WRITGO.NL REFACTORING PLAN

==============================

STATUS: READY FOR EXECUTION

Datum: 16 december 2025

EXECUTIVE SUMMARY

- **Totale bestanden**: 886 TypeScript/TSX bestanden
- **Client API Routes**: 275 routes
- **Potentially Unused**: 375 van 582 routes (64%)
- **Doel**: Reduceren tot ~150-200 goed gestructureerde routes

FASE 1: VERWIJDER DUIDELIJK ONGEBRUIKTE CODE ✓

1.1 Backup & Unused Directories

Te verwijderen:

```
❌ app/api/client/chat/_conversation_unused/
❌ app/api/client/chat/_conversations_backup/
```

Impact: Geen - expliciet gemarkeerd als unused/backup

1.2 Duplicate Late.dev Routes

Situatie: 4 verschillende directories voor dezelfde functionaliteit:
- `app/api/client/getlate/` (9 references)
- `app/api/client/late-dev/` (15 references)
- `app/api/client/latedev/` (7 references) ← MEEST COMPLEET
- `app/api/client/latedev-config/` (config file)

Actie:
1. **Behoud:** `app/api/client/latedev/` (meest complete feature set)
2. **Verwijder:** `getlate/`, `late-dev/`, `latedev-config/`
3. **Update:** Alle references naar `/api/client/latedev/`

Reden: Latedev heeft callback, disconnect, invite, post - meest complete set

1.3 Duplicate Bolcom Search Routes

Situatie: 3 verschillende implementaties:
- `app/api/client/bolcom/search/`

- `app/api/client/bolcom/search-products/`
- `app/api/client/bolcom/ai-search/`

Actie:
1. Analyseer welke het meest gebruikt wordt
2. Consolideer naar 1-2 routes (ai-search + basic search)
3. Verwijder duplicate implementaties

FASE 2: CONSOLIDEER SIMPLIFIED ↔ CLIENT OVERLAP ✓

2.1 Content Planning

Overlapping routes:

Client:
- GET `/api/client/content-plan/route.ts`
- POST `/api/client/content-plan/add-ideas/route.ts`
- POST `/api/client/content-plan/refresh/route.ts`

Simplified:
- GET/POST `/api/simplified/content-plan/route.ts` ✓ BETER
- POST `/api/simplified/content-plan/analyze-wordpress/route.ts` ✓

Beslissing:
- **Behoud:** Simplified routes (gecombineerde GET/POST is efficiënter)
- **Actie:** Voeg client-specifieke features toe aan simplified routes
- **Redirect:** Client routes → Simplified implementatie

2.2 Stats & Analytics

Overlapping:
- `/api/client/stats/route.ts` (Client-specific stats)
- `/api/simplified/stats/route.ts` (Simplified stats)

Beslissing:
- **Behoud beide** maar merge implementatie
- Client route kan extra admin features hebben
- Simplified blijft gebruiksvriendelijk

2.3 Social Media

Probleem: 38 client social media routes vs 4 simplified routes!

Client routes groepen:
1. Posts management (10 routes) → Consolideer naar 3-4 routes
2. Ideas & topics (6 routes) → Consolideer naar 2 routes
3. Configuration (8 routes) → Consolideer naar 2-3 routes
4. Publishing (4 routes) → Consolideer naar 1-2 routes
5. Analytics & scheduling (10 routes) → Consolideer naar 3-4 routes

Target: 38 routes → 12-15 goed gestructureerde routes

2.4 Projects Management

Overlapping:
- 21 client project routes
- 2 simplified project routes

Consolidatie strategie:

```
Basis CRUD: 2 routes (GET/POST combined)
├── /api/client/projects/route.ts (list, create)
├── /api/client/projects/[id]/route.ts (get, update, delete)

Specifieke acties: ~8 routes
├── /api/client/projects/[id]/wordpress/route.ts
├── /api/client/projects/[id]/knowledge/route.ts
├── /api/client/projects/[id]/sitemap/route.ts
├── /api/client/projects/[id]/rescan/route.ts
├── /api/client/projects/[id]/collaborators/route.ts
├── /api/client/projects/[id]/content-strategy/route.ts
└── ... (keep only actively used)
```

Target: 21 routes → 10-12 routes

FASE 3: HERORGANISEER API STRUCTUUR ✓

3.1 Nieuwe Directory Structuur

```
app/api/
├── client/                     # Client-facing API (hoofdgebruik)
│   ├── content/               # Content generation & management
│   │   ├── generate/          # All generation endpoints
│   │   ├── plan/              # Content planning (merged)
│   │   ├── library/           # Content library
│   │   └── research/          # Research tools
│   │
│   ├── projects/              # Project management (consolidated)
│   ├── social/                # Social media (consolidated)
│   ├── publishing/            # WordPress, WooCommerce publishing
│   ├── integrations/          # External integrations
│   │   ├── latedev/           # Social media scheduler
│   │   ├── bolcom/            # Product search
│   │   └── wordpress/         # WordPress API
│   │
│   └── user/                  # User-specific (profile, stats, etc.)
│
├── simplified/                # Keep for backwards compatibility
│   └── (redirect to client where possible)
│
├── admin/                     # Admin panel API (unchanged)
└── cron/                      # Background jobs (unchanged)
```

3.2 Naming Conventions

Gebruik RESTful principes:
- GET `/api/client/content` - List content

- POST `/api/client/content` - Create content
- GET `/api/client/content/[id]` - Get specific content
- PUT `/api/client/content/[id]` - Update content
- DELETE `/api/client/content/[id]` - Delete content

FASE 4: UPDATE FRONTEND REFERENCES ✓

4.1 Automated Find & Replace

```
# Update late-dev references
find app -type f \( -name "*.tsx" -o -name "*.ts" \) -exec sed -i 's|api/client/get-late|api/client/latedev|g' {} +
find app -type f \( -name "*.tsx" -o -name "*.ts" \) -exec sed -i 's|api/client/late-dev|api/client/latedev|g' {} +
```

4.2 Manual Updates Required

- Content plan routes update
- Project routes update
- Social media routes update

IMPLEMENTATIE VOLGORDE

Week 1: Opschoning ✓

1. ✅ Verwijder backup directories
2. ✅ Consolideer late-dev routes
3. ✅ Update alle references
4. ✅ Test basis functionaliteit

Week 2: Consolidatie ✓

1. Merge content-plan routes
2. Consolideer social media routes
3. Consolideer project routes
4. Update frontend calls

Week 3: Herstructurering ✓

1. Herorganiseer directory structuur
2. Implementeer redirect routes voor backwards compatibility
3. Update alle API documentatie
4. Performance testing

Week 4: Testing & Cleanup ✓

1. Volledige applicatie testing
2. Fix edge cases
3. Remove unused dependencies

4. Final documentation update

RISICO MITIGATIE

---

1. Breaking Changes

Strategie: Implementeer redirect routes

```
// Old route: /api/client/content-plan/route.ts
export async function GET(req: Request) {
  // Redirect to new unified route
  return NextResponse.redirect('/api/client/content/plan')
}
```

2. Database Dependencies

Check: Geen schema wijzigingen verwacht
Actie: Behoud alle database queries zoals ze zijn

3. External Webhooks

Check: Cron jobs en webhooks blijven ongewijzigd
Actie: Alleen frontend-facing routes worden aangepast

SUCCESS METRICS

Voor refactoring:
- 582 API routes totaal
- 275 Client routes
- 375 mogelijk ongebruikte routes (64%)

Na refactoring (target):
- ~200-250 API routes totaal
- ~120-150 Client routes (45% reductie)
- <50 ongebruikte routes (80% reductie)

Code quality:
- Betere route organisatie
- Consistent naming
- Minder duplicatie
- Beter onderhoudbaar

NEXT STEPS

---

1. ✅ Get approval voor plan
2. 🔄 Create feature branch: `refactor/api-consolidation`

3. ⏳ Start met Fase 1: Opschoning
4. ⏳ Incrementele commits per logische stap
5. ⏳ Testing na elke fase
6. ⏳ Merge naar main na volledige verificatie