Crypto Sim Backend API 文档

基础信息

- **Base URL**: `http://localhost:3000/api`
- **环境**: Development
- **版本**: 1.0.0

开发模式配置

当前系统配置为开发模式，已禁用以下安全特性以方便测试：

- **JWT 验证**: 已跳过 (`AUTH_SKIP_JWT_VERIFICATION=true`)
- **Redis**: 已禁用 (`REDIS_ENABLED=false`)

目录

1. 认证接口
2. 用户接口
3. 交易流水接口

认证接口

1. 用户注册

POST `/api/auth/register`

注册新用户账户。

请求体:

```
{
  "email": "user@example.com",
  "password": "password123",
  "displayName": "用户名",
  "roles": ["trader"]  // 可选，默认为 ["trader"]
}
```

响应 (201 Created):

```
{
  "user": {
    "id": "uuid",
    "email": "user@example.com",
    "displayName": "用户名",
    "roles": ["trader"],
```

```
    "accountBalance": 10000.00000000,
    "totalProfitLoss": 0.00000000,
    "winRate": 0.00,
    "totalTrades": 0,
    "verificationStatus": "UNVERIFIED",
    "isActive": true,
    "createdAt": "2025-10-22T00:00:00.000Z"
  },
  "accessToken": "eyJhbGc...JWT_TOKEN",
  "refreshToken": "eyJhbGc...REFRESH_TOKEN"
}
```

2. 用户登录

POST `/api/auth/login`

使用邮箱和密码登录。

请求体:

```
{
  "email": "user@example.com",
  "password": "password123"
}
```

响应 (200 OK):

```
{
  "user": {
    "id": "uuid",
    "email": "user@example.com",
    "displayName": "用户名",
    "roles": ["trader"],
    "accountBalance": 9900.00000000,
    "totalProfitLoss": -100.00000000,
    "winRate": 45.50,
    "totalTrades": 10,
    "verificationStatus": "VERIFIED"
  },
  "accessToken": "eyJhbGc...JWT_TOKEN",
  "refreshToken": "eyJhbGc...REFRESH_TOKEN"
}
```

3. 刷新令牌

POST `/api/auth/refresh`

使用刷新令牌获取新的访问令牌。

请求体:

```
{
  "refreshToken": "eyJhbGc...REFRESH_TOKEN"
}
```

响应 (200 OK):

```
{
  "accessToken": "eyJhbGc...NEW_JWT_TOKEN",
  "refreshToken": "eyJhbGc...NEW_REFRESH_TOKEN"
}
```

4. 获取当前用户信息

GET `/api/auth/me`

获取当前登录用户的详细信息。

Headers:

```
Authorization: Bearer {accessToken}
```

响应 (200 OK):

```
{
  "id": "uuid",
  "email": "user@example.com",
  "displayName": "用户名",
  "roles": ["trader"],
  "accountBalance": 9900.00000000,
  "totalProfitLoss": -100.00000000,
  "winRate": 45.50,
  "totalTrades": 10,
  "verificationStatus": "VERIFIED",
  "isActive": true,
  "lastLoginAt": "2025-10-22T00:00:00.000Z",
  "createdAt": "2025-10-22T00:00:00.000Z",
  "updatedAt": "2025-10-22T00:00:00.000Z"
}
```

5. 登出

POST `/api/auth/logout`

登出当前用户，撤销刷新令牌。

Headers:

```
Authorization: Bearer {accessToken}
```

响应 (200 OK):

```
{
  "success": true
}
```

交易流水接口

1. 创建交易

POST `/api/transactions`

创建新的交易订单（买涨/买跌）。

Headers:

```
Authorization: Bearer {accessToken}
```

请求体:

```
{
  "assetType": "BTC",           // 资产类型: BTC, ETH, ADA, SOL等
  "direction": "CALL",          // CALL=买涨, PUT=买跌
  "duration": 60,               // 时长（秒）, 如60秒
  "investAmount": 100,          // 投入金额
  "returnRate": 0.85            // 报酬率, 0.85表示85%
}
```

响应 (201 Created):

```
{
  "id": "uuid",
  "userId": "user-uuid",
  "orderNumber": "TXN1729581234567ABC123",  // 自动生成的订单号
  "assetType": "BTC",
  "direction": "CALL",
  "entryTime": "2025-10-22T09:00:00.000Z",    // 入场时间
  "expiryTime": "2025-10-22T09:01:00.000Z",   // 到期时间
  "duration": 60,
  "entryPrice": 65000.00000000,               // 入场价
  "currentPrice": 65000.00000000,             // 当前价
  "exitPrice": null,                          // 出场价（未结算时为null）
  "spread": 6.50000000,                       // 点差
  "investAmount": 100.00000000,               // 投入金额
  "returnRate": 0.8500,                       // 报酬率
  "actualReturn": 0.00000000,                 // 实得（未结算时为0）
  "status": "PENDING",                        // PENDING=进行中
  "createdAt": "2025-10-22T09:00:00.000Z",
  "updatedAt": "2025-10-22T09:00:00.000Z",
```

```
  "settledAt": null
}
```

错误响应:

```
// 余额不足
{
  "statusCode": 400,
  "message": "账户余额不足。当前余额: 50, 需要: 100",
  "error": "Bad Request"
}

// 用户不存在
{
  "statusCode": 404,
  "message": "用户不存在",
  "error": "Not Found"
}
```

2. 获取交易列表

GET `/api/transactions`

获取当前用户的交易记录列表，支持分页和筛选。

Headers:

```
Authorization: Bearer {accessToken}
```

Query Parameters:

- `page` (可选): 页码，默认 1
- `limit` (可选): 每页数量，默认 20
- `assetType` (可选): 资产类型筛选，如 BTC
- `direction` (可选): 交易方向筛选，CALL 或 PUT
- `status` (可选): 状态筛选，PENDING/SETTLED/CANCELED

示例请求:

```
GET /api/transactions?page=1&limit=10&assetType=BTC&status=SETTLED
```

响应 (200 OK):

```
{
  "data": [
    {
      "id": "uuid",
      "userId": "user-uuid",
```

```
      "orderNumber": "TXN1729581234567ABC123",
      "assetType": "BTC",
      "direction": "CALL",
      "entryTime": "2025-10-22T09:00:00.000Z",
      "expiryTime": "2025-10-22T09:01:00.000Z",
      "duration": 60,
      "entryPrice": 65000.00000000,
      "currentPrice": 65100.00000000,
      "exitPrice": 65100.00000000,
      "spread": 6.50000000,
      "investAmount": 100.00000000,
      "returnRate": 0.8500,
      "actualReturn": 185.00000000,      // 盈利：本金100 + 收益85
      "status": "SETTLED",
      "createdAt": "2025-10-22T09:00:00.000Z",
      "updatedAt": "2025-10-22T09:01:05.000Z",
      "settledAt": "2025-10-22T09:01:05.000Z"
    }
  ],
  "total": 25,        // 总记录数
  "page": 1,          // 当前页
  "limit": 10         // 每页数量
}
```

3. 获取交易详情

GET `/api/transactions/:orderNumber`

根据订单号获取交易的详细信息。

Headers:

```
Authorization: Bearer {accessToken}
```

示例请求:

```
GET /api/transactions/TXN1729581234567ABC123
```

响应 (200 OK):

```
{
  "id": "uuid",
  "userId": "user-uuid",
  "orderNumber": "TXN1729581234567ABC123",
  "assetType": "BTC",
  "direction": "CALL",
  "entryTime": "2025-10-22T09:00:00.000Z",
  "expiryTime": "2025-10-22T09:01:00.000Z",
  "duration": 60,
  "entryPrice": 65000.00000000,
```

```
  "currentPrice": 65050.00000000,
  "exitPrice": null,
  "spread": 6.50000000,
  "investAmount": 100.00000000,
  "returnRate": 0.8500,
  "actualReturn": 0.00000000,
  "status": "PENDING",
  "createdAt": "2025-10-22T09:00:00.000Z",
  "updatedAt": "2025-10-22T09:00:30.000Z",
  "settledAt": null
}
```

错误响应:

```
{
  "statusCode": 404,
  "message": "订单 TXN1729581234567ABC123 不存在",
  "error": "Not Found"
}
```

4. 手动结算交易

POST `/api/transactions/:orderNumber/settle`

手动结算交易（测试用途，正常情况下由系统自动结算）。

Headers:

```
Authorization: Bearer {accessToken}
```

示例请求:

```
POST /api/transactions/TXN1729581234567ABC123/settle
```

响应 (200 OK):

```
{
  "id": "uuid",
  "userId": "user-uuid",
  "orderNumber": "TXN1729581234567ABC123",
  "assetType": "BTC",
  "direction": "CALL",
  "entryTime": "2025-10-22T09:00:00.000Z",
  "expiryTime": "2025-10-22T09:01:00.000Z",
  "duration": 60,
  "entryPrice": 65000.00000000,
  "currentPrice": 65100.00000000,
  "exitPrice": 65100.00000000,          // 结算时的价格
  "spread": 6.50000000,
  "investAmount": 100.00000000,
```

```
  "returnRate": 0.8500,
  "actualReturn": 185.00000000,        // 盈利
  "status": "SETTLED",
  "createdAt": "2025-10-22T09:00:00.000Z",
  "updatedAt": "2025-10-22T09:01:05.000Z",
  "settledAt": "2025-10-22T09:01:05.000Z"
}
```

盈亏计算规则:

- **买涨(CALL)盈利**: 出场价 > 入场价 → 返还本金 + 收益 (investAmount * (1 + returnRate))
- **买涨(CALL)亏损**: 出场价 ≤ 入场价 → 损失本金 (actualReturn = -investAmount)
- **买跌(PUT)盈利**: 出场价 < 入场价 → 返还本金 + 收益
- **买跌(PUT)亏损**: 出场价 ≥ 入场价 → 损失本金

错误响应:

```
// 订单不存在
{
  "statusCode": 404,
  "message": "订单 TXN1729581234567ABC123 不存在",
  "error": "Not Found"
}

// 订单已结算
{
  "statusCode": 400,
  "message": "订单 TXN1729581234567ABC123 已经结算或取消",
  "error": "Bad Request"
}
```

5. 取消交易

POST `/api/transactions/:orderNumber/cancel`

取消进行中的交易，退还本金（只能取消未结算的交易）。

Headers:

```
Authorization: Bearer {accessToken}
```

示例请求:

```
POST /api/transactions/TXN1729581234567ABC123/cancel
```

响应 (200 OK):

```
{
  "id": "uuid",
```

```
  "userId": "user-uuid",
  "orderNumber": "TXN1729581234567ABC123",
  "assetType": "BTC",
  "direction": "CALL",
  "entryTime": "2025-10-22T09:00:00.000Z",
  "expiryTime": "2025-10-22T09:01:00.000Z",
  "duration": 60,
  "entryPrice": 65000.00000000,
  "currentPrice": 65000.00000000,
  "exitPrice": null,
  "spread": 6.50000000,
  "investAmount": 100.00000000,
  "returnRate": 0.8500,
  "actualReturn": 0.00000000,        // 取消不计算盈亏
  "status": "CANCELED",
  "createdAt": "2025-10-22T09:00:00.000Z",
  "updatedAt": "2025-10-22T09:00:30.000Z",
  "settledAt": null
}
```

错误响应:

```
{
  "statusCode": 400,
  "message": "订单 TXN1729581234567ABC123 不能取消",
  "error": "Bad Request"
}
```

6. 获取用户统计数据

GET `/api/transactions/statistics`

获取当前用户的交易统计数据。

Headers:

```
Authorization: Bearer {accessToken}
```

响应 (200 OK):

```
{
  "accountBalance": 10500.00,        // 当前账户余额
  "totalProfitLoss": 500.00,         // 总盈亏
  "winRate": 65.50,                  // 胜率 (%)
  "totalTrades": 20,                 // 总交易次数（包括未结算）
  "settledTrades": 18,               // 已结算交易次数
  "winningTrades": 12,               // 盈利交易次数
  "losingTrades": 6                  // 亏损交易次数
}
```

7. 自动结算过期交易

POST `/api/transactions/auto-settle`

触发自动结算所有过期的交易（管理员或定时任务使用，已设置为 Public）。

响应 (200 OK):

```
15  // 返回结算成功的交易数量
```

数据模型

User (用户)

```
{
  id: string;                       // UUID
  email: string;                    // 邮箱（唯一）
  displayName: string;              // 显示名称
  passwordHash: string;             // 密码哈希
  refreshTokenHash?: string;        // 刷新令牌哈希
  roles: string[];                  // 角色数组 ["trader", "admin"]
  isActive: boolean;                // 是否激活
  lastLoginAt?: Date;               // 最后登录时间

  // 交易相关字段
  accountBalance: number;           // 账户余额，默认 10000
  totalProfitLoss: number;          // 总盈亏
  winRate: number;                  // 胜率 (0-100)
  totalTrades: number;              // 交易次数
  verificationStatus: string;       // 身份状态：UNVERIFIED | VERIFIED

  createdAt: Date;                  // 创建时间
  updatedAt: Date;                  // 更新时间
}
```

TransactionLog (交易流水)

```
{
  id: string;                       // UUID
  userId: string;                   // 用户 ID
  orderNumber: string;              // 订单号（唯一）

  // 交易资产和方向
  assetType: string;                // 资产类型：BTC, ETH, ADA, SOL等
  direction: TradeDirection;        // CALL=买涨, PUT=买跌

  // 时间相关
  entryTime: Date;                  // 入场时间
  expiryTime: Date;                 // 到期时间
```

```
  duration: number;                 // 时长（秒）

  // 价格相关
  entryPrice: number;               // 入场价
  currentPrice?: number;            // 当前价格
  exitPrice?: number;               // 出场价格
  spread: number;                   // 点差

  // 金额相关
  investAmount: number;             // 投入金额
  returnRate: number;               // 报酬率 (0.85 = 85%)
  actualReturn: number;             // 实得（可正可负）

  // 状态
  status: TransactionStatus;        // PENDING | SETTLED | CANCELED

  // 时间戳
  createdAt: Date;                  // 创建时间
  updatedAt: Date;                  // 更新时间
  settledAt?: Date;                 // 结算时间
}
```

枚举类型

```
enum TradeDirection {
  CALL = "CALL",  // 买涨
  PUT = "PUT"     // 买跌
}

enum TransactionStatus {
  PENDING = "PENDING",       // 进行中
  SETTLED = "SETTLED",       // 已结算
  CANCELED = "CANCELED"      // 已取消
}

enum VerificationStatus {
  UNVERIFIED = "UNVERIFIED",  // 未验证
  VERIFIED = "VERIFIED"       // 已验证
}
```

错误响应格式

所有错误响应遵循统一格式：

```
{
  "statusCode": 400,
  "timestamp": "2025-10-22T09:00:00.000Z",
  "path": "/api/transactions",
  "message": "错误描述信息",
  "error": "Bad Request"
}
```

常见状态码：

- `400 Bad Request`：请求参数错误
- `401 Unauthorized`：未认证或令牌无效
- `403 Forbidden`：权限不足
- `404 Not Found`：资源不存在
- `500 Internal Server Error`：服务器内部错误

测试用例

完整交易流程示例

```
# 1. 注册用户
curl -X POST http://localhost:3000/api/auth/register \
  -H "Content-Type: application/json" \
  -d '{
    "email": "trader@example.com",
    "password": "password123",
    "displayName": "测试交易员"
  }'

# 2. 登录获取令牌
curl -X POST http://localhost:3000/api/auth/login \
  -H "Content-Type: application/json" \
  -d '{
    "email": "trader@example.com",
    "password": "password123"
  }'

# 3. 创建交易（买涨 BTC，60秒，投入100）
curl -X POST http://localhost:3000/api/transactions \
  -H "Content-Type: application/json" \
  -H "Authorization: Bearer YOUR_ACCESS_TOKEN" \
  -d '{
    "assetType": "BTC",
    "direction": "CALL",
    "duration": 60,
    "investAmount": 100,
    "returnRate": 0.85
  }'
```

```bash
# 4. 查看交易列表
curl -X GET "http://localhost:3000/api/transactions?page=1&limit=10" \
  -H "Authorization: Bearer YOUR_ACCESS_TOKEN"

# 5. 查看统计数据
curl -X GET http://localhost:3000/api/transactions/statistics \
  -H "Authorization: Bearer YOUR_ACCESS_TOKEN"

# 6. 手动结算交易
curl -X POST http://localhost:3000/api/transactions/TXN1729581234567ABC123/settle \
  -H "Authorization: Bearer YOUR_ACCESS_TOKEN"

# 7. 再次查看统计数据（观察余额、盈亏、胜率变化）
curl -X GET http://localhost:3000/api/transactions/statistics \
  -H "Authorization: Bearer YOUR_ACCESS_TOKEN"
```

注意事项

1. **开发模式**: 当前 `AUTH_SKIP_JWT_VERIFICATION=true`，JWT 验证被跳过，生产环境必须设置为 `false`
2. **价格数据**: 当前使用模拟价格，需要集成真实的市场数据源（如 Binance API）
3. **自动结算**: 需要配置定时任务（Cron Job）来自动调用 `/api/transactions/auto-settle` 结算过期交易
4. **数据库**: 使用 PostgreSQL + Prisma ORM，支持可视化管理工具 Prisma Studio (http://localhost:5555)
5. **并发处理**: 交易创建和结算操作需要考虑并发情况，建议添加数据库事务处理
6. **风控**: 建议添加：
    - 单笔交易金额限制
    - 每日交易次数限制
    - 最小持仓时间限制
    - 最大持仓金额限制

技术栈

- **框架**: NestJS
- **数据库**: PostgreSQL
- **ORM**: Prisma
- **认证**: JWT + Passport
- **验证**: class-validator
- **WebSocket**: Socket.io (实时价格推送)
- **缓存**: Redis (可选)

- **队列**: BullMQ (可选)

后续开发建议

1. **实时价格推送**: 通过 WebSocket 推送实时价格和交易状态更新
2. **市场数据集成**: 集成 Binance/CoinGecko API 获取真实市场数据
3. **定时任务**: 实现自动结算过期交易的 Cron Job
4. **风控系统**: 添加交易限额、频率限制等风控规则
5. **通知系统**: 交易结算后发送邮件/短信通知
6. **管理后台**: 添加管理员接口，监控系统运行状态
7. **数据分析**: 添加用户行为分析、盈亏分析等报表功能
8. **安全加固**: 启用 JWT 验证、添加请求签名、IP 白名单等

生成时间: 2025-10-22
版本: v1.0.0