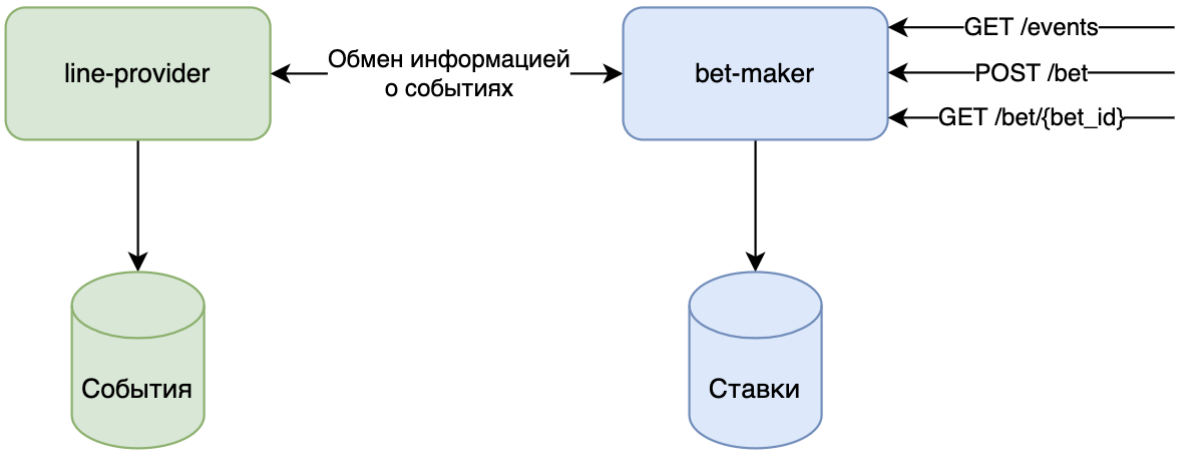
Общая информация

Вам предстоит разработать небольшую систему, принимающую пользовательские ставки на определённые события (например, спортивные).

Система должна состоять из двух независимых сервисов:

- сервис **line-provider** — провайдер информации о событиях,
- сервис **bet-maker**, принимающий ставки на эти события от пользователя.



Описание сервиса line-provider

Сервис должен выдавать информацию о событиях, на которые можно совершать ставки. Наша система будет максимально упрощённой, поэтому мы будем принимать ставки только на выигрыш первой команды.

Таким образом, информация о событии должна содержать как минимум:

- *уникальный идентификатор события* — строка или число,
- *коэффициент ставки на выигрыш* — строго положительное число с двумя знаками после запятой,
- *дедлайн для ставок* — таймстемп, до которого на событие принимаются ставки,

- *текущий статус события*.

В нашей простой системе событие может иметь один из трёх статусов:

- *незавершённое*,
- *завершено выигрышем первой команды*,
- *завершено выигрышем второй команды* и, соответственно, поражением первой (ничьих в наших событиях не бывает).

API сервиса **line-provider** не регламентировано и остаётся на ваше усмотрение. Информация о событиях может храниться в памяти, без использования каких-либо сторонних хранилищ.

Логично предусмотреть утилитарное API, позволяющее динамически создавать новые события и менять статус существующих.

Шаблонную имплементацию сервиса **line-provider** можно взять на гитхабе: https://github.com/SuminAndrew/bsw-test-line-provider. В ней информация о событиях хранится в памяти, реализовано простейшее апи для получения и изменения событий. От этой имплементации можно отталкиваться при проектировании сервиса.

Описание сервиса bet-maker

Сервис **bet-maker** отвечает за постановку ставок на события пользователями.

Информация о событиях должна получаться из сервиса **line-provider**. В частности, сервису **bet-maker** необходимо узнавать об изменении статуса событий (переход в статус *завершено* с выигрышем или поражением), чтобы понять выиграла ставка или проиграла.

Взаимодействие между сервисами может быть реализовано, к примеру, запросами в сервис **line-provider**, вызовом callback-урла **bet-maker** при изменении статуса события на стороне **line-provider** или обменом сообщений между сервисами через очередь.

Сервис **bet-maker** должен обладать следующим API:

GET /events
Возвращает список событий, на которые можно совершить ставку. Таковыми считаются все события, для которых ещё не наступил *дедлайн для ставок*.

Список событий предоставляется сервисом **line-provider**, в целях оптимизации взаимодействий допускается небольшое отставание в «свежести» списка для новых событий (новые события могут появляться в API сервиса **bet-maker** с небольшим отставанием от **line-provider**).

POST /bet
Совершает ставку на событие.
В теле запроса необходимо передать JSON-объект, содержащий:

- *идентификатор события* — строка или число,
- *сумму ставки* — строго положительное число с двумя знаками после запятой.

В ответе необходимо возвращать как минимум уникальный идентификатор ставки.

GET /bets
Возвращает историю всех сделанных ставок.

Должен возвращать массив JSON-объектов, содержащих информацию о ставках: их идентификаторы и текущие статусы.

Статус ставки может быть одним из следующих:

- *ещё не сыграла* (соответствующее событие ещё не завершилось),
- *выиграла* (событие завершилось выигрышем первой команды),
- *проиграла* (событие завершилось проигрышем первой команды).

Информация о ставках должна сохраняться в хранилище на ваш выбор (Redis, PostreSQL, ...).

Требования к фреймворкам, инфраструктуре и коду

Рекомендованный фреймворк для реализации сервисов — fastapi, версия Python — 3.10. Взаимодействия между сервисами **line-provider** и **bet-maker**, а также между сервисами и их хранилищами должны быть полностью асинхронными.

Сервисы **line-provider** и **bet-maker**, а также дополнительные хранилища и прочие инфраструктурные элементы должны быть докеризированы и запускаться через docker compose.

Отдельными плюсами будут следование стандарту PEP8, наличие тестов, использование type hints.