)



def update_pilot(form, db):
    db_pilot = db.query(models.Pilot).filter(models.Pilot.id == form.id).first()
    db_pilot.name = form.name
    db_pilot.stability = form.stability
    db_pilot.pace = form.pace
    db_pilot.rate = form.rate
    db.commit()
    db.refresh(db_pilot)
```

/web/routers/home.py

```
from fastapi import APIRouter, Request, Depends
from fastapi.templating import Jinja2Templates
from sqlalchemy.orm import Session

from ...core.ai.perceptron import train_perceptron
from ...db.connection_pool import get_session
from ...service.karts_service import get_karts
from ...service.pilots_service import get_pilots

router = APIRouter()
templates = Jinja2Templates(directory="../resources/templates")



@router.get("/")
async def home(request: Request, db: Session = Depends(get_session)):
    # train_perceptron(db)
    karts = get_karts(db)
    pilots = get_pilots(db)
    return templates.TemplateResponse("dashboard.html", {
        "request": request,
        "karts": karts,
        "pilots": pilots
    })
```

/web/routers/login.py

```
from fastapi import APIRouter, Request, Depends
from fastapi.templating import Jinja2Templates
from sqlalchemy.orm import Session

from ...db.connection_pool import get_session

router = APIRouter()
templates = Jinja2Templates(directory="../resources/templates")



@router.get("/")
async def login(request: Request, db: Session = Depends(get_session)):
    return templates.TemplateResponse("login.html", {
        "request": request,
    })
```

/web/routers/register.py

```
from fastapi import APIRouter, Request, Depends
from fastapi.templating import Jinja2Templates
from sqlalchemy.orm import Session

from ...db.connection_pool import get_session

router = APIRouter()
templates = Jinja2Templates(directory="../resources/templates")



@router.get("/")
async def register(request: Request, db: Session = Depends(get_session)):
    return templates.TemplateResponse("register.html", {
```

```
        "request": request,
    })
```

/web/routers/router.py

```
from fastapi import APIRouter

from app.web.routers import home, settings, login, register

router = APIRouter()

router.include_router(home.router, prefix="")
router.include_router(settings.router, prefix="/settings")
router.include_router(login.router, prefix="/login")
router.include_router(register.router, prefix="/register")
```

/web/routers/settings.py

```
from fastapi import APIRouter, Request, Depends
from fastapi.templating import Jinja2Templates
from sqlalchemy.orm import Session

from app.service import pilots_service, karts_service, data_service
from ...db.connection_pool import get_session

router = APIRouter()
templates = Jinja2Templates(directory="../resources/templates")



@router.get("")
async def settings(request: Request):
    return templates.TemplateResponse("settings.html", {
        "request": request
    })



@router.put("/recalculate")
async def settings(request: Request, db: Session = Depends(get_session)):
    data_service.recalculate(db)
    return templates.TemplateResponse("settings.html", {
        "request": request
    })



@router.put("/reset")
async def settings(request: Request, db: Session = Depends(get_session)):
    data_service.reset(db)
    return templates.TemplateResponse("settings.html", {
        "request": request
    })



@router.post("/pilots")
async def add_pilot(request: Request, db: Session = Depends(get_session)):
    form = request.form()
    pilots_service.create_pilot(form, db)
    return templates.TemplateResponse("settings.html", {
        "request": request
```

```
    })


@router.put("/pilots/{id}")
async def delete_pilot(request: Request, db: Session = Depends(get_session)):
    form = request.form()
    pilots_service.update_pilot(form, db)
    return templates.TemplateResponse("settings.html", {
        "request": request
    })



@router.post("/karts")
async def add_kart(request: Request, db: Session = Depends(get_session)):
    form = request.form()
    karts_service.create_kart(form, db)
    return templates.TemplateResponse("settings.html", {
        "request": request
    })



@router.put("/karts/{id}")
async def delete_kart(request: Request, db: Session = Depends(get_session)):
    form = request.form()
    karts_service.update_kart(form, db)
    return templates.TemplateResponse("settings.html", {
        "request": request
    })
```

main.py

```
import uvicorn
from fastapi import FastAPI
from fastapi.staticfiles import StaticFiles

from app.core.cron.timing_cron import recalculate_data
from app.web.routers.router import router



def get_application() -> FastAPI:
    application = FastAPI()
    application.include_router(router)
    application.add_event_handler("startup", recalculate_data)
    application.mount("/static", StaticFiles(directory="../resources/static"),
name="static")
    return application



app = get_application()

if __name__ == "__main__":
    uvicorn.run(app, host="127.0.0.1", port=5000)
```

/resourses/db/V1.0__Init.sql

```
SELECT 1
```

/resourses/db/V1.1__Create_base_version.sql

```
CREATE TABLE pilots
(
    id        SERIAL PRIMARY KEY,
    name      TEXT  NOT NULL,
    stability FLOAT NOT NULL DEFAULT 0,
    pace      FLOAT NOT NULL DEFAULT 0,
    rate      FLOAT NOT NULL DEFAULT 0.5
);

CREATE TABLE karts
(
    id        SERIAL PRIMARY KEY,
    number    INTEGER NOT NULL,
    stability FLOAT   NOT NULL DEFAULT 0,
    pace      FLOAT   NOT NULL DEFAULT 0,
    rate      FLOAT   NOT NULL DEFAULT 0.5
);

CREATE TABLE laps
(
    id       SERIAL PRIMARY KEY,
    number   INTEGER NOT NULL,
    time     FLOAT   NOT NULL,
    pilot_id INTEGER REFERENCES pilots (id),
    kart_id  INTEGER REFERENCES karts (id)
);

CREATE TABLE stints
(
    id        SERIAL PRIMARY KEY,
    best_lap FLOAT NOT NULL,
    pilot_id  INTEGER REFERENCES pilots (id),
    kart_id   INTEGER REFERENCES karts (id)
);
```

/resourses/db/V1.2__Populate_base_version.sql

```
INSERT INTO karts (number) VALUES (1);
INSERT INTO karts (number) VALUES (2);
INSERT INTO karts (number) VALUES (3);
INSERT INTO karts (number) VALUES (4);
INSERT INTO karts (number) VALUES (5);
INSERT INTO karts (number) VALUES (6);
INSERT INTO karts (number) VALUES (7);
INSERT INTO karts (number) VALUES (8);
INSERT INTO karts (number) VALUES (9);
INSERT INTO karts (number) VALUES (10);
INSERT INTO karts (number) VALUES (11);
INSERT INTO karts (number) VALUES (12);
INSERT INTO karts (number) VALUES (13);
INSERT INTO karts (number) VALUES (14);
INSERT INTO karts (number) VALUES (15);
INSERT INTO karts (number) VALUES (16);
INSERT INTO karts (number) VALUES (17);
INSERT INTO karts (number) VALUES (18);
INSERT INTO karts (number) VALUES (21);
INSERT INTO karts (number) VALUES (44);
INSERT INTO karts (number) VALUES (69);
INSERT INTO karts (number) VALUES (33);
```

```
INSERT INTO pilots (name) VALUES ('Єлманов Сергій');
INSERT INTO pilots (name) VALUES ('Слащов Руслан');
INSERT INTO pilots (name) VALUES ('Голоцван Дмитро');
INSERT INTO pilots (name) VALUES ('Поліщук Спартак');
INSERT INTO pilots (name) VALUES ('Гетьманцев Данило');
INSERT INTO pilots (name) VALUES ('Башинський Богдан');
INSERT INTO pilots (name) VALUES ('Бегішев Максим');
INSERT INTO pilots (name) VALUES ('Кригін Ілля');
INSERT INTO pilots (name) VALUES ('Кравченко Дмитро');
INSERT INTO pilots (name) VALUES ('Рульов Гліб');
INSERT INTO pilots (name) VALUES ('Якусик Саша');
INSERT INTO pilots (name) VALUES ('Карлюк Марина');
INSERT INTO pilots (name) VALUES ('Хижняк Антон');
INSERT INTO pilots (name) VALUES ('Черненко Дмитро');
INSERT INTO pilots (name) VALUES ('Лихошерст Олексій');
INSERT INTO pilots (name) VALUES ('Нужний Олексій');
INSERT INTO pilots (name) VALUES ('Литвиненко Віктор');
INSERT INTO pilots (name) VALUES ('Чуб Дмитро');
INSERT INTO pilots (name) VALUES ('Дорохін Антон');
INSERT INTO pilots (name) VALUES ('Булавінов Андрій');
INSERT INTO pilots (name) VALUES ('Бурім Сергій');
INSERT INTO pilots (name) VALUES ('Пашковський Павло');
INSERT INTO pilots (name) VALUES ('Ференс Дмитро');
INSERT INTO pilots (name) VALUES ('Ждан-Пушкін Антон');
INSERT INTO pilots (name) VALUES ('Кравчонок Олександр');
INSERT INTO pilots (name) VALUES ('Харченко Артем');
INSERT INTO pilots (name) VALUES ('Пікулін Павло');
INSERT INTO pilots (name) VALUES ('Бахмацький Олег');
INSERT INTO pilots (name) VALUES ('Танцюра Денис');
INSERT INTO pilots (name) VALUES ('Маніло Денис');
INSERT INTO pilots (name) VALUES ('Лантушенко Ігор');
INSERT INTO pilots (name) VALUES ('Ткаченко Кирило');
INSERT INTO pilots (name) VALUES ('Міфтахутдінов Ільяс');
INSERT INTO pilots (name) VALUES ('Якусик Дмитро');
INSERT INTO pilots (name) VALUES ('Васильчук Назар');
INSERT INTO pilots (name) VALUES ('Шиленко Олександр');
```

/resourses/db/V1.3__Populate_stints.sql

```
INSERT INTO stints (pilot_id, kart_id, best_lap) VALUES (3,2,41.706
);INSERT INTO stints (pilot_id, kart_id, best_lap) VALUES (4,1,42.102
);INSERT INTO stints (pilot_id, kart_id, best_lap) VALUES (3,4,41.407
);INSERT INTO stints (pilot_id, kart_id, best_lap) VALUES (4,1,42.056
);INSERT INTO stints (pilot_id, kart_id, best_lap) VALUES (3,14,41.889
);INSERT INTO stints (pilot_id, kart_id, best_lap) VALUES (4,1,42.391
);INSERT INTO stints (pilot_id, kart_id, best_lap) VALUES (3,18,42.015
);INSERT INTO stints (pilot_id, kart_id, best_lap) VALUES (4,21,42.230
);INSERT INTO stints (pilot_id, kart_id, best_lap) VALUES (5,11,41.274
);INSERT INTO stints (pilot_id, kart_id, best_lap) VALUES (5,14,41.861
);INSERT INTO stints (pilot_id, kart_id, best_lap) VALUES (6,1,41.855
);INSERT INTO stints (pilot_id, kart_id, best_lap) VALUES (5,3,41.707
);INSERT INTO stints (pilot_id, kart_id, best_lap) VALUES (6,21,41.683
);INSERT INTO stints (pilot_id, kart_id, best_lap) VALUES (5,1,42.432
);INSERT INTO stints (pilot_id, kart_id, best_lap) VALUES (5,6,41.659
);INSERT INTO stints (pilot_id, kart_id, best_lap) VALUES (6,3,41.527
);INSERT INTO stints (pilot_id, kart_id, best_lap) VALUES (7,21,41.616
);INSERT INTO stints (pilot_id, kart_id, best_lap) VALUES (8,19,41.473
);INSERT INTO stints (pilot_id, kart_id, best_lap) VALUES (7,12,42.685
```

```
);INSERT INTO stints (pilot_id, kart_id, best_lap) VALUES (8,5,41.067
);INSERT INTO stints (pilot_id, kart_id, best_lap) VALUES (7,4,41.666
);INSERT INTO stints (pilot_id, kart_id, best_lap) VALUES (8,2,41.713
);INSERT INTO stints (pilot_id, kart_id, best_lap) VALUES (7,9,41.491
);INSERT INTO stints (pilot_id, kart_id, best_lap) VALUES (8,15,41.329
);INSERT INTO stints (pilot_id, kart_id, best_lap) VALUES (9,8,41.310
);INSERT INTO stints (pilot_id, kart_id, best_lap) VALUES (10,6,41.879
);INSERT INTO stints (pilot_id, kart_id, best_lap) VALUES (9,21,41.483
);INSERT INTO stints (pilot_id, kart_id, best_lap) VALUES (10,10,41.693
);INSERT INTO stints (pilot_id, kart_id, best_lap) VALUES (9,20,41.358
);INSERT INTO stints (pilot_id, kart_id, best_lap) VALUES (10,5,41.325
);INSERT INTO stints (pilot_id, kart_id, best_lap) VALUES (9,3,41.713
);INSERT INTO stints (pilot_id, kart_id, best_lap) VALUES (10,18,42.003
);INSERT INTO stints (pilot_id, kart_id, best_lap) VALUES (11,1,41.418
);INSERT INTO stints (pilot_id, kart_id, best_lap) VALUES (12,5,41.566
);INSERT INTO stints (pilot_id, kart_id, best_lap) VALUES (11,20,41.265
);INSERT INTO stints (pilot_id, kart_id, best_lap) VALUES (12,9,41.900
);INSERT INTO stints (pilot_id, kart_id, best_lap) VALUES (11,19,41.344
);INSERT INTO stints (pilot_id, kart_id, best_lap) VALUES (12,22,42.037
);INSERT INTO stints (pilot_id, kart_id, best_lap) VALUES (11,20,41.204
);INSERT INTO stints (pilot_id, kart_id, best_lap) VALUES (12,13,41.954
);INSERT INTO stints (pilot_id, kart_id, best_lap) VALUES (13,17,41.179
);INSERT INTO stints (pilot_id, kart_id, best_lap) VALUES (14,2,41.497
);INSERT INTO stints (pilot_id, kart_id, best_lap) VALUES (13,14,41.392
);INSERT INTO stints (pilot_id, kart_id, best_lap) VALUES (14,20,41.478
);INSERT INTO stints (pilot_id, kart_id, best_lap) VALUES (13,18,41.527
);INSERT INTO stints (pilot_id, kart_id, best_lap) VALUES (14,10,41.268
);INSERT INTO stints (pilot_id, kart_id, best_lap) VALUES (13,7,41.154
);INSERT INTO stints (pilot_id, kart_id, best_lap) VALUES (14,2,41.524
);INSERT INTO stints (pilot_id, kart_id, best_lap) VALUES (15,15,42.229
);INSERT INTO stints (pilot_id, kart_id, best_lap) VALUES (15,12,41.531
);INSERT INTO stints (pilot_id, kart_id, best_lap) VALUES (16,22,41.171
);INSERT INTO stints (pilot_id, kart_id, best_lap) VALUES (15,2,41.146
);INSERT INTO stints (pilot_id, kart_id, best_lap) VALUES (16,1,41.590
);INSERT INTO stints (pilot_id, kart_id, best_lap) VALUES (16,11,40.981
);INSERT INTO stints (pilot_id, kart_id, best_lap) VALUES (15,21,41.265
);INSERT INTO stints (pilot_id, kart_id, best_lap) VALUES (16,4,41.241
);INSERT INTO stints (pilot_id, kart_id, best_lap) VALUES (17,16,40.956
);INSERT INTO stints (pilot_id, kart_id, best_lap) VALUES (18,21,41.689
);INSERT INTO stints (pilot_id, kart_id, best_lap) VALUES (17,3,41.577
);INSERT INTO stints (pilot_id, kart_id, best_lap) VALUES (18,22,41.393
);INSERT INTO stints (pilot_id, kart_id, best_lap) VALUES (17,17,41.194
);INSERT INTO stints (pilot_id, kart_id, best_lap) VALUES (18,20,41.312
);INSERT INTO stints (pilot_id, kart_id, best_lap) VALUES (17,11,41.165
);INSERT INTO stints (pilot_id, kart_id, best_lap) VALUES (18,20,41.273
);INSERT INTO stints (pilot_id, kart_id, best_lap) VALUES (19,5,40.925
);INSERT INTO stints (pilot_id, kart_id, best_lap) VALUES (20,11,40.956
);INSERT INTO stints (pilot_id, kart_id, best_lap) VALUES (19,13,41.206
);INSERT INTO stints (pilot_id, kart_id, best_lap) VALUES (20,8,40.946
);INSERT INTO stints (pilot_id, kart_id, best_lap) VALUES (19,3,41.572
);INSERT INTO stints (pilot_id, kart_id, best_lap) VALUES (20,2,41.347
);INSERT INTO stints (pilot_id, kart_id, best_lap) VALUES (19,16,41.257
);INSERT INTO stints (pilot_id, kart_id, best_lap) VALUES (20,11,41.109
);INSERT INTO stints (pilot_id, kart_id, best_lap) VALUES (21,13,41.168
);INSERT INTO stints (pilot_id, kart_id, best_lap) VALUES (21,9,40.950
);INSERT INTO stints (pilot_id, kart_id, best_lap) VALUES (22,7,41.430
);INSERT INTO stints (pilot_id, kart_id, best_lap) VALUES (22,19,41.581
);INSERT INTO stints (pilot_id, kart_id, best_lap) VALUES (21,15,41.099
);INSERT INTO stints (pilot_id, kart_id, best_lap) VALUES (22,7,41.414
);INSERT INTO stints (pilot_id, kart_id, best_lap) VALUES (21,13,41.428
);INSERT INTO stints (pilot_id, kart_id, best_lap) VALUES (22,8,41.367
```

```
);INSERT INTO stints (pilot_id, kart_id, best_lap) VALUES (23,3,41.013
);INSERT INTO stints (pilot_id, kart_id, best_lap) VALUES (24,20,41.020
);INSERT INTO stints (pilot_id, kart_id, best_lap) VALUES (23,6,41.048
);INSERT INTO stints (pilot_id, kart_id, best_lap) VALUES (24,14,41.087
);INSERT INTO stints (pilot_id, kart_id, best_lap) VALUES (24,9,41.018
);INSERT INTO stints (pilot_id, kart_id, best_lap) VALUES (23,19,41.090
);INSERT INTO stints (pilot_id, kart_id, best_lap) VALUES (24,5,40.694
);INSERT INTO stints (pilot_id, kart_id, best_lap) VALUES (23,6,40.982
);INSERT INTO stints (pilot_id, kart_id, best_lap) VALUES (25,9,40.914
);INSERT INTO stints (pilot_id, kart_id, best_lap) VALUES (26,10,41.327
);INSERT INTO stints (pilot_id, kart_id, best_lap) VALUES (25,17,41.055
);INSERT INTO stints (pilot_id, kart_id, best_lap) VALUES (26,16,41.011
);INSERT INTO stints (pilot_id, kart_id, best_lap) VALUES (25,7,41.102
);INSERT INTO stints (pilot_id, kart_id, best_lap) VALUES (26,6,41.143
);INSERT INTO stints (pilot_id, kart_id, best_lap) VALUES (25,14,41.193
);INSERT INTO stints (pilot_id, kart_id, best_lap) VALUES (25,16,41.180
);INSERT INTO stints (pilot_id, kart_id, best_lap) VALUES (27,7,41.025
);INSERT INTO stints (pilot_id, kart_id, best_lap) VALUES (28,14,41.167
);INSERT INTO stints (pilot_id, kart_id, best_lap) VALUES (27,16,40.736
);INSERT INTO stints (pilot_id, kart_id, best_lap) VALUES (28,15,42.547
);INSERT INTO stints (pilot_id, kart_id, best_lap) VALUES (27,7,41.132
);INSERT INTO stints (pilot_id, kart_id, best_lap) VALUES (28,5,40.830
);INSERT INTO stints (pilot_id, kart_id, best_lap) VALUES (27,14,41.207
);INSERT INTO stints (pilot_id, kart_id, best_lap) VALUES (28,22,41.024
);INSERT INTO stints (pilot_id, kart_id, best_lap) VALUES (29,4,41.026
);INSERT INTO stints (pilot_id, kart_id, best_lap) VALUES (30,15,41.211
);INSERT INTO stints (pilot_id, kart_id, best_lap) VALUES (29,2,41.391
);INSERT INTO stints (pilot_id, kart_id, best_lap) VALUES (30,17,40.929
);INSERT INTO stints (pilot_id, kart_id, best_lap) VALUES (29,16,41.323
);INSERT INTO stints (pilot_id, kart_id, best_lap) VALUES (30,17,40.996
);INSERT INTO stints (pilot_id, kart_id, best_lap) VALUES (29,15,41.246
);INSERT INTO stints (pilot_id, kart_id, best_lap) VALUES (30,5,40.630
);INSERT INTO stints (pilot_id, kart_id, best_lap) VALUES (31,19,40.675
);INSERT INTO stints (pilot_id, kart_id, best_lap) VALUES (32,3,41.085
);INSERT INTO stints (pilot_id, kart_id, best_lap) VALUES (31,19,41.021
);INSERT INTO stints (pilot_id, kart_id, best_lap) VALUES (32,8,40.870
);INSERT INTO stints (pilot_id, kart_id, best_lap) VALUES (31,6,40.999
);INSERT INTO stints (pilot_id, kart_id, best_lap) VALUES (32,4,40.942
);INSERT INTO stints (pilot_id, kart_id, best_lap) VALUES (31,16,40.871
);INSERT INTO stints (pilot_id, kart_id, best_lap) VALUES (32,17,40.865
);INSERT INTO stints (pilot_id, kart_id, best_lap) VALUES (33,10,40.729
);INSERT INTO stints (pilot_id, kart_id, best_lap) VALUES (34,17,41.066
);INSERT INTO stints (pilot_id, kart_id, best_lap) VALUES (33,9,40.659
);INSERT INTO stints (pilot_id, kart_id, best_lap) VALUES (34,4,41.089
);INSERT INTO stints (pilot_id, kart_id, best_lap) VALUES (34,11,41.029
);INSERT INTO stints (pilot_id, kart_id, best_lap) VALUES (33,13,41.086
);INSERT INTO stints (pilot_id, kart_id, best_lap) VALUES (34,8,41.219
);INSERT INTO stints (pilot_id, kart_id, best_lap) VALUES (33,7,40.839
);INSERT INTO stints (pilot_id, kart_id, best_lap) VALUES (36,6,40.743
);INSERT INTO stints (pilot_id, kart_id, best_lap) VALUES (35,4,40.964
);INSERT INTO stints (pilot_id, kart_id, best_lap) VALUES (36,8,40.829
);INSERT INTO stints (pilot_id, kart_id, best_lap) VALUES (35,11,40.759
);INSERT INTO stints (pilot_id, kart_id, best_lap) VALUES (36,6,40.868
);INSERT INTO stints (pilot_id, kart_id, best_lap) VALUES (35,8,41.128
);INSERT INTO stints (pilot_id, kart_id, best_lap) VALUES (36,4,40.854
);INSERT INTO stints (pilot_id, kart_id, best_lap) VALUES (35,19,40.999);
```

/resourses /static/css/fonts.css

```
@font-face {
    font-family: 'Dorsa';
    src: url('../fonts/Dorsa-Regular.eot');
```

```
    src: url('../fonts/Dorsa-Regular.eot?#iefix') format('embedded-opentype'),
    url('../fonts/Dorsa-Regular.woff2') format('woff2');
    font-weight: normal;
    font-style: normal;
}

@font-face {
    font-family: 'Open Sans Condensed';
    src: url('../fonts/OpenSansCondensed-Light.eot');
    src: url('../fonts/OpenSansCondensed-Light.eot?#iefix') format('embedded-
opentype'),
    url('../fonts/OpenSansCondensed-Light.woff2') format('woff2'),
    url('../fonts/OpenSansCondensed-Light.ttf') format('truetype');
    font-weight: 300;
    font-style: normal;
}

@font-face {
    font-family: 'Pathway Gothic One';
    src: url('../fonts/PathwayGothicOne-Regular.eot');
    src: url('../fonts/PathwayGothicOne-Regular.eot?#iefix') format('embedded-
opentype'),
    url('../fonts/PathwayGothicOne-Regular.woff2') format('woff2'),
    url('../fonts/PathwayGothicOne-Regular.ttf') format('truetype');
    font-weight: normal;
    font-style: normal;
}
```

/resourses /static/css/reset.css

```
html, body, div, span, applet, object, iframe,
h1, h2, h3, h4, h5, h6, p, blockquote, pre,
a, abbr, acronym, address, big, cite, code,
del, dfn, em, img, ins, kbd, q, s, samp,
small, strike, strong, sub, sup, tt, var,
b, u, i, center,
dl, dt, dd, ol, ul, li,
fieldset, form, label, legend,
table, caption, tbody, tfoot, thead, tr, th, td,
article, aside, canvas, details, embed,
figure, figcaption, footer, header, hgroup,
menu, nav, output, ruby, section, summary,
time, mark, audio, video {
    margin: 0;
    padding: 0;
    border: 0;
    font-size: 100%;
    font: inherit;
    vertical-align: baseline;;
}

article, aside, details, figcaption, figure,
footer, header, hgroup, menu, nav, section {
    display: block;
}

body {
    line-height: 1;
}
```

```
ol, ul {
    list-style: none;
}

blockquote, q {
    quotes: none;
}

blockquote:before, blockquote:after,
q:before, q:after {
    content: '';
    content: none;
}

table {
    border-collapse: collapse;
    border-spacing: 0;
}
```

/resourses /static/css/style.css

```
body {
    background: #fff8dc no-repeat center;
    background-size: cover;
    font-family: 'Pathway Gothic One', sans-serif;
    color: #000;
}

ul {
    list-style: none;
}

a {
    text-decoration: none;
    color: #000;
}

.wrapper {
    width: 100%;
    margin: 0;
}

header {
    width: 100%;
    height: 68px;
    background-color: #f0f8ff;
    font-family: 'Pathway Gothic One', sans-serif;
    font-size: 1.75rem;
    position: relative;
}

nav {
    position: relative;
    text-align: right;
    color: #000;
}

nav > ul {
```

```
    width: 70%;
    padding: 20px 100px;
    display: flex;
}

nav > ul > li {
    text-transform: uppercase;
    display: inline-block;
    margin: 0 0 0 62px;
    float: right;
}

nav > ul > li:nth-child(-n+3) {
    float: left;
    margin: 0 60px 0 0;
}

nav > ul > li > a:hover {
    cursor: pointer;
}

.menu-item {
    position: relative;
}

main {
    font-size: 1.25rem;
    text-transform: uppercase;
    font-family: 'Pathway Gothic One', sans-serif;
}

.home-info {
    background-color: #fff;
    width: 80%;
    margin: 0 auto;
    padding: 20px;
}

.dashboard {
    display: flex;
    align-content: space-between;
    background-color: #fff;
}

table {
    width: 50%;
    padding: 10px;
    border: #000 solid 1px;
}

tr {
    width: 100%;
    height: 40px;
    display: flex;
    justify-content: space-around;
    border-bottom: #000 solid 1px;
}
```

```
td {
    padding: 10px;
    width: 18%;
}

.pilot-name {
    width: 25%;
    font-size: 0.9rem;
}

.table-head {
    font-size: 1.25rem;
}

.settings {
    width: 80%;
    margin: 0 auto;
    padding: 20px;
    background-color: #fff;
}

form {
    margin: auto 20px;
}

input {
    height: 1.25rem;
    margin: auto 10px;
}

label {
    display: inline-block;
    width: 4rem;
}

.id {
    width: 1.5rem;
}

.btn {
    width: 6.3rem;
    display: inline-flex;
    margin: 10px;
    text-decoration: none;
    position: relative;
    font-size: 1.25rem;
    line-height: 20px;
    padding: 8px 10px;
    color: #000000;
    text-align: center;
    text-transform: uppercase;
    font-family: 'Pathway Gothic One', sans-serif;
    background: #f0f8ff;
    cursor: pointer;
    outline: #f0f8ff solid 1px;
    outline-offset: 0;
```

```
    text-shadow: none;
    transition: all 1.5s cubic-bezier(0.19, 1, 0.22, 1);
}
```

/resourses /static/js/script.js

```
const BASE_URL = 'http://127.0.0.1:5000';

const kartIdField = document.querySelector('#kartId');
const numberField = document.querySelector('#number');
const pilotIdField = document.querySelector('#pilotId');
const nameField = document.querySelector('#name');
const swsIdField = document.querySelector('#swsId');
const addKart = document.querySelector('#addKart');
const updateKart = document.querySelector('#updateKart');
const addPilot = document.querySelector('#addPilot');
const updatePilot = document.querySelector('#updatePilot');

addKart.addEventListener('click', () => addKartHandler());
updateKart.addEventListener('click', () => updateKartHandler());
addPilot.addEventListener('click', () => addPilotHandler());
updatePilot.addEventListener('click', () => updatePilotHandler());

const addKartHandler = async () => {
    const kart = {
        number: numberField.value
    };

    try {
        const response = fetch('${BASE_URL}/karts', {
            method: 'post',
            headers: {
                'Content-type': 'application/json'
            },
            body: JSON.stringify({kart})
        });
    } catch (error) {
        console.log(error);
    }
};

const updateKartHandler = async () => {
    const kart = {
        id: kartIdField.value,
        number: numberField.value
    };

    try {
        const response = fetch('${BASE_URL}/karts', {
            method: 'put',
            headers: {
                'Content-type': 'application/json'
            },
            body: JSON.stringify({kart})
        });
    } catch (error) {
        console.log(error);
    }
```

```
};

const addPilotHandler = async () => {
    const pilot = {
        name: nameField.value,
        swsId: swsIdField.value
    };

    try {
        const response = fetch('${BASE_URL}/pilots', {
            method: 'post',
            headers: {
                'Content-type': 'application/json'
            },
            body: JSON.stringify({pilot})
        });
    } catch (error) {
        console.log(error);
    }
};

const updatePilotHandler = async () => {
    const pilot = {
        id: pilotIdField.value,
        name: nameField.value,
        swsId: swsIdField.value
    };

    try {
        const response = fetch('${BASE_URL}/pilots', {
            method: 'put',
            headers: {
                'Content-type': 'application/json'
            },
            body: JSON.stringify({pilot})
        });
    } catch (error) {
        console.log(error);
    }
};

function liveUpdate() {
    const textResult = document.getElementById("result")

    setInterval(function () {
        fetch("/dashboard").then(function (response) {
            return response
        }).then(function () {
            textResult.textContent = {
            {
                result | safe
            }
        }
        }).catch(function (error) {
            console.log(error)
        })
    }, 1000)
}
```

```
document.addEventListener("DOMContentLoaded", function () {
    // liveUpdate()
});
```

/resourses/templates/dashboard.html

```
{% extends "layout.html" %}

{% block content %}
<button type="submit" id="resetData" class="btn">Reset</button>
<button type="submit" id="recalculateData" class="btn">Recalculate</button>
<div class="dashboard">
    <table class="left-table">
        <thead>
        <tr>
            <td>Position</td>
            <td>Kart</td>
            <td>Pace</td>
            <td>Stability</td>
            <td>Rate</td>
        </tr>
        </thead>
        <tbody>
        {%- for kart in karts %}
        <tr>
            <td></td>
            <td>{{ kart.number }}</td>
            <td>{{ kart.pace }}</td>
            <td>{{ kart.stability }}</td>
            <td>{{ kart.rate }}</td>
        </tr>
        {% endfor %}
        </tbody>
    </table>
    <table class="right-table">
        <thead>
        <tr>
            <td>Position</td>
            <td class="pilot-name table-head">Pilot</td>
            <td>Stability</td>
            <td>Pace</td>
            <td>Rate</td>
        </tr>
        </thead>
        <tbody>
        {%- for pilot in pilots %}
        <tr>
            <td></td>
            <td class="pilot-name">{{ pilot.name }}</td>
            <td>{{ pilot.pace }}</td>
            <td>{{ pilot.stability }}</td>
            <td>{{ pilot.rate }}</td>
        </tr>
        {% endfor %}
        </tbody>
    </table>
</div>
{% endblock %}
```

```
{% block footer %}
{% endblock %}
```

/resourses/templates/home.html

```
{% extends "layout.html" %}

{% block content %}
<button type="submit" id="resetData" class="btn">Reset</button>
<button type="submit" id="recalculateData" class="btn">Recalculate</button>
<div class="dashboard">
    <table class="left-table">
        <thead>
        <tr>
            <td>Position</td>
            <td>Kart</td>
            <td>Pace</td>
            <td>Stability</td>
            <td>Rate</td>
        </tr>
        </thead>
        <tbody>
        {%- for kart in karts %}
        <tr>
            <td></td>
            <td>{{ kart.number }}</td>
            <td>{{ kart.pace }}</td>
            <td>{{ kart.stability }}</td>
            <td>{{ kart.rate }}</td>
        </tr>
        {% endfor %}
        </tbody>
    </table>
    <table class="right-table">
        <thead>
        <tr>
            <td>Position</td>
            <td class="pilot-name table-head">Pilot</td>
            <td>Stability</td>
            <td>Pace</td>
            <td>Rate</td>
        </tr>
        </thead>
        <tbody>
        {%- for pilot in pilots %}
        <tr>
            <td></td>
            <td class="pilot-name">{{ pilot.name }}</td>
            <td>{{ pilot.pace }}</td>
            <td>{{ pilot.stability }}</td>
            <td>{{ pilot.rate }}</td>
        </tr>
        {% endfor %}
        </tbody>
    </table>
</div>
{% endblock %}

{% block footer %}
{% endblock %}
```

/resourses/templates/layout.html

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>Title</title>
    <link rel="stylesheet" type="text/css" href="{{ url_for('static',
path='css/reset.css') }}">
    <link rel="stylesheet" type="text/css" href="{{ url_for('static',
path='css/fonts.css') }}">
    <link rel="stylesheet" type="text/css" href="{{ url_for('static',
path='css/style.css') }}">
    <script type="text/javascript" src="{{ url_for('static',
path='js/script.js')}}"></script>
</head>
<body>
<div class="wrapper">
    <header>
        <nav>
            <ul>
                <li>
                    <a href="/">dashboard</a>
                </li>
                <li class="menu-item">
                    <a href="/settings">settings</a>
                </li>
                <li>
                    <a href="/login">login</a>
                </li>
            </ul>
        </nav>
    </header>
    <main>
        {% block content %}
        {% endblock %}
    </main>
    <footer>
        {% block footer %}
        {% endblock %}
    </footer>
</div>
</body>
</html>
```

/resourses/templates/login.html

```
{% extends "layout.html" %}

{% block content %}
<div class="settings">
    <form>
        <label for="username">username</label>
        <input type="text" id="username">
        <label for="password">password</label>
        <input type="text" id="password">
        <button type="submit" id="login" class="btn">login</button>
        <button class="btn"><a href="/register">register</a></button>
    </form>
</div>
{% endblock %}
```

```
{% block footer %}
{% endblock %}
```

/resourses/templates/register.html

```
{% extends "layout.html" %}

{% block content %}
<div class="settings">
    <form>
        <label for="username">username</label>
        <input type="text" id="username">
        <label for="password">password</label>
        <input type="text" id="password">
        <label for="confirmPassword">confirm password</label>
        <input type="text" id="confirmPassword">
        <button type="submit" id="register" class="btn">register</button>
    </form>
</div>
{% endblock %}

{% block footer %}
{% endblock %}
```

/resourses/templates/settings.html

```
{% extends "layout.html" %}

{% block content %}
<div class="settings">
    <form>
        <label for="kart_id" class="id">id</label>
        <input type="number" id="kart_id">
        <label for="number">number</label>
        <input type="text" id="number">
        <button type="submit" id="addKart" class="btn">add kart</button>
        <button type="submit" id="updateKart" class="btn">update kart</button>
    </form>
    <form>
        <label for="pilot_id" class="id">id</label>
        <input type="number" id="pilot_id">
        <label for="name">name</label>
        <input type="text" id="name">
        <button type="submit" id="addPilot" class="btn">add pilot</button>
        <button type="submit" id="updatePilot" class="btn">update pilot</button>
    </form>
</div>
{% endblock %}

{% block footer %}
{% endblock %}
```