Admin Wallet & Balance Management Guide

Огляд

Ця інструкція описує функціонал управління балансом користувачів та обробку депозитів через адмін-панель.

📋 Зміст

---

1. Crypto Deposit Flow

1.1 Як працює Deposit через Crypto адресу

Автоматична система без ручного approve:

Система працює повністю автоматично через WebSocket та polling механізм:

Крок 1: Користувач ініціює депозит

```
// backend/api/finance/deposit/spot/index.post.ts
POST /api/finance/deposit/spot
{
  "currency": "BTC",
  "chain": "BTC",
  "trx": "transaction_hash_here"
}
```

Що відбувається:

- Створюється transaction запис зі статусом `PENDING`
- `amount: 0` (поки не підтверджено)
- `type: "DEPOSIT"`
- `referenceId: trx` (transaction hash)

Крок 2: Автоматична верифікація (WebSocket)

```
// backend/api/finance/deposit/spot/index.ws.ts
```

Система автоматично:

1. Запускає `startSpotVerificationSchedule()` через WebSocket
2. Кожні **15 секунд** перевіряє транзакцію на біржі
3. Максимум **30 хвилин** очікування
4. Автоматично зупиняє перевірку після підтвердження або таймауту

Крок 3: Підтвердження депозиту

```
// Функція verifyTransaction() автоматично:
1. Отримує список депозитів з біржі (Binance/KuCoin/OKX)
   - exchange.fetchDeposits(currency)
   - або exchange.fetchTransactions()

2. Шукає транзакцію за referenceId (txid)

3. Якщо deposit.status === "ok":
   - Оновлює transaction.status = "COMPLETED"
   - Додає amount та fee
   - Оновлює wallet.balance
   - Відправляє email підтвердження
   - Нараховує welcome bonus (якщо налаштовано)
```

1.2 Важливі моменти

❌ Manual Approve НЕ ПОТРІБЕН:

- Система повністю автоматична
- Адміну не потрібно нічого підтверджувати вручну для депозитів

⚠️ Обмеження часу (можна вимкнути):

```
// Якщо settings.depositExpiration === "true":
- Транзакція має бути в межах ±15 хвилин від створення запиту
- Максимум 45 хвилин з моменту створення на біржі
- Інакше статус стає "TIMEOUT"
```

🔄 Підтримувані біржі:

- Binance
- KuCoin
- OKX
- XT

2. Управління Балансом

2.1 Ручне додавання/зняття коштів

API Endpoint:

```
POST /api/admin/finance/wallet/:id/balance
```

Файл: `backend/api/admin/finance/wallet/[id]/balance.post.ts`

2.2 Параметри запиту

```
{
  "type": "ADD" | "SUBTRACT",  // Додати або зняти
  "amount": 100.50             // Сума
}
```

2.3 Приклади використання

Додати кошти (Credit):

```
POST /api/admin/finance/wallet/wallet-uuid-here/balance
{
  "type": "ADD",
  "amount": 1000
}
```

Зняти кошти (Debit):

```
POST /api/admin/finance/wallet/wallet-uuid-here/balance
{
  "type": "SUBTRACT",
  "amount": 500
}
```

2.4 Що відбувається автоматично

```
// Функція updateWalletBalance():

1. Перевіряє wallet існування
```

```
2. Розраховує новий баланс:
   - ADD: newBalance = balance + amount
   - SUBTRACT: newBalance = balance - amount
3. Перевіряє достатність коштів (якщо SUBTRACT)
4. Оновлює wallet.balance
5. Створює transaction запис:
   - type: "INCOMING_TRANSFER" (ADD) або "OUTGOING_TRANSFER" (SUBTRACT)
   - status: "COMPLETED"
   - metadata.method: "ADMIN"
6. Відправляє email користувачу
```

2.5 Захист від помилок

```
// Автоматичні перевірки:
☑ Wallet існує
☑ User існує
☑ Достатньо коштів (для SUBTRACT)
☑ newBalance >= 0
❌ Якщо помилка -> throw Error
```

---

3. Approve/Reject Withdrawals

3.1 Approve Withdrawal

Файл: `backend/api/admin/finance/wallet/[id]/withdraw/approve.post.ts`

```
POST /api/admin/finance/wallet/:transactionId/withdraw/approve
```

Процес:

1. Перевіряє transaction.status === "PENDING"
2. Виконує withdraw на біржі
3. Оновлює статус транзакції
4. Відправляє email підтвердження

3.2 Reject Withdrawal

Файл: `backend/api/admin/finance/wallet/[id]/withdraw/reject.post.ts`

```
POST /api/admin/finance/wallet/:transactionId/withdraw/reject
{
  "message": "Причина відхилення"
}
```

Процес:

1. Оновлює transaction.status = "REJECTED"
2. Додає metadata.note з причиною
3. Повертає кошти на wallet (REFUND_WITHDRAWAL)
4. Відправляє email з причиною відхилення

---

4. Адмін UI

4.1 Wallets Management

URL: `/admin/finance/wallet`

Файл: `src/pages/admin/finance/wallet/index.tsx`

Функціонал:

- ☑ Перегляд всіх гаманців користувачів
- ☑ Фільтрація по currency, type, status
- ☑ Сортування по balance, inOrder
- ☑ Switch для status (активація/деактивація)
- ❌ canCreate: false (не можна створювати)
- ❌ canView: false (немає детального view)
- ☑ hasAnalytics (аналітика доступна)

Відображувані колонки:

```
- User (firstName + lastName, email, avatar)
- Currency (BTC, ETH, USD тощо)
- Type (FIAT, SPOT, ECO)
- Balance (поточний баланс)
- In Order (заморожені кошти)
- Status (активний/неактивний)
```

Permission: `Access Wallet Management`

4.2 Transactions Management

URL: `/admin/finance/transaction`

Файл: `src/pages/admin/finance/transaction/index.tsx`

Функціонал:

- ☑ Перегляд всіх транзакцій
- ☑ Фільтрація по type, status
- ☑ Сортування по amount, fee, date
- ☑ View деталей транзакції

- ☑ Edit транзакції (якщо PENDING)
- ❌ canCreate: false
- ☑ hasAnalytics

Відображувані колонки:

```
- Transaction ID
- User (з аватаром та email)
- Wallet Currency (з типом)
- Type (DEPOSIT, WITHDRAWAL, TRANSFER тощо)
- Amount (з precision: 8)
- Fee (з precision: 8)
- Status (PENDING, COMPLETED, REJECTED тощо)
- Created At (дата та час)
```

Transaction Types:

```
- DEPOSIT
- WITHDRAWAL
- INCOMING_TRANSFER
- OUTGOING_TRANSFER
- PAYMENT
- REFUND
- BINARY_ORDER
- EXCHANGE_ORDER
- AI_INVESTMENT
- INVESTMENT
- AI_INVESTMENT_ROI
- INVESTMENT_ROI
- COMMISSION
- DIVIDEND
- REFERRAL_REWARD
```

Permission: `Access Transaction Management`

4.3 Як знайти wallet ID для balance update

Спосіб 1: Через UI

1. Відкрити `/admin/finance/wallet`
2. Знайти користувача через пошук
3. В колонці можна побачити walletId (хоча UI не показує напряму)

Спосіб 2: Через Transaction

1. Відкрити `/admin/finance/transaction`
2. Знайти транзакцію користувача
3. В sublabel відображається walletId

Спосіб 3: Через API/DB

```
SELECT id, userId, currency, type, balance
FROM wallet
WHERE userId = 'user-uuid-here' AND currency = 'BTC' AND type = 'SPOT';
```

---

5. Тестування

5.1 Швидке нарахування тестових коштів

Метод 1: Через API (Рекомендовано)

```
# 1. Знайти wallet ID користувача
GET /api/admin/finance/wallet?userId=USER_UUID

# 2. Додати кошти
POST /api/admin/finance/wallet/WALLET_UUID/balance
Authorization: Bearer ADMIN_TOKEN
Content-Type: application/json

{
  "type": "ADD",
  "amount": 10000
}
```

Метод 2: Прямо через SQL (Швидко, але без email/transaction)

```
-- УВАГА: Цей метод не створює transaction запис і не відправляє email!
-- Використовуйте тільки для тестування!

-- Додати 10000 BTC користувачу
UPDATE wallet
SET balance = balance + 10000
WHERE userId = 'user-uuid-here'
  AND currency = 'BTC'
  AND type = 'SPOT';

-- Перевірити баланс
SELECT currency, type, balance, inOrder
FROM wallet
WHERE userId = 'user-uuid-here';
```

Метод 3: Створити тестову транзакцію

```
-- Створити COMPLETED deposit транзакцію
INSERT INTO transaction (
  id, userId, walletId, type, status, amount, fee, description, metadata
) VALUES (
  UUID(),
  'user-uuid-here',
  'wallet-uuid-here',
  'DEPOSIT',
  'COMPLETED',
  5000,
  0,
  'Test deposit for development',
  '{"method": "TEST", "note": "Development test transaction"}'
);

-- Потім оновити баланс гаманця
UPDATE wallet
SET balance = balance + 5000
WHERE id = 'wallet-uuid-here';
```

5.2 Тестування Deposit Flow

Тест 1: Імітація crypto deposit

```
// 1. Створити PENDING deposit
POST /api/finance/deposit/spot
{
  "currency": "BTC",
  "chain": "BTC",
  "trx": "test-transaction-hash-123"
}

// 2. Manually завершити (якщо немає біржі)
// В production: система автоматично verify через WebSocket
UPDATE transaction
SET status = 'COMPLETED', amount = 1.5, fee = 0.0001
WHERE referenceId = 'test-transaction-hash-123';

UPDATE wallet
SET balance = balance + 1.4999
WHERE id = (SELECT walletId FROM transaction WHERE referenceId = 'test-
transaction-hash-123');
```

Тест 2: Перевірка timeout

```
// Створити deposit більше 45 хвилин тому
INSERT INTO transaction (
  id, userId, walletId, type, status, amount,
  referenceId, createdAt, metadata
) VALUES (
  UUID(),
  'user-uuid',
  'wallet-uuid',
  'DEPOSIT',
  'PENDING',
  0,
  'old-transaction-hash',
  DATE_SUB(NOW(), INTERVAL 50 MINUTE),
  '{"currency": "BTC", "chain": "BTC", "trx": "old-transaction-hash"}'
);

-- Verification система автоматично встановить TIMEOUT
```

5.3 Тестові сценарії

Сценарій 1: Повний цикл deposit

```
1. Користувач отримує crypto адресу
2. Надсилає кошти на адресу
3. Вводить transaction hash
4. Система автоматично verify (15 сек інтервал)
5. Після підтвердження - баланс оновлюється
6. Email підтвердження відправляється
```

Сценарій 2: Адмін додає кошти вручну

```
1. Адмін шукає користувача в /admin/finance/wallet
2. Отримує wallet ID
3. POST /api/admin/finance/wallet/:id/balance
4. Користувач отримує email
5. Transaction створюється з type INCOMING_TRANSFER
```

Сценарій 3: Withdrawal approve/reject

```
1. Користувач створює withdrawal request (PENDING)
2. Адмін переглядає в /admin/finance/transaction
3. Адмін approve або reject:
   - Approve: виконується на біржі, статус COMPLETED
```

```
   - Reject: кошти повертаються, статус REJECTED
4. Email відправляється користувачу
```

---

6. API Reference

6.1 Admin Wallet Endpoints

Get All Wallets

```
GET /api/admin/finance/wallet
Query: {
  page?: number
  perPage?: number
  sortField?: string
  sortOrder?: 'asc' | 'desc'
  userId?: string
  currency?: string
  type?: 'FIAT' | 'SPOT' | 'ECO'
}
```

Update Wallet Balance

```
POST /api/admin/finance/wallet/:id/balance
Body: {
  type: 'ADD' | 'SUBTRACT'
  amount: number
}
Permission: "Access Wallet Management"
```

Update Wallet Status

```
PUT /api/admin/finance/wallet/:id/status
Body: {
  status: boolean
}
```

Get Wallet by ID

```
GET /api/admin/finance/wallet/:id
Response: {
  id: string
```

```
  userId: string
  currency: string
  type: string
  balance: number
  inOrder: number
  status: boolean
  user: User
  transactions: Transaction[]
}
```

6.2 Admin Transaction Endpoints

Get All Transactions

```
GET /api/admin/finance/transaction
Query: {
  page?: number
  perPage?: number
  type?: TransactionType
  status?: TransactionStatus
  userId?: string
}
```

Update Transaction Status

```
PUT /api/admin/finance/transaction/:id/status
Body: {
  status: 'PENDING' | 'COMPLETED' | 'REJECTED' | 'FAILED' | 'CANCELLED' |
'EXPIRED' | 'TIMEOUT'
  metadata?: {
    message?: string
  }
}
```

Approve Withdrawal

```
POST /api/admin/finance/wallet/:transactionId/withdraw/approve
Permission: "Access Wallet Management"
```

Reject Withdrawal

```
POST /api/admin/finance/wallet/:transactionId/withdraw/reject
Body: {
  message: string  // Required: причина відхилення
}
Permission: "Access Wallet Management"
```

6.3 User Deposit Endpoints

Initiate Spot Deposit

```
POST /api/finance/deposit/spot
Body: {
  currency: string
  chain: string
  trx: string  // Transaction hash
}
```

Verify Deposit (WebSocket)

```
WS /api/finance/deposit/spot
Message: {
  payload: {
    trx: string
  }
}

// Автоматично:
// - Запускає verification кожні 15 секунд
// - Максимум 30 хвилин
// - Відправляє статус через WebSocket
```

---

7. Database Schema

7.1 Wallet Table

```
CREATE TABLE wallet (
  id VARCHAR(36) PRIMARY KEY,
  userId VARCHAR(36) NOT NULL,
  type ENUM('FIAT', 'SPOT', 'ECO', 'FUTURES', 'FOREX', 'STOCK', 'INDEX') NOT
NULL,
  currency VARCHAR(255) NOT NULL,
  balance DOUBLE NOT NULL DEFAULT 0,
  inOrder DOUBLE DEFAULT 0,
```

```
  address JSON,
  status BOOLEAN NOT NULL DEFAULT TRUE,
  createdAt TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
  updatedAt TIMESTAMP DEFAULT CURRENT_TIMESTAMP ON UPDATE CURRENT_TIMESTAMP,
  deletedAt TIMESTAMP NULL,

  UNIQUE KEY (userId, currency, type),
  FOREIGN KEY (userId) REFERENCES user(id) ON DELETE CASCADE
);
```

7.2 Transaction Table

```
CREATE TABLE transaction (
  id VARCHAR(36) PRIMARY KEY,
  userId VARCHAR(36) NOT NULL,
  walletId VARCHAR(36) NOT NULL,
  type ENUM('DEPOSIT', 'WITHDRAWAL', 'INCOMING_TRANSFER', 'OUTGOING_TRANSFER',
...) NOT NULL,
  status ENUM('PENDING', 'COMPLETED', 'REJECTED', 'FAILED', 'CANCELLED',
'EXPIRED', 'TIMEOUT') DEFAULT 'PENDING',
  amount DOUBLE NOT NULL DEFAULT 0,
  fee DOUBLE DEFAULT 0,
  description TEXT,
  metadata JSON,
  referenceId VARCHAR(255) UNIQUE,
  createdAt TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
  updatedAt TIMESTAMP DEFAULT CURRENT_TIMESTAMP ON UPDATE CURRENT_TIMESTAMP,

  FOREIGN KEY (userId) REFERENCES user(id) ON DELETE CASCADE,
  FOREIGN KEY (walletId) REFERENCES wallet(id) ON DELETE CASCADE,
  INDEX (walletId)
);
```

---

8. Важливі примітки

8.1 Security & Permissions

Потрібні permissions:

- `Access Wallet Management` - для всіх wallet операцій
- `Access Transaction Management` - для перегляду транзакцій

8.2 Email Notifications

Система автоматично відправляє emails:

- ☑ Deposit підтвердження
- ☑ Withdrawal approve/reject

- ☑ Balance update (admin manual)
- ☑ Transaction status changes

8.3 Обробка помилок

```
// Типові помилки:
❌ "Wallet not found"
❌ "User not found"
❌ "Insufficient funds"
❌ "Transaction not found"
❌ "Transaction is not pending"
❌ "Withdrawal failed"
❌ "Invalid deposit currency"
❌ "Deposit expired"
```

8.4 Best Practices

Для адміністраторів:

1. ☑ Завжди перевіряйте user перед додаванням коштів
2. ☑ Використовуйте ADD/SUBTRACT через API (не SQL)
3. ☑ Документуйте причину manual adjustments
4. ☑ Перевіряйте transaction history перед approve withdrawals
5. ⚠ Не редагуйте balance напряму в DB (використовуйте API)

Для розробників:

1. ☑ Завжди створюйте transaction record при зміні балансу
2. ☑ Використовуйте updateWalletBalance() функцію
3. ☑ Перевіряйте достатність коштів перед SUBTRACT
4. ☑ Логуйте всі balance changes
5. ☑ Відправляйте email notifications

---

9. Troubleshooting

Проблема: Deposit не підтверджується автоматично

Перевірити:

1. WebSocket з'єднання активне
2. Exchange credentials правильні
3. Transaction hash правильний
4. Не вийшов timeout (30 хвилин)
5. Deposit з'явився на біржі

Рішення:

```
-- Перевірити статус
SELECT * FROM transaction WHERE referenceId = 'trx-hash';

-- Manually завершити (якщо потрібно)
UPDATE transaction SET status = 'COMPLETED', amount = 1.0 WHERE id =
'transaction-id';
UPDATE wallet SET balance = balance + 1.0 WHERE id = 'wallet-id';
```

Проблема: Balance не оновлюється

Перевірити:

1. Wallet існує
2. User має permission
3. Amount правильний (позитивний)
4. Wallet не заблокований (status = true)

Рішення:

```
// Перевірити wallet
GET /api/admin/finance/wallet/:id

// Спробувати знову
POST /api/admin/finance/wallet/:id/balance
{
  "type": "ADD",
  "amount": 100
}
```

Проблема: Withdrawal не можна approve

Перевірити:

1. Transaction status = 'PENDING'
2. Exchange credentials налаштовані
3. Достатньо коштів на біржі
4. Network/chain правильний

---

10. Швидкий старт для тестування

```
# 1. Знайти користувача
GET /api/admin/crm/user?email=test@example.com

# 2. Знайти wallet
GET /api/admin/finance/wallet?userId=USER_UUID&currency=BTC&type=SPOT
```

```
# 3. Додати тестові кошти
POST /api/admin/finance/wallet/WALLET_UUID/balance
{
  "type": "ADD",
  "amount": 10000
}

# 4. Перевірити баланс
GET /api/admin/finance/wallet/WALLET_UUID

# 5. Переглянути transaction
GET /api/admin/finance/transaction?userId=USER_UUID
```

Контакти та підтримка

Для питань звертайтесь до:

- Backend розробників для API issues
- DevOps для database/infrastructure
- QA для test scenarios

Версія документу: 1.0
Дата: 2026-01-02
Автор: AI Assistant