

# CURSUS CONTENT INTEGRATIE PLAN

## Aansluiting op Bestaande DatingAssistant App

---

### ANALYSE: WAT JE AL HEBT

#### Relevante Bestaande Structuur

```
src/
├── app/
│   └── (waarschijnlijk) /cursussen of /modules route
├── components/
│   └── (waarschijnlijk) LesModule, Quiz, VideoPlayer, etc.
├── lib/
│   └── services voor data fetching
└── Database tabellen voor:
    └── modules, lessen, quizzes, user_progress, etc.
```

#### Bestaande Module Structuur (uit screenshots)

Je hebt al werkende modules:

- Module 1: Zelfkennis
- Module 2: Psychologie van Aantrekkelijkheid
- Module 3: Profieltekst optimalisatie

Met componenten voor:

- Video player (HeyGen integratie)
- Quizzen met scoring
- Reflectie opdrachten
- AI Coach (Iris) integratie
- Voortgang tracking

---

### WAT WE MOETEN BOUWEN

We hoeven **geen nieuwe code** te schrijven - alleen **content** die in je bestaande systeem past:

Component	Wat we maken	Format
Les Content	Teksten, titels, beschrijvingen	Database records of JSON
Video Scripts	Tekst voor HeyGen/Iris	Markdown
Video Slides	Achtergronden voor video	PNG afbeeldingen
Quiz Vragen	Multiple choice + feedback	Database records of JSON
Reflecties	Open vragen + prompts	Database records of JSON
AI Coach Context	Instructies voor Iris per les	Prompt teksten
Werkboek	Download materiaal	PDF/DOCX

## ❓ CRUCIALE VRAGEN

Om precies het juiste format te leveren:

### 1. Database of Hardcoded?

**Hoe laad je nu module content?**

typescript

```
// Optie A: Database (PostgreSQL/Neon)
const les = await db.query('SELECT * FROM lessen WHERE id = $1', [lesId]);

// Optie B: Hardcoded in code
const les = MODULES_DATA.profielfoto.les1;

// Optie C: JSON files
import lesData from '@/data/cursussen/profielfoto/les-1.json';

// Optie D: CMS/API
const les = await fetch('/api/cursussen/profielfoto/les-1');
```

### 2. Bestaand Schema?

**Heb je al een database schema voor cursussen?**

Bijvoorbeeld:

sql

```
-- Bestaat dit al?
CREATE TABLE cursussen (
    id SERIAL PRIMARY KEY,
    titel VARCHAR(255),
    beschrijving TEXT,
    ...
);

CREATE TABLE lessen (
    id SERIAL PRIMARY KEY,
    cursus_id INTEGER REFERENCES cursussen(id),
    titel VARCHAR(255),
    video_url VARCHAR(500),
    ...
);

CREATE TABLE quiz_vragen (
    id SERIAL PRIMARY KEY,
    les_id INTEGER REFERENCES lessen(id),
    vraag TEXT,
    opties JSONB,
    ...
);
```

### 3. Component Props?

**Wat verwachten je bestaande componenten?**

typescript

```
// Hoe ziet je LesModule component eruit?
interface LesProps {
    titel: string;
    videoUrl?: string;
    content: ContentBlock[];
    quiz?: QuizVraag[];
    // etc.
}

// Hoe ziet een quiz vraag eruit?
interface QuizVraag {
    id: string;
    vraag: string;
    opties: { id: string; tekst: string; correct: boolean }[];
    feedback?: { correct: string; incorrect: string };
}
```

## TWEE AANPAK OPTIES

### Optie A: "Plug & Play" Content Package

Ik lever alle content in een **universeel JSON format** dat je eenvoudig kunt importeren:

```
json

{
  "cursus": {
    "id": "profielfoto",
    "titel": "De Perfecte Profielfoto in 5 Stappen",
    "type": "gratis",
    "lessen": [
      {
        "id": "les-1",
        "titel": "De 3 Grote Foto-Fouten",
        "duur": "7 min",
        "video": {
          "script": "...",
          "slides": ["slide-1.png", "slide-2.png"]
        },
        "content": [...],
        "quiz": [...],
        "reflectie": [...],
        "aiCoach": {...}
      }
    ]
  }
}
```

Jij schrijft dan een simpel import script:

```
typescript

// scripts/import-cursus.ts
import cursusData from './profielfoto-cursus.json';
await importCursusToDatabase(cursusData);
```

### Optie B: Direct Database Format

Je deelt je bestaande database schema met mij, en ik lever:

- SQL INSERT statements
- Of Prisma seed data
- Of exact het format dat je ORM verwacht

## WAT IK NU NODIG HEB

Om het perfecte format te leveren, deel één van deze:

### Optie 1: Bestaand Voorbeeld (Makkelijkst)

Deel de data van een bestaande module, bijvoorbeeld:

- Module 1: Zelfkennis database records
- Of de JSON/TypeScript file met module data
- Of een API response van /api/modules/1

### Optie 2: Database Schema

```
sql  
-- Export van je relevante tabellen  
\d cursussen  
\d lessen  
\d quiz_vragen  
-- etc.
```

### Optie 3: TypeScript Interfaces

```
typescript  
// Je bestaande types voor modules/lessen  
interface Module { ... }  
interface Les { ... }  
interface Quiz { ... }
```

## VOORLOPIG PLAN

Zonder de exacte specs ga ik uit van een pragmatische aanpak:

### Stap 1: Ik bouw nu

1. Complete content in gestructureerd JSON
2. HeyGen video scripts in Markdown
3. Slide designs als PPTX + PNG
4. Werkboek als DOCX

### Stap 2: Jij past aan

1. Review de content
2. Map naar jouw database schema
3. Import in je app

### Stap 3: We itereren

1. Wat werkt niet? → Ik pas format aan
  2. Templates verfijnen
  3. Volgende cursus bouwen
- 

## ACTIE NU

Kies één:

- A) "Bouw maar, ik map het zelf naar mijn database" → Ik lever universeel JSON + alle assets
  - B) "Wacht, ik deel eerst mijn schema/voorbeelden" → Je stuurt me database schema of bestaande module data
  - C) "Laten we simpel starten met alleen scripts + slides" → Ik focus eerst op HeyGen content, database later
- 

## MIJN AANBEVELING

Gezien je **productie-ready app** met **50+ tabellen**, heb je waarschijnlijk al een goed doordacht schema.

**Meest efficiënt:**

1. Deel je `lessen` en `quiz_vragen` table schema (of Prisma model)
2. Ik lever content in exact dat format
3. Jij doet `INSERT` of `prisma db seed`
4. Klaar!

Wat wil je doen?