Annual of Konstantin Preslavsky University of Shumen
Faculty of Mathematics and Informatics
vol. XXVI C, 2025, pp. 94 – 102, https://doi.org/10.46687/DDLN5162

DESIGN AND IMPLEMENTATION OF A WEB-BASED CONTENT MANAGEMENT SYSTEM *

TODOR G. GROZDEV, HRISTO I. PARASKEVOV

ABSTRACT: *This paper presents the design and implementation of a web-based Content Management System (CMS) developed using modern technologies and frameworks such as React, Next.js, Tailwind CSS, PHP, MySQL, and Node.js. The main goal of the project is to create a functional and scalable platform that provides an intuitive user interface, a secure server architecture, and role-based access control. The article describes the selected technologies, the system architecture, and the installation and configuration process, as well as the organization of the codebase according to the MVC model. Performance testing confirms high responsiveness on the client side, enabled by the Single Page Application (SPA) architecture. The system is suitable for real-world deployment and provides a solid foundation for future development through API integrations and external system extensions.*

KEYWORDS: *content management system (CMS), web development, role-based access control*

2020 Math. Subject Classification: *68N30; 68U35*

ИЗРАБОТКА НА СОФТУЕРНА СИСТЕМА ЗА УПРАВЛЕНИЕ НА ИНТЕРНЕТ СЪДЪРЖАНИЕ †

ТОДОР Г. ГРОЗДЕВ, ХРИСТО И. ПАРАСКЕВОВ

*This paper is (partially) supported by Scientific Research Grant RD-08-106/05.02.2025 of Konstantin Preslavsky University of Shumen
†Статията е частично финансирана по проект № РД-08-106/05.02.2025 на ШУ „Еп. Константин Преславски“

1 Въведение

В съвременната дигитална среда, ефективното управление на интернет съдържание е от съществено значение за информационната активност на организации, бизнес структури и институции. С нарастващите изисквания за достъпност, бързодействие, сигурност и интеграция, необходимостта от съвременни и гъвкави софтуерни решения става все по-осезаема. Системите за управление на съдържание (Content Management Systems – CMS) предлагат платформа за създаване, редактиране и поддържане на уеб съдържание, без да е нужно дълбоко програмистко познание от страна на крайните потребители [1].

Настоящата статия представя резултатите от проектирането и реализацията на софтуерна система за управление на интернет съдържание, разработена с цел да отговори на актуалните технологични и функционални изисквания. Разработката се основава на утвърдени практики в съвременната уеб разработка и използва иновативни технологии като Next.js, React, Tailwind CSS, PHP/MySQL, Node.js, jQuery, Git и NPM, с фокус върху модулност, адаптивност и лесна поддръжка [2][3][4][5][6]

.

2 Цел, задачи и обхват на разработката

Основната цел на проекта е създаване на съвременна уеб базирана система за управление на съдържание, която да предлага интуитивен интерфейс, сигурна архитектура и гъвкава структура, подходяща както за крайни потребители, така и за разработчици и администратори. Разработката има за задача да демонстрира възможностите на съвременния технологичен стек при реализация на реална и функционална CMS платформа[1][2].

Конкретните задачи, изпълнени в рамките на разработката, включват:

- изграждане на двуезичен потребителски интерфейс с възможност за лесно добавяне на нови езикови локализации;

- осигуряване на сигурно и ефективно съхранение на съдържание чрез PHP/MySQL сървър с API комуникация[2];
- реализация на адаптивен (responsive) дизайн, който гарантира коректна визуализация на съдържанието върху различни видове устройства – компютри, таблети и мобилни телефони[5];
- прилагане на JSON API комуникация за интеграция с външни приложения и интерфейси;
- внедряване на версионен контрол чрез Git за проследимост на промените и поддръжка на дългосрочна разработка[6];
- използване на съвременни UI фреймуъркове и CSS библиотеки за модерен и функционален интерфейс[4][5].

Обхватът на изследването включва технологичен анализ, архитектурно моделиране, имплементация и тестване на ключови функционалности на системата. Статията разглежда избора на инструменти, принципите на организация на кода и интерфейса, както и възможностите за бъдещо разширение на платформата.

3 Инфраструктурна платформа и среди за разработка

3.1. Хостинг и платформена съвместимост

Изборът на подходяща хостинг среда е от решаващо значение за производителността и стабилността на уеб базираните системи. Разработената CMS система е базирана на AMP (Apache, MySQL, PHP) стек, което позволява широк спектър от хостинг конфигурации — от споделен хостинг до облачни решения[2]. В зависимост от очаквания трафик и обем на съдържанието, системата може да бъде разположена както на традиционни сървъри, така и в облачни среди като Amazon S3, където се реализира високоскоростен достъп до мултимедийни ресурси чрез субдомейни.

Клиентската версия на продукта изисква минимум следната сървърна конфигурация:

- Уеб сървър Apache 2.4+ или LiteSpeed;
- MySQL база данни версия 8.0+;
- PHP интерпретатор версия 8.2+ с активиран GD модул за работа с изображения[2].

За нуждите на разработката се използва локална инсталация чрез AMPPS, както и интеграция с Git, Node.js v23+, NPM v10+ и съвременни front-end библиотеки, което осигурява унифицирана среда за разработка и тестване[4][6][10]. Разграничението между продуктовата и разработващата среда е ясно дефинирано, което съдейства за по-добра поддръжка и версиционен контрол.

3.2. Фронтенд архитектура и потребителски интерфейс

Графичният интерфейс на системата е реализиран чрез двустепенен подход: административната зона е изградена върху Bootstrap 5, а клиентската част използва React/Next.js и Tailwind CSS, осигуряващи адаптивен дизайн и висока производителност[3][4][5]. Tailwind CSS е избран заради утилитарния си стил и възможностите за прецизен контрол на стилизацията, докато Bootstrap предлага готови компоненти, подходящи за административна логика[5].

Проектът използва Next.js архитектура, базирана на TypeScript, React и PostCSS, с автоматично маршрутизиране, предварително рендиране и API комуникация чрез Axios. Файловата структура на клиента е ясно структурирана в модули (Layout, Page, lib/api-client.ts), което улеснява поддръжката, мащабирането и интеграцията с външни системи[4].

4 Инсталационна процедура и функционалност на системата

4.1. Процес на инсталация и конфигурация

Инсталирането на системата е реализирано по два начина — чрез Git clone от публично хранилище или чрез ръчно разархивиране на публикуван дистрибутив. Инсталационният процес включва:

- настройка на локален AMP сървър;
- активиране на PHP разширения (напр. GD);
- създаване на MySQL база и потребител;
- изпълнение на уеб-базиран инсталатор през браузър;
- отделна инсталация и компилация на клиентската част чрез Node.js и NPM.

Допълнително се предлага препоръчана среда за разработка чрез Visual Studio Code, включваща Git интеграция и автоматично наблюдение на промените чрез npm run dev.

4.2. Функционалност, базирана на ролеви достъп

Системата е изградена с разграничение между администраторски и обикновени потребителски права. Администраторите разполагат с пълен достъп до:

- конфигурации на сайта (наименование, домейн, автор, слоган);
- редактор на менюта с поддръжка на йерархична структура;
- създаване и редакция на съдържание от различни типове (страници, новини, блог постове);
- управление на потребители и профили.

Потребителите, от своя страна, имат достъп само до съдържанието, което самите те са създали. Това осигурява контролирана среда, подходяща за мултиавторски платформи и съвместна работа, като едновременно запазва сигурността и целостта на данните.

5 Софтуерна архитектура и класова организация

Архитектурата на разработената CMS система следва обектно-ориентиран подход, базиран на разделяне на отговорности чрез специализирани класове: Core, Admin и API. Всеки от тях наследява обща логика и осигурява отделни функционални нива – сървърна логика, административен бекенд и комуникационен слой между клиента и сървъра.

Класът Core служи като базов компонент за достъп до база данни, обработка на входни параметри, работа с файлове и изображения, конвертиране на стойности и изпращане на електронни съобщения. Над него се изграждат класовете Admin и API, които разширяват функционалността чрез:

- управление на административната зона и достъпа до административни изгледи (Admin);
- обработка на AJAX заявки и структурирани отговори за React клиента в JSON формат (API).

Внедрен е ясен механизъм за разпределяне на правомощия и динамично зареждане на изгледи и компоненти според ролята на потребителя. По този начин архитектурата постига модулност, преизползваемост на код и лесна поддръжка.

6 Файлова структура и MVC организация

Изграждането на CMS системата следва концепцията на Model–View–Controller (MVC), при която се разделя бизнес логиката от потребителския интерфейс. Структурата на проекта е организирана в логически групирани директории [7][9]:

- /assets/classes/ съдържа основните класове (Core, Api, Admin);
- /assets/views/ – съдържа логиката за зареждане, редакция и изтриване на съдържание, менюта и потребителски профили;
- /assets/html/ – предоставя шаблони за потребителски и административен интерфейс на различни езици;
- /assets/scripts/ – включва JavaScript файлове за AJAX обработка, визуални ефекти и WYSIWYG редактор;
- /assets/styles/ – дефинира основни и специфични CSS стилове за оформяне на съдържанието;
- /templates/ – съдържа визуални теми, които могат да бъдат променяни от администратора.

Тази организация подпомага не само ефективната разработка, но и възможността за бъдещи разширения или интеграции чрез REST API или React компоненти.

7 Производителност и поведение при реални условия

С оглед практическото приложение на системата, е проведено тестване с акцент върху времето за зареждане и визуализация на съдържанието в браузърната среда. Благодарение на използването на React и архитектурата тип Single Page Application (SPA), се постига минимално време за реакция при смяна на страници и компоненти[3][4]. Само мултимедийните ресурси (снимки, видео) се зареждат динамично при необходимост, което допълнително оптимизира първоначалното зареждане на системата.

Тестовете показват, че клиентската част на системата отговаря на съвременните изисквания за производителност и потребителско изживяване, включително при динамична работа с API сървъра и натоварване със съдържание от различен тип.

8 Заключение и насоки за бъдещо развитие

Разработената софтуерна система за управление на интернет съдържание демонстрира ефективното прилагане на съвременни технологии в изграждането на модулна, сигурна и лесно разширяема уеб платформа. Чрез интеграцията на React, Next.js, Tailwind CSS, PHP/MySQL и Node.js, проектът успява да съчетае удобство за крайния потребител с гъвкавост за разработчиците и администраторите.

Използваната архитектура тип Single Page Application (SPA) и прилагането на ролево базиран достъп допринасят за висока производителност и сигурност, което прави системата подходяща за реална експлоатация. Внедреният MVC модел, ясната файлова структура и автоматизираната разработваща среда осигуряват добра поддръжка и възможност за бъдещо разширяване.

Предстоящите подобрения могат да включват:

- интеграция на изкуствен интелект за автоматични препоръки или категоризация на съдържание;
- разработване на мобилно приложение с React Native;
- внедряване на коментари, форуми и интерактивни компоненти;
- автоматичен превод на съдържание и по-добра поддръжка на многоезичност.

В заключение, разработката представлява устойчиво решение, което съчетава добри инженерни практики с висока приложимост, и може да служи като основа за по-широки софтуерни решения в сферата на управление на дигитално съдържание.

ЛИТЕРАТУРА:

[1] Deane Barker. Web Content Management: Systems, Features, and Best Practices. O'Reilly Media, 2016.

[2] Valade, Janet. PHP and MySQL Web Development. Pearson Education, 5th Edition, 2016.

[3] Alex Banks, Eve Porcello. Learning React: Functional Web Development with React and Redux. O'Reilly Media, 3rd Edition, 2022.

[4] Tim Neutkens, The Vercel Team. The Next.js Handbook. Vercel Docs, 2023.

[5] Adam Wathan et al. Tailwind CSS: From Zero to Production. Tailwind Labs, 2022.

[6] Flanagan, David. JavaScript: The Definitive Guide. O'Reilly Media, 7th Edition, 2020.

[7] Martin Fowler. Patterns of Enterprise Application Architecture. Addison-Wesley, 2002.

[8] Rouse, Margaret. Role-Based Access Control (RBAC) – Definition. TechTarget, [Online Resource].

[9] Eric Freeman, Elisabeth Robson. Head First Design Patterns. O'Reilly Media, 2nd Edition, 2020.

[10] Krolczyk, Adam. Modern Full-Stack Development: Using TypeScript, React, Node.js, Webpack, and Docker. Apress, 2020.

Тодор Гроздев
Сектор Пътна Полиция към ОД на МВР гр. Добрич
e-mail: todorgeorgievgrozdev@gmail.com

Христо Параскевов
Факултет по математика и информатика
Шуменски университет „Епископ К. Преславски“
E-mail: h.paraskevov@shu.bg