Київський національний університет імені Тараса Шевченка
Факультет інформаційних технологій
Кафедра програмних систем і технологій
Освітньо-кваліфікаційний рівень бакалавр
Спеціальність 121 “Інженерія програмного забезпечення”

ЗАТВЕРДЖЕНО
Зав. кафедри програмних систем і технологій
_____ (Олексій БИЧКОВ)

ЗАВДАННЯ
НА ВИПУСКНУ КВАЛІФІКАЦІЙНУ БАКАЛАВРСЬКУ РОБОТУ
СТУДЕНТУ

Гоголаурі Ростиславу Олександровичу

(прізвище, ім’я, по батькові)

1. **Тема випускної кваліфікаційної бакалаврської роботи** **“**Створення парсеру об'єктів нерухомості та побудова фільтрації”
затверджена наказом вищого навчального закладу від „__”_____20__р. №_____
2. **Строк здачі студентом закінченої роботи** „__” _____ 2021 р.
3. **Вихідні дані до роботи** підручники, навчальні посібники, статті, Інтернет-ресурси
4. **Зміст пояснювальної записки (перелік питань, що їх належить розробити)**

Аналітична частина:

- Обґрунтування актуальності та мети розробки парсерів для об’єктів нерухомості та систем фільрації
- Опис предметного середовища та огляд наявних аналогів
- Постановка задачі фільтрації

Практична частина

− визначити особливості архітектурного рішення_____

− спроектувати структуру програмної системи_____

− розробити програмне забезпечення системи_____

− проаналізувати результати розробленої системи_____.

5. Перелік графічного матеріалу (з точним забезпеченням обов'язкових креслень)

1. Схема структурна функціональної моделі системи
2. Схема структурна діяльності
3. Схема бази даних
4. Схема структурна класів програмного забезпечення
5. Діаграма послідовності
6. Креслення вигляду екранних форм

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

Розділ	Консультант	Підпис, дата	
		Завдання видав	Завдання прийняв
Розділ 1 Огляд існуючих рішень	Тетяна КОВАЛЮК		
Розділ 2 Проектування та розробка системи	Тетяна КОВАЛЮК		
Розділ 3 Результат роботи програми	Тетяна КОВАЛЮК		

7. Дата видачі завдання _____

Керівник _____ /Тетяна КОВАЛЮК/

Завдання прийняв до виконання _____/Ростислав ГОГОЛАУРІ/

КАЛЕНДАРНИЙ ПЛАН

№ п/п	Назви етапів бакалаврської роботи	Термін виконання етапів роботи	Відмітка про виконання
1.	Уточнення постановки задачі		Виконано
2.	Аналіз літератури		Виконано
3.	Аналіз існуючих систем парсингу даних		Виконано
4.	Обґрунтування вибору рішення		Виконано
5.	Опис архітектури		Виконано
6.	Побудова архітектури		Виконано
7.	Розроблення програмного забезпечення		Виконано
8.	Тестування розробленого програмного забезпечення		Виконано
9.	Оформлення і друк пояснювальної записки		Виконано
10.	Оформлення презентації		Виконано
11.	Отримання рецензії		Виконано
12.	Затвердження пояснювальної записки роботи завідувачем кафедри		Виконано
13.	Захист дипломної роботи		Виконано

Студент – бакалавр _____/Ростислав ГОГОЛАУРІ/

Керівник роботи _____/Тетяна КОВАЛЮК/

АНОТАЦІЯ

Випускна кваліфікаційна бакалаврська робота: 43 с., 13 рис., 1 додаток, 6 джерел.

Тема: Створення парсеру об'єктів нерухомості та побудова фільтрації.

Об’єкт дослідження: технологія парсингу даних.

Предмет дослідження: платформа для оренди нерухомості.

Мета роботи: розробка платформи для оренди нерухомості з парсером об’яв зі сторонніх джерел.

Результати дослідження: в результаті дослідження було оцінено можливість використання технології для реалізації швидкого парсеру для обробки великої кількості даних.

Висновок: В результаті виконання роботи було розроблено сервіс для оренди нерухомості зі сбором великої кількості об’яв з більш ніж 60 веб-сайтів.. Розроблений веб-додаток представляє з себе сукупність:

- Парсеру даних з 60 веб ресурсів;
- Модулю перевірки на легитимність;
- Клієнтську частину веб-додатку для взаємодії з системою;
- Датабази MongoDB та пов’язану з системою IPFS на блокчейні Ethereum та Binance Smart Chain;

АННОТАЦИЯ

Выпускная квалификационная бакалаврская работа: 43 с., 13 рис., 1 приложение, 6 источников.

Тема: Создание парсера объектов недвижимости и построение фильтрации.

Объект исследования: технология парсинга данных.

Предмет исследования: платформа для аренды недвижимости.

Цель работы: разработка платформы для аренды недвижимости с парсером объявлений из посторонних источников.

Результаты исследования: в результате исследования было оценено возможность использования технологии для реализации быстрого парсеру для обработки большого количества данных.

Вывод: В результате выполнения работы был разработан сервис для аренды недвижимости со сбором большого количества объявлений с более чем 60 сайтов. Разработан веб-приложение представляет из себя совокупность:

- Парсера данных с 60 веб ресурсов;
- Модулю проверки на легитимность;
- Клиентскую часть веб-приложения для взаимодействия с системой;
- Датабазы MongoDB и связанную с системой IPFS на блокчейне Ethereum и Binance Smart Chain;

ANNOTATION

Final qualifying bachelor's work: 43 p., 13 fig., 1 appendix, 6 sources.

Topic: Creation of a parser for real estate objects and construction of filtering.

Object of research: data parsing technology.

Subject of research: real estate rental platform.

Purpose: development of a platform for renting real estate with a parser for ads from third-party sources.

Results: as a result of the study, the possibility of using the technology to implement a fast parser for processing large amounts of data was assessed.

Conclusion: As a result of the work, a service for renting real estate was developed with the collection of a large number of real estate listings from more than 60 websites. The developed web application is a collection of:

- Data parser from 60 web resources;
- The legitimacy checker;
- the client part of the web application to interact with the system;
- MongoDB Database and IPFS Related to Ethereum Blockchain and Binance Smart Chain;

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ

IPFS – Inter Planetary File System (протокол для збереження даних).

BSC – блокчейн Binance Smart Chain.

TheGraph – хостинг для децетралізованого ПЗ.

Selenium – частина парсеру для автоматихації скрейпінгу.

CITVY – система для оренди нерухомості на базі якої працюють парсери.

ВСТУП

Парсинг - це автоматизований процес збору даних з сайтів, застосовується для збору контенту: цін конкурентів, описів товарів, контактів, відгуків і рейтингів, файлів і для будь-яких інших завдань, коли потрібно зібрати великий обсяг інформації. Умовно інструменти для парсингу поділяються на дві частини. Одна з них - це розробка парсеру під задачу, цим займаються програмісти, які часто використовують різні мови програмування, наприклад, Python або JavaScript, щоб ретельно продумати логіку майбутнього скрипта до деталей, що вимагає часу і обчислювальних ресурсів, але в кінцевому підсумку дає найкращі результати. Завдання при такому підході вирішуються точно, як потрібно, можна збирати дані з необхідними умовами, швидкістю і обсягами.

Інша частина - непрофесійні користувачі, яким доводиться вибирати між безліччю існуючих парсерів, програм або сервісів, кожен з яких включає набір готових команд і обмежений реалізованої функціональністю. Це часто змушує користувачів витрачати час і гроші на вивчення кількох інструментів в спробі охопити широкий спектр можливих сценаріїв парсингу сайтів.

Метою випускної кваліфікаційної роботи є розробка парсеру для об’єктів нерухомості та побудова зручної системи фільтрації об’єктів нерухомості для веб-сайту з оренди житла.

Відповідно до поставленої мети в роботі вирішувалися наступні завдання:

1. Аналіз існуючих аналогів парсеру для об’єктів нерухомості та систем фільтрації даних.
2. Аналіз та огляд існуючих інструментів для розробки парсеру об’єктів нерухомості.
3. Розробка ефективного швидкого веб-скрейперу.
4. Розробка моделі та баз даних для збереження великої кількості даних.
5. Розробка та тестування системи фільтрації з парсерами.

1. ПОНЯТТЯ ПАРСИНГУ ОБ’ЄКТІВ НЕРУХОМОСТІ ТА ОГЛЯД ІСНУЮЧИХ ТЕХНОЛОГІЙ

1.1. Поняття парсеру об’єктів нерухомості та його застосування для автоматизації процесів пошуку, збору та фільтрації сайтів

Парсер об’єктів нерухомості необхідний для швидкого та автоматизованого збору даних про конкретні квартири які цікавлять клієнта за фільтрами. Парсер не буде завантажувати об'єкти нерухомості з однаковими контактами. Спочатку парсер нерухомості налаштовується відповідно до конкретних фільтрів. Ми можемо вибрати, які властивості хочемо зібрати. Наприклад, лише 1 та 2 кімнатні квартири в Києві та Одесі. Або просто комерційна нерухомість у певній місцевості. Фільтр встановлений в одному інтерфейсі і негайно застосовується до всіх сайтів оренди нерухомості. Ви також можете вказати фільтр за ціною, місцем розташування, поверхом та іншими параметрами.

1.2. Огляд технологій парсингу

Веб-скрапінг знаходить все більшу актуальність серед підприємств по всьому світу, зростаюча потреба у якісних даних. Мережа схожа на нескінченний океан неструктурованих даних, і разом із цими даними з’являються незвідані можливості.

Веб-скрапінг - це метод, який допомагає взяти або витягти вміст з веб-сайту з метою його використання в цілях, що не залежать від безпосереднього контролю власника сайту. Першим використанням веб-скрейпінгу було збирання текстових файлів. За допомогою таких інструментів, як Selenium, такі компанії, як IP-Label, створюють продукти, що дозволяють веб-розробникам та веб-майстрам щодня контролювати ефективність парсингу веб-сайтів.

Веб-скрапінг схожий на веб-індексацію - процес, за допомогою якого пошукові системи індексують веб-вміст. Різниця полягає у "правилі" robots.txt, яке визначає, куди боти можуть заходити на сайт. Веб-індексатори («хороші боти») дотримуються правил; веб-скрейпери, навпаки, просто крадуть будь-який вміст, який вони можеть отримати - ціни, акції, пропозиції або інформацію, яка в іншому випадку була б доступна лише платним абонентам або уповноваженим діловим партнерам.

Веб-сканери відвідують веб-сторінки, отримують дані та виявляють нові сторінки зі «насінних» сторінок. Хоча більшість людей вважають, що Google був, мабуть, першим сканером, який повністю просканував Інтернет, але веб-сканування як технологія має досить довгу та цікаву історію. Хоча початкові сканери могли сканувати не всі дані тоді, коли сучасні веб-сканери набагато розумніші, оскільки вони здатні контролювати веб-програми на предмет уразливості та доступності.

Selenium - це інструмент автоматизації веб-браузера, який має можливості виконувати широкий спектр завдань на автопілоті. Навчання користуванню selenium, безумовно, допоможе зрозуміти, як працюють веб-сайти. Він може бути використаний для імітації людини, яка відвідує веб-сторінку за допомогою звичайного браузера. Звідси можна отримати точні дані, які відвідувач-людина бачить на сторінці дані.

Він часто використовується для емуляції викликів ajax у веб-скрапінгу. Завдяки своїм потужним функціям автоматизації, selenium може допомогти набагато більше, ніж ми можемо уявити наприклад, тестування веб-сайтів та автоматизацію будь-якої трудомісткої діяльності, пов’язаної з Інтернетом. Коли витяг чистого тексту разом із пов’язаними заголовками є вимогою, Boilerpipe - чудовий варіант. Boilerpipe - це бібліотека Java, створена виключно для вилучення даних із веб-сторінок, структурованих чи неструктурованих. Він може розумно видаляти непотрібні теги HTML та інші шуми, виявлені на сторінках.

Родзинка Boilerpipe полягає в тому, що він може витягувати відповідний вміст за лічені мілісекунди та з мінімальним введенням від користувача. Точність вражаюче висока, що робить його одним з найпростіших інструментів для зішкрябування даних. Ознайомлення з цим інструментом може негайно покращити навички парсингу веб-сторінок.

Nutch розрекламований як золотий стандарт технологій вискоблювання веб-сторінок. Це не що інше, як програма з відкритим кодом, яка може сканувати та витягувати дані з веб-сторінок з блискавичною швидкістю. Nutch можна використовувати для сканування, вилучення та зберігання даних, запрограмованих відповідно до конкретних вимог. За лаштунками знаходиться дуже складний і потужний алгоритм сканування, що робить його одним з найкращих інструментів для сканування Інтернету.

Для проведення вишкрібання веб-сторінок для обходу та вилучення даних із необхідності кодування в Nutch вручну. Після налаштування він сканує сторінки у списку та отримує необхідні дані на сервер. Є багато простих команд, які використовуються для вишкрібання за допомогою Nutch, що полегшує роботу. Nutch - дуже корисний інструмент, коли справа доходить до парсингу.

Watir (яскрава вода) - це сімейство бібліотек з відкритим кодом Ruby, яке можна використовувати для автоматизації веб-браузера. Він простий у використанні та гнучкий. Він може взаємодіяти з браузером так само, як це робить людина.

Watir може виконувати такі функції, як перехід за посиланнями, заповнення форм, натискання кнопок і буквально все, що робить людина на веб-сторінці. Завдяки функціональності Ruby, Watir із задоволенням використовується та налаштовується. Як і будь-яка інша мова програмування, Ruby надає нам можливості читати файли даних, експортувати XML, підключатися до баз даних і записувати електронні таблиці.

Celerity - це обгортка JRuby, створена навколо HtmlUnit - браузера Java з підтримкою JavaScript. Він має простий у використанні API, який можна використовувати для програмної навігації веб-додатками. Це вражаюче швидко, оскільки немає трудомісткого візуалізації графічного інтерфейсу або непотрібних завантажень. Будучи масштабованим і ненав’язливим, він може працювати у фоновому режимі після початкового налаштування. Celerity - чудовий інструмент автоматизації браузера, за допомогою якого можна ефективно та швидко сканувати Інтернет.

1.3. Збір даних за допомогою парсингу

Парсинг html-сторінки можна розділити на три частини.

1. Отримання вихідного коду веб-сторінки - завантажити програмний код сторінки сайту, з якої ви хочете отримати інформацію. У різних мовах для цього існують різні способи, наприклад, в PHP 18 найчастіше використовують бібліотеку cURL.
2. Виймання необхідних даних з html-коду. Отримавши сторінку, необхідно обробити - відокремити звичайний текст від розмітки гіпертексту, побудувати ієрархічне дерево елементів документа, правильно взяти потрібну інформацію за допомогою регулярних виразів.
3. Фіксація результату. Успішно обробивши дані на сторінці,

нам необхідно зберегти їх в необхідному вигляді для подальшої обробки.

1.3.1. Основи збору текстових даних за допомогою парсингу

У деяких системах машинного перекладу та обробки природних мов письмові тексти людськими мовами аналізуються за допомогою комп'ютерних програм. Речення нелегко аналізувати програмами, оскільки існує значна неоднозначність у структурі мови, використання якої полягає у передачі значення

(або семантики) серед потенційно необмеженого кола можливостей, але лише деякі з них є загальними для конкретного випадку. Отже, вислів "Людина кусає собаку" проти "Собака кусає людину" є певним для однієї деталі, але іншою мовою може виглядати як "Людина собаку кусає" з опорою на ширший контекст, щоб розрізнити ці дві можливості, якщо справді ця різниця була занепокоєння. Важко підготувати офіційні правила для опису неформальної поведінки, хоча очевидно, що деяких правил дотримуються.

Для того, щоб проаналізувати дані на природній мові, дослідники повинні спочатку домовитись про граматику, яку слід використовувати. На вибір синтаксису впливають як лінгвістичні, так і обчислювальні проблеми; наприклад, деякі системи синтаксичного аналізу використовують лексичну функціональну граматику, але загалом аналіз граматик цього типу, як відомо, є неповним. Граматика будови фраз - це ще один лінгвістичний формалізм, який був популярний серед парсингового співтовариства, але інші дослідницькі зусилля були зосереджені на менш складних формалізмах, таких як той, що використовується в Penn Treebank. Неглибокий розбір має на меті знайти лише межі основних складових, таких як іменні фрази. Ще однією популярною стратегією уникнення лінгвістичних суперечок є розбір граматики залежностей.

Більшість сучасних аналізаторів є принаймні частково статистичними; тобто вони покладаються на навчальні дані, який вже анотовано (проаналізовано вручну). Цей підхід дозволяє системі збирати інформацію про частоту, з якою різні конструкції трапляються в конкретному контексті. Підходи, які були використані, включають прямі PCFG (імовірнісні безконтекстні граматики), максимальну ентропію та нейронні мережі. Більшість успішних систем використовують лексичну статистику (тобто вони враховують ідентичність залучених слів, а також їх частину мови). Однак такі системи вразливі до переобладнання і вимагають певного згладжування, щоб бути ефективними.

Алгоритми синтаксичного розбору для природної мови не можуть покладатися на граматику, яка має приємні властивості, як у граматиках, розроблених вручну для мов програмування. Як уже згадувалося раніше, деякі граматичні формалізми дуже важко обчислювати обчислювально; загалом, навіть якщо бажана структура не є контекстною, для виконання першого проходження використовується якийсь без контекстового наближення до граматики. Алгоритми, що використовують безконтекстні граматики, часто покладаються на варіант алгоритму CYK, зазвичай з деякою евристикою, щоб обрізати малоймовірний аналіз, щоб заощадити час. Однак деякі системи обмінюються швидкістю для точності, використовуючи, наприклад, версії алгоритму зменшення зсуву лінійного часу. Дещо недавно відбувся синтаксичний аналіз переоцінки, в якому аналізатор пропонує деяку кількість аналізів, а більш складна система вибирає найкращий варіант.

1.3.2. Обхід блокування зі сторони веб-сайтів

Надсилання повторних запитів з тієї ж IP-адреси - це чіткий слід того, що ви автоматизуєте HTTPS / HTTP-запити. Власники веб-сайтів можуть виявити та заблокувати наші веб-скрепери, перевіривши IP-адресу у своїх файлах журналів сервера. Часто існують автоматизовані правила, наприклад, якщо ви робите понад 100 запитів за 1 годину, ваш IP буде заблокований. Щоб цього уникнути, використовуйте проксі-сервери або віртуальну приватну мережу, щоб надсилати свої запити через низку різних IP-адрес. Ваш справжній IP буде прихований. Відповідно, ви зможете стягнути більшість веб-сайтів без проблем.

Корисно використовувати Google Cloud Functions або AppEngine як платформу для розміщення веб-скреперів. Це пов’язано з тим, що в поєднанні зі зміною вашого агента користувача на GoogleBot, власникам веб-сайтів може здатися, що ви насправді GoogleBot.

Справжні веб-браузери матимуть безліч різних заголовків, будь-який з них може перевіряти веб-сайти, щоб заблокувати ваш веб-скрепер. Щоб зробити ваш веб-скрепер більш реалістичним, ви можете скопіювати всі заголовки з httpbin.org/anything. (Це заголовки, які зараз використовує ваш браузер).

При використанні веб-служб скрапінгу дуже важливо використати дані якомога швидше. Однак, коли людина залишається на веб-сайті, їх швидкість перегляду досить низька в порівнянні зі сканерами.

Повільний темп - не єдина особливість людської роботи в мережі. Люди однозначно обробляють веб-сайти. Також слід враховувати різний час перегляду, випадкові клацання, коли користувачі відвідують веб-сайт. Однак боти дотримуються однакової схеми перегляду. Веб-сайти можуть легко ідентифікувати ботів, коли знаходять повторювані та подібні дії перегляду. Тому вам слід час від часу застосовувати різні шаблони вишкрібання, коли витягуєте дані з веб-сайтів. Деякі веб-сайти можуть мати вдосконалені механізми боротьби з ботами.

Розгляньте можливість поєднання кількох клацань, рухів миші або перемішування та поєднання випадкових дій, щоб ваш скребок виглядав як людина. За допомогою алгоритму за яким працює мій парсер, а саме поєднання різних випадкових клацань та імітація людських дій за допомогою неоднозначних та непослідовних дій парсеру.

1.3.3. Підтримка ефективності та швидкості парсингу

Система threading піклується про перемикання між потоками у операційній системі. Це приносить великі проблеми, якщо доведеться отримати доступ до деяких типових структур даних, оскільки ми ніколи не знаємо, який інший потік наразі отримує доступ до наших даних. Потім ми починаємо грати з синхронізованими блоками, блокуваннями, семафорами - просто для синхронізації доступу до наших спільних структур даних.

З eventlet це набагато простіше - ми завжди запускаємо лише один потік і стрибаємо між ними лише за вказівками вводу-виводу або під час інших викликів eventlet. Решта нашого коду працює безперервно, і без ризику що інший потік змішає наші дані.

Нам потрібно подбати лише про те що всі операції вводу-виводу мають бути неблокуючими (це в основному легко, eventlet надає неблокуючі версії для більшості потрібних нам операцій вводу-виводу).

Наш код, що залишився, не повинен бути дорогим для процесора, оскільки він довше блокував би перемикання між «зеленими» потоками, і потужність «зеленого» багатопоточності була б втрачена.

Великою перевагою eventlet є те, що він дозволяє писати код прямо, не псуючи його (занадто) сильно за допомогою Locks, Semaphores тощо.

1.3.4. Автоматизація парсингу / скрейпінг зображень

Парсер нерухомості обов’язково повинен парсити всі доступні зображення для кожної об’яви, для збереження зображень краще використовувати CDN, або напряму парсити URL зображень та заносити їх до бази даних.

1.3.5. Selenium

Selenium - це в першу чергу набір бібліотек для різних мов програмування. Ці бібліотеки використовуються для відправки HTTP запитів драйверу (звідси і назва WebDriver), за допомогою протоколу JsonWireProtocol, в яких зазначено дію, яку повинен зробити браузер в рамках поточної сесії. Прикладами таких команд можуть бути команди знаходження елементів по локатору, перехід по посиланнях, парсинг тексту сторінки / елемента, натискання кнопок або перехід по посиланнях на сторінці веб-сайту. Існують як офіційні прив'язки бібліотеки до популярних мов програмування, так і аматорські. Наприклад, бібліотека для підтримки мови PHP не є офіційною і розробляється Facebook.

Проектом Selenium і співтовариством підтримується робота з браузерами Microsoft Internet Explorer, Google Chrome, Mozilla Suite і Mozilla Firefox під управлінням операційних систем Microsoft Windows, Linux і Apple Macintosh.

Selenium, починаючи з версії 3.x почав вимагати для роботи з браузером Firefox окремий драйвер - GeckoDriver, що раніше називався Marionette

2. АЛГОРИТМИ ФІЛЬТРАЦІЇ ТА ГРУПУВАННЯ ДАНИХ

Колаборативні системи фільтрації мають багато форм, але багато загальних систем можна звести до двох етапів: пошук даних, які мають однакові шаблони, побудування матриці елемент-елемент, що визначає взаємозв'язки між парами елементів, визначення поточного об’єкта нерухомості, вивчивши матрицю та зіставивши дані цього об’єкту.

Інша форма колаборативної фільтрації може базуватися на неявних спостереженнях нормальної поведінки користувачів (на відміну від штучної поведінки, накладеної рейтинговим завданням). Ці системи спостерігають за тим, що робив користувач разом із тим, що робили всі користувачі, і використовують ці дані для прогнозування поведінки користувача в майбутньому або для прогнозування того, як може сподобатися користувачеві поводитися, даючи можливість. Потім ці прогнози потрібно відфільтрувати через бізнес-логіку, щоб визначити, як вони можуть вплинути на дії бізнес-системи.

Колаборативні алгоритми фільтрації розроблені для систем рекомендацій та застосовуються у багатьох реальних додатках, таких як Netflix , YouTube та Amazon. Як проаналізувати такий масштабний набір даних, має вирішальне значення. Факторизація матриць і тензорів з імовірнісною обробкою або без неї є можливою для здійснення ефективних алгоритмів для вирішення цієї проблеми. Загалом, спостерігається матриця містить рейтингові значення M користувачів та N елементів (фільми, відео чи товари), які однозначно не є негативними. Деякі записи в X можуть бути відсутніми. Може бути зібрана розріджена матриця. Метою спільної фільтрації є аналіз минулої історії користувацьких уподобань та рейтингів та використання проаналізованої інформації для прогнозування майбутнього рейтингу для користувача m щодо конкретного елемента n . У роботі Salakhutdinov and Mnih (2008) пропонується факторизація імовірнісної матриці (PMF) для наближення спостережуваної рейтингової матриці використання

продукту матриці користувача та матрицю елементів . Це можна порівняти зі стандартним методом поділу джерела за допомогою NMF, де спостерігається матриця збирається як спектрограми часової величини на різних частотних бункерах m та часових межах n . Негативна матриця X факторізуются як твір базис матриці B і вагова матриця W . Нехай матриці B і W будуть представлені відповідними векторами рядків і стовпців, тобто PMF будується відповідно до імовірнісних припущень. Передбачається, що функція ймовірності формування рейтингової матриці відповідає гауссовому розподілу де і позначають K -вимірні специфічні для користувача та специфічні для елемента вектори прихованих ознак відповідно, є спільним параметром дисперсії для всіх записів у X та позначає показник, який є одиницею, коли спостерігається і дорівнює нулю. Попередні щільності параметрів ПМФ приймаються як нульові середні показники Гаусса із загальними параметрами точності і для всіх записів у матрицях B і W, заданих. Максимальні апостеріорні (MAP) оцінки B і W обчислюються шляхом максимізації логарифму заднього розподілу за користувачем та матриць елементів, заданих фіксованими дисперсіями. Відповідно, максимізація логарифму заднього розподілу еквівалентна мінімізації наступної цільової функції: де параметри регуляризації визначаються з використанням різних параметрів дисперсії, у реалізації вибору належних гіперпараметрів або параметри регуляризації має вирішальне значення для регуляризації моделі. Узагальнення для майбутніх даних на основі PMF залежить від контролю складності. Зокрема, узагальнення за наявності розріджених та незбалансованих наборів даних є критичним у реальних додатках. Вибірка Гіббса була розроблена для реалізації байєсівського PMF з фіксованими гіперпараметрами або параметрами регуляризації.

2.1. Поняття рекомендаційних систем та їх типи

Одним із підходів до проектування систем, що рекомендують, є широке застосування колаборативної фільтрації. Колаборативна фільтрація базується на

припущенні, що люди, які домовились у минулому, погоджуватимуться в майбутньому, і що їм сподобаються подібні види предметів, як і раніше. Система формує рекомендації, використовуючи лише інформацію про рейтингові профілі для різних користувачів або предметів. Розташовуючи однорангових користувачів / елементи з історією оцінок, подібною до поточного користувача чи елемента, вони генерують рекомендації, використовуючи цей метод. Методи спільної фільтрації класифікуються як на основі пам'яті та на основі моделі. Добре відомим прикладом підходів, заснованих на пам'яті, є алгоритм, заснований на користувачах, тоді як підходів на основі моделей є рекомендатором картографування ядра.

Ключовою перевагою підходу до колаборативної фільтрації є те, що вона не покладається на вміст, що аналізується машиною, і тому він здатний точно рекомендувати складні елементи, такі як об’єкти нерухомості, не вимагаючи "розуміння" самого елемента. Багато алгоритмів використовувались для вимірювання схожості користувачів або подібності елементів у системах, що рекомендують. Наприклад, підхід k-найближчого сусіда (k-NN) та кореляція Пірсона, вперше реалізовані Алленом.

При побудові моделі за поведінкою користувача часто розрізняють явні та неявні форми збору даних.

Холодний старт: для нового користувача чи елемента недостатньо даних, щоб дати точні рекомендації. Примітка: одним із загальноприйнятих рішень цієї проблеми є алгоритм багаторукого бандита.

Масштабованість: У багатьох середовищах, в яких ці системи дають рекомендації, є мільйони користувачів та продуктів. Таким чином, для обчислення рекомендацій часто необхідна велика кількість обчислювальної потужності.

Економність: кількість товарів, що продаються на основних сайтах електронної комерції, надзвичайно велика. Найактивніші користувачі оцінили

лише невелику підмножину загальної бази даних. Таким чином, навіть найпопулярніші предмети мають дуже мало рейтингів.

Одним з найвідоміших прикладів є колаборативна фільтрація по елементах (люди, які купують x, також купують y), алгоритм, популяризований системою рекомендацій Amazon.com.

Багато соціальних мереж спочатку використовували спільну фільтрацію, щоб рекомендувати нових друзів, групи та інші соціальні зв’язки, досліджуючи мережу зв’язків між користувачем та їх друзями. колаборативна фільтрація все ще використовується як частина гібридних систем.

2.1.1. Технічна складова системи рекомендацій

На основі пам'яті: підхід, заснований на пам'яті, використовує дані рейтингу користувачів для обчислення подібності між користувачами або елементами. Типовими прикладами цього підходу є CF на основі сусідства та рекомендації top-N на основі предметів / користувачів. Наприклад, у підходах, заснованих на користувачах, значення оцінок, які користувач u дає елементу i, обчислюється як сукупність рейтингу подібних користувачів щодо елемента

$$r_{u,i} = \mathrm{aggr}_{u' \in U} r_{u',i}$$

де U позначає набір найпопулярніших N користувачів, які найбільш схожі на користувача u, який оцінив елемент i. Деякі приклади функції агрегування включають:

$$r_{u,i} = \frac{1}{N} \sum_{u' \in U} r_{u',i}$$

$$r_{u,i} = k \sum_{u' \in U} \mathrm{simil}(u, u') r_{u',i}$$

де k - коефіцієнт нормування, визначений як

$$k = 1 / \sum_{u' \in U} |\operatorname{simil}(u, u')|$$

де $\bar{r_u}$ - середня оцінка користувача u для всіх елементів, оцінених u.

$$r_{u,i} = \bar{r_u} + k \sum_{u' \in U} \operatorname{simil}(u, u')(r_{u',i} - \bar{r_{u'}})$$

Алгоритм, що базується на сусідстві, обчислює схожість між двома користувачами або елементами та виробляє прогноз для користувача, беручи середньозважене значення всіх оцінок. Обчислення подібності між елементами або користувачами є важливою частиною цього підходу. Для цього використовуються різні виміри, такі як кореляція Пірсона та подібність векторних косинусів.

Подібність кореляції Пірсона двох користувачів x, y визначається як

$$\operatorname{simil}(x, y) = \frac{\sum_{i \in I_{xy}} (r_{x,i} - \bar{r_x})(r_{y,i} - \bar{r_y})}{\sqrt{\sum_{i \in I_{xy}} (r_{x,i} - \bar{r_x})^2} \sqrt{\sum_{i \in I_{xy}} (r_{y,i} - \bar{r_y})^2}}$$

Підхід, заснований на косинусах, визначає подібність косинусів між двома користувачами x та y як:

$$\operatorname{simil}(x, y) = \cos(\vec{x}, \vec{y}) = \frac{\vec{x} \cdot \vec{y}}{||\vec{x}|| \times ||\vec{y}||} = \frac{\sum_{i \in I_{xy}} r_{x,i} r_{y,i}}{\sqrt{\sum_{i \in I_x} r_{x,i}^2} \sqrt{\sum_{i \in I_y} r_{y,i}^2}}$$

2.1.2. Рекомендаційна система на основі фільтрації вмісту

Іншим поширеним підходом при розробці рекомендаційних систем, є фільтрація на основі вмісту. Методи фільтрації на основі вмісту базуються на описі товару та профілі уподобань користувача. Ці методи найкраще підходять для

ситуацій, коли є відомі дані про елемент (ім'я, місцезнаходження, опис тощо), але не на користувач. Рекомендатори, що базуються на вмісті, розглядають рекомендацію як специфічну для користувача проблему класифікації та вивчають класифікатор, який сподобається і не сподобається користувачеві, виходячи з особливостей товару.

У цій системі ключові слова використовуються для опису елементів, а профіль користувача будується для позначення типу товару, який подобається цьому користувачеві. Іншими словами, ці алгоритми намагаються рекомендувати предмети, подібні до тих, що сподобалися користувачеві в минулому або досліджують сьогодні. Він не покладається на механізм входу користувача для створення цього часто тимчасового профілю. Зокрема, різні предмети-кандидати порівнюються з елементами, попередньо оціненими користувачем, і рекомендуються найоптимальніші елементи. Цей підхід сягає корінням у дослідження інформації та фільтрування інформації.

Для створення профілю користувача система здебільшого фокусується на двох типах інформації:

1. Модель уподобань користувача.
2. Історія взаємодії користувача з системою рекомендацій.

В основному, ці методи використовують профіль елемента (тобто набір дискретних атрибутів та ознак), що характеризує елемент у системі. Для абстрагування особливостей елементів у системі застосовується алгоритм подання елементів. Широко використовуваним алгоритмом є подання tf – idf (також зване представлення векторного простору). Система створює контентний профіль користувачів на основі зваженого вектора елементів елемента. Вагові коефіцієнти означають важливість кожної функції для користувача і можуть бути розраховані з індивідуально оцінених векторів вмісту за допомогою різних методів. Прості підходи використовують середні значення оціненого вектора елемента, тоді як інші складні методи використовують методи машинного навчання, такі як байєсівські

класифікатори, кластерний аналіз, дерева рішень та штучні нейронні мережі, щоб оцінити ймовірність того, що елемент сподобається користувачеві.

Ключовою проблемою фільтрації на основі вмісту є те, чи здатна система дізнатися вподобання користувачів з дій користувачів щодо одного джерела вмісту та використовувати їх для інших типів вмісту. Коли система обмежується рекомендуванням вмісту того самого типу, який вже використовує користувач, значення із системи рекомендацій значно менше, ніж тоді, коли можна рекомендувати інші типи вмісту з інших служб. Наприклад, рекомендувати статті на основі перегляду новин корисно, але було б набагато корисніше, коли музика, відео, товари, дискусії тощо з різних служб можуть бути рекомендовані на основі перегляду новин. Щоб подолати це, зараз більшість систем, що рекомендують контент, використовують певну форму гібридної системи.

Рекомендаційні системи на основі вмісту можуть також включати системи рекомендацій на основі думок. У деяких випадках користувачам дозволяється залишати текстовий огляд або відгук про елементи. Ці тексти, створені користувачами, є неявними даними для системи рекомендацій, оскільки вони є потенційно багатим ресурсом як функцій / аспектів товару, так і оцінки / настроїв користувачів до товару. Функції, вилучені з оглядів, створених користувачами, - це вдосконалені метадані елементів елементів, оскільки, оскільки вони також відображають такі аспекти елемента, як метадані, користувачі отримують значну увагу щодо вилучених функцій. Настрої, вилучені з оглядів, можна розглядати як оцінки користувачів за відповідними функціями. Популярні підходи системи рекомендацій, заснованої на думках, використовують різні методи, включаючи аналіз тексту, пошук інформації, аналіз настроїв (див. Також Мультимодальний аналіз настроїв) та глибоке навчання.

2.1.3. Технічна складова системи групування

В інтелектуальному аналізі даних ми часто хочемо аналізувати дані за деякими категоріями. У SQL рядок операторів GROUP BY групує рядки, що мають однакові значення категорій, у зведені рядки. У Pandas операція SQL GROUP BY виконується за допомогою аналогічного методу groupby(). Pandas 'groupby() дозволяє розділити дані на окремі групи для виконання обчислень для кращого аналізу.

2.1.4. Переваги і недоліки рекомендаційних систем

Рекомендаційні системи мають певні переваги та недоліки.

Плюси:

1. Прості рекомендації зменшують кількість пошуків, а іноді трапляються недобрі пропозиції
2. Відгуки користувачів дадуть точну інформацію, це також є перевагою, якщо ви купуєте в Інтернеті, оскільки ви також можете бачити інші відгуки, в більшості випадків чесні
3. Прискорити процес прийняття рішень та придбання на основі попередньої статистики

Мінуси:

1. Якщо система рекомендує продукцію з упередженням, тоді клієнт потраплятиме в неправильні угоди
2. Швидше за все, деякі веб-сайти можуть пропонувати продукти неправильно на основі аналізу небагато зібраної інформації

2.1.5. Порівняння рекомендаційних систем

Хоча більшість рекомендаційних систем, роблять пропозиції для окремих користувачів, у багатьох випадках вибрані елементи (наприклад, об'єкти

нерухомості) призначені не для особистого користування, а для споживання в групах. Цей документ досліджує, як можна створити ефективні групові рекомендації щодо об’єктів нерухомості, поєднуючи уподобання членів групи (як це виражено рейтингами), або комбінуючи рекомендації членів групи. Ці дві стратегії групування, які перетворюють традиційні алгоритми рекомендацій в алгоритми групових рекомендацій, поєднуються з п’ятьма загальновживаними алгоритмами рекомендацій для обчислення групових рекомендацій для різних композицій груп. Рекомендації групи оцінюються не лише з точки зору точності, але й з точки зору інших якісних аспектів, важливих для користувачів, таких як різноманітність, охоплення та випадковість. Крім того, у нас обговорюється вплив чисельності та складу групи на якість рекомендацій. Ми отримуємо, що стратегія групування, яка дає найбільш точні результати, залежить від алгоритму, який використовується для генерації окремих рекомендацій. Тому в нас пропонується поєднання стратегій групування, що перевершує кожну окрему стратегію з точки зору точності. Крім того, результати показують, що точність рекомендацій групи зростає із збільшенням подібності між членами групи. Крім того, різноманітність, охоплення та випадковість рекомендацій групи значною мірою залежать від використовуваної стратегії групування та алгоритму рекомендацій. Отже, для (комерційних) систем, що рекомендують групу, стратегію та алгоритм групування слід вибирати ретельно, щоб оптимізувати бажані показники якості рекомендацій групи. Висновки можуть бути використані як настанови для цього процесу відбору.

2.1.6. Що потрібно для створення фільтрації

Для створення фільтрації необхідні дані які ми будемо фільтрувати, та критерії які найбільш популярні серед користувачів для фільтрації. У нашому випадку ми маємо базу даних mongodb яка дозволяє нам за допомогою бібліотеки

mongoose виконувати фільтрацію за полями які є у нас в коллекції. А саме поля: price, date, location, coordinates[], floor, rooms, sqm, isOwner, verified.

3. СЕРВІС ДЛЯ ОРЕНДИ ЖИТЛА

В процесі виконання роботи був розроблений сервіс для оренди житла, з використанням бази даних MongoDB та IPFS (interplanetary file system) зі смарт-контрактами.

При розробці додатку використовувалися такі технології:

- HTML;
- CSS;
- Bootstrap;
- Мова JavaScript;
- Ethereum Web3;
- Мова Solidity.

Для реалізації клієнтської частини використовувалися HTML, CSS, Bootsrap та мова JavaScript. Було використано комбінацію бази даних mongodb та блокчейну BSC. Оскільки головна ідею проекту полягає у тому, щоб якнайшвидше напарсити об’яви з багатьох ресурсів, та забезпечити максимальну безпеку зберігання даних, та їх незмінюваність. Для цього було вирішено використовувати смарт-контракти та IPFS як своєрідну базу зберігання інформації, яка буде зберігатись в надійному протоколі, а не тільки на виділеному сервері.

Для взаємодії з IPFS, і нашими смарт-контрактами зокрема, використовувалося BSC JavaScript API Web3. Це дозволяє працювати з смарт-контрактами, та використовувати функції, які було написано в них, а відповідно використовувати смарт-контракти, викликати потрібні функції прямо з веб-додатку.

3.1. Поліпшення технології роботи системи для пошуку актуальних об’яв

Кешована версія сторінки (деякий або весь вміст, необхідний для її відображення), що зберігається в робочій пам’яті пошукової системи, швидко надсилається запитувачу. Якщо візит запізнився, пошукова машина може просто виступати в якості веб-проксі. У цьому випадку сторінка може відрізнятися від індексованих пошукових термінів. Кешована сторінка має вигляд версії, слова якої раніше індексувались, тому кешована версія сторінки може бути корисною веб-сайту, коли фактична сторінка втрачена, але ця проблема також вважається м’якою формою посилання.

Як правило архітектура високого рівня стандартного веб-сканера, коли користувач вводить запит у пошукову систему, це кілька ключових слів. В індексі вже є назви сайтів, що містять ключові слова, і вони миттєво отримуються з індексу. Реальне навантаження на обробку полягає у формуванні веб-сторінок, які є списком результатів пошуку: Кожна сторінка у всьому списку повинна бути зважена відповідно до інформації в індексах. Тоді верхній елемент результату пошуку вимагає пошуку, реконструкції та розмітки фрагментів, що відображають контекст відповідних ключових слів. Це лише частина обробки, яку вимагає кожна веб-сторінка результатів пошуку, а інші сторінки (поруч із вершиною) вимагають більшої кількості цієї обробки.

Окрім простих пошуків ключових слів, пошукові системи пропонують власні керовані графічним інтерфейсом або командами оператори та параметри пошуку для уточнення результатів пошуку. Вони забезпечують необхідні елементи керування для користувача, задіяного в циклі зворотного зв'язку, який користувачі створюють за допомогою фільтрації та зважування при уточненні результатів пошуку, враховуючи початкові сторінки перших результатів пошуку. Наприклад, з 2007 року пошукова система Google.com дозволила фільтрувати за датою, натиснувши "Показати інструменти пошуку" в крайньому лівому стовпці початкової сторінки результатів пошуку, а потім вибравши бажаний діапазон дат. Також можна зважити за датою, оскільки кожна сторінка має час модифікації.

Більшість пошукових систем підтримують використання логічних операторів AND, OR і NOT, щоб допомогти кінцевим користувачам уточнити пошуковий запит. Логічні оператори призначені для буквального пошуку, що дозволяє користувачеві уточнити та розширити умови пошуку. Двигун шукає слова чи фрази точно так, як вони були введені. Деякі пошукові системи забезпечують розширену функцію, яка називається близьким пошуком, що дозволяє користувачам визначати відстань між ключовими словами. Існує також пошук на основі концепції, де дослідження передбачає використання статистичного аналізу на сторінках, що містять слова чи фрази, які ви шукаєте.

Корисність пошукової системи залежить від релевантності набору результатів, який вона дає. Незважаючи на те, що веб-сторінок, що містять певне слово чи фразу, може бути мільйони, деякі сторінки можуть бути більш релевантними, популярними чи авторитетними, ніж інші. У більшості пошукових систем використовуються методи ранжирування результатів для отримання найкращих результатів. Те, як пошукова система вирішує, які сторінки найкраще відповідають, і в якому порядку повинні відображатися результати, різниться від одного механізму до іншого. Методи також змінюються з часом, коли змінюється використання Інтернету та розвиваються нові методи. Існує два основних типи пошукової системи, яка розвинулася: одна - це система заздалегідь визначених та впорядкованих за ієрархією ключових слів, яку люди широко програмували. Інша - це система, яка генерує "перевернутий індекс", аналізуючи розміщені в ній тексти. Ця перша форма набагато більше покладається на сам комп'ютер, щоб виконати основну частину роботи.

3.1.1. Реєстрація користувача

Реєстрація необхідна для додавання у базу даних додаткових полів та верифікації користувача.

Процес реєстрації користувача проходить наступним чином:

- Підтвердити свій номер телефону;
- Створити особистий електронний гаманець - необов’язково;
- Заповнення форми реєстрації (Ім’я, Прізвище) - необов’язково;
- Верифікація BankID або KYC за допомогою personal ID чи паспорту;
- Введені дані передаються до функції смарт-контракту, за допомогою API Web3;
- Дані передаються через смарт контракт на оракл та IPFS, паралельно йде збереження в mongoDB;
- Відправка повідомлення про успішну реєстрацію.

3.1.2. Підтвердження користувача

Знай свого клієнта (know your customer, скорочено KYC) - потрібен для верифікації користувачів чиї об’яви були стягнуті парсером, термін банківського і біржового регулювання для фінансових інститутів і букмекерських контор, а також інших компаній, що працюють з грошима приватних осіб, що означає, що вони повинні ідентифікувати і встановити особу контрагента, перш ніж проводити фінансову операцію.

Ця вимога поширюється на отримання розумно повних відомостей про контрагентів-юридичних особах, характер їх бізнесу і окремих господарських операцій, для забезпечення яких проводиться фінансова операція.

Вважається, що така практика перешкоджає відмиванню грошей, фінансуванню тероризму та ухилення від сплати податків.

В даний час вимоги і стандарти, спрямовані на реалізацію цього принципу, встановлюються на рівні національного законодавства, нормативних документів банківських регуляторів і міжнародних організацій, таких як FATF.

Послідовне впровадження принципу Знай свого клієнта призвело до поступової заборони обслуговування анонімних банківських рахунків.

3.1.3. Процес створення акаунтів для авторів об’яв

Кожен користувач, об’яву якого було додано в нашу базу даних, отримує смс повідомлення на свій телефон (який ми також парсимо з об’яви) з гіперпосиланням чату з потенційними орендодавцями.

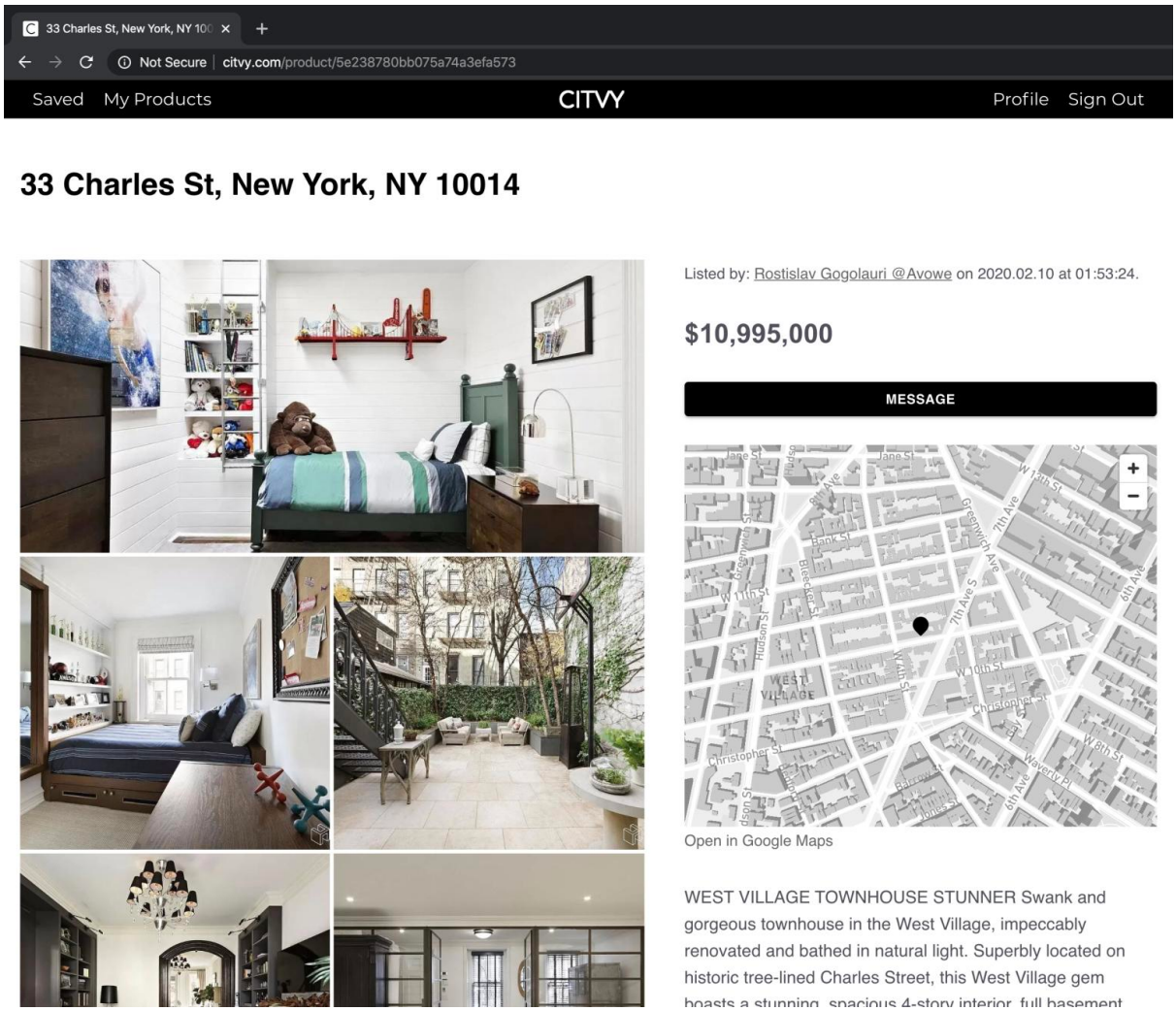

Рис. 3.2. Сторінка об’яви.

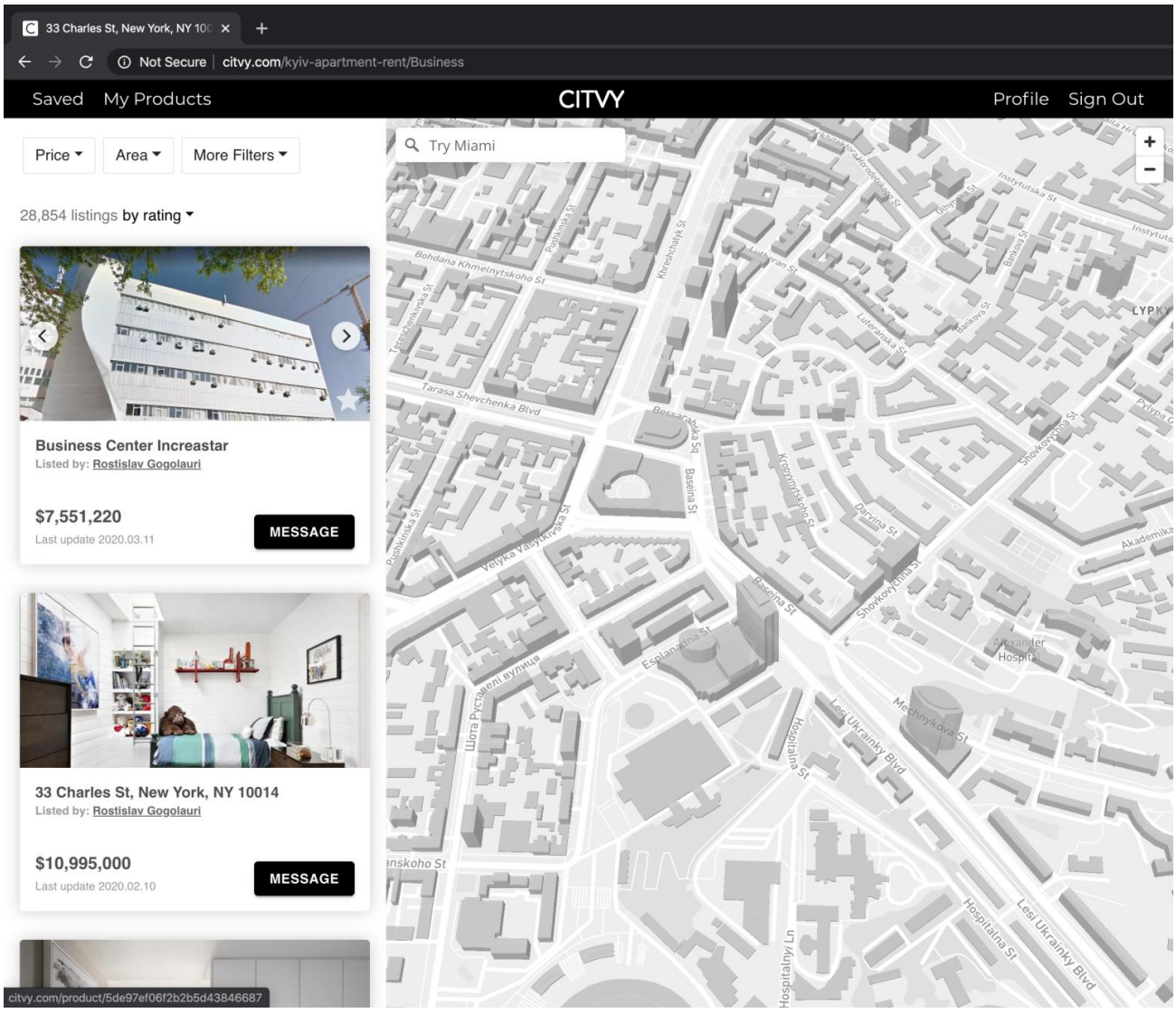

Рис. 3.3. Головна сторінка сайту.

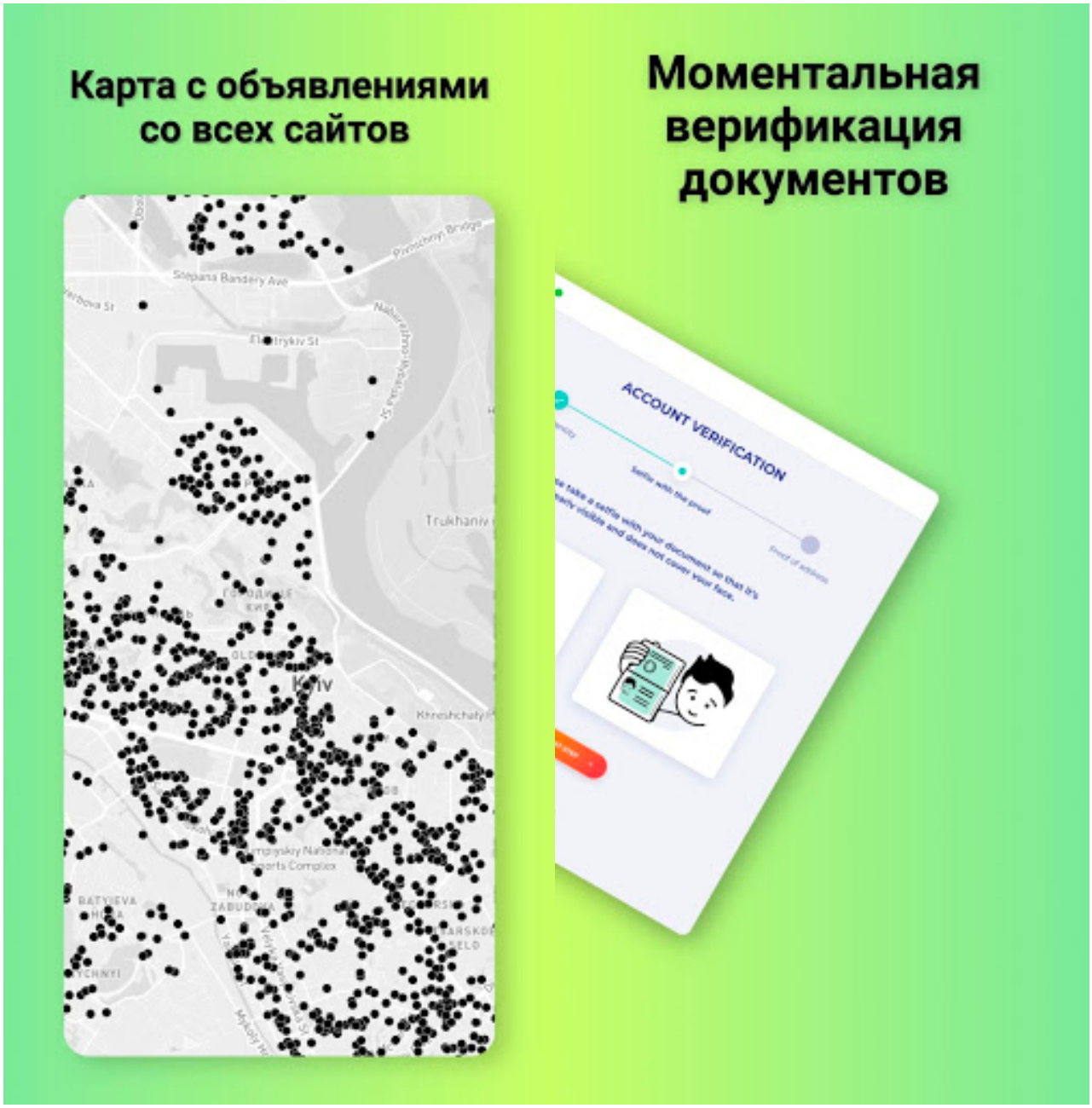

Рис. 3.4. Сторінка об’яв на карті у додатку.

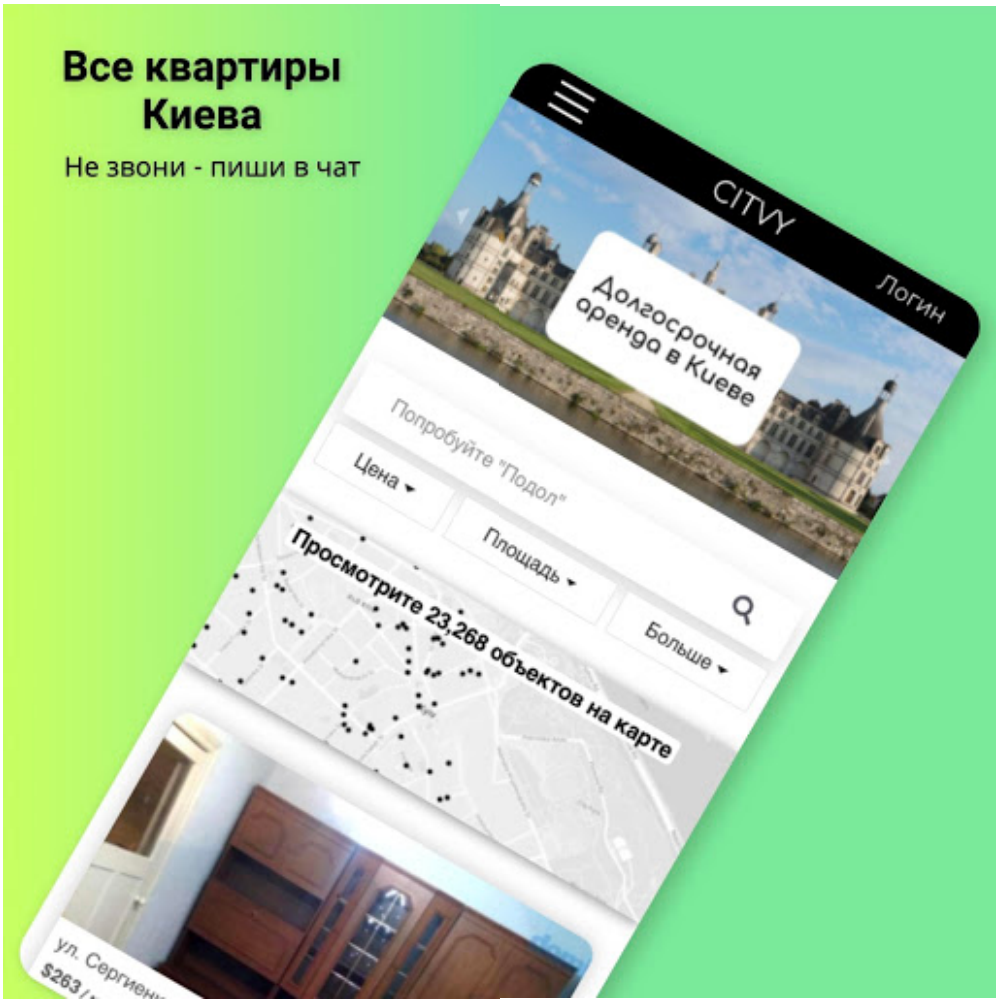

Рис 3.5. Сторінка з коротким гайдом для телефону.

3.2. Огляд функціональності адміністратора

Для переходу на панель адміністратора потрібно натиснути на вкладку *Admin* на навігаційній панелі сайту. Ця кнопка буде доступна тільки якщо у вас є обліковий запис адміністратора.

4. ПРАКТИЧНА РЕАЛІЗАЦІЯ

4.1 Проектування розробки програми

Розробка почалася з складання діаграм та проектування архітектури парсерів для платформи OLX.

4.1.1 Діаграма станів

Діаграма станів - орієнтований граф для кінцевого автомата, в якому:

- вершини позначають стан
- дуги показують переходи між двома станами

Спроектуємо діаграму станів програми-парсера (рис. 4.1 – Діаграма станів).

Рис 4.1. Діаграма станів.

4.1.2 IDEF0 діаграма

IDEF0 використовується для створення функціональної моделі (рис. 4.2), що відображає структуру і функції системи, а також потоки інформації і матеріальних об'єктів, що зв'язують ці функції.

Рис 4.2. IDEF0 діаграма.

4.1.3 Діаграма потоків даних

Діаграма потоків даних (англ. *Data Flow Diagram*) — модель проектування, графічне представлення «потоків» даних в інформаційній системі. Діаграма потоків даних також може використовуватись для візуалізації процесів обробки даних. Діаграми потоків даних містять чотири типи графічних елементів:

1. процеси - являють собою трансформацію даних в рамках описуваної системи;
2. сховища даних (репозиторії);
3. зовнішні по відношенню до системи сутності;
4. потоки даних між елементами трьох попередніх типів.

Спроектуємо DFD діаграму для нашого завдання (рис. 4.3).

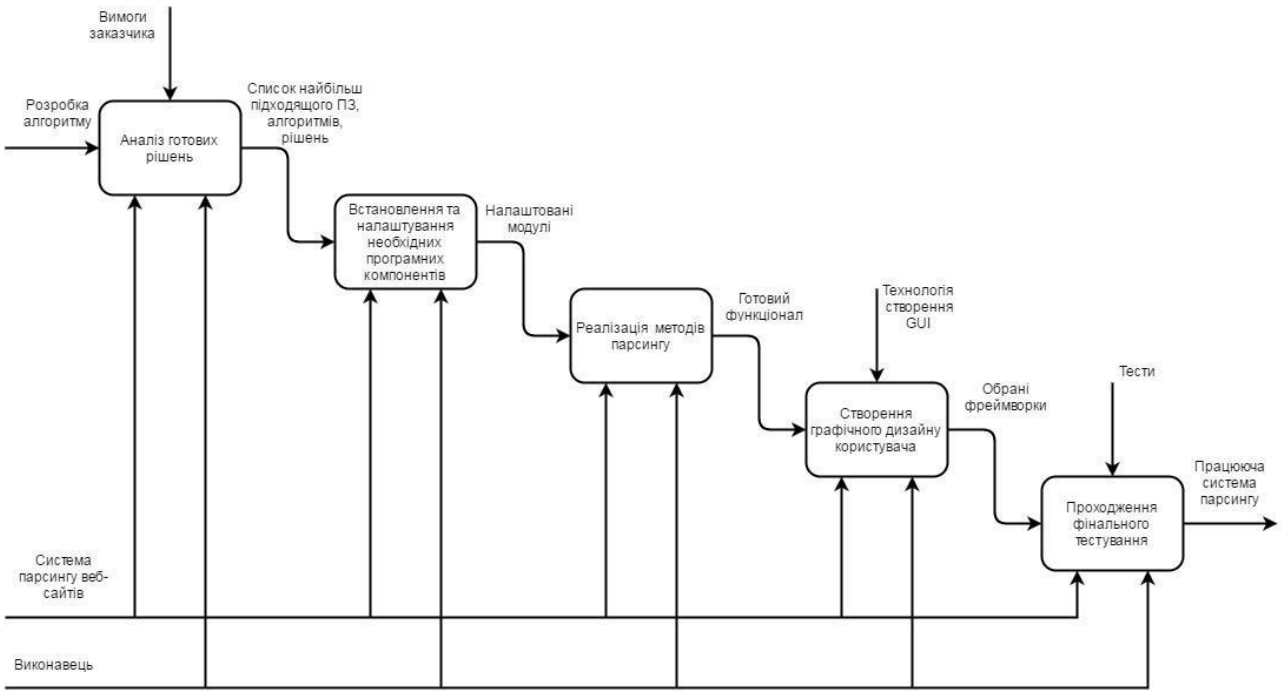

Рис 4.3. DFD діаграма.

4.1.4 Діаграма прецедентів

Діаграма прецедентів візуально зображає різноманітні сценарії взаємодії між акторами (користувачами) і прецедентами (випадками використання), описує функціональні аспекти системи (бізнес логіку).

Діаграми прецедентів відіграють важливу роль не тільки у комунікації між збирачами вимог до проекту і потенційними користувачами. Діаграми прецедентів дописані бізнес логікою і детальними специфікаціями прецедентів, як джерельна інформація, успішно використовують учасники розробки проекту на всіх його фазах (зародження, дизайн, програмування, тестування, документування). Добре продумані і завершені специфікації прецедентів легко перевтілюються у тестові випадки.

Елементи діаграми прецедентів:

1. Актор – користувач.
2. Прецедент – випадок використання, дія. Позначається овалом.
3. Граничні межі системи охоплюють усі випадки використання у системі. Позначається прямокутником.

Елементи взаємодії діаграми прецедентів:

1. Використовує – користувач виконує дію.
2. Включає – один прецедент використовує іншого.
3. Розширює – представлення дочірніх прецедентів.
4. Вимагає – наступний прецедент вимагає виконання попереднього.
5. Схожий – прецеденти подібні, але описують різну функціональність.
6. Рівнозначний - подібна функціональність, але користувач сприймає, як різну.

Спроектуємо діаграму перецедентів (рис. 4.4).

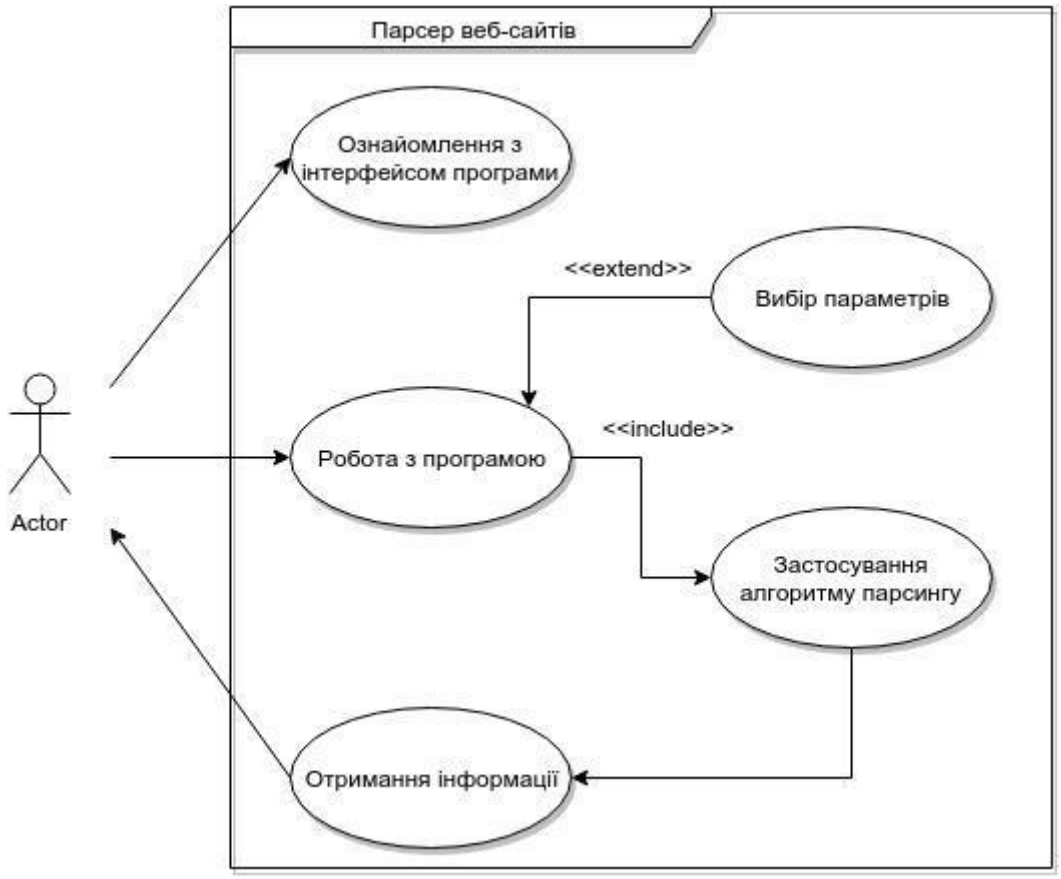

Рис 4.4. Діаграма прецедентів.

4.2 Розробка функціоналу програми

Парсер написаний на мові Python3 та використовує ряд бібліотек та API таких як google maps geocoder для переводу адрес вулиць у координати широти та довготи.

4.2.1 Реалізація методу завантаження веб-сторінки та парсингу

Ми маємо метод який приймає XPATH різних сайтів нерухомості як параметри та виконує завантаження веб-сторінки та збереження у базу даних mongodb:

```
from bs4 import BeautifulSoup as BS
import lxml.html
from bson.objectid import ObjectId
from .config import URL, last_page, apikey
import googlemaps
from . import uploader
import requests
from requests.exceptions import Timeout
```

```
import datetime
import re
from urllib.parse import urljoin, urlparse
from .sms import SMS


gmaps = googlemaps.Client(key='AIzaSyHn5ZBsN-K09fjLG7mVv2mcNQg')
HEADERS = {"User-Agent": "Mozilla/5.0 (X11; Ubuntu; Linux x86_64; rv:78.0)
Gecko/20100101 Firefox/78.0"}
SPLASH_SERVER_URL = "http://127.0.0.1:8050/execute"
net = urlparse(URL)
base_url = f"{net.scheme}://{net.netloc}/"

sms_block = ['38044']

excepted_urls = [
    'mesto.ua',
]

url_in_style = {
    'novolipki.com.ua': ['div', {'class': 'pic'}],
    'воскресенье.укр': ['span', {'class': 'control-slide'}],
}


path_num_tel = {
    'aviso.ua':                                ['//*[@id="show-number-button"]',
'/html/body/div/div[3]/div[1]/div[2]/div/div[3]/p[1]'],
    'rieltor.ua':
['/html/body/div/div/div/div/div/div/div/div[1]/div[2]/div[3]/div[2]/div[2]/button',
'/html/body/div/div/div/div/div/div/div/div/div[1]/div[2]/div[3]/div[2]/div[2]/div/div'],
    'fn.ua':            ['/html/body/div[1]/div[5]/div/div[3]/div[4]/button',
'/html/body/div[1]/div[5]/div/div[3]/div[4]/p[@class="ad-contacts"][last()]'],
    'anck.com.ua':
['/html/body/div[2]/div/aside/section/div[1]/div[2]/div[1]/p[1]',
'/html/body/div[2]/div/aside/section/div[1]/div[2]/div[1]/p[1]'],
    'thecapital.com.ua': ['', '/html/body/header/div[1]/div/div[2]/a[1]'],
    'mesto.ua':
['/html/body/div[5]/div[1]/div[1]/div/div[2]/div[2]/div[2]/div[1]/div[1]/form/button'
,
'/html/body/div[5]/div[1]/div[1]/div/div[2]/div[2]/div[2]/div[1]/div[1]/form/p[1]/spa
n/a'],
```

```
          'nextrealty.com.ua':                                                    ['',
'/html/body/div[4]/div/section[1]/div/div/div[3]/span[4]/a'],
          '100realty.ua':
['/html/body/div[2]/div/div[2]/div[3]/div[1]/div[1]/div/div[1]/div[1]/div[8]/div[1]/a
',
'/html/body/div[2]/div/div[2]/div[3]/div[1]/div[1]/div/div[1]/div[1]/div[8]/div[1]/di
v/p'],
          'megamakler.com.ua':
['/html/body/div[3]/div[1]/div[1]/div[1]/div[2]/div[3]/a',
'/html/body/div[3]/div[1]/div[1]/div[1]/div[2]/div[3]/span'],
          'allstar.com.ua': ['', '/html/body/div[2]/div[2]/div[5]/p'],
          'remax.ua':
['/html/body/form/div[3]/div[5]/div/div[2]/div[2]/div[1]/div[1]/div[1]/div/div[3]/div
/div[2]/div[1]/div/div/div[@id="ShowAgentPhone"]',
'/html/body/form/div[3]/div[5]/div/div[2]/div[2]/div[1]/div[1]/div[1]/div/div[3]/div/
div[2]/div[1]/div/div/div[1]'],
          'sto-grad.com.ua': ['', '/html/body/div/header/div/div/div[2]/div[1]/a'],
          'geo-svit.com.ua': ['', '//header//button'],
          'address.ua':
['/html/body/div[2]/form/div/div[2]/div[1]/div[2]/div/div[2]/a[1]',
'//*[@id="phone_a1"]'],
          'est.ua':
['/html/body/div[4]/div[2]/div/div/div/div[4]/div[2]/div/div[2]/div/div[2]/div[1]/div
[2]/div',
'/html/body/div[4]/div[2]/div/div/div/div[4]/div[2]/div/div[2]/div/div[2]/div[2]'],
          'country.ua':
['/html/body/div[1]/main/div/div[1]/div/div/div/aside/div/div/div[3]/div[1]/a',
'/html/body/div[1]/main/div/div[1]/div/div/div/aside/div/div/div[3]/div[1]/div/span[2
]/a'],
          'valion.ua': ['', '/html/body/div/div[1]/div/div[3]/div[1]/div/div[1]/a'],
          'kievrealtor.com.ua':                                                   ['',
'/html/body/div[1]/div[2]/div[3]/div/div/div/div[2]/div/div[1]/div[2]/div[1]/div[2]/d
iv[2]/div[3]/a'],
          'atlanta.ua':                                ['//*[@id="show-number-button"]',
'/html/body/div[3]/div/div[3]/div/div[2]/div[2]/div[2]/div[2]/div[2]'],
          'novolipki.com.ua':                                                     ['',
'/html/body/div[3]/header/div[1]/div[2]/div[2]/div[1]/p/span/a'],
          'asnu.net':
['/html/body/div[7]/div/table/tbody/tr/td[2]/table[1]/tbody/tr/td/table/tbody/tr/td[1
]/div[3]/div/p[4]/span[1]/a',
'/html/body/div[7]/div/table/tbody/tr/td[2]/table[1]/tbody/tr/td/table/tbody/tr/td[1]
/div[3]/div/p[4]/span[2]'],
```

```
        'mayak.kiev.ua': ['', '/html/body/div[2]/header/div/div[3]/div[2]/p'],
        'wellhome.kiev.ua':                                                    ['',
'/html/body/div[2]/div/div/section/div/div[1]/div[1]/div[8]/div[2]/div/section/div/di
v/div[2]/div/ul/li[1]/a'],
        'hubestate.com.ua':                                                    ['',
'/html/body/div[2]/div[1]/div[1]/header/div[1]/div[2]/div/div/div/div[3]/div/div[2]/s
trong'],
        'воскресенье.укр':                                                     ['',
'/html/body/main/div[2]/div[2]/div/div[2]/div[2]/ul[2]/li[1]/a'],
        'dom.ria.com':      ['//div[@class="mb-10      phoneShowLink      pointer
callsActionShow"]', '//div/a/canvas'],
    }


    path_to_images = {
        'aviso.ua': [True, '', ['div', {'class': 'carousel-inner'}], None],
        'rieltor.ua': [True, '', ['ul', {'class': 'ov-gallery__main'}], None],
        'address.ua': [True, '', ['div', {'class': 'ff'}], ['/sm/', '/bg/']],
        'est.ua':   [False,   '//*[@class="app__gallery-photos-count"]',   ['div',
{'class': 'app__gallery'}], None],  # ['360x270/crop', '1280x1280/resize']
        'fn.ua': [True, '', ['div', {'class': 'slick-track'}], None],
        'dom.ria.com':                                                       [False,
'/html/body/div[1]/div[3]/div/div[3]/main/div[1]/div/div[1]',    ['div',    {'class':
'tumbsInner'}], ['m.jpg', 'fl.jpg']],
        'country.ua':                                                        [False,
'/html/body/div[1]/main/div/div[1]/div/div/div/div/div[1]/div[2]/div[1]/div/div/div[1
]/a', ['div', {'class': 'gallery__side'}], None],
        'thecapital.com.ua': [True, '', ['div', {'class': 'lightSlider'}], None],
        'valion.ua': [True, '', ['div', {'class': 'swiper-container'}], None],
        'mesto.ua': [True, '', ['div', {'class': 'slick-track'}], None],
        'nextrealty.com.ua':   [True,   '',   ['div',   {'class':   'owl-stage'}],
['preview_', '']],
        '100realty.ua': [True, '', ['div', {'class': 'slick-track'}], None],
        'megamakler.com.ua':  [True,  '',  ['div',  {'class':  'fotorama__grab'}],
None],
        'allstar.com.ua':   [True,   '',   ['div',   {'class':   'DivListFotoIm'}],
['immovables/apartment-rent-kiev-real-estate-agency-kiev-', 'immovables/']],
        'kievrealtor.com.ua': [True, '', ['div', {'class': 'owl-stage'}], None],
        'atlanta.ua':   [True,   '',   ['div',   {'class':   'swiper-wrapper'}],
['objects/375px', 'objects/805px']],
        'novolipki.com.ua': [True, '', ['div', {'class': 'slick-track'}], None],
```

```
            'asnu.net': [True, '', ['div', {'class': 'object-gallery-wrap'}], ['s.JPG',
'.JPG']],
            'mayak.kiev.ua': [True, '', ['div', {'class': 'slick-track'}], None],
            'remax.ua': [True, '', ['div', {'class': 'sp-slides'}], None],
            'wellhome.kiev.ua': [True, '', ['div', {'class': 'owl-wrapper'}], None],
            'sto-grad.com.ua': [True, '', ['div', {'class': 'object__gallery'}], None],
            'hubestate.com.ua': [True, '', ['div', {'class': 'images-wrapper'}],
['600x401', '600x401']],
            'anck.com.ua': [False, '/html/body/div[2]/div/article/div[3]', ['div',
{'class': 'slides'}], None],
            'geo-svit.com.ua': [True, '', ['ul', {'class': 'slides'}], None],
            'воскресенье.укр': [True, '', ['div', {'class': 'sli-links'}], None],
      }


      XPATH = {
            'of': '//article[{}]/div/div[1]/a/div[2]/div[2]',
            'redirect_link': '//article[{}]/div/div[2]/a[1]',
            'name': '//article[{}]/div/div[2]/a[2]/div',
            'district': '//article[{}]/div/div[2]/div[1]',
            'info': '//article[{}]/div/div[2]/div[2]/div[2]',
            'price': '//article[{}]/div/div[2]/div[2]/div[1]/a/div',
            'description': '//article[{}]/div/div[2]/div[3]/div',
            'initial_company': '//article[{}]/div/div[1]/a/div[2]/div[1]',
            'owner': '//article[{}]/div/div[1]/a/span',
            'update_string': '//article[{}]/div/div[2]/div[2]/div[1]/div',
      }


      def equal(arr):
            try:
                  f = arr[0]
                  s = arr[1]
            except IndexError:
                  return True
            if f == s:
                  return True
            return False


      def unique(arr):
            unique_list = list()
```

```
    for i in arr:
          if i not in unique_list:
                unique_list.append(i)
    return unique_list


def prettify(line):
    tel = ""
    count = 0
    counter = 0
    code = ""
    start_code = ""
    for ch in line:
          if ch.isdigit():
                code += ch
                break
    if code == "3":
          counter = 12
          start_code = "+"
    elif code == "0":
          counter = 10
          start_code = "+38"
    for ch in line:
          if ch.isdigit():
                tel += ch
                count += 1
          if count == counter:
                tel = start_code + tel
                if len(tel) == 13:
                      tmp = ""
                      tmp += tel[:4]
                      tmp += " ("
                      tmp += tel[4:6]
                      tmp += ") "
                      tmp += tel[6:9]
                      tmp += "-"
                      tmp += tel[9:11]
                      tmp += "-"
                      tmp += tel[11:13]
                      tel = tmp
                      return tel
                else:
```

```
                        return tel


    def get_url(s):
        start = s.index("url(")+4
        end = s[start:].index(")")+start
        return s[start:end].replace("'", "")


    def extract_integers(value):
        try:
              return int(''.join(re.findall("\d", value)))
        except:
              return 0


    def real_url(redirect_link):
        LUA_SCRIPT = "function main(splash, args) \n\
        headers = { \n\
            [\"user-agent\"] = \"Mozilla/5.0 (X11; Ubuntu; Linux x86_64; rv:78.0)
Gecko/20100101 Firefox/78.0\", \n\
        } \n\
        splash:set_custom_headers(headers) \n\
        splash.images_enabled = false \n\
        assert(splash:go(args.url)) \n\
        assert(splash:wait(2)) \n\
        return { \n\
          url = splash:url(), \n\
        } \n\
        end"

        response = requests.post(
              url=SPLASH_SERVER_URL,
              headers=HEADERS,
              json={
                    'lua_source': LUA_SCRIPT,
                    'url': redirect_link,
              },
              timeout=20,
        )

        if response.ok:
```

```
                return response.json()['url']

            return redirect_link


      LUA_SCRIPT = "function main(splash, args) \n\
        headers = { \n\
            [\"user-agent\"] = \"Mozilla/5.0 (X11; Ubuntu; Linux x86_64; rv:78.0)
Gecko/20100101 Firefox/78.0\", \n\
        } \n\
        splash:set_custom_headers(headers) \n\
        splash.images_enabled = false \n\
        assert(splash:go(args.url)) \n\
        assert(splash:wait(2)) \n\
        return { \n\
          html = splash:html(), \n\
        } \n\
        end"


      def get_cards():
          for number_page in range(1, last_page + 1):
                with open('./v_phones/1.csv', 'r') as csv:
                      verified_numbers    =    {number.strip()    for    number    in
csv.read().split(',') if number.strip() != ''}

                tmp_url = URL + '?page=' + str(number_page)
                success = False
                color = "\x1B[34;1m"
                status = "REFRESH"
                clear = "\x1B[0m"
                page_cards = list()
                while not success:
                      try:
                            response = requests.post(
                                  url=SPLASH_SERVER_URL,
                                  headers=HEADERS,
                                  json={
                                        'lua_source': LUA_SCRIPT,
                                        'url': tmp_url,
                                  },
                                  timeout=20,
```

```
                        )
                        if response.ok:
                              content = response.json()['html']
                              tree = lxml.html.document_fromstring(content)
                              articles = tree.xpath('//article')
                              if len(articles) == 30:
                                    success = True
                                    color = "\x1B[32;1m"
                                    status = "SUCCESS"
                              else:
                                    color = "\x1B[34;1m"
                                    status = "REFRESH"
                        except Timeout:
                              color = "\x1B[33;1m"
                              status = "SLOW INTERNET"

                        if success:
                              print(color,    f"\r[    {status}    ]",    clear,    end="
\n")
                        else:
                              print(color,    f"\r[    {status}    ]",    clear,    end="
")

                  content = response.json()['html']
                  tree = lxml.html.document_fromstring(content)

                  for i in range(30):
                        redirect_link                =                urljoin(base_url,
tree.xpath(XPATH['redirect_link'].format(i+1))[0].get('href').replace('\\\"', ''))

                        name = tree.xpath(XPATH['name'].format(i+1))[0].text_content()
                        district                                                    =
tree.xpath(XPATH['district'].format(i+1))[0].text_content()
                        update_string                                               =
tree.xpath(XPATH['update_string'].format(i+1))[0].text
                        owner = "без комиссии"

                        try:
                              owner                                                 =
tree.xpath(XPATH['owner'].format(i+1))[0].text_content()
                        except:
                              pass
```

```
                        address = name + ', Киев'
                        payload = {
                              'apikey': apikey,
                              'q': address
                        }

                        try:
                              try:
                                    location                                    =
requests.get("https://geocode.search.hereapi.com/v1/geocode", params=payload)
                                    lat                                         =
location.json()['items'][0]['position']['lat']
                                    lon                                         =
location.json()['items'][0]['position']['lng']
                              except:
                                    lat                                         =
gmaps.geocode(address)[0]['geometry']['location']['lat']
                                    lon                                         =
gmaps.geocode(address)[0]['geometry']['location']['lng']
                        except:
                              lat = "50.4501"
                              lon = "30.5234"

                        try:
                              all_info                                          =
tree.xpath(XPATH['info'].format(i+1))[0].xpath('//li')
                        except:
                              all_info = []

                        parsed_data = {
                              'info': [x.text for x in all_info],
                              'price': tree.xpath(XPATH['price'].format(i+1))[0].text,
                        }
                        description                                             =
tree.xpath(XPATH['description'].format(i+1))[0].text

                        try:
                              initial_company                                   =
tree.xpath(XPATH['initial_company'].format(i+1))[0].text
                        except:
                              try:
```

```
                                    initial_company                                    =
tree.xpath('//article[{}]/div/div[1]/a/div/div[1]')[0].text
                              except:
                                    initial_company = ""

                        if len(parsed_data['info']) > 0 and len(parsed_data['info']) <
6:
                              rooms = parsed_data['info'][0]
                              area = None
                              kitchenArea = None
                              livingArea = None
                              floor = None
                              re_type = None
                              year_built = None
                              bulding_type = None
                              for info in parsed_data['info']:
                                    if 'м²' in info:
                                          area  =  info.split('/')[0].replace(' ', '')
if '/' in info else info.replace('м²', '').replace(' ', '')
                                          kitchenArea       =       info.replace('м²',
'').split('/')[2].replace(' ', '') if '/' in info else ''
                                          livingArea = info.split('/')[1].replace(' ',
'') if '/' in info else ''
                                    elif 'этаж' in info:
                                          floor = info
                                    elif 'г' in info:
                                          year_built = info.replace('г', '')
                                    elif not year_built:
                                          re_type = info
                        elif len(parsed_data['info']) == 0:
                              rooms,  area,  kitchenArea,  livingArea,  floor,  re_type,
year_built, bulding_type = "", "", "", "", "", "", "", ""
                        else:
                              rooms = parsed_data['info'][0]
                              area = parsed_data['info'][1].split('/')[0].replace(' ',
'')  if  '/'  in  parsed_data['info'][1]  else  parsed_data['info'][1].replace('м²',
'').replace(' ', '')
                              kitchenArea    =    parsed_data['info'][1].replace('м²',
'').split('/')[2].replace(' ', '') if '/' in parsed_data['info'][1] else ''
                              livingArea                                                =
parsed_data['info'][1].split('/')[1].replace('     ',     '')     if     '/'     in
parsed_data['info'][1] else ''
```

```
                floor = parsed_data['info'][2]
                re_type = parsed_data['info'][3]
                year_built = parsed_data['info'][4].replace('г', '')
                bulding_type = parsed_data['info'][5]

            datetime_obj = datetime.datetime.now()
            number_of_storey = None
            if floor:
                try:
                    number_of_storey = int(floor.split()[-1].strip())
                    floor = int(floor.split()[0].strip())
                except:
                    pass

            price = extract_integers(parsed_data['price'])
            area = extract_integers(area)
            kitchenArea = extract_integers(kitchenArea)
            livingArea = extract_integers(livingArea)
            year_built = extract_integers(year_built)
            if 'однокомнатная' in rooms:
                rooms = 1
            else:
                rooms = extract_integers(rooms)

            Card = {
                'name': name,
                'district': district,
                'price': int(round(price)) if '$' in
parsed_data['price'] else None if price == 0 else int(round(price * 0.037593985)),
                'location': {
                    'latitude': lat,
                    'longitude': lon
                    },
                'description': description,
                'publisher': owner,
                'update_string': update_string,
                'redirect_link': redirect_link,
                'initial_company': initial_company,
                'otherObjects': {
                    'rubric': 'Долгосрочная аренда квартир, комнат'
                },
                'telephone': None,
```

```
                        'images': [],
                        'isApproved': True,
                        'isParsed': True,
                        'currency': '$',
                        'floor': floor if floor else None,
                        'area': area if area else None,
                        'kitchenArea': kitchenArea if kitchenArea else None,
                        'livingArea': livingArea if livingArea else None,
                        'bulding_type': bulding_type,
                        'year_built': year_built if year_built else None,
                        're_type': re_type,
                        'rooms': str(rooms) if rooms else None,
                        'owner': ObjectId('5e812796a8630ef4a7bc49f6'),
                        'created': datetime_obj,
                        'updated': datetime_obj,
                        'number_of_storey': number_of_storey if number_of_storey
else None,
                        'areUpdatedImages': False,
                        'isVerified': False
                  }
                  page_cards.append(Card)

            for card_num in range(len(page_cards)):
                  redirect_link = page_cards[card_num]['redirect_link']
                  current_url = ""
                  telephone = None
                  images = []
                  while True:
                        current_url = real_url(redirect_link)
                        if "https://flatfy.lun.ua" not in current_url:
                              break

                  telephone, images = get_media(current_url)
                  telephone = str(extract_integers(telephone))
                  if telephone and images:
                        page_cards[card_num]['telephone'] = telephone
                        page_cards[card_num]['images'] = images
                        page_cards[card_num]['isVerified'] = True if telephone
in verified_numbers else False
                  else:
                        page_cards[card_num] = None
```

```
                        if page_cards[card_num]:
                              print(current_url[:30], "...\t", telephone, sep="")
                              if   not   any(map(lambda   c:   telephone.startswith(c),
sms_block)):
                                    sms = SMS()
                                    sms.tel = telephone
                                    sms.analyze()

                              uploader.uploading(page_cards[card_num])

                  print(f"Page number [{number_page}] has successfully parsed")
            print("All data has successfully parsed")


      def get_media(current_url):
            LUA_SCRIPT = "function main(splash, args) \n\
            headers = { \n\
               [\"user-agent\"] = \"Mozilla/5.0 (X11; Ubuntu; Linux x86_64; rv:78.0)
Gecko/20100101 Firefox/78.0\", \n\
            } \n\
            splash:set_custom_headers(headers) \n\
            splash.images_enabled = false \n\
            assert(splash:go(args.url)) \n\
            assert(splash:wait(2)) \n"
            lua_end = "return { \n\
              html = splash:html(), \n\
            } \n\
            end"
            click_by_path = "assert(splash:runjs(\"var tel = document.evaluate('{}',
document, null, XPathResult.FIRST_ORDERED_NODE_TYPE, null).singleNodeValue; if (tel
!= null) {{tel.click();}}\")) \n assert(splash:wait(2)) \n"

            sites = path_num_tel.keys()

            for site in sites:
                  if site in current_url:
                        r = urlparse(current_url)
                        domain = r.netloc
                        if site in excepted_urls:
                              domain      =      domain[domain.index('.')+1:]      if
domain.count('.') == 2 else domain
                        index = f'https://{domain}'
```

```
                        path_click, path_to_tel = path_num_tel[site]
                        is_easy, image_click_path, images_container, replace_with =
path_to_images[site]
                        tag, attrs = images_container

                        if path_click:
                              LUA_SCRIPT                                          +=
click_by_path.format(path_click.replace('"', '\\"'))

                        if not is_easy:
                              LUA_SCRIPT                                          +=
click_by_path.format(image_click_path.replace('"', '\\"'))

                        LUA_SCRIPT += lua_end

                        try:
                              response = requests.post(
                                    url=SPLASH_SERVER_URL,
                                    headers=HEADERS,
                                    json={
                                          'lua_source': LUA_SCRIPT,
                                          'url': current_url,
                                    },
                                    timeout=20,
                              )

                              content = response.json()['html']

                              tree = lxml.html.document_fromstring(content)
                              telephone                                           =
prettify(tree.xpath(path_to_tel)[0].text_content().strip())

                              soup = BS(content, 'html.parser')
                              container = soup.find(tag, attrs)

                              images = {urljoin(index, img.get('src')) for img in
container.find_all('img')}
                              if site in url_in_style.keys():
                                    tag, attrs = url_in_style[site]
                                    images = container.find_all(tag, attrs)
```

```
                                images = {urljoin(index, get_url(img.get('style'))) for img in images}
                            if site == 'est.ua':
                                images = {a.get('href') for a in container.find_all('a', {'class': 'app__gallery-img'})}
                            if replace_with:
                                curr, new = replace_with
                                images = [img_url.replace(curr, new) for img_url in images if curr in img_url]
                            images = [img_url for img_url in images if img_url.startswith('http')]

                            return (
                                telephone,
                                images
                            )
                        except Timeout:
                            break

        return (
            None,
            []
        )
```

ВИСНОВКИ

В ході виконання роботи був розроблений сервіс для оренди нерухомості на основі парсингу даних об’яв зі сторонніх джерел:

- Парсери з 60 ресурсів з об’явами;
- Клієнтську частину веб-додатку, за допомогою якої користувачі взаємодіють з розробленою системою.

В результаті аналізу технологій парсингу даних були виявлені переваги і недоліки даної технології. В процесі розробки сервісу були створені парсери, які реалізують весь запланований на стадії проектування функціонал та користувальницький інтерфейс для взаємодії з цим функціоналом.

Розроблений сервіс володіє наступними характеристиками:

- безпека, що забезпечується збереженням об’яв не тільки у звичайній датабазі, а з паралельним збереження за допомогою смарт-контракта у блокчейн Binance Smart Chain та IPFS;
- швидкість наповнення бази даних новими об’явами;
- інформування за рахунок повідомлення нових користувачів за допомогою смс про те що їх об’яви зі сторонніх веб-сайтів були додані у нашу датабазу;

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Data-Oriented Parsing (Studies in Computational Linguistics) 1st Edition by Rens Bod (Editor), Remko Scha (Editor), Khalil Sima'an (Editor) 2003.
2. Parsing Techniques: A Practical Guide (Monographs in Computer Science) 2nd ed. 2008 Edition.
3. Learn XPath Fast: A beginner-friendly, exercise-based course for people who want to use XPath in Selenium, SQL Server, XQuery or anywhere else Paperback 2020.
4. Hands-On Web Scraping with Python: Perform advanced scraping operations using various Python libraries and tools such as Selenium, Regex, and others 2019.
5. Selenium Testing Tools Interview Questions You'll Most Likely Be Asked: Second Edition 2019.
6. Real Estate in New Reality published by KPMG 2020.

ДОДАТОК А

Код смарт-контракту для маркетплейсу з даними нерухомості:

pragma solidity ^0.6.6;

```
import "openzeppelin-solidity/contracts/token/ERC20/ERC20.sol";
import "openzeppelin-solidity/contracts/math/SafeMath.sol";

import "./PurchaseListener.sol";
import "./Ownable.sol";


interface IMarketplace {
    enum ProductState {
        NotDeployed,                // non-existent or deleted
        Deployed                    // created or redeployed
    }

    enum Currency {
        DATA,                       // "token wei" (10^-18 DATA)
        USD                         // attodollars (10^-18 USD)
    }

    enum WhitelistState{
        None,
        Pending,
        Approved,
        Rejected
    }
```

```
        function getSubscription(bytes32 productId, address subscriber) external view returns (bool isValid, uint endTimestamp);
```

```
            function getPriceInData(uint subscriptionSeconds, uint
price, Currency unit) external view returns (uint datacoinAmount);
    }
    interface IMarketplace1 is IMarketplace{
            function getProduct(bytes32 id) external view returns
(string memory name, address owner, address beneficiary, uint
pricePerSecond, Currency currency, uint minimumSubscriptionSeconds,
ProductState state);
    }
    interface IMarketplace2 is IMarketplace{
            function getProduct(bytes32 id) external view returns
(string memory name, address owner, address beneficiary, uint
pricePerSecond, Currency currency, uint minimumSubscriptionSeconds,
ProductState state, bool requiresWhitelist);
         function buyFor(bytes32 productId, uint subscriptionSeconds,
address recipient) external;
    }
    /**
     * @title Streamr Marketplace
     * @dev note about numbers:
     *   All prices and exchange rates are in "decimal fixed-point",
that is, scaled by 10^18, like ETH vs wei.
     *  Seconds are integers as usual.
     *
     * Next version TODO:
     *  - EIP-165 inferface definition; PurchaseListener
     */
    contract Marketplace is Ownable, IMarketplace2 {
        using SafeMath for uint256;

        // product events
```

```
        event ProductCreated(address indexed owner, bytes32 indexed
id, string name, address beneficiary, uint pricePerSecond, Currency
currency, uint minimumSubscriptionSeconds);
        event ProductUpdated(address indexed owner, bytes32 indexed
id, string name, address beneficiary, uint pricePerSecond, Currency
currency, uint minimumSubscriptionSeconds);
        event ProductDeleted(address indexed owner, bytes32 indexed
id, string name, address beneficiary, uint pricePerSecond, Currency
currency, uint minimumSubscriptionSeconds);
        event ProductImported(address indexed owner, bytes32 indexed
id, string name, address beneficiary, uint pricePerSecond, Currency
currency, uint minimumSubscriptionSeconds);
           event ProductRedeployed(address indexed owner, bytes32
indexed id, string name, address beneficiary, uint pricePerSecond,
Currency currency, uint minimumSubscriptionSeconds);
        event ProductOwnershipOffered(address indexed owner, bytes32
indexed id, address indexed to);
           event ProductOwnershipChanged(address indexed newOwner,
bytes32 indexed id, address indexed oldOwner);

        // subscription events
        event Subscribed(bytes32 indexed productId, address indexed
subscriber, uint endTimestamp);
          event NewSubscription(bytes32 indexed productId, address
indexed subscriber, uint endTimestamp);
            event SubscriptionExtended(bytes32 indexed productId,
address indexed subscriber, uint endTimestamp);
            event SubscriptionImported(bytes32 indexed productId,
address indexed subscriber, uint endTimestamp);
           event SubscriptionTransferred(bytes32 indexed productId,
address indexed from, address indexed to, uint secondsTransferred);

        // currency events
```

```
        event ExchangeRatesUpdated(uint timestamp, uint dataInUsd);

        // whitelist events
         event WhitelistRequested(bytes32 indexed productId, address
indexed subscriber);
          event WhitelistApproved(bytes32 indexed productId, address
indexed subscriber);
          event WhitelistRejected(bytes32 indexed productId, address
indexed subscriber);
        event WhitelistEnabled(bytes32 indexed productId);
        event WhitelistDisabled(bytes32 indexed productId);

        //txFee events
        event TxFeeChanged(uint256 indexed newTxFee);


        struct Product {
            bytes32 id;
            string name;
            address owner;
             address beneficiary;        // account where revenue is
directed to
            uint pricePerSecond;
            Currency priceCurrency;
            uint minimumSubscriptionSeconds;
            ProductState state;
              address newOwnerCandidate;  // Two phase hand-over to
minimize the chance that the product ownership is lost to a
non-existent address.
            bool requiresWhitelist;
            mapping(address => TimeBasedSubscription) subscriptions;
            mapping(address => WhitelistState) whitelist;
        }
```

```
        struct TimeBasedSubscription {
            uint endTimestamp;
        }

        /////////////// Marketplace lifecycle /////////////////

        ERC20 public datacoin;

        address public currencyUpdateAgent;
        IMarketplace1 prev_marketplace;
        uint256 public txFee;

                    constructor(address datacoinAddress, address
currencyUpdateAgentAddress, address prev_marketplace_address)
Ownable() public {
            _initialize(datacoinAddress, currencyUpdateAgentAddress,
prev_marketplace_address);
        }

            function _initialize(address datacoinAddress, address
currencyUpdateAgentAddress, address prev_marketplace_address)
internal {
            currencyUpdateAgent = currencyUpdateAgentAddress;
            datacoin = ERC20(datacoinAddress);
                                                prev_marketplace =
IMarketplace1(prev_marketplace_address);
        }

        ////////////////// Product management /////////////////

        mapping (bytes32 => Product) public products;
        /*
```

```
            checks this marketplace first, then the previous
        */
        function getProduct(bytes32 id) public override view returns
(string memory name, address owner, address beneficiary, uint
pricePerSecond, Currency currency, uint minimumSubscriptionSeconds,
ProductState state, bool requiresWhitelist) {
            (name, owner, beneficiary, pricePerSecond, currency,
minimumSubscriptionSeconds, state, requiresWhitelist) =
_getProductLocal(id);
            if (owner != address(0))
                return (name, owner, beneficiary, pricePerSecond,
currency, minimumSubscriptionSeconds, state, requiresWhitelist);
            (name, owner, beneficiary, pricePerSecond, currency,
minimumSubscriptionSeconds, state) = prev_marketplace.getProduct(id);
            return (name, owner, beneficiary, pricePerSecond,
currency, minimumSubscriptionSeconds, state, false);
        }

        /**
        checks only this marketplace, not the previous marketplace
         */

        function _getProductLocal(bytes32 id) internal view returns
(string memory name, address owner, address beneficiary, uint
pricePerSecond, Currency currency, uint minimumSubscriptionSeconds,
ProductState state, bool requiresWhitelist) {
            Product memory p = products[id];
            return (
                p.name,
                p.owner,
                p.beneficiary,
                p.pricePerSecond,
                p.priceCurrency,
```

```
                p.minimumSubscriptionSeconds,
                p.state,
                p.requiresWhitelist
            );
        }

        // also checks that p exists: p.owner == 0 for non-existent
products
        modifier onlyProductOwner(bytes32 productId) {
            (,address _owner,,,,,) = getProduct(productId);
            require(_owner != address(0), "error_notFound");
              require(_owner == msg.sender || owner == msg.sender,
"error_productOwnersOnly");
            _;
        }

        /**
          * Imports product details (but NOT subscription details)
from previous marketplace
         */
         function _importProductIfNeeded(bytes32 productId) internal
returns (bool imported){
            Product storage p = products[productId];
            if (p.id != 0x0) { return false; }
                  (string memory _name, address _owner, address
_beneficiary,    uint    _pricePerSecond,    IMarketplace1.Currency
_priceCurrency,          uint          _minimumSubscriptionSeconds,
IMarketplace1.ProductState                  _state)                =
prev_marketplace.getProduct(productId);
            if (_owner == address(0)) { return false; }
            p.id = productId;
            p.name = _name;
            p.owner = _owner;
```

```
            p.beneficiary = _beneficiary;
            p.pricePerSecond = _pricePerSecond;
            p.priceCurrency = _priceCurrency;
                                p.minimumSubscriptionSeconds    =
_minimumSubscriptionSeconds;
            p.state = _state;
                      emit  ProductImported(p.owner,  p.id,  p.name,
p.beneficiary,           p.pricePerSecond,          p.priceCurrency,
p.minimumSubscriptionSeconds);
            return true;
        }

            function  _importSubscriptionIfNeeded(bytes32  productId,
address subscriber) internal returns (bool imported) {
                                    bool    _productImported    =
_importProductIfNeeded(productId);

                // check that subscription didn't already exist in
current marketplace
             (Product storage product, TimeBasedSubscription storage
sub) = _getSubscriptionLocal(productId, subscriber);
            if (sub.endTimestamp != 0x0) { return false; }

                // check that subscription exists in the previous
marketplace(s)
                // only call prev_marketplace.getSubscription() if
product exists there
             // consider e.g. product created in current marketplace
but subscription still doesn't exist
            // if _productImported, it must have existed in previous
marketplace so no need to perform check
            if (!_productImported) {
```

```
                                    (,address  _owner_prev,,,,,)  =
prev_marketplace.getProduct(productId);
                if (_owner_prev == address(0)) { return false; }
            }
                                    (,   uint   _endTimestamp)   =
prev_marketplace.getSubscription(productId, subscriber);
            if (_endTimestamp == 0x0) { return false; }
                                product.subscriptions[subscriber]   =
TimeBasedSubscription(_endTimestamp);
                    emit SubscriptionImported(productId, subscriber,
_endTimestamp);
            return true;
        }
           function createProduct(bytes32 id, string memory name,
address beneficiary, uint pricePerSecond, Currency currency, uint
minimumSubscriptionSeconds) public whenNotHalted {
               _createProduct(id, name, beneficiary, pricePerSecond,
currency, minimumSubscriptionSeconds, false);
        }

            function createProductWithWhitelist(bytes32 id, string
memory name, address beneficiary, uint pricePerSecond, Currency
currency, uint minimumSubscriptionSeconds) public whenNotHalted {
               _createProduct(id, name, beneficiary, pricePerSecond,
currency, minimumSubscriptionSeconds, true);
            emit WhitelistEnabled(id);
        }


           function _createProduct(bytes32 id, string memory name,
address beneficiary, uint pricePerSecond, Currency currency, uint
minimumSubscriptionSeconds, bool requiresWhitelist) internal {
            require(id != 0x0, "error_nullProductId");
```

```
                                    require(pricePerSecond    >    0,
"error_freeProductsNotSupported");
            (,address _owner,,,,,,) = getProduct(id);
            require(_owner == address(0), "error_alreadyExists");
                products[id] = Product({id: id, name: name, owner:
msg.sender, beneficiary: beneficiary, pricePerSecond: pricePerSecond,
                 priceCurrency: currency, minimumSubscriptionSeconds:
minimumSubscriptionSeconds,       state:       ProductState.Deployed,
newOwnerCandidate:          address(0),          requiresWhitelist:
requiresWhitelist});
              emit ProductCreated(msg.sender, id, name, beneficiary,
pricePerSecond, currency, minimumSubscriptionSeconds);
        }

        /**
        * Stop offering the product
        */
               function   deleteProduct(bytes32   productId)   public
onlyProductOwner(productId) {
            _importProductIfNeeded(productId);
            Product storage p = products[productId];
                        require(p.state   ==   ProductState.Deployed,
"error_notDeployed");
            p.state = ProductState.NotDeployed;
                  emit  ProductDeleted(p.owner,  productId,  p.name,
p.beneficiary,          p.pricePerSecond,          p.priceCurrency,
p.minimumSubscriptionSeconds);
        }

        /**
        * Return product to market
        */
```

```
    function redeployProduct(bytes32 productId) public
onlyProductOwner(productId) {
        _importProductIfNeeded(productId);
        Product storage p = products[productId];
            require(p.state == ProductState.NotDeployed,
"error_mustBeNotDeployed");
        p.state = ProductState.Deployed;
            emit ProductRedeployed(p.owner, productId, p.name,
p.beneficiary, p.pricePerSecond, p.priceCurrency,
p.minimumSubscriptionSeconds);
    }

    function updateProduct(bytes32 productId, string memory
name, address beneficiary, uint pricePerSecond, Currency currency,
uint minimumSubscriptionSeconds, bool redeploy) public
onlyProductOwner(productId) {
        require(pricePerSecond > 0,
"error_freeProductsNotSupported");
        _importProductIfNeeded(productId);
        Product storage p = products[productId];
        p.name = name;
        p.beneficiary = beneficiary;
        p.pricePerSecond = pricePerSecond;
        p.priceCurrency = currency;
        p.minimumSubscriptionSeconds =
minimumSubscriptionSeconds;
        emit ProductUpdated(p.owner, p.id, name, beneficiary,
pricePerSecond, currency, minimumSubscriptionSeconds);
        if (redeploy) {
            redeployProduct(productId);
        }
    }
```

```
        /**
          * Changes ownership of the product. Two phase hand-over minimizes the chance that the product ownership is lost to a non-existent address.
        */
          function offerProductOwnership(bytes32 productId, address newOwnerCandidate) public onlyProductOwner(productId) {
            _importProductIfNeeded(productId);
                  // that productId exists is already checked in onlyProductOwner
                            products[productId].newOwnerCandidate = newOwnerCandidate;
             emit ProductOwnershipOffered(products[productId].owner, productId, newOwnerCandidate);
        }

        /**
          * Changes ownership of the product. Two phase hand-over minimizes the chance that the product ownership is lost to a non-existent address.
        */
           function claimProductOwnership(bytes32 productId) public whenNotHalted {
            _importProductIfNeeded(productId);
             // also checks that productId exists (newOwnerCandidate is zero for non-existent)
            Product storage p = products[productId];
                        require(msg.sender == p.newOwnerCandidate, "error_notPermitted");
                emit ProductOwnershipChanged(msg.sender, productId, p.owner);
            p.owner = msg.sender;
            p.newOwnerCandidate = address(0);
```

```
        }

        /////////////// Whitelist management ///////////////

            function setRequiresWhitelist(bytes32 productId, bool
_requiresWhitelist) public onlyProductOwner(productId) {
            _importProductIfNeeded(productId);
            Product storage p = products[productId];
            require(p.id != 0x0, "error_notFound");
            p.requiresWhitelist = _requiresWhitelist;
            if (_requiresWhitelist) {
                emit WhitelistEnabled(productId);
            } else {
                emit WhitelistDisabled(productId);
            }
        }

             function whitelistApprove(bytes32 productId, address
subscriber) public onlyProductOwner(productId) {
            _importProductIfNeeded(productId);
            Product storage p = products[productId];
            require(p.id != 0x0, "error_notFound");
                                        require(p.requiresWhitelist,
"error_whitelistNotEnabled");
            p.whitelist[subscriber] = WhitelistState.Approved;
            emit WhitelistApproved(productId, subscriber);
        }

              function whitelistReject(bytes32 productId, address
subscriber) public onlyProductOwner(productId) {
            _importProductIfNeeded(productId);
            Product storage p = products[productId];
            require(p.id != 0x0, "error_notFound");
```

```
                                        require(p.requiresWhitelist,
"error_whitelistNotEnabled");
            p.whitelist[subscriber] = WhitelistState.Rejected;
            emit WhitelistRejected(productId, subscriber);
        }

        function whitelistRequest(bytes32 productId) public {
            _importProductIfNeeded(productId);
            Product storage p = products[productId];
            require(p.id != 0x0, "error_notFound");
                                        require(p.requiresWhitelist,
"error_whitelistNotEnabled");
             require(p.whitelist[msg.sender] == WhitelistState.None,
"error_whitelistRequestAlreadySubmitted");
            p.whitelist[msg.sender] = WhitelistState.Pending;
            emit WhitelistRequested(productId, msg.sender);
        }

            function getWhitelistState(bytes32 productId, address
subscriber) public view returns (WhitelistState wlstate) {
            (, address _owner,,,,,) = getProduct(productId);
            require(_owner != address(0), "error_notFound");
              // if product is not local (maybe in old marketplace)
this will return 0 (WhitelistState.None)
            Product storage p = products[productId];
            return p.whitelist[subscriber];
        }

        /////////////// Subscription management ///////////////

             function getSubscription(bytes32 productId, address
subscriber) public override view returns (bool isValid, uint
endTimestamp) {
```

```
            (,address _owner,,,,,,) = _getProductLocal(productId);
            if (_owner == address(0)) {
                                                                return
prev_marketplace.getSubscription(productId,subscriber);
            }

                        (,  TimeBasedSubscription  storage  sub)  =
_getSubscriptionLocal(productId, subscriber);
            if (sub.endTimestamp == 0x0) {
                  // only call prev_marketplace.getSubscription() if
product exists in previous marketplace too
                                    (,address  _owner_prev,,,,,)  =
prev_marketplace.getProduct(productId);
                if (_owner_prev != address(0)) {
                                                                return
prev_marketplace.getSubscription(productId,subscriber);
                }
            }
            return (_isValid(sub), sub.endTimestamp);
        }

          function getSubscriptionTo(bytes32 productId) public view
returns (bool isValid, uint endTimestamp) {
            return getSubscription(productId, msg.sender);
        }

        /**
            * Checks if the given address currently has a valid
subscription
         * @param productId to check
         * @param subscriber to check
         */
```

```
            function hasValidSubscription(bytes32 productId, address
subscriber) public view returns (bool isValid) {
                (isValid,) = getSubscription(productId, subscriber);
            }

            /**
             * Enforces payment rules, triggers PurchaseListener event
             */
              function _subscribe(bytes32 productId, uint addSeconds,
address subscriber, bool requirePayment) internal {
                _importSubscriptionIfNeeded(productId, subscriber);
                     (Product storage p, TimeBasedSubscription storage
oldSub) = _getSubscriptionLocal(productId, subscriber);
                          require(p.state == ProductState.Deployed,
"error_notDeployed");
                 require(!p.requiresWhitelist || p.whitelist[subscriber]
== WhitelistState.Approved, "error_whitelistNotAllowed");
                uint endTimestamp;

                if (oldSub.endTimestamp > block.timestamp) {
                    require(addSeconds > 0, "error_topUpTooSmall");
                    endTimestamp = oldSub.endTimestamp.add(addSeconds);
                    oldSub.endTimestamp = endTimestamp;
                            emit SubscriptionExtended(p.id, subscriber,
endTimestamp);
                } else {
                     require(addSeconds >= p.minimumSubscriptionSeconds,
"error_newSubscriptionTooSmall");
                    endTimestamp = block.timestamp.add(addSeconds);
                                TimeBasedSubscription memory newSub =
TimeBasedSubscription(endTimestamp);
                    p.subscriptions[subscriber] = newSub;
```

```
                                emit  NewSubscription(p.id,  subscriber,
endTimestamp);
            }
            emit Subscribed(p.id, subscriber, endTimestamp);

            uint256 price = 0;
            uint256 fee = 0;
            address recipient = p.beneficiary;
            if (requirePayment) {
                price = getPriceInData(addSeconds, p.pricePerSecond,
p.priceCurrency);
                fee = txFee.mul(price).div(1 ether);
                require(datacoin.transferFrom(msg.sender, recipient,
price.sub(fee)), "error_paymentFailed");
                if (fee > 0) {
                    require(datacoin.transferFrom(msg.sender, owner,
fee), "error_paymentFailed");
                }
            }

            uint256 codeSize;
               assembly { codeSize := extcodesize(recipient) }  //
solhint-disable-line no-inline-assembly
            if (codeSize > 0) {
                // solhint-disable-next-line avoid-low-level-calls


    pragma solidity ^0.4.24;

    // File: contracts/upgradable/ProxyStorage.sol

    contract ProxyStorage {
```

```
  /**
   * Current contract to which we are proxing
   */
  address public currentContract;
  address public proxyOwner;
}

// File: contracts/upgradable/OwnableStorage.sol

contract OwnableStorage {

  address public owner;

  constructor() internal {
    owner = msg.sender;
  }

}

// File: erc821/contracts/AssetRegistryStorage.sol

contract AssetRegistryStorage {

  string internal _name;
  string internal _symbol;
  string internal _description;

  /**
   * Stores the total count of assets managed by this registry
   */
  uint256 internal _count;

  /**
```

```
        * Stores an array of assets owned by a given account
        */
       mapping(address => uint256[]) internal _assetsOf;

       /**
        * Stores the current holder of an asset
        */
       mapping(uint256 => address) internal _holderOf;

       /**
         * Stores the index of an asset in the `_assetsOf` array of its holder
        */
       mapping(uint256 => uint256) internal _indexOfAsset;

       /**
        * Stores the data associated with an asset
        */
       mapping(uint256 => string) internal _assetData;

       /**
         * For a given account, for a given operator, store whether that operator is
        * allowed to transfer and modify assets on behalf of them.
        */
         mapping(address => mapping(address => bool)) internal _operators;

       /**
        * Approval array
        */
       mapping(uint256 => address) internal _approval;
     }
```

```
// File: contracts/estate/IEstateRegistry.sol

contract IEstateRegistry {
   function mint(address to, string metadata) external returns (uint256);
    function ownerOf(uint256 _tokenId) public view returns (address _owner); // from ERC721

  // Events

  event CreateEstate(
    address indexed _owner,
    uint256 indexed _estateId,
    string _data
  );

  event AddLand(
    uint256 indexed _estateId,
    uint256 indexed _landId
  );

  event RemoveLand(
    uint256 indexed _estateId,
    uint256 indexed _landId,
    address indexed _destinatary
  );

  event Update(
    uint256 indexed _assetId,
    address indexed _holder,
    address indexed _operator,
    string _data
```

```
    );

    event UpdateOperator(
      uint256 indexed _estateId,
      address indexed _operator
    );

    event UpdateManager(
      address indexed _owner,
      address indexed _operator,
      address indexed _caller,
      bool _approved
    );

    event SetLANDRegistry(
      address indexed _registry
    );

    event SetEstateLandBalanceToken(
      address indexed _previousEstateLandBalance,
      address indexed _newEstateLandBalance
    );
  }

  // File: contracts/minimeToken/IMinimeToken.sol

  interface IMiniMeToken {
  ////////////////
  // Generate and destroy tokens
  ////////////////

      /// @notice Generates `_amount` tokens that are assigned to
`_owner`
```

```
        /// @param _owner The address that will be assigned the new
tokens
        /// @param _amount The quantity of tokens generated
        /// @return True if the tokens are generated correctly
            function generateTokens(address _owner, uint _amount)
external returns (bool);


        /// @notice Burns `_amount` tokens from `_owner`
        /// @param _owner The address that will lose the tokens
        /// @param _amount The quantity of tokens to burn
        /// @return True if the tokens are burned correctly
            function destroyTokens(address _owner, uint _amount)
external returns (bool);

          /// @param _owner The address that's balance is being
requested
        /// @return The balance of `_owner` at the current block
          function balanceOf(address _owner) external view returns
(uint256 balance);

         event Transfer(address indexed _from, address indexed _to,
uint256 _amount);
    }

    // File: contracts/land/LANDStorage.sol

    contract LANDStorage {
      mapping (address => uint) public latestPing;

                    uint256       constant       clearLow       =
0xffffffffffffffffffffffffffffffff00000000000000000000000000000000;
```

```
                    uint256        constant        clearHigh        =
0x00000000000000000000000000000000ffffffffffffffffffffffffffffffff;
        uint256 constant factor = 0x100000000000000000000000000000000;

                mapping      (address      =>      bool)      internal
_deprecated_authorizedDeploy;

        mapping (uint256 => address) public updateOperator;

        IEstateRegistry public estateRegistry;

        mapping (address => bool) public authorizedDeploy;

           mapping(address   =>   mapping(address   =>   bool))   public
updateManager;

        // Land balance minime token
        IMiniMeToken public landBalance;

        // Registered balance accounts
        mapping(address => bool) public registeredBalance;
    }

    // File: contracts/Storage.sol

    contract     Storage     is     ProxyStorage,     OwnableStorage,
AssetRegistryStorage, LANDStorage {
    }

    // File: contracts/upgradable/Ownable.sol

    contract Ownable is Storage {
```

```
    event OwnerUpdate(address _prevOwner, address _newOwner);

    modifier onlyOwner {
      assert(msg.sender == owner);
      _;
    }

    function transferOwnership(address _newOwner) public onlyOwner
{
      require(_newOwner != owner, "Cannot transfer to yourself");
      owner = _newOwner;
    }
  }

  // File: contracts/upgradable/IApplication.sol

  contract IApplication {
    function initialize(bytes data) public;
  }

  // File: openzeppelin-solidity/contracts/math/SafeMath.sol

  /**
   * @title SafeMath
   * @dev Math operations with safety checks that throw on error
   */
  library SafeMath {

    /**
    * @dev Multiplies two numbers, throws on overflow.
    */
     function mul(uint256 _a, uint256 _b) internal pure returns
(uint256 c) {
```

```
        // Gas optimization: this is cheaper than asserting 'a' not
being zero, but the
        // benefit is lost if 'b' is also tested.
                                                    //           See:
https://github.com/OpenZeppelin/openzeppelin-solidity/pull/522
        if (_a == 0) {
          return 0;
        }

        c = _a * _b;
        assert(c / _a == _b);
        return c;
      }

      /**
       * @dev Integer division of two numbers, truncating the
quotient.
      */
       function div(uint256 _a, uint256 _b) internal pure returns
(uint256) {
         // assert(_b > 0); // Solidity automatically throws when
dividing by 0
        // uint256 c = _a / _b;
         // assert(_a == _b * c + _a % _b); // There is no case in
which this doesn't hold
        return _a / _b;
      }

      /**
       * @dev Subtracts two numbers, throws on overflow (i.e. if
subtrahend is greater than minuend).
      */
```

```
        function sub(uint256 _a, uint256 _b) internal pure returns
(uint256) {
          assert(_b <= _a);
          return _a - _b;
        }

        /**
        * @dev Adds two numbers, throws on overflow.
        */
        function add(uint256 _a, uint256 _b) internal pure returns
(uint256 c) {
          c = _a + _b;
          assert(c >= _a);
          return c;
        }
      }

      // File: erc821/contracts/IERC721Base.sol

      interface IERC721Base {
        function totalSupply() external view returns (uint256);

        // function exists(uint256 assetId) external view returns
(bool);
        function ownerOf(uint256 assetId) external view returns
(address);

        function balanceOf(address holder) external view returns
(uint256);

        function safeTransferFrom(address from, address to, uint256
assetId) external;
```

```
        function safeTransferFrom(address from, address to, uint256 assetId, bytes userData) external;

        function transferFrom(address from, address to, uint256 assetId) external;

       function approve(address operator, uint256 assetId) external;
        function setApprovalForAll(address operator, bool authorized) external;

        function getApprovedAddress(uint256 assetId) external view returns (address);
          function isApprovedForAll(address assetHolder, address operator) external view returns (bool);

        function isAuthorized(address operator, uint256 assetId) external view returns (bool);

       /**
       * @dev Deprecated transfer event. Now we use the standard with three parameters
        * It is only used in the ABI to get old transfer events. Do not remove
       */
       event Transfer(
         address indexed from,
         address indexed to,
         uint256 indexed assetId,
         address operator,
         bytes userData,
         bytes operatorData
       );
       /**
```

```
         * @dev Deprecated transfer event. Now we use the standard
with three parameters
         * It is only used in the ABI to get old transfer events. Do
not remove
         */
        event Transfer(
          address indexed from,
          address indexed to,
          uint256 indexed assetId,
          address operator,
          bytes userData
        );
        event Transfer(
          address indexed from,
          address indexed to,
          uint256 indexed assetId
        );
        event ApprovalForAll(
          address indexed holder,
          address indexed operator,
          bool authorized
        );
        event Approval(
          address indexed owner,
          address indexed operator,
          uint256 indexed assetId
        );
      }

      // File: erc821/contracts/IERC721Receiver.sol

      interface IERC721Receiver {
        function onERC721Received(
```

```
        address _operator,
        address _from,
        uint256 _tokenId,
        bytes   _userData
      ) external returns (bytes4);
    }

    // File: erc821/contracts/ERC165.sol

    interface ERC165 {
       function supportsInterface(bytes4 interfaceID) external view
returns (bool);
    }

    // File: erc821/contracts/ERC721Base.sol

    contract ERC721Base is AssetRegistryStorage, IERC721Base, ERC165
{
      using SafeMath for uint256;

                                  //             Equals             to
`bytes4(keccak256("onERC721Received(address,address,uint256,bytes)"))
`
      bytes4 private constant ERC721_RECEIVED = 0x150b7a02;

      bytes4 private constant InterfaceId_ERC165 = 0x01ffc9a7;
      /*
       * 0x01ffc9a7 ===
       *   bytes4(keccak256('supportsInterface(bytes4)'))
       */

      bytes4 private constant Old_InterfaceId_ERC721 = 0x7c0633c6;
      bytes4 private constant InterfaceId_ERC721 = 0x80ac58cd;
```

```
        /*
        * 0x80ac58cd ===
        *   bytes4(keccak256('balanceOf(address)')) ^
        *   bytes4(keccak256('ownerOf(uint256)')) ^
        *   bytes4(keccak256('approve(address,uint256)')) ^
        *   bytes4(keccak256('getApproved(uint256)')) ^
        *   bytes4(keccak256('setApprovalForAll(address,bool)')) ^
        *   bytes4(keccak256('isApprovedForAll(address,address)')) ^
                                                                    *
bytes4(keccak256('transferFrom(address,address,uint256)')) ^
                                                                    *
bytes4(keccak256('safeTransferFrom(address,address,uint256)')) ^
                                                                    *
bytes4(keccak256('safeTransferFrom(address,address,uint256,bytes)'))
        */

       //
       // Global Getters
       //

       /**
        * @dev Gets the total amount of assets stored by the contract
        * @return uint256 representing the total amount of assets
        */
       function totalSupply() external view returns (uint256) {
         return _totalSupply();
       }
       function _totalSupply() internal view returns (uint256) {
         return _count;
       }

       //
       // Asset-centric getter functions
```

```
    //

    /**
     * @dev Queries what address owns an asset. This method does not throw.
     * In order to check if the asset exists, use the `exists` function or check if the
     * return value of this call is `0`.
     * @return uint256 the assetId
     */
     function ownerOf(uint256 assetId) external view returns (address) {
     return _ownerOf(assetId);
    }
     function _ownerOf(uint256 assetId) internal view returns (address) {
     return _holderOf[assetId];
    }

    //
    // Holder-centric getter functions
    //
    /**
     * @dev Gets the balance of the specified address
     * @param owner address to query the balance of
     * @return uint256 representing the amount owned by the passed address
     */
     function balanceOf(address owner) external view returns (uint256) {
     return _balanceOf(owner);
    }
```

```
        function _balanceOf(address owner) internal view returns
(uint256) {
        return _assetsOf[owner].length;
      }

      //
      // Authorization getters
      //

      /**
       * @dev Query whether an address has been authorized to move
any assets on behalf of someone else
       * @param operator the address that might be authorized
          * @param assetHolder the address that provided the
authorization
       * @return bool true if the operator has been authorized to
move any assets
       */
         function isApprovedForAll(address assetHolder, address
operator)
        external view returns (bool)
      {
        return _isApprovedForAll(assetHolder, operator);
      }
         function _isApprovedForAll(address assetHolder, address
operator)
        internal view returns (bool)
      {
        return _operators[assetHolder][operator];
      }

      /**
```

```
        * @dev Query what address has been particularly authorized to
move an asset
        * @param assetId the asset to be queried for
         * @return bool true if the asset has been approved by the
holder
        */
        function getApproved(uint256 assetId) external view returns
(address) {
         return _getApprovedAddress(assetId);
       }
        function getApprovedAddress(uint256 assetId) external view
returns (address) {
         return _getApprovedAddress(assetId);
       }
        function _getApprovedAddress(uint256 assetId) internal view
returns (address) {
         return _approval[assetId];
       }

       /**
        * @dev Query if an operator can move an asset.
        * @param operator the address that might be authorized
         * @param assetId the asset that has been `approved` for
transfer
         * @return bool true if the asset has been approved by the
holder
        */
         function isAuthorized(address operator, uint256 assetId)
external view returns (bool) {
         return _isAuthorized(operator, assetId);
       }
         function _isAuthorized(address operator, uint256 assetId)
internal view returns (bool)
```

```
    {
      require(operator != 0);
      address owner = _ownerOf(assetId);
      if (operator == owner) {
        return true;
      }
              return   _isApprovedForAll(owner,   operator)   ||
_getApprovedAddress(assetId) == operator;
    }

    //
    // Authorization
    //

    /**
       * @dev Authorize a third party operator to manage (send)
msg.sender's asset
     * @param operator address to be approved
     * @param authorized bool set to true to authorize, false to
withdraw authorization
     */
    function setApprovalForAll(address operator, bool authorized)
external {
      return _setApprovalForAll(operator, authorized);
    }
    function _setApprovalForAll(address operator, bool authorized)
internal {
      if (authorized) {
        require(!_isApprovedForAll(msg.sender, operator));
        _addAuthorization(operator, msg.sender);
      } else {
        require(_isApprovedForAll(msg.sender, operator));
        _clearAuthorization(operator, msg.sender);
```

```
        }
        emit ApprovalForAll(msg.sender, operator, authorized);
      }

      /**
       * @dev Authorize a third party operator to manage one particular asset
       * @param operator address to be approved
       * @param assetId asset to approve
       */
      function approve(address operator, uint256 assetId) external {
        address holder = _ownerOf(assetId);
        require(msg.sender == holder || _isApprovedForAll(msg.sender, holder));
        require(operator != holder);

        if (_getApprovedAddress(assetId) != operator) {
          _approval[assetId] = operator;
          emit Approval(holder, operator, assetId);
        }
      }

      function _addAuthorization(address operator, address holder) private {
        _operators[holder][operator] = true;
      }

      function _clearAuthorization(address operator, address holder) private {
        _operators[holder][operator] = false;
      }

      //
```

```
        // Internal Operations
        //

        function _addAssetTo(address to, uint256 assetId) internal {
          _holderOf[assetId] = to;

          uint256 length = _balanceOf(to);

          _assetsOf[to].push(assetId);

          _indexOfAsset[assetId] = length;

          _count = _count.add(1);
        }

          function _removeAssetFrom(address from, uint256 assetId)
internal {
          uint256 assetIndex = _indexOfAsset[assetId];
          uint256 lastAssetIndex = _balanceOf(from).sub(1);
          uint256 lastAssetId = _assetsOf[from][lastAssetIndex];

          _holderOf[assetId] = 0;

            // Insert the last asset into the position previously
occupied by the asset to be removed
          _assetsOf[from][assetIndex] = lastAssetId;

          // Resize the array
          _assetsOf[from][lastAssetIndex] = 0;
          _assetsOf[from].length--;

           // Remove the array if no more assets are owned to prevent
pollution
```

```
        if (_assetsOf[from].length == 0) {
          delete _assetsOf[from];
        }

        // Update the index of positions for the asset
        _indexOfAsset[assetId] = 0;
        _indexOfAsset[lastAssetId] = assetIndex;

        _count = _count.sub(1);
      }

        function _clearApproval(address holder, uint256 assetId)
internal {
        if (_ownerOf(assetId) == holder && _approval[assetId] != 0)
{
          _approval[assetId] = 0;
          emit Approval(holder, 0, assetId);
        }
      }

      //
      // Supply-altering functions
      //

        function _generate(uint256 assetId, address beneficiary)
internal {
        require(_holderOf[assetId] == 0);

        _addAssetTo(beneficiary, assetId);

        emit Transfer(0, beneficiary, assetId);
      }
```

```
function _destroy(uint256 assetId) internal {
  address holder = _holderOf[assetId];
  require(holder != 0);

  _removeAssetFrom(holder, assetId);

  emit Transfer(holder, 0, assetId);
}

//
// Transaction related operations
//

modifier onlyHolder(uint256 assetId) {
  require(_ownerOf(assetId) == msg.sender);
  _;
}

modifier onlyAuthorized(uint256 assetId) {
  require(_isAuthorized(msg.sender, assetId));
  _;
}

modifier isCurrentOwner(address from, uint256 assetId) {
  require(_ownerOf(assetId) == from);
  _;
}

modifier isDestinataryDefined(address destinatary) {
  require(destinatary != 0);
  _;
}
```

```
    modifier destinataryIsNotHolder(uint256 assetId, address to) {
      require(_ownerOf(assetId) != to);
      _;
    }

    /**
     * @dev Alias of `safeTransferFrom(from, to, assetId, '')`
     *
     * @param from address that currently owns an asset
     * @param to address to receive the ownership of the asset
     * @param assetId uint256 ID of the asset to be transferred
     */
     function safeTransferFrom(address from, address to, uint256 assetId) external {
      return _doTransferFrom(from, to, assetId, '', true);
    }

    /**
     * @dev Securely transfers the ownership of a given asset from one address to
     * another address, calling the method `onNFTReceived` on the target address if
     * there's code associated with it
     *
     * @param from address that currently owns an asset
     * @param to address to receive the ownership of the asset
     * @param assetId uint256 ID of the asset to be transferred
     * @param userData bytes arbitrary user information to attach to this transfer
     */
     function safeTransferFrom(address from, address to, uint256 assetId, bytes userData) external {
      return _doTransferFrom(from, to, assetId, userData, true);
```

```
    }

    /**
     * @dev Transfers the ownership of a given asset from one address to another address
     * Warning! This function does not attempt to verify that the target address can send
     * tokens.
     *
     * @param from address sending the asset
     * @param to address to receive the ownership of the asset
     * @param assetId uint256 ID of the asset to be transferred
     */
     function transferFrom(address from, address to, uint256 assetId) external {
     return _doTransferFrom(from, to, assetId, '', false);
    }

    function _doTransferFrom(
      address from,
      address to,
      uint256 assetId,
      bytes userData,
      bool doCheck
    )
      onlyAuthorized(assetId)
      internal
    {
      _moveToken(from, to, assetId, userData, doCheck);
    }

    function _moveToken(
      address from,
```

```
    address to,
    uint256 assetId,
    bytes userData,
    bool doCheck
  )
    isDestinataryDefined(to)
    destinataryIsNotHolder(assetId, to)
    isCurrentOwner(from, assetId)
    private
  {
    address holder = _holderOf[assetId];
    _clearApproval(holder, assetId);
    _removeAssetFrom(holder, assetId);
    _addAssetTo(to, assetId);
    emit Transfer(holder, to, assetId);

    if (doCheck && _isContract(to)) {
                                          //       Equals       to
`bytes4(keccak256("onERC721Received(address,address,uint256,bytes)"))
      require(
        IERC721Receiver(to).onERC721Received(
          msg.sender, holder, assetId, userData
        ) == ERC721_RECEIVED
      );
    }
  }

  /**
   * Internal function that moves an asset from one holder to
another
   */

  /**
```

```
            * @dev Returns `true` if the contract implements
`interfaceID` and `interfaceID` is not 0xffffffff, `false` otherwise
         * @param  _interfaceID The interface identifier, as specified
in ERC-165
         */
        function supportsInterface(bytes4 _interfaceID) external view
returns (bool) {

          if (_interfaceID == 0xffffffff) {
            return false;
          }
          return _interfaceID == InterfaceId_ERC165 || _interfaceID ==
Old_InterfaceId_ERC721 || _interfaceID == InterfaceId_ERC721;
        }

        //
        // Utilities
        //

          function  _isContract(address  addr)  internal  view  returns
(bool) {
          uint size;
          assembly { size := extcodesize(addr) }
          return size > 0;
        }
      }

      // File: erc821/contracts/IERC721Enumerable.sol

      contract IERC721Enumerable {

        /**
         * @notice Enumerate active tokens
```

```
        * @dev Throws if `index` >= `totalSupply()`, otherwise SHALL
NOT throw.
        * @param index A counter less than `totalSupply()`
        * @return The identifier for the `index`th asset, (sort order
not
        *  specified)
        */
       // TODO (eordano): Not implemented
        // function tokenByIndex(uint256 index) public view returns
(uint256 _assetId);

       /**
        * @notice Count of owners which own at least one asset
        *  Must not throw.
        * @return A count of the number of owners which own asset
        */
       // TODO (eordano): Not implemented
        // function countOfOwners() public view returns (uint256
_count);

       /**
        * @notice Enumerate owners
        * @dev Throws if `index` >= `countOfOwners()`, otherwise must
not throw.
        * @param index A counter less than `countOfOwners()`
        * @return The address of the `index`th owner (sort order not
specified)
        */
       // TODO (eordano): Not implemented
        // function ownerByIndex(uint256 index) public view returns
(address owner);

       /**
```

```
        * @notice Get all tokens of a given address
        * @dev This is not intended to be used on-chain
        * @param owner address of the owner to query
        * @return a list of all assetIds of a user
        */
         function tokensOf(address owner) external view returns
(uint256[]);

       /**
        * @notice Enumerate tokens assigned to an owner
        * @dev Throws if `index` >= `balanceOf(owner)` or if
        *  `owner` is the zero address, representing invalid assets.
        *  Otherwise this must not throw.
         * @param owner An address where we are interested in assets
owned by them
        * @param index A counter less than `balanceOf(owner)`
         * @return The identifier for the `index`th asset assigned to
`owner`,
        *   (sort order not specified)
        */
       function tokenOfOwnerByIndex(
         address owner, uint256 index
       ) external view returns (uint256 tokenId);
     }

     // File: erc821/contracts/ERC721Enumerable.sol

     contract     ERC721Enumerable     is     AssetRegistryStorage,
IERC721Enumerable {

       /**
        * @notice Get all tokens of a given address
        * @dev This is not intended to be used on-chain
```

```
        * @param owner address of the owner to query
        * @return a list of all assetIds of a user
        */
         function tokensOf(address owner) external view returns
(uint256[]) {
         return _assetsOf[owner];
       }

       /**
        * @notice Enumerate tokens assigned to an owner
        * @dev Throws if `index` >= `balanceOf(owner)` or if
        *  `owner` is the zero address, representing invalid assets.
        *  Otherwise this must not throw.
         * @param owner An address where we are interested in assets
owned by them
        * @param index A counter less than `balanceOf(owner)`
         * @return The identifier for the `index`th asset assigned to
`owner`,
        *   (sort order not specified)
        */
       function tokenOfOwnerByIndex(
         address owner, uint256 index
       )
         external
         view
         returns (uint256 assetId)
       {
         require(index < _assetsOf[owner].length);
         require(index < (1<<127));
         return _assetsOf[owner][index];
       }

     }
```

```
// File: erc821/contracts/IERC721Metadata.sol

contract IERC721Metadata {

  /**
   * @notice A descriptive name for a collection of NFTs in this
contract
   */
  function name() external view returns (string);

  /**
   * @notice An abbreviated name for NFTs in this contract
   */
  function symbol() external view returns (string);

  /**
   * @notice A description of what this DAR is used for
   */
  function description() external view returns (string);

  /**
   * Stores arbitrary info about a token
   */
   function tokenMetadata(uint256 assetId) external view returns
(string);
}

// File: erc821/contracts/ERC721Metadata.sol

contract ERC721Metadata is AssetRegistryStorage, IERC721Metadata
{
  function name() external view returns (string) {
```

```
        return _name;
      }
      function symbol() external view returns (string) {
        return _symbol;
      }
      function description() external view returns (string) {
        return _description;
      }
      function tokenMetadata(uint256 assetId) external view returns
(string) {
        return _assetData[assetId];
      }
      function _update(uint256 assetId, string data) internal {
        _assetData[assetId] = data;
      }
    }

    // File: erc821/contracts/FullAssetRegistry.sol

    contract FullAssetRegistry is ERC721Base, ERC721Enumerable,
ERC721Metadata {
      constructor() public {
      }

      /**
       * @dev Method to check if an asset identified by the given id
exists under this DAR.
       * @return uint256 the assetId
       */
      function exists(uint256 assetId) external view returns (bool)
{
        return _exists(assetId);
      }
```

```
    function _exists(uint256 assetId) internal view returns (bool)
{
      return _holderOf[assetId] != 0;
    }

    function decimals() external pure returns (uint256) {
      return 0;
    }
  }

  // File: contracts/land/ILANDRegistry.sol

  interface ILANDRegistry {

    // LAND can be assigned by the owner
     function assignNewParcel(int x, int y, address beneficiary)
external;
      function assignMultipleParcels(int[] x, int[] y, address
beneficiary) external;

    // After one year, LAND can be claimed from an inactive public
key
    function ping() external;

    // LAND-centric getters
     function encodeTokenId(int x, int y) external pure returns
(uint256);
    function decodeTokenId(uint value) external pure returns (int,
int);
    function exists(int x, int y) external view returns (bool);
     function ownerOfLand(int x, int y) external view returns
(address);
```

```
        function ownerOfLandMany(int[] x, int[] y) external view
returns (address[]);
        function landOf(address owner) external view returns (int[],
int[]);
        function landData(int x, int y) external view returns
(string);

    // Transfer LAND
    function transferLand(int x, int y, address to) external;
        function transferManyLand(int[] x, int[] y, address to)
external;

    // Update LAND
    function updateLandData(int x, int y, string data) external;
        function updateManyLandData(int[] x, int[] y, string data)
external;

    // Authorize an updateManager to manage parcel data
        function setUpdateManager(address _owner, address _operator,
bool _approved) external;

    // Events

    event Update(
      uint256 indexed assetId,
      address indexed holder,
      address indexed operator,
      string data
    );

    event UpdateOperator(
      uint256 indexed assetId,
      address indexed operator
```

```
    );

    event UpdateManager(
      address indexed _owner,
      address indexed _operator,
      address indexed _caller,
      bool _approved
    );

    event DeployAuthorized(
      address indexed _caller,
      address indexed _deployer
    );

    event DeployForbidden(
      address indexed _caller,
      address indexed _deployer
    );

    event SetLandBalanceToken(
      address indexed _previousLandBalance,
      address indexed _newLandBalance
    );
  }

  // File: contracts/metadata/IMetadataHolder.sol

  contract IMetadataHolder is ERC165 {
    function getMetadata(uint256 /* assetId */) external view
returns (string);
  }
```