Documentación de la API

Usuarios

1. `/users`

GET

- **Descripción**: Obtiene una lista de todos los usuarios registrados.
- **Parámetros**: Ninguno.
- **Respuestas**:
  - **200 OK**:

```json
[
  {
    "user_id": "1",
    "email": "usuario1@example.com",
    "is_admin": false,
    "wallet": 100.0
  },
  ...
]
```

POST

- **Descripción**: Crea un nuevo usuario.
- **Parámetros (body)**:

- `email` (string, requerido): Correo del usuario.
- `wallet` (float, opcional): Cantidad inicial en la billetera.
- `is_admin` (boolean, opcional): Si el usuario será administrador.

- **Respuestas**:
  - **201 Created**:

```json
{
  "user_id": "1",
  "email": "usuario1@example.com",
  "is_admin": true,
  "wallet": 100.0
}
```

  - **400 Bad Request**:

```json
{ "email": ["Este campo es obligatorio."] }
```

---

2. `/users/admin/<user_id>`

GET

- **Descripción**: Verifica si un usuario es administrador.
- **Parámetros**:
  - `user_id` (string, requerido): ID del usuario.
- **Respuestas**:
  - **200 OK**:

```json
{ "is_admin": true }
```

- **404 Not Found**:

```json
{ "is_admin": false, "error": "Usuario no encontrado" }
```

---

3. `/users/<user_id>`

GET

- **Descripción**: Obtiene los detalles de un usuario específico.
- **Parámetros**:
  - `user_id` (string, requerido): ID del usuario.
- **Respuestas**:
  - **200 OK**:

```json
{
  "user_id": "1",
  "email": "usuario1@example.com",
  "is_admin": false,
  "wallet": 100.0
}
```

  - **404 Not Found**:

```json
{ "error": "No encontrado" }
```

PUT

- **Descripción**: Actualiza los datos de un usuario.
- **Parámetros (body)**: Cualquier campo permitido por el modelo del usuario (`email`, `wallet`, etc.).
- **Respuestas**:

- **200 OK**:

```json
{
  "user_id": "1",
  "email": "nuevo_correo@example.com",
  "wallet": 150.0
}
```

- **400 Bad Request**:

```json
{ "email": ["Formato inválido."] }
```

DELETE

- **Descripción**: Elimina un usuario.
- **Parámetros**: Ninguno.
- **Respuestas**:
  - **204 No Content**
  - **404 Not Found**:

```json
{ "error": "Usuario no encontrado" }
```

---

4. `/users/wallet`

PATCH

- **Descripción**: Agrega una cantidad de dinero a la wallet de un usuario.
- **Parámetros (body)**:
  - `user` (string, requerido): ID del usuario.
  - `quantity` (float, requerido): Cantidad a agregar.

- **Respuestas**:
  - **200 OK**:

```json
{
  "message": "Cantidad agregada a la wallet exitosamente",
  "user_id": "1",
  "new_balance": 200.0
}
```

  - **400 Bad Request**:

```json
{ "error": "user y quantity son campos requeridos" }
```

  - **404 Not Found**:

```json
{ "error": "Usuario no encontrado" }
```

  - **500 Internal Server Error**:

```json
{ "error": "Error al procesar la operación" }
```

Fixtures

1. `/fixtures`

GET

- **Descripción**: Obtiene una lista de fixtures (partidos) futuros.
- **Parámetros (query)**:
  - `home` (string, opcional): Filtrar por nombre del equipo local.

- `away` (string, opcional): Filtrar por nombre del equipo visitante.
- `date` (string, opcional): Filtrar por fecha.

- **Respuestas**:
  - **200 OK**:

```json
[
  {
    "fixture_id": "1",
    "home_team_name": "Equipo A",
    "away_team_name": "Equipo B",
    "date": "2024-12-01T15:00:00Z"
  },
  ...
]
```

POST

- **Descripción**: Crea un nuevo fixture.
- **Parámetros (body)**: Campos según el modelo de `Fixture`.
- **Respuestas**:
  - **201 Created**:

```json
{
  "fixture_id": "1",
  "home_team_name": "Equipo A",
  "away_team_name": "Equipo B",
  "date": "2024-12-01T15:00:00Z"
}
```

  - **400 Bad Request**: Datos inválidos.

---

2. `/fixtures/<fixture_id>`

GET

- **Descripción**: Obtiene un fixture específico.
- **Parámetros**:
  - `fixture_id` (string, requerido): ID del fixture.
- **Respuestas**:
  - **200 OK**: Detalles del fixture.
  - **404 Not Found**:

```json
{ "error": "Fixture no encontrado" }
```

PUT

- **Descripción**: Actualiza un fixture.
- **Parámetros (body)**: Campos según el modelo de `Fixture`.
- **Respuestas**:
  - **200 OK**:

```json
{
  "fixture_id": "1",
  "home_team_name": "Equipo A",
  "away_team_name": "Equipo B",
  "date": "2024-12-01T15:00:00Z"
}
```

  - **400 Bad Request**: Datos inválidos.

---

3. `/fixtures/requests`

POST

- **Descripción**: Procesa solicitudes de compra de bonos desde el canal MQTT.

- **Parámetros (body)**:
  - `fixture_id` (string, requerido): ID único del partido.
  - `quantity` (integer, requerido): Cantidad de bonos solicitados.
  - `wallet` (boolean, requerido): Indica si se usará la billetera.
- **Respuestas**:
  - **201 Created**:

```json
{
  "request_id": "REQ123",
  "message": "Bonos reservados temporalmente"
}
```

  - **400 Bad Request**:

```json
{ "error": "Datos inválidos" }
```

  - **404 Not Found**:

```json
{ "error": "Fixture no encontrado" }
```

4. `/bonos/validations/{request_id}`

PUT

- **Descripción**: Valida o rechaza una solicitud de bonos y ajusta los bonos disponibles si la solicitud es rechazada.
- **Parámetros (body)**:
  - `valid` (boolean, requerido): Indica si la solicitud es válida.
- **Respuestas**:
  - **200 OK**:

    ```
    json

    {
      "message": "Compra con request_id REQ123 aprobada."
    }
    ```

  - **400 Bad Request**:

    ```
    json

    { "error": "El campo 'valid' no fue encontrado en los datos de validación" }
    ```

  - **404 Not Found**:

    ```
    json

    { "error": "Solicitud no encontrada" }
    ```

---

5. `/bonos/history`

POST

- **Descripción**: Procesa información histórica de partidos y actualiza el estado de los bonos asociados.
- **Parámetros (body)**:
  - `fixtures` (array, requerido): Lista de objetos con información de partidos y goles.

    ```
    json
    ```

```
[
  {
    "fixture": {"id": "FIX123", ...},
    "goals": {"home": 2, "away": 1}
  }
]
```

- **Respuestas:**
  - **200 OK:**

    ```json
    {
      "status": "success",
      "message": "Bonos procesados correctamente",
      "fixtures_not_found": ["FIX456"],
      "fixtures_processed": 1,
      "total_fixtures": 2
    }
    ```

  - **400 Bad Request:**

    ```json
    { "error": "JSON inválido o datos faltantes" }
    ```

  - **500 Internal Server Error:**

    ```json
    { "error": "Error en el procesamiento de bonos o en la determinación de resultados" }
    ```

Transacciones

1. `/verificar-transaccion`

POST

- **Descripción**: Confirma el estado de una transacción realizada a través de Webpay. Recibe un token de transacción y verifica si el pago fue exitoso.
- **Parámetros (body)**:
  - `token_ws` (string, requerido): Token único de la transacción.
- **Respuestas**:
  - **200 OK**:

    ```json
    {
      "message": "Pago exitoso",
      "buy_order": "ORDEN123",
      "amount": 1000,
      "transaction_date": "2024-11-25T10:30:00Z",
      "card_detail": {"card_number": "XXXX-XXXX-XXXX-1234"},
      "status": "AUTHORIZED"
    }
    ```

  - **400 Bad Request**:

    ```json
    { "error": "Token no proporcionado o error en la transacción" }
    ```

  - **500 Internal Server Error**:

    ```json
    { "error": "Error al confirmar la transacción" }
    ```

Workers

1. `/workers`

GET

- **Descripción**: Verifica el estado de un servicio externo (Job Master) enviando una solicitud de heartbeat.
- **Respuestas**:
  - **200 OK**:

    ```json
    {
      "status": "success",
      "data": { ... }
    }
    ```

  - **500 Internal Server Error**:

    ```json
    { "error": "Fallo al conectar con Job Master o respuesta no válida" }
    ```

Recomendaciones

1. `/store-recommendation`

POST

- **Descripción**: Guarda una nueva recomendación generada para un usuario.
- **Parámetros (body)**: Los parámetros varían según el modelo `Recommendation`.
- **Respuestas**:
  - **201 Created**:

    ```json
    {
      "id": 1,
      "user_id": "1",
      "fixture_id": "1",
      "benefit_score": 0.9,
    ```

```
  "recommendation_text": "Recomendación para el usuario"
}
```

- **400 Bad Request**: Datos inválidos.

---

2. `/user-recommendations/{user_id}`

GET

- **Descripción**: Obtiene las recomendaciones de un usuario, ordenadas por el beneficio de la recomendación.
- **Parámetros**:
  - `user_id` (string, requerido): ID del usuario.
- **Respuestas**:
  - **200 OK**:

```
json

[
  {
    "fixture_id": "1",
    "home_team_name": "Equipo A",
    "away_team_name": "Equipo B",
    "date": "2024-12-01T15:00:00Z"
  },
  ...
]
```

  - **404 Not Found**:

```
json

{ "message": "No hay recomendaciones para este usuario." }
```

---

3. `/user-purchases/{user_id}`

GET

- **Descripción**: Obtiene la lista de bonos comprados por un usuario.
- **Parámetros**:
  - `user_id` (string, requerido): ID del usuario.
- **Respuestas**:
  - **200 OK**:

```json
[
  {
    "request_id": "123e4567-e89b-12d3-a456-426614174000",
    "fixture_id": "1",
    "quantity": 2,
    "group_id": "6",
    "league_name": "Premier League",
    "round": "10",
    "datetime": "2024-11-24T12:00:00Z",
    "date": "2024-11-25",
    "seller": 6,
    "wallet": false,
    "acierto": false
  }
]
```
