Clerk Integration Guide - TheShoeBolt

Overview

Tài liệu này mô tả cách tích hợp và sử dụng Clerk authentication service trong TheShoeBolt backend.

Prerequisites

1. **Clerk Account**: Tạo tài khoản tại clerk.com
2. **Clerk Application**: Tạo một application mới trong Clerk dashboard
3. **API Keys**: Lấy Secret Key và Publishable Key từ Clerk dashboard

Environment Configuration

Cập nhật file `.env` với thông tin Clerk của bạn:

```
# Clerk Configuration
CLERK_SECRET_KEY=sk_test_xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
CLERK_PUBLISHABLE_KEY=pk_test_xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
CLERK_WEBHOOK_SECRET=whsec_xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
```

Core Components

1. ClerkModule (`src/modules/clerk/clerk.module.ts`)

Module chính quản lý Clerk SDK initialization:

```
// Sử dụng trong AppModule
ClerkModule.forRootAsync()
```

2. ClerkAuthGuard (`src/modules/auth/guards/clerk-auth.guard.ts`)

Guard để xác thực request bằng Clerk JWT token:

```
@UseGuards(ClerkAuthGuard)
@Get('protected-endpoint')
async protectedEndpoint(@Request() req) {
  // req.user chứa thông tin user từ Clerk
  // req.session chứa thông tin session
  return { user: req.user };
}
```

3. RolesGuard (Updated)

Guard đã được cập nhật để đọc role từ Clerk's `publicMetadata`:

```
@UseGuards(ClerkAuthGuard, RolesGuard)
@Roles(UserRole.ADMIN)
@Get('admin-only')
async adminOnly() {
  return { message: 'Admin access granted' };
}
```

4. ClerkSessionService (`src/modules/clerk/clerk.session.service.ts`)

Service quản lý sessions:

```
// Get user sessions
await clerkSessionService.getSessionList(userId);

// Revoke specific session
await clerkSessionService.revokeSession(sessionId);

// Revoke all user sessions
await clerkSessionService.revokeAllUserSessions(userId);
```

API Endpoints

Authentication Testing

1. Get User Profile

```
GET /auth/profile
Authorization: Bearer <clerk_jwt_token>
```

Response:

```
{
  "message": "Profile retrieved successfully",
  "user": {
    "id": "user_xxxxxxxxx",
    "email": "user@example.com",
    "firstName": "John",
    "lastName": "Doe",
    "publicMetadata": {
      "role": "user"
    }
  },
  "session": {
    "id": "sess_xxxxxxxxx",
    "status": "active"
  }
}
```

2. Admin Only Endpoint

```
GET /auth/admin-only
Authorization: Bearer <admin_clerk_jwt_token>
```

Yêu cầu user có `role: "admin"` trong `publicMetadata`.

Session Management

1. Get Current User Sessions

```
GET /clerk/sessions
Authorization: Bearer <clerk_jwt_token>
```

2. Revoke Specific Session

```
DELETE /clerk/sessions/{sessionId}
Authorization: Bearer <clerk_jwt_token>
```

3. Revoke All Current User Sessions

```
DELETE /clerk/sessions
Authorization: Bearer <clerk_jwt_token>
```

4. Admin: Get Any User Sessions

```
GET /clerk/admin/users/{userId}/sessions
Authorization: Bearer <admin_clerk_jwt_token>
```

5. Admin: Revoke All User Sessions

```
DELETE /clerk/admin/users/{userId}/sessions
Authorization: Bearer <admin_clerk_jwt_token>
```

User Role Management

Setting User Roles in Clerk

Roles được lưu trong Clerk's `publicMetadata`. Để set role cho user:

1. **Via Clerk Dashboard**:

- Vào Users section
- Chọn user cần set role
- Trong Public metadata, thêm:

```
{
  "role": "admin"
}
```

2. **Via Clerk API** (từ backend):

```
await clerkClient.users.updateUser(userId, {
  publicMetadata: {
    role: UserRole.ADMIN
  }
});
```

Available Roles

```
export enum UserRole {
  ADMIN = 'admin',
  USER = 'user',
}
```

Authentication Flow

1. Frontend Integration

Trên frontend, sử dụng Clerk React/Next.js components:

```
import { ClerkProvider, SignIn, SignUp, useAuth } from '@clerk/nextjs';

// Lấy token để gọi API
const { getToken } = useAuth();
const token = await getToken();

// Gọi API với token
fetch('/api/auth/profile', {
  headers: {
    'Authorization': `Bearer ${token}`
  }
});
```

2. Backend Verification

Backend tự động verify token thông qua `ClerkAuthGuard`:

1. Extract token từ `Authorization: Bearer <token>` header
2. Verify token với Clerk SDK
3. Lấy thông tin user và session
4. Attach vào `request.user` và `request.session`

Error Handling

Common Errors

1. **Invalid Token**:

```
{
  "statusCode": 401,
  "message": "Authentication failed: Invalid token"
}
```

2. **Missing Authorization Header**:

```
{
  "statusCode": 401,
  "message": "Missing or invalid authorization header"
}
```

3. **Insufficient Permissions**:

```
{
  "statusCode": 403,
  "message": "Forbidden resource"
}
```

Development Setup

1. Install Dependencies

```
npm install @clerk/clerk-sdk-node
```

2. Setup Clerk Development Instance

1. Tạo development application trong Clerk
2. Enable development mode
3. Configure allowed origins
4. Setup social providers (optional)

3. Test Authentication

Sử dụng Clerk's development mode để test:

```
# Start development server
npm run start:dev

# Test endpoints
curl -H "Authorization: Bearer <dev_token>" http://localhost:3000/auth/profile
```

Production Considerations

1. Environment Variables

Đảm bảo production keys được set đúng:

- `CLERK_SECRET_KEY`: Production secret key
- `CLERK_PUBLISHABLE_KEY`: Production publishable key
- `CLERK_WEBHOOK_SECRET`: Webhook secret cho production

2. CORS Configuration

Cập nhật CORS settings để allow frontend domain:

```
// main.ts
app.enableCors({
  origin: process.env.FRONTEND_URL,
  credentials: true,
});
```

3. Rate Limiting

Implement rate limiting cho authentication endpoints:

```
@UseGuards(ThrottlerGuard)
@Throttle(10, 60) // 10 requests per minute
```

Migration from JWT

Existing Endpoints

Endpoints hiện tại sử dụng `JwtAuthGuard` có thể được migrate dần:

1. Thay `JwtAuthGuard` bằng `ClerkAuthGuard`
2. Update code để đọc user info từ `req.user` (Clerk format)

3. Update role checking logic nếu cần

Dual Authentication Support

Có thể support cả JWT và Clerk temporarily:

```
@UseGuards(JwtAuthGuard || ClerkAuthGuard)
```

Webhook Integration (Optional)

Để sync user data với local database:

```
@Controller('webhooks/clerk')
export class ClerkWebhookController {
  @Post()
  async handleWebhook(@Body() payload: any) {
    // Handle user.created, user.updated, user.deleted events
    // Sync with local User entity if needed
  }
}
```

Troubleshooting

Common Issues

1. **Token Verification Fails**:
    - Check issuer URL trong ClerkAuthGuard
    - Verify environment variables
    - Check network connectivity
2. **Role Check Fails**:
    - Verify `publicMetadata.role` is set in Clerk
    - Check role enum values match
3. **Session Management Issues**:
    - Ensure proper permissions for session operations
    - Check user ID matches session owner

Debug Tips

```
// Enable debug logging
const logger = new Logger('ClerkAuth');
logger.debug('Token:', token);
logger.debug('User:', user);
```

Next Steps

1. **Frontend Integration**: Setup Clerk React/Next.js components
2. **User Sync**: Implement webhook để sync user data
3. **Advanced Features**: Implement MFA, social logins
4. **Monitoring**: Add logging và monitoring cho auth flows