Квалификационный раунд

Задача “Построитель отчетов”

Введение

Вам предстоит разработать с использованием Spring Framework бэкенд сервиса, реализующего функционал построителя отчетов. Разработка фронтенда системы в задачу не входит.
От системы требуется реализация API, описанного ниже.

Требования к API

Создание таблиц

Создание новой таблицы POST /api/table/create-table

№	Название	Тип	Обязательность	Описание
Входные параметры				
1	tableName	string	ИСТИНА	Имя таблицы
2	columnsAmount	integer	ИСТИНА	Количество полей
3	primaryKey	string	ИСТИНА	Ключ
4	columnInfos	list	ИСТИНА	Список столбцов
Выходные параметры				
1	result code	integer	ИСТИНА	Код результата

Описание

Метод предназначен для создания новой таблицы в рамках выбранной ранее базы данных. В параметре columnInfos должен быть передан список полей. Как минимум должна быть передана колонка, в которой находится первичный ключ, остальные колонки могут быть заведены уже отдельным запросом.

Особенности обработки

По результатам должна быть добавлена таблица с заданным именем. При невозможности добавления должны быть обработаны следующие ошибки:

- Таблица с таким именем уже существует
- Недопустимое имя таблицы
- Недопустимое название типа данных
- Список полей содержит недопустимые имена полей
- База данных недоступна
- Недостаточно прав на редактирование БД

При возникновении таких ошибок должен вернуться код ошибки 406.
Если ошибок нет, то должен вернуться код 201.

Примеры правильных JSON:

```
{
  "tableName": "Customer",
  "columnsAmount": 12,
  "columnInfos": [
    {
      "title": "CustomerId",
      "type": "int4"
    },
    {
      "title": "FirstName",
      "type": "VARCHAR(40)"
    },
    {
      "title": "LastName",
      "type": "VARCHAR(20)"
    },
    {
      "title": "Company",
      "type": "VARCHAR(80)"
    },
    {
      "title": "Address",
      "type": "VARCHAR(70)"
    },
    {
      "title": "City",
```

```
            "type": "VARCHAR(40)"
        },
        {
            "title": "Country",
            "type": "VARCHAR(40)"
        },
        {
            "title": "PostalCode",
            "type": "VARCHAR(10)"
        },
        {
            "title": "Phone",
            "type": "VARCHAR(24)"
        },
        {
            "title": "Fax",
            "type": "VARCHAR(24)"
        },
        {
            "title": "Email",
            "type": "VARCHAR(60)"
        },
        {
            "title": "SupportRepId",
            "type": "int4"
        }
    ],
    "primaryKey": "CustomerId"
}
```

```
{
    "tableName": "Artists",
    "columnsAmount": 3,
    "columnInfos": [
        {
            "title": "id",
            "type": "int4"
        },
        {
            "title": "name",
            "type": "varchar"
        },
        {
```

```
            "title": "age",
            "type": "int4"
        }
    ],
    "primaryKey": "id"
}
```

Примеры неправильных JSON:

```
{
    "tableName": "CustomerЛАЛА",
    "columnsAmount": 12,
    "columnInfos": [
        {
            "title": "CustomerId",
            "type": "int4"
        },
        {
            "title": "FirstName",
            "type": "VARCHAR(40)"
        },
        {
            "title": "LastName",
            "type": "VARCHAR(20)"
        },
        {
            "title": "Company",
            "type": "VARCHAR(80)"
        },
        {
            "title": "Address",
            "type": "VARCHAR(70)"
        },
        {
            "title": "City",
            "type": "VARCHAR(40)"
        },
        {
            "title": "Country",
            "type": "VARCHAR(40)"
        },
        {
            "title": "PostalCode",
```

```
            "type": "VARCHAR(10)"
        },
        {
            "title": "Phone",
            "type": "VARCHAR(24)"
        },
        {
            "title": "Fax",
            "type": "VARCHAR(24)"
        },
        {
            "title": "Email",
            "type": "VARCHAR(60)"
        },
        {
            "title": "SupportRepId",
            "type": "int4"
        }
    ],
    "primaryKey": "CustomerId"
}
```

```
{
    "tableName": "Artists",
    "columnsAmount": 3,
    "columnInfos": [
        {
            "title": "id",
            "type": "int4"
        },
        {
            "title": "name",
            "type": "varchar"
        },
        {
            "title": "age",
            "type": "int4"
        }
    ],
    "primaryKey": "zoo"
}
```

Получение структуры таблицы GET /api/table/get-table-by-name/{name}

№	Название	Тип	Обязательность	Описание
Входные параметры				
1	name	string(50)	ИСТИНА	Имя таблицы
Выходные параметры				
1	tableName	string	ИСТИНА	Имя таблицы
2	columnsAmount	integer	ИСТИНА	Количество полей
3	primaryKey	string	ИСТИНА	Ключ
4	columnInfos	list	ИСТИНА	Список столбцов

Описание

Метод предназначен для чтения структуры таблицы в рамках выбранной ранее базы данных. В параметре fields должен вернуться список полей.

Особенности обработки

По результатам должна быть добавлена таблица с заданным именем. При невозможности добавления должны быть обработаны следующие ошибки:

- Таблицы с таким именем не существует

Если таблицы с таким именем нет, то должен вернуться код 200, но тело ответа должно быть полностью пустым.
Если таблица с таким именем существует, должен прийти код 200 и выходной JSON,в котором значения полей title и type будут написаны в верхнем регистре, а primaryKey в нижнем. Значение type должно быть приведено к стандартному виду, смотрите в примерах ниже.

Пример правильного запроса на Artists:

```
{
  "tableName": "Artists",
  "columnsAmount": 3,
  "primaryKey": "id",
  "columnInfos": [
    {
      "title": "ID",
      "type": "INTEGER"
    },
    {
      "title": "NAME",
      "type": "CHARACTER VARYING"
    },
    {
      "title": "AGE",
      "type": "INTEGER"
    }
  ]
}
```

Пример правильного запроса на Customer:

```
{
  "tableName": "Customer",
  "columnsAmount": 12,
  "primaryKey": "customerid",
  "columnInfos": [
    {
      "title": "CUSTOMERID",
      "type": "INTEGER"
    },
    {
      "title": "FIRSTNAME",
      "type": "CHARACTER VARYING"
    },
    {
      "title": "LASTNAME",
      "type": "CHARACTER VARYING"
    },
    {
      "title": "COMPANY",
```

```
        "type": "CHARACTER VARYING"
      },
      {
        "title": "ADDRESS",
        "type": "CHARACTER VARYING"
      },
      {
        "title": "CITY",
        "type": "CHARACTER VARYING"
      },
      {
        "title": "COUNTRY",
        "type": "CHARACTER VARYING"
      },
      {
        "title": "POSTALCODE",
        "type": "CHARACTER VARYING"
      },
      {
        "title": "PHONE",
        "type": "CHARACTER VARYING"
      },
      {
        "title": "FAX",
        "type": "CHARACTER VARYING"
      },
      {
        "title": "EMAIL",
        "type": "CHARACTER VARYING"
      },
      {
        "title": "SUPPORTREPID",
        "type": "INTEGER"
      }
    ]
}
```

Удаление таблицы DELETE /api/table/drop-table-by-name/{name}

№	Название	Тип	Обязательность	Описание
Входные параметры				
1	name	string(50)	ИСТИНА	Имя таблицы
Выходные параметры				
1	result code	integer	ИСТИНА	Код результата

Описание

Метод предназначен для удаления таблицы в рамках выбранной ранее базы данных.Должен вернуться код результата.

Особенности обработки

По результатам должна быть удалена таблица с заданным именем. При невозможности удаления должны быть обработаны следующие ошибки:

- Таблицы с таким именем не существует

Если таблицы с таким именем нет, то должен вернуться код 406.
Если таблица с таким именем существует, должен прийти код 201.

Пример правильного запроса:

DELETE api/table/drop-table-by-name/Artists

Пример неправильного запроса:

DELETE api/table/drop-table-by-name/Artistshaha

Запросы таблиц

Создание нового запроса таблицы POST /api/table-query/add-new-query-to-table

№	Название	Тип	Обязательность	Описание
Входные параметры				
1	queryId	integer	ИСТИНА	id запроса
2	tableName	string(50)	ИСТИНА	Имя таблицы
3	query	string(120)	ИСТИНА	Запрос
Выходные параметры				
1	status	string	ИСТИНА	Код результата

Описание

Метод предназначен для создания sql-запроса в рамках выбранной таблицы.

Особенности обработки

По результатам должен быть добавлен sql-запрос для заданной таблице. При невозможности добавления должны быть обработаны следующие ошибки:

- Таблица с таким именем не существует
- Недопустимый формат id

При возникновении ошибок должен прийти код результата 406.
Если ошибок нет, то 201.

Примеры правильного JSON:

```
{
  "queryId": 1,
  "tableName": "Artists",
  "query": "раз dva"
}
```

```
{
   "queryId": 2,
   "tableName": "Artists",
   "query": "select * from Artists"
}
```

```
{
   "queryId": 3,
   "tableName": "Customer",
   "query": "select * from Customer"
}
```

Примеры неправильного JSON:

```
{
   "queryId": 1,
   "tableName": "Artist",
   "query": "раз dva"
}
```

```
{
   "queryId": abc,
   "tableName": "Artists",
   "query": "select * from Artists"
}
```

Изменение запроса таблицы PUT /api/table-query/modify-query-in-table

№	Название	Тип	Обязательность	Описание
Входные параметры				
1	queryId	integer	ИСТИНА	id запроса
2	tableName	string(50)	ИСТИНА	Имя таблицы
3	query	string(120)	ИСТИНА	Запрос
Выходные параметры				
3	status	integer	ИСТИНА	Код результата

Описание

Метод предназначен модификации sql-запроса в рамках выбранной таблицы и id.

Особенности обработки

По результатам должен быть изменен существующий запрос, привязанный к таблице. Привязка говорит о целевой таблице запроса. При невозможности добавления должны быть обработаны следующие ошибки:

- Таблицы с таким именем не существует
- Запроса с таким id не существует
- Недопустимый формат id

При возникновении ошибок должен прийти код результата 406.
Если ошибок нет, то 200.

Примеры правильного JSON:

```
{
  "queryId": 1,
  "tableName": "Artists",
  "query": "раз dva"
}
```

```
{
  "queryId": 2,
  "tableName": "Artists",
  "query": "select * from Artists"
}
```

```
{
  "queryId": 3,
  "tableName": "Customer",
  "query": "select * from Customer"
}
```

Примеры неправильного JSON:

```
{
  "queryId": 1,
  "tableName": "Artist",
  "query": "раз dva"
}
```

```
{
  "queryId": 12a,
  "tableName": "Artists",
  "query": "select * from Artists"
}
```

Изменение запроса таблицы DELETE /api/table-query/delete-table-query-by-id/{id}

№	Название	Тип	Обязательность	Описание
Входные параметры				
1	id	integer	ИСТИНА	id запроса
Выходные параметры				
3	status	integer	ИСТИНА	Код результата

Описание

Метод предназначен для удаления sql-запроса в рамках выбранной таблицы по id.

Особенности обработки

По результатам должен быть удален существующий запрос, привязанный к таблице. Привязка говорит о целевой таблице запроса. При невозможности удаления должны быть обработаны следующие ошибки:

- Запроса с таким id не существует

При возникновении ошибок должен прийти код результата 406.
Если ошибок нет, то 202.

Пример правильного запроса:

DELETE /api/table-query/delete-table-query-by-id/1

Пример неправильного запроса:

DELETE /api/table-query/delete-table-query-by-id/sting

Запуск запроса таблицы GET /api/table-query/execute-table-query-by-id/{id}

№	Название	Тип	Обязательность	Описание
Входные параметры				
1	id	integer	ИСТИНА	id запроса
Выходные параметры				
3	status	integer	ИСТИНА	Код результата

Описание

Метод предназначен для запуска sql-запроса в рамках выбранной таблицы по id.

Особенности обработки

По результатам должен быть запущен существующий запрос, привязанный к таблице. Привязка говорит о целевой таблице запроса. При невозможности запуска должны быть обработаны следующие ошибки:

- Запроса с таким id не существует
- Синтаксис запроса неверный

При возникновении ошибок должен прийти код результата 406.
Если ошибок нет, то 201.

Пример правильного запроса:

GET /api/table-query/execute-table-query-by-id/1

Пример неправильного запроса:

GET /api/table-query/execute-table-query-by-id/sting

Получение всех запросов таблицы GET /api/table-query/get-all-queries-by-table-name/{name}

№	Название	Тип	Обязательность	Описание
Входные параметры				
1	name	string	ИСТИНА	Имя таблицы
Выходные параметры				
3	tableQueries	list	ИСТИНА	Список запросов

Описание

Метод предназначен для получения всех sql-запросов в рамках выбранной таблицы имени таблицы.

Особенности обработки

По результатам должен быть получен JSON с запросами по конкретной таблице. Привязка говорит о целевой таблице запроса. При невозможности запуска должны быть обработаны следующие ошибки:

- Таблицы с таким именем не существует

Если таблицы не существует, должен вернуться полностью пустой ответ с кодом 200. Если ошибок нет, то код 200 со всеми запросами.

Пример правильного запроса:

GET /api/table-query/get-all-queries-by-table-name/Customer

Пример правильного ответа:

```
[
  {
    "queryId": 2,
    "tableName": "Customer",
    "query": "select * from Customer"
  },
  {
    "queryId": 3,
    "tableName": "Customer",
    "query": "select * from Customer where queryId = 2"
  }
]
```

Получение запроса таблицы по id GET /api/table-query/get-table-query-by-id/{id}

№	Название	Тип	Обязательность	Описание
Входные параметры				
1	id	integer	ИСТИНА	id таблицы
Выходные параметры				
1	queryId	integer	ИСТИНА	id запроса
2	tableName	string(50)	ИСТИНА	Имя таблицы
3	query	string(120)	ИСТИНА	Запрос

Описание

Метод предназначен для получения sql-запроса по id.

Особенности обработки

По результатам должен быть получен JSON с запросом по id. При невозможности запуска должны быть обработаны следующие ошибки:

- Запроса с таким id не существует

Если запроса не существует, ответ с кодом 500.
Если ошибок нет, то код 200 с запросом.

Пример правильного запроса:

GET /api/table-query/get-table-query-by-id/2

Пример правильного ответа:

```
{
  "queryId": 2,
  "tableName": "Customer",
  "query": "select * from Customer"
}
```

Пример неправильного запроса:

GET /api/table-query/get-table-query-by-id/20

Пример ответа на неправильный запрос:


```
{
    "timestamp": "2022-08-12T01:12:22.322+00:00",
    "path": "/api/table-query/get-table-query-by-id/20",
    "status": 500,
    "error": "Internal Server Error",
    "requestId": "f240e5ad-53"
}
```

Получение всех запросов GET

/api/table-query/get-all-table-queries

№	Название	Тип	Обязательность	Описание
Выходные параметры				
1	tableQueriesList	list	ИСТИНА	Все запросы

Описание

Метод предназначен для получения всех sql-запросов.

Особенности обработки

По результатам должен быть получен JSON со всеми запросами.

Пример правильного запроса:

GET /api/table-query/get-all-table-queries

Примеры правильного ответа:

```
[
  {
    "queryId": 2,
    "tableName": "Customer",
    "query": "select * from Customer"
  },
  {
    "queryId": 3,
    "tableName": "Customer",
    "query": "select * from Customer where queryId = 2"
  }
]
```

Если запросов нет:

```
[]
```

Универсальные запросы

Создание нового запроса таблицы POST /api/single-query/add-new-query

№	Название	Тип	Обязательность	Описание
Входные параметры				
1	queryId	integer	ИСТИНА	id запроса
2	query	string(120)	ИСТИНА	Запрос
Выходные параметры				
1	status	string	ИСТИНА	Код результата

Описание

Метод предназначен для создания sql-запроса.

Особенности обработки

По результатам должен быть добавлен sql-запрос. При невозможности добавления должны быть обработаны следующие ошибки:

- Таблица с таким именем не существует
- Недопустимый формат id

При возникновении ошибок должен прийти код результата 400.

Если ошибок нет, то 201.

Примеры правильного JSON:

```
{
    "queryId": 1,
    "query": "select * from Cars"
}
```

```
{
    "queryId": 2,
    "query": "select * from Motos"
}
```

```
{
  "queryId": 3,
  "query": "select * from Customer"
}
```

Примеры неправильного JSON:

```
{
  "queryId": abc,
  "query": "раз dva"
}
```

Пример ответа на неправильный JSON:

```
{
  "timestamp": "2022-08-12T01:32:09.533+00:00",
  "path": "/api/single-query/add-new-query",
  "status": 400,
  "error": "Bad Request",
  "requestId": "c493e828-7"
}
```

Изменение запроса PUT

/api/single-query/modify-query

№	Название	Тип	Обязательность	Описание
Входные параметры				
1	queryId	integer	ИСТИНА	id запроса
3	query	string(120)	ИСТИНА	Запрос
Выходные параметры				
3	status	integer	ИСТИНА	Код результата

Описание

Метод предназначен модификации sql-запроса.

Особенности обработки

По результатам должен быть изменен существующий запрос, привязанный к таблице. Привязка говорит о целевой таблице запроса. При невозможности добавления должны быть обработаны следующие ошибки:

- Запроса с таким id не существует
- Недопустимый формат id

При возникновении ошибок должен прийти код результата 406.
Если ошибок нет, то 200.

Примеры правильного JSON:

```
{
    "queryId": 1,
    "query": "select * from Planes"
}
```

Примеры неправильного JSON:

```
{
    "queryId": 1a,
    "query": "select * from Planes"
}
```

Удаление запроса DELETE /api/single-query/delete-single-query-by-id/{id}

№	Название	Тип	Обязательность	Описание
Входные параметры				
1	id	integer	ИСТИНА	id запроса
Выходные параметры				
3	status	integer	ИСТИНА	Код результата

Описание

Метод предназначен для удаления sql-запроса по id.

Особенности обработки

По результатам должен быть удален существующий запрос. При невозможности удаления должны быть обработаны следующие ошибки:

- Запроса с таким id не существует

При возникновении ошибок должен прийти код результата 406.
Если ошибок нет, то 202.

Пример правильного запроса:
DELETE /api/single-query/delete-single-query-by-id/1

Пример неправильного запроса:
DELETE /api/single-query/delete-single-query-by-id/sting

Запуск запроса GET /api/single-query/execute-single-query-by-id/{id}

№	Название	Тип	Обязательность	Описание
Входные параметры				
1	id	integer	ИСТИНА	id запроса
Выходные параметры				
3	status	integer	ИСТИНА	Код результата

Описание

Метод предназначен для запуска sql-запроса по id.

Особенности обработки

По результатам должен быть запущен существующий запрос. При невозможности запуска должны быть обработаны следующие ошибки:

- Запроса с таким id не существует
- Синтаксис запроса неверный

При возникновении ошибок должен прийти код результата 406.
Если ошибок нет, то 201.

Пример правильного запроса:

GET /api/single-query/execute-single-query-by-id/1

Пример неправильного запроса:

GET /api/single-query/execute-single-query-by-id/sting

Получение запроса по id GET /api/single-query/get-single-query-by-id/{id}

№	Название	Тип	Обязательность	Описание
Входные параметры				
1	id	integer	ИСТИНА	id таблицы
Выходные параметры				
1	queryId	integer	ИСТИНА	id запроса
3	query	string(120)	ИСТИНА	Запрос

Описание

Метод предназначен для получения sql-запроса по id.

Особенности обработки

По результатам должен быть получен JSON с запросом по id. При невозможности запуска должны быть обработаны следующие ошибки:

- Запроса с таким id не существует

Если запроса не существует, ответ с кодом 500.
Если ошибок нет, то код 200 с запросом.

Пример правильного запроса:

GET /api/single-query/get-single-query-by-id/1

Пример правильного ответа:

```
{
  "queryId": 1,
  "query": "select * from Cars"
}
```

Пример неправильного запроса:

GET /api/single-query/get-single-query-by-id/10

Пример ответа на неправильный запрос:


```
{
    "timestamp": "2022-08-12T01:47:20.416+00:00",
    "path": "/api/single-query/get-single-query-by-id/10",
    "status": 500,
    "error": "Internal Server Error",
    "requestId": "c493e828-11"
}
```

Получение всех запросов GET

/api/single-query/get-all-single-queries

№	Название	Тип	Обязательность	Описание
Выходные параметры				
1	querysList	list	ИСТИНА	Все запросы

Описание

Метод предназначен для получения всех sql-запросов.

Особенности обработки

По результатам должен быть получен JSON со всеми запросами.

Пример правильного запроса:

GET /api/single-query/get-all-single-queries

Примеры правильного ответа:

```
[
  {
    "queryId": 1,
    "query": "select * from Cars"
  },
  {
    "queryId": 2,
    "query": "select * from Motos"
  }
]
```

Если запросов нет:

```
[]
```

Создание отчетов

Запрос отчета по id GET /api/report/get-report-by-id/{id}

№	Название	Тип	Обязательность	Описание
Входные параметры				
1	id	integer	ИСТИНА	id отчета
Выходные параметры				
1	reportId	integer	ИСТИНА	id отчета
2	tableAmount	string	ИСТИНА	кол-во таблиц
3	tables	list	ИСТИНА	Список с информацией о таблицах в запросе

Описание

Метод предназначен для чтения структуры отчета в рамках выбранной ранее базы данных. В параметре tableInfo должен вернуться список полей таблицы в отчете. Поле size должно в себе содержать кол-во записей в данном столбце.

Особенности обработки

По результатам должен быть получен отчет. Должны быть обработаны следующие ошибки:

- Неверный формат reportId
- Отчет с таким reportId не существует

При возникновении ошибок должен вернуться код результата 406.
Если ошибок нет, то 201.

Пример правильного JSON:

```
{
  "reportId": 2,
  "tableAmount": 2,
  "tables": [
     {
       "tableName": "Artists",
       "columns": [
          {
```

```
            "title": "id",
            "type": "int4",
            "size": "10"
          },
          {
            "title": "name",
            "type": "varchar",
            "size": "10"
          },
          {
            "title": "age",
            "type": "int4",
            "size": "10"
          }
        ]
      },
      {
        "tableName": "Job",
        "columns": [
          {
            "title": "id",
            "type": "int4",
            "size": "10"
          },
          {
            "title": "name",
            "type": "varchar",
            "size": "10"
          },
          {
            "title": "salary",
            "type": "int4",
            "size": "10"
          },
          {
            "title": "address",
            "type": "varchar",
            "size": "10"
          }
        ]
      }
    ]
}
```

Формирование отчета по POST /api/report/create-report

№	Название	Тип	Обязательность	Описание
Выходные параметры				
1	reportId	integer	ИСТИНА	id отчета
2	tableAmount	integer	ИСТИНА	кол-во таблиц в отчете
3	tables	list	ИСТИНА	список таблиц с параметрами
Входные параметры				
1	status	integer	ИСТИНА	Код результата

Описание

Метод предназначен формирования отчета по заданным параметрам.

Особенности обработки

По результатам должен быть добавлен отчет. При невозможности добавления должны быть обработаны следующие ошибки:

- tableAmount не совпадает с кол-вом таблиц в списке tables
- Неверный формат reportId
- Отчет с таким reportId уже существует
- Таблицы с таким именем не существует
- Колонки с таким именем не существует
- Неверный тип колонки

При возникновении ошибок должен вернуться код результата 406.
Если ошибок нет, то 201.

Примеры правильных JSON:

```
{
  "reportId": 1,
  "tableAmount": 1,
  "tables": [
    {
      "tableName": "Artists",
      "columns": [
        {
          "title": "id",
          "type": "int4"
        },
        {
          "title": "name",
          "type": "varchar"
        },
        {
          "title": "age",
          "type": "int4"
        }
      ]
    }
  ]
}
```

```
{
  "reportId": 2,
  "tableAmount": 2,
  "tables": [
    {
      "tableName": "Artists",
      "columns": [
        {
          "title": "id",
          "type": "int4"
        },
        {
          "title": "name",
          "type": "varchar"
```

```
            },
            {
                "title": "age",
                "type": "int4"
            }
        ]
    },
    {
        "tableName": "Job",
        "columns": [
            {
                "title": "id",
                "type": "int4"
            },
            {
                "title": "name",
                "type": "varchar"
            },
            {
                "title": "salary",
                "type": "int4"
            },
            {
                "title": "address",
                "type": "varchar"
            }
        ]
    }
  ]
}
```

Примеры неправильных JSON:

```
{
  "reportId": 2,
  "tableAmount": 5,
  "tables": [
     {
        "tableName": "Artists",
        "columns": [
            {
                "title": "id",
                "type": "int4"
            },
```

```
            {
                "title": "name",
                "type": "varchar"
            },
            {
                "title": "age",
                "type": "int4"
            }
        ]
    },
    {
        "tableName": "Job",
        "columns": [
            {
                "title": "id",
                "type": "int4"
            },
            {
                "title": "name",
                "type": "varchar"
            },
            {
                "title": "salary",
                "type": "int4"
            },
            {
                "title": "address",
                "type": "varchar"
            }
        ]
    }
  ]
}
```

```
{
  "reportId": 1,
  "tableAmount": 1,
  "tables": [
     {
        "tableName": "!!LoremIpsum$$",
        "columns": [
            {
                "title": "id",
```

```
            "type": "dogs"
          },
          {
            "title": "name",
            "type": "varchar"
          },
          {
            "title": "age",
            "type": "int4"
          }
        ]
      }
  ]
}
```