Эндпоинты:

1. Получить данные АЗС из первого источника (список АЗС)
2. Получить данные АЗС из второго источника (список АЗС)
3. Получить данные АЗС из БД по id (на вход — id, результат — словарь с данными)
4. Получить список АЗС из БД (результат — список словарей с данными)
5. Получить список всех видов топлива из БД (результат — список словарей с данными)
6. Получить список всех дополнительных услуг из БД (результат — список словарей с данными)
7. Получить данные по топливу из БД (на вход id или название топлива, результат — словарь с данными)
8. Получить данные по дополнительной услуге из БД (на вход id или название доп. услуги, результат — словарь с данными)
9. Изменить данные топлива в БД (на вход — пары название поля-его новое значение, результат — словарь с обновленными данными)
10. Изменить данные дополнительной услуги в БД (на вход — пары название поля-его новое значение, результат — словарь с обновленными данными)
11. Изменить данные АЗС в БД (на вход — пары название поля-его новое значение, результат — словарь с обновленными данными)
12. Добавить новую АЗС в БД (на вход — словарь с данными новой записи в БД, на выход — статус операции (успешно/не успешно) и айди новой АЗС)
13. Добавить новое топливо в БД (на вход — словарь с данными новой записи в БД, на выход — статус операции (успешно/не успешно) и айди нового топлива)
14. Добавить новую доп. услугу в БД (на вход — словарь с данными новой записи в БД, на выход — статус операции (успешно/не успешно) и айди новой доп. услуги)

Схема БД:

1. *Таблица АЗС:*
    a. ID
    b. Координаты
    c. Номер
    d. Адрес
    e. Список URL изображений
    f. Список дополнительных услуг (ID)
    g. Список цен на топливо (ID)
    h. Флаг — полная информация или нет.
2. *Топливо:*

    a. ID
    b. Название топлива (уникальное поле)
    c. Цена топлива
    d. Валюта цены топлива
    e. Иконка (ссылка на изображение)
3. *Дополнительные услуги*
    a. ID
    b. Название услуги (уникальное поле)
    c. Иконка (ссылка на изображение)
    d. Другие поля (если есть, не указано в ТЗ)

Примерный стек:

1. Python + Flask/Django + GraphQL/REST API
2. Реляционная СУБД для хранения данных (MySQL/PostgeSQL)
3. Asyncio + Redis для асинхронных задач

Схема работы приложения:

1. Первый запуск — проход по обоим источникам по очереди. Если айди АЗС нет в таблице — добавляем данные с флагом, что информация неполная. Если айди АЗС есть в таблице и флаг со значением False, дополняем информацию и устанавливаем флаг True. Если айди АЗС есть в таблице и флаг со значением True пропускаем АЗС.
2. При добавлении АЗС в таблицу — в случае, если в таблицах топлива/доп.услуг нет данных из списков этой АЗС, внести их, после чего внести в запись в таблице АЗС их айди.
3. В случае необходимости обновления данных из источников: пройтись по обоим спискам из источников, если айди АЗС нет в таблице — добавить, повторив шаги из пункта 1, если айди АЗС есть в таблице — сверить изменились ли поля, при необходимости обновить данные в БД. Во время пересбора данных необходима блокировка изменений БД другими сервисами (доступ только на чтение) для избежания коллизий. Частота пересбора изменений зависит от срочности получения свежих данных (не очевидно по ТЗ), как вариант — по расписанию один раз ночью и один раз днём в период минимальной активности пользователей (если такой есть, возможно, в обеденный перерыв). Можно вывести данный функционал в асинхронность/отдельный сервис на другой машине, чтобы не мешать получению данных клиентами.
4. Дополнительно — разделение видов пользователей (у администраторов доступ на просмотр и изменение данных, у клиента — только на просмотр)