WritGo Application Architecture

Version: 2.0
Last Updated: 18 december 2024
Status: In Progress - Route Consolidation

Overview

WritGo is een AI-powered content platform voor WordPress sites. De applicatie helpt gebruikers met content planning, generatie, en distributie via een moderne Next.js interface.

Technology Stack

Frontend

- **Framework:** Next.js 14 (App Router)
- **Language:** TypeScript
- **Styling:** Tailwind CSS
- **UI Components:** shadcn/ui
- **State Management:** React Hooks + Server Components
- **Authentication:** NextAuth.js

Backend

- **API:** Next.js API Routes (Edge Runtime ready)
- **Database:** PostgreSQL (Prisma ORM)
- **File Storage:** Supabase Storage
- **AI:** Abacus.AI + OpenAI APIs
- **WordPress Integration:** REST API

Deployment

- **Platform:** Vercel (recommended) / Docker
- **Database:** Supabase / Railway
- **CDN:** Vercel Edge Network
- **Monitoring:** Vercel Analytics

Application Structure

```
writgo/
├── nextjs_space/                     # Main Next.js application
│   ├── app/                          # Next.js App Router
│   │   ├── (simplified)/             # ✅ New simplified interface (client)
│   │   │   ├── layout.tsx            # Simplified layout with dark theme
│   │   │   ├── page.tsx              # Dashboard
│   │   │   ├── blog/                 # Blog management
│   │   │   ├── content/              # Content creation
│   │   │   ├── platforms/            # Platform connections
│   │   │   └── account/              # Account settings
│   │   │
│   │   ├── (simplified-admin)/       # 🔄 Planned admin interface
│   │   │   └── ... (Phase 3.2)
│   │   │
│   │   ├── api/                      # API Routes
│   │   │   ├── simplified/           # ✅ Consolidated API (19 routes)
│   │   │   ├── admin/                # Admin API (155 routes)
│   │   │   ├── client/               # Client API (267 routes)
│   │   │   └── ...                   # Other specialized APIs
│   │   │
│   │   ├── admin/                    # ⚠️ Legacy admin interface (70 pages)
│   │   ├── dashboard/                # ⚠️ Legacy dashboard
│   │   ├── client-dashboard/         # ⚠️ Legacy client dashboard
│   │   └── ...                       # Other legacy routes
│   │
│   ├── components/                   # React Components
│   │   ├── SimplifiedLayout.tsx      # ✅ Main layout
│   │   ├── SimplifiedNavigation.tsx  # ✅ Navigation
│   │   ├── ui/                       # shadcn/ui components (49)
│   │   ├── dashboard/                # ⚠️ Legacy dashboard components
│   │   ├── admin/                    # Admin components (42)
│   │   └── ...                       # Other components (259 total)
│   │
│   ├── lib/                          # Shared utilities
│   │   ├── db.ts                     # Prisma client
│   │   ├── auth-options.ts           # NextAuth config
│   │   ├── aiml-api.ts               # AI integrations
│   │   ├── services/                 # Business logic
│   │   │   └── wordpress-content-fetcher.ts
│   │   └── ...
│   │
│   ├── prisma/                       # Database schema
│   │   └── schema.prisma             # Prisma schema
│   │
│   └── public/                       # Static assets
│
└── docs/                             # Documentation
    ├── ARCHITECTURE.md               # This file
    ├── API_DOCUMENTATION.md          # API docs
    ├── API_INVENTORY.md              # API route inventory
    ├── COMPONENT_CONSOLIDATION_STRATEGY.md
    ├── ADMIN_INTERFACE_STRATEGY.md
    ├── MIGRATION_STRATEGY.md         # Route migration guide
    ├── CHANGES.md                    # Change log
    └── ...
```

Architecture Patterns

1. Route Structure

Simplified Interface (New)

```
/(simplified)/
  - Clean, modern dark theme interface
  - Mobile-first responsive design
  - Minimal navigation (8 core items)
  - Consistent dark theme with orange accents
  - Server-side authentication
```

Legacy Interfaces (Deprecated)

```
/admin/*            - 70 pages (legacy admin)
/dashboard/*        - Legacy dashboard
/client-dashboard/* - Legacy client dashboard
```

Migration Status:
- ✅ 17 routes migrated to simplified (7%)
- ⏳ ~190+ legacy routes remaining
- 🎯 Target: 80-100 core routes in simplified

2. API Architecture

Three-Layer API Structure

Layer 1: Simplified API (Preferred)

```
/api/simplified/*
  - Consolidated, consistent endpoints
  - Standard error handling
  - Input validation (Zod)
  - TypeScript types
  - Current: 19 routes
  - Target: 40-50 routes
```

Layer 2: Client/Admin APIs (Legacy)

```
/api/client/*  - 267 routes (client-specific)
/api/admin/*   - 155 routes (admin-specific)
```

Layer 3: Specialized APIs (Keep)

```
/api/financien/*      - Financial module
/api/cron/*           - Scheduled jobs
/api/ai-agent/*       - AI integrations
/api/content-hub/*    - Content management
```

3. Component Architecture

Component Hierarchy

```
SimplifiedLayout (Root)
  └── SimplifiedNavigation (Sidebar)
      ├── NavigationItem (Links)
      └── UserMenu (Account)

Page Components
  └── Card (shadcn/ui)
      ├── CardHeader
      └── CardContent
          └── Business Logic
```

Component Categories

Core Components (Keep & Maintain)
- `SimplifiedLayout.tsx` - Main layout
- `SimplifiedNavigation.tsx` - Navigation
- `ui/*` - shadcn/ui components (49)

Specialized Components (Keep)
- `blog/*` - Blog-specific (12)
- `chat/*` - Chat interface (19)
- `email-marketing/*` - Email tools (5)

Deprecated Components (Phase Out)
- `dashboard/*` - Legacy dashboard (18)
- `admin-complex/*` - Legacy admin (4)
- `client-dashboard/*` - Legacy client (3)
- `dashboard-client/*` - Legacy client (4)

Total Component Consolidation:
- Before: 259 components (37+ duplicates)
- Target: ~220 components (remove ~30 duplicates)

4. Database Architecture

Prisma Schema Overview

Core Tables:
- `Client` - User accounts
- `Project` - WordPress sites
- `Post` - Generated content
- `SocialPost` - Social media posts

Relationship Pattern:

```
Client (1) ──> (N) Project
Project (1) ──> (N) Post
Client (1) ──> (N) SocialPost
```

Key Features:
- UUID primary keys
- Timestamps (createdAt, updatedAt)

- Soft deletes (isActive flags)
- JSON fields for flexible data

5. Authentication & Authorization

NextAuth.js Configuration

Session Strategy:

```
session: {
  strategy: "jwt",
  maxAge: 30 * 24 * 60 * 60, // 30 days
}
```

Supported Providers:
- Email/Password (credentials)
- Google OAuth
- Email Magic Links (planned)

Authorization Levels:

```
type UserRole = 'client' | 'admin' | 'superadmin';
```

RBAC Implementation:

```
// middleware.ts
if (path.startsWith('/admin')) {
  requireRole(['admin', 'superadmin']);
}

if (path.startsWith('/simplified')) {
  requireAuth();
}
```

6. AI Integration

AI Services Used

Content Generation:
- Abacus.AI AIML API (primary)
- OpenAI GPT-4 (fallback)
- Claude 3 (planned)

Image Generation:
- DALL-E 3
- Stable Diffusion
- Midjourney API (planned)

Video Generation:
- Vadoo.tv API
- Luma AI (planned)

SEO Analysis:
- DataForSEO API
- Google Search Console API

7. WordPress Integration

WordPress REST API

Endpoints Used:

```
/wp-json/wp/v2/posts      - Posts management
/wp-json/wp/v2/media      - Media upload
/wp-json/wp/v2/categories - Categories
/wp-json/wp/v2/users      - Author info
```

Authentication:
- Application Passwords
- OAuth (planned)

Features:
- Auto-publish to WordPress
- Sitemap parsing
- Content sync
- Media management

Data Flow

Content Generation Flow

```
User Input (Topic)
    ↓
[API: /api/simplified/generate]
    ↓
AI Service (Content Generation)
    ↓
Database (Save Draft)
    ↓
[Optional: Review & Edit]
    ↓
[API: /api/simplified/publish/wordpress]
    ↓
WordPress Site (Published)
```

WordPress Content Sync Flow

```
WordPress Site (Sitemap)
    ↓
[Service: wordpress-content-fetcher]
    ↓
Parse Sitemap
    ↓
Fetch Post Data (REST API)
    ↓
Store in Database
    ↓
Display in Simplified Interface
```

Dashboard Stats Flow

```
[API: /api/simplified/stats]
    ↓
Fetch from Multiple Sources:
  - Database (Generated content)
  - WordPress (Published content)
  - Social Media (Posts)
    ↓
Aggregate & Calculate
    ↓
Return to Frontend
    ↓
Display in Dashboard
```

Security Considerations

1. Authentication

- Session-based with JWT
- HTTP-only cookies
- CSRF protection
- Rate limiting on login

2. API Security

- All routes require authentication
- Input validation (Zod schemas)
- SQL injection prevention (Prisma)
- XSS protection (React escaping)

3. Data Privacy

- GDPR compliance
- Data encryption at rest
- Secure password hashing (bcrypt)
- PII data handling

4. WordPress Security

- Application passwords (not main password)
- Encrypted credential storage
- API key rotation
- Rate limiting

Performance Optimizations

1. Frontend

- Server Components (reduce JS bundle)
- Image optimization (next/image)
- Code splitting (dynamic imports)
- Route caching

2. API

- Edge runtime where possible
- Database connection pooling
- Query optimization (Prisma)
- Response caching

3. Database

- Indexed fields
- Efficient queries
- Connection pooling
- Read replicas (planned)

Deployment Architecture

Production Setup

```
┌─────────────────┐
│   Vercel Edge   │  ← CDN & Edge Functions
└────────┬────────┘
         │
┌────────▼────────┐
│   Next.js App   │  ← Application Server
└────────┬────────┘
         │
    ┌────┴────┐
    │         │
┌───▼──┐  ┌───▼───┐
│ DB   │  │ Files │
│ PG   │  │ S3    │
└──────┘  └───────┘
```

Environment Variables

Required:
- `DATABASE_URL` - PostgreSQL connection
- `NEXTAUTH_SECRET` - Session encryption
- `NEXTAUTH_URL` - App URL
- `AIML_API_KEY` - AI API key

Optional:
- `GOOGLE_CLIENT_ID` - OAuth
- `GOOGLE_CLIENT_SECRET` - OAuth
- `SUPABASE_URL` - File storage
- `SUPABASE_KEY` - File storage

Migration Status

Completed (Phase 1-2)

- ✅ Simplified interface structure
- ✅ Blog route migration
- ✅ Platforms page

- ✅ Account page
- ✅ 10 redirects
- ✅ Dark theme implementation
- ✅ 19 simplified API routes
- ✅ Component analysis
- ✅ Admin strategy

In Progress (Phase 3-4)

- 🔄 Admin interface planning
- 🔄 Component consolidation
- 🔄 Documentation
- 🔄 Testing

Planned (Future)

- ⏳ Complete admin migration
- ⏳ Remove deprecated components
- ⏳ Performance optimizations
- ⏳ Advanced AI features

Development Guidelines

1. New Features

- Always use simplified routes
- Follow dark theme guidelines
- Use TypeScript strictly
- Add API documentation

2. Code Standards

- ESLint + Prettier
- TypeScript strict mode
- React best practices
- Component documentation

3. Testing

- Unit tests (Jest)
- Integration tests (Playwright)
- E2E tests (Cypress)
- Manual testing checklist

4. Git Workflow

- Feature branches
- Pull request reviews
- Commit message standards
- Changelog updates

Monitoring & Observability

Metrics to Track

- API response times
- Error rates
- User engagement
- Content generation stats
- WordPress publish success rate

Tools

- Vercel Analytics (frontend)
- Prisma Studio (database)
- LogTail (logs)
- Sentry (error tracking - planned)

Future Improvements

Short-term (1-3 months)

1. Complete admin interface migration
2. Remove deprecated components
3. Improve mobile experience
4. Add more AI models
5. Better error handling

Long-term (3-6 months)

1. Multi-language support
2. Advanced analytics
3. Team collaboration features
4. Custom AI model training
5. Enterprise features

Conclusion

WritGo is transitioning from a complex multi-interface application to a **clean, consolidated simplified interface** with specialized modules for advanced features. The architecture supports this transition through clear separation of concerns, consistent patterns, and gradual migration paths.

Current Progress: ~7% migrated to simplified interface
Target: 100% core functionality in simplified interface
Timeline: Gradual migration over next 3-6 months

For detailed migration plans, see:
- MIGRATION_STRATEGY.md (./MIGRATION_STRATEGY.md)
- ADMIN_INTERFACE_STRATEGY.md (./ADMIN_INTERFACE_STRATEGY.md)
- COMPONENT_CONSOLIDATION_STRATEGY.md (./COMPONENT_CONSOLIDATION_STRATEGY.md)