# Техническое задание: DT-2997 — Задача 1 “Хранение секретов и подготовка БД”

## 1. Цель

Добавить в Custom Integration Request базовую возможность **хранить секреты** (в зашифрованном виде) и **подготовить схему данных** для вычисляемых полей, чтобы далее использовать это в подстановках `{{secret.*}}` / `{{computed.*}}` и в JS-вычислениях.

## 2. Контекст

- Существующий `Custom Integration` поддерживает URL/method/payload/headers, но подстановки сейчас работают фактически только в payload.
- По требованиям DT-2997 секреты должны:
  - храниться отдельно от основной конфигурации интеграции;
  - храниться **в зашифрованном виде**;
  - быть доступны по имени для подстановок и вычислений;
  - иметь флаг **конфиденциальности**.

## 3. Требования

### 3.1. Изменения в БД

#### 3.1.1. Новая таблица `custom_integration_secrets`

Назначение: хранение секретов интеграции отдельно от `custom_integrations`.

- **Поля**:
  - `id` : PK
  - `custom_integration_id` : FK на `custom_integrations.id`
  - `name` : строковый ключ секрета (пример: `API_TOKEN`)
  - `encrypted_value` : строка/текст, **зашифрованное** значение секрета
  - `is_confidential` : boolean (по умолчанию `false`)
  - `created_at`, `updated_at`

- **Ограничения**:
  - уникальность: `(custom_integration_id, name)`
  - при удалении интеграции секреты удаляются автоматически (cascade)
- **Индексы**:
  - индекс по `custom_integration_id`

### 3.1.2. Модификация таблицы `custom_integrations`

Добавить JSON-поле **`computed_fields`** для хранения конфигурации вычисляемых полей на уровне интеграции (MVP — без отдельной таблицы).

- **Поле**: `computed_fields` (JSON/JSONB)
- **Значение по умолчанию**: пустой массив `[]`
- **Формат данных (структура массива)**:
  - `name` : имя переменной (ключ внутри `computed` )
  - `type` : тип результата ( `string|number|boolean|object|array` )
  - `script` : JS-скрипт (строка)
  - `is_confidential` : boolean

## 3.2. Модель и связи

- Добавить модель **`CustomIntegrationSecret`** .
- Связи:
  - `CustomIntegration` **hasMany** `CustomIntegrationSecret`
  - `CustomIntegrationSecret` **belongsTo** `CustomIntegration`

## 3.3. Сервис секретов (логика уровня домена)

Нужен сервис (далее “SecretService”) для управления секретами интеграции.

- **Операции**:
  - создать секрет для интеграции
  - обновить секрет (в т.ч. возможность обновить только метаданные, не меняя значение)
  - удалить секрет
  - получить список секретов интеграции (для UI/API — только маска значения)
- **Правила хранения**:
  - `encrypted_value` хранит только зашифрованное значение.
  - расшифрованные значения не пишутся в логи и не возвращаются в API.
- **Правило обновления значения**:
  - если при обновлении значение не передано — хранится прежнее `encrypted_value` ;

- если передано пустое значение — это считается явным действием пользователя (поведение должно быть определено и зафиксировано в коде; по умолчанию: запрещать “обнуление” без явного флага).

# 3.4. API для управления секретами (CRUD)

API должно позволять управлять секретами в рамках конкретной `custom_integration`.

## 3.4.1. Общие требования

- В ответах API **никогда не возвращать** реальное значение секрета.
- Для UI возвращать:
  - `name`
  - `is_confidential`
  - признак “значение задано” (например, `is_set: true/false`) или `masked_value: "********"` при наличии.

## 3.4.2. Эндпоинты (логическая спецификация)

- `GET /api/v2/custom_integrations/{integration_id}/secrets`
  - **Response**: список секретов с маской/ `is_set`.
- `POST /api/v2/custom_integrations/{integration_id}/secrets`
  - **Request**: `name`, `value`, `is_confidential`
  - **Response**: созданный секрет (без value, только маска/ `is_set`).
- `PUT /api/v2/custom_integrations/{integration_id}/secrets/{secret_id}`
  - **Request**: `name?`, `value?`, `is_confidential?`
  - **Response**: обновлённый секрет (без value).
- `DELETE /api/v2/custom_integrations/{integration_id}/secrets/{secret_id}`
  - **Response**: признак успеха.

# 3.5. Безопасность и конфиденциальность

- Секрет хранится **только в зашифрованном виде** (ключ шифрования — системный ключ приложения).
- Доступ к управлению секретами — только пользователям с правами настройки интеграций (конкретные права/роли берутся из существующей модели доступа продукта).
- Поле `is_confidential` обязано сохраняться и возвращаться в API (в дальнейшем будет использоваться для контроля доступа/аудита при использовании секретов в шаблонах/вычислениях).
- Логи/трассировки:

- запрещено логировать `value` и расшифрованные секреты;
- в ошибках допускается логировать только идентификаторы ( `integration_id` , `secret_id` , `name` ) и причины.

## 3.6. Граничные случаи

- **Дубликат имени** внутри одной интеграции ( `(custom_integration_id, name)` ): возвращать ошибку валидации.
- **Переименование секрета**: должно учитывать уникальность по имени.
- **Удаление интеграции**: секреты должны удаляться автоматически.
- **Миграции/обратная совместимость**:
  - существующие интеграции без секретов должны работать без изменений;
  - `computed_fields` по умолчанию пустой массив.

# 4. Критерии приёмки

- В БД создана таблица `custom_integration_secrets` с указанными полями, индексами и уникальностью `(custom_integration_id, name)` .
- В `custom_integrations` добавлено поле `computed_fields` (JSON) с дефолтом `[]` .
- Секрет можно создать/обновить/удалить через API.
- В API и логах **ни при каких условиях** не появляется реальное значение секрета (только маска/ `is_set` ).
- При попытке создать секрет с тем же `name` в рамках одной интеграции возвращается понятная ошибка валидации.