|  |  |
| --- | --- |
|  | **ТЕХНИЧЕСКИ УНИВЕРСИТЕТ – СОФИЯ**  **ФАКУЛТЕТ КОМПЮТЪРНИ СИСТЕМИ И ТЕХНОЛОГИИ** |

**Дипломна работа**

**ТЕМА: Разработване на уеб система с чат бот за търсене на новини**

**Ръководител:** доц. д-р Г.Павлова

**София, 2025**

**Съдържание**

**1. I. УВОД .................................................. 3**

**2. II. ОБЗОР НА УЕБ ТЕХНОЛОГИИ .................................. 4**

2.1. Цели .................................................. 4

2.2. Задачи ................................................ 4

2.3. Обхват ............................................... 4-5

**3. III. ОБЗОР НА ИЗПОЛЗВАНИТЕ ТЕХНОЛОГИИ ....................... 6**

3.1. ASP.NET Core .......................................... 6

3.2. Entity Framework Core ................................ 7

3.3. PostgreSQL ........................................... 8

3.4. React.js ............................................. 8-9

3.5. TypeScript ........................................... 9-10

3.6. Material-UI .......................................... 10

3.7. OpenAI GPT-4o ........................................ 11

3.8. Google Custom Search API ............................. 11

3.9. Vite ................................................. 12

**4. IV. ПРОЕКТИРАНЕ НА СИСТЕМАТА ................................ 13**

4.1. Архитектура на системата ............................. 13-14

4.2. Структура на backend приложението .................... 15-21

4.3. Структура на frontend приложението ................... 22-24

4.4. Структура на базата данни ............................ 24-26

**5. V. ПРОГРАМНА РЕАЛИЗАЦИЯ ..................................... 26**

5.1. Реализация на backend функционалност ................. 26-28

5.2. Създаване и управление на статии ..................... 29-31

5.3. Чатбот с AI интеграция ............................... 31-32

5.4. Търсене с хибридна система ........................... 33-35

**6. VI. РЪКОВОДСТВО ЗА ИЗПОЛЗВАНЕ ............................... 36**

6.1. Главната страница на системата ....................... 36-37

6.2. Преглед и управление на категории .................... 38-39

6.3. Преглед и управление на статии ....................... 40-41

6.4. Работа с коментари ................................... 42-43

6.5. Използване на чатбот ................................. 43-44

6.6. Системи за търсене ................................... 45

6.7. Административни функционалности ....................... 46-47

6.8. Потребителски интерфейс .............................. 47-49

6.9. Интеграция ........................................... 49

**7. VII. ЗАКЛЮЧЕНИЕ ............................................. 50**

7.1. Постигнати резултати ................................. 50

7.2. Възможности за развитие .............................. 50

**8. VIII. ИЗПОЛЗВАНИ ИЗТОЧНИЦИ .................................. 51**

**I. УВОД**

В настоящата дипломна работа се разглежда проектирането и реализацията на съвременна система за управление на новини с интегриран интелигентен чатбот. Целта на проекта е да се създаде комплексно решение за публикуване, организация и търсене на новинарскo съдържание, както и интерактивен помощник за потребителите.

Със засиленето развитие на дигиталните технологии и нарастващата потребност от бърз достъп до актуална информация, необходимостта от ефективни системи за управление на новини става все по-важна. Традиционните новинарски платформи често срещат предизвикателства при организацията на големи обеми информация и предоставянето на персонализирано потребителско изживяване.

Разработената система за управление на новини адресира тези предизвикателства чрез:

* Модерна архитектура базирана на ASP.NET Core и React.js за осигуряване на високопроизводително и скалируемо решение
* Интелигентен чатбот с AI възможности за интерактивно търсене и отговор на въпроси
* Хибридна система за търсене комбинираща текстово и семантично търсене за по-точни резултати
* Интуитивен потребителски интерфейс за лесно навигиране и управление на съдържанието

• Гъвкава система за категоризация и тагове за по-добра организация

* Система за коментари и обратна връзка от потребителите

Настоящата работа е структурирана в седем основни части:

• В първата част се прави увод и се поставят целите на проекта

• Във втората част се разглеждат използваните уеб технологии

• В третата част се представя архитектурата на системата

• В четвъртата част се описва програмната реализация

• В петата част се предоставя ръководство за използване

• В шестата част се правят заключения

• В седмата част са посочени използваните източници

**II. ОБЗОР НА УЕБ ТЕХНОЛОГИИ**

**Цели**

Основната цел на проекта е създаването на съвременна, скалируема и потребителски ориентирана система за управление на новини с интегриран AI чатбот. Системата трябва да предоставя ефективни инструменти за създаване, публикуване, категоризиране и търсене на новинарски съдържание.

**Главни цели:**

1. Създаване на интуитивна система за управление на новини

2. Интеграция на AI чатбот за интерактивно търсене и помощ

3. Реализация на хибридна система за търсене с високa точност

4. Осигуряване на отзивчив и съвременен потребителски интерфейс

5. Създаване на скалируема архитектура за бъдещо развитие

**Задачи:**

1. Проектиране и реализация на RESTful API с ASP.NET Core

2. Създаване на модерен frontend с React.js и TypeScript

3. Интеграция с OpenAI GPT-4o за AI функционалности

4. Реализация на хибридно търсене (текстово + семантично)

5. Създаване на система за категории и тагове

6. Имплементация на система за коментари

7. Разработка на административен панел

8. Оптимизация на производителността и SEO

**Обхват:**

Проектът обхваща пълноценна уеб система за управление на новини със следните основни компоненти:

**Backend API:**• RESTful API с ASP.NET Core 8.0

• Автентификация и авторизация с JWT токени

• Entity Framework Core за работа с база данни

• Интеграция с OpenAI API за AI функционалности

• Google Custom Search API за уеб търсене

• Системи за кеширане и оптимизация

**Frontend приложение:**• Single Page Application (SPA) с React.js

• TypeScript за статично типизиране

• Material-UI за съвременен дизайн

• Responsive дизайн за всички устройства

• Интерактивен AI чатбот

• Разширени възможности за търсене

**База данни:**

• PostgreSQL за основно съхранение на данни

• Оптимизирани схеми за бързи заявки

• Индексиране за подобрена производителност

**III. ОБЗОР НА ИЗПОЛЗВАНИТЕ ТЕХНОЛОГИИ**

**ASP.NET Core [1]**

ASP.NET Core е безплатна софтуерна рамка за уеб разработка с отворен код. Също така тя се явява и следващата стъпка в еволюцията на ASP.NET. Тя е разработена съвместно от Microsoft и общността, която е събрала през годините на своето развитие. ASP.NET Core е модулярна софтуерна рамка, която върви на крос-платформената .NET Core. Въпреки, че е нова софтуерна рамка изградена върху нов уеб стек, тя има висока степен на съвместимост и прилики с ASP.NET.

Спрямо своите предшественици ASP.NET Core поддържа нова функция – т.нар. “side by side versioning”. При нея различни приложения, които използват една и съща машина могат да таргетират различни версии на ASP.NET Core, в зависимост от версиите (и нуждите) си. Това не е възможно с по-стари издания на ASP.NET.

Съвместим е с Windows, MacOS и Linux, което го прави гъвкав и удобен за използване. Подходящ е за употреба, тъй като има интегрирани множество от библиотеки и компоненти, които се изпълняват по време на работа и компилация.

В случая не само платформите са от голяма значимост, но и това, че разработваната система има нужда от висока производителност и надграждане. .NET Core и ASP.NET Core са най-добрата опция в случая, тъй като предоставят нужната производителност за стартиране и функциониране на сървърната част. Освен това, управляваното време за изпълнение улеснява разработката, garbage-collection-а и гарантира безопасно изпълнение.

.NET Core е набор от изпълними, библиотечни и компилационни компоненти, които могат да се иползват в различни конфигурации за работа на устройства и в облака. Междуплатформено и с отворен код .NET Core предоставя олекотен модел за разработване и гъвкавостта да се работи с различни инструменти за разработване и платформени ОС. .NET Core е достъпно в GitHub под лиценза MIT като включва няколко технологии - .NET Core, ASP.NET Core, Entity Framework Core и др.  
  
ASP.NET Core е модерна, кроспластформена рамка за разработка на уеб приложения от Microsoft. Версия 8.0 предлага подобрена производителност, нови функционалности и по-добра поддръжка за cloud-native приложения.

Ключови предимства на ASP.NET Core:

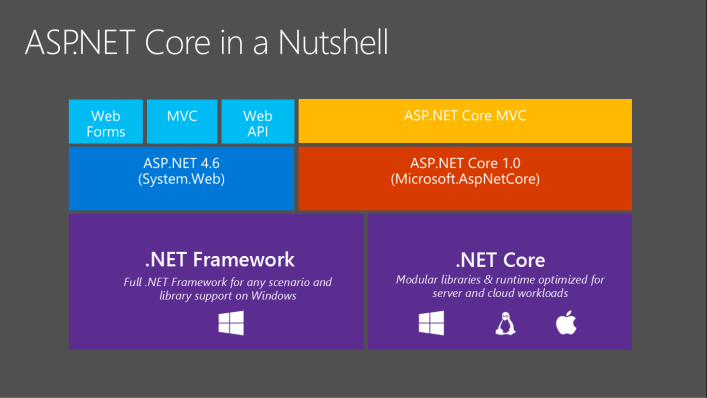
• Високопроизводителна и скалируема архитектура

• Кроспластформена поддръжка (Windows, Linux, macOS)

• Вградена поддръжка за dependency injection

• Мощни възможности за API разработка

• Модулен дизайн позволяващ гъвкавост при развитието



Фиг. 1. Шаблон с възможности на ASP.NET Core

**Entity Framework Core [2]**

Entity Framework Core е лека, разширяема и многоплатформена версия на популярната технология за достъп и поддръжка на данни Entity Framework. Тя може да ни служи за създаване на връзка между обекти, позволявайки на разработчиците да работят с база от данни с .NET обекти и елиминирайки нуждата от голяма част от кода за достъп до данните, която би се наложило да напишат.

Има много начини, чрез които може да се проектира такава библиотека, EF Core е създадена като object-relational mapper (ORM). Този тип дизайн работи като създава връзката между две страни базата данни с API-то, което използва и обектно ориентирана софтуерна страна. Това е един от най-бързите начини, чрез които разработчиците на софтуер достъпват базата от данни бързо и лесно. Именно затова, EF Core се използва за разработката на текущото приложение – за достъп и контрол на база от данни, която съхранява данните на приложението.

Основни характеристики:

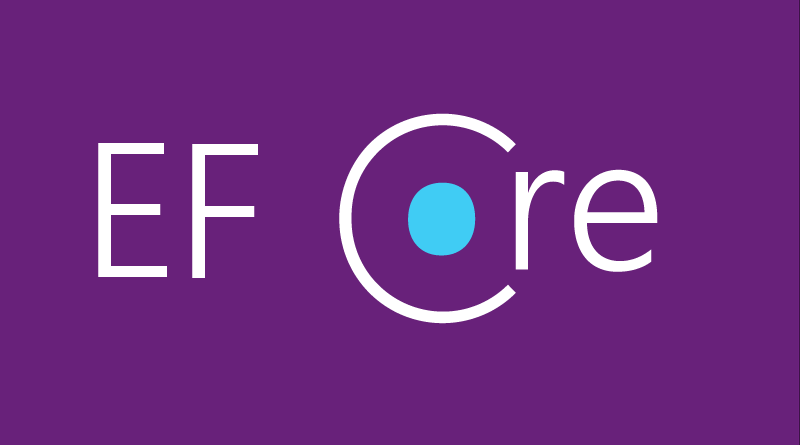
• Code First подход за моделиране на данни

• LINQ заявки за елегантен код

• Автоматично генериране на миграции

• Поддръжка за множество доставчици на бази данни

• Оптимизация на заявките за подобрена производителност



Фиг. 2. Лого на Entity Framework Core

**PostgreSQL [3]**

PostgreSQL е мощна, open source релационна база данни с разширена поддръжка за JSON, пълнотекстово търсене и геопространствени данни. Избрана е заради стабилността, производителността и богатите функционалности.

Предимства:

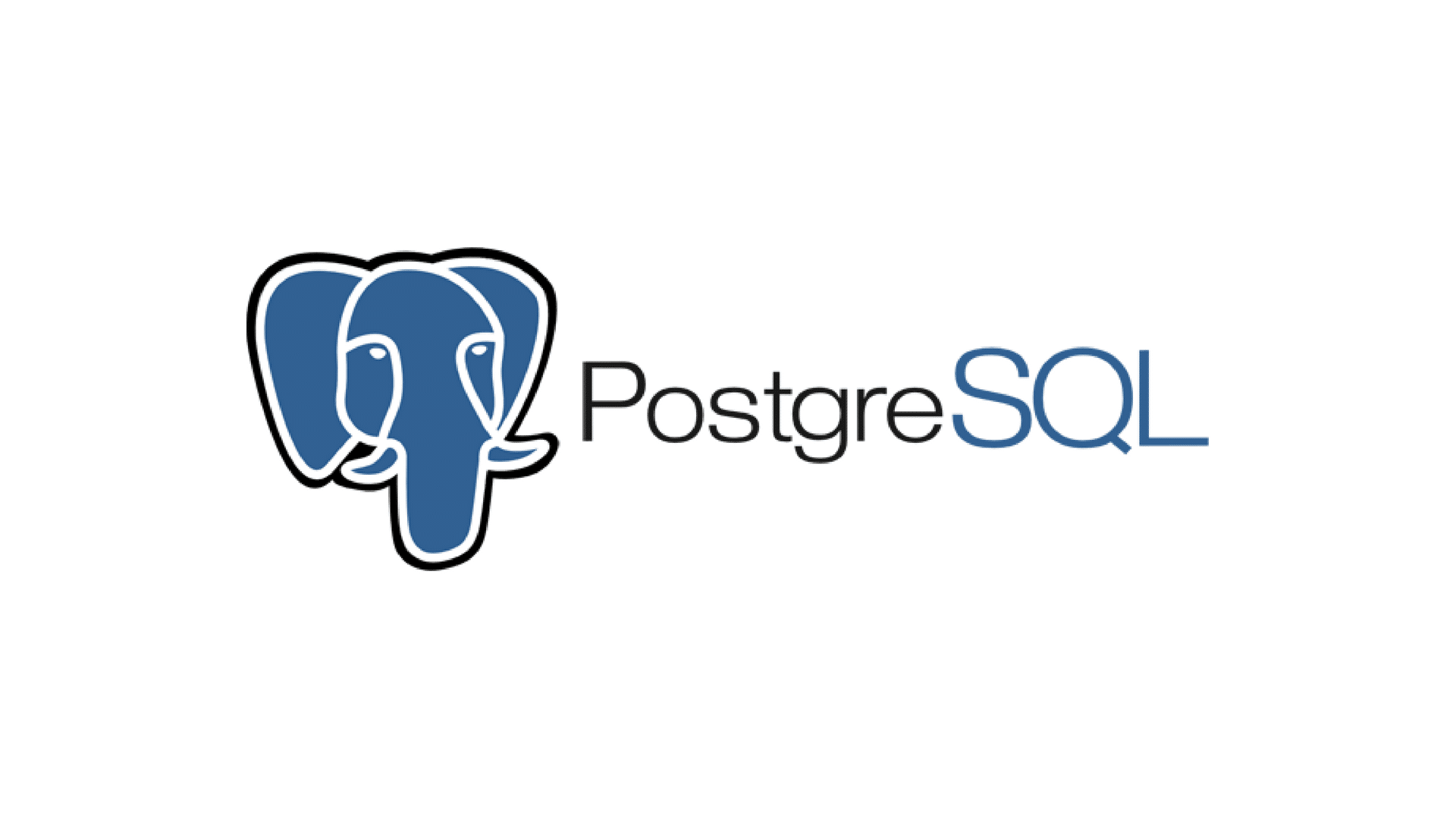
• Отлична производителност при сложни заявки

• Поддръжка за JSON и документни структури

• Вградено пълнотекстово търсене

• ACID съвместимост и надеждност

• Активна общност и развитие



Фиг. 3. Лого на PostgreSQL

**React.js [4]**

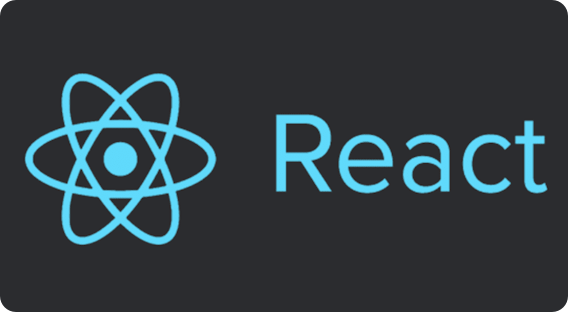
React.js е популярна JavaScript библиотека за изграждане на потребителски интерфейси. Създадена от Facebook, тя позволява създаването на интерактивни и отзивчиви уеб приложения чрез компонентен подход.

Основни особености:

• Virtual DOM за оптимизирана производителност

• Компонентна архитектура за преизползваемост

• Односторонен поток от данни

• Богата екосистема от библиотеки  
• Отлична поддръжка за SEO  
  
  


Фиг. 4. Лого на React JS

**TypeScript [5]**

TypeScript е надстройка над JavaScript, която добавя статично типизиране. Развива се от Microsoft и се компилира до чист JavaScript код.

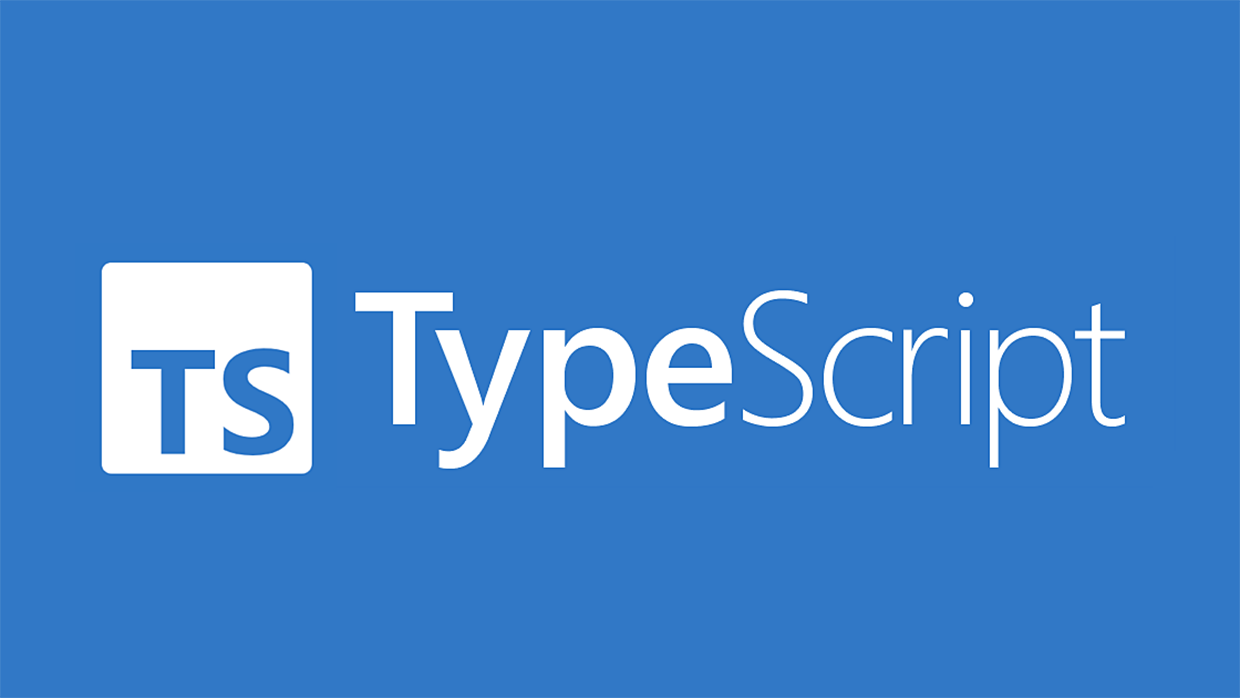
Предимства:

• Статично типизиране за по-малко грешки

• Подобрена поддръжка в IDE

• Обектно-ориентирано програмиране

• Съвместимост с JavaScript екосистемата

• Подобрена поддръжка при рефакториране  
  
  
  


Фиг. 5. Лого на TypeScript

**Material-UI [6]**

Material-UI (MUI) е популярна React библиотека за компоненти, базирана на Google's Material Design принципите. Предоставя готови, красиво оформени компоненти за бърза разработка.

Особености:

• Съвременен и консистентен дизайн

• Богат набор от готови компоненти

• Отзивчив дизайн out-of-the-box

• Тематизиране и персонализация

• Достъпност и интернационализация



Фиг. 6. Лого на Material UI

**OpenAI GPT-4o [7]**

GPT-4o е модел на OpenAI с мултимодални възможности. Интегриран е в системата за предоставяне на интелигентни отговори и помощ на потребителите.

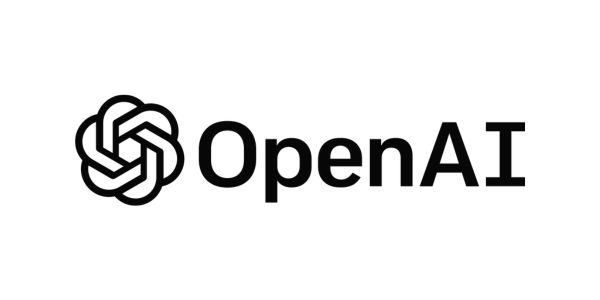
Възможности:

• Разбиране на естествен език

• Генериране на съдържание

• Отговаряне на въпроси

• Резюмиране на текстове

• Многоезична поддръжка  
  


Фиг. 7. Лого на Open AI

**Google Custom Search API [8]**

Google Custom Search API позволява интеграция на Google търсене в приложения. Използва се за разширяване на възможностите за търсене извън локалното съдържание.

Функционалности:

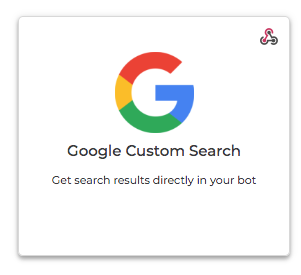
• Мощни възможности за търсене

• Персонализирани резултати

• Филтриране по домейни

• Програматичен достъп

• Висока точност на резултатите



Фиг. 8. Лого на Google Custom Search API

**Vite [9]**

Vite е модерен build tool за frontend разработка, който предлага бързо development окружение и оптимизирани build процеси.

Предимства:

• Моментално стартиране на dev server

• Hot Module Replacement (HMR)

• Оптимизирани production builds

• Plugin екосистема

• TypeScript поддръжка

  
  
  
Фиг. 9. Лого на Vite JS

**IV. ПРОЕКТИРАНЕ НА СИСТЕМАТА**

**Архитектура на системата**

Системата за управление на новини е проектирана като модерно трислойно приложение, състоящо се от frontend уеб интерфейс, backend API и база данни. Архитектурата следва принципите на separation of concerns и осигурява високо ниво на скалируемост и поддръжка.

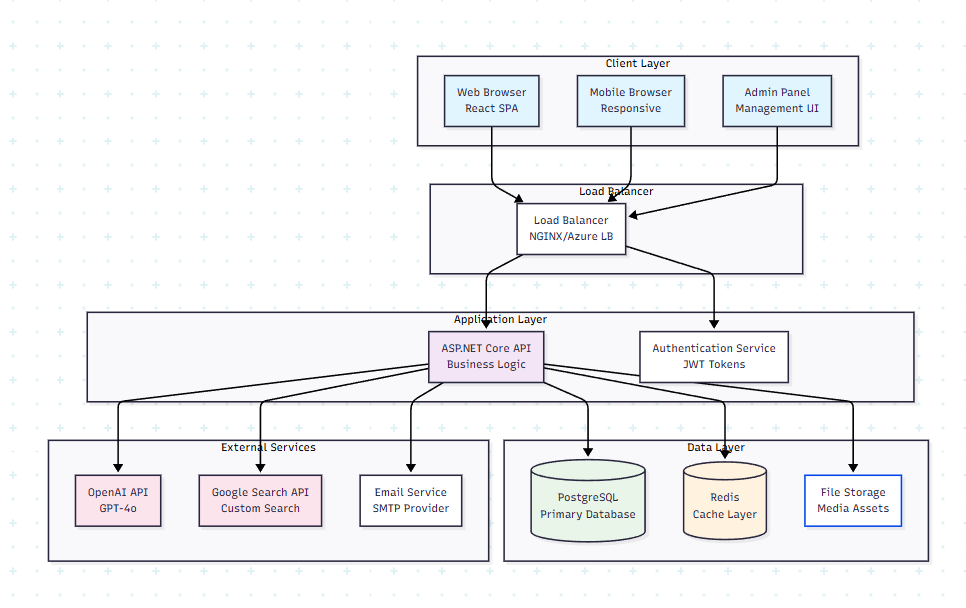
Основни компоненти:

• Frontend (React.js) - потребителски интерфейс

• Backend API (ASP.NET Core) - бизнес логика и данни

• База данни (PostgreSQL) - съхранение на информация

• Външни услуги (OpenAI, Google Search) - AI и търсене



Фиг. 10. Общ преглед на системната архитектура

**Структура на backend приложението**

Backend приложението е организирано в няколко основни слоя, следвайки clean architecture принципите:

**Controllers:**

Съдържа MVC контролери, които обработват HTTP заявките и връщат отговори. Всеки контролер е отговорен за специфична функционалност.

1. ArticlesController - управление на статии

• GET /api/articles - извличане на всички статии

• GET /api/articles/{id} - извличане на конкретна статия

• POST /api/articles - създаване на нова статия

• PUT /api/articles/{id} - актуализиране на статия

• DELETE /api/articles/{id} - изтриване на статия

2. CategoriesController - управление на категории

• GET /api/categories - извличане на всички категории

• POST /api/categories - създаване на нова категория

• PUT /api/categories/{id} - актуализиране на категория

• DELETE /api/categories/{id} - изтриване на категория

3. CommentsController - управление на коментари

• GET /api/comments - извличане на коментари

• POST /api/comments - създаване на коментар

• DELETE /api/comments/{id} - изтриване на коментар

4. ChatbotController - AI чатбот функционалности

• POST /api/chatbot/ask - изпращане на въпрос към AI

• GET /api/chatbot/history - история на разговорите

5. HybridSearchController - хибридно търсене

• POST /api/search/hybrid - търсене в локални и външни източници

• GET /api/search/suggestions - предложения за търсене

6. AuthController - автентификация и авторизация

• POST /api/auth/login - вход в системата

• POST /api/auth/register - регистрация на потребител

• POST /api/auth/refresh - обновяване на токен

**Services:**

Съдържа бизнес логиката на приложението, отделена от контролерите за по-добра тестваемост и преизползваемост.

1. ArticleService - бизнес логика за статии

• Валидация на данни

• Управление на статус на публикуване

• Търсене и филтриране

2. CategoryService - управление на категории

• Йерархична организация

• Валидация на категории

3. CommentService - управление на коментари

• Модерация на съдържание

• Nested коментари

4. ChatbotService - AI интеграция

• Комуникация с OpenAI API

• Обработка на контекст

• Кеширане на отговори

5. HybridSearchService - търсене

• Локално търсене в базата

• Интеграция с Google Search

• Подреждане на резултати

**Data Access:**

Data Access слоят е отговорен за комуникацията с базата данни чрез Entity Framework Core.

1. ApplicationDbContext - основен database context

2. Repository паттерн за абстракция над данните

3. Миграции за управление на схемата

**DTOs (Data Transfer Objects):**

Обекти за трансфер на данни между слоевете:

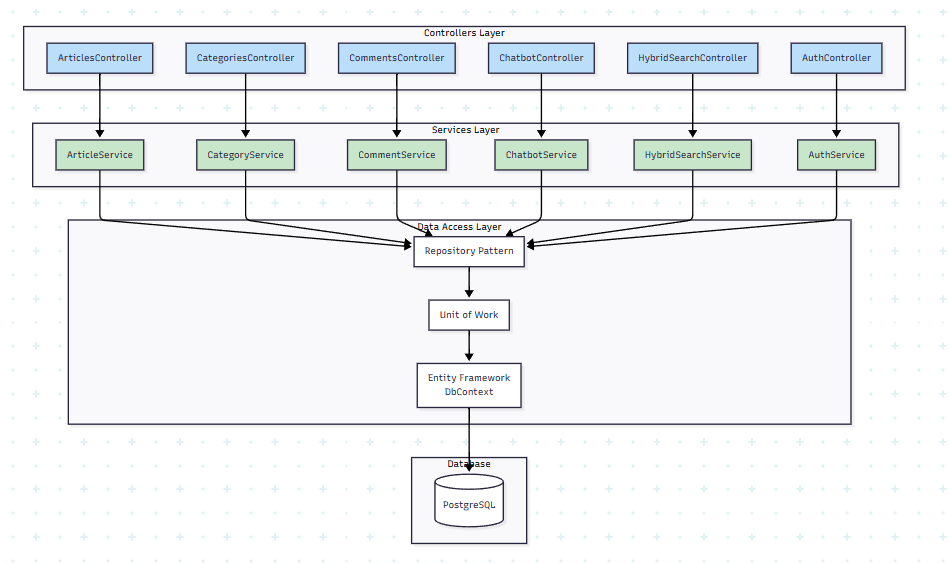
1. ArticleDTO - данни за статии

2. CategoryDTO - данни за категории

3. CommentDTO - данни за коментари

4. ChatRequestDTO/ChatResponseDTO - чатбот комуникация

5. SearchResultDTO - резултати от търсене



Фиг. 11. Диаграма на backend архитектурата

**Структура на frontend приложението**

Frontend приложението е построено като Single Page Application (SPA) с React.js и TypeScript. Архитектурата следва компонентния подход с ясно разделение на отговорностите.

Основни директории:

1. src/components/ - преизползваеми компоненти

• ArticleCard - карта за показване на статия

• CategoryTree - йерархично дърво от категории

• CommentSection - секция за коментари

• SearchBox - поле за търсене

• Pagination - компонент за страниране

2. src/pages/ - основни страници

• HomePage - начална страница

• ArticlePage - детайлна страница за статия

• CategoryPage - страница за категория

• AdminPage - административен панел

3. src/services/ - API комуникация

• ArticleService - заявки към articles API

• CategoryService - заявки към categories API

• ChatbotService - комуникация с чатбот

• AuthService - автентификация

4. src/hooks/ - custom React hooks

• useArticles - hook за управление на статии

• useAuth - hook за автентификация

• useDebounce - hook за debouncing

• useInfiniteScroll - безкрайно скролиране

5. src/context/ - React context за глобално състояние

• AuthContext - контекст за автентификация

• ThemeContext - контекст за тема

• NotificationContext - контекст за уведомления

6. src/utils/ - помощни функции

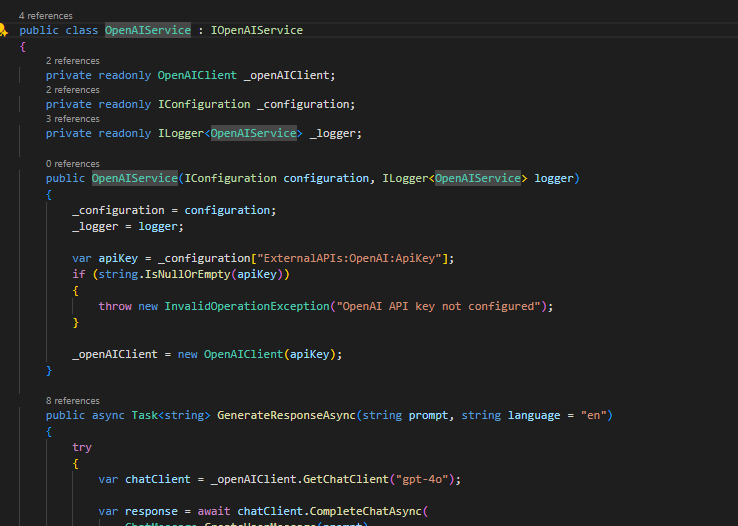
• dateHelpers - работа с дати

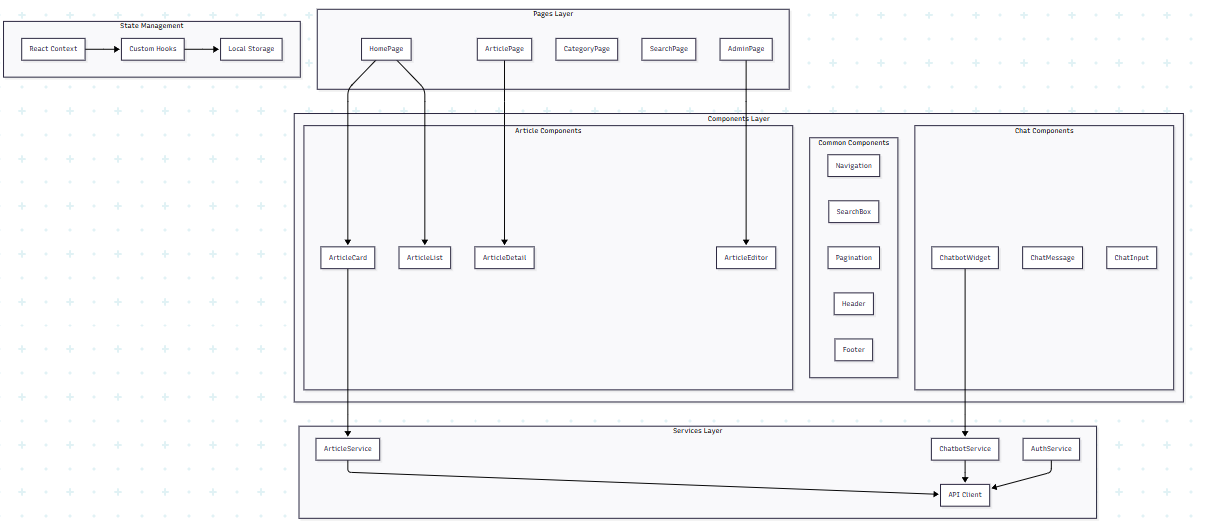
• textHelpers - форматиране на текст

• apiClient - HTTP клиент

**AI Чатбот компонент:**

**Специализиран компонент за интерактивна комуникация с AI:**





Фиг. 11. Диаграма на frontend архитектурата

**Структура на базата данни**

Базата данни е проектирана за оптимална производителност и гъвкавост при управление на новинарското съдържание.

Основни таблици:

Articles - основна таблица за статии

• Id (PK, int, not null) - уникален идентификатор

• Title (varchar(500), not null) - заглавие на статията

• Content (text, not null) - съдържание на статията

• Summary (varchar(1000), null) - кратко резюме

• CategoryId (FK, int, not null) - връзка към категория

• AuthorId (FK, int, not null) - връзка към автор

• PublishedAt (timestamp, null) - дата на публикуване

• CreatedAt (timestamp, not null) - дата на създаване

• UpdatedAt (timestamp, not null) - дата на последна актуализация

• IsPublished (boolean, not null) - статус на публикуване

• ViewCount (int, default 0) - брой прегледи

• Tags (varchar[], null) - тагове за категоризация

Categories - категории за статии

• Id (PK, int, not null) - уникален идентификатор

• Name (varchar(100), not null) - име на категорията

• Description (varchar(500), null) - описание на категорията

• ParentId (FK, int, null) - родителска категория

• CreatedAt (timestamp, not null) - дата на създаване

• UpdatedAt (timestamp, not null) - дата на актуализация

• IsActive (boolean, not null) - активен статус

Comments - коментари към статии

• Id\*\* (PK, int, not null) - уникален идентификатор

• ArticleId\*\* (FK, int, not null) - връзка към статия

• AuthorName\*\* (varchar(100), not null) - име на автора

• AuthorEmail\*\* (varchar(255), not null) - email на автора

• Content\*\* (text, not null) - съдържание на коментара

• ParentId\*\* (FK, int, null) - родителски коментар

• CreatedAt\*\* (timestamp, not null) - дата на създаване

• IsApproved\*\* (boolean, default false) - одобрен статус

Users - потребители на системата

• Id (PK, int, not null) - уникален идентификатор

• Username (varchar(50), not null) - потребителско име

• Email (varchar(255), not null) - email адрес

• PasswordHash (varchar(255), not null) - хеширана парола

• FirstName (varchar(100), null) - собствено име

• LastName (varchar(100), null) - фамилно име

• Role (varchar(20), not null) - роля (Admin, Editor, User)

• CreatedAt (timestamp, not null) - дата на създаване

• LastLoginAt (timestamp, null) - последен вход

ChatConversations - история от разговори с чатбот

• Id (PK, int, not null) - уникален идентификатор

• SessionId (varchar(100), not null) - идентификатор на сесия

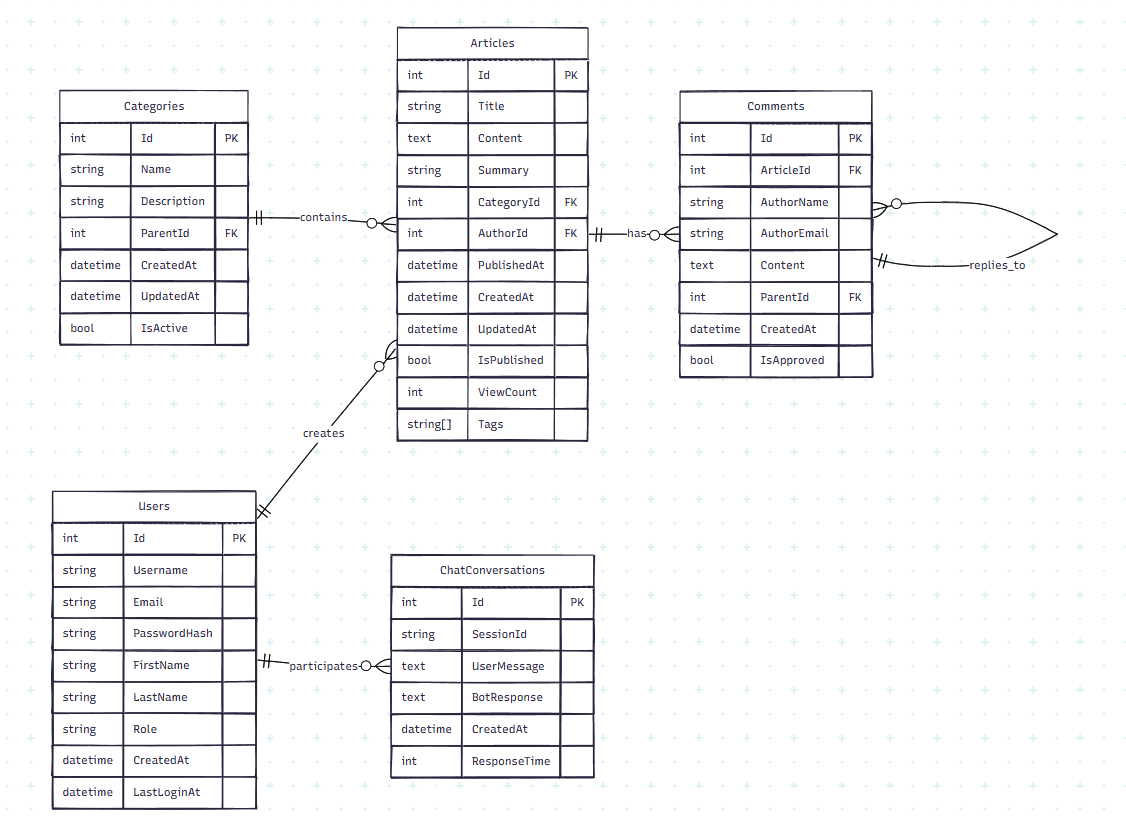
• UserMessage (text, not null) - съобщение от потребител

• BotResponse (text, not null) - отговор от бот

• CreatedAt (timestamp, not null) - дата на създаване

• ResponseTime (int, null) - време за отговор в милисекунди  
  
Както може да се види от таблицата по-долу връзките към отделните таблици са както следва:

|  |  |  |  |
| --- | --- | --- | --- |
| From Table | To Table | Relationship | Type |
| User | NewsArticle | Author → Articles | 1:M |
| Category | NewsArticle | Category → Articles | 1:M |
| User | Comment | User → Comments | 1:M |
| NewsArticle | Comment | Article → Comments | 1:M |
| User | ArticleLike | User → Likes | 1:M |
| NewsArticle | ArticleLike | Article → Likes | 1:M |
| User | NewsArticle | Users ↔ Liked Articles | M:M |



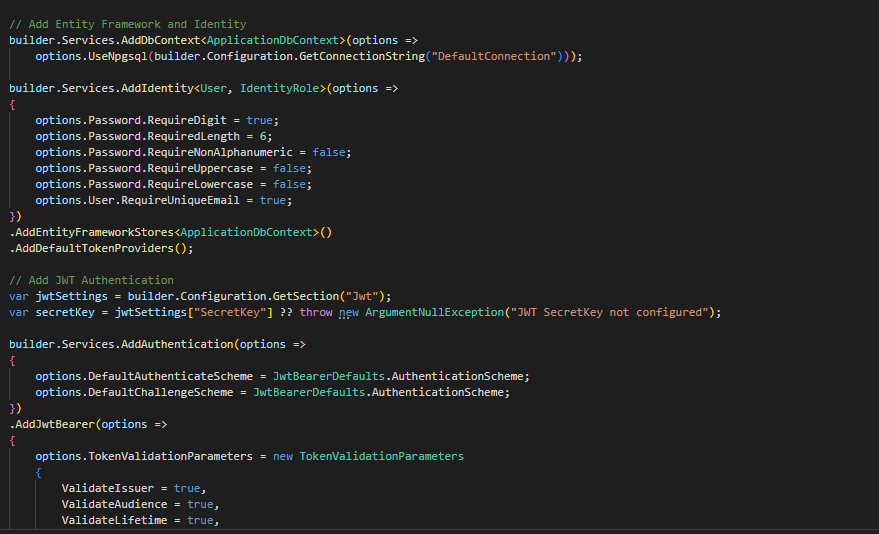
Фиг. 11. ER Диаграма на Базата Данни

**V. ПРОГРАМНА РЕАЛИЗАЦИЯ**

**Реализация на backend функционалност**

Backend API е реализиран следвайки RESTful принципите и clean architecture подхода. Всеки слой има ясно дефинирани отговорности и минимални зависимости.

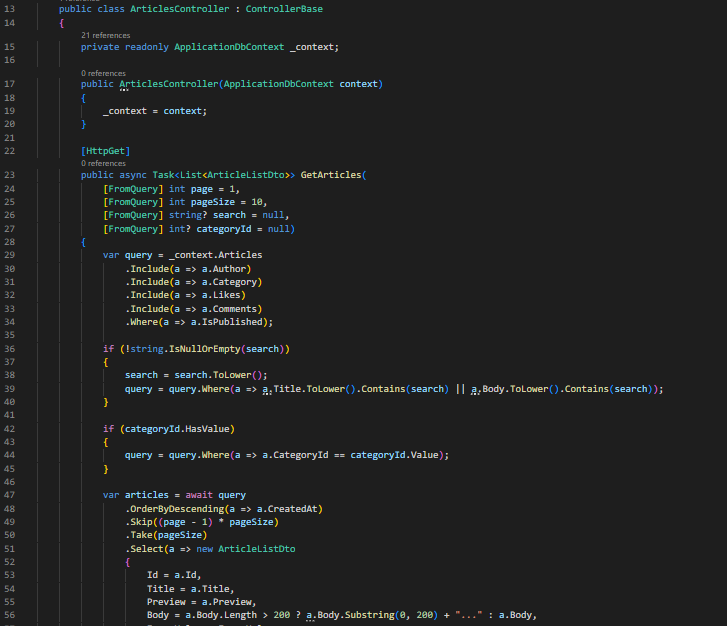
Част от Кода показващ конфигурацията на приложението:

**

**Създаване и управление на статии**

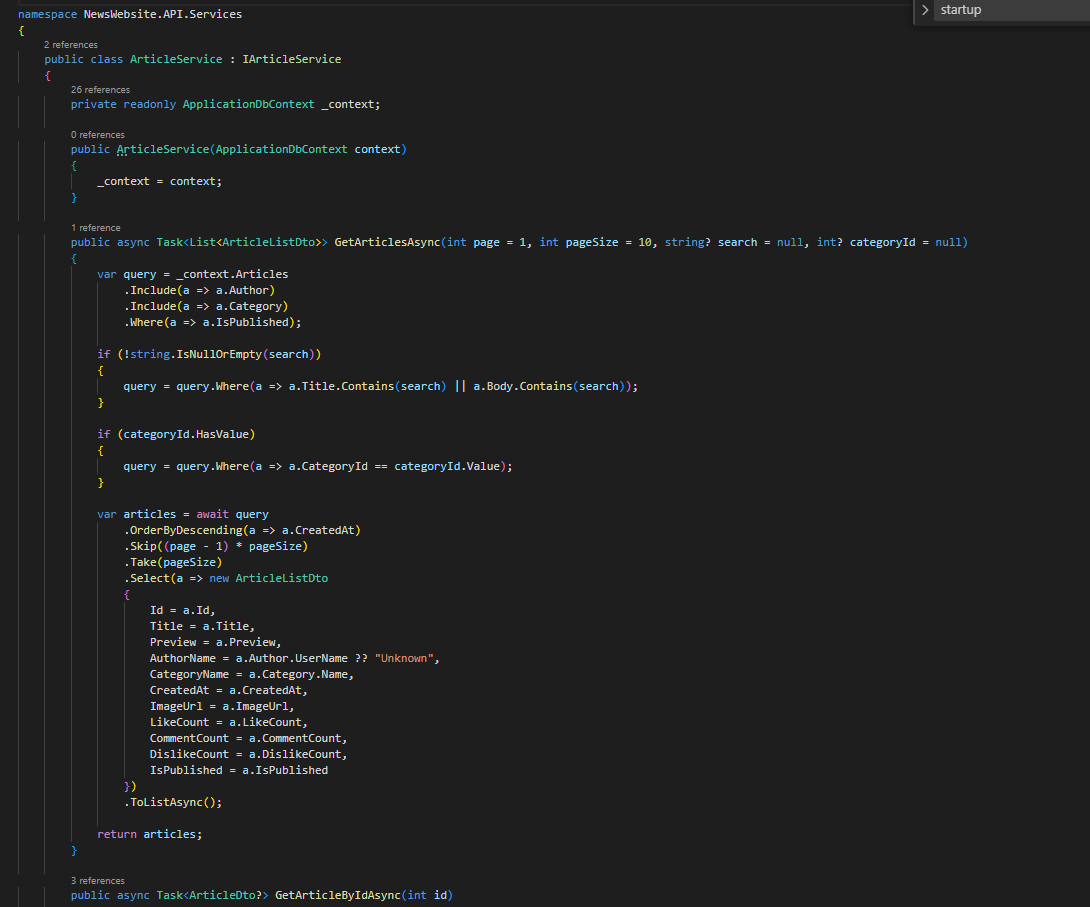
Системата за управление на статии предоставя пълен CRUD функционал с валидация, кеширане и SEO оптимизации.

ArticleController имплементация:



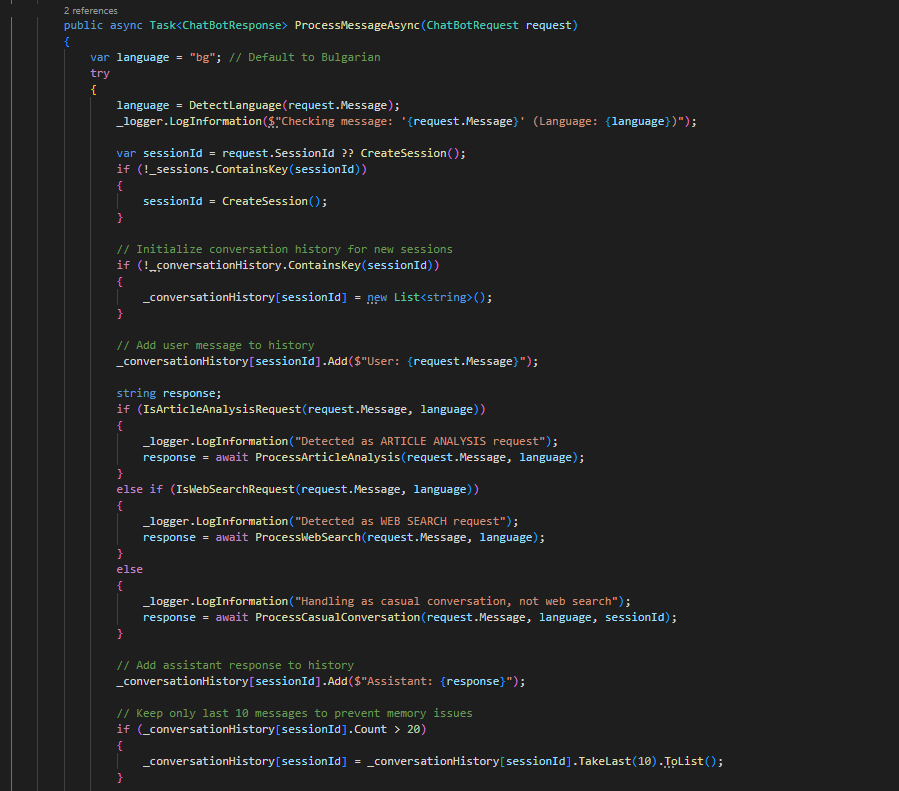
**ArticleService бизнес логика:**ArticleService е основният компонент за управление на новинарските статии в нашето приложение. Той действа като посредник между нашите контролери и базата данни, като обработва всички операции, свързани със статиите.

ArticleService е по същество „мозъкът“ на нашият новинарски уебсайт, обработвайки цялата сложна бизнес логика за управление на статии, взаимодействия с потребителите и откриване на съдържание, като същевременно поддържа сигурност и производителност!



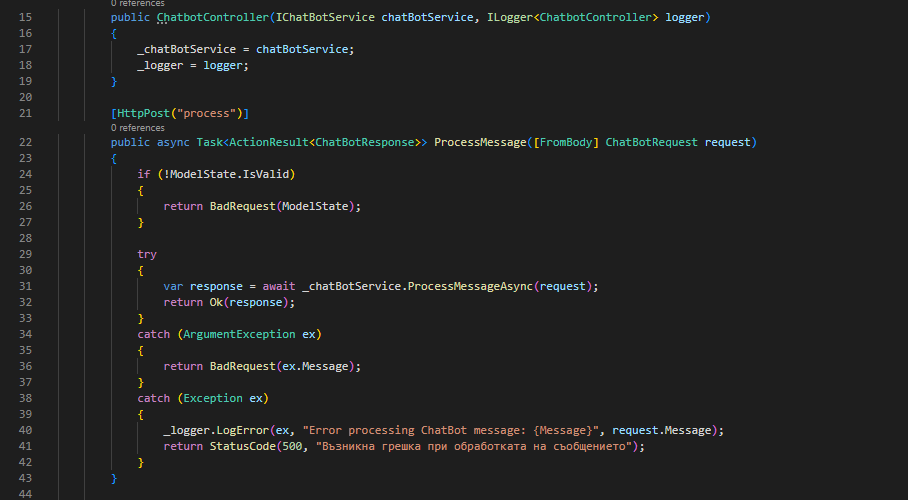
**Чатбот с AI интеграция**

AI чатботът е интегриран с OpenAI GPT-4o за предоставяне на интелигентни отговори на потребителските въпроси, като поддържа анализ на статии и не формален разговор с потребителя. Тук се намира и логиката за определеяне на вида търсене (уеб или локално).



**ChatbotController:**

Контролерът на ChatBot представлява усъвършенстван, управляван от изкуствен интелект, разговорен интерфейс, който служи като интелигентен мост между потребителите и вашата новинарска екосистема.



**Търсене с хибридна система**

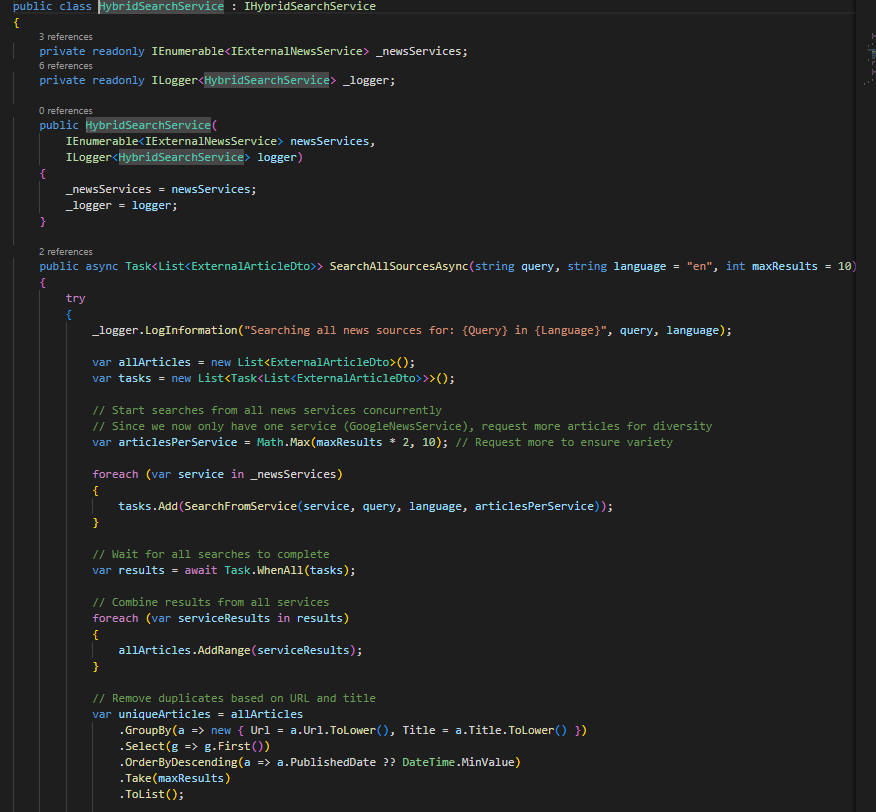
Хибридната система за търсене комбинира локално търсене в базата данни с външни източници чрез Google Custom Search API.  
HybridSearchService е усъвършенстван механизъм за генериране на новини от множество източници, който предоставя възможност за търсене на нашият новинарски уебсайт.

HybridSearchService действа като интелигентен оркестратор за генериране на новини, който комбинира множество източници на информация, за да предостави изчерпателни резултати от търсенето, подобрени с изкуствен интелект.

Системата използва RankArticlesByRelevanceAsync на OpenAI, за да оценява интелигентно външни статии въз основа на релевантността на заявката.

По същество HybridSearchService трансформира нашият новинарски уебсайт в цялостен информационен център, който интелигентно комбинира нашето редакционно съдържание с глобални новини в реално време, всичко това подобрено от класиране по релевантност, задвижвано от изкуствен интелект. В този случай имаме интелигентен редактор на новини, който може незабавно да получи достъп както до нашият архив, така и до целия интернет, за да намери най-подходящата информация за всяка заявка.

HybridSearchService:



**VI. РЪКОВОДСТВО ЗА ИЗПОЛЗВАНЕ**

**Главната страница на системата**

Главната страница предоставя достъп до всички основни функционалности на системата. Потребителите могат да разглеждат най-новите статии, да търсят съдържание и да използват AI чатбота.

Основни елементи:

• Навигационно меню с категории

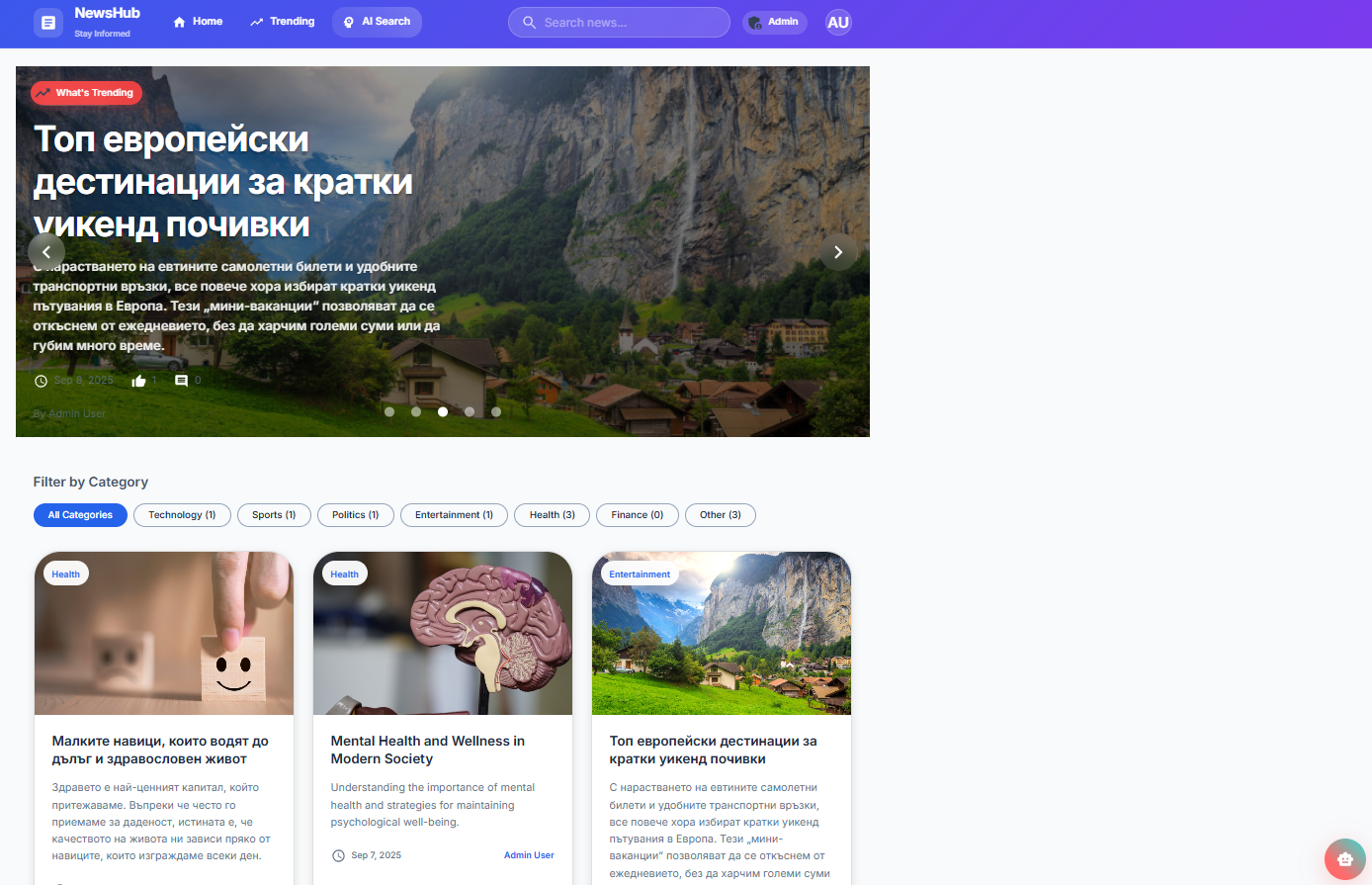
• Секция за най-нови статии

• Поле за търсене с автодопълване

• AI чатбот в долния десен ъгъл

• Секция за популярни статии

• Footer с допълнителна информация



Фиг. 12. Главна страница на системата

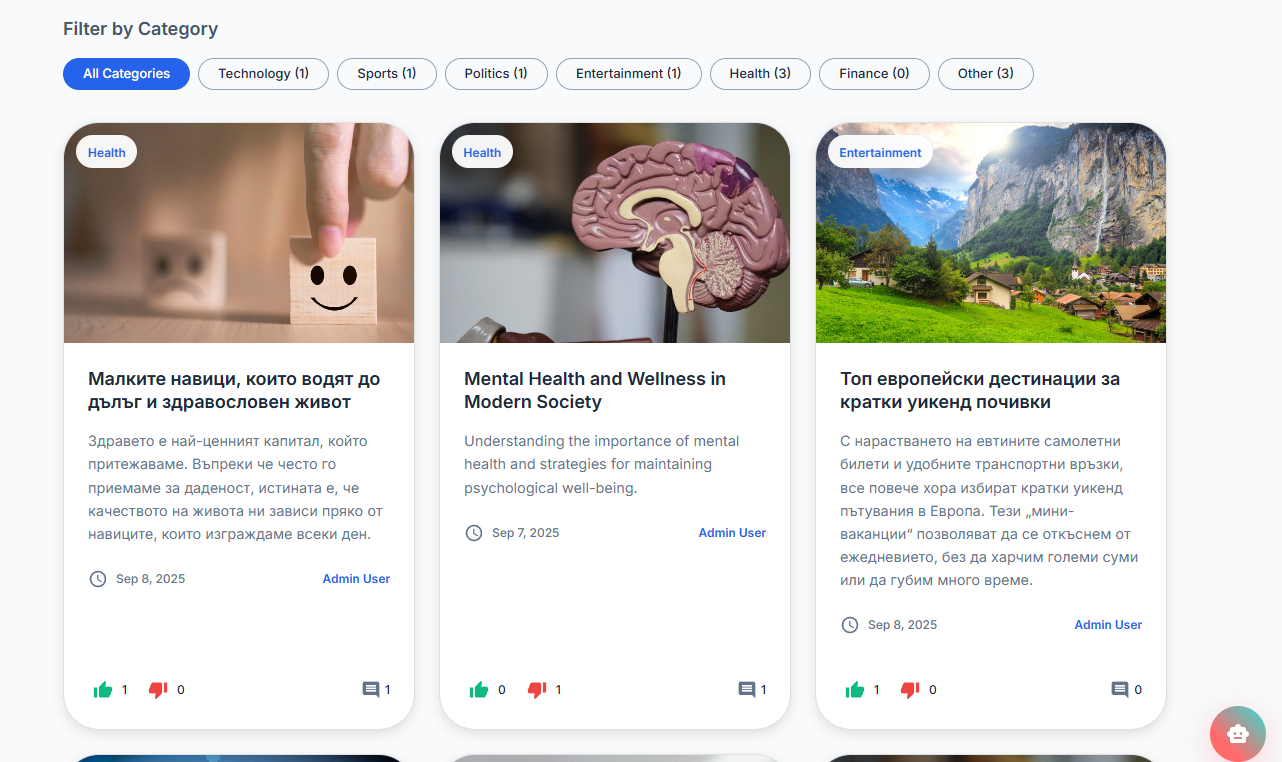
**Преглед и управление на категории**

Системата поддържа йерархична структура от категории за по-добра организация на съдържанието.

Функционалности за потребители:

• Разглеждане на категории в дървовидна структура

• Филтриране на статии по категория

• Навигация между родителски и дъщерни категории  
  


Фиг. 13. Филтриране на статии по категории

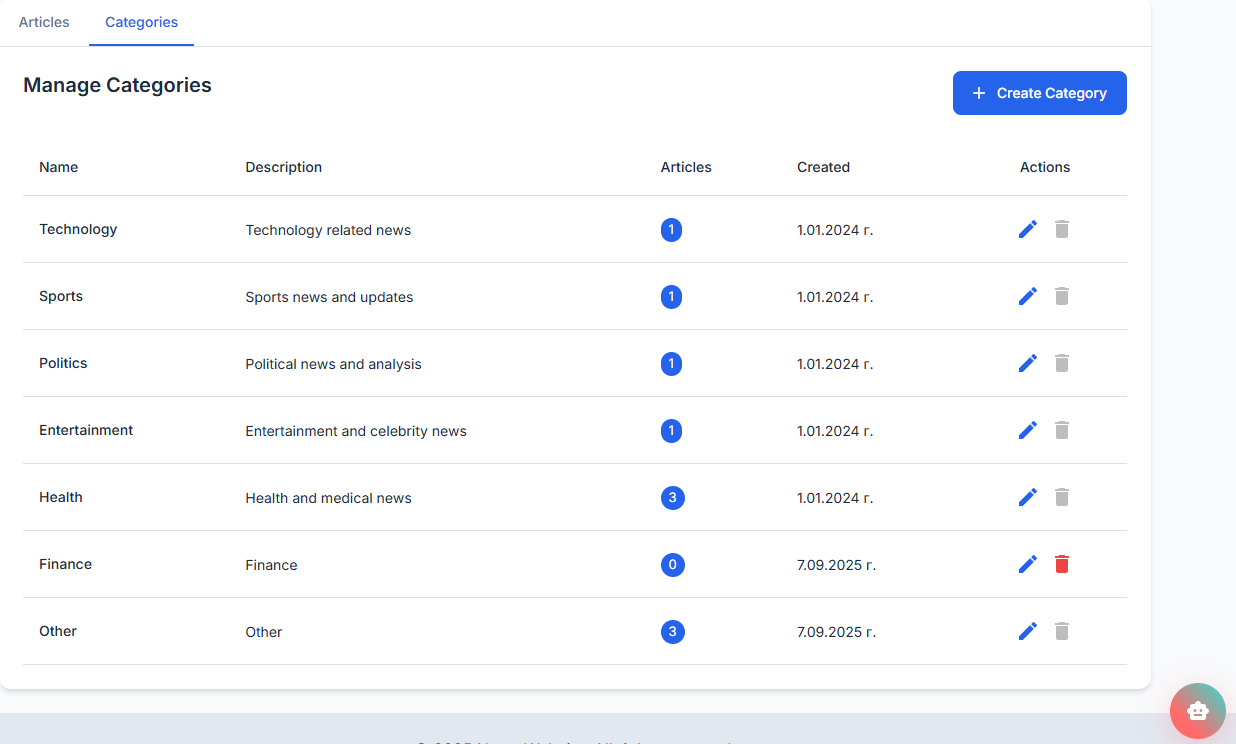
**Административни функции:**

• Създаване на нови категории

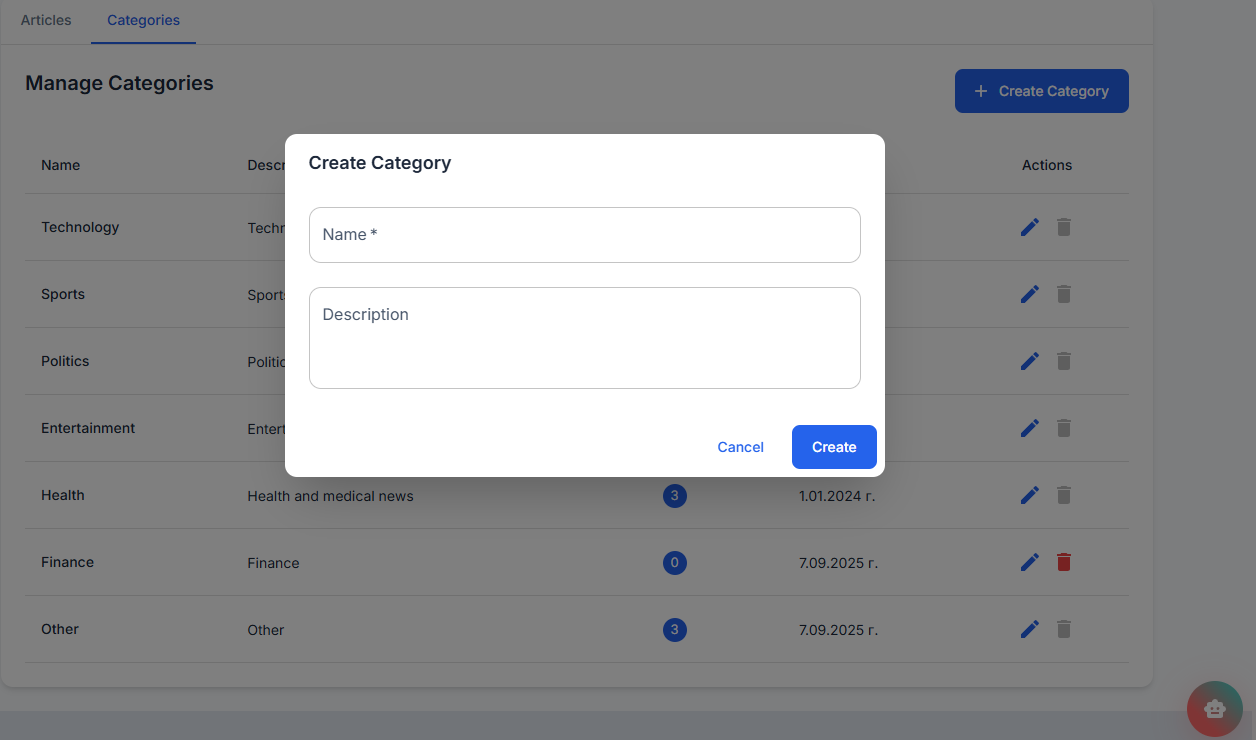
• Редактиране на съществуващи категории

• Преместване на категории в йерархията

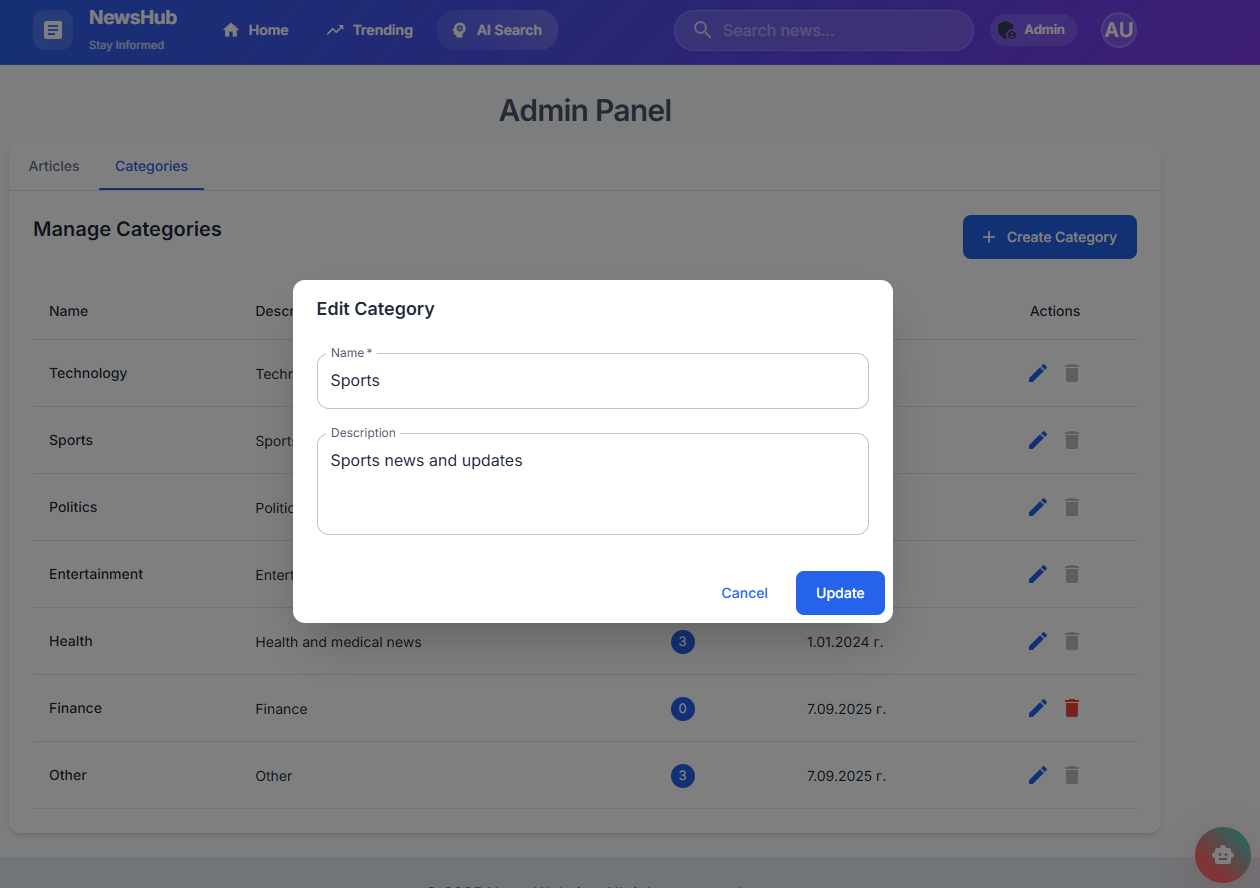
• Деактивиране на категории



Фиг. 14. Админ Панел с възможнот за добавяне или промяна на категории



Фиг. 15. Админ Панел с възможнот за добавяне на нова категория



Фиг. 15. Админ Панел с възможнот за промяна на вече съществуваща категория

**Преглед и управление на статии**

Статиите са основният тип съдържание в системата. Потребителите могат да ги четат, коментират и споделят.

Преглед на статии:

• Списъчен изглед с пагинация

• Детайлен изглед на статия

• Свързани статии

• Секция за коментари

• Бутони за споделяне в социални мрежи

Административно управление:

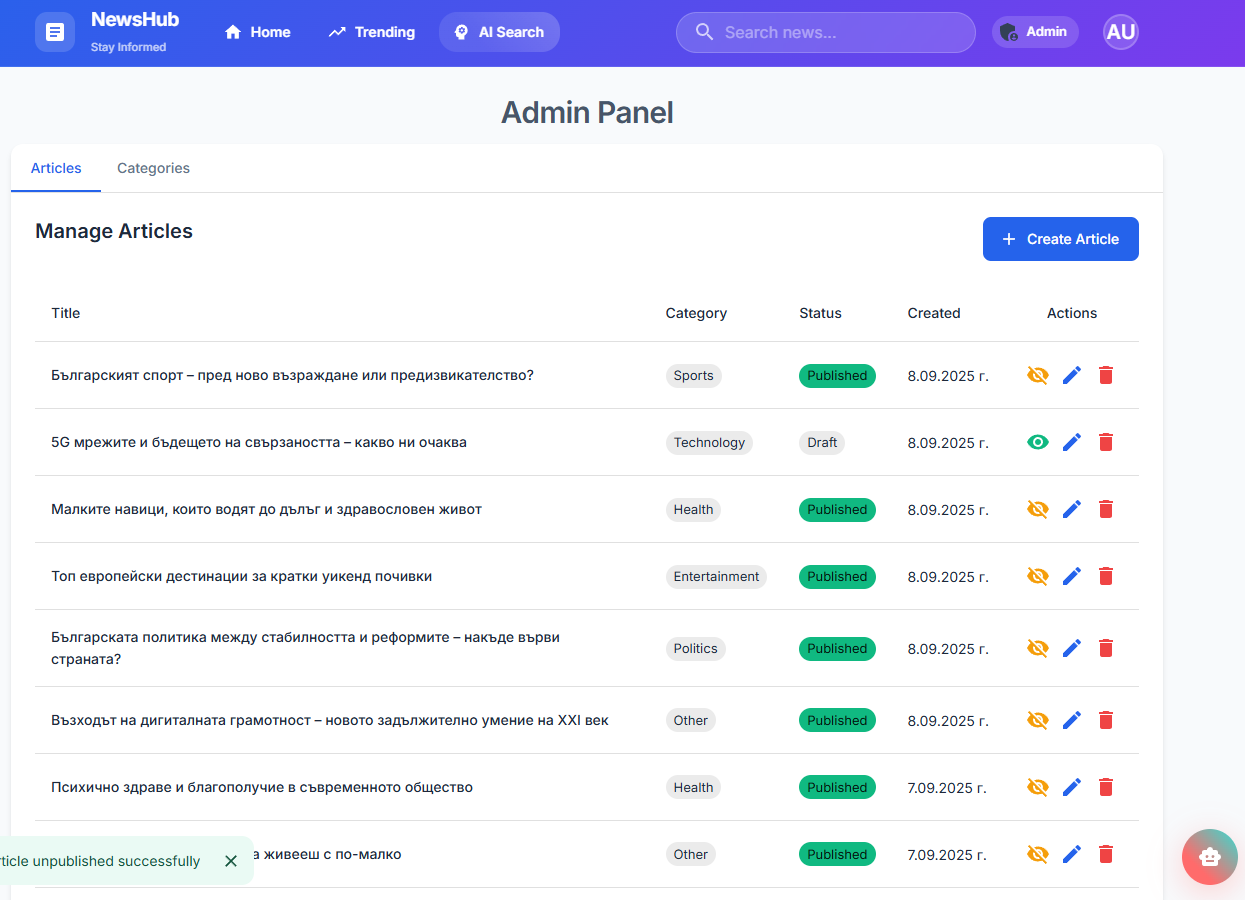
• Създаване на нови статии с rich text editor

• Редактиране на съществуващи статии

• Публикуване/скриване на статии

• Управление на тагове

• SEO настройки



Фиг. 16. Админ Панел за добавяне / редактиране на статии

**Работа с коментари**

Системата поддържа вложени коментари с модерация.

Функционалности:

• Добавяне на коментари от регистрирани потребители

• Отговаряне на съществуващи коментари

• Модерация от администратори

• Известяване за нови коментари



Фиг. 17. Добавяне на коментар под статия

**Използване на чатбот**

AI чатботът предоставя интерактивна помощ и отговори на въпроси.

Възможности:

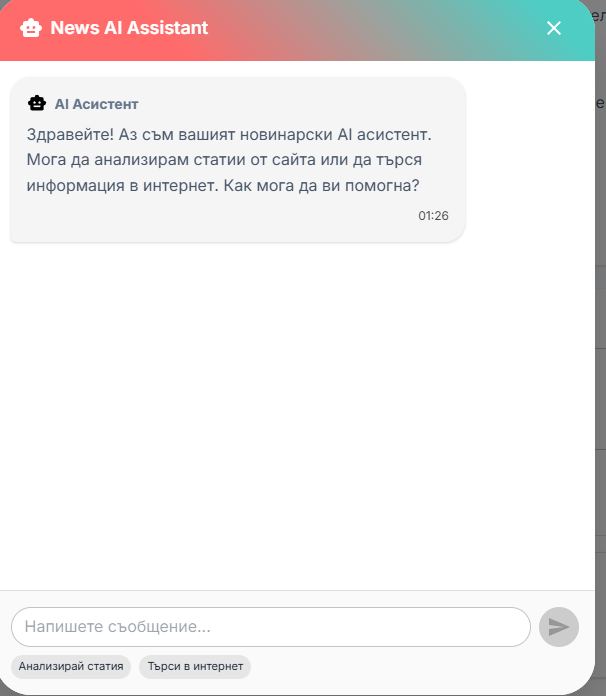
• Въпроси за конкретни статии

• Търсене на информация

• Резюмиране на съдържание

• Препоръки за четене

• Общи въпроси за сайта



Фиг. 18. AI ChatBot асистент

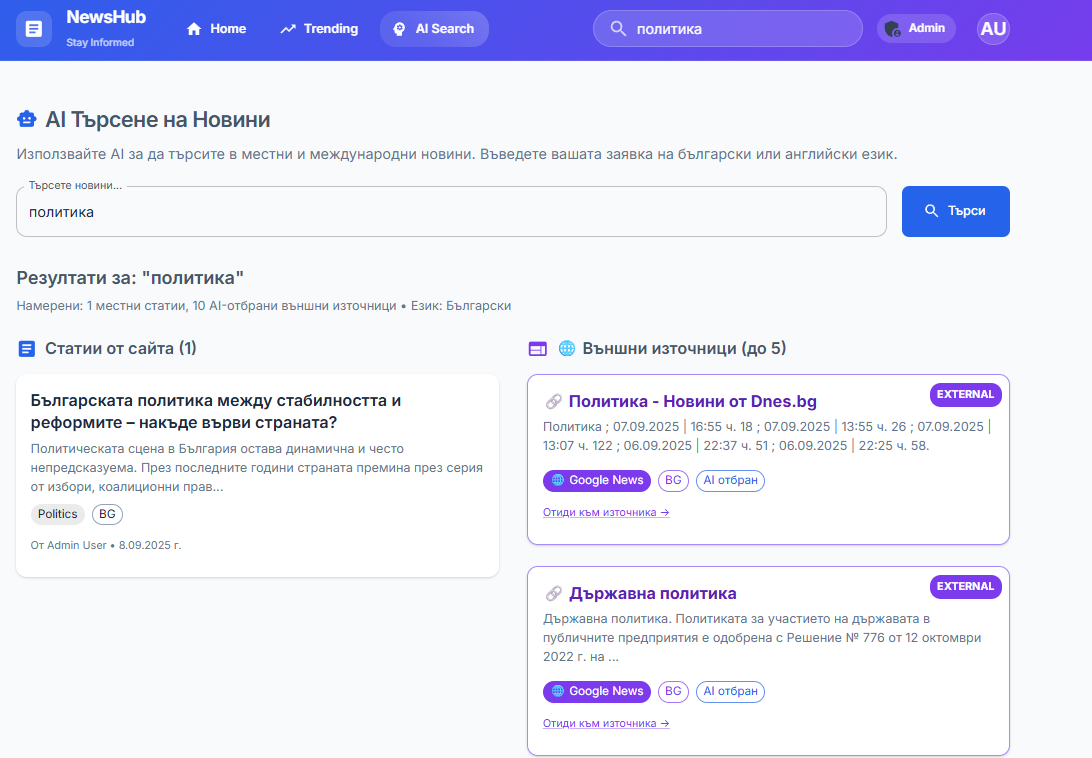
**Хибридно търсене:**

• Комбинация от локални и външни резултати

• Интеграция с Google Search

• Ранжиране на резултати

• Предложения за търсене



**VII. ЗАКЛЮЧЕНИЕ**

**Постигнати резултати**

Успешно е разработена съвременна система за управление на новини с интегриран AI чатбот, която покрива всички поставени цели и изисквания.

**Основни постижения:**

• Създадена е скалируема архитектура с ясно разделение на отговорностите

• Реализиран е интуитивен и отзивчив потребителски интерфейс

• Интегриран е мощен AI чатбот с GPT-4o за интерактивна помощ

• Имплементирана е хибридна система за търсене с високи точност

• Осигурена е надеждна система за автентификация и авторизация

• Създадени са административни инструменти за пълно управление

**Техническите решения се характеризират с:**

• Високопроизводителна backend архитектура с ASP.NET Core

• Модерен frontend с React.js и TypeScript

• Оптимизирана PostgreSQL база данни

• Интеграция с външни AI и търсещи услуги

• Responsive дизайн за всички типове устройства

**Възможности за развитие:**

Системата е проектирана с възможност за бъдещо разширение и подобрения:

Краткосрочни подобрения:

• Добавяне на push уведомления

• Интеграция със социални мрежи

• Подобрена SEO оптимизация

• Мобилно приложение

• Многоезична поддръжка

Дългосрочни разширения:

• Машинно обучение за персонализирани препоръки

• Автоматично генериране на съдържание с AI

• Разширена аналитика и reporting

• Интеграция с CMS системи

• Микросервисна архитектура за по-голяма скалируемост

Системата демонстрира успешната интеграция на съвременни уеб технологии с AI възможности за създаване на иновативно решение в областта на управлението на новинарско съдържание.

**VIII. ИЗПОЛЗВАНИ ИЗТОЧНИЦИ**

[1] ASP.NET Core Documentation - <https://docs.microsoft.com/en-us/aspnet/core/>

[2] Entity Framework Core - <https://docs.microsoft.com/en-us/ef/core/>

[3] PostgreSQL Documentation - <https://www.postgresql.org/docs/>

[4] React.js Official Documentation - <https://reactjs.org/docs/>

[5] TypeScript Handbook - <https://www.typescriptlang.org/docs/>

[6] Material-UI Documentation - <https://mui.com/>

[7] OpenAI API Documentation - <https://platform.openai.com/docs/>

[8] Google Custom Search API - <https://developers.google.com/custom-search/>

[9] Vite Documentation - <https://vitejs.dev/guide/>

[10] Clean Architecture Principles - Robert C. Martin

[11] RESTful API Design Best Practices - <https://restfulapi.net/>

[12] Modern Web Development with React - <https://react.dev/>