دليل منصة ZEREX CARBON الشامل

سوق الكربون الأوروبي الطوعي

تاريخ الإنشاء: ٢٠٢٥/٩/٢٤
الإصدار: 1.0.0
اللغة: العربية

مقدمة

هذا الدليل الشامل يغطي جميع جوانب منصة ZEREX CARBON، من الواجهة الأمامية إلى الخادم الخلفي، ومن تطبيق الهاتف المحمول إلى النشر والإنتاج. يهدف هذا الدليل إلى توفير فهم عميق لجميع مكونات النظام وكيفية عملها معاً.

دليل منصة ZEREX CARBON - الجزء الأول

سوق الكربون الأوروبي الطوعي

فهرس المحتويات

الجزء الأول: نظرة عامة على المشروع

1. [مقدمة عن المنصة](#مقدمة-عن-المنصة)
2. [الميزات الرئيسية](#الميزات-الرئيسية)
3. [التقنيات المستخدمة](#التقنيات-المستخدمة)
4. [هيكل المشروع](#هيكل-المشروع)

الجزء الثاني: الواجهة الأمامية (Frontend)

5. [ملفات HTML الأساسية](#ملفات-html-الأساسية)
6. [ملفات JavaScript](#ملفات-javascript)
7. [التصميم والواجهات](#التصميم-والواجهات)

الجزء الثالث: الخادم الخلفي (Backend)

8. [خادم Express.js](#خادم-expressjs)
9. [المتحكمات (Controllers)](#المتحكمات-controllers)
10. [الخدمات (Services)](#الخدمات-services)
11. [المسارات (Routes)](#المسارات-routes)

الجزء الرابع: قاعدة البيانات

12. [مخطط قاعدة البيانات](#مخطط-قاعدة-البيانات)
13. [النماذج والعلاقات](#النماذج-والعلاقات)

الجزء الخامس: تطبيق الهاتف المحمول

14. [تطبيق Flutter](#تطبيق-flutter)
15. [الهيكل والميزات](#الهيكل-والميزات)

الجزء السادس: النشر والإنتاج

16. [النشر والإعداد](#النشر-والإعداد)
17. [الأمان والمراقبة](#الأمان-والمراقبة)

مقدمة عن المنصة

ما هي ZEREX CARBON؟

ZEREX CARBON هي منصة تجارية شاملة وجاهزة للإنتاج لسوق الكربون الأوروبي الطوعي. تمكن هذه المنصة المستخدمين من شراء وبيع وتداول أرصدة الكربون مع دعم مشاريع تعويض الكربون المعتمدة.

الهدف من المنصة

- **الشفافية**: معلومات مفصلة لجميع مشاريع الكربون (الموقع، التأثير، الشهادات)
- **التكامل مع APIs**: يمكن للأنظمة الخارجية الاستعلام عن المشاريع والأرصدة والمعاملات
- **التحقق المستقل**: جميع أرصدة الكربون يتم التحقق منها بشكل مستقل
- **دعم المجتمعات المحلية**: تتبع المشاريع التي تفيد السكان المحليين
- **التسعير المرن**: أسعار قابلة للتعديل لكل مشروع ونوع ائتمان
- **التكنولوجيا المتقدمة**: تتبع ذكي/قمر صناعي للتحقق من الكربون (محاكاة)
- **تقييم المشاريع**: تقييمات المستخدمين وتتبع التقدم

- **لوحات المعلومات المرئية**: تتبع التأثير والتقارير
- **التقارير الواضحة**: شفافية البيانات البيئية والمالية

الميزات الرئيسية

ميزات المنصة الأساسية

لوحة تحكم الإدارة

- **نظرة عامة على لوحة التحكم**: مقاييس في الوقت الفعلي للمستخدمين والمشاريع والمعاملات والإيرادات
- **الإشعارات**: تنبيهات فورية للرسائل الجديدة والودائع والمشاريع أو الأخطاء
- **صندوق الرسائل**: التواصل المباشر مع المستخدمين/المشاريع
- **لوحة الإيرادات**: رسوم بيانية للأرباح اليومية والأسبوعية والشهرية
- **التحكم في الموقع**: إدارة المستخدمين والمشاريع والمعاملات والتسعير والإعدادات
- **الوصول القائم على الأدوار**: صلاحيات Super Admin / Moderator / Viewer
- **التنبيهات**: إشعارات المعاملات الفاشلة أو عالية القيمة
- **البحث السريع**: وظائف البحث للمستخدمين والمشاريع
- **تصدير التقارير**: قدرات تصدير CSV و PDF
- **إدارة المشاريع**: الموافقة/رفض المشاريع وإصدار ائتمانات الكربون

لوحة تحكم المستخدم

- **عرض المحفظة/الرصيد**: تاريخ المعاملات وإدارة الرصيد
- **شراء/بيع أرصدة الكربون**: التسعير الديناميكي واختيار المشروع
- **تتبع المشاريع**: التأثير والتقدم والشهادات المكتسبة
- **الإشعارات**: تحديثات المشاريع وتنبيهات المعاملات
- **التقارير والشهادات**: تحميل تقارير PDF/CSV
- **إعدادات الحساب**: إدارة الملف الشخصي وكلمة المرور وطريقة الدفع

التقنيات المستخدمة

الخادم الخلفي (Backend)

- **Node.js* + *TypeScript**
- **Express.js** - إطار عمل الويب
- **PostgreSQL** - قاعدة البيانات (مدارة عبر Supabase)
- **Prisma ORM** - نمذجة قاعدة البيانات والاستعلام
- **JWT + Refresh Tokens** - المصادقة
- **bcrypt** - تشفير كلمات المرور
- **Stripe** - معالجة المدفوعات (sandbox)
- **Winston** - التسجيل
- **Jest + Supertest** - الاختبار

الواجهة الأمامية (Frontend)

- **HTML5/CSS3/JavaScript** - التقنيات الأساسية للويب
- **Tailwind CSS** - إطار عمل CSS utility-first
- **Chart.js** - تصور البيانات
- **Font Awesome** - الأيقونات

الأمان والمراقبة

- **Helmet** - رؤوس الأمان
- **Rate Limiting** - تقييد الطلبات
- **SQL Injection Protection** - تنظيف المدخلات
- **XSS Protection** - منع هجمات XSS
- **Performance Monitoring** - مقاييس في الوقت الفعلي

- **Audit Logging** - مسار تدقيق كامل

تطبيق الهاتف المحمول

- **Flutter 3.10+** - إطار عمل متعدد المنصات
- **Dart 3.0+** - لغة البرمجة
- **BLoC Pattern** - إدارة الحالة
- **SQLite** - قاعدة البيانات المحلية مع التشفير
- **Firebase** - الإشعارات والتحليلات
- **OAuth 2.0** - المصادقة

هيكل المشروع

```
/carbon-marketplace
├── /prisma
│   └── schema.prisma # مخطط قاعدة البيانات
├── /src
│   ├── /controllers # متحكمات المسارات
│   │   ├── adminController.ts
│   │   ├── authController.ts
│   │   ├── messageController.ts
│   │   ├── notificationController.ts
│   │   ├── paymentController.ts
│   │   └── transactionController.ts
│   ├── /middleware # Express middleware
│   │   ├── auth.ts
│   │   ├── errorHandler.ts
│   │   ├── monitoring.ts
│   │   ├── performance.ts
│   │   ├── security.ts
│   │   └── validation.ts
│   ├── /routes # مسارات API
```

```
│   ├── admin.ts
│   ├── auth.ts
│   ├── market.ts
│   ├── messages.ts
│   ├── notifications.ts
│   ├── payments.ts
│   ├── transactions.ts
│   └── users.ts
├── services/ # منطق الأعمال
│   ├── monitoringService.ts
│   └── transactionService.ts
├── types/ # أنواع TypeScript
│   └── index.ts
├── utils/ # الأدوات المساعدة
│   ├── advanced-logger.ts
│   └── logger.ts
├── views/ # عروض الواجهة الأمامية
│   ├── admin/
│   │   ├── dashboard.html
│   │   ├── enhanced-dashboard.html
│   │   └── monitoring-dashboard.html
│   └── user/
│       └── enhanced-dashboard.html
└── server.ts # ملف الخادم الرئيسي
├── tests/ # ملفات الاختبار
│   ├── admin.test.ts
│   ├── auth.test.ts
│   ├── integration.test.ts
│   ├── market.test.ts
│   ├── notifications.test.ts
│   ├── performance.test.ts
│   ├── setup.ts
│   ├── testUtils.ts
│   └── transactions.test.ts
├── public/ # الملفات الثابتة
│   ├── admin/
│   ├── auth/
│   ├── js/
│   ├── downloads/
│   └── *.html
```

```
├── /mobile-app # تطبيق الهاتف المحمول
│   ├── /lib
│   ├── /android
│   ├── /ios
│   └── pubspec.yaml
├── .eslintrc.js # إعدادات ESLint
├── .env.example # قالب متغيرات البيئة
├── jest.config.js # إعدادات Jest
├── package.json # التبعيات والبرامج النصية
├── tsconfig.json # إعدادات TypeScript
└── README.md # هذا الملف
```

يتبع في الجزء الثاني: الواجهة الأمامية والكود المصدري

دليل منصة ZEREX CARBON - الجزء الثاني

الواجهة الأمامية والكود المصدري

الواجهة الأمامية (Frontend)

الملف الرئيسي: index.html

```
html `
```

ZEREX CARBON

Europe's fastest way to buy, sell and retire voluntary & compliance carbon credits. Live EUA & .VER prices, one-click KYC, PDF certificates

Get Started

Live Prices

Featured Projects

`

ملف JavaScript الرئيسي: main.js

```
javascript
// main.js - الملف الرئيسي للواجهة الأمامية
import { AuthService } from './auth-service.js';
import { MarketService } from './market-service.js';
import { NotificationService } from './notification-service.js';
import { ThemeService } from './theme-service.js';
import { ResponsiveService } from './responsive-service.js';

class ZerexCarbonApp {
    constructor() {
        this.authService = new AuthService();
        this.marketService = new MarketService();
        this.notificationService = new NotificationService();
        this.themeService = new ThemeService();
        this.responsiveService = new ResponsiveService();

        this.init();
    }

    async init() {
```

```
} try
// تهيئة الخدمات
await this.initializeServices();

// تحميل البيانات الأولية
await this.loadInitialData();

// إعداد المستمعين للأحداث
this.setupEventListeners();

// بدء المراقبة في الوقت الفعلي
this.startRealTimeMonitoring();

console.log('ZEREX CARBON App initialized successfully');
} catch (error) {
console.error('Failed to initialize app:', error);
this.showErrorMessage('فشل في تحميل التطبيق. يرجى تحديث الصفحة.');
{
{

async initializeServices() {
// تهيئة خدمة المصادقة
await this.authService.initialize();

// تهيئة خدمة السوق
await this.marketService.initialize();

// تهيئة خدمة الإشعارات
await this.notificationService.initialize();

// تطبيق الإعدادات المحفوظة
this.themeService.applySavedSettings();
this.responsiveService.setupResponsiveHandlers();
{

async loadInitialData() {
// تحميل أسعار الكربون الحية
await this.loadLivePrices();

// تحميل المشاريع المميزة
```

```
;()await this.loadFeaturedProjects

// تحميل إحصائيات السوق
;()await this.loadMarketStats
{

} ()async loadLivePrices
} try
;()const prices = await this.marketService.getLivePrices
;this.renderPriceTicker(prices)
;this.renderPriceChart(prices)
} catch (error) {
;console.error('Failed to load live prices:', error)
;this.showErrorMessage('فشل في تحميل الأسعار الحية')
{
{

} ()async loadFeaturedProjects
} try
;()const projects = await this.marketService.getFeaturedProjects
;this.renderFeaturedProjects(projects)
} catch (error) {
;console.error('Failed to load featured projects:', error)
;this.showErrorMessage('فشل في تحميل المشاريع المميزة')
{
{

} ()async loadMarketStats
} try
;()const stats = await this.marketService.getMarketStats
;this.renderMarketStats(stats)
} catch (error) {
;console.error('Failed to load market stats:', error)
{
{

} renderPriceTicker(prices)
;const ticker = document.getElementById('ticker')
;if (!ticker) return
```

```
` <= ticker.innerHTML = prices.map(price
```

{price.symbol}$

{price.currency}$ {price.current}$

%{price.change}${'' : '+' ? price.change >= 0}$

```
;('').join(`
{

} renderPriceChart(prices)
;const chartContainer = document.getElementById('chart')
;if (!chartContainer) return

// استخدام Chart.js لعرض الرسوم البيانية
;const ctx = document.createElement('canvas')
;chartContainer.appendChild(ctx)

} ,new Chart(ctx
,'type: 'line
} :data
,labels: prices.map(p => p.symbol)
}] :datasets
,'label: 'Carbon Credit Prices
,data: prices.map(p => p.current)
,'borderColor: '#00F2A9
,'backgroundColor: 'rgba(0, 242, 169, 0.1)
tension: 0.4
[{
,{
} :options
,responsive: true
,maintainAspectRatio: false
} :plugins
} :legend
display: false
```

```
{
,{
} :scales
} :y
,beginAtZero: false
} :ticks
'color: '#fff
,{
} :grid
'color: 'rgba(255, 255, 255, 0.1)
{
,{
} :x
} :ticks
'color: '#fff
,{
} :grid
'color: 'rgba(255, 255, 255, 0.1)
{
{
{
{
;({
{

} renderFeaturedProjects(projects)
;const container = document.getElementById('projects')
;if (!container) return

` <= container.innerHTML = projects.map(project
```

{project.name}$

{project.type}$

{project.description}$

السعر:
${project.pricePerCredit} ${project.currency}

الائتمانات المتاحة:
${project.availableCredits.toLocaleString()}

البلد:
${project.country}

عرض التفاصيل
تداول

`).join('');
{

} renderMarketStats(stats)
// عرض إحصائيات السوق
console.log('Market stats:', stats);
{

} setupEventListeners()
// مستمعون للأحداث العامة
window.addEventListener('online', () => {
this.notificationService.show('تم استعادة الاتصال بالإنترنت', 'success');
this.loadInitialData();
});

window.addEventListener('offline', () => {
this.notificationService.show('تم فقدان الاتصال بالإنترنت', 'warning');
});

// مستمعون للتفاعل مع المشاريع

```
window.app = this; // للوصول من HTML
{

} ()startRealTimeMonitoring
// بدء مراقبة الأسعار في الوقت الفعلي
} <= () setInterval(async
} try
;()await this.loadLivePrices
} catch (error) {
;console.error('Failed to update live prices:', error)
{
}, 30000); // كل 30 ثانية

// بدء مراقبة الإشعارات
;()this.notificationService.startRealTimeMonitoring
{

} viewProject(projectId)
// عرض تفاصيل المشروع
;window.location.href = `/project.html?id=${projectId}`
{

} tradeProject(projectId)
// فتح واجهة التداول
} if (!this.authService.isAuthenticated())
;('يجب تسجيل الدخول أولاً', 'warning')this.notificationService.show
;'window.location.href = '/auth.html
;return
{

;window.location.href = `/trade.html?project=${projectId}`
{

} showErrorMessage(message)
// عرض رسالة خطأ للمستخدم
;const errorDiv = document.createElement('div')
;'errorDiv.className = 'error-message
;errorDiv.textContent = message
` = errorDiv.style.cssText
;position: fixed
```

```
;top: 20px
;right: 20px
;background: #ef4444
;color: white
;padding: 1rem
;border-radius: 0.5rem
;z-index: 1000
;max-width: 300px
;`

;document.body.appendChild(errorDiv)

} <= ())setTimeout
;()errorDiv.remove
;(5000 ,{
{
{

// تهيئة التطبيق عند تحميل الصفحة
} <= () ,'document.addEventListener('DOMContentLoaded
;()new ZerexCarbonApp
;{}

// تصدير للاستخدام في ملفات أخرى
;export { ZerexCarbonApp }
`
```

خدمة المصادقة: auth-service.js

```
` javascript
// auth-service.js - خدمة المصادقة
} export class AuthService
} ()constructor
;this.token = localStorage.getItem('accessToken')
;this.refreshToken = localStorage.getItem('refreshToken')
;this.user = null
;this.isInitialized = false
```

```
{

} ()async initialize
} try
} if (this.token)
;()await this.validateToken
;()await this.loadUserProfile
{
;this.isInitialized = true
} catch (error) {
;console.error('Auth initialization failed:', error)
;()this.clearTokens
{
{

} ()async validateToken
} try
} ,'const response = await fetch('/api/auth/validate
,'method: 'GET
} :headers
,Authorization': `Bearer ${this.token}` '
'Content-Type': 'application/json'
{
;({

} if (!response.ok)
;throw new Error('Token validation failed')
{

;return true
} catch (error) {
// محاولة تجديد الرمز المميز
;()return await this.refreshAccessToken
{
{

} ()async refreshAccessToken
} try
} ,'const response = await fetch('/api/auth/refresh
,'method: 'POST
```

```
} :headers
'Content-Type': 'application/json'
,{
})body: JSON.stringify
refreshToken: this.refreshToken
({
;({

} if (!response.ok)
;throw new Error('Token refresh failed')
{

;()const data = await response.json
;this.token = data.tokens.accessToken
;localStorage.setItem('accessToken', this.token)

;return true
} catch (error) {
;console.error('Token refresh failed:', error)
;()this.clearTokens
;return false
{
{

} ()async loadUserProfile
} try
} ,'const response = await fetch('/api/users/me
} :headers
,Authorization': `Bearer ${this.token}` '
'Content-Type': 'application/json'
{
;({

} if (!response.ok)
;throw new Error('Failed to load user profile')
{

;()const data = await response.json
;this.user = data.user
;return this.user
```

```
    } catch (error) {
        console.error('Failed to load user profile:', error);
        return null;
    }
}

    } async login(email, password)
        } try
            } ,'const response = await fetch('/api/auth/login
                ,'method: 'POST
                } :headers
                    'Content-Type': 'application/json'
                ,{
                body: JSON.stringify({ email, password })
            });{

            const data = await response.json();()

            } if (!response.ok)
                throw new Error(data.error || 'Login failed');
            {

            this.token = data.tokens.accessToken;
            this.refreshToken = data.tokens.refreshToken;
            this.user = data.user;

            localStorage.setItem('accessToken', this.token);
            localStorage.setItem('refreshToken', this.refreshToken);

            return { success: true, user: this.user };
        } catch (error) {
            console.error('Login error:', error);
            return { success: false, error: error.message };
        }
    }

    } async register(userData)
        } try
            } ,'const response = await fetch('/api/auth/register
                ,'method: 'POST
```

```
} :headers
'Content-Type': 'application/json'
,{
body: JSON.stringify(userData)
;({

;()const data = await response.json

} if (!response.ok)
;throw new Error(data.error || 'Registration failed')
{

;this.token = data.tokens.accessToken
;this.refreshToken = data.tokens.refreshToken
;this.user = data.user

;localStorage.setItem('accessToken', this.token)
;localStorage.setItem('refreshToken', this.refreshToken)

;return { success: true, user: this.user }
} catch (error) {
;console.error('Registration error:', error)
;return { success: false, error: error.message }
{
{

} ()async logout
} try
} if (this.refreshToken)
} ,'await fetch('/api/auth/logout
,'method: 'POST
} :headers
'Content-Type': 'application/json'
,{
body: JSON.stringify({ refreshToken: this.refreshToken })
;({
{
} catch (error) {
;console.error('Logout error:', error)
} finally {
```

```
;()this.clearTokens
{
{

} ()clearTokens
;this.token = null
;this.refreshToken = null
;this.user = null
;localStorage.removeItem('accessToken')
;localStorage.removeItem('refreshToken')
{

} ()isAuthenticated
;return !!this.token && !!this.user
{

} ()getUser
;return this.user
{

} ()getToken
;return this.token
{
{
```

خدمة السوق: market-service.js

```
javascript `
// market-service.js - خدمة السوق
} export class MarketService
} ()constructor
;'this.baseUrl = '/api/market
;this.wsConnection = null
;[] = this.priceCallbacks
;this.isInitialized = false
{
```

```
} ()async initialize
} try
;()await this.connectWebSocket
;this.isInitialized = true
} catch (error) {
;console.error('Market service initialization failed:', error)
{
{

} ()async connectWebSocket
} <= return new Promise((resolve, reject)
} try
;':const protocol = window.location.protocol === 'https:' ? 'wss:' : 'ws
;const wsUrl = `${protocol}//${window.location.host}/ws/market`

;this.wsConnection = new WebSocket(wsUrl)

} <= () = this.wsConnection.onopen
;console.log('WebSocket connected to market service')
;()resolve
;{

} <= this.wsConnection.onmessage = (event)
} try
;const data = JSON.parse(event.data)
;this.handleWebSocketMessage(data)
} catch (error) {
;console.error('Failed to parse WebSocket message:', error)
{
;{

} <= () = this.wsConnection.onclose
;console.log('WebSocket connection closed')
// محاولة إعادة الاتصال بعد 5 ثوان
} <= ()setTimeout
;()this.connectWebSocket
;(5000 ,{
;{
```

```
} <= this.wsConnection.onerror = (error)
;console.error('WebSocket error:', error)
;reject(error)
;{
} catch (error) {
;reject(error)
{
;({
{

} handleWebSocketMessage(data)
} switch (data.type)
:'case 'PRICE_UPDATE
;this.priceCallbacks.forEach(callback => callback(data.payload))
;break
:'case 'MARKET_STATS
// تحديث إحصائيات السوق
;break
:default
;console.log('Unknown WebSocket message type:', data.type)
{
{

} onPriceUpdate(callback)
;this.priceCallbacks.push(callback)
{

} ()async getLivePrices
} try
;const response = await fetch( `${this.baseUrl}/prices` )
} if (!response.ok)
;throw new Error('Failed to fetch live prices')
{
;()return await response.json
} catch (error) {
;console.error('Error fetching live prices:', error)
;throw error
{
{
```

```
} ()async getFeaturedProjects
} try
;const response = await fetch( `${this.baseUrl}/projects/featured` )
} if (!response.ok)
;throw new Error('Failed to fetch featured projects')
{
;()return await response.json
} catch (error) {
;console.error('Error fetching featured projects:', error)
;throw error
{
{

} ()async getMarketStats
} try
;const response = await fetch( `${this.baseUrl}/stats` )
} if (!response.ok)
;throw new Error('Failed to fetch market stats')
{
;()return await response.json
} catch (error) {
;console.error('Error fetching market stats:', error)
;throw error
{
{

} async getProjectDetails(projectId)
} try
;const response = await fetch( `${this.baseUrl}/projects/${projectId}` )
} if (!response.ok)
;throw new Error('Failed to fetch project details')
{
;()return await response.json
} catch (error) {
;console.error('Error fetching project details:', error)
;throw error
{
{

} async tradeCarbonCredits(projectId, amount, type = 'BUY')
```

```
} try
} ,const response = await fetch( ${this.baseUrl}/trade
,'method: 'POST
} :headers
,'Content-Type': 'application/json'
Authorization': Bearer ${this.getAuthToken()} '
,{
})body: JSON.stringify
,projectId
,amount
type
({
;({

;()const data = await response.json

} if (!response.ok)
;throw new Error(data.error || 'Trade failed')
{

;return data
} catch (error) {
;console.error('Error trading carbon credits:', error)
;throw error
{
{

} ()getAuthToken
;return localStorage.getItem('accessToken')
{
{
`
```

يتبع في الجزء الثالث: الخادم الخلفي والكود المصدري

دليل منصة ZEREX CARBON - الجزء الثالث

الخادم الخلفي والكود المصدري

الخادم الخلفي (Backend)

ملف الخادم الرئيسي: server.ts

```typescript
import express, { NextFunction } from 'express';
import { createServer } from 'http';
import cors from 'cors';
import dotenv from 'dotenv';

import logger from '@/utils/logger';
import {
  helmetConfig,
  generalRateLimit,
  authRateLimit,
  sensitiveRateLimit,
  securityHeaders,
  requestLogger,
  sqlInjectionProtection,
  xssProtection,
  requestSizeLimit
} from '@/middleware/security';
```

```
} import
,performanceMonitor
,memoryMonitor
,rateMonitor
,errorRateMonitor
healthCheck
;'from '@/middleware/performance {
;'import { cleanupExpiredRequests } from '@/middleware/idempotency
;'import MonitoringService from '@/services/monitoringService
;'import RealtimeNotificationService from '@/services/realtimeNotificationService
;'import redisService from '@/services/redisService
;'import healthCheckService from '@/services/healthCheckService
;'import circuitBreakerService from '@/services/circuitBreakerService

// استيراد المسارات
;'import authRoutes from '@/routes/auth
;'import adminRoutes from '@/routes/admin
;'import transactionRoutes from '@/routes/transactions
;'import userRoutes from '@/routes/users
;'import marketRoutes from '@/routes/market
;'import notificationRoutes from '@/routes/notifications
;'import messageRoutes from '@/routes/messages
;'import paymentRoutes from '@/routes/payments
;'import snapshotRoutes from '@/routes/snapshot
;'import gdprRoutes from '@/routes/gdpr

// تحميل متغيرات البيئة
;()dotenv.config

;()const app = express
;const server = createServer(app)
;const PORT = process.env.PORT || 3000

// تهيئة Socket.IO
;RealtimeNotificationService.initializeSocketIO(server)

// أمان الوسطاء
;app.use(helmetConfig)
;app.use(securityHeaders)
;app.use(requestLogger)
```

```
app.use(requestSizeLimit('10mb')); // حد 10 ميجابايت
app.use(sqlInjectionProtection);
app.use(xssProtection);

// مراقبة الأداء والوسطاء
app.use(performanceMonitor);
app.use(rateMonitor());
app.use(errorRateMonitor());

app.use(cors({
  origin: process.env.FRONTEND_URL || 'http://localhost:3000',
  credentials: true,
  methods: ['GET', 'POST', 'PUT', 'DELETE', 'OPTIONS'],
  allowedHeaders: ['Content-Type', 'Authorization', 'X-CSRF-Token', 'X-API-Key']
}));

// تقييد المعدل
app.use(generalRateLimit);

// الملفات الثابتة
app.use(express.static('public'));

// وسطاء تحليل الجسم
app.use(express.json({
  limit: '10mb',
  verify: (req, res, buf) => {
    try {
      JSON.parse(buf.toString());
    } catch (e) {
      // خطأ في تحليل JSON - سيتم التعامل معه بواسطة middleware الأخطاء
      throw new Error('Invalid JSON format');
    }
  }
}));
app.use(express.urlencoded({ extended: true }));

// معالج أخطاء تحليل JSON - سيتم نقله بعد المسارات

// وسطاء تسجيل الطلبات
app.use((req, res, next) => {
```

```
} ,logger.info( `${req.method} ${req.path}
,ip: req.ip
,userAgent: req.get('User-Agent')
()timestamp: new Date().toISOString
;({
;()next
;({

// المسارات مع الأمان المحسن
;app.use('/api/auth', authRateLimit, authRoutes)
;app.use('/api/admin', sensitiveRateLimit, adminRoutes)
;app.use('/api/transactions', sensitiveRateLimit, transactionRoutes)
;app.use('/api/users', userRoutes)
;app.use('/api/market', marketRoutes)
;app.use('/api/notifications', notificationRoutes)
;app.use('/api/messages', messageRoutes)
;app.use('/api/payments', sensitiveRateLimit, paymentRoutes)
;app.use('/api/snapshot', snapshotRoutes)
;app.use('/api/gdpr', gdprRoutes)

// نقاط فحص الصحة
} <= app.get('/api/health', async (req, res)
} try
;()const healthStatus = await healthCheckService.performHealthCheck
;const statusCode = healthStatus.status === 'healthy' ? 200 : 503

})res.status(statusCode).json
,'success: healthStatus.status === 'healthy
,data: healthStatus
()timestamp: new Date().toISOString
;({
} catch (error) {
;logger.error('Health check failed:', { error: (error as Error).message })
})res.status(503).json
,success: false
,'error: 'Health check failed
()timestamp: new Date().toISOString
;({
{
;({
```

```
} <= app.get('/api/health/quick', async (req, res)
} try
;()const quickHealth = await healthCheckService.getQuickHealthCheck
;const statusCode = quickHealth.status === 'healthy' ? 200 : 503

})res.status(statusCode).json
,'success: quickHealth.status === 'healthy
,data: quickHealth
()timestamp: new Date().toISOString
;({
} catch (error) {
;logger.error('Quick health check failed:', { error: (error as Error).message })
})res.status(503).json
,success: false
,'error: 'Quick health check failed
()timestamp: new Date().toISOString
;({
{
;({

// نقاط المراقبة
} <= app.get('/api/monitoring/dashboard', async (req, res)
} try
;const { PrismaClient } = await import('@prisma/client')
;()const prisma = new PrismaClient
;const monitoringService = new MonitoringService(prisma)

;()const dashboardData = await monitoringService.getDashboardData

})res.json
,success: true
,data: dashboardData
()timestamp: new Date().toISOString
;({
} catch (error) {
;logger.error('Monitoring dashboard failed:', { error: (error as Error).message })
})res.status(500).json
,success: false
,'error: 'Failed to load monitoring data
```

```
()timestamp: new Date().toISOString
;{{
{
;{{

} <= app.get('/api/monitoring/metrics', async (req, res)
} try
;const { PrismaClient } = await import('@prisma/client')
;()const prisma = new PrismaClient
;const monitoringService = new MonitoringService(prisma)

;()const metrics = monitoringService.getMetrics

})res.json
,success: true
,data: metrics
()timestamp: new Date().toISOString
;{{
} catch (error) {
;logger.error('Metrics retrieval failed:', { error: (error as Error).message })
})res.status(500).json
,success: false
,'error: 'Failed to load metrics
()timestamp: new Date().toISOString
;{{
{
;{{

// معالج أخطاء تحليل JSON
} <= app.use((error: any, req: any, res: any, next: any)
} if (error instanceof SyntaxError && error.message.includes('JSON'))
})return res.status(400).json
,success: false
,'error: 'Invalid JSON format
,'code: 'INVALID_JSON
()timestamp: new Date().toISOString
;{{
{
;next(error)
;{{
```

```
// معالج أخطاء عالمي
import { globalErrorHandler } from './middleware/errorHandler';
app.use(globalErrorHandler);

// معالج 404
app.use('*', (req, res) => {
  res.status(404).json({
    success: false,
    error: 'Route not found',
    data: { code: 'ROUTE_NOT_FOUND' },
    timestamp: new Date().toISOString()
  });
});

// إغلاق متدرج
const gracefulShutdown = async (signal: string) => {
  logger.info(`${signal} received, shutting down gracefully`);

  try {
    // إغلاق الخادم
    server.close(() => {
      logger.info('HTTP server closed');
    });

    // إغلاق اتصالات قاعدة البيانات
    const { PrismaClient } = await import('@prisma/client');
    const prisma = new PrismaClient();
    await prisma.$disconnect();

    // إغلاق اتصال Redis
    await redisService.disconnect();

    // تنظيف خدمة فحص الصحة
    await healthCheckService.cleanup();

    logger.info('Graceful shutdown completed');
    process.exit(0);
  } catch (error) {
    logger.error('Error during graceful shutdown:', { error: (error as Error).message });
```

```
;process.exit(1)
{
;{

;process.on('SIGTERM', () => gracefulShutdown('SIGTERM'))
;process.on('SIGINT', () => gracefulShutdown('SIGINT'))
nodemon لـ // ;process.on('SIGUSR2', () => gracefulShutdown('SIGUSR2'))

// بدء الخادم
} <= () const startServer = async
} try
// تهيئة المراقبة
;()memoryMonitor

// إعداد مهام التنظيف
تنظيف كل ساعة // ;setInterval(cleanupExpiredRequests, 60 *60* 1000)

} <= () ,server.listen(PORT
;logger.info( 🚀 `Server running on port ${PORT}` )
;logger.info( 📊 `Environment: ${process.env.NODE_ENV || 'development'}` )
;logger.info( 🔗 `Health check: http://localhost:${PORT}/api/health` )
logger.info(📈 `Monitoring dashboard:
; http://localhost:${PORT}/api/monitoring/dashboard`)
;logger.info( 🔌 `Socket.IO initialized for real-time notifications` )
;logger.info( 🛡️ `Idempotency middleware enabled` )
;logger.info( 📝 `Enhanced audit logging enabled` )
;({
} catch (error) {
;logger.error('Failed to start server:', { error: (error as Error).message })
;process.exit(1)
{
;{

// بدء الخادم فقط إذا لم نكن في بيئة الاختبار
} if (process.env.NODE_ENV !== 'test')
;()startServer
{

;export default app
```

متحكم المصادقة: authController.ts

```typescript
import { Request, Response } from 'express';
import { PrismaClient } from '@prisma/client';
import { validationResult } from 'express-validator';
import crypto from 'crypto';
import {
  generateAccessToken,
  generateRefreshToken,
  hashPassword,
  comparePassword,
  storeRefreshToken,
  revokeRefreshToken,
  revokeAllUserTokens
} from '@/middleware/auth';
import { RegisterRequest, AuthResponse, LoginRequest } from '@/types';
import logger, { logAuth } from '@/utils/logger';
import captchaService from '@/services/captchaService';
import kycScreeningService from '@/services/kycScreeningService';

const prisma = new PrismaClient();

/**
 * تسجيل مستخدم جديد
 */
export const register = async (req: Request, res: Response) => {
  try {
    // التحقق من صحة المدخلات
    const errors = validationResult(req);
    if (!errors.isEmpty()) {
      return res.status(400).json({
        success: false,
        error: 'Validation failed',
        data: { details: errors.array() }
      });
```

```
()timestamp: new Date().toISOString
;(as any {
{

const { email, password, firstName, lastName, phone, dateOfBirth, countryCode = 'DE',
;address, captchaToken } = req.body

// التحقق من CAPTCHA إذا تم تكوينها
} if (captchaService.isConfigured())
} if (!captchaToken)
})return res.status(400).json
,success: false
,'error: 'CAPTCHA verification is required
,data: { code: 'CAPTCHA_REQUIRED' }
()timestamp: new Date().toISOString
;({
{

const captchaResult = await captchaService.verifyTokenV3(captchaToken, 'register', 0.5,
;req.ip)
} if (!captchaResult.success)
})return res.status(400).json
,success: false
,'error: 'CAPTCHA verification failed
,data: { code: 'CAPTCHA_FAILED', reason: captchaResult.reason }
()timestamp: new Date().toISOString
;({
{
{

// التحقق من وجود المستخدم
})const existingUser = await prisma.user.findUnique
where: { email }
;({

} if (existingUser)
})return res.status(409).json
,success: false
,'error: 'User already exists with this email
,data: { code: 'USER_EXISTS' }
```

```
()timestamp: new Date().toISOString
;(as any {
{

// تشفير كلمة المرور
;const passwordHash = await hashPassword(password)

// إجراء فحص KYC للمستخدمين الجدد
;let kycScreeningResult = null
} try
)kycScreeningResult = await kycScreeningService.performScreening(
,firstName
,lastName
,dateOfBirth
countryCode
;(

// تسجيل نتيجة الفحص
)await kycScreeningService.logScreeningResult(
", // userId سيتم تعيينه بعد إنشاء المستخدم
,kycScreeningResult
,req.ip
req.get('User-Agent')
;(
} catch (error) {
} ,':logger.warn('KYC screening failed during registration
,error: (error as Error).message
,firstName
lastName
;({
{

// إنشاء المستخدم في المعاملة
} <= const result = await prisma.$transaction(async (tx)
// إدراج المستخدم
})const user = await tx.user.create
} :data
,email
,passwordHash
,firstName
```

```
,lastName
,phone
,dateOfBirth: dateOfBirth ? new Date(dateOfBirth) : null
,countryCode
,address: address ? JSON.stringify(address) : undefined
'kycStatus: kycScreeningResult?.overallRisk === 'HIGH' ? 'REJECTED' : 'PENDING
{
;({

// إنشاء محفظة EUR افتراضية
})await tx.wallet.create
} :data
,userId: user.id
,'currency: 'EUR
,balance: 0
lockedBalance: 0
{
;({

// تخزين نتيجة فحص KYC إذا كانت متاحة
} if (kycScreeningResult)
})await tx.auditLog.create
} :data
,userId: user.id
,'action: 'KYC_SCREENING_RESULT
,'tableName: 'users
,recordId: user.id
} :newValues
,screeningResult: kycScreeningResult
,overallRisk: kycScreeningResult.overallRisk
recommendation: kycScreeningResult.recommendation
,{
,ipAddress: req.ip
userAgent: req.get('User-Agent')
{
;({
{

;return user
;({
```

```
// إنشاء الرموز المميزة
const accessToken = generateAccessToken({ userId: result.id, email: result.email });
const refreshToken = generateRefreshToken({ userId: result.id, email: result.email });

// تشفير رمز التحديث للتخزين
const refreshTokenHash = crypto.createHash('sha256').update(refreshToken).digest('hex');

// تخزين رمز التحديث
await storeRefreshToken(result.id, refreshTokenHash, {
  ip: req.ip,
  userAgent: req.get('User-Agent')
});

logAuth.register(result.id, result.email, {
  ip: req.ip,
  userAgent: req.get('User-Agent')
});

res.status(201).json({
  success: true,
  user: {
    id: result.id,
    email: result.email,
    firstName: result.firstName,
    lastName: result.lastName,
    kycStatus: result.kycStatus,
    isVerified: result.isVerified
  },
  tokens: {
    accessToken,
    refreshToken,
    expiresIn: process.env.JWT_EXPIRES_IN || '15m'
  },
  timestamp: new Date().toISOString()
});
return;

} catch (error) {
logger.error('Registration error:', { error: (error as Error).message });
```

```
})return res.status(500).json
,success: false
,'error: 'Registration failed
,data: { code: 'REGISTRATION_FAILED' }
()timestamp: new Date().toISOString
;(as any }
{
;{

/
* تسجيل دخول المستخدم
/*
} <= export const login = async (req: Request, res: Response)
} try
// التحقق من صحة المدخلات
;const errors = validationResult(req)
} if (!errors.isEmpty())
})return res.status(400).json
,success: false
,'error: 'Validation failed
,data: { details: errors.array() }
()timestamp: new Date().toISOString
;(as any }
{

;const { email, password, captchaToken } = req.body

// التحقق من CAPTCHA إذا تم تكوينها
} if (captchaService.isConfigured())
} if (!captchaToken)
})return res.status(400).json
,success: false
,'error: 'CAPTCHA verification is required
,data: { code: 'CAPTCHA_REQUIRED' }
()timestamp: new Date().toISOString
;({}
{

;const captchaResult = await captchaService.verifyTokenV3(captchaToken, 'login', 0.5, req.ip)
} if (!captchaResult.success)
```

```
})return res.status(400).json
,success: false
,'error: 'CAPTCHA verification failed
,data: { code: 'CAPTCHA_FAILED', reason: captchaResult.reason }
()timestamp: new Date().toISOString
;({
{
{

// الحصول على المستخدم مع تشفير كلمة المرور
})const user = await prisma.user.findUnique
,where: { email }
} :select
,id: true
,email: true
,passwordHash: true
,firstName: true
,lastName: true
,isActive: true
,isVerified: true
,kycStatus: true
lastLogin: true
{
;({

} if (!user)
} ,logAuth.login('', email, false
,ip: req.ip
userAgent: req.get('User-Agent')
;({
})return res.status(401).json
,success: false
,'error: 'Invalid credentials
,data: { code: 'INVALID_CREDENTIALS' }
()timestamp: new Date().toISOString
;(as any {
{

// التحقق من نشاط الحساب
} if (!user.isActive)
```

```
})return res.status(401).json
,success: false
,'error: 'Account is deactivated
,data: { code: 'ACCOUNT_DEACTIVATED' }
()timestamp: new Date().toISOString
;(as any { }
{

// التحقق من كلمة المرور
;const isPasswordValid = await comparePassword(password, user.passwordHash)
} if (!isPasswordValid)
} ,logAuth.login(user.id, email, false
,ip: req.ip
userAgent: req.get('User-Agent')
;({
})return res.status(401).json
,success: false
,'error: 'Invalid credentials
,data: { code: 'INVALID_CREDENTIALS' }
()timestamp: new Date().toISOString
;(as any { }
{

// تحديث آخر تسجيل دخول
})await prisma.user.update
,where: { id: user.id }
data: { lastLogin: new Date() }
;({

// إنشاء الرموز المميزة
;const accessToken = generateAccessToken({ userId: user.id, email: user.email })
;const refreshToken = generateRefreshToken({ userId: user.id, email: user.email })

// تشفير رمز التحديث للتخزين
;const refreshTokenHash = crypto.createHash('sha256').update(refreshToken).digest('hex')

// تخزين رمز التحديث
} ,await storeRefreshToken(user.id, refreshTokenHash
,ip: req.ip
userAgent: req.get('User-Agent')
```

```
;({

} ,logAuth.login(user.id, email, true
,ip: req.ip
userAgent: req.get('User-Agent')
;({

})res.json
,success: true
} :user
,id: user.id
,email: user.email
,firstName: user.firstName
,lastName: user.lastName
,kycStatus: user.kycStatus
isVerified: user.isVerified
,{
} :tokens
,accessToken
,refreshToken
'expiresIn: process.env.JWT_EXPIRES_IN || '15m
,{
()timestamp: new Date().toISOString
;({
;return

} catch (error) {
;logger.error('Login error:', { error: (error as Error).message })
})return res.status(500).json
,success: false
,'error: 'Login failed
,data: { code: 'LOGIN_FAILED' }
()timestamp: new Date().toISOString
;(as any {
{
;{

/
* تجديد رمز الوصول باستخدام رمز التحديث
/*
```

```
} <= export const refreshToken = async (req: Request, res: Response)
} try
;const { refreshToken } = req.body

} if (!refreshToken)
})return res.status(400).json
,success: false
,'error: 'Refresh token is required
,data: { code: 'MISSING_REFRESH_TOKEN' }
()timestamp: new Date().toISOString
;({
{

// تشفير رمز التحديث المقدم
;const refreshTokenHash = crypto.createHash('sha256').update(refreshToken).digest('hex')

// التحقق من صحة رمز التحديث وعدم إلغائه
})const tokenRecord = await prisma.refreshToken.findFirst
} :where
,tokenHash: refreshTokenHash
,isRevoked: false
expiresAt: { gt: new Date() }
{
;({

} if (!tokenRecord)
})return res.status(401).json
,success: false
,'error: 'Invalid or expired refresh token
,data: { code: 'INVALID_REFRESH_TOKEN' }
()timestamp: new Date().toISOString
;({
{

// الحصول على معلومات المستخدم الحالي
})const user = await prisma.user.findUnique
,where: { id: tokenRecord.userId }
} :select
,id: true
,email: true
```

```
,firstName: true
,lastName: true
,isActive: true
,isVerified: true
kycStatus: true
{
;({

} if (!user || !user.isActive)
})return res.status(401).json
,success: false
,'error: 'User not found or inactive
,data: { code: 'USER_NOT_FOUND' }
()timestamp: new Date().toISOString
;({
{

// إنشاء رمز وصول جديد
;const newAccessToken = generateAccessToken({ userId: user.id, email: user.email })

})res.json
,success: true
} :data
} :tokens
,accessToken: newAccessToken
'expiresIn: process.env.JWT_EXPIRES_IN || '15m
{
,{
()timestamp: new Date().toISOString
;({
;return

} catch (error) {
;logger.error('Token refresh error:', { error: (error as Error).message })
})return res.status(401).json
,success: false
,'error: 'Token refresh failed
,data: { code: 'REFRESH_FAILED' }
()timestamp: new Date().toISOString
;({
```

```
{
;{

/
* تسجيل خروج المستخدم وإلغاء رمز التحديث
/*
} <= export const logout = async (req: Request, res: Response)
} try
;const { refreshToken } = req.body

} if (!refreshToken)
})return res.status(400).json
,success: false
,'error: 'Refresh token is required
,data: { code: 'MISSING_REFRESH_TOKEN' }
()timestamp: new Date().toISOString
;({
{

// تشفير رمز التحديث
;const refreshTokenHash = crypto.createHash('sha256').update(refreshToken).digest('hex')

// إلغاء رمز التحديث
;await revokeRefreshToken(refreshTokenHash)

} ,'')logAuth.logout
,ip: req.ip
userAgent: req.get('User-Agent')
;({

})res.json
,success: true
,data: { message: 'Logout successful' }
()timestamp: new Date().toISOString
;({
;return

} catch (error) {
;logger.error('Logout error:', { error: (error as Error).message })
})return res.status(500).json
```

```
,success: false
,'error: 'Logout failed
,data: { code: 'LOGOUT_FAILED' }
()timestamp: new Date().toISOString
;({
{
;{

/
* تسجيل خروج المستخدم من جميع الأجهزة
/*
} <= export const logoutAll = async (req: Request, res: Response)
} try
;const userId = (req as any).user?.id
} if (!userId)
})return res.status(401).json
,success: false
,'error: 'Authentication required
,data: { code: 'AUTH_REQUIRED' }
()timestamp: new Date().toISOString
;({
{

// إلغاء جميع رموز التحديث للمستخدم
;await revokeAllUserTokens(userId)

} ,logAuth.logout(userId
,ip: req.ip
userAgent: req.get('User-Agent')
;({

})res.json
,success: true
,data: { message: 'All sessions logged out successfully' }
()timestamp: new Date().toISOString
;({
;return

} catch (error) {
;logger.error('Logout all error:', { error: (error as Error).message })
```

```
})return res.status(500).json
,success: false
,'error: 'Logout all failed
,data: { code: 'LOGOUT_ALL_FAILED' }
()timestamp: new Date().toISOString
;({
{
;{

/
* الحصول على معلومات المستخدم الحالي
/*
} <= export const getMe = async (req: Request, res: Response)
} try
;const userId = (req as any).user?.id
} if (!userId)
})return res.status(401).json
,success: false
,'error: 'Authentication required
,data: { code: 'AUTH_REQUIRED' }
()timestamp: new Date().toISOString
;({
{

})const user = await prisma.user.findUnique
,where: { id: userId }
} :select
,id: true
,email: true
,firstName: true
,lastName: true
,phone: true
,dateOfBirth: true
,countryCode: true
,address: true
,isVerified: true
,isActive: true
,kycStatus: true
,createdAt: true
lastLogin: true
```

```
{
});

} if (!user)
})return res.status(404).json
,success: false
,'error: 'User not found
,data: { code: 'USER_NOT_FOUND' }
()timestamp: new Date().toISOString
});
{

})res.json
,success: true
} :data
} :user
,user...
address: user.address ? JSON.parse(user.address as string) : null
{
,{
()timestamp: new Date().toISOString
});
;return

} catch (error) {
;logger.error('Get user info error:', { error: (error as Error).message })
})return res.status(500).json
,success: false
,'error: 'Failed to get user info
,data: { code: 'GET_USER_FAILED' }
()timestamp: new Date().toISOString
});
{
;{
`
```

يتبع في الجزء الرابع: قاعدة البيانات والخدمات

دليل منصة ZEREX CARBON - الجزء الرابع

قاعدة البيانات والخدمات

قاعدة البيانات

مخطط قاعدة البيانات: schema.prisma

```
prisma `
// This is your Prisma schema file,
// learn more about it in the docs: https://pris.ly/d/prisma-schema

generator client {
  provider = "prisma-client-js"
}

datasource db {
  provider = "postgresql"
  url = env("DATABASE_URL")
}

// ========================================== //
```

```
// إدارة المستخدمين
// ==========================================

} model User
id String @id @default(uuid())
email String @unique
passwordHash String @map("password_hash")
firstName String @map("first_name")
lastName String @map("last_name")
?phone String
dateOfBirth DateTime? @map("date_of_birth") @db.Date
countryCode String @default("DE") @map("country_code")
?address Json
kycStatus KycStatus @default(PENDING) @map("kyc_status")
isVerified Boolean @default(false) @map("is_verified")
isActive Boolean @default(true) @map("is_active")
stripeCustomerId String? @unique @map("stripe_customer_id")
profileImage String? @map("profile_image")
profilePicture String? @map("profile_picture")
?bio String
language Language @default(EN) @map("language")
googleId String? @unique @map("google_id")
("{}")preferences Json @default
createdAt DateTime @default(now()) @map("created_at")
updatedAt DateTime @updatedAt @map("updated_at")
lastLogin DateTime? @map("last_login")

// العلاقات
[]wallets Wallet
[]transactions Transaction
[]userCarbonCredits UserCarbonCredit
[]refreshTokens RefreshToken
[]auditLogs AuditLog
[]processedRequests ProcessedRequest
[]notifications Notification
sentMessages Message[] @relation("MessageSender")
receivedMessages Message[] @relation("MessageReceiver")
[]projectSubscriptions ProjectSubscription
[]userRoles UserRole
?admin Admin
```

```
[]kycDetails KycDetail
[]subscriptions Subscription
[]bankAccounts BankAccount

index([email])@@
index([kycStatus])@@
index([isActive])@@
index([createdAt])@@
index([lastLogin])@@
map("users")@@
{

} model Admin
id String @id @default(uuid())
userId String @unique @map("user_id")
role AdminRole @default(ADMIN)
("{}")permissions Json @default
createdAt DateTime @default(now()) @map("created_at")
createdBy String? @map("created_by")

// العلاقات
user User @relation(fields: [userId], references: [id], onDelete: Cascade)
createdByAdmin Admin? @relation("AdminCreatedBy", fields: [createdBy], references: [id])
createdAdmins Admin[] @relation("AdminCreatedBy")
[]auditLogs AuditLog

map("admins")@@
{

} model Role
id String @id @default(uuid())
name String @unique
?description String
("{}")permissions Json @default
isActive Boolean @default(true) @map("is_active")
createdAt DateTime @default(now()) @map("created_at")
updatedAt DateTime @updatedAt @map("updated_at")

// العلاقات
[]userRoles UserRole
```

```
map("roles")@@
{

} model UserRole
id String @id @default(uuid())
userId String @map("user_id")
roleId String @map("role_id")
createdAt DateTime @default(now()) @map("created_at")

// العلاقات
user User @relation(fields: [userId], references: [id], onDelete: Cascade)
role Role @relation(fields: [roleId], references: [id], onDelete: Cascade)

unique([userId, roleId])@@
map("user_roles")@@
{

// ==========================================
// المشاريع وأرصدة الكربون
// ========================================== //

} model Project
id String @id @default(uuid())
name String
?description String
projectType ProjectType @map("project_type")
country String
?region String
{lat, lng} // ?Json coordinates للخرائط
verificationStandard String @map("verification_standard")
vintageYear Int @map("vintage_year")
totalCredits Decimal @map("total_credits") @db.Decimal(20, 8)
availableCredits Decimal @map("available_credits") @db.Decimal(20, 8)
pricePerCredit Decimal @map("price_per_credit") @db.Decimal(10, 4)
currency Currency @default(EUR)
status ProjectStatus @default(PENDING)
isActive Boolean @default(true) @map("is_active")
startDate DateTime? @map("start_date")
endDate DateTime? @map("end_date")
```

```
images String[] // مصفوفة من روابط الصور
documents Json @default("[]") // مصفوفة من كائنات المستندات
impactMetrics Json @default("{}") // بيانات التأثير البيئي
communityBenefits Json @default("{}") // فوائد المجتمع المحلي
aiVerification Json @default("{}") // بيانات التحقق الذكي/القمر الصناعي
rating Decimal? @db.Decimal(3, 2) // 0.00 إلى 5.00
progress Decimal @default(0) @db.Decimal(5, 2) // 0.00 إلى 100.00
createdAt DateTime @default(now()) @map("created_at")
updatedAt DateTime @updatedAt @map("updated_at")
approvedAt DateTime? @map("approved_at")
approvedBy String? @map("approved_by")

// العلاقات
carbonCredits CarbonCredit[]
subscriptions ProjectSubscription[]
projectUpdates ProjectUpdate[]
projectDocuments ProjectDocument[]

@@map("projects")
}

model CarbonCredit {
id String @id @default(uuid())
projectId String @map("project_id")
name String
description String?
projectType ProjectType @map("project_type")
country String
verificationStandard String @map("verification_standard")
vintageYear Int @map("vintage_year")
totalSupply Decimal @map("total_supply") @db.Decimal(20, 8)
availableSupply Decimal @map("available_supply") @db.Decimal(20, 8)
pricePerTon Decimal @map("price_per_ton") @db.Decimal(10, 4)
currency Currency @default(EUR)
isActive Boolean @default(true) @map("is_active")
createdAt DateTime @default(now()) @map("created_at")
updatedAt DateTime @updatedAt @map("updated_at")

// العلاقات
project Project @relation(fields: [projectId], references: [id], onDelete: Cascade)
```

```
[]transactions Transaction
[]userCarbonCredits UserCarbonCredit
[]priceHistory CarbonCreditPrice

map("carbon_credits")@@
{

} model ProjectUpdate
id String @id @default(uuid())
projectId String @map("project_id")
title String
content String
images String[] // مصفوفة من روابط الصور
isPublic Boolean @default(true) @map("is_public")
createdAt DateTime @default(now()) @map("created_at")

// العلاقات
project Project @relation(fields: [projectId], references: [id], onDelete: Cascade)

map("project_updates")@@
{

} model ProjectDocument
id String @id @default(uuid())
projectId String @map("project_id")
name String
type String // شهادة، تقرير، تحقق، إلخ
url String
size Int? // حجم الملف بالبايت
createdAt DateTime @default(now()) @map("created_at")

// العلاقات
project Project @relation(fields: [projectId], references: [id], onDelete: Cascade)

map("project_documents")@@
{

} model ProjectSubscription
id String @id @default(uuid())
userId String @map("user_id")
```

```
projectId String @map("project_id")
isActive Boolean @default(true) @map("is_active")
createdAt DateTime @default(now()) @map("created_at")

// العلاقات
user User @relation(fields: [userId], references: [id], onDelete: Cascade)
project Project @relation(fields: [projectId], references: [id], onDelete: Cascade)

unique([userId, projectId])@@
map("project_subscriptions")@@
{

// ========== //
// النماذج المالية
// ========== //

} model Wallet
id String @id @default(uuid())
userId String @map("user_id")
currency Currency @default(EUR)
balance Decimal @default(0) @db.Decimal(20, 8)
lockedBalance Decimal @default(0) @map("locked_balance") @db.Decimal(20, 8)
createdAt DateTime @default(now()) @map("created_at")
updatedAt DateTime @updatedAt @map("updated_at")

// العلاقات
user User @relation(fields: [userId], references: [id], onDelete: Cascade)

unique([userId, currency])@@
map("wallets")@@
{

} model Transaction
id String @id @default(uuid())
userId String @map("user_id")
type TransactionType
status TransactionStatus @default(PENDING)
amount Decimal @db.Decimal(20, 8)
currency Currency @default(EUR)
fee Decimal @default(0) @db.Decimal(20, 8)
```

```
netAmount Decimal @map("net_amount") @db.Decimal(20, 8)
externalTransactionId String? @map("external_transaction_id")
paymentMethod String? @map("payment_method")
paymentIntentId String? @map("payment_intent_id")
carbonCreditId String? @map("carbon_credit_id")
carbonCreditsAmount Decimal? @map("carbon_credits_amount") @db.Decimal(20, 8)
carbonCreditsPrice Decimal? @map("carbon_credits_price") @db.Decimal(10, 4)
?description String
("{}")metadata Json @default
createdAt DateTime @default(now()) @map("created_at")
updatedAt DateTime @updatedAt @map("updated_at")
completedAt DateTime? @map("completed_at")

// العلاقات
user User @relation(fields: [userId], references: [id], onDelete: Cascade)
carbonCredit CarbonCredit? @relation(fields: [carbonCreditId], references: [id])

index([userId])@@
index([status])@@
index([type])@@
index([createdAt])@@
index([carbonCreditId])@@
map("transactions")@@
{

} model CarbonCreditPrice
id String @id @default(uuid())
carbonCreditId String @map("carbon_credit_id")
pricePerTon Decimal @map("price_per_ton") @db.Decimal(10, 4)
price Decimal? @db.Decimal(10, 4)
priceChange Decimal? @map("price_change") @db.Decimal(5, 2)
volumeTraded Decimal @default(0) @map("volume_traded") @db.Decimal(20, 8)
volume Decimal? @db.Decimal(20, 8)
marketCap Decimal? @map("market_cap") @db.Decimal(20, 2)
?market String
currency Currency @default(EUR)
timestamp DateTime @default(now())
?source String
change24h Decimal? @map("change_24h") @db.Decimal(5, 2)
createdAt DateTime @default(now()) @map("created_at")
```

```
// العلاقات
carbonCredit CarbonCredit @relation(fields: [carbonCreditId], references: [id], onDelete:
Cascade)

map("carbon_credit_prices")@@
{

} model UserCarbonCredit
id String @id @default(uuid())
userId String @map("user_id")
carbonCreditId String @map("carbon_credit_id")
amount Decimal @default(0) @db.Decimal(20, 8)
averagePurchasePrice Decimal @default(0) @map("average_purchase_price")
@db.Decimal(10, 4)
totalInvested Decimal @default(0) @map("total_invested") @db.Decimal(20, 8)
createdAt DateTime @default(now()) @map("created_at")
updatedAt DateTime @updatedAt @map("updated_at")

// العلاقات
user User @relation(fields: [userId], references: [id], onDelete: Cascade)
carbonCredit CarbonCredit @relation(fields: [carbonCreditId], references: [id], onDelete:
Cascade)

unique([userId, carbonCreditId])@@
map("user_carbon_credits")@@
{

// ==========
// الإشعارات والرسائل
// ==========

} model Notification
id String @id @default(uuid())
userId String @map("user_id")
type NotificationType
title String
message String
("{}")data Json @default
isRead Boolean @default(false) @map("is_read")
```

```
isImportant Boolean @default(false) @map("is_important")
createdAt DateTime @default(now()) @map("created_at")
readAt DateTime? @map("read_at")

// العلاقات
user User @relation(fields: [userId], references: [id], onDelete: Cascade)

index([userId])@@
index([isRead])@@
index([type])@@
index([createdAt])@@
map("notifications")@@
{

} model Message
id String @id @default(uuid())
senderId String @map("sender_id")
receiverId String @map("receiver_id")
?subject String
content String
isRead Boolean @default(false) @map("is_read")
isImportant Boolean @default(false) @map("is_important")
attachments Json @default("[]") // مصفوفة من كائنات المرفقات
createdAt DateTime @default(now()) @map("created_at")
readAt DateTime? @map("read_at")

// العلاقات
sender User @relation("MessageSender", fields: [senderId], references: [id], onDelete: Cascade)
receiver User @relation("MessageReceiver", fields: [receiverId], references: [id], onDelete: Cascade)

map("messages")@@
{

// ==========
// المصادقة
// ========== //

} model RefreshToken
```

```
  id String @id @default(uuid())
  userId String @map("user_id")
  token String @unique
  tokenHash String @unique @map("token_hash")
  expiresAt DateTime @map("expires_at")
  createdAt DateTime @default(now()) @map("created_at")
  isRevoked Boolean @default(false) @map("is_revoked")
  deviceInfo Json @default("{}") @map("device_info")

  // العلاقات
  user User @relation(fields: [userId], references: [id], onDelete: Cascade)

  @@index([userId])
  @@index([expiresAt])
  @@index([isRevoked])
  @@map("refresh_tokens")
}

// ============================================
// تتبع الطلبات
// ============================================

model ProcessedRequest {
  id String @id @default(uuid())
  requestId String @unique @map("request_id")
  userId String? @map("user_id")
  endpoint String
  method String
  result Json // تخزين نتيجة الاستجابة
  status String @default("COMPLETED") // COMPLETED, FAILED, PROCESSING
  createdAt DateTime @default(now()) @map("created_at")
  expiresAt DateTime @map("expires_at") // TTL للتنظيف

  // العلاقات
  user User? @relation(fields: [userId], references: [id], onDelete: SetNull)

  @@map("processed_requests")
}

// ============================================
```

```
// التدقيق والتسجيل
// ========================================== //

} model AuditLog
id String @id @default(uuid())
userId String? @map("user_id")
adminId String? @map("admin_id")
action String
tableName String? @map("table_name")
recordId String? @map("record_id")
oldValues Json? @map("old_values")
newValues Json? @map("new_values")
ipAddress String? @map("ip_address")
userAgent String? @map("user_agent")
createdAt DateTime @default(now()) @map("created_at")

// العلاقات
user User? @relation(fields: [userId], references: [id], onDelete: SetNull)
admin Admin? @relation(fields: [adminId], references: [id], onDelete: SetNull)

index([userId])@@
index([adminId])@@
index([action])@@
index([tableName])@@
index([createdAt])@@
map("audit_logs")@@
{

// ========================================== //
// إدارة KYC
// ========================================== //

} model KycDetail
id String @id @default(uuid())
userId String @map("user_id")
status KycStatus @default(PENDING)
firstName String @map("first_name")
lastName String @map("last_name")
dateOfBirth DateTime? @map("date_of_birth") @db.Date
?nationality String
```

```
?address Json
?phone String
submittedAt DateTime @default(now()) @map("submitted_at")
reviewedAt DateTime? @map("reviewed_at")
reviewedBy String? @map("reviewed_by")
rejectionReason String? @map("rejection_reason")
createdAt DateTime @default(now()) @map("created_at")
updatedAt DateTime @updatedAt @map("updated_at")

// العلاقات
user User @relation(fields: [userId], references: [id], onDelete: Cascade)
[]documents KycDocument

map("kyc_details")@@
{

} model KycDocument
id String @id @default(uuid())
kycDetailId String @map("kyc_detail_id")
type KycDocumentType
filename String
filepath String
fileSize Int @map("file_size")
mimeType String @map("mime_type")
isVerified Boolean @default(false) @map("is_verified")
verifiedAt DateTime? @map("verified_at")
verifiedBy String? @map("verified_by")
createdAt DateTime @default(now()) @map("created_at")

// العلاقات
kycDetail KycDetail @relation(fields: [kycDetailId], references: [id], onDelete: Cascade)

map("kyc_documents")@@
{

// ==========
// سندات الكربون والشهادات
// ========== //

} model CarbonBond
```

```
  id String @id @default(uuid())
  name String
  ?description String
  issuerId String @map("issuer_id")
  totalSupply Decimal @map("total_supply") @db.Decimal(20, 8)
  availableSupply Decimal @map("available_supply") @db.Decimal(20, 8)
  faceValue Decimal @map("face_value") @db.Decimal(10, 4)
  interestRate Decimal @map("interest_rate") @db.Decimal(5, 4)
  maturityDate DateTime @map("maturity_date")
  issueDate DateTime @map("issue_date")
  status String @default("ACTIVE")
  isActive Boolean @default(true) @map("is_active")
  createdAt DateTime @default(now()) @map("created_at")
  updatedAt DateTime @updatedAt @map("updated_at")

  // العلاقات
  []purchases CarbonBondPurchase
  []certificates CarbonBondCertificate
  []payments BondInterestPayment

  map("carbon_bonds")@@
}

} model CarbonBondPurchase
  id String @id @default(uuid())
  userId String @map("user_id")
  carbonBondId String @map("carbon_bond_id")
  amount Decimal @db.Decimal(20, 8)
  purchasePrice Decimal @map("purchase_price") @db.Decimal(10, 4)
  purchaseDate DateTime @default(now()) @map("purchase_date")
  status String @default("ACTIVE")
  createdAt DateTime @default(now()) @map("created_at")

  // العلاقات
  carbonBond CarbonBond @relation(fields: [carbonBondId], references: [id], onDelete: Cascade)

  map("carbon_bond_purchases")@@
{

} model CarbonBondCertificate
```

```
  id String @id @default(uuid())
  carbonBondId String @map("carbon_bond_id")
  certificateNumber String @unique @map("certificate_number")
  issueDate DateTime @default(now()) @map("issue_date")
  expiryDate DateTime @map("expiry_date")
  status String @default("ACTIVE")
  createdAt DateTime @default(now()) @map("created_at")

  // العلاقات
  carbonBond CarbonBond @relation(fields: [carbonBondId], references: [id], onDelete: Cascade)

  map("carbon_bond_certificates")@@
}

} model BondInterestPayment
  id String @id @default(uuid())
  carbonBondId String @map("carbon_bond_id")
  amount Decimal @db.Decimal(10, 4)
  paymentDate DateTime @default(now()) @map("payment_date")
  status String @default("COMPLETED")
  createdAt DateTime @default(now()) @map("created_at")

  // العلاقات
  carbonBond CarbonBond @relation(fields: [carbonBondId], references: [id], onDelete: Cascade)

  map("bond_interest_payments")@@
}

// ============================================== //
// الوساطة والسمسرة
// ============================================== //

} model Mediation
  id String @id @default(uuid())
  userId String @map("user_id")
  sourcePlatform String @map("source_platform")
  targetPlatform String @map("target_platform")
  sourceAmount Decimal @map("source_amount") @db.Decimal(20, 8)
  targetAmount Decimal @map("target_amount") @db.Decimal(20, 8)
  commission Decimal @db.Decimal(10, 4)
```

```
actualProfit Decimal @map("actual_profit") @db.Decimal(10, 4)
status String @default("PENDING")
requestedAt DateTime @default(now()) @map("requested_at")
completedAt DateTime? @map("completed_at")
createdAt DateTime @default(now()) @map("created_at")

// العلاقات
[]transactions MediationTransaction

map("mediations")@@
{

} model MediationTransaction
id String @id @default(uuid())
mediationId String @map("mediation_id")
sourceAmount Decimal @map("source_amount") @db.Decimal(20, 8)
targetAmount Decimal @map("target_amount") @db.Decimal(20, 8)
commission Decimal @db.Decimal(10, 4)
actualProfit Decimal @map("actual_profit") @db.Decimal(10, 4)
status String @default("PENDING")
createdAt DateTime @default(now()) @map("created_at")
completedAt DateTime? @map("completed_at")

// العلاقات
mediation Mediation @relation(fields: [mediationId], references: [id], onDelete: Cascade)

map("mediation_transactions")@@
{

// ============================================== //
// الاشتراكات والمدفوعات
// ============================================== //

} model Subscription
id String @id @default(uuid())
userId String @map("user_id")
type SubscriptionType
status SubscriptionStatus @default(PENDING)
startDate DateTime @default(now()) @map("start_date")
endDate DateTime? @map("end_date")
```

```
price Decimal @db.Decimal(10, 4)
currency Currency @default(EUR)
isActive Boolean @default(true) @map("is_active")
createdAt DateTime @default(now()) @map("created_at")
updatedAt DateTime @updatedAt @map("updated_at")

// العلاقات
user User @relation(fields: [userId], references: [id], onDelete: Cascade)
[]payments Payment

map("subscriptions")@@@
{

} model Payment
id String @id @default(uuid())
subscriptionId String @map("subscription_id")
amount Decimal @db.Decimal(10, 4)
currency Currency @default(EUR)
status PaymentStatus @default(PENDING)
provider PaymentProviderType
providerTransactionId String? @map("provider_transaction_id")
paidAt DateTime? @map("paid_at")
createdAt DateTime @default(now()) @map("created_at")

// العلاقات
subscription Subscription @relation(fields: [subscriptionId], references: [id], onDelete: Cascade)

map("payments")@@@
{

// ============================================== //
// الحسابات المصرفية وإعداد النظام
// ============================================== //

} model BankAccount
id String @id @default(uuid())
userId String @map("user_id")
bankName String @map("bank_name")
accountNumber String @map("account_number")
routingNumber String? @map("routing_number")
```

```
?iban String
?swift String
isActive Boolean @default(true) @map("is_active")
createdAt DateTime @default(now()) @map("created_at")
updatedAt DateTime @updatedAt @map("updated_at")

// العلاقات
user User @relation(fields: [userId], references: [id], onDelete: Cascade)

map("bank_accounts")@@
{

} model SystemConfig
id String @id @default(uuid())
key String @unique
value Json
?category String
isActive Boolean @default(true) @map("is_active")
createdAt DateTime @default(now()) @map("created_at")
updatedAt DateTime @updatedAt @map("updated_at")

map("system_config")@@
{

// ============================================ //
// التعدادات
// ============================================ //

} enum KycStatus
PENDING
UNDER_REVIEW
APPROVED
REJECTED
{

} enum AdminRole
SUPER_ADMIN
ADMIN
MODERATOR
COMPLIANCE
```

```
    VIEWER
}

} enum Currency
    EUR
    USD
    GBP
    CHF
{

} enum TransactionType
    DEPOSIT
    WITHDRAWAL
    PURCHASE
    SALE
    TRANSFER
    REFUND
{

} enum TransactionStatus
    PENDING
    COMPLETED
    FAILED
    CANCELLED
    REFUNDED
{

} enum ProjectType
    RENEWABLE_ENERGY
    FOREST_CONSERVATION
    WASTE_MANAGEMENT
    ENERGY_EFFICIENCY
    CARBON_CAPTURE
    AFFORESTATION
    BLUE_CARBON
    AGRICULTURE
    TRANSPORT
    INDUSTRY
{
```

```
} enum ProjectStatus
  PENDING
  UNDER_REVIEW
  APPROVED
  REJECTED
  ACTIVE
  COMPLETED
  SUSPENDED
{

} enum NotificationType
  TRANSACTION
  PROJECT_UPDATE
  SYSTEM_ALERT
  MESSAGE
  PROJECT_APPROVAL
  PAYMENT_FAILED
  HIGH_VALUE_TRANSACTION
  KYC_UPDATE
  GENERAL
{

} enum Language
  EN
  FR
  DE
  ES
  PT
  ET
{

} enum PaymentProviderType
  STRIPE
  PAYPAL
  BANK_TRANSFER
  CRYPTO
{

} enum KycDocumentType
  PASSPORT
```

ID_CARD
DRIVER_LICENSE
UTILITY_BILL
BANK_STATEMENT
PROOF_OF_ADDRESS
PROOF_OF_INCOME
OTHER
{

} enum SubscriptionType
BASIC
PREMIUM
ENTERPRISE
CUSTOM
{

} enum SubscriptionStatus
ACTIVE
INACTIVE
CANCELLED
EXPIRED
PENDING
{

} enum PaymentStatus
PENDING
COMPLETED
FAILED
CANCELLED
REFUNDED
{

يتبع في الجزء الخامس: تطبيق الهاتف المحمول والخدمات

دليل منصة ZEREX CARBON - الجزء الخامس

تطبيق الهاتف المحمول والخدمات

تطبيق الهاتف المحمول

ملف التطبيق الرئيسي: main.dart

```dart
import 'package:flutter/material.dart';
import 'package:flutter_localizations/flutter_localizations.dart';
import 'package:flutter_bloc/flutter_bloc.dart';
import 'package:firebase_core/firebase_core.dart';
import 'package:shared_preferences/shared_preferences.dart';

import 'core/config/app_config.dart';
import 'core/config/firebase_config.dart';
import 'core/di/injection_container.dart';
import 'core/localization/app_localizations.dart';
import 'core/localization/localization_bloc.dart';
import 'core/theme/app_theme.dart';
import 'core/routing/app_router.dart';
import 'features/auth/presentation/bloc/auth_bloc.dart';
import 'features/sync/presentation/bloc/sync_bloc.dart';
import 'features/notifications/presentation/bloc/notification_bloc.dart';
```

```
} void main() async
;()WidgetsFlutterBinding.ensureInitialized

// تهيئة Firebase
)await Firebase.initializeApp
,options: DefaultFirebaseOptions.currentPlatform
;(

// تهيئة حقن التبعيات
;()await initializeDependencies

// تهيئة إعدادات التطبيق
;()await AppConfig.initialize

;runApp(const ZerexCarbonApp())
{

} class ZerexCarbonApp extends StatelessWidget
;const ZerexCarbonApp({super.key})

override@
} Widget build(BuildContext context)
)return MultiBlocProvider
] :providers
)BlocProvider
,create: (context) => getIt()..add(LoadLocalization())
,(
)BlocProvider
,create: (context) => getIt()..add(CheckAuthStatus())
,(
)BlocProvider
,()create: (context) => getIt
,(
)BlocProvider
,create: (context) => getIt()..add(InitializeNotifications())
,(
,[
)child: BlocBuilder
} builder: (context, state)
)return MaterialApp.router
```

```
,'title: 'ZEREX CARBON
,debugShowCheckedModeBanner: false

// الترجمة
,locale: state.locale
] supportedLocales: const
,Locale('en', '')
,Locale('de', '')
,Locale('fr', '')
,Locale('pt', '')
,Locale('et', '')
,Locale('ar', '')
,[
] localizationsDelegates: const
,AppLocalizations.delegate
,GlobalMaterialLocalizations.delegate
,GlobalWidgetsLocalizations.delegate
,GlobalCupertinoLocalizations.delegate
,[

// السمة
,theme: AppTheme.lightTheme
,darkTheme: AppTheme.darkTheme
,themeMode: ThemeMode.system

// التوجيه
,routerConfig: AppRouter.router

// دعم RTL
} builder: (context, child)
)return Directionality
,textDirection: state.isRTL ? TextDirection.rtl : TextDirection.ltr
,!child: child
;(
,{
;(
,{
,(
;(
{
```

{

`

إعدادات التطبيق: app_config.dart

dart `

```
import 'package:shared_preferences/shared_preferences.dart';
import 'package:package_info_plus/package_info_plus.dart';
import 'package:device_info_plus/device_info_plus.dart';

class AppConfig {
  static late String _version;
  static late String _buildNumber;
  static late String _deviceId;
  static late String _platform;
  static late Map _config;

  // الحصول على معلومات التطبيق
  static String get version => _version;
  static String get buildNumber => _buildNumber;
  static String get deviceId => _deviceId;
  static String get platform => _platform;

  // تهيئة إعدادات التطبيق
  static Future initialize() async {
    try {
      // الحصول على معلومات الحزمة
      final packageInfo = await PackageInfo.fromPlatform();
      _version = packageInfo.version;
      _buildNumber = packageInfo.buildNumber;

      // الحصول على معلومات الجهاز
      final deviceInfo = DeviceInfoPlugin();
      if (deviceInfo is AndroidDeviceInfo) {
        _platform = 'Android';
        _deviceId = deviceInfo.id;
      } else if (deviceInfo is IosDeviceInfo) {
```

```
;'platform = 'iOS_
;'deviceId = deviceInfo.identifierForVendor ?? 'unknown_
{

// تحميل الإعدادات المحفوظة
;()await _loadSavedConfig

;print('App config initialized successfully')
} catch (e) {
;print('Failed to initialize app config: $e')
;rethrow
{
{

// تحميل الإعدادات المحفوظة
} static Future _loadSavedConfig() async
;()final prefs = await SharedPreferences.getInstance

} = config_
,'theme': prefs.getString('theme') ?? 'system'
,'language': prefs.getString('language') ?? 'en'
,biometricEnabled': prefs.getBool('biometric_enabled') ?? false'
,notificationsEnabled': prefs.getBool('notifications_enabled') ?? true'
,offlineModeEnabled': prefs.getBool('offline_mode_enabled') ?? true'
,autoSyncEnabled': prefs.getBool('auto_sync_enabled') ?? true'
;{
{

// حفظ الإعدادات
} static Future saveConfig(String key, dynamic value) async
;()final prefs = await SharedPreferences.getInstance

} if (value is String)
;await prefs.setString(key, value)
} else if (value is bool) {
;await prefs.setBool(key, value)
} else if (value is int) {
;await prefs.setInt(key, value)
} else if (value is double) {
;await prefs.setDouble(key, value)
```

```
{

;config[key] = value_
{

// الحصول على إعداد
} static T? getConfig(String key)
;?return _config[key] as T
{

// الحصول على جميع الإعدادات
} ()static Map getAllConfig
;return Map.from(_config)
{
{
```

خدمة المصادقة: auth_bloc.dart

```dart
;'import 'package:flutter_bloc/flutter_bloc.dart
;'import 'package:equatable/equatable.dart

;'import '../../domain/entities/user.dart
;'import '../../domain/usecases/login_usecase.dart
;'import '../../domain/usecases/register_usecase.dart
;'import '../../domain/usecases/logout_usecase.dart
;'import '../../domain/usecases/refresh_token_usecase.dart

// الأحداث
} abstract class AuthEvent extends Equatable
;()const AuthEvent

override@
;[] <= List get props
{
```

```
{} class CheckAuthStatus extends AuthEvent

} class LoginRequested extends AuthEvent
  ;final String email
  ;final String password
  ;final bool useBiometric

})const LoginRequested
  ,required this.email
  ,required this.password
  ,this.useBiometric = false
  ;({

  override@
;List get props => [email, password, useBiometric]
  {

} class RegisterRequested extends AuthEvent
  ;final String email
  ;final String password
  ;final String firstName
  ;final String lastName
  ;final String? phone
  ;final String countryCode

})const RegisterRequested
  ,required this.email
  ,required this.password
  ,required this.firstName
  ,required this.lastName
  ,this.phone
  ,'this.countryCode = 'DE
  ;({

  override@
;List get props => [email, password, firstName, lastName, phone, countryCode]
  {

{} class LogoutRequested extends AuthEvent
```

```
{} class RefreshTokenRequested extends AuthEvent

{} class BiometricLoginRequested extends AuthEvent

// الحالات
} abstract class AuthState extends Equatable
;()const AuthState

override@
;[] <= List get props
{

{} class AuthInitial extends AuthState

{} class AuthLoading extends AuthState

} class AuthAuthenticated extends AuthState
;final User user

;const AuthAuthenticated({required this.user})

override@
;List get props => [user]
{

{} class AuthUnauthenticated extends AuthState

} class AuthError extends AuthState
;final String message

;const AuthError({required this.message})

override@
;List get props => [message]
{

BLoC //
} class AuthBloc extends Bloc
;final LoginUsecase _loginUsecase
;final RegisterUsecase _registerUsecase
```

```
;final LogoutUsecase _logoutUsecase
;final RefreshTokenUsecase _refreshTokenUsecase

})AuthBloc
,required LoginUsecase loginUsecase
,required RegisterUsecase registerUsecase
,required LogoutUsecase logoutUsecase
,required RefreshTokenUsecase refreshTokenUsecase
,loginUsecase = loginUsecase_ : ({
,registerUsecase = registerUsecase_
,logoutUsecase = logoutUsecase_
,refreshTokenUsecase = refreshTokenUsecase_
} super(AuthInitial())
;on(_onCheckAuthStatus)
;on(_onLoginRequested)
;on(_onRegisterRequested)
;on(_onLogoutRequested)
;on(_onRefreshTokenRequested)
;on(_onBiometricLoginRequested)
{

)Future _onCheckAuthStatus
,CheckAuthStatus event
,Emitter emit
} async (
} try
;emit(AuthLoading())

// التحقق من وجود رمز مميز محفوظ
;()final result = await _refreshTokenUsecase

} if (result.isRight())
;final user = result.getOrElse(() => throw Exception())
;emit(AuthAuthenticated(user: user))
} else {
;emit(AuthUnauthenticated())
{
} catch (e) {
;emit(AuthUnauthenticated())
{
```

```
{

)Future _onLoginRequested
,LoginRequested event
,Emitter emit
} async (
} try
;emit(AuthLoading())

)final result = await _loginUsecase
)LoginParams
,email: event.email
,password: event.password
,useBiometric: event.useBiometric
,(
;(

} if (result.isRight())
;final user = result.getOrElse(() => throw Exception())
;emit(AuthAuthenticated(user: user))
} else {
;final failure = result.fold((l) => l, (r) => throw Exception())
;emit(AuthError(message: failure.message))
{
} catch (e) {
;((''{()e.toString}$ فشل في تسجيل الدخول: 'message)AuthError)emit
{
{

)Future _onRegisterRequested
,RegisterRequested event
,Emitter emit
} async (
} try
;emit(AuthLoading())

)final result = await _registerUsecase
)RegisterParams
,email: event.email
,password: event.password
```

```
,firstName: event.firstName
,lastName: event.lastName
,phone: event.phone
,countryCode: event.countryCode
,(
;(

} if (result.isRight())
;final user = result.getOrElse(() => throw Exception())
;emit(AuthAuthenticated(user: user))
} else {
;final failure = result.fold((l) => l, (r) => throw Exception())
;emit(AuthError(message: failure.message))
{
} catch (e) {
;(('{()e.toString}$ :فشل في التسجيل' :message)AuthError)emit
{
{

)Future _onLogoutRequested
,LogoutRequested event
,Emitter emit
} async (
} try
;emit(AuthLoading())

;()await _logoutUsecase
;emit(AuthUnauthenticated())
} catch (e) {
;(('{()e.toString}$ :فشل في تسجيل الخروج' :message)AuthError)emit
{
{

)Future _onRefreshTokenRequested
,RefreshTokenRequested event
,Emitter emit
} async (
} try
;()final result = await _refreshTokenUsecase
```

```
} if (result.isRight())
;final user = result.getOrElse(() => throw Exception())
;emit(AuthAuthenticated(user: user))
} else {
;emit(AuthUnauthenticated())
{
} catch (e) {
;emit(AuthUnauthenticated())
{
{

)Future _onBiometricLoginRequested
,BiometricLoginRequested event
,Emitter emit
} async (
} try
;emit(AuthLoading())

// تنفيذ تسجيل الدخول البيومتري
// هذا يتطلب تنفيذ خدمة المصادقة البيومترية

// للآن، سنقوم بإرسال خطأ
;'المصادقة البيومترية غير متاحة حالياً' :message)AuthError)emit
} catch (e) {
;'فشل في المصادقة البيومترية: ${()e.toString}' :message)AuthError)emit
{
{
{
```

صفحة لوحة التحكم: dashboard_page.dart

```dart
;'import 'package:flutter/material.dart
;'import 'package:flutter_bloc/flutter_bloc.dart

;'import '../../../../core/localization/app_localizations.dart
```

```
';'import '../../../../core/theme/app_theme.dart
';'import '../bloc/dashboard_bloc.dart
';'import '../bloc/dashboard_event.dart
';'import '../bloc/dashboard_state.dart
';'import '../widgets/portfolio_summary_card.dart
';'import '../widgets/recent_transactions_list.dart
';'import '../widgets/market_overview_card.dart
';'import '../widgets/quick_actions_row.dart

} class DashboardPage extends StatefulWidget
;const DashboardPage({super.key})

override@
;()State createState() => _DashboardPageState
{

} class _DashboardPageState extends State
override@
} ()void initState
;()super.initState
;context.read().add(LoadDashboardData())
{

override@
} Widget build(BuildContext context)
;!final l10n = AppLocalizations.of(context)

)return Scaffold
)appBar: AppBar
,title: Text(l10n.dashboard)
,backgroundColor: AppTheme.primaryColor
,foregroundColor: Colors.white
,elevation: 0
] :actions
)IconButton
,icon: const Icon(Icons.notifications_outlined)
} () :onPressed
// التنقل إلى صفحة الإشعارات
,{
,(
```

```
)IconButton
,icon: const Icon(Icons.settings_outlined)
} () :onPressed
// التنقل إلى صفحة الإعدادات
,{
,(
,[
,(
)body: BlocBuilder
} builder: (context, state)
} if (state is DashboardLoading)
)return const Center
,()child: CircularProgressIndicator
;(
{

} if (state is DashboardError)
)return Center
)child: Column
,mainAxisAlignment: MainAxisAlignment.center
] :children
)Icon
,Icons.error_outline
,size: 64
,color: Colors.red.shade300
,(
,const SizedBox(height: 16)
)Text
,state.message
,style: Theme.of(context).textTheme.titleMedium
,textAlign: TextAlign.center
,(
,const SizedBox(height: 16)
)ElevatedButton
} () :onPressed
;context.read().add(LoadDashboardData())
,{
,child: Text(l10n.retry)
,(
,[
```

```
),
);
{

} if (state is DashboardLoaded)
)return RefreshIndicator
} onRefresh: () async
;context.read().add(LoadDashboardData())
,{
)child: SingleChildScrollView
,()physics: const AlwaysScrollableScrollPhysics
,padding: const EdgeInsets.all(16)
)child: Column
,crossAxisAlignment: CrossAxisAlignment.start
] :children
// ملخص المحفظة
)PortfolioSummaryCard
,portfolio: state.dashboardData.portfolio
),

,const SizedBox(height: 16)

// نظرة عامة على السوق
)MarketOverviewCard
,marketData: state.dashboardData.marketOverview
),

,const SizedBox(height: 16)

// الإجراءات السريعة
)QuickActionsRow
} () :onBuyCredits
// التنقل إلى صفحة شراء الأرصدة
,{
} () :onSellCredits
// التنقل إلى صفحة بيع الأرصدة
,{
} () :onViewProjects
// التنقل إلى صفحة المشاريع
,{
```

```
onGenerateReport: () {
// توليد تقرير
},
),

const SizedBox(height: 24),

// المعاملات الأخيرة
Text(
l10n.recentTransactions,
style: Theme.of(context).textTheme.titleLarge?.copyWith(
fontWeight: FontWeight.bold,
),
),

const SizedBox(height: 12),

RecentTransactionsList(
transactions: state.dashboardData.recentTransactions,
),

const SizedBox(height: 24),

// إحصائيات إضافية
_buildAdditionalStats(state.dashboardData),
],
),
),
);
}

return const SizedBox.shrink();
},
),
);
}

Widget _buildAdditionalStats(DashboardData data) {
return Card(
child: Padding(
```

```
,padding: const EdgeInsets.all(16)
)child: Column
,crossAxisAlignment: CrossAxisAlignment.start
] :children
)Text
,AppLocalizations.of(context)!.additionalStats
)style: Theme.of(context).textTheme.titleMedium?.copyWith
,fontWeight: FontWeight.bold
,(
,(
,const SizedBox(height: 16)
)Row
] :children
)Expanded
)child: _buildStatItem
,AppLocalizations.of(context)!.totalInvested
,'EUR {data.portfolio.totalInvested.toStringAsFixed(2)}$'
,Icons.account_balance_wallet
,Colors.green
,(
,(
)Expanded
)child: _buildStatItem
,AppLocalizations.of(context)!.carbonCreditsOwned
,()data.portfolio.totalCredits.toString
,Icons.eco
,Colors.blue
,(
,(
,[
,(
,const SizedBox(height: 16)
)Row
] :children
)Expanded
)child: _buildStatItem
,AppLocalizations.of(context)!.averagePrice
,'EUR {data.portfolio.averagePrice.toStringAsFixed(2)}$'
,Icons.trending_up
,Colors.orange
```

```
,(
,(
)Expanded
)child: _buildStatItem
,AppLocalizations.of(context)!.portfolioValue
,'EUR {data.portfolio.currentValue.toStringAsFixed(2)}$'
,Icons.assessment
,Colors.purple
,(
,(
,[
,(
,[
,(
,(
;(
{

} Widget _buildStatItem(String title, String value, IconData icon, Color color)
)return Column
] :children
)Icon
,icon
,color: color
,size: 24
,(
,const SizedBox(height: 8)
)Text
,value
)style: Theme.of(context).textTheme.titleMedium?.copyWith
,fontWeight: FontWeight.bold
,color: color
,(
,(
,const SizedBox(height: 4)
)Text
,title
,style: Theme.of(context).textTheme.bodySmall
,textAlign: TextAlign.center
,(
```

```
],
);
{
{
```

خدمة التزامن: sync_bloc.dart

```dart
import 'package:flutter_bloc/flutter_bloc.dart';
import 'package:equatable/equatable.dart';

import '../../domain/usecases/sync_portfolio_usecase.dart';
import '../../domain/usecases/sync_market_data_usecase.dart';
import '../../domain/usecases/sync_transactions_usecase.dart';

// الأحداث
abstract class SyncEvent extends Equatable {
const SyncEvent();

@override
List get props => [];
{

class SyncPortfolio extends SyncEvent {}

class SyncMarketData extends SyncEvent {}

class SyncTransactions extends SyncEvent {}

class SyncAll extends SyncEvent {}

class StartAutoSync extends SyncEvent {}

class StopAutoSync extends SyncEvent {}

// الحالات
```

```
} abstract class SyncState extends Equatable
;()const SyncState

override@
;[] <= List get props
{

{} class SyncInitial extends SyncState

} class SyncInProgress extends SyncState
;final String operation
;final double progress

})const SyncInProgress
,required this.operation
,this.progress = 0.0
;({

override@
;List get props => [operation, progress]
{

} class SyncCompleted extends SyncState
;final String operation
;final DateTime completedAt

})const SyncCompleted
,required this.operation
,required this.completedAt
;({

override@
;List get props => [operation, completedAt]
{

} class SyncError extends SyncState
;final String operation
;final String message

})const SyncError
```

```
,required this.operation
,required this.message
;({

override@
;List get props => [operation, message]
{

{} class AutoSyncEnabled extends SyncState

{} class AutoSyncDisabled extends SyncState

BLoC //
} class SyncBloc extends Bloc
;final SyncPortfolioUsecase _syncPortfolioUsecase
;final SyncMarketDataUsecase _syncMarketDataUsecase
;final SyncTransactionsUsecase _syncTransactionsUsecase

})SyncBloc
,required SyncPortfolioUsecase syncPortfolioUsecase
,required SyncMarketDataUsecase syncMarketDataUsecase
,required SyncTransactionsUsecase syncTransactionsUsecase
,syncPortfolioUsecase = syncPortfolioUsecase_ : ({
,syncMarketDataUsecase = syncMarketDataUsecase_
,syncTransactionsUsecase = syncTransactionsUsecase_
} super(SyncInitial())
;on(_onSyncPortfolio)
;on(_onSyncMarketData)
;on(_onSyncTransactions)
;on(_onSyncAll)
;on(_onStartAutoSync)
;on(_onStopAutoSync)
{

)Future _onSyncPortfolio
,SyncPortfolio event
,Emitter emit
} async (
} try
;emit(const SyncInProgress(operation: 'portfolio'))
```

```
;()final result = await _syncPortfolioUsecase

} if (result.isRight())
)emit(SyncCompleted
,'operation: 'portfolio
,()completedAt: DateTime.now
;((
} else {
;final failure = result.fold((l) => l, (r) => throw Exception())
)emit(SyncError
,'operation: 'portfolio
,message: failure.message
;((
{
} catch (e) {
)emit(SyncError
,'operation: 'portfolio
,()message: e.toString
;((
{
{

)Future _onSyncMarketData
,SyncMarketData event
,Emitter emit
} async (
} try
;emit(const SyncInProgress(operation: 'market_data'))

;()final result = await _syncMarketDataUsecase

} if (result.isRight())
)emit(SyncCompleted
,'operation: 'market_data
,()completedAt: DateTime.now
;((
} else {
;final failure = result.fold((l) => l, (r) => throw Exception())
)emit(SyncError
```

```
,'operation: 'market_data
,message: failure.message
;((
{
} catch (e) {
)emit(SyncError
,'operation: 'market_data
,()message: e.toString
;((
{
{

)Future _onSyncTransactions
,SyncTransactions event
,Emitter emit
} async (
} try
;emit(const SyncInProgress(operation: 'transactions'))

;()final result = await _syncTransactionsUsecase

} if (result.isRight())
)emit(SyncCompleted
,'operation: 'transactions
,()completedAt: DateTime.now
;((
} else {
;final failure = result.fold((l) => l, (r) => throw Exception())
)emit(SyncError
,'operation: 'transactions
,message: failure.message
;((
{
} catch (e) {
)emit(SyncError
,'operation: 'transactions
,()message: e.toString
;((
{
{
```

```
)Future _onSyncAll
,SyncAll event
,Emitter emit
} async (
} try
;emit(const SyncInProgress(operation: 'all'))

// تزامن المحفظة
;()final portfolioResult = await _syncPortfolioUsecase
} if (portfolioResult.isLeft())
;final failure = portfolioResult.fold((l) => l, (r) => throw Exception())
)emit(SyncError
,'operation: 'all
,'message: 'Portfolio sync failed: ${failure.message}
;((
;return
{

// تزامن بيانات السوق
;()final marketResult = await _syncMarketDataUsecase
} if (marketResult.isLeft())
;final failure = marketResult.fold((l) => l, (r) => throw Exception())
)emit(SyncError
,'operation: 'all
,'message: 'Market data sync failed: ${failure.message}
;((
;return
{

// تزامن المعاملات
;()final transactionsResult = await _syncTransactionsUsecase
} if (transactionsResult.isLeft())
;final failure = transactionsResult.fold((l) => l, (r) => throw Exception())
)emit(SyncError
,'operation: 'all
,'message: 'Transactions sync failed: ${failure.message}
;((
;return
{
```

```
)emit(SyncCompleted
,'operation: 'all
,()completedAt: DateTime.now
;((
} catch (e) {
)emit(SyncError
,'operation: 'all
,()message: e.toString
;((
{
{

)Future _onStartAutoSync
,StartAutoSync event
,Emitter emit
} async (
;emit(AutoSyncEnabled())

// بدء التزامن التلقائي كل 30 ثانية
// هذا يتطلب تنفيذ Timer أو Stream
{

)Future _onStopAutoSync
,StopAutoSync event
,Emitter emit
} async (
;emit(AutoSyncDisabled())

// إيقاف التزامن التلقائي
{
{
`
```

يتبع في الجزء السادس والأخير: النشر والإنتاج والأمان

دليل منصة ZEREX CARBON - الجزء السادس والأخير

النشر والإنتاج والأمان

النشر والإعداد

ملف Docker: Dockerfile

dockerfile `

استخدام Node.js 18 كصورة أساسية

FROM node:18-alpine AS base

تثبيت التبعيات اللازمة للنشر

RUN apk add --no-cache libc6-compat
WORKDIR /app

نسخ ملفات التبعيات

```
/. COPY package*.json
/. COPY tsconfig.json
/COPY prisma ./prisma
```

تثبيت التبعيات

```
RUN npm ci --only=production && npm cache clean --force
```

بناء التطبيق

```
FROM base AS builder
. . COPY
RUN npm run build
```

مرحلة الإنتاج

```
FROM base AS runner
WORKDIR /app
```

إنشاء مستخدم غير جذر

```
RUN addgroup --system --gid 1001 nodejs
```

```
RUN adduser --system --uid 1001 nextjs
```

نسخ الملفات المبنية

```
COPY --from=builder /app/dist ./dist
COPY --from=builder /app/node_modules ./node_modules
COPY --from=builder /app/package.json ./package.json
COPY --from=builder /app/prisma ./prisma
COPY --from=builder /app/public ./public
```

تعيين المالك

```
USER nextjs
```

كشف المنفذ

```
EXPOSE 3000
```

متغيرات البيئة

```
ENV NODE_ENV=production
ENV PORT=3000
```

بدء التطبيق

```
CMD ["npm", "start"]
```

Docker Compose: docker-compose.yml ملف

```yaml
version: '3.8'

services:
  # قاعدة البيانات
  postgres:
    image: postgres:15-alpine
    container_name: zerex_carbon_db
    restart: unless-stopped
    environment:
      POSTGRES_DB: zerex_carbon
      POSTGRES_USER: zerex_user
      POSTGRES_PASSWORD: ${DB_PASSWORD}
    volumes:
      - postgres_data:/var/lib/postgresql/data
      - ./database/init.sql:/docker-entrypoint-initdb.d/init.sql
    ports:
      - "5432:5432"
    networks:
      - zerex_network
    healthcheck:
      test: ["CMD-SHELL", "pg_isready -U zerex_user -d zerex_carbon"]
      interval: 30s
      timeout: 10s
      retries: 3

  # Redis للتخزين المؤقت
  redis:
    image: redis:7-alpine
    container_name: zerex_carbon_redis
    restart: unless-stopped
    command: redis-server --appendonly yes --requirepass ${REDIS_PASSWORD}
```

```
    volumes:
      - redis_data:/data
    ports:
      - "6379:6379"
    networks:
      - zerex_network
    healthcheck:
      test: ["CMD", "redis-cli", "ping"]
      interval: 30s
      timeout: 10s
      retries: 3

  # التطبيق الرئيسي
  app:
    build: .
    container_name: zerex_carbon_app
    restart: unless-stopped
    environment:
      NODE_ENV: production
      PORT: 3000
      DATABASE_URL: postgresql://zerex_user:${DB_PASSWORD}@postgres:5432/zerex_carbon
      REDIS_URL: redis://:${REDIS_PASSWORD}@redis:6379
      JWT_SECRET: ${JWT_SECRET}
      JWT_REFRESH_SECRET: ${JWT_REFRESH_SECRET}
      STRIPE_SECRET_KEY: ${STRIPE_SECRET_KEY}
      STRIPE_WEBHOOK_SECRET: ${STRIPE_WEBHOOK_SECRET}
      FRONTEND_URL: ${FRONTEND_URL}
    ports:
      - "3000:3000"
    depends_on:
      postgres:
        condition: service_healthy
      redis:
        condition: service_healthy
    networks:
      - zerex_network
    volumes:
      - logs:/app/logs/.
      - uploads:/app/uploads/.
    healthcheck:
```

```
test: ["CMD", "curl", "-f", "http://localhost:3000/api/health"]
interval: 30s
timeout: 10s
retries: 3

# Nginx كخادم وكيل
nginx:
image: nginx:alpine
container_name: zerex_carbon_nginx
restart: unless-stopped
ports:
- "80:80"
- "443:443"
volumes:
- nginx.conf:/etc/nginx/nginx.conf/.
- ssl:/etc/nginx/ssl/.
- public:/usr/share/nginx/html/.
depends_on:
- app
networks:
- zerex_network

volumes:
postgres_data:
redis_data:

networks:
zerex_network:
driver: bridge
```

إعدادات Nginx: nginx.conf

```
nginx
} events
worker_connections 1024;
{
```

```
} http
    ;include /etc/nginx/mime.types
    ;default_type application/octet-stream

    # إعدادات السجل
    ' "log_format main '$remote_addr - $remote_user [$time_local] "$request
    ' "status $body_bytes_sent "$http_referer$'
    ;'"http_user_agent" "$http_x_forwarded_for$"'

    ;access_log /var/log/nginx/access.log main
    ;error_log /var/log/nginx/error.log

    # إعدادات الأداء
    ;sendfile on
    ;tcp_nopush on
    ;tcp_nodelay on
    ;keepalive_timeout 65
    ;types_hash_max_size 2048

    # ضغط Gzip
    ;gzip on
    ;gzip_vary on
    ;gzip_min_length 1024
    ;gzip_proxied any
    ;gzip_comp_level 6
    gzip_types
        text/plain
        text/css
        text/xml
        text/javascript
        application/json
        application/javascript
        application/xml+rss
        application/atom+xml
        ;image/svg+xml

    # حد معدل الطلبات
    ;limit_req_zone $binary_remote_addr zone=api:10m rate=10r/s
    ;limit_req_zone $binary_remote_addr zone=auth:10m rate=5r/s
```

```
# خادم HTTP (إعادة توجيه إلى HTTPS)
server {
    listen 80;
    server_name zerexcarbon.com www.zerexcarbon.com;
    return 301 https://$server_name$request_uri;
}

# خادم HTTPS الرئيسي
server {
    listen 443 ssl http2;
    server_name zerexcarbon.com www.zerexcarbon.com;

    # شهادات SSL
    ssl_certificate /etc/nginx/ssl/cert.pem;
    ssl_certificate_key /etc/nginx/ssl/key.pem;
    ssl_protocols TLSv1.2 TLSv1.3;
    ssl_ciphers ECDHE-RSA-AES256-GCM-SHA512:DHE-RSA-AES256-GCM-SHA512:ECDHE-
    RSA-AES256-GCM-SHA384:DHE-RSA-AES256-GCM-SHA384;
    ssl_prefer_server_ciphers off;
    ssl_session_cache shared:SSL:10m;
    ssl_session_timeout 10m;

    # أمان إضافي
    add_header Strict-Transport-Security "max-age=31536000; includeSubDomains" always;
    add_header X-Frame-Options DENY always;
    add_header X-Content-Type-Options nosniff always;
    add_header X-XSS-Protection "1; mode=block" always;
    add_header Referrer-Policy "strict-origin-when-cross-origin" always;

    # الملفات الثابتة
    location / {
        root /usr/share/nginx/html;
        index index.html;
        try_files $uri $uri/ /index.html;
    }

    # تخزين مؤقت للملفات الثابتة
    location ~* \.(js|css|png|jpg|jpeg|gif|ico|svg|woff|woff2|ttf|eot)$ {
        expires 1y;
        add_header Cache-Control "public, immutable";
    }
```

```
{
{

API #
} /location /api
;limit_req zone=api burst=20 nodelay

;proxy_pass http://app:3000
;proxy_http_version 1.1
;proxy_set_header Upgrade $http_upgrade
;'proxy_set_header Connection 'upgrade
;proxy_set_header Host $host
;proxy_set_header X-Real-IP $remote_addr
;proxy_set_header X-Forwarded-For $proxy_add_x_forwarded_for
;proxy_set_header X-Forwarded-Proto $scheme
;proxy_cache_bypass $http_upgrade
;proxy_read_timeout 86400
{

API # المصادقة (حد أبطأ)
} /location /api/auth
;limit_req zone=auth burst=10 nodelay

;proxy_pass http://app:3000
;proxy_http_version 1.1
;proxy_set_header Host $host
;proxy_set_header X-Real-IP $remote_addr
;proxy_set_header X-Forwarded-For $proxy_add_x_forwarded_for
;proxy_set_header X-Forwarded-Proto $scheme
{

WebSocket #
} /location /ws
;proxy_pass http://app:3000
;proxy_http_version 1.1
;proxy_set_header Upgrade $http_upgrade
;"proxy_set_header Connection "upgrade
;proxy_set_header Host $host
;proxy_set_header X-Real-IP $remote_addr
;proxy_set_header X-Forwarded-For $proxy_add_x_forwarded_for
```

```
    proxy_set_header X-Forwarded-Proto $scheme;
}

# رفض الوصول للملفات الحساسة
location ~ /\. {
    deny all;
}

location ~ \.(env|log)$ {
    deny all;
}
}
}
```

ملف متغيرات البيئة: env.production.

```
env
```

إعدادات البيئة

```
NODE_ENV=production
PORT=3000
```

قاعدة البيانات

```
DATABASE_URL=postgresql://zerex_user:your_secure_password@postgres:5432/zerex_carbon
```

Redis

```
REDIS_URL=redis://:your_redis_password@redis:6379
```

JWT

```
JWT_SECRET=your_super_secure_jwt_secret_key_here
JWT_REFRESH_SECRET=your_super_secure_refresh_secret_key_here
JWT_EXPIRES_IN=15m
JWT_REFRESH_EXPIRES_IN=7d
```

Stripe

```
STRIPE_SECRET_KEY=sk_live_your_stripe_secret_key
STRIPE_WEBHOOK_SECRET=whsec_your_webhook_secret
STRIPE_PUBLISHABLE_KEY=pk_live_your_stripe_publishable_key
```

URLs

```
FRONTEND_URL=https://zerexcarbon.com
API_URL=https://api.zerexcarbon.com
WS_URL=wss://ws.zerexcarbon.com
```

حدود المعدل

```
RATE_LIMIT_WINDOW_MS=900000
RATE_LIMIT_MAX_REQUESTS=100
AUTH_RATE_LIMIT_WINDOW_MS=300000
AUTH_RATE_LIMIT_MAX_REQUESTS=5
```

```
SENSITIVE_RATE_LIMIT_WINDOW_MS=60000
SENSITIVE_RATE_LIMIT_MAX_REQUESTS=3
```

التسجيل

```
LOG_LEVEL=info
LOG_FILE_PATH=/app/logs
```

Firebase (للتطبيق المحمول)

```
FIREBASE_PROJECT_ID=zerex-carbon-mobile
FIREBASE_PRIVATE_KEY_ID=your_firebase_private_key_id
FIREBASE_PRIVATE_KEY="-----BEGIN PRIVATE KEY-----\nyour_firebase_private_key\n-----
"END PRIVATE KEY-----\n
FIREBASE_CLIENT_EMAIL=firebase-adminsdk-xxxxx@zerex-carbon-
mobile.iam.gserviceaccount.com
FIREBASE_CLIENT_ID=your_firebase_client_id
FIREBASE_AUTH_URI=https://accounts.google.com/o/oauth2/auth
FIREBASE_TOKEN_URI=https://oauth2.googleapis.com/token
```

CAPTCHA

```
RECAPTCHA_SECRET_KEY=your_recaptcha_secret_key
RECAPTCHA_SITE_KEY=your_recaptcha_site_key
```

البريد الإلكتروني

```
SMTP_HOST=smtp.gmail.com
```

```
SMTP_PORT=587
SMTP_USER=your_email@gmail.com
SMTP_PASS=your_app_password
FROM_EMAIL=noreply@zerexcarbon.com
```

المراقبة

```
SENTRY_DSN=your_sentry_dsn
NEW_RELIC_LICENSE_KEY=your_new_relic_license_key
```

الأمان

```
BCRYPT_ROUNDS=12
SESSION_SECRET=your_session_secret
ENCRYPTION_KEY=your_32_byte_encryption_key
```

ميزات التطبيق

```
ENABLE_BIOMETRIC_AUTH=true
ENABLE_OFFLINE_MODE=true
ENABLE_PUSH_NOTIFICATIONS=true
ENABLE_DEEP_LINKS=true
ENABLE_ANALYTICS=true
```

الأمان والمراقبة

خدمة المراقبة: monitoringService.ts

```typescript
import { PrismaClient } from '@prisma/client';
import logger from '@/utils/logger';

export default class MonitoringService {
  private prisma: PrismaClient;
  private metrics: Map = new Map();

  constructor(prisma: PrismaClient) {
    this.prisma = prisma;
    this.initializeMetrics();
  }

  private initializeMetrics() {
    this.metrics.set('requests_total', 0);
    this.metrics.set('requests_duration', []);
    this.metrics.set('errors_total', 0);
    this.metrics.set('active_users', 0);
    this.metrics.set('memory_usage', 0);
    this.metrics.set('cpu_usage', 0);
  }

  async getDashboardData() {
    try {
      const [
        totalUsers,
        totalTransactions,
        totalProjects,
        activeUsers,
        recentTransactions,
        systemHealth
      ] = await Promise.all([
        this.getTotalUsers(),
```

```
,()this.getTotalTransactions
,()this.getTotalProjects
,()this.getActiveUsers
,()this.getRecentTransactions
()this.getSystemHealth
;([

} return
} :overview
,totalUsers
,totalTransactions
,totalProjects
,activeUsers
systemHealth
,{
} :recentActivity
,transactions: recentTransactions
,()errors: await this.getRecentErrors
()performance: await this.getPerformanceMetrics
,{
} :charts
,()userGrowth: await this.getUserGrowthChart
,()transactionVolume: await this.getTransactionVolumeChart
()projectPerformance: await this.getProjectPerformanceChart
{
;{
} catch (error) {
;logger.error('Failed to get dashboard data:', { error: (error as Error).message })
;throw error
{
{

} private async getTotalUsers(): Promise
})return await this.prisma.user.count
where: { isActive: true }
;({
{

} private async getTotalTransactions(): Promise
;()return await this.prisma.transaction.count
```

```
{

} private async getTotalProjects(): Promise
})return await this.prisma.project.count
where: { isActive: true }
;({
{

} private async getActiveUsers(): Promise
;const oneDayAgo = new Date(Date.now() - 24 *60 * 60* 1000)

})return await this.prisma.user.count
} :where
} :lastLogin
gte: oneDayAgo
{
{
;({
{

} private async getRecentTransactions(limit: number = 10)
})return await this.prisma.transaction.findMany
,take: limit
,orderBy: { createdAt: 'desc' }
} :include
} :user
} :select
,id: true
,firstName: true
,lastName: true
email: true
{
,{
} :carbonCredit
} :select
,id: true
,name: true
projectType: true
{
{
```

```
{
;({
{

} ()private async getSystemHealth
;()const memoryUsage = process.memoryUsage
;()const cpuUsage = process.cpuUsage

} return
,()uptime: process.uptime
} :memory
,used: memoryUsage.heapUsed
,total: memoryUsage.heapTotal
,external: memoryUsage.external
rss: memoryUsage.rss
,{
} :cpu
,user: cpuUsage.user
system: cpuUsage.system
,{
,()database: await this.checkDatabaseHealth
()redis: await this.checkRedisHealth
;{
{

} private async checkDatabaseHealth(): Promise
} try
;await this.prisma.$queryRaw SELECT 1
;return true
} catch (error) {
;logger.error('Database health check failed:', { error: (error as Error).message })
;return false
{
{

} private async checkRedisHealth(): Promise
} try
;const redis = await import('@/services/redisService')
;()await redis.default.ping
;return true
```

```
    } catch (error) {
;logger.error('Redis health check failed:', { error: (error as Error).message })
;return false
    {
    {

  } private async getRecentErrors(limit: number = 10)
  })return await this.prisma.auditLog.findMany
    ,take: limit
    } :where
    } :action
    'contains: 'ERROR
    {
    ,{
    ,orderBy: { createdAt: 'desc' }
    } :select
    ,id: true
    ,action: true
    ,ipAddress: true
    ,userAgent: true
    ,createdAt: true
    } :user
    } :select
    email: true
    {
    {
    {
  ;({{
    {

  } ()private async getPerformanceMetrics
  ;[] || const requests = this.metrics.get('requests_duration')
  ;const totalRequests = this.metrics.get('requests_total') || 0
  ;const totalErrors = this.metrics.get('errors_total') || 0

  const avgResponseTime = requests.length > 0
  requests.reduce((a, b) => a + b, 0) / requests.length ?
  ;0 :

  const errorRate = totalRequests > 0
```

```
100 * (totalErrors / totalRequests) ?
;0 :

} return
,avgResponseTime
,errorRate
,totalRequests
,totalErrors
طلبات في الدقيقة // throughput: totalRequests / (process.uptime() / 60)
;{
{

} private async getUserGrowthChart(days: number = 30)
;const startDate = new Date(Date.now() - days *24 * 60 * 60* 1000)

})const users = await this.prisma.user.findMany
} :where
} :createdAt
gte: startDate
{
,{
} :select
createdAt: true
{
;({

// تجميع المستخدمين حسب التاريخ
;()const userGrowth = new Map
} <= users.forEach(user
;const date = user.createdAt.toISOString().split('T')[0]
;userGrowth.set(date, (userGrowth.get(date) || 0) + 1)
;({

}) <= return Array.from(userGrowth.entries()).map(([date, count])
,date
count
;(({
{

} private async getTransactionVolumeChart(days: number = 30)
```

```
;const startDate = new Date(Date.now() - days 24 * 60 * 60 1000)

})const transactions = await this.prisma.transaction.findMany
  } :where
  } :createdAt
  gte: startDate
  {
  ,{
  } :select
  ,createdAt: true
  ,amount: true
  currency: true
  {
  ;{{

// تجميع المعاملات حسب التاريخ
;()const transactionVolume = new Map
  } <= transactions.forEach(transaction
;const date = transaction.createdAt.toISOString().split('T')[0]
;const amount = parseFloat(transaction.amount.toString())
;transactionVolume.set(date, (transactionVolume.get(date) || 0) + amount)
;{{

  }) <= return Array.from(transactionVolume.entries()).map(([date, volume])
  ,date
  volume
  ;(({
  {

} ()private async getProjectPerformanceChart
})const projects = await this.prisma.project.findMany
  ,where: { isActive: true }
  } :select
  ,id: true
  ,name: true
  ,progress: true
  ,rating: true
  ,totalCredits: true
  availableCredits: true
  {
```

```
;({

}) <= return projects.map(project
,id: project.id
,name: project.name
,progress: parseFloat(project.progress.toString())
,rating: project.rating ? parseFloat(project.rating.toString()) : 0
creditsSold: parseFloat(project.totalCredits.toString()) -
,parseFloat(project.availableCredits.toString())
totalCredits: parseFloat(project.totalCredits.toString())
;(({
{

// تحديث المقاييس
} recordRequest(duration: number)
;const total = this.metrics.get('requests_total') || 0
;this.metrics.set('requests_total', total + 1)

;[] || const durations = this.metrics.get('requests_duration')
;durations.push(duration)

// الاحتفاظ بآخر 1000 طلب فقط
} if (durations.length > 1000)
;()durations.shift
{

;this.metrics.set('requests_duration', durations)
{

} ()recordError
;const total = this.metrics.get('errors_total') || 0
;this.metrics.set('errors_total', total + 1)
{

} ()updateMemoryUsage
;()const memoryUsage = process.memoryUsage
;this.metrics.set('memory_usage', memoryUsage.heapUsed)
{

} ()updateCpuUsage
```

```
;()const cpuUsage = process.cpuUsage
;this.metrics.set('cpu_usage', cpuUsage.user + cpuUsage.system)
{

} ()getMetrics
} return
} :requests
,total: this.metrics.get('requests_total')
,()avgDuration: this.getAverageRequestDuration
()errorRate: this.getErrorRate
,{
} :system
,memory: this.metrics.get('memory_usage')
,cpu: this.metrics.get('cpu_usage')
()uptime: process.uptime
,{
} :business
activeUsers: this.metrics.get('active_users')
{
;{
{

} private getAverageRequestDuration(): number
;[] || const durations = this.metrics.get('requests_duration')
;if (durations.length === 0) return 0

;return durations.reduce((a, b) => a + b, 0) / durations.length
{

} private getErrorRate(): number
;const total = this.metrics.get('requests_total') || 0
;const errors = this.metrics.get('errors_total') || 0

;return total > 0 ? (errors / total) * 100 : 0
{
{
`
```

خدمة فحص الصحة: healthCheckService.ts

```typescript
import { PrismaClient } from '@prisma/client';
import redisService from '@/services/redisService';
import logger from '@/utils/logger';

class HealthCheckService {
  private prisma: PrismaClient;
  private isHealthy: boolean = true;
  private lastCheck: Date = new Date();
  private checkInterval: NodeJS.Timeout | null = null;

  constructor() {
    this.prisma = new PrismaClient();
    this.startPeriodicChecks();
  }

  private startPeriodicChecks() {
    // فحص كل 30 ثانية
    this.checkInterval = setInterval(async () => {
      try {
        await this.performHealthCheck();
      } catch (error) {
        logger.error('Periodic health check failed:', { error: (error as Error).message });
      }
    }, 30000);
  }

  async performHealthCheck(): Promise {
    const startTime = Date.now();
    const checks = await Promise.allSettled([
      this.checkDatabase(),
      this.checkRedis(),
      this.checkMemory(),
      this.checkDiskSpace()
    ]);

    const results = {
      database: checks[0].status === 'fulfilled' ? checks[0].value : { status: 'error', message:
```

```
,checks[0].reason }
redis: checks[1].status === 'fulfilled' ? checks[1].value : { status: 'error', message:
,checks[1].reason }
memory: checks[2].status === 'fulfilled' ? checks[2].value : { status: 'error', message:
,checks[2].reason }
disk: checks[3].status === 'fulfilled' ? checks[3].value : { status: 'error', message:
checks[3].reason }
;{

;const allHealthy = Object.values(results).every((check: any) => check.status === 'healthy')
;this.isHealthy = allHealthy
;()this.lastCheck = new Date

;const responseTime = Date.now() - startTime

} return
,'status: allHealthy ? 'healthy' : 'unhealthy
,()timestamp: this.lastCheck.toISOString
,responseTime
,checks: results
,()uptime: process.uptime
,'version: process.env.npm_package_version || '1.0.0
'environment: process.env.NODE_ENV || 'development
;{
{

} async getQuickHealthCheck(): Promise
} try
// فحص سريع لقاعدة البيانات فقط
;await this.prisma.$queryRaw SELECT 1

} return
,'status: 'healthy
,()timestamp: new Date().toISOString
()uptime: process.uptime
;{
} catch (error) {
} return
,'status: 'unhealthy
,()timestamp: new Date().toISOString
```

```
        error: (error as Error).message
      };{
    {
  {

} private async checkDatabase(): Promise
  } try
;()const startTime = Date.now

// فحص الاتصال
;await this.prisma.$queryRaw SELECT 1

// فحص عدد الجداول
`const tableCount = await this.prisma.$queryRaw
  SELECT COUNT(*) as count
  FROM information_schema.tables
  'WHERE table_schema = 'public
  ;`

;const responseTime = Date.now() - startTime

  } return
  ,'status: 'healthy
  ,responseTime
  tables: (tableCount as any)[0].count
  ;{
} catch (error) {
  } return
  ,'status: 'error
  message: (error as Error).message
  ;{
  {
  {

} private async checkRedis(): Promise
  } try
;()const startTime = Date.now

;()await redisService.ping
```

```
;()const info = await redisService.info
;const responseTime = Date.now() - startTime

} return
,'status: 'healthy
,responseTime
,version: info.redis_version
,usedMemory: info.used_memory_human
connectedClients: info.connected_clients
;{
} catch (error) {
} return
,'status: 'error
message: (error as Error).message
;{
{
{

} private async checkMemory(): Promise
} try
;()const memoryUsage = process.memoryUsage
;()const totalMemory = require('os').totalmem
;()const freeMemory = require('os').freemem
;const usedMemory = totalMemory - freeMemory
;const memoryUsagePercent = (usedMemory / totalMemory) * 100

;'const status = memoryUsagePercent > 90 ? 'warning' : 'healthy

} return
,status
,heapUsed: memoryUsage.heapUsed
,heapTotal: memoryUsage.heapTotal
,external: memoryUsage.external
,rss: memoryUsage.rss
,systemTotal: totalMemory
,systemUsed: usedMemory
usagePercent: Math.round(memoryUsagePercent * 100) / 100
;{
} catch (error) {
} return
```

```
,'status: 'error
message: (error as Error).message
;{
{
{

} private async checkDiskSpace(): Promise
} try
;const fs = require('fs').promises
;const path = require('path')

// فحص مساحة القرص في المجلد الحالي
;('.')const stats = await fs.stat
} <= const diskUsage = await new Promise((resolve, reject)
} <= require('child_process').exec('df -h .', (error: any, stdout: string)
;if (error) reject(error)
;else resolve(stdout)
;({
;({

// تحليل إخراج df
;const lines = (diskUsage as string).split('\n')
;const dataLine = lines[1]
;const parts = dataLine.split(/\s+/)

;const total = parts[1]
;const used = parts[2]
;const available = parts[3]
;const usagePercent = parseInt(parts[4])

;'const status = usagePercent > 90 ? 'warning' : usagePercent > 95 ? 'error' : 'healthy

} return
,status
,total
,used
,available
usagePercent
;{
} catch (error) {
```

```
} return
,'status: 'error
message: (error as Error).message
;{
{
{

} ()async cleanup
} if (this.checkInterval)
;clearInterval(this.checkInterval)
;this.checkInterval = null
{

;()await this.prisma.$disconnect
{
{

;()export default new HealthCheckService
```

سكريبت النشر: deploy.sh

```
bash
bin/bash/!#
```

سكريبت النشر لمنصة ZEREX CARBON

يضمن النشر الآمن والمتسق للتطبيق

```
set -e # إيقاف التنفيذ عند حدوث خطأ
```

الألوان للرسائل

```
'RED='\033[0;31m
'GREEN='\033[0;32m
'YELLOW='\033[1;33m
NC='\033[0m' # No Color
```

متغيرات البيئة

```
"PROJECT_NAME="zerex-carbon
"DOCKER_COMPOSE_FILE="docker-compose.yml
"ENV_FILE=".env.production
"BACKUP_DIR="/backup/$(date +%Y%m%d_%H%M%S)
```

وظائف مساعدة

```
} ()log_info
"echo -e "${GREEN}[INFO]${NC} $1
{

} ()log_warn
"echo -e "${YELLOW}[WARN]${NC} $1
{

} ()log_error
"echo -e "${RED}[ERROR]${NC} $1
{
```

التحقق من المتطلبات

```
} ()check_requirements
"التحقق من المتطلبات..." log_info

if ! command -v docker &> /dev/null; then
"Docker غير مثبت" log_error
exit 1
fi

if ! command -v docker-compose &> /dev/null; then
"Docker Compose غير مثبت" log_error
exit 1
fi

if [ ! -f "$ENV_FILE" ]; then
"ملف البيئة $ENV_FILE غير موجود" log_error
exit 1
fi

"جميع المتطلبات متوفرة" log_info
{
```

إنشاء نسخة احتياطية

```
} ()create_backup
"إنشاء نسخة احتياطية..." log_info

"mkdir -p "$BACKUP_DIR

# نسخ قاعدة البيانات
docker-compose exec -T postgres pg_dump -U zerex_user zerex_carbon >
""$BACKUP_DIR/database.sql
```

```
# نسخ الملفات المرفوعة
if [ -d "uploads" ]; then
"/cp -r uploads "$BACKUP_DIR
fi

# نسخ السجلات
if [ -d "logs" ]; then
"/cp -r logs "$BACKUP_DIR
fi

log_info "تم إنشاء النسخة الاحتياطية في: $BACKUP_DIR"
{
```

إيقاف الخدمات

```
} ()stop_services
log_info "إيقاف الخدمات..."
docker-compose down
{
```

بناء الصور

```
} ()build_images
log_info "بناء صور Docker..."
docker-compose build --no-cache
{
```

بدء الخدمات

```
} ()start_services
```

```
log_info "بدء الخدمات..."
docker-compose up -d

# انتظار بدء الخدمات
log_info "انتظار بدء الخدمات..."
sleep 30

# فحص صحة الخدمات
check_services_health
{
```

فحص صحة الخدمات

```
} ()check_services_health
log_info "فحص صحة الخدمات..."

# فحص قاعدة البيانات
if ! docker-compose exec -T postgres pg_isready -U zerex_user -d zerex_carbon; then
    log_error "قاعدة البيانات غير جاهزة"
    return 1
fi

# فحص Redis
if ! docker-compose exec -T redis redis-cli ping | grep -q PONG; then
    log_error "Redis غير جاهز"
    return 1
fi

# فحص التطبيق
local max_attempts=30
local attempt=1

while [ $attempt -le $max_attempts ]; do
    if curl -f http://localhost:3000/api/health > /dev/null 2>&1; then
        log_info "التطبيق جاهز"
        return 0
    fi
```

```
log_info "محاولة $attempt/$max_attempts - انتظار التطبيق..."
sleep 10
((++attempt))
done

log_error "التطبيق فشل في البدء"
return 1
{
```

تشغيل الهجرات

```
run_migrations() {
log_info "تشغيل هجرات قاعدة البيانات..."

docker-compose exec app npx prisma migrate deploy

log_info "تم تشغيل الهجرات بنجاح"
{
```

تنظيف الموارد غير المستخدمة

```
cleanup() {
log_info "تنظيف الموارد غير المستخدمة..."

docker system prune -f
docker volume prune -f

log_info "تم التنظيف"
{
```

إرسال إشعار النشر

```
} ()send_deployment_notification
local status=$1
""=local message

if [ "$status" = "success" ]; then
"(date)$ في بنجاح ZEREX CARBON تم نشر منصة ✅"=message
else
"$2 :(date)$ في ZEREX CARBON فشل في نشر منصة ❌"=message
fi

# إرسال إشعار (يمكن تخصيص هذا حسب النظام المستخدم)
"message$ :إشعار النشر" log_info

# مثال: إرسال إلى Slack
# curl -X POST -H 'Content-type: application/json' \
# --data "{\"text\":\"$message\"}" \
# YOUR_SLACK_WEBHOOK_URL
{
```

الدالة الرئيسية

```
} ()main
"...ZEREX CARBON بدء عملية النشر لمنصة" log_info

# التحقق من المتطلبات
check_requirements

# إنشاء نسخة احتياطية
create_backup

# إيقاف الخدمات الحالية
stop_services

# بناء الصور الجديدة
build_images
```

```
# بدء الخدمات
if start_services; then
# تشغيل الهجرات
run_migrations

# تنظيف الموارد
cleanup

# إرسال إشعار النجاح
"send_deployment_notification "success

log_info "تم النشر بنجاح! 🎉"
log_info "المنصة متاحة على: https://zerexcarbon.com"
else
# إعادة تشغيل الخدمات السابقة في حالة الفشل
log_warn "محاولة استعادة الخدمات السابقة..."

# يمكن إضافة منطق استعادة الخدمات السابقة هنا

# إرسال إشعار الفشل
"send_deployment_notification "error" "فشل في فحص صحة الخدمات"

log_error "فشل في النشر"
exit 1
fi
{
```

تشغيل الدالة الرئيسية

```
"@$" main
```

خاتمة

هذا الدليل الشامل يغطي جميع جوانب منصة ZEREX CARBON، من الواجهة الأمامية إلى الخادم الخلفي، ومن تطبيق الهاتف المحمول إلى النشر والإنتاج.

النقاط الرئيسية:

1. **التصميم المعماري**: استخدام Node.js + TypeScript للخادم الخلفي، Flutter للتطبيق المحمول، وPostgreSQL لقاعدة البيانات
2. **الأمان**: تطبيق معايير أمان عالية مع JWT، تشفير البيانات، وحماية من الهجمات الشائعة
3. **المراقبة**: نظام مراقبة شامل لتتبع الأداء والصحة العامة للنظام
4. **النشر**: عملية نشر آمنة مع Docker ونسخ احتياطية تلقائية
5. **التوافق**: دعم متعدد اللغات والمنصات مع امتثال GDPR

الملفات المرفقة:

- **zerex_carbon_guide_part1.md**: نظرة عامة وهيكل المشروع
- **zerex_carbon_guide_part2.md**: الواجهة الأمامية والكود المصدري
- **zerex_carbon_guide_part3.md**: الخادم الخلفي والمتحكمات
- **zerex_carbon_guide_part4.md**: قاعدة البيانات والخدمات
- **zerex_carbon_guide_part5.md**: تطبيق الهاتف المحمول
- **zerex_carbon_guide_part6.md**: النشر والإنتاج والأمان

هذا الدليل يخدم كمرجع شامل للمطورين والمهندسين الذين يعملون على منصة ZEREX CARBON، ويوفر فهماً عميقاً لجميع مكونات النظام وكيفية عملها معاً.
