SteelEye API Developer Technical Test

Introduction

Developing a Trade Management API can be a difficult but immensely fulfilling experience, especially when venturing into the realm of FastAPI and Pydantic with limited prior knowledge.

Selecting the Appropriate Tools

Understanding FastAPI - As a developer new to FastAPI, gaining a solid grasp of this framework was paramount. FastAPI was chosen due to its intriguing blend of impressive performance capabilities and the curiosity to explore something new. Despite being a relative newcomer to FastAPI, it swiftly proved itself as the ideal framework for constructing a responsive and efficient API.

Pydantic for Data Validation - Pydantic's reputation for robust data validation made it a natural companion to FastAPI. Its user-friendliness and seamless alignment with our data modeling needs rendered it the obvious choice.

The Learning Curve

Mocking the Database - One of the most significant challenges arose from the absence of Elasticsearch, SteelEye's primary data storage technology. To surmount this obstacle, I embarked on the journey of creating a mock database interaction layer from the ground up. This exercise turned out to be a treasure trove of insights into generating randomized trade data while delving into the intricacies of Python's data manipulation capabilities.

Studying Elasticsearch - My unwavering passion for learning prompted me to take the initiative to study Elasticsearch, even though it wasn't a prerequisite for the project. This self-driven learning experience added a profound layer of depth to my understanding of data storage technologies. It reflects my commitment to continuous learning and my eagerness to expand my knowledge horizons. This newfound knowledge may prove invaluable in future projects.

Defining the Schema Model - At the core of any API lies its schema model, a role impeccably fulfilled by Pydantic models. These models provided the essential underpinning, encapsulating vital trade information such as asset class, counterparty, and instrument ID. Mastering the art of structuring these models became a logical step in the quest for maintaining data consistency and integrity.

Endpoint Design and Implementation

Listing Trades - The journey commenced with the creation of the trade-listing endpoint, a logical starting point. However, the quest to accommodate optional query parameters like asset class, date range, and price range required an in-depth exploration of logical thinking. Crafting a versatile API capable of gracefully handling an array of filtering criteria became the objective.

Retrieving a Single Trade - Extending the API's functionality to retrieve a single trade by its unique ID was a logical progression. This endeavor showcased how FastAPI simplifies routing and encourages logical thinking in endpoint design.

Searching Trades - Facilitating users to conduct searches across trades represented a natural evolution. Capitalizing on FastAPI's adept handling of query parameters and Pydantic's prowess in data validation, I embarked on a journey marked by intricate logical thinking. The task involved unraveling the complexities of searching across multiple fields to fetch data aligned with a user's query.

Advanced Filtering - The demand for advanced filtering capabilities added an intriguing layer of complexity to the project. The task at hand involved supporting optional query parameters such as asset class, date range, and trade type. This called for the application of logical thinking and meticulous coding, giving rise to intricate conditional logic to accommodate inclusive maximum and minimum values for price filtering.

Pagination and Sorting - In pursuit of bonus points, the implementation of pagination and sorting emerged as the next logical step. Here, FastAPI's adept handling of query parameters proved invaluable, while the dynamic sorting logic based on different fields demanded logical thinking and adaptability.

Conclusion

The journey of crafting a Trade Management API using FastAPI and Pydantic transcended the mere realm of technology; it was an expedition into the realms of learning and logical thinking. My passion for continuous learning was a driving force throughout this endeavor, prompting me to explore Elasticsearch independently, despite it not being a project requirement. This journey serves as a testament to the significance of learning and logical thinking, highlighting how they are the cornerstone of any successful development endeavor.