ChatGPT

Анализ проекта SnowOps (ANPR-Service и Number-Service)

Общий обзор архитектуры

Проект **SnowOps** разделён на несколько микросервисов, два из которых – `snowops-anpr-service` и `snowops-number-service` – отвечают за работу с автомобильными номерами. В базе данных они используют общие таблицы для номеров и списков (белых/чёрных списков). Микросервис **ANPR-Service** принимает события распознавания номеров (ANPR events) и сохраняет их, а **Number-Service** управляет списками номеров (добавление в белый/чёрный список, проверка номера). Оба сервиса написаны на Go, используют веб-фреймворк (Gin) для REST API и подключаются к одной базе (PostgreSQL через GORM). Ниже подробно рассмотрены ключевые аспекты: распознавание номеров, проверка по базе, доступные API, интеграция с камерой Hikvision DS-TCG406-E через облако Hik-Connect, и рекомендации по интеграции.

1. Реализация распознавания автомобильных номеров

Отсутствие встроенного OCR: В коде сервиса **snowops-anpr-service** нет собственной реализации OCR или обработки изображений. Сервис **не выполняет распознавание символов на изображении**, а ожидает, что распознанный номер будет передан ему извне (например, камерой или другим компонентом). Это видно из того, что точка входа для события (`POST /api/v1/anpr/events`) сразу принимает JSON с уже распознанным номером (поле `plate`) и данными о событии. В функции обработчика `createANPREvent` запрос парсится в структуру `EventPayload` и проверяется, что поле номера заполнено. Затем вызывается бизнес-логика `ANPRService.ProcessIncomingEvent(...)` для обработки этого события.

Нормализация номера: Перед сохранением сервис выполняет нормализацию текста номерного знака. Функция `utils.NormalizePlate(raw string) string` удаляет пробелы, дефисы и приводит номер к верхнему регистру【27†source】. Например, `"AB 123-CD"` превратится в `"AB123CD"`. Если после нормализации строка пустая (например, распознавание дало только пробелы), событие отклоняется как некорректное. Такая нормализация позволяет унифицировать хранение номеров (например, чтобы *A123BC* и *A123BC* хранились одинаково).

Обработка события ANPR: Функция `ANPRService.ProcessIncomingEvent` реализует логику приема распознанного номера. Основные шаги внутри нее: 1. **Валидация входных данных:** проверяется наличие `Plate` (номер), `CameraID` (идентификатор камеры) и времени события. Если каких-то обязательных данных нет, возвращается ошибка `ErrInvalidInput`. 2. **Нормализация номера:** как описано, с помощью `NormalizePlate`. Если результат пустой – ошибка. 3. **Сохранение номера в базе:** вызывается `repo.GetOrCreatePlate(ctx, normalized, original)` – репозиторий ищет запись в таблице `plates` по нормализованному номеру. Если номер уже есть, получает его `ID`; если нет – создает новую запись. Таким образом, каждый уникальный номер хранится один раз (поле `normalized` уникально в БД). 4. **Создание события:** формируется структура события `ANPR Event` с сылкой на PlateID, с исходным номером (как распознан камерой) и нормализованным. Добавляются данные камеры: идентификатор, модель (если не указана в payload – подставляется `defaultCameraModel` из конфигурации),

направление движения, номер полосы, уровень уверенности (confidence), информация о машине (цвет, тип), URL снимка и пр. Эти поля соответствуют структуре `anpr.EventPayload` и частично заполняются на основе пришедших данных. Репозиторий сохраняет событие в таблицу `anpr_events` с текущей меткой времени. 5. **Проверка по спискам:** сразу после сохранения события вызывается `repo.FindListsForPlate(ctx, plateID)`. Эта функция выполняет SQL JOIN между таблицей списков и элементов списков, чтобы найти, в какие списки (если вообще) включен данный номер. Возвращается список попаданий – с идентификаторами списков, их именами и типами (например, ListName `default_whitelist` с типом `WHITELIST`). 6. **Результат обработки:** сервис возвращает структуру `ProcessResult`, где указывает ID нового события, ID номера, сам нормализованный номер и массив найденных списков (hits). Этот результат отправляется клиенту вызова. Например, ответ на **POST /api/v1/anpr/events** выглядит как:

```
{
  "status": "ok",
  "event_id": 123,
  "plate_id": 45,
  "plate": "AB123CD",
  "hits": [ { "list_id": 1, "list_name": "default_blacklist", "list_type":
"BLACKLIST" } ]
}
```

если номер был найден, например, в черном списке.

Вывод: Распознавание номеров как таковое в коде **не реализовано** – предполагается, что эту функцию выполняет камера Hikvision или другое устройство, а сервис SnowOps лишь принимает уже готовый результат (строку номера) для дальнейшей обработки. Основная логика сервиса – нормализовать и сохранить номер, зарегистрировать событие и сразу определить, относится ли номер к каким-либо предопределенным спискам (см. следующий раздел).

2. Проверка распознанных номеров по базе данных

Структура базы данных: Оба сервиса используют общую базу с такими основными таблицами: - **plates** – все уникальные номера машин. Поля: `id` (PK), `number` (исходный вид номера, как ввели/распознали), `normalized` (нормализованный номер, уникальный индекс), опционально страна и регион, `created_at`.
- **anpr_events** – события распознавания. Поля: `id`, `plate_id` (ссылка на plates), `camera_id`, модель камеры, направление (`direction`), полоса (`lane`), `raw_plate` (номер как распознан, до нормализации), `normalized_plate`, `confidence` (уверенность), `vehicle_color`, `vehicle_type`, `snapshot_url` (ссылка на фото машины или номера), `event_time` (время события) и `raw_payload` (полный JSON/данные события, хранится как JSONB), `created_at`. Индексы созданы по времени и plate_id для ускорения выборки. - **lists** – списки номеров. Поля: `id`, `name` (уникальное имя списка), `type` (тип списка, строка, например `"WHITELIST"` или `"BLACKLIST"`), описание, `created_at`. - **list_items** – связи номеров со списками (многие-ко-многим). Поля: составной PK из `list_id` и `plate_id`, плюс `note` (примечание) и `created_at`.

При развертывании базы оба сервиса выполняют миграции, которые создают эти таблицы, а также автоматически добавляют два списка по умолчанию: **default_whitelist** (тип WHITELIST) и

default_blacklist (тип BLACKLIST). То есть, сразу после запуска будут существовать белый и чёрный списки с этими именами.

Проверка номера в списках: Когда поступает новое событие распознавания, как описано, ANPR-Service сразу узнаёт, числится ли данный номер в каком-либо списке. Это происходит через репозиторий: `FindListsForPlate` выполняет SQL-запрос, выбирающий все списки, связанные с данным PlateID. Результат – массив структур `ListHit` с полями `list_id`, `list_name`, `list_type`. Таким образом, сервис может мгновенно определить, например, что номер принадлежит «белому списку» (разрешённые) или «чёрному списку» (запрещённые, или подозрительные) и далее среагировать (в рамках SnowOps возможно есть ещё сервис реагирования, например, для открытия шлагбаума или сигнала тревоги – вероятно, **snowops-acts-service** делает это, хотя его код мы не анализируем).

Хранение и поиск номеров: Сервис `snowops-number-service` предназначен для управления списками и вручную запрошенных проверок. Он имеет аналогичный метод `NumberService.NormalizeAndCheck(plate)`, который: 1. Нормализует номер так же, как ANPR-Service. 2. Через репозиторий `GetOrCreatePlate` убеждается, что номер есть в таблице plates (создаёт, если нового номера ещё не было). 3. Вызывает `FindListsForPlate` и возвращает результат – аналогично, список попаданий. Эта операция доступна по API как **POST /api/v1/numbers/check**, куда можно послать JSON `{"plate": "A123BC"}` и получить в ответ информацию, есть ли номер в списках. Если номер не найден вообще, он будет создан в базе (но ни в один список не помещён, hits будет пустым).

Добавление в списки: Number-Service также предоставляет методы для управления белым/черным списком: - **POST /api/v1/numbers/whitelist** – добавить указанный номер в белый список. - **POST /api/v1/numbers/blacklist** – добавить в чёрный список. - **DELETE /api/v1/numbers/whitelist?plate=...** – убрать номер из белого списка. - **DELETE /api/v1/numbers/blacklist?plate=...** – убрать из чёрного списка.

Эти обработчики вызывают методы `AddToWhitelist`, `AddToBlacklist`, `RemoveFromWhitelist`, etc., которые внутри себя используют общую функцию `addToList`/`removeFromList`. Например, `AddToWhitelist` просто вызывает `addToList(..., listName="default_whitelist")`. В реализации `addToList`, сервис проверяет/создает Plate, затем получает через репозиторий объект списка по имени (`GetListByName`) и добавляет связь `ListItem` между списком и номером. Таким образом, логика списков достаточно проста: есть два специальных списка с фиксированными именами, куда можно через API добавлять/удалять номера. (Заметим, что структура базы позволяет иметь и произвольные другие списки, но текущий API оперирует именно двумя «default» списками.)

Вывод: Проверка распознанных номеров по базе выполняется мгновенно при поступлении каждого события и при явных запросах. Хранится история всех сканирований (события) и реестр всех увиденных номеров. Система поддерживает как минимум два типа списков (белый/чёрный) для классификации номеров. Код, отвечающий за это, включает: - модели `Plate`, `List`, `ListItem` и запросы GORM для поиска списка по номеру; - в ANPR-Service: метод `FindListsForPlate` (через JOIN получает списки для заданного plate_id); - в Number-Service: методы `NormalizeAndCheck`, `AddToWhitelist/Blacklist` и соответствующие функции репозитория (`AddPlateToList`, `RemovePlateFromList`, `GetListByName` и т.п.).

Все эти части работают совместно, используя одну базу данных, поэтому **ANPR-Service и Number-Service фактически дополняют друг друга** – первый пишет события и проверяет списки, второй предоставляет API для работы с самими списками и ручной проверки номера.

3. REST API и точки входа для получения распознанных номеров

Сервисы предоставляют REST API (JSON over HTTP) для интеграции. Ниже перечислены ключевые конечные точки (endpoints):

ANPR-Service API (порт по умолчанию 8080):
- `POST /api/v1/anpr/events` – **приём события распознавания номера.** Ожидает JSON в теле запроса, соответствующий структуре `EventPayload`. Пример содержания:

```
{
  "camera_id": "CAM123",
  "camera_model": "DS-TCG406-E",
  "plate": "AB123CD",
  "confidence": 97.5,
  "direction": "forward",
  "lane": 2,
  "event_time": "2025-11-24T06:15:00Z",
  "vehicle": { "color": "white", "type": "car" },
  "snapshot_url": "http://camera/snap.jpg",
  "raw_payload": { ... любое доп. поле ... }
}
```

Многие поля опциональны (модель камеры, цвет и т.д.). В обработчике, если `event_time` не задан, он проставится текущим временем. В ответе будет статус и данные о созданном событии, как описано ранее. Этот метод – **главный вход для новых распознанных номеров**. - `GET /api/v1/plates?plate=<номер>` – **поиск номеров**. Возвращает список номеров, совпадающих с переданным (с учётом нормализации). Например, если запросить `?plate=AB123CD`, сервис нормализует и найдёт все записи в таблице plates, где `normalized = 'AB123CD'` (их может быть несколько, теоретически, если один и тот же номер был введён с разным оригинальным написанием или разной раскладкой). Каждая запись снабжается полем `last_event_time` – время последнего события, связанного с этим номером. Это полезно, чтобы узнать, когда данный автомобиль последний раз обнаруживался. - `GET /api/v1/events?plate=<номер>&from=<ISO8601>&to=<ISO8601>&limit=<N>&offset=<M>` – **просмотр событий распознавания.** Позволяет выгрузить историю событий, опционально фильтруя по конкретному номеру (plate), по диапазону времени (`from` – `to` в формате RFC3339), а также постранично (`limit`, `offset`). Если `plate` указан, сервис снова нормализует его и фильтрует по полю `normalized_plate`. Если указаны `from` / `to` – конвертирует их в Time и фильтрует по `event_time`. Параметры `limit` по умолчанию 50 (максимум 100), `offset` по умолчанию 0. Результат – список событий с подробностями (`EventInfo` содержит все поля события: камера, время, номер, уверенность, и т.д.). Этот эндпоинт позволяет, например, вывести последние N проездов, или все события по конкретному номеру за период.

Примечание: В ANPR-Service также заложены механизмы аутентификации (JWT) – в коде есть middleware `Auth`, parser для токенов, однако в `Register` видно, что вышеупомянутые endpoints добавлены в `public` группу (без авторизации). То есть, на данный момент они открыты. При необходимости можно перенести их в защищенную группу с токенами.

Number-Service API (порт, вероятно, иной, например 8081):
Базовый префикс у этого сервиса `/api/v1/numbers`. Его обработчик регистрирует такие маршруты: - `POST /api/v1/numbers/check` – **проверка номера.** Принимает JSON `{"plate": "<номер>"}`. Возвращает, присутствует ли номер в списках. По сути, это обёртка вокруг `NumberService.NormalizeAndCheck` – на выходе будет JSON с полями `plate_id`, `plate` (нормализованный), `original` (оригинальный запрос) и `hits` (массив найденных списков). Например:

```
{ "data": {
    "plate_id": 45, "plate": "AB123CD", "original": "A6123Cd",
    "hits": [ { "list_id":1, "list_name":"default_whitelist",
"list_type":"WHITELIST" } ]
} }
```

(Здесь номер найден в белом списке). - `POST /api/v1/numbers/whitelist` – **добавить в белый список.** Тело: `{"plate": "<номер>", "note": "опциональное примечание"}`. Номер нормализуется, добавляется (если нового номера не было – создастся запись в plates) и в таблицу list_items в связке с списком `default_whitelist`. Примечание, если указано, сохранится. - `POST /api/v1/numbers/blacklist` – **добавить в чёрный список** (аналогично выше). - `DELETE /api/v1/numbers/whitelist?plate=<номер>` – **удалить из белого списка.** Номер указывается как query-param. Сервис нормализует его, находит соответствующую запись Plate и удаляет связь из list_items для default_whitelist. Если номера или связи не было – вернёт 404 или ошибку. - `DELETE /api/v1/numbers/blacklist?plate=<номер>` – **удалить из чёрного списка** (аналогично выше).

Таким образом, **для получения распознанных номеров и результатов их проверки доступны**: - активная отправка событий (POST /anpr/events), - просмотр сохранённых данных (GET /plates, /events), - а также API для взаимодействия со списками и проверки (в Number-Service). Эти HTTP-интерфейсы могут использоваться веб-приложением, мобильным клиентом или интегрироваться с другим ПО. Например, можно построить веб-страницу, которая отображает последние распознанные номера, обращаясь к `/events`, или страничку поиска номеров через `/plates`.

Стоит отметить, что **альтернативных точек входа помимо REST API не замечено** – например, нет явных упоминаний о чтении сообщений из очередей или о websocket-уведомлениях. Вероятно, система задумывалась для работы в режиме запрос/ответ через HTTP. Если нужен push-уведомление о новых событиях, можно либо опрашивать `/events` периодически, либо (лучше) доработать систему – например, отправлять сообщения в Kafka/RabbitMQ или через WebSocket – но из коробки этого нет.

4. Возможность связать с камерой Hikvision DS-TCG406-E (через Hik-Connect)

Камера **Hikvision DS-TCG406-E** – это специализированная ANPR-камера, которая **самостоятельно распознаёт номера** автомобилей на въезде/выезде. Задача – интегрировать её с описанными сервисами так, чтобы распознанные камерой номера автоматически попадали в SnowOps и сравнивались с базой.

Способности камеры: Согласно документации, современные камеры Hikvision ANPR поддерживают **несколько способов интеграции** с внешними системами. Во-первых, они предоставляют **API** (протокол ISAPI) и могут работать по стандарту **ONVIF Profile M** для передачи метаданных распознавания [1]. Проще говоря, камера может либо: - **Отправлять события на указанный сервер (HTTP Listening)** – камера сама делает HTTP POST запросы при каждом срабатывании ANPR. - **Предоставлять поток событий (Streaming) для клиента** – по протоколу Hikvision ISAPI или ONVIF, где внешняя программа подключается и слушает события.

Hik-Connect облако: Hik-Connect – это облачная платформа Hikvision для удалённого доступа к камерам (через P2P, без прямого проброса портов). На скриншоте пользователя видно раздел **«Платформа Hik-Connect»** с включенной опцией и доменом litedev.hik-connect.com. Это означает, что камера привязана к аккаунту и доступна удалённо через облако. Однако, **прямого публичного API у Hik-Connect для сторонних приложений нет** (по крайней мере, Hikvision не предоставляет открытой документации для получения событий через их облако). Чаще Hik-Connect используется через фирменное приложение или HikCentral. Для интеграции лучше опираться на возможности самой камеры по LAN/интернет-протоколам:

1. HTTP Listening (Alarm Server): Самый прямой путь – настроить камеру отправлять ANPR-события на SnowOps ANPR-Service. В веб-интерфейсе камеры есть раздел конфигурации (обычно **Network > Advanced Settings > Alarm Server** или аналогично). Туда можно ввести адрес сервера (IP или доменное имя), порт и URL пути, куда камера будет делать POST запрос при событии распознавания. Также нужно включить нужные типы событий (ANR – Automatic Number plate Recognition). На форуме отмечают: *«в настройках камеры открыть Network -> Advanced -> Alarm Server, добавить HTTP endpoint (например, путь* `/hikvision` *) и включить ANPR»*. После этого камера при каждом распознавании отправит HTTP POST на указанный адрес.

Формат такого запроса у Hikvision – **multipart/form-data** с XML и изображениями. В частности, камера шлёт XML-файл (например, `anpr.xml`) с детальной информацией о событии, и может прикреплять снимки (общий план и крупный план номера). В XML содержится номерной знак и атрибуты. Пример (на основе данных интеграции, упомянутых на StackOverflow):

```
<EventNotificationAlert version="2.0" ...>
    <ipAddress>...</ipAddress>
    <eventType>ANPR</eventType>
    <eventDescription>ANPR</eventDescription>
    <ANPR>
        <licensePlate>PECT600</licensePlate>
        <confidenceLevel>97</confidenceLevel>
        <vehicleType>vehicle</vehicleType>
        <color>white</color>
        ...
```

```
    </ANPR>
    <vehicleInfo>...</vehicleInfo>
    <pictureInfoList>...</pictureInfoList>
</EventNotificationAlert>
```

Как видно, номер передаётся внутри `<licensePlate>`, также есть `confidenceLevel` (уверенность), цвет машины, и прочие данные. Отдельно, jpeg-изображения идут как части multipart с файлами.

Чтобы интегрировать это с SnowOps, придётся написать обработчик для таких запросов. В текущем ANPR-Service ожидается JSON, поэтому напрямую камера туда постить не сможет (она отправит XML и изображения, что сервис не распарсит). Есть два варианта: - **Расширить ANPR-Service**: Добавить эндпойнт (например, `/api/v1/anpr/hikvision`) специально для Hikvision. Он бы принимал `Content-Type: multipart/form-data` и парсил входящие данные: извлекать XML, оттуда брать номер, время и пр., затем вызывать внутренний `ProcessIncomingEvent`. (В Go/Gin это реализуемо – надо читать `c.Request.MultipartForm` и XML unmarshal). - **Создать промежуточный адаптер**: отдельное небольшое приложение (можно даже скрипт на Python, используя библиотеку hikvisionapi или просто HTTP сервер). Этот адаптер будет слушать на определённом порту, принимать POST от камеры, извлекать номер и затем делать запрос **в SnowOps API** (например, отправить JSON на `POST /anpr/events`). Такой адаптер может также сохранять изображения при необходимости.

Оба подхода рабочие. Первый менее звеньев (камере сразу SnowOps), но требует изменить код сервиса. Второй – сохраняет сервис нетронутым, вся логика преобразования вынесена.

Важно: **камера и сервер должны быть в одной сети или сервер должен быть доступен по интернету**. Если SnowOps развёрнут локально рядом с камерой – проблем нет. Если SnowOps в облаке, а камера в локальной сети без «белого» IP, то прямой HTTP POST невозможен без проброса порта. Здесь как раз и помогает Hik-Connect – но повторимся, Hik-Connect не предоставляет способ указать кастомный сервер для событий. Он используется для Hikvision-экосистемы (например, посылать события в HikCentral Cloud).

2. Прямое подключение по API (без push): Альтернативно, можно **запрашивать события самой камерой**. Например, использовать Python-библиотеку `hikvisionapi` (как в вопросе на StackOverflow) или посылать GET запросы к ISAPI endpoint камеры (например, `http://<camera_ip>/ISAPI/Event/notification/alertStream` для потока уведомлений). При локальной доступности камеры SnowOps мог бы сам постоянно слушать поток событий. Однако, подобная логика **не реализована** в текущем коде – нет фонового потока или упоминания Hikvision API. Поля конфигурации `Camera.RTSPURL`, `HTTPHost` и `HikConnect` в `app.env` намекают, что разработчики **планировали** интеграцию с камерой: - `RTSPURL` – возможно, URL потока видео RTSP (для других модулей, например, детекции заполненности стоянки – упоминается `EnableSnowVolumeAnalysis`). - `HTTPHost` – вероятно, адрес веб-интерфейса камеры (например, `http://192.168.1.101`) для прямых API запросов. - `HikConnect` – мог быть UID или адрес для облачного подключения.

Тем не менее, в коде **нет реализации**, которая бы использовала эти параметры. Видимо, интеграцию с камерой отложили или вынесли за рамки этих сервисов. Таким образом, чтобы сейчас связать систему с камерой, нужно разработать дополнительный компонент (или расширить существующий).

3. ONVIF Profile M: Камера поддерживает ONVIF для передачи метаданных (распознанных номеров). Теоретически, можно подключиться к камере как ONVIF-клиент и подписаться на события. ONVIF Prof.M стандартизует события распознавания номера в XML/JSON формате. Если есть готовые библиотеки ONVIF для Go или Python, это путь к интеграции. Но он сложнее, чем HTTP listening, и тоже требует прямого доступа к камере (ONVIF через Hik-Connect, скорее всего, не работает). Поэтому, **ONVIF полезен, если сервер в той же сети**: тогда SnowOps-сервис (доработанный) мог бы зарегистрироваться как слушатель ONVIF Events и получать события. Однако, проще всё-таки настроить push от самой камеры.

Вывод по Hik-Connect: Использование облака Hik-Connect напрямую для нашей задачи ограничено. Hik-Connect позволяет получить видеопоток через их сервис (например, в виде «облачного» RTSP с UID, через Ezviz/Ezviz API), но **события о номерах во внешнее приложение Hik-Connect не рассылает**. Скорее всего, оптимальное решение: либо разместить SnowOps сервисы рядом с камерой (на объекте) чтобы подключаться локально, либо настроить VPN/порт для прямого взаимодействия. Hik-Connect может оставаться включённым для параллельного мониторинга, но для интеграции со сторонней системой лучше опираться на прямые методы.

Подытожим: **камера уже сама определяет номера и может отправлять данные наружу**, нужно лишь принять их. SnowOps может быть интегрирован с камерой Hikvision путем настройки **Alarm Server** на камере (HTTP POST при событиях) или реализации опроса событий через API. Оба варианта требуют написать парсинг XML->JSON. Далее эти данные легко вписываются в существующую логику: номер будет передан в `ProcessIncomingEvent`, который сразу добавит его в базу и проверит списки.

5. Рекомендации по интеграции камеры с существующими сервисами

A) Настройка камеры: В веб-интерфейсе камеры DS-TCG406-E перейдите в настройки сети и найдите раздел для отправки событий (может называться «Alarm Server», «HTTP Listening» или «Integration Protocols»). Включите отправку ANPR-событий. Укажите адрес сервера, где работает SnowOps ANPR-Service, и путь API. Например, если ANPR-Service доступен по `http://192.168.1.100:8080`, можно задать Host = `192.168.1.100`, Port = `8080`, URL = `/api/v1/anpr/events` (или другой, если решите сделать отдельный эндпоинт). Камера, к сожалению, не умеет сама отправлять JSON в нашем формате, она пошлёт multipart XML. Поэтому **необходимо обеспечить совместимость на стороне сервера**.

B) Адаптация SnowOps ANPR-Service: Самый прямой подход – доработать существующий сервис: - Добавить новый HTTP-обработчик, например `POST /api/v1/anpr/hikvision` (название произвольное). В нём отключить требование `ShouldBindJSON` и вместо этого читать сырые данные запроса. Использовать Go-пакет `mime/multipart` или `http.Request.ParseMultipartForm` чтобы выделить части. Найти часть с XML (имя файла обычно `anpr.xml`) – это текст ~2-3KB. Распарсить XML (через encoding/xml). Извлечь нужные поля: номер (`licensePlate`), время события (`dateTime` внутри XML, конвертировать в time.Time), направление (`direction`), номер полосы (`line`), уверенность (`confidenceLevel`), цвет и тип машины (`vehicleType`, `color`). Также можно получить IP камеры из XML (поле ipAddress), но лучше использовать заданный CameraID (например, задать CameraID = серийный номер устройства, либо статично настроить в конфиге). - Извлечённые данные поместить в структуру `anpr.EventPayload` – она почти совпадает по полям с тем, что даёт камера. Например, `payload.Plate = licensePlate`, `CameraID = <ID этой камеры>`,

`Confidence = confidenceLevel`, `Direction = direction`, `Lane = line`, `EventTime = dateTime` (распарсенный). URL снимка: камера в XML передаёт пути к файлам (но сами изображения уже пришли в POST). Можно сохранить их локально или загрузить в хранилище, получив URL – но это опционально. На первое время достаточно игнорировать изображение или сохранить путь к нему локально. - Затем вызвать `ANPRService.ProcessIncomingEvent(ctx, payload, defaultCameraModel)` – последний параметр можно взять из конфигурации (например, `cfg.Camera.Model` – вероятно, там можно указать `"DS-TCG406-E"`). - Вернуть простой ответ камере. Камера ожидает код 200 OK. Можно ответить статически `"Received"` – это неважно, главное подтверждение без ошибки. - Таким образом, буквально несколько десятков строк кода позволят напрямую принимать события от камеры. Наша система получит эти номера как будто присланные внешним POST JSON.

C) Внешний адаптер (альтернативно): Если нет возможности менять код SnowOps, напишите отдельный скрипт/службу: - Например, Python-скрипт, запускающий HTTPServer на порту (того же хоста, где сервис, либо другого, лишь бы камера могла достучаться). Он принимает POST, парсит XML и затем делает HTTP запрос **к SnowOps API**: `requests.post("http://localhost:8080/api/v1/anpr/events", json=payload)`. Payload – словарь с полями по формату EventPayload (как показано выше). SnowOps примет это как обычное событие. - Этот адаптер можно упаковать в контейнер или запустить как системную службу. Его плюс – не трогаем боевой сервис; минус – дополнительный компонент, возможна задержка на двойную пересылку.

D) Размещение и сеть: Убедитесь, что камера может достучаться до сервиса. Если сервис в облаке, придётся открыть публичный адрес/порт (что нежелательно без защиты). В таком случае лучше установить VPN между объектом и сервером, либо разместить SnowOps локально. Если SnowOps будет работать *внутри* той же сети, где камера (например, на мини-ПК или сервере на парковке), то можно оставить его закрытым в локальной сети, а доступ для администратора организовать через Hik-Connect (для видео) и, например, веб-интерфейс SnowOps через VPN.

E) Проверка работы: После настройки, при проезде автомобиля камера распознает номер и отправит событие. SnowOps ANPR-Service получит его (через прямой или адаптерный маршрут) и сохранит: появится запись в таблице events, а номер – в таблице plates (если новый). Сервис автоматически определит, есть ли номер в белом/черном списке, и выдаст это в ответе. Далее на основе `hits` можно принять решение: - Например, если hits содержит `default_whitelist`, можно послать команду открыть шлагбаум. В архитектуре SnowOps, возможно, за это отвечает **snowops-acts-service** – он мог бы подписываться на новые события или периодически опрашивать события. Но данных по нему мало. Если такого механизма нет, можно внедрить: например, Number-Service при check может возвращать статус или внешний компонент сам проверяет (через /numbers/check) и управляет оборудованием.

F) Использование возможностей камеры: Hikvision-камера помимо номера даёт **фото** и характеристики машины. Эти данные тоже можно сохранять или использовать: - URL снимка (`snapshot_url`) можно заполнить, если, скажем, настроить на камере сохранение снимков на FTP и формирование URL, либо сделать API запрос чтобы получить кадр по ID события. (Hikvision ISAPI имеет метод запроса изображения события). - Цвет, тип машины – уже приходят, их можно сохранить (в `vehicle.color`, `vehicle.type` поля EventPayload). - Таким образом, SnowOps база может обогащаться информацией, а интерфейсы – показывать фото нарушителя или цвет авто.

G) Hik-Connect облако: Если прямой доступ труден, и нужно через облако – рассмотрите вариант **Hik-ProConnect** или **HikCentral Cloud**. Эти платформы имеют API, но они предназначены для партнеров и масштабных решений. Прямого легкого способа «подписаться» на события через Hik-Connect нет. Возможно, Hikvision предоставляет SDK, работающий с устройством по серийному номеру и паролю (P2P соединение). Если такой SDK доступен, можно написать сервис, который подключается к камере через облако и получает события. Однако, это довольно сложный путь. Проще поставить шлюз (например, miniPC с подключением к обеим сетям: локальной и интернет) и использовать стандартные методы.

Выделение ключевых функций и классов (резюме): В коде SnowOps, на который следует обратить внимание при интеграции: - *Парсинг ANPR-событий:* `snowops-anpr-service/internal/http/handler.go` – функция `createANPREvent` (приём JSON события) и код, вызывающий `ProcessIncomingEvent`. Также, если будет сделан свой обработчик для Hikvision, он появится здесь. - *Бизнес-логика события:* `internal/service/anpr_service.go` – функция `ProcessIncomingEvent` (описана подробно выше) – нормализация, запись, проверка списка. - *Хранилище номеров:* `internal/repository/anpr_repository.go` – функции `GetOrCreatePlate` (работа с таблицей plates), `CreateANPREvent` (вставка в anpr_events). - *Хранилище списков:* Это реализовано и в ANPRRepository, и дублируется в NumberRepository – структуры `List` и `ListItem` и методы, например `FindListsForPlate`, `AddPlateToList`, `RemovePlateFromList`. Эти методы выполняют SQL запросы к таблицам lists и list_items. - *Проверка «номер в списке»:* `snowops-number-service/internal/service/number_service.go` – метод `NormalizeAndCheck` объединяет все шаги (создать Plate при необходимости, найти списки). Его результат `CheckResult` содержит поле `Hits` – именно его возвращает API `/check`. Эта же логика фактически используется при каждом событии в ANPR-Service (там аналогично вызывается поиск списков). - *Конфигурация камеры:* `snowops-anpr-service/internal/config/config.go` – структура `CameraConfig` (поля RTSPURL, HTTPHost, HikConnect, Model). Хотя код не использует их, можно задействовать `cfg.Camera.Model` для указания модели по умолчанию, которая подставляется, если камера не передала свою (в `ProcessIncomingEvent` это и делается).

Напоследок, если система требует **автоматической реакции** (например, открытие ворот для белого списка), убедитесь, что есть компонент, который читает `hits` и предпринимает действие. Возможно, SnowOps-Acts-Service предназначен для этого – например, он может слушать появление новых событий (через БД или сообщение) и, если найден whitelist hit, активировать реле. Если такого механизма нет, можно реализовать простой скрипт-демон, который периодически запрашивает последние события через `/events` и проверяет, есть ли среди них нужные (или использовать webhook).

Заключение: Проект SnowOps уже содержит основу для регистрации и проверки автомобильных номеров. Интеграция с камерой Hikvision ANPR сведётся к получению от неё уже готовых распознанных номеров. Рекомендуется использовать функцию **HTTP push** с парсингом XML, так как она наиболее надежна и не требует постоянного опроса. Настроив это, вы получите автоматическое добавление номеров в систему и их сравнение с базой в режиме реального времени. Далее можно наращивать функциональность – уведомления охране, автоматическое управление доступом, накопление статистики по потокам машин и т.д., используя данные, которые собирают ANPR-Service и Number-Service. Все необходимые для этого точки расширения (REST API, структура БД, кодовые модули) в проекте присутствуют и готовы к использованию. [1]

---

1 ANPR (Automatic Number Plate Recognition) - Solutions by Function - Hikvision Europe
https://www.hikvision.com/europe/solutions/solutions-by-function/anpr--automatic-number-plate-recognition-/