Схема БД и готовая спецификация по OpenAPI приведены в конце файла.

Бизнес-требования:

1. Блокировка платежей клиента.
2. Разблокировка клиента.
3. Проверка статуса блокировки.
4. Разделение блокировок на типы: мошенничество (FRAUD) и ошибки в реквизитах (INCORRECT_DETAILS) – добропорядочные транзакции.

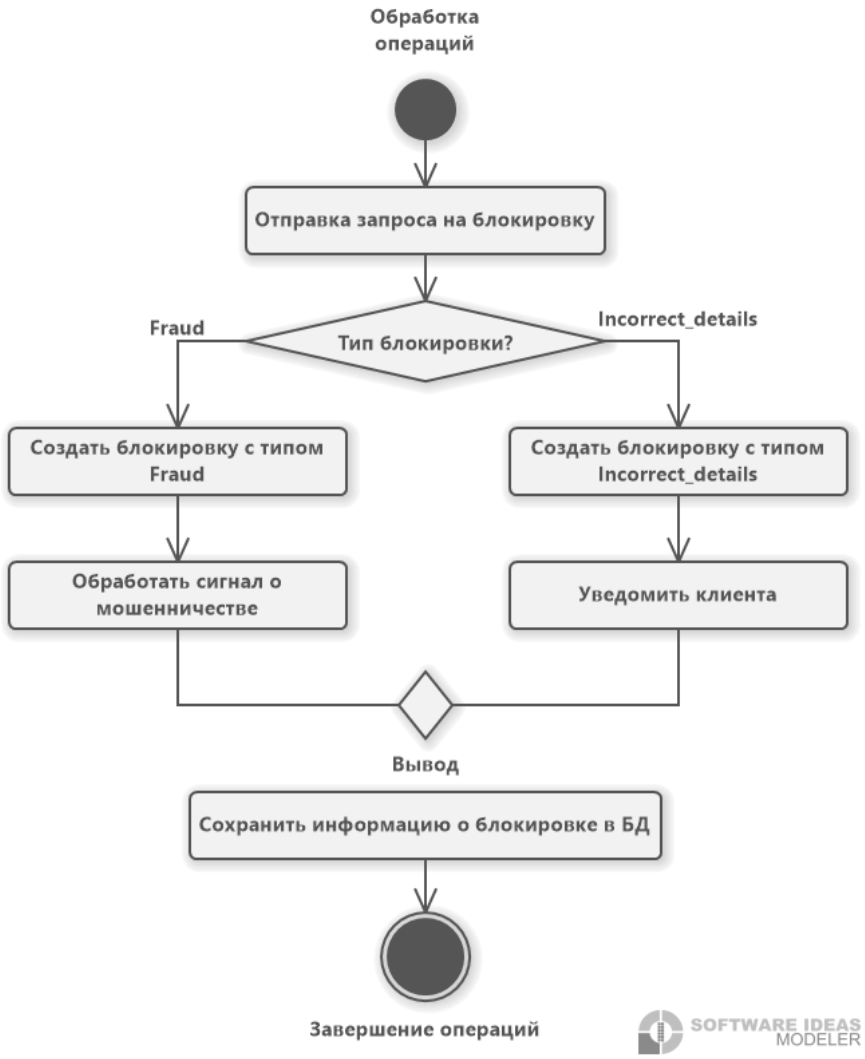
Функциональные требования:

- Оператор должен иметь возможность вручную заблокировать клиента через интерфейс.
- Аудит всех операций блокировки/разблокировки.

3. Концептуальные модели

3.1. Модель действий (Activity diagram)

Отображает предполагаемые процессы обработки блокировки. Как определение типа блокировки с соответствующими действиями в остальных местах системы, так и сохранение состояния блокировки в соответствующей БД(таблице).



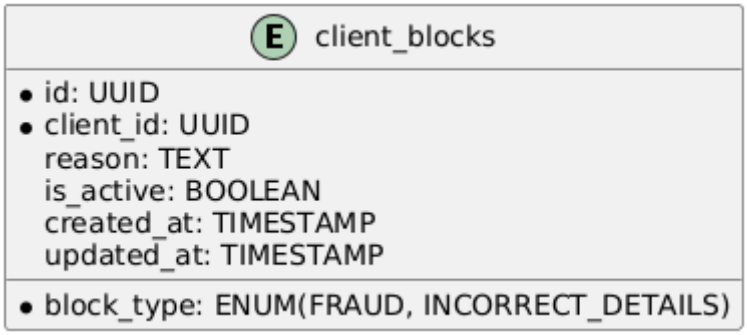
3.2. Use Case диаграмма

Показывает взаимодействие пользователей и других систем с системой. Предполагается возможность оператору или САУ после анализа операций клиента вводить блокировку и сопутствующие действия, возможность проверки статуса и разблокировки клиента для оператора

4. Проектирование

4.1. Диаграмма сущность-связь

Структура таблицы client_blocks. Так как данные либо являются производными блокировок(дата/время) или внешней ссылкой(client_id) то всю информацию для аудита блокировок возможно разместить в одной таблице:



4.2. REST API (OpenAPI)

Так как используется иерархия REST API на которой основана спецификация OpenAPI, то мы можем так кратко специфицировать эндпоинты:

- **POST** /clients/{clientId}/blocks – блокировка.
- **DELETE** /clients/{clientId}/blocks – разблокировка.
- **GET** /clients/{clientId}/blocks – проверка статуса.

Подробная OpenAPI спецификация:

```
openapi: 3.0.3
info:
  title: Client Block Service API
  version: 1.0.0
  description: API для управления блокировками платежей клиентов.

paths:
  /clients/{clientId}/blocks:
    post:
      summary: Заблокировать клиента
      description: Создает новую блокировку платежей для клиента. Если клиент уже заблокирован, возвращает ошибку.
      parameters:
        - name: clientId
          in: path
          required: true
          schema:
            type: string
            format: uuid
      requestBody:
        required: true
        content:
          application/json:
            schema:
              $ref: "#/components/schemas/BlockRequest"
      responses:
        201:
          description: Блокировка успешно создана.
          content:
            application/json:
```

```
        schema:
          $ref: "#/components/schemas/BlockResponse"
    409:
      description: Клиент уже заблокирован.
delete:
  summary: Разблокировать клиента
  description: Снимает активную блокировку клиента.
  parameters:
    - name: clientId
      in: path
      required: true
      schema:
        type: string
        format: uuid
  responses:
    200:
      description: Блокировка успешно снята.
    404:
      description: Активная блокировка не найдена.
get:
  summary: Проверить статус блокировки
  description: Возвращает информацию о текущей блокировке клиента.
  parameters:
    - name: clientId
      in: path
      required: true
      schema:
        type: string
        format: uuid
  responses:
```

```
      200:
        description: Статус блокировки.
        content:
          application/json:
            schema:
              $ref: "#/components/schemas/BlockStatusResponse"
      404:
        description: Клиент не заблокирован.

components:
  schemas:
    BlockType:
      type: string
      enum: [FRAUD, INCORRECT_DETAILS]
      description: Тип блокировки (мошенничество/некорректные реквизиты).

    BlockRequest:
      type: object
      required:
        - type
      properties:
        type:
          $ref: "#/components/schemas/BlockType"
        reason:
          type: string
          description: Описание причины блокировки.

    BlockResponse:
      type: object
      required: [id, clientId, type, createdAt, isActive]
```

```
  properties:
    id:
      type: string
      format: uuid
    clientId:
      type: string
      format: uuid
    type:
      $ref: "#/components/schemas/BlockType"
    reason:
      type: string
    createdAt:
      type: string
      format: date-time
    isActive:
      type: boolean

BlockStatusResponse:
  type: object
  required: [isBlocked, type, blockedAt]
  properties:
    isBlocked:
      type: boolean
    type:
      $ref: "#/components/schemas/BlockType"
    reason:
      type: string
    blockedAt:
      type: string
      format: date-time
```