Общая информация

Вам предстоит разработать небольшой сервис, принимающий пользовательские ставки на спортивные события. В реальном мире пользователи могут совершать ставки на различные исходы событий (результат игры, разница в счёте и пр.). Мы же для простоты будем рассматривать единственный исход — победу первой команды.

Система должна состоять из единственного сервиса **bet-maker**.

Описание сервиса bet-maker

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

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

- *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 — 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 или ничьей).

PUT /events/{event_id}
Сообщает о том, что в событии с указанным *event_id* произошли изменения.

В теле запроса необходимо передать JSON-объект, содержащий единственное поле — *новый статус события*. Поле может принимать одно из следующих строковых значений: *WIN*, *LOSE*.

Обработчик запроса должен соответствующим образом обновлять статусы всех совершённых на данное событие ставок.

Информация о ставках должна сохраняться в хранилище на ваш выбор (Redis, PostreSQL, ...). События как таковые хранить не требуется, мы будем работать только с их идентификаторами.

Требования к фреймворкам, инфраструктуре и коду

Рекомендованный фреймворк для реализации сервисов — fastapi, минимальная рекомендованная версия Python — 3.10. Все взаимодействия должны быть полностью асинхронными.

Сервис **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.