

Александр Сырых

# Автоматизация конвейера подготовки отчетов: интеграция Firebird и n8n

# Обо мне

Ведущий программист отдела перспективных разработок компании ООО «Быстрые отчеты»

## Опыт:

разработка ПО и компонентов на:

- Delphi (VCL, FMX, Lazarus)
- C# (.NET, WinForms, WPF, Avalonia UI, ASP.NET Core)
- C++ (нативные плагины для Aurora OS)
- Dart (Flutter)
- JavaScript /TypeScript (React)

## DevOps

- разворачивание и настройка серверов
- настройка реверс прокси и сетей
- настройка CI/CD

## QA

- написание автотестов unit и UI



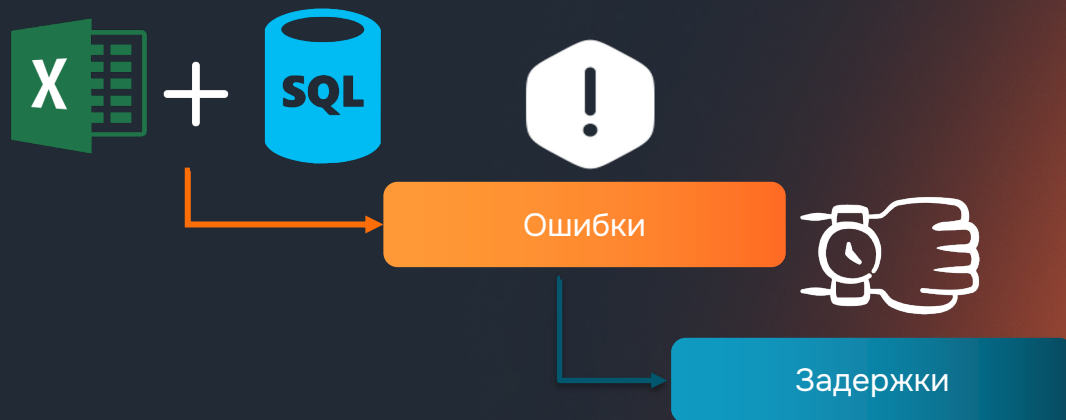
# Введение

Для чего это все нужно?  
А вы что-нибудь автоматизировали?



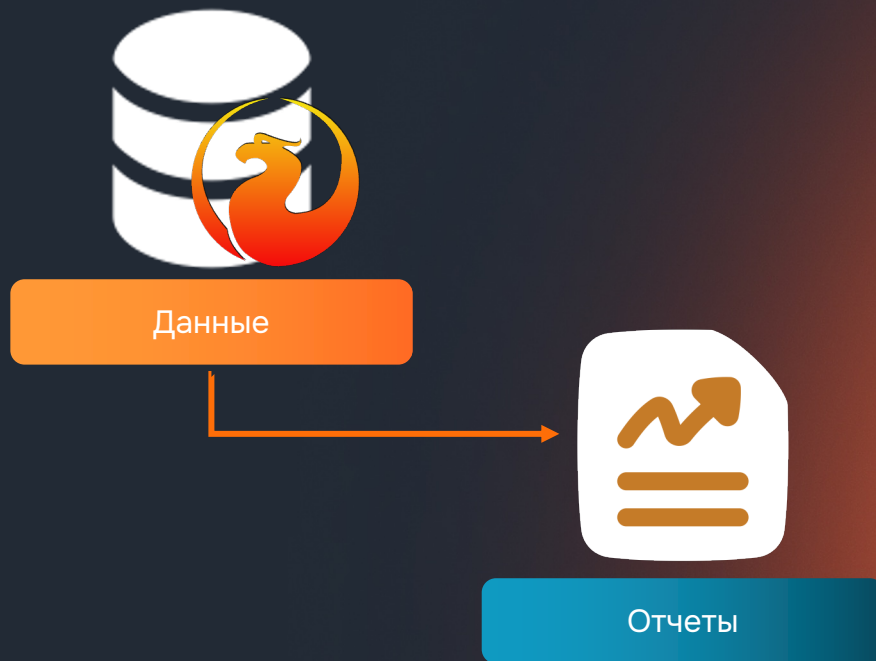
# Проблема

- ручная подготовка отчетов
- Excel + SQL
- ошибки
- задержки



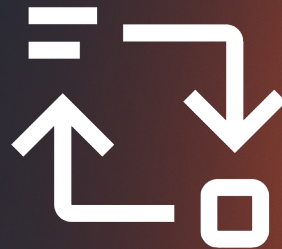
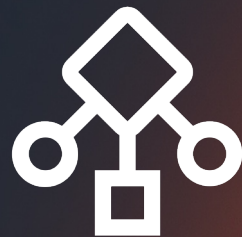
# Контекст Firebird

- Firebird уже используется
- данные есть
- система стабильна
  
- Но хочется быстро добавить отчетности без танцев с бубном и минимум кода



# Главная идея

- не трогаем Firebird
- строим конвейер (pipeline) вокруг него
- оставляем масштабируемость конвейера
- автоматизируем процессы подготовки отчета



# Вводная часть про n8n

А вы знаете про n8n? Или признаете только bash-скрипты и cron?



# Варианты автоматизации процессов

- написать свои скрипты и запускать по триггеру
- использовать git CI/CD
- использовать визуальные инструменты автоматизации:
  - Make
  - n8n
  - Zapier
  - Power Automate
  - RAGFlow

# Что такое n8n?

n8n – бесплатная платформа для автоматизации рабочих процессов и интеграций, ориентированная на технические команды. Это «low-code» решение с визуальным редактором и возможностью как облачного использования, так и полного self-hosting.

И по-простому n8n – это:

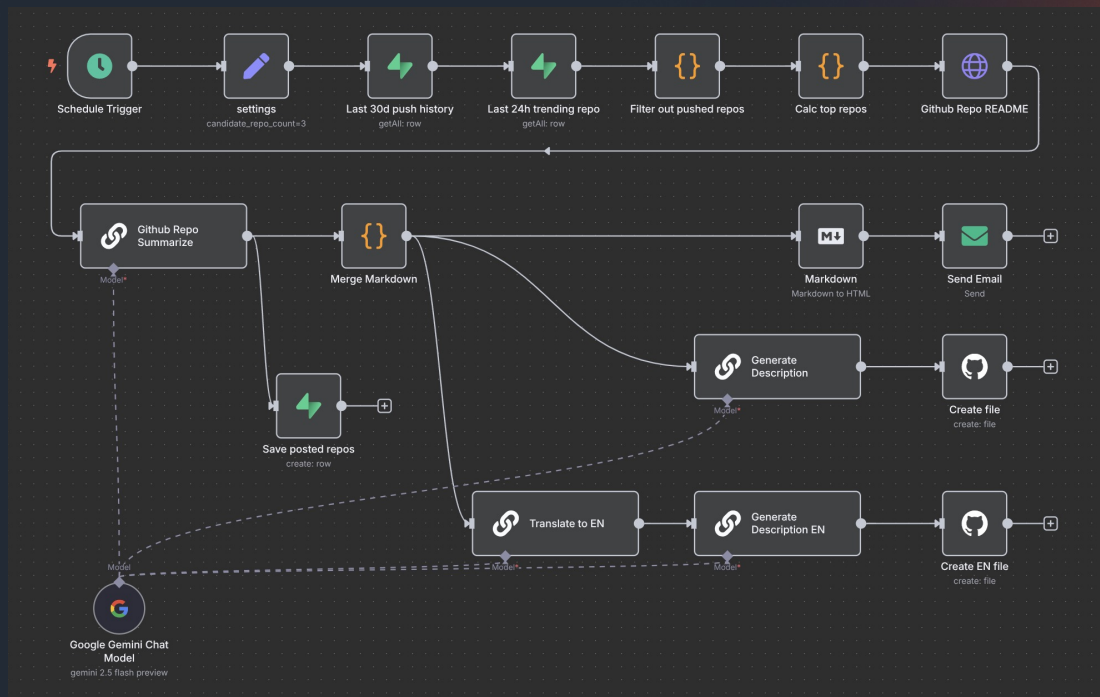
- инструмент автоматизации
- оркестрационный слой, который **координирует работу компонентов системы**
- визуальные рабочие процессы



# Как работает n8n

Типичный конвейер:

- Триггер
- Получение данных
- Обработка
- Отправка



# Зачем n8n в этом кейсе

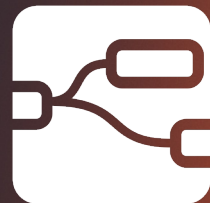
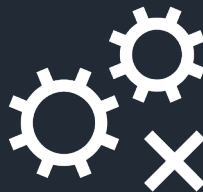
Firebird умеет:

- хорошо хранить данные ✓



Firebird не умеет:

- запускать процессы ✗
- отправлять email ✗
- строить конвейер ✗



# Архитектура решения

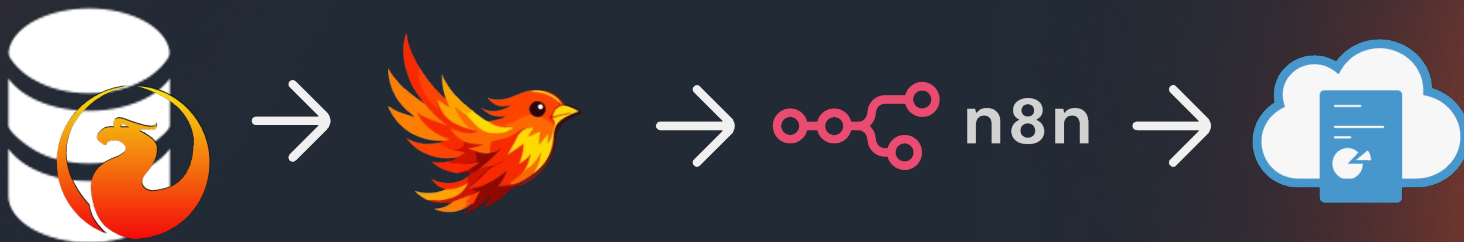
А схемы будут?



# Базовая схема

Вариант А:

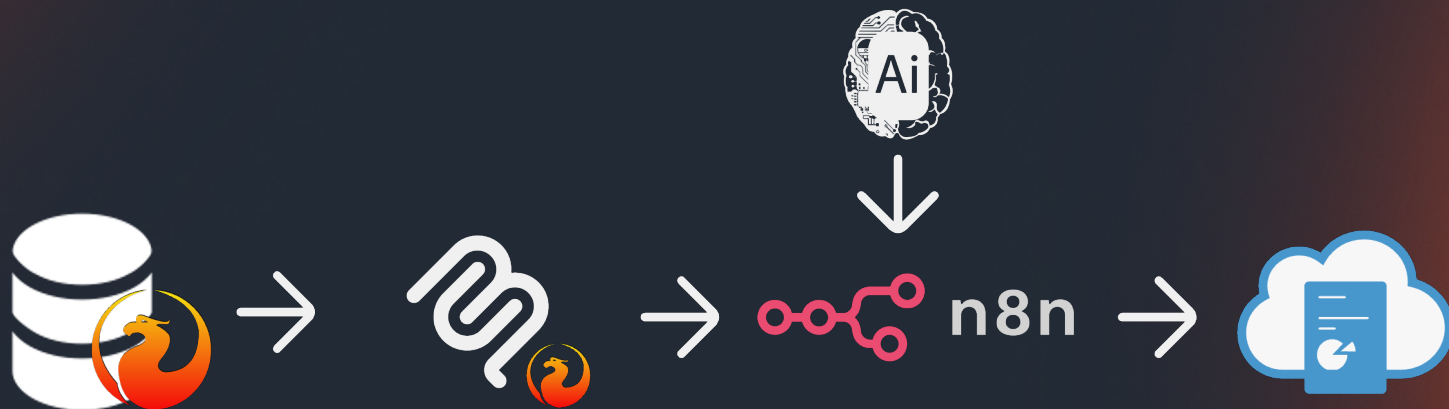
Firebird → API → n8n → МоиОтчеты Публикатор



# Расширенная схема

Вариант Б (опционально):

Firebird → MCP → n8n → AI → МоиОтчеты Публикатор



# Роли компонентов

Firebird → данные

API → доступ

n8n → управление

МоиОтчеты Публикатор → отчеты в PDF  
(или в 32 других формата)

AI → анализ данных

MCP → для взаимодействия языковых  
моделей (LLM) с внешними источниками  
данных и инструментами



# Где брать API Сервис?

Как преобразовать FireBird в готовую REST API-платформу?

А вы знаете про FastBirdieREST?



**FIREBIRD  
CONF'26**



# Что такое FastBirdieREST?

FastBirdieREST –  
open source REST API-платформа для Firebird.

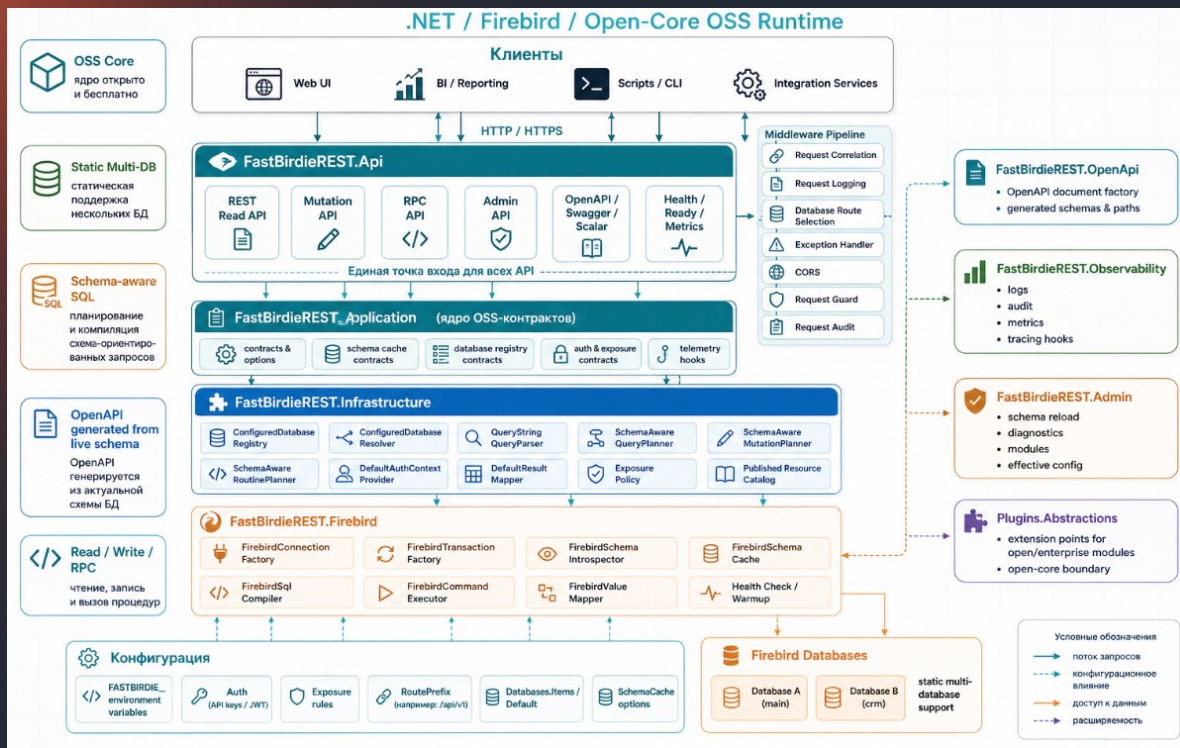
Написан на .NET 8.

FastBirdieREST превращает вашу базу Firebird  
в безопасную, масштабируемую и удобную  
для разработчиков REST API-платформу с:

- CRUD
- RPC
- OpenAPI (Swagger и Scalar)
- поддержкой нескольких статических подключений к БД



# Архитектура FastBirdieREST



# Как происходит запрос

## Поток запроса в FastBirdieREST OSS

От HTTP-запроса до Firebird и обратно



# Как это работает

1. Docker контейнер
2. Настройка подключения к FireBird
3. И все работает

FastBirdieREST  
автоматически сделает  
REST API для ваших таблиц  
и RPC для вызова процедур  
из вашей БД

**db:main** Published table and view routes for the default database. Unique routes: branch-daily-sales, branches, category-growth, daily-sales, management-monthly, management-overview, managers, receivables-snapshot, revenue-anomalies ^

GET	/fastbirdierest/db/main/branch-daily-sales	List rows from branch-daily-sales in database main.	🔒	⌵
GET	/fastbirdierest/db/main/branches	List rows from branches in database main.	🔒	⌵
POST	/fastbirdierest/db/main/branches	Insert rows into branches in database main.	🔒	⌵
PATCH	/fastbirdierest/db/main/branches	Update rows in branches in database main.	🔒	⌵
DELETE	/fastbirdierest/db/main/branches	Delete rows from branches in database main.	🔒	⌵

**rpc:main** Published routine routes for the default database. Unique routes: branch-daily-sales, branches, category-growth, daily-sales, management-monthly, management-overview, managers, receivables-snapshot, revenue-anomalies ^

GET	/fastbirdierest/db/main/rpc/category-growth	Invoke read-only routine category-growth in database main.	🔒	⌵
POST	/fastbirdierest/db/main/rpc/category-growth	Invoke routine category-growth in database main.	🔒	⌵

# Кому будет полезен FastBirdieREST?

К чему можно подключать:

- веб-приложениям
- мобильным приложениям
- партнёрским системам
- внутренним кабинетам и дашбордам
- средствам автоматизации
- внешним интеграциям

Продукт поможет компаниям:

- использовать уже существующую базу вместо дорогой и рискованной замены
- связывать старые системы с современными приложениями
- публиковать данные и бизнес-логику через API
- уменьшать объём кастомной backend-разработки
- стандартизировать доступ к данным
- готовить систему к дальнейшему росту и enterprise-интеграциям

# Где можно скачать FastBirdieREST?

- Лицензия Apache 2.0



# Демо (фин. отчетность)

Когда уже ближе к делу и кейсу?



**FIREBIRD  
CONF'26**



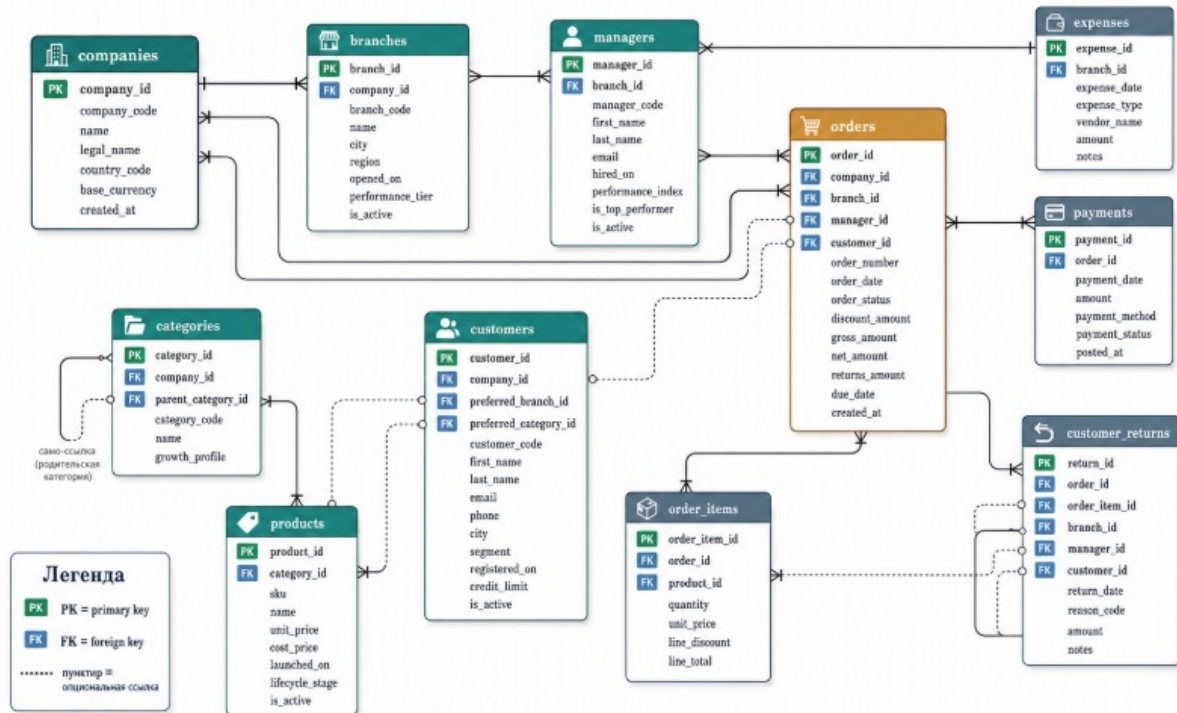
# Бизнес-сценарий

- продажи
- выручка
- дебиторка
- менеджеры



# Структура данных

## Схема таблиц БД Firebird Demo



# Почему этот кейс

- понятен всем
- легко показать эффект
- есть метрики

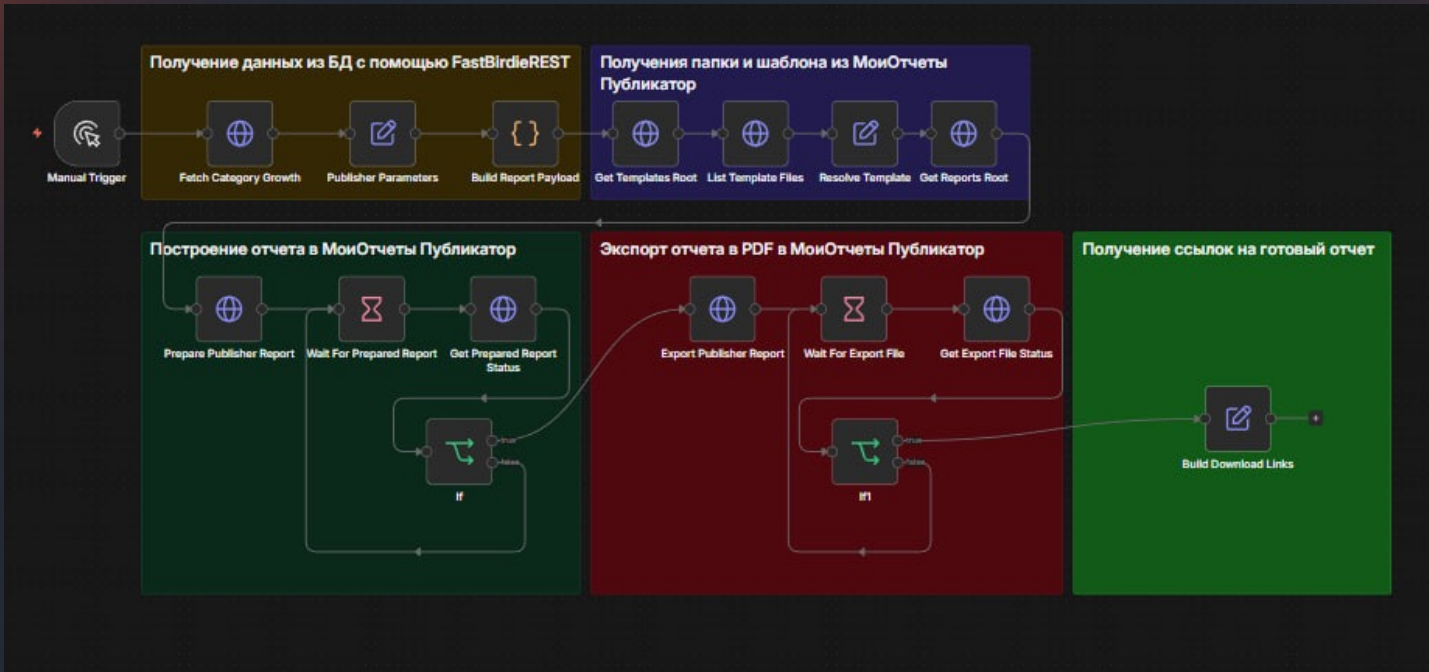


# Конвейер шаг за шагом

Когда уже ближе к делу и кейсу?



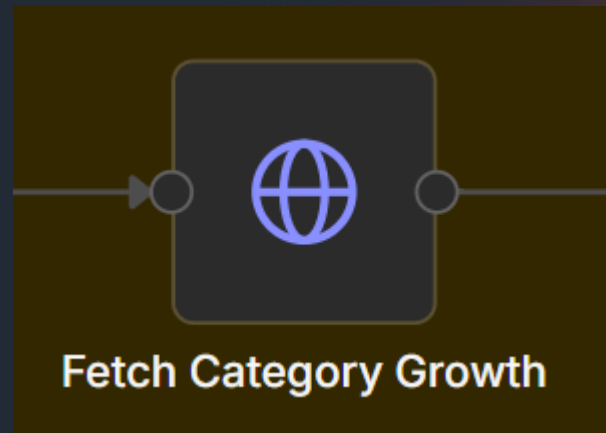
# Пример конвейера Отчет о росте категории



# Получение данных

n8n → FastBirdieREST → Firebird

```
REPORTING_API_BASE_URL +  
'/fastbirdierest/rpc/category-  
growth?p_date_from=2026-03-  
25&p_date_to=2026-04-01'
```



# Подготовка данных

- агрегаты
- КРІ
- группировки

```
1 const cfg = $(Publisher.Parameters).first().json;
2 const items = $(Category.Growth).size(1).json.items ?? [];
3 const formatMoney = (value) => Number(value || 0).toLocaleString('ru-RU', { style: 'currency', currency: 'RUB', minimumFractionDigits: 0, maximumFractionDigits: 2 });
4 const pct = (value) => `${Number(value || 0).toLocaleString('ru-RU', { minimumFractionDigits: 0, maximumFractionDigits: 1 })}%`;
5 const totalCurrent = items.reduce((acc, item) => acc + Number(item.REVENUE_CURRENT ?? 0), 0);
6 const totalPrevious = items.reduce((acc, item) => acc + Number(item.REVENUE_PREVIOUS ?? 0), 0);
7 const growing = items.filter((item) => Number(item.REVENUE_GROWTH_PCT ?? 0) > 0);
8 const declining = items.filter((item) => Number(item.REVENUE_GROWTH_PCT ?? 0) < 0);
9 const avgGrowth = items.length ? items.reduce((acc, item) => acc + Number(item.REVENUE_GROWTH_PCT ?? 0), 0) / items.length : 0;
10 const leaders = [...items].sort((a, b) => Number(b.REVENUE_GROWTH_PCT ?? 0) - Number(a.REVENUE_GROWTH_PCT ?? 0)).slice(0, 3);
11 const laggards = [...items].sort((a, b) => Number(b.REVENUE_GROWTH_PCT ?? 0) - Number(a.REVENUE_GROWTH_PCT ?? 0)).slice(0, 3);
12 const leader = leaders[0];
13 const laggard = laggards[0];
14 const topBranchesText = leaders.length ? leaders.map((item, index) => `${index + 1}. ${item.CATEGORY_NAME} | ${formatMoney(item.REVENUE_GROWTH_PCT)} | ${formatMoney(item.REVENUE_CURRENT)}).join('\n') : 'Категория не определена.';
15 const managementOverviewText = laggards.length ? laggards.map((item, index) => `${index + 1}. ${item.CATEGORY_NAME} | ${formatMoney(item.REVENUE_GROWTH_PCT)} | ${formatMoney(item.REVENUE_PREVIOUS)}).join('\n') : 'Категория не определена.';
16 const highlights = [
17   leader ? `* Top performer: ${leader.CATEGORY_NAME} | ${formatMoney(leader.REVENUE_GROWTH_PCT)} | ${formatMoney(leader.REVENUE_CURRENT)} : null,
18   laggard ? `* Low performer: ${laggard.CATEGORY_NAME} | ${formatMoney(laggard.REVENUE_GROWTH_PCT)} : null,
19   `Категория в покое: ${growing.length} vs ${items.length}`.
```

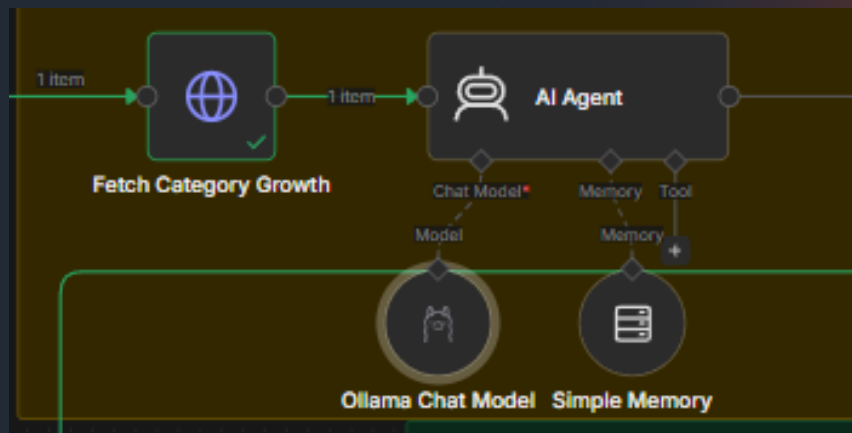
publisherId	templateId	templateName
category-growth-report-2026-04-01.txt	6f8a4a61d133353b3a7657659	category-growth-report.fx

# AI-предобработка

- инсайты
- аномалии
- объяснения

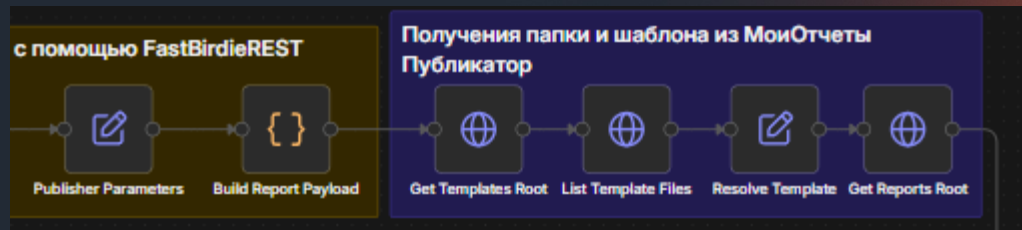
Можно использовать:

- локальные модели через Ollama
- Облачные модели



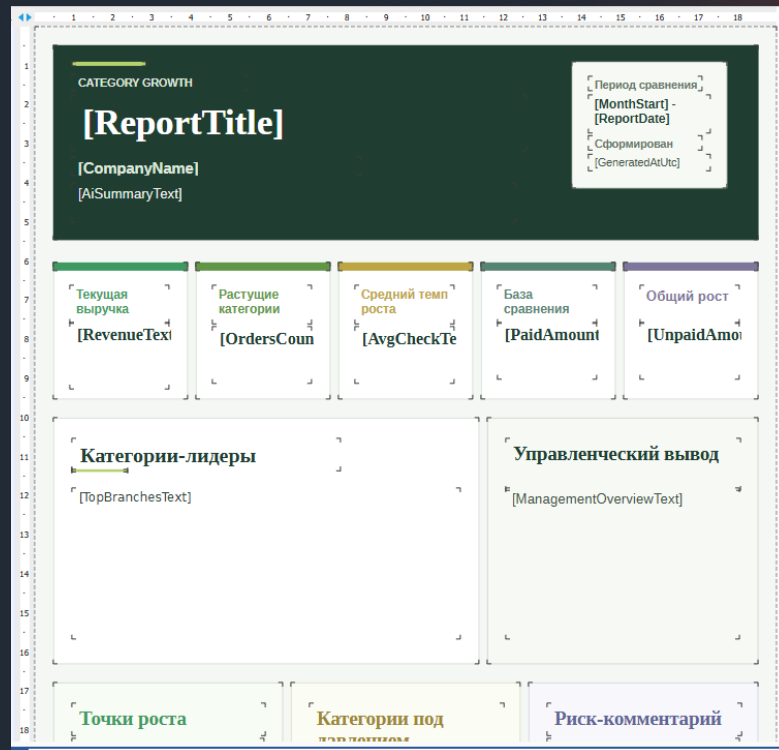
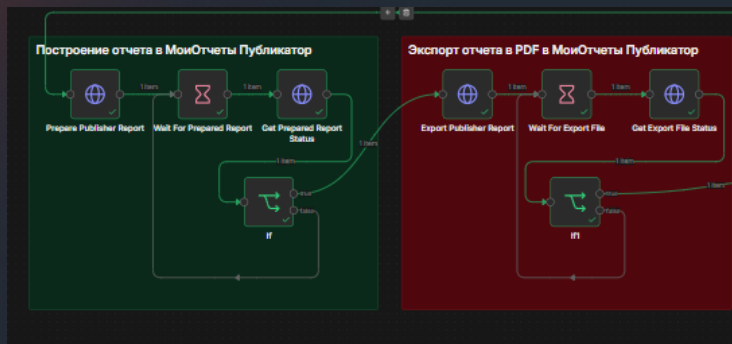
# Формирование запросов

- JSON
- структура под отчет



# Генерация отчета

- МоиОтчеты Публикатор
- шаблон (.frx)
- экспорт в pdf

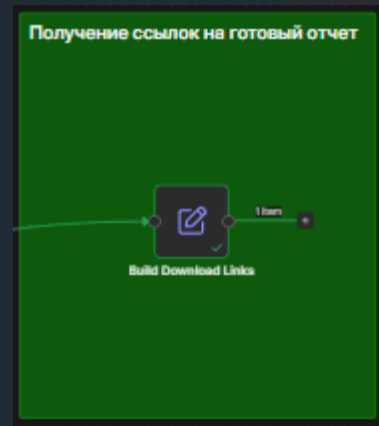


# Доставка

Несколько вариантов как через n8n, так и через МоиОтчеты

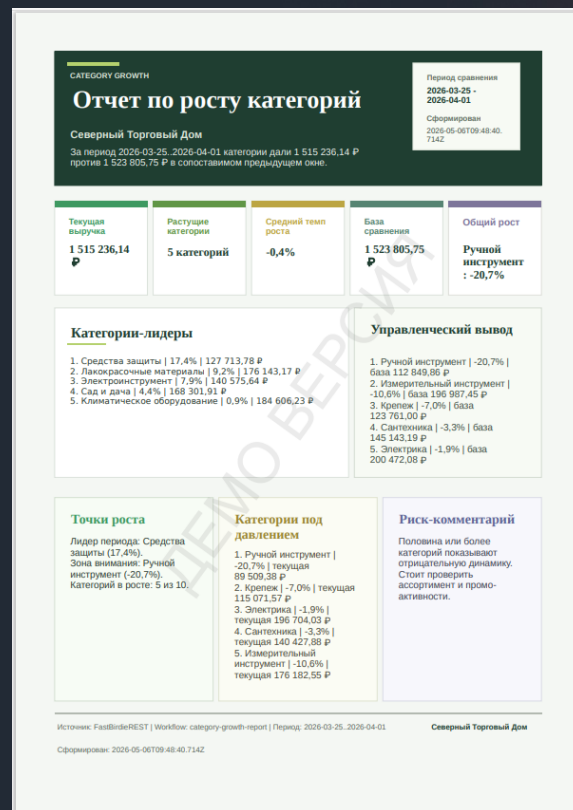
Публикатор

- Email
- Webhook
- storage



# Пример

- Лидер периода: Средства защиты (17,4%).
- Зона внимания: Ручной инструмент (-20,7%).
- Категорий в росте: 5 из 10.



# МСР как слой расширения

- гибкость
- добавление логики без изменения API
- AI-готовая архитектура

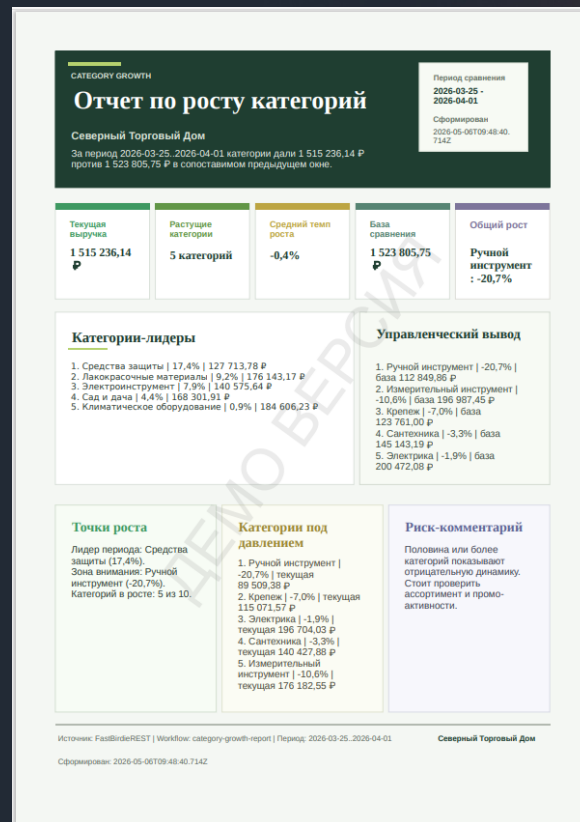
# Отчеты

А отчеты будут или нет?



# Типы отчетов в демо

- Пульс северной торговли
- Отчет по росту категорий
- Недельная динамика
- Отчет по дебиторской задолженности



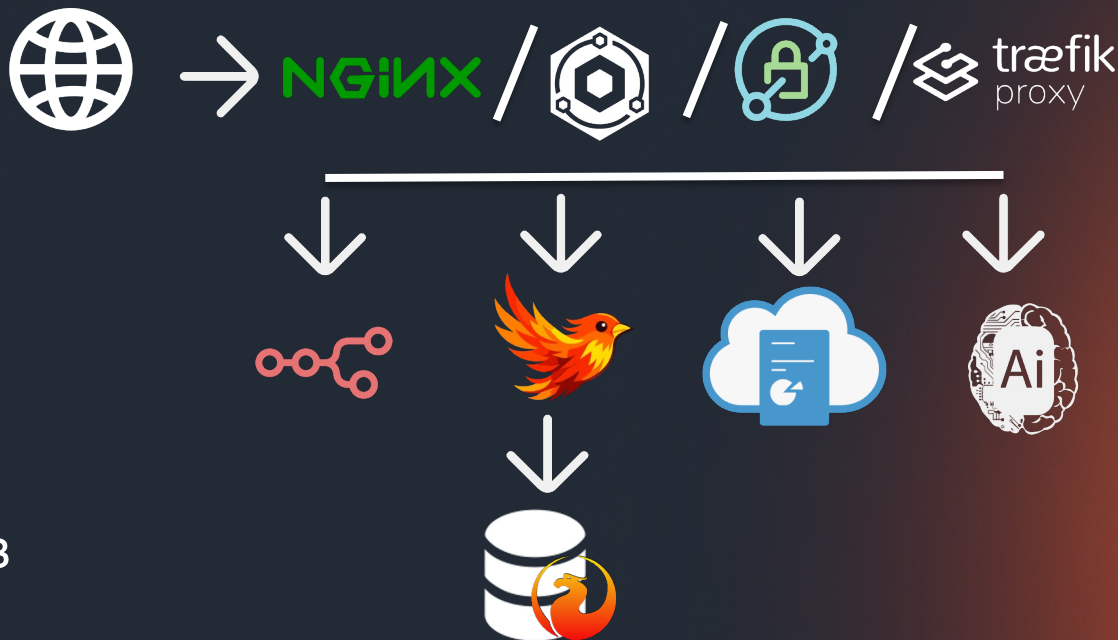
# Разворачивание

Как это все запустить?



# Как это разворачивается

- наружу не торчит Firebird
- наружу не торчат внутренние контейнеры
- весь внешний доступ идет через reverse proxy



# Docker Compose

- идеально для demo-стенда
- удобно для dev/test среды
- можно быстро показать весь pipeline локально
- легко перенести на сервер или VM

# Какие сервисы описываются в docker compose

```
services:  
  firebird:  
    image: firebirdsql/firebird
```

```
  fastbirdierest:  
    depends_on:  
      - firebird
```

```
  n8n:  
    image: n8nio/n8n  
    depends_on:  
      - api
```

```
  mcp-firebird:  
    image: node:20  
    depends_on:  
      - firebird
```

```
  publisher:  
    image: myreports-publisher:latest
```

```
  reverse-proxy:  
    image: caddy:latest
```

# Результаты

Что поменялось и чего можно добиться?



# До / После

Было	Стало
Excel	конвейер
2 часа	5 минут
ручной труд	автоматизация

# Эффект

- скорость
- стабильность
- меньше ошибок



# Безопасность

А это точно безопасно?



# Что обеспечит безопасность

- Firebird закрыт
- API слой (FastBirdieREST)
- API ключи
- read-only доступ
- правильная настройка FastBirdieREST
- правильная настройка реверс прокси

# Эволюция архитектуры

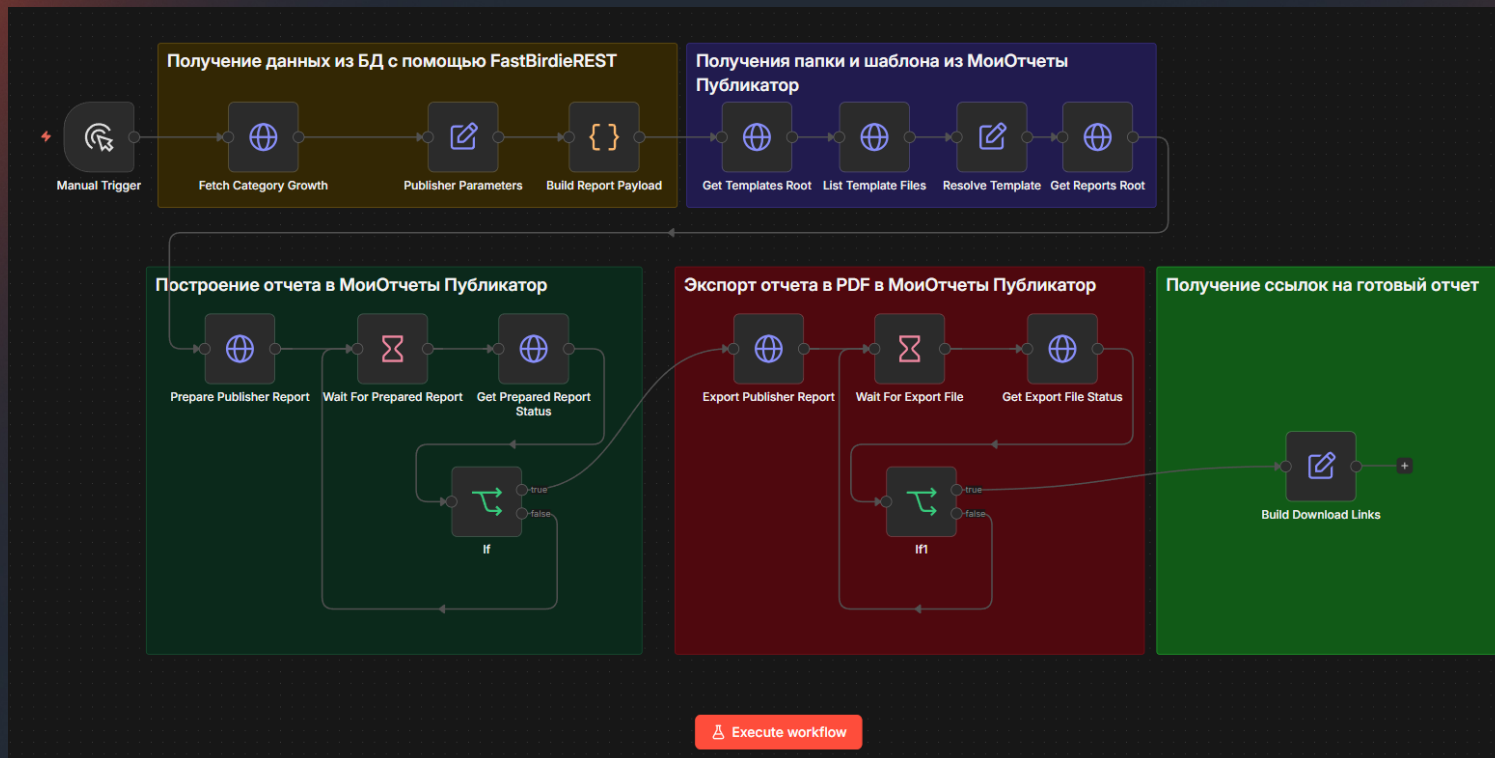
К чему это приведет?



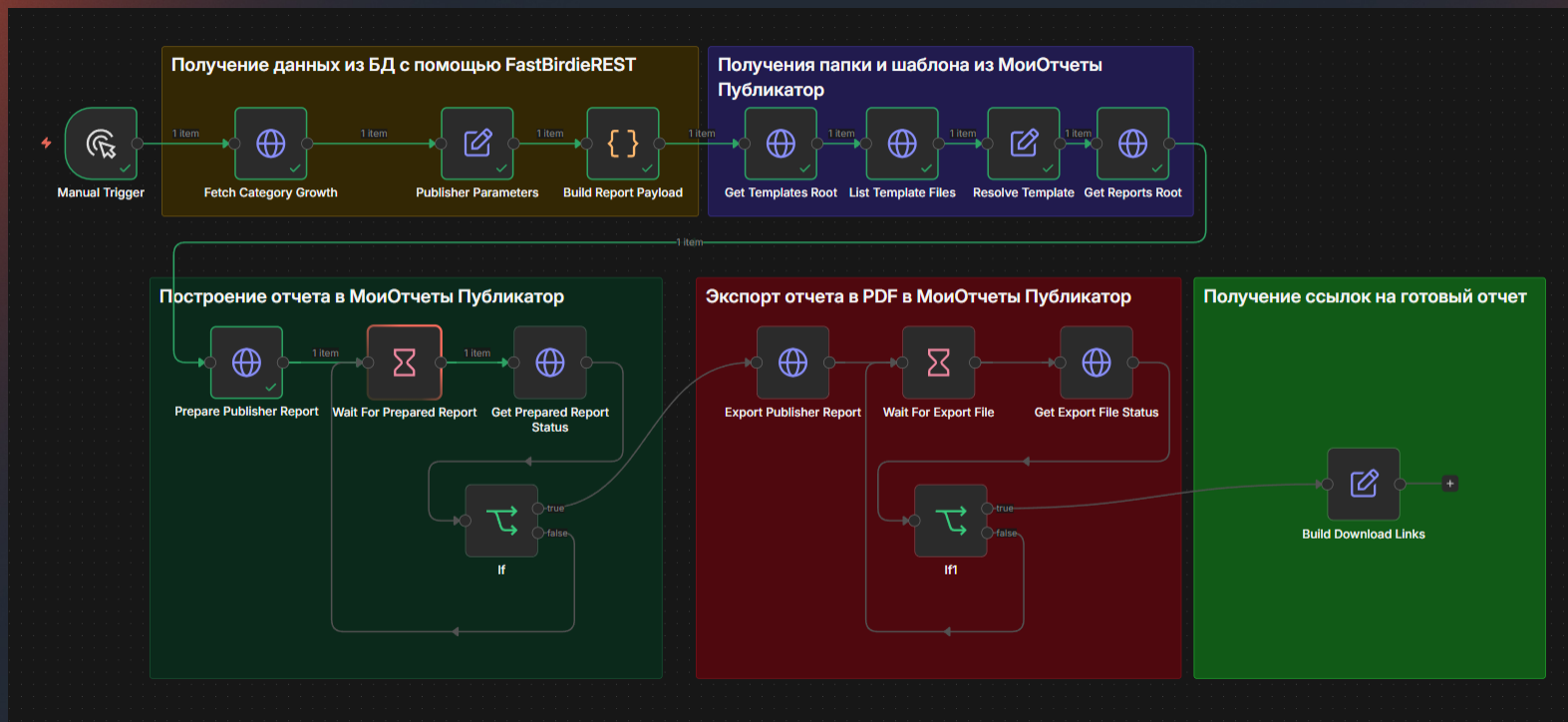
# Эволюция архитектуры

- Firebird → Excel отчет
- ↓
- Firebird → FastBirdieREST → n8n → 32 формата отчетов
- ↓
- Firebird → MCP → n8n + AI → 32 формат отчетов

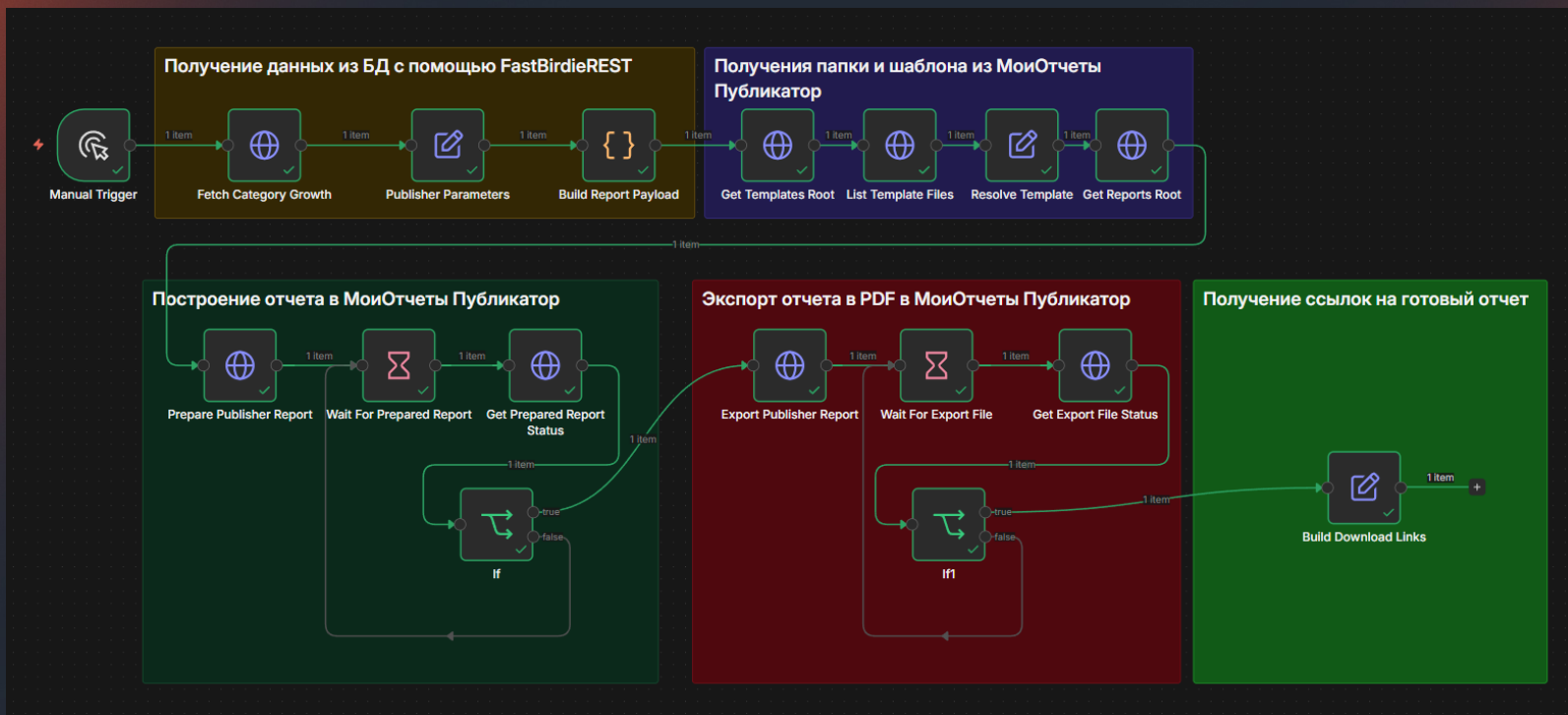
# Показ демки 1 этап



# Показ демки 2 этап



# Показ демки 3 этап



# Показ демки 4 этап

The screenshot displays the 'Build Download Links' application interface. The left pane shows the 'INPUT' JSON schema with a 'Keep Only Set' parameter set to 'True Branch (1 item)'. The middle pane shows the configuration for the 'Keep Only Set' parameter, with the 'Name' field set to 'templateId' and the 'Value' field set to a JavaScript expression: `{{('${Resolve Template').first().json.resolvedTemplateId}}`. The right pane shows the 'OUTPUT' JSON, which is a single item containing a list of download links with various metadata fields.

```
INPUT
```

```
{
  "$t": "ExportVM",
  "format": "Pdf",
  "reportId": "6a05d000f565066f465fbd5d",
  "templateId": "69fa258a93660098544545dc",
  "taskMessageId": "045437e3-0a00-4c48-bde6-ae11462a2e4d",
  "id": "6a05d005f565066f465fbd64",
  "createTime": "2026-05-14T13:37:09.269Z",
  "creatorUserId": "608bc568686ffc46b8d43f62",
  "editedTime": "2026-05-14T13:37:09.465Z",
  "editorUserId": "608bc568686ffc46b8d43f62",
  "name": "category-growth-report-2026-04-01 (5).pdf",
  "parentId": "608bc00fc215f4e1075bf3c",
  "type": "File",
  "size": 183994,
  "subscriptionId": "608bc00fc215f4e1075bf3d",
  "status": "Success",
  "statusReason": "AllRight"
}
```

Parameters

Keep Only Set

Values to Set

String

Name

templateId

Value

```
{{('${Resolve Template').first().json.resolvedTemplateId}}
```

69fa258a93660098544545dc

Name

templateName

Value

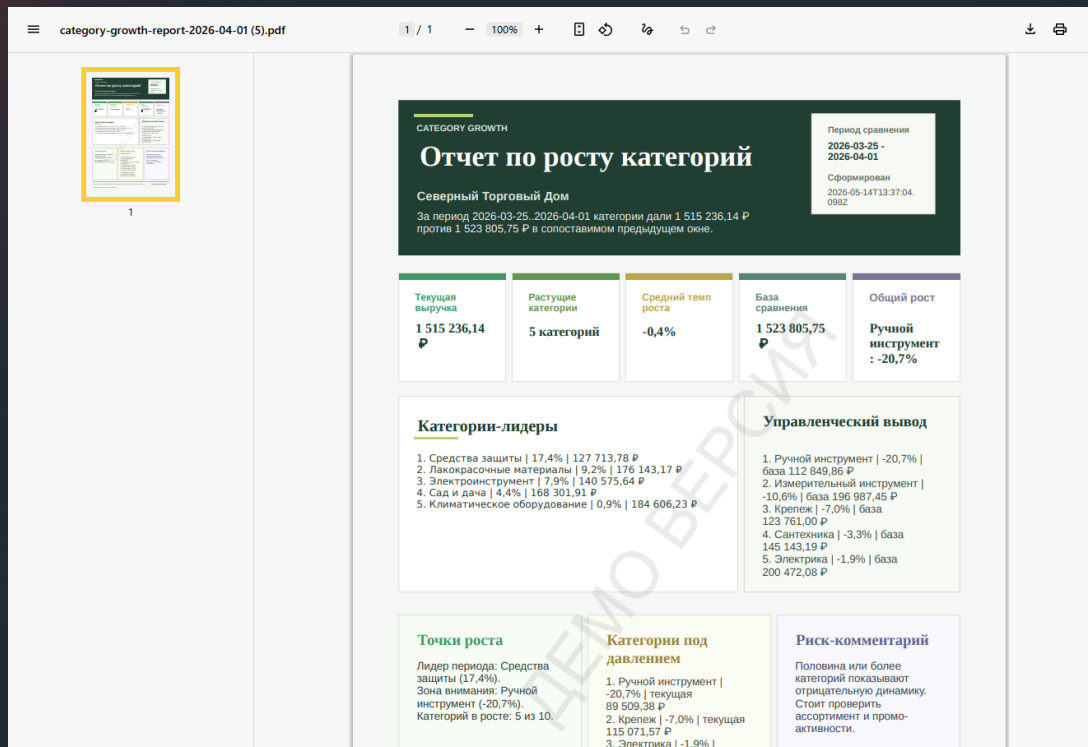
```
{{('${Resolve Template').first().json.resolvedTemplateName}}
```

category-growth-report.frx

OUTPUT

```
1 item
[
  {
    "templateId": "69fa258a93660098544545dc",
    "templateName": "category-growth-report.frx",
    "reportId": "6a05d000f565066f465fbd5d",
    "reportStatus": "Success",
    "exportId": "6a05d005f565066f465fbd64",
    "exportStatus": "Success",
    "exportFileName": "category-growth-report-2026-04-01 (5).pdf",
    "internalDownloadUrl": "http://host.docker.internal:8090/download/e/6a05d005f565066f465fbd64",
    "publicDownloadUrl": "http://localhost:8090/download/e/6a05d005f565066f465fbd64",
    "internalPreparedReportUrl": "http://host.docker.internal:8090/download/r/6a05d000f565066f465fbd5d",
    "publicPreparedReportUrl": "http://localhost:8090/download/r/6a05d000f565066f465fbd5d"
  }
]
```

# Показ демки 5 этап



# Ссылки на демо проект



Спасибо за внимание!