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

ВИПУСКНА КВАЛІФІКАЦІЙНА РОБОТА БАКАЛАВРА

на тему:

«Програмний модуль визначення та прогнозування позицій web-сайтів у пошуковій системі Google»

Галузь знань 12 «Інформаційні технології»

Спеціальність 122 «Комп'ютерні науки»

Освітня програма «Комп'ютерні науки»

Освітній рівень: бакалавр

Виконав: студент 4 курсу, групи КН-42

____Пилипенко В.І.
(прізвище та ініціали)

Керівник ___Мінаєва Ю. І._____
(прізвище та ініціали)

_____канд. техн. наук, доц._____
(науковий ступінь, звання)

Випускна кваліфікаційна робота бакалавра допущена до захисту рішенням кафедри *інтелектуальних технологій*
Протокол №11 від 06.02.2022р.
зав. кафедри_____ доц. Іларіонов О.Є.

Київ – 2022

КИЇВСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ ІМЕНІ ТАРАСА ШЕВЧЕНКА

Факультет інформаційних технологій

Кафедра інтелектуальних технологій

Спеціальність 122 «Комп'ютерні науки»

ЗАТВЕРДЖУЮ

Завідувач кафедри інтелектуальних технологій

Іларіонов О.Є.

_____

"___"_____202 2 р.

ЗАВДАННЯ
НА ВИПУСКНУ КВАЛІФІКАЦІЙНУ РОБОТУ СТУДЕНТОВІ

Пилипенку Вадиму Ігоровичу

(прізвище, ім'я, по батькові)

1. Тема проекту (роботи)

Програмний модуль визначення та прогнозування позицій web-сайтів у пошуковій системі Google

затверджена протоколом засідання кафедри від «23 » грудня 2022 р. №4

2. Термін здачі студентом закінченого проекту (роботи) 29 травня 2022 року

3. Вихідні дані до проекту (роботи)

Програмний модуль визначення та прогнозування позицій web-сайтів у пошуковій системі Google

4. Зміст розрахунково-пояснювальної записки (перелік питань, що їх належить розробити)

4.1 Аналітичний огляд програмного модулю отримання та прогнозування позицій
4.2 Проектування програмного модулю отримання та прогнозування позицій у пошуковій системі Google
4.3 Технологічні особливості реалізації програмного модулю отримання та прогнозування позицій у пошуковій системі Google

5. Перелік презентаційного матеріалу (з точним зазначенням обов’язкових презентацій)

5.1 Розгляд наявних систем отримання позицій 3 - 4 слайди

5.2 Основні вимоги до застосунку: 5 слайд

5.3 Проектування програмного модуля 6 - 12 слайди

5.4 Технологічні особливості реалізації програмного модулю отримання та прогнозування позицій у пошуковій системі Google 13 – 15 слайди

6. Консультанти з випускної кваліфікаційної роботи із зазначенням розділів випускної кваліфікаційної роботи, що їх стосуються

Розділ	Консультант	Підпис, дата	
		Завдання видав	Завдання прийняв
1			
2			
3			

7. Дата видачі завдання 15 лютого 2022 року

Керівник _____ / Мінаєва Ю.І./
(підпис) (ПІБ)

Завдання прийняв до виконання _____ / Пилипенко В.І./
(підпис) (ПІБ)

КАЛЕНДАРНИЙ ПЛАН

По р. №	Назва етапів випускної кваліфікаційної роботи	Термін виконання етапів випускної кваліфікаційної роботи	Приміт ка
1.	Обговорення з керівником постановки завдання та змісту пояснювальної записки	15.02.2022 - 22.02.2022	
2.	Аналіз постановки задачі, формалізація задачі, вибір методів та засобів реалізації поставленої задачі, аналіз літературних джерел	23.02.2022 - 08.03.2022	
3.	Проектування програмного модулю модулю визначення та прогнозування	09.03.22 - 28.03.22	

	позицій		
4.	Розробка та тестування програмного модулю визначення та прогнозування позицій	29.03.22 - 15.04.22	
5.	Оформлення пояснювальної записки, підготовка презентації	16.05.2022 - 29.05.2022	

Студент-дипломник  / Пилипенко В.І. /
(підпис) (ПІБ)

Керівник випускної кваліфікаційної роботи  / Мінаєва Ю.І. /
(підпис) (ПІБ)

Анотація

Пилипенко Вадим Ігорович виконав випускну кваліфікаційну роботу на тему «Програмний модуль визначення та прогнозування позицій web-сайтів у пошуковій системі Google» за спеціальністю 122 – «Комп’ютерні науки».

У випускній кваліфікаційній роботі проведено аналіз методів отримання позицій, досліджено метод прогнозування, розглянуто особливості обраних засобів, спроектована архітектура та реалізовано програмний модуль, як серверний додаток, що отримує позиції і прогнозує їх зміни на цій основі.

Ключові слова: оптимізація у пошуковій системі, серверний додаток, отримання позицій.

Summary

This bachelor's degree project, under the name of «Software module for capturing and forecasting positions of website pages in Google search engine» was completed by Vadym Pylypenko, a student of the 122 specialty – «Computer Science».

This graduation thesis studies methods of capturing positions of web-pages in the Google Search Engine and how to work with the parsed data, in particular – how to create a prognosis based on those positions. The architecture and the user interface were designed during completion. The used technology stack and selected tools of development were utilized to their full potential.

Keywords: seo, ruby application, ruby on rails server app, forecast.

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ І СКОРОЧЕНЬ

«Пошукова машина» («пошуковий система») - програмно - апаратний комплекс з веб-інтерфейсом, призначений для пошуку серед електронних документів в інтернеті [12].

«Ранжування» — процес сортування сайтів у пошуковій видачі за спаданням релевантності [11].

«Парсер» (з англ. - parser, to parse) - програма, здатна розпізнавати

попередньо вказані елементи коду з більших, глобальних сегментів коду [12].

«Стек технологій» (з англ. - technology stack) - умовна множина програм, яким користується команда розробників для досягнення певної мети.

«Позиція» - місце, яке займає посилання на сторінку у пошуковій видачі [11].

«Видимість» - відсотковий показник, який визначає кількість показів за вибраними ключовими фразами, виходячи з «позицій» сайту, які він займає у видачі пошукової системи [11].

«URI» (з англ. - Uniform Resource Identifier) - уніфікований ідентифікатор ресурсу. Це послідовність символів, що ідентифікує абстрактний або фізичний ресурс [11].

«URL» (з англ. - Uniform Resource Locator) - система уніфікованих адрес електронних ресурсів, або одноманітний визначник місцезнаходження ресурсу (файлу) [11].

«DNS» (з англ. Domain Name System - система доменних імен) - система, що дозволяє перетворювати символьні імена доменів на IP-адреси (і навпаки) в мережах TCP/IP [12].

«ТОП видачі» - сторінки «органічної» видачі. Наприклад, ТОП-1 буде першою сторінкою результату пошуку, ТОП-3 - третя сторінка і так далі [11].

«Трафік сайту» - кількість відвідувачів, які прийшли на сайт за певний проміжок часу (зазвичай за добу) [11].

«Органічна видача» — частина видачі пошукової системи, на яку не впливає рекламна видача. Формується під час ранжування сайтів пошукової системи [11].

«Семантичне ядро сайту» (або СЯ) - база пошукових слів, їх словосполучень та морфологічних форм, що найбільш точно характеризують електронний документ (сторінку), товари або послуги, які пропонує сайт [11].

«Google Analytics» (скорочено GA) – аналітичний інструмент, який надає Google для отримання статистики відвідувачів веб-сайтів [11].

«Web Worker» (або воркер, агент) - засіб для запуску скриптів у фоновому потоці. Потік Worker'а може виконувати завдання без втручання в інтерфейс користувача. До того ж, вони можуть здійснювати введення/виведення, використовуючи XMLHttpRequest. Існуючий Worker може надсилати повідомлення JavaScript коду-творцю через обробник подій, зазначений цим кодом і навпаки [12].

ВСТУП

У часи, коли інтернет зв’язаний з нашим життям більше ніж коли-небудь, а ринок послуг у різних галузях перенасичений, перспектива власного бізнесу може здатися недосяжною. Ці причини слугують основними чинниками створення та розвитку сфери SEO. Існує безліч способів оптимізації та відстеження сайту у пошуковій системі, більшість з яких дорогі та децентралізовані.

Тому, метою даної випускної кваліфікаційної роботи є реалізація централізованого та комплексного рішення для пошукової оптимізації та відстеження позицій веб-сайту.

Об’єкт дослідження – інтерфейсна реалізація взаємодії користувача з серверним додатком у вікні браузера у зв’язці з модулем отримання та прогнозування позицій веб-сайту у пошуковій системі.

РОЗДІЛ 1. АНАЛІТИЧНИЙ ОГЛЯД ОБЛАСТІ ЗАСТОСУВАННЯ РОЗРОБЛЮВАНОЇ СИСТЕМИ

1.1 Аналітичний огляд

У наш час широкого поширення набули SEO агентства. Основою їх «стеку технологій» є мова програмування PHP. Завдяки їй, можна збільшити «видимість» сайту, оптимізувати швидкість завантаження сторінок або навіть переробити алгоритм побудови «URI». Люди, які користуються пошуковою системою - цілеспрямовані користувачі мережі. Вони вже сформували свої інтереси та шукають саме той товар або послугу, що пропонує умовний сайт. Роботи з позиціонування сайту в пошукових системах — один з найважливіших заходів щодо розвитку бізнесу (підтримки конкурентоспроможності та залучення цільової аудиторії).

Таким чином з’являється потреба у керуванні наповнення веб-сторінок сайту та відстеження ефективності проведених робіт. Мета – полегшити процес та залишитися у межах власного додатку. Однак треба враховувати, що обсяг можливостей інтерфейсу обмежений, тому реалізація більш складних задач оптимізації залишаються поза ним.

Інтерес до SEO може надходити абсолютно з будь-якої галузі. Варто зазначити, що на момент написання цієї роботи, переважна кількість клієнтів у бізнесі пошукової оптимізації – інтернет магазини. Власники згаданих веб-сайтів найбільш схильні звернутися по SEO-послуги через потенціальну конкурентоспроможність у їх сфері. Кожен з них бажає досягти певних показників продаж, а це в свою чергу підтримує актуальність SEO-послуг. Наведемо приклади.

Розглянемо інтернет магазини, сайти нерухомості, послуг та подібні. Веб-сайти такого роду потребують постійної оптимізації через

непостійний та конкурентоспроможний ринок відповідної галузі. Сайти, що продають мають спільну ціль - закріпити позиції у «ТОП видачі» і залишатися там як якомога довше, враховуючи спроби конкурентів досягти того ж самого.

Наприклад, магазин автозапчастин. В інтернеті запчастини на продаж можна знайти всюди, і не обов'язково від імені магазинів. Отже, конкуренція в галузі буде більше ніж звичайно. Масштаб такого магазину дасть зібрати велике «семантичне ядро», що збільшить кількість запитів для «видачі». Отже, на такому проекті будуть доступні всі можливі задачі оптимізації, що буде підтримувати конкурентоспроможність проекту.

Схожу ситуацію можна знайти на ринку техніки та ігор. Магазини у цій галузі повинні конкурувати не тільки між собою, а ще і з незалежними продавцями. Тут, задачі оптимізації одночасно покращать видачу і допоможуть у закріпленні бренду на ринку.

У галузі нерухомості це пов’язано з підвищенням цін житлової площі. Наприклад, розглянемо два агентства. Чим більше клієнтів агентство зможе залучити, тим більша кількість потенційних продаж. Отже, якщо оптимізація першого агентства буде гірше другого, швидше за все воно призупинить діяльність швидше. Клієнт, звичайно, хоче зробити все можливе, щоби завадити цьому. За цих обставин, клієнти будуть більш ладні погодитися на умови, які будуть запропоновані.

У галузі медицини, наприклад, стоматологічні клініки, також будуть змагатись за клієнтів на просторі інтернету, за рахунок підвищення довіри до їх бренду. З цим їм допоможе підвищення видачі, одна з головних цілей SEO.

Веб-сайти, які продають товари або послуги ведуть нескінченні змагання за статус найбільшого постачальника. Це передбачає підвищення

видачі сайту та довіри до бренду. Від галузі до галузі, ціль залишається незмінною. Така динаміка підтримує актуальність та розвиток послуг у SEO сфері.

1.2 Визначення профілів зацікавлених сторін

Профілі зацікавлених сторін відображені в табл. 1.1.

Таблиця 1.1. Профілі зацікавлених сторін

Зацікавлена сторона	Вигода	Очікування (неявні потреби)	Основні інтереси (розкривають вигоди)	Обмеження
Інтернет магазини	Досягнення бажаної кількості продажів	Оптимізація та тимчасова підтримка веб-сайту	Підвищення «позицій» проекту у пошуковій системі	Апаратні обмеження
Медичні установи	Поліпшення довіри до проекту			

Також можна розглянути діаграму зацікавлених сторін ситуації/підприємців на прикладі моделі Менделоу (модель С). В обраній моделі зацікавлені сторони групуються в залежності від комбінації двох

змінних: влади і інтересу щодо результатів проекту. Влада – це рівень повноважень, що визначає здатність впливати на проект або компанію. Інтерес – рівень зацікавленості, який визначається бажанням підприємців впливати на проект або компанію. Зацікавлена сторона, що володіє істотною владою і має значний інтерес є більш впливовою, ніж та, у якої ступінь влади і зацікавленості нижче. Модель Менделоу дозволяє визначити зацікавлені сторони, які будуть найбільш впливовими при реалізації проекту, а також заздалегідь позначити зони потенційних конфліктів інтересів (як правило, це відноситься до сторін, що потрапляють в зону перетину влади високого рівня і інтересу високого рівня). [1]

Ідентифікація зацікавлених сторін за моделлю Менделоу відображена на рис. 1.1.

Рисунок 1.1 – Зацікавлені сторони за моделлю С

1.3 Існуючі підходи

Коректне розпізнавання посилань на сторінки конкретного веб-сайту важливе для відстеження прогресу оптимізації. Сучасні сервіси розпізнавання мають ряд недоліків, таких як, похибки у позиціях, відсутність можливості звітності за період часу, більший одного місяця, низька швидкість і продуктивність при отриманні позицій одного або

одразу декількох проектів. Перший пріоритет займає отримання позицій. Тому задача має бути реалізована з уникненням асоційованих проблем.

Існує декілька принципові підходів до отримання позицій: вручну і за допомоги сервісів «парсингу» по ключовим словам.

Переваги першого підходу полягають у простому застосувані. Не доведеться чекати і сплачувати за сервіс, але цей спосіб передбачає людський фактор. Отже мінусами цього підходу є велика ймовірність помилки та зайві витрати часу.

Другий підхід передбачає роботу зі сторонніми компаніями. Вони, в свою чергу, надають сервіс з безкоштовним і платним планом. Функціонал різних сервісів може відрізняється, тому одразу можна виділити наступні переваги: низький коефіцієнт помилки, зручність у відображенні статистики. З мінусів: обмеження сервісу по кількості запитів, відсутність можливості підтримки відстеження статистики протягом великого періоду часу та інші обмеження, унікальні на різних платформах.

1.4 Огляд існуючих рішень

Google Search Console

Компанія Google надає власний інструмент для відстеження позицій та відвідувань веб-сайту. Сервіс забезпечує можливості аналізу даних, їх відображення у вигляді графіків.

Переваги:

- безкоштовний
- розширена статистика
- можливість групування ключових слів

Недоліки:

- обмеження кількості доданих запитів
- відсутність статистики конкуренції
- відсутність можливості додавання запитів
- передбачає розміщення специфічного коду на веб-сайті

Serpstat

Serpstat це платформа пошукової оптимізації. Вона дає можливість аналізу ключових слів, відстеження рангів, перегляд зворотних посилань, надає аудит сайту та деталізований аналіз ключових слів конкуренції. Serpstat має наступні функції: кластеризація ключових слів та текстова аналітика на основі штучного інтелекту.

Переваги:

- розширена статистика
- модель SaaS
- безкоштовна версія
- ведення статистики

  Недоліки:

- обмеження внесення ключових слів

Topvisor

Схожий на попередні, Topvisor надає свої послуги не безкоштовно, але надає ширший функціонал. Наприклад, пряме задання конкуруючих веб-сайтів, або вибір між пошуковими «роботами», регіональна статистика.

Переваги:

- легкість в налаштуванні;
- відсутність ліміту по запитам
- розширена аналітика

Недоліки:

- Вартість.

Усі розглянуті рішення в тій чи іншій мірі вже використовують функцію отримання позицій. Різниця полягає у швидкостях та масштабу досліджуваних даних. Тому необхідність проектування оригінального рішення залежить тільки від поставленої задачі та доступних у використанні засобів.

Найбільш подібне до теми, що розглядається, готове рішення Topvisor і Serpstat. Але в межах випускної кваліфікаційної роботи був обраний компроміс між двома варіантами реалізації.

1.5 Постановка задачі програмного модулю визначення та прогнозування позицій веб-сайтів

Задачі випускної кваліфікаційної роботи:

Тема випускної кваліфікаційної роботи: Програмний модуль визначення та прогнозування позицій веб-сайтів у пошуковій системі Google.

Основною метою даної роботи є реалізація одного із можливих варіантів методів отримання позицій сайту на основі структури сторінки видачі пошукової системи Google.

Об’єкт дослідження – засоби оптимізації та отримання позицій сторінок у пошуковій системі Google.

Предметом є метод визначення позицій з графічним відображенням та маніпуляція HTML-кодом веб-сторінок.

У ході виконання випускної кваліфікаційної роботи має бути реалізована система визначення та прогнозування позицій веб-сайтів у пошуковій системі Google, яка буде складатися з front- та back- end частин.

Отже, для досягнення визначеної мети необхідно вирішити наступні завдання:

- змоделювати систему;
- розробити макет інтерфейсу;
- визначити метод тлумачення даних веб-сторінок Google;
- написати програмний код, який керуватиме моделями графіку в залежності від отриманих даних із «парсера»;
- розробити програму для тестування.

У результаті буде отримана система захоплення позицій сторінок із можливістю подальшого використання за рахунок контексту.

Замість представлення програмного модулю визначення позицій веб-сайтів як «чорної скрині» для подальшого моделювання можна використати методологію IDEF0. IDEF0 використовується для представлення функціональних меж системи, основні елементи якої: діяльність (або процес), вхідні та вихідні дані, обмеження або елементи керування, та механізми, що здійснюють активність. Характерною рисою даної нотації є декомпозиція процесів. Тому, моделі IDEF0 чітко визначені, добре структуровані, легкі для розуміння, модифікації та використання. [1,2,3,4]

Контекстна діаграма є вершиною деревовидної структури діаграм та показує призначення системи (основну функцію) і її взаємодію з зовнішнім середовищем. Для системи, що розглядається основним процесом є зовнішнє керування серверним додатком.

Елементи керування:

Математичні методи – використовуються при визначенні нових координат на графіках, що відображають статистику позицій, а також для прогнозування. У рамках цієї роботи будуть задіяні статистичні методи прогнозування, а саме - метод зваженого ковзного середнього;

Актуальні тенденції SEO – правила та тренди пошукової системи, на які потрібно спиратися для маніпулювання HTML-наповненням

Структура сторінок Google – правила отримання позицій.

Механізми:

SEO-спеціаліст – такий користувач, якому довірено оптимізацію проекту.

Вхідні дані:

Дані проекту – передбачає семантичне ядро веб-сайту, його сторінки, дані доступу та звіти щодо сплачених послуг;

Дані користувача – логін та пароль користувача, які будуть визначати роль авторизованого користувача. Вона в свою чергу диктує обмеження повноважень у додатку для підтримання модульності;

Дані позицій - отримані парсером позиції сторінок проекту у пошуковій видачі.

Вихідні дані:

Змінений HTML-код - результат маніпуляції наповненням сторінок користувачем;

Статистика позицій веб-сайту - результат отримання позицій;

Прогноз позицій веб-сайту – передбачуваний результат оптимізації.

Контекстна діаграма для програмного модулю визначення та прогнозування позицій web-сайтів у пошуковій системі Google на рис. 1.1.

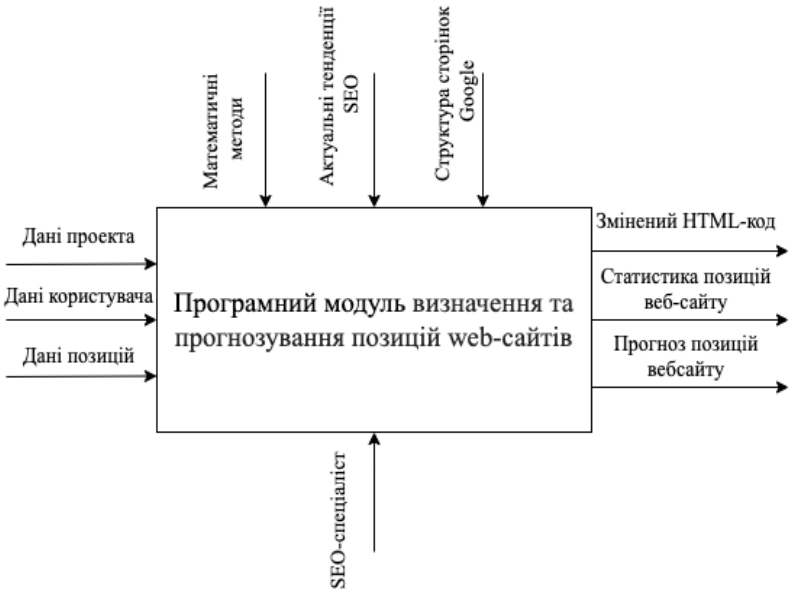

Рисунок 1.1 – Контекстна діаграма програмного модулю визначення та прогнозування позицій

Вимоги до розроблювальної системи

Системні вимоги до розроблюваної системи:

- підтримка можливих форматів файлів таблиць;
- встановлене програмне середовище.

Функціональні вимоги до розроблюваної системи:

- бізнес-вимоги (система має відобразити статистику позицій графічно);
- користувацькі вимоги (система має виконувати визначені команди: отримання позицій, маніпуляція HTML-контентом);
- технічні вимоги (система має виводити стан позицій до їх отримання, якщо отримали їх раніше);
- системні вимоги (вищеописані).

Нефункціональні вимоги:

- бізнес-правила (розглядаються в розділі 2.1);
- зовнішній інтерфейс (інтерфейс має бути інтуїтивно зрозумілим);

- атрибути якості (система має бути продуктивною, зручною в експлуатації та розширюваною);
- обмеження (система може працювати в рамках ПЗ (програмного забезпечення) описаного в системних вимогах).

Висновок до першого розідлу

У результаті завершення першого розділу ми розглянули предметну область застосування розроблюваної системи, розглянули зацікавлені сторони та провели порівняння з існуючими рішеннями.

На основі отриманої інформації була поставлена задача для програмного модуля визначення і прогнозування позицій. Були виділені тема, мета, об’єкт дослідження та предмет роботи. Для представлення програмного модулю визначення позицій веб-сайтів була побудована контекстна діаграма за методологію IDEF0. В рамках діаграми були визначені основні елементи: процес, вхідні та вихідні дані, елементи керування, та механізми, що здійснюють активність. Побудована діаграма чітко визначена, добре структурована та легка для розуміння, модифікації та використання.

РОЗДІЛ 2. ПРОЕКТУВАННЯ ПРОГРАМНОГО МОДУЛЮ ОТРИМАННЯ ТА ПРОГНОЗУВАННЯ ПОЗИЦІЙ

2.1 Функціональний та бізнес-аналіз програмного модулю отримання та прогнозування позицій

Для більш детального опису архітектури програмного модулю визначення та прогнозування позицій web-сайтів у пошуковій системі Google, далі було наведено дерево функцій. Воно допоможе поліпшити розуміння результату та можливих напрямків розвитку роботи.

Дерево функцій для програмного модулю визначення та прогнозування позицій web-сайтів у пошуковій системі Google зображене на рис. 1.2.

Рисунок 2.1 – Дерево функцій програмного модулю визначення та прогнозування позицій

Опис процесів

1. Прогнозування позицій веб-сайту – ведення статистики позицій та прогнозування на її основі.

A11 - Отримання позицій веб-сайту – автоматизований процес, який запускається SEO-спеціалістом. Його результатом будуть позиції у пошуковій видачі сторінок веб-сайту по ключовим словам;

A12 - Прогнозування позицій веб-сайту - передбачення майбутнього стану позицій веб-сайту, на основі аналізу минулих результатів отримання позицій;

2. Облік проектів клієнтів – ведення бази даних клієнтських веб-сайтів.

A21- Облік сторінок проекту – ведення списку доступних сторінок веб-сайту;

A22 - Маніпуляція HTML контентом – процес, який передбачає зміну вмісту певних семантично-важливих елементів HTML-коду веб-сторінки. Наприклад, заміна тексту у <title></title> [11].

3. Ведення обліку користувачів – впорядкування потенціальних користувачів для підтримки безпечного функціонування програмного модулю.

A31 - Додання користувача – введення нового користувача до відповідної таблиці у базі даних, створення нового логіну і паролю;

A33 - Надання користувачу ролі – введення спеціальних обмежень на діяльність користувачів у додатку, спираючись на їх

діяльність поза додатком. Наприклад: роль «SEO-спеціаліст», «Адміністратор», «Розробник»;

A32 - Закріплення користувача за проектом – призначення запису конкретного користувача до запису конкретного проекту для чіткого розподілення відповідальності;

A34 - Видалення користувача – видалення запису конкретного користувача з таблиці у базі даних.

4. Формування розкладу – впорядкування завдань у рамках програмного модуля

A41- Призначення користувача до поставленої задачі – розподіл завдань між користувачами;

A42 - Редагування розкладу – передбачає надання завдань, видалення та зміни їх статусу, наприклад з «перевіряється» на «здано»;

A43 - Звільнення користувача від поставленої задачі – перерозподіл відповідальності певного користувача.

5. Формування рахунків за послуги – ведення звітності стосовно наданих послуг у рамках програмного модулю.

Після проведення функціонального аналізу можна перейти до декомпозиції розглянутої раніше контекстної діаграми «програмного модулю визначення та прогнозування позицій » (рис 1.1).

Після авторизації та завантаження (якщо ядро ще не було завантажене у модуль) користувачем, за відповідної команди, програмний модуль почне процес отримання позицій. При виборі команди “Отримати позиції”, модуль буде створювати n-кількість воркерів, число n залежить

від кількості сторінок веб-сайту, а також одного додаткового воркера, який по завершенню процесу отримання позицій, переранжує отримані позиції у ТОП-видачі. На основі отриманої статистики, користувач може отримати прогноз, вибравши відповідну функцію. Декомпозиція процесу «Прогнозування позицій» А1, відображена на рис. 2.2.

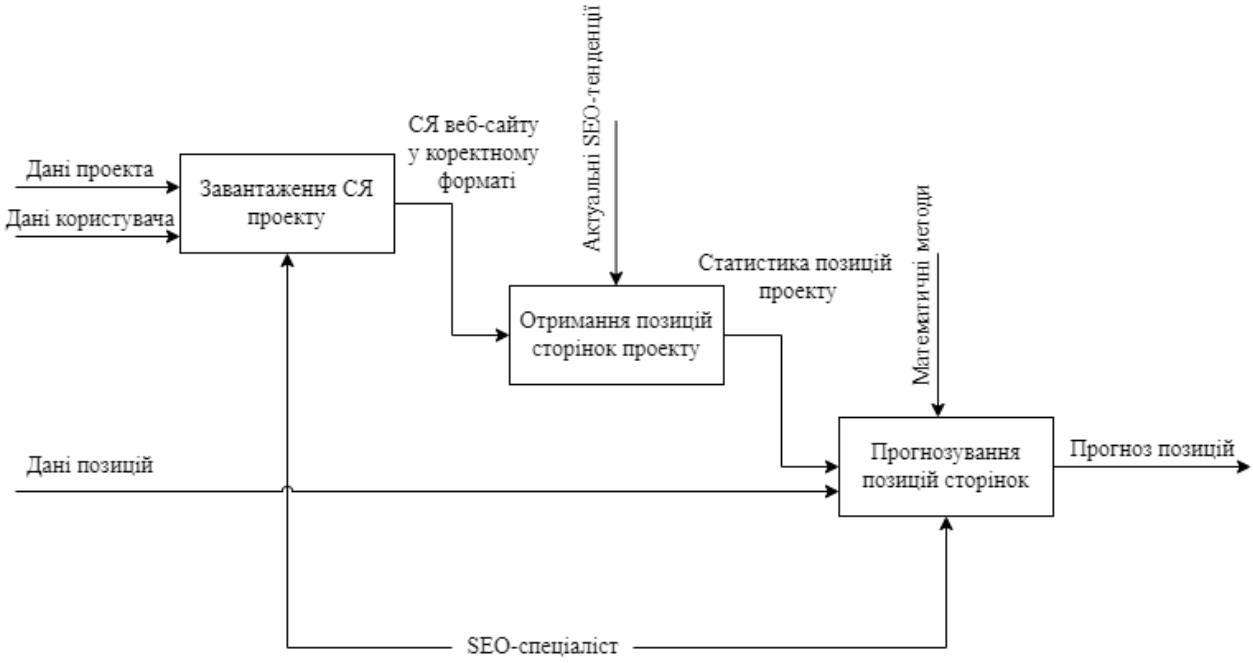

Рисунок 2.2 – Декомпозиція процесу «Прогнозування позицій А1»

У відповідності до вищезгаданого можна визначити ієрархію процесів. Узагальнений процес роботи програмного модулю складається з таких підпроцесів:

- Авторизація у програмному модулі;
- З'єднання з сервером;
- Робота з модулем.

Процес роботи програмного модулю у нотації VAD (Value Added Chain Diagram) відображено на рис. 2.3.

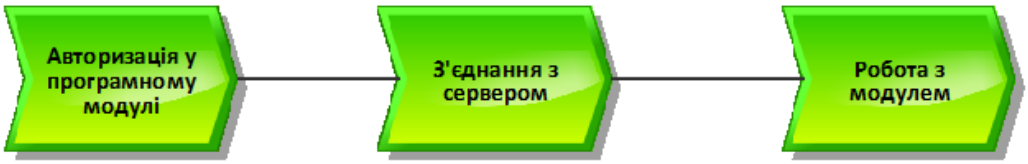

Рисунок 2.3 – Процес «Узагальнена робота модуля» у нотації VAD

Процес робота з модулем складається із таких підпроцесів:

- Отримання даних;
- Доповнення моделі;
- Обробка отриманих даних;
- Відображення змін на представленні.

Узагальнений процес роботи з модулем у нотації VAD відображено на рис. 2.5.

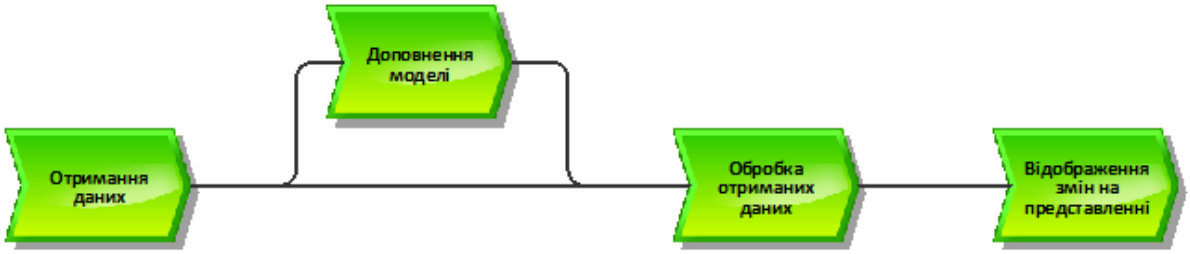

Рисунок 2.4 – Процес «Робота з модулем» VAD

Спираючись на отримані діаграми, можемо зобразити узагальнений життєвий цикл програмного модулю на наступному рисунку.

Рисунок 2.5 – Життєвий цикл програмного модулю

Більш детальний процес роботи програмного модулю у нотації EPC (Event-Driven Process Chain) відображено на рис. 2.6.

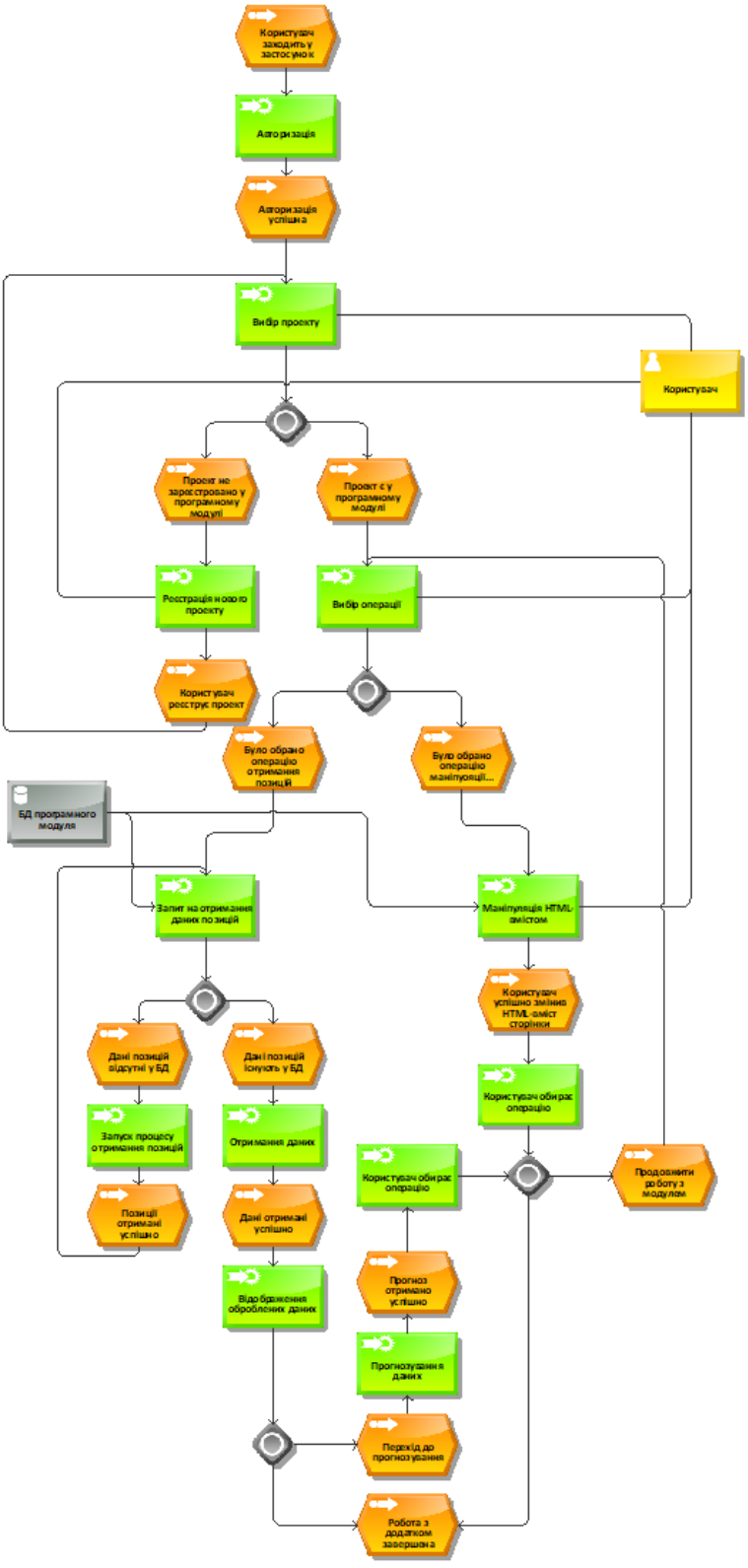

Рисунок 2.6 – Процес роботи програмного модулю у нотації EPC

2.2 Архітектура програмного модуля отримання та прогнозування позицій

Оскільки мета роботи передбачає роботу за даними, далі потрібно спроектувати БД програмного модуля. Спочатку, наведемо ключові об’єкти та їх атрибути.

На основі наведнего вище, маємо наступні об’єкти:

Проект:

- Номер;
- Протокол з’єднання;
- Домен;
- Статус проекту;
- Дата створення;
- Дата оновлення.

Конфігурація проекту:

- Номер;
- Номер проекту;
- Ключ проекту;
- Остання дата отримання позицій;
- Статус отримання позицій;
- Домен-нотатка;
- Користувач домену;

- Пароль домену;
- Хост домену;
- GIT-нотатка;
- Користувач GIT;
- Пароль GIT;
- Посилання на GIT;
- Користувач CloudFlare;
- Пароль CloudFlare;
- Посилання на CloudFlare;
- Хостинг- користувач;
- Пароль від хостингу;
- Хостинг- посилання;
- Хостинг- нотатка;
- Користувач адмін-панелі;
- Пароль адмін-панелі;
- Посилання на адмін-панель;
- Користувач SSH;
- Хост SHH;
- Порт SHH;

- Користувач FTP;
- Хост FTP;
- Пароль FTP;
- Порт FTP;
- Дата створення;
- Дата оновлення.

Позиції проекту:

- Номер;
- Номер проекту;
- Позиція;
- Середня позиція;
- Видимість;
- Дата створення;
- Дата оновлення.

Сторінки:

- Номер;
- Статус;
- Номер проекту;
- Перевірка

- Посилання;
- Заголовок H1;
- Назва сторінки;
- Опис сторінки;
- Текстове наповення;
- JSON-текст;
- Перевірка

Конфігурація сторінки:

- Номер;
- Номер сторінки;
- Заголовок H1;
- Назва сторінки;
- Опис сторінки;
- Текстове наповення;
- Статус синхронізації;
- Дата синхронізації;
- Автор синхронізації;
- Теги сторінки;
- Дата створення;

- Дата оновлення;
- Автор оновлення.

Семантика:

- Номер;
- Номер сторінки;
- Запити;
- Дата створення;
- Дата оновлення.

Звіти семантики:

- Номер;
- Номер сторінки;
- Звіт;
- Дата створення;
- Дата оновлення

В результаті визначення об’єктів предметної області, на наступному етапі визначимо їх зв’язки.

Таблиця 1. Визначення зв’язків між сутностями

№	Сутності, що утворюють зв’язок	Тип зв’язку	Пояснення

1	Projects – Project_Positions (Проект – Позиції проекта)	Один – до - багатьох	Один проект може мати багато записів в табличці позицій, так як сайт має n-кількість сторінок.
2	Projects – Project_Config (Проект – Конфігурація проекта)	Один – до - багатьох	Один проект має одну відповідну таблицю конфігурації. Проекти не можуть мати однакові конфігурації, тому що це буде означати що вони утворюють один проект.
3	Projects – Pages (Проект – Сторінки)	Один – до - багатьох	Один проект може мати багато записів в таблиці сторінок, так як сайт має n-кількість сторінок.
4	Pages – Pages_Config (Сторінки – Конфігурація сторінки)	Один – до - багатьох	Одна сторінка має багато параметрів.

5	Pages – Semantic_Reports (Сторінки – Семантичні звіти)	Один – до - багатьох	Одна сторінка має n-кількість запитів, які надають відображення відповідної сторінки(семантика). З цього випливає, шо одна сторінка може мати n-звітів відповідно до обсягу семантики.
6	Pages – Pages_Config (Сторінки – Конфігурація сторінки)	Один – до - багатьох	Одна сторінка має багато параметрів.

7	Pages – Semantics (Сторінки – Семантика)	Один – до - багатьох	Одна сторінка має n-кількість запитів, які результують у відображенні відповідної сторінки.

У результаті визначення об’єктів предметної області перейдемо до побудови діаграми «Сутність - зв’язок». Результат побудови діаграми представлено на рис. 2.3.

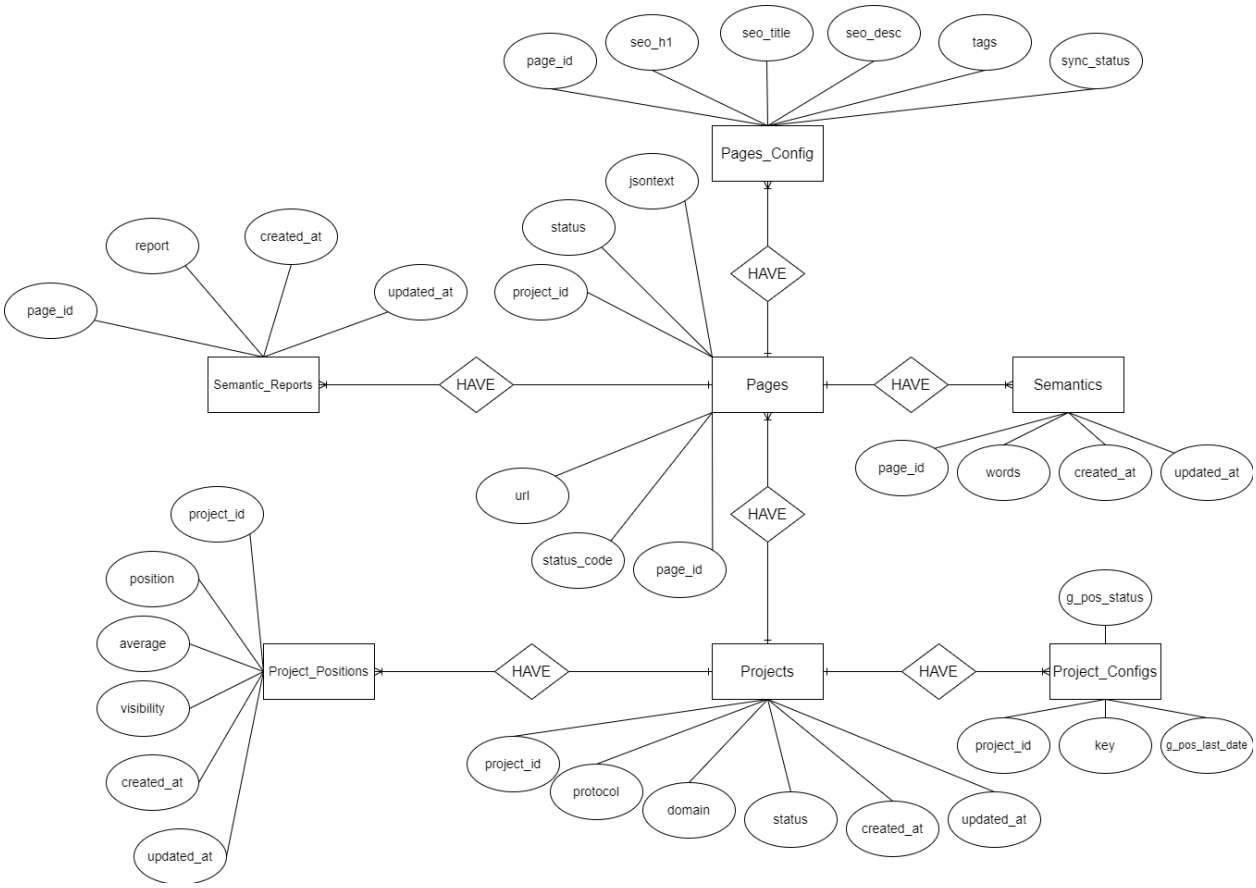

Рисунок 2.3 – Діаграма «Сутність-зв’язок»

Наступним етапом опишемо склад та характеристики атрибутів таблиць даталогічної моделі.

Таблиця 2. Склад та характеристики атрибутів таблиць логічної моделі

Склад та характеристики атрибутів таблиць логічної моделі

Сутність	Назва атрибуту	Ключ	Обов’язкове значення	Тип даних	Обмеження
Таблиця Projects – «Проекти»					
project_id	номер	ПК	Так	Ціле число	Від 1 і більше
protocol	Протокол з’єднання		Ні	Текст	
domain	домен		Ні	Текст	
status	Статус проекту		Ні	Ціле число	Від 1 і більше
created_at	Дата створення		Так	Дата число	більше

updated_at	Дата оновлення		Так	Текст	

Продовження таблиці 2

Таблиця Project_Configs – «Конфігурації проекту»					
id	номер	ПК	Так	Ціле число	Від 1 і більше
project_id	номер проекту	ЗК	Так	Ціле число	Від 1 і більше
key	Ключ проекту		Так	Текст	
g_pos_last_date	Остання дата отримання позицій		Ні	Дата	Від 1 і більше
g_pos_status	Статус отримання позицій		Ні	Ціле число	255 символів
paneldomain_notic	Домен-		Ні	Текст	255

e	нотатка				символів
paneldomain_user	Користувач домену		Ні	Текст	255 символів
paneldomain_pass	Пароль домену		Ні	Текст	255 символів
paneldomain_host	Хост домену		Ні	Текст	255 символів
git_notice	GIT-нотатка		Ні	Текст	255 символів
git_user	Користувач GIT		Ні	Текст	255 символів
git_pass	Пароль GIT		Ні	Текст	255 символів
git_url	Посилання на GIT		Ні	Текст	255 символів
cloudflare_user	Користувач		Ні	Текст	255

	CloudFlare				символів
cloudflare_pass	Пароль CloudFlare		Ні	Текст	255 символів
cloudflare_url	Посилання на CloudFlare		Ні	Текст	255 символів
host_user	Хостинг-користувач		Ні	Текст	255 символів
host_pass	Пароль від хостингу		Ні	Текст	255 символів
host_url	Хостинг-посилання		Ні	Текст	255 символів
host_notice	Хостинг-нотатка		Ні	Текст	255 символів
admin_user	Користувач адмін-панелі		Ні	Текст	255 символів
admin_pass	Пароль		Ні	Текст	255

	адмін-панелі				символів
admin_url	Посилання на адмін-панель		Ні	Текст	255 символів
ssh_user	Користувач SSH		Ні	Текст	255 символів
ssh_host	Хост SHH		Ні	Текст	255 символів
ssh_port	Порт SHH		Ні	Текст	255 символів
ftp_user	Користувач FTP		Ні	Текст	255 символів
ftp_host	Хост FTP		Ні	Текст	255 символів
ftp_pass	Пароль FTP		Ні	Текст	255 символів
ftp_port	Порт FTP		Ні	Текст	255

					символів
created_at	Дата створення		Так	Дата	
updated_at	Дата оновлення		Так	Дата	
Таблиця Project_Positions – «Позиції проекту»					
id	номер	ПК	Так	Ціле число	Від 1 і більше
project_id	номер проекту	ЗК	Так	Ціле число	Від 1 і більше
position	рівень вологості		Так	Ціле число	Від 1 і більше
average	Середня позиція		Ні	Ціле число	Від 1 і більше
visibility	Видимість		Ні	Ціле число	Від 1 і більше

created_at	Дата створення		Так	Дата	
updated_at	Дата оновлення		Так	Дата	
Таблиця Pages – «Сторінки»					
Page_id	номер	ПК	Так	Ціле число	Від 1 і більше
status_code	Статус		Ні	Ціле число	0-1
project_id	номер проекту	ЗК	Так	Ціле число	Від 1 і більше
check	Перевірка		Ні	Дата	
url	Посилання	ЗК	Так	Ціле число	Від 1 і більше
h1	Заголовок H1		Ні	Текст	

title	Назва сторінки		Ні	Текст	
description	Опис сторінки		Ні	Текст	
content_text	Текстове наповення		Ні	Текст	
jsontext	JSON-текст		Ні	Текст	
status	Перевірка		Ні	Ціле число	0-1
Таблиця Page_Configs – «Конфігурації сторінки»					
id	номер	ПК	Так	Ціле число	Від 1 і більше
page_id	номер сторінки	ЗК	Так	Ціле число	Від 1 і більше
Seo_h1	Заголовок H1		Ні	Текст	

Seo_title	Назва сторінки		Ні	Текст	
Seo_description	Опис сторінки		Ні	Текст	
Seo_content_text	Текстове наповення		Ні	Текст	
Sync_status	Статус синхронізації		Ні	Ціле число	0-1
Sync_data	Дата синхронізації		Ні	Дата	
Sync_author	Автор синхронізації		Так	Текст	
tags	Теги сторінки		Ні	Текст	
created_at	Дата створення		Так	Дата	
updated_at	Дата		Так	Дата	

	оновлення				
update_author	Автор оновлення		Ні	Текст	
Таблиця Semantics – «Семантика»					
id	номер		Так	Ціле число	Від 1 і більше
Page_id	Номер сторінки	ЗК	Так	Ціле число	Від 1 і більше
words	Запити		Ні	Дата	
created_at	Дата створення		Так	Дата	created_at
updated_at	Дата оновлення		Так	Дата	updated_at
Таблиця Semantics – «Звіти_Семантики»					

id	номер		Так	Ціле число	Від 1 і більше
Page_id	Номер сторінки	ЗК	Так	Ціле число	Від 1 і більше
report	Звіт		Ні	Дата	
created_at	Дата створення		Так	Дата	created_at
updated_at	Дата оновлення		Так	Дата	updated_at

На наступному кроці побудуємо логічну модель із використання опису наведеного у таблиці 2. Результат побудови моделі наведено на рис. 2.4.

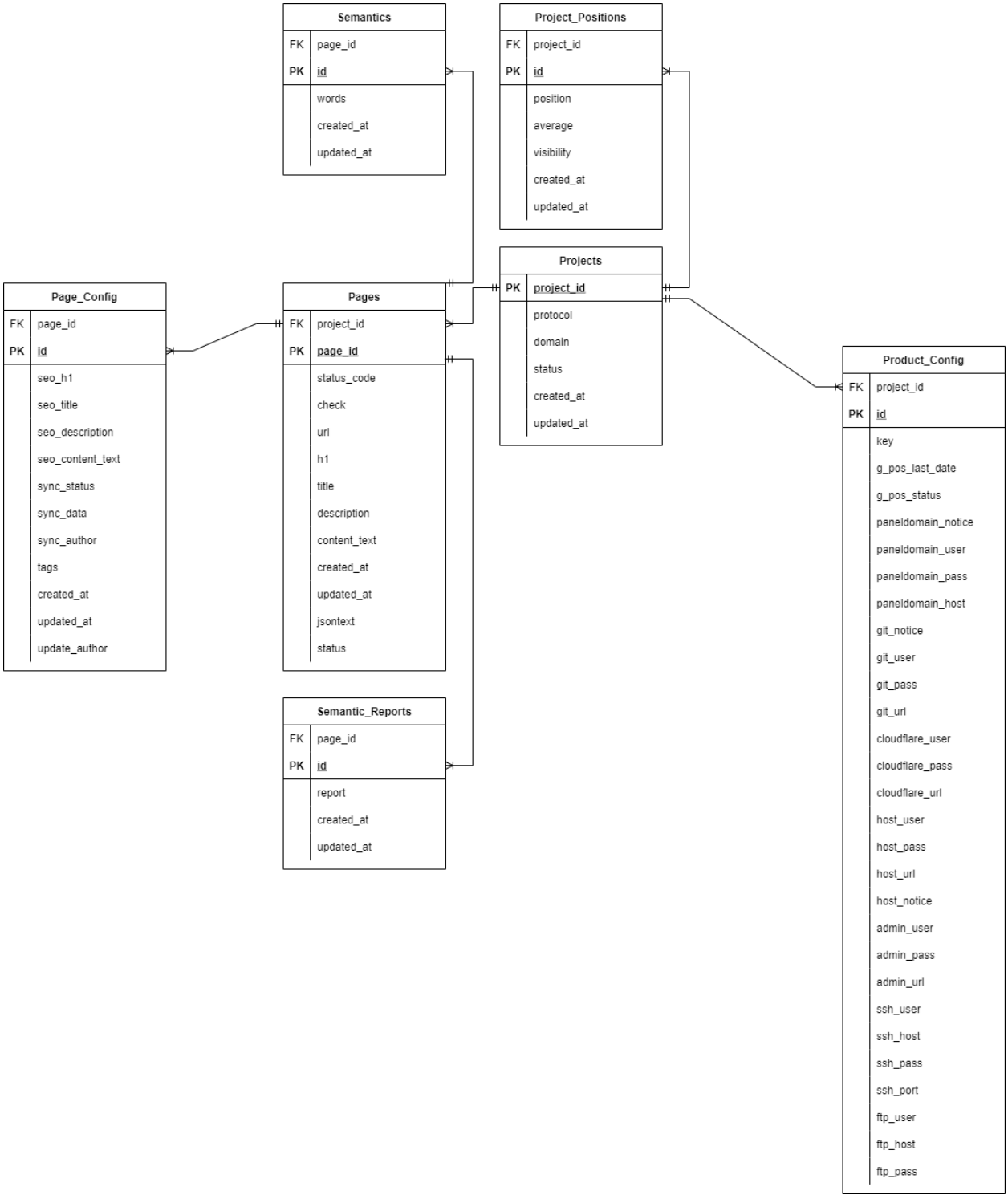

Рисунок 2.4– Представлення логічної моделі

Використаємо результат побудови логічної моделі та перейдемо до фізичної. Для цього використаємо інструмент MySQL Workbench за

вказівками джерела [5]. На цьому етапі визначимо зовнішні ключі. Результат побудови моделі наведено на рис. 2.5.

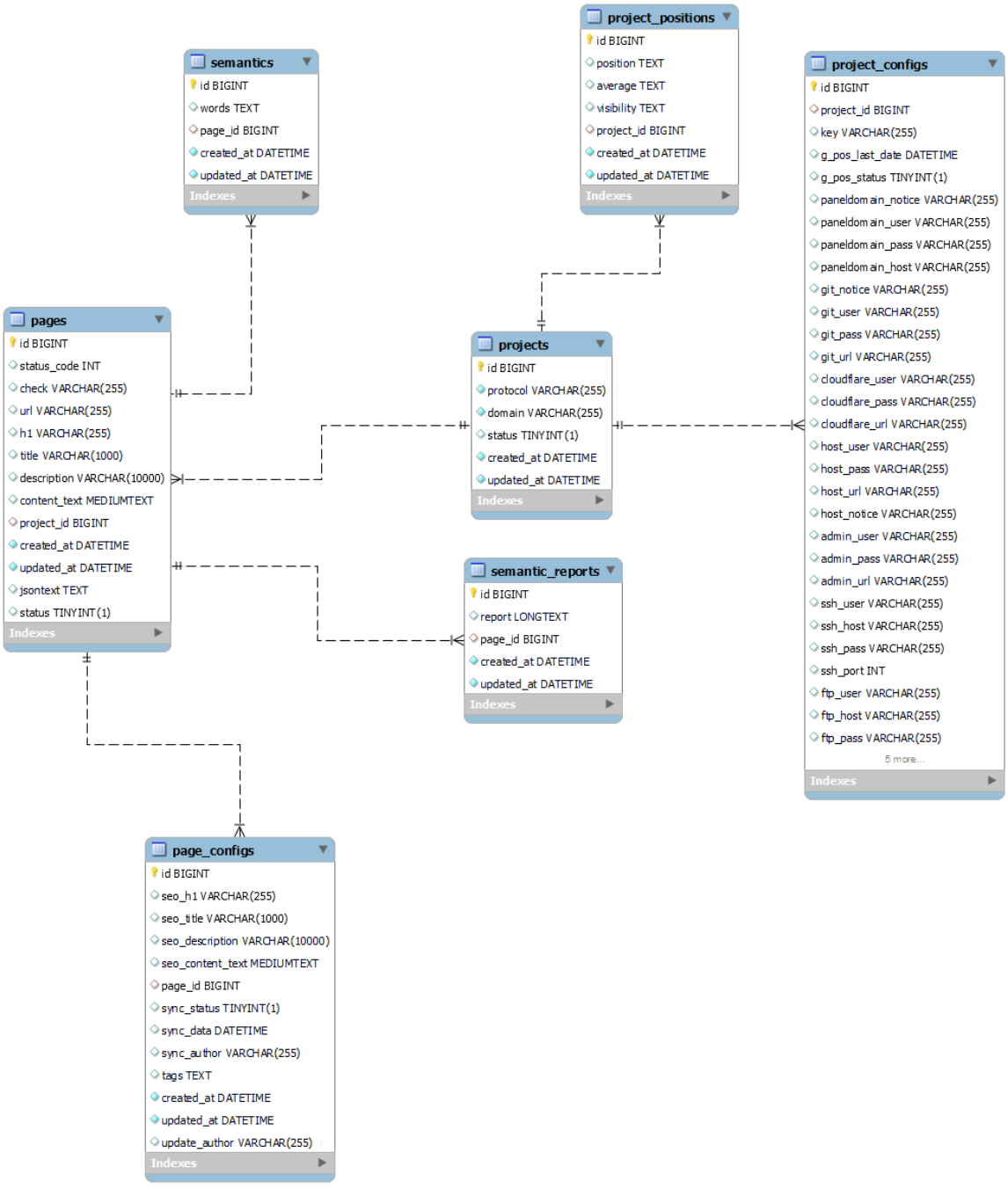

Рисунок 2.5 – Представлення фізичної моделі

2.3 Узагальнений опис програмного модулю отримання та прогнозування позицій

Головна задача програмного модулю – отримати дані та видати прогноз на їх основі. В рамках дипломної роботи, дані – позиції сторінок web-сайту, а прогноз – зміна позицій на наступний місяць.

У програмному модулі буде облік проектів, з якими він працює на поточний час. Кожний проект матиме списки асоційованих сторінок, даних доступу до сервера, статистики позицій та завдань. Модуль буде зберігати дані семантичного ядра відповідних сайтів для відстеження статистики позицій сторінок проекту за відповідним ключовим запитом. Користувач може змінювати вміст конкретних сторінок web-сайту з допомогою модуля.

У майбутньому планується розширення та поліпшення додаткового функціоналу модуля організації завдань та відстеження рахунків за надані послуги.

2.4 Опис обраних рішень як складових програмного модулю отримання та прогнозування позицій

Середовище розробки

Visual Studio Code (або VS Code) — редактор коду, створений Microsoft для Windows, Linux і macOS. Його набір функцій включає підтримку налагодження, виділення синтаксису, інтелектуальне завершення коду, фрагменти, рефакторинг коду та вбудований контроль версій.

Переваги, актуальні у рамках роботи:

- Легкий у використанні;

- Не вимагає багато ресурсів системи;
- Широка бібліотека розширень, які полегшують робочий процес;
- Підтримка обраних мов програмування;
- Розширення контролю версій.

Враховуючи, що робота буде виконуватися на ноутбуці та потребує чіткої документації для звітування у наступних розділах, можна сказати, що VSCode повністю задовольняє потреби для розробки рішення дипломної роботи. Найголовніше, що підтверджує це - обране середовище не потребує багато ресурсів від персонального комп’ютера і підтримує контроль версій.

macOS Terminal
це стандартний застосунок операційної системи macOS, емулятор консолі (терміналу) компанії Apple Inc. Оскільки робота буде виконана на операційній системі macOS, Terminal буде основним інструментом для виконання різноманітних задач. Наприклад, перегляд файлів логів, встановлення самого Ruby, Ruby on Rails та сумісних бібліотек та інші необхідні задачі можливо буде здійснити через Terminal.

Мови програмування

Ruby це інтерпретована мова програмування високого рівня загального призначення, яка підтримує кілька парадигм програмування. Вона була розроблена з акцентом на продуктивність і простоту програмування. У Ruby все є об’єктом, включаючи примітивні типи даних [8].

Ruby динамічно типізується і використовує збір сміття та компіляцію «точно і вчасно». Як згадано вище, вона підтримує кілька парадигм програмування, включаючи процедурне, об’єктно-орієнтоване та функціональне програмування. За словами творця, на Ruby вплинули Perl, Smalltalk, Eiffel, Ada, BASIC і Lisp [8].

Переваги:

- Лаконічний і простий синтаксис;
- Повністю об'єктно-орієнтована мова програмування;
- Містить автоматичний прибиральник сміття;
- Має незалежну від ОС підтримку невитискальної багатонитковості.

Ruby було обрано для дипломної роботи через вищезгадані переваги. У порівнянні з PHP, ця мова програмування краще задовольняє потреби роботи. Це значно зменшить час розробки, спростить програмний код та буде сприяти роботі над модулем у майбутньому.

Ruby on Rails
це об'єктно-орієнтований програмний фреймворк для створення веб-застосунків, написаний на мові програмування Ruby. Ruby on Rails використовує патерн модель-вид-контролер (Model-View-Controller) для веб-застосунків, а також забезпечує їхню інтеграцію з веб-сервером і сервером бази даних [9].
Переваги:

- Ruby on Rails надає механізми повторного використання, що дозволяють мінімізувати дублювання коду у веб-застосунку;

JavaScript
динамічна, об'єктно-орієнтована прототипна мова програмування. Реалізація стандарту ECMAScript. Найчастіше використовується для створення сценаріїв веб-сторінок, що надає можливість взаємодіяти з користувачем, керувати браузером, асинхронно обмінюватися даними з сервером, змінювати структуру та зовнішній вигляд веб-сторінки. Дуже зручно, коли потрібно поновити вміст сторінки з великим обсягом інформації без потреби перезавантаження цілого документа.

JavaScript класифікують як прототипну (підмножина об'єктно-орієнтованої), скриптову мову програмування з динамічною типізацією. Крім прототипної, JavaScript також частково підтримує інші парадигми програмування (імперативну та частково функціональну) і деякі відповідні архітектурні властивості, зокрема: динамічна та слабка типізація, автоматичне керування пам'яттю, прототипне наслідування, функції як об'єкти першого класу [10].
Переваги:

- Можливість оновлення вмісту веб-сторінки без її перезавантаження;
- Потужна бібліотека класів;

React
це відкрита бібліотека JavaScript для створення інтерфейсів користувача, яка покликана вирішувати проблеми часткового оновлення вмісту веб-сторінки. React дозволяє створювати великі веб-застосунки, які використовують дані, котрі змінюються з часом, без перезавантаження сторінки. Його мета полягає в тому, щоб бути

швидким, простим. React обробляє тільки користувацький інтерфейс у застосунках. Це відповідає патерну MVC, і може бути використане у поєднанні з іншими JavaScript бібліотеками або в великих фреймворках MVC. Він також може бути використаний з React на основі надбудов, щоб піклуватися про частини без користувацького інтерфейсу побудови веб-застосунків [6].

Методи прогнозування

Ковзне середнє

один із інструментів аналізу випадкових процесів та часових рядів, що полягає в обчисленні середнього підмножини значень.

Розрахунок показника на прогнозований момент часу будується шляхом усереднення значень цього показника за кілька попередніх моментів часу. Ковзне середнє не є скаляром, а є випадковим процесом. Розмір підмножини, від якої обчислюється середнє значення може бути як сталим, так і змінним [9].

У методі рухомого середнього виділяють основні різновиди: просте ковзне середнє, зважене ковзне середнє і експоненційне ковзне середнє. У рамках роботи було обрано метод зваженого ковзного середнього [9].

Метод зваженого ковзного середнього

розширення методу простого ковзного середнього. У ньому враховується, що дані за минулі періоди часу впливають на можливі зміни неоднаково. Для цього вводиться поняття "ваги". Зручно в якості ваг брати частки, що демонструють ступінь впливу даних

вихідного часового ряду в залежності від їх віддаленості від прогнозованих [9]. Формула для розрахунків має наступний вигляд:

$$f_k = \frac{\sum_{i=1}^{N} \omega_{k-1}\, x_{k-1}}{\sum_{i=1}^{N} \omega_{k-1}} \qquad (2.1)$$

Де: f_k - прогноз позицій на момент часу;

t_{ki}. N - число попередніх моментів часу, що використовуються при розрахунку;

x_{ki} - реальне значення показника в момент часу t_{ki},

ω_k- вага, з яким показник x_{ki} використовується в розрахунках [9].

Висновок до другого розділу

По завершенню другого розділу було спроектовано БД прогармного модуля, опсано його роботу у нотаціях VAD та EPC, а також визначено його життєвий цикл. При проектуванні БД усі сутності були описані повністю і зв’язки між ними перевірені.

Також були обрані рішення для подальшої розробки. Під час викоаннння пункту 2.4 ми визначили середовище розробки, мови програмування і метод прогнозування. Обрані рішення допоможуть досягти поставлених задач наприкінці попереднього розділу.

РОЗДІЛ 3. ТЕХНОЛОГІЧНІ ОСОБЛИВОСТІ РЕАЛІЗАЦІЇ ПРОГРАМНОГО МОДУЛЮ ОТРИМАННЯ ТА ПРОГНОЗУВАННЯ ПОЗИЦІЙ У ПОШУКОВІЙ СИСТЕМІ GOOGLE

3.1 Опис реалізації складових програмного модуля

Відповідно до наведеної архітектури, визначимо складові програмного модуля. Сам модуль розроблений за патерном програмування MVC (Model-View-Controller) та працює з БД. Отже кожний тип сторінки має свою модель, контроллер та представлення, які працюють з відповідними даними.

У першу чергу розглянемо реалізацію інтерфейсу програмного модуля, далі - реалізацію отримання і прогнозування позицій.

Представлення

Задача полягає в тому, щоб розробити лаконічний та легкий для сприйняття користувацький інтерфейс. Так як у майбутньому планується розширення функціоналу, що в свою чергу означає збільшення обсягу користувачів, потрібно врахувати їх потреби та поведінку у програмному модулі.

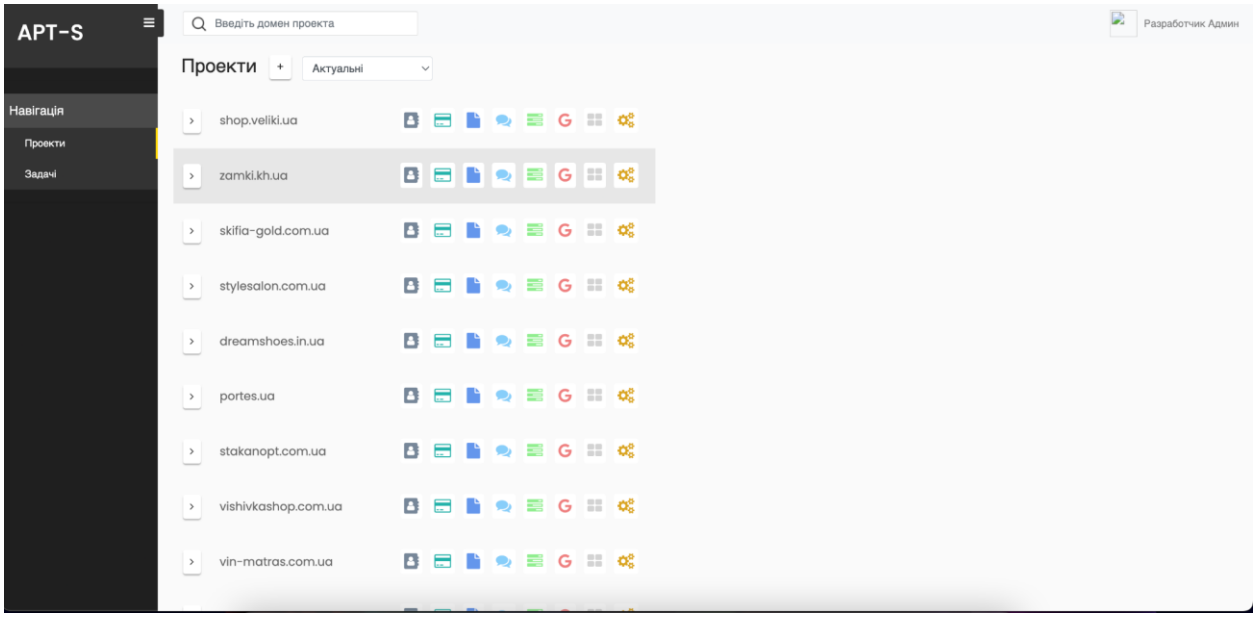

Рисунок 3.1 - Головна сторінка програмного модуля

Розберемо макет головної сторінки. Цей шаблон можна поділити (зображено на рисунку 3.2) на три основні частини (виділені кольором). Синім кольором виділено панель навігації. Вона використовується у всіх шаблонах, її також можна зменшити для зручності (зображено на рисунку 3.3). Зеленим кольором виділено головну область додатку - список проектів, з якими ми працюємо. По правий бік назви кожного проекта є набір швидких функцій, потрібних для роботи з відповідним проектом. Їх функціонал буде розглянуто пізніше у розділі. Червоним кольором виділено область можливих розширень інтерфейсу у майбутньому. На рисунку 3.3 зображено приклад розширення - виведення інформації фінансових звітів по наданим послугам.

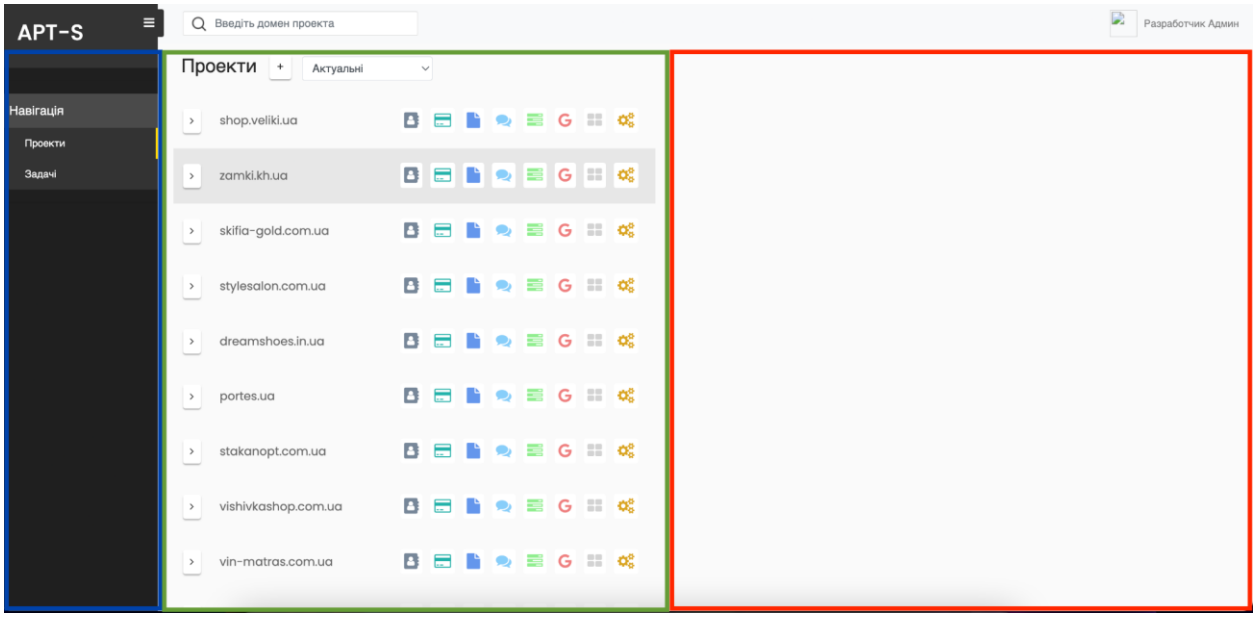

Рисунок 3.2 - Основні зони програмного модуля

На наступному рисунку (3.3), можна побачити приклад реалізації фінансового розширення програмного модулю. Тут планується коротко виводити важливу інформацію щодо виставлених рахунків за послуги з можливістю переходу до відповідного рахунку.

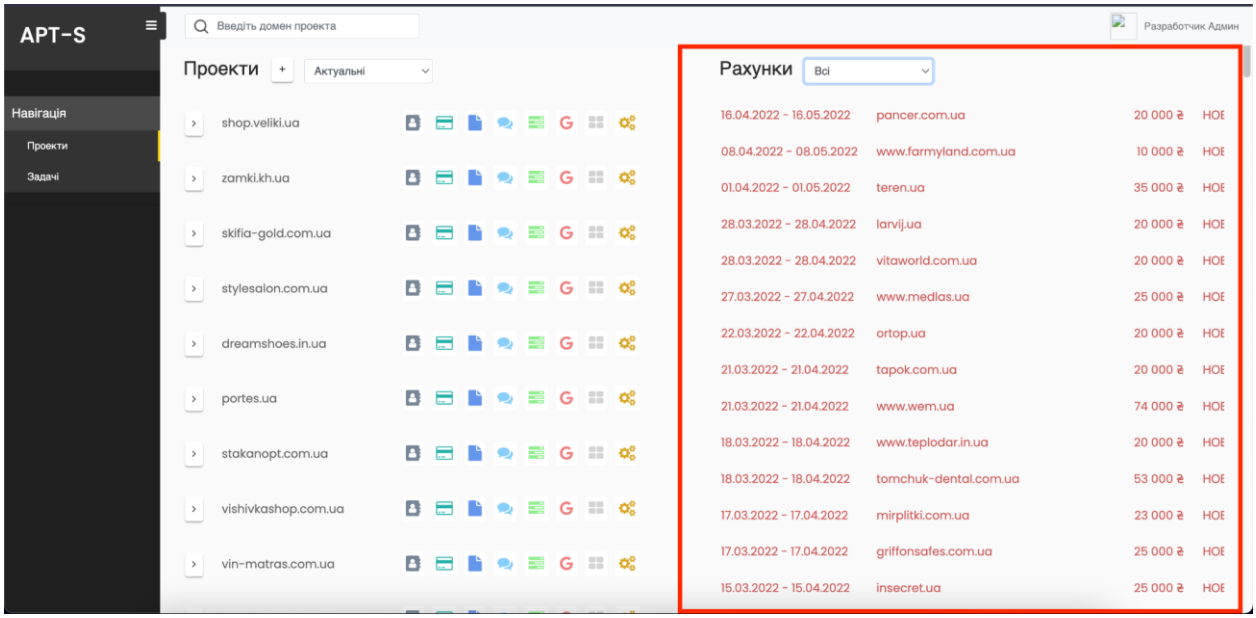

Рисунок 3.3 - Приклад розширення головної сторінки (виділено червоним)

Далі роздивимось функціонал основної частини сторінки. На рисунку 3.5 зеленим кольором виділено раніше встановлену основну зону інтерфейсу програмного додатку. У цій зоні виводиться список проектів, які були внесені у БД програмного модулю з можливістю сортування за критеріями актуальності, зображених на рисунку 3.4. Актуальні - поточні проекти, з якими умовне агентство продовжує роботу на момент часу. Не актуальні - проекти, співпраця з якими була завершена.

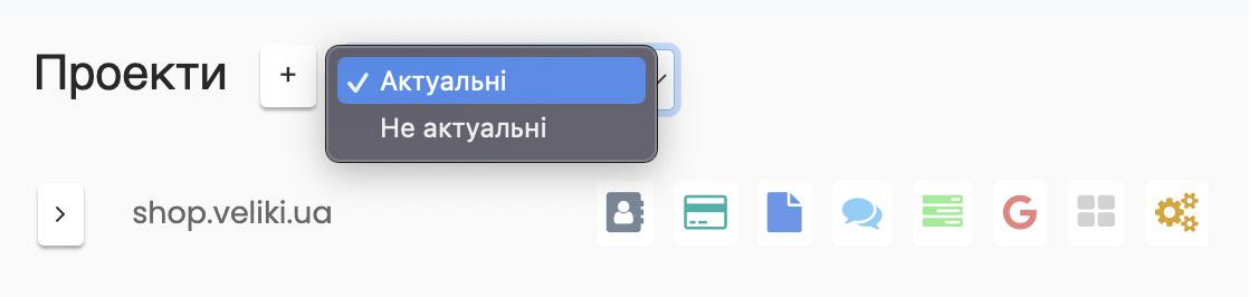

Рисунок 3.4 - Критерії сортування

Блідо-зеленим виділено зону відповідного проекту. З лівого боку зображена назва проекту, з правого боку - перелік швидких функцій, виділених жовтим кольром.

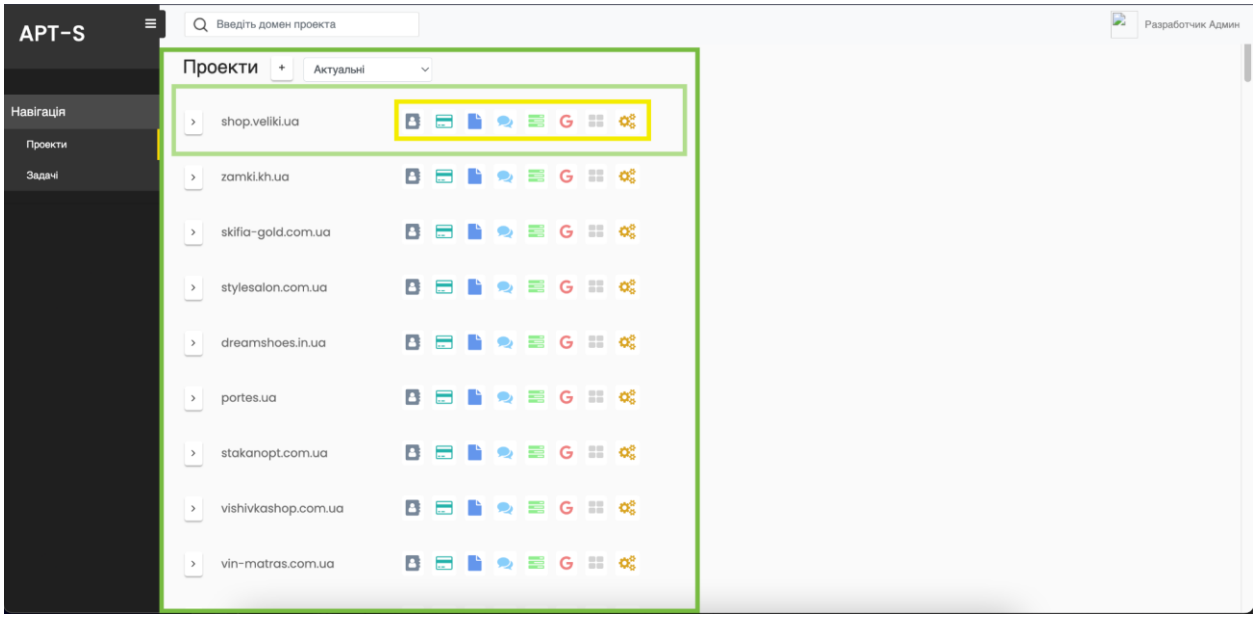

Рисунок 3.5 - Зона проектів програмного модуля

Розглянемо набір швидких функцій, виділених жовтим.

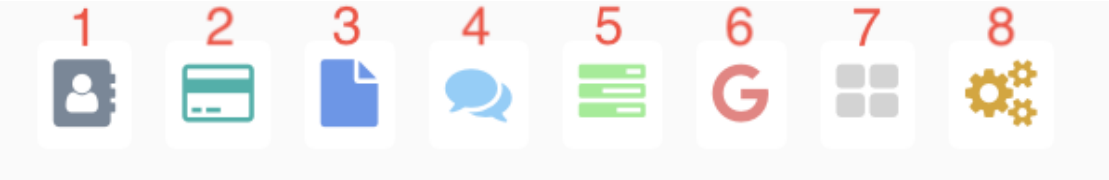

Рисунок 3.6 - Швидкі функції для роботи з проектом

1. Картка клієнта - Контактна інформація відповідальних за проект та базова інформація стосовно проекту;

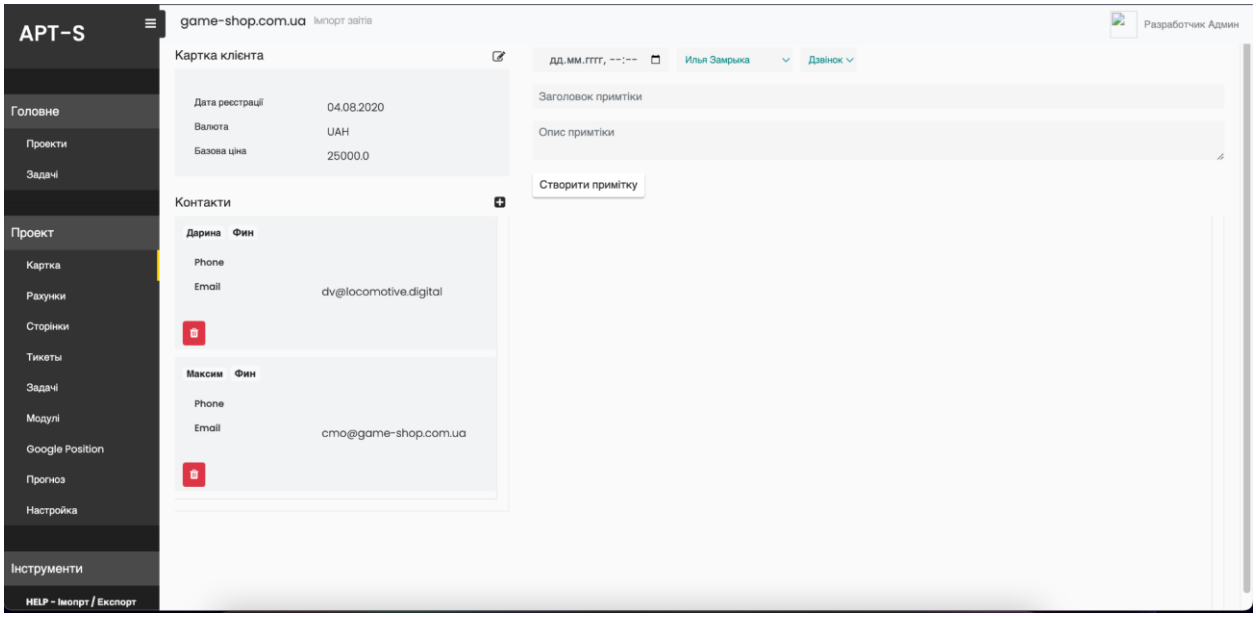

Рисунок 3.6.1 - Сторінка “Картка клієнта”

2. Рахунки - список наданих послуг по відповідному проекту;

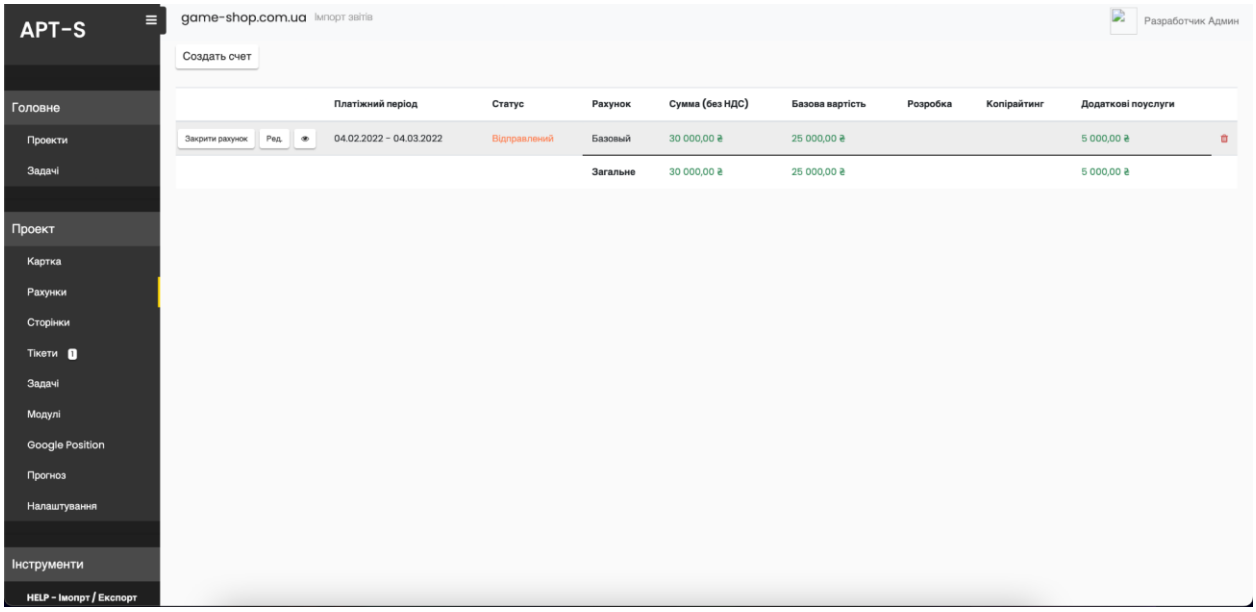

Рисунок 3.6.2 - Сторінка “Рахунки про проекту”

3. Сторінки - список сторінок проекту;

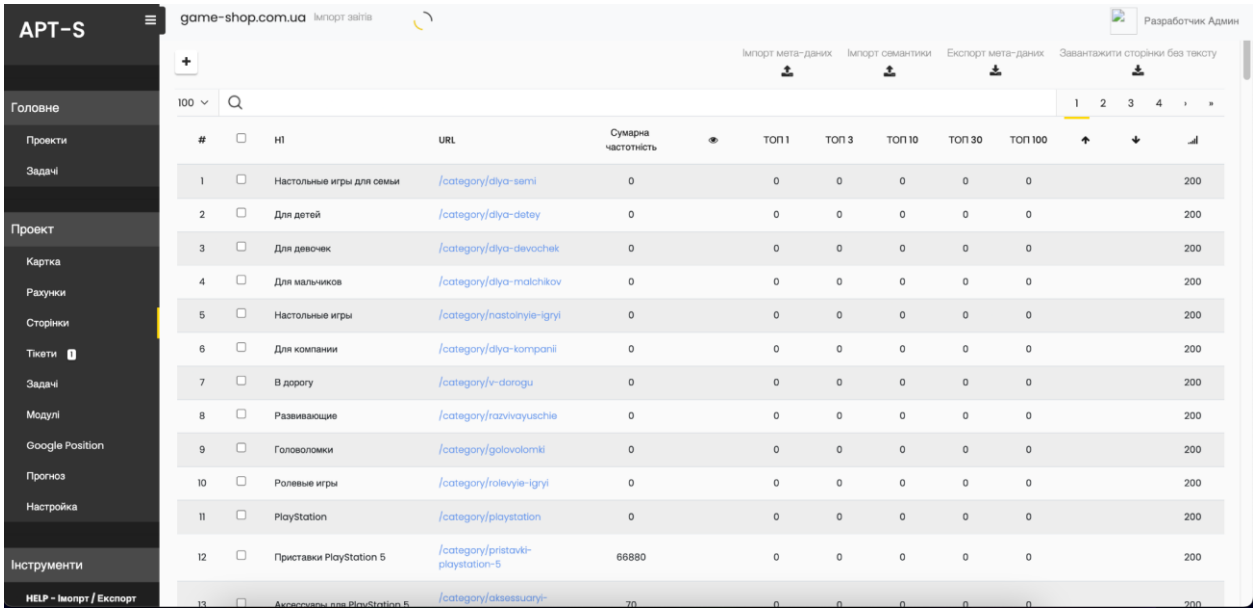

Рисунок 3.6.3 - Сторінка з переліком “Сторінки” проекту

4. Тікети - перелік повідомлень технічного характеру;

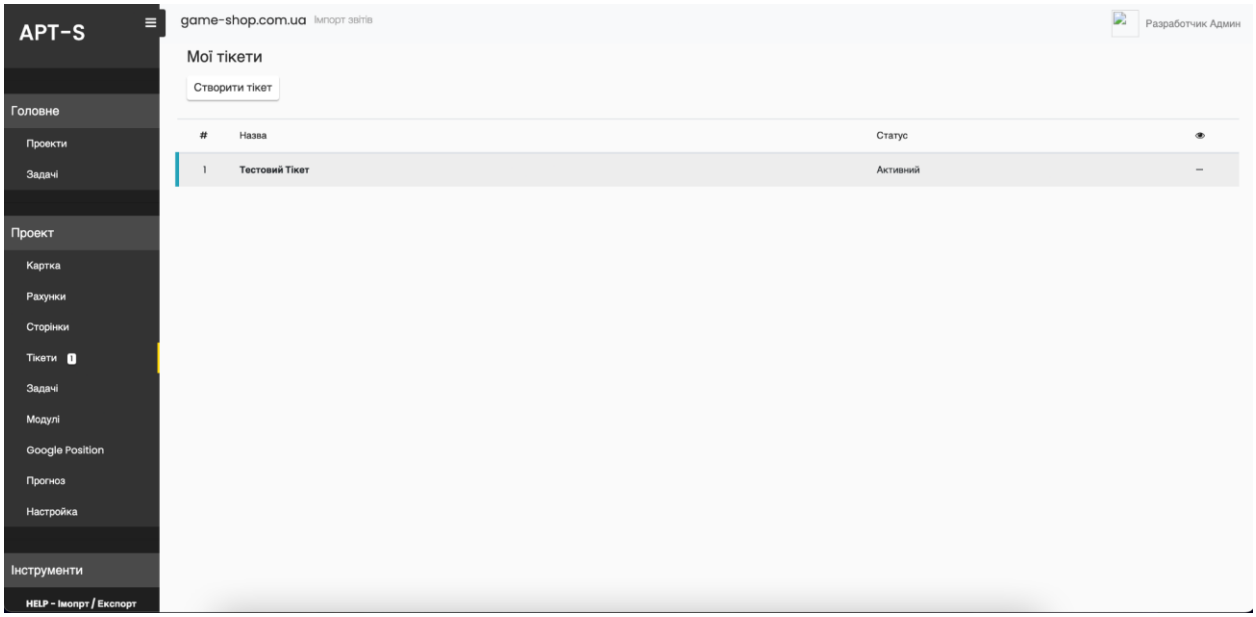

Рисунок 3.6.4 - Сторінка з переліком “Тікетів” проекту

5. Задачі - розклад задач по відповідному проекту;

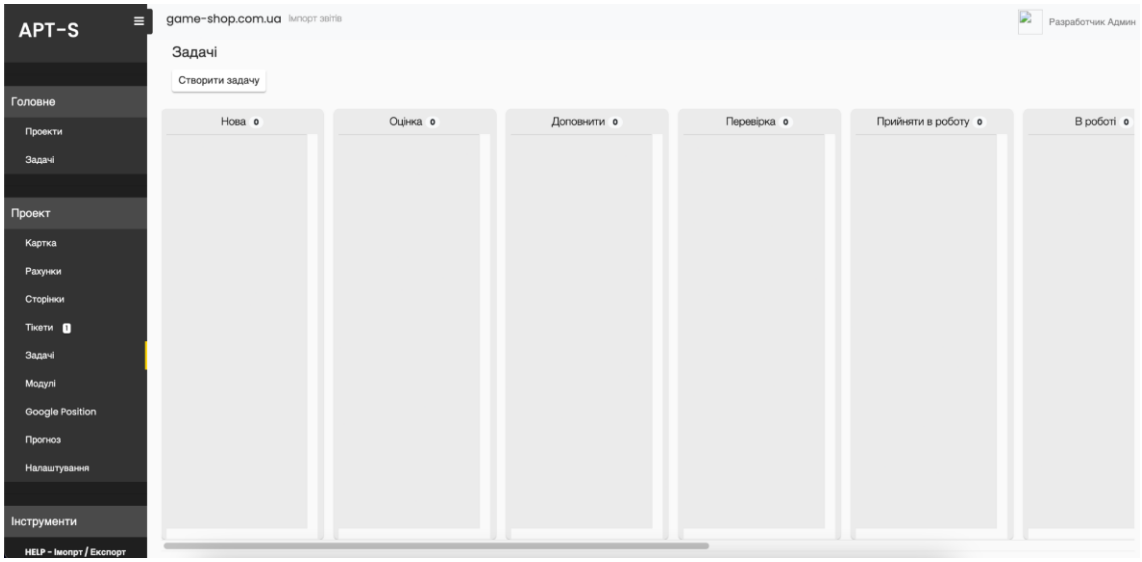

Рисунок 3.6.5 - Сторінка з переліком задач проекту

6. Звіти позицій - сторінка з візуалізацією звіту з позицій web-сайту у пошуковій системі;

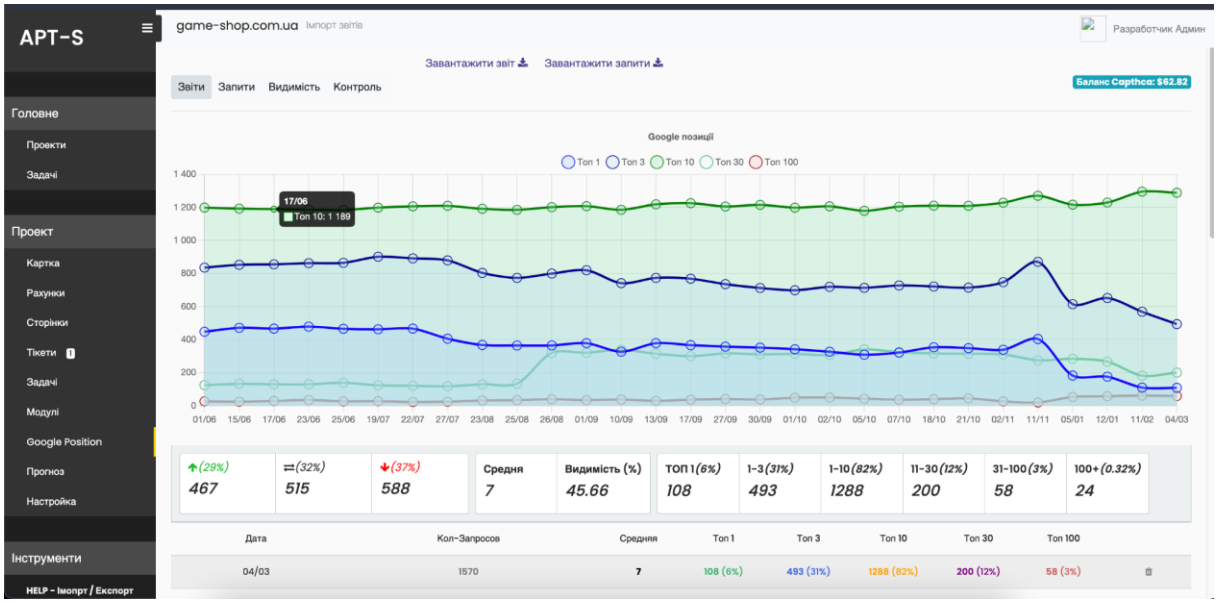

Рисунок 3.6.6 - Сторінка звітів позицій проекту

7. Встановлені модулі - перелік модулів, наданих проекту умовним агентством;

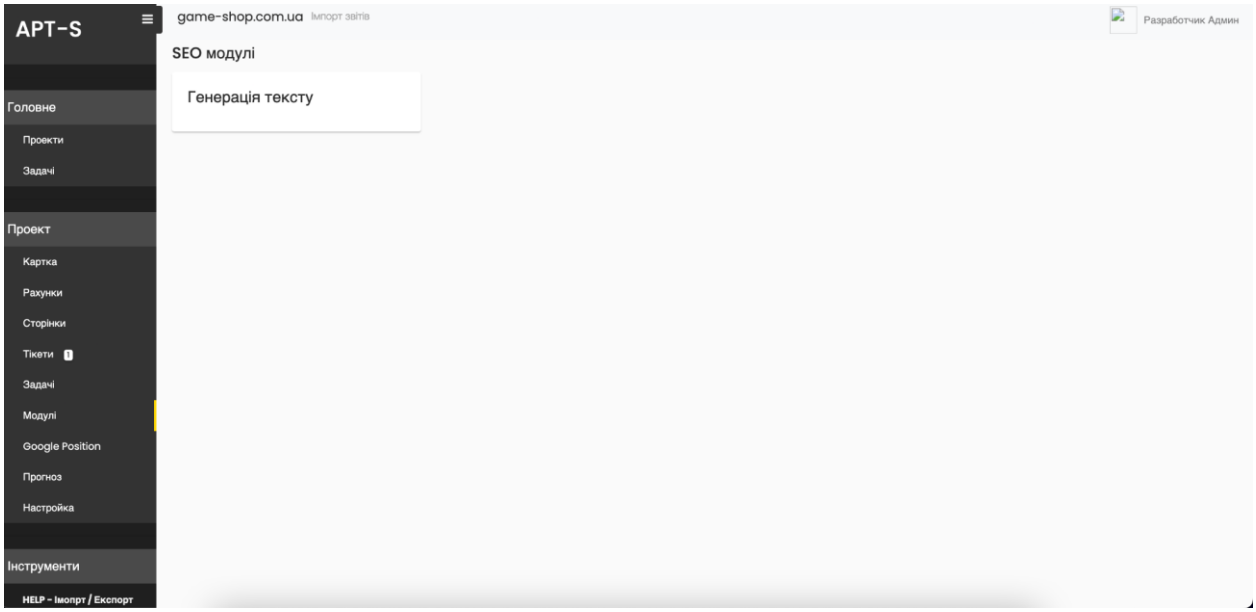

Рисунок 3.6.7 - Сторінка з переліком модулів проекту

8. Налаштування проекту - перелік усіх необхідних даних доступів до файлів проекту.

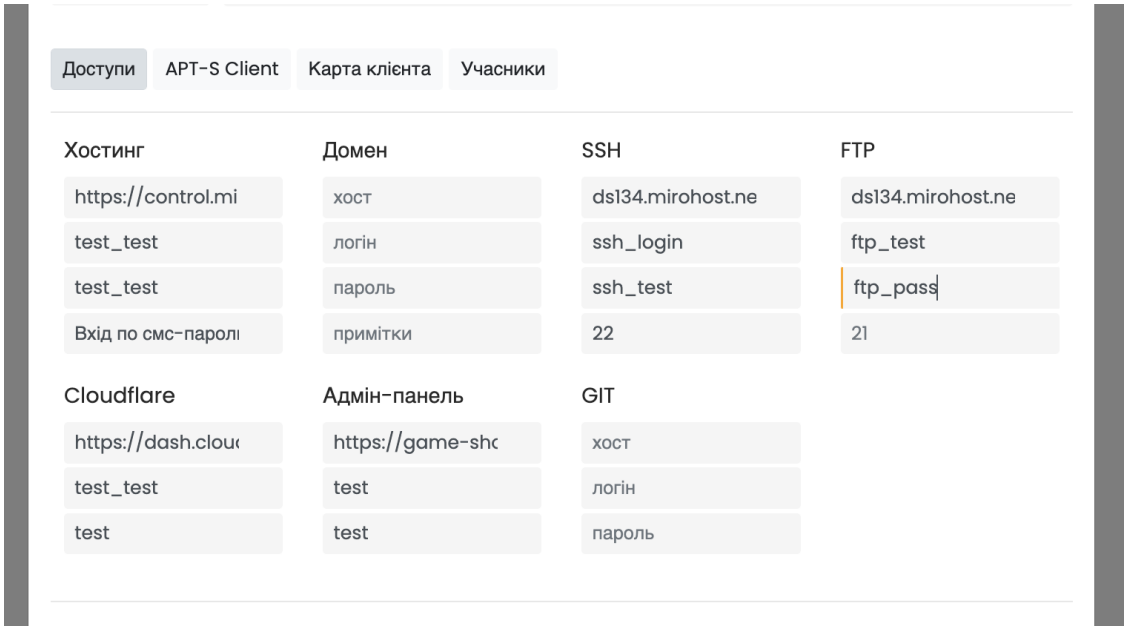

Рисунок 3.6.8 - Сторінка налаштувань проекту

Далі буде розглянуто представлення сторінки отримання позицій. На рисунку 3.7 наведено зображення інтерфейсу сторінки звіту отриманих позицій. Кольором виділимо три основні зони. Зеленим виділено заголовок сторінки, фіолетовим виділено графічне зображення отриманих позицій і оранжевим виділено таблицю значень, за якими будується графік.

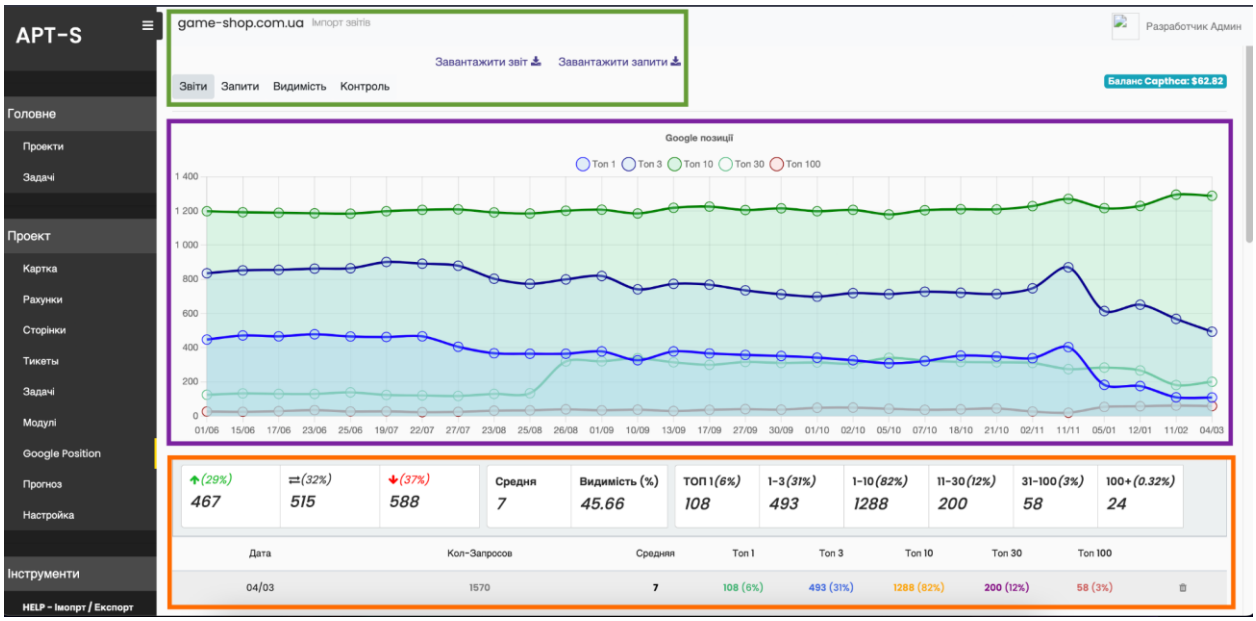

Рисунок 3.7 - Сторінка звітів позицій проекту

У заголовку сторінки, зображеного на рисунку 3.8, можна побачити наступний функціонал:

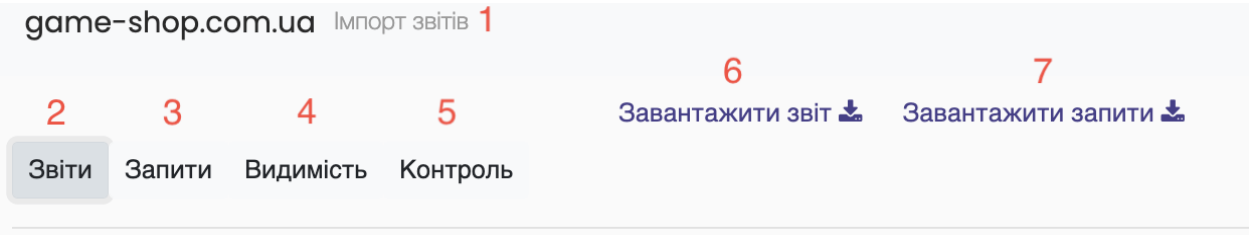

Рисунок 3.8 - Перелік функціоналу у заголовку сторінки позицій

1. Імпорт звітів - імпорт звітів позицій з інших ресурсів;
2. Вкладка “Звіти” - (поточна вкладка) поновлює вміст сторінки для виведення графічного зображення позицій запитів;
3. Вкладка “Запити” - оновлює вміст сторінки для виведення ключових запитів с СЯ;

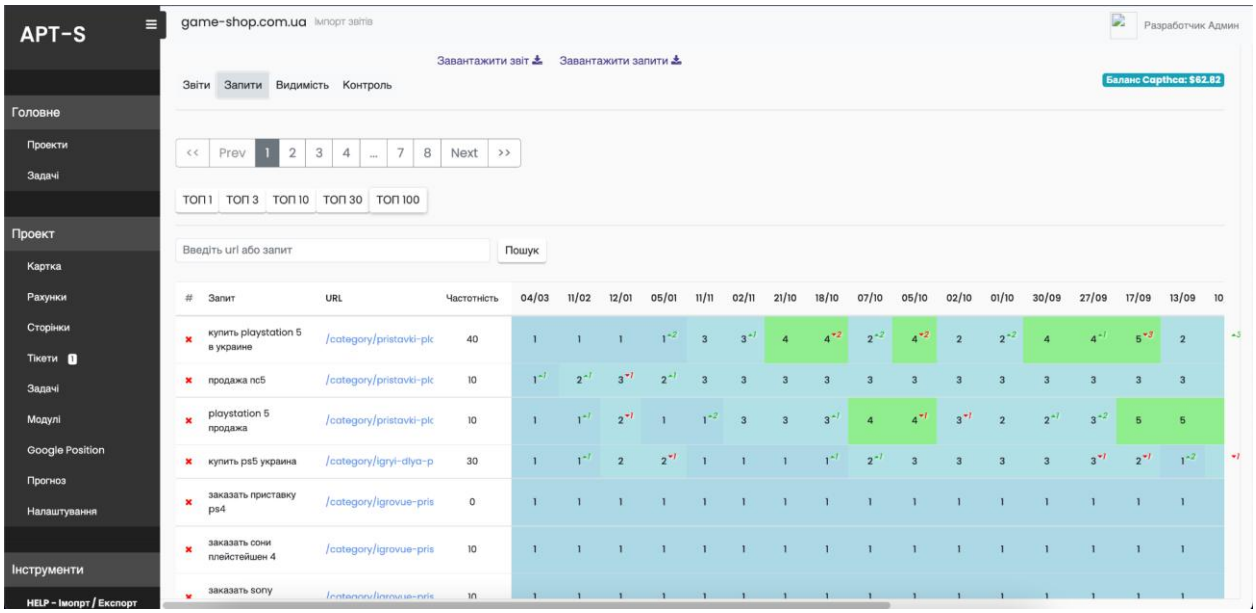

Рисунок 3.8.1 - Сторінка “Запити”

4. Вкладка “Видимість” - поновлює вміст сторінки для виведення графічного зображення критерію видимості;

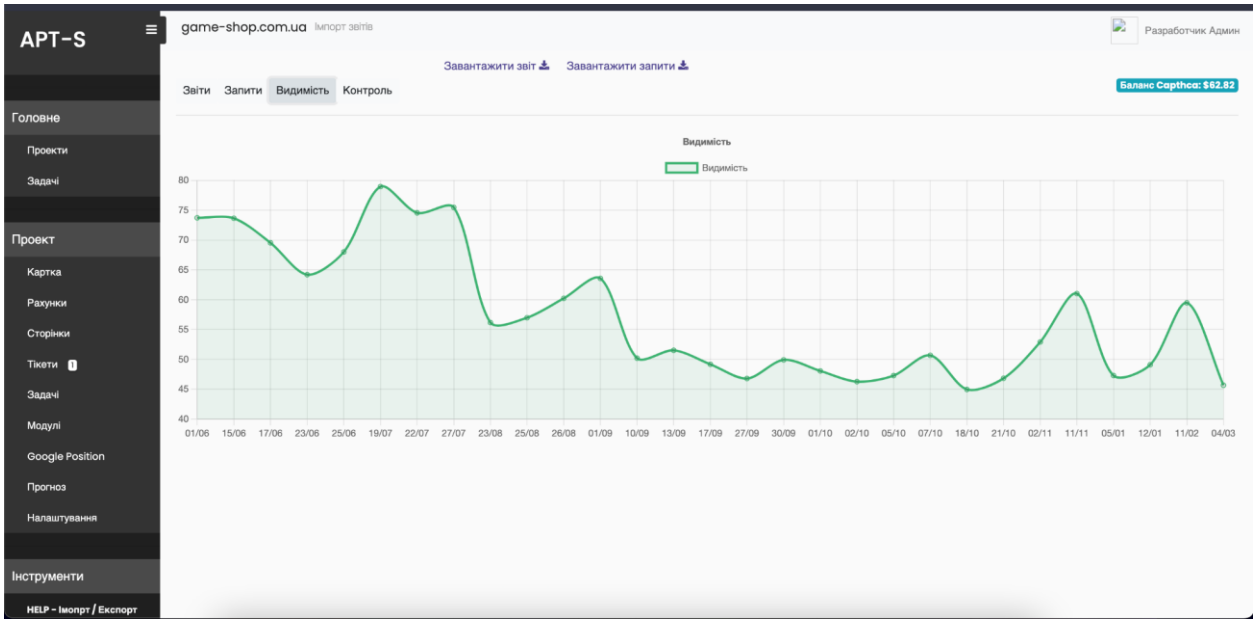

Рисунок 3.8.2 - Сторінка “Видимість”

5. Вкладка “Контроль” - поновлює вміст сторінки, дає доступ до функції отримання позицій;

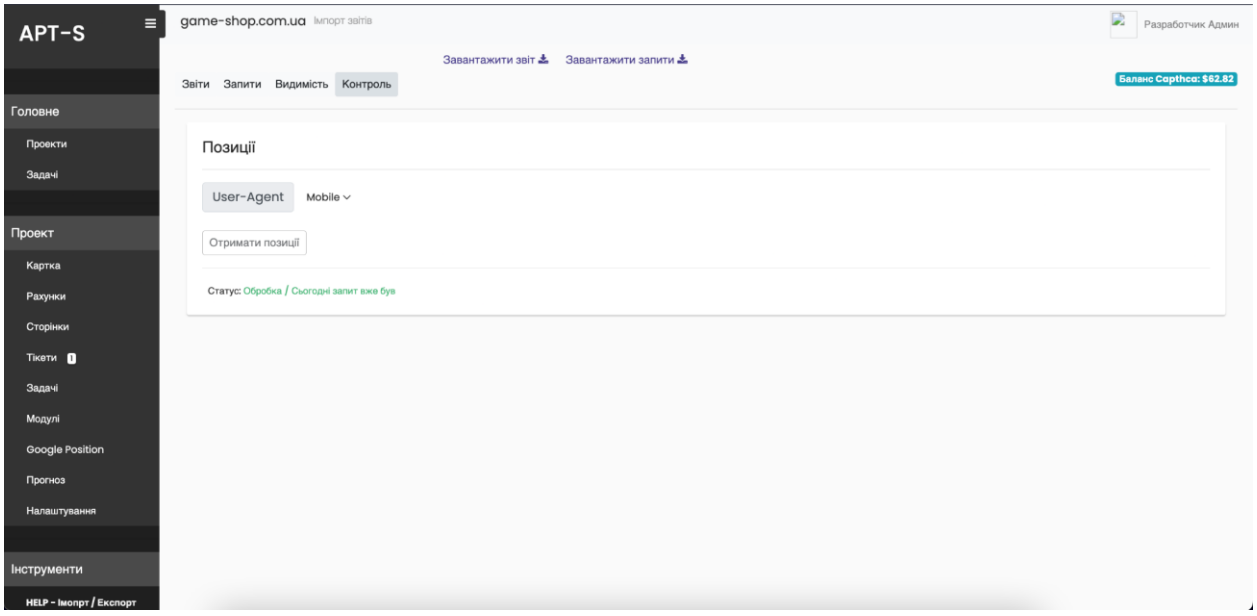

Рисунок 3.8.2 - Сторінка “Контроль”

6. Завантажити звіт - завантажити поточний звіт позицій у форматі .xlsx;

7. Завантажити запити - завантажити використані запити для отримання позицій у форматі .xlsx;

3.2 Опис процесу отримання позицій

Для початку процесу отримання позицій потрібно встановити наступне ПЗ:

- Chrome Driver;
- Sidekiq;
- Redis;

З міркувань зручності, вважаємо що проект вже розвернуто і він працює на локально. Розпочати процес варто з запуску файлу Chrome_Driver_executable:

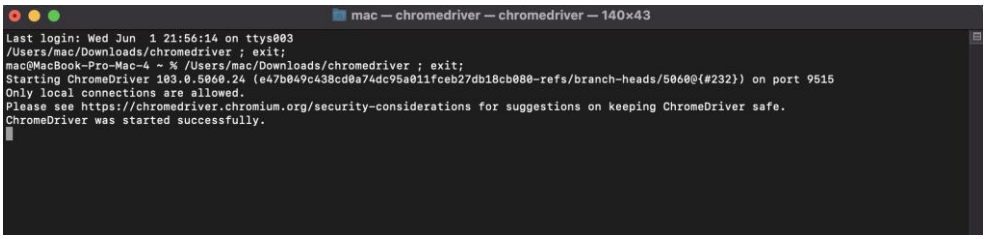

Рисунок 3.9 - Вікно терміналу після запуску Chrome_Driver_executable

Наступним кроком буде запуск Sidekiq командою у терміналі bundle exec sidekiq у корені проекту:

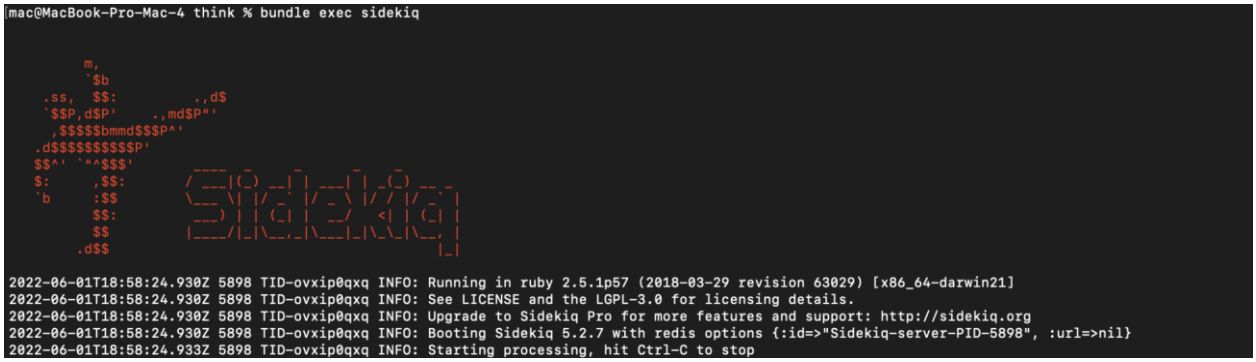

Рисунок 3.10 - Вікно терміналу після запуску Sidekiq

Після чого можна перейти до відповідної сторінки і розпочати процес отримання позицій. Відстежувати процес можна на сторінці Sidekiq, якщо перейти за посиланням на кшталт - HTTP_HOST/sidekiq/.

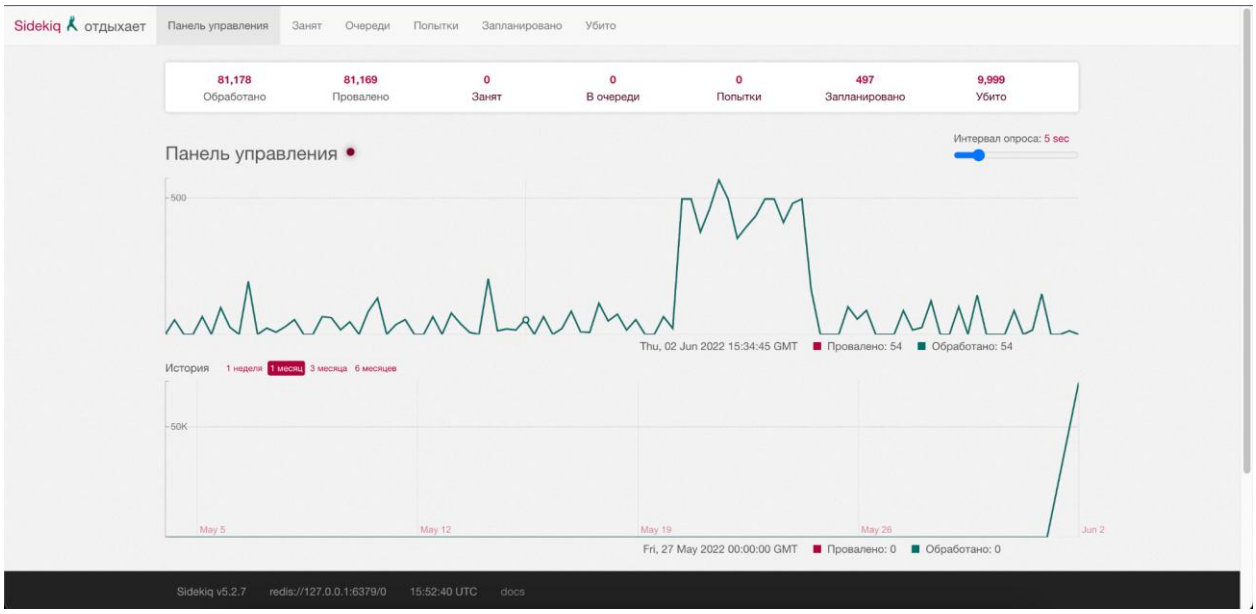

Рисунок 3.11 - Сторінка Sidekiq

Після завершення процесу отримання позицій (близько чотирьох годин), ми зможемо відстежити зміни на графічному зображенні.

3.3 Опис інструктивних матеріалів для користування та розробки

Для запуску программного модулю у локальному середовищі знадобиться наступне ПЗ:

- VSCode;
- Утиліта Homebrew;
- Ruby >= 3.5.1;
- Сумісна версія Ruby on Rails;
- Сумісна версія MySQL Gem;
- Bundler;
- Встановити Rake через Bundler;
- Встановити Foreman через Bundler;
- Встановити Yarn через Bundler.

Коли всі згадані кроки були виконані успішно, потрібно перейти до кореневого каталогу проекту, запустити в ній Terminal або CMD і ввести наступну команду:

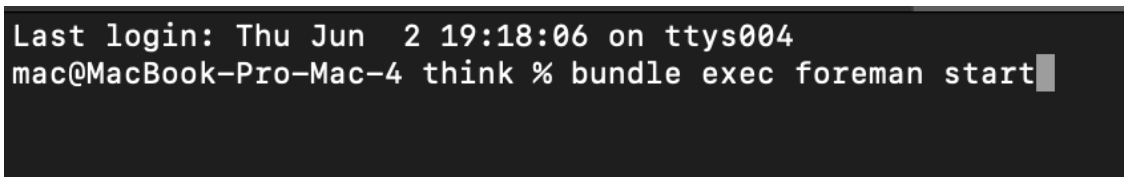

Рисунок 3.12 - Вікно Terminal за адресою кореневого каталога програмного модуля

Результатом виконання команди буде наступна відповідь вікна:

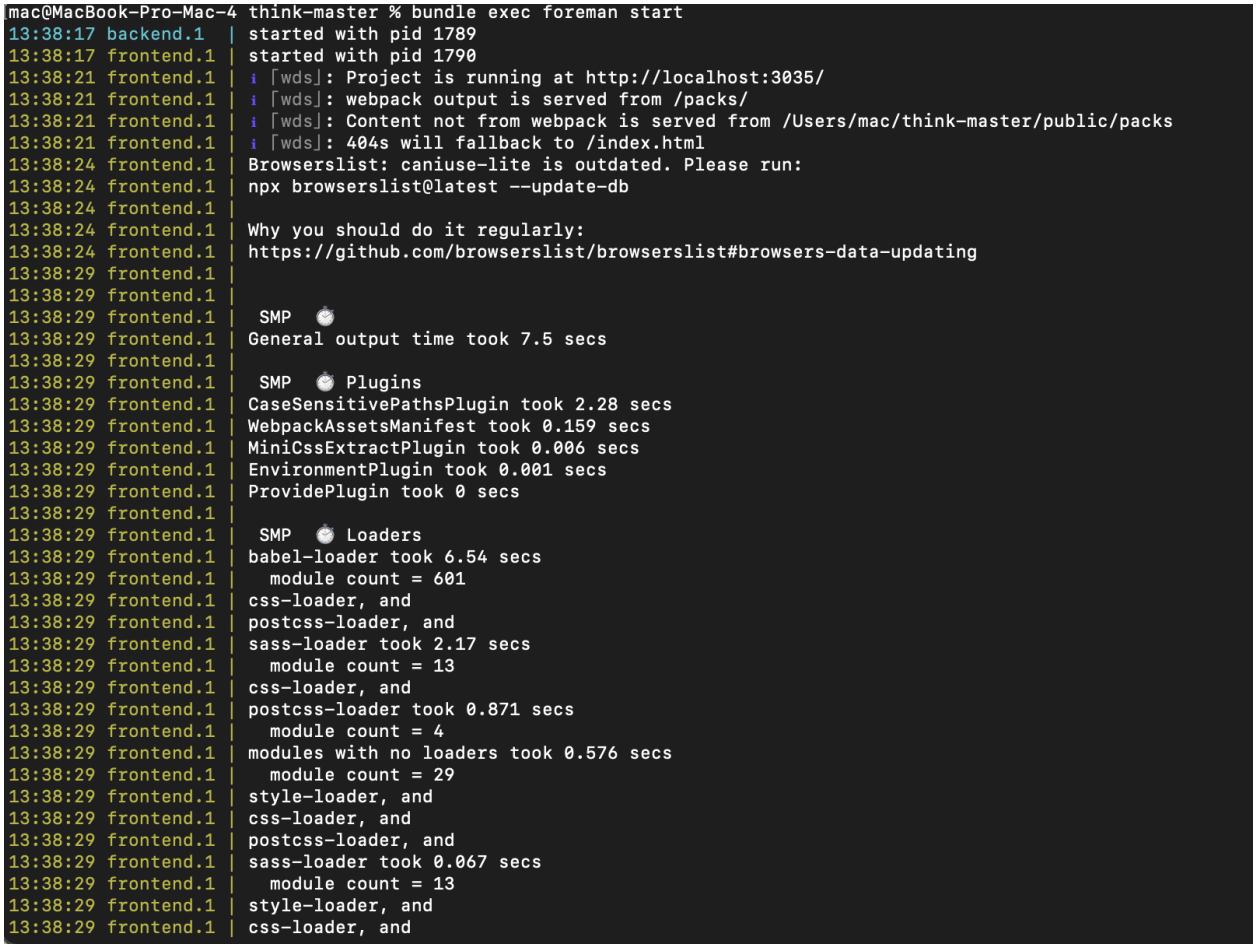

Рисунок 3.13 - Вікно Terminal після виконання команди bundle exec foreman start

На MacOS, проект буде розміщено на 3000 порт за замовчуванням. За адресою http://localhost:3000/ у вікні браузера ми можемо переглянути результат виконання команд.

ВИСНОВКИ

У результаті даної випускної кваліфікаційної роботи був отриманий програмний модуль отримання та прогнозування позицій web-сайтів у пошуковій системі Google. Була розглянута взаємодія мови програмування Ruby з бібліотекою React (мови програмування JS) та створено інтерфейс взаємодії з пошуковою видачею Google Search Engine.

З сильних сторін модуля можна виділити відсутність похибки при отриманні позицій, легкий для сприйняття інтерфейс, експорт звітів та можливість імпорту звітів з інших сервісів.

Варто зазнчити, що програмний модуль розроблявся з потенціалом до майбутнього удосконалення та розширення функціоналу. З ряду розширення функціоналу можна виділити завершення розробки фінансового модуля та модуля розкладу. Удосконалення передбачають можливі зміни та поліпшення користувацького інтерфейсу, швидкодії процесу отримання позицій та рефакторинг контролерів програмного модуля.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1) Integration Definition for Function Modeling (IDEF0). Software Standard, Modeling Techniques. – National Institute of Standards and Technology, 1993

2) The Use of IDEF0 for the Design and Specification of Methodologies [Електронний ресурс]. – Режим доступу: https://www.researchgate.net/publication/2447898_The_Use_of_IDEF0_for_the_Design_and_Specification_of_Methodologies

3) Automation of strategy using IDEF0 — A proof of concept [Електронний ресурс]. – Режим доступу: https://www.sciencedirect.com/science/article/pii/S2214716015000111

4) МЕТОДОЛОГИЯ IDEF0 [Електронний ресурс]. – Режим доступу: https://sites.google.com/site/anisimovkhv/learning/pris/lecture/tema6/tema6_2

5) MySQL® Notes for Professionals book:

   https://books.goalkicker.com/MySQLBook/

6) React JS Notes for Professionals book:

   https://books.goalkicker.com/ReactJSBook/

7) Ruby® Notes for Professionals book:

   https://books.goalkicker.com/RubyBook/

8) Ruby® on Rails Notes for Professionals book:

   https://books.goalkicker.com/RubyOnRailsBook/

9) Golyandina, N., V. Nekrutkin and A. Zhigljavsky (2001): Analysis of Time Series Structure: SSA and related techniques. Chapman and Hall/CRC. doi: 10.1201/9781420035841

https://books.goalkicker.com/GitBook/

10) JavaScript® Notes for Professionals book:

https://books.goalkicker.com/JavaScriptBook/

11) Шаміна І. Total SEO. Повне практичне керівництво по просуванню сайтів. / Шаміна І., Носаченко А. М: Інфра-Інженерія,2021. - 604 с. - ISBN 978-5-9729-0544-7

12) Ашманов І. Оптимізація і просування у пошукових системах / Ашманов І. М: Інфра-Інженерія,2019. - 512 с. - ISBN 978-5-4461-1161-9

Додаток А

```
class Api::V1::ReportsController < ApplicationController

  def get_reports

    @project = Project.find(params[:project_id])

    position = ProjectPosition.find_by(project_id: params[:project_id])

    proj_config = ProjectConfig.find_by(project_id: params[:project_id])


    position ||= self.create_project_position_entry(params[:project_id])


    @pages_reports_position = position.position

    @average = position.average

    @visibility = position.visibility

    @average = eval(@average)

    @visibility = eval(@visibility)

    @pages_reports_position = eval(@pages_reports_position)


    if proj_config.pagination_count.nil?

      ReportCacheCreateWorker.perform_async(params[:project_id], 1)

      @pages = @project.pages.all.page_num(1).per(5).eager_load(:semantic_report).where("`semantic_reports`.`report` is not null")

      total_pages = @pages.total_pages

    else

      total_pages = proj_config.pagination_count

    end


    client = current_user.has_role? :client


    balance_captcha = Rails.cache.fetch("balance_cache", :expires_in => 2.hours) do

      res = HTTParty.get("https://api.anti-captcha.com/getBalance",

        :body => {

          :clientKey => '89b7b7aea8532a2ec7b2a6f1a89c98bc',

          }.to_json,

        :headers => {'Content-Type' => 'application/json'}).to_h
```

```
      res["balance"]
    end

    report_changes = {
      up: position.up,
      stay: position.stay,
      down: position.down
    }

    render json: {
      pagesReportsPosition: @pages_reports_position,
      average: @average,
      visibility: @visibility,
      report_changes: report_changes,
      totalPages: total_pages,
      balance: balance_captcha,
      isClient: client
    }
  end

  def get_forecast
    @project = Project.find(params[:project_id])
    position = ProjectPosition.find_by(project_id: params[:project_id])
    proj_config = ProjectConfig.find_by(project_id: params[:project_id])

    position ||= self.create_project_position_entry(params[:project_id])

    @pages_reports_position = position.position
    @average = position.average
    @visibility = position.visibility
    @average = eval(@average)
    @visibility = eval(@visibility)
    @pages_reports_position = eval(@pages_reports_position)
```

```
    forecast = calculate_forecast(@pages_reports_position)
    client = current_user.has_role? :client

    report_changes = {
        up: position.up,
        stay: position.stay,
        down: position.down
    }

    render json: {
        pagesReportsPosition: @pages_reports_position,
        average: @average,
        visibility: @visibility,
        report_changes: report_changes,
        isClient: client
    }
end
#TODO - get this method done
def calculate_forecast (reports)
    reports.each do |report|
        p report
    end
end



def reverse_position
    # ProjectUnlockGooglePositionWorker.perform_async(params[:project_id], DateTime.now)
    # position = ProjectPosition.find_by(project_id: params[:project_id])
    # av = eval(position.average).to_a.reverse.to_h
    # vis = eval(position.visibility).to_a.reverse.to_h
    # pos = eval(position.position).to_a.reverse.to_h
    # position.update_attributes(:position => pos, :average => av, :visibility => vis)
    # report_changes = ReportChanges::CountChanges.new(params[:project_id])
```

```
    # report_changes.count_reports_number_changes


    cache_report = Cache::Report.new(params[:project_id])
    cache_report.create_cache_of_report


    render json: {
      isDone: true
    }
end
##########################################################
def pages_reports
    page_num = params[:page_num].to_i
    reports = CacheReport.find_by(project_id: params[:project_id])
    reports = JSON::parse(reports.last_report)


    start = page_num * 200 - 200
    end_arr = start + 200
    reports = reports[start..end_arr]


    render json: reports
end


##########################################################
# return sorted pages reports with freq                  #
##########################################################
def pages_reports_freq
    project_id = params[:project_id].to_i
    @reports = CacheReport.find_by(project_id: project_id)
    @reports = JSON::parse(@reports.last_report)


    start = params[:page_num].to_i * 200 - 200
    end_arr = start + 200
```

```
    status = params[:status].to_i

    if status == 0
        @reports.sort_by! { |element| -element["freq"] }
    else
        @reports.sort_by! { |element| element["freq"] }
    end

    reports = @reports[start..end_arr]

    render json: reports
end
#########################################################
# return pages reports with top                         #
#########################################################
def pages_reports_position
    pos = params[:position].to_i
    project_id = params[:project_id]

    conf = ProjectConfig.find_by(project_id: project_id)
    last_date = conf.g_pos_last_date

    @reports = CacheReport.find_by(project_id: project_id)

    if @reports.nil?
        ReportCacheCreateWorker.perform_async(params[:project_id], 0)
        sleep(10)
        @reports = CacheReport.find_by(project_id: project_id)
    end

    @reports = JSON::parse(@reports.last_report)

    case pos
```

```
        when 1
            reports = @reports.select { |element| element["top1"] == 1 }
        when 3
            reports = @reports.select { |element| element["top3"] == 1 }
        when 10
            reports = @reports.select { |element| element["top10"] == 1 }
        when 30
            reports = @reports.select { |element| element["top30"] == 1 }
        when 100
            reports = @reports.select { |element| element["top100"] == 1 }
    end

    render json: reports
end
############################################################

def get_position_status
    project = Project.find(params[:project_id])
    status = project.project_config.g_pos_status

    render json: status
end

def get_searched_pages
    @reports = CacheReport.find_by(project_id: params[:project_id])
    @reports = JSON::parse(@reports.last_report)

    string = params[:word]
    key = params[:searchFor]

    # reports = @reports.select { |element| element["#{search_for}"].include? string }
    reports = @reports.select do |element|
        curr_elem = element["#{key}"]
```

```
        if curr_elem.nil?
          return false
        end

        curr_elem.include?(string)
      end

      render json: reports
    end

    # артефакты
    # sem_words => {word: "example", freq: "100"}
    # sem_words => {"example", {word: "example", freq: "100"}}
    def start
      user_agent = params[:user_agent]
      project = Project.find(params[:project_id])
      if !project.project_config.g_pos_status
        project.project_config.update_attributes(g_pos_status: true)

        month = Date.today.strftime("%m").to_i
        day = Date.today.strftime("%d").to_i
        year = Date.today.strftime("%Y").to_i

        result = project.pages.collect(&:semantic).flatten

        result.each_with_index do |sem, idx|
          unless sem.nil?
            PageScanPositionWorker.perform_async(project.protocol+project.domain, sem.page_id, sem.words, user_agent, month, day,
year)
          end
        end

        ProjectUnlockGooglePositionWorker.perform_at(4.hours.from_now, project.id, DateTime.now)
```

```
        # WARNING, BAD PART OF CODE

        begin

          notice = Notice.new(content: "Позиції будуть отримані", domain: project.domain, status: 5, link:
"/projects/#{params[:project_id]}/reports")

          users = []

          users << User.find(53)

          notice.users = users

          notice.save

        rescue => exception

          p exception

        end



        render json: {

          status: 200,

          message: 'Усі сторінки проекту додані до черги зняття позицій'

        }

      else

        render json: {

          status: 220,

          message: 'Обробка...'

        }

      end

    end



    # need to refactor

    def delete

      result = Hash.new { |hash, key| hash[key] = [] }

      key = params[:report_date]

      dats = key.split('/')

      p_day = dats[0].to_i

      p_mounth = dats[1].to_i



      project = Project.find(params[:project_id])
```

```
pages = project.pages.all.eager_load(:semantic_report)


pages.each do |page|
  unless page.semantic_report.nil?
    new_page_mounths = Hash.new { |hash, key| hash[key] = [] }


    page.semantic_report.report.each do |year, month_values|
      month_values.each do |month, vals|
        new_page_days = Hash.new { |hash, key| hash[key] = [] }
        _mounth = month.to_i


        begin
          vals.each do |day, valz|
            _day = day.to_i
            if _day != p_day
              new_page_days[_day] = valz
            end
          end
        rescue Exception => e
          result = {
            status: 220,
            message: "(1) Что-то пошло не так"
          }
        end


        begin
          if not new_page_days.empty?
            new_page_mounths[_mounth] = new_page_days
          end
        rescue Exception => e
          result = {
            status: 220,
            message: "(2) Что-то пошло не так"
```

```
                }
            end
        end

        begin
            page.semantic_report.update!({report: new_page_mounths })
        rescue Exception => e
            result = {
                status: 220,
                message: "(3) Что-то пошло не так"
            }
        end
    end
  end
end

proj_pos = ProjectPosition.find_by(project_id: params[:project_id])
pos = eval(proj_pos.position)
av = eval(proj_pos.average)
vis = eval(proj_pos.visibility)

pos = pos.except!(key)
av = av.except!(key)
vis = vis.except!(key)

proj_pos.update_attributes(:position => pos, :average => av, :visibility => vis)

if result.empty?
  result = {
    status: 200,
    message: "Отчет #{params[:report_date]} успешно удален!"
  }
end
```

```
    render json: result
end

def create_project_position_entry(project_id)
    @project = Project.find(project_id)
    @pages = @project.pages.all.eager_load(:semantic_report, :semantic)
    @pages_reports = Hash.new { |hash, key| hash[key] = [] }
    @pages_reports_position = Hash.new { |hash, key| hash[key] = {
        count_keys: 0,
        top1: 0,
        top3: 0,
        top10: 0,
        top30: 0,
        top100: 0
    }}
    @pages.each do |page|
        reports = Hash.new { |hash, key| hash[key] = [] }

        unless page.semantic_report.nil?
            semantic_check = true

            page.semantic_report.report.each do |year, month_values|
                month_values.each do |month, values|
                    values.each do |day, words|

                        if month.to_i < 10 and !month.to_s.index("0")
                            month = '0'+month.to_s
                        end

                        if day.to_i < 10 and !day.to_s.index("0")
                            day = '0'+day.to_s
                        end
```

```
      unless page.semantic.nil? and semantic_check
        semantic_check = false
      end

      words.each do |word, vals|
        unless vals.nil?
          sumfreq = 0
          unless page.semantic.nil?
            page.semantic.words.each do |word_sem|
              if word_sem.is_a?(Hash) and word_sem.key?(:freq) and word_sem[:freq] != '-'
                if word == word_sem[:word]
                  sumfreq += word_sem[:freq]
                end
              end
            end
          end

          pos_z = vals[:position]
          if pos_z == 0
            pos_z = "-"
          end

          reports[word] << {
            date: day.to_s+'/'+month.to_s,
            pos: pos_z,
            url: vals[:url],
            freq: sumfreq
          }
        end
      end
    end
end
```

```
        end

        unless reports.nil? or reports.empty?
            @pages_reports[page.id] = reports
        end
    end
end

@pages.each do |page|
    unless page.semantic_report.nil?
        semantic_check = true

        page.semantic_report.report.each do |year, month_values|
            month_values.each do |month, values|
                values.each do |day, words|

                    if month.to_i < 10 and !month.to_s.index("0")
                        month = '0'+month.to_s
                    end

                    if day.to_i < 10 and !day.to_s.index("0")
                        day = '0'+day.to_s
                    end

                    unless page.semantic.nil?
                        @pages_reports_position[day.to_s+'/'+month.to_s][:count_keys] += page.semantic.words.count
                        semantic_check = false
                    end

                    words.each do |word, vals|
                        unless vals.nil?
                            posit = vals[:position].to_i
                            if posit != 0
```

```
                    if posit == 1
                      @pages_reports_position[day.to_s+'/'+month.to_s][:top1] += 1
                    end
                    if posit <= 3
                      @pages_reports_position[day.to_s+'/'+month.to_s][:top3] += 1
                    end
                    if posit <= 10
                      @pages_reports_position[day.to_s+'/'+month.to_s][:top10] += 1
                    end
                    if posit > 10 and posit <= 30
                      @pages_reports_position[day.to_s+'/'+month.to_s][:top30] += 1
                    end
                    if posit > 30 and posit <= 100
                      @pages_reports_position[day.to_s+'/'+month.to_s][:top100] += 1
                    end
                  end
                end
              end
            end
          end
        end
      end
    end
  end

  average = self.calculate_average(@pages_reports)
  vision = self.calculate_visibility(@pages_reports)

  position = ProjectPosition.new
  position.average = average
  position.visibility = vision
  position.position = @pages_reports_position
  position.project_id = @project.id
  position.save
```

```
    return position
end

def calculate_visibility(pages_reports)
    visibility = Hash.new { |hash, key| hash[key] = [] }
    numbers = Hash.new { |hash, key| hash[key] = {
        up: 0,
        down: 0
    }}

    pages_reports.each do |key, report|
        report.each do |k, inside|
            inside.each do |req|
                case req[:pos]
                when 1..3
                    numbers[req[:date]][:up] += req[:freq]
                when 4
                    numbers[req[:date]][:up] += req[:freq] * 0.85
                when 5
                    numbers[req[:date]][:up] += req[:freq] * 0.6
                when 6..7
                    numbers[req[:date]][:up] += req[:freq] * 0.5
                when 8..9
                    numbers[req[:date]][:up] += req[:freq] * 0.3
                when 10
                    numbers[req[:date]][:up] += req[:freq] * 0.2
                else
                end

                numbers[req[:date]][:down] += req[:freq]
            end
        end
```

```
    end

    numbers.each do |key, value|
        val = (value[:up] / value[:down]) * 100

        if val > 0
            visibility[key] = val
        else
            visibility[key] = 0
        end
    end

    return visibility
end

def calculate_average(reports)
    all_pos = Hash.new { |hash, key| hash[key] = [] }
    dates = Hash.new { |hash, key| hash[key] = [] }
    average = 0

    reports.each do |key, report|
        report.each do |k, inside|
            inside.each do |req|
                if req[:pos] != "-" and not req[:pos].nil?
                    if all_pos.key?(req[:date])
                        all_pos[req[:date]] += req[:pos].to_i
                    else
                        all_pos[req[:date]] = req[:pos].to_i
                    end

                    if dates.key?(req[:date])
                        dates[req[:date]] += 1
                    else
```

```
                    dates[req[:date]] = 1
                end
            end
        end
    end
  end

  all_pos.each do |key, value|
    all_pos[key] = value/dates[key]
  end
  return all_pos
 End
end
```

Додаток Б

```
class PageScanPositionWorker
  include Sidekiq::Worker
  sidekiq_options :queue => :scan_position_google, :retry => 0


  def perform(protocol_domain, page_id, words, user_agent, mounth, day, year)
    browser = create_browser(user_agent)
    browser_pid = nil
    if browser
      browser_pid = browser.driver.instance_variable_get(:@service).instance_variable_get(:@process).instance_variable_get(:@pid)
      logger.info("Browser PID / #{browser_pid}")
      semantic_report = Hash.new { |hash, key| hash[key] = [] }
      count_pages = 'num=100'


      bad_words = []
      words.each do |element|
        word = nil
        if element.is_a? Hash and element.key?("word")
          word = element["word"].to_s
        else
          word = element.to_s
        end


        unless word.nil?
          begin
            browser.goto('https://www.google.com/search?q='+word+'&'+count_pages)
            logger.info("NOT-ERROR / goes to page")
          rescue Exception => e
            logger.info("ERROR-1 / #{e.message}")
            browser.quit
            PageScanPositionWorker.perform_in(1.minutes, protocol_domain, page_id, words, user_agent, mounth, day, year)
            return true
          end
```

```
doc = Nokogiri::HTML(browser.html)

logger.info("NOT-ERROR / html is here")

begin
  if browser.url.index("sorry/index?continue")
    doc_captcha = Nokogiri::HTML browser.html

    doc_captcha.search('//div[@class="g-recaptcha"]').each do |node|
      node_hash = node.to_h
      data_site_key = node_hash["data-sitekey"]
      data_s = node_hash["data-s"]

      task_id = self.create_task(data_site_key, data_s, browser.url)
      task_status = true
      task_count = 0

      if task_id
        while task_status and task_count < 15
          sleep(4)

          begin
            res_task = get_task(task_id)
          rescue Exception => e
            logger.info("TASK ERROR-10 / #{e.message}")
          end

          if res_task[:status] == 'ready'
            browser.execute_script("document.getElementById('g-recaptcha-response').style.display = 'block';")
            browser.textarea(id: "g-recaptcha-response").set res_task[:g_recaptcha_response]
            browser.execute_script("document.getElementById('g-recaptcha-response').style.display = 'none';")
            sleep(1)
```

```
                browser.form(id: "captcha-form").submit
                task_status = false

                if !browser.url.index("sorry/index")
                  doc = Nokogiri::HTML browser.html
                  res = self.parse_position(protocol_domain, page_id, doc)

                  if res[:position] == false
                    bad_words << word
                  else
                    semantic_report[word] = res
                  end
                else
                  logger.info("ERROR-2 / sorry/index")
                  browser.quit
                  PageScanPositionWorker.perform_in(2.minutes, protocol_domain, page_id, words, user_agent, mounth, day,
year)

                  # unless bad_words.empty?
                  #   PageScanPositionWorker.perform_in(1.minutes, protocol_domain, page_id, bad_words, user_agent,
mounth, day, year)
                  # end

                  # if !semantic_report.empty?
                  #   sem = []
                  #   words.each do |element|
                  #     if element.is_a? Hash and element.key?("word")
                  #       sem << element["word"].to_s
                  #     else
                  #       sem << element.to_s
                  #     end
                  #   end

                  #   keys = semantic_report.keys
```

```
                        #   back_words = sem.select {|elem| not keys.include?(elem)}
                        #   back_words = back_words.select {|elem| not bad_words.include?(elem)}

                        #   unless back_words.empty?
                        #       PageScanPositionWorker.perform_in(2.minutes, protocol_domain, page_id, back_words, user_agent,
mounth, day, year)
                        #   end

                        #   self.set_position(semantic_report, page_id, mounth, day, year)
                        # else
                        #   PageScanPositionWorker.perform_in(2.minutes, protocol_domain, page_id, words, user_agent, mounth,
day, year)
                        # end

                        return true
                      end
                    else
                      if task_count < 15
                        task_count += 1
                      end
                    end
                  end
                end
              end
            else
              begin
                res = self.parse_position(protocol_domain, page_id, doc)
              rescue Exception => e
                logger.info("PARSE ERROR-4 / #{e.message}")
              end

              if res[:position] == false
                bad_words << word
              else
```

```
                    semantic_report[word] = res
                end
            end
        rescue Exception => e
            logger.info("ERROR-3 / #{e.message}")
            browser.quit
            return true
        end
    end
  end

  unless bad_words.empty?
    PageScanPositionWorker.perform_in(1.minutes, protocol_domain, page_id, bad_words, user_agent, mounth, day, year)
  end
else
  PageScanPositionWorker.perform_in(1.minutes, protocol_domain, page_id, words, user_agent, mounth, day, year)
  browser.quit
  return true
end

browser.quit

begin
  # проверка, чтобы ключи не выпадали
  sem = []
  words.each do |element|
    if element.is_a? Hash and element.key?("word")
      sem << element["word"].to_s
    else
      sem << element.to_s
    end
  end
```

```
        keys = semantic_report.keys

        back_words = sem.select {|elem| not keys.include?(elem)}

        back_words = back_words.select {|elem| not bad_words.include?(elem)}


        unless back_words.empty?

          PageScanPositionWorker.perform_in(2.minutes, protocol_domain, page_id, back_words, user_agent, mounth, day, year)

        end


        self.set_position(semantic_report, page_id, mounth, day, year)

      rescue Exception => e

        logger.info("SETPOS ERROR-5 / #{e.message}")

      end


      return true

    end


    def create_browser(user_agent)

      browser = nil

      if user_agent == 'mobile'

        user_agent = 'Mozilla/5.0 (iPhone; CPU iPhone OS 13_2_3 like Mac OS X) AppleWebKit/605.1.15 (KHTML, like Gecko)
Version/13.0.3 Mobile/15E148 Safari/604.1'

      else

        user_agent = 'Mozilla/5.0 (Macintosh; Intel Mac OS X 11_1_0) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/88.0.4324.192
Safari/537.36'

      end


      begin

        Watir.default_timeout = 130


        options = Selenium::WebDriver::Chrome::Options.new

        options.add_argument('--headless')

        options.add_argument('--no-sandbox')

        options.add_argument('--disable-gpu')

        options.add_argument('--lang=ru')
```

```
        options.add_argument('--user-agent=' + user_agent)

        driver = Selenium::WebDriver.for :chrome, options: options
        browser = Watir::Browser.new driver
    rescue Exception => e
        logger.info("ERROR-4 / #{e.message}")
        browser = nil
    end

    return browser
end

def create_task(data_site_key, data_s, captcha_url)
    task_id = nil

    logger.info("ERROR ERROR 123 // #{data_site_key} gogogo")

    response = HTTParty.post("https://api.anti-captcha.com/createTask",
        :body => {
            :clientKey => '89b7b7aea8532a2ec7b2a6f1a89c98bc',
            :task => {
                type: "NoCaptchaTaskProxyless",
                    websiteURL: captcha_url,
                websiteKey: data_site_key,
                recaptchaDataSValue: data_s
            }
        }.to_json,
        :headers => {'Content-Type' => 'application/json'})

    response_hash = response.to_h
    if response_hash["errorId"] == 0
        task_id = response_hash["taskId"]
    end
```

```
    return task_id
end

def get_task(task_id)
    res_result = Hash.new
    g_recaptcha_response = nil
    g_cookies = nil

    response = HTTParty.post("https://api.anti-captcha.com/getTaskResult",
        :body => {
            :clientKey => '89b7b7aea8532a2ec7b2a6f1a89c98bc',
            :taskId => task_id
        }.to_json,
        :headers => {'Content-Type' => 'application/json'}).to_h

    case response["status"]
    when 'ready'
        res_result = {
            status: 'ready',
            g_cookies: response["solution"]["cookies"],
            g_recaptcha_response: response["solution"]["gRecaptchaResponse"]
        }

    when 'processing'
        res_result = { status: 'processing' }
    else
        res_result = { status: response["status"] }
    end

    return res_result
end
```

```
def parse_position(protocol_domain, page_id, doc)
    res = nil
    position = 1
    ggt = false
    last_link = ''

    # yuRubf
    doc.search("div[class='mnr-c xpd O9g5cc uUPGi']").map do |elem|
        ahref = elem.search("a").first
        ahref = ahref.to_h

        if ahref.key?('href') and last_link != ahref['href']
            if ahref['href'].index(protocol_domain)
                res = {
                    position: position.to_i,
                    url: ahref['href']
                }

                return res
            end

            position += 1
        end

        last_link = ahref['href']
        ggt = true
    end

    if ggt == true
            res = {
                position: 0,
                url: "-"
            }
```

```
      else
        res = {
          position: false,
          url: "-"
        }
      end

      return res
    end

    def set_position(data, page_id, month, day, year)
      ex_data = Hash.new { |hash, key| hash[key] = [] }
      ex_data_month = Hash.new { |hash, key| hash[key] = [] }

      ex_data_month[month] = { day => data }
      ex_data[year] = ex_data_month

      semantic_report = SemanticReport.find_by(page_id: page_id)
      if !semantic_report
        SemanticReport.create({report: ex_data, page_id: page_id})
      else
        report = semantic_report.report
        report.deep_merge!(ex_data)
        semantic_report.update!({report: report })
        # unset
        report = nil
        ex_data = nil
        ex_data_month = nil
      end
    end
  end
```