ДЕРЖАВНИЙ УНІВЕРСИТЕТ ТЕЛЕКОМУНІКАЦІЙ

НАВЧАЛЬНО–НАУКОВИЙ ІНСТИТУТ ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ

Кафедра інженерії програмного забезпечення

Пояснювальна записка

до магістерської роботи
на ступінь вищої освіти магістр

на тему: «**Оптимізація в сфері надання послуг автомобільного сервісу за допомогою методів машинного навчання**»

Виконав: студент 6 курсу, групи ПДМ–61
спеціальності
121 Інженерія програмного забезпечення
(шифр і назва спеціальності/спеціалізації)

Тараканов Д.С.
(прізвище та ініціали)

Керівник Шевченко С.М.
(прізвище та ініціали)

Рецензент _____
(прізвище та ініціали)

Київ –2021

ДЕРЖАВНИЙ УНІВЕРСИТЕТ ТЕЛЕКОМУНІКАЦІЙ

НАВЧАЛЬНО-НАУКОВИЙ ІНСТИТУТ ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ

Кафедра Інженерії програмного забезпечення
Ступінь вищої освіти -«Магістр»
Спеціальність підготовки – 121 «Інженерія програмного забезпечення»

ЗАТВЕРДЖУЮ
Завідувач кафедри
Інженерії програмного забезпечення
Негоденко О.В.
"____" _____2020 року

З А В Д А Н Н Я
НА МАГІСТЕРСЬКУ РОБОТУ СТУДЕНТУ

ТАРАКАНОВУ ДМИТРУ СЕРГІЙОВИЧУ
(прізвище, ім'я, по батькові)

1. Тема роботи: «Оптимізація в сфері надання послуг автомобільного сервісу за допомогою методів машинного навчання»

Керівник роботи: Шевченко Світлана Миколаївна, к.т.н., доцент кафедри ІПЗ
(прізвище, ім'я, по батькові, науковий ступінь, вчене звання)

Затверджені наказом вищого навчального закладу від «13» жовтня 2020 року №230.
2. Строк подання студентом роботи «24» грудня 2020 року
3. Вхідні дані до роботи
Методи обробки зображень;
Науково-технічна література з питань, пов'язаних з програмним забезпеченням щодо розпізнавання тексту з зображень;

4. Зміст розрахунково-пояснювальної записки(перелік питань, які потрібно розробити).
4.1 Системи розпізнавання та вилучення текстової інформації з зображень.
4.2 Вимоги та оцінка якості системи.
4.3 Опис проектування системи.
4.4 Опис використаних технологій.

5. Перелік демонстраційного матеріалу (назва основних слайдів)

1. Актуальність проблеми
2. Існуюче програмне забезпечення та методи розпізнавання
3. Вилучення полів
4. Вилучення таблиць
5. Архітектура бази даних
6. Логічна діаграма компонентів архітектури програмного забезпечення
7. Інтерфейс системи

6. Дата видачі завдання «02» листопада 2020

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів магістерської роботи	Строк виконання етапів роботи	Примітка
1	Підбір науково-технічної літератури	02.11-07.11	Виконано
2	Вимоги до системи	08.11-10.11	Виконано
3	Створення та навчання моделі для вилучення полів	11.11-18.11	Виконано
4	Створення та навчання моделі для вилучення таблиць	19.11-28.11	Виконано
5	Концепція та архітектура програмного забезпечення	29.11-04.12	Виконано
6	Вступ, висновки, реферат	05.12-12.12	Виконано
7	Розробка обов‘язкових демонстраційних матеріалів	13.11-15.12	Виконано
8	Попередній захист роботи	16.12	
9	Здача роботи	24.12	

Студент _____ _____
(підпис) (прізвище та ініціали)

Керівник роботи _____ _____
(підпис) (прізвище та ініціали)

РЕФЕРАТ

Текстова частина магістерської роботи : 64 с., 11 рис., 30 джерел.

Об'єкт дослідження – покращення та вдосконалення роботи у сфері надання послуг автомобільного сервісу.

Предмет дослідження – додаток *для підбору потрібного сервісного центру для клієнта основуючись на наявності деталей та присутності вільного часу у правильного спеціаліста.*

Мета роботи – дослідження прикладів застосування методики машинного навчання Q-learning та її різновидів.

Методи дослідження – основні складові методики Q-learning та використання модифікацій Q-learning.

У роботі подано огляд загальної інформації про машинне навчання та розглядаються основні елементи Q-навчання. Проведено аналіз сучасного використання модифікацій Q-Learning: Fuzzy Q-навчання та ін. та методів прогнозування машинного навчання.

Загальною проблемою цього додатку є низька кількість різних характеристик для вводу, що ускладнює можливість прогнозування Q-learning. Тому, було взято сучасну модифікацію Q-Learning - Fuzzy.

Особливістю модифікації є можливість автоматично масштабувати комп'ютерні ресурси під час виконання для вдоволення потреб продуктивності і мінімізації витрат ресурсів. За основу було взято алгоритм AdaBoost та мова програмування Python з серверної сторони. AngularJS заснований на JavaScript та мова програмування TypeScript з клієнтської сторони для основного функціоналу.

Також деякий функціонал було розроблено, використовуючи бібліотеку ReactJS. Додаток працює з вже існуючими базами даних. В якості серверу баз даних було взято MySQL та бібліотеку Entity Framework для взаємодії з нею.

Отже, розроблено та описано веб-додаток, завданням якого є прогнозування та вдосконалення роботи у сфері надання послуг автомобільного сервісу.

Даний додаток може бути використано у автомобільних сервісах та інших зацікавлених структурах.

Галузь використання – автомобільні сервіси.

АВТОМОБІЛЬНІ СЕРВІСИ, МОДЕЛІ МАШИННОГО НАВЧАННЯ, ДОДАТОК TARGET, ADABOST, FUZZY Q-НАВЧАННЯ ,TYPESCRIPT, PYTHON, JAVASCRIPT, ANGULARJS, ВЕБ ДОДАТОК

ЗМІСТ

Перелік умовних позначень

QL – Q-Learning

FQL – Fuzzy Q-Learning

JS – Java Script

TP – Type Script

ML – Машинне навчання

SML – Часткове навчання

RL – Навчання із підкріпленням

UP – Навчання без вчителя

AB – AdaBoost

AJS – AngularJS

ВСТУП

Обгрунтування вибору теми та її актуальність: Проблема витрати часу на знаходження спеціаліста та наявності деталей у сервісі для якомого найшвидшого отримання результату.

В сучасному суспільстві значну роль відіграють інформаційні технології. Фактично, в сьогоденні вони використовуються в будь-якій сфері людського життя. Поміж всіх усіх інформаційних технологій, в сьогоднішньому світі особливе місце займають технології машинного навчання.

Початком історії машинного навчання можна назвати працю Томаса Баєса “An Essay towards solving a Problem in the Doctrine of Chances”[1], яку опублікували через 2 роки після його смерті Річардом Прайсом.

Після цього були відкриті ланцюги Маркова і Метод найменших квадратів, але справжнім початком розвитку машинного навчання вважається

1950 рік. Саме тоді, у жовтневому журналі “Mind” Алан Тюрінг опублікував статтю “COMPUTING MACHINERY AND INTELLIGENCE”[2] в якій запропонував машини, які можуть вчитися і ставати інтелектуальними. Це було передбаченням генетичних алгоритмів.

Перша нейромережева машина появилась в 1951 році під назвою SNARC “Stochastic neural analog reinforcement calculator”[3]

Згодом, в 1957 році, Френк Розенблат придумує персептрон, який був реалізований в 1960 році у вигляді нейрокомп’ютера Марк-1. Після цього відкриття дослідження пришвидшились. Були відкриті алгоритми “найближчого сусіда” і автоматичного розподілу. Так же Марвіном Мінським та Сеймуром Папертом в 1969 була написана книга “Perceptrons” в якій були описані обмеження персептронів і нейронних мереж.

В 80-х роках дослідження пожвавились. Саме в ці роки були відкриті рекурентні нейронні мережі і алгоритм зворотного поширення помилки. Так же, в

кінці цього десятиліття, а саме в 1989 році був винайдений алгоритм Q-Leaning, який значно покращив практичність навчання із підкріпленням.

І хоча QL був відкритий аж в 1989 році, він залишається актуальним і в сьогоденні. Прикладом цього є нова варіація алгоритму QL: Fuzzy.

Об'єктом дослідження є покращення та вдосконалення роботи у сфері надання послуг автомобільного сервісу.

Предмет дослідження є додаток *для підбору потрібного сервісного центру для клієнта основуючись на наявності деталей та присутності вільного часу у правильного спеціаліста.*

Метою роботи є дослідження прикладів застосування методики машинного навчання Q-learning та її різновидів..

Методи дослідження – основні складові методики Q-learning та використання модифікацій QL.

Завданням роботи є *підбір потрібного сервісного центру для клієнта основуючись на наявності деталей та присутності вільного часу у правильного спеціаліста*.

Методика дослідження: Перш за все, потрібно було визначити метод прогнозування машинного навчання, вибрати адекватну архітектуру та стек технологій із забезпечення ефективного використання програмного продукту автомобільними компаніями, за можливості підтримки його розробником з метою ефективного додавання нового функціоналу або виправлення виявлених недоліків.

Найбільш доречними виявився підхід Q-learning описаний в посібнику Стюарт Рассел, Пітер Норвіг, «II - сучасний підхід» [4] та ін. методів машинного прогнозування,тих, що представлені у роботі додатків та основній властивості машинного навчання – Марковській властивості.

Враховуючи вимоги до специфіки програмного продукту, найкращим вирішенням є веб-додаток, що має клієнт-серверну архітектуру, де клієнтська сторона відповідає за роботу та логіку програмного продукту, а серверна – за обчислення. Такий тип технології дає можливість використання складних логічних

та математичних обчислень, не призводячи до необхідності введення великої кількості даних.

Для клієнтської сторони веб-додаток може виглядати, форма в яку треба вписати усі ушкодження авто.

Результати пошуку відображаються на мапі де буде показаний найближчий сервісний центр ,який найбільш підходить, а також у вигляді таблиці.

Результатом обчислення є такі основні характеристики, як вільний час привильного майстра та наявність деталей на його робочому місці.

Наукова новизна роботи: Таким чином, наукова новизна полягає в удосконаленні існуючого методу Machine Learning та розробці програмного продукту, що має можливість підбору потрібного сервісного центру для клієнта.

Практична значущість результатів: Даний продукт може бути використано автомобільними компаніями, а також іншими зацікавленими.

1 ДОСЛІДЖЕННЯ ПРЕДМЕТНОЇ ОБЛАСТІ

1.1. Машинне навчання

Машинне навчання (Machine Learning) - це складна одиниця штучного інтелекту, яка досліджує методи побудови алгоритмів навчання. Машинне навчання - це інтерфейс між математичною статистикою, методами оптимізації та класичними математичними дисциплінами, яка також має свої специфічні особливості, пов'язані з обчислювальною продуктивністю та проблемами перенавчання. Багато методів індуктивного навчання розроблено як альтернативу класичним статистичним підходам. Багато методів тісно пов'язані з пошуком інформації та обробкою даних.

Більшість теоретичних розділів машинного навчання пов'язані в окремому напрямку, теорії обчислювального навчання.

ML - це не лише математична, а й практична, технічна дисципліна. Чиста теорія зазвичай не призводить одразу до методів та алгоритмів, які можна застосувати на практиці. Щоб вони добре працювали, необхідно винайти додаткові механізми, щоб компенсувати невідповідність теоретичних припущень умовам реальних проблем. Практично жодне дослідження машинного навчання не проходить без експерименту з моделлю чи реальними даними, що підтверджує практичну доцільність цього методу.

1.1.1. Часткове навчання

Часткове навчання (semi-supervised learning) використовує як марковані, так і спільні дані під час навчання. Зазвичай використовується невелика кількість розмічених тегів та значна кількість нерозмічених даних. SML - це компроміс між навчанням без вчителя (без позначених даних про навчання) та навчанням з викладачем (з повністю розміченим набором навчання). Було встановлено, що непомічені дані з невеликою кількістю позначених даних можуть значно

підвищити якість навчання. Під якістю навчання мається на увазі якийсь функціонал якості, наприклад, середньоквадратична помилка.

Збір розмічених даних для навчального завдання часто вимагає кваліфікованого експерта для ручної класифікації навчальних об’єктів. Витрати, пов’язані з процесом розмітки, можуть бути настільки високими, що створити повний набір неможливо, в той час як збір нерозподілених даних коштує порівняно недорого. У таких ситуаціях значення часткового навчання важко переоцінити.

1.1.2. Навчання із підкріпленням

Навчання із підкріпленням (reinforcement learning) вивчає, як агент повинен поводитися в середовищі, щоб отримати довгостроковий прибуток. Часткові алгоритми навчання намагаються знайти стратегію, щоб гарантувати, що агент обирає найкращі дії у будь-якому стані навколишнього середовища. В економічній і ігровій теорії посилене навчання розглядається як тлумачення того, як досягти рівноваги.

На відміну від RL, посилене навчання не забезпечує правильних пар введення та відповіді, і прийняття майже оптимальних рішень (надання місцевого екстремуму) прямо не обмежується. Розширене навчання намагається знайти компроміс між дослідженням невивчених областей та застосуванням наявних знань.

1.1.3. Навчання із учителем

У навчанні з учителем ідея полягає у тому, що між відповідями та предметами існує взаємозв'язок, але це невідомо.

Відомий лише остаточний набір прецедентів - пари "об'єкт, відповідь", які називаються тренувальними зразками. На основі цих даних необхідно відновити залежність, тобто побудувати алгоритм, який здатний дати досить точну відповідь для кожного об’єкта.

Для вимірювання точності відповідей певним чином вводиться функція якості.

Під вчителем означається або сам педагогічний зразок, або той, що дав правильні відповіді на задані об’єкти. (Рис. 1.1)

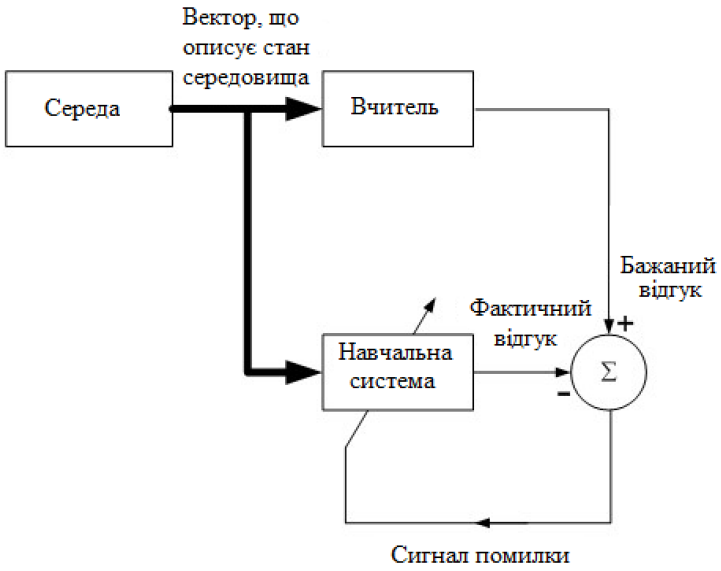


Рисунок 1.1 – Навчання із учителем[5]

Функціонал якості зазвичай визначається як середня помилка відповідей, заданих алгоритмом для всіх об'єктів вибірки.

1.1.4. Навчання без вчителя

Навчання без вчителя — один зі способів машинного навчання, при вирішенні яких випробовувана система спонтанно навчається виконувати поставлене завдання, без втручання з боку експериментатора.

UP вирішує задачі, коли відомий лише опис багатьох об'єктів. І ви повинні знайти внутрішні зв’язки, відносини, які існують між об'єктами. Прикладами таких завдань є кластеризація даних та візуалізація.

1.2 Q-Learning

Q-навчання - метод, що застосовується в штучному інтелекті з агентурним підходом. Відноситься до навчання із підкріпленням (reinforcement learning). На основі винагороди, яку він отримав від оточення, агент формує функцію корисності Q, яка також дає йому можливість не випадково вибирати стратегію поведінки, а враховувати досвід попередніх взаємодій. Однією з переваг Q-learning є те, що він може порівнювати очікувані переваги наявних заходів без створення екологічної моделі.

1.2.1. Агент і середовище

Агент і навколишнє середовище взаємодіють на кожному з дискретних кроків, насправді, фактично середовище чекає відповіді агента. На кожному кроці агент отримує стан навколишнього середовища Санкт -∈, де S – множина можливих станів та аналізу стану, він вибирає можливі дії ∈ A(ст), де a(ст) – – множина можливих дій для стану s_t. На наступному етапі агент отримує винагороду rt+1, як результат вибраної дії. (Рис. 1.2)

Рисунок 1.2 – Взаємодія агента і середовища [7]

Елемент, який навчається і приймає рішення - називається агентом. Сукупність всіх об'єктів, що знаходяться поза агентом, і з якими цей агент взаємодіє, позначається терміном середовище. Така взаємодія відбувається постійно: агент вибирає дії, навколишнє середовище реагує на них, створюючи нові ситуації для агента). Навколишнє середовище є також джерелом винагород

(спеціальні числові значення, які агент постійно намагається збільшити). Повний опис характеристики навколишнього середовища, визначає завдання. [6]

Ця дуже загальна і гнучка структура може використовуватися для вирішення різних завдань різними засобами. Наприклад, тимчасові інтервали не обов'язково мають фіксований інтервал у режимі реального часу, ви можете вибрати відповіді на будь-якому наступному етапі прийняття рішень та виконання дій.

Строго кажучи, межа між агентом і середовищем не завжди є такою ж, як фізична межа тіла робота або живого організму. Зазвичай ця межа ближче до агента, ніж фізична межа його тіла. Наприклад, мотори та сенсорні пристрої робота повинні розглядатися як об'єкти навколишнього середовища, а не як частина агента. Якщо дивитися з цієї точки зору, м'язи, скелет та органи сприйняття також повинні бути частиною навколишнього середовища. Винагорода зазвичай розраховується в живому організмі або навчається в штучній системі, але також як зовнішня по відношенню до агента.

Існує правило: все, що не може довільно змінюватися агентом вважається навколишнім середовищем. Агент зазвичай досить обізнаний про те, як його винагорода та функції обчислюються з його дій та станів, у яких ці дії відбуваються. Однак процес нарахування винагороди завжди є агентом зовнішнім, оскільки цей процес визначає завдання агента, і через це агент не має можливості змінити його за бажанням.

1.2.2. Цілі і винагороди

У навчанні з підкріпленням мета агента формалізується за допомогою спеціального сигналу винагороди, що надходить до агента з навколишнього середовища. На кожному етапі t винагороду представляє звичайне число $r_t \in R$. Метою агента є максимізація загальної суми нагород. Це означає, що потрібно максимізувати НЕ винагороду поточного кроку, а сукупну підсумкову винагороду за результатами досить тривалої послідовності кроків.

Використання сигналу винагороди для формалізації цілей є найбільш яскравою особливістю навчання з підкріпленням. Незважаючи на те, що спочатку цей спосіб формалізації цілей здається обмеженим, він на практиці довів свою гнучкість.

Агент вчиться постійно максимізувати отриману винагороду. Якщо ми хочемо, щоб він щось зробив для нас, ми повинні компенсувати встановлення, щоб при максимізуванні винагороди наші цілі здобувалися. Отже, необхідно встановити вказану винагороду правильно, так щоб було зрозуміло, чого ми хочемо досягти. Зокрема, сигнал винагороди не повинен наділяти агента апріорними знаннями про те, як досягти того, чого ми від нього хочемо. Наприклад, агент для гри в шахи, винагорода повинна бути лише за фактичну перемогу, але не за досягнення якихось проміжних цілей, наприклад, захоплення фігур суперника або контролю над центром дошки. Якщо досягнення таких середньострокових цілей нагороджувалось, то агент міг знайти спосіб їх досягнення, не досягаючи при цьому головну мету.

Наприклад, він може знайти спосіб захопити фігури суперника, навіть ціною програшу гри. Сигнал про винагороду - це можливість повідомити агента про ціль, якої він повинен досягнути, а не інформацію про те як він повинен це зробити[6]

1.2.3. Алгоритм Q-Learning

Найпростіша форма, однокрокового QL, визначається наступним чином:

$$Q(s_t, a_t) \leftarrow Q(s_t, a_t) + \alpha[r_{t+1} + \gamma \max Q(s_{t+1}, \alpha) - Q(s_t, a_t)] \qquad (1.1)$$

Де s_t – стан середовища в момент t;

a_t – дія, яку виконав агент в під час стану s_t;

r_t – нагорода після виконання агентом дії a_t.

В цьому випадку шукана функція цінності дії Q, безпосередньо апроксимує Q*, оптимальну функцію цінності дії, незалежно від використаної стратегії. . Це

значно спрощує аналіз алгоритму та зберігає раніше запропоновані докази конвергенції.

Стратегія також визначає, які пари статусу активності відвідуються та адаптуються. Однак для забезпечення конвергенції необхідно, лише щоб усі пари продовжували коригуватися. Це мінімальна вимога в тому сенсі, що кожний метод гарантовано знайде оптимальну процедуру, де загалом ця умова буде виконана.

За цих умов і в цьому варіанті типових умов стохастичного наближення для послідовності значень довжини кроків було показано, що функція Qt збігається з ймовірністю від 1 до Q *.

Алгоритм QL в процедурній формі можна показати так: [6]

Ініціалізувати Q (s, a) випадковими даними;

Повторювати (для кожного епізоду);

Ініціалізувати s;

Знайти a по s, використовуючи стратегію отриману із Q;

Виконати дію a, знайти r, s′:

$Q(s_t, a_t) \leftarrow Q(s_t, a_t) + \alpha[r_{t+1} + \gamma \max Q(s_{t+1}, \alpha) - Q(s_t, a_t)]$ $s \leftarrow s'$;

Поки s не стане кінцевим станом.

1.3 Fuzzy Q-Learning

Fuzzy Q-Learning має на меті реагувати на вимоги програми шляхом автоматичного масштабування комп'ютерних ресурсів під час виконання, щоб задовольнити вимоги до продуктивності та мінімізувати витрати на ресурси. Існуючі контролери часто витрачаються на стратегію масштабування, яка визначається набором правил масштабування. Однак значна частина хмарної інфраструктури часто виглядає як чорний ящик, що ускладнює створення оптимальних правил.

Тому було замінено правила на нейронну мережу з використанням Fuzzy Q-Learning, нечіткої логічної версії QL.

1.3.1 Причини вибору алгоритму

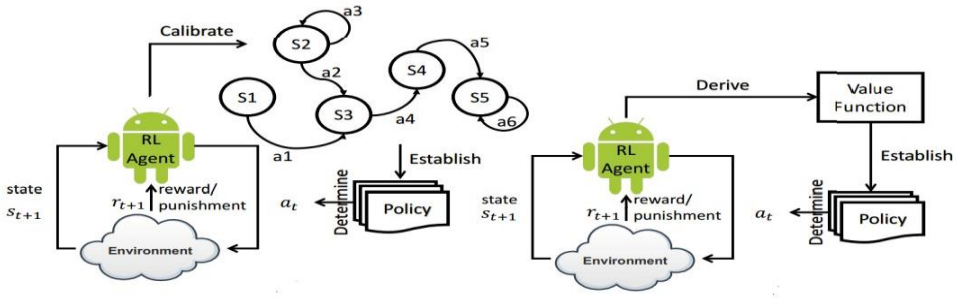
Є кілька функцій, які ставлять під сумнів існуючі методи автоматичного масштабування та пристрої:

• Навколишнє середовище не є епізодичним, це означає, що наступні вибори вплинуть на наступні дії.

• Моделювання хмарної інфраструктури дуже складне.

• Навантаження нерегулярні та динамічні.

Враховуючи характеристики середовища, було прийнято рішення використовувати нейронну мережу на навчанні із підкріпленням.



а) б)

Рисунок 1.3. – Варіанти RL нейронної мережі: а) основана на моделі; б) без модельна[7]

Ці властивості середовища роблять необхідним вирішення проблеми послідовних рішень, в яких кожна дія поточного стану впливає на майбутнє. Поширений спосіб вирішити цю проблему - створити план, правило або стратегію дій, які будуть працювати до тих пір, поки не виникне специфічна ситуація.

Агент вибирає дії на основі правила, що збільшує майбутню винагороду. RL на основі моделі (рис. 1.3 а) може бути корисним при виборі рішення, але його дуже важко використовувати через складність моделювання хмарної інфраструктури в цій RL.

Вирішення цієї проблеми полягає у виборі без модельної версії RL (рис. 1.3 б). Рішення на користь QL було також пов'язано з тим, що він має оптимальну політику для отримання негайної та відстроченої винагороди. Через високу складність та складність хмарного середовища не вдалося отримати повну інформацію для належного налаштування діяльності. Завдяки FQL ви можете використовувати неповні знання. Нечіткі правила вдосконалюються під час збору даних у режимі реального часу.

QL було обрано через непередбачуваність завантаження програми в хмарі, що фактично не дозволило отримати дату навчальної мережі. Для QL не потрібно навчати. FQL був обраний через неповну проблему знань.

1.4 Алгоритм AdaBoost

AdaBoost - алгоритм машинного навчання, запропонований Йоавом Фрейндом та Робертом Шапірою. Цей алгоритм можна використовувати в поєднанні з декількома алгоритмами класифікації для підвищення їх ефективності. Алгоритм посилює класифікатори, об'єднуючи їх у "комітет". AdaBoost є адаптивним в тому сенсі, що кожен наступний комітет класифікатора базується на об'єктах, які були неправильно класифіковані попередніми комітетами. AdaBoost чутливий до шуму даних та переживань. Однак він менш схильний до перепідготовки порівняно з іншими алгоритмами машинного навчання.

AB викликає слабкі класифікатори в циклі t = 1, ..., T. Після кожного виклику розподіл ваг D t, що відповідає важливості кожного з об'єктів у навчальному наборі для класифікації, оновлюється. Кожна ітерація збільшує вагу кожного неправильно класифікованого об'єкта, тому новий комітет класифікатора зосереджується на цих об'єктах.

Переваги алгоритму

- Хороша узагальнююча здатність. В реальних задачах (не завжди, але часто) вдається будувати композиції, що перевершують за якістю базові алгоритми. Узагальнююча здатність може поліпшуватися (в деяких завданнях) по мірі збільшення числа базових алгоритмів.
- Простота реалізації.
- Власні накладні витрати бустинга невеликі. Час побудови композиції практично повністю визначається часом навчання базових алгоритмів.

Недоліки алгоритму

- AB схильний до перенавчання при наявності значного рівня шуму в даних. Експоненціальна функція втрат занадто сильно збільшує ваги найбільш важких об'єктів, на яких помиляються багато базові алгоритми. Однак саме ці об'єкти найчастіше виявляються шумовими викидами. В результаті AdaBoost починає налаштовуватися на шум, що веде до перенавчання. Проблема вирішується шляхом видалення викидів або застосування менш агресивних функцій втрат.
- AB вимагає досить довгих навчальних вибірок. Інші методи лінійної корекції, зокрема, бэггинг, здатні будувати алгоритми порівнянного якості з меншим вибірках даних.
- Жадібна стратегія послідовного додавання призводить до побудови неоптимального набору базових алгоритмів. Для поліпшення композиції можна періодично повертатися до раніше побудованих алгоритмів і навчати їх заново. Для поліпшення коефіцієнтів можна оптимізувати їх ще раз по закінченні процесу бустинга за допомогою якого-небудь стандартного методу побудови лінійної розділяє поверхні. Рекомендується використовувати для цієї мети SVM (машини опорних векторів).
- Бустинг може призводити до побудови громіздких композицій, що складаються з сотень алгоритмів. Такі композиції виключають можливість

змістовної інтерпретації, вимагають великих обсягів пам'яті для зберігання базових алгоритмів і істотних витрат часу на обчислення класифікацій.

1.5 Марковська властивість

Під час навчання з підкріпленням рішення агента виступають як сигнал від навколишнього середовища, який називається станом навколишнього середовища. В ідеалі цей сигнал стану повинен коротко підсумувати минулі відчуття, щоб зберегти всю необхідну інформацію. Сигнал стану, який містить всю необхідну інформацію, називається сигналом Маркова.

Розглянемо, як середовище при t + 1 може реагувати на заходи, вжиті при t. Загалом, середовище реагування залежить від попередніх подій. У цьому випадку динаміка середовища визначається лише за умови повного розподілу ймовірностей (1.1).

$$P_r\{s_{t+1} = s',\ r_{t+1} = r \mid s_t,\ a_t,\ r_t,\ s_{t-1},\ a_{t-1},\ r_{t-1}. \ldots . ,\ r_1,\ s_0,\ a_0\} \qquad (1.2)$$

Де s_i – стан середовища в момент i, r_i – нагорода в момент i, a_i – дія в момент i.

З іншого боку, якщо сигнал стану має властивість Маркова, реакція середовища в момент st + 1 залежить лише від представлення стану та дії в момент t. У цьому випадку можна визначити динаміку навколишнього середовища. Знати лише попередній стан (1.3)

$$P_r\{s_{t+1} = s',\ r_{t+1} = r \mid s_t,\ a_t\} \qquad (1.3)$$

Іншими словами, стан має властивість Маркова і, отже, є марківським станом тоді і лише тоді, коли (1.2) дорівнює (1.3). Якщо середовище має властивість Маркова, ми можемо використовувати його одномоментну динаміку (1.3) для прогнозування наступного стану та наступної очікуваної винагороди, що є результатом поточного стану та дії. Коли для цього співвідношення виконуються ітераційні розрахунки, можна передбачити всі майбутні стани та очікувані

винагороди, а також усю історію станів, відомий лише поточний стан. Звідси випливає, що марківський стан пропонує найкращі з усіх можливих підстав для вибору заходів. Таким чином, найкраща стратегія вибору дій, визначена як функція стану Маркова, настільки ж ефективна, як і найкраща стратегія, сформована як функція всієї історії станів.

Навіть якщо сигнал стани не є марковським, він все одно повинен розглядатися в навчальній задачі з підвищення рівня як наближення стану Маркова. Зокрема, завжди бажано, щоб ця умова була гарною основою для передбачення подальших винагород та вибору діяльності. Марковські стани - найкраща основа для досягнення всіх цих цілей. Чим більше характеристики ціого стану схожі з характеристиками станів Маркова, тим кращою буде система навчання з підкріпленням. Таким чином, на кожному часовому етапі стан переважно вважається наближенням стану Маркова, хоча слід пам’ятати, що він може не повністю задовольняти властивість Маркова. [6]

1.5.1 Марковський процес прийняття рішень

Завдання навчання з підкріпленням, що відповідають марковським властивостям, називається процесом прийняття рішень Маркова. Обидва терміни «марковська властивість» та «сильна марковська властивість» застосовувалися

у зв'язку з особливою властивістю «відсутності пам'яті» експоненційного розподілу.

Процес рішення Маркова - це конкретизація послідовного завдання рішення для повністю спостережуваного середовища з марковською перехідною моделлю та додатковими винагородами. Цей процес має такі функції:

- Зовнішнє середовище розвивається на основі законів ймовірності припускаючи скінченну множину дискретних станів. Однак ці умови не включають даних попередньої статистики.
- Є багато можливих дій, які система може виконувати в кожному статусі

- Є багато можливих дій, які система може виконувати в кожному статусі
- Під час акції стягується плата або надається ціна.

Ма́рковські проце́си вири́шування забезпечують математичну систему для моделювання ухвалення рішень у ситуаціях, в яких наслідки є частково випадковими, а частково контрольованими ухвалювачем рішення. МПВ є корисними для дослідження широкого спектра задач оптимізації, розв'язуваних динамічним програмуванням та навчанням з підкріпленням. МПВ були відомі щонайменше з 1950-х років (пор. Bellman, 1957). Основна маса досліджень марковських процесів вирішування стала результатом книги Рональда Говарда, опублікованої 1960 року, «Динамічне програмування та марковські процеси» . Їх застосовують у широкій області дисциплін, включно з робототехнікою, автоматизованим керуванням, економікою та виробництвом.

Якщо точніше, то марковський процес вирішування є стохастичним процесом керування дискретного часу. На кожному кроці часу процес перебуває в якомусь стані s, і ухвалювач рішення може обрати будь-яку дію a, доступну в стані s.

Ймовірність переходу процесу до його нового стану s знаходиться під впливом обраної дії. Конкретно, вона задається функцією переходу стану P a (s , s ′). Таким чином, наступний стан s ′ залежить від поточного стану s та від дії ухвалювача рішення a. Але для заданих s та a він є умовно незалежним від усіх попередніх станів та дій; іншими словами, переходи станів процесу МПВ задовольняють марковську властивість.

2 ОПИС ЗАСОБІВ РЕАЛІЗАЦІЇ

2.1. Мова програмування Python для написання серверної частини

Python - інтерпретована мова програмування високого рівня та загального призначення. Філософія дизайну Python підкреслює читабельність коду завдяки помітному використанню значних пробілів. Його мовні конструкції та об’єктно-орієнтований підхід мають на меті допомогти програмістам писати чіткий логічний код для малих та великих проектів. [24]

Python підтримує кілька парадигм програмування, включаючи структуроване (зокрема, процедурне), об'єктно-орієнтоване та функціональне програмування. Python часто описують як мову, що включає батареї, завдяки своїй повній стандартній бібліотеці. [25]

Python був створений наприкінці 1980-х років, а вперше випущений у 1991 році Гвідо ван Россумом як наступник мови програмування ABC. Python 2.0, випущений у 2000 році, представив нові функції, такі як розуміння списків, та систему збору сміття з підрахунком посилань, і був припинений з версією 2.7 у 2020 році [26]. Python 3.0, випущений в 2008 році, був серйозним переглядом мови, яка не є повністю сумісною із зворотною суттю, і більша частина коду Python 2 не працює незміненою на Python 3.

Інтерпретатори Python доступні для багатьох операційних систем. Глобальне співтовариство програмістів розробляє та підтримує CPython, безкоштовну та з відкритим кодом [27], довідкову реалізацію. Некомерційна організація, Фонд програмного забезпечення Python, управляє та направляє ресурси для розробки Python та CPython. В даний час вона пов’язана з Java як другою за популярністю мовою програмування у світі. [28] [29]

Python був задуманий наприкінці 1980-х [25] Гвідо ван Россумом з Centrum Wiskunde & Informatica (CWI) в Нідерландах як наступник мови програмування ABC, натхненної SETL) [26], здатний обробляти винятки та взаємодіяти. з операційною системою Amoeba. [27] Його реалізація розпочалась у

грудні 1989 р. [36] Ван Россум взяв на себе виключну відповідальність за проект, як провідний розробник, до 12 липня 2018 року, коли він оголосив про свою "постійну відпустку", виконуючи обов'язки як "Доброзичливий диктатор за життя" Пітона, титул, який йому надала спільнота Пітона, щоб відобразити його довгий термін зобов’язання як головного керівника проекту. [28] Зараз він ділиться своїм керівництвом як член керівної ради з п'яти осіб [29] [30] [31]. У січні 2019 року активні розробники ядра Python обрали Бретта Кеннона, Ніка Коглана, Барі Варшаву, Керол Віллінг та Ван Россума членами керівної ради, що керуватимуть проектом. [32] З тих пір Гвідо ван Россум відкликав свою кандидатуру на посаду Керівної ради 2020 року. [33]

Python 2.0 був випущений 16 жовтня 2000 року з багатьма основними новими можливостями, включаючи збирач сміття, що виявляє цикл, та підтримку Unicode. [43]

Python 3.0 був випущений 3 грудня 2008 року. Це був серйозний перегляд мови, який не є повністю сумісним із зворотною стороною. [34] Багато його основних функцій були передані до версій Python 2.6.x [35] та 2.7.x. Випуски Python 3 включають утиліту 2to3, яка автоматизує (принаймні частково) переклад коду Python 2 на Python 3. [36]

Дата закінчення терміну служби Python 2.7 спочатку була встановлена на 2015 рік, а потім перенесена на 2020 рік через занепокоєння тим, що велика кількість існуючого коду не може бути легко перенесена на Python 3. [37] [39] Більше виправлень безпеки та інших удосконалень для нього випускати не буде [39] [40]. У кінці життя Python 2 підтримується лише Python 3.6.x [41] та пізніші версії.

Python - мова програмування з багатьма парадигмами. Об'єктно-орієнтоване програмування та структуроване програмування повністю підтримуються, і багато його функцій підтримують функціональне програмування та аспектно-орієнтоване програмування (в тому числі шляхом метапрограмування [52] та метаоб'єктів (магічні методи)). [53] Багато інших парадигм підтримуються за допомогою

розширень, включаючи дизайн за контрактом [54] [55] та логічне програмування. [56]

Python використовує динамічне введення тексту та комбінацію підрахунку посилань та збирач сміття, що визначає цикл, для управління пам’яттю. Він також має динамічну роздільну здатність імен (пізня прив'язка), яка зв'язує імена методів та змінних під час виконання програми.

Дизайн Python пропонує певну підтримку функціонального програмування за традицією Lisp. Він має функції фільтрування, картографування та зменшення; перелічити розуміння, словники, набори та вирази генераторів. [58] Стандартна бібліотека має два модулі (itertools та functools), що реалізують функціональні інструменти, запозичені у Haskell та Standard ML. [59]

Основна філософія мови узагальнена в документі Zen of Python (PEP 20), який включає такі афоризми, як: [60]

- Красиве - краще, ніж потворне.
- Явне краще, ніж неявне.
- Просте - краще, ніж складне.
- Комплексне краще, ніж складне.
- Підрахунок читабельності.

Замість того, щоб усі його функціональні можливості були вбудовані в його ядро, Python був розроблений таким чином, щоб бути дуже розширюваним. Ця компактна модульність зробила його особливо популярним як засіб додавання програмованих інтерфейсів до існуючих програм. Бачення Ван Россума щодо малої основної мови з великою стандартною бібліотекою та легко розширюваним перекладачем випливало з його розчарувань у ABC, який підтримував протилежний підхід. [34]

Python прагне до більш простого, менш захламленого синтаксису та граматики, одночасно надаючи розробникам можливість вибору методології кодування. На відміну від девізу Perl "існує більше ніж один спосіб", Python

охоплює "філософію дизайну" повинен бути один - і бажано лише один - очевидний спосіб зробити це [60]. Алекс Мартеллі, співробітник Фонду програмного забезпечення Python та автор книги про Python, пише, що "Описувати щось як розумне — не вважається компліментом у культурі Python" [61].

Розробники Python прагнуть уникнути передчасної оптимізації та відкидають виправлення для некритичних частин еталонної реалізації CPython, які пропонують незначне збільшення швидкості ціною ясності. [62] Коли швидкість важлива, програміст Python може переносити критично важливі для часу функції до модулів розширення, написаних мовами, такими як C, або використовувати PyPy, своєчасний компілятор. Також доступний Cython, який перекладає сценарій Python на C і здійснює прямі виклики API рівня C в інтерпретатор Python.

Важливою метою розробників Python є підтримка задоволення від використання. Це відображається в назві мови - данина пам’яті британському гумористичному гурту Монті Пайтон [63] - та у випадкових грайливих підходах до навчальних посібників та довідкових матеріалів, таких як приклади, що стосуються спаму, стандартних foo and bar. [64] [65]

Поширеним неологізмом у спільноті Python є пітонічний, який може мати широкий діапазон значень, пов’язаних зі стилем програми. Сказати, що код є пітонічним, означає сказати, що він добре використовує ідіоми Python, що він природний або демонструє вільну мову, що відповідає мінімалістичній філософії Python та наголосу на читабельності. На відміну від цього, код, який важко зрозуміти або читається як груба транскрипція з іншої мови програмування, називається непітонічним.

Користувачів та шанувальників Python, особливо тих, хто вважається обізнаним або досвідченим, часто називають Pythonistas. [66] [67]

Python призначений для легко читаної мови. Його форматування візуально не захаращене, і в ньому часто використовуються англійські ключові слова, де інші мови використовують розділові знаки. На відміну від багатьох інших мов, він не

використовує фігурні дужки для розмежування блоків, а крапки з комою після операторів необов’язкові. Він має менше синтаксичних винятків та особливих випадків, ніж C або Pascal. [68]

Заяви та контроль потоку

Заяви Python включають (серед іншого):

- Оператор присвоєння (маркер '=', знак рівності). Це діє інакше, ніж у традиційних імперативних мовах програмування, і цей фундаментальний механізм (включаючи природу версії змінних Python) висвітлює багато інших особливостей мови. Призначення в C, наприклад, x = 2, перекладається як "введене ім'я змінної x отримує копію числового значення 2". Значення (праворуч) копіюється у виділене місце зберігання, для якого ім'я змінної (ліворуч) є символічною адресою. Пам'ять, виділена для змінної, досить велика (потенційно досить велика) для оголошеного типу. У найпростішому випадку призначення Python, використовуючи той самий приклад, x = 2, перекладається на "(загальне) ім'я x отримує посилання на окремий, динамічно виділений об'єкт числового (int) типу значення 2." Це називається прив'язками імені до об'єкта. Оскільки місце зберігання імені не містить зазначеного значення, неправильно називати його змінною. Потім імена можуть бути в будь-який час відбиті до об’єктів дуже різного типу, включаючи рядки, процедури, складні об’єкти з даними та методами тощо. Послідовне присвоєння загального значення кільком іменам, наприклад, x = 2; y = 2; z = 2 призводить до розподілу пам’яті до (щонайбільше) трьох імен та одного числового об’єкта, до якого прив’язані всі три імена. Динамічни масивом називають такий масив, розмір якого можна змінювати під час виконання програми. Оскільки ім'я є загальним власником посилань, нерозумно пов'язувати з ним фіксований тип даних. Однак в даний момент часу ім'я буде прив'язане до якогось об'єкта, який матиме тип; таким чином відбувається динамічне введення тексту.
- Оператор if, який умовно виконує блок коду, разом з else та elif (скорочення else-if).

- Оператор for, який переглядає ітерабельний об’єкт, захоплюючи кожен елемент до локальної змінної для використання вкладеним блоком.
- Оператор while, який виконує блок коду, доки його умова відповідає істині.
- Оператор try, який дозволяє вилученням, що містяться у вкладеному блоці коду, ловити та обробляти за винятком речень; він також гарантує, що код очищення в блоці нарешті завжди буде виконуватися незалежно від того, як блок виходить.
- Оператор підвищення, що використовується для отримання вказаного винятку або повторного підняття спійманого винятку.
- Оператор класу, який виконує блок коду та приєднує його локальний простір імен до класу для використання в об'єктно-орієнтованому програмуванні.
- Оператор def, що визначає функцію або метод.
- Оператор with з Python 2.5, випущений у вересні 2006 р. [70], який охоплює блок коду в диспетчері контексту (наприклад, отримання блокування перед запуском блоку коду та звільнення блокування після цього, або відкриття файлу, а потім закриваючи його), дозволяючи схожість ресурсів на подібну до ініціалізації (RAII) поведінку, і замінює загальну ідіому спроби / остаточно. [71]
- Оператор break, виходить із циклу.
- Оператор continue, пропускає цю ітерацію і продовжує наступний пункт.
- Заява про передачу, яка виконує функцію NOP. Це синтаксично потрібно для створення порожнього блоку коду.
- Оператор assert, який використовується під час налагодження для перевірки умов, які слід застосувати. Макрос assert() додає до програми процедуру діагностики. Після виконання, якщо визначення хибне(тобто, порівняння рівне 0), assert() пише інформацыю про виклик, котрий виконався з помилкою на stderr і викликає функцію abort().

- Оператор yield, який повертає значення з функції генератора. З Python 2.5, yield також є оператором. Ця форма використовується для реалізації спільних програм.
- Оператор return, який використовується для повернення значення з функції.
- Оператор import, який використовується для імпорту модулів, функції чи змінні яких можна використовувати в поточній програмі. Є три способи використання імпорту: import <module name> [as <alias>] або from <module name> import * або from <module name> import <definition 1> [as <alias 1>], <definition 2> [as <alias 2>], ...
- Оператор print змінено на функцію print () у Python 3.

Python не підтримує оптимізацію хвостових викликів або першокласні продовження, і, за словами Гвідо ван Россума, це ніколи не буде. [72] [73] Однак краща підтримка подібних до корутинів функціональних можливостей надається в 2.5, розширюючи генератори Python. [74] До 2.5 генератори були ледачими ітераторами; інформація передавалась в односпрямованому напрямку з генератора. З Python 2.5 можна передавати інформацію назад у функцію генератора, а з Python 3.3 інформацію можна передавати через кілька рівнів стека. [75]

Вирази

Деякі вирази Python подібні до таких мов, як C та Java, а деякі - ні:

- Додавання, віднімання та множення однакові, але поведінка ділення відрізняється. У Python існує два типи поділів. Вони є поділом підлоги (або цілочисельним поділом) // та плаваючою комою / поділом. [76] Python також додав оператор ** для додавання до степені.
- З Python 3.5 був представлений новий оператор @ infix. Він призначений для використання бібліотеками, такими як NumPy, для множення матриць. [77] [78]
- З Python 3.8 був введений синтаксис: =, який називається «моржовим оператором». Він присвоює значення змінним як частину більшого виразу. [79]

- У Python == порівнює за значенням порівняно з Java, яка порівнює числові значення за значенням [80], а об'єкти за посиланням. [81] (Порівняння значень в Java на об'єктах можна виконувати методом equals ().) Оператор Python is is може використовуватися для порівняння ідентифікацій об'єктів (порівняння за посиланням). У Python порівняння можуть бути ланцюговими, наприклад a <= b <= c.
- Python використовує слова та, або, не для своїх булевих операторів, а не для символічного &&, ||,! використовується в Java та C.
- Python має тип вираження, що називається розумінням списку. Python 2.4 розширив розуміння списку в більш загальний вираз, який називають генераторським виразом. [58]
- Анонімні функції реалізовані за допомогою лямбда-виразів; однак вони обмежені тим, що тіло може бути лише одним виразом.
- Умовні вирази в Python записуються як x, якщо c else y [82] (відрізняється за порядком операндів від оператора c? X: y, загального для багатьох інших мов).
- Python розрізняє списки та кортежі. Списки записуються як [1, 2, 3], змінюються та не можуть використовуватися як ключі словників (ключі словника повинні бути незмінними в Python). Кортежі записуються як (1, 2, 3), є незмінними і, отже, можуть використовуватися як ключі словників, за умови, що всі елементи кортежу є незмінними. Оператор + може бути використаний для об'єднання двох кортежів, що безпосередньо не змінює їх вміст, а створює новий кортеж, що містить елементи обох передбачених кортежів. Таким чином, враховуючи змінну t, спочатку рівну (1, 2, 3), виконання t = t + (4, 5) спочатку обчислює t + (4, 5), що дає (1, 2, 3, 4, 5) , який потім призначається назад до t, тим самим ефективно "модифікуючи вміст" t, одночасно відповідаючи незмінній природі об'єктів кортежу. Дужки не є обов’язковими для кортежів у однозначному контексті. [83]
- Python розпаковує послідовність, при якій декілька виразів, кожен з яких обчислює будь-що, до чого можна присвоїти (змінну, властивість, що

записується, тощо), пов’язані ідентично тому, що формує кортежні літерали, і, як ціле, розміщуються на лівий бік знака рівності у твердженні про присвоєння. Оператор очікує ітерабельний об'єкт праворуч від знака рівності, який видає таку ж кількість значень, що і надані вираз для запису, коли його перебирають і буде перебирати його, присвоюючи кожне із створених значень відповідному виразу зліва . [84]

- Python має оператор " string format". Це функціонує аналогічно рядкам формату printf на C, наприклад "spam=%s eggs=%d" % ("blah", 2) evaluates to "spam=blah eggs=2". В Python 3 та 2.6+, це було реалізовано за допомогою метоу format(), e.g. "spam={0} eggs={1}".format("blah", 2). Python 3.6 включає "f-strings": blah = "blah"; eggs = 2; f'spam={blah} eggs={eggs}'.[85]
- Python має різні типи рядкових літералів:
- Рядки, розділені одинарними або подвійними лапками. На відміну від оболонок Unix, мови Perl та Perl, що впливають на Perl, одинарні лапки та подвійні лапки функціонують однаково. Обидва типи рядків використовують зворотну скісну риску (\) як символ виходу. Інтерполяція рядків стала доступною в Python 3.6 як "відформатовані рядкові літерали". [85]
- Рядки з потрійними лапками, які починаються та закінчуються серією з трьох одинарних або подвійних лапок. Вони можуть охоплювати кілька рядків і функціонувати, як тут документи в оболонках, Perl і Ruby.
- Сирі різновиди рядків, що позначаються префіксом рядкового літералу перед r. Послідовності втечі не інтерпретуються; отже, необроблені рядки корисні там, де буквенні зворотні слеші є загальними, такі як регулярні вирази та шляхи у стилі Windows. Порівняйте "@ -citation" у C #.
- Python має індекси масивів та вирази нарізування масивів у списках, що позначаються як [ключ], [старт: зупинка] або [старт: зупинка: крок]. Індекси базуються на нулі, а негативні індекси відносно кінця. Зрізи беруть елементи від початкового індексу до, але не включаючи, індексу зупинки. Третій параметр зрізу, який називається кроком або кроком, дозволяє пропускати та обертати елементи.

Індекси зрізів можуть бути опущені, наприклад, [:] повертає копію всього списку. Кожен елемент зрізу є неглибокою копією.

2.2. Мова програмування TypeScript для написання клієнту

TypeScript - мова програмування з відкритим кодом, розроблена та підтримувана Microsoft. Це суворий синтаксичний набір JavaScript і додає до мові необов'язкове статичне введення тексту. TypeScript призначений для розробки великих програм та транскопіляцій у JavaScript. Оскільки TypeScript є набором JavaScript, існуючі програми JavaScript також є дійсними програмами TypeScript.

TypeScript може використовуватися для розробки програм JavaScript для виконання як на стороні клієнта, так і на стороні сервера (як у Node.js або Deno). Існує кілька варіантів транскопіляції. Може використовуватися перевірка TypeScript за замовчуванням, або компілятор Babel може бути викликаний для перетворення TypeScript в JavaScript.

TypeScript підтримує файли визначення, які можуть містити інформацію про типи існуючих бібліотек JavaScript, як і файли заголовків C ++, можуть описувати структуру існуючих об'єктних файлів. Це дає можливість іншим програмам використовувати значення, визначені у файлах, так, як якщо б вони були статично набраними сутностями TypeScript. Існують сторонні файли заголовків для популярних бібліотек, таких як jQuery, MongoDB і D3.js. Також доступні заголовки TypeScript для основних модулів Node.js, що дозволяє розробляти програми Node.js в TypeScript.

Компілятор TypeScript сам пишеться в TypeScript і компілюється в JavaScript. Він ліцензований за ліцензією Apache 2.0. TypeScript включений як першокласна мова програмування в Microsoft Visual Studio 2013 Update 2 та новіших версій, поряд із C # та іншими мовами Microsoft. Офіційне розширення дозволяє Visual Studio 2012 також підтримувати TypeScript. Андер Хейлсберг, провідний архітектор C # і творець Delphi і Turbo Pascal, працював над розробкою TypeScript[7].

Вперше TypeScript був оприлюднений у жовтні 2012 року (у версії 0.8) після двох років внутрішньої розробки в Microsoft. Незабаром після оголошення Мігель де Іказа похвалив саму мову, але розкритикував відсутність зрілої підтримки IDE, окрім Microsoft Visual Studio, яка в той час не була доступною для Linux та OS X. Сьогодні існує підтримка в інших IDE, зокрема в Eclipse, за допомогою плагіна, який надає Palantir Technologies. Різні текстові редактори, включаючи Emacs, Vim, Webstorm, Atom та власний код Visual Studio Microsoft, також підтримують TypeScript.

TypeScript 0.9, випущений у 2013 році, додав підтримку для дженериків. TypeScript 1.0 був випущений на конференції розробників Microsoft у 2014 році. Visual Studio 2013 Update 2 забезпечує вбудовану підтримку TypeScript.

У липні 2014 року команда розробників оголосила про новий компілятор TypeScript, який вимагає 5 × підвищення продуктивності. Одночасно вихідний код, який спочатку розміщувався на CodePlex, був переміщений до GitHub.

22 вересня 2016 року було випущено TypeScript 2.0; він ввів декілька функцій, включаючи можливість програмістів необов'язково запобігати присвоєнню змінним нульових значень, іноді їх називають помилкою в мільярд доларів.

TypeScript 3.0 був випущений 30 липня 2018 року, який містить багато мовних доповнень, таких як кортежі в параметрах спокою та вирази розповсюдження, параметри відпочинку з типами кортежу, загальні параметри відпочинку тощо.

TypeScript виникла з недоліків JavaScript для розробки масштабних додатків як у Microsoft, так і серед їх зовнішніх клієнтів. Проблеми, що стосуються роботи зі складним кодом JavaScript, спричинили потребу в користувальницьких інструментах для полегшення розробки компонентів мови.

Розробники TypeScript шукали рішення, яке не порушило б сумісність зі стандартом та його міжплатформою підтримкою. Знаючи, що поточна стандартна пропозиція ECMAScript обіцяє майбутню підтримку програмування на основі класів, TypeScript базувався на цій пропозиції. Це призвело до компілятора JavaScript з набором синтаксичних розширень мови, суперсети на основі пропозиції, що перетворює розширення в звичайний JavaScript. У цьому сенсі TypeScript був попереднім переглядом того, чого слід очікувати від ECMAScript 2015. Унікальним аспектом, який не в пропозиції, а доданий до TypeScript, є необов'язкове статичне введення тексту, що дозволяє статичний аналіз мови, що полегшує інструментарій та підтримку IDE.

TypeScript є суворим набором ECMAScript 2015, який сам по собі є набором ECMAScript 5, який зазвичай називають JavaScript.Таким чином, програма JavaScript також є дійсною програмою TypeScript, і програма TypeScript може безперешкодно споживати JavaScript. За замовчуванням компілятор орієнтований на ECMAScript 5, поточний переважаючий стандарт, але також може генерувати конструкції, що використовуються в ECMAScript 3 або 2015.

За допомогою TypeScript можна використовувати існуючий код JavaScript, включати популярні бібліотеки JavaScript та викликати код, сформований TypeScript з іншого JavaScript. Декларації типів для цих бібліотек надаються вихідним кодом[2].

Середовища і редактори.

- Microsoft надає плагін для Visual Studio 2012 та WebMatrix, повну інтегровану підтримку Visual Studio 2013, Visual Studio 2015 та підтримку базового текстового редактора для Emacs та Vim. WebMatrix – це інструмент, який об’єднує в собі всі засоби, необхідні для автономної web-розробки на платформі.
- Visual Studio Code - це (в основному) редактор вихідного коду з платформою з відкритим кодом, розроблений Microsoft на базі Electron. Він підтримує TypeScript на додаток до кількох інших мов і пропонує такі функції, як налагодження та інтелектуальне завершення коду.

- alm.tools - це хмара IDE з відкритим кодом для TypeScript, побудована за допомогою TypeScript, ReactJS та TypeStyle.
- JetBrains підтримує TypeScript з доповненням коду, рефакторингом та налагодженням його IDE, побудованих на платформі IntelliJ, таких як PhpStorm 6, WebStorm 6 та IntelliJ IDEA, , а також їх надбудови та розширення Visual Studio, ReSharper 8.1.
- У Atom є плагін TypeScript від Basarat з підтримкою завершення коду, навігації, форматування та швидкої компіляції.
- Онлайн Cloud9 IDE та Codenvy підтримують TypeScript.
- Плагін доступний для IDE NetBeans.
- Плагін доступний для IDE Eclipse (версія Kepler)
- TypEcs доступний для IDE Eclipse.
- Cross Cloud Cloud IDE Codeanywhere підтримує TypeScript.
- Webclipse Плагін Eclipse, призначений для розробки TypeScript і Angular 2.
- Angular IDE Автономний IDE, доступний через npm для розробки додатків TypeScript і Angular 2, з інтегрованою підтримкою терміналів.
- Tide - Інтерактивне середовище розробки TypeScript для Emacs.

2.3. AngularJS

AngularJS - це веб-система з відкритим вихідним кодом, заснована на JavaScript, в основному підтримується Google та спільнотою осіб та корпорацій для вирішення багатьох проблем, що виникають при розробці програм на одній сторінці. Він спрямований на спрощення як розробки, так і тестування таких додатків, забезпечуючи структуру для клієнтської архітектури модель-перегляд-контролер (MVC) та модель-перегляд-перегляд-модель (MVVM), а також компоненти, що часто використовуються в багатих Інтернет-додатках.

AngularJS - це пряма частина стека MEAN, що складається з бази даних MongoDB, рамки сервера веб-додатків Express.js, самого Angular.js та середовища

виконання сервера Node.js. Версія 1.7.x працює на довгостроковій підтримці до 1 липня 2021 року. Після цієї дати AngularJS більше не оновлюватиметься, а натомість запропоновано Angular (2.0+).

Рамка AngularJS працює, попередньо прочитавши сторінку мови розмітки гіпертексту (HTML), на якій вбудовані додаткові спеціальні атрибути HTML. Кутова інтерпретує ці атрибути як директиви для прив’язки вхідних або вихідних частин сторінки до моделі, яка представлена стандартними змінними JavaScript. Значення цих змінних JavaScript можна встановити вручну в коді або отримати зі статичних або динамічних ресурсів JSON[11].

AngularJS побудований на вірі, що декларативне програмування слід використовувати для створення інтерфейсів користувача та підключення програмних компонентів, тоді як імперативне програмування краще підходить для визначення ділової логіки програми. Рамка адаптує та розширює традиційний HTML для представлення динамічного контенту за допомогою двостороннього зв’язування даних, що дозволяє здійснювати автоматичну синхронізацію моделей та поглядів. Як результат, AngularJS децензує увагу на явній маніпуляції з об'єктною моделлю документа (DOM) з метою поліпшення перевірки та продуктивності[8].

Цілі проектування AngularJS включають:

- щоб від'єднати маніпуляцію DOM від логіки програми. На складність цього різко впливає спосіб структурування коду. Об’єктна модель документа(DOM) – специфікація прикладного програмного інтрефейсу для роботи зі структурованими документами (як правило, документами XML).
- для роз'єднання клієнтської програми програми з боку сервера. Це дозволяє паралельно прогресувати роботи з розробки та дозволяє повторно використовувати обидві сторони.
- надати структуру для побудови програми: від проектування інтерфейсу, через написання ділової логіки, до тестування.

AngularJS реалізує шаблон MVC для розділення компонентів презентації, даних та логіки. Використовуючи введення залежностей, Angular традиційно постачає сервіси на стороні сервера, такі як контролери, залежні від перегляду, до веб-додатків на стороні клієнта. Отже, значна частина навантаження на сервер може бути зменшена.

AngularJS використовує термін "сфера дії" таким чином, що схожий на основи інформатики.

Сфера застосування інформатики описує, коли в програмі діє певне прив'язка. Специфікація ECMA-262 визначає область як: лексичне середовище, в якому об'єкт функції виконується в веб-скриптах на стороні клієнта; схожий на те, як визначається область застосування в обчисленні лямбда.

Як частина архітектури "MVC", область містить "Модель", а всі змінні, визначені в області, можуть отримати доступ до "Перегляду", а також "Контролера". Область поводиться як клей і зв’язує «Вид» і «Контролер».

Завдання, яке виконується завантажувачем AngularJS, відбувається в три фази після завантаження DOM[4]:

- Створення нового інжектора
- Складання директив, що прикрашають DOM
- Зв’язування всіх директив із сферою застосування

Директиви AngularJS дозволяють розробнику задавати користувацькі та багаторазові HTML-подібні елементи та атрибути, що визначають прив'язку даних та поведінку компонентів презентації. Деякі з найбільш часто використовуваних директив:

ng-animate

Модуль забезпечує підтримку JavaScript, CSS3 переходу та анімаційних гачків анімації ключових кадрів CSS3 в рамках існуючих основних та спеціальних директив.

Оскільки атрибути ng- * не відповідають дійсності в специфікаціях HTML, дані-ng- * можуть також використовуватися як префікс. Наприклад, і ng-app, і data-ng-додаток дійсні в AngularJS.

ng-app

Заявляє кореневий елемент програми AngularJS, згідно з яким директиви можуть використовуватися для оголошення прив’язок та визначення поведінки.

ng-aria

Модуль підтримки доступності загальних атрибутів ARIA.

ng-bind

Встановлює текст елемента DOM на значення виразу. Наприклад,   відображає значення "name" всередині елемента span. Будь-яка зміна змінної "ім'я" в області застосування програми негайно відображається в DOM.

ng-class

Умовно застосуйте клас, залежно від значення булевого виразу.

ng-controller

Визначає клас контролера JavaScript, який оцінює вирази HTML.

ng-if

Основна, якщо директива твердження, яка створює наступний елемент, якщо умови справджуються. Коли умова помилкова, елемент видаляється з DOM. Якщо це правда, клон складеного елемента повторно вставляється.

ng-init

Викликається один раз при ініціалізації елемента.

ng-model

Аналогічно ng-bind, але встановлює двостороннє зв’язування даних між представленням і областю.

ng-model

Аналогічно ng-bind, але встановлює двостороннє зв’язування даних між представленням і областю.

ng-model-options

Забезпечує налаштування способів оновлення моделі.

ng-repeat

Екземплярує елемент один раз на предмет із колекції.

ng-show & ng-hide

Умовно покажіть або прихойте елемент, залежно від значення булевого виразу. Показати та приховати досягається встановленням стилю відображення CSS.

ng-switch

Умовно інстанціюйте один шаблон із набору варіантів, залежно від значення виразу вибору.

ng-view

Базова директива, відповідальна за обробку маршрутів, які вирішують JSON перед наданням шаблонів, керованих зазначеними контролерами.

Випуск 1.6 додав багато понять Angular до AngularJS, включаючи концепцію архітектури додатків на основі компонентів. Цей випуск серед інших видалив пісочницю, яка, як вважають багато розробників, забезпечила додаткову безпеку, незважаючи на численні вразливості, які виявили обхід пісочниці[11]. Поточний (станом на березень 2020 року) стабільний випуск AngularJS становить 1,7,9.

AngularJS був спочатку розроблений у 2009 році Мішко Гевери у компанії Brat Tech LLC [14] як програмне забезпечення, що стоїть на онлайновому сервісі зберігання даних JSON, яке було б коштуватиме мегабайт, для зручних для роботи підприємств. Це підприємство було розміщене у веб-домені "GetAngular.com", і

мало декількох передплатників, перш ніж вони вирішили відмовитися від бізнес-ідеї та випустити Angular як бібліотеку з відкритим кодом.

У січні 2018 року було оголошено графік поступового припинення AngularJS: після випуску 1.7.0 активна розробка AngularJS триватиме до 30 червня 2018 року. Згодом 1.7 буде підтримуватися до 30 червня 2021 року як довгострокова підтримка[3].

Подальші версії AngularJS просто називають Angular. Angular - це несумісне перезапис AngularJS на основі TypeScript. Кутовий 4 був оголошений 13 грудня 2016 року, пропускаючи 3, щоб уникнути плутанини через невідповідність версії пакету маршрутизатора, яка вже поширювалася як v3.3.0.

AngularDart працює над Dart, який є об'єктно-орієнтованим, визначеним класом, єдиною мовою програмування успадкування, використовуючи синтаксис стилю C, який відрізняється від Angular JS (для якого використовується JavaScript) та Angular 2 / Angular 4 (для якого використовується TypeScript). Angular 4 випущений у березні 2017 року, при цьому версія рамки узгоджується з номером версії маршрутизатора, який він використовував. Angular 5 вийшов 1 листопада 2017 року. Основні вдосконалення в Angular 5 включають підтримку прогресивних веб-додатків, оптимізатор збірки та вдосконалення, пов'язані з дизайном матеріалів. Angular 6 було випущено 3 травня 2018 року, Angular 7 було випущено 18 жовтня 2018 року, а Angular 8 було випущено 28 травня 2019 року. Angular зобов’язався забезпечити 6 місяців активної підтримки для кожної основної версії, а потім 12 місяців довгострокової підтримки. Основні випуски є щорічно, з 1 до 3 незначних випусків для кожного головного випуску[19].

2.4. Побудова дерева цілей

Слід зауважити, що на практиці, як правило, існує кілька цілей і тому важливо, окрім визначення головної мети, не упустити деякі з суттєвих серед інших. Для цього застосовують метод побудови дерева цілей, що був запропонований ще 1957 року групою американських учених та успішно

використаний в ряді військових та промислових програм у США, а нині є повсякденним інструментом практично будь-якого сучасного менеджера. Метод побудови дерева цілей являє собою один із найрозповсюдженіших та найефективніших способів аналізу слабко структурованих завдань, що стоять перед економічними об’єктами. Він допомагає знаходити найкращі шляхи та засоби вирішення існуючих проблем. Деревоподібні ієрархічні структури використовуються і при дослідженні та удосконаленні організаційних структур.

Отже для побудови дерева цілей будемо використовувати поділ загальної цілі на під цілі, тобто визначимо які основні цілі будуть а які глобальні.

Головна мета (генеральна мета).

Створення оптимізованої системи сервісу на основі Python технології буде головною метою нашої проектної розробки. Це є загальне визначення генеральної мети. В результаті у нас повинна функціонувати система обміну даних між користувачами, яка включає в себе всі цілі та досягнення мети.

Під цілі першого рівня.

Під цілі першого рівня є головними цілями що проектуються після генеральної мети. Для визначення цілей першого рівня потрібно організувати структуру та основні функції роботи нашої системи, а саме:

- вибір середовища програмування;
- авторизація та реєстрація користувачів у системі;
- профільна частина користувача;
- обмін повідомленнями;
- кросбраузерність.

Під цілі другого рівня.

Для висвітлення цих цілей потрібно чітко зрозуміти які основні розподіли цілей першого рівня існують у системі. Це зумовлено проектуванням на рівні деталізації цілей похідних рівнів. На цьому рівні деталізуються багатообіцяючі цілі які ми проектуємо. Серед таких цілей є:

- побудова сесії підтримки користувача;
- збереження сеансу користувача;
- статуси користувачів;
- статуси та прихід повідомлень;
- розлогінення користувачів;
- використання універсального фреймворка для кросбраузерності javascript та typescript коду.

Під цілі третього рівня

- автоматичне оновлення індикаторів повідомлень. (впровадження AngularJS);
- автоматичне оновлення індикаторів Користувачів. (впровадження TypescriptJS);
- створення кімнати спілкування для користувача;

Після деталізації та розподілу загальної мети можемо побудувати дерево цілей. Дерево цілей показує які головні цілі нам потрібно виконати і дає змогу вирішити всі явні проблеми на різних етапах а це є основною проблемою. Отже дерево цілей буде мати 4 гілки: головна мета(генеральна мета на цьому етапі проектування), під цілі першого рівня, під цілі другого рівня, під цілі третього рівня. В **додатку А** нижче показане дерево цілей яке спроектоване на цьому етапі.

Отже як бачимо із діаграми цілей системного аналізу інформаційної системи обміну даних всі цілі чітко характеризуються поетапним виконанням поставленого завдання.

2.5 Побудова дерева завдань

Побудова дерева проблеми включає в себе необхідний етап побудови дерева завдань. Дерево завдань включає в себе висновки щодо правильної роботи системи, правильної роботи окремих функціональних складових та всіх гілок дерева цілей.[11]

Основні завдання, які повинні бути реалізовані у системі обміну інформацією на основі Python технології, можна класифікувати за наступними критеріями:

- інноваційний підхід
- простота програмного коду
- кросбраузерність додатку.
- створення зручного інтерфейсу користувача

Кожен із критеріїв що відповідають поставленому завданню класифікується як окремі складові. Використання такого підходу дає нам можливість розглянути конкретні завдання та розробити алгоритм їх вирішення. Нижче наведені складові для кожного критерію поставленого завдання.

Інноваційний підхід.

Використання сучасного підходу до створення веб додатків вимагає впровадженні все нових та складніших технологій у сфері інформаційних технологій. Інноваційний підхід у нашому випадку включає в себе впровадження сучасного стилю програмування, тобто використання найсучасніших програмних компонентів для отримання результату. Основні цілі в цьому контексті є автоматичне оновлення даних без перевантаження веб сторінки користувача, обробка запитів у фоновому режимі, використання кросбраузерного підходу у проектуванні нашої системи. Все це можна реалізувати за допомогою AngularJS інноваційного підходу, який широко використовуються у веб програмуванні. Потрібно реалізувати використання технології AngularJS, що дасть змогу автоматично оновлювати дані для користувача та відправляти запити у фоновому режимі без перевантаження веб сторінки. Також потрібно використання сучасного стилю програмування AngularJS підходу. Програмування функціональної частини додатку буде дуже зручним та простим із використання мови сценаріїв python. Отже програмування веб додатку буде здійснюватися мовою сценаріїв python а впровадження технології AngularJS буде реалізовано клієнтською мовою програмування JavaScript та TypeScript.

Простота програмного коду.

Для модульного принципу проектування простота програмного коду дуже необхідна. Сценарії програмування повинні бути зрозумілими та структурованими, це дає змогу з легкістю проводити тестування та правлення коду програми. Для кращого розуміння всієї структури програмного коду буде використано модульну структуру програмування. Головні частини коду будуть поділені на окремі програмні файли та підключені у тих місцях де це необхідно. Кожна функціональна частина додатку буде окремим файлом. Також важливою складовою простоти використання є повторне використання коду. Один і той самий сценарій може бути використаний у декількох місцях програмної структури, це буде необхідних принципом для економії часу. Отже для оптимально створення програмного коду є використання модульного принципу та повторного використання коду.

Кросбраузерність додатку.

Існує безліч клієнтських програм які користувач використовує для перегляду веб сторінок перебуваючи у мережі Інтернет. Один і той самий додаток може по різному інтерпретуватись у різних версіях браузерів. Це зумовлено тим що кожний браузер спроектований різними розробниками які у свою чергу не дотримуються ідентичного стандарту опрацювання веб сторінок. Варіантів для створення програмного коду що однаково інтерпретується браузерами є досить багато. Для створення такого результату потрібно використовувати верстку сайту прийнятну для всіх типів браузерів, а сучасні мови гіпертекстової розмітки сторінок дають змогу це втілити у реальність.

Попросту кажучи, таку характеристику дають сайтам, дизайн яких однаковий як в Firefox, так і в Google Chrome. Відображення сайту при цьому може відрізнятися лише дрібними деталями (наприклад закругленими куточками, рівнем тіні, градієнтної або суцільною заливкою), але не кольоровою гамою, розташуванням елементів, а також, що найважливіше у визначенні даного поняття, структурою сайту, адже особливо багато проблем виникає з “блочною” версткою. Часто виникає багато проблем з версіями Internet Explorer нижче 10, тому багато розробників веб-сайтів не турбуються про кросбраузерність, а ставлять сторінку-

загрузку, тобто користувач Firefox може бачити лише одну сторінку, на якій йому рекомендується змінити браузер, без можливості перегляду контенту.

Інтерфейс користувача у нашому прикладі відповідає за сприйняття та простоту динаміки роботи веб додатку. Потрібно створити простий та зрозумілий інтерфейс роботи користувача та веб додатку що й полягає в основу проектування цієї частини роботи. Потрібно створити динамічно зрозумілу веб сторінку яка буде правильно функціонувати і при цьому виконувати правильно всі команди користувача. Отже після опрацювання всіх критеріїв що були розглянуті вище можемо переходити до побудови діаграми дерева завдань (Додаток Б).

2.6 Обґрунтування та агрегування критеріїв

Під критеріями розуміють кількісні показники якісних цілей, які мають точніше їх характеризувати. Критерії мають якомога точніше відповідати цілям, хоча і не можуть повністю збігатися з ними, оскільки вони фіксуються в різних шкалах вимірювання: цілі – в номінальних, а критерії – в шкалах, що передбачають упорядкування.

При формуванні критеріїв головним є не їх кількість, а те, наскільки повно вони характеризують ціль. Тому тут прагнуть досягти компромісу між повнотою описування цілей та кількістю критеріїв. Для повноти описування проблемної ситуації необхідно розглядати три взаємодіючі системи:

- систему, в якій існуюча ситуація розглядається як проблема;
- систему, в рамках якої можна вплинути на проблему для її вирішення;
- зовнішнє середовище, в якому існують та з яким взаємодіють ці дві системи.

Формулювання критеріїв потрібно якомога більш наблизити до цілей які ми побудували. Цілі являють собою якісний показник роботи системи. Для представлення критеріїв будемо опиратись на такі особливості як надійність, захищеність, швидкість опрацювання, середовище роботи, інноваційний підхід, користувацький інтерфейс. Як бачимо ці критерії найточніше характеризуються

нашу проектну ціль при розробці. Серед дуже важливих картерів можемо виділити такі, як захищеність та інноваційний підхід.

Надійність системи.

Критерій надійності системи в першу чергу відповідає за захищеність усіх даних, що мають обіг у системі а також надійність роботи системи з точки зору апаратних та технічних можливостей. Оскільки наша в першу чергу є система обміну інформацією тому слід врахувати такий фактор як невідказність роботи системи та робота при максимальному навантаження користувачів. Оскільки в системі постійно йде обмін даними між користувачами прослідкувати за всіма транзакціями користувачів з метою запобігання несинхронної роботи системи. При достатньому навантаженні користувачів у системі потрібно прослідкувати за синхронізацією роботи запитів та навантаження роботи на базу даних щоб це не призвело до так званого “зависання” роботи системи. Надійність – властивість технічних об’єктів зберігати у встановлених межах часу значення всіх параметрів, які характеризують здатність виконувати потрібні функції в заданих режимах та умовах застосування, технічного обслуговування, зберігання та транспортування

Захищеність.

У нашому прикладі захищеність стосується приватних даних усіх користувачів що реєструються у системі обміну інформацією. Кожний користувач має свій власний профіль за допомогою якого він може працювати у системі. Ця можливість надається авторизацією та реєстрацією у системі. Всі дані користувача мають бути захищені з середини системи і це дасть змогу здійснювати лише санкціонований обіг даних користувача. Ця можливість втілюється за допомогою створення окремої бази даних що містись всі дані користувачів та створення так званої “сесії користувача” середовищем мови програмування python. Тобто в загальному розумінні доступ користувачам до своєї інформації буде лише після авторизації або реєстрації у системі.

Середовище роботи.

Для повнофункціональної роботи системи потрібно середовище її виконання. Таким середовищем роботи системи є браузер користувача, тобто веб переглядач сторінок. Існує безліч нюансів роботи веб оглядача. Як було сказано раніше повинна бути підтримка роботи мови сценаріїв JavaScript, що зазвичай деякими користувачами відключається, в такому випадку повнофункціональна робота системи неможлива. Веб-браузер у кожного користувача на його власному комп'ютері відрізняється, тому постає проблема кросплатформенності. Потрібно реалізувати роботу системи у всіх середовищах виконання однаково, що дасть змогу зробити це.

Інноваційний підхід.

Побудова функціоналу роботи системи зосереджена на використанні сучасних версій мов програмування та синхронізації роботи користувача. Для програмування буде використовуватись мова python. Особливість вибору такого рішення є те що python створений для реалізації саме для таких цілей які нам потрібно і при цьому дуже зручний та простий у використанні. Для того щоб уникнути класної моделі поведінки веб додатку було прийнято рішення використання технології AngularJS, яка дасть змогу створити синхронізацію роботи користувача із системою, а саме уникнути перевантаження веб сторінок при відправці даних про що йшлось раніше.

Користувацький інтерфейс.

Робота користувача у системі повинна бути для нього прозора та зручна саме за це буде відповідати створюваний на даному етапі проектування інтерфейс. Основним принципом зручності інтерфейсу користувача буде використання технології Angular. Як говорилось раніше всі користувацькі запити будуть синхронно відправлятись на опрацювання і при цьому робота користувача не буде перериватись на час виконання обробки запиту.

Аналіз альтернативних шляхів вирішення завдання.

Цим етапом системного аналізу є генерування альтернатив, тобто ідей та можливих шляхів досягнення визначеної мети. Генерування альтернатив є творчим процесом. Тут немає чіткого плану щодо альтернативних рішень та їх побудови, варто визначити основні принципи та напрямки вирішення завдання аналогічним чином. Генерування альтернативних ідей щодо вирішення визначеної проблеми полягає у використанні подібних способів створення результату на основі вже існуючих.

Використання альтернативних технологій.Програмну реалізацію веб додатку було вирішено розробити трьома компонентами, а саме:

- python;
- javaScript;
- angular.

Використання мови програмування python було зумовлене простотою та зручністю для програміста що дало можливість оцінити її як основний варіант для програмної розробки. Альтернативним шляхом вирішення може слугувати мова програмування Ruby. Використовується як мова загального призначення для створення як веб додатків так і програмного забезпечення різного рівня складності. Ruby запозичує можливості з багатьох інших мов програмування, як то C, shell scripting, AWK та sed. Мова надає потужні можливості для обробки тексту без довільних обмежень на довжину даних багатьох сучасних інструментів Unix, полегшуючи процес маніпуляції текстових файлів. Використовується для програмування графіки, системного адміністрування, у мережному програмуванні, у написанні програмного забезпечення, яке взаємодіє з базами даних, у програмуванні CGI для веб.

Отже доцільним альтернативним рішення програмного рівня буде використання мови Ruby, що підтримує усі потрібні можливості для побудови веб додатку.

Для створення динаміки інтерфейсу користувача було прийнято рішення використання мови сценаріїв, що виконується на клієнтській машині, а саме

JavaScript. Альтернативним рішенням щодо використання JavaScript буде Rhino. Rhino – рушій JavaScript з відкритим сирцевим кодом. Він написаний повністю на Java та підтримується Mozilla Foundation. Mozilla Foundation також підтримує іншу реалізацію рушія JavaScript, написану на C SpiderMonkey.

Rhino перетворює JavaScript скрипти в Java класи. Rhino працює і у компільованому, та інтерпретованих режимах. Він призначений для використання у веб-застосунках на серверному боці, тому в ньому немає вбудованої підтримки для об'єктів браузера, які зазвичай асоціюються з JavaScript.

Rhino може використовуватися при використанні Rhino Shell. Він також може використовуватися в застосунках при вбудовуванні Rhino.

Використання AngularJS є основною метою проектної розробки нашого завдання дипломного проектування. Для використання цієї технології було об’єднання трьох суміжних технологій, а саме:

Використання DHTML для динамічної зміни змісту сторінки.

Використання XMLHttpRequest для звернення до сервера «на льоту», не перезавантажуючи всю сторінку повністю[12]

- альтернативний метод – динамічне під вантаження коду JavaScript в тег <SCRIPT> з використанням DOM, що здійснюється із використанням формату JSON)[13];
- динамічне створення дочірніх фреймів.

Щодо використання альтернативної заміни Angular тут можливо спроектувати використання концепції несинхронної обробки даних. Тобто по суті без фонового режиму роботи веб додатку. Для імітування принципу Angular можливе створення динамічного під вантаження коду JavaScript в тег <SCRIPT> з використанням DOM, що здійснюється із використанням формату JSON).[14]

Отже враховуючи проведені всі вище системні аналізи та проектні дослідження можемо побудувати відобразити концептуальну модель розв'язку завдання.

Дерево проблем у нашому випадку буде складатись із трьох гілок, а саме:

- дерево цілей;
- дерево завдань;
- дерево альтернативних рішень.

2.7. Прийняття рішення системного аналізу

Судячи із проведеного системного аналізу та проектної побудови вирішення завдання можемо прийняти чітке рішення, щодо всіх етапів побудови веб додатку. З метою доцільного виконання проектної розробки для прийняття рішення було враховано критеріальні оцінки при системному аналізі та можливості альтернативних варіантів вирішення завдання.

Технології програмування додатку.

Розглянуті вище способи програмної реалізації додатку будуть найкраще підходити для вирішення цієї задачі. Використання таких мов програмування як python та JavaScript дасть змогу реалізувати всі поставлені завдання що стосуються проектно-функціональної розробки додатку. Програмування користувацького інтерфейсу повністю покладається на мову програмування JavaScript. Базовою технологією що використовується для нашого проектування використовується Angular. Принцип фонового режиму роботи додатку повністю покладається на суміжну технологію Python що використовується наряду із JavaScript.

Алгоритм роботи додатку.

Алгоритм роботи додатку був вище розглянутий і на даному етапі проектування він буде прийнятий як базовий. Розглядаючи дерево цілей стає зрозуміло, що всі гілки дерева відповідають функціональним проектним можливостям додатку. Включаючи альтернативні рішення алгоритмів роботи додатку можемо взяти приклад із готових варіантів роботи систем, що надані для прикладу як аналоги додатку що проектується у даному дипломному проекті. Оцінюючи алгоритм роботи додатку спираємось на критеріальні рішення які

допоможуть визначитись із правильною постановкою роботи всіх пунктів алгоритму додатку.

Використання системи збереження даних.

Усі дані, що приймають обіг у системі мають бути збережені у надійній базі даних. Як дані особистих профілів користувачів так і всі повідомлення що користувачі відправляли та отримували під час роботи веб додатку мають бути збережені у відповідній базі даних. Таким рішенням є база даних MySQL. Це база даних реляційного типу, що повністю відповідає потребам проектної розробки додатку. Доступність використання та простота використання бази даних mysql на програмному рівні створює зручний спосіб її використання.

Отже виконання системного аналізу дало нам можливість побачити шляхи вирішення завдання дипломного проектування, побудова дерева завдань дало змогу визначити чіткі принципи роботи системи та всі привальні шляхи її вирішення. Виходячи із альтернативних шляхів вирішення завдання маємо можливість побачити додаткові шляхи вирішення деякі з проблем поставленого завдання. Оглядаючи всі критерії щодо роботи веб додатку стає зрозумілим та точнішим виконання поставленого завдання. Загалом головне завдання системних досліджень полягає в пошуку простоти у складному, а також ефективних методів та засобів дослідження й управління об'єктами. Принципи системного підходу – це положення загального характеру, що є узагальненням досвіду дослідження людиною складних систем. Їх часто вважають ядром методології.

3 ОТИМІЗАЦІЯ В СФЕРІ НАДАННЯ ПОСЛУГ АВТОМОБІЛЬНОГО СЕРВІСУ ЗА ДОПОМОГОЮ МЕТОДІВ МАШИННОГО НАВЧАННЯ

3.1. Загальні відомості

Даний розділ буде розглядати основні теоретичні шляхи вирішення проблеми дипломного проектування. Буде розглянуто концептуальний алгоритм роботи додатку та його всі основні функціональні можливості, побудова діаграм uml для характеристики роботи додатку а також розглянуто спосіб інтеграції Angular технології та всі її особливості на програмному рівні веб додатку.(рис. 1.4)

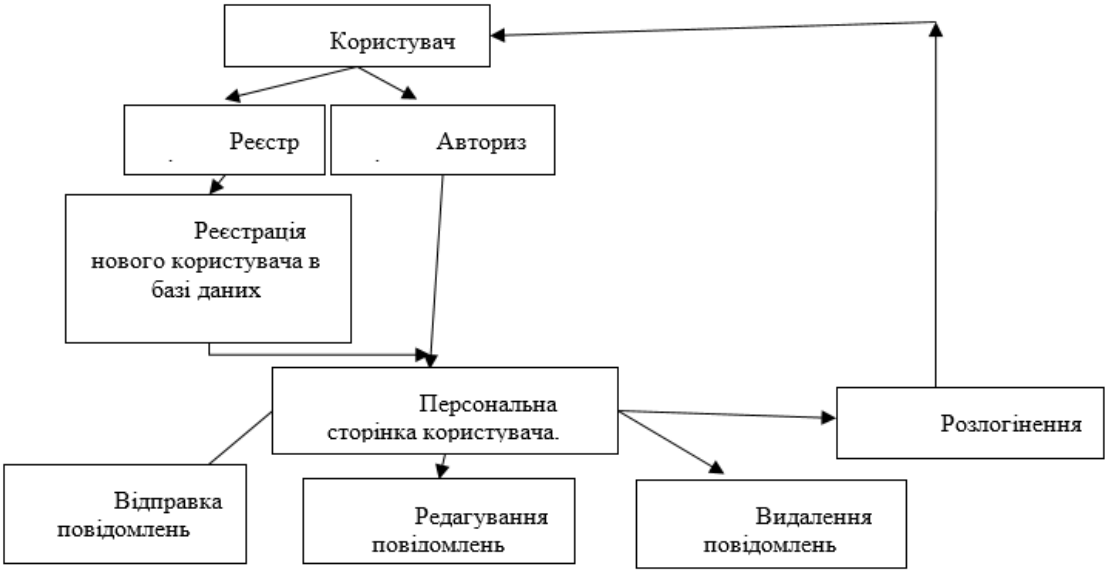


Рисунок 1.4. - Концептуальний алгоритм роботи веб додатку

Алгоритм роботи демонструє в простоті роботу веб додатку що проектується у даному дипломному проекту.

Як бачимо із схеми головним прецедентом є користувач а також алгоритм має декілька розгалужень в роботі, які в залежності від вибору користувача виконують свої певні дії. Користувач користуючись браузером своєї операційної частини завантажує веб ресурс по заданому адресу та виконує чіткі дії які йому потрібні. Користувач має можливість пройти реєстрацію або авторизацію у системі

веб додатку, про це свідчить на схемі алгоритму роботи дві розгалужувальні гілки які йдуть після претендента. Виконуючи авторизацію користувач потрапляє у власну профільну частину веб додатку, і у іншому випадку, коли користувач обирає реєстрацію у системі йому надається можливість створення власного профілю у системі. Для цього він слідує всім вказівкам системи та після вже створюється для нього окремий профіль у системі. Після виконання всіх вище наведених кроків алгоритму користувач потрапляє у власну профільну сторінку користувача, тобто на персональну веб сторінку у веб додатку. Далі користувачу надається можливість обирати наступні дії, а саме відправку повідомлень, видалення повідомлень, редагування та вкінці при завершення роботи можливий вихід із власного профілю користувача тобто розлогінення.

Використання технології динамічного звернення до веб сервера на “льоту” без перевантаження всіє веб сторінки, наприклад :

- з використання XMLHttpRequest (основний об'єкт);
- через динамічне створення дочірніх фреймів;
- через динамічне створення тегу <script>.

Через динамічне створення тегу <img>, як це реалізовано Google analytics.

Використання DHTML для динамічного методу оновлення веб сторінки.

3.1.1 Функціональне призначення веб додатку

Інтенсивний розвиток інформаційного суспільства, швидка динаміка розвитку інформаційних технологій привели до необхідності оптимізації сервісів надання послуг. Функціональне призначення сайту:

Продаж деталей та надання ремонтних робіт.

Фінансова інформація про надання послуг з продажу товарів.

Інформування про інновації та розробки в сфері автомобільного ремонту.

Інтернет і численні розробки спеціалізованих електронних продуктів відкривають доступ до нових джерел пізнання інформації й істотно розширює інформаційне середовище.

Інтернет-комунікації реалізуються через:

Взаємодію клієнтів з інтернет-ресурсами (пошук, обробку й аналіз інформації); такі способи спілкування в Інтернеті, як чати, веб-конференції, електронна пошта;

самостійну роботу, використовуючи тематичну літературу, представлену в Інтернеті;

Таким чином, сформуємо завдання веб-сайту:

- комунікативні – отримання відповідей від спеціаліста, які клієнт йому задав за допомогою спілкування за допомогою форуму;
- інформаційні – розширення інформаційної бази за рахунок вільного доступу до інформації;
- методичні – знайомство з різними формами подання інформації, організація самостійної діяльності;

Рішення зазначених завдань виводить оптимізацію алгоритма на новий рівень, наповнюючи його інноваційними інформаційно-комунікативним змістом, що відповідає вимогам інформатизації й модернізації суспільства, сприяє формуванню й розвитку інформаційної, комунікативної, методичної й предметної компетентності.

Зберігання і накопичення є одними з основних дій, здійснюваних над інформацією і головним засобом забезпечення її доступності протягом деякого проміжку часу. В даний час визначальним напрямом реалізації цієї операції є концепція бази даних, в ролі складу (сховища) даних та технологія AngularJS для обміну даних. База даних може бути визначена як сукупність взаємозв'язаних даних, використовуваних декількома користувачами, які зберігаються з регульованою надлишковістю. Збереження даних, не залежать від програм

користувачів, для модифікації і внесення змін застосовується загальний управляючий метод який доцільно реалізувати за використанням JavaScript. Банк даних - система, що представляє певні послуги із зберігання і пошуку даних певній групі користувачів з певної тематики. Система баз даних – сукупність управляючої системи прикладного програмного забезпечення, бази даних, операційної системи і технічних засобів, що забезпечують інформаційне обслуговування користувачів. Ще одним важливим напрямом розвитку баз даних є репозиторії. Репозиторій, в спрощеному вигляді, можна розглядати просто як базу даних, призначену для зберігання не користувацьких, а системних даних. Технологія репозиторіїв виникає із словників даних, які у міру збагачення новими функціями і можливостями набували рис інструменту для управління метаданими де знову ж таки умисно використовувати JavaScript. Кожен з учасників дії (користувач, група користувачів, «фізична пам'ять») має своє представлення про інформацію. По відношенню до користувачів застосовують трирівнева представлення для опису предметної області: концептуальне, логічне і внутрішнє (фізичне) (Рис. 3.1).

Рисунок 3.1. – Опис предметної області збереження даних

Концептуальний рівень пов'язаний з особистим представленням даних групи користувачів у вигляді зовнішньої схеми, що об'єднуються спільністю використовуваної інформації. Кожен конкретний користувач працює з частиною БД і представляє її у вигляді зовнішньої моделі. Цей рівень характеризується

різноманітністю використовуваних моделей (модель «сутність-зв'язок», ER-модель, модель Чена, бінарні і інфологічні моделі, семантичні мережі). На рисунку 3.2 представлений фрагмент наочної бази даних «Збут» і одне з можливих його концептуальних представлень, яке відображає не тільки об'єкти і їх властивості, але і взаємозв'язки між ними.

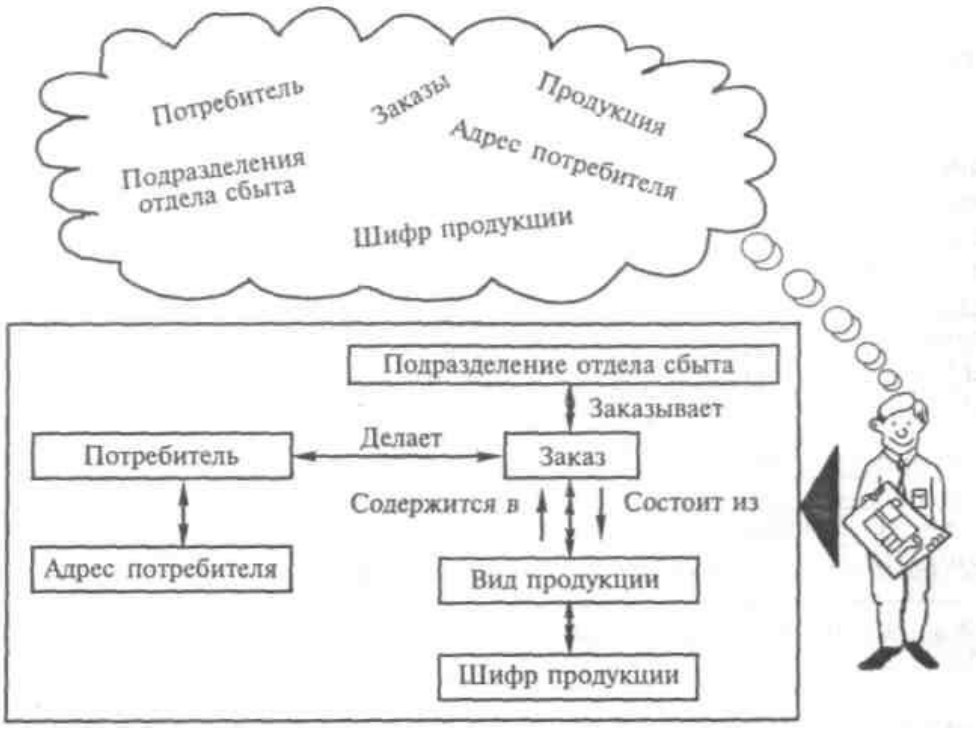


Рисунок 3.2. – Фрагмент наочної бази даних «Збут» і одне з його можливих концептуальних представлень

База даних – сукупність даних, організованих відповідно до концепції, яка описує характеристику цих даних і взаємозв'язки між їх елементами; ця сукупність підтримує щонайменше одну з областей застосування . В загальному випадку база даних містить схеми, таблиці, подання, збережені процедури та інші об'єкти. Дані у базі організовують відповідно до моделі організації даних. Таким чином, сучасна база даних, крім саме даних, містить їх опис та може містити засоби для їх обробки.

В загальному випадку базою даних можна вважати будь-який впорядкований набір даних. Наприклад, паперову картотеку з формулярами про працівників

підприємства у відділі кадрів. Але дана стаття зосереджена на використанні баз даних в інформаційних системах. На даний час застосунки для роботи з базами даних є одними з найпоширеніших прикладних програм.

Модель функціонування програми можна побачити на діаграма прецедентів, що представлена на рисунку 3.3.

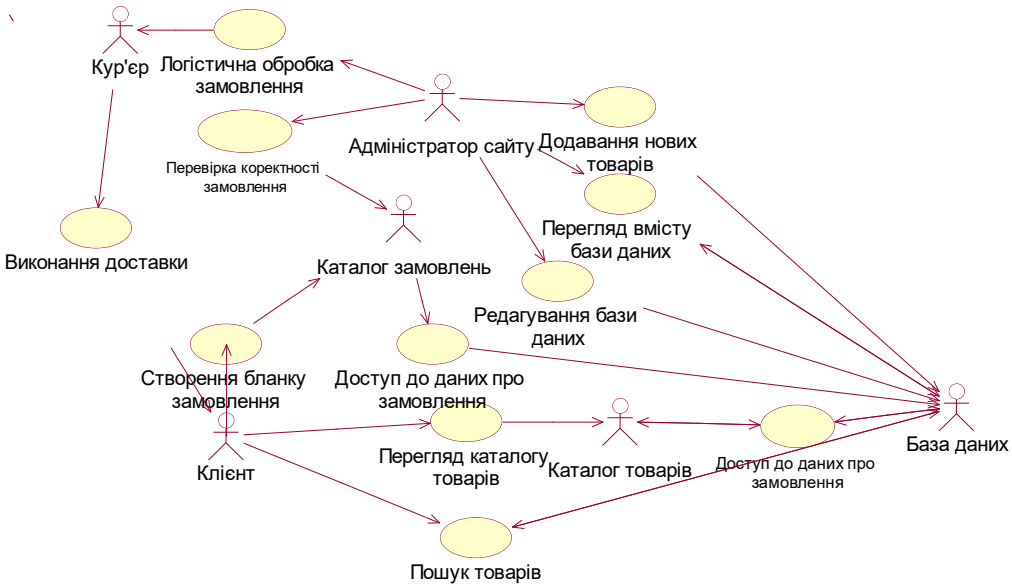


Рисунок 3.3. – UML UseCase діаграма веб – орієнтованої системи

Діаграма прецедентів — в UML, діаграма, на якій зображено відношення між акторами та прецедентами в системі. Також, перекладається як діаграма варіантів використання.

3.1.2. Сховище даних веб додатку

Сховище даних — предметно орієнтований, інтегрований, незмінний набір даних, що підтримує хронологію і здатний бути комплексним джерелом достовірної інформації для оперативного аналізу та прийняття рішень. В основі концепції сховища даних (СД) лежить розподіл інформації, що використовують в системах оперативної обробки даних (OLTP) і в системах підтримки прийняття

рішень (СППР). Такий розподіл дозволяє оптимізувати як структури даних оперативного зберігання для виконання операцій введення, модифікації, знищення та пошуку, так і структури даних, що використовуються для аналізу. В СППР ці два типи даних називаються відповідно оперативними джерелами даних (ОДД) та сховищем даних.

Призначення сховища даних - інформаційна підтримка прийняття рішень, а не оперативна обробка даних.

Основні принципи організації сховищ даних наступні :

1. Предметна орієнтація. У оперативній базі даних зазвичай підтримується декілька предметних областей, кожна з яких може послужити джерелом даних для СХД. База даних і сховище даних не є однаковими поняттями. Архітектура СХД представлена на рисунку 3.4.

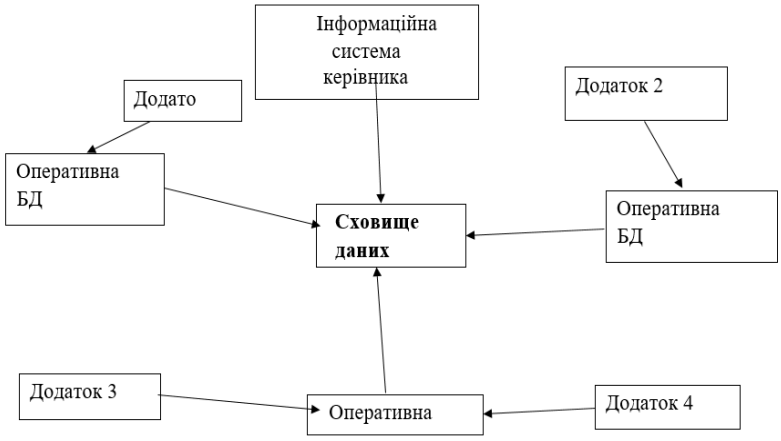


Рисунок 3.4. – Архітектура СХД

Наприклад, для магазина, торгуючого відео- і музичною продукцією, інтерес представляють наступні предметні області: клієнти, відеокасети, CD-диски і аудіокасети, співробітники, постачальники. Явно простежується аналогія між предметними областями СХД і класами об'єктів в об'єктно-орієнтованих базах даних. Це говорить про можливість застосування методів проектування, вживаних в об'єктно-орієнтованих СУБД.

2.Засоби інтеграції. Приведення різних представлень однієї і тієї ж суті до деякого загального типу.

3.Постійність даних. У СХД не підтримуються операції модифікації в сенсі традиційних баз даних. У СХД підтримується модель «масових завантажень» даних, здійснюваних в задані моменти часу по встановлених правилах на відміну від традиційної моделі індивідуальних модифікацій.

4. Хронологія даних. Завдяки засобам інтеграції реалізується певний хронологічний часовий аспект, властивий вмісту СХД.

Основні функції репозиторіїв:

- парадигма включення/виключення і деякі формальні процедури для об'єктів;
- підтримка множинних версій об'єктів і процедури управління конфігураціями для об'єктів;
- повідомлення інструментальних і робочих систем про події, що цікавлять їх;
- управління контекстом і різні способи огляду об'єктів репозиторія;
- визначення потоків робіт.

Розглянемо коротко основні напрями наукових досліджень в області баз даних:

- розвиток теорії реляційних баз даних;
- моделювання даних і розробка конкретних моделей різноманітного призначення;
- відображення моделей даних, направлених на створення методів їх перетворення і конструювання комутативних відображень, розробку архітектурних аспектів відображення моделей даних і специфікацій визначення відображень для конкретних моделей даних;
- розробка, вибір і оцінка методів доступу;
- створення само описуваних баз даних, що дозволяють застосовувати єдині методи доступу даних і метаданих;

- розвиток системипрограмування баз даних і знань, які додатків, так і для управління даними;
- вдосконалення машини баз даних;
- розробка дедуктивних баз даних, заснованих на застосуванні апарату математичної логіки і засобів логічного програмування, а також просторово-часових баз даних;
- інтеграція неоднорідних інформаційних ресурсів.

3.1.3. Виклик і завантаження

Інтерфейс - це, говорячи з розумним вираженням обличчя, певна сукупність програмних засобів, які забезпечують взаємодію графічної оболонкаи з її користувачем. Дана програма використовує загально прийнятий інтерфейс браузера.

В нашому проекті використовується взаємодія WWW-сервера з запуском програми CGI – Common Gateway Interface. Наш вибір програмного засобу – мова програмування Python. WWW — інформаційна система, якій не можна дати конкретного визначення.

Браузер — програмне забезпечення для комп'ютера або іншого електронного пристрою, як правило, під'єднаного до Інтернету, що дає можливість користувачеві взаємодіяти з текстом, малюнками або іншою інформацією на гіпертекстовій вебсторінці. Тексти та малюнки можуть містити посилання на інші вебсторінки, розташовані на тому ж вебсайті або на інших вебсайтах. Вебпереглядач з допомогою гіперпосилань дозволяє користувачеві швидко та просто отримувати інформацію, розміщену на багатьох вебсторінках. Вебпереглядач під'єднується до сервера HTTP, отримує з нього документ і форматує його для представлення користувачеві або намагається викликати зовнішню програму, яка це зробить, залежно від формату документа. Найпершим вебпереглядачем був Mosaic, розроблений в Національному центрі застосування суперкомп'ютерів (NCSA) Іллінойського університету в Урбана-Шампейн.

Викликати нашу WEB інформаційну систему можемо з любого WEB браузера встановленого на комп’ютері клієнта.

3.2. Структурна схема нейронної мережі

Віддаленість від клієнта необхідна для сортування фінальних результатів, від найближчого, до найдальшого. Структура СНМ представлена на рисунку 3.5.

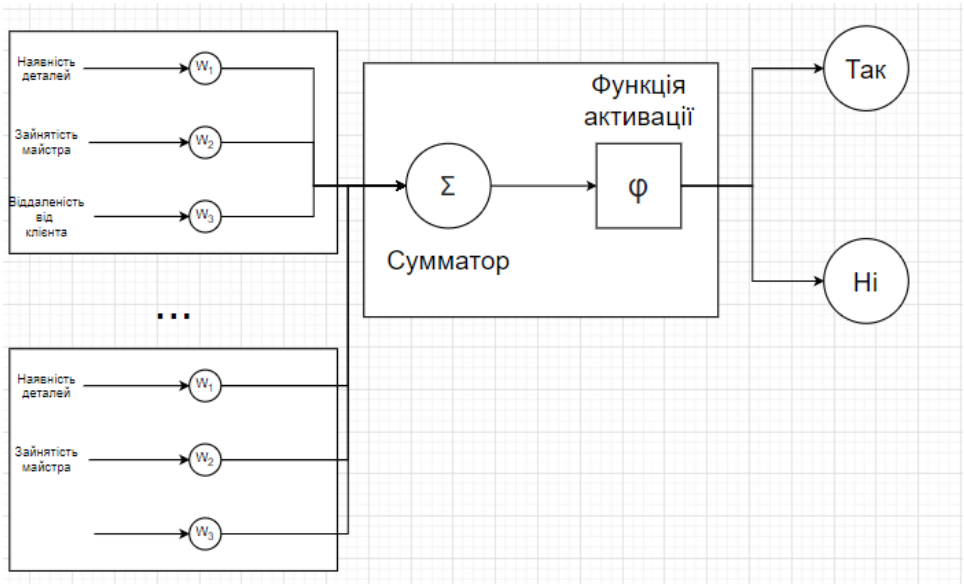


Рис. 3.5. Структурна схема нейронної мережі

Розроблена нейронна мережа перебирає данні з усіх автосервісів, до даних яких має доступ, та визначає наявність необхідних деталей, наявність вільного часу у майстра та віддаленість від клієнта. Далі ці данні передаються в суматор, який сумує ваги отриманих характеристик і викликає функцію активації, яка в залежності від отриманого результату робить однозначний висновок, чи підходить даний автосервіс під вимоги користувача.

Наявність деталей та наявність вільного часу у майстра є абсолютно необхідними критеріями, якщо хоча б один з них не буде справджуватися — автосервіс автоматично відкидається як непідходящий.

3.2.1. Розробка клієнтського веб-сервісу

Ідея сервіс-орієнтованої архітектури і мікросервісної архітектури як її підвиду в тому, що ви поділяєте весь свій великий проект на окремі сервіси, які можуть працювати незалежно і обмінюються даними з іншими частинами проекту.

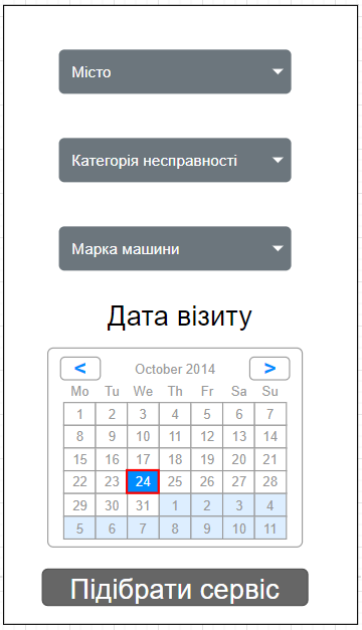


Рис. 3.6. Головна форма сервісу

Кожен веб-сервіс - це окремий проект, в якому можливо збудувати ієрархії з сервісів, коли одні сервіси використовують інші. Такий підхід сприяє масштабування системи і команди, над системою можуть одночасно працювати десятки програмістів без проблем з пошуком складних логічних помилок і взаємозалежності одних частин системи від інших. Приклад головної форми сервісу представлено на рисунку 3.6

Вебслужба, вебсервіс (англ. web service) — програмна система, що ідентифікується URI, і публічні інтерфейси та прив'язки якої визначені та описані мовою XML. Опис цієї програмної системи може бути знайдено іншими програмними системами, які можуть взаємодіяти з нею відповідно до цього опису

з використанням повідомлень, що базуються на XML та передаються за допомогою інтернет-протоколів.

Дуже важливим та зручним є можливість кліентів відразу в сервісі бачити карту з потрібними точками шляху. Приклад відображення результатів на мапі представлено на рисунку 3.7

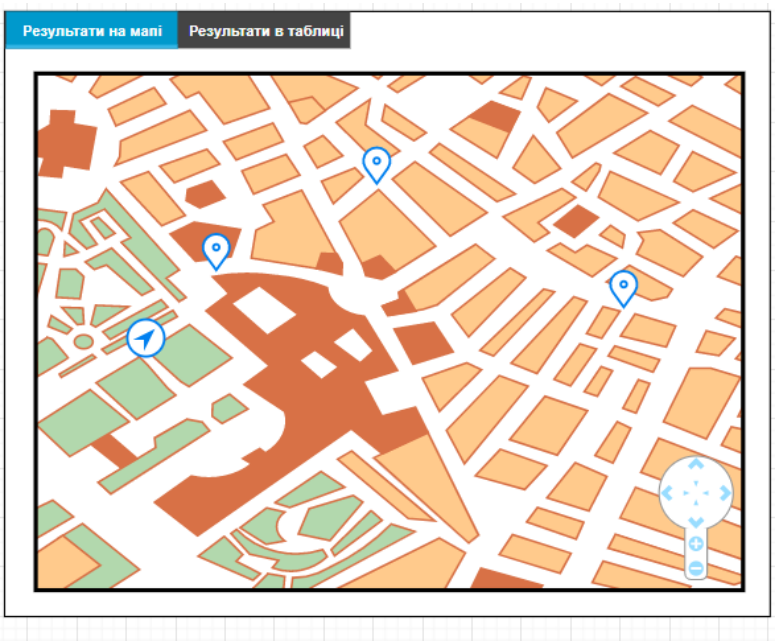


Рис. 3.7. Відображення результатів на мапі

Електро́нна ка́рта місце́вості - цифрова модель, що вміщує та відображає інформацію про об'єкти місцевості та їх географічні і інші атрибути, яка призначена для використання в геоінформаційних системах.

Кожна конкретна електронна карта є засобом оперативного контролю і існує лише в певний момент часу, як правило нетривалий, поки видно на пристрої відеозображення. У цьому їх головна відмінність від інших візуальних картографічних матеріалів, відображених на твердій підкладці (папір, пластик) засобами графічного виводу (наприклад, принтерами).

На рис. 3.8. а також 3.6 та 3.7 зображено макети графічного інтерфейсу користувача, які включають в себе введення даних щодо несправностей та бажаного часу візиту, виведення результатів у формі мапи та таблиці.

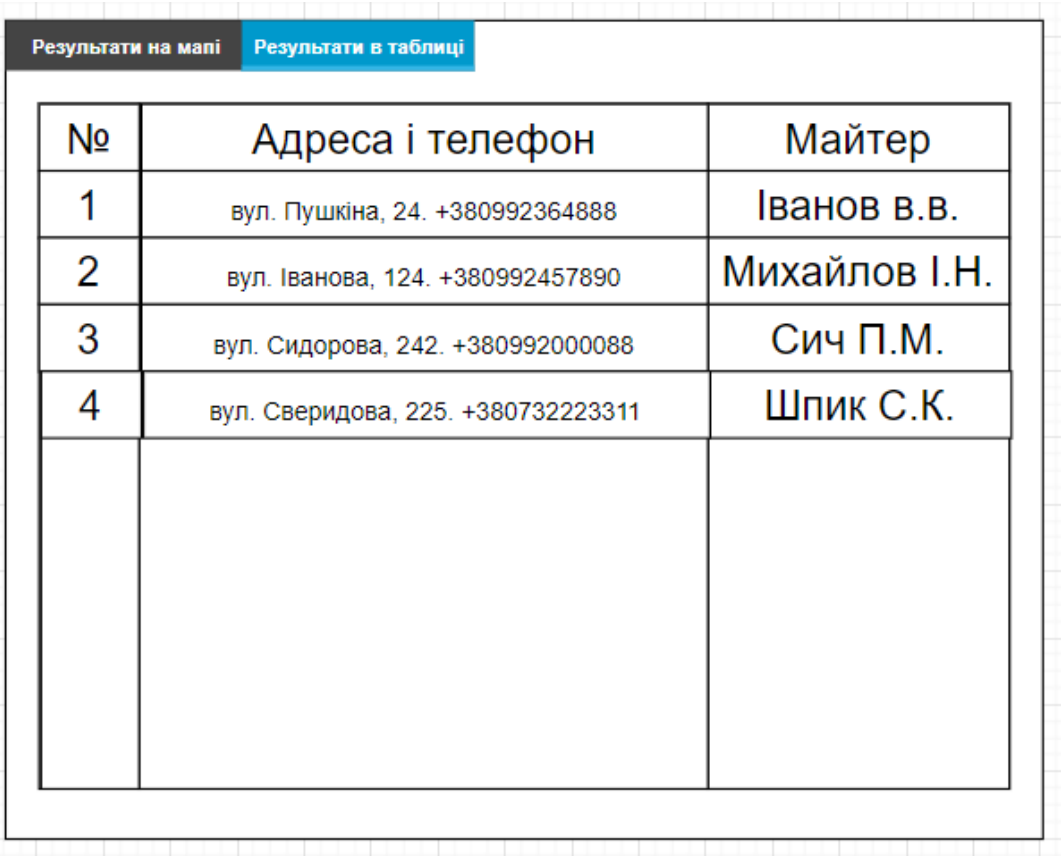


Рис. 3.8. Відображення результатів в формі таблиці

3.2.2. Інструкція адміністративної частини

Адміністративна частина - це простір з обмеженим доступом, в якому адміністратори і автори можуть писати статті або новини, змінювати структуру сайту, моніторити і обмінюватися повідомленнями, які розміщуються на форумах, і так далі. Щоб дістатися до адміністративної частини, потрібна реєстрація, тобто логін і пароль.

Стосовно адміністрування на сайті було створено каталог де будуть викладені сторінки адміністрування, у хост – провайдера дана папка буде закрита певним логіном і паролем. Для входу в адміністративну частину набираємо в

адресному рядку браузера http://magazin/admin/login, після входу бачимо головну сторінку рис 3.9.

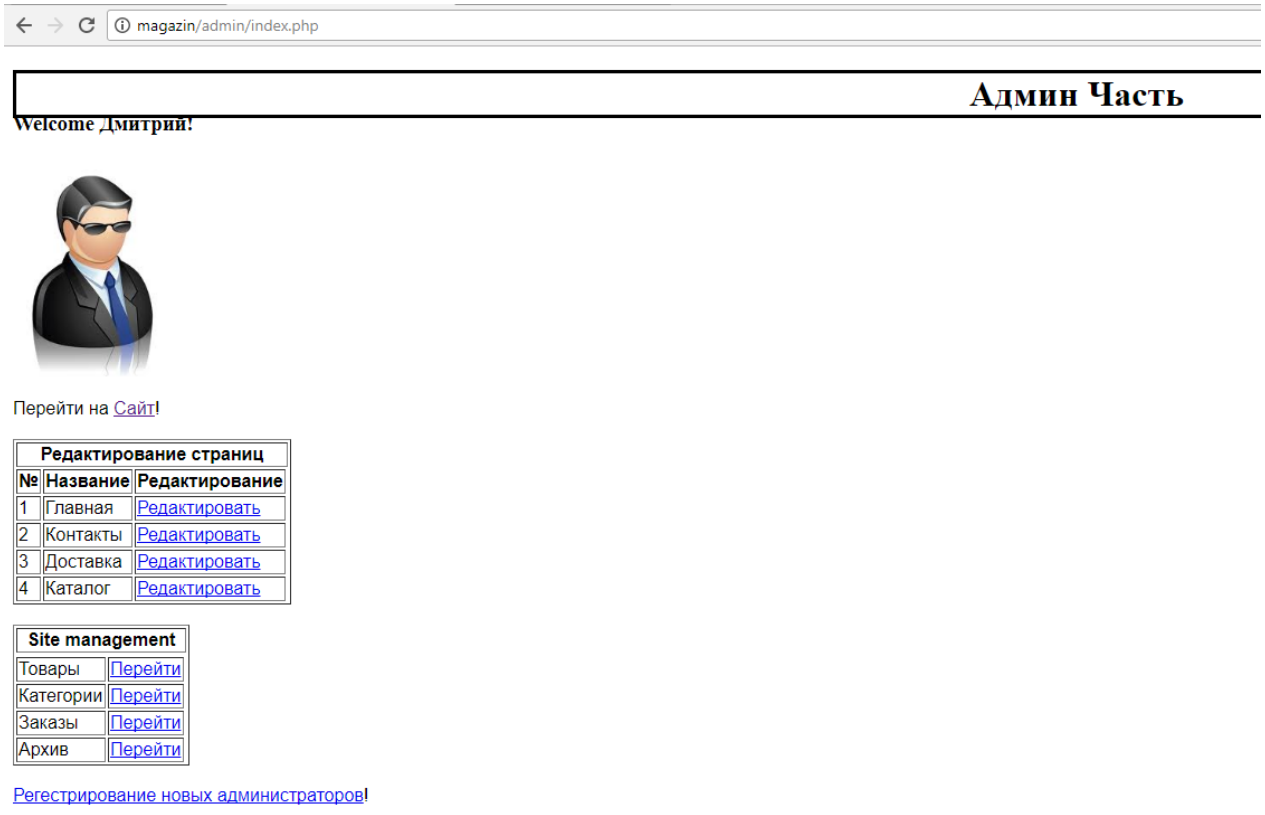


Рисунок 3.9 – Головна сторінка адміністративної частини

Як Ви можете спостерігати зліва меню яке дозволяє працювати з певними розділами сайту. Для того щоб сайт частіше відвідували на сайті більш широко описана діяльність фірми. Треба зазначити що всі сторінки сайту завантажуються з бази даних динамічно і в залежності від потреб власника можуть змінювати своє наповнення.

Редагування — це приведення об'єкта редагування у відповідність із чинними у певний час у конкретному суспільстві нормами, а також його творча оптимізація, метою яких є отримання заданого соціального ефекту.

Складається з двох рівноправних процедур: контролю (аналізу) та виправлення (реконструкції) авторського оригіналу.

Приклад редагування сторінок сайту представлено на рисунку 3.10.

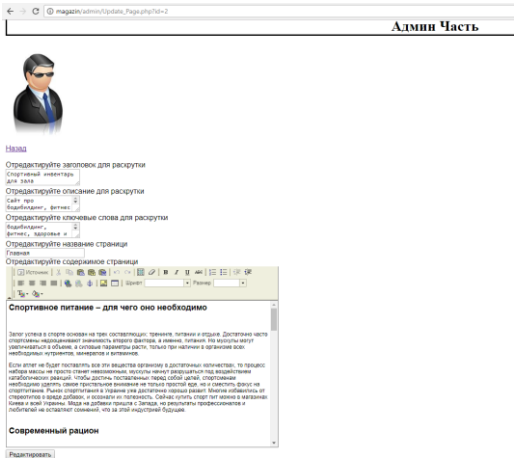
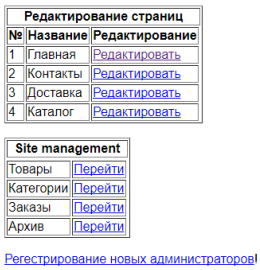


Рисунок 3.10 – Редагування сторінок сайту

Зміна інформації на сторінці особливо важлива, для того щоб потенційний клієнт відвідав сайт як можна частіше..На рисунку 3.11 представлено головну сторінка адміністративної частини.



Редактирование страниц		
№	Название	Редактирование
1	Главная	Редактировать
2	Контакты	Редактировать
3	Доставка	Редактировать
4	Каталог	Редактировать

Site management	
Товары	Перейти
Категории	Перейти
Заказы	Перейти
Архив	Перейти

Регестрирование новых администраторов

Рисунок 3.11 - Головна сторінка адміністративної частини.

Та для того щоб ресурс краще знаходили в інтернеті, його контент можна оптимізувати і вдосконалювати по даній тематиці. Метою редагування є трансляція («перетворення») повідомлень для отримання заданого соціального ефекту.

3.3.Тестування та аналіз роботи програми

На етапі розробки програми виникали помилки різного роду, що не завжди відразу можна було виявити і зрозуміти у чому їх причина. Розробка програми пройшла декілька етапів.

1-й етап. Розробка основних сторінок. На цьому етапі виникали такі помилки:

- дизайн сайту можна назвати умовним, але основний акцент був на програмування;
- крім того перелік розділів, на головній сторінці був не виразний, і збоку робочої сторінки.

2-й етап. Розробка адміністративної частини.

Цей етап можна вважати найбільш складний, бо потребував уважного написання коду Python та запитів до бази даних.

На цьому етапі виникали такі помилки:

По перше коли користувач редагував зміст сторінок сайту вони не зберігалися в базі даних, помилка виявилась в написані назви скрипта який виконується після відправки форми.

Крім того коли редагували матеріали які використовуються, вони не зберігалися в базі даних, це відбувалося тому, що в запиті SQL були помилки в назві полів бази даних.

3.4. Структура інформаційної системи

Клієнт, використовуючи клієнтський веб-додаток вводить необхідні данні, які передаються на сервер.

Сервер, отримавши дані від клієнтської частини, підтягує дані автосервісів за допомогою інтерфейсів розробників (АПІ) та парсерів,що продемонстровано на рисунку 3.12, які збирають дані на ресурсах різноманітних автосервісів та передає

сукупність клієнтських та зібраних власноруч даних нейронній мережі, яка вибирає параметри «вільний час» та «наявність запчастин», місцеположення автосервісів і на базі сукупностей цих факторів робить висновки щодо того, чи підходить даний автосервіс даному конкретному клієнту.

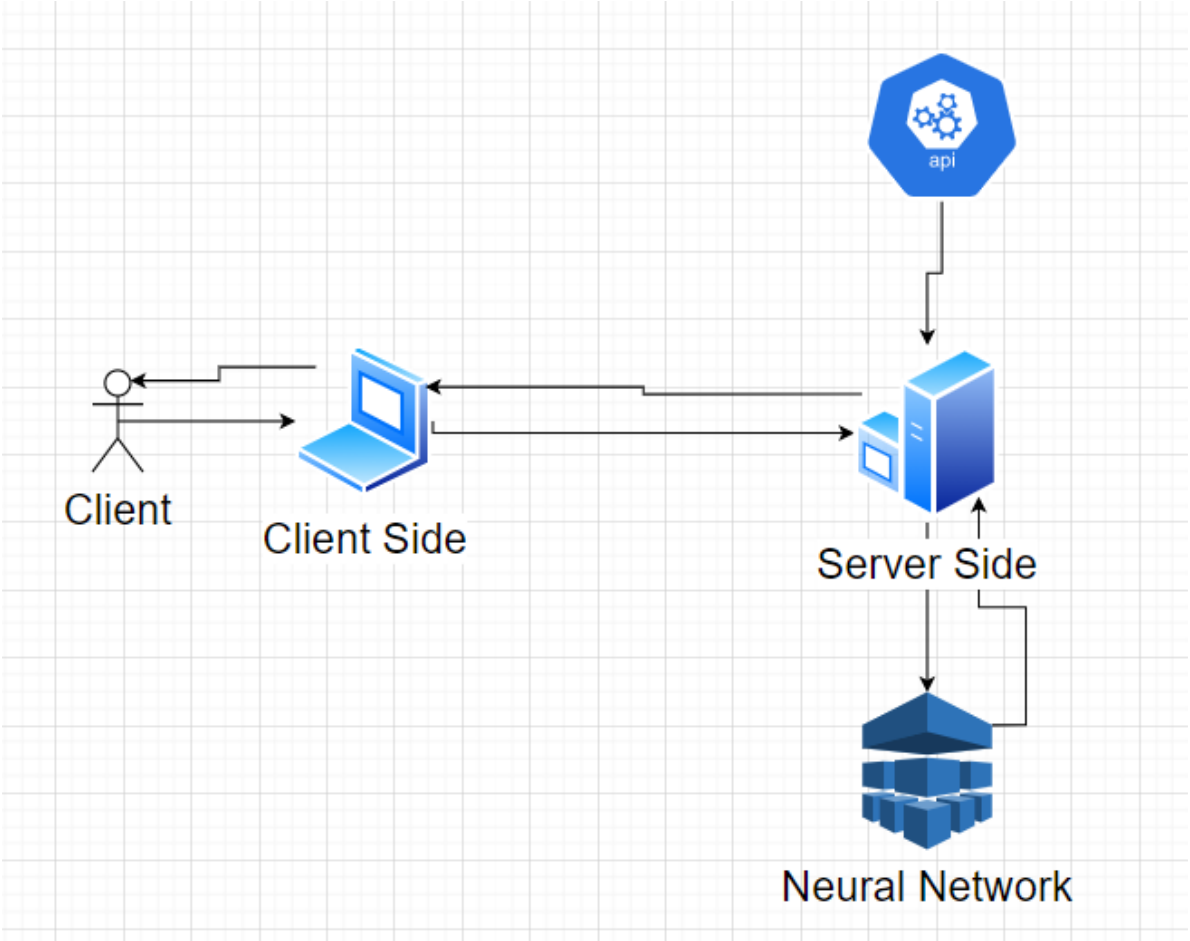


Рис. 3.12. Загальна структура інформаційної системи

Нейронна мережа передає сукупність результатів з вузла «Так» назад на сервер. Сервер — у комп'ютерній термінології термін може стосуватися окремого комп'ютера чи програми. Головною ознакою в обох випадках є здатність машини чи програми переважну кількість часу працювати автономно, без втручання людини, реагуючи на зовнішні події відповідно до встановленого програмного забезпечення. Сервер сортує результати виходячи з значення параметру

«розташування» та відправляє дані на клієнську частину, яка займається вимальовуванням результатів на мапі та в таблиці.

3.5. Наукова новизна розробленого продукту

Агент вибирає дії на основі правила, що збільшує майбутню винагороду. RL на основі моделі (рис. 1.3 а) може бути корисним при виборі рішення, але його дуже важко використовувати через складність моделювання хмарної інфраструктури в цій RL.

Вирішення цієї проблеми полягає у виборі без модельної версії RL (рис. 1.3 б). Рішення на користь QL було також пов'язано з тим, що він має оптимальну політику для отримання негайної та відстроченої винагороди. Через високу складність та складність хмарного середовища не вдалося отримати повну інформацію для належного налаштування діяльності. Завдяки FQL ви можете використовувати неповні знання. Нечіткі правила вдосконалюються під час збору даних у режимі реального часу.

Агент вибирає дії на основі правила, що збільшує майбутню винагороду. RL на основі моделі (рис. 1.3 а) може бути корисним при виборі рішення, але його дуже важко використовувати через складність моделювання хмарної інфраструктури в цій RL.

Вирішення цієї проблеми полягає у виборі без модельної версії RL (рис. 1.3 б). Рішення на користь QL було також пов'язано з тим, що він має оптимальну політику для отримання негайної та відстроченої винагороди. Через високу складність та складність хмарного середовища не вдалося отримати повну інформацію для належного налаштування діяльності. Завдяки FQL ви можете використовувати неповні знання. Нечіткі правила вдосконалюються під час збору даних у режимі реального часу.

В даній роботі новизну отриманих результатів слід визначати по таким параметрам:

- Новизна використання технологій
- Новизна технічного рішення на ринку
- Новизна технічного рішення відносно об’єкту дослідження
- Наукова новизна ідеї розробки
- Наукова новизна поєднання технологій

3.5.1. Новизна використання технологій

З точки зору новизни використання сучасних технологій слід звернути увагу на незвичайну зв’язку технологій, які використовуються для того, щоб пов’язати клієнтську сторону та серверну сторону.

В нашому конкретному випадку серверна сторона реалізована за допомогою мови програмування Python, так як вона найбільш часто використовується для побудови і написання нейронних мереж різного роду, але, окрім цього, вона використовується і для написання серверних частин різного роду програмного забезпечення, побудованого на базі архітектури «клієнт-сервер».

Для побудови клієнтської частини було використано мову TypeScript та фреймворк Angular, як стек технологій, доступний в розумінні і використанні, який часто використовується для розробки клієнтських частин за рахунок простої схеми зв’язку з серверами на інших мовах програмування, без необхідності використання сторонніх модулів.

Новизна полягає в самій ідеї зв’язки Python — Angular.

З точки зору розробника — такий стек технологій є максимально зручним і простим у використанні для fullstack-розробників. Проте, не зважаючи на велику кількість переваг, такого роду зв’язки використовуються вкрай рідко, надаючи перевагу більш складним системам.

3.5.2. Новизна технічного рішення на ринку

З точки зору ринку розроблений програмний продукт є абсолютно новаторським, адже у вільному доступі є маса парсерів, інтерфейсів розробників, тощо, самих різних специфікацій і тематик, проте об’єднання декількох варіацій різних програмних продуктів — ідея нова і нереалізована на сучасному ринку програмного забезпечення.

Пошук тієї чи іншої інформації за заданими параметрами за допомогою нейронних мереж — взагалі нерозвідана тема в сфері безкоштовного програмного забезпечення, а тим паче в межах обраної тематики.

Така новизна продукту пов’язана з тим, що нейронними мережами користуються, як правило, закриті корпорації, які використовують їх для збору і класифікації різного роду інформації, наприклад відвідувачів сайту, класифікації цільової аудиторії, розпізнавання обличь, тощо.

3.5.3. Новизна технічного рішення відносно об’єкту дослідження

З точки зору об’єкту дослідження дане технічне рішення є цілком новим і нерозробленим. Автоматизація роботи підприємств, подібних автосервісам, можна назвати темою, яка не розробляється зовсім.

Процес пошуку клієнтів в підприємствах такого характеру, як правило, відбувається за допомогою реклами в інтернеті, в паперових носіях інформації, яскравих вивісок та усного спілкування.

Виходячи з вищесказаного, клієнти не мають можливості заздалегідь дізнатися, підходить чи ні той чи інший автосервіс. Звертаючи увагу на цей фактор, можна зробити висновок про високу новизну розробленого програмного забезпечення відносно об’єкту дослідження.

3.5.4. Наукова новизна ідеї розробки

Ідея розробки уніфікованого програмного забезпечення, що поєднує в собі парсери, інтерфейси розробників та нейронну мережу є абсолютно новою і може претендувати на звання «новатора» в темі розробки програмного забезпечення.

Зазвичай, розроблюване програмне забезпечення має лише частковий функціонал того, що було розроблено в рамках даної роботи. Наприклад, набір парсерів, які збирають дані з різних сайтів по короткому переліку параметрів, або навпаки нейронна мережа, яка залишає користувачеві пошук і збір не класифікованої інформації.

В нашому ж випадку, розроблена інформаційна система звільняє користувача від збору взагалі будь-якої інформації.

ВИСНОВКИ

Робота присвячена дослідженню методів машинного навчання Q-навчання.

У першому розділі описано цілі курсової роботи та сформульовано завдання, які слід виконувати в ній.

Перший розділ був також присвячений вивченню предмета. Було розглянуто область машинного навчання, а Q-навчання було розглянуто більш детально. Принцип роботи агента, що слід враховувати навколишньому середовищу та як встановлюються цілі та винагорода. Також було розглянуто марковську властивість і марковський процес прийняття рішень.

Розглянуті приклади застосування методики Q-навчання машинного навчання.

Описані властивості Маркова потребують більшості згаданих вище властивостей і дуже точні та значущі при використанні.

Таким чином специфіка даних програмного забезпечення не дозволяє їм швидко застосовуватися в надзвичайних ситуаціях, а марковські властивості вимагають велику кількість специфічних даних. Виходячи з цих фактів, важливим завданням є питання використання моделі та створення прототипу системи, що дозволить оперативно прогнозувати роботу у сфері надання послуг автомобільного сервісу.

Веб - технології набули високого рівня та складають значну частину програмного забезпечення в цілому. Також веб технології інтегрувалися в життя людини, полегшуючи та прискорюючи життєві процеси. Вони використовуються буквально у всіх сферах IT-побуту. Веб - технології не стоять на одному місці і продовжують розвиватися і AngularJS одна з найцікавіших можливостей. На думку експертів, в майбутньому AngularJS буде у більшості веб ресурсів.

На мій погляд, мова програмування Python це одна з найперспективніших технологій в нашому веб -світі.

Використання цих підходів дозволяє створювати набагато зручніші веб-інтерфейси користувача на тих сторінках сайтів, де необхідна активна взаємодія з користувачем.

Зробивши детальний системний аналіз та обґрунтувавши проблему застосування AngularJS можемо прийти до висновку що технологія необхідна в реальному житті.

Це дає змогу розробити певний та чіткий алгоритм застосування та в свою чергу приводить до правильного підходу розробки веб продуктів. У нашому випадку було проведено аналіз системи обміну інформації для веб сайту. Було створено алгоритм роботи системи, проаналізувавши структуру роботи та виділені основні аспекти розв'язку завдання. Найголовнішою метою другого розділу був аналіз проблеми та побудова дерева проблеми. Після висвітлення матеріалу цього розділу, стає очевидним що застосування технології AngularJS є важливою необхідністю. Вкінці даного розділу було створено критеріальні, алгоритмічні оцінки роботи системи з використанням Angular. Проведено дослідження пояснює шляхи вирішення проблеми, та відповідає на питання навіщо потрібен Angular. Розроблена єдина методика системного аналізу застосування.

Отже як бачимо при проектуванні та розробці веб систем необхідною є використання технології Angular в профільній частині користувача.

В третьому розділі було розглянуто основні теоретичні шляхи вирішення проблеми дипломного проектування. Було розглянуто концептуальний алгоритм роботи додатку та його всі основні функціональні можливості, побудова діаграм uml для характеристики роботи додатку а також було розглянуто наукову новизну розробленого продукту.

Отже виконання системного аналізу дає нам можливість побачити шляхи вирішення завдання дипломного проектування, розробка реального проекту показала вузькі місця застосування Angular та Python та реальні переваги всієї технології в цілому. Виходячи із альтернативних шляхів вирішення завдання бачимо додаткові шляхи вирішення деяких з проблем поставленого завдання.

Оглядаючи всі критерії щодо роботи веб додатку стає зрозумілим та точнішим виконання посаленого завдання. Загалом можна вважати що головне завдання системних досліджень було вирішено в дипломному проекті, а також застосовані ефективні методи та засоби дослідження й управління об'єктами. В цілому технологія AngularJS значно полегшує використання веб додатків, роблячи їх значно зручнішими, швидкими та надійними.

СПИСОК ПОСИЛАНЬ

1. An Essay towards solving a Problem in the Doctrine of. Режим доступу: https://royalsocietypublishing.org/doi/pdf/10.1098/rstl.1763.0053 Дата звернення: 04.05.2020.
2. A COMPUTING MACHINERY AND INTELLIGENCE Режим доступу: https://phil415.pbworks.com/f/TuringComputing.pdf Дата звернення: 07.05.2020.
3. SNARC “Stochastic neural analog reinforcement calculator”[. Режим доступу: https://historyof.ai/snarc/ Дата звернення: 02.04.2020.
4. Стюарт Рассел, Пітер Норвіг, «II - сучасний підхід. Режим доступу: http://i.uran.ru/webcab/system/files/bookspdf/iskusstvennyy-intellekt-sovremennyy-podhod/229021.pdf Дата звернення: 04.05.2020.
5. Моделювання процесів навчання в нейронних мережах.. Режим доступу: http://old.exponenta.ru/soft/others/mvs/stud3/3.asp. Дата звернення: 04.05.2020.
6. Саттон Р.С Навчанняя з підкріпленням. / Саттон Р.С, Э. Г. Барто // БИНОМ, Лаборатория знаний, 2014 – С. 42-96. Дата звернення: 04.05.2020.
7. CS234: Reinforcement Learning Режим доступу: http://web.stanford.edu/class/cs234/index.html. Дата звернення: 09.05.2020
8. JavaScript|MDN. Режим доступу: https://developer.mozilla.org/uk/docs/Web/JavaScript. Дата звернення: 04.06.2020.
9. ECMAScript 2020 Language Specification. Режим доступу: https://www.ecma-international.org/ecma-262/10.0/index.html#Title. Дата звернення: 24.08.2020.
10. React - Javascript бібліотека для створення користувацьких інтерфейсів. Режим доступу: https://uk.reactjs.org. Дата звернення: 12.06.2020.
11. ReactDOM – React. Режим доступу: https://uk.reactjs.org/docs/react-dom.html. Дата звернення: 15.06.2020.
12. TypeScript - JavaScript that scales. Режим доступу: https://www.typescriptlang.org/index.html. Дата звернення: 21.05.2020.

13. Google Maps Platform - Google Developers. Режим доступу: https://developers.google.com/maps/documentation. Дата звернення: 24.05.2020.Charts - Google Developer. Режим доступу: https://developers.google.com/chart/?hl=uk. Дата звернення: 27.05.2020.

14. ApexChart.js. Режим доступу: https://apexcharts.com. Дата звернення: 12.05.2020.

15. Kuhlman, Dave. "A Python Book: Beginning Python, Advanced Python, and Python Exercises". Section 1.1. Archived from the original (PDF) on 23 June 2012.

16. "About Python". Python Software Foundation. Retrieved 24 April 2012., second section "Fans of Python use the phrase "batteries included" to describe the standard library, which covers everything from asynchronous processing to zip files."

17. Peterson, Benjamin (20 April 2020). "Python Insider: Python 2.7.18, the last release of Python 2". Python Insider. Retrieved 27 April 2020.

18. "History and License". Retrieved 5 December 2016. "All Python Releases are Open Source"

19. The RedMonk Programming Language Rankings: January 2020. Режим доступу: https://redmonk.com/sogrady/2020/02/28/language-rankings-1-20/

20. Python Is More Popular Than Ever. Режим доступу: https://www.wired.com/story/python-language-more-popular-than-ever/

21. Venners, Bill (13 January 2003). "The Making of Python". Artima Developer. Artima. Retrieved 22 March 2007.

22. Van Rossum, Guido (29 August 2000). "SETL (was: Lukewarm about range literals)". Python-Dev (Mailing list). Retrieved 13 March 2011.

23. "Why was Python created in the first place?". General Python FAQ. Python Software Foundation. Retrieved 22 March 2007

24. van Rossum, Guido (20 January 2009). "A Brief Timeline of Python". The History of Python. Retrieved 20 January 2009.

25. Fairchild, Carlie (12 July 2018). "Guido van Rossum Stepping Down from Role as Python's Benevolent Dictator For Life". Linux Journal. Retrieved 13 July 2018.

26. "Guido van Rossum Stepping Down from Role as Python's Benevolent Dictator For Life | Linux Journal". www.linuxjournal.com.

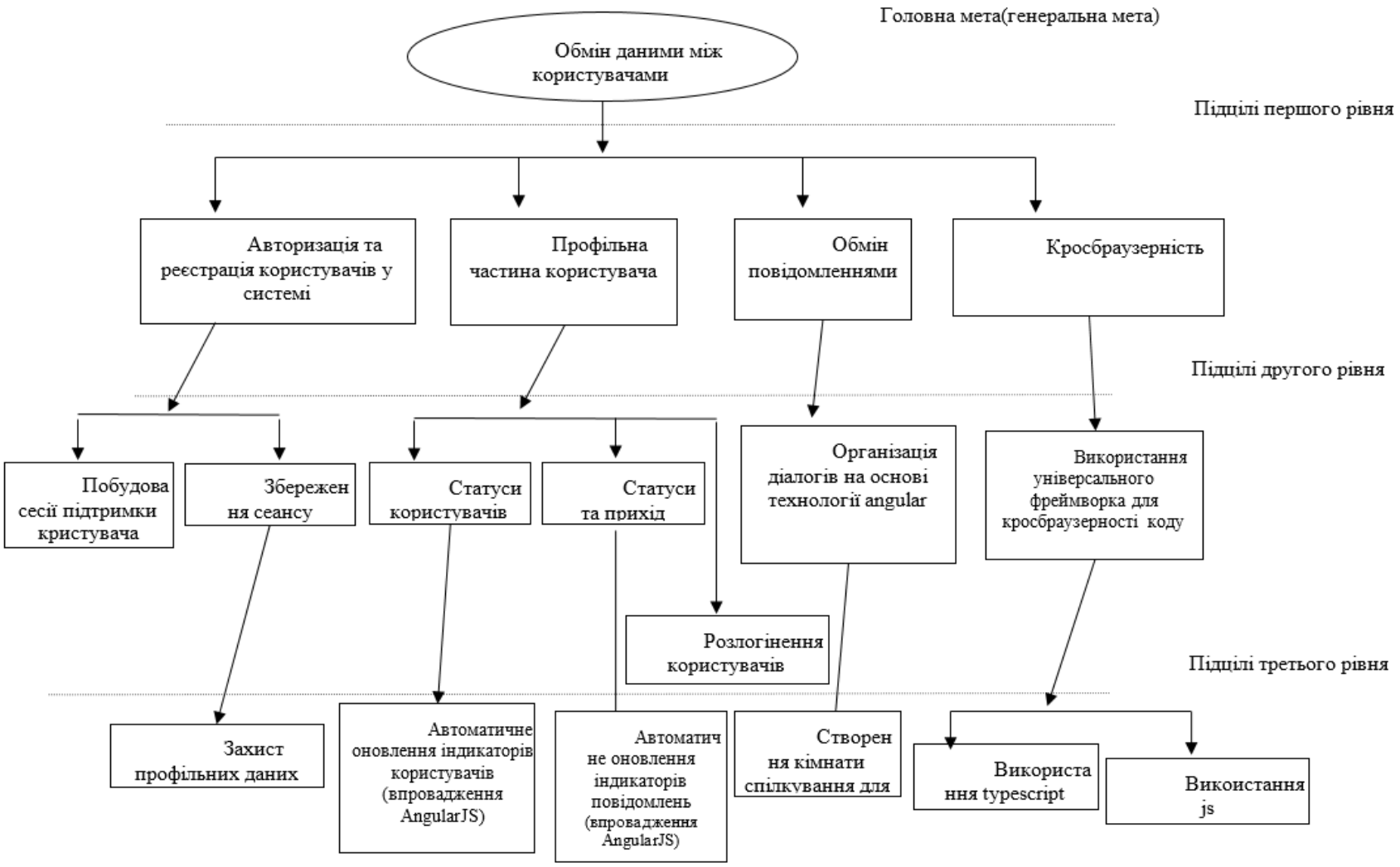
27. "Python boss Guido van Rossum steps down after 30 years". The Inquirer.

28. "PEP 8100". python. Python Software Foundation. Retrieved 4 May 2019.

29. "Python Developer's Guide — Python Developer's Guide". devguide.python.org. Retrieved 17 December 2019.

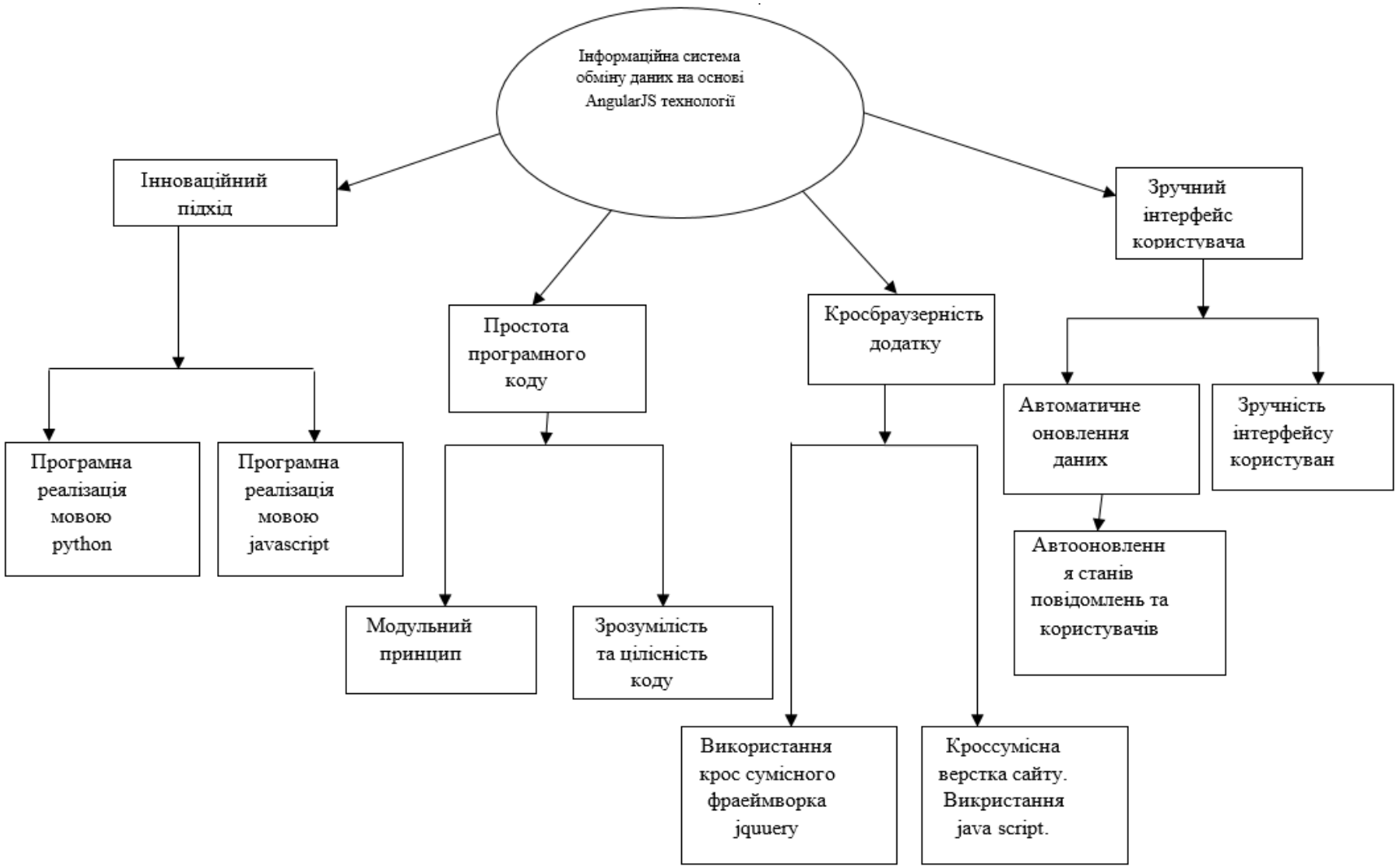
30. The Cain Gang Ltd. "Python Metaclasses: Who? Why? When?" (PDF). Archived from the original (PDF) on 30 May 2009. Retrieved 27 June 2009.

Додаток А



Діаграма цілей системного аналізу

Додаток Б



Діаграма дерева завдань

Демонстраційні матеріали

ДЕРЖАВНИЙ УНІВЕРСИТЕТ ТЕЛЕКОМУНІКАЦІЙ

ФАКУЛЬТЕТ ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ

КАФЕДРА ІНЖЕНЕРІЇ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

Оптимізація в сфері надання послуг автомобільного сервісу за допомогою методів машинного навчання

Виконав студент Групи ПДМ-61

Тараканов Дмитро Сергійович

Науковий керівник: доцент кафедри Шевченко С.М.

Київ 2020

МЕТА РОБОТИ. ОБ'ЄКТ ТА ПРЕДМЕТ ДОСЛІДЖЕННЯ

Мета роботи – вдосконалення роботи в сфері надання послуг автомобільного сервісу за допомогою розробленої інформаційної системи обміну данними між користувачами основаній на методах машинного навчання.

Об`єкт дослідження – низька швидкість роботи інформаційної системи обміну даних між користувачами в мережі Інтернет.

Предмет дослідження – практична розробка веб-додатку для підбору потрібного сервісного центру для клієнта основуючись на наявності деталей та присутності вільного часу у правильного спеціаліста з використанням новітніх технологій у цій галузі.

Актуальність

- Вирішення проблеми витрати часу на знаходження спеціаліста та наявності деталей у сервісі для якомога найшвидшого отримання результату.
- В сучасному суспільстві значну роль відіграють інформаційні технології. Фактично, в сьогоденні вони використовуються в будь-якій сфері людського життя. Поміж усіх інформаційних технологій, в сьогоднішньому світі особливе місце займають технології машинного навчання. Все більше і більше компаній починають використовувати ці технології для покращення роботи з клієнтами.

Наукова новизна

- Розробка програмного продукту, що має можливість підбору потрібного сервісного центру для клієнта та об’єднання декількох варіацій різних програмних продуктів в один.
- З точки зору ринку розроблений програмний продукт є абсолютно новаторським, адже у вільному доступі є маса парсерів, інтерфейсів розробників, тощо, самих різних специфікацій і тематик, проте об’єднання декількох варіацій різних програмних продуктів — ідея нова і нереалізована на сучасному ринку програмного забезпечення.
- Пошук тієї чи іншої інформації за заданими параметрами за допомогою нейронних мереж — нерозвідана тема в сфері безкоштовного програмного забезпечення, а тим паче в межах обраної тематики.

ТЕХНІЧНЕ ЗАВДАННЯ

Розроблення веб-додатку використовуючи технології програмування з технологіями машинного навчання, що спрощують та прискорюють роботу з конфігурацією веб-застосунків.

Функції програми повинні включати в себе:

- Реєстрацію користувачів
- Авторизацію користувачів.
- Відправку повідомлень.
- Редагування повідомлень.
- Знаходження потрібного сервісу враховуючи вимоги клієнта.
- Значне збільшення швидкості роботи програми.

Аналоги

Логотип	Назва	Призначення	Переваги	Недоліки
	SPOTIFY	• модель коллаборативної фільтрації, яка порівнює ваші музичні уподобання до переваг інших користувачів • модель обробки природної мови, яка працює на основі аналізу тексту • аудіо-модель, яка аналізує необроблене аудіо	Spotify став найпопулярнішим у світі музичним потоковим сервісом завдяки саме такій потужній системі рекомендацій. Як би рідко ви не слухали музику і якою б своєрідною вона не була - алгоритми порадять вам щось схоже.	•громідзкий інтерфейс; •вимагає дуже багато часу для опрацювання інформації та видання результату; •повільна швидкість роботи.
	LINKEDIN	ІІ-асистент, який допомагає шукачам складати резюме: помічник підбирає схожі анкети в потрібній області і дає поради. Також LinkedIN пропонує використовувати різні терміни, що дозволяють зробити документ більш інформативним і цікавим для роботодавців.	LinkedIn використовує комбінацію алгоритмів машинного навчання з роботою оператора-людини. Це дозволяє класифікувати контент за якістю.	•Дуже повільно працює у режимі реального часу •Потребує досить велику кількість часу для моменту коли зможе видавати правильний результат без помилок •Багато помилок на початку роботи
	SALESFORCE	Salesforce.com - американський розробник однойменної CRM-системи, що надається замовникам виключно за моделлю SaaS (Software as a Service).	У 2016 році компанія запустила інтеграцію з Einstein AI і додала нові можливості в систему в 2017 році. Einstein - це асистент на основі ІІ, призначений для аналізу даних в сфері CRM. Його головною функцією є допомога компаніям виявляти, прогнозувати, рекомендувати і автоматизувати найбільш ефективні бізнес-процеси.ІІ-асистент використовує Data Mining і МО для прогнозування майбутніх показників продажів компанії.	•Працює лише на англійській та німецьких мовах. •Велика швидкість лише на початку роботи програми, потім йде велике падіння швидкості

Порівняльна діаграма

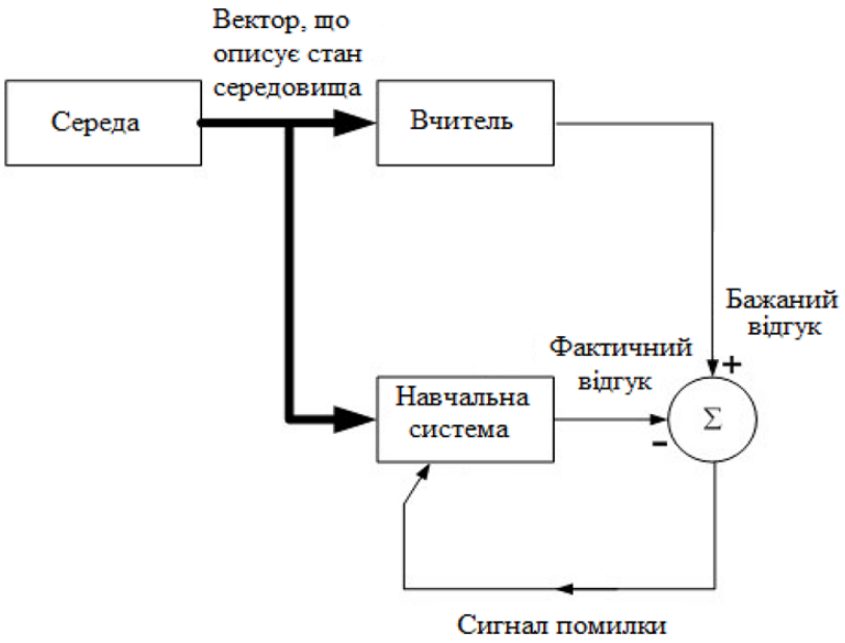
Марковська властивість

У теорії ймовірностей та статистиці термін ма́рковська власти́вість відноситься до властивості відсутності пам'яті[en] в стохастичного процесу. Стан має властивість Маркова і, отже, є марківським станом тоді і лише тоді, коли:

$$\Pr\{st{+}1 = s' \, , \, rt{+}1 = r \, | st, \, at, \, rt, \, st{-}1, \, at{-}1, \, rt{-}1. \, . \, . \, . \, , \, r1, \, s0, \, a0 \, \}$$
$$=$$
$$\Pr\{st{+}1 = s' \, , \, rt{+}1 = r \, | st, \, at\}$$

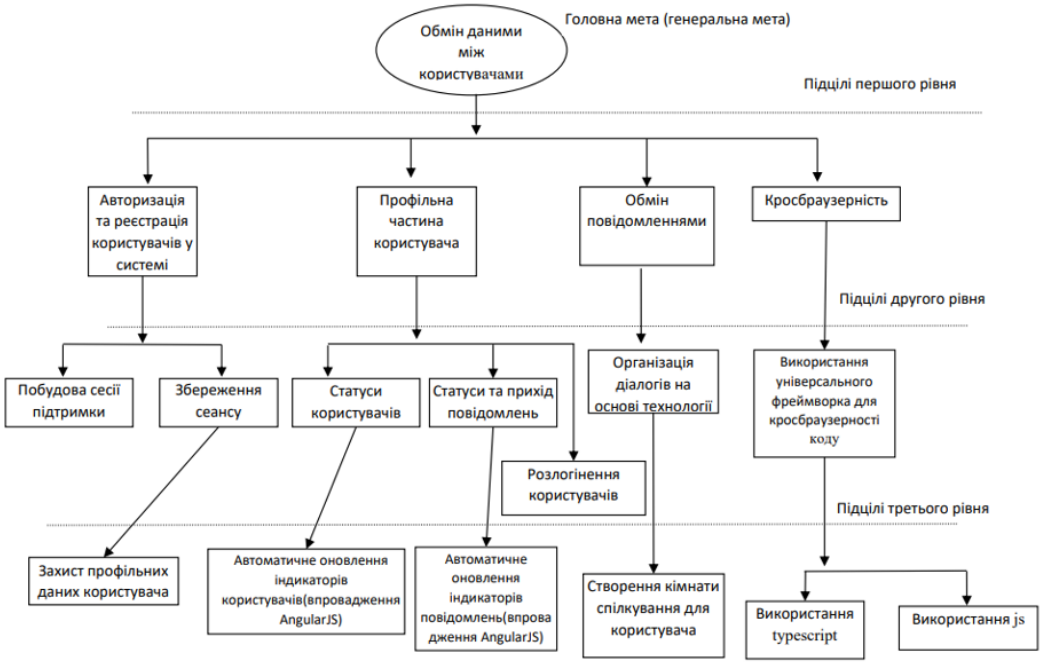
Під час навчання з підкріпленням рішення агента виступають як сигнал від навколишнього середовища, який називається станом навколишнього середовища. В ідеалі цей сигнал стану повинен коротко підсумувати минулі відчуття, щоб зберегти всю необхідну інформацію. Сигнал стану, який містить всю необхідну інформацію, називається сигналом Маркова.

НАВЧАННЯ ІЗ УЧИТЕЛЕМ



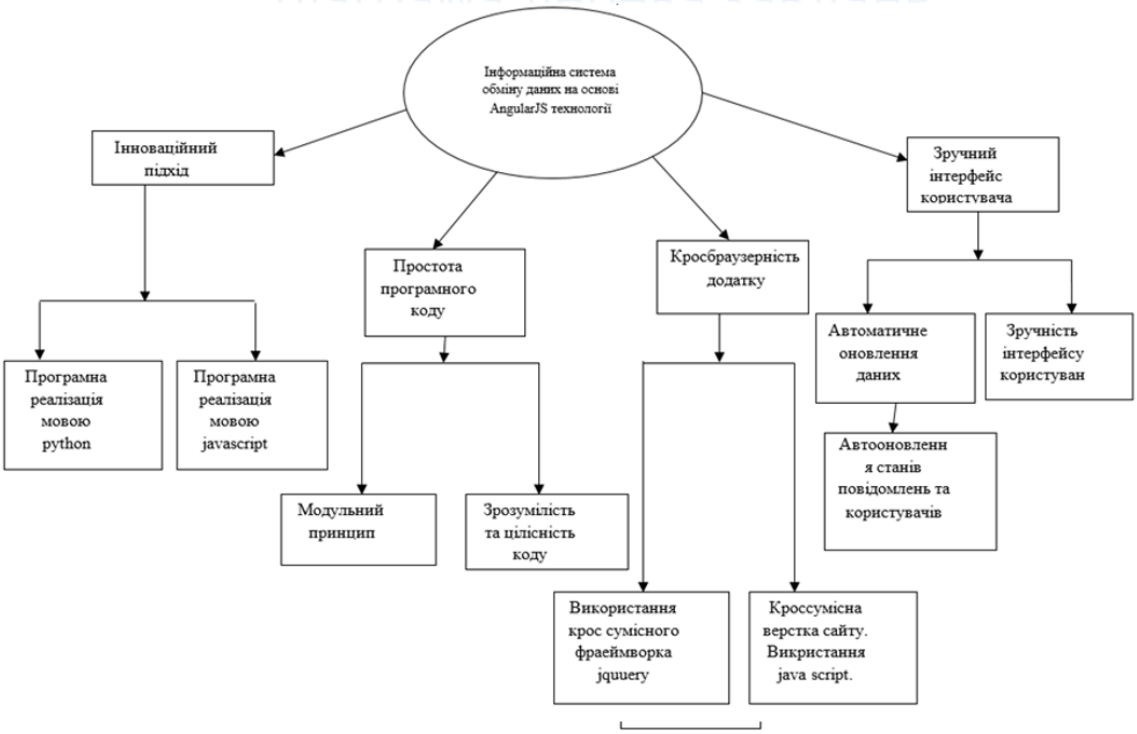
Діаграма цілей системного аналізу

Додаток А

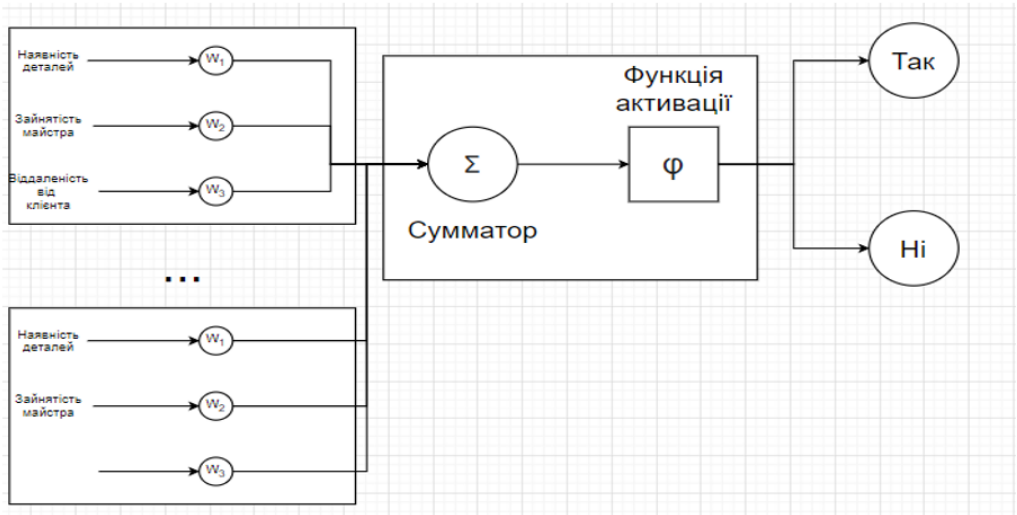


Діаграма цілей системного аналізу

Діаграма дерева завдань



Структурна схема проекту

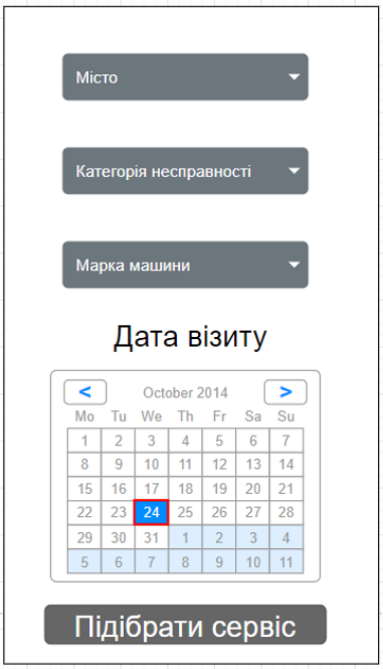


Функція Ейлера $\varphi(n)$, де n — натуральне число, — це цілочисельна функція, яка показує кількість натуральних чисел, що є меншими за n і взаємно простих з ним.
Функцію Ейлера можна подати у вигляді так званого добутку Ейлера:

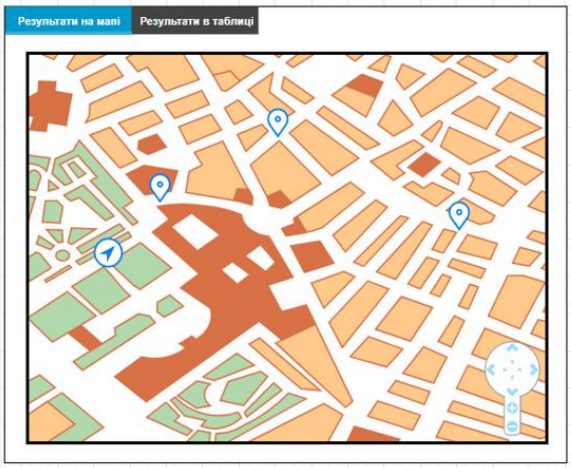
$$\varphi(n) = n \prod_{p|n} \left(1 - \frac{1}{p}\right),$$

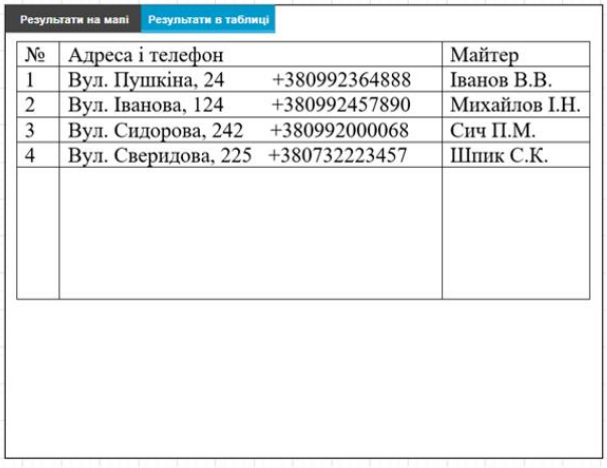
де p — просте число.

МАКЕТ ГРАФІЧНОГО ІНТЕРФЕЙСУ



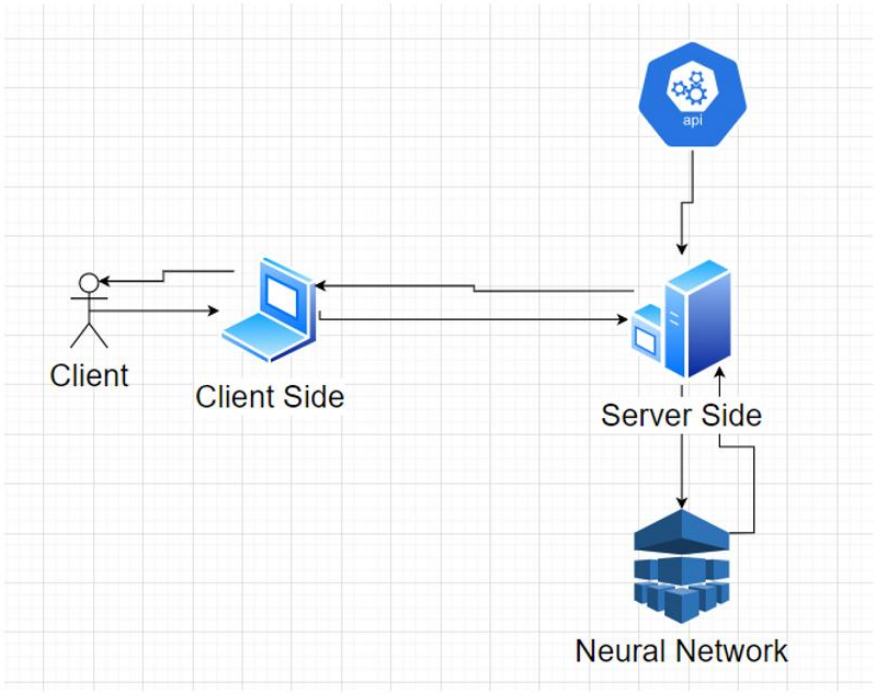
ВИГЛЯД РОЗРОБЛЕНОГО ІНТЕРФЕЙСУ





№	Адреса і телефон		Майтер
1	Вул. Пушкіна, 24	+380992364888	Іванов В.В.
2	Вул. Іванова, 124	+380992457890	Михайлов І.Н.
3	Вул. Сидорова, 242	+380992000068	Сич П.М.
4	Вул. Сверидова, 225	+380732223457	Шпик С.К.

СТРУКТУРА ІНФОРМАЦІЙНОЇ СИСТЕМИ



Спеціалізовані технології

Програмний продукт був створений за допомогою мови програмування python та TypeScript, а також було використано веб-систему Angular JS

Обрані мови програмування: Python, TypeScript.

Використані додаткові програмні засоби:

1. OPEN SERVER - це портативна серверна платформа і програмне середовище, створене спеціально для веб-розробників.
2. Ps — графічний редактор, який використовувався для створення графічних матеріалів.

Висновки та апробація результатів магістерської роботи

Головним результатом дипломної роботи є вдосконалення роботи в сфері надання послуг автомобільного сервісу за допомогою розробленої інформаційної системи обміну даними між користувачами основаній на методах машинного навчання та розгортання веб проекту в якому застосовано веб-систему AngularJS.

В результаті проведеної роботи:

- розроблено та прийнято проектне рішення де застосовується Q-learning;
- досліджено переваги Q-learning;
- досліджено недоліки Q-learning;
- розроблено опис функціональності об’єкту дослідження;
- було розроблено веб-додаток з використанням AngularJS та Q-learning для підвищення ефективності роботи з інтерфейсом та реєстрацією;
- розроблена інформаційна система, яка звільняє користувача від збору взагалі будь-якої інформації.

В цілому технологія AngularJS в зв’язку з Q-learning значно полегшує використання веб додатків, роблячи їх значно зручнішими, швидшими та надійними.

Апробація результатів магістерської роботи.

Даний дипломний проект пройшов апробацію на XI НАУКОВО-ТЕХНІЧНІЙ КОНФЕРЕНЦІЇ "СУЧАСНІ ІНФОКОМУНІКАЦІЙНІ ТЕХНОЛОГІЇ" у дут, м.Київ, 12 грудня 2020 року, а також опублікована стаття на тему "МАШИННЕ НАВЧАННЯ ЯК МЕТОД ОПТИМІЗАЦІЇ АЛГОРИТМУ ВЗАЄМОДІЇ У СФЕРІ НАДАННЯ ПОСЛУГ АВТОМОБІЛЬНОГО СЕРВІСУ" у 6-му випуску журнала "Зв'язок".

Дякую за увагу!