техосновы 4 Зименкова

На семинаре мы с командой выполняли Кейс 4 "Госуслуги", поэтому я возьму микросервисы из него для выполнения этой работы.

Описание микросервиса

Сервис уведомлений и коммуникаций
Функции:

- Отправка уведомлений (SMS, email, push, личный кабинет).
- Получение уведомлений по событиям других сервисов (статус заявок, долги, штрафы, ответы поддержки) для дальнейшей отправки пользователю.

Описание структуры таблиц БД

notifications - уведомления пользователям

- id INT PK
- user_id INT NOT NULL FK → user.id - ссылка на пользователя
- type VARCHAR(50) NOT NULL - sms/email/push/inbox (входящие в личном кабинете)
- status VARCHAR(30) NOT NULL - pending (в отправке)/sent (отправлено)/error (ошибка)
- title NVARCHAR(255) - заголовок (для email/push/inbox)
- message NVARCHAR(MAX) - текст сообщения
- destination NVARCHAR(255) - email/телефон/токен устройства
- created_at DATETIME NOT NULL - дата и время создания
- sent_at DATETIME NULL - дата и время отправки
- error_message NVARCHAR(500) NULL - сообщение об ошибке если отправка не удалась

event_queue - входящие события от других сервисов

- id INT PK
- user_id INT NOT NULL FK → user.id - ссылка на пользователя
- source_service_id INT NOT NULL FK → services.id - какой сервис прислал уведомление
- event_type VARCHAR(100) NOT NULL - тип события (обновление статуса, штраф, ...)

- payload NVARCHAR(MAX) NOT NULL - JSON-данные для построения уведомления
- processed BOOLEAN NOT NULL DEFAULT 0 - обработано (перешло в отправку) или нет
- created_at DATETIME NOT NULL - дата и время создания
- processed_at DATETIME NULL - дата и время обработки

templates - шаблоны уведомлений

- id INT PK
- code VARCHAR(100) NOT NULL - уникальный идентификатор (например "NEW_DOCUMENT") для логов
- type VARCHAR(50) NOT NULL - sms/email/push/inbox (входящие в личном кабинете)
- title_template NVARCHAR(255) NOT NULL - стандартный заголовок
- body_template NVARCHAR(255) NOT NULL - стандартный текст
- is_used BOOLEAN NOT NULL - используется ли сейчас этот шаблон или уже устарел
- created_at DATETIME NOT NULL - дата и время создания шаблона

API сервиса уведомлений и коммуникаций

1. Приём события от другого сервиса

POST /events/push

Сервис-продуцент (заявки, штрафы, платежи и т.д.) отправляет событие, которое нужно обработать и позже превратить в уведомление.

Формат запроса:

```
{
    "user_id": 1542,
    "source_service_id": 3,
    "event_type": "APPLICATION_STATUS_UPDATED",
    "payload": {
        "application_id": 91244,
        "old_status": "submitted",
        "new_status": "approved"
    }
}
```

Формат ответа:

```
{
   "result": true,
   "event_id": 4819
}
```

2. Получение списка всех необработанных событий

(используется системным воркером, который формирует уведомления)

GET /events/pending

Формат ответа:

```
{
   "result": true,
   "events": [
       {
           "id": 4819,
           "user_id": 1542,
           "source_service_id": 3,
           "event_type": "APPLICATION_STATUS_UPDATED",
           "payload": "{...}",
           "created_at": "2025-01-10T12:44:00"
       }
   ]
}
```

3. Обработка события → Создание уведомления через шаблон

Воркер вызывает ручку, чтобы по event_type и payload сгенерировать текст и добавить запись в notifications.

POST /events/process

Формат запроса:

```
{
   "event_id": 4819
}
```

Формат ответа:

```
{
   "result": true,
   "notification_id": 9931
}
```

4. Создать уведомление вручную (админ / системная логика)

Например, массовая рассылка или форсированные сообщения.

POST /notifications/create

Формат запроса:

```
{
   "user_id": 1542,
   "type": "email",
   "title": "Новый документ доступен",
   "message": "Ваш новый ПТС сформирован и доступен для
скачивания.",
   "destination": "test@example.com"
}
```

Формат ответа:

```
{
   "result": true,
   "notification_id": 9932
}
```

5. Получить список уведомлений пользователя

GET /notifications/user/{user_id}

Формат ответа:

```
{
   "result": true,
   "notifications": [
       {
           "id": 9931,
           "type": "email",
           "status": "pending",
           "title": "Статус вашей заявки обновлён",
           "message": "Заявка №91244 одобрена.",
           "created_at": "2025-01-10T12:50:00",
           "sent_at": null
       }
   ]
}
```

6. Отметить уведомление как отправленное

(после ответа внешнего провайдера)

PUT /notifications/{id}/mark-sent

Формат запроса:

```
{
   "provider_response": "OK"
}
```

Формат ответа:

```
{
   "result": true
}
```

7. Отметить уведомление как отправку с ошибкой

PUT /notifications/{id}/mark-error

Формат запроса:

```
{
   "error_message": "SMS provider timeout"
}
```

Формат ответа:

```
{
   "result": true
}
```

8. Получить список шаблонов уведомлений

GET /templates

Формат ответа:

```
{
   "result": true,
   "templates": [
       {
           "id": 5,
           "code": "APPLICATION_STATUS_UPDATED",
           "type": "email",
           "title_template": "Статус вашей заявки обновлён",
           "body_template": "Заявка №{{application_id}}: новый
статус - {{new_status}}",
           "is_used": true
       }
   ]
}
```

9. Получить шаблон по event_type

GET /templates/by-code/{code}

Формат ответа:

```
{
    "result": true,
    "template": {
        "id": 5,
        "code": "APPLICATION_STATUS_UPDATED",
        "type": "email",
        "title_template": "Статус вашей заявки обновлён",
        "body_template": "Заявка №{{application_id}}: новый
статус - {{new_status}}",
        "is_used": true
    }
}
```

10. Добавить новый шаблон

POST /templates/create

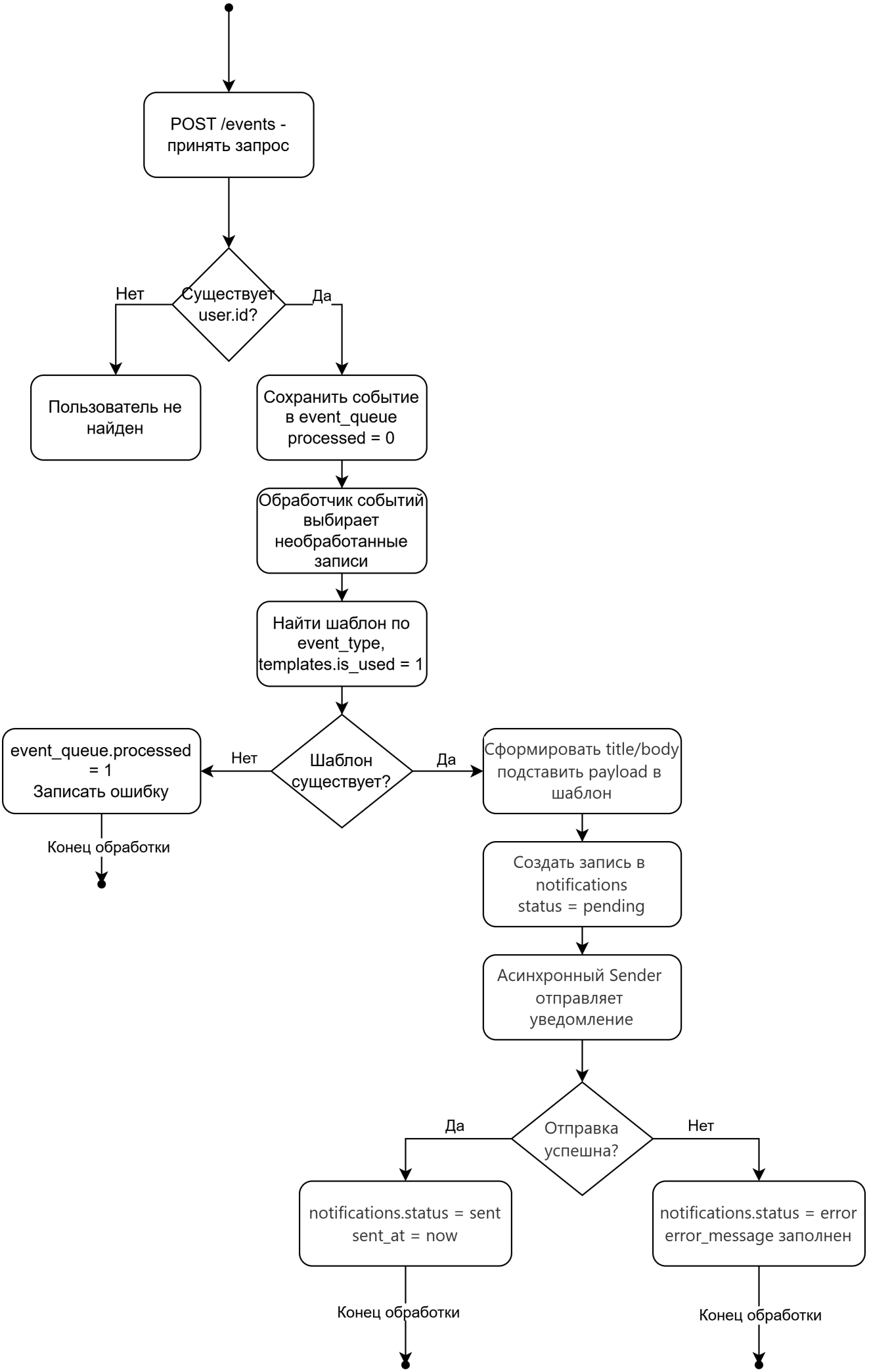
Формат запроса:

```
{
    "code": "NEW_FINE",
    "type": "sms",
    "title_template": "Уведомление о штрафе",
    "body_template": "У вас новый штраф: сумма {{amount}}.",
    "is_used": true
}
```

Формат ответа:

```
{
    "result": true,
    "template_id": 6
}
```

Диаграмма активности для POST /events


POST /events - принять запрос
Существует user.id?
Нет
Да
Пользователь не найден
Сохранить событие в event_queue processed = 0
Обработчик событий выбирает необработанные записи
Найти шаблон по event_type, templates.is_used = 1
Шаблон существует?
Нет
Да
event_queue.processed = 1
Записать ошибку
Конец обработки
Сформировать title/body подставить payload в шаблон
Создать запись в notifications status = pending
Асинхронный Sender отправляет уведомление
Отправка успешна?
Да
Нет
notifications.status = sent sent_at = now
Конец обработки
notifications.status = error error_message заполнен
Конец обработки