TASK #5

ЗАДАЧА

1. Развернуть сервис на Golang, Postgres, Clickhouse, Nats (альтернатива kafka), Redis
2. Описать модели данных и миграций
3. В миграциях Postgres
   a. Проставить primary-key и индексы на указанные поля
   b. При добавлении записи в таблицу устанавливать приоритет как макс приоритет в таблице +1. Приоритеты начинаются с 1
   c. При накатке миграций добавить одну запись в Campaigns таблицу по умолчанию
      i. id = serial
      ii. name = Первая запись
4. Реализовать CRUD методы на GET-POST-PATCH-DELETE данных в таблице ITEMS в Postgres
5. При редактировании данных в Postgres ставить блокировку на чтение записи и оборачивать все в транзакцию. Валидируем поля при редактировании.
6. При редактировании данных в ITEMS инвалидируем данные в REDIS
7. Если записи нет (проверяем на PATCH-DELETE), выдаем ошибку (статус 404)
   a. code = 3
   b. message = “**errors.item.notFound**“
   c. details = {}
8. При GET запросе данных из Postgres кешировать данные в Redis на минуту. Пытаемся получить данные сперва из Redis, если их нет, идем в БД и кладем их в REDIS
9. При добавлении, редактировании или удалении записи в Postgres писать лог в Clickhouse через очередь Nats (альтернатива kafka). Логи писать пачками в Clickhouse

МИГРАЦИИ POSTGRES

1. **Синий цвет** - primary key
2. * - индоссируемые параметры
3. C - создание
4. U - обновление
5. R - обязательное поле или нет при редактировании

CAMPAIGNS

Атрибут	Тип	Описание	Пример	C	U	R
id *	int	id записи	1	+		+
name	string	название	Запись 1	+	+	+

Добавляем при старте приложения сразу одну запись в таблицу

ITEMS

Атрибут	Тип	Описание	Пример	C	U	R
id *	int	id записи	1	+		+
campaign_id *	int	id кампании	1	+		+
name *	string	название	Запись 1	+	+	+
description	string	описание	Описание		+	
priority	int	приоритет	1	max+1	+	+
removed	bool	статус удаления	false	false	+	+
created_at	timestamp	дата и время	timestamp	now		+

МИГРАЦИИ CLICKHOUSE

Атрибут	Тип	Описание	Пример
Id *	int	идентификатор	1
CampaignId *	int	идентификатор	1
Name *	string	название	Запись 1
Description	string	описание	Описание
Priority	int	приоритет	1
Removed	bool	статус удаления	false
EventTime	timestamp	дата и время	timestamp

REST API

POST /item/create

request

```
TYPE: Post
Header: empty
URL: campaignId=int

Payload: {
  "name": "string"
}
```

response

```
{
  "id": int,
  "campaignId": int,
  "name": "string",
  "description": "string",
  "priority": int,
  "removed": false,
```

```
  "createdAt": "timestamp"
}
```

PATCH /item/update

request

```
TYPE: Patch
Header: empty
URL: id=int, campaignId=int // проверяем, что запись есть

Payload: {
  "name": "string", // проверяем, чтобы не было пустым
  "description": "string" // необязательное поле
}
```

response

```
{
  "id": int,
  "campaignId": int,
  "name": "string",
  "description": "string",
  "priority": int,
  "removed": false,
  "createdAt": "timestamp"
}
```

DELETE /item/remove

request

```
TYPE: Patch
Header: empty
URL: id=int, campaignId=int // проверяем, что запись есть

Payload: {}
```

response

```
{
  "id": int,
  "campaignId": int,
  "removed": true
}
```

GET /items/list

request

```
TYPE: Patch
Header: empty
URL: empty

Payload: {}
```

response

```
{
  [{
    "id": int,
    "campaignId": int,
    "name": "string",
    "description": "string",
    "priority": int,
    "removed": false,
    "createdAt": "timestamp"
  }]
}
```

- если записей нет, возвращаем пустой массив