EPIC 4. Тикеты

1) Цель

Дать ТОО возможность назначать задания (тикеты) подрядчикам, подрядчикам - распределять водителей, водителям - отмечать статус работы. Учёт выполняется фактами из камер: `lpr_event`, `volume_event` и связями в `trip`. Никаких план-нормативов - только фактические значения из событий.

2) Базовые сущности (по ERD)

- `ticket`: участок + подрядчик + период + статусы/временные метки. Поля для «норматива» в схеме отсутствуют - и не нужны.
- `ticket_assignment`: назначение водителей/техники на тикет.
- `trip`: рейс в рамках тикета с привязкой к входным/выходным `lpr_event` / `volume_event` и полю `detected_volume_entry/exit`.
- `volume_event`: факт измерения объёма с камеры (въезд/выезд) + фото кузова.
- `lpr_event`: распознавание номера (въезд/выезд) + фото номера.
- `cleaning_area`: участок уборки (геометрия и ответственный при необходимости).

3) Роли и права

- **Акимат**: только просмотр всех тикетов и их деталей. Без создания/редактирования.
- **ТОО**: создаёт/редактирует/удаляет (soft) тикеты для своих подрядчиков; просматривает прогресс и рейсы.
- **Подрядчик**: не создаёт тикеты; назначает водителей/машины (`ticket_assignment`), следит за выполнением, может перевести тикет в «завершён».
- **Водитель**: видит свои тикеты, нажимает «В работе»/«Завершено». Фактический объём и рейсы считаются автоматически по событиям камер.

4) Жизненный цикл тикета

значения колонки status

- `PLANNED` - создан ТОО, ещё нет факта работ (`fact_start` NULL).

- `IN_PROGRESS` - либо водитель отметил «В работе», либо появился первый `trip` с `entry_*` событиями → ставим `fact_start`.
- `COMPLETED` - подрядчик/водитель отметил завершение и нет активных рейсов (последние `exit_*` получены) → ставим `fact_end`.
- `CLOSED` - ТОО/Акимат закрыли после проверки.
- `CANCELLED` - ТОО отменило (до фактов).
  Храним в `ticket.status` + используем `fact_start/fact_end` из таблицы.

5) Расчёт метрик

Без нормативов. Показываем две ключевые метрики, считаемые из `trip`:

- **Кол-во рейсов** = `COUNT(trip.id)` по `ticket_id`.
- **Объём вывезен (м³)** = сумма по тикету:
  вариант MVP - `SUM(trip.detected_volume_entry)` (объём загруженного кузова на въезде);
  фиксируем пустой кузов на выезде через `detected_volume_exit` (ожидание 0<x<0.2).
  Эти поля уже в `trip` и получаются из привязанных `volume_event` (`entry_volume_event_id` / `exit_volume_event_id`).

6) Нарушения

Кодируем в `trip.status` (бейджи в UI):

- `OK` - рейс валиден;
- `ROUTE_VIOLATION` - GPS показал отсутствие работ в зоне `cleaning_area` перед въездом на полигон;
- `MISMATCH_PLATE` - распознанный номер не совпал с назначенным `vehicle_id`;
- `NO_ASSIGNMENT` - по камерам зафиксирован рейс, но водитель/машина не назначены на тикет;
- `SUSPICIOUS_VOLUME` - объём ниже 70% объема кузова, volume_event.volume_m3 > 0.7*vehicle.body_volume_m3.

Тогда «наличие нарушений» для таблицы тикетов = `EXISTS trip WHERE status <> 'OK'`.

7) UI по ролям

А) Акимат - `/akimat/tickets`

Фильтры: Статус, ТОО, Подрядчик, Участок, Период.

Таблица колонок:

- ID тикета, Участок, Подрядчик, Период (`planned_start_at-planned_end_at`), Статус,
- **Кол-во рейсов** (факт), **Объём вывезен (м³)** (факт), **Нарушения** (иконка при `EXISTS trip.status <> 'OK'`).
  Действие: «Подробнее» → карточка тикета (участок, подрядчик, период, `fact_start/fact_end` , список `ticket_assignment` , список рейсов с фото из `lpr_event.plate_image_url` и `volume_event.body_image_url`).

Б) ТОО - `/too/tickets`

Может: `+ Создать тикет` , редактировать, отменить (если нет фактов).

Создание тикета (форма):

Участок (`cleaning_area_id`), Подрядчик (`contractor_id`), Период (`planned_start_at` , `planned_end_at`), Комментарий (`description`).

Статус по умолчанию - `PLANNED` .

Редактировать: до начала фактов (при наличии `trip` или `fact_start` - только описание/период по бизнес-правилам).

Удалить (soft): разрешено до фактов (или перевод в `CANCELLED`).

Просмотр: те же поля, что у Акимата, плюс действия над статусом:

- Перевести в `CLOSED` после проверки.

В) Подрядчик - `/contractor/tickets`

Назначение водителей/техники: через `ticket_assignment` (многие водители на один тикет). Поля: `ticket_id` , `driver_id` , `vehicle_id` .

Статусы: может перевести в `IN_PROGRESS` (когда реально стартуют) и в `COMPLETED` (когда водители закончили, и последних `exit_*` достаточно).

Таблица колонок: ID, Участок, Период, Статус, Назначено водителей, **Кол-во рейсов**, **Объём (м³)**, Нарушения, действия «Назначить», «Подробнее».

Г) Водитель — `/driver`

Вкладка «Мои задания»

- Карточка тикета:
  - Название участка, адрес, мини-карта.
  - Кнопки: **«В работе»** / **«Завершено»**

    Меняют *только* признак готовности со стороны водителя (driver_mark), не закрывают тикет в системе. Объём и рейсы фиксируются только камерами (LPR/Volume) и собираются в `trip`.
  - Блок «Мои рейсы по тикету» (список `trip`):
    - Дата/время въезда/выезда.
    - Госномер, объём, полигон.
    - Статус рейса (OK / ROUTE_VIOLATION / FOREIGN_AREA / MISMATCH_PLATE / OVER_CAPACITY / NO_AREA_WORK и т.п.).
    - **Если статус ≠ OK → кнопка «Обжаловать»**.
      - Нажатие открывает форму обжалования по **конкретному нарушению** этого рейса.

Форма «Обжаловать нарушение» (модалка):

- Поля:
  - Нарушение (readonly из рейса, например `ROUTE_VIOLATION`).
  - Основание (select): *«Ошибка камеры/распознавания», «Ехал транзитом через участок», «Был назначен другой участок/задание», «Другая причина»*.
  - Комментарий (textarea, обязательно).
- Кнопки: **«Отправить»**, «Отмена».
- Результат: создаётся `appeal` в статусе `SUBMITTED`; водитель видит карточку обжалования в «История обжалований» внутри тикета.

Карточка обжалования (для водителя):

- Статус: `UNDER_REVIEW` / `NEED_INFO` / `APPROVED` / `REJECTED` / `CLOSED`.
- Лента комментариев (водитель, подрядчик, ТОО, Акимат) с метками времени.
- Вложения обеих сторон.

- Если статус `NEED_INFO` → кнопка **«Добавить информацию»** (добавляет коммент/фото и переводит обратно в `UNDER_REVIEW`).
- Уведомления: пуш/SMS при смене статуса и при новых комментариях.

8) Алгоритмы привязки фактов к тикету (ключевое)

1. **Создание/обновление рейса (`trip`)**
   При срабатывании камер на полигоне:

- Парсим `entry_lpr_event` и `entry_volume_event` → создаём/находим `trip`:
  - `vehicle_id` - по `plate_number` из `lpr_event` + справочник `vehicle`;
  - `driver_id` - по активному `ticket_assignment` для этого тикета/подрядчика/водителя (или по текущей привязке `vehicle.driver_id`);
  - `ticket_id` - находим активный тикет (`PLANNED/IN_PROGRESS`) для этого подрядчика и участка, к которому относится водитель/машина (логика назначения);
  - заполняем `polygon_id` по `camera.polygon_id`, `start_time/polygon_entry_time`, `entry_*_event_id`, `detected_volume_entry`.
- На выезде - обновляем тот же `trip`: `exit_*_event_id`, `polygon_exit_time`, `detected_volume_exit`.

2. **Переход тикета в `IN_PROGRESS`**
   Если появился первый `trip` или водитель нажал «В работе» → `status=IN_PROGRESS`, `fact_start=NOW()`.
3. **Переход в `COMPLETED`**
   Подрядчик/водитель жмёт «Завершено» и система проверяет, что **нет незакрытых рейсов** (у последних есть `exit_*`/кузов пустой). Ставим `fact_end=NOW()`.
   (Нормативов нет - только факт.)
4. **Агрегация показателей тикета**

- `total_trips` = `COUNT(trip WHERE ticket_id=...)`;
- `total_volume_m3` = `SUM(trip.detected_volume_entry)`;
  Обновляем по событию `trip` или ночной задачей (materialized view).

9) Фильтры и таблица

Фильтры: `ticket.status`, `contractor_id`, `cleaning_area_id`, период по `planned_*` / `fact_*`.

Колонки таблицы (для всех UI):

- ID тикета
- Участок (`cleaning_area.name`)
- Подрядчик (`organization.name`)
- Период (`planned_start_at-planned_end_at`)
- Статус (`ticket.status`)
- **Кол-во рейсов** (факт, агрегация по `trip`)
- **Объём вывезен (м³)** (факт: `SUM trip.detected_volume_entry`)
- Нарушения (иконка, если `EXISTS trip WHERE status <> 'OK'`)

  Все поля/связи есть в ERD.