Technisch ontwerp

Project	:	The Kembit Times
Opdrachtgever:		Mike Wijnen
Auteur	:	Christian Schroth
Datum	:	13-05-2025
Versie	:	1.0

1. Inleiding

The Kembit Times is een platform dat zowel automatisch verzamelde nieuwsberichten als door gebruikers geschreven artikelen samenbrengt. Bezoekers kunnen actuele berichten uit verschillende bronnen bekijken, terwijl geregistreerde gebruikers eigen nieuwscontent kunnen creëren en publiceren. Het doel is een centraal nieuwsportaal te bieden dat zowel informatief als interactief is.

2. Oplossing in kaart

We gaan een nieuwe website maken waar iedereen recent nieuwsberichten van meerdere genres kan lezen. Bovendien wordt het ook mogelijk om een account aan te maken en daarna zelf een nieuwsbericht te schrijven. Deze nieuwsberichten worden ook voor iedereen zichtbaar gemaakt. De site wordt geoptimaliseerd voor zowel desktop als mobiele apparaten. Daarnaast maken we gebruik van een snel backend-system om een snelle laadtijd en hoge prestatie te waarborgen.

3. Architectuurontwerp

De applicatie bestaat uit een React-frontend die communiceert met een ASP.NET Core API. De API raadpleegt een SQL Server-database voor het ophalen, opslaan en verwerken van nieuwsberichten, gebruikersgegevens en door gebruikers gegenereerde content.

Een Feeder-module verzamelt externe nieuwsartikelen en verwerkt deze automatisch in de database.

Gebruikte technologieën

- **Frontend:** React + TypeScript
- **Backend:** ASP.NET Core
- **Database:** SQL Server
- **ORM / Data-access:** Dapper
- **Feeder:** externe nieuwsverwerking
- **Extra functies:** ondersteuning voor gebruikerscontent, accountbeheer

Componenten en communicatie

De backend volgt een gelaagde architectuur:

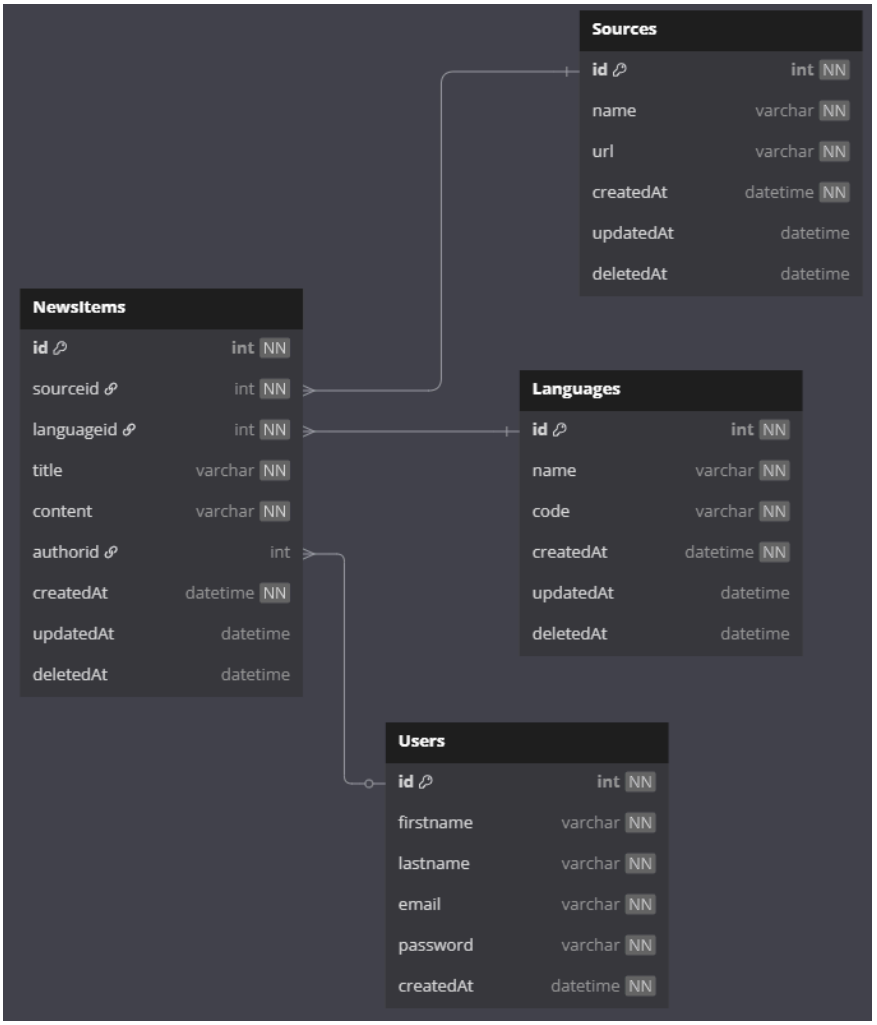
- **Controller Layer**
  Verantwoordelijk voor het ontvangen en valideren van HTTP-verzoeken en het terugsturen van API-responses.
- **Business Layer**
  Bevat de kern logica, zoals het verwerken van binnenkomende nieuwsfeeds, validatie van gebruikersinvoer en logica rond het aanmaken van gebruikers artikelen.
- **Repository Layer**
  Verzorgt alle database-interacties via Dapper en levert gegevens aan de Business Layer.

4. Databaseontwerp

ERD (Entity-Relationship Diagram):
Het datamodel bevat tabellen voor gebruikers, nieuwsartikelen, categorieën, talen en bronnen. Relaties zijn uitgewerkt in een ER-diagram dat is ontworpen in dbdiagram.io.

Belangrijkste tabellen

- **Sources**
  Informatie over nieuws leveranciers, zoals naam, type en URL.
- **Languages**
  Taalinstellingen voor nieuwsberichten (bijvoorbeeld naam, code).
- **News**
  Opslag van artikelen met informatie zoals titel, inhoud, HTML-versie, afbeeldingen, bron en taal.
- **Users**
  Gegevens van geregistreerde gebruikers..



- **Tabellen en relaties:**
  - Primary keys op alle ID-kolommen
  - Foreign keys sleutels:
    - Users.id → NewsItems.authorId
    - Languages.id → NewsItems.languageId
    - Sources.id → NewsItems.sourceId

5. API Ontwerp

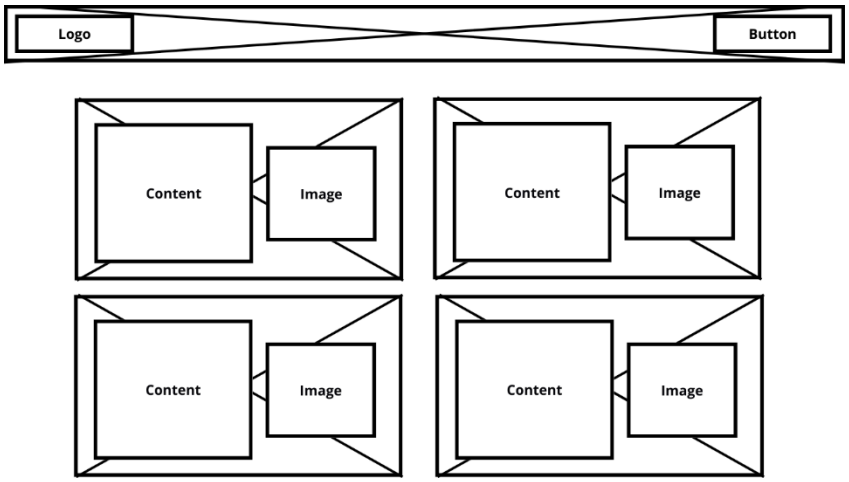
De API is ontwikkeld met ASP.NET Core en volgt een gelaagde architectuur om een duidelijke scheiding van verantwoordelijkheden te waarborgen:

- **Controller Layer:**
    - Ontvangt HTTP-requests van de frontend
    - Voert validatie uit
    - Retourneert responses in JSON
- **Business Layer:**
    - Regelt de logica rondom feed-verwerking, gebruikersacties en publicatie van artikelen
    - Coördineert bewerkingen met de Repository-laag
- **Repository Layer:**
    - Voert SQL-query's uit via Dapper
    - Geeft data terug aan de Business Layer
- **Voorbeeld-endpoints:**
    - GET /api/news – Ophalen van recente nieuwsberichten.
    - GET /api/news/{id} – Ophalen van één specifiek nieuwsbericht.
- **Foutafhandeling:**
    - Middleware registreert fouten en handelt uitzonderingen centraal.

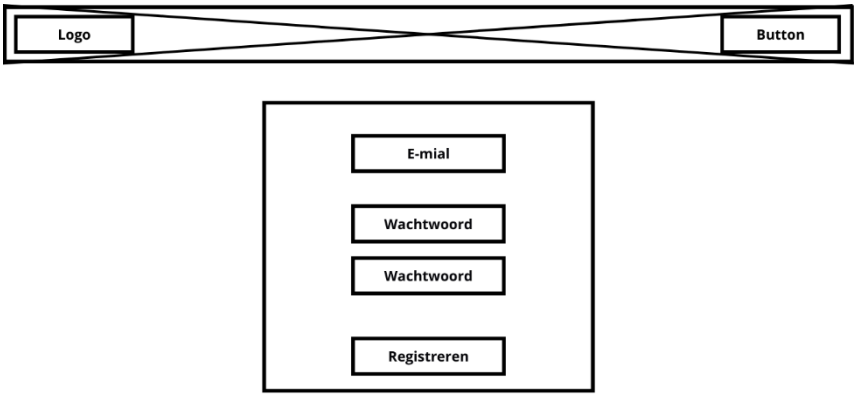
6. Frontend Ontwerp

- **Wireframe:**
  De wireframes tonen de structuur van de belangrijkste pagina's:

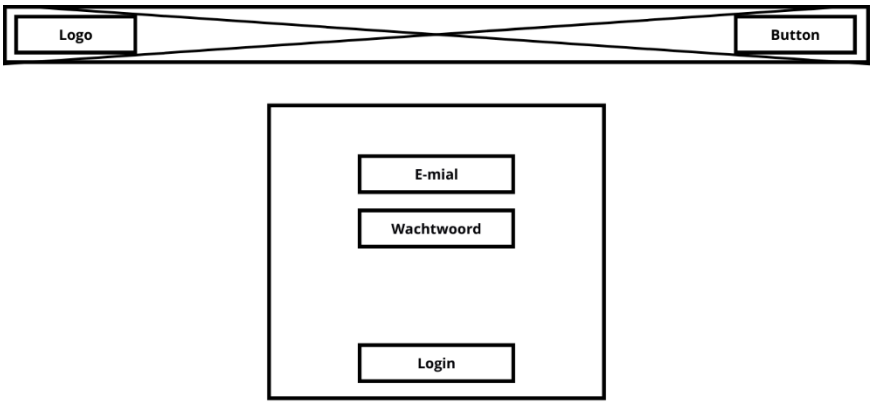
  Homepagina:



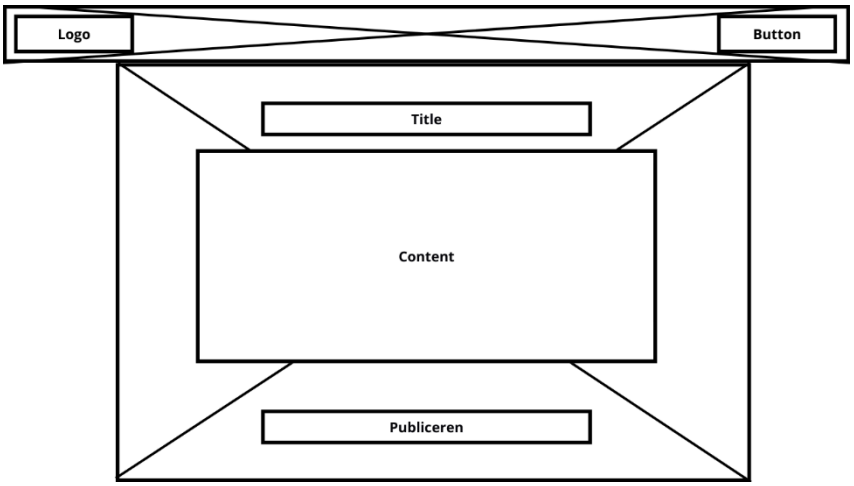
Registratiepagina:



Loginpagina:



Publiceer page (Pagina voor het schrijven van een nieuwsbericht)

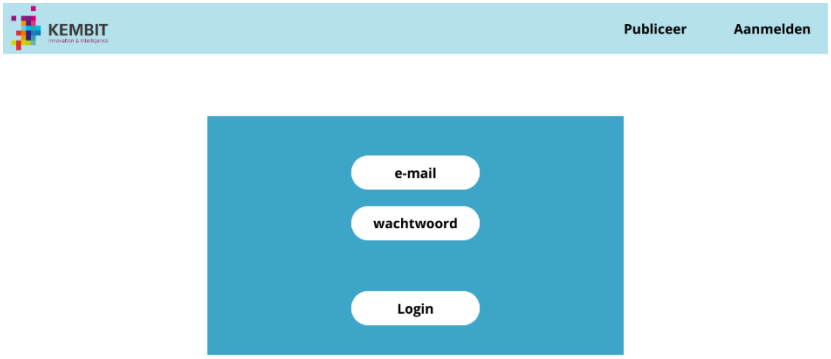


- **UI/UX**

  De interface benadrukt eenvoud en toegankelijkheid:

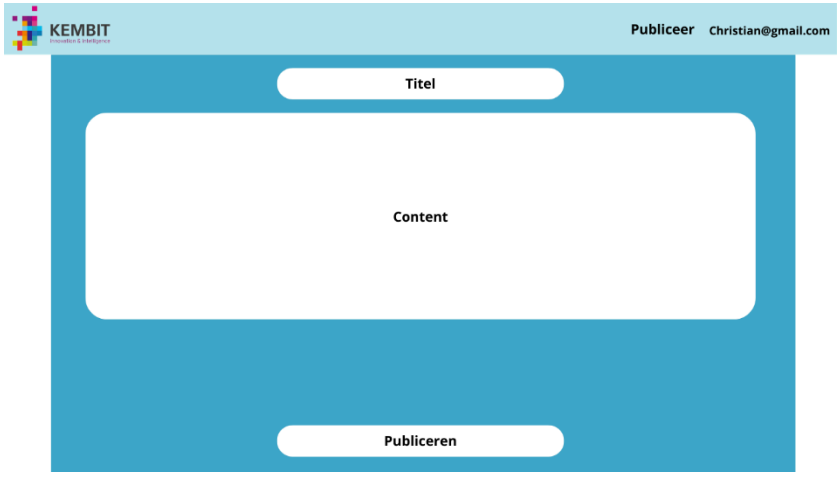
Homepagina:

Loginpagina:

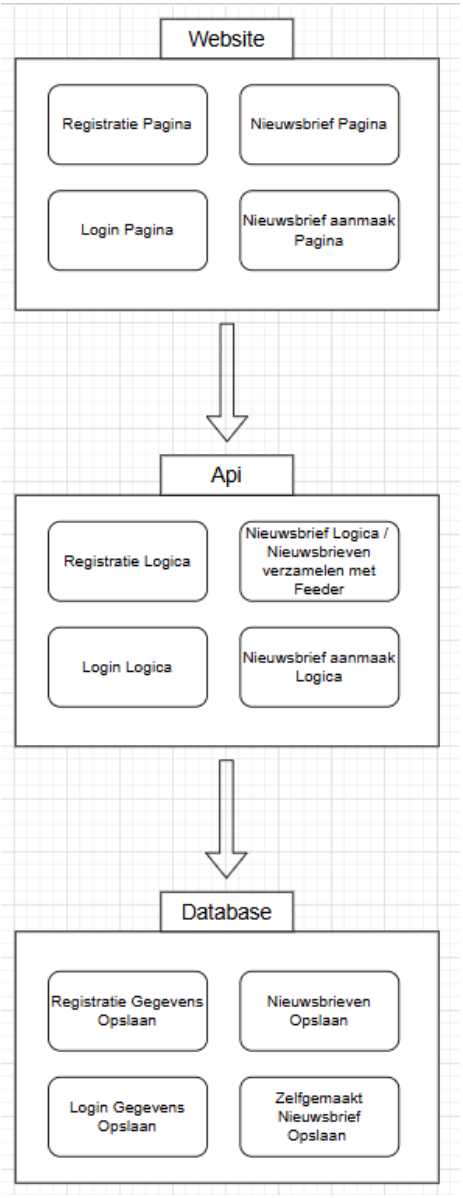


Registratiepagina:

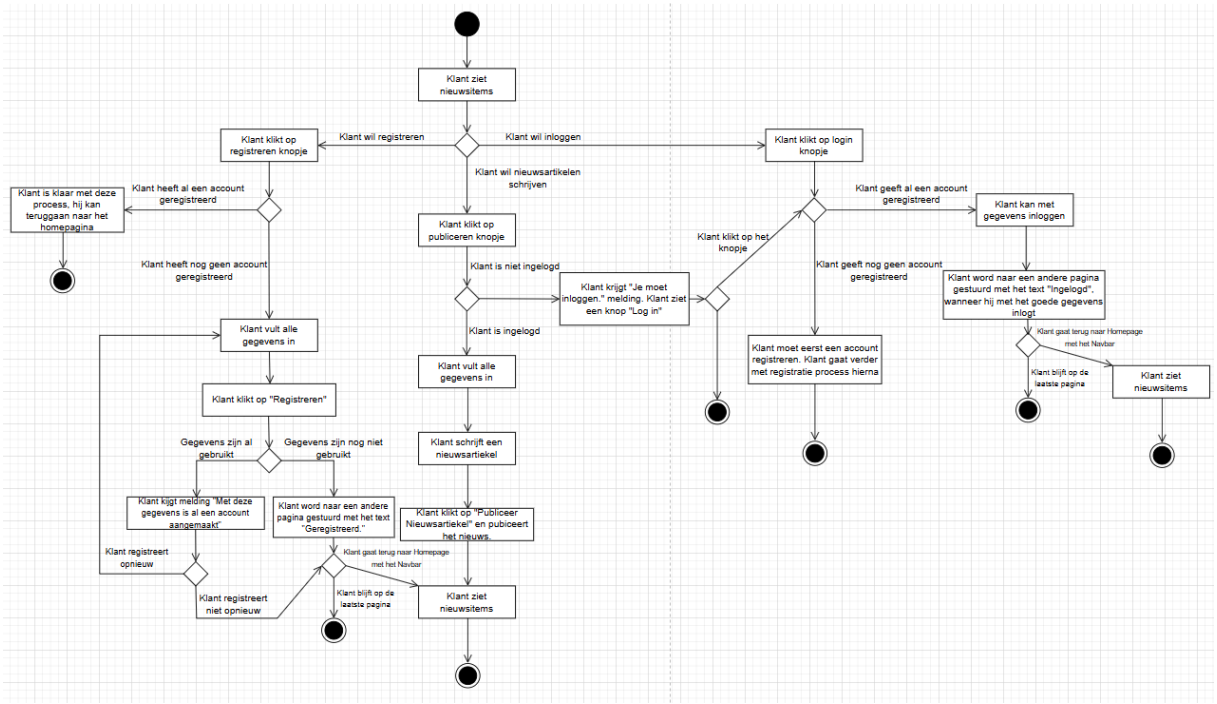
Nieuwsbrief aanmaak pagina:



- **Componenten Diagram**
  Componenten interactie binnen de frontend



- **Activiteiten Diagram**



7. User Story Traceability

User Story	Relatie met technisch ontwerp
Bezoeker wil nieuwsartikelen van meerdere bronnen zien	Feeder-module verzamelt externe nieuwsfeeds; API /api/news levert artikelen; frontend toont overzichtspagina.
Bezoeker wil artikelen in meerdere talen lezen	Database Languages koppelt taal aan artikelen; frontend toont taalkeuze via dropdown; API filtert per taal.

Gebruiker wil account registreren	Registratiepagina in frontend; API endpoint /users/register; data wordt veilig opgeslagen in Users-tabel.
Gebruiker wil inloggen	Loginpagina frontend; API endpoint /users/login; JWT-authenticatie beheert sessies.
Ingelogde gebruiker wil nieuws schrijven en publiceren	Frontend editor; API endpoint /api/news/post; Business Layer verwerkt en slaat artikelen op in News-tabel.
Bezoeker wil nieuws van andere gebruikers zien naast externe artikelen	Business Layer combineert externe feeds en gebruikersartikelen; frontend toont een gecombineerde lijst met auteur of bron.
Bezoeker wil artikelen automatisch per categorie zien	Categorieën-tabel; API verwerkt sortering; frontend groepeert artikelen per categorie.
Bezoeker wil veilige en gebruiksvriendelijke website	Validatie op frontend + backend; beveiligde opslag in SQL Server; UI/UX ontwerp zorgt voor toegankelijkheid.
Beheerder wil nieuwsbronnen kunnen beheren	Admin-API + CRUD-functionaliteit op Sources-tabel; alleen toegankelijk voor gebruikers met beheerdersrechten.
Website moet responsive en snel laden	Frontend is mobielvriendelijk; lazy loading; caching in API en database; minimale payloads.
Gebruiker wil bronvermelding en thema-optie	Frontend toont bron; CSS-thema's en user settings voor licht/donker en 3–4 visuele thema's.

8. Onderhoud en Schaalbaarheid

Dit hoofdstuk beschrijft hoe The KEMBIT Times onderhoudbaar, stabiel en schaalbaar blijft bij groei van gebruikers, nieuwsbronnen en content. Het laat zien welke maatregelen zijn genomen voor continuïteit, prestaties en uitbreidbaarheid.

Onderhoudsstrategie:

Onderdeel	**Beschrijving**
Code / Versiebeheer	Gebruik van Git, feature branches, pull requests en code reviews voor overzichtelijke ontwikkeling.
Testen	Unit tests voor backend en business logica; handmatige en geautomatiseerde UI-tests voor frontend.
Logging / monitoring	Middleware registreert fouten; real-time monitoring detecteert prestatieproblemen en verdachte activiteit.
Updates	Regelmatige updates van frontend, backend en dependencies om beveiliging en stabiliteit te waarborgen.

Schaalbaarheid:

Schaalbaarheid Aspect	Implementatie
Backend	Gebruik van Git, feature branches, pull requests en code reviews voor overzichtelijke ontwikkeling.
Database	SQL Server met indexen op veel gebruikte velden, partitioning en caching voor grote datasets.
Frontend	Lazy loading van componenten en mediabestanden, minimale payloads.
Hosting & infrastructuur	Azure App Services voor cloud hosting, gebruik van CDN's voor afbeeldingen en audiobestanden.

Performanceoptimalisaties:

- API-optimalisatie: Query's efficiënt gemaakt, caching toegepast voor veelgevraagde endpoints.
- Frontend-optimalisatie: Lazy loading, minimalisatie van scripts en mediabestanden, efficiënte bundeling.
- Database-optimalisatie: Indexering van tabellen (bijv. NewsItems, Users) en geoptimaliseerde relaties voor snelle queries.
- Content delivery: Afbeeldingen en media via CDN, compressie en caching van statische bestanden.

Veiligheid & continuïteit:

Onderdeel	Beschrijving
Back-ups	Periodieke back-ups van database en mediabestanden.
Beveiliging	JWT-authenticatie, input validatie, rolgebaseerde toegang voor admins.
Privacy	Minimale opslag van persoonsgegevens, geen gevoelige data in logs.
Monitoring	Detectie van foutmeldingen, verdachte activiteiten en performance issues