Refresh Token — محاضرة عملية

> **كيفية إدارة جلسات التسجيل الآمنة والمريحة**

جدول المحتويات

مقدمة خفيفة

Token يعني إيه؟

Token = تذكرة دخول مؤقتة بتقول "هذا الشخص عندو صلاحية".

```
التسلسل:
1) أنت → بتسجل دخول (email + password)
2) الـ Server → بيتفحص الـ credentials
3) الـ Server → يرجعلك token (مثل التذكرة)
4) أنت → بتحط الـ token في كل طلب بعدها
5) الـ Server → بيتفحص الـ token (موافق ✅ إيه أنت؟!)
```

مشكلة الـ Token العادي

لو الـ token valid لمدة طويلة (مثلاً سنة):

- ❌ لو اتسرق، الهاكر يستخدمه طول السنة
- ❌ ما فيش طريقة تبطله من غير ما تمسح الـ database

الحل؟ Refresh Token!

الأساسيات اللي لازم تعرفها

1) Access Token vs Refresh Token

الـ Token	الصلاحية	الوقت	الاستخدام
Access Token	كل شيء	15-30 دقيقة	في كل API request
Refresh Token	إعادة الـ access token	أيام/أسابيع	ينتهي access لما الـ

المفهوم:

```
Access Token = تذكرة قصيرة المدى (مثل تذكرة اليوم)
Refresh Token = تذكرة طويلة المدى (مثل الـ membership card)
```

2) العملية كاملة

```
الخطوة 1: التسجيل الأول
┌─────────────────────────────────────┐
│ POST /login                         │
│ {email: "ahmed@example.com",        │
│  password: "123456"}                │
└─────────────────────────────────────┘
          ↓
┌─────────────────────────────────────┐
│ Server Response (200 OK):           │
│ {                                   │
│   access_token: "abc123...",        │
│   refresh_token: "xyz789...",       │
│   expires_in: 900 (15 دقيقة/ثواني) │
│ }                                   │
└─────────────────────────────────────┘

الخطوة 2: الاستخدام (access token صالح)
┌─────────────────────────────────────┐
│ GET /profile                        │
│ Header: Authorization: Bearer abc... │
└─────────────────────────────────────┘
          ↓
┌─────────────────────────────────────┐
│ Server: ✅ Token صحيح               │
│ Response: {name: "Ahmed", ...}      │
└─────────────────────────────────────┘

الخطوة 3: بعد 15 دقيقة (access token انتهى)
┌─────────────────────────────────────┐
│ GET /profile                        │
│ Header: Authorization: Bearer abc... │
└─────────────────────────────────────┘
          ↓
┌─────────────────────────────────────┐
│ Server: ❌ Token انتهى              │
│ Response: 401 Unauthorized          │
└─────────────────────────────────────┘

الخطوة 4: استخدام الـ Refresh Token
┌─────────────────────────────────────┐
│ POST /refresh                       │
│ {refresh_token: "xyz789..."}        │
└─────────────────────────────────────┘
```

```
        ↓
┌─────────────────────────────────────┐
│ Server: ✅ Refresh token صحيح       │
│ Response: {                         │
│   access_token: "new_abc...",       │
│   refresh_token: "new_xyz..."       │
│ }                                   │
└─────────────────────────────────────┘

الخطوة 5: استخدام الـ New Access Token
┌─────────────────────────────────────┐
│ GET /profile                        │
│ Header: Authorization: Bearer new...│
└─────────────────────────────────────┘
        ↓
┌─────────────────────────────────────┐
│ Server: ✅ Token صحيح               │
│ Response: {name: "Ahmed", ...}      │
└─────────────────────────────────────┘
```

3) Token Structure (JWT)

Token عادة بتاعك بيكون JWT (JSON Web Token):

```
Header.Payload.Signature

مثال:
eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9
.eyJzdWIiOiIxMjM0NTY3ODkwIiwibmFtZSI6IkFobWVkIiwiaWF0IjoxNTE2MjM5MDIyfQ
.SflKxwRJSMeKKF2QT4fwpMeJf36POk6yJV_adQssw5c
```

Decoded:

```
// Header (Algorithm)
{"alg": "HS256", "typ": "JWT"}

// Payload (Data)
{"sub": "1234567890", "name": "Ahmed", "iat": 1516239022}

// Signature (Verification)
HMACSHA256(header + payload, secret_key)
```

💻 Refresh Token في Laravel

التطبيق العملي: Authentication System

الخطوة 1: Model + Migration

```
php artisan make:model RefreshToken -m
```

```
// database/migrations/xxxx_create_refresh_tokens_table.php
Schema::create('refresh_tokens', function (Blueprint $table) {
    $table->id();
    $table->foreignId('user_id')->constrained()->onDelete('cascade');
    $table->string('token')->unique();
    $table->timestamp('expires_at');
    $table->timestamps();
});
```

```
// app/Models/RefreshToken.php
namespace App\Models;

class RefreshToken extends Model {
    protected $fillable = ['user_id', 'token', 'expires_at'];
    protected $casts = ['expires_at' => 'datetime'];

    public function user() {
        return $this->belongsTo(User::class);
    }

    public function isExpired(): bool {
        return now()->isAfter($this->expires_at);
    }
}
```

الخطوة 2: Authentication Routes

```
// routes/api.php
use App\Http\Controllers\AuthController;

Route::post('/login', [AuthController::class, 'login']);
Route::post('/refresh', [AuthController::class, 'refresh']);
Route::post('/logout', [AuthController::class, 'logout'])-
>middleware('auth:sanctum');
```

الخطوة 3: Authentication Controller

```
// app/Http/Controllers/AuthController.php
namespace App\Http\Controllers;

use App\Models\User;
```

```php
use App\Models\RefreshToken;
use Illuminate\Http\Request;
use Illuminate\Support\Facades\Hash;

class AuthController extends Controller {
    public function login(Request $request) {
        $validated = $request->validate([
            'email' => 'required|email',
            'password' => 'required|min:6',
        ]);

        $user = User::where('email', $validated['email'])->first();

        if (!$user || !Hash::check($validated['password'], $user->password)) {
            return response()->json([
                'success' => false,
                'message' => 'Invalid credentials',
            ], 401);
        }

        // حط الـــ access token
        $accessToken = $user->createToken('access_token', ['*'], now()->addMinutes(15))->plainTextToken;

        // حط الـــ refresh token
        $refreshToken = new RefreshToken([
            'user_id' => $user->id,
            'token' => bin2hex(random_bytes(32)),
            'expires_at' => now()->addDays(7),
        ]);
        $refreshToken->save();

        return response()->json([
            'success' => true,
            'access_token' => $accessToken,
            'refresh_token' => $refreshToken->token,
            'expires_in' => 900, // ثواني (15 دقيقة)
            'token_type' => 'Bearer',
        ], 200);
    }

    public function refresh(Request $request) {
        $validated = $request->validate([
            'refresh_token' => 'required|string',
        ]);

        // تفحص الـــ refresh token
        $refreshToken = RefreshToken::where('token', $validated['refresh_token'])->first();

        if (!$refreshToken || $refreshToken->isExpired()) {
            return response()->json([
                'success' => false,
                'message' => 'Invalid or expired refresh token',
```

```
            ], 401);
        }

        $user = $refreshToken->user;

        // حط الـ new access token
        $newAccessToken = $user->createToken('access_token', ['*'], now()->addMinutes(15))->plainTextToken;

        // الاختيار: هل تحدّث الـ refresh token أم لا؟
        // ✅ الأفضل: حدّثه (rotation)
        $refreshToken->delete();
        $newRefreshToken = RefreshToken::create([
            'user_id' => $user->id,
            'token' => bin2hex(random_bytes(32)),
            'expires_at' => now()->addDays(7),
        ]);

        return response()->json([
            'success' => true,
            'access_token' => $newAccessToken,
            'refresh_token' => $newRefreshToken->token,
            'expires_in' => 900,
            'token_type' => 'Bearer',
        ], 200);
    }

    public function logout(Request $request) {
        // حذف الـ access token
        $request->user()->tokens()->delete();

        // حذف الـ refresh token (optional)
        RefreshToken::where('user_id', $request->user()->id)->delete();

        return response()->json([
            'success' => true,
            'message' => 'Logged out successfully',
        ]);
    }
}
```

الخطوة 4: Protected Routes

```
// في routes/api.php
Route::middleware('auth:sanctum')->group(function () {
    Route::get('/profile', fn(Request $r) => $r->user());
    Route::post('/logout', [AuthController::class, 'logout']);
});
```

الخطوة 5: أمثلة CURL

```
# 1 التسجيل الأول
curl -X POST http://localhost:8000/api/login \
  -H "Content-Type: application/json" \
  -d '{
    "email": "ahmed@example.com",
    "password": "password123"
  }'

# Response:
# {
#   "success": true,
#   "access_token": "abc123def456...",
#   "refresh_token": "xyz789uvw012...",
#   "expires_in": 900,
#   "token_type": "Bearer"
# }

# 2 استخدام الـ access token
curl http://localhost:8000/api/profile \
  -H "Authorization: Bearer abc123def456..."

# Response:
# {"id": 1, "name": "Ahmed", "email": "ahmed@example.com", ...}

# 3 عمل refresh بعد انتهاء الـ access token
curl -X POST http://localhost:8000/api/refresh \
  -H "Content-Type: application/json" \
  -d '{
    "refresh_token": "xyz789uvw012..."
  }'

# Response:
# {
#   "success": true,
#   "access_token": "new_abc123...",
#   "refresh_token": "new_xyz789...",
#   "expires_in": 900
# }

# 4 التسجيل الخروج
curl -X POST http://localhost:8000/api/logout \
  -H "Authorization: Bearer abc123def456..."

# Response:
# {"success": true, "message": "Logged out successfully"}
```

⚠ الأخطاء الشائعة

1) Refresh Token بصلاحية طويلة جداً

```
// ❌ سنة واحدة؟ كتير جداً!
$refreshToken->expires_at = now()->addYear();

// ✅ أيام معقولة
$refreshToken->expires_at = now()->addDays(7);
```

2) عدم عمل Token Rotation

```
// ❌ الـ refresh token ما يتغيّر، لو اتسرق يفضل صالح للأبد
public function refresh(Request $request) {
    // ... استخدم نفس الـ refresh token
}

// ✅ حدّث الـ refresh token في كل مرة
public function refresh(Request $request) {
    $oldToken->delete();
    $newToken = RefreshToken::create([...]);
}
```

3) تخزين الـ Tokens في الـ Session

```
// ❌ الـ session بتتحذف، token ينحذف معها
session(['access_token' => $token]);

// ✅ احفظها في الـ localStorage من الـ frontend
// الـ backend بس يتفحصها
```

4) نسيان حذف القديم

```
// ❌ تراكم الـ tokens
public function refresh() {
    $newToken = RefreshToken::create([...]);
    // ما حذفت الـ old token
}

// ✅ نظّف
$refreshToken->delete();
$newToken = RefreshToken::create([...]);
```

👔 Senior Corner

1) Token Revocation (الإبطال)

```
// لما user يغيّر كلمة السر، بطّل كل الـ tokens
public function changePassword(Request $request) {
    $user = $request->user();

    // تحديث كلمة السر
    $user->update([
        'password' => Hash::make($request->new_password)
    ]);

    // حذف كل الـ tokens (force logout من كل الأجهزة)
    $user->tokens()->delete();
    RefreshToken::where('user_id', $user->id)->delete();

    return response()->json(['message' => 'Password changed, please login
again']);
}
```

2) Sliding Window (Expiration Extension)

```
// لكل استخدام للـ refresh token، مدّد صلاحيته
public function refresh(Request $request) {
    $refreshToken = RefreshToken::where('token', $request->refresh_token)-
>first();

    // بدل ما تحذفه، حدّث الـ expiration
    $refreshToken->update([
        'expires_at' => now()->addDays(7)
    ]);

    $newAccessToken = $refreshToken->user->createToken(...)->plainTextToken;

    return response()->json([
        'access_token' => $newAccessToken,
        'refresh_token' => $refreshToken->token, // نفس الـ token
    ]);
}
```

3) Rate Limiting على الـ Refresh

```
// In routes/api.php
Route::post('/refresh', [AuthController::class, 'refresh'])
    ->middleware('throttle:5,1'); // 5 requests في الدقيقة

// في الـ controller
public function refresh(Request $request) {
    // لو الـ user جاول يعمل refresh كتير، يتبلك
}
```

4) Device Tracking

```php
// database/migrations/xxxx_add_device_to_refresh_tokens.php
Schema::table('refresh_tokens', function (Blueprint $table) {
    $table->string('device_name')->nullable(); // "iPhone", "Chrome", etc
    $table->string('ip_address')->nullable();
});

// في الـــ login
$refreshToken = RefreshToken::create([
    'user_id' => $user->id,
    'token' => bin2hex(random_bytes(32)),
    'device_name' => $request->header('User-Agent'),
    'ip_address' => $request->ip(),
    'expires_at' => now()->addDays(7),
]);

// User يقدر يشوف الأجهزة اللي محتة على حسابه
Route::middleware('auth:sanctum')->get('/devices', function (Request $r) {
    return RefreshToken::where('user_id', $r->user()->id)->get(['device_name',
'created_at']);
});
```

5) Testing Refresh Token

```php
// tests/Feature/AuthTest.php

public function test_refresh_token_renews_access_token() {
    $user = User::factory()->create();

    $response = $this->post('/api/login', [
        'email' => $user->email,
        'password' => 'password',
    ]);

    $refreshToken = $response['refresh_token'];

    $this->post('/api/refresh', ['refresh_token' => $refreshToken])
        ->assertStatus(200)
        ->assertJsonStructure(['access_token', 'refresh_token']);
}

public function test_expired_refresh_token_fails() {
    $user = User::factory()->create();
    RefreshToken::factory()->create([
        'user_id' => $user->id,
        'expires_at' => now()->subDay(),
    ]);

    $this->post('/api/refresh', [
```

```
            'refresh_token' => 'expired_token'
        ])->assertStatus(401);
    }
```

🎓 خاتمة + تحدي

تحديك ليك:

اعمل feature: **logout من جميع الأجهزة**

```
DELETE /api/logout-all
من كل الأجهزة tokens لما يضغط عليها، يحذف كل الـ
```

الحل:

```
public function logoutAll(Request $request) {
    $user = $request->user();

    // حذف كل الـ access tokens
    $user->tokens()->delete();

    // حذف كل الـ refresh tokens
    RefreshToken::where('user_id', $user->id)->delete();

    return response()->json([
        'success' => true,
        'message' => 'Logged out from all devices'
    ]);
}

// في routes
Route::middleware('auth:sanctum')->delete('/logout-all', [AuthController::class,
'logoutAll']);
```

Checklist ليك:

- ✅ فهمت الفرق بين Access و Refresh Token
- ✅ عملت نظام Login/Refresh/Logout كامل
- ✅ عرفت ليه Token Rotation مهم
- ✅ عملت Protected Routes
- ✅ عملت Device Tracking (optional)
- ✅ عملت Rate Limiting

📋 Cheat Sheet

Token Endpoints

```
POST /login          → اجلب access + refresh tokens
POST /refresh        → جدّد الـــ access token
POST /logout         → احذف tokens حالـيين
DELETE /logout-all   → احذف tokens من كل الأجهزة
```

Token Lifetimes

```
Access Token: 15-30 دقـيقـة
Refresh Token: 7 أيـام - شهر واحد
Session: عادة مـا يـستخدم فـي APIs
```

Best Practices

الممارسة	ليه؟
Access token قصير المدى	Security
Refresh token طويل المدى	Convenience
Token rotation على الـ refresh	Prevent token theft
Store refresh في الـ database	Track active sessions
HTTPS فقط	Prevent man-in-the-middle
HttpOnly cookie للـ tokens	Prevent XSS
CORS properly configured	Prevent CSRF

النهاية 🚀

الفكرة الأساسية:

> Short-lived access tokens + long-lived refresh tokens = أمان + سهولة.

يلا.. طبّق! 💪

□□ ♡ □□ □□□□□□ *Backend Instructor*