内部股票行情数据分析系统

设计说明书

**2025年07月19日**

**资料版本：** V0.5

**日 期： 2025**年07月

**密 级：** 🞎公开资料 🗹内部资料 🞎保密资料 🞎机密资料

**状 态：** 🗹初稿 🞎讨论稿 🞎发布

**修改记录**

|  |  |  |  |
| --- | --- | --- | --- |
| 日期 | 作者 | 版本 | 修改记录 |
| 2025-07-19 | 王学武 | 0.5 | 创建 |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

**审阅**

|  |  |  |
| --- | --- | --- |
| 姓名 | 时间 | 职位 |
|  |  |  |
|  |  |  |
|  |  |  |

**目录**

[1. 概述 5](#_Toc204369399)

[*1.1.* 目的 5](#_Toc204369400)

[*1.2.* 核心系统架构范围 5](#_Toc204369401)

[*1.3.* 系统组件图 6](#_Toc204369402)

[*1.4.* 数据流管道图 7](#_Toc204369403)

[*1.5.* 关键组件 8](#_Toc204369404)

[1.5.1. 数据采集基础设施 8](#_Toc204369405)

[1.5.2. 后端服务 8](#_Toc204369406)

[1.5.3. 前端应用程序 9](#_Toc204369407)

[*1.6.* 数据库架构概述 9](#_Toc204369408)

[*1.7.* 开发路线图 9](#_Toc204369409)

[*1.8.* 系统入口点 10](#_Toc204369410)

[2. 系统架构 10](#_Toc204369411)

[*2.1.* 目的和范围 10](#_Toc204369412)

[*2.2.* 顶层架构概述 10](#_Toc204369413)

[3. 数据采集系统 13](#_Toc204369414)

[*3.1.* 采集调度 13](#_Toc204369415)

[3.1.1. 目的和范围 13](#_Toc204369416)

[3.1.2. 调度器架构 13](#_Toc204369417)

[3.1.3. 系统启动流程 15](#_Toc204369418)

[3.1.4. 采集任务计划 16](#_Toc204369419)

[3.1.5. 数据采集执行流程 18](#_Toc204369420)

[3.1.6. 数据流图 19](#_Toc204369421)

[*3.2.* Akshare数据采集器 19](#_Toc204369422)

[3.2.1. 系统架构 19](#_Toc204369423)

[3.2.2. 采集器层次结构 19](#_Toc204369424)

[3.2.3. 数据流架构 19](#_Toc204369425)

[3.2.4. 采集器基类 19](#_Toc204369426)

[3.2.5. 实时数据采集器 20](#_Toc204369427)

[3.2.5.1 股票报价采集器 20](#_Toc204369428)

[3.2.5.2 股票报价采集器 20](#_Toc204369429)

[3.2.5.3 股票报价采集器 20](#_Toc204369430)

[3.2.5.4 股票报价采集器 20](#_Toc204369431)

[3.2.6. 数据流架构 20](#_Toc204369432)

[4. 后端API服务 20](#_Toc204369433)

[5. 前端应用程序 20](#_Toc204369434)

[6. 数据库和存储 20](#_Toc204369435)

[7. 配置和部署 20](#_Toc204369436)

[*7.1.* 高可用性需求 21](#_Toc204369437)

# 概述

本文档对中国股市分析系统进行了高层次的介绍，涵盖其架构、核心组件和开发路线图。该系统旨在全面收集、分析和可视化中国股市数据，并具备实时监控功能。

有关特定子系统的详细信息，请参阅[系统架构](https://deepwiki.com/wangj9601/stock_quote_analayze/2-system-architecture)了解整体设计模式、[数据收集系统](https://deepwiki.com/wangj9601/stock_quote_analayze/3-data-collection-system)了解自动数据收集、[后端 API 服务](https://deepwiki.com/wangj9601/stock_quote_analayze/4-backend-api-services)了解 REST 服务实现、[前端应用程序](https://deepwiki.com/wangj9601/stock_quote_analayze/5-frontend-applications)了解用户界面以及[数据库和存储](https://deepwiki.com/wangj9601/stock_quote_analayze/6-database-and-storage)了解数据持久性。

## 目的

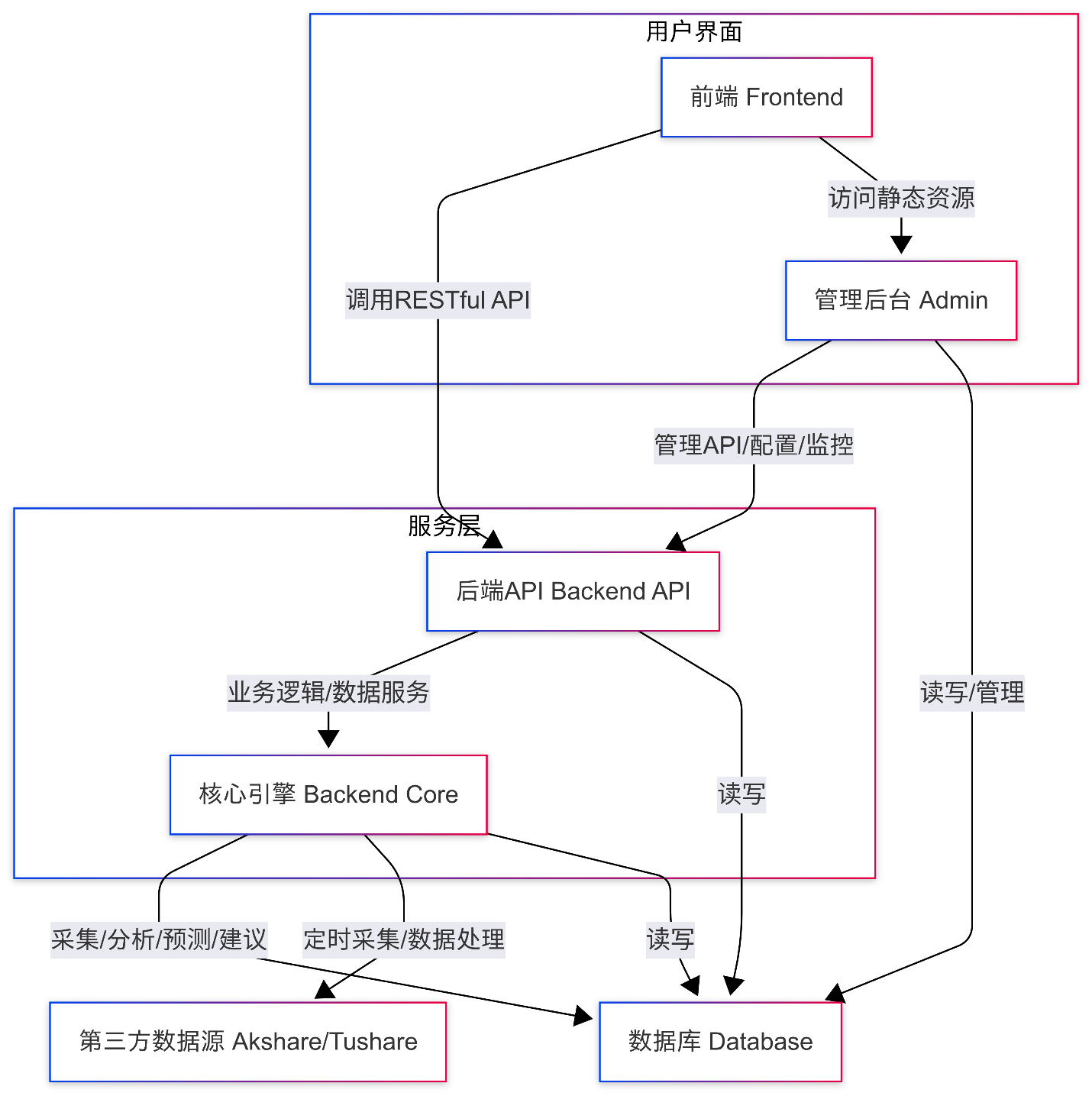
股票分析系统是针对中国股市数据分析的综合平台，提供：

* **自动数据收集：**通过 Akshare 和 Tushare API 定期收集实时报价、历史数据、新闻和财务指标
* **REST API 服务：**基于 FastAPI 的后端提供数据访问、用户管理和业务逻辑
* **Web 界面：**用于股票分析、投资组合跟踪和市场监控的交互式前端
* **数据存储：**具有结构化模式的 PostgreSQL 数据库，可实现高效的数据检索和分析

## 核心系统架构范围

本系统遵循分层架构模式，数据收集、存储、API 服务和表示层之间有明确的分离。。

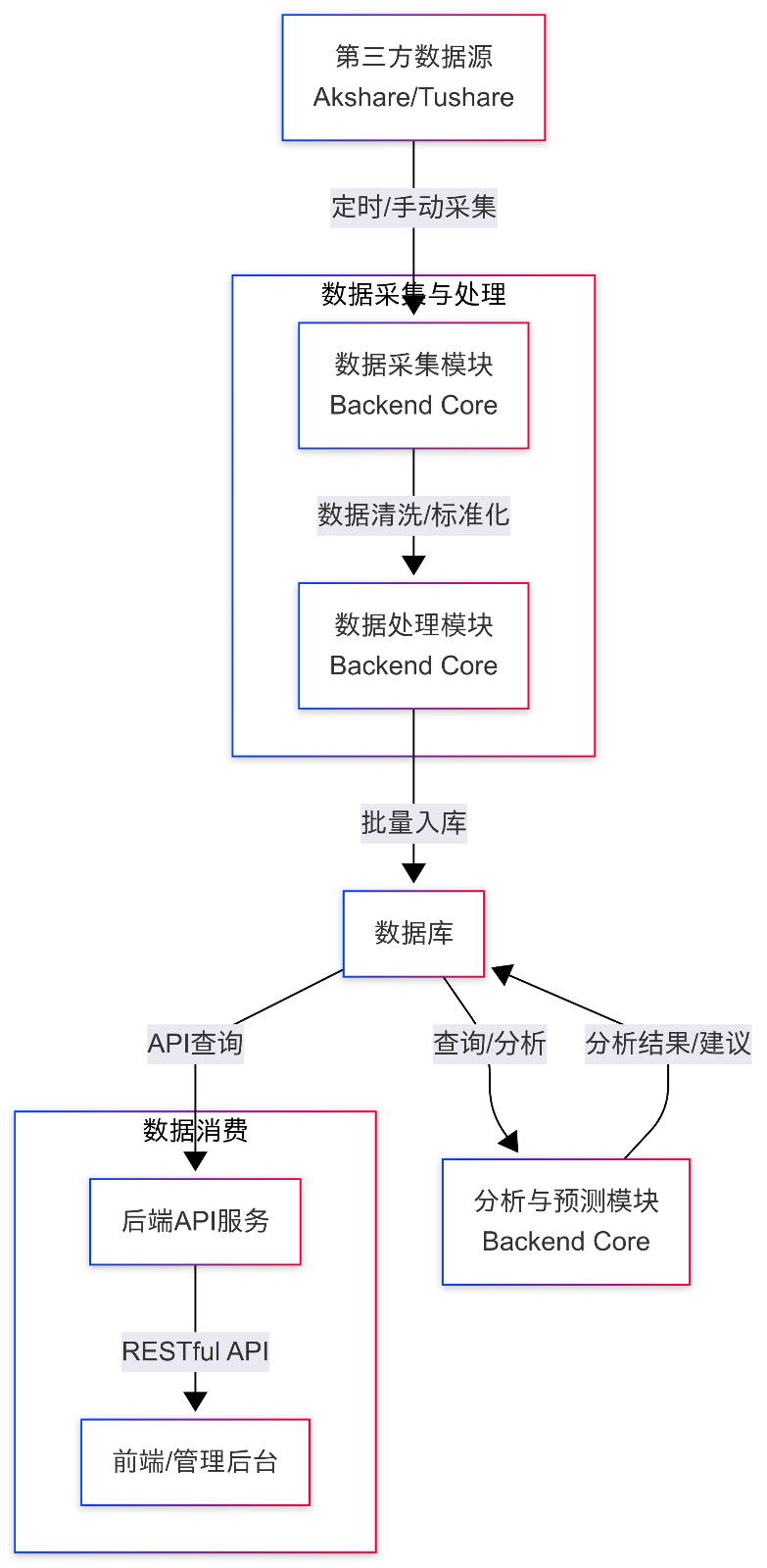
## 系统组件图

****

股票分析系统的主要组件及其关系如下：

* 前端（Frontend）：为普通用户提供行情、分析、选股、资讯、自选股等页面，通过RESTful API与后端交互。
* 管理后台（Admin）：为管理员提供用户、数据、采集、系统配置、日志、模型等管理功能，也通过 API与后端交互。
* 后端API（Backend API）：基于 FastAPI，负责用户认证、数据服务、业务逻辑处理、系统管理等，向前端和管理后台提供统一接口。
* 核心引擎（Backend Core）：负责数据采集（akshare/tushare）、数据清洗、分析计算、模型预测、交易建议等核心功能，定时采集并处理第三方数据。
* 数据库（Database）：存储行情、用户、日志、配置、分析结果等所有结构化数据。
* 第三方数据源（Akshare/Tushare）：为系统提供实时及历史行情、财务、新闻等原始数据。

## 数据流管道图

****

**说明：**

* 数据从Akshare/Tushare 等第三方数据源采集，经采集模块进入数据处理模块进行清洗和标准化。
* 处理后的数据批量写入数据库。
* 分析与预测模块对数据库中的数据进行分析、建模和生成建议，结果也存回数据库。
* 后端API服务从数据库读取数据，向前端和管理后台提供RESTful API接口。
* 前端和管理后台通过API获取所需数据进行展示和管理。

## 关键组件

### 数据采集基础设施

|  |  |  |
| --- | --- | --- |
| **组件** | **用途** | **关键类** |
| **主程序** | 集中数据采集协调 | backend\_core.data\_collectors.main |
| **Akshare 采集器** | 实时报价、行业板块、指数行情、新闻公告等 | AkshareRealtimeQuoteCollector, RealtimeIndexSpotAkCollector |
| **Tushare 采集器** | 历史数据、财务指标等 | HistoricalQuoteCollector, RealtimeQuoteCollector |
| **调度器** | 自动采集调度 | APScheduler.BlockingScheduler |

### 后端服务

|  |  |  |
| --- | --- | --- |
| **服务层** | **端点** | **主要功能** |
| **股票 API** | /api/stock/\* | 实时行情、K线数据、财务指标等 |
| **市场 API** | /api/market/\* | 市场指数、行业表现等 |
| **授权认证 API** | /api/autht/\* | 用户认证、JWT token管理等 |
| **管理 API** | /api/admin/\* | 系统管理、用户管理等 |

### 前端应用程序

|  |  |  |
| --- | --- | --- |
| **界面** | **文件** | **主要特性** |
| **主页** | index.html | 市场概览、指数、关注列表预览、新闻等 |
| **股票分析** | stock.html | 交互式图表、财务数据、新闻、公告等 |
| **关注列表管理** | watchlist.html | 投资组合跟踪、股票分组等 |
| **市场行情探索** | markets.html | 行业分析、股票排名等 |

## 数据库架构概述

系统使用 PostgreSQL 数据库名称stock\_analysis，并包含以下核心表：

* **股票数据：**stock\_realtime\_quote、historical\_quotes、stock\_basic\_info…
* **用户管理：**users、watchlist…
* **市场信息：**industry\_board\_realtime\_quotes，stock\_news…
* **系统操作：**operation\_logs…

## 开发路线图

**近期开发**

* 个股数据采集与分析
* 交易详情，包括时间、价格、交易量和金额数据
* 开发环境搭建、应用和数据库服务器部署

**中期目标**

* 高级分析模型与算法
* 股票选择和筛选能力

**远期愿景**

* 交易策略管理系统
* 策略回测框架

## 系统入口点

主系统可以通过几个入口点启动：

* **数据采集主程序：**python -m backend\_core.data\_collectors.main
* **API：**端口 5000 上的 FastAPI 应用程序
* **数据库访问：**使用管理员凭据访问 PostgreSQL 服务器 5446 端口
* **前端：**8000端口访问股票数据分析前端
* **管理端：**8001端口访问后台管理端

# 系统架构

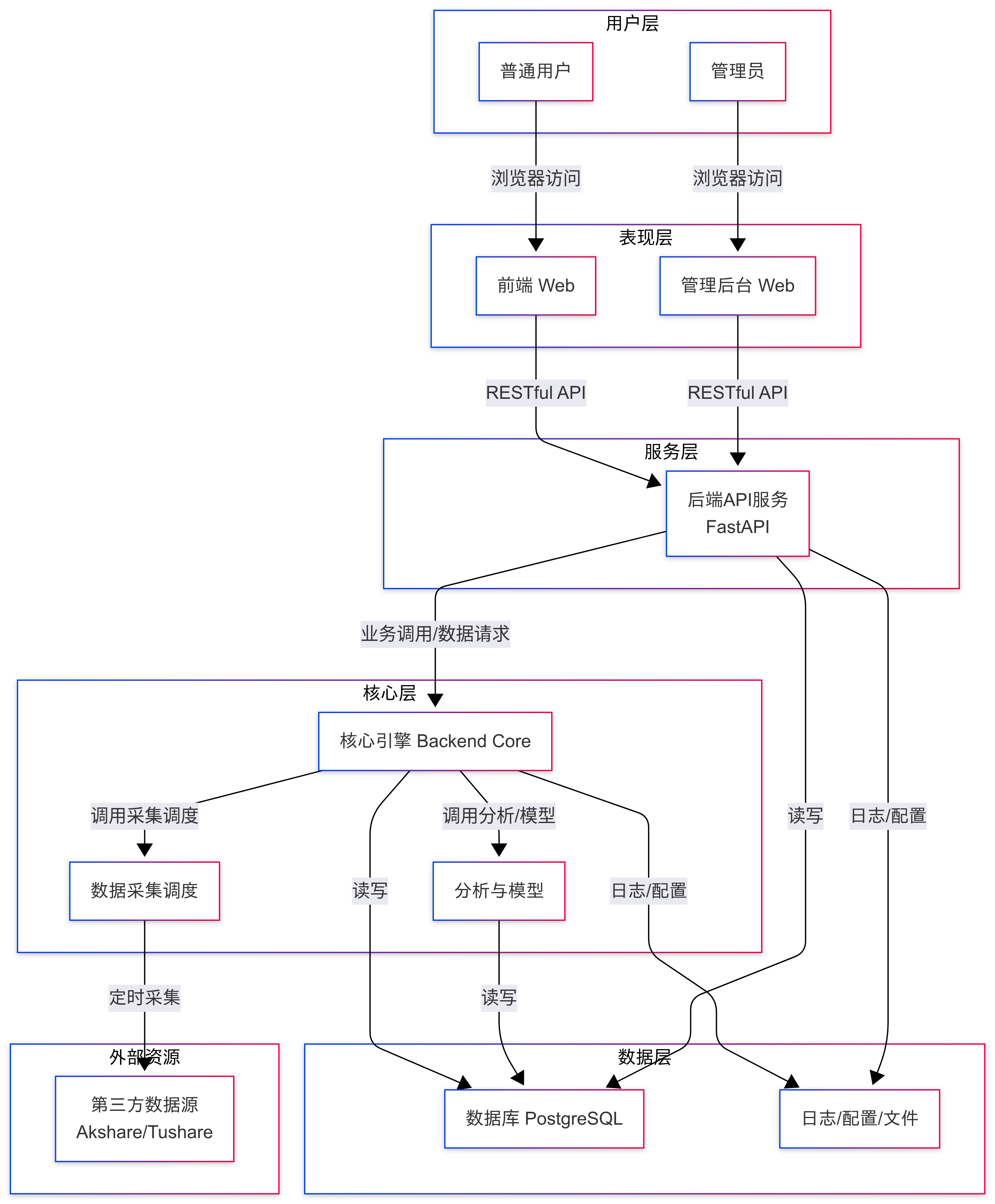
## 目的和范围

本章节全面概述股票分析系统的架构，涵盖整体系统设计、数据流模式以及整个平台的组件关系。文档描述主要子系统的交互方式，以及实现自动数据收集、实时市场分析和用户投资组合管理的关键架构决策。

有关特定子系统的详细信息，请参阅[**数据采集系统**](https://deepwiki.com/wangj9601/stock_quote_analayze/3-data-collection-system)**、**[**后端 API 服务**](https://deepwiki.com/wangj9601/stock_quote_analayze/4-backend-api-services)**、**[**前端应用程序**](https://deepwiki.com/wangj9601/stock_quote_analayze/5-frontend-applications)和[**数据库与存储**](https://deepwiki.com/wangj9601/stock_quote_analayze/6-database-and-storage)。

## 顶层架构概述

本系统采用分层架构模式，数据收集、API 服务和用户界面之间清晰划分。该架构支持实时数据处理、定时批量操作和多用户 Web 访问。



1. **用户层**

* **普通用户**：通过浏览器访问前端页面，获取行情、分析、资讯、自选股等服务。
* **管理员**：通过浏览器访问管理后台，进行用户、数据、采集、系统配置、日志、模型等管理操作。

1. **表现层**

* **前端 Web：**为普通用户提供股票行情、分析、图表、资讯等可视化界面，所有数据均通过后端API获取。
* **管理后台 Web：**为管理员提供系统管理界面，包括用户管理、数据源配置、采集任务、日志监控、模型管理等。

1. **服务层**

* **后端API服务（FastAPI）：**统一对外提供RESTful API，负责用户认证、数据服务、业务逻辑处理、系统管理等。前端和管理后台均通过API与后端交互。

1. **核心层**

* **核心引擎（Backend Core）：**负责数据采集、清洗、标准化、分析、预测、交易建议等核心业务逻辑。
* **分析与模型：**实现技术分析、基本面分析、机器学习/深度学习模型预测等功能，为用户和管理端提供智能分析结果。
* **数据采集调度：**定时或手动从第三方数据源采集行情、财务、新闻等数据，保证数据的实时性和完整性。

1. **数据层**

* **数据库（PostgreSQL）：**存储所有结构化数据，包括行情、用户、日志、配置、分析结果等，支持高效查询和批量写入。
* **日志/配置/文件：**存储系统运行日志、采集日志、配置文件、导出文件等，便于系统监控和维护。

1. **外部资源**

* **第三方数据源（Akshare/Tushare）：**为系统提供A股、ETF、指数、财务、新闻等原始数据，是数据采集的基础。

**主要交互说明**

* 用户和管理员均通过浏览器访问各自界面，所有数据和操作均通过后端API完成。
* 后端API服务作为唯一的数据和业务逻辑入口，既服务前端，也服务管理后台。
* 核心引擎负责所有数据的采集、处理、分析和预测，并与数据库进行读写交互。
* 数据采集调度模块定时从第三方数据源拉取最新数据，经过清洗和标准化后入库。
* 日志和配置文件为系统运维和监控提供支持，所有关键操作均有日志记录。

# 数据采集系统

## 采集调度

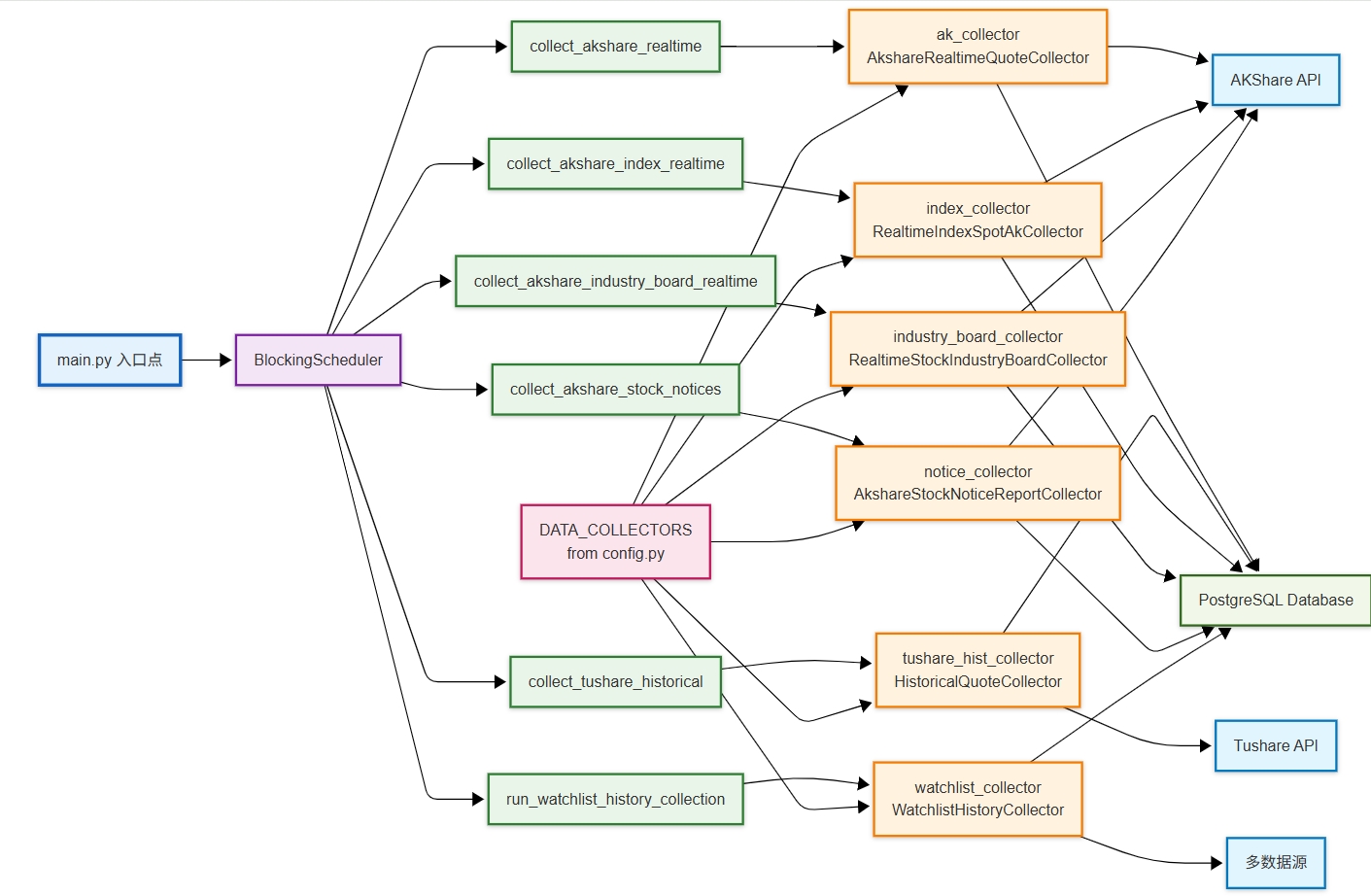
### 目的和范围

数据采集调度模块为股票分析平台中的所有自动化数据收集任务提供集中调度和协调。该模块使用 APSchedulerBlockingScheduler来管理来自 Akshare 和 Tushare 等多个外部 API 的数据收集器的定时执行。

本章节涵盖了主要的调度器实现、任务配置以及协调逻辑。关于具体的数据收集器，请参阅[**Akshare 数据收集器**](https://deepwiki.com/wangj9601/stock_quote_analayze/3.2-akshare-data-collectors)、[**Tushare 数据收集器**](https://deepwiki.com/wangj9601/stock_quote_analayze/3.3-tushare-data-collectors)和[**Watchlist 数据收集器**](https://deepwiki.com/wangj9601/stock_quote_analayze/3.4-watchlist-data-collection)等。

### 调度器架构

编排系统围绕一个中央实例构建BlockingScheduler，该实例协调多个专用数据收集器。每个收集器只需初始化一次，即可在计划执行期间重复使用。



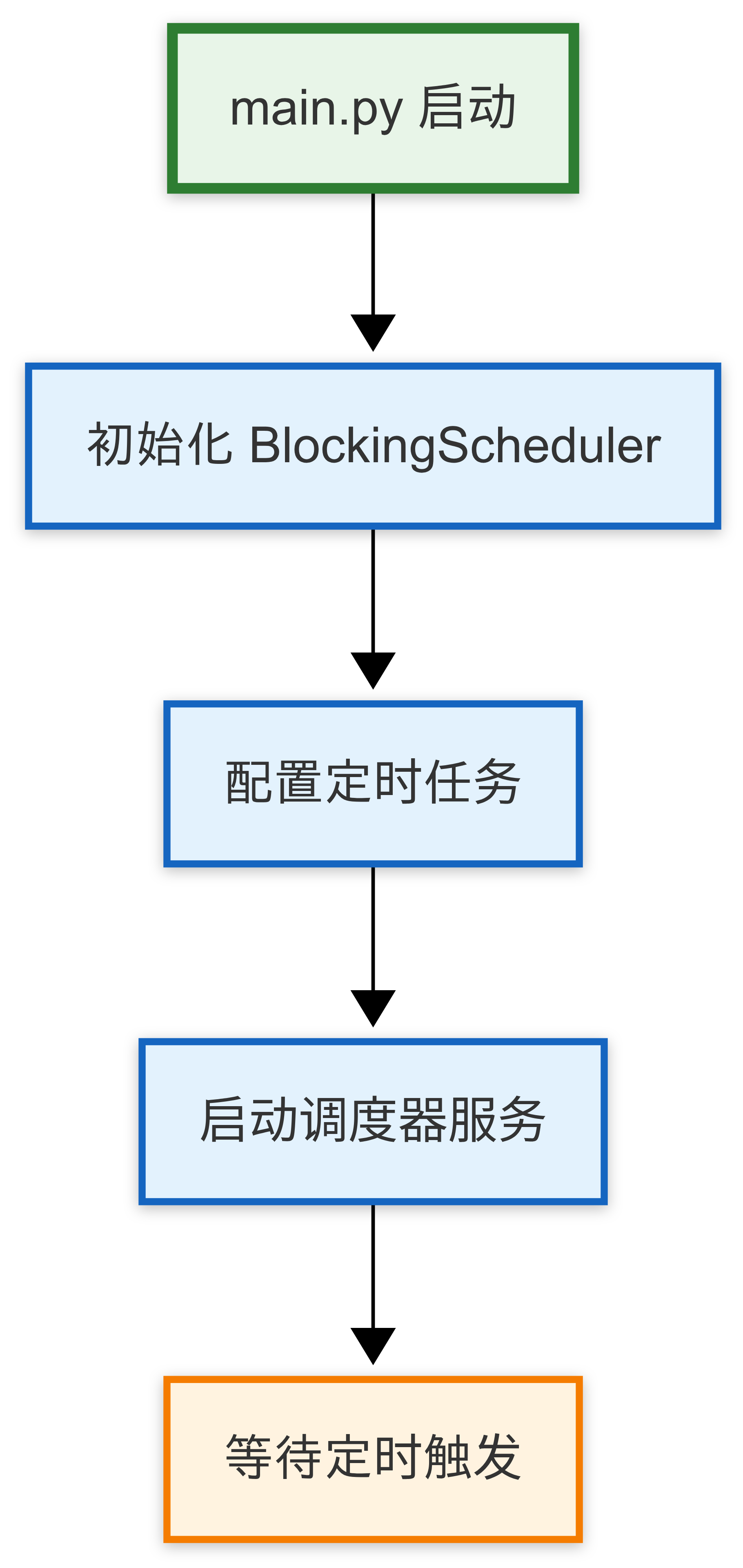
**关键组件说明**

* **编排层(Orchestration Layer)**
* main.py入口点
* BlockingScheduler调度器
* **包装函数层(Wrapper Functions Layer)**

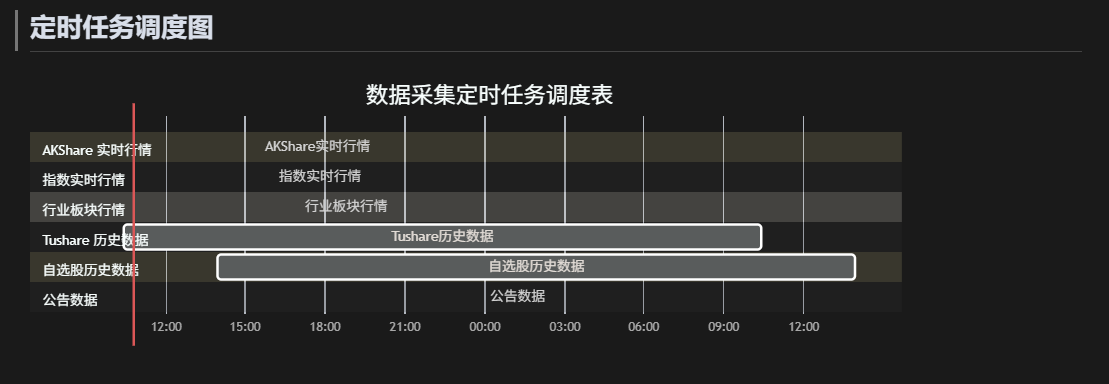
6个主要的包装函数，对应不同的数据采集任务

* collect\_akshare\_realtime(): AKShare实时行情采集
* collect\_tushare\_historical(): Tushare历史数据采集
* collect\_akshare\_index\_realtime(): 指数实时行情采集
* collect\_akshare\_industry\_board\_realtime(): 行业板块行情采集
* collect\_akshare\_stock\_notices(): 公告数据采集
* run\_watchlist\_history\_collection(): 自选股历史数据采集
* **采集器实例层(Collector Instances Layer)**
* Tushare采集器：历史数据、实时数据
* ak\_collector: AKShare实时行情采集器
* tushare\_hist\_collector: Tushare历史数据采集器
* index\_collector: 指数实时行情采集器
* industry\_board\_collector: 行业板块行情采集器
* notice\_collector: 公告数据采集器
* watchlist\_collector: 自选股历史数据采集器
* **配置层(Configuration Layer)**
* DATA\_COLLECTORS统一配置管理

### 系统启动流程

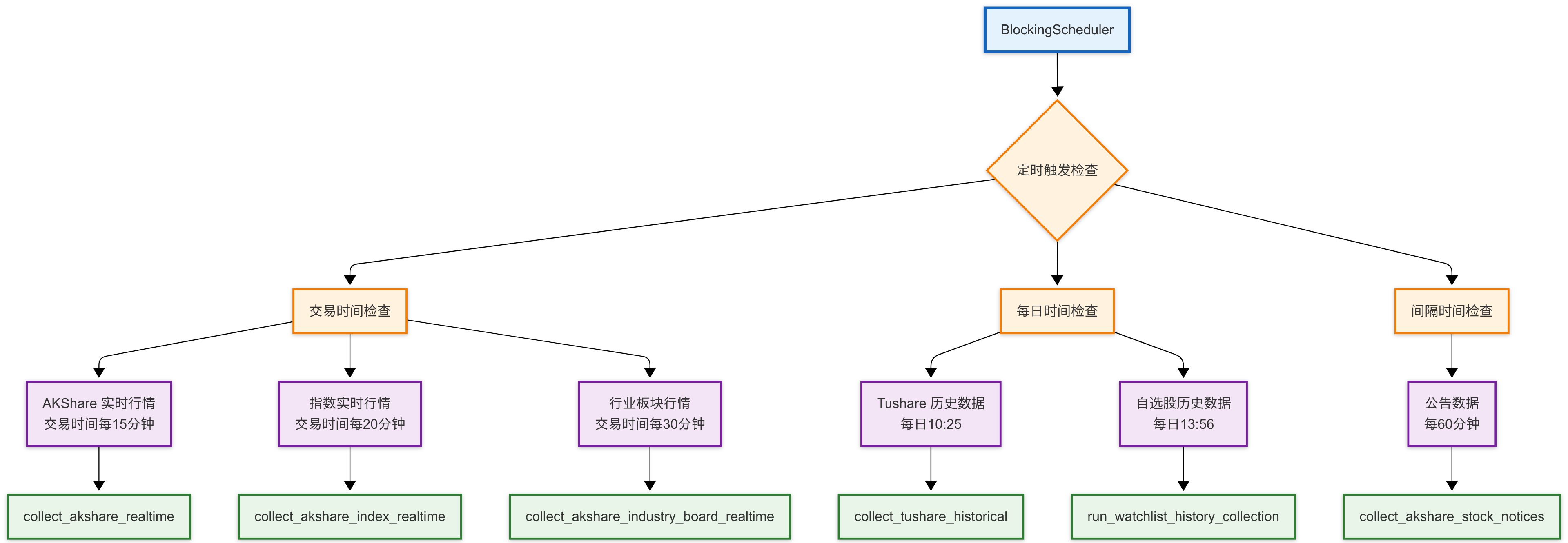


### 采集任务计划

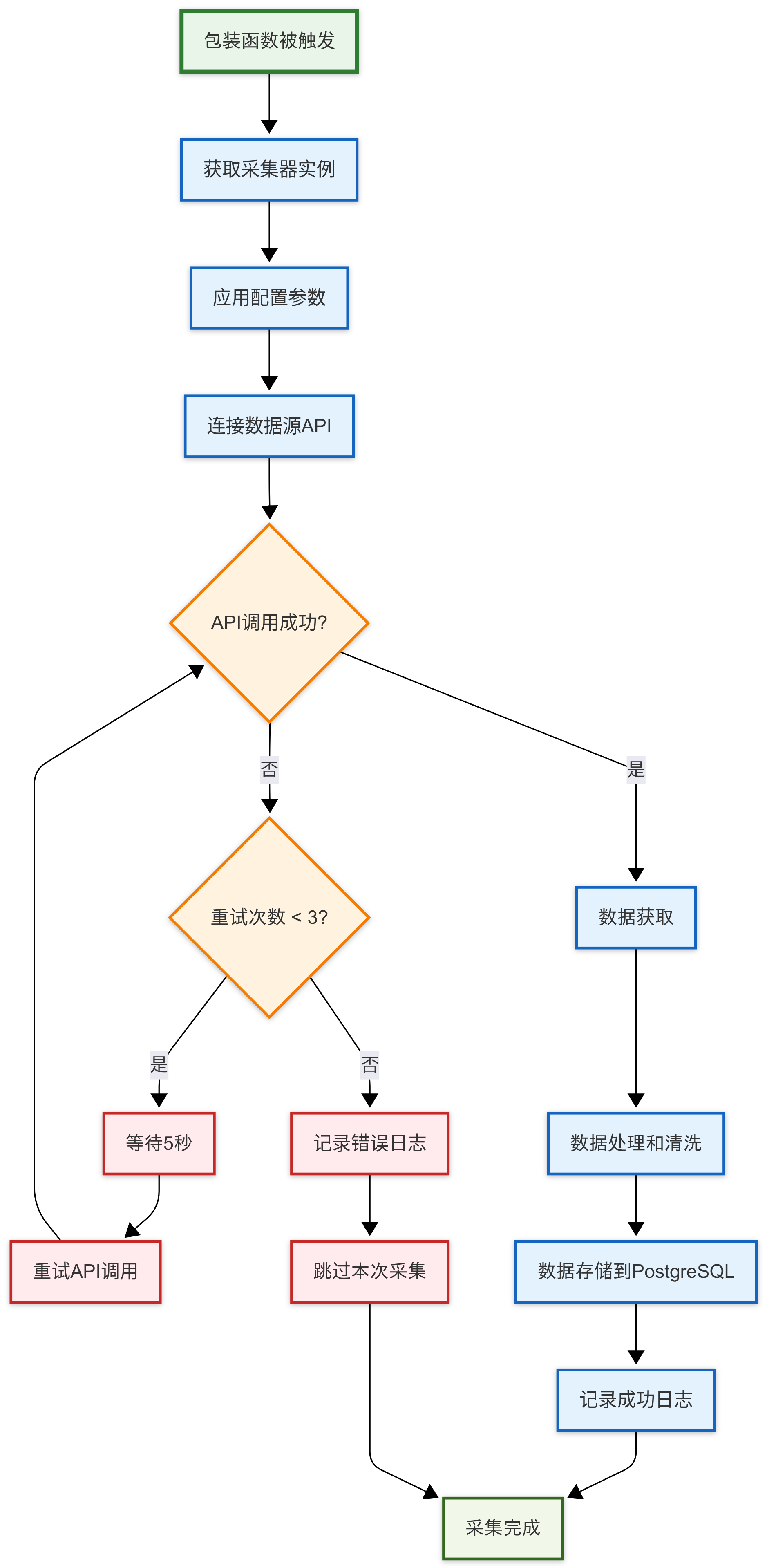


|  |  |  |  |
| --- | --- | --- | --- |
| **任务名称** | **调度策略** | **包装函数** | **采集器实例** |
| **AKShare 实时行情** | 交易时间每15分钟 | collect\_akshare\_realtime() | ak\_ collector |
| **Tushare 历史数据** | 每日10:25 | collect\_tushare\_historical() | tushare\_hist\_collector |
| **指数实时行情** | 交易时间每20分钟 | collect\_akshare\_index\_realtime() | index\_collector |
| **行业板块行情** | 交易时间每30分钟 | collect\_akshare\_industry\_board\_realtime() | industry\_board\_collector |
| **公告数据** | 每60分钟 | collect\_akshare\_stock\_notices() | notice\_collector |
| **自选股历史数据** | 每日13:56 | run\_watchlist\_history\_collection() | watchlist\_collector |

**定时任务调度逻辑**



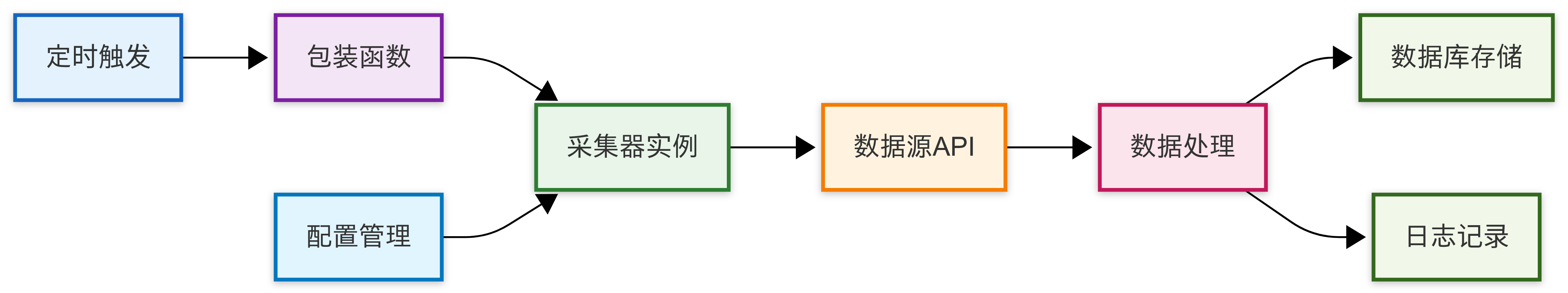
### 数据采集执行流程



**错误处理和重试机制**

* **重试配置**
* 最大重试次数: 3次
* 重试延迟: 5秒
* 请求超时: 30秒
* 最大连接错误: 10次

### 数据流图



## Akshare数据采集器

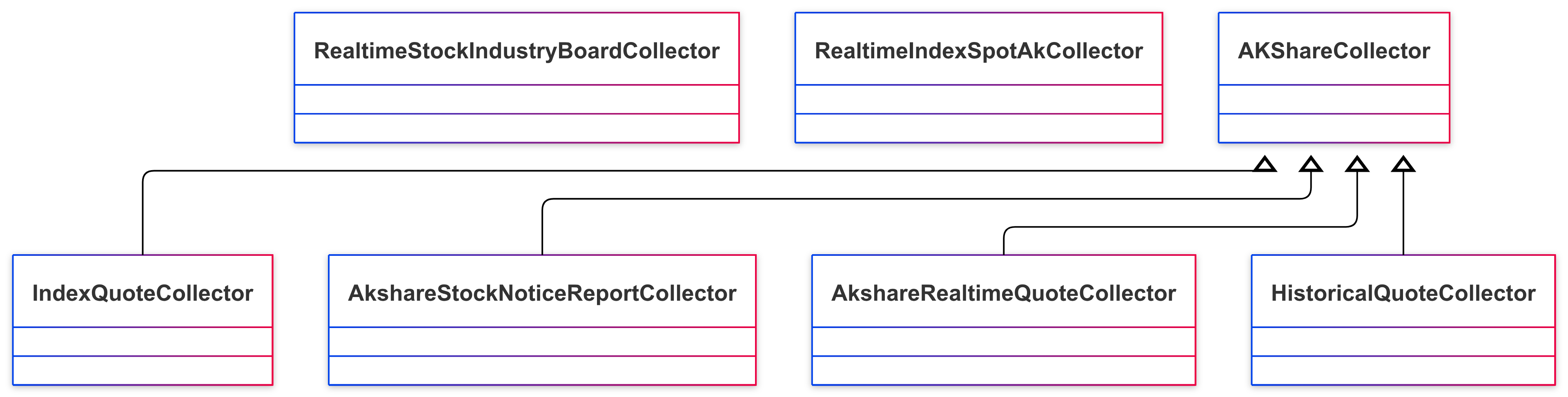
本章节介绍 Akshare 数据收集子系统，该系统负责通过 Akshare API 自动收集中国股市数据。该系统收集实时行情、行业板块表现、股票公告、指数数据和历史交易数据，并将其存储在 PostgreSQL 数据库表中。

整体数据采集编排和调度请参见[**采集调度**](https://deepwiki.com/wangj9601/stock_quote_analayze/3.1-collection-orchestration)。 Tushare API 的历史数据采集请参见[**Tushare 数据采集器**](https://deepwiki.com/wangj9601/stock_quote_analayze/3.3-tushare-data-collectors)。

### 系统架构

Akshare 数据收集器采用分层架构，并具有通用基类，为 API 调用、错误处理和数据库操作提供共享功能。每个收集器都专门处理 Akshare API 中的特定数据类型。

### 采集器层次结构



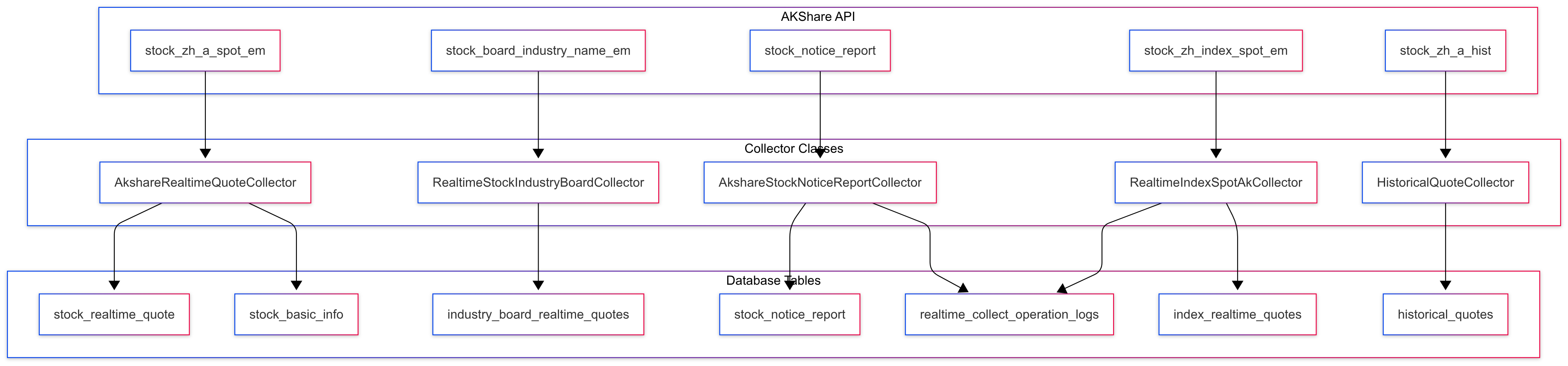
说明

* 核心基类：AKShareCollector，其子类负责不同类型的数据采集。
* 部分采集器（如行业板块、指数实时采集器）未继承基类，但实现方式类似，属于同一采集体系。
* 采集器之间通过配置、数据库操作等实现复用和扩展。

代码路径：

* [backend\_core/data\_collectors/akshare/base.py](https://github.com/wangj9601/stock_quote_analayze/blob/f38b222c/backend_core/data_collectors/akshare/base.py#L13-L17)
* [backend\_core/data\_collectors/akshare/realtime.py](https://github.com/wangj9601/stock_quote_analayze/blob/f38b222c/backend_core/data_collectors/akshare/realtime.py#L18-L19)
* [backend\_core/data\_collectors/akshare/historical.py](https://github.com/wangj9601/stock_quote_analayze/blob/f38b222c/backend_core/data_collectors/akshare/historical.py#L21-L22)
* [backend\_core/data\_collectors/akshare/realtime\_stock\_notice\_report\_ak.py](https://github.com/wangj9601/stock_quote_analayze/blob/f38b222c/backend_core/data_collectors/akshare/realtime_stock_notice_report_ak.py#L31-L32)
* [backend\_core/data\_collectors/akshare/realtime\_stock\_industry\_board\_ak.py](https://github.com/wangj9601/stock_quote_analayze/blob/f38b222c/backend_core/data_collectors/akshare/realtime_stock_industry_board_ak.py#L10-L14)
* [后端核心/数据收集器/akshare/realtime\_index\_spot\_ak.py](https://github.com/wangj9601/stock_quote_analayze/blob/f38b222c/backend_core/data_collectors/akshare/realtime_index_spot_ak.py#L10-L15)

### 数据流架构



**Akshare 数据采集器数据流图架构说明**

* AKShare API 层

**作用**：作为数据采集的源头，负责从互联网获取原始金融数据。

**主要接口**：

* stock\_zh\_a\_spot\_em：获取A股实时行情数据。
* stock\_board\_industry\_name\_em：获取行业板块实时数据。
* stock\_notice\_report：获取股票公告数据。
* stock\_zh\_index\_spot\_em：获取指数实时行情数据。
* stock\_zh\_a\_hist：获取A股历史行情数据。
* Collector Classes层

**作用：**采集器类负责调度、调用API、处理数据，并将数据写入数据库。

**主要采集器：**

* AkshareRealtimeQuoteCollector：采集A股实时行情，写入stock\_realtime\_quote和stock\_basic\_info表。
* RealtimeStockIndustryBoardCollector：采集行业板块实时数据，写入industry\_board\_realtime\_quotes表。
* AkshareStockNoticeReportCollector：采集公告数据，写入stock\_notice\_report和realtime\_collect\_operation\_logs表。
* RealtimeIndexSpotAkCollector：采集指数实时行情，写入index\_realtime\_quotes和realtime\_collect\_operation\_logs表。
* HistoricalQuoteCollector：采集A股历史行情，写入historical\_quotes表。
* Database Tables 层

作用**：存储采集到的结构化数据，供后续分析、查询和展示使用。**

主要表：

* stock\_realtime\_quote：A股实时行情数据。
* stock\_basic\_info：股票基础信息。
* industry\_board\_realtime\_quotes：行业板块实时行情。
* stock\_notice\_report：股票公告数据。
* realtime\_collect\_operation\_logs：采集操作日志，记录采集过程中的操作和异常。
* index\_realtime\_quotes：指数实时行情数据。
* historical\_quotes：A股历史行情数据。
* 数据流动说明
* 每个API接口的数据流向对应的采集器类。
* 采集器类对数据进行处理后，写入一到多个数据库表。
* 采集器在采集和写入过程中，会将操作日志写入realtime\_collect\_operation\_logs表，便于追踪和监控。

### 采集器基类

### 实时数据采集器

#### 股票报价采集器

#### 股票报价采集器

#### 股票报价采集器

#### 股票报价采集器

### 数据流架构

# 后端API服务

# 前端应用程序

# 数据库和存储

行情数据延迟：实时行情数据在数据源更新后，前端展示延迟不超过 [5] 秒。

页面加载速度：各主要页面（自选、个股详情）加载时间不超过 [5] 秒。

图表响应：K线图缩放、平移、切换周期、叠加指标等操作响应流畅，延迟不超过 [500] 毫秒。

API响应时间：大多数API请求应在 [5] 秒内响应。

数据采集效率：能够在规定时间内完成日终、周终等数据采集任务。

# 配置和部署

* + 用户认证与授权：确保用户账号安全，API调用权限控制。
  + 数据传输安全：前后端、后端各服务之间使用加密通信（HTTPS, TLS）。
  + 敏感数据保护：用户密码、API密钥等敏感信息加密存储，防止数据泄露。
  + 防御常见攻击：防御SQL注入、XSS、CSRF、DDoS等安全威胁。
  + 预测模型安全：保护预测模型算法和数据不被非法获取。

## 高可用性需求

系统可用性：系统核心功能（行情、个股详情）年可用率目标达到 [99% 或更高]。

数据一致性：确保存储和展示的数据准确和一致。

容错性：系统应能处理部分组件故障，避免整个系统崩溃。

数据备份与恢复：定期备份数据，具备快速恢复能力。