AvaTrade Dev Home Assignment

Juan David Berovides

05.05.2023

Table of contents

Business case scope

The focus is to enrich the news section of AvaTrade marketing web site and landing pages.

The objective is to create centralized service that:

1. Consume trading news every 1 hour from any provider available i.e. https://polygon.io/
2. For each news items
   a. Enrich the data with any information you can think of i.e., ticker chart
   b. Save the data in any storage solution.
3. Create web API that any client can reach and implement the following:
   a. a. [Authorize] Get all news.
   b. b. [Authorize] Get all news from today – {n} days.
   c. c. [Authorize] Get all news per instrument name include news limit (default limit = 10).
   d. d. [Authorize] Get all news that contains {text}.
   e. e. [Authorize] Allow customer to subscribe.
   f. f. [Public] Get latest new (top latest 5 different instruments) for conversion tool.

Architectural proposals

1- Microservices-based architecture

In this architecture, each functionality is broken down into independent microservices, allowing for greater scalability and flexibility. Each microservice handles a specific task, and communication between them is done through a RESTful API. For the centralized service solution described, the following microservices could be implemented:

- News collection service: This service is responsible for collecting trading news every hour from the news provider, such as Polygon.io.
- Data enrichment service: This service enriches the news data with additional information, such as stock quote charts for the ticker symbol in question.
- Data storage service: This service is responsible for storing the data in a storage solution, such as a SQL or NoSQL database.
- Web API service: This service provides a RESTful API for clients to access the data stored in the data storage service. The web API service also handles client authentication and authorization.

2- Event-driven architecture

In this architecture, the different functionalities are broken down into independent events, with each event handling a specific task. The events communicate through an event bus, allowing for extremely high scalability and flexibility. For the centralized service solution described, the following events could be implemented:

- News collection event: This event is responsible for collecting trading news every hour from the news provider, such as Polygon.io.
- Data enrichment event: This event enriches the news data with additional information, such as stock quote charts for the ticker symbol in question.
- Data storage event: This event is responsible for storing the data in a storage solution, such as a SQL or NoSQL database.
- Web API event: This event provides a RESTful API for clients to access the data stored in the data storage event. The web API event also handles client authentication and authorization.
- Subscription event: This event handles client subscriptions.

Pros and Cons for each architecture

Architecture	PROS	CONS
Microservices	Scalability: Microservices allow for scaling individual components independently, making it easier to handle high traffic and load. Flexibility: Each microservice can be developed and deployed independently, enabling faster iterations and changes. Resilience: If one microservice fails, it does not necessarily mean that the entire system fails. Maintainability: Since each microservice is independent, it is easier to maintain and update them.	Complexity: The microservices-based architecture can become complex to manage, as there are multiple independent components that need to be coordinated. Increased overhead: Since each microservice requires a separate runtime environment, there is an overhead in terms of resources and costs. Communication overhead: The communication between microservices can introduce additional latency and complexity. Testing: Testing a microservices-based architecture can be challenging as there are multiple independent components.
Event-driven	Scalability: Event-driven architectures allow for horizontal scaling, enabling handling of large amounts of data and events. Flexibility: Since each event can trigger a different action, the architecture is highly flexible and can be adapted to different use cases. Resilience: If one component fails, it does not necessarily mean that the entire system fails. Decoupling: The architecture components are loosely coupled, making it easier to change or update individual components.	Complexity: The event-driven architecture can become complex to manage, as there are multiple independent components that need to be coordinated. Debugging: Debugging event-driven systems can be challenging due to the distributed nature of the system. Testing: Testing event-driven architectures can be challenging as there are multiple independent components. Event ordering: Handling events in the right order can be a challenge, as different events can arrive at different times.

Recommended proposal

In summary, both architectures have their own set of advantages and disadvantages, and the choice depends on the specific requirements and constraints of the system. Microservices-based architecture is best suited for systems that require independent scaling of components, while event-driven architecture is best suited for systems that require handling of large amounts of data and events.

For a small-scale project, I would recommend a microservices-based architecture. The main reason is that microservices architecture is more suitable for smaller, less complex projects because it is simpler to design, implement, and maintain. Microservices architecture allows for greater flexibility and scalability, which are key factors in any project, regardless of its size.

Event-driven architecture is more complex, and it requires a higher level of expertise to design and implement. Moreover, it is more suitable for larger, complex systems that require handling of large volumes of data and events.

In summary, while both architectures have their own set of advantages and disadvantages, the microservices-based architecture is more suitable for smaller, less complex projects, as it is simpler to design, implement, and maintain.

Data storage

Additionally, regarding the data storage, using a NoSQL database can provide flexibility in storing the data provided by trading APIs like Polygon.io. NoSQL databases are designed to handle unstructured or semi-structured data, which is often the case in trading APIs that provide data in JSON or other non-tabular formats.

A NoSQL database also provides the ability to scale horizontally, which is important for microservices-based architectures. This means that you can add more nodes or servers to the database cluster to handle increased traffic or data

volumes as needed, without needing to modify the database schema or application code.

Moreover, NoSQL databases typically offer better performance for read-heavy workloads, which may be the case in a trading news application that requires frequent reads of the data.

Initial backlog proposal

The proposed development team composition is one senior developer that will act a tech lead, plus 3 mid-level developers with an average experience of 5 years. Development methodology is SCRUM with a 2-week sprint cycle. For each main task or user story, here will be included a rough estimation in man-hours.

Backend base implementation tasks

1- Solution scaffolding (30 man-hours)

Description: This task involves setting up the initial infrastructure and architecture of the system, including the microservices and the chosen technologies. The goal is to create a modular and scalable system that can handle the data pipeline and the Web API.

Acceptance criteria:

- The solution includes separate services for data storage, news collection, and data enrichment, each with its own API endpoints and data models.
- The solution uses MongoDB as the primary data storage solution, with appropriate data models and schemas defined.
- The solution uses Hangfire as the job scheduling framework, with appropriate jobs and tasks defined.
- The solution uses .NET Core 7 as the Web API framework, with appropriate endpoints and routes defined.
- The solution includes appropriate logging, monitoring, and security measures, Serilog for logging, Prometheus for monitoring, and JWT for authentication.
- The solution is well-documented and follows best practices, such as SOLID principles, DRY, and KISS.

2- Data Storage Service (120 man-hours)

Description: This task involves the implementation of the *Data Storage Service* microservice.

Acceptance criteria: The following tasks must be completed:

- Set up a MongoDB database instance for storing the news data and configure the connection string and credentials in the configuration file.
- Define the data models and schemas for the news data, such as NewsItem, Source, and Category.
- Implement the CRUD operations for the news data, such as Create, Read, Update, and Delete, using the MongoDB driver and appropriate error handling.
- Implement a Redis caching mechanism, to improve the performance and scalability of the data storage service.
- Implement appropriate logging and monitoring mechanisms, using Serilog and Prometheus respectively, to track and analyse the performance and usage of the data storage service.

3- News Collection Service (160 man-hours)

Description: This task involves the implementation of the *News Collection Service* microservice.

Acceptance criteria: The following tasks must be completed:

- Define the sources and categories of the news data and configure them in the configuration file.
- Implement the news data retrieval logic for each source and category.
- Implement the data transformation and normalization logic for the news data, including removing duplicates, filtering by keywords and extracting metadata.
- Implement the data validation and error handling logic for the news data, including checking for null or invalid data and logging errors.
- Implement appropriate scheduling and job management mechanism, using Hangfire, to ensure that the news data is collected and processed at regular intervals.

4- Data Enrichment Service (200 man-hours)

Description: This task involves the implementation of the *Data Enrichment Service* microservice.

Acceptance criteria: The following tasks must be completed:

- Define the enrichment tasks and algorithms for the news data, such as sentiment analysis, entity recognition, or keyword extraction, and configure them in the configuration file.
- Implement the enrichment logic for each task and algorithm, using appropriate APIs and libraries, such as Azure Cognitive Services.
- Implement the data transformation and normalization logic for the enriched data, such as aggregating scores or summarizing text.
- Implement the data validation and error handling logic for the enriched data, such as checking for null or invalid data and logging errors.
- Implement appropriate logging and monitoring mechanisms, using Serilog and Prometheus respectively, to track and analyse the performance and usage of the data enrichment service.

User Stories

1. As a news consumer, I want to be able to access the latest trading news from Polygon.io, so that I can stay informed about the latest market trends. (20 man-hours)

Acceptance criteria:

- A background task fetches news data from Polygon.io every hour.
- The news data is stored in the main MongoDB database.
- The news data includes relevant information, such as ticker symbols, headlines, and timestamps.
- Logging is implemented to record any errors or exceptions that occur during the data retrieval process.

2. As a news consumer, I want to be able to search for news by date range, so that I can filter the news based on my preferences. (30 man-hours)

Acceptance criteria:

- An API endpoint is available that accepts a date range parameter.
- The API endpoint returns the news data that falls within the specified date range.
- The news data is returned in JSON format.

3. As a news consumer, I want to be able to search for news by instrument name, so that I can filter the news based on specific stocks or securities. (20 man-hours)

Acceptance criteria:

- An API endpoint is available that accepts an instrument name parameter.
- The API endpoint returns the news data that matches the specified instrument name.
- The news data includes relevant information, such as ticker symbols, headlines, and timestamps.
- The API endpoint includes a default limit of 10 news items per request.

4. As a news consumer, I want to be able to search for news by keyword, so that I can filter the news based on specific topics or themes. (40 man-hours)

Acceptance criteria:

- An API endpoint is available that accepts a keyword parameter.
- The API endpoint returns the news data that matches the specified keyword.
- The news data includes relevant information, such as ticker symbols, headlines, and timestamps.
- The API endpoint includes a default limit of 10 news items per request.

5. As a news consumer, I want to be able to subscribe to news updates, so that I can receive notifications about relevant news items. (20 man-hours)

Acceptance criteria:

An API endpoint is available that accepts a subscription request.

- The subscription request includes relevant information, such as the subscriber's email address or other contact information.
- The subscription data is stored in the database.
- The subscriber receives notifications about relevant news items via email or other communication channel.

6. As a news consumer, I want to be able to access the top 5 latest news items for different instruments, so that I can use them for a conversion tool or other purpose. (30 man-hours)

Acceptance criteria:

- A public API endpoint is available that returns the top 5 latest news items for different instruments.
- The API endpoint is available to all users without authentication or authorization.
- The API endpoint includes a caching mechanism to improve performance and reliability.

7. As a news consumer, I want to be able to authenticate myself before accessing the news data, so that I can ensure the security and privacy of my data. (20 man-hours)

Acceptance criteria:

- An JWT authentication mechanism is implemented.
- The API endpoints are secured using the authentication mechanism.
- The authentication mechanism supports various authentication modes, such as username/password, API key, or social login.

8. As a news consumer, I want to be able to retrieve news data in various formats, so that I can integrate it with different applications or tools. (30 man-hours)

Acceptance criteria:

- The API endpoints support various output formats, such as JSON, XML, or CSV.
- The API endpoints support various query parameters, such as sort order, pagination, or filtering.
- The API endpoints are well-documented and easy to use, with clear examples and explanations.

9. As a news consumer, I want to be able to receive notifications about relevant news items in real-time, so that I can stay up to date with the latest market trends. (40 man-hours)

Acceptance criteria:

- A real-time messaging mechanism is implemented, such as WebSockets or SignalR.

- The messaging mechanism supports various notification modes, such as email, SMS, or push notifications.
- The messaging mechanism is secure and reliable, with appropriate error handling and fallback mechanisms.

10. As a news consumer, I want to be able to view historical news data and trends, so that I can analyse past market behaviour and make informed decisions. (30 man-hours)

Acceptance criteria:

- An API endpoint is available that accepts a date range parameter.
- The API endpoint returns the historical news data that falls within the specified date range.
- The historical news data includes relevant information, such as ticker symbols, headlines, and timestamps.
- The historical news data is returned in JSON format.

11. As a news consumer, I want to be able to share news items with my colleagues or friends, so that I can collaborate and discuss market trends. (20 man-hours)

Acceptance criteria:

- A share feature is implemented that allows users to share news items via email, social media, or other communication channels.
- The share feature is user-friendly and easy to use, with clear instructions and feedback.
- The share feature includes appropriate security and privacy measures, such as spam protection or GDPR compliance.

12. As a news consumer, I want to be able to receive personalized news recommendations based on my preferences and behaviour, so that I can save time and improve my productivity. (50 man-hours)

Acceptance criteria:

- A recommendation system is implemented that analyses the user's behaviour and preferences.
- The recommendation system suggests news items that are relevant and interesting to the user.

- The recommendation system includes appropriate privacy and consent mechanisms, such as GDPR compliance or opt-out options.

Estimation summary

Solution scaffolding – 30 hours

Data Storage Service - 120 hours

News Collection Service - 160 hours

Data Enrichment Service - 200 hours

User story 1 - 20 hours

User story 2 - 30 hours

User story 3 - 20 hours

User story 4 - 40 hours

User story 5 - 20 hours

User story 6 - 30 hours

User story 7 - 20 hours

User story 8 - 30 hours

User story 9 - 40 hours

User story 10 - 30 hours

User story 11 - 20 hours

User story 12 - 50 hours

Additional testing and adjustments – 80 hours

Grand total: 940 man-hours

Estimated sprint breakdown: 80%

Total sprints: 6