Отчет Блокировка Платежей

📌 Проблема:

Возникновение ситуаций, когда платежи в сервисе по их отправке надо приостановить, потому что клиент ненадежен (есть подозрение на мошенничество и нужно время разобраться), либо клиент предоставил не те реквизиты для перечисления и платежи отбиваются банком клиента (нужно время на выяснение верных реквизитов).

⚙️ Запрос:

- возможность заблокировать платежи конкретного клиента;
- возможность разблокировать платежи клиента;
- возможность проверить, заблокирован ли клиент;
- возможность отличать блокировки мошенников от блокировок добропорядочных клиентов.

💻 API-спецификация (Swagger):

Данный API позволяет управлять блокировкой клиентов в системе. Он предоставляет методы для:

✅ Блокировки клиента по причине мошенничества или некорректных реквизитов.

✅ Разблокировки клиента.

✅ Проверки статуса блокировки.

✅ Получения списка всех заблокированных клиентов.

Методы API:

Метод	Эндпоинт	Описание
POST	/block-client	заблокировать клиента
POST	/unblock-client	разблокировать клиента
GET	/client-status	проверить статус блокировки клиента
GET	/blocked-clients	получить список заблокированных клиентов

OpenAPI-документ:

```
openapi: 3.0.0
info:
  title: Client Payment Blocking API
  description: API для блокировки и разблокировки платежей клиентов.
```

```
  version: 1.0.0

paths:
  /block-client:
    post:
      summary: Заблокировать клиента
      description: Блокирует платежи клиента по указанной причине.
      requestBody:
        required: true
        content:
          application/json:
            schema:
              $ref: '#/components/schemas/BlockClientRequest'
      responses:
        '200':
          description: Клиент успешно заблокирован.
          content:
            application/json:
              schema:
                $ref: '#/components/schemas/BlockClientResponse'
        '400':
          description: Некорректные данные.

  /unblock-client:
    post:
      summary: Разблокировать клиента
      description: Снимает блокировку платежей с клиента.
      requestBody:
        required: true
        content:
          application/json:
            schema:
              $ref: '#/components/schemas/UnblockClientRequest'
      responses:
        '200':
          description: Клиент успешно разблокирован.
          content:
          application/json:
              schema:
                $ref: '#/components/schemas/UnblockClientResponse'
        '400':
```

```
        description: Некорректные данные.

  /client-status:
    get:
      summary: Проверить статус блокировки клиента
      description: Возвращает информацию о блокировке платежей для указанного клиента
      parameters:
        - name: client_id
          in: query
          required: true
          schema:
            type: string
      responses:
        '200':
          description: Информация о статусе клиента.
          content:
            application/json:
              schema:
                $ref: '#/components/schemas/ClientStatusResponse'

  /blocked-clients:
    get:
      summary: Получить список заблокированных клиентов
      description: Возвращает список всех заблокированных клиентов с деталями.
      responses:
        '200':
          description: Список заблокированных клиентов.
          content:
            application/json:
              schema:
                type: array
                items:
                  $ref: '#/components/schemas/BlockedClient'

components:
  schemas:
    BlockClientRequest:
      type: object
      required:
        - client_id
        - reason
```

```
    properties:
      client_id:
        type: string
        example: "12345"
      reason:
        type: string
        enum: [fraud, incorrect_details]
        example: "fraud"

  BlockClientResponse:
    type: object
    properties:
      client_id:
        type: string
        example: "12345"
      status:
        type: string
        example: "blocked"

  UnblockClientRequest:
    type: object
    required:
      - client_id
    properties:
      client_id:
        type: string
        example: "12345"

  UnblockClientResponse:
    type: object
    properties:
      client_id:
        type: string
        example: "12345"
      status:
        type: string
        example: "unblocked"

  ClientStatusResponse:
    type: object
    properties:
```

```
        client_id:
          type: string
          example: "12345"
        is_blocked:
          type: boolean
          example: true
        block_reason:
          type: string
          example: "fraud"
        blocked_at:
          type: string
          format: date-time
          example: "2024-03-20T12:00:00Z"

    BlockedClient:
      type: object
      properties:
        client_id:
          type: string
          example: "12345"
        block_reason:
          type: string
          example: "fraud"
        blocked_at:
          type: string
          format: date-time
          example: "2024-03-20T12:00:00Z"
```

Тестировние YAML-кода на платформе SwaggerEditor.

Client Payment Blocking API 1.0.0 OAS 3.0

API для блокировки и разблокировки платежей клиентов.

default

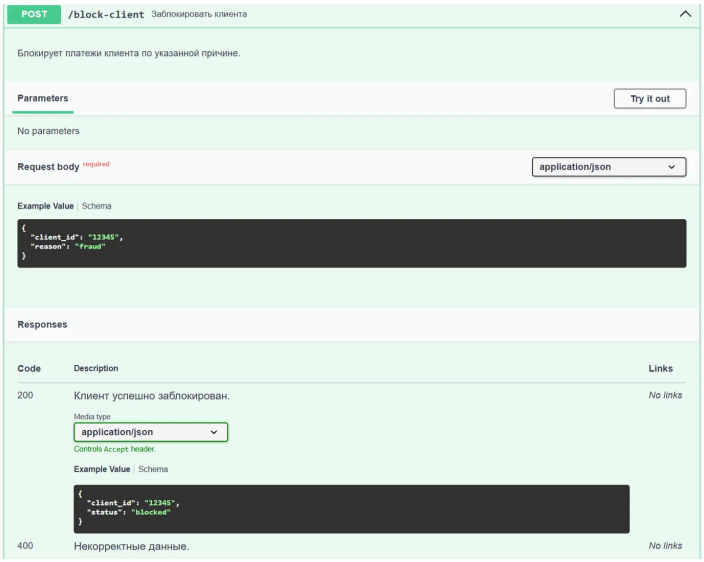
POST /block-client Заблокировать клиента

POST /unblock-client Разблокировать клиента

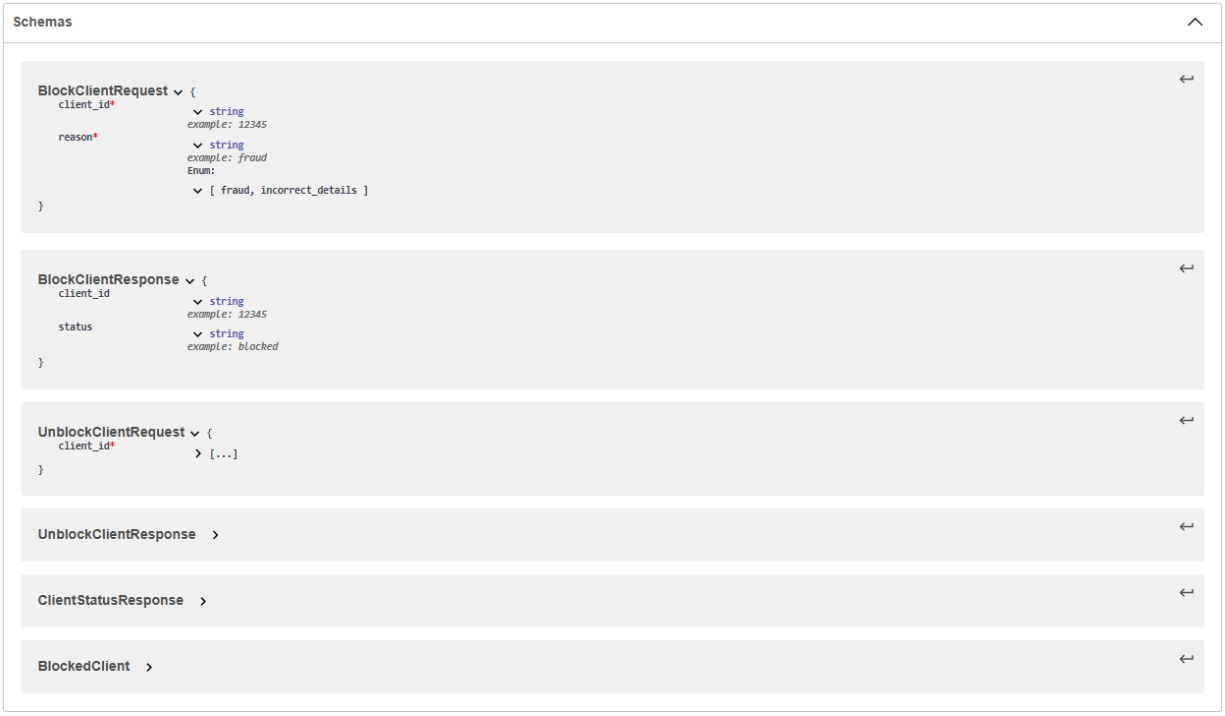
GET /client-status Проверить статус блокировки клиента

GET /blocked-clients Получить список заблокированных клиентов

Отображение всех эндпоинтов



Развернутое описание первого.



Компоненты, схемы (= структуры данных)

Примеры запросов и ответов API

Запрос: Блокировка клиента

POST /block-client
Content-Type: application/json

```
{
  "client_id": "12345",
  "reason": "fraud"
}
```

```
{
  "client_id": "1",
  "status": "blocked"
}
```

Запрос: Получение списка заблокированных клиентов

```
GET /blocked-clients
```

```
[
 {
  "client_id": "12345",
  "block_reason": "fraud",
  "blocked_at": "2024-03-20T12:00:00Z"
 },
 {
  "client_id": "67890",
  "block_reason": "incorrect_details",
  "blocked_at": "2024-03-21T09:30:00Z"
 }
]
```

🔐 Структура хранения нужной информации в базе данных

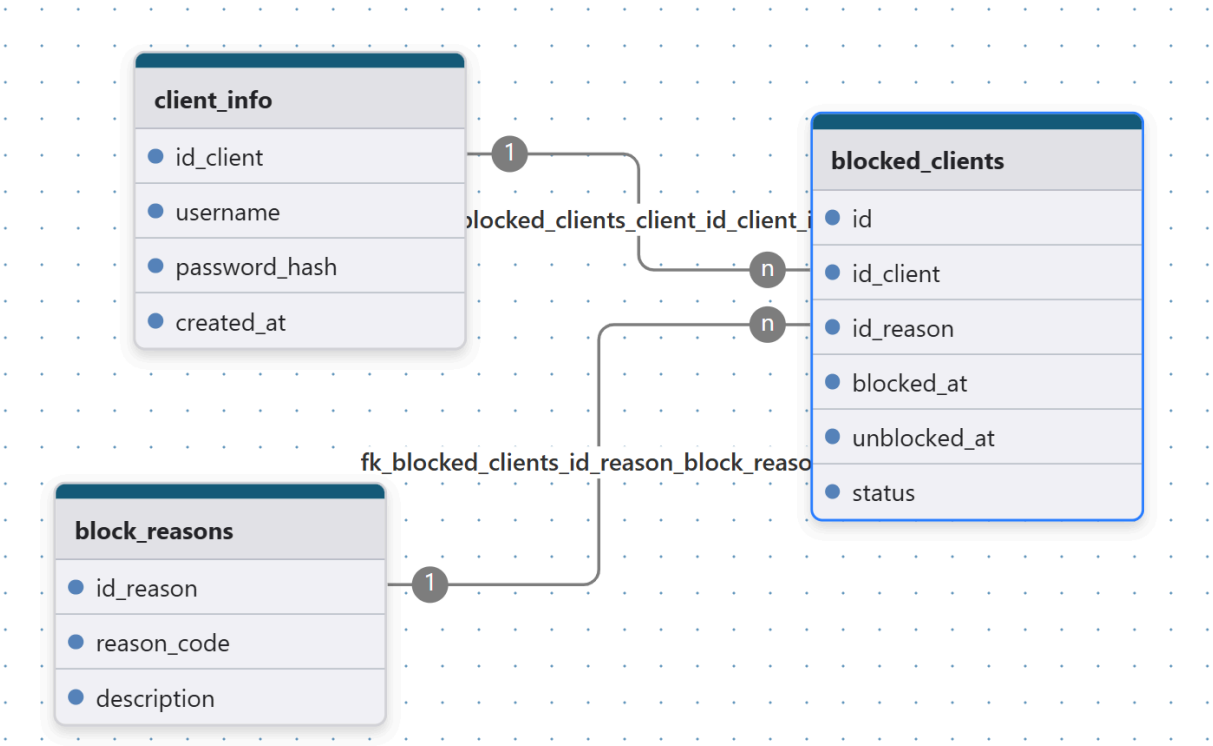


схема в case-системе drawDB

```
-- Таблица причин блокировки
CREATE TABLE block_reasons (
    id_reason SERIAL PRIMARY KEY,
    reason_code VARCHAR(20) NOT NULL,
    description TEXT,
    CONSTRAINT unique_reason_code UNIQUE (reason_code)
);

-- Таблица клиентов
CREATE TABLE client_info (
    id_client SERIAL PRIMARY KEY,
    username VARCHAR(100) NOT NULL,
    password_hash VARCHAR(100) NOT NULL,
    created_at TIMESTAMP WITH TIME ZONE DEFAULT CURRENT_TIMESTAMP,
    CONSTRAINT unique_username UNIQUE (username)
);

-- Таблица блокировок
CREATE TABLE blocked_clients (
    id SERIAL PRIMARY KEY,
    id_client INTEGER NOT NULL REFERENCES client_info(id_client),
    id_reason INTEGER NOT NULL REFERENCES block_reasons(id_reason),
    blocked_at TIMESTAMP WITH TIME ZONE DEFAULT CURRENT_TIMESTAMP,
    unblocked_at TIMESTAMP WITH TIME ZONE,
    status VARCHAR(10) NOT NULL CHECK (status IN ('blocked', 'unblocked')),

    CONSTRAINT fk_client FOREIGN KEY (client_id) REFERENCES client_info(id_client),
    CONSTRAINT fk_reason FOREIGN KEY (id_reason) REFERENCES block_reasons(id_reason)
);

-- Создаем индекс для быстрого поиска активных блокировок
CREATE INDEX idx_active_blocks ON blocked_clients (client_id) WHERE status = 'blocked';
```

Связи: один ко многим (`client_info` → `blocked_clients`), один ко многим (`block_reasons` → `blocked_clients`).

Вставляем данные:

```
INSERT INTO block_reasons (reason_code, description) VALUES
('fraud', 'Подозрение на мошенничество'),
```

```
('incorrect_details', 'Некорректные реквизиты');
```

id_reason	reason_code	description
1	fraud	подозрение на мошенничество
2	incorrect_details	некорректные реквизиты

```
INSERT INTO client_info (username, password_hash)
VALUES ('jl', 'hashed_password_12345');
```

id_client	username	password_hash	created_at
1	jl	hashed_password_12345	2024-03-23 12:00:00

Блокируем клиента

```
INSERT INTO blocked_clients (client_id, id_reason, status)
VALUES (1, 1, 'blocked');
```

id	id_client	id_reason	blocked_at	unblocked_at	status
1	1	1	2024-03-25 18:25:12	NULL	blocked

Проверяем, заблокирован ли клиент

```
SELECT c.username, bc.status, br.reason_code, bc.blocked_at
FROM blocked_clients bc
JOIN client_info c ON bc.client_id = c.id_client
JOIN block_reasons br ON bc.id_reason = br.id_reason
WHERE bc.status = 'blocked';
```

username	status	reason_code	blocked_at
jl	blocked	1	2024-03-25 18:25:12

Разблокировать клиента

```
UPDATE blocked_clients
SET status = 'unblocked', unblocked_at = NOW()
WHERE client_id = 1 AND status = 'blocked';
```

id	id_client	id_reason	blocked_at	unblocked_at	status
1	1	1	2024-03-25 18:25:12	2024-03-25 22:50:00	unblocked

Реализованная система позволяет эффективно управлять блокировками клиентов. API даёт возможность заблокировать и разблокировать клиента, а также проверить статус. База данных оптимизирована с помощью индексов и нормализации данных. В будущем можно добавить автоматическую разблокировку через определённое время.