Тестовое задание

К тестовому заданию прилагается клиентская документация Mascot Gaming в виде PDF-файла. Понадобится лишь только та часть, которая описывает взаимодействие по **Seamless API**.

Ваша задача – реализовать обработчик **Seamless API**.

Вы выступаете в роли нашего клиента, который интегрирует нашу игровую платформу в свой бизнес.

Клиент получает по API возможность создавать игровые сессии для своих игроков на нашей игровой платформе.

Каждая игровая сессия при действиях внутри игры обращается к клиенту по **Seamless API**.
Чаще всего это просто "вращение барабанов" – ставка, расчёт комбинации и соответствующий выигрыш. Иногда бывают “бесплатные вращения” в процессе игры. Это когда вы не платите за ставку, но можете получить выигрыш.

Seamless API – это интерфейс взаимодействия игровой сессии с кошельком игрока на стороне клиента. Обработчик этого API со стороны клиента и нужно реализовать.

На выходе вы должны получить приложение с HTTP-сервером с одним эндпоинтом на корне, который принимает данные POST-ом по протоколу JSON-RPC 2.0 over HTTP и имеет 3 метода.

Для хранения балансов игроков и транзакций используйте базу данных Postgres.

В pdf-файле есть ссылки на спецификации, которые мы использовали для реализации данного API и по которому, как предполагается, мы будем взаимодействовать с этим сервисом.

Полезные ссылки из PDF-файла:

- Спецификация JSON-RPC 2.0: https://www.jsonrpc.org/specification
- Спецификация HTTP-транспорта: https://www.simple-is-better.org/json-rpc/transport_http.html

По сути сервис будет работать с тремя действиями от платформы:

- **getBalance:** Вернуть актуальный баланс игрока.

- **withdrawAndDeposit:** Принять денежную транзакцию (**withdraw** – ставка и **deposit** – выигрыш) от игровой платформы, совершить операции над балансом и вернуть актуальный баланс игрока в ответе.

  В любом запросе значения полей **deposit**, **withdraw** — целые неотрицательные числа (то есть они в том числе могут быть равны нулю).

  Операции с балансом всегда выполняются в такой последовательности:
  1) balance -= **withdraw** (с валидацией, что в результате balance >= 0)
  2) balance += **deposit**

- **rollbackTransaction:** Откатить денежную транзакцию. Ожидается успешный ответ, иначе запрос будет повторяться примерно 1 раз в минуту до тех пор, пока не пройдёт 24 часа или не вернётся успешный ответ (успешным считается любой неошибочный ответ).

Нужно учесть, что дублирующийся идентичный запрос (с одинаковым **transactionRef**) к **withdrawAndDeposit** или к **rollbackTransaction** не должен приводить к повторению операции (запросы идемпотентные).

В случае, если пришёл запрос **rollbackTransaction** с **transactionRef**, который ещё не был зарегистрирован в сервисе, нужно сохранить денежную транзакцию и пометить её, как откаченная. Если позже придёт запрос на **withdrawAndDeposit** с таким же **transactionRef**, сервис должен ответить ошибкой о том, что транзакция не может быть совершена, т.к. была откачена.

Важные замечания:
1. Денежные значения всегда указываются в "копейках" (например, **1 EUR** будет выглядеть как **100** единиц с валютой **EUR**).
2. Один игрок – одна валюта. В примерах игрок **player1** имеет баланс только в валюте **EUR**.
3. Поле **callerId** в API – это идентификатор нашего клиента, который в данном случае можно просто игнорировать, т.к. он всегда будет одним и тем же (равен 1 в примерах).
4. **playerName** – это уникальный идентификатор игрока.

Сопутствующие пояснения:
1. **gameRoundRef** – это идентификатор игрового раунда в конкретной игровой сессии. В каждом раунде игровой сессии может быть >= 0 денежных транзакций.
2. Поле **reason** сигнализирует о том, что раунд в игровой сессии завершился ("**GAME_PLAY_FINAL**") или не завершился и продолжится ("**GAME_PLAY**").

Обычно алгоритм такой:

1. Игрок создаёт игровую сессию.
2. Клиенту приходит запрос **getBalance**.
3. Игрок совершает действия в игре – приходят **withdrawAndDeposit**.
4. В случае какой-либо ошибки (в том числе сетевой) иногда могут приходить **rollbackTransaction**.
5. Игрок прекращает играть.

Запрос на получение баланса игрока **player1** с валютой **EUR** может выглядеть так:

```
POST /mascot/seamless HTTP/1.1
Host: localhost:8080
Content-Length: 126
Accept: application/json
Content-Type: application/json
Accept-Encoding: gzip

{"jsonrpc":"2.0","method":"getBalance","params":{"callerId":1,"playerName":"player1",
"currency":"EUR","gameId":"riot"},"id":0}

---

HTTP/1.1 200 OK
Connection: keep-alive
Content-Type: application/json
Date: Mon, 22 Aug 2022 00:00:00 GMT
Vary: Accept-Encoding

{"jsonrpc":"2.0","id":0,"result":{"balance":10000}} // 100 EUR
```

Запрос на совершение денежной транзакции **1:UOwGgNHPgq3OkqRE** (ставка **4 EUR**, выигрыш **2 EUR**) в игровой сессии **qx9sgvvpihtrlug** игрока **player1** может выглядеть так:

```
POST /mascot/seamless HTTP/1.1
Host: localhost:8080
Content-Length: 339
Accept: application/json
Content-Type: application/json
Accept-Encoding: gzip

{"jsonrpc":"2.0","method":"withdrawAndDeposit","params":{"callerId":1,"playerName":"p
layer1","withdraw":400,"deposit":200,"currency":"EUR","transactionRef":"1:UOwGgNHPgq3
OkqRE","gameRoundRef":"1wawxl:39","gameId":"riot","reason":"GAME_PLAY_FINAL","session
Id":"qx9sgvvpihtrlug","spinDetails":{"betType":"spin","winType":"standart"}},"id":0}

---

HTTP/1.1 200 OK
Connection: keep-alive
Content-Type: application/json
Date: Mon, 22 Aug 2022 00:00:00 GMT
Vary: Accept-Encoding

{"jsonrpc":"2.0","id":0,"result":{"newBalance":9800,"transactionId":"1413628395"}}
```

Запрос на откат денежной транзакции **1:UOwGgNHPgq3OkqRE** в игровой сессии **qx9sgvvpihtrlug** игрока **player1** может выглядеть так:

```
POST /mascot/seamless HTTP/1.1
Host: localhost:8080
Content-Length: 213
Accept: application/json
Content-Type: application/json
Accept-Encoding: gzip

{"jsonrpc":"2.0","method":"rollbackTransaction","params":{"callerId":1,"playerName":"
player1","transactionRef":"1:UOwGgNHPgq3OkqRE","gameId":"riot","sessionId":"qx9sgvvpi
htrlug","gameRoundRef":"1wawxl:39"},"id":0}

---

HTTP/1.1 200 OK
Connection: keep-alive
Content-Type: application/json
Date: Mon, 22 Aug 2022 00:00:00 GMT
Vary: Accept-Encoding

{"jsonrpc":"2.0","id":0,"result":null} // успешный ответ
```