# 🏗️ BacklineIT Platform — Architektúra Terv

**Dokumentum típus:** Architecture Plan
**Verzió:** 1.0.0
**Dátum:** 2025. december 15.
**Státusz:** Production
**Tulajdonos:** BacklineIT Solutions — Engineering Team

---

## 📋 Tartalomjegyzék

---

## 1. Vezetői Összefoglaló

### 1.1 Architektúra Filozófia

A BacklineIT Platform egy **cloud-native, serverless architektúrára** épülő modern webalkalmazás, amely a következő alapelveket követi:

| Alapelv | Megvalósítás |
|---|---|
| **Egyszerűség** | Monolitikus Next.js alkalmazás, minimális külső függőségek |
| **Skálázhatóság** | Serverless hosting (Vercel), auto-scaling képesség |
| **Biztonság** | Defense in depth, zero-trust network, OAuth 2.0 |
| **Sebesség** | Edge computing, SSR/SSG, agresszív caching |
| **Fenntarthatóság** | TypeScript, tiszta kódbázis, dokumentált API-k |
| **Költséghatékonyság** | Pay-as-you-go modellek, serverless funkciók |

### 1.2 Kulcs Döntések

**Miért Next.js 16?**

- Full-stack framework egy kódbázisban
- Automatikus code splitting és optimalizálás
- Built-in SSR/SSG/ISR támogatás
- Vercel platform natív integrációja

**Miért Serverless?**

- Nulla infrastruktúra menedzsment
- Automatikus skálázás terhelés alapján
- Költséghatékony kis/közepes terhelés mellett
- Globális edge network

**Miért Neon PostgreSQL?**

- Serverless, auto-scaling adatbázis
- PostgreSQL kompatibilitás (érett ökoszisztéma)
- Automatikus connection pooling
- Git-szerű branching development-hez

---

## 2. Rendszer Áttekintés

### 2.1 Magas Szintű Architektúra

**Három fő réteg:**

```
┌─────────────────────────────────────────────────────────────┐
│                    CLIENT LAYER                             │
│  - Web Browser (Desktop, Mobile, Tablet)                    │
│  - Future: PWA (Progressive Web App)                        │
└─────────────────────────────────────────────────────────────┘
                              ↓ HTTPS
┌─────────────────────────────────────────────────────────────┐
│                EDGE & APPLICATION LAYER                      │
│  ┌──────────────┐  ┌──────────────┐  ┌──────────────┐       │
│  │ Cloudflare   │→│  Vercel Edge │→│   Next.js    │       │
│  │ CDN + WAF    │  │   Network    │  │  App Router  │       │
│  └──────────────┘  └──────────────┘  └──────────────┘       │
└─────────────────────────────────────────────────────────────┘
                              ↓
┌─────────────────────────────────────────────────────────────┐
│                   DATA & SERVICES LAYER                      │
│  ┌──────────────┐  ┌──────────────┐  ┌──────────────┐       │
│  │  Neon DB     │  │     n8n      │  │  External    │       │
│  │ PostgreSQL   │  │  Automation  │  │  Services    │       │
│  └──────────────┘  └──────────────┘  └──────────────┘       │
└─────────────────────────────────────────────────────────────┘
```

### 2.2 Komponens Kapcsolatok

| Komponens | Kapcsolódik → | Protokoll/Módszer | Cél |
|---|---|---|---|
| **Web Browser** | Cloudflare | HTTPS (TLS 1.3) | DNS, SSL, DDoS védelem |
| **Cloudflare** | Vercel Edge | HTTPS | Edge caching, routing |
| **Vercel Edge** | Next.js App | Internal | Request handling |
| **Next.js App** | Neon DB | Prisma (PostgreSQL wire protocol) | Adatbázis műveletek |

| | | | |
|---|---|---|---|
| **Next.js App** | n8n | HTTPS Webhook | Automatizációs triggerek |
| **Next.js App** | SimplePay | HTTPS API | Fizetés inicializálás |
| **SimplePay** | Next.js App | HTTPS Webhook (IPN) | Fizetés visszaigazolás |
| **n8n** | Számlázz.hu | HTTPS API | Számla generálás |
| **n8n** | SMTP | SMTP/TLS | Email küldés |

### 2.3 Adatfolyam Típusok

**1. Syncrhon (Real-time):**

- Felhasználói interakciók (form submit, navigáció)
- Adatbázis lekérdezések (< 100ms)
- API válaszok

**2. Asynchron (Background):**

- Email küldés (n8n)
- Számla generálás
- Analytics reporting
- Backup folyamatok

**3. Event-driven:**

- Webhook triggerek (SimplePay IPN, Ticket létrehozás)
- n8n workflow indítások
- Real-time frissítések

---

## 3. Alkalmazás Architektúra

### 3.1 Frontend Architektúra

**Technológiai Stack:**

| Réteg | Technológia | Verzió | Felelősség |
|---|---|---|---|
| **Framework** | Next.js | 16.x | App Router, SSR/SSG, API Routes |
| **UI Library** | React | 19.x | Komponens-alapú UI |
| **Language** | TypeScript | 5.x | Type safety |
| **Styling** | Tailwind CSS | 4.x | Utility-first CSS framework |
| **Components** | Radix UI | Latest | Headless UI primitives |
| **Animation** | Framer Motion | 12.x | Deklaratív animációk |
| **Icons** | Lucide React | Latest | SVG ikonkészlet |
| **Forms** | React Hook Form | 7.x | Form state management |
| **Validation** | Zod | 4.x | Schema validation |

**Komponens Architektúra:**

```
src/components/
├── ui/                    # Atomic komponensek (Button, Input, Card)
│   ├── button.tsx
│   ├── input.tsx
│   └── card.tsx
├── layout/                # Layout komponensek
│   ├── header.tsx
│   ├── footer.tsx
│   └── navigation.tsx
├── sections/              # Page sections (Hero, Features, CTA)
│   ├── hero.tsx
│   ├── features.tsx
│   └── testimonials.tsx
├── ecommerce/             # E-commerce specifikus
│   ├── cart-button.tsx
│   ├── product-card.tsx
│   └── checkout-form.tsx
├── dashboard/             # Dashboard komponensek
│   ├── stats-card.tsx
│   ├── ticket-list.tsx
│   └── license-table.tsx
├── templates/             # Page templates
│   ├── service-layout.tsx
│   └── blog-layout.tsx
└── analytics/             # Analytics komponensek
    ├── google-tags.tsx
    └── vercel-analytics.tsx
```

**Rendering Stratégia:**

| Oldal Típus | Rendering | Indoklás |
|---|---|---|
| **Landing Page** | SSG (Static) | Ritkán változik, max sebesség |
| **Blog Posts** | SSG + ISR | On-demand revalidation új postnál |
| **Termék Katalógus** | SSR | Dinamikus árak, készlet |
| **Dashboard** | CSR | Felhasználó-specifikus, auth required |
| **Admin Panel** | CSR | Real-time adatok, auth required |
| **Checkout** | SSR | SEO nem számít, de gyors betöltés igen |

## 3.2 Backend Architektúra

**Next.js App Router Struktúra:**

```
src/app/
├── [locale]/               # Internationalization wrapper
│   ├── page.tsx           # Homepage (SSG)
│   ├── layout.tsx          # Root layout (Auth provider, Theme)
│   ├── szolgaltatasok/     # Service pages (SSG)
│   ├── termekek/           # Products (SSR)
│   ├── blog/               # Blog (SSG + ISR)
│   ├── dashboard/          # User portal (CSR, protected)
│   ├── admin/              # Admin panel (CSR, protected)
│   ├── checkout/           # Checkout flow (SSR)
│   └── payment/            # Payment callbacks
│
├── api/                    # API Routes
│   ├── auth/               # Auth.js routes
│   ├── webhooks/           # External webhooks (SimplePay, n8n)
│   ├── tickets/            # Ticket CRUD
│   ├── products/           # Product CRUD (admin)
│   └── admin/              # Admin API endpoints
│
├── sitemap.ts              # Dynamic sitemap generation
├── robots.ts               # Robots.txt
└── not-found.tsx           # 404 page
```

**Server Actions:**

Next.js Server Actions használata form submission-höz és data mutation-höz:

```
// Példa: Ticket létrehozás Server Action
'use server'

export async function createTicket(formData: FormData) {
  const session = await auth();
  if (!session) throw new Error('Unauthorized');

  const data = ticketSchema.parse({
    subject: formData.get('subject'),
    description: formData.get('description'),
    // ...
  });

  const ticket = await db.ticket.create({ data });

  // Trigger n8n webhook
  await fetch(process.env.N8N_WEBHOOK_URL!, {
    method: 'POST',
    body: JSON.stringify({ event: 'ticket_created', ticket })
  });

  return ticket;
}
```

## 3.3 State Management

**Stratégia:**

| Típus | Megoldás | Használat |
|---|---|---|
| **Server State** | React Query | API adatok cache-elése, auto-refetch |
| **Client State** | React Context | Theme, language, auth state |
| **Form State** | React Hook Form | Form inputs, validation |
| **URL State** | Next.js Router | Pagination, filters, search |
| **Local Storage** | localStorage API | Kosár, user preferences |
| **Session Storage** | sessionStorage | Checkout flow state |

**Nincs globális state management library (Redux, Zustand) - egyszerűség kedvéért.**

---

## 4. Adatbázis Architektúra

### 4.1 Adatbázis Választás: Neon PostgreSQL

**Előnyök:**

- ✅ Serverless (auto-scaling, pay-per-use)
- ✅ PostgreSQL kompatibilitás (sql standard, mature)
- ✅ Git-style branching (dev/staging/prod ágak)
- ✅ Automatikus connection pooling
- ✅ Sub-100ms latency (edge compatible)

**Kapcsolódás:**

```
// lib/db.ts
import { PrismaClient } from '@prisma/client'

const globalForPrisma = global as unknown as { prisma: PrismaClient }

export const db = globalForPrisma.prisma || new PrismaClient({
  log: process.env.NODE_ENV === 'development' ? ['query', 'error'] : ['error'],
})

if (process.env.NODE_ENV !== 'production') globalForPrisma.prisma = db
```

**Connection String példa:**

```
DATABASE_URL="postgresql://user:pass@eu-central-1.neon.tech/backlineit?sslmode=require"
DIRECT_URL="postgresql://user:pass@eu-central-1.neon.tech/backlineit"
```

### 4.2 Schema Design Elvek

**Normalizáció:**

- 3NF (Third Normal Form) követése
- Redundancia minimalizálás
- Referential integrity (foreign keys)

**Indexelés Stratégia:**

| Tábla | Index Típus | Mezők | Indoklás |
|---|---|---|---|
| **User** | Unique | email | Gyors auth lookup |
| **User** | Unique | referralCode | Affiliate link resolution |
| **Order** | Unique | orderRef | Rendelés azonosítás |
| **Order** | B-tree | userId, createdAt | User rendeléseinek lekérése |
| **License** | Unique | key | Licenc validáció |
| **Ticket** | Unique | ticketNumber | Ticket lookup |
| **Ticket** | B-tree | userId, status | User ticket listázás |
| **Product** | Unique | slug | URL-based lookup |

**Query Optimalizálás:**

```
// ❌ Rossz: N+1 query probléma
const orders = await db.order.findMany();
for (const order of orders) {
  const items = await db.orderItem.findMany({ where: { orderId: order.id } });
}

// ✅ Jó: Prisma include használata
const orders = await db.order.findMany({
  include: {
    items: {
      include: {
        product: true
      }
    }
  }
});
```

### 4.3 Adatbázis Migráció

**Prisma Migrate Workflow:**

```
# 1. Schema változtatás (schema.prisma szerkesztése)

# 2. Migráció generálás
npx prisma migrate dev --name add_referral_system

# 3. Production deploy
npx prisma migrate deploy
```

**Migráció Best Practices:**

- ✅ Mindig backward compatible migration
- ✅ Explicit default values új mezőknél
- ✅ Rollback terv minden migration-höz
- ✅ Staging környezetben tesztelés először

### 4.4 Adatbázis Biztonsági Megfontolások

| Fenyegetés | Védelem |
|---|---|
| **SQL Injection** | Prisma ORM (parameterized queries) |
| **Data Breach** | Encrypted at rest (Neon native), TLS in transit |
| **Unauthorized Access** | Row Level Security (majdani feature), application-level auth |
| **Data Loss** | Automatikus daily backups (Neon), point-in-time recovery |

---

## 5. API Design és Integráció

### 5.1 API Struktúra

**Next.js API Routes vs Server Actions:**

| Használat | Megoldás | Példa |
|---|---|---|
| **Form Submission** | Server Actions | Ticket létrehozás, user update |
| **External Webhooks** | API Routes | SimplePay IPN, n8n callbacks |
| **Public API** | API Routes | `/api/products`, `/api/blog` |
| **Admin CRUD** | Server Actions | Termék szerkesztés, user kezelés |

### 5.2 Webhook Integráció

**Bejövő Webhookok:**

| Forrás | Endpoint | Esemény | Akció |
|---|---|---|---|
| **SimplePay** | `/api/webhooks/simplepay/ipn` | Fizetés sikeres | Rendelés státusz frissítés, licenc generálás, n8n trigger |
| **SimplePay** | `/api/webhooks/simplepay/ipn` | Fizetés sikertelen | Rendelés cancel, értesítés |
| **n8n** | `/api/webhooks/n8n/email-received` | Email érkezett | Ticket létrehozás email-ből |

**Webhook Security:**

```
// SimplePay IPN signature validation
import crypto from 'crypto';

function validateSimplePaySignature(data: any, signature: string): boolean {
  const secret = process.env.SIMPLEPAY_SECRET_KEY!;
  const hash = crypto
    .createHmac('sha256', secret)
    .update(JSON.stringify(data))
    .digest('hex');

  return hash === signature;
}

export async function POST(req: Request) {
  const signature = req.headers.get('X-SimplePay-Signature');
  const data = await req.json();

  if (!validateSimplePaySignature(data, signature)) {
    return new Response('Invalid signature', { status: 401 });
  }

  // Process webhook...
}
```

**Kimenő Webhookok (n8n triggerek):**

| Esemény | n8n Workflow | Akció |
|---|---|---|
| **user.registered** | Email Verification | Verification email küldés |
| **ticket.created** | Ticket Notification | Admin email + user visszaigazolás |
| **order.completed** | Order Fulfillment | Számla generálás, licenc email |
| **newsletter.subscribed** | Newsletter Welcome | Üdvözlő email + lead export |

### 5.3 Külső API Integrációk

**SimplePay Payment Gateway:**

```
// lib/simplepay.ts
export async function createPaymentSession(order: Order) {
  const response = await fetch('https://api.simplepay.hu/payment/start', {
    method: 'POST',
    headers: {
      'Content-Type': 'application/json',
      'Authorization': `Bearer ${process.env.SIMPLEPAY_API_KEY}`
    },
    body: JSON.stringify({
      merchant: process.env.SIMPLEPAY_MERCHANT_ID,
      orderRef: order.orderRef,
      currency: 'HUF',
      total: order.totalAmount / 100, // fillérből forintra
      customer: {
        email: order.customerEmail,
        name: order.customerName
      },
      methods: ['CARD'],
      url: {
        success: `${process.env.NEXTAUTH_URL}/payment/success`,
        fail: `${process.env.NEXTAUTH_URL}/payment/fail`,
        cancel: `${process.env.NEXTAUTH_URL}/payment/cancel`,
        timeout: `${process.env.NEXTAUTH_URL}/payment/timeout`
      }
    })
  });

  const data = await response.json();
  return data.paymentUrl;
}
```

**Számlázz.hu Integration (via n8n):**

n8n workflow kezeli (nem direct integration az app-ból):

1. n8n fogadja az `order.completed` webhook-ot
2. n8n meghívja Számlázz.hu API-t XML formátumban
3. n8n visszaküldi a PDF számlát email-ben

---

# 6. Biztonsági Architektúra

## 6.1 Autentikáció és Autorizáció

**Auth.js (NextAuth.js v5) Setup:**

```
// lib/auth.ts
import NextAuth from "next-auth"
import Google from "next-auth/providers/google"
import GitHub from "next-auth/providers/github"
import Credentials from "next-auth/providers/credentials"
import { PrismaAdapter } from "@auth/prisma-adapter"
import { db } from "./db"
import bcrypt from "bcryptjs"

export const { handlers, auth, signIn, signOut } = NextAuth({
  adapter: PrismaAdapter(db),
  providers: [
    Google({
      clientId: process.env.GOOGLE_CLIENT_ID!,
      clientSecret: process.env.GOOGLE_CLIENT_SECRET!
    }),
    GitHub({
      clientId: process.env.GITHUB_ID!,
      clientSecret: process.env.GITHUB_SECRET!
    }),
    Credentials({
      credentials: {
        email: { label: "Email", type: "email" },
        password: { label: "Password", type: "password" }
      },
      authorize: async (credentials) => {
        const user = await db.user.findUnique({
          where: { email: credentials.email as string }
        });

        if (!user || !user.password) return null;

        const valid = await bcrypt.compare(
          credentials.password as string,
          user.password
        );

        return valid ? user : null;
      }
    })
  ],
  callbacks: {
    session: async ({ session, token }) => {
      if (token.sub) {
        const user = await db.user.findUnique({
          where: { id: token.sub }
        });
        session.user.role = user?.role || 'USER';
      }
      return session;
    }
  },
  pages: {
    signIn: '/login',
    error: '/login'
  }
})
```

**Role-Based Access Control (RBAC):**

| Szerepkör | Hozzáférés |
|---|---|
| **GUEST** | Publikus oldalak, termékek böngészése |
| **USER** | Dashboard, licencek, ticket létrehozás |
| **ADMIN** | Admin panel, összes user/order/ticket kezelés |

**Middleware-based Protection:**

```
// middleware.ts
import { auth } from '@/lib/auth'
import { NextResponse } from 'next/server'

export default auth((req) => {
  const { pathname } = req.nextUrl
  const isAdmin = req.auth?.user?.role === 'ADMIN'

  // Admin only routes
  if (pathname.startsWith('/admin') && !isAdmin) {
    return NextResponse.redirect(new URL('/login', req.url))
  }

  // Protected routes
  if (pathname.startsWith('/dashboard') && !req.auth) {
    return NextResponse.redirect(new URL('/login', req.url))
  }

  return NextResponse.next()
})

export const config = {
  matcher: ['/dashboard/:path*', '/admin/:path*']
}
```

### 6.2 Adatvédelem és GDPR

**Személyes Adatok Kezelése:**

| Adat Típus | Tároló Hely | Megőrzési Idő | Törlés Módja |
|---|---|---|---|
| **Email cím** | Neon DB (User tábla) | Account törlésig | Soft delete, 30 nap után hard delete |
| **Jelszó** | Neon DB (bcrypt hash) | Account törlésig | GDPR request alapján |
| **IP cím** | Vercel logs | 30 nap | Automatikus |
| **Session cookie** | Browser | Session vége | Browser zárásakor |
| **Analitika** | Google Analytics | 14 hónap | GA setting szerint |

**GDPR Compliance Features:**

- ✅ Cookie Banner (Consent management)
- ✅ Adatvédelmi tájékoztató oldal ( `/adatvedelem` )
- ✅ User data export funkció (admin panel)
- ✅ Account törlés lehetőség (soft delete)
- ✅ Email opt-out minden marketing emailben

### 6.3 Security Headers

**Next.js next.config.ts:**

```
const securityHeaders = [
  {
    key: 'X-DNS-Prefetch-Control',
    value: 'on'
  },
  {
    key: 'Strict-Transport-Security',
    value: 'max-age=63072000; includeSubDomains; preload'
  },
  {
    key: 'X-Frame-Options',
    value: 'SAMEORIGIN'
  },
  {
    key: 'X-Content-Type-Options',
    value: 'nosniff'
  },
  {
    key: 'X-XSS-Protection',
    value: '1; mode=block'
  },
  {
    key: 'Referrer-Policy',
    value: 'strict-origin-when-cross-origin'
  },
  {
    key: 'Permissions-Policy',
    value: 'camera=(), microphone=(), geolocation=()'
  }
]

export default {
  async headers() {
    return [
    {
      source: '/:path*',
      headers: securityHeaders
    }
    ]
  }
}
```

### 6.4 Rate Limiting

**Cloudflare Rate Limiting:**

- 100 req/min per IP a `/api/*` végpontokon
- 10 req/min per IP a `/api/webhooks/*` végpontokon (csak allowlist IP-k)

**Application-level Rate Limiting:**

```
// lib/rate-limit.ts
import { Ratelimit } from "@upstash/ratelimit"
import { Redis } from "@upstash/redis"

const ratelimit = new Ratelimit({
  redis: Redis.fromEnv(),
  limiter: Ratelimit.slidingWindow(10, "10 s"),
})

export async function checkRateLimit(identifier: string) {
  const { success } = await ratelimit.limit(identifier)
  return success
}
```

---

## 7. Deployment Architektúra

### 7.1 Hosting: Vercel Platform

**Deployment Konfiguráció:**

| Környezet | Branch | Domain | Purpose |
|---|---|---|---|
| **Production** | `main` | `backlineit.hu` | Éles környezet |
| **Preview** | `feat/*` | `*.vercel.app` | Feature preview |
| **Development** | `dev` | `dev.backlineit.hu` | Integration testing |

**vercel.json:**

```json
{
  "buildCommand": "prisma generate && next build",
  "installCommand": "npm install",
  "framework": "nextjs",
  "regions": ["fra1"],
  "env": {
    "DATABASE_URL": "@database_url",
    "NEXTAUTH_SECRET": "@nextauth_secret"
  }
}
```

### 7.2 CI/CD Pipeline

**GitHub Actions Workflow (.github/workflows/ci.yml):**

```yaml
name: CI/CD Pipeline

on:
  push:
    branches: [main, dev]
  pull_request:
    branches: [main]

jobs:
  test:
    runs-on: ubuntu-latest
    steps:
      - uses: actions/checkout@v3
      - uses: actions/setup-node@v3
        with:
          node-version: '20'
          cache: 'npm'

      - name: Install dependencies
        run: npm ci

      - name: Type check
        run: npm run type-check

      - name: Lint
        run: npm run lint

      - name: Test
        run: npm run test

  deploy-preview:
    needs: test
    if: github.ref != 'refs/heads/main'
    runs-on: ubuntu-latest
    steps:
      - uses: actions/checkout@v3
      - uses: amondnet/vercel-action@v20
        with:
          vercel-token: ${{ secrets.VERCEL_TOKEN }}
          vercel-org-id: ${{ secrets.ORG_ID}}
          vercel-project-id: ${{ secrets.PROJECT_ID}}
```

**Automatic Deployment (Vercel):**

- Push to `main` → Automatic production deploy
- Push to `dev` → Automatic staging deploy
- Open PR → Automatic preview deploy + comment with URL

## 7.3 Environment Variables

**Környezeti Változók Kezelése:**

```
# Development (.env.local)
DATABASE_URL="postgresql://..."
NEXTAUTH_URL="http://localhost:3000"
NEXTAUTH_SECRET="dev-secret-key"

# Production (Vercel Dashboard)
DATABASE_URL="postgresql://prod@..."
NEXTAUTH_URL="https://backlineit.hu"
NEXTAUTH_SECRET="<secure-random-string>"
SIMPLEPAY_API_KEY="<encrypted>"
```

**Secrets Management:**

- ✅ Vercel Environment Variables (encrypted at rest)

- ✅ Külön értékek (dev/staging/prod)
- ✅ Automatikus injection build time-ban
- ✅ No secrets in git ( `.env.local` in `.gitignore` )

### 7.4 CDN és Edge Network

**Cloudflare Integration:**

| Feature | Konfiguráció |
|---|---|
| **DNS** | Cloudflare Nameservers |
| **SSL/TLS** | Full (strict) mode, Auto HTTPS Rewrites |
| **Caching** | Cache Level: Standard, Browser TTL: 4 hours |
| **Security** | WAF enabled, Bot Fight Mode, DDoS protection |
| **Page Rules** | Cache Everything for `/blog/*`, `/szolgaltatasok/*` |

**Vercel Edge Network:**

- Automatic deployment to 20+ global regions
- Smart routing to nearest edge location
- Image optimization at edge (next/image)

---

## 8. Skálázhatósági Stratégia

### 8.1 Horizontal Scaling (Serverless Auto-scaling)

**Vercel Serverless Functions:**

- Automatikus skálázás concurrent request alapján
- Cold start: ~100-300ms (Next.js)
- Warm: ~10-50ms
- Max concurrent: 1000+ (Pro plan)

**Bottleneck Elemzés:**

| Komponens | Skálázási Limit | Mitigáció |
|---|---|---|
| **Next.js Functions** | 1000 concurrent | Vercel auto-scale, cache növelés |
| **Neon DB** | 100 connections | Prisma connection pooling, PgBouncer |
| **n8n** | 100 req/s | Queue system, batch processing |
| **SimplePay** | 50 req/s | Retry logic, exponential backoff |

### 8.2 Vertical Scaling (Performance Optimization)

**Database Query Optimization:**

```
// Példa: Pagination large datasets
async function getOrdersPaginated(userId: string, page: number = 1, limit: number = 20) {
  const skip = (page - 1) * limit;

  const [orders, total] = await Promise.all([
    db.order.findMany({
      where: { userId },
      skip,
      take: limit,
      orderBy: { createdAt: 'desc' },
      select: {
        id: true,
        orderRef: true,
        totalAmount: true,
        status: true,
        createdAt: true,
        _count: {
          select: { items: true }
        }
      }
    }),
    db.order.count({ where: { userId } })
  ]);

  return { orders, total, pages: Math.ceil(total / limit) };
}
```

### 8.3 Caching Stratégia

**Multi-Layer Caching:**

```
┌──────────────────┐
│  Browser Cache   │  → Static assets (immutable), 1 year
└──────────────────┘
         ↓
┌──────────────────┐
│ Cloudflare CDN   │  → HTML pages (public), 1 hour
└──────────────────┘
         ↓
┌──────────────────┐
│  Vercel Edge     │  → ISR pages, API responses, 60s
└──────────────────┘
         ↓
┌──────────────────┐
│  React Query     │  → Client-side API cache, 5 min
└──────────────────┘
         ↓
┌──────────────────┐
│  Database        │  → Source of truth
└──────────────────┘
```

**Cache Invalidation:**

```
// ISR Revalidation (Next.js)
export const revalidate = 60 // seconds

// On-demand revalidation
import { revalidatePath } from 'next/cache'

export async function updateProduct(id: string, data: any) {
  await db.product.update({ where: { id }, data });

  // Invalidate product page cache
  revalidatePath(`/termekek/${data.slug}`)
  revalidatePath('/termekek')
}
```

---

## 9. Monitoring és Logging

### 9.1 Application Performance Monitoring (APM)

**Sentry Integration:**

```
// sentry.client.config.ts
import * as Sentry from "@sentry/nextjs";

Sentry.init({
  dsn: process.env.NEXT_PUBLIC_SENTRY_DSN,
  environment: process.env.NODE_ENV,
  tracesSampleRate: process.env.NODE_ENV === 'production' ? 0.1 : 1.0,
  replaysOnErrorSampleRate: 1.0,
  replaysSessionSampleRate: 0.1,

  beforeSend(event, hint) {
    // Filter out sensitive data
    if (event.request) {
      delete event.request.cookies;
      delete event.request.headers;
    }
    return event;
  }
});
```

**Monitoring Dashboard:**

| Metrika | Tool | Alert Threshold |
|---|---|---|
| **Error Rate** | Sentry | > 1% |
| **Response Time (p99)** | Vercel Analytics | > 3s |
| **Uptime** | Vercel | < 99.5% |

|  |  |  |
|---|---|---|
| **Database Latency** | Neon Dashboard | > 200ms |
| **API Success Rate** | Sentry | < 95% |

### 9.2 Logging Architecture

**Structured Logging (Winston):**

```
// lib/logger.ts
import winston from 'winston';

export const logger = winston.createLogger({
  level: process.env.NODE_ENV === 'production' ? 'info' : 'debug',
  format: winston.format.combine(
    winston.format.timestamp(),
    winston.format.errors({ stack: true }),
    winston.format.json()
  ),
  defaultMeta: { service: 'backlineit-app' },
  transports: [
    new winston.transports.Console({
      format: winston.format.combine(
        winston.format.colorize(),
        winston.format.simple()
      )
    })
  ]
});

// Usage
logger.info('Order created', { orderId, userId, amount });
logger.error('Payment failed', { error, orderId });
```

**Log Aggregation:**

- Vercel Logs (built-in, 7 days retention)
- Sentry Breadcrumbs (30 days)
- Optional: Logtail / Datadog (long-term retention)

### 9.3 Analytics és Business Intelligence

**Google Analytics 4:**

- Page views, user journeys
- Conversion tracking (product purchase, ticket creation)
- Custom events (button clicks, form submissions)

**Vercel Analytics:**

- Web Vitals (LCP, FID, CLS)
- Real User Monitoring (RUM)
- Geographic distribution

**Custom Analytics (Prisma):**

```
// Daily stats aggregation (cron job via Vercel Cron)
export async function generateDailyStats(date: Date) {
  const stats = await db.$queryRaw`
    SELECT
      COUNT(DISTINCT user_id) as new_users,
      COUNT(*) as total_orders,
      SUM(total_amount) as revenue,
      AVG(total_amount) as avg_order_value
    FROM "Order"
    WHERE created_at::date = ${date}
  `;

  await db.dailyStats.create({ data: stats });
}
```

---

## 10. Backup és Disaster Recovery

### 10.1 Adatbázis Backup Stratégia

**Neon Automated Backups:**

- **Frequency**: Teljes backup naponta 02:00 UTC
- **Retention**: 30 nap rolling
- **Point-in-Time Recovery (PITR)**: Bármely időpontra az elmúlt 7 napból

**Manual Backup Trigger:**

```
# pg_dump via Neon connection
pg_dump "postgresql://user:pass@neon.tech/backlineit" \
  --format=custom \
  --file=backup-$(date +%Y%m%d).dump
```

### 10.2 Disaster Recovery Plan

**Recovery Time Objective (RTO): 1 óra**
**Recovery Point Objective (RPO): 24 óra**

**Disaster Scenarios:**

| Szcenárió | Probability | Recovery Steps |
|---|---|---|
| **Vercel Outage** | Alacsony | Automatic failover Vercel edge-en, kommunikáció |
| **Neon DB Failure** | Nagyon alacsony | Restore from last backup, inform users |
| **Data Corruption** | Közepes | PITR to last known good state |
| **Accidental Delete** | Közepes | Soft delete recovery, vagy backup restore |
| **Security Breach** | Alacsony | Revoke credentials, security audit, user notification |

**Runbook (DB Restore):**

```
# 1. Letöltés legutóbbi backup
neon branches create --name recovery-$(date +%Y%m%d)

# 2. Restore backup to new branch
neon pg-restore --branch recovery-$(date +%Y%m%d) backup.dump

# 3. Verify data integrity
psql $RECOVERY_DATABASE_URL -c "SELECT COUNT(*) FROM \"User\""

# 4. Cutover (update DATABASE_URL in Vercel)
vercel env add DATABASE_URL production < new_connection_string

# 5. Redeploy
vercel --prod
```

### 10.3 File Storage Backup

**Vercel Blob Storage:**

- Replikálás 3 lokációban (auto)
- No manual backup needed

**Critical Files (Code):**

- Git repository (GitHub) → distributed backup
- Vercel deployment artifacts (30 days retention)

---

## 11. Teljesítmény Optimalizálás

### 11.1 Frontend Optimalizálás

**Code Splitting:**

```
// Lazy loading komponensek
import dynamic from 'next/dynamic'

const HeavyChart = dynamic(() => import('@/components/heavy-chart'), {
  loading: () => <Skeleton />,
  ssr: false // Client-side only
})
```

**Image Optimization:**

```
import Image from 'next/image'

<Image
  src="/hero.jpg"
  width={1200}
  height={600}
  alt="BacklineIT Hero"
  priority // LCP image
  placeholder="blur"
  blurDataURL="data:image/..." // LQIP
/>
```

**Font Optimization:**

```
// app/layout.tsx
import { Inter } from 'next/font/google'

const inter = Inter({
  subsets: ['latin', 'latin-ext'],
  display: 'swap',
  variable: '--font-inter'
})
```

### 11.2 Backend Optimalizálás

**Database Query Optimization:**

- ✅ Indexes létrehozása gyakori query-khez
- ✅ Select only needed fields
- ✅ Use pagination
- ✅ Avoid N+1 queries (use `include` )

**API Response Compression:**

```
// middleware.ts
export function middleware(request: NextRequest) {
  const response = NextResponse.next()

  // Enable compression for API routes
  if (request.nextUrl.pathname.startsWith('/api/')) {
    response.headers.set('Content-Encoding', 'gzip')
  }

  return response
}
```

### 11.3 Target Performance Metrics

| Metrika | Target | Mérés |
|---|---|---|
| **LCP (Largest Contentful Paint)** | < 2.5s | Lighthouse, Vercel Analytics |
| **FID (First Input Delay)** | < 100ms | Real User Monitoring |
| **CLS (Cumulative Layout Shift)** | < 0.1 | Lighthouse |
| **TTFB (Time to First Byte)** | < 600ms | Server Response Time |
| **Bundle Size (JS)** | < 200KB (gzipped) | `next build` analysis |

**Performance Budget:**

```
// next.config.ts
const withBundleAnalyzer = require('@next/bundle-analyzer')({
  enabled: process.env.ANALYZE === 'true'
})

module.exports = withBundleAnalyzer({
  experimental: {
    optimizePackageImports: ['@radix-ui/react-icons']
  },
  webpack: (config, { isServer }) => {
    if (!isServer) {
      config.optimization.concatenateModules = true
    }
    return config
  }
})
```

---

## 12. Fejlesztői Workflow

### 12.1 Development Environment Setup

**Követelmények:**

- Node.js 20+
- pnpm / npm
- PostgreSQL (local vagy Neon dev branch)
- Git

**Quick Start:**

```
# 1. Clone repository
git clone https://github.com/backlineit/platform.git
cd platform

# 2. Install dependencies
npm install

# 3. Setup environment
cp .env.example .env.local
# Edit .env.local with your credentials

# 4. Setup database
npx prisma generate
npx prisma migrate dev

# 5. Seed data (optional)
npm run db:seed
```

```
# 6. Start dev server
npm run dev
```

### 12.2 Git Workflow

**Branch Strategy (GitHub Flow):**

```
main (production)
  ↑
  └── feat/user-profile-page
  └── fix/checkout-validation
  └── chore/update-dependencies
```

**Commit Convention (Conventional Commits):**

```
feat: add user profile page
fix: resolve checkout validation error
chore: update Next.js to 16.1
docs: update API documentation
style: format code with prettier
refactor: extract cart logic to hook
test: add unit tests for product service
```

**Pull Request Checklist:**

- [ ] Code review by 1+ developer
- [ ] All tests passing
- [ ] No TypeScript errors
- [ ] Lighthouse score > 90
- [ ] Updated documentation (if needed)

### 12.3 Code Quality Tools

**Tooling Setup:**

| Tool | Purpose | Config File |
|---|---|---|
| **ESLint** | Linting | `.eslintrc.json` |
| **Prettier** | Formatting | `.prettierrc` |
| **TypeScript** | Type checking | `tsconfig.json` |
| **Husky** | Git hooks | `.husky/pre-commit` |
| **Lint-staged** | Staged files linting | `package.json` |

**Pre-commit Hook:**

```
#!/bin/sh
# .husky/pre-commit
```

```
npm run type-check
npm run lint
npm run format
```

### 12.4 Testing Strategy

**Test Pyramid:**

```
          /\
         /  \       E2E Tests (Playwright) - 10%
        /────\
       /  \   \     Integration Tests (Vitest) - 30%
      /────────\
     /  \   \   \   Unit Tests (Vitest) - 60%
    /────────────\
```

**Példa Unit Test:**

```
// __tests__/lib/cart.test.ts
import { describe, it, expect } from 'vitest'
import { calculateTotal, applyDiscount } from '@/lib/cart'

describe('Cart calculations', () => {
  it('should calculate total correctly', () => {
    const items = [
      { price: 1000, quantity: 2 },
      { price: 500, quantity: 1 }
    ]
    expect(calculateTotal(items)).toBe(2500)
  })

  it('should apply 10% discount', () => {
    expect(applyDiscount(1000, 0.1)).toBe(900)
  })
})
```

---

## 📊 Összefoglalás

### Architektúra Kiemelt Jellemzői

✅ **Cloud-Native**: Serverless, auto-scaling, pay-as-you-go
✅ **Security-First**: OAuth 2.0, encryption, GDPR compliance
✅ **Performance-Optimized**: Edge network, caching, SSR/SSG
✅ **Developer-Friendly**: TypeScript, modern tooling, clear structure
✅ **Cost-Effective**: No infrastructure overhead, efficient resource usage
✅ **Scalable**: Handles 1,000+ concurrent users, horizontal scaling ready

### Következő Lépések

1. **Monitoring Dashboard Setup** → Sentry + Vercel Analytics integráció finomítása
2. **Load Testing** → Artillery / k6 terhelési tesztek 1000+ concurrent user-rel

3. **Security Audit** → Penetration testing, OWASP Top 10 ellenőrzés
4. **Performance Baseline** → Lighthouse CI setup, performance regression alerts
5. **Documentation** → API dokumentáció (Swagger/OpenAPI), architectural decision records (ADR-ek)

---

*Készítette: Engineering Team — Architecture Guild*
*Jóváhagyta: CTO*
*Utolsó felülvizsgálat: 2025. december 15.*