На вибір даємо виконати одне з тестових завдань.

Варіант 1

Опис

Ми працюємо над новим дизайном і хочемо протестувати гіпотезу, використовуючи АБ - тести. Для цього нам потрібна система, що є простою REST API, що складається з 2 ендпоітнів.

API та розподіл

Web-додаток (Клієнт) при запиті до вашого API генерує деякий унікальний ID клієнта, який зберігається між сесіями, та запитує експеримент, додаючи GET параметр device-token. У відповідь сервер дає експеримент.

Для кожного експерименту клієнт отримує:

- Ключ: назва експерименту (X-name). Передбачається, що в клієнті є код, який змінюватиме якусь поведінку залежно від значення цього ключа
- Значення: рядок, одна з можливих опцій (див. нижче)

Важливо, щоб девайс попадав в одну групу і завжди залишався в ній на основі device-token.

Query:

Type: GET

Parameters: "device-token"

Return:

{key: "X-name", value: "string"}

Експерименти

1. У нас є гіпотеза, що колір кнопки «купити» впливає на конверсію на покупку

Ключ: button_color

Опції:

#FF0000 → 33.3%

#00FF00 → 33.3%

#0000FF → 33.3%

Так після 600 запитів до API з різними DeviceToken кожен колір повинні отримати по +-200 девайсів.

2. У нас є гіпотеза, що зміна вартості покупки в додатку може вплинути на наш маржинальний прибуток. Але щоб не втрачати гроші у разі невдалого експерименту, 75% користувачів будуть отримувати стару ціну і лише на малій частині аудиторії ми протестуємо зміну:

Ключ: price

Опції:

10 → 75%

20 → 10%

50 → 5%

5 → 10%

Приклад 1:

Запит клієнта: GET: https://yourdomain.com/experiment/button-color?device-token=randomstring1

JSON відповідь сервера: {key: "button_color", value: "#FF0000”}

Приклад 2 (другий запит того самого клієнта):

Запит клієнта: GET: https://yourdomain.com/experiment/button-color?device-token=randomstring1

JSON відповідь сервера: {key: "button_color", value: "#FF0000"}

Приклад 3:

Запит клієнта: GET: https://yourdomain.com/experiment/button-color?device-token=randomstring2

JSON відповідь сервера: {key: "button_color", value: "#00FF00"}

Вимоги та обмеження

1. Якщо девайс одного разу отримав значення, то він завжди отримуватиме лише його
2. Експеримент проводиться тільки для нових девайсів: якщо експеримент створений після першого запиту від девайсу, то девайс не повинен нічого знати про цей експеримент

Завдання:

1. Спроектуйте, опишіть та реалізуйте API. Воно має працювати через swagger чи Postman
2. Створіть сторінку для статистики (на вибір):
    a. проста таблиця зі списком експериментів, загальна кількість девайсів, що беруть участь в експерименті та їх розподіл між опціями
    b. Статистика у форматі JSON зі списком експериментів, загальна кількість девайсів, що беруть участь в експерименті та їх розподіл між опціями
3. Використовуйте MS SQL базу даних для зберігання інформації про експерименти та їх результати
4. Використовуйте прямі запити або процедури, що зберігаються для CRUD операцій з БД.
5. Надайте структуру БД разом із результатом вашої роботи
6. Ваше рішення має бути розміщене на GitHub у відкритому репозиторії.

Можна використовувати будь-які технології та бібліотеки в рамках .NET

Плюсом буде:

- Наявність тестів (UNIT)

- Заповнений GIT README
- Обробка винятків (try|catch)
- Коментарі в коді
- Оптимізація (Обґрунтування проведеної оптимізації у вигляді коментарів у коді)

Варіант 2

Опис:

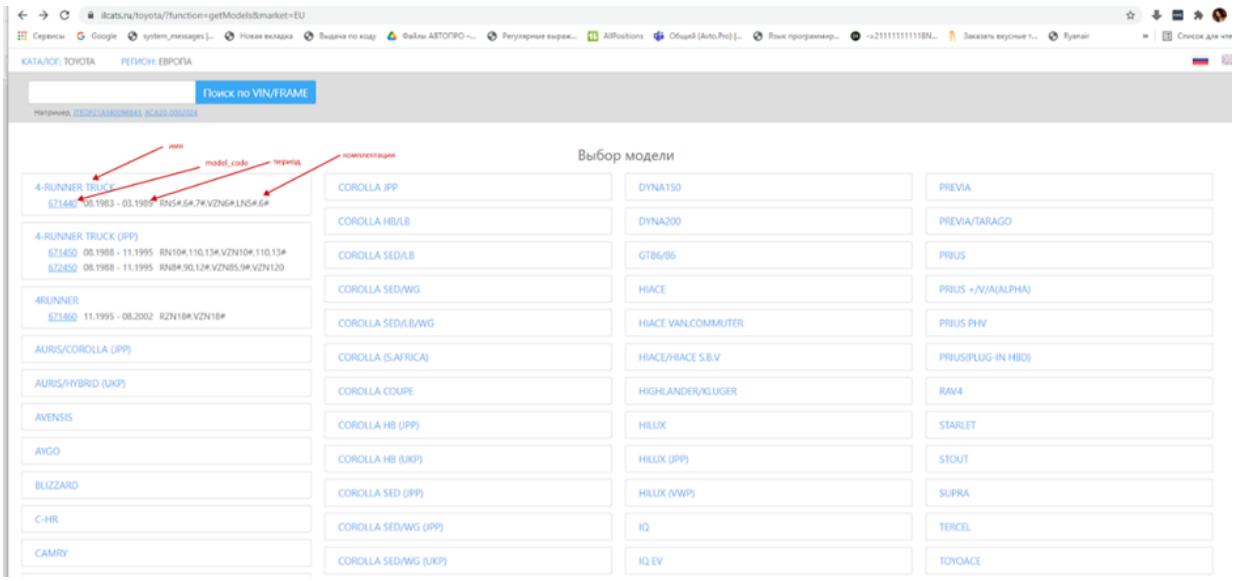
Завдання полягає у розпарсенні даних з іншого сайту. Потрібно написати консольний застосунок (або WPF, або MVVM), який ходитиме розділами сайту і розпарсиватиме за допомогою регулярних виразів потрібні нам значення, зберігаючи все це до бази.

Завдання:

Є два сайти www.ilcats.ru, catcar.info

У ilcats більш надійна система антипарсингу, парсинг цього сайту зараховуватиметься вищим балом. У catcar менш надійна система антипарсингу, що побудована на кількості запитів у один період часу. Який сайт парсити, обирайте самі. Головне, щоб ви зрозуміли алгоритм, що описаний на прикладі сайту ilcats.ru. У catcar теж є ті самі дані, але по-іншому розташовані та зі своїми посиланнями.

1. Парсинг починається зі списку моделей до певної марки. Наприклад у Toyota
https://www.ilcats.ru/toyota/?function=getModels&market=EU

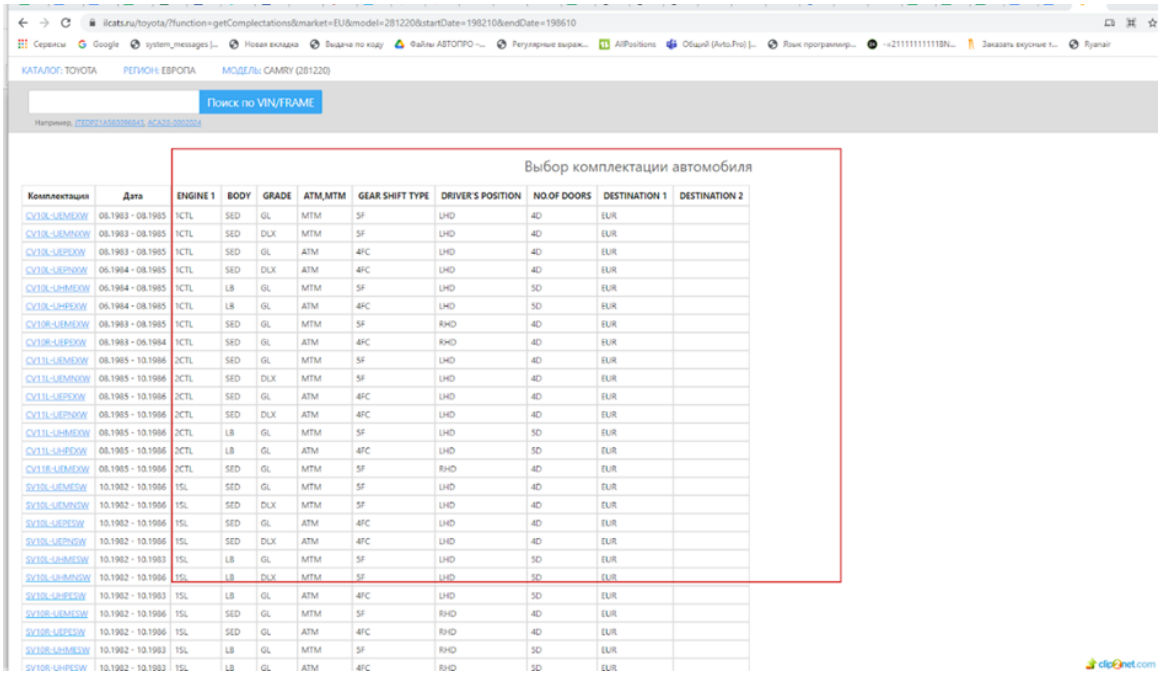


Зберігаємо у БД:

- ім’я моделі;
- код моделі

- дата виробництва
- комплектації

2. Збереження даних про комплектації. Тиснемо на посилання «Код моделі» у попередньому пункті



https://www.ilcats.ru/toyota/?function=getComplectations&market=EU&model=281220&startDate=198210&endDate=198610

Зберігаємо у БД:

Усі поля, що представлені на сторінці, треба зберегти у окремій таблиці, що пов’язана з попередньою таблицею моделей.

3. Збереження даних на першому рівні каталогу
https://www.ilcats.ru/toyota/?function=getGroups&market=EU&model=281220&modification=CV10L-UEMEXW&complectation=001

Зберігаємо у БД:

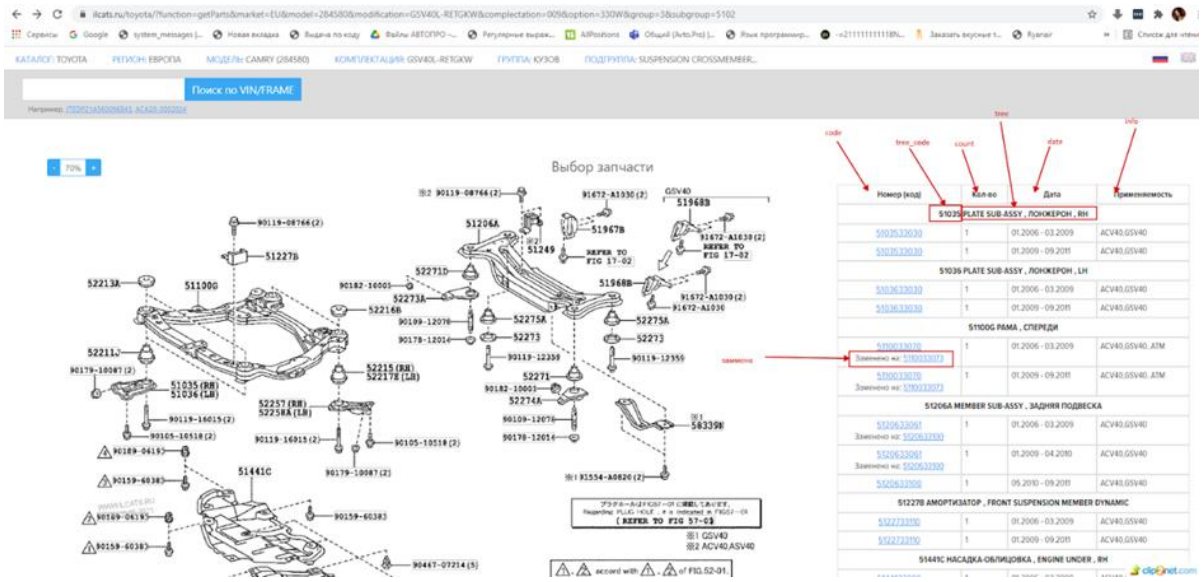
Назву та пов’язуємо з попередньою таблицею

4. Збереження даних на другому рівні каталогу
https://www.ilcats.ru/toyota/?function=getSubGroups&market=EU&model=281220&modification=CV10L-UEMEXW&complectation=001&group=1

Зберігаємо у БД:

Назву та пов’язуємо з попередньою таблицею

5. Збереження даних про схему
https://www.ilcats.ru/toyota/?function=getParts&market=EU&model=281220&modification=CV10L-UEMEXW&complectation=001&group=3&subgroup=5202



Зберігаємо у БД:

- код запчастини (code);
- кількість (count);
- інфо (info);
- код назви (tree_code);
- назву (tree);
- дату (date);
- посилання на попередню таблицю.

Зберегти локально картинку у папці images і надати їй довільне унікальне ім’я у цій папці. Зберегти отримане ім’я у окремому полі у БД.

База даних:

Спроєктуйте самі базу даних MS SQL, врахуйте відразу усі моменти, щоб не зберігати дублікати даних. Постарайтеся спроєктувати базу таким чином, ніби ви працюватиме не з 10к записів у таблицях, з 10 млн, тому зберігання інформації, що дублює, займатиме зайве місце.

Але при цьому не втратьте зручність у наступних вибірках інформації з бази даних і швидкість цих вибірок на великому об’ємі даних. Для цього створіть необхідні індекси.

Після наповнення вашої бази даних якоюсь частиною значень, зробіть вибірку, що збере усі дані в усіх таблицях в один запит VIEW, та збережіть його.

Зафіксуйте час, який у вас піде на виконання завдання.

Якщо воно здасться вам трудомістким і займе багато часу, виконайте тільки частину і надішліть результат. Описані задачі та побажання не є суворо обов'язковими для виконання. Ми підшукуємо відразу 2-3 кандидатів на цю вакансію, тому рівень реалізації приймаємо будь-який.

Ми чекаємо від вас:

- Фіксацію витраченого часу на виконання завдання.
- Вихідний код (бажано прокоментований).
- Архітектуру бази даних і у цілому алгоритм роботи (необов'язково, але вітається).
- Частково наповнену базу даних (нам не обов'язково випарсивати увесь сайт, достатньо побачити частину збережених даних).
- Результат вибірки з бази даних.
- Коментарі, які зробили у проєктуванні архітектури бази даних з ціллю економії розміру та зручності вибірок. (Необов'язково, але вітається.)
- Як можна не за описаною вище схемою, а по-іншому отримати усі дані з цього сайту більш економічним шляхом? Тобто як би ви вирішили цю задачу не по ТЗ, а по своєму алгоритму, щоб отримати абсолютно ті ж дані? (Необов'язково, але вітається.)