Блокировки платежей (API + БД)

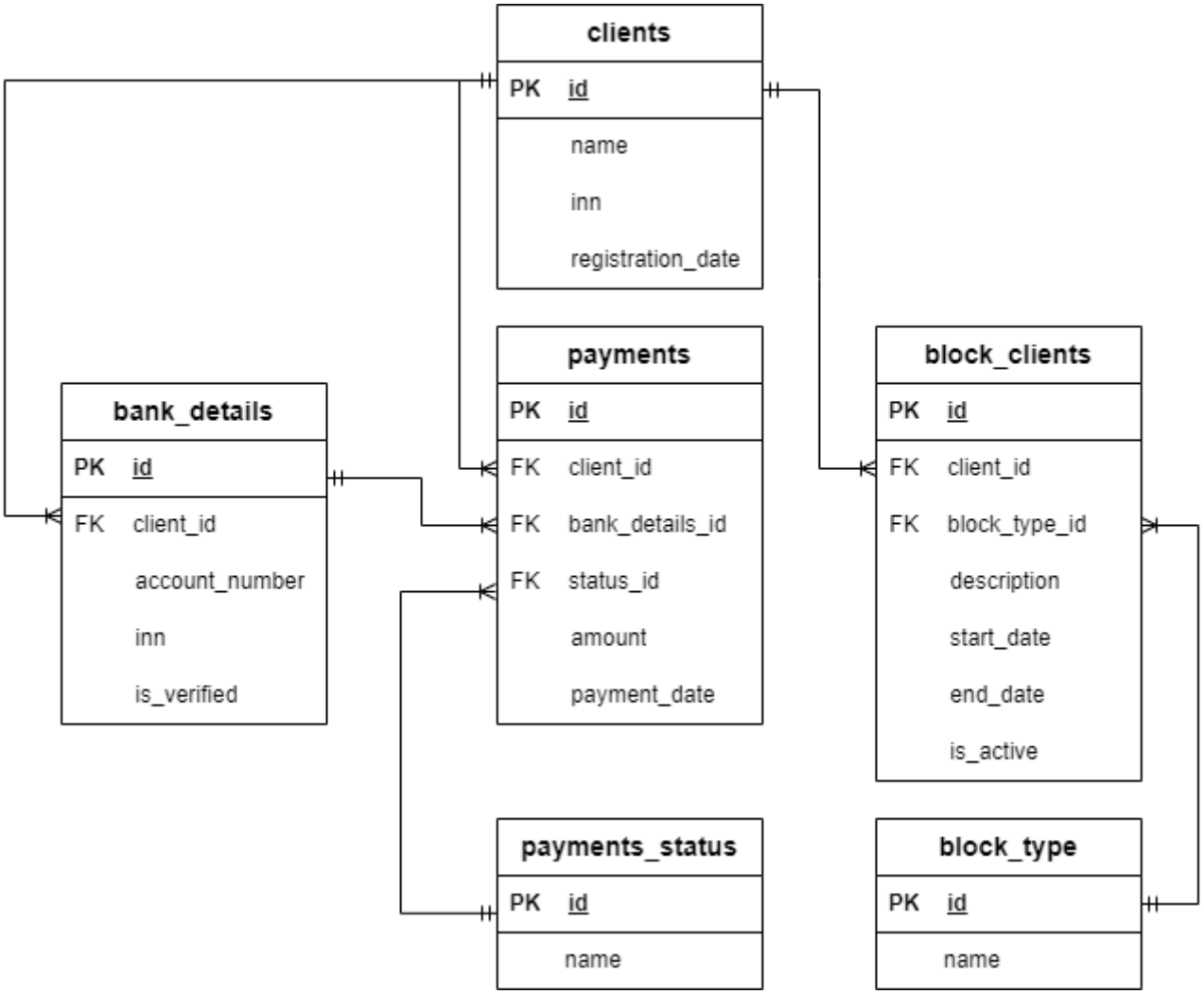
Оглавление

1.Структура БД

1.1 Допущения

1. Используемая СУБД - PostgreSQL
2. Справочники Статус платежа, Тип блокировки будем называть “справочники-классификаторы”.
3. Все таблицы расположены в одной базе.
4. Значимый атрибутивный состав таблицы Клиенты, Банковские реквизиты, Платежи упрощен, т.к. в данном задании нам не принципиальны полные данные.

1.2 Схема структуры базы данных



1.3 Таблицы

Таблица	Сущность	Комментарий
clients	Клиенты	Таблица с данными клиентов
bank_details	Банковские реквизиты	Таблица с данными банковских реквизитов клиентов Один клиент может иметь несколько счетов в разных банках (связь “один ко многим”)

payments	Платежи	Таблица с данными платежей в банк клиента Один платеж может совершаться, только на один расчетный счет банка получателя (связь “один ко многим”)
payments_status	Статус платежа	Для выбора значения в Платежи.Статус Один статус платежа может быть у нескольких платежей (связь “один ко многим”)
block_clients	Блокировка клиента	Таблица с данными о блокировках клиентов Одна блокировка может распространяться только на одного клиента (связь “один к одному”)
block_type	Тип блокировки	Для выбора значения в Блокировка клиента.Тип Один тип блокировки может быть у нескольких платежей (связь “один ко многим”)

1.3.1 clients (клиенты)

Свойство таблицы	**Атрибут сущности**	**Тип данных**	**Комментарий**
id	ИД	Bigserial	Первичный ключ. На формах не отображается. Не редактируется.
name	Название юр.лица	Varchar	Максимум 255 символов
inn	ИНН	Varchar	Максимум 10 символов
registration_date	Дата регистрации	Date	

1.3.2 bank_details (Банковские реквизиты)

Свойство таблицы	**Атрибут сущности**	**Тип данных**	**Комментарий**

id	ИД	Bigserial	Первичный ключ. На формах не отображается. Не редактируется.
client_id	ИД клиента	Bigint	Внешний ключ к таблице clients
account_number	Расчетный счет	Varchar	Максимум 20 символов
inn	ИНН	Varchar	Максимум 10 символов
is_verified	Подтверждены ли реквизиты	Boolean	По умолчанию FALSE

1.3.3 payments (Платежи)

Свойство таблицы	**Атрибут сущности**	**Тип данных**	**Комментарий**
id	ИД	Bigserial	Первичный ключ. На формах не отображается. Не редактируется.
client_id	ИД клиента	Bigint	Внешний ключ к таблице clients
bank_details_id	ИД банковских реквизитов	Bigint	Внешний ключ к таблице bank_details
amount	Сумма платежа	Decimal(11, 2)	Значение с плавающей точкой, с 9 значениями до запятой и 2 после
payment_date	Дата и время отправки платежа	Timestamp	
status_id	Статус платежа	Smallint	Внешний ключ к таблице payments_status

1.3.4 block_clients (блокировка клиентов)

Свойство таблицы	Атрибут сущности	Тип данных	Комментарий
id	ИД	Bigserial	Первичный ключ. На формах не отображается. Не редактируется.
client_id	ИД клиента	Bigint	Внешний ключ к таблице clients. Уникальное значение.
block_type_id	ИД типа блокировки	Bigint	Внешний ключ к таблице block_type
description	Комментарий	Varchar	Максимум 255 символов По умолчанию NULL
start_date	Дата и время блокировки	Timestamp	Дата и время начала блокировки клиента
end_date	Дата и время снятия ограничений	Timestamp	Дата и время автоматического снятия временного ограничения. По умолчанию NULL
is_active	Активный статус блокировки	Boolean	По умолчанию TRUE

1.4 Таблицы справочников-классификаторов

Имеют одинаковую структуру:

- payments_status (статус платежа)
- block_type (тип блокировки)

Свойство таблицы	Атрибут сущности	Тип данных	Комментарий
id	ИД	Bigserial	Первичный ключ. Не редактируется.
name	Название	Varchar	Максимум 64 символа

2. Архитектура сервиса

2.1 Классы

Справочник – класс, предназначенный для работы с данными. Не предполагает вовлеченности в бизнес-процесс.

- статус платежа
- тип блокировки

Тип объекта – класс, предназначенный для обработки объектов в бизнес-процессах.

- Платежи

Тип коллекции – класс, предназначенный для реализации подчиненной коллекции типа объекта.

- блокировка клиентов

Класс платформы - предназначен для предоставления форм просмотра, редактирования, табличной формы, др. общих функций.

Методы:

1. Формирование view табличной формы
2. Формирование view формы просмотра
3. Формирование view формы редактирования\ввода
4. Формирование view формы справочника (для выбора значения из справочника, для наполнения, просмотра справочника).
5. Формирование общего view модуля
6. …

Класс констант – для хранения константных значений и предоставления их в виде списков.

2.2 Идентификация / Аутентификация / Авторизация

Работа с пользователем – отдельный модуль.

В рамках описания доработок по данному бизнес-требованию не рассматривается.

2.3 Модули

Сервис может быть разбит на большее кол-во модулей, чтобы исключить усложнения тестового задания, считаем, как один.

Взаимодействие осуществляется с помощью REST API.

В данном случае рассмотрим подраздел блокировки клиента.

2.3.1 Блокировка клиента

Предназначен для работы с блокировкой платежей клиента.

2.3.2 Тип блокировки

Наследник класса констант платформы

Содержит значения:

- Мошенничество

- Некорректные реквизиты

Метод класса - предоставить список констант.

2.3.3 Свойства

Свойство	**Поле таблицы blocks_clients**	**Тип данных**	**Доступно для редактирования**	**Комментарий**
ИД	id	type: integer format: int64	Нет	Идентификатор
ИД клиента	client_id	type: integer	Да	Указывается в URL метода
Тип блокировки	block_type	type: integer format: int16	Да	Выбор из справочника “Тип блокировки”
Комментарий	description	type: string maxLength: 255	Да	Не обязательное поле. Содержит краткое описание причины блокировки
Дата и время блокировки	start_date	type: string format: ISO 8601	Нет	Проставляется автоматически при запросе.
Дата и время снятия ограничений	end_date	type: string format: ISO 8601	Да	Не обязательное поле. Указывается для автоматического снятия ограничений.
Активный статус блокировки	is_active	type: boolean	Нет	Проставляется автоматически при запросе

2.3.4 Методы

Метод	**Для чего**	**В спецификации**
Заблокировать платежи клиенту	Для блокировки осуществляемых клиенту платежей, как на время, так и на постоянной основе(кол-во не ограничено)	POST /api/client/{client_id}/block

Разблокировать платежи клиенту	Для разблокировки всех осуществляемых клиенту платежей	POST /api/client/{client_id}/unblock
Получить статус блокировки клиента	Просмотр статуса и всех блокировок у клиента	GET /api/client/{client_id}/is_block

3. Эндпоинты

3.1 Метод блокировки платежей клиента POST /api/client/{client_id}/block

3.1.1 Описание

Метод вызывается для блокировки платежей пользователей внутри сервиса, где в URL в качестве атрибута передается client_id - уникальный идентификатор юридического лица в системе.

Формат обмена данными – JSON.

Метод можно вызвать только после получения уникального идентификатора клиента.

Запрос выполняется во внутреннем сервисе с использованием технологии синхронного взаимодействия.

Участники информационного обмена используют POST метод с URL /client. В заголовке REST запроса указывается авторизационный токен.

Успешный вызов метода добавляет запись о блокировки в таблицу БД blocks_clients (администратор указывает: клиента, тип блокировки и дата/время автоматического снятия блокировки. Остальные параметры заполняются автоматически на основе правил). Допускаются добавление одному клиенту разных блокировок.

3.1.2 Входные данные

Содержание	**Тип данных**	**Обязательность**	**Описание**
block_type	integer	Да	Выбор из справочника “Тип блокировки”
description	string	Нет	Описание пользователем причины блокировки
end_date	string	Нет	Дата и время автоматического снятия блокировки Формат: YYYY-MM-DDThh:mm:ss

3.1.3 Выходные данные

Содержание	Тип данных	Обязательность	Описание
success	boolean	Да	Статус выполнения запроса
data			Обязательный блок, если success = TRUE
- block_id	integer	Да	Уникальный идентификатор записи блокировки в системе
error			Обязательный блок, если success = FALSE
- code	integer	Да	Код ошибки обработчика
- message	string	Да	Текст сообщения обработчика с причиной ошибки

3.1.4 Логика работы

После получение запроса в сервисе, происходят следующие проверки:

1. Поиск клиента по указанному в URL запросе идентификатору. Если ID по запрашиваемому клиенту не найден, тогда происходит ответ с сообщением HTTP кодом запроса 200:

```
{
   "data": null,
   "success": false,
   "error": {
      "code": "404",
      "message": "Клиент не найден"
   }
}
```

2. При наличии внутренних ошибок, сервис направляет ответ с HTTP кодом запроса 500. Пример сообщения:

```
{
   "data": null,
   "success": false,
   "error": {
      "code": "500",
      "message": "Произошло ошибка! Exception …"
   }
}
```

3. Если в теле запросы передан не корректно параметр справочного типа, то сервис вернет ответ с HTTP кодом запроса 200. Пример сообщения:


```
{
   "data": null,
   "success": false,
   "error": {
      "code": "404",
      "message": "Тип блокировки не найден"
   }
}
```


4. Если в теле запроса не передан обязательный параметр или в его название атрибута допущена ошибка, то сервис вернет ответ с HTTP кодом запроса 422. Пример сообщения:


```
{
   "detail": [
      {
         "type": "missing",
         "loc": [
            "body",
            "block_type"
         ],
         "msg": "Field required",
         "input": {}
      }
   ]
}
```


5. Если данные об клиенте найдены и все обязательные атрибуты переданы корректно, то в ответе приходит сообщение с HTTP кодом запроса 200 и идентификатором записи. Пример сообщения:


```
{
   "data": {
      "block_id": 8
   },
   "success": true,
   "error": null
}
```


3.2 Метод разблокировки платежей клиента POST /api/client/{client_id}/unblock

3.2.1 Описание

Метод вызывается для разблокировки платежей клиента внутри сервиса.
Успешное выполнение метода выставляет в БД blocks_clients флаг is_active == False и дату с временем снятия блокировки.

3.2.2 Входные данные

В URL в качестве атрибута передается client_id - уникальный идентификатор юридического лица в системе.

Тело запроса пустое (данный тип запроса позволяет гибко расширить функционал в дальнейшем)

3.2.3 Выходные данные

Содержание	**Тип данных**	**Обязательность**	**Описание**
success	boolean	Да	Статус выполнения запроса
data			Обязательный блок, если success = TRUE
- count_unblocked	integer	Да	Количество снятых блокировок у клиента
error			Обязательный блок, если success = FALSE
- code	integer	Да	Код ошибки обработчика
- message	string	Да	Текст сообщения обработчика с причиной ошибки

3.2.4 Логика работы

1. Поиск клиента по указанному в URL запросе идентификатору. Если ID по запрашиваемому клиенту не найден, тогда приходит ответ с сообщением HTTP кодом запроса 200

2. При наличии внутренних ошибок, сервис направляет ответ с HTTP кодом запроса 500.

3. Если данные об клиенте найдены и клиент не заблокирован (is_active == False), то в ответе приходит сообщение с HTTP кодом запроса 200 и текстом ошибки. Пример сообщения:

```
{
   "data": null,
   "success": false,
   "error": {
      "code": "404",
      "message": "Активных блокировок не найдено"
   }
}
```

4. Если данные об клиенте найдены и клиент заблокирован (is_active == True), то в ответе приходит сообщение с HTTP кодом запроса 200 и количеством разблокированных записей. Пример сообщения:


```
{
   "data": {
      "count_unblocked": 2
   },
   "success": true,
   "error": null
}
```


3.3 Метод получения статуса блокировки клиента GET /api/client/{client_id}/is_block

3.3.1 Описание

Метод вызывается для получения информации о статусе блокировки платежей клиента внутри сервиса.
Носит информативный характер для службы поддержки пользователей и решения инцидентов с проблемой переводов.

3.3.2 Входные данные

В URL в качестве атрибута передается client_id - уникальный идентификатор юридического лица в системе.

3.3.3 Выходные данные

Содержание	Тип данных	Обязательность	Описание
success	boolean	Да	Статус выполнения запроса
data			Обязательный блок, если success = TRUE
- is_blocked	boolean	Да	Статус об успешном снятии блокировки
- block	array	Да	Объект, содержащий данные о блокировке клиента
- - type	string	Нет	Текст типа блокировки
- - description	string	Нет	Описание администратора при выставлении блокировки
- - end_date	string	Нет	Дата автоматического снятия блокировки.

			Формат: YYYY-MM-DDThh:mm:ss
error			Обязательный блок, если success = FALSE
- code	integer	Да	Код ошибки обработчика
- message	string	Да	Текст сообщения обработчика с причиной ошибки

3.3.4 Логика работы

1. Поиск клиента по указанному в URL запросе идентификатору. Если ID по запрашиваемому клиенту не найден, тогда происходит ответ с сообщением HTTP кодом запроса 200

2. При наличии внутренних ошибок, сервис направляет ответ с HTTP кодом запроса 500.

3. Если данные об клиенте найдены и клиент не заблокирован (is_active == False), то в ответе приходит сообщение с HTTP кодом запроса 200 и пустым объектом. Пример сообщения:


```
{
   "data": {
      "is_blocked": false,
      "block": {}
   },
   "success": true,
   "error": null
}
```


4. Если данные о клиенте найдены и клиент заблокирован (is_active == True), то в ответе приходит сообщение с HTTP кодом запроса 200 и массив объектов с данными. Пример сообщения:


```
{
   "data": [{
      "is_blocked": true,
      "block":
         {
            "type": "Мошенничество",
            "description": "Большое кол-во переводов за короткий промежуток",
            "end_date": null
         }
   }],
   "success": true,
   "error": null
}
```


ПРИЛОЖЕНИЕ

Запросы:

```
-- Создание таблицы clients
CREATE TABLE clients (
    id INTEGER PRIMARY KEY,
    name VARCHAR(255) NOT NULL,
    inn VARCHAR(10) NOT NULL,
    registration_date TIMESTAMP DEFAULT CURRENT_TIMESTAMP
);

-- Создание таблицы bank_details
CREATE TABLE bank_details (
    id INTEGER PRIMARY KEY,
    client_id INT NOT NULL REFERENCES clients(id) ON DELETE CASCADE,
    account_number VARCHAR(20) NOT NULL,
    inn VARCHAR(10) NOT NULL,
    is_verified BOOLEAN DEFAULT FALSE
);

-- Создание таблицы payments_status
CREATE TABLE payments_status (
    id INTEGER PRIMARY KEY,
    name VARCHAR(64) NOT NULL
);

-- Создание таблицы block_type
CREATE TABLE block_type (
    id INTEGER PRIMARY KEY,
    name VARCHAR(64) NOT NULL
);

-- Создание таблицы block_clients
CREATE TABLE block_clients (
    id INTEGER PRIMARY KEY,
    client_id INTEGER NOT NULL,
    block_type_id INTEGER NOT NULL,
```

```
    description VARCHAR(255),
    start_date TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
    end_date TIMESTAMP,
    is_active BOOLEAN DEFAULT TRUE,
    FOREIGN KEY (client_id) REFERENCES clients(id),
    FOREIGN KEY (block_type_id) REFERENCES block_type(id)
);

-- Создание таблицы payments
CREATE TABLE payments (
    id INTEGER PRIMARY KEY,
    client_id INT NOT NULL,
    bank_details_id INT NOT NULL,
    amount DECIMAL(11, 2) NOT NULL,
    payment_date TIMESTAMP NOT NULL,
    status_id INTEGER NOT NULL,
    FOREIGN KEY (bank_details_id, client_id) REFERENCES bank_details(id, client_id),
    FOREIGN KEY (status_id) REFERENCES payments_status(id)
);
```