МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

ІНЖЕНЕРНИЙ НАВЧАЛЬНО-НАУКОВИЙ ІНСТИТУТ
ім. Ю.М. Потебні
ЗАПОРІЗЬКОГО НАЦІОНАЛЬНОГО УНІВЕРСИТЕТУ

КАФЕДРА ЕЛЕКТРОНІКИ, ІНФОРМАЦІЙНИХ СИСТЕМ ТА ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

Кваліфікаційна робота

другий (магістерський)
(рівень вищої освіти)

на тему **Особливості застосування генетичного алгоритму при розробці мобільного застосунку для системи обслуговування**

Виконав: студент 2 курсу, групи 8.1211-2іпз
спеціальності 121 Інженерія програмного забезпечення
(код і назва спеціальності)
освітньої програми Інженерія програмного забезпечення
(код і назва освітньої програми)
М.Ю. Калашник
(підпис, ініціали та прізвище)
Керівник доцент Заяц В.1.,
(посада, вчене звання, науковий ступінь, підпис, ініціали та прізвище)
Рецензент директор ТОВ Дискус П.О. Лютий
(посада, вчене звання, науковий ступінь, підпис, ініціали та прізвище)

Запоріжжя
2022

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ІНЖЕНЕРНИЙ НАВЧАЛЬНО-НАУКОВИЙ ІНСТИТУТ
ім. Ю.М. Потебні
ЗАПОРІЗЬКОГО НАЦІОНАЛЬНОГО УНІВЕРСИТЕТУ

Кафедра Електроніки, інформаційних систем та програмного забезпечення
Рівень вищої освіти _____другий (магістерський)_____
Спеціальність ___121_Інженерія програмного забезпечення_____
(код та назва)
Освітня програма ___Інженерія програмного забезпечення
(код та назва)

ЗАТВЕРДЖУЮ

Завідувач кафедри_____Т.В. Критська_____
“_12_” __вересня_____2022_року

З А В Д А Н Н Я
НА КВАЛІФІКАЦІЙНУ РОБОТУ СТУДЕНТОВІ

Калашник Максим Юрійович
(прізвище, ім’я, по батькові)

1.Тема роботи _____Особливості застосування генетичного алгоритму при розробці мобільного застосунку для системи обслуговування___
керівник роботи ___Заяц Валерій Іванович, доцент
(прізвище, ім’я, по батькові, науковий ступінь, вчене звання)
затверджені наказом ЗНУ від 02.06.2022 р. №597-с

2. Строк подання студентом кваліфікаційної роботи 1 грудня 2022 р.
3. Вихідні дані магістерської роботи
- комплект нормативних документів ;
- технічне завдання до роботи.

4. Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити)
- огляд та збір літератури стосовно теми кваліфікаційної роботи;
- огляд та аналіз існуючих рішень та аналогів;
- дослідження проблеми розпізнавання мов та розробка методів її вирішення;
- створення програмного продукту та його опис;
- перелік вимог для роботи програми;
- дослідження поставленої проблеми та розробка висновків та пропозицій.

5. Перелік графічного матеріалу (з точним зазначенням обов’язкових креслень)

6. Консультанти розділів магістерської роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата
		Завдання прийняв

7. Дата видачі завдання 01.09.2022

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів магістерської роботи	Строк виконання етапів магістерської роботи	Примітка
1	Аналіз предметної області	02.06-17.06.2022	виконано
2	Формулювання основної задачі дипломної роботи та узгодження її з науковим керівником	18.06-19.06.2022	виконано
3	Аналіз існуючих рішень	20.06-8.07.2022	виконано
4	Дослідження сучасних методів реалізації генетичного алгоритму	09.07-20.07.2022	виконано
5	Розробка веб застосунка (серверної частини)	21.07-21.09.2022	виконано
6	Розробка модуля та аналіз його інтеграції в застосунок	22.09-07.10.2022	виконано
7	Узгодження подальших дій з науковим керівником	08.10-9.10.2022	виконано
8	Розробка back частини застосунка	10.10-28.10.2022	виконано
9	З'єднання модуля с генетичним алгоритмом до серверної частини	29.10-07.11.2022	виконано
10	Представлення отриманих результатів науковому керівнику та узгодження плану подальшого дослідження	08.11-09.11.2022	виконано
11	Аналіз отриманих результатів дослідження	10.11-17.11.2022	виконано
12	Оцінка доцільності використання генетичного алгоритму при розробці мобільного застосунка	18.11-22.11.2022	виконано
13	Оформлення звіту	22.11-30.11.2022	виконано

Студент _____ М.Ю. Калашник
(підпис) (ініціали та прізвище)

Керівник роботи _____ В.І. Заяц
(підпис) (ініціали та прізвище)

Нормоконтроль пройдено

Нормоконтролер _____ І.А. Скрипник
(підпис) (ініціали та прізвище)

ЗМІСТ

ВСТУП

Актуальність теми

Розвиток інформаційних технологій, зокрема штучного інтелекту, робить повсякденне життя людини більш оптимізованим та безпечним. Завдяки цьому дослідженню можна вирішити проблему з підбором індивідуального меню з урахуванням потреб кожного користувача застосунку та зменшити витрати на кваліфікований обслуговуючий персонал компанії.

Мета і завдання дослідження

Мета і завдання дослідження полягає у вивченні генетичного алгоритму та створенні якісного та швидкого веб-сервісу, який буде використовуватися у мобільному застосунку будь-якої платформи та буде повністю автоматизований і не вимагатиме зайвого втручання спеціалістів, тим самим зменшить ризик зіткнення із людським фактором.

Об’єкт дослідження

Об’єктом дослідження є реальні клієнтські параметри та їх побажання щодо страв, способів харчування та індивідуальних потреб.

Предмет дослідження

Предметом дослідження є додаток з їстівним меню, яке буде відповідати вимогам клієнта та всім нормам нутринології.

Методи дослідження

Для вирішення поставленої задачі використовуються наступні методи дослідження:

1. Аналізування існуючих рішень та різних модифікацій для генетичного алгоритму.

2. Аналізування використання унікальних систем штрафів для досягнення поставлених цілей.
3. Аналізування допоміжних бібліотек та програмного забезпечення.
4. Повне розуміння предметної області застосунку.

Наукова новизна одержаних результатів

Наукова новизна одержаних результатів дослідження полягає у тому, що для вирішення задачі автоматизації створення їстівного меню з урахуванням всіх потреб клієнта був використаний генетичний алгоритм пошуку. Після розгляду конкурентів, це буде унікальна система, яка застосовується вперше з цією метою.

Практичне значення одержаних результатів

Практичне значення одержаних результатів дослідження полягає у тому, що була розроблена ціла екосистема пов’язаних між собою сервісів, яка працює в реальному часі та виконує всі поставлені задачі за короткий термін виконання скрипту.

Апробація одержаних результатів

Результатом дослідження є користування застосунком по всьому світу. Цей застосунок буде включати в себе два способи оплати (підписку та через відділ продажу) та приносити дохід компанії. Таким чином ці алгоритми візьмуть на себе виконання ручної роботи нутринологів і тим самим знизять витрати компанії.

Глосарій

Схрещування — один із кроків для досягнення результату в генетичному алгоритмі. Основна його ідея: взяти випадкову пару двох батьків і перемішати між собою фрагменти їх хромосом, в результаті цього

отримуємо новий набір генів. Після завершення обміну, дані додаються до популяції.

Популяція — містить в собі інформацію заданої кількості хромосом.

Мутація — виникає випадково змінює значення окремих генів.

Хромосома — бойова одиниця популяції.

Індивід — інша назва хромосоми.

Генетичний алгоритм — це алгоритм пошуку, який використовується для вирішення задач оптимізації та моделювання шляхом випадкового підбору, комбінування та варіації шуканих параметрів з використанням механізмів, аналогічних природного відбору.

Монетизація — одержання прибутку з проекту за рахунок введення в нього платних послуг. В моєму випадку — підписка на мобільних платформах та прямий продаж курсів.

Нутриціологія — наука про харчування. Вона вивчає вплив харчування на організм людини, питання поповнення дефіциту вітамінів, взаємодію мікро- та макроелементів тощо.

Нутриціолог — це фахівець з правильного та збалансованого харчуванню. Ця людина розбирається у хімічному складі продуктів, їх негативних та позитивних властивостях, взаємодії різних поживних речовин, способах приготування та кратності прийому їжі з метою запобігання розвитку захворювань.

QA інженер — фахівець, який тестує різноманітні програми, застосунки та сервіси для того, щоб переконатися, що вони працюють коректно, виявити ймовірні помилки та вразливості.

Ген — єдина складова хромосоми.

Добір — природній відбір. Іншими словами: із популяції вибираємо найкращі хромосоми та утворюємо із них нову популяція, а всі інші хромосоми покидають участь в генерації.

Нутрієнти — основні споживчі матеріали для правильного функціонування організму людини. До них відносяться: калорії, білки, жири, вуглеводи, клітчатка та всі похідні від них матеріали.

РОЗДІЛ 1 ДОСЛІДЖЕННЯ ПРОБЛЕМИ ГЕНЕТИЧНОГО АЛГОРИТМУ ДЛЯ СИСТЕМИ ОБСЛУГОВУВАННЯ

1.1 Огляд проблем для бізнесу

На сьогоднішній день найактуальнішими для компанії є такі проблеми:

1. Великі витрати на кваліфікованих людей — нутриціологів, які індивідуально для кожного клієнта збирають меню по харчуванню.
2. Великі витрати часу цих людей для збору даних клієнтів і генерування меню.
3. Не завжди вистачає часу для обробки кожного клієнта і клієнти можуть покинути проект.

1.2 Огляд застосунків конкурентів

На першому етапі було розглянуто застосунки конкурентів які були доступні в App Store (для телефонів з операційною системою IOS) та Play Market (для телефонів з операційною системою Android. Застосунки фільтрувалися за категоріями: “Фітнес”, “Здорове харчування”, “Спорт” тощо.

Під час аналізу було розглянуто близько 20 застосунків і через те, що жоден із них не міг в реальному часі зібрати меню для харчування, ми зіткнулися з двома варіантами альтернатив алгоритму:

1. Меню виписувалося кваліфікованою людиною, яка спеціалізується у сфері нутриціології, через що процес підготовки меню затягувався на 3-5 днів. Цей варіант є не дуже зручним для користувача, адже клієнт повинен був заповнювати достатньо об’ємні таблиці, в яких треба було вказувати детальну інформацію щодо винятків у його

харчуванні, після чого проводилася бесіда із спеціалістом, де обговорювався кожен пункт цієї таблиці.

2. У застосунку вже були заготовленні шаблони під певні групи клієнтів, що обмежувало кількість рецептів та не завжди відображало коректні варіанти меню. Також застосунки з таким типом меню не були розроблені для людей із певними обмеженнями у харчуванні.

1.3 Огляд загальних вимог для застосунку

На основі загальних проблем бізнесу, огляду застосунків конкурентів та реальних відгуків клієнтів, було створено основні вимоги:

1. Автоматичне створення меню для харчування з урахуванням таких параметрів:
    a. зріст, вага, вік та добова активність;
    b. добова норма калорій, білків (рослиного та тваринного походження), жирів (насичені та ненасичені), вуглеводів (прості та складні) та клітчатки;
    c. уподобання клієнта — їжа, яка йому до вподоби та їжа, яку він не хотів би включати до меню;
    d. медичні протипоказання при діабеті, алергічних реакціях та непереносимості лактози;
    e. індивідуальна система харчування для вегетаріанців;
    f. географічний стан клієнта;
    g. урахування сезону для актуального підбору продуктів.
2. Тип монетизації:
    a. підписка в застосунку — використовуючи сервіси Apple та Google. В цьому випадку клієнт буде обслуговуватися автоматично. Тобто автоматичне створення курсу та автоматичне його продовження. При цьому меню для користувача є безперервним;

   b. ручне оформлення — через відділ з продажу. В цьому випадку клієнт буде обслугований полуавтоматично. Тобто людина генерує меню та вручну його активує. По закінченню курсу, він буде видалений та замість нього буде створений спеціальний курс для гостей.

3. Працездатність алгоритму:
   a. швидке генерування меню;
   b. вміння генерувати для 1, 3, 7, 14, 21, 28, 30, 31, 32 та 37 днів.
4. Смачне та збалансоване меню. Меню яке зібрано за всіма правилам нутринології:
   a. сніданок, обід, вечеря — основні прийоми їжі;
   b. три динамічні перекуси, які можуть бути між основними прийомами їжі;
   c. обмеження по стравам із яєць (не більше однієї страви на день);
   d. одна порція червоного м’яса на тиждень;
   e. не більше 4 порцій картоплі на тиждень;
   f. дві порції риби на тиждень;
   g. правила солодкого сніданку (сніданок повинен мати в собі солодкі фрукти, а перекус бути калорійним) для стабілізації калорій на першу половину доби;
   h. основні страви не повинні повторюватися протягом одного дня (тобто сніданок, обід та вечеря повинні відрізнятися за складом один від одного);
   i. дотримуватися правильного розподілу продуктів харчування: перша страва повинна включати в себе продукти із високим вмістом білка. Друга страва включає в себе вуглеводний гарнір. Третя страва опціональна і залежить від дефіциту нутрієнтів зв день — додавання салатів для отримання рослинних білків або додавання овочів до гарніру при дефіциті клітчатки та вуглеводів;

j. уникати несумісних між собою страв (напр. молоко + рибні страви).

1.4 Огляд еволюційних та генетичних алгоритмів

Еволюційний алгоритм — це напрям в штучному інтелекті (розділ еволюційного моделювання), що використовує і моделює біологічну еволюцію [1].

Такий тип алгоритмів найчастіше використовують як пошукові алгоритми, які змінюють набір рішення завдяки перебору та модифікування результатів або як їх ще називають – трансформацією. Основную їх особливістю є те, що вони використовують біологічні терміни які пов’язані з теорії біологічної еволюції (схрещування, мутація та природній відбір) батьком якої був Чарльз Дарвін.

На сьогоднішній день існує три базові різновиди еволюційних алгоритмів:

1. Еволюційне програмування — такий метод звертаю більшу увагу на корекцію окремих індивідів та підборів найкращих генів, аніж на еволюцію всієї популяції. Цей підхід застосовує безстатеве схрещування та вищою вірогідності мутації. Таким чином, першу популяції потрібно запрограмувати таким чином, щоб на етапі першого відбору функція пристосованості обрала найкращі розв’язки.
2. Еволюційні стратегії — основна особливість такого підходу полягає в тому, що вся логіка кинута на реалізацію механізму мутації. Цей механізм не лише шукає потрібні розв’язки поставленої задачі, а й на еволюційний оператор мутації.
3. Генетичний алгоритм — основою цього підходу є використання оператору схрещування, як основний механізм для розв'язання поставленої задачі. Основана на гіпотезі: що вдале рішення може бути

у різних індивідах в однієї популяції. І завдяки цій гіпотезі це рішення може успадковуватися.

В основному ці всі існуючі базові різновиди еволюційного алгоритму дуже схожі між собою. Всі вони мають базові кроки реалізації: підбирається базова популяція, а далі застосовуємо функцію пристосованості для кожного індивіда.

1.4.1 Еволюційні алгоритми

Еволюційні алгоритми[14] — це сімейство алгоритмів стохастичного пошуку, які імітують природну еволюцію, запропоновану Чарльзом Дарвіном у 1858 році. Якщо ми розглядаємо інтелект як певну здатність особи адаптуватися до постійно мінливого середовища, ми можемо розглядати еволюційні алгоритми як підрозділ цієї теми. Ці алгоритми складаються з кількох ітерацій базового циклу еволюції [2].

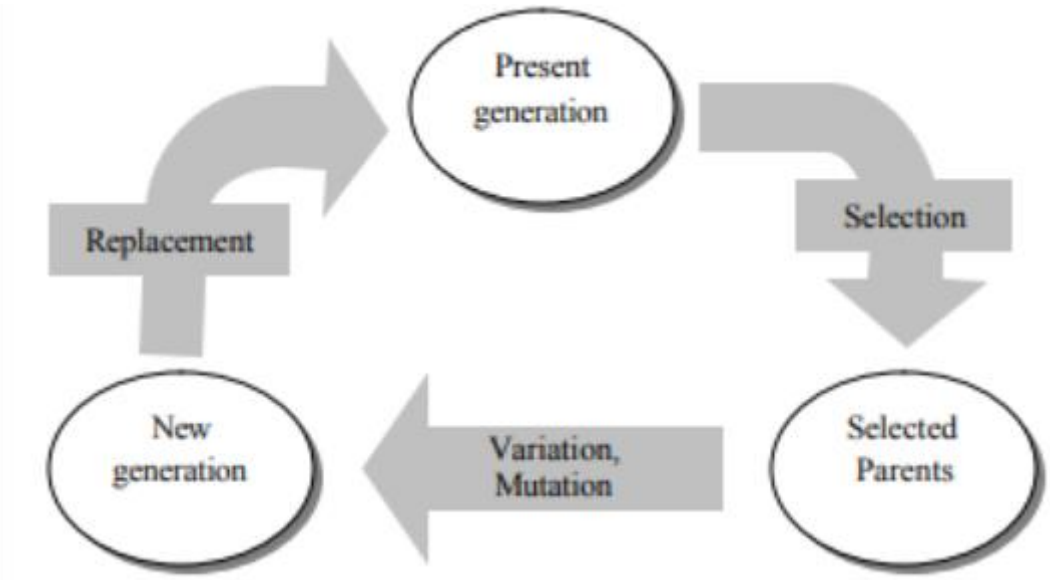


Рис. *1 Приклад ітерації базового циклу еволюції*

[13]Різні варіанти еволюційних обчислень включають один і той же базовий цикл з різними моделями подання або конкретними комбінаціями методів схрещування, мутації, та відбору. Цікавим моментом у реалізації є

баланс між двома протилежними операціями. З одного боку, операція «Відбір» призначена для зменшення різноманітності популяції (безліч можливих рішень), а з іншого боку, оператори «Схрещування» та «Мутація» намагаються збільшити різноманітність популяції. Цей факт призводить до швидкості збіжності та якості рішення. Як алгоритм оптимізації еволюційні алгоритми слід аналізувати, щоб знайти відповіді на такі питання, як:

1. Швидкість збіжності.
2. Якість розробленого рішення.
3. Обчислювальні вимоги.

До цих пір не було запропоновано загальної схеми рішення для аналізу загальної форми еволюційних алгоритмів, але деякі конкретні варіанти або реалізації можуть бути зосереджені на двох напрямках досліджень: теоретичному та емпіричному.

Теоретичний підхід намагається виявити математичні істини алгоритмів, які застосовуються у досить великій області додатків. Однак емпіричний підхід намагається оцінити продуктивність реалізації у конкретній галузі застосування. Обидва методи мають свої переваги та недоліки, і на практиці їх слід використовувати як додаткові засоби для розробки та налаштування конкретного екземпляра алгоритму.

1.4.2 Генетичний алгоритм

Генетичний алгоритм [3] — це стохастичні методи пошуку, які імітують метафору природної біологічної еволюції, моделюючи природні процеси, такі як відбір, схрещування, мутація, міграція, місцевість і сусідство. Їх успішно використовували в широкому спектрі застосувань для пошуку рішень для проблем жорсткої оптимізації, у тому числі в багатьох областях обчислювальної біології. Ця техніка корисна для пошуку оптимальних або близьких до оптимальних рішень для задач комбінаторної оптимізації, які традиційні методи не можуть ефективно вирішити.

Генетичний алгоритм — це метод, заснований на популяції, де кожен окремий представник популяції представляє варіант вирішення цільової проблеми. Ця популяція рішень розвивається протягом кількох поколінь, загалом починаючи з випадково згенерованого. У кожному поколінні еволюційного процесу всі особини в популяції оцінюються функцією пристосованості, яка вимірює, наскільки хорошим є рішення, представлене особиною, для цільової проблеми. Відібрані особини (зазвичай ті, що мають найвищу пристосованість) стають батьками і дають «потомство», тобто нових особин, які успадковують деякі особливості від своїх батьків, тоді як інші (з нижчою пристосованістю) відкидаються. Генерація нових нащадків від обраних батьків поточного покоління здійснюється за допомогою генетичних операторів. Цей процес ітеративно повторюється, доки не буде знайдено задовільне рішення або не буде досягнуто певного критерію зупинки, наприклад максимальної кількості поколінь.

1.5 Застосування та дослідження генетичного алгоритму

Одне з перших застосувань генетичного алгоритму для множинного вирівнювання послідовностей було реалізовано в вирівнювачі SAGA[4], незадовго до аналогічної роботи Чжана. У SAGA кожна особина в популяції є повним множинним вирівнюванням, і визначено оператори, які вставляють і зміщують прогалини у вирівнювання випадковим або напів випадковим чином. Під час етапу ініціалізації генерується популяція вирівнювання, яка є максимально різноманітною, або генерується випадковим чином, або з використанням, наприклад, динамічного програмування. Придатність популяції оцінюється шляхом оцінки кожного узгодження з заданою функцією пристосованості. Потім створюється нова популяція за допомогою операторів, таких як схрещування та мутація. Схрещення генерують дочірнє вирівнювання шляхом поєднання двох батьківських вирівнювань і є важливими для сприяння обміну

високоякісними генами. Потім дочірні елементи можна мутувати, наприклад, вставивши або видаливши ген. Лише найслабша половина популяції замінюється новим потомством, тоді як інша половина переноситься до наступного покоління. Процес завершується, коли досягається емпіричний критерій: після певної кількості поколінь або коли більше не спостерігається покращення.

Протягом багатьох років були представлені інші стратегії вирівнювання кількох послідовностей, засновані на генетичного алгоритмі. Вони ґрунтуються на принципі, подібному до SAGA, але реалізують кращі оператори мутації, які покращують ефективність і точність алгоритмів. Шю запропонував альтернативний підхід під назвою MSA-EC, де оптимізація консенсусної послідовності за допомогою генетичного алгоритму означає, що кількість ітерацій, необхідних для пошуку оптимального рішення, приблизно однакова незалежно від кількості послідовностей, які вирівнюються. MSA-GA — ще один простий метод на основі генетичного алгоритму, де початкова популяція генерується за допомогою попарного динамічного програмування. Інший алгоритм, GA-ant colony optimization (GA-ACO), поєднує оптимізацію колонії мурах з генетичним алгоритмом, щоб подолати проблему потрапляння в пастку локальних оптимумів. Для цього при кожній генерації GA вибирається найкраще вирівнювання та застосовується ACO. Техніка гумової стрічки-GA поєднує оптимізацію GA з RBT. RBT натхненний поведінкою еластичної гумової стрічки на пластині з кількома полюсами, що аналогічно місцям у вхідних послідовностях, які, швидше за все, пов’язані. RBT використовується для створення колонок вирівнювання з ідентичними або тісно пов’язаними залишками. У вертикальній декомпозиції за допомогою генетичного алгоритму при кожному поколінні GA послідовності поділяються вертикально на підпослідовності, які потім вирівнюються за допомогою методу прогресивного вирівнювання та комбінуються для побудови повного множинного вирівнювання.

Остання робота була зосереджена на покращенні точності генетичного алгоритму, зокрема, з використанням багатоцільових алгоритмів, таких як MO-SAStrE, який використовує вісім класичних інструментів MSA для отримання початкових вирівнювань, і три різні бали включені для оцінки кожного вирівнювання. Багатокритерійна процедура повертає підмножину недомінованих вирівнювання. Ці отримані вирівнювання однаково хороші, і неможливо вирішити, яке з них точніше відповідно до трьох цілей. Тому вибір найкращого вирівнювання залежить лише від мети, яку користувачі вважають більш корисною щодо конкретних вирівняних послідовностей. У MSAGMOGA[5] придатність індивідуума оцінюється на основі кількості збігів залишків, штрафу за афінний проміжок і оцінки «підтримки», яка вимірює кількість добре вирівняних послідовностей у рівнянні. Експерименти, проведені на п’яти наборах даних від BAliBASE, показали, що цей підхід дає високоякісні результати та має кращу ефективність, ніж інші протестовані методи генетичного алгоритму.

Багато з цих вирівнювачів на основі генетичного алгоритму продемонстрували потенційне підвищення точності вирівнювання в контрольних тестах, як правило, з використанням невеликих підмножин тесту BAliBASE. Хоча це чітко вказує на зацікавленість генетичного алгоритму у сфері MSA, це також ілюструє деякі їхні обмеження. Наприклад, отримані остаточні рішення можуть не відповідати оптимальному вирівнюванню, оскільки генетичний алгоритм може потрапити в пастку локальних рішень. Отримані результати також можуть бути суперечливими, навіть якщо повторити генетичний алгоритм з тими самими параметрами. Це може бути причиною занепокоєння при використанні результуючого множинного вирівнювання в низхідних системах висновку. [15]Генетичний алгоритм також має ще один недолік: тривалий обчислювальний час, необхідний для корисних результатів. Тим не менш, генетичний алгоритм є неявно паралельною технікою, тому її можна дуже ефективно реалізувати на потужних паралельних комп’ютерах

для вирішення великомасштабних проблем корисних результатів. Тим не менш, генетичний алгоритм є неявно паралельною технікою, тому її можна дуже ефективно реалізувати на потужних паралельних комп’ютерах для вирішення великомасштабних проблем.

1.6 Висновки з розділу 1

Отже, з огляду на все вищесказане, ми можемо підсумувати, що головною проблемою старого застосунку було використання великої кількості мануальної роботи та роботи спеціалістів у галузі нутриціології. Це викликало незручності як зі сторони компанії, так і зі сторони клієнта.

Одна із проблем для компанії полягала в тому, що з частою періодичністю потрібно було виділяти бюджет на нутриціолога, який розробляв індивідуальне меню для кожного клієнта. Із цього випливає наступна проблема — велика завантаженість нутріціологів і служби підтримки, що впливало на швидкість обслуговування клієнта. А як нам всім відомо — час — це гроші. Відповідно до цього, втрачала гроші, які могла виручити шляхом обслуговування більшої кількості клієнтів.

Для клієнта ж незручності проявлялися у великому періоді очікування, що могло призвести до втрати інтересу клієнта до застосунку і як результат — відмова від нього. На виході ми отримували витрачений бюджет на нутриціолога, витрачений час на написання меню, втрачений клієнт і тим самим мінус у бюджеті компанії.

Все це призвело до пошуку нових шляхів оптимізації застосунку. Провівши аналіз застосунків на платформах IOS та Android у категоріях “Фітнес”, “Спорт”, “Здорове харчування” тощо, ми виділили сильні та слабкі сторони застосунків конкурентів. Але в одному всі ці застосунки були схожі між собою — в жодному з них не було задіяно алгоритму, який спрощував би процес створення індивідуального меню.

Саме тому ми поставили для себе за мету створити застосунок, який би був не тільки конкурентоспроможним, але і перевершував інші застосунки в плані точності індивідуального меню та швидкості його створення. Тому ми з впевненістю можемо заявити, що даний предмет дослідження є інноваційним в своїй галузі.

Основними вимогами до застосунку були:

1. Автоматичне створення меню для харчування.
2. Підтримка двох типів монетизації (підписка та ручне оформлення).
3. Швидке генерування меню та вміння генерувати за певною кількістю днів.
4. здорове та збалансоване меню, яке відповідало би всім нормам нутриціології.

Як результат, за допомогою генетичного алгоритму, ми створили застосунок, який повністю відповідає вимогам замовника і зв своєю новизною не має гідних аналогів у подібних йому застосунках.

РОЗДІЛ 2 ДОСЛІДЖЕННЯ ЗАСОБУ ГЕНЕТИЧНОГО АЛГОРИТМУ ДЛЯ СИСТЕМИ ОБСЛУГОВУВАННЯ

2.1 Огляд генетичного алгоритму

Генетичний алгоритм[7] — відноситься до сімейства еволюційних алгоритмів пошуку. Головною ідеєю цього алгоритму стала теорія Чарльза Дарвіна — “еволюційна теорія”. Цей алгоритм симулює процес природного відбору, шляхом визначення більш сільніших індивідів із популяції які переживають більш слабких і створюють наступне покоління індивідів.

2.2 Природній відбір

Процес природного відбору починається з вибору сильних осіб з популяції. Їхнє потомство успадковує характеристики батьків і є частиною наступного покоління особин. Якщо обидва батьки сильні, то їхнє потомство буде сильнішим за батьків.

Це ітеративний процес і завершується, коли знайдено найсильніших особин. Ця ідея застосовується для пошуку. Розглядається набір рішень для проблеми та відбирається набір кращих із них.

2.3 Основні етапи генетичного алгоритму

Навчання генетичного алгоритму можно поділити на 5 етапів:

1. Створення базової популяції.
2. Функція знаходження найсильніших осіб в популяції.
3. Відбір найбільш сильніших індивідів із популяції.
4. Обмін особливостей між двома особинами.
5. Мутація окремих індивідів.

6. Нова ітерація з створенням нової популяції, але вже виходячи з попередньої популяції.

Процес роботи алгоритму починається із набору особин, що створюють — популяцію. Кожна особа — це поставлені умови, які комбінуються між собою. Особа характеризує набір параметрів (змін), які називаються — генами. Об’єднані гени формуються в одну сутність (хромосому) — рішення нашої поставленої задачі.

2.4 Принцип роботи генетичного алгоритму

Розглянемо класичний та відомий приклад задачи під назвою “OneMax” [8].

В цій задачи ми маємо список певної довжини що складається з нулів та одиниць.

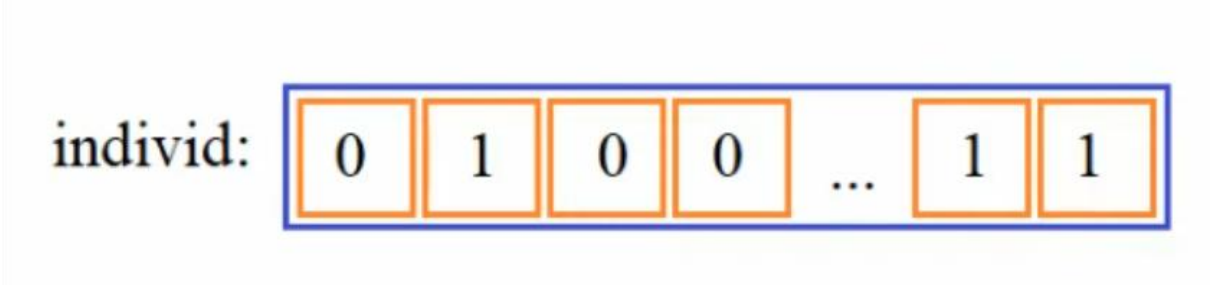


Рис. *1 Приклад списку для задачі “OneMax” певної довжини, що містить нулі та одиниці*

Ціль цієї задачі в тому щоб знайти рішення яка б давала максимальну суму чисел цього списку. Критерій якості буде нараховуватися по формулі — це сума чисел яка знаходиться в цьому списку.

$$fitness = \sum_{i=0}^{N-1} individ[i]$$

Рис. *2 Формула для підрахунку суми чисел*

Очевидно що, найкращим варіантом рішення цієї задачи буде список який складається з одиниць.

Рис. *3 Вигляд вирішеної задачи*

Це ми можемо легко вирішити і без реалізації генетичного алгоритму, но на цьому прикладі добре та легко показати принцип його роботи.

2.4.1 Формування першої популяції

Все починається з формування першої популяції. Тобто безліч особин одного виду.

Рис. *4 Вигляд першої популяції з N числом особин*

Особини в задачі “OneMax” складаються з нулів та одиниць і це типове представлення особин. В кожній задачи для генетичного алгоритму намагаємося формулювати так, щоб особини були представлені списком цілих або дійсних чисел. В цьому випадку ми отримуємо природну біологічну інтерпретацію цих списків як хромосом який містить в собі набір ген.

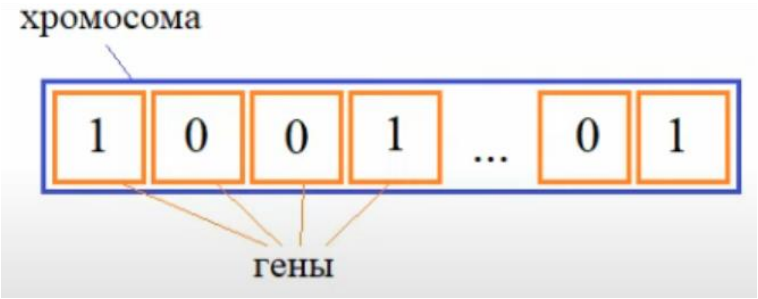


Рис. *5 Біологічна інтерпретація списка, як хромосома яка містить в собі набір ген*

Підбір початкових значень ген відбувається за допомогою датчика випадкових чисел. В результаті отримаємо множину самих різних особин, що дуже добре для процесу еволюційної адаптації всій популяції.

2.4.2 Функція пристосованості

В процесі еволюційного відбору в популяції, як правило, залишаються самі приспособлені особини. Ми повинні визначити цю степінь приспособленности, щоб імітувати процес еволюції.

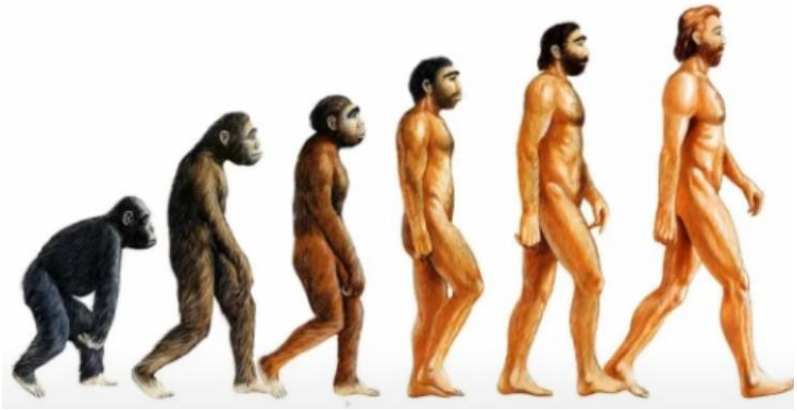


Рис. *6 Демонстрація еволюційного відбору*

Для задач генетичного алгоритму ступінь пристосованості — це отримання найкращого результату. Ті особи які представляють найкраще рішення поставленої задачі являються більш пристосовані, а щоб оцінювати якість рішення для кожного конкретного індивіда необхідно задати — функцію пристосованості (цільова функція). ЇЇ значення залежить від хромосоми конкретного індивіда. Список генів передається до функції пристосованості і на його основі обчислюються певні значення. Вважається що, чим воно більше тим більше пристосований індивід.

Наприклад, в нашій задачи “OneMax” чим більше одиниць буде в хромосомі індивіда тим він буде більш пристосований для рішення поставленої задачі.

Для різних задач використовується своя функція пристосованості і її конкретний математичний вигляд визначається розробником генетичного алгоритму виходячи з поставленого завдання оптимізації.

2.5 Імітація еволюційного процесу

Після того як в нас сформовано початкова популяція та визначена функція яка дає оцінку пристосованості кожній конкретній особини ми готові к імітації еволюційного процесу.

Будемо виробляти зміни в популяції від покоління до покоління в надії отримати все більш і більш пристосованих особин, а значить йти по шляху покращення рішення поставленого завдання.

Згідно теорії Чарльзу Дарвіна — рухаючої сили еволюції визначається наступними трьома процесами:

1. Відбір найбільш пристосованих особин.
2. Схрещування батьків для отримання нових індивідів.
3. Мутація — випадкова заміна окремих генів.

Рис. *7 Імітація еволюційного процесу*

Пояснення до рисунку 7:

1. У нас є початкова популяція.
2. Розраховуємо пристосованість для кожного індивіду.
3. По циклу йде оператори: відбору, схрещування та мутації.
4. Перевіряємо умову зупинки генетичного алгоритму.
5. І після того як виконали багато-багато разів пункт 4 ми в останньому поколінні обираємо найкращого індивіда — це і буде рішенням поставленого завдання за допомогою генетичного алгоритму.

2.5.1 Відбір

Відбір — це перша дія виконання генетичного алгоритму при імітації еволюційного процесу. Ціль цього оператору — з однієї сторони залишити в популяції найбільш пристосованих особин, но з другої сторони зберегти популяційну різноманітність.

Наприклад, якщо залишати найбільш пристосованих, то можно втратити важливі фрагменты хромосом у наименьш пристосованих особин. Можливість знаходження оптимального рішення досягається — схрещуванням одного найбільш пристосованого батька з іншим найменш пристосованим. Якщо на етапі відбору другий батько буде втраченим, то оптимальне рішення, можливо, буде досягатися за більше кроків роботи генетичного алгоритму, а може бути втраченим назавжди. Щоб зменшити вірогідність такого результату в наступному поколінні потрібно залишати не тільки самих кращих, но і давати можливість найменш пристосованим конкурентам залишатися в популяції.

Існує багато способів реалізації механізму відбору. В рамках задачи “OneMax” використаємо ідеєю турнірного відбору, який найбільш часто використовується на практиці.

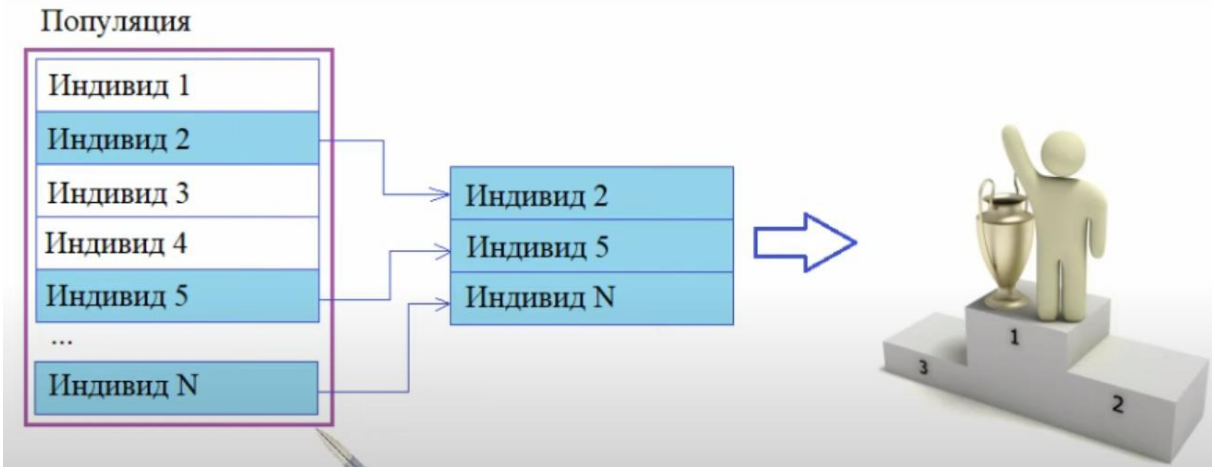


Рис. *8 Ідея турнірного відбору*

Ідея його дуже проста. Спочатку во всій популяції випадковим чином відбираються декілька індивідів. Потім серед них відбирається найкращий. Тобто найбільш пристосований. В результаті переможець проходить відбір і буде використаний як батько для операції схрещування.

У відповідність цей турнірний відбір повторюється з вихідною популяцією до тих пір, поки не буде сформовано новий список із потенційних батьків число яких буде співпадати з розміром початкової популяції.

2.5.2 Схрещування

Етап схрещування ще називають — кроссовером. На цьому етапі відбувається обмін даних між частинами хромосом батьків.

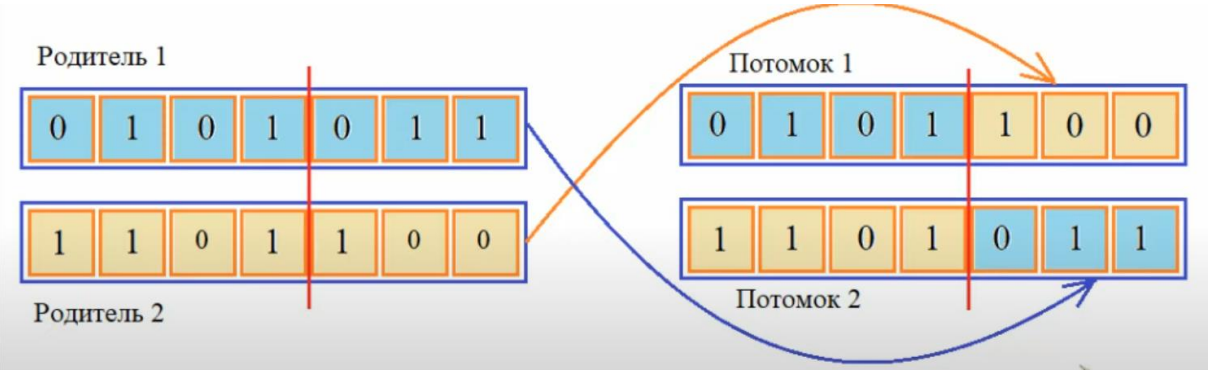


Рис. *9 Схрещування батьків та отримання нових потомків*

Зазвичай в популяції перебирають пари батьків і фрагменти їх хромосом перемішують — отримуючи новий набір генів в хромосомах потомків.

В даному випадку представлена схема одно точечного схрещування. Коли випадково обирається деяка точка розрізу хромосом у батьків, а далі здійснюється обмін їх частин. Так з’являються два потомки.

Ця операція виконується з деякою високою вірогідністю. При цьому батьки давши потомство, як правило, не проходять до наступного покоління, а батьки не давши потомство зберігаються, тобто дублюються в наступне покоління. Результаті цього розмір популяції після операції схрещування не змінюється.

2.5.3 Мутація

Останній оператор імітації процесу еволюції — це мутація. Вона застосовується до отриманої популяції і випадково, с малою вірогідністю, змінює значення окремих ген.

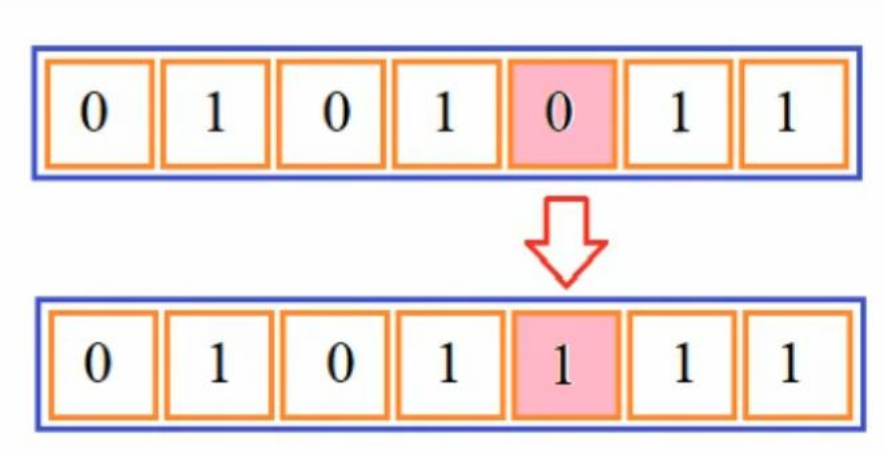


Рис. *10 Процес мутації*

В самому простому варіанті двійкового кодування мутація виконує конвертування бита. Цей механізм дозволяє розширювати область пошуку рішення задачи і зберегти різноманітність популяції. Можливо завдяки корисної мутації особина придбає нові властивості і стане більш конкурентно спроможною в своїй популяції. Надалі у неї є всі можливості дати потомство і закріпити корисний признак.

Саме так, завдяки мутації відбувається покращення результату рішення, авжеж може статися зворотній ефект — погіршення

пристосування індивіду. Тоді він не зможе конкурувати більш пристосованими особами і незабаром буде відсіяний в процесі відбору.

2.6 Висновок з розділу 2

1. Було розглянуто генетичний алгоритм та основні його етапи.
2. Розглянуто основну теорію Чарльза Дарвіна та як базується на цьому генетичний алгоритм.
3. Дослідження роботи генетичного алгоритму на прикладі легкої та популярної задачи “OneMax”.
4. Розглянуто основні принципи по імітації еволюційного процесу: відбір, схрещування та матація.
5. Розглянуто функцію пристосованості популяції.
6. Досліджено термінологічну базу та як вона взаємодіє в генетичного алгоритму.

РОЗДІЛ 3 РОЗРОБКА ВЕБ ЗАСТОСУНКА ГЕНЕТИЧНОГО АЛГОРИТМУ ДЛЯ СИСТЕМИ ОБСЛУГОВУВАННЯ

3.1 Зміст команди

Під час роботи над цим проектом, наша команда складалася з чотирьох людей:

1. Project manager — людина яка займалась постановкою задач та була прошарком між командою розробників та замовником.
2. Front-end розробник — людина яка займалась розробкою мобільного застосунку та панелі адміністрування всім проектом.
3. Back-end розробник — це я. Займався розробкою серверного слою та працював над самим генетичним алгоритмом.
4. QA інженер — людина, яка перевіряла весь проект та повідомляла про всі баги та проблеми які виникали під час роботи.

3.2 Вибір мови програмування

Моя спеціалізація — мови програмування PHP та Python. Основним питаннями стали швидкодія скрипта та його легкість написання.

PHP[8] — інтерпретована мова програмування. З версії PHP > 8 має тестовий інструмент, який дозволяє компілювати скрипт і тим самим пришвидшити вичислювальні здібності. Але великий мінус цієї мови для реалізації генетичного алгоритму не має проміжних бібліотек, які би дозволяли полегшити роботу з великими об’ємами даними.

Python3[9] — теж інтерпретована мова програмування, яка була написана мовою C (сі). Дозволяє свій код зробити компільованим кодом. Має у себе велику кількість бібліотек для роботи з великими об’ємами та, навіть, прискорює швидкість виконання скрипту.

Отже було вибрано мову Python, тому що вона містить весь комплект необхідних бібліотек і вона трохи швидше ніж нова версія PHP.

3.3 Використане програмне забезпечення та технології.

1. PHP — основна мова програмування для серверної частини. Використовувалася, на той час, версія 7.4.
2. Python — мова програмування для написання самого скрипта за генетичним алгоритмом. Версія мови 3.9.
3. Фреймворк Laravel[10] — основний фреймворк для серверної частини. Який підтримує мову PHP. Версія фреймворку 8. Завдяки цьому встановлений зв'язок між клієнтом і сервером. За логіку обробки всіх запитів відповідає Laravel.
4. PHPStorm — інтелектуальний редактор коду для PHP та підтримки фреймворку Laravel. Це той випадок коли IDE пише за тебе код сам та підказує твої помилки в code-style.
5. Visual Studio Code — редактор для Python.
6. MYSQL — реляційна база даних.
7. MongoDB — документоорінтована база даних.
8. Бібліотеки, які були необхідні для написання генетичного алгоритму:
    a. numpy[11] — бібліотека для мови програмування Python. Яка підтримує багатовимірні масиви, дозволяє працювати з векторами та містить в собі високорівневі математичні функції. З останніх версій дозволяє використовувати технологію векторизації процесору — це дозволить пришвидшити виконання відрахувань;
    b. pandas[12] — бібліотека для мови програмування Python. Яка дозволяє обробляти та аналізувати великий об’єм даних. Великий плюс цієї бібліотеки — вона підтримує інтеграцію з numpy.

9. MySQL Workbench — клієнтський застосунок, який надає зручний інтерфейс для роботи з реляційною базою даних. Дозволяє віддалено підключатися до баз даних.
10. Compass DB — теж платний клієнтський застосунок, який надає графічний інтерфейс для взаємодії з базою mongo.
11. Postman — це інструмент який дозволяє тестувати та налаштовувати автоматизовані тести для API.
12. Операційна система Ubuntu. Дозволяє мати повний контроль на файлами та налаштувати адміністрування над проектом.
13. Nginx — веб сервер, через який проходить всі запити від клієнта. Дозволяє обробити запити ще до попадання їх до моєї API.
14. Docker — сервіс контейнерів. Дозволяє всі сервіси (база даних, налаштування веб серверу nginx, основний фреймворк Laravel та Python скрипт) зробити незалежними між собою та використовувати однакове середовище для розробки, що для інших розробників, що на сервері.
15. Beanstalk — спрощена та легка система для черг. Яка дозволяє API виставляти черги та обробляти їх для генерування меню. Тому що потрібно деякий час коли скрипт із алгоритмом підбере актуальне меню для клієнта і це дає велику завантаженість для веб серверу. Завдяки цій системі черг веб сервер може виконувати генерації в фоновому режимі.
16. Використання єдиного код-стайлу для PHP — PSR-2.
17. Використання єдиного код-стайлу для Python — pep8.

3.4 Налаштування середовища для розробки

В першу чергу було налаштовано веб-сервер nginx та система контейнірів Docker.

Лістинг 1 демонструє налаштування веб серверу nginx. Основні параметри для налаштування — це сертифікат https, `client_max_body_size` — вказує на максимальну величину для тіла запиту, `fastcgi_read_timeout` — як довго клієнт чекатиме на відповідь від серверу, параметр `root` який показує точку для входу до застосунку Laravel, параметр `index` відповідає за файл, який знаходиться за маршрутом `root`.

Лістинг *1 Налаштування конфігу для nginx*

```
listen 80;
server_name _;
root /var/www/laravel-docker/public;
client_max_body_size 2000M;
fastcgi_read_timeout 2000;
index index.php;
error_log /var/log/nginx/error.log;

location / {
   try_files $uri /$uri /index.php?$query_string;
}
```

Лістинг 2 демонструє налаштування Docker. В ньому піднімалися контейнери для таких служб: nginx — налаштування для веб серверу який був описаний в Лістинг 1, fpm — налаштування для всіх залежностей та бібліотек PHP та Python, mysql — піднімає реляційну базу даних, надає для контейнеру fpm доступи та робить редирект на вільний порт, mongo — піднімає документоорієнтовану базу даних та робить редирект на вільний порт, beanstalkd — піднімає на своєму образі сервіс який буде адмініструвати черги для всього застосунку.

Лістинг *2 Налаштування для Docker (для локальної розробки)*

```
nginx:
    build:
        context: .
        dockerfile: docker/Nginx.Dockerfile
    ports:
        - 8098:80
fpm:
    build:
        context: .
        dockerfile: docker/Fpm.Dockerfile
mysql:
    image: mysql
    ports:
        - 33061:3306
    environment:
        - MYSQL_ROOT_PASSWORD=root
        - MYSQL_DATABASE=new_youslim_api
mongo:
    image: mongo
    environment:
        - MONGO_INITDB_ROOT_USERNAME=root
        - MONGO_INITDB_ROOT_PASSWORD=password
        - MONGO_INITDB_DATABASE=youslim
    ports:
        - 27018:27017
beanstalkd:
    image: schickling/beanstalkd
    ports: - 11301:11300
```

3.5 Налаштування баз даних

Для баз даних було виділено багато часу. Через те, що успіх виконання та досягнення поставлених умов залежить від правильності заповнення та розмітки страв. На розробку тільки однієї бази було витрачено тиждень, і доповнювалася вона протягом року розробки застосунку. Велику увагу було приділено правильності розмітки таблиць, указання типів даних та на зв’язок між таблицями. Була розроблена система тегів, в якій міститься близько 100 тегів для розмітки страв та їх віддача до генетичного алгоритму.

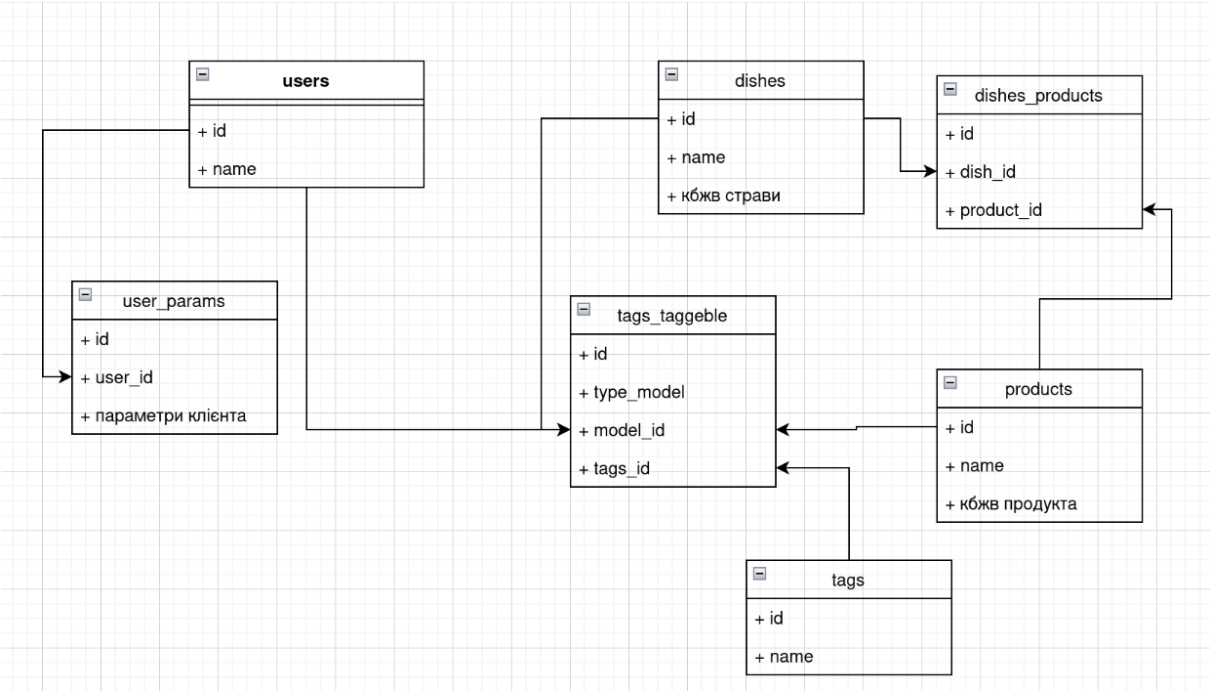


Рис. *11 Скорочена UML діаграма моделей бази даних MySQL*

3.5.1 Опис залежностей бази даних MySQL

Модель `users` — в ній зберігається вся контактна інформація клієнта та його налаштування для застосунку.

Модель `user_params` — містить всі параметри для клієнта. Має залежність один к багатьом з таблицею `users`. Це дає нам можливість

зберігати історію клієнта і послідовність змін протягом користання застосунком.

Модель `tags` — інформація про теги, якими ми можемо розмітити страви, клієнтів та продукти.

Модель `products` — містить всю інформацію про продукт: назву та його нутринологічні параметри (калорії, білки, жири та вуглеводи). Має залежність багато до багатьох від моделі `dishes` — це дає нам змогу використовувати різні продукти в різних стравах.

Модель `dishes` — інформація для страв та їх нутринологічні параметри, які розраховуються в залежності від змісту продуктів. Має проміжну таблицю `dishes_products` для зв'язку з таблицею `products`.

Модель `tags_taggeble` — містить багато до багатьох морфологічних зв'язків з таблицями: `users`, `tags`, `products` та `dishes`.

3.5.2 Опис документоорієнтованої бази даних MongoDB

Документоорієнтована база даних MongoDB зберігає в собі готовий результат згенерованого меню від генетичного алгоритму. Завдяки такому вибору можна гибко зберігати всю структуру генерації та завдяки драйверу текстового пошуку, який вже налаштований в цій базі, заощадити час для операцій CRUD.

Рис. *12 Структура документо-орієнтованої бази даних MongoDB*

Ця база містить в собі 497 полей, які використовуються для застосунку.

Найважливіші поля:

1. _id — містить унікальний захешированний ідентифікатор.
2. Об’єкт menus — в ньому знаходиться вся інформація про меню, яке було створено за допомогою генетичного алгоритму:
    a. Об'єкт menu містить всі страви для кожного прийому їжі;
    b. menu_structure — інформація про те, з яким типом правил було створене меню;
    c. result_nutrients_generate — нутринологічні результати меню.
3. user_id — унікальний ідентифікатор клієнта, для якого було зроблене це меню, та для пошуку.

4. target_value — параметри клієнта, для якого було створене меню (зріст, вік, вага, його цільова вага та нутриціологічна норма на день).

3.6 Розрахунок добової норми нутрієнтів для клієнта

Основними критеріями для розрахунку функції пристосованості виступають такі параметри, як: білки (тваринні та рослинні), жири (насичені та ненасичені), вуглеводи (прості та складні) та клітчатка. Для їх розрахунку використовується – формула Миффлина – Сан Жеора.

Формула Миффлина – Сан Жеора – це одна з самих популярніших формула розрахунку калорійності для оптимального схуднення або збереження ваги людини.

Для розрахунку добової калорійності для чоловіків.

для мужчин: (10 x вес (кг) + 6.25 x рост (см) – 5 x возраст (г) + 5) x A;

Рис. *13 Формула розрахунку добової норми калорій для чоловіків.*

Для розрахунку добової норми калорійності для жінок.

для женщин: (10 x вес (кг) + 6.25 x рост (см) – 5 x возраст (г) – 161) x A.

Рис. *14 Формула розрахунку добової норми калорій для жінок*

Де А — це коефіцієнт активності клієнта:

1. Мінімальна активність — без тренувань на тиждень і А = 1,2.
2. Слабка активність — одне тренування на тиждень і А = 1,375.
3. Середня активність — два тренування на тиждень і А = 1,55.
4. Висока активність — три тренування на тиждень і А = 1,725.
5. Ектримальна активність — більше трьох тренувань на тиждень і А = 1.9.

Після того як розрахували добову калорійність клієнта, розраховуємо інші нутрієнти – білки (рослинні та тваринні), жири (насичені та ненасичені) та вуглеводи (прості та складні):

1. Формула для розрахунку білку: вага клієнта * 1,5 * 4.
2. Формула для розрахунку тваринного білку: результат формули 1 * 0,6.
3. Формула для розрахунку рослинного білку: результат формули 1 * 0,4.
4. Для жирів діє константне значення — 600.
5. Для насичених жирів — 200.
6. Для ненасичених жирів — 400.
7. Формула розрахунку вуглеводів: різниця між добовою нормою калорійності клієнта та формулою 1 та відняти 600.
8. Формула для розрахунку простих вуглеводів: результат формули 7 * 0,15.
9. Формула для розрахунку складних вуглеводів: результат формули 7 * 0,85.

   Добова норма клітчатки для всіх — 25 грам на добу.

3.7 Підготовка даних для генерації

Через те, що генетичний алгоритм — це алгоритм пошуку і йому для повної реалізації потрібні підготовлені актуальні дані, цим процесом займається відповідний сервіс фреймворк Laravel. Під час запиту від клієнта до веб серверу йде сигнал до команди генерування файлів. Параметром запиту є ідентифікатор користувача. Це дозволяє нам під час генерування файлів відтягувати свіжі та актуальні дані користувача.

Для скрипту генетичного алгоритму потрібно 6 файлів, у середньому їх вага 100 МБ:

1. dishes.json — файл, який містить інформацію про всі страви з повною інформацією, яка відповідає характеристикам клієнта. Враховуються

вподобання клієнта за допомогою системи тегів, медичні обмеження, сезон, його місцезнаходження та бюджет.

2. target_value.json — в цьому файлі зберігаються актуальні нутринологічні дані на добу. Щоб під час генерування скрипт міг налаштуватися під цю метрику.
3. weight.json — в цьому файлі зберігається перша система штрафів. Під час знаходження найкращих індивідів розраховується різниця квадратів всіх нутринологічних параметрів.
4. dishes_for_each_position.json — файл, який розбитий по кожному прийому їжі і в кожному прийомі знаходяться відповідні страви.
5. menu_structure.json — файл, в якому знаходиться структура, по якій генетичний алгоритм повинен зібрати меню.
6. penalty_table.csv — файл з таблицею (матриця), в якій міститься інформація сумісності кожної страви. Наприклад, якщо в базі міститься 500 страв, то таблиця буде 500 * 500. Цей файл згенеровано завчасно, тому що він потребує великих ресурсів для розрахунків.

3.7.1 Генерування та збір файлу penalty_table.csv

Ця система штрафів (друга система штрафів) дозволяє вирішити проблему несумісних та повторюваних страв. Всі таблиці знаходяться на Google Sheets. І кожного дня відбувається нова генерація таблиць, через те, що кожен день відбуваються операції над стравами: додавання нових, видалення неактуальних та оновлення страв. Генерація таблиці штрафів відбувається в 3 етапи:

1. Таблиця 1. Генерація таблиці штрафів за тегами. Вона містить в собі матрицю 30 * 30 тегів (тегів для продуктів) які можуть між собою конфліктувати і потрібно ці конфлікти вирішувати. Таблиця заповнювалася вручну кваліфікованими нутринологами.

Таблиця 1

Приклад таблиці штрафів по тегам

	1/бобовое	2/грибы	3/крупа	9/рыба	10/сыр	25/картофель
1/бобовое	1	0	1	0	0	1
2/грибы	0	1	0	0	0	0
3/крупа	1	0	1	0	0	1
9/рыба	0	0	0	1	0	0
10/сыр	0	0	0	0	1	0
25/картофель	1	0	1	0	0	1

2. Таблиця 2. Із таблиці штрафів тегів вже автоматично генерується таблиця штрафів продуктів. В цій матриці йде розрахунок всіх продуктів, які є в базі даних. Продукти розмічені нутринологами. Під час генерування виставляються коєфіцієнти штрафів від 0 (продукти сумісні між собою) до 1 (продукти не сумісні між собою). Ці коефіцієнти слугують як множники, які далі будуть надані для останньої таблиці.

Таблиця 2

Приклад таблиці штрафів по продуктам

	6/Баранина	8/Утиное Мясо	9/Бедра куриные (филе)	11/Куриное Крыло	12/Мясо Кролика	13/Куриные Голени
6/Баранина	1	1	1	1	1	1
8/Утиное Мясо	1	1	1	1	1	1
9/Бедра куриные (филе)	1	1	1	1	1	1
11/Курино	1	1	1	1	1	1

е Крыло						
12/Мясо Кролика	1	1	1	1	1	1
13/Куриные Голени	1	1	1	1	1	1
14/Перепелиное Мясо	1	1	1	1	1	1

3. Таблиця 3. Остання таблиця на цьому етапі, яка йде до скрипту генетичного алгоритму — таблиця штрафів страв. Для кожної пари страв береться список всіх продуктів та вираховується середнє значення — це і буде коефіцієнтом штрафу пар страв. Якщо бодай одна пара продуктів між собою має коефіцієнт штрафу 1 (повністю несумісні) то всій парі страв ставиться 1.

Таблиця 3

Приклад таблиці штрафів по стравам

	32/Творожно—фруктовое суфле с семенами чиа	34/Тушеная куриная печень	37/Вареные куриные голени	39/Жареная куриная печень	44/Баранья отбивная	47/Тушеная баранина
32/Творожно-фруктовое суфле с семенами чиа	1	0	0	0	0	0
34/Тушеная куриная печень	0	1	1	1	1	1
37/Вареные куриные голени	0	1	1	1	1	1
39/Жареная	0	1	1	1	1	1

куриная печень						
44/Баранья отбивная	0	1	1	1	1	1
47/Тушеная баранина	0	1	1	1	1	1

3.7.2 Приклад розрахунків пар страв

На прикладі двох страв “Омлет” та “Відварна сочевиця”.

Склад продуктів “Омлету”:

- куряче яйце;
- молоко.

Склад продуктів “Відварної сочевиці”:

- сочевиця.

Формула розрахунку коефіцієнта штрафу:

$$\frac{(\textit{куряче яйце} \times \textit{сочевиця}) + (\textit{молоко} \times \textit{сочевиця})}{2} = \textit{коефіцієнт штрафу}$$

3.7.3 Генерування файлу dishes_for_each_position.json

Основним критерієм правильного підбору меню є врахування добових та тижневих правил. Правила були прописані кваліфікованими нутринологами. Генеруючи цей файл потрібно враховувати всі правила. До скрипту передається інформація страв на тиждень. Генерація для курсу харчування відбувається подобово.

Правила які застосовуються для генерації:

1. 2 рибні страви на тиждень.
2. 1 страва з червоним м’ясом на тиждень.

3. Не більше 3 солодких сніданків на тиждень.
4. Не більше однієї яєчної страви на добу.
5. Врахування перекусів (опція на вибір).

3.8 Реалізація генетичного алгоритму

Після ознайомлення з самим принципом генетичного алгоритму я приступив до його реалізації мовою Python. Але перед початком його написання, було витрачено час на вивчення документації бібліотек numpy та pandas.

За допомогою pandas я можу проаналізувати весь об’єм отриманих файлів для генерації: всі відповідні страви з повною нутринологічною інформацією та їх тегами, цільові значення клієнта, ваги для підрахунків результатів кожної популяції, за допомогою суми зважених квадратів різниць, страви для кожного прийому їжі та матриці штрафів страв. Після читання та аналізування файлів, всі їх форматуємо до формату бібліотеки numpy за допомогою метода `to_numpy()`.

Лістинг *3 Читання та аналіз файлів за допомогою бібліотеки numpy*

```
penalty_table = pd.read_json(penaltyTable_path)
penalty_table = np.asarray(penalty_table).ravel()
dishes = pd.read_json(dishes_path)
targetValue = pd.read_json(targetValue_path,
orient="index")
weights_json = pd.read_json(weights_path, orient="index")
dishes = dishes.drop(["portion"], axis=1)
dishes_for_each_position =
pd.read_json(dishes_for_each_position_path)
menuStructure = pd.read_json(menuStructure_path)
dishes_for_each_position.fillna(0, inplace=True)
```

```
number_of_dished_per_day =
dishes_for_each_position.shape[0]
dishes_for_each_position =
dishes_for_each_position.to_numpy()
```

Далі надаємо базові налаштування обмежень щодо роботи алгоритму:

1. Кількість хромосом в одній популяції — 200.
2. Кількість популяцій — 100.
3. Відсоток випадковості до схрещування хромосом — 75%.
4. Відсоток випадкості до мутації хромосом — 25%.

Лістинг *4 Задання обмежень налаштування щодо виконання генетичного алгоритму*

```
population_size = 200
epochs = 100
mutation_coef = 25
crossing_coef = 100 - mutation_coef
```

Першим кроком для початку роботи над алгоритмом було написання функції, яка генерує випадкову першу популяцію індивідів. Принцип написання був легкий. Беремо масив всіх існуючих страв та генеруємо популяцію в кількості 200 хромосом і додаємо до кожної хромосоми по необхідній кількості ген (кількість ген вказано в змінній menu_sctructure).

Вхідні параметри функції:

1. dishes_for_each_position — массив всіх існуючих страв.
2. population_size — розмір популяції.

Повертає функція — numpy массив для першої популяції випадкових ідентифікаторів страв.

Лістинг *5 Функція для генерації першої популяції*

```
def generate_population(dishes_for_each_position,
population_size):
```

```
    population = []
    for position in dishes_for_each_position:
        population.append(np.random.choice(position,
population_size))
    return np.array(population).transpose()
```

```
array([[2154, 2223, 1625,  892,  411,  817, 1303,  625, 1031,  800],
       [2247, 2220, 1627,  401, 1113,  428,  815,  624, 1031,  804],
       [ 515, 2221, 1626,  875,  411,  817,  624,  906, 2261,  803],
       [2244, 2221, 1624,  878, 2112,  816,  791,  906, 1375,  806],
       [ 514, 2221, 1625, 1224, 1103, 1375,  788,  905, 1317,  801],
       [ 124, 2220, 1626,  872,  416, 1592, 1302,  905, 1516,  792],
       [2153, 2220, 1625,  401, 1104, 1037, 1303,  905, 1449,  800],
```

Рис. *15 Приклад першої випадкової популяції*

Лістинг 6 демонструє одну із функцій особливості генетичного алгоритму — схрещення. Принцип його роботи: отримуємо пару батьків та ділимо кількість хромосом іншого батька навпіл — результатом буде нова хромосома, де початок буде від першого батька, а кінець від другого. Основна його модифікація полягає в тому, що — він схрещує цілу популяцію.

Лістинг *6 Функції схрещування*

```
def crossover(population_size, sorted_population,
number_of_dished_per_day):
    offspring = []
    cp = int(round(number_of_dished_per_day/2))
    for i in range(0,population_size,2):

offspring.append(np.concatenate((sorted_population[i][0:cp
], sorted_population[i+1][cp:])))
offspring.append(np.concatenate((sorted_population[i+1][0:
cp], sorted_population[i][cp:])))
    return offspring
```

Лістинг 7 демонструє другу функцію особливості генетичного алгоритму — мутацію. Якщо скрипт потрапив у випадковість в 25% — то хромосома буде мутована. Таким чином в цієї хромосоми буде повністю замінений один випадковий ген. Це може як зробити результат хромосоми кращим, так і погіршити цей результат.

Для кожної хромосоми в популяції генерується випадковість до мутації. Якщо випадковість потрапляє в 25%, то замінюємо ген.

Лістинг *7 Функція для мутації популяції*

```
def mutation(population, number_of_dished_per_day, 
dishes_for_each_position):
   mutation_probabilities = random.randint(0, 100, 
size=len(population))
   if_mutate = mutation_probabilities — threshhold
   indexes_to_mutate = np.argwhere(if_mutate > 0)
   for j in indexes_to_mutate.flatten():
       position_to_replace = random.randint(0, 
number_of_dished_per_day)
       replacement = 
np.random.choice(dishes_for_each_position[position_to_repl
ace])
       population[j][position_to_replace] = replacement

   return population
```

Останнім кроком для вибору найкращої популяції є розрахунок фітнес функції. За допомогою цієї метрики ми можемо штрафувати хромосоми та вибирати найкращі індивіди. Формулою для штрафу є сума зважених квадратів різниць. Основним доповненням є обчислення для всієї популяції, що буде продемонстровано в Лістингу 8.

Лістинг *8 Функція для розрахунку фітнес-функції*

```
def ff(goal, population, dishes_data,
number_of_dished_per_day, weights=[1, 1, 1, 1, 1]):
   nutrition_data_by_individual_flattered =
np.take(dishes_data, population.ravel(),axis=0)
   n = dishes_data.shape[1]
   ndif = nutrition_data_by_individual_flattered
   newshape =
(int(ndif.shape[0]/number_of_dished_per_day),number_of_dis
hed_per_day, n)
   results = np.reshape(ndif, newshape).sum(1)

   return np.sum(weights*(100*(goal - results)/goal)**2,
axis=1)
```

Результатом, який повертає фітнес функція буде масив розрахованих штрафів популяції для кожної хромосоми.

```
array([ 46606.67251819, 171664.39850179,  31421.25550868,  28022.58795124,
        19058.2294415 ,  29664.27433093,  29589.68041852,  45516.1116238 ,
        23685.96400685,  34619.24745644, 161267.84890379,  19175.01535982,
       167772.96817265,  18746.49491195,  22703.96597474,  26873.65800557,
        35645.56060442,  36811.22384827,  44503.95351359,  41951.82945735,
        28644.00317997,  41330.07270396,  32528.72150329,  48189.69462832,
        27864.58780118, 183243.4999124 , 164583.57827623, 169907.83521272,
        37726.76640084,  16870.15980544,  52094.41794482,  20832.76168916,
        66283.82285051,  32020.29913841,  32785.95527171,  26389.40249982,
       163899.88030089, 165764.1737935 ,  42854.77624952, 159900.60221329,
```

Рис. *16 Приклад розрахунку фітнес-функції*

Поєднуємо всі попередні кроки. Створюємо функцію, яка буде розраховувати та знаходити найкращу хромосому, це і буде нашим

результатом генерації. Принципом всієї роботи скрипту є генерування першої популяції, ітерація по кількості заданих популяцій в налаштуваннях скрипту, виконання схрещування та мутацій хромосом в кожній популяції, розрахунок фітнес функції для всіх хромосом в популяції та вибір найкращої популяції.

Лістинг *9 Функція виконання всього генетичного алгоритму*

```
def GA_itterator(population_size, epochs, mutation_coef,
threshhold, weights):
   stats = []
   pop = generate_population(dishes_for_each_position,
population_size)

   for i in range(epochs):
       fitness = ff(goal, pop, dishes_data,
number_of_dished_per_day, weights)
       sorted_population = pop[fitness.argsort()]
       offspring = crossover(population_size,
sorted_population, number_of_dished_per_day)
       pop = np.concatenate((pop,offspring))
       fitness = ff(goal, pop, dishes_data,
number_of_dished_per_day, weights)
       mutation_probabilities = random.randint(0, 100,
size=len(pop))
       if_mutate = mutation_probabilities — threshhold
       indexes_to_mutate = np.argwhere(if_mutate > 0)

       for i in indexes_to_mutate.flatten():
           position_to_replace = random.randint(0,
number_of_dished_per_day)
           replacement =
np.random.choice(dishes_for_each_position[position_to_repl
ace])
           pop[i][position_to_replace] = replacement


       best_idx = fitness.argsort()[: population_size]
       fitness = fitness[best_idx]
       stats.append(fitness[0])
```

```
        best_individual = pop[best_idx[0]]
        pop = pop[best_idx]

    return best_individual, fitness, pop, stats
```

```
best_fitness 1366.0
[1415, 1119, 841, 1146, 1024, 49, 458, 1450, 39, 568]
       id                                          name  calories
0    3328                                       Творог     185.0
1    2412  Оладушки из батата с соусом из йогурта и тахини     354.0
2    2182                                 Овощи для вок      40.0
3    2595                                       Фрукты      72.0
4    2337           Салат с овощами гриль и сыром тофу     129.0
5     418                             Отварные макароны     174.0
6    1519                                Орехи на выбор     193.0
7    3367                              Салат с омлетом      231.0
8     408                               Вареный кускус     301.0
9    1809                            Йогурт натуральный      95.0
10  Result                                         NaN    1774.0
11     NaN                                         NaN    1770.0
```

Рис. *17 Результат виконання скрипту*

Як ми бачимо за результатом виконання скрипту для клієнта з його параметрами:

1. Найкращий результат фітнес-функції — 1366. Методом дослідження було доведено, що найкращим варіантом є значення, яке приблизно дорівнює 1500.
2. Отриманий список страв на 3 основні прийому їжі та на 3 сніданки. Тобто за структурою було підібрано 10 страв.
3. Рядок 10 та 11 демонструє добову норму калорійності клієнта та добову норму згенерованого меню і вони майже є однаковими.

3.9 Логіка та запуск всієї екосистеми

На рисунку 18 проілюстрована UML діаграма запиту до генерації курсу харчування.

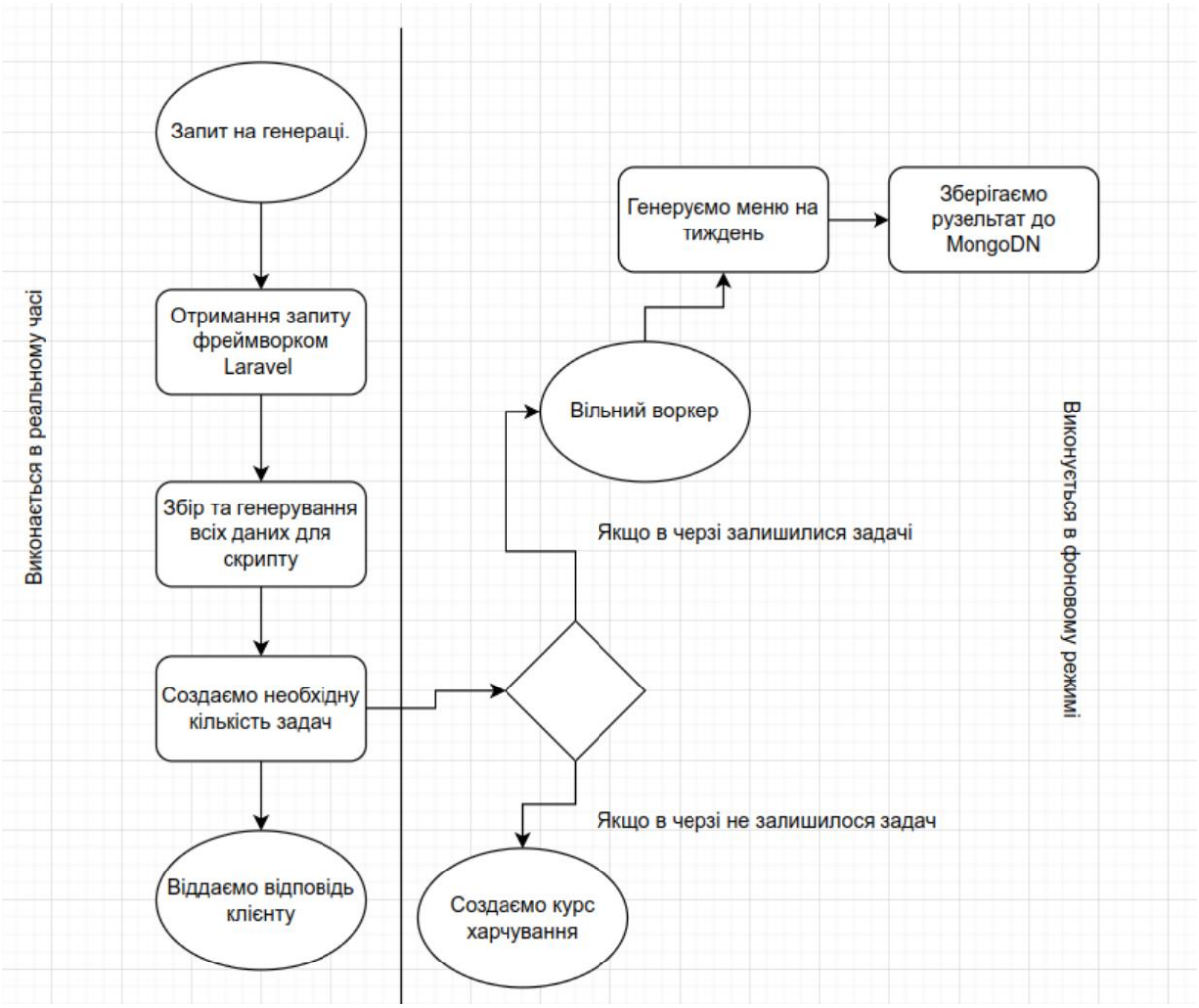


Рис. *18 UML діаграма принципу роботи всієї екосистеми*

3.9.1 Виконання в реальному часі

Початковою точкою відправлення слугує запит від клієнта. POST запит виконується з параметром user_id (унікальний ідентифікатор користувача, для якого потрібно зібрати курс харчування). За допомогою цього, під час підготовки до генерації, ми можемо дізнатися всю необхідну інформацію про клієнта.

Далі цей запит передається веб-сервером до фреймворку Laravel. Фреймворк дістає за допомогою user_id всі необхідні моделі (таблиці з

MySql) та передає їх до генерування допоміжних файлів для генетичного алгоритму.

На основі отриманих моделей ми генеруємо всі необхідні та актуальні файли для скрипту генератора та створюємо чергу задач з параметрами генерації.

Відправляємо відповідь клієнту, де сповіщаємо, що дані прийняті і потрібно зачекати поки згенерується меню.

3.9.2 Виконання в фоновому режимі

Вільний воркер перевіряє базу черг і якщо в базі з’явилася задача, він одразу береться за її виконання.

Воркер бере всі файли та дає команду скрипту “Старт”.

Лістинг *10 Запуск скрипту генетичного алгоритму воркером.*

```
$dishes =
storage_path("app/generator/$userId/dishes.json");
       $targetValue =
storage_path("app/generator/$userId/targetValue.json");
       $weights =
storage_path("app/generator/$userId/weights.json");
       $dishesForEachPosition =
storage_path("app/generator/$userId/dishesForEachPosition.
json");
       $menuStructure =
storage_path("app/generator/$userId/menuStructure.json");
       $penaltyTable = $this->penaltyFileName;
       $python = env('PYTHON', 'python3');
       $scriptPath = public_path('generator-final-new-
data.py');
       $output = null;
       exec(
```

```
            "$python $scriptPath " .
            "$dishes $targetValue $weights
$dishesForEachPosition $menuStructure $penaltyTable",
            $output
        );
return json_decode($output[0], true);
```

Коли воркер закінчив свою роботу, все згенероване меню поміщається до документоорієнтованої бази даних MongoDB.

Фінальний крок — після генерації всіх тижнів меню, ми об'єднуємо їх до однієї структури та віддаємо фінальному користувачу.

3.10 Висновки з розділу 3

1. Була обрана мова програмування для реалізації генетичного алгоритму — Python.
2. Розглянутий список використаного програмного забезпечення та використаних бібліотек для поставленої задачі.
3. Налаштоване середовище для правильного та безперервного роботи всього застосунку. Налаштування локального середовища для розробки.
4. Розглянуто розрахунок всіх нутрієнтів. Використання формули Миффлина – Сан Жеора для визначення добової норми калорій клієнта. Та розрахунок вуглеводів, білків та жирів.
5. Було розглянуто та побудовано реляційна база даних MySQL та документоорієнтована база даних MongoDB.
6. Описаний весь процесс розробки генетичного алгоритму. Всі його налаштування та працездатність.
7. Налаштована система воркерів, яка обробляє скрипт генетичного алгоритму.

РОЗДІЛ 4 ДОСЛІДЖЕННЯ РЕЗУЛЬТАТІВ РОБОТИ ЗАСТОСУНКА З ГЕНЕТИЧНИМ АЛГОРИТМОМ ДЛЯ СИСТЕМИ ОБСЛУГОВУВАННЯ

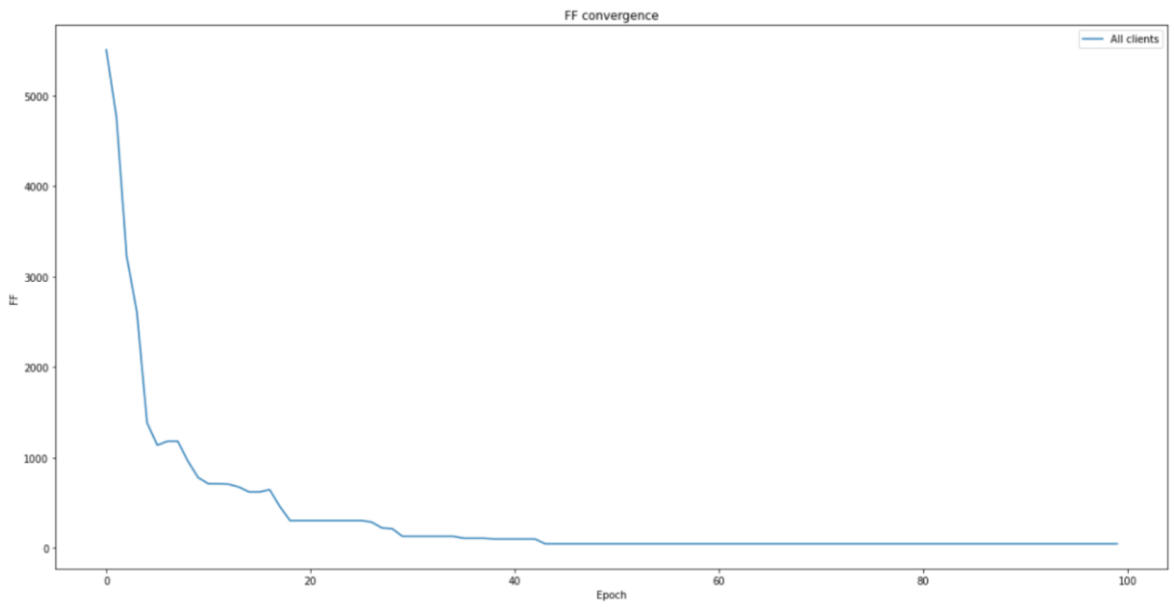
4.1 Результати налаштування конфігурацій генетичного алгоритму для системи обслуговування людей

В цьому розділі буде розглянуто дослідження налаштування конфігурацій для скрипту генетичного алгоритму та демонстрація змін виконання в часі і зміна при цьому результату.

Основні параметри для конфігурації:

1. population_size — відповідає за розмір популяції.
2. epoch — відповідає за кількість популяцій.

Початковими параметрами було взято такі показники population_size = 200 та epoch = 100.



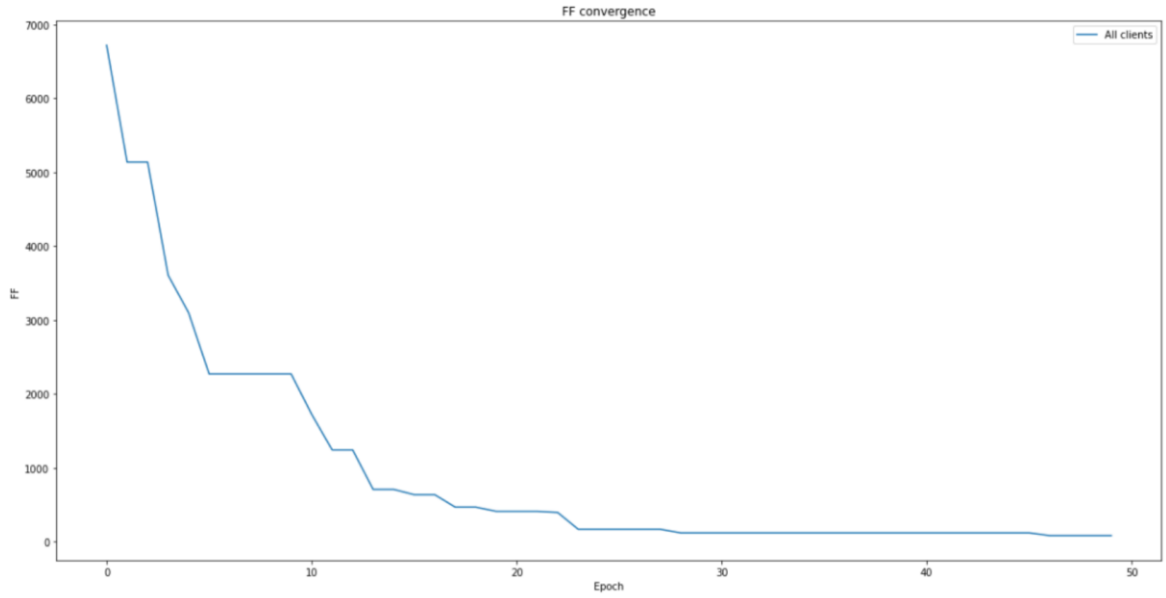

```
Wall time: 4512 ms
```

Рис. *19 Демонстрація виконання скрипту з початковими параметрами запуску*

Як показує початкова конфігурація вона має декілька проблем, а саме:

1. Неприємний час виконання скрипту Wall time: 4.5 секунди.
2. Знаходження найкращої хромосоми вже в 43 популяції, а з цього можна зробити висновки, що 67 популяцій скрипт працював марно і віддавав одну і ту хромосому.

Після отримання цих даних, я ще аналізував та тестував конфігурацію. Під час цього я помітив, що знаходження найкращої хромосоми відбувається на проміжку 30 — 44 ітерацій популяцій. І таким чином я зробив калібровку конфігурацій і самим оптимальним рішенням виявилася комбінація з: population_size = 100 та epoch = 50.




```
Wall time: 2345 ms
```

Рис. *20 Результат фінальної калібровки параметрів налаштування для генетичного алгоритму*

З цього можна зробити висновки, що з цим налаштуванням зменшився час виконання з 4.5 секунд до 2 — 2.3 секунд та на якість генерації це ніяк не вплинуло.

4.2 Дослідження налаштування корректного меню для реальних користувачів застосунку

Основною задачею виконання скрипту генетичного алгоритму — підібрати правильне, їстівне, враховуючи всі побажання користувача та дотримання нутринологічних добових норм.

```
best_fitness 133897.0
[378, 475, 843, 1146, 1451, 51, 458,
        id                        name
0     1335              Яйцо всмятку
1     1604    Цельнозерновые хлебцы
2     2184             Овощи для вок
3     2595                    Фрукты
4     3368           Салат с омлетом
5      420         Отварные макароны
6     1519            Орехи на выбор
7     3366           Салат с омлетом
8      384  Вареная пшеничная крупа
9     1812        Йогурт натуральный
```

Рис. *21 Приклад генерації*

Проаналізувавши рисунок 9 можно зробити такі висновки:

1. Дуже великий показник фітнес функції — 134 000. Це нам говорить про те що меню подібране по нутринологічним нормам не підходить добовій нормі клієнта (калорії, білки, вуглеводи, жири, клітчатки та похідні від них).
2. Не їстівне меню. Якщо його розділити на 3 основні прийоми їжі: сніданок — 0 1 2, обід — 4 5 та вечеря — 7 8. То маємо великі проблеми:
    a. страви повторяются;
    b. великий недобір по нутрієнтам;
    c. зовсім не їстівне меню;

d. не дотримуються правила поставлені нутринологами (не більше одного яйця на добу; немає жодної м’ясної страви).

4.2.1 Налаштування системи штрафів для корекції фітнес-функції

Для вирахування фітнес-функції використовується сума зважених квадратів різниць. Принцип роботи формули заключається в тому, щоб знайти відсоток відхилення від поставленої цілі та отриманого результату в квадраті та помножити їх на вагові коефіцієнти. Враховується всі нутринологічні параметри.

```
"calories": 1,
"proteins": 1,
"vegetable_protein": 1,
"animal_protein": 1,
"fats": 1,
"saturated_fat": 1,
"unsaturated_fats": 1,
"carbohydrates": 1,
"simple_carbohydrates": 1,
"complex_carbohydrates": 1,
"cellulose": 1
```

Рис. *22 Базові вагові параметри штрафів для фітнес-функції*

Пояснення до рисунку:

1. calories — калорійність;
2. proteins — білки;
3. vegetable_protein — рослинний білок;
4. animal_protein — тваринний білок;

5. fats — жири;
6. saturated_fat — насичені жири;
7. unsaturated_fat — ненасичені жири;
8. carbohydrates — вуглеводи;
9. simple_carbohydrates — прості виглеводи;
10. complex_carbohydrates — складні вуглеводи;
11. cellulose — клітчатка.

Основною мірою нутрієнтів виступають — калорії. Для більш точних розрахунків фітнес функції грами переводились до калорій. Формула конвертації білків (і також рослинного та тваринного походження) та вуглеводів (простих та складних) 1 грамм * 4 калорії, для жирів (насыщенных та ненасичених) 1 грамм * 9 калорій, а клітчатка залишається в грам так як її неможливо перевести до калорій.

Шляхом багатьох повторів та досліджень було виявлено така комбінація штрафів.

```
"calories": 0.2,
"proteins": 0.5,
"vegetable_protein": 0.5,
"animal_protein": 0.5,
"fats": 0.1,
"saturated_fat": 0.1,
"unsaturated_fats": 0.1,
"carbohydrates": 0.3,
"simple_carbohydrates": 0.3,
"complex_carbohydrates": 0.3,
"cellulose": 5
```

Рис. *23 Оптимальна комбінація штрафів для розрахунку фітнес-функція генетичного алгоритму*

```
best_fitness 1052.0
[382, 31, 841, 1147, 1450, 22, 457, 1450, 39,
```

	id	name	calories
0	1339	Яйцо пашот	130.0
1	400	Вареная овсяная крупа	273.0
2	2182	Овощи для вок	40.0
3	2596	Фрукты	72.0
4	3367	Салат с омлетом	231.0
5	391	Вареное пшено	215.0
6	1518	Орехи на выбор	128.0
7	3367	Салат с омлетом	231.0
8	408	Вареный кускус	301.0
9	1811	Йогурт натуральный	158.0
10	Result	NaN	1779.0
11	NaN	NaN	1770.0

Рис. *24 Результат після калібровки штрафів фітнес-функції*

Після калібровки бачимо результат найкращої хромосоми в якої фітнес-функція = 1052 та маємо мінімальне відхилення по калоріям на добу: добова калорійність клієнта 1770, а результатом згенерованого 1779 — відхилення складає всього 9 калорій.

Шляхом дослідженням було виявлено діапазон значень фітнес-функції які задовольняють від 0 до 1500.

4.2.2 Вирішення проблеми “їстівного меню”

Як бачимо на рисунку 9, то результату ще далеко до ідеалу. Для цього командую кваліфікованих фахівців нутриціологів було зроблено декілька варіантів моделей меню. Ця модель служить орієнтиром для генетического алгоритму і вона по ньому збирає меню. В цій моделі вказується: прийом їжі, кількість страв на цей прийом, та теги для кожної їх страв.

Лістинг *11 Один з прикладів моделі меню, який надається до скрипту генетичного алгоритму*

```
"menu_structure": {
```

```
            "breakfast": ["білкове", "вуглеводне",
"фруктове"],
            "morningSnack": ["ранковий сніданок"],
            "lunch": ["білкове", "вуглеводне", "салат"],
            "afternoonSnack": ["обідній сніданок"],
            "dinner": ["білкове", "вуглеводне", "овощі"],
            "dinnerSnack": ["вечерній сніданок"]
        }
```

```
best_fitness 71.0
[1132, 1024, 1365, 36, 496, 413, 643, 2104, 157, 1900]
        id                            name  calories
0     2213  Соленый овсяноблин с начинкой     260.0
1     2069         Летняя овощная нарезка      23.0
2     2594                         Фрукты      72.0
3      205      Запечённая куриная грудка     154.0
4     1293       Пшеничная каша с грибами     208.0
5     1146        Салат с зеленой фасолью      87.0
6     1520                Орехи на выбор     257.0
7     3837    Индейка с грибами и овощами     227.0
8      410                 Киноа на воде     187.0
9     3529                  Травяной чай       0.0
10  Result                           NaN    1475.0
11     NaN                           NaN    1445.0
```

Рис. *25 Результат генерації після введення моделі меню*

Після введення цих моделей, результат виконання скрипту генетичного алгоритму став збирати комбінацію страв яка вже підходить і можна давати клієнтам на використання.

4.2.3 Дослідження проблем повторів страв та несумісних комбінацій

Під час генерування меню було виявлено ряд проблем по підбору страв:

1. Страви повторюються на протязі одного дня для основного прийому їжі.
2. Траплялися несумісні комбінації (молочні страви + рибні страви).

Таким чином було винайдено систему несумісностей страв (повна реалізація описано в 3 розділі). Ця система штрафів доповнювала вже існуючу метрику. Для її вирішення, кваліфіковані нутриціологи заповнювали матрицю (на даному етапі вона досягає розмірів 649 * 649) коефіцієнтами від 0 (страва повністю сумісна) до 1 (ставилася в тому випадку, коли страва не сумісна і сама на себе). Після її ведення зіткнулися з великим часом виконання скрипту до 30 секунд на добу. Після довгого аналізу та задавання питання на StackOverFlow результат був досягнутий.

Лістинг *12 Приклад вирішення проблеми з таблицею штрафів страв*

```
check_penalty = []
   for key in alll_codes:
       check_penalty.append(penalty_table.get(key))

   for k, i in enumerate(check_penalty):
       if (check_penalty[k] == None):
           check_penalty[k] = False
       check_penalty[k] = check_penalty[k] * penalty
```

Проблема була в тому, що під час читання цієї матриці використовується бібліотека numpy та тип даних від неї. Таким чином вона записувала таблицю як багатовимірний масив та потребувала більше

ітерацій для пошуку потрібної комбінації. Але за допомогою вбудованих типів даних в мову програмування Python то вся матриця була переведена в хєш таблицю. Результатом цього було вирішено проблеми: повторів страв, виділення страв які несумісні та зменшення в рази швидкість виконання скрипту — до 3 секунд.

4.3 Фінальний огляд отриманих результатів

Всі фінальні заміри відбувалися на людині яка користується даним застосунком.

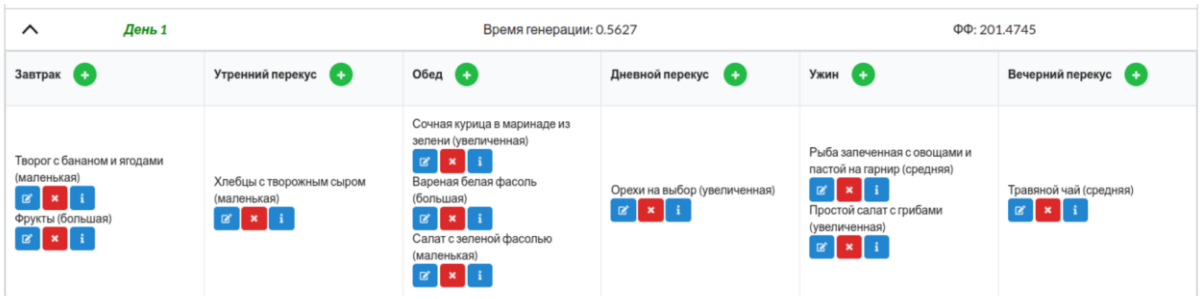


Рис. *26 Фінальний результат добового меню згенерований за допомогою генетичного алгоритму*

Із рисунку бачимо — меню яке задовольняє всім поставленим умов. Значення фітнес-функції яка дорівнює 200 та час виконання скрипту 0.5 секунд.

	Σ К	Σ Б	Σ ЖБ	Σ РБ	Σ Ж	Σ НЖ	Σ ННЖ	Σ У	Σ ПУ	Σ СУ	Σ клат
Цель	1445	390	234	156	540	108	432	571	85	485	25
Актуальные	1445	387	276	111	514	111	403	560	182	378	21

Рис. *27 Фінальний результат добової норми нутрієнтів*

По результатам виконання генерування генетичного алгоритму було підібрано такі страви, які повністю задовольняють добовій нормі клієнта по:

білкам (тваринним та рослинним), жирам (насичені та ненасичені), вуглеводам (складним та простим), калоріям та клітковини.

4.4 Висновок з розділу 4

1. Досліджена проблема з налаштуванням основних параметрів конфігурації генетичного алгоритму. Було продемонстровано базова конфігурація та її результати. Проаналізувавши всі проблеми була створена сама ідеальна конфігурація при яких отримали задовільні результати.
2. Дослідження та усунення проблем з якістю згенерованого меню. Завдяки введення до генетичного алгоритму таких систем штрафів, як: сума зважених квадратів різниць та багатовимірної матриці несумісних страв.
3. Розглянуто проблеми не “їстівного меню” та шлях її вирішення за допомогою моделей меню. Одну із моделей в прикладі розглянуто.
4. Було здійснено основна мета проекту.

ВИСНОВКИ

1. Поставлена проблема по генерації меню за допомогою генетичного алгоритму для клієнта з її параметрами, бажаннями, добовою нормою нутрієнтів (білків, жирів, вуглеводів та клітчатки).
2. Досліджені результати роботи розробленого скрипту генетичного алгоритму для генерування меню. З урахуванням всіх поставлених цілей. Точність працездатності системи залежить від розмічених страв нутриціологів. Допустима похибка в результаті 10%, що є допустимим результатом.
3. Розроблена та налаштована екосистема для працездатності срипту генетичного алгоритму.
4. Досліджено результати та шляхи калібровки параметрів налаштування генетичного алгоритму.
5. Впроваджено власна система штрафів, несумісність страв, для функції пристосованості популяцій. Налаштування та калібровка цієї системи для зниження потреб в системних ресурсах.
6. Досліджено принцип роботи генетичного алгоритму. Розглянута термінологія і як вона використовується в системі генетичного алгоритму.
7. Розглянуто основні оператори для працездатності генетичного алгоритму: відбір, схрещування та мутація.
8. Досліджена проблема генетичного алгоритму при розробці мобільного застосунку для системи обслуговування в реальному та фоновому часі.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Carlos Coello Coello David A. Van Veldhuizen Gary B. Lamont. Evolutionary Algorithms for Solving Multi-Objective Problems. 2013. P. 172–186.
2. Мічалевіц З. Генетичні алгоритми+структура данних=еволюційна програма. 94 с.
3. Mallawaarachchi V. Introduction to Genetic Algorithms — Including. URL: https://towardsdatascience.com/introduction-to-genetic-algorithmsincluding-example-code-e396e98d8bf3. (дата звернення 1.12.2022 р.)
4. JulieDawn Thompson: Statistics for Bioinformatics. 2016. P.43–51.
5. Jennifer S.Hallinan: Computational Intelligence in the Design of Synthetic Microbial Genetic Systems. 2013. P. 1–37
6. Что такое генетический алгоритм ? URL: https://neurohive.io/ru/osnovy-data-science/chto-takoe-geneticheskie-algoritmy/ (дата звернення 1.12.2022 р.)
7. Обзор методов отбора, скрещивания и мутации | Генетические алгоритмы на Python. URL: https://www.youtube.com/watch?v=ond8h5NqtGQ&t=23s (дата звернення 1.12.2022 р.)
8. Morgan Kaufmann. Multi-Tier Application Programming with PHP. 2004.

William Menke. Environmental Data Analysis with MatLab® or Python. 2019.

9. Andri Sunardi Suharjito: MVC Architecture: A Comparative Study Between Laravel Framework and Slim Framework in Freelancer Project Monitoring System Web Based. 2019. P. 134-141.
10. Preeti Saraswat. NumPy for Beginners: First Step to learn Data Science. 2017.
11. Wes McKinney. Python for Data Analysis: Data Wrangling with pandas, NumPy, and Jupyter 3rd Edition. 2022.

12. Altenberg L. The Schema Theorem and Price’s Theorem. Foundations of genetic algorithms. 1995. P. 65-73 .
13. Еволюційний алгоритм. URL: https://www.wiki.uk-ua.nina.az/ (дата звернення 1.12.2022 р.)
14. Shaban Mohammadi: Mathematics and Computers in Simulation. 2022. P. 243–312.

Декларація

академічної доброчесності

здобувача ступеня вищої освіти ЗНУ

Я, Калашник Максим Юрійович , студент 2 курсу, форми навчання денної, Інженерного навчально-наукового інституту ім. Ю.М. Потебні, спеціальність 121 Інженерія програмного забезпечення, адреса електронної пошти ipz17bd-24@stu.zsea.edu.ua, — підтверджую, що написана мною кваліфікаційна робота на тему «**Особливості застосування генетичного алгоритму при розробці мобільного застосунку для системи обслуговування**» відповідає вимогам академічної доброчесності та не містить порушень, що визначені у ст.42 Закону України «Про освіту», зі змістом яких ознайомлений.

— заявляю, що надана мною для перевірки електронна версія роботи є ідентичною її друкованій версії;

- згоден на перевірку моєї роботи на відповідність критеріям академічної доброчесності у будь-який спосіб, у тому числі за допомогою інтернет-системи, а також на архівування моєї роботи в базі даних цієї системи.

Дата 30.11.2022 _____ Калашник Максим Юрійович
(студент)

Дата 30.11.2022 _____ Заяц Валерій Іванович
(науковий керівник)