lemon.markets

🚀 Case Study: Backend Engineering

The purpose of the task is to understand your way of working, assess your ability to develop highly available and scalable APIs, and your approach to testing the application.

Intro

As we are offering an API for FinTechs to launch investing products or features, we designed a task that gives you a real insight into the challenges that we are working with. Focusing on one of the core processes of our platform, placing a trade. You don't need any special knowledge as it is a logical process, but we would lie if we say it doesn't help. See this as a simplified version of what we are dealing with on a day-to-day basis at lemon.markets. 🍋

Tech stack and repository

We **strongly** encourage you to use **Python for the task**. Please note, that we will take into consideration if you have been working on a different tech stack in the past, but we'd like to see how confident you are picking up Python when you join lemon.markets.

For the implementation, we will provide you with a repository with a basic Python setup to use as the foundation of your solution.

The repository contains a bare setup for a REST API to place orders via a POST - /orders request which can be found in ./app/api.py. Additionally, it contains a building block that symbolizes placing an order at the stock exchange. The folder tests contains the setup for tests with the framework pytest.

Forking and submitting

Please submit a zip file via the submission link at least 24h before the review meeting.

Your task

Your main task is to finalize the missing parts of the above repository. To give you some guidance we have structured the exercise in 5 steps that you should walk through in your solution. If you get all excited about it, we also have some bonus questions, that came up when we were solving the task ourselves.

Finalize the missing parts in the repository so that the following requirements are fulfilled:

1. A valid request to POST /orders should result in the order being stored in a database of your choice.
2. The created order should be placed at the stock exchange (if you're working in Python, use the method place_order provided in stock_exchange.py, else please create a similar function or method based on the Python example)
3. The endpoint should return a status code of 201 and the created order details, provided that the order has been saved in the database *and* it is guaranteed that the order *will* be placed on the stock exchange.
4. In case of an error in the endpoint, it should return the status code 500 and the body {"message": "Internal server error while placing the order"}
5. The API should be highly scalable and reliable. The reliability of the provided stock exchange (place_order function) should not impact the reliability of the POST /orders endpoint.

☝️ Additionally, please add some tests and document how you would test the application as a whole and its pieces.

📝 Please include a solution.md file where you document your decisions, assumptions, and also improvements you would like to incorporate in the future.

⏰ We value your time, so we do not expect you to spend more than 6 hours preparing the solution. 🤗

Bonus tasks
How would you change the system if we would receive a high volume of async updates to the orders placed through a socket connection on the stock exchange, e.g. execution information? Please outline the changes in the solution.md.

Feel free to add a GitHub actions workflow to test the application.

Feel free to add a Dockerfile.

We are excited to discuss your solution with you and hope you enjoy this task 🍋