
Weiterentwicklung eines WordPress-Plugins zur gamifizierten Spendenverteilung: Gutenberg-Integration und Plugin-Architektur am Beispiel von “Charigame” in Kooperation mit der elancer-team GmbH

Bachelorarbeit zur Erlangung des akademischen Grades
Bachelor of Science
im Studiengang Medieninformatik
an der Fakultät für Informatik und Ingenieurwissenschaften
der Technischen Hochschule Köln

vorgelegt von: Christian Krenn

eingereicht bei: Prof. Dr. Hoai Viet Nguyen
Zweitgutachter*in: Tobias Derksen

Gummersbach, 01.09.2025

Kurzfassung / Abstract

Diese Arbeit beschreibt die Weiterentwicklung des WordPress-Plugins *Charigame*, einem WordPress Plugin für gamifizierte Spendenaktionen. Ausgangspunkt war eine historisch gewachsene Codebasis mit eingeschränkter Editierbarkeit, uneinheitlichen Strukturen und Sicherheitslücken.

Auf Basis der theoretischen Grundlagen zu WordPress, dem Gutenberg-Editor und Security-Standards wurde eine modulare Pluginarchitektur entworfen. Der Ansatz ist objektorientiert, trennt Verantwortlichkeiten und bindet den Gutenberg Editor für die Bearbeitung im Backend ein. Zentrale Klassen sind konzipiert und entwickelt worden darunter ein Login Handler, ein Donation Manager und ein Asset Manager.

In der Umsetzung wurde ACF Pro durch Carbon Fields ersetzt. Für das Plugin entwickelte Gutenberg-Blöcke machen die Landingpage-Inhalte im Editor vollständig bearbeitbar. PHPCS stellt die Einhaltung der WordPress Coding Standards sicher. Sicherheitsmaßnahmen wie Nonce-Validierung, Validierung/Sanitizing/Escaping und Session-Management wurden konsequent umgesetzt. Ein konsistentes Design-System auf Basis von Tailwind/shadcn verbessern Performance und UX.

Die Ergebnisse sind eine deutlich erhöhte Wartbarkeit, eine klare Strukturierung der Verantwortlichkeiten im Code, höhere Sicherheit und eine verbesserte Redaktionserfahrung im Backend. Ein neu gestaltetes Dashboard fasst Kennzahlen zusammen und erleichtert die Kampagnensteuerung.

Der Ausblick umfasst tiefergehende Analytics mit Exportfunktionen, Testautomatisierung, Internationalisierung sowie zusätzliche Spieltypen und Integrationen. Damit zeigt die Arbeit, wie sich bestehende WordPress-Plugins mit aktuellen Entwicklungspraktiken nachhaltig weiterentwickeln lassen.

Inhaltsverzeichnis

Tabellenverzeichnis	IV
Abbildungsverzeichnis	V
Glossar	VII
Abkürzungsverzeichnis	IX
1 Einleitung	1
1.1 Problemstellung und Motivation	1
1.2 Zielsetzung	2
1.3 Methodisches Vorgehen	3
2 Theoretische Grundlagen	4
2.1 WordPress als Content-Management-System	4
2.2 Plugin-Entwicklung mit WordPress	5
2.2.1 Grundlagen der Plugin-Architektur	5
2.2.2 Best Practices in der Plugin-Programmierung	9
2.2.3 Plugin Security und Privacy	11
2.2.4 Weiterführende Funktionen und Paradigmen	15
2.3 Gutenberg-Editor: Konzept und technische Grundlagen	17
2.3.1 Dateistruktur eines Blocks	17
2.3.2 Schlüsselkonzepte im Block Editor	18
2.4 Gamification im Kontext digitaler Anwendungen	19
2.5 Überblick über Corporate Social Responsibility und Charity-Plattformen	20
3 Projektkontext	22
3.1 Das Projekt Charigame	22
3.2 Deskriptiver Stand	24
3.2.1 WordPress Frontend und Backend	24
3.2.2 Verwendete Technologien	35
3.2.3 Architektur	36
4 Konzeption und Design	38
4.1 Stakeholderanalyse und Nutzungskontext	38

4.2	Anforderungen an die Weiterentwicklung	39
4.2.1	Technische Anforderungen	40
4.2.2	Funktionale Anforderungen	40
4.3	Abgrenzung des Projektumfangs	40
4.4	Plugin-Architektur	41
4.4.1	Einzusetzende Technologien	41
4.4.2	Aufbau der Komponenten	42
4.4.3	Custom Post Types	43
4.4.4	Verwaltungs- und Darstellungsebene	44
5	Implementierung	45
5.1	Entwicklungsumgebung und Tools	45
5.2	Plugin-Struktur und Architektur	46
5.2.1	Ausgangsbasis und Bereinigung	46
5.2.2	Migration und Anpassung der Komponenten	46
5.2.3	Backend-Integration und Dashboard	51
5.3	Gutenberg-Block-Entwicklung	54
5.4	Spendenverteilung und Schlussfolgerung	56
6	Fazit und Ausblick	57
6.1	Zusammenfassung der Ergebnisse	57
6.2	Einordnung der Zielerreichung	58
6.3	Reflexion und Grenzen	58
6.4	Ausblick	59
6.5	Schlussbemerkung	59
Literatur		60
Anhang		65

Tabellenverzeichnis

2.1	Übersicht der Plugin-Header-Felder in WordPress nach [8]	8
2.2	Übersicht Validierungs-Philosophien mit Beispielen (eigene Darstellung)	12
2.3	Auswahl wichtiger WordPress-Sanitization-Funktionen (eigene Darstellung)	13
2.4	Überblick zentraler Escaping-Funktionen in WordPress (eigene Darstellung)	14
1	Übersicht über konfigurierbare Eingabefelder der General Settings (eigene Darstellung)	72
2	Übersicht über konfigurierbare Eingabefelder der Donation Recipients (eigene Darstellung)	73
3	Übersicht über konfigurierbare Eingabefelder der Game Types (eigene Darstellung)	73
4	Übersicht über konfigurierbare Eingabefelder der User (eigene Darstel- lung)	74
5	Übersicht über angezeigte Metriken der Data Table (eigene Darstellung)	75
6	Übersicht über die Eingabefelder der Intro Section (eigene Darstellung)	76
7	Übersicht über die Eingabefelder der Recipient Section (eigene Dar- stellung)	76
8	Übersicht über die Eingabefelder der Donation Section (eigene Dar- stellung)	77
9	Übersicht über die Eingabefelder eines How-to-Play-Items (eigene Dar- stellung)	77
10	Übersicht über die Eingabefelder der Game Section (eigene Darstellung)	78
11	Übersicht über die Eingabefelder des Email Templates (eigene Darstel- lung)	79

Abbildungsverzeichnis

2.1	Unveränderte WordPress-Verzeichnisstruktur der Version 6.8.2 (eigene Darstellung)	6
2.2	Empfohlene Plugin Ordnerstruktur (eigene Darstellung angelehnt an WordPress Developer Resources [7])	7
2.3	Minimaler Plugin-Header in WordPress (eigene Darstellung angelehnt an WordPress Developer Resources [8])	7
2.4	Dateistruktur eines WordPress-Blocks (eigene Darstellung)	18
3.1	Funktionsweise <i>Charigame</i> (eigene Darstellung)	23
3.2	Charigame Menü im WordPress Dashboard (eigene Darstellung)	24
3.3	Screenshot aus dem WordPress-Backend (eigene Darstellung). Enthaltene Medien: Hintergrundbild © Aditya Romansa / Unsplash [39], Logo © WordPress Foundation [40].	25
3.4	Login-Form einer <i>Charigame</i> -Testkampagne im Frontend (eigene Darstellung)	26
3.5	Donation Recipients im WordPress Backend Enthaltenes Logos © WWF [41] (eigene Darstellung)	27
3.6	Donation Recipients im WordPress Frontend (eigene Darstellung) Enthaltene Logos © UNICEF, Deutsches Rotes Kreuz, WWF [41]	27
3.7	Game Types im WordPress Backend (eigene Darstellung)	28
3.8	Game Types im WordPress Frontend (eigene Darstellung)	29
3.9	Users im WordPress Backend (eigene Darstellung)	30
3.10	Pipedrive im WordPress Backend (eigene Darstellung)	30
3.11	E-Mail Settings im WordPress Backend (eigene Darstellung)	32
3.12	Data Table im WordPress Backend (eigene Darstellung)	33
4.1	Stakeholderanalyse Matrix (eigene Darstellung)	38
4.2	Relation der Custom Post Types (eigene Darstellung)	44
5.1	Aufbau der Klasse Charigame_Carbon_Fields (eigene Darstellung)	47
5.2	Donation Recipients im WordPress Backend Enthaltenes Logos © WWF [41] (eigene Darstellung)	48
5.3	Neu aufgebautes Dashbord im Backend (eigene Darstellung)	52

5.4	E-Mail-Template-System im Gutenberg Editor (eigene Darstellung) . . .	53
5.5	Automatisierte Gutenberg Blockregistrierung (eigene Darstellung) . . .	54
1	Dateistruktur eines WordPress-Blocks [29]	65
2	Gesamte Landingpage von Charigame im WordPress Frontend Enthaltene Logos © UNICEF, Deutsches Rotes Kreuz, WWF [41] (eigene Darstellung)	66
3	Spendenverteilungsseite von Charigame im WordPress Frontend Enthaltene Logos © UNICEF, Deutsches Rotes Kreuz, WWF [41] (eigene Darstellung)	67
4	Dankesseite von Charigame im WordPress Frontend Enthaltene Logos © UNICEF, Deutsches Rotes Kreuz, WWF [41] (eigene Darstellung)	68
5	Backendansicht der Intro-Section (eigene Darstellung)	68
6	Backendansicht der Recipient-Section Enthaltene Logos © UNICEF, Deutsches Rotes Kreuz, WWF [41] (eigene Darstellung)	69
7	Backendansicht der Donation-Section (eigene Darstellung)	69
8	Backendansicht der How-To-Play-Section Enthaltene Logos © UNICEF, Deutsches Rotes Kreuz, WWF [41] (eigene Darstellung)	69
9	Backendansicht der Game-Section (eigene Darstellung)	70
10	Backendansicht der Email-Edit-Section (eigene Darstellung)	70
11	Gesamte Landingpage in der Backendansicht Enthaltene Logos © UNICEF, Deutsches Rotes Kreuz, WWF [41] (eigene Darstellung)	71

Glossar

Action Hook-Typ in WordPress, der es erlaubt, zu bestimmten Ereignissen eigenen Code auszuführen (z. B. bei Plugin-Aktivierung). 9, 44

Block Editor (Gutenberg) Seit WordPress 5 eingeführter Editor, der Inhalte aus modularen Blöcken zusammensetzt und direkt im Backend bearbeitbar macht. 17, 42, 43

Carbon Fields PHP-Bibliothek zur Definition von Feldern und Metadaten in WordPress, genutzt als Alternative zu ACF Pro. 41, 46, 47

Content-Management-System Softwaresystem zur Erstellung, Verwaltung, Bearbeitung und Publikation von digitalen Inhalten wie Texten, Bildern, Videos und Multimedia-Dokumenten. 4, 5

Corporate Social Responsibility Unter "Corporate Social Responsibility" (CSR) ist die gesellschaftliche Verantwortung von Unternehmen im Sinne eines nachhaltigen Wirtschaftens zu verstehen.[42]. 1, 20

Custom Post Type Ein Custom Post Type ist ein individuell definierter Inhaltstyp in WordPress. Er wird über die WordPress Core Post API registriert und speichert die Daten in der posts-Tabelle der Datenbank. Ein Beispiel für einen Core-CPT ist der Datentyp „Seite“ (page).[45]. 43

dbDelta WordPress-Funktion zum Erstellen/Aktualisieren von Datenbanktabellen anhand einer Schema-Definition. 37

Enqueue Verfahren in WordPress zum korrekten Registrieren und Laden von Skripten und Styles (wp_enqueue_script, wp_enqueue_style). 35

Filter Hook-Typ in WordPress, der Daten vor der Ausgabe verändert, indem Rückgabewerte durch eigene Funktionen gefiltert werden. 9

Grunt JavaScript-Task-Runner zur Automatisierung von Build- und Wartungsaufgaben. 36

Hook Erweiterungspunkt in WordPress, an dem eigener Code ausgeführt werden kann. Unterschieden wird zwischen Actions und Filtern. 9, 42

jQuery JavaScript-Bibliothek zur DOM-Manipulation und Eventbehandlung, häufig über ein CDN eingebunden. 35

Nonce Einmal-Token in WordPress zur Absicherung von Formularen und Aktionen gegen Cross-Site Request Forgery. 15, 50

Object Cache Caching-Schicht in WordPress zum Zwischenspeichern von Objekten und Datenbankergebnissen für bessere Performance. 50

PHPMailer Bibliothek zum Versenden von E-Mails in PHP, in WordPress über wp_mail nutzbar. 35

Plugin Softwareerweiterung bestehend aus Codepaketen, die die Kernfunktionalität von WordPress erweitern. Plugins bestehen aus PHP-Code und können weitere Assets wie Bilder, CSS und JavaScript enthalten.
(eigene Übersetzung nach [5]). 1, 5–7, 17

Tailwind CSS Utility-First CSS-Framework, das über einen CLI/Build-Prozess Klassen generiert und in Projekten eingebunden wird. 36, 42

Template Include Mechanismus in WordPress, um die geladene Template-Datei programmgesteuert zu überschreiben bzw. zu steuern. 35

Three.js JavaScript-3D-Bibliothek zur Darstellung von 3D-Grafiken im Browser. 35

WP List Table WordPress-API zur Erstellung tabellarischer Listenansichten im Admin-Dashboard. 35

WP-CRON WordPress-eigener Pseudo-Cron zur zeitgesteuerten Ausführung von Aufgaben (Cron API), getriggert durch Seitenaufrufe. 16, 35

wpdb WordPress-Datenbankschnittstelle (Klasse *wpdb*) für direkte SQL-Abfragen und Datenbankoperationen. 37

Abkürzungsverzeichnis

ACF PRO Advanced Custom Fields Pro. 3

AJAX Asynchronous JavaScript and XML. 16, 35, 36, 44

API Application Programming Interface. 4, 16, 36

B2B Business-to-Business. 39

B2C Business-to-Consumer. 39

CDN Content Delivery Network. 35

CI Corporate Identity. 72

CLI Command Line Interface. 36, 37

CRM Customer-Relationship-Management. 30

CSRF Cross-Site Request Forgery. 15

CSS Cascading Style Sheets. 5, 36, 37

CTA Call to Action. 35

GNU GNU's Not Unix. 4

GPL General Public License. 4

GSAP GreenSock Animation Platform. 35, 37

HTML Hypertext Markup Language. 13, 14, 18

HTTP Hypertext Transfer Protocol. 4

JS JavaScript. 35

JSON JavaScript Object Notation. 16

KPI Key Performance Indicator. 36

MP4 Moving Picture Experts Group 4 Part 14. 72

MySQL My Structured Query Language. 4

NPM Node Package Manager. 35, 36

ORM Object-Relational Mapping. 37

PHP Hypertext Preprocessor. 4, 5, 10, 36

PHPCS PHP CodeSniffer. 41

REST Representational State Transfer. 4, 16

SMTP Simple Mail Transfer Protocol. 31, 36

SSL Secure Sockets Layer. 31

TLS Transport Layer Security. 31

URI Uniform Resource Identifier. 8

URL Uniform Resource Locator. 8, 14–16

XML Extensible Markup Language. 14

XSL Extensible Stylesheet Language. 14

XSS Cross-Site-Scripting. 13

1 Einleitung

Die vorliegende Arbeit behandelt die Weiterentwicklung des WordPress-Plugins *Charigame*. Hierzu wird vornehmlich die Gutenberg-Integration und Plugin-Architektur betrachtet. Das WordPress-Plugin *Charigame* der Agentur elancer-team GmbH bietet Unternehmen die Möglichkeit, Spendenaktionen über gamifizierte Inhalte online anzubieten. Die Nutzer beteiligen sich an interaktiven Kampagnen, die als Memory- oder Geschicklichkeitsspiele gestaltet sind. Je nach Interaktion der Kunden wird bestimmt, welche sozialen Projekte aus einem vom Unternehmen bereitgestellten Spendentopf unterstützt werden. Hierdurch werden unternehmerische Corporate Social Responsibility-Maßnahmen mit spielerischer Nutzerbeteiligung kombiniert.

„Clean code always looks like it was written by someone who cares.“

— Robert C. Martin

Diese Philosophie bildet die Grundlage für die Analyse und Weiterentwicklung der bestehenden Struktur. Für diese Weiterentwicklung wird der deskriptive Zustand des Projekts beleuchtet. Anschließend wird ein Konzept erarbeitet, das auf den zuvor gesammelten theoretischen Erkenntnissen aufbaut. Die Ausarbeitung sieht vor, das Plugin *Charigame* wartungsfreundlicher zu gestalten und Entwicklern mehr Flexibilität bei Anpassungen zu bieten. Des Weiteren gilt es Redakteuren eine intuitive Nutzung im WordPress-Backend und die Konfiguration von bestehenden Bereichen zu vereinfachen sowie neue Anpassungsmöglichkeiten bereitzustellen.

1.1 Problemstellung und Motivation

Die deskriptive Codebasis des WordPress-Plugins *Charigame* weist einen strukturellen Optimierungsbedarf auf, der auf die Entstehungsgeschichte des Plugins zurückzuführen ist. Ursprünglich wurde das Plugin als internes Projekt der elancer-team GmbH entwickelt und später gezielt auf die Bedürfnisse eines spezifischen Kunden angepasst.

Während dieser Phase der kundenspezifischen Entwicklung entstanden viele Funktionen, die stark auf diesen besonderen Anwendungsfall zugeschnitten sind. Eine übergreifende und flexibel erweiterbare Architektur wurde hierbei nicht berücksichtigt. Das Hauptproblem liegt darin, dass die aktuelle Implementierung gängige WordPress-Standards nur unzureichend berücksichtigt. Aufgrund des ursprünglich gesetzten Zeitrahmens entstand eine funktionale Codebasis, die jedoch nicht vollständig den etablierten Best Practices der WordPress-Plugin-Entwicklung entspricht. Das zeigt sich beispielsweise in mangelnder Modularität, lückenhafter Dokumentation und einer Struktur, die weitere Anpassungen erschwert.

Motivation

Die Motivation für diese Arbeit ergibt sich aus drei zentralen Aspekten:

- **Eigene Weiterentwicklung:** Die Neuausrichtung des Plugins bietet die Chance, sich intensiv mit modernen WordPress-Entwicklungsmethoden zu beschäftigen. Besonders im Bereich modularer Architektur und Block-Entwicklung im Gutenberg-Umfeld können tiefgehende Erkenntnisse gesammelt werden.
- **Technologischer Fortschritt:** Die Integration des Gutenberg-Editors als zukunftsähnlicher, visueller Backend-Builder bringt das Plugin auf den neuesten Stand der WordPress-Technologie. Das verbessert nicht nur die redaktionelle Nutzererfahrung, sondern schafft auch die Basis für eine nachhaltigere technische Grundlage.
- **Mehrwert für die Agentur:** Eine klar strukturierte, wartbare Codebasis bringt konkrete Vorteile für die elancer-team GmbH. Sie ermöglicht eine effizientere Teamarbeit, erleichtert den Wissenstransfer und verringert die Abhängigkeit von Einzelpersonen. Langfristig schafft das mehr Flexibilität, Stabilität und Skalierbarkeit im Projekt.

Die Weiterentwicklung des Plugins ist somit ein logischer nächster Schritt.

1.2 Zielsetzung

Das Ziel dieser Arbeit ist die technische Weiterentwicklung des bestehenden WordPress-Plugins *Charigame* mit besonderem Fokus auf Wartbarkeit, Standardkonformität und redaktionelle Nutzbarkeit. Durch die Anpassung der Architektur soll der Code strukturierter und anhand der WordPress-Plugin-Standards gestaltet werden.

Diese Arbeit beschäftigt sich mit den theoretischen Grundlagen und der Anpassung des Plugins an bewährte WordPress-Standards. Zunächst wird der aktuelle Entwicklungsstand analysiert, anschließend werden Potenziale identifiziert und Anpassungen konzipiert. Auf Grundlage des Konzepts erfolgt dann die praktische Umsetzung. Ein zentraler Aspekt der Zielsetzung ist es, den Gutenberg Editor einzubinden und die Struktur der Codegrundlage zu optimieren. Dabei werden verschiedene Lösungsansätze betrachtet und die getroffenen Entscheidungen transparent erläutert.

1.3 Methodisches Vorgehen

Die Zielerreichung erfolgt praxisnah und methodisch strukturiert. Damit das gesetzte Ziel erreicht werden kann, wird ein methodisches Vorgehen durchgeführt, das sich in vier größere Bestandteile gliedert lässt.

1. **Recherche und Wissensaufbau der theoretischen Grundlagen:** Zuerst wird theoretisches Fachwissen im Bezug auf die WordPress Plugin Programmierung und den Gutenberg Editor gebündelt und verschriftlicht. Dies stellt die Grundlage für den theoretischen Teil dar.
2. **Ermitteln des deskriptiven Stands:** Im zweiten Schritt wird die aktuelle Plugin-Struktur auf Basis der offiziellen WordPress-Guidelines beleuchtet. Hier wird grundlegend eine modulare Architektur verfolgt, damit die langfristige Wartbarkeit und Erweiterbarkeit gegeben ist.
3. **Konzeption der Umsetzung:** Die Konzeption der Änderungen werden im dritten Schritt erfasst und dienen als Grundlage für die Anpassung des Systems. Darauf hinaus gilt es eine Reduktion externer Abhängigkeiten zu schaffen. Im Detail ist hier die Nutzung von Advanced Custom Fields Pro (ACF PRO) gemeint, welche durch eine andere lizenzzfreie Lösung ersetzt werden soll.
4. **Integration des Gutenberg-Editors:** Der letzte Schritt sieht vor, eine Blockbasierte Verwaltung im Backend zu entwickeln. Diese Lösung soll auf dem Gutenberg-Editor basieren und somit den aktuellen Standards von WordPress entsprechen.

Die Arbeit beginnt mit dem Erfassen und Sammeln der theoretischen Grundlagen hinsichtlich der WordPress-Entwicklungsstandards. Darauf folgt eine Analyse der aktuellen Codebasis und der bestehenden Architektur, worauf basierend ein neues technisches Konzept erarbeitet und implementiert wird. Die Umsetzung wird schließlich im Hinblick auf Struktur, Funktionalität und redaktionelle Bedienbarkeit evaluiert.

2 Theoretische Grundlagen

In diesem Kapitel werden die zentralen theoretischen Grundlagen vermittelt, die für das Verständnis der Arbeit und die spätere Umsetzung notwendig sind. Es werden sowohl technische Aspekte von WordPress und der Plugin-Entwicklung als auch die Prinzipien des Gutenberg-Editors betrachtet. Ergänzend werden konzeptionelle Überlegungen zu Gamification und gesellschaftlich relevanten Plattformen diskutiert, um den fachlichen Rahmen der Arbeit abzugrenzen.

2.1 WordPress als Content-Management-System

WordPress ist ein freies, unter der GNU's Not Unix (GNU) General Public License (GPL) v2 lizenziertes Content-Management-System und mit einem Marktanteil von 61,1% das weltweit am häufigsten verwendete Content-Management-System [1]. Die Plattform zeichnet sich durch seine Flexibilität, Erweiterbarkeit und Bedienbarkeit aus, was WordPress zu einer Lösung für simple Websites und Blogs bis hin zu komplexen Web-Applikationen macht [2]. Die technische Struktur verfolgt ein modulares Paradigma, das eine klare Trennung zwischen Kernfunktionen, Design und Erweiterungen ermöglicht:

- Themes: Kontrollieren die Präsentationslogik und das Design der Website
- Plugins: Erweitern die Funktionalität
- Core System: Stellt die grundlegenden Funktionen des Content-Management-System bereit

Technisch gesehen besteht WordPress aus einer Hypertext Preprocessor (PHP)-basierten Architektur in Verbindung mit der persistenten Speicherlösung My Structured Query Language (MySQL) als relationale Datenbank. Durch diese weit verbreiteten Technologien ist WordPress mit den meisten gängigen Hosting-Umgebungen kompatibel. Grundlage hierfür ist ein Webserver, welcher PHP und MySQL unterstützt. Offiziell ist Apache oder Nginx empfohlen [3].

Darüber hinaus bietet WordPress eine Representational State Transfer (REST)-Application Programming Interface (API) mit der eine entkoppelte Inhaltsverwaltung möglich ist und Inhalte über das Hypertext Transfer Protocol (HTTP)-Protokoll in

verschiedene Anwendungen und Plattformen integriert werden können.

WordPress Coding Standards

WordPress Coding Standards dienen als Richtlinien und Best Practices für Entwickler die an WordPress Projekten arbeiten. Diese Standards haben sich in der WordPress Community etabliert und ermöglichen eine konsistente Codestruktur. Ziel ist es, die Codequalität so zu gestalten, dass Wartung und kollaborative Weiterentwicklung erleichtert werden. Für die Ausarbeitung der Weiterentwicklung von *Charigame* sollen die offiziellen WordPress Coding Standards beachtet und umgesetzt werden [4].

2.2 Plugin-Entwicklung mit WordPress

Die offizielle WordPress-Definition besagt: „Plugins sind Codepakete, die die Kernfunktionalität von WordPress erweitern. WordPress-Plugins bestehen aus PHP-Code und können weitere Assets wie Bilder, Cascading Style Sheets (CSS) und JavaScript enthalten“ [5] (eigene Übersetzung).

Für die Entwicklung solcher Plugins ist grundlegend ein tiefergehendes Verständnis des WordPress-Content-Management-System vorausgesetzt. Die Grundlagen sowie weiterführende Themengebiete sollen im Folgenden ermittelt werden.

2.2.1 Grundlagen der Plugin-Architektur

Die Entwicklung von Plugins in WordPress setzt ein Basiswissen der Dateistruktur und des internen Aufbaus des Content-Management-Systems voraus. Insbesondere ist es erforderlich, die Plugin-bezogenen Dateien korrekt zu strukturieren und im entsprechenden Verzeichnis innerhalb der WordPress-Installation zu platzieren.

Plugins werden standardmäßig im Verzeichnis `wp-content/plugins` abgelegt. Jeder Plugin-Ordner enthält dabei alle notwendigen Dateien zur Funktionalität des jeweiligen Moduls.

Im folgenden Schaubild ist eine typische, nicht modifizierte Verzeichnisstruktur einer WordPress-Installation dargestellt. Diese dient als Ausgangspunkt für die Entwicklung und Integration von Plugins:

Name	Größe	Art
.htaccess	553 Byte	Dokument
index.php	405 Byte	PHP-Skript
license.txt	20 KB	Reiner Text
local-xdebuginfo.php	20 Byte	PHP-Skript
readme.html	7 KB	HTML-Text
wp-activate.php	7 KB	PHP-Skript
wp-admin	--	Ordner
wp-blog-header.php	351 Byte	PHP-Skript
wp-comments-post.php	2 KB	PHP-Skript
wp-config-sample.php	3 KB	PHP-Skript
wp-config.php	3 KB	PHP-Skript
wp-content	--	Ordner
index.php	28 Byte	PHP-Skript
languages	--	Ordner
plugins	--	Ordner
themes	--	Ordner
uploads	--	Ordner
wp-cron.php	6 KB	PHP-Skript
wp-includes	--	Ordner
wp-links-opml.php	3 KB	PHP-Skript
wp-load.php	4 KB	PHP-Skript
wp-login.php	51 KB	PHP-Skript
wp-mail.php	9 KB	PHP-Skript
wp-settings.php	30 KB	PHP-Skript
wp-signup.php	35 KB	PHP-Skript
wp-trackback.php	5 KB	PHP-Skript
xmlrpc.php	3 KB	PHP-Skript

Abbildung 2.1: Unveränderte WordPress-Verzeichnisstruktur der Version 6.8.2 (eigene Darstellung)

Abbildung 2.1 visualisiert, dass sich im wp-content Ordner bereits initial in der Standardkonfiguration von WordPress das Verzeichnis plugins befindet. Der plugins Ordner ist die zentrale Destination für alle im WordPress System erstellten Erweiterungen. Die nachfolgende Untersuchung berücksichtigt weitere Bestandteile von WordPress-Plugins, die fortlaufend detailliert beschrieben werden.

Aufbau, Plugin-Header und Metadateien

Für die Entwicklung von WordPress-Plugins empfehlen die offiziellen WordPress Developer Resources eine klare und strukturierte Ordnerorganisation [6]. Als empfohlen wird folgende grundlegende Verzeichnisstruktur vorgeschlagen (vgl. Abbildung 2.2):

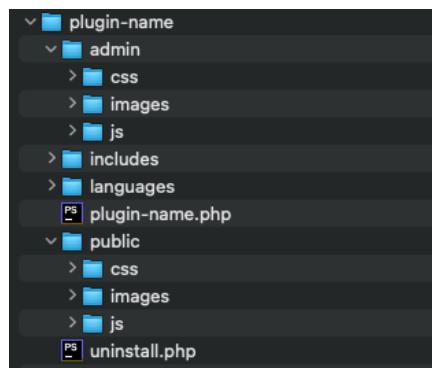


Abbildung 2.2: Empfohlene Plugin Ordnerstruktur (eigene Darstellung angelehnt an WordPress Developer Resources [7])

Die Datei `plugin-name.php` stellt im gezeigten Beispielaufbau die zentrale Plugin-Datei dar. In dieser Datei muss eine von WordPress vorgegebene Headerstruktur enthalten sein, damit das System die Datei als Plugin erkennt und korrekt lädt. Das minimale erforderliche Header-Format ist dabei wie in Abbildung 2.3 definiert:

```

  plugin-name.php
  -----
  1  <?php
  2  /*
  3  * Plugin Name: YOUR PLUGIN NAME
  4  */
  
```

Abbildung 2.3: Minimaler Plugin-Header in WordPress (eigene Darstellung angelehnt an WordPress Developer Resources [8])

Darüber hinaus können weitere optionale Felder definiert werden wie z. B. Author URI für die Website des Plugin-Entwicklers, Text Domain zur Internationalisierung und Domain Path für den Pfad zu Sprachdateien. Alle möglichen Felder werden in der folgenden Tabelle aufgeführt:

Tabelle 2.1: Übersicht der Plugin-Header-Felder in WordPress nach [8]

Feldname	Beschreibung
Plugin Name (erforderlich)	Der Name des Plugins, wie er in der Plugin-Übersicht im WordPress-Adminbereich angezeigt wird.
Plugin Uniform Resource Identifier (URI)	Eine eindeutige Uniform Resource Locator (URL) zur Startseite des Plugins, idealerweise auf der eigenen Website. WordPress.org-URLs sind hier nicht erlaubt.
Description	Eine kurze Beschreibung des Plugins (max. 140 Zeichen), wie sie im WordPress-Adminbereich angezeigt wird.
Version	Die aktuelle Versionsnummer des Plugins, z. B. 1.0 oder 1.0.3.
Requires at least	Die minimale WordPress-Version, mit der das Plugin kompatibel ist.
Requires PHP	Die minimale benötigte PHP-Version für das Plugin.
Author	Der Name des Autors oder der Autoren des Plugins. Mehrere Autoren können durch Kommata getrennt angegeben werden.
Author URI	Die Website des Autors oder ein öffentliches Profil, z. B. auf WordPress.org.
License	Die Kurzbezeichnung der Lizenz, unter der das Plugin veröffentlicht wird (z. B. GPLv2).
License URI	Ein Link zum vollständigen Lizenztext (z. B. https://www.gnu.org/licenses/gpl-2.0.html).
Text Domain	Die Textdomain für die Internationalisierung des Plugins, erforderlich für Übersetzungen mit <code>gettext</code> .
Domain Path	Das Verzeichnis, in dem WordPress die Übersetzungssdateien findet (z. B. <code>/languages</code>).
Network	Gibt an, ob das Plugin netzwerkweit (Multisite) aktiviert werden kann. Nur <code>true</code> ist erlaubt, andernfalls Feld weglassen.
Update URI	Eine eindeutige URI zur Updatequelle, um versehentliche Überschreibungen durch gleichnamige Plugins im WordPress.org-Verzeichnis zu vermeiden.

Fortsetzung auf nächster Seite

Fortsetzung von vorheriger Seite

Feldname	Beschreibung
Requires Plugins	Eine durch Kommata getrennte Liste von Plugin-Slugs aus dem WordPress.org-Verzeichnis, die als Abhängigkeit benötigt werden. Kommata innerhalb der Slugs sind nicht erlaubt.

Hooks und Filter

Es gibt eine Kardinalregel in der WordPress Entwicklung die besagt: Don't touch WordPress core [9]. Damit Anpassungen an bestehende Funktionalitäten möglich sind, gibt es Hooks und Filters. Hooks erlauben es, an spezifischen Stellen einzugreifen, um das Verhalten von WordPress zu ändern, ohne dabei die Kern-Dateien bearbeiten zu müssen. Allgemein gibt es zwei Arten von Hooks in WordPress: Actions und Filters. Mit Actions können Funktionen hinzugefügt oder geändert werden. Filters wiederum ändern Inhalte, während diese geladen und dem Website-Nutzer angezeigt werden. Hooks sind nicht nur für die Plugin-Entwicklung gedacht, sondern werden ebenfalls häufig verwendet, um Standardfunktionen durch den WordPress-Kern selbst bereitzustellen.

Zu den zentralen Hooks im Kontext von WordPress-Plugins gehören insbesondere `register_activation_hook()`, `register_deactivation_hook()` sowie `register_uninstall_hook()` [10].

- **register_activation_hook()**: Dieser Hook wird ausgeführt, wenn das Plugin im WordPress-Backend aktiviert wird. Dies wird oftmals verwendet, um Funktionen zwecks Einrichtung des Plugins bereitzustellen.
- **register_deactivation_hook()**: Der Deaktivierungs-Hook wird ausgeführt, wenn das Plugin deaktiviert wird. Diese Funktion ermöglicht es die vom Plugin temporär gespeicherten Daten und Einträge zu entfernen.
- **register_uninstall_hook()**: Die Ausführung findet statt, wenn das Plugin aus dem Backend gelöscht wird. Die Verwendung zielt häufig darauf ab, alle vom Plugin erstellen Optionen oder Datenbanktabellen restlos zu löschen.

2.2.2 Best Practices in der Plugin-Programmierung

Die WordPress Developer Resources geben dem Entwickler einige Best Practices vor, welche helfen den Code zu organisieren. Ferner wird durch die Hinweise sichergestellt, dass der Code verlässlich mit dem WordPress Kern und anderen Plugins funktioniert.

Nachfolgend sollen zentrale Best Practices aufgeführt werden, welche im Verlauf der Arbeit in der Konzeption berücksichtigt werden.

Vermeiden von Namenskonflikten

Namenskollisionen können durch die gleiche Benennung von Variablen, Funktionen oder Klassen zustande kommen. Um einen solchen Namenskonflikt zu vermeiden, wird es empfohlen, einen globalen Namespace zu definieren. Dieser Namespace ermöglicht das Überschreiben von anderen Variablen, Funktionen und Plugins. Innerhalb von Funktionen und Klassen definierte Variablen, sind hiervon nicht betroffen [11].

Voranstellen eines Prefix

Unter Zuhilfenahme eines Prefix können potenzielle Konflikte im Hinblick auf Aufrufe und Ausführungen mit anderen Plugins verhindert werden. Das Handbuch empfiehlt hierzu ein einzigartiges Wort, welches vor verschiedene Deklarationen vorangestellt wird [12]. Auf das *Charigame* gemünzte Projekt könnte dies dann wie folgt aussehen:

- function charigame_save_post();
- define ('CHARIGAME_LICENSE', true);
- class CHARIGAME_Admin
- namespace ChariGame;
- update_option('charigame_settings', \$settings)

Prüfung vorhandener Implementierungen

Damit der Code robust und weniger fehleranfällig ist, gilt es in Anbetracht der Best Practices auf bestehende Implementationen von Programmierelementen zu prüfen [13]. Dies geschieht unter Zuhilfenahme der folgenden von PHP zur Verfügung gestellten Funktionen:

- Variablen: isset()
- Function_exists()
- Classes class_exists()
- Constants defined()

Architekturmuster

Die WordPress Developer Resources geben eine Hand voll mögliche Architekturmuster vor. Diese können grob in drei Variationen kategorisiert werden:

- Einzelne Plugin-Datei, die Funktionen enthält
- Einzelne Plugin-Datei, die eine Klasse, ein instanziertes Objekt und optionale Funktionen enthält
- Eine Haupt-Plugin-Datei, sowie eine oder mehrere Klassendateien

Je nachdem welchen Umfang das Plugin anstrebt, gilt es die passende Lösung des Architekturmusters zu wählen. Eine einzelne Plugin-Datei, die Funktionen enthält, stellt eine solide Basis für ein kleines Plugin zur Verfügung. Die Variante eine Haupt-Plugin-Datei zu erstellen und mehrere Klassendateien aufzubauen, bietet sich eher für Plugins mit einer größeren Codebasis an.

2.2.3 Plugin Security und Privacy

Ein essenzieller Aspekt bei der Programmierung von WordPress Plugins ist die Plugin Security und Privacy. Damit die Sicherheit von Eingaben gegeben ist, werden Methoden wie Validating, Sanitizing und Escaping genutzt.

Validating

Das Validieren von Eingaben ist der Prozess, bei dem Daten anhand eines vordefinierten Musters (oder mehreren Mustern) mit einem eindeutigen Ergebnis getestet werden: gültig oder ungültig. Die Validierung ist im Vergleich zur Bereinigung ein spezifischerer Ansatz, aber beide haben ihre Berechtigung.

(eigene Übersetzung nach [14])

Es werden auch seitens der WordPress Developer Ressourcen einfache Validierungsbeispiele genannt [14]:

- Überprüfung, ob Pflichtfelder ausgefüllt wurden
- Überprüfung, ob eine eingegebene Telefonnummer nur Zahlen und Satzzeichen enthält
- Überprüfung, ob eine eingegebene Zeichenfolge eine von fünf gültigen Optionen ist
- Überprüfung, ob ein Mengenfeld größer als 0 ist

Die Validierung selbst wird in verschiedene Philosophien gegliedert, die nun kurz exemplarisch aufgefasst werden:

Philosophie	Beschreibung	Beispiel Anwendung
Safelist (Allowlist)	Nur Werte aus einer vordefinierten Liste erlauben. Alles andere wird abgelehnt.	Dropdown-Auswahl für Sortieroptionen (z. B. "date", "author"). Nur erlaubte Keys werden akzeptiert.
Blocklist (Denylist)	Bestimmte bekannte, unerwünschte Werte verbieten. Unsicher, da neue unerlaubte Werte nicht erfasst werden.	Sperren von bekannten schädlichen Dateiendungen (z. B. ".exe", ".bat").
Format Detection	Prüfen, ob Eingabe einem bestimmten Muster entspricht.	Postleitzahl-Validierung mit Regex, E-Mail-Prüfung mit <code>is_email()</code> , nur Ziffern mit <code>ctype_digit()</code> .
Format Correction (Sanitization)	Eingaben akzeptieren, aber in ein sicheres Format umwandeln.	Benutzername mit <code>sanitize_title()</code> , Typ-Cast von String zu Integer <code>(int)\$input</code> .

Tabelle 2.2: Übersicht Validierungs-Philosophien mit Beispielen (eigene Darstellung)

Wenn keine Validierung möglich ist, bedienen sich WordPress Entwickler der Möglichkeit des Sanitizing und Escaping von Werten.

Sanitizing

Die Bereinigung von Eingaben ist der Prozess der Sicherung/Bereinigung/Filterung von Eingabedaten. Die Validierung ist der Bereinigung vorzuziehen, da sie spezifischer ist. Wenn jedoch eine „spezifischere“ Lösung nicht möglich ist, ist die Bereinigung die nächstbeste Option.

(eigene Übersetzung nach [15])

Bevor also Daten, die von Nutzenden stammen, in der Datenbank abgelegt oder weiterverarbeitet werden, sollten sie bereinigt werden. Damit werden Sicherheitslücken als auch ungewollte Inhalte vermieden. Dafür stellt WordPress verschiedene

Funktionen bereit, die sicherstellen, dass Eingaben ausschließlich die vorgesehenen Zeichen oder Strukturen enthalten. Ein häufig genutztes Beispiel ist die Verwendung von `sanitize_text_field()` für Freitextfelder. Nachfolgend werden die seitens WordPress bereitgestellten Funktionen erläutert:

Funktion	Beschreibung
<code>sanitize_text_field()</code>	Entfernt Tags, ungültige UTF-8-Zeichen, Zeilenumbrüche, Tabs und überflüssige Leerzeichen.
<code>sanitize_email()</code>	Validiert und bereinigt eine E-Mail-Adresse.
<code>sanitize_file_name()</code>	Entfernt unerlaubte Zeichen aus Dateinamen.
<code>sanitize_hex_color()</code>	Prüft und bereinigt Hex-Farbwerde (#RRGGBB).
<code>sanitize_html_class()</code>	Bereinigt Klassennamen für Hypertext Markup Language (HTML)-Attribute.
<code>sanitize_key()</code>	Bereinigt Array-Keys oder Optionsnamen.
<code>sanitize_option()</code>	Bereinigt Optionswerte in der Datenbank.
<code>sanitize_title()</code>	Macht einen String zu einem URL-freundlichen Titel.
<code>sanitize_user()</code>	Bereinigt Benutzernamen.
<code>wp_kses()</code>	Filtert HTML nach erlaubten Tags/Attributen.
<code>wp_kses_post()</code>	Wie <code>wp_kses()</code> , aber mit Standardregeln für Posts.

Tabelle 2.3: Auswahl wichtiger WordPress-Sanitization-Funktionen
(eigene Darstellung)

Escaping

Escaping transformiert Ausgabedaten so, dass kritische Zeichen (z. B. “<“ wird zu “<“) nicht mehr als Code interpretiert werden. Dies verhindert Cross-Site-Scripting (XSS)-Angriffe.

Das Prinzip lautet: Daten unverändert speichern, erst bei der Ausgabe escapen. WordPress bietet kontextspezifische Funktionen wie `esc_html()`, `esc_attr()`, `esc_url()` und `esc_js()` [16]. Die richtige Funktionswahl je nach Ausgabekontext ist entscheidend für die Sicherheit. Genau wie beim Sanitizing bietet WordPress hier ein Set an

Funktionen, welche speziell für das Escaping genutzt werden:

Funktion	Anwendungsfall / Beschreibung
<code>esc_html()</code>	Escaped Text für den Einsatz innerhalb von HTML-Elementen. Entfernt sämtliches HTML.
<code>esc_attr()</code>	Für Werte innerhalb von HTML-Attributten, z. B. <code><input value="...></code> .
<code>esc_url()</code>	Für URLs in <code>src</code> oder <code>href</code> -Attributten.
<code>esc_url_raw()</code>	Roh-URL-Escaping für Speicherung in der Datenbank oder nicht-encodierte Weitergabe.
<code>esc_js()</code>	Für Inline-JavaScript oder Übergabe von Variablen in Skripte.
<code>esc_xml()</code>	Absicherung von Ausgaben in Extensible Markup Language (XML) -/Extensible Stylesheet Language (XSL)-Kontexten.
<code>esc_textarea()</code>	Encodiert Texte für die Verwendung innerhalb eines <code><textarea></code> Elements.
<code>wp_kses()</code>	Filtert HTML-Inhalte und erlaubt nur eine definierte Menge an Tags/Attributten.
<code>wp_kses_post()</code>	Variante von <code>wp_kses()</code> , die die Standardmenge an Post-HTML erlaubt.
<code>wp_kses_data()</code>	Variante von <code>wp_kses()</code> , die nur HTML erlaubt, das in Kommentaren zugelassen ist.

Tabelle 2.4: Überblick zentraler Escaping-Funktionen in WordPress
(eigene Darstellung)

Ein wichtiger Grundsatz ist das späte Escaping: Daten sollten erst unmittelbar vor der Ausgabe escaped werden. Dadurch wird die Code-Überprüfung vereinfacht, das Fehlerrisiko verringert und die Anwendung widerstandsfähiger gegen zukünftige Anpassungen. Wenn ein spätes Escaping nicht umsetzbar ist (etwa bei generiertem JavaScript-Code), muss der Entwickler sicherstellen, dass Variablen bereits in einer als „sicher“ gekennzeichneten Form vorliegen (z. B. `$variable_escaped`) [17].

Nonces

Ein Nonce (number used once) ist ein Sicherheitsmechanismus in WordPress zum Schutz von URLs und Formularen vor unbefugten Manipulationen. Insbesondere wird der Schutz vor Cross-Site Request Forgery (CSRF) sichergestellt. Anders als die Definition von Nonces handelt es sich nicht um Einmalzahlen, sondern um Hash-Werte. Diese Hashwerte sind für einen bestimmten Benutzer und Kontext über eine begrenzte Zeit gültig (Standard: 12–24 Stunden) [18].

Erstellung: `wp_create_nonce('action')`, `wp_nonce_url()`, `wp_nonce_field()`

Validierung: `wp_verify_nonce()`, `check_admin_referer()`, `check_ajax_referer()`

Sie sollten stets actionsspezifisch sein und niemals als alleinige Sicherheitsmaßnahme dienen.

2.2.4 Weiterführende Funktionen und Paradigmen

In diesem Kapitel sind weitere für das Projekt verwendete Funktionen und Paradigmen aus WordPress aufgeführt.

Themes

Ein Theme in WordPress bestimmt das Erscheinungsbild einer Website. Es legt Farben, Schriftarten, Layout-Strukturen sowie die gesamte Darstellung des Frontends fest und kann auch Funktionen im Backend hinzufügen [19]. Für das Projekt *Charigame* spielt das konkret gewählte Theme jedoch keine zentrale Rolle. Die entwickelte Lösung soll unabhängig von einem bestimmten Theme funktionieren.

Metadaten

Unter Metadaten versteht man ergänzende Informationen zu bestehenden Inhalten, sogenannte “Informationen über Informationen“. Im WordPress-Kontext handelt es sich dabei um zusätzliche Datensätze, die verschiedenen Objekten wie Artikeln, Nutzerprofilen, Kommentaren oder Kategorien hinzugefügt werden können. Praktische Anwendungsbeispiele sind etwa Custom Fields in Blogbeiträgen, erweiterte Benutzerinformationen in Profilen oder spezielle Attribute für Kategorienseiten. WordPress zeichnet sich durch ein besonders anpassungsfähiges Metadaten-System aus. Jedes Element kann mit einer unbegrenzten Anzahl zusätzlicher Informationen ausgestattet werden [20].

API Endpoints

Die WordPress-REST-API stellt unter dem Endpunkt /wp-json/ eine Schnittstelle zur Verfügung, über die Inhalte und Strukturen einer WordPress-Installation abgerufen und verändert werden können. Der zentrale Namespace wp/v2/ umfasst dabei alle wesentlichen Ressourcen wie Beiträge, Seiten, Benutzer, Medien oder Taxonomien. Technisch fundiert die Kommunikation im Gutenberg-Editor selbst auf der REST-API. Änderungen im Block Editor werden im Hintergrund als Anfragen an die API umgesetzt. Entwickler können über die JavaScript-Schnittstellen wie @wordpress/api-fetch eigene Blöcke implementieren oder externe Datenquellen einbinden [21].

AJAX

WordPress bietet eine eigene Asynchronous JavaScript and XML (AJAX)-Schnittstelle, die auf den Endpunkt admin-ajax.php verweist. Entwickler können hier eigene Aktionen registrieren, die bei einer AJAX-Anfrage ausgeführt werden. Jede Aktion wird über einen eindeutigen Namen (action) identifiziert, wodurch mehrere unabhängige AJAX-Funktionen parallel existieren können. Die Kommunikation erfolgt typischerweise über POST- oder GET-Anfragen, die Daten in JavaScript Object Notation (JSON)- oder URL-kodierter Form zurückliefern [22].

WP-CRON

In WordPress werden zeitgesteuerte Aufgaben über ein eigenes System gesteuert, das sogenannte WP-CRON. WP-CRON bietet eine einfache Möglichkeit, zeitbasierte Prozesse zu implementieren. Solche Events werden typischerweise beim Planen von Beiträgen oder dem Versenden von E-Mail-Benachrichtigungen genutzt [23].

2.3 Gutenberg-Editor: Konzept und technische Grundlagen

Der Block Editor (Gutenberg) ist ein modernes Paradigma für die Erstellung und Veröffentlichung von WordPress-Websites. Er verwendet ein modulares System aus Blöcken zum Erstellen und Formatieren von Inhalten und wurde entwickelt, um reichhaltige und flexible Layouts für Websites und digitale Produkte zu erstellen.

(eigene Übersetzung nach [24])

Ein Block ist ein eigenständiges Element wie beispielsweise ein Absatz, eine Überschrift, ein Medienelement oder eine Einbettung. Jeder Block wird als separates Element mit individuellen Bearbeitungs- und Formatierungsoptionen behandelt. Wenn alle diese Komponenten zusammengefügt werden, bilden sie den Inhalt der Seite oder des Beitrags, der dann in der WordPress-Datenbank gespeichert wird [24].

Die Relevanz des Editors wird durch die erhobene Statistik der Seite Gutenberg in Numbers deutlich. Demnach beträgt die Anzahl aktiver Installationen des Gutenberg-Editors auf WordPress- und Jetpack-Seiten aktuell 87,3 Mio. Innerhalb der letzten drei Jahre wurden 282 Mio. Beiträge mit dem Gutenberg-Editor erstellt. Die Daten basieren auf einem Dreijahreszeitraum und umfassen nur WordPress.com- und Jetpack-Seiten, die den Einsatz des Editors melden [25].

2.3.1 Dateistruktur eines Blocks

Die Developer Resources geben Entwicklern Grundlagen der Blockentwicklung mit, welche nachfolgend erläutert werden.

Seitens WordPress wird eine Dateistruktur für die Gutenberg Blöcke empfohlen. Diese Struktur erleichtert die Trennung von Metadaten, Logik und Darstellung. Abbildung 1 im Anhang der Arbeit zeigt den grundsätzlichen Aufbau eines Blocks, wie er in der offiziellen Dokumentation beschrieben ist.

Die Abbildung verdeutlicht, dass zentrale Dateien wie `block.json` zur Definition der Block-Metadaten sowie Skripte, Stylesheets und Build-Artefakte in klar getrennten Verzeichnissen organisiert werden.

Wenn ein Block entwickelt wird, ist es angeraten, diesen als Plugin und nicht innerhalb des Themes zu registrieren. Dies hat den Vorteil, dass der erstellte Block unabhängig vom verwendeten Theme genutzt werden kann.

2.3.2 Schlüsselkonzepte im Block Editor

Kombinierbarkeit

Blöcke sind darauf ausgelegt, auf vielfältige Weise miteinander kombiniert zu werden. Sie folgen einem hierarchischen Aufbau, bei dem ein Block in einen anderen eingebettet werden kann. Verschachtelte Blöcke und ihr umschließender Block werden auch als Kinder und Eltern bezeichnet. Ein Spalten-Block kann beispielsweise als übergeordneter Block fungieren und mehrere untergeordnete Blöcke in seinen einzelnen Spalten enthalten. Die Programmierschnittstelle, welche die Verwendung von Unter-Blöcken regelt, wird InnerBlocks genannt [26].

Daten und Attribute

Blöcke verstehen Inhalte als Attribute. Ein Block kann eine beliebige Anzahl von Attributen beinhalten und diese werden in einer Key Value Relation definiert. Der Key spiegelt den Namen wider und der Value den Wert. Diese Inhalte sind serialisierbar in HTML und werden im Block HTML (vgl. Abbildung 2.4) gespeichert und so auch in der Datenbank als post_content abgelegt.

```

1      <!-- wp:paragraph {"key": "value"} -->
2      <p>Welcome to the world of blocks.</p>
3      <!-- /wp:paragraph -->
```

Abbildung 2.4: Dateistruktur eines WordPress-Blocks (eigene Darstellung)

Blöcke lassen sich in zwei Kategorien unterteilen: statisch und dynamisch. Statische Blöcke bestehen aus bereits gerenderten Inhalten sowie einem Attribut-Objekt, das bei Änderungen für das erneute Rendern verwendet wird. Dynamische Blöcke hingegen benötigen serverseitige Daten und werden erst während der Inhaltsgenerierung gerendert [27].

Blockmuster

Ein Blockermuster oder auch Pattern ist eine Gruppe von Blöcken, die zu einem Designmuster zusammengefasst werden. Solche Designmuster bieten für die schnellere Erstellung komplexer Seiten einen Ausgangspunkt. Die Größe des Blockmusters ist variabel und kann von einem einzelnen Block bis hin zu einer gesamten Seite reichen. Themes können Designmuster registrieren, um Benutzern schnelle Startpunkte für das Verwenden von Blöcken zu bieten [28].

Block Themes

Anders als der Beitragseditor konzentrieren sich die Block Themes auf die Deklaration und das Bearbeiten der gesamten Website. Hier können unter Zuhilfenahme von Blöcken die Kopf- bis hin zur Fußzeile Inhalte angepasst werden. Die Block Themes selbst werden unterteilt in Vorlagen, die die gesamte Seite beschreiben oder Vorlagenteile, die wiederverwendbare Bereiche sein können. Angepasste Blockvorlagen umfassen statische als auch dynamische Seiten wie Archive, Singular, Home, 404 usw. [30]

2.4 Gamification im Kontext digitaler Anwendungen

Deterding et al. definieren Gamification als “the use of game design elements in non-game contexts“ [31]. Diese Definition erfasst ein Phänomen, das in seiner Grundform nicht neu ist. Leistungsbezogene Vergütungssysteme, Rankings und spielerische Wettkämpfe existieren in Fabriken, dem Vertrieb und Bildungseinrichtungen bereits seit Jahrzehnten [32]. Die Digitalisierung hat seit etwa 2010 eine weitreichende Transformation des Gamification-Konzepts eingeleitet. Durch die Verbreitung von Smartphones und insbesondere Wearables wie Smartwatches und Fitnessbändern haben sich neue Dimensionen der spielerischen Motivationsgestaltung eröffnet [33]. Diese Entwicklung stellt einen Wechsel in der Gestaltung des Mensch-Computer-Interaktionsparadigmas dar.

Gamification hat sich von simplen Belohnungssystemen zu einem komplexen Gesamtkonzept entwickelt, das die Wahl spezifischer Elemente und Mechaniken, die Berücksichtigung des Anwendungskontextes, der Zielgruppe sowie der zu erreichenden Ziele umfasst [32].

Abgrenzung von Serious Games

Im Projektkontext von *Charigame* ist eine klare Abgrenzung zwischen Gamification und Serious Games erforderlich. Während Serious Games als eigenständige Spiele konzipiert werden, die primär pädagogische Lernziele verfolgen, versteht sich die Gamification-Implementierung in *Charigame* als funktionale Ergänzung. Ziel des gamifizierten Parts ist es, die Nutzerbeteiligung zu steigern und nicht als Wissensvermittlung zu dienen.

Charigame verbindet somit die konventionelle (Charity)-Spendenaktion durch die Integration spielmechanischer Elemente, ohne dabei den ursprünglichen Zweck der Aktion zu verändern. Der User Experience-orientierte Ansatz zielt darauf ab, die Customer Journey zu optimieren und die emotionale Bindung der Nutzer zur Spendenaktion zu erhöhen. Dieser spielerische Ansatz dient als Mittel zur Aktivierung und nicht als Selbstzweck.

Strategische Bedeutung von Gamification für Spendenaktionen

Die Integration von Gamification in Spendenaktionen adressiert mehrere strukturelle Herausforderungen traditioneller Fundraising-Ansätze. Spendenaufzüge leiden häufig unter geringem Nutzerengagement und mangelnder Transparenz bezüglich der Verwendung der Mittel. Gamification-Elemente schaffen hier Abhilfe durch die Visualisierung des sozialen Impacts und erhöhte Nutzerbeteiligung. Die Integration spielerischer Elemente steigert sowohl die Aufmerksamkeit als auch die Reichweite sozialer Kampagnen. Parallel dazu ermöglicht dieser Ansatz eine authentische Kommunikation des Markenimages. Das Unternehmen kann dadurch seine gesellschaftliche Verantwortung auf eine partizipative Weise demonstrieren [35].

2.5 Überblick über Corporate Social Responsibility und Charity-Plattformen

Corporate Social Responsibility (CSR) hat sich von einem optionalen Unternehmensengagement zu einer Strategie moderner Unternehmensmaßnahmen entwickelt. Die Europäische Kommission definiert CSR als die Verantwortung von Unternehmen für ihre Auswirkungen auf die Gesellschaft. Hier wird ein Konzept beschrieben, das weit über die Einhaltung gesetzlicher Bestimmungen hinausgeht [34]. Diese gesellschaftliche Erwartungshaltung spiegelt sich in der zunehmenden Nachfrage nach transparenten und messbaren CSR-Maßnahmen wider, die von Verbrauchern eingefordert werden. Wie aus der Studie der Bertelsmann-Stiftung zur gesellschaftlichen Verantwortung von Unternehmen hervorgeht, identifizieren 60% der befragten Unternehmen die Erwartungen der Kunden als einen der wichtigsten äußeren Faktoren für ihr gesellschaftliches Engagement [36].

Die Verbindung von CSR-Maßnahmen mit digitalen, interaktiven Ansätzen stellt eine potenzielle Antwort auf diese gestiegenen Erwartungen dar. Gamifizierte Charity-Plattform ermöglichen es dabei, Transparenzanforderungen zu erfüllen als auch das häufig geringe Engagement bei konventionellen Spendenaktionen zu erhöhen.

Charity-Plattformen

Parallel zu dieser Entwicklung hat die Digitalisierung neue Möglichkeiten für die Umsetzung und Kommunikation von CSR-Initiativen eröffnet. Online-Charity-Plattformen ermöglichen es Unternehmen, ihre gesellschaftliche Verantwortung digital abzubilden und dabei gleichzeitig die Reichweite zu erhöhen. Diese digitalen Plattformen fungieren als Vermittler zwischen Unternehmen, gemeinnützigen Organisationen und der Öffentlichkeit [37].

Gamifizierte Charity-Plattformen

Die Kombination von CSR-Maßnahmen mit spielerischen Inhalten eröffnet Unternehmen neue Möglichkeiten, gesellschaftliches Engagement sichtbar und interaktiv zu gestalten. Gamification wirkt hier als Verbindungsstück, das das in CSR-Anwendungen häufig geringe Nutzerengagement adressiert und erhöht. Charity-Plattformen, die auf Gamification-Elemente setzen, verbinden damit zwei wesentliche Aspekte. Die wachsende Erwartungshaltung an transparente CSR-Initiativen und den Trend zu interaktiven, nutzerzentrierten digitalen Anwendungen. Damit ist der theoretische Rahmen gesetzt, innerhalb dessen das Projekt *Charigame* verortet werden kann. Im folgenden Kapitel wird der Projektkontext dargestellt, um die praktische Umsetzung und Weiterentwicklung der Plugin-Architektur nachvollziehbar zu machen.

3 Projektkontext

In diesem Kapitel wird der Kontext des Projekts vorgestellt, um ein Verständnis für dessen Ursprung, Zielsetzung und aktuellen Entwicklungsstand zu schaffen. Neben der Entstehungsgeschichte von *Charigame* werden die derzeitige Systemarchitektur sowie die Funktionsweise des bestehenden WordPress-Plugins beschrieben. Diese Darstellung bildet die Grundlage für die anschließende Analyse und dient als Ausgangspunkt für die konzeptionelle Weiterentwicklung in Kapitel 4.

3.1 Das Projekt Charigame

Das Projekt *Charigame* entstand ursprünglich als interne Initiative der elancer-team GmbH mit dem Ziel eine digitale Spendenaktion zu ermöglichen. Diese Idee des Projekts lässt sich auf eine durch die Agentur erstellte Weihnachtsaktion zurückführen, die auf einer dedizierten Website¹ implementiert wurde. Bei der Spendenaktion hat die elancer-team GmbH die Spendensumme bereitgestellt und die Agenturkunden zur Teilnahme eingeladen, sodass die Kunden durch ihr Engagement den Spendentopf erhöhen konnten. Diese Plattform ermöglichte es den Kunden der Agentur erstmals, über die gamifizierte Benutzeroberfläche Spenden zu erspielen und zu verteilen.

Die positive Resonanz der Agenturkunden auf diesen gamifizierten Ansatz führte zu der Überlegung, das Konzept weiterzuentwickeln. Aufgrund der erfolgreichen Durchführung äußerte ein Kunde der Agentur den Wunsch, eine vergleichbare Spendenaktion für das eigene Unternehmen zu realisieren.

Diese erste kundenspezifische Umsetzung basierte auf einer abgewandelten Kopie der Weihnachts-Spendenaktion. Die Anfrage stellte dann den Übergang von einem internen Projekt zu einem eigenständigen Dienstleistungsangebot dar. Daraufhin wurde durch den Autor das Projekt *Charigame* als eigenständiges WordPress-Plugin realisiert. Das daraus resultierende WordPress-Plugin befindet sich aktuell bei einem Kunden im produktiven Einsatz und bildet die Grundlage für die in dieser Arbeit dokumentierten technischen Analyse und Weiterentwicklung.

¹<https://hohoho.elancer-team.de/>

Funktionsweise von Charigame

Die Funktionsweise von *Charigame* lässt sich grob in vier Schritte unterteilen, die in Abbildung 3.1 veranschaulicht sind:

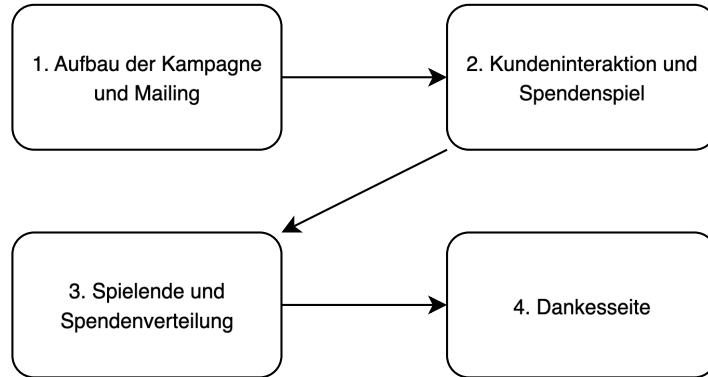


Abbildung 3.1: Funktionsweise *Charigame* (eigene Darstellung)

1. Aufbau der Kampagne und Mailing

Eine Spendenkampagne wird im Backend von *Charigame* angelegt. Anschließend werden Personendaten der Kunden in das System importiert. Das System versendet dann basierend auf den Kampagneneinstellungen, automatisierte E-Mails mit dem Link zur Spendenaktion.

2. Kundeninteraktion und Spendenspiel

Der Kunde öffnet den personalisierten Code in der E-Mail oder gibt den darin enthaltenen Code auf der Login-Page ein. Die in der Kampagne eingestellte *Charigame*-Landingpage wird angezeigt und der Kunde kann aktiv beim Spendenspiel teilnehmen.

3. Spielende und Spendenverteilung

Nachdem das Spendenspiel seitens des Kunden absolviert wurde, kann dieser den erspielten Beitrag prozentual auf einen bis hin zu drei verschiedenen Spendenempfängern verteilen.

4. Dankesseite

Eine Dankesseite wird angezeigt und der Kunde kann nach Belieben erneut an dem Spiel teilnehmen, um seine Punktzahl zu verbessern. Ferner werden weitere Handlungsauforderungen in Form von CTAs ausgespielt, die den Kunden gezielt auf Bereiche der Unternehmenswebsite leiten können.

3.2 Deskriptiver Stand

3.2.1 WordPress Frontend und Backend

Der deskriptive Stand von *Charigame* befindet sich in einem funktionsfähigen, jedoch technisch und konzeptionell ausbaufähigen Zustand. Das WordPress-Plugin integriert sich in das CMS WordPress und erweitert den Funktionsumfang um gamifizierte Spendenaktionen. Bevor die Spendenaktion bereitsteht, ist es notwendig, dass verschiedene Einstellungen getroffen werden. Die erforderlichen Einstellungen können über das WordPress-Backend Menü aufgerufen werden.

Hierzu erzeugt ein *Charigame* Menüpunkt im WordPress Dashboard wie in Abbildung 3.2 visualisiert. In diesem Menü sind die wichtigsten Einstellmöglichkeiten als Menüpunkte definiert.

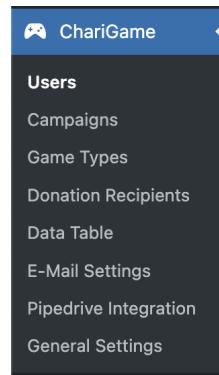


Abbildung 3.2: Charigame Menü im WordPress Dashboard (eigene Darstellung)

Die im Menü angezeigten Einstellungen und Informationen lassen sich in mehrere Kategorien gliedern. Diese spiegeln den Aufbau der einzelnen Bausteine vom Projekt wider und werden nachfolgend erläutert.

General Settings

Die General Settings enthalten grundlegende Angaben zum Unternehmen, das eine Spendenkampagne durchführt. Neben rechtlichen Informationen wie dem Impressum und den AGBs können hier auch gestalterische Parameter wie Farben der Corporate Identity hinterlegt werden. Die Backendansicht der Einstellungsseite ist in Abbildung 3.3 dargestellt.

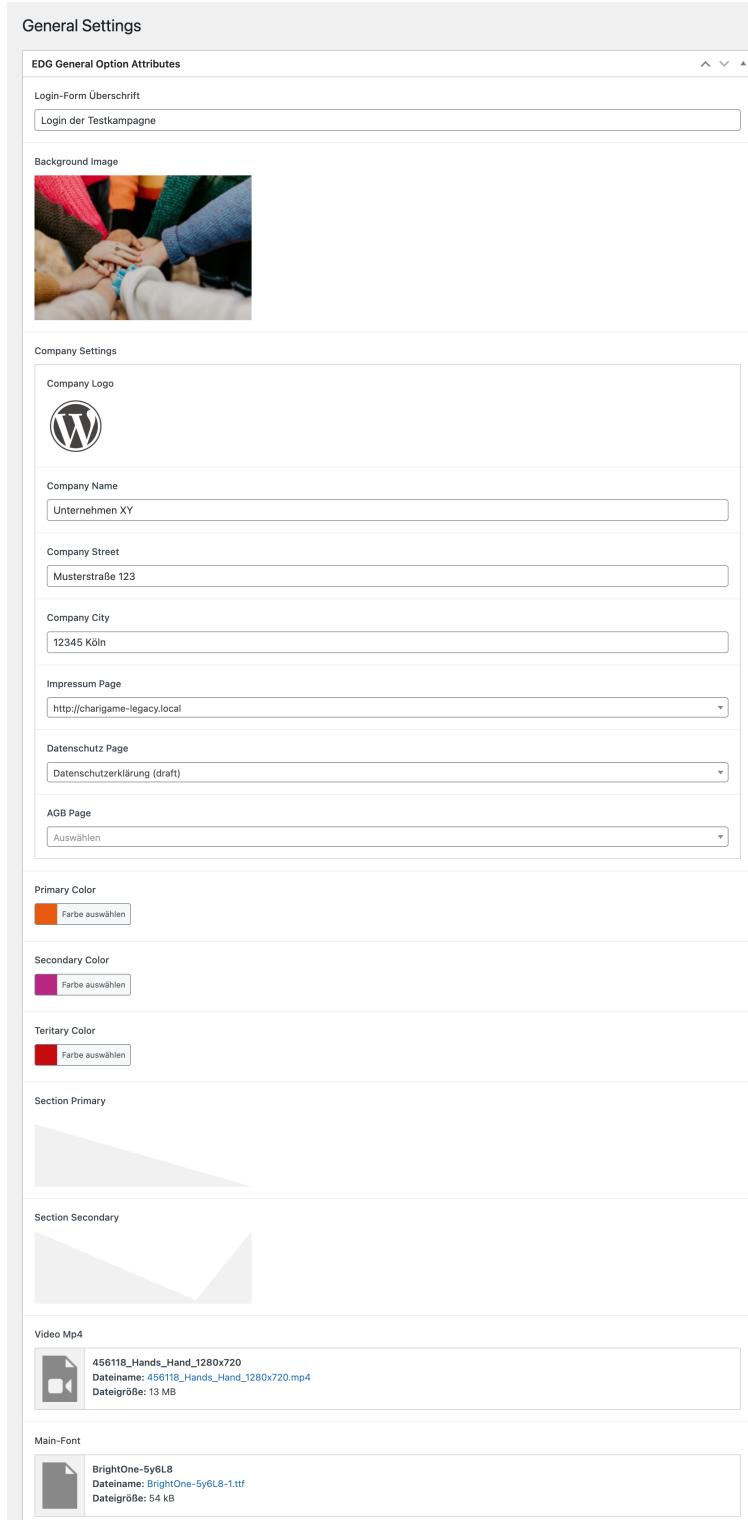


Abbildung 3.3: Screenshot aus dem WordPress-Backend (eigene Darstellung). 25

Enthaltene Medien:

Hintergrundbild © Aditya Romansa / Unsplash [39],
Logo © WordPress Foundation [40].

Es können grundlegende Einstellungen konfiguriert werden, z. B.

- Überschriften der Login-Form
- Hintergrund- und Logo-Uloads
- Farbschema (Primär, Sekundär, Tertiär)

Eine vollständige Übersicht der Eingabefelder ist im Anhang zu finden (vgl. Tabelle 1).

Die getroffenen Einstellungen bestimmen direkt das Erscheinungsbild der Login-Form im Frontend (vgl. Abbildung 3.4).

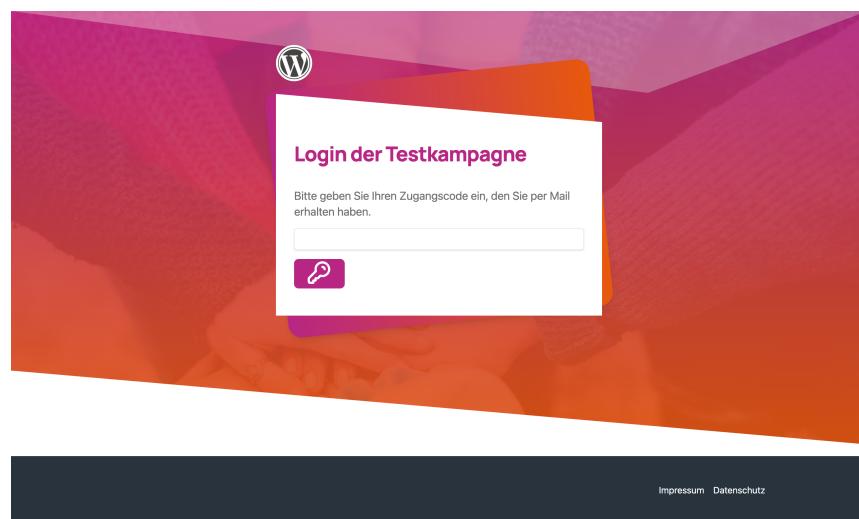


Abbildung 3.4: Login-Form einer *Charigame*-Testkampagne im Frontend
(eigene Darstellung)

Donation Recipients

Die Donation Recipients stellen die Empfänger der Spenden dar. Für jede Kampagne müssen mindestens drei Empfänger angelegt werden. Pro Empfänger werden ein Name, ein Bild sowie ein beschreibender Text hinterlegt (vgl. Tabelle 2). Die Vorabberstellung der Recipients ist im Verlauf der Einstellungen zwingend erforderlich, da sie in den Kampagneneinstellungen referenziert werden. Die Backendlansicht der Donation Recipients ist in Abbildung 3.5 veranschaulicht.



Abbildung 3.5: Donation Recipients im WordPress Backend
Enthaltenes Logos © WWF [41] (eigene Darstellung)

Die im Backend gesetzten Einstellungen werden in folgender Form wie im Frontend dargestellt (vgl. Abbildung 3.6):



Abbildung 3.6: Donation Recipients im WordPress Frontend (eigene Darstellung)
Enthaltene Logos © UNICEF, Deutsches Rotes Kreuz, WWF [41]

Game Types

Unter Game Types werden die verfügbaren Spieltypen definiert. Jedes Spiel kann vom Unternehmen mit einer „How-To-Play“-Anleitung versehen werden. Diese besteht aus einer Abfolge von Schritten, die jeweils mit einem Icon, einer Überschrift und einem erklärenden Text ergänzt werden können. Im Detail können die verfügbaren Attributionen aus der Tabelle 3 entnommen werden. Die Backendansicht der Game Types ist in Abbildung 3.7 ersichtlich.

The screenshot shows the 'EDG Gametype Attributes' section in the WordPress admin. It includes fields for 'How To Play Headline' (containing 'So wird gespielt:'), 'How To Play Steps' (a table with columns for Step Icon, Step Headline, Step Text, and Step Color), and a 'Step Icon' dropdown menu with options like Dashicons, Mediathek, and URL, currently set to 'Mediathek'.

Step Icon	Step Headline	Step Text	Step Color
	Start des Spiels	Das Spiel generiert ein Raster mit verdeckten Karten. Jede Karte besitzt ein identisches Gegenstück.	Primary

Abbildung 3.7: Game Types im WordPress Backend (eigene Darstellung)

Wenn entsprechende Informationen eingepflegt sind, erzeugt der Bereich der Game Types den folgenden Frontend View wie in Abbildung 3.8 zu sehen:



Abbildung 3.8: Game Types im WordPress Frontend (eigene Darstellung)

Users & Pipedrive Integration

Die Menüpunkte Users und Pipedrive Integration dienen maßgeblich der Verwaltung im Backend und haben keine direkten Auswirkungen auf das Frontend. Im Bereich der User werden die Teilnehmer der Spendenkampagnen verwaltet. Diesen können die folgenden Werte zugeordnet werden:

- Vorname
- Nachname
- Geburtsdatum
- E-Mail-Adresse
- Flags: Imported, Email sent

Die genauen Werte und Eingabetypen können aus der Tabelle 4 entnommen werden. Im WordPress Backend wird die User Sektion wie folgt dargestellt (vgl. Abbildung 3.9):

The screenshot shows a form titled "EDG User Attributes" for creating a new user. The fields filled in are:

- First Name: Max
- Last Name: Mustermann
- E-Mail: maxmustermann@charigame.de
- Next Birthday: 20/02 (Note: As the year of the individual user is not used, the date is set to the current year for data protection reasons.)
- Imported: An unchecked checkbox.
- Email sent: An unchecked checkbox.

Abbildung 3.9: Users im WordPress Backend (eigene Darstellung)

Der Bereich der Pipedrive Integration bietet die Möglichkeit, Nutzer direkt über die API des Customer-Relationship-Management (CRM) Systems Pipedrive zu beziehen. Diese Funktion wurde spezifisch für ein Kundenprojekt entwickelt und bietet deshalb eine angepasste und keine universelle Funktionalität. Die Darstellung im Backend der Pipedrive Integration zeigt Abbildung 3.10:

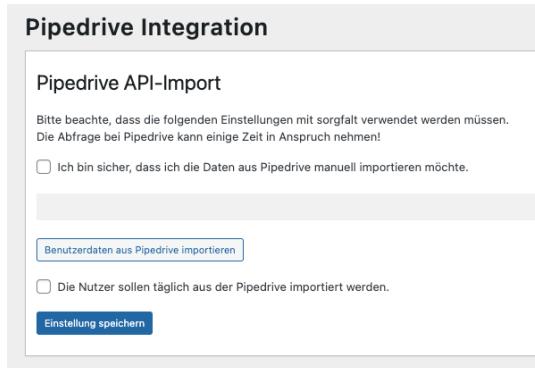


Abbildung 3.10: Pipedrive im WordPress Backend (eigene Darstellung)

E-Mail-Settings

Das Plugin erlaubt es, anstelle des standardmäßigen WordPress-Mailers einen eigenen Simple Mail Transfer Protocol (SMTP)-Server einzubinden. Dafür stehen folgende Eingabefelder bereit:

- SMTP-Host
- SMTP-Port
- Benutzername
- Passwort
- Verschlüsselung (Transport Layer Security (TLS)
oder Secure Sockets Layer (SSL))
- Absender-E-Mail
- Absender-Name

Aus Sicherheitsgründen kann das Passwort optional in der wp-config.php als Konstante „CHARIGAME_WPMS_SMTP_PASS“ hinterlegt werden, sodass es nicht in der Datenbank gespeichert wird. Zusätzlich bietet die Oberfläche eine Funktion zum Versand einer Testmail, mit der sich die Konfiguration überprüfen lässt. Die Eingabemaske, befüllt mit Beispielwerten, ist in Abbildung 3.11 ersichtlich.

SMTP-Einstellungen

SMTP Host	smtp.meinsmtp-server.de
SMTP Port	587
Benutzername	smtp4492121de
Passwort
Verschlüsselung	TLS (Port 587) ▾
Absender E-Mail	charigame@meinsmtp-server.de
Absender Name	Charigame Gewinnspiel

SMTP Einstellungen speichern

Test-E-Mail versenden

Absender E-Mail	
Empfänger E-Mail	

Testmail senden

Abbildung 3.11: E-Mail Settings im WordPress Backend (eigene Darstellung)

Darüber hinaus bietet der Backend-View der E-Mail-Settings die Funktion, eine Testmail an eine ausgewählte Adresse zu senden, um die Einstellungen auf ihre Richtigkeit prüfen zu können.

Data Table

Die Data Table dient als zentrales Dashboard, um die Ergebnisse und Kennzahlen der Spendenkampagnen auf einen Blick darzustellen. Angezeigt werden u. a.

- erstellte Kampagnen
- zugehörige Spendenempfänger
- Benutzername
- aktueller Spendenstand (in Euro)
- Nutzerübersicht mit einzigartigen Game Codes

Darüber hinaus werden weitere Metriken bereitgestellt, darunter Valid From, Valid Until, Last Played, Highscore, Recipient 1–3 sowie E-Mail sent, wie in Abbildung 3.12 zu sehen ist. Die genaue Bedeutung dieser Kennzahlen ist in Tabelle 5 im Anhang erläutert.

The screenshot shows the 'EDG Data Table Page' under the 'EDG Campaigns' section. A single entry for a campaign named 'Testkampagne' is displayed. The table has columns for Email, Game Type, Game Code, Valid From, Valid Until, Code Used, Last Played, Highscore, Recipient 1, Recipient 2, Recipient 3, and E-Mail Sent. The data shows one entry for 'maxmustermann@fakemail.com' with a game type of 'memory' and a game code of 'NUJDFTYLSPX7V'. The valid period is from 2025-08-03 to 2025-09-07. The last play was at 2025-08-21 22:36:15. The highscore is 0, and all recipient fields are set to 0. An email was sent. At the bottom, there are buttons for 'Benutzerdaten aktualisieren' and 'Benutzerdaten löschen', and a link 'Alle Nutzer aus der Tabelle EDG User löschen'.

Email	Game Type	Game Code	Valid From	Valid Until	Code Used	Last Played	Highscore	Recipient 1	Recipient 2	Recipient 3	E-Mail Sent
maxmustermann@fakemail.com	memory	NUJDFTYLSPX7V	2025-08-03	2025-09-07	2025-08-21 22:36:15		0	0	0	0	0

Abbildung 3.12: Data Table im WordPress Backend (eigene Darstellung)

Campaigns

Die Campaigns stellen das zentrale Element von *Charigame* dar und repräsentieren den konzeptionell und technisch komplexesten Bereich des Plugins. Dieser Bereich ermöglicht die Steuerung über eine Vielzahl von Konfigurationsmöglichkeiten. Der gesamte Ablauf einer Kampagne kann innerhalb des WordPress-Backends abgebildet werden. Dies umfasst sowohl die Auswahl von Spieleinstellungen als auch die Kommunikation mit den Teilnehmenden.

Zu den wesentlichen Komponenten der Campaigns gehören:

- **Allgemeine Kampagneninformationen**

Jede Kampagne erhält einen Titel, eine kurze Bezeichnung sowie eine ausführliche Beschreibung. Diese Inhalte bilden die Grundlage für die Frontend-Darstellung und dienen der inhaltlichen Darstellung im oberen Bereich der Landingpage. Zusätzliche Medien wie Logos oder Illustrationen können hochgeladen und der Kampagne zugeordnet werden.

- **Spielmechanik**

Der gewünschte Spieltyp wird aus den im System verfügbaren Game Types ausgewählt.

- **Spendenempfänger**

Jede Kampagne referenziert mindestens drei zuvor definierte Donation Recipients. Diese Empfänger werden im Spiel zur Auswahl gestellt, um den erspielten Beitrag unter ihnen zu verteilen.

- **Punkte- und Spendenlogik**

Durch eine Highscore-Logik wird festgelegt, wie viele Punkte maximal erreichbar sind und welche Spendenwerte daraus resultieren. Gewinnkategorien ermöglichen eine Staffelung nach erreichten Punktzahlen (beispielsweise ab 20 Punkten 4 Euro, ab 50 Punkten 10 Euro). Eine übergeordnete Spendensumme, welche vom Unternehmen unabhängig der erzielten Spenden durch die Teilnehmer erfolgt, kann ebenfalls definiert werden.

- **Zeitliche Steuerung**

In der Kampagne kann zwischen zwei Methoden der Versendung gewählt werden. Entweder wird der Zeitpunkt zum Geburtstag der jeweiligen Teilnehmer oder an einem ausgewählten Datum gesetzt. Darüber hinaus lässt sich die Uhrzeit wählen, an der die E-Mail versendet werden soll. Ferner lässt sich die Gültigkeit der Spielcodes definieren, die angibt, wie lange sich der Teilnehmer mit seinem Code einloggen kann. Aus den getroffenen Einstellungen resultiert dann das Enddatum der Kampagne. Start- und Enddatum bestimmen den Gültigkeitszeitraum.

- **Kommunikation**

Für jede Kampagne können individuelle E-Mail-Texte konfiguriert werden. Diese umfassen Betreffzeile, Header, Haupttext sowie eine Signatur. Die eingestellten Inhalte werden dann in der E-Mail genutzt, die die Nutzer über ihre Teilnahme informieren. Optional können Social-Media-Kanäle angegeben werden, welche anschließend in der Mail angezeigt werden.

- **Call-to-Action**

Abschließend lassen sich Call to Action Bereiche aufbauen. Hierzu werden

Headline, Beschreibungstext, Button-Beschriftung und Ziel-URL definiert. Der Call to Action (CTA) motiviert Teilnehmende nach Abschluss des Spiels zu weiteren Aktionen, beispielsweise dem Besuch einer Unternehmensseite.

Im Anhang der Arbeit ist die gesamte Landingpage (vgl. Abbildung 2), die Seite der Spendenverteilung (vgl. Abbildung 3) und die Dankesseite (vgl. Abbildung 4) dargestellt. Die Abbildungen dienen der Visualisierung der oben genannten Komponenten und dem Verständnis des deskriptiven Zustands.

3.2.2 Verwendete Technologien

Für die Umsetzung der *Charigame*-Einstellungen im Frontend und Backend kam eine Vielzahl an Technologien zum Einsatz. Im Folgenden werden diese näher vorgestellt und in ihrem technischen Kontext erläutert.

- **WordPress Plugin-Stack**

Die Kernfunktionalität basiert auf den von WordPress bereitgestellten APIs. Eingesetzt wurden u. a. WP List Table für die Darstellung von Tabellen im Admin-Dashboard, die Enqueue-Mechanismen von WordPress zum Einbinden von Skripten und Styles sowie AJAX Hooks für asynchrone Requests.

Zusätzlich werden Register Activation Hooks und Template Include genutzt, um das Plugin bei der Aktivierung einzubinden und die Views zu steuern.

jQuery wird aus einem Content Delivery Network (CDN) eingebunden und dient an einigen Stellen als Hilfsbibliothek für Interaktionen.

Für den E-Mail-Versand kommt PHPMailer über die WordPress-Funktion wp_mail zum Einsatz.

Mit Advanced Custom Fields (ACF/ACF Pro) werden zusätzliche Metadaten für Inhalte verwaltet, die über die Funktionen get_field() intensiv genutzt werden. Außerdem wurden WP-CRON-Jobs bzw. Scheduled Tasks vorbereitet, um zeitgesteuerte Abläufe im Plugin zu ermöglichen.

- **Frontend und JavaScript**

Für Animationen wird die GreenSock Animation Platform (GSAP) 3 eingesetzt. Neben der Kernbibliothek sind auch Plugins von GSAP wie Observer, ScrollTrigger, MotionPathPlugin und Draggable integriert, die komplexe Animationen und Interaktionen ermöglichen.

Visuell wird ebenfalls Confetti JavaScript (JS) für animierte Partikeleffekte eingesetzt. Für das Tower-Spiel wird Three.js genutzt, eine leistungsfähige 3D-Bibliothek, die sowohl lokal eingebunden als auch als Node Package Manager (NPM)-Abhängigkeit installiert ist.

Ferner existiert ein eigenes kleines Spiele-Framework im Ordner “games“, das das Memory- und Tower-Spiel enthält und durch eigene Hilfsskripte wie helper.js und picker.js ergänzt wird.

- **CSS und Build-Prozess**

Das Styling basiert auf Tailwind CSS CSS, das über einen Command Line Interface (CLI)-Buildprozess in Kombination mit NPM-Skripten (build:css, watch:css) verarbeitet wird. Zusätzlich werden dynamische Theme-Farben generiert, die in einer tailwind-colors.css gespeichert und über eine PHP-Funktion automatisch aktualisiert werden.

- **Package- und Task-Runner**

NPM dient als zentraler Package Manager für JavaScript-Bibliotheken und Build-Skripte. Ergänzend kommt Grunt als Task-Runner zum Einsatz, unter anderem für Internationalisierung und die automatisierte Erstellung von Readme-Dateien. Grunt selbst wurde in der praktischen Anwendung jedoch nie verwendet.

- **Weitere Aspekte**

Neben den genannten Technologien werden AJAX-APIs für die Kommunikation zwischen Frontend und Backend genutzt, beispielsweise um aktuelle Spendenbeiträge dynamisch abzufragen.

Custom Post Types und Admin-Menüs erweitern die Standardfunktionalität von WordPress, sodass eigene Inhalte und Einstellungen im Backend verwaltet werden können.

3.2.3 Architektur

Trotz der Vielzahl an Funktionalität ist die Architektur des Plugins insgesamt einfach gehalten und stark auf eine zentrale Datei ausgerichtet. Sämtliche Kernfunktionen laufen in der Hauptdatei elancer-donate-games-plugin.php zusammen.

Hier sind die Custom Post Types, Admin-Menüs, Datenbankoperationen, AJAX-Endpunkte, Template-Overrides sowie das Enqueueing von Skripten und Styles koordiniert.

Der Code ist überwiegend prozedural, enthält aber einzelne objektorientierte Elemente (z. B. in den Klassen EDGAddons oder EDG Data Table), die alle im Namespace *elancer* organisiert sind.

- **Backend**

Im Backend werden Menüs und Tabellenansichten sowie eine SMTP-Konfigurationsseite bereitgestellt. Das Dashboard ist jedoch sehr rudimentär: Es zeigt lediglich Rohdaten in Tabellenform, bietet aber weder aussagekräftige Key

Performance Indicator (KPI)s noch visuelle Darstellungen oder interaktive Analysefunktionen.

- **Frontend**

Das Frontend umfasst die Einbindung von Tailwind CSS, die Spiele-Module sowie externe Bibliotheken wie GSAP und Three.js. Über Template-Overrides werden eigene Views in WordPress eingebracht, was eine gewisse Anpassung erlaubt. Die Gestaltung bleibt insgesamt funktional, aber ohne ausgeprägtes visuelles Design. Es ist anzumerken, dass viele Texte, die im Frontend dargestellt werden, nicht bearbeitbar sind. Die Views selbst sind in 3.2.1 beschrieben und visualisiert.

- **Datenzugriff**

Der Datenzugriff erfolgt direkt über die WordPress-Datenbankschnittstelle wpdb. Eine eigene Tabelle (wp_edg_game_data) wird per dbDelta angelegt, ohne zusätzliche Abstraktionsschicht oder Object-Relational Mapping (ORM). Inhalte und Konfigurationen basieren stark auf ACF Pro, das an vielen Stellen über get_field() aufgerufen wird. Dynamische Farben werden über eine in PHP generierte CSS-Datei (tailwind-colors.css) eingebunden.

- **Build- und Tooling-Umfeld**

Für den Build-Prozess kommt ein Tailwind-CLI-Workflow zum Einsatz, gesteuert über npm-Skripte. Der Task-Runner Grunt unterstützt Aufgaben wie Internationalisierung und die Generierung von Readme-Dateien.

- **Einschränkungen und Schwächen**

Die Bearbeitung von Landingpages ist ausschließlich eingeschränkt möglich, viele Texte sind fest im Code hinterlegt und bieten damit wenig Flexibilität. Das Dashboard liefert lediglich eine einfache Datenübersicht und bleibt visuell schlicht. Sicherheitsaspekte wurden nur teilweise berücksichtigt. Manche AJAX-Endpunkte verzichten aufNonce- oder Berechtigungsprüfungen und an einigen Stellen fehlt das Output-Escaping. Auch die Datenbank-Constraints sind uneinheitlich, wodurch doppelte Spielcodes möglich sind. Darüber hinaus erschwert die starke Bündelung der Logik in einer Datei die Wartbarkeit. Designprinzipien wie Service- oder Repository-Muster sind nicht umgesetzt. Ebenfalls ist die Einbindung von Assets wenig optimiert, da Skripte und Bibliotheken ohne Bündelung oder Minifizierung geladen werden.

Insgesamt ergibt sich damit ein pragmatisches Architekturkonzept, das die Umsetzung beschleunigt hat, jedoch klare Grenzen in Bezug auf Flexibilität, Wartbarkeit und Sicherheit erkennen lässt.

4 Konzeption und Design

Aufbauend auf der Analyse des Projektkontexts in Kapitel 3 werden in diesem Kapitel die konzeptionellen Überlegungen und das technische Design der angestrebten Lösung vorgestellt. Ziel ist es, die identifizierten Schwachstellen zu adressieren und eine tragfähige Architektur für die Umsetzung des WordPress-Plugins zu entwerfen. Dabei werden sowohl die Struktur von Backend, Frontend und Helper-Layer als auch die Integration des Gutenberg-Editors konzeptionell beschrieben, um die Basis für die Implementierung in Kapitel 5 zu schaffen.

4.1 Stakeholderanalyse und Nutzungskontext

Zuerst werden die Stakeholder und der Nutzungskontext des Projekts *Charigame* beleuchtet. Je nach Branche und Geschäftsmodell ergeben sich dabei unterschiedliche Anforderungen an die Funktionalität und den Einsatzkontext. Eine Übersicht der potenziellen Stakeholdergruppen ist in Abbildung 4.1 veranschaulicht.

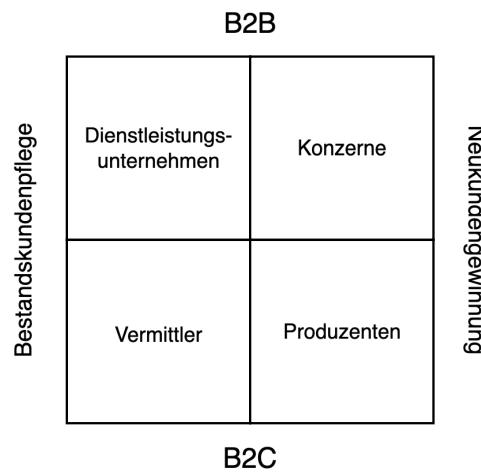


Abbildung 4.1: Stakeholderanalyse Matrix (eigene Darstellung)

1. Dienstleistungsunternehmen

Dienstleistungsunternehmen mit langfristigen Kundenbeziehungen fokussieren sich häufig primär auf die Erhaltung ihrer bestehenden Geschäftspartnerschaften. Hier geht es weniger um die Neukundengewinnung als mehr um die Pflege des bestehenden Kundenstamms. Eine gamifizierte CSR-Lösung bietet die Möglichkeit, die soziale Verantwortung sichtbar zu machen und die Bindung zu stärken.

2. Konzerne

Großunternehmen nutzen CSR-Maßnahmen strategisch zur Stärkung von Geschäftspartnerschaften und zur Förderung des Markenimages [43]. Für diese Zielgruppe ist es wichtig, dass die Skalierbarkeit nahtlos und ohne Probleme geschieht. *Charigame* kann hier einen innovativen Ansatz bieten, um Mitarbeitende, Partner und die Öffentlichkeit einzubinden.

3. Vermittler

Unternehmen mit direktem Endkundenkontakt in wettbewerbsintensiven Märkten profitieren von Alleinstellungsmerkmalen [44]. *Charigame* schafft durch seinen spielerischen Ansatz einen niedrigschwälligen Zugang zu CSR-Aktivitäten. Dieser kann die Kundenerfahrung emotional aufwerten, wodurch sich Wettbewerbsvorteile erschließen.

4. Produzenten

Für produzierende Unternehmen bietet gamifizierte CSR die Möglichkeit, eine emotionale Bindung zwischen Endverbrauchern und der Marke herzustellen [46]. Spielerisch gestaltete Spendenaktionen lassen sich nahtlos in Marketingkampagnen integrieren. Diese Aktionen stärken das Markenimage und erhöhen die Reichweite in digitalen Kanälen.

Zusammenfassend lässt sich festhalten, dass das Plugin *Charigame* sowohl im Business-to-Business (B2B)- als auch im Business-to-Consumer (B2C)-Kontext unterschiedliche Mehrwerte schaffen kann. Während im B2B-Bereich die Aspekte Reputation und Kundenbindung im Vordergrund stehen, sind es im B2C-Bereich vor allem Markenerlebnis und Differenzierung, die den Einsatz attraktiv machen.

4.2 Anforderungen an die Weiterentwicklung

Aus den Erkenntnissen der Stakeholderanalyse und den identifizierten Schwächen des deskriptiven Stands sowie der zu bearbeitenden These dieser Arbeit lassen sich diverse

Anforderungen ableiten. Diese Anforderungen wurden entsprechend dem definierten Projektumfang auf die wesentlichsten Aspekte fokussiert und werden nachfolgend adressiert.

4.2.1 Technische Anforderungen

- [T1] Das System muss die Abhängigkeit von der PRO-Version des Plugins ACF (Advanced Custom Fields) entfernen.
- [T2] Das System muss gängige WordPress-Plugin-Best-Practices befolgen.
- [T3] Das System soll gängige Security-Best-Practices der WordPress-Developer-Ressourcen berücksichtigen.

4.2.2 Funktionale Anforderungen

- [F1] Das System muss die Landingpage mit dem Gutenberg-Editor editierbar machen.
- [F2] Das System muss Gutenberg-Blöcke bereitstellen, die die bestehende Landingpage abbilden können.
- [F3] Das System soll Funktionen zum Bearbeiten von Inhalten bereitstellen, wie z. B. Erstellen, Ändern, Löschen und Speichern.
- [F4] Das System soll ein Dashboard mit mindestens 3 Elementen (z. B. Balkendiagrammen, Prozentualen Werten, Statistiken) bereitstellen.
- [F5] Das System soll dem Redakteur eine Startseite bereitstellen, auf der die wichtigsten Metriken (z. B. zuletzt gespielt, Code eingegeben, erspielter Highscore) direkt ersichtlich sind.

4.3 Abgrenzung des Projektumfangs

Zur Abgrenzung des Projektumfangs der Arbeit wurden die Anforderungen im Muss-Kann-Soll-Schema definiert. Das bestehende Projekt bietet weiterführendes Potenzial, wesentlich mehr Anforderungen zu definieren, die nicht abgedeckt wurden, um den Umfang zu begrenzen. Die definierten Anforderungen werden im weiteren Verlauf der Konzeption tiefgehend betrachtet, damit der angestrebte präskriptive Zustand erreicht wird.

4.4 Plugin-Architektur

Die Weiterentwicklung des Plugins *Charigame* strebt eine modulare und wartungsfreundliche Plugin-Architektur an, wie einleitend in Kapitel 1 skizziert. Dabei wird das Ziel verfolgt, eine klare Trennung von Verantwortlichkeiten der Funktionen umzusetzen. Ferner wird die Erweiterbarkeit und die Möglichkeit, das Projekt skalierbar zu entwickeln, ein wesentlicher Bestandteil der Architektur. Grundlegend wird ein objektorientierter Ansatz verfolgt, um die Wiederverwendbarkeit des Codes zu gewährleisten. Folgend werden die wichtigsten definierten Anforderungen im Zusammenhang mit den einzusetzenden Technologien adressiert.

4.4.1 Einzusetzende Technologien

Carbon Fields

Die Abhängigkeit des externen Plugins ACF Pro vollständig zu entfernen, geht aus der zuvor definierten Anforderung [T1] hervor. Anstelle von ACF Pro wird das Framework Carbon Fields genutzt, das über Composer eingebunden werden kann. Carbon Fields ermöglicht die flexible Definition und Verwaltung von Metafeldern innerhalb von WordPress. Im Vergleich zu ACF hat diese Lösung den Vorteil, dass sie als Bibliothek in das Plugin integriert werden kann. Darüber hinaus ist Carbon Fields eine Open Source Lösung und es ist keine kostengünstige Lizenz notwendig.

Gutenberg Blockeditor

Ein weiterer Kernbestandteil der Technologieauswahl ist die Integration des Gutenberg Block Editors. Dies erlaubt die Entwicklung maßgeschneiderter Blöcke für den Editor und stellt sicher, dass Inhalte direkt im Backend in einer Live-Vorschau bearbeitet werden können. Für das Editor-UI wird React verwendet, da es die von WordPress empfohlene und im Gutenberg-Editor standardmäßig eingesetzte Technologie ist. Die Kopplung von PHP im Backend und React im Editor ermöglicht eine klare Trennung von Daten- und Darstellungsschicht. Durch den Einsatz dieser Lösung können die Anforderungen [F1], [F2] und [F3] behandelt werden.

PHPCS

Eine weitere zentrale Thematik der Weiterentwicklung bildet die Anforderung [T2]. Unter Zuhilfenahme des PHP CodeSniffer (PHPCS) wird sichergestellt, dass die WordPress Coding Standards konsequent eingehalten werden. Diese Verwendung ermöglicht eine automatisierte Überprüfung des Codes, sodass Verstöße gegen die definierten Standards unmittelbar angezeigt werden. Auf diese Weise wird nicht nur die

Lesbarkeit, sondern auch die langfristige Wartbarkeit des Codes erheblich verbessert.

Aus der geplanten Architektur und den eingesetzten Technologien geht weiterführend der Aufbau der Komponenten hervor.

4.4.2 Aufbau der Komponenten

Das Plugin soll eine zentrale Einstiegsklasse `class-charigame.php` besitzen, die in der Hauptdatei des Plugins initialisiert wird. Durch den definierten Plugin-Header sowie die Implementierung der WordPress-Hooks für Aktivierung und Deaktivierung wird diese Klasse geladen und stößt die Instanziierung aller weiteren benötigten Abhängigkeiten an. Die Architektur sieht eine Reihe spezialisierter Klassen vor, welche die unterschiedlichen Funktionsbereiche des Plugins abbilden. Dazu gehören unter anderem:

- **Charigame Blocks:** Verantwortlich für die Registrierung und Bereitstellung der im Plugin genutzten Gutenberg-Blöcke (Block Editor (Gutenberg)). Darüber hinaus wird eine eigene Block-Kategorie eingeführt und die notwendigen Assets werden eingebunden.
- **Login Handler:** Kümmert sich um die Verwaltung der Loginsessions der Teilnehmer, prüft Gamecodes und stößt das Rendering der Landingpage an.
- **Color Manager:** Ermöglicht die Registrierung von Kampagnenfarben und deren dynamische Integration in das DOM.
- **Donation Manager:** Übernimmt die Logik der Spendenverteilung, speichert und aktualisiert Ergebnisse, berechnet Anteile und stellt diese für die weitere Verarbeitung bereit.
- **Email Sender:** Implementiert Funktionen zum E-Mail-Versand. Dazu gehören die Konfiguration des SMTP, das Versenden von Test- und Kampagnenmails, die Generierung des HTML-Templates sowie das Bereitstellen von Variablen.
- **Carbon Fields:** Verantwortlich für die Initialisierung und Bereitstellung des Frameworks Carbon Fields, das zur Verwaltung individueller Metafelder eingesetzt wird.

Ergänzend wird eine statische Konfigurationsdatei vorgesehen, die allgemeine Design-Mappings wie Abstände, Textausrichtungen oder Flexbox-Alignments enthält. Diese statischen Definitionen basieren auf Tailwind CSS und werden innerhalb der Gutenberg-Blöcke genutzt.

4.4.3 Custom Post Types

Die zentrale Datenstruktur des Plugins wird über die Definition mehrerer Custom Post Types (Custom Post Type) realisiert. Folgende Post Types werden vorgesehen:

- **Campaign:** Enthält alle zentralen Informationen einer Kampagne. Dazu gehören die Wahl des Spieltyps, spezifische Spieleinstellungen, Verlinkungen zu Landingpages und E-Mail-Templates sowie die Konfigurationen zur Spendenverteilung und der Zeitsteuerung. Zudem werden in der Campaign die Einstellungen für das Login-Formular hinterlegt.
- **Landingpage:** Stellt einen CPT zur Verfügung, der die Bearbeitung von Landingpages direkt im Block Editor (Gutenberg) ermöglicht.
- **Game:** Dient als Zuordnungstyp für die verschiedenen im Plugin angebotenen Spiele.
- **Game-Settings:** Wird als neuer CPT erstellt, der die Spieleinstellungen aus der Kampagne bezieht und diese wiederverwendbar macht.
- **Recipient:** Enthält Informationen zu den begünstigten Empfängern einer Kampagne, darunter Name, Logo und Beschreibung.
- **User:** Speichert Teilnehmerinformationen wie z. B. Vorname, Nachname, E-Mail-Adresse und Geburtsdatum. Außerdem werden hier Import- und E-Mail-Versand-Status dokumentiert. Die ursprüngliche Implementierung mit ACF wird durch Carbon Fields ersetzt.
- **E-Mail Template:** Ermöglicht die Verwaltung von E-Mail-Templates innerhalb des Gutenberg-Editors.

Die verwendeten Custom Post Types, stehen wie in Abbildung 4.2 visualisiert, in Relation zueinander. Hier ist ersichtlich, dass die Campaign als zentraler Punkt die diversen Custom Post Types nutzt, um alle notwendigen Informationen für die Spendenkampagne zu aggregieren. Zudem kann man erkennen, dass das geplante Dashboard die Daten aus der Kampagne und den Nutzern bezieht und diese durch weitere Parameter anreichert.

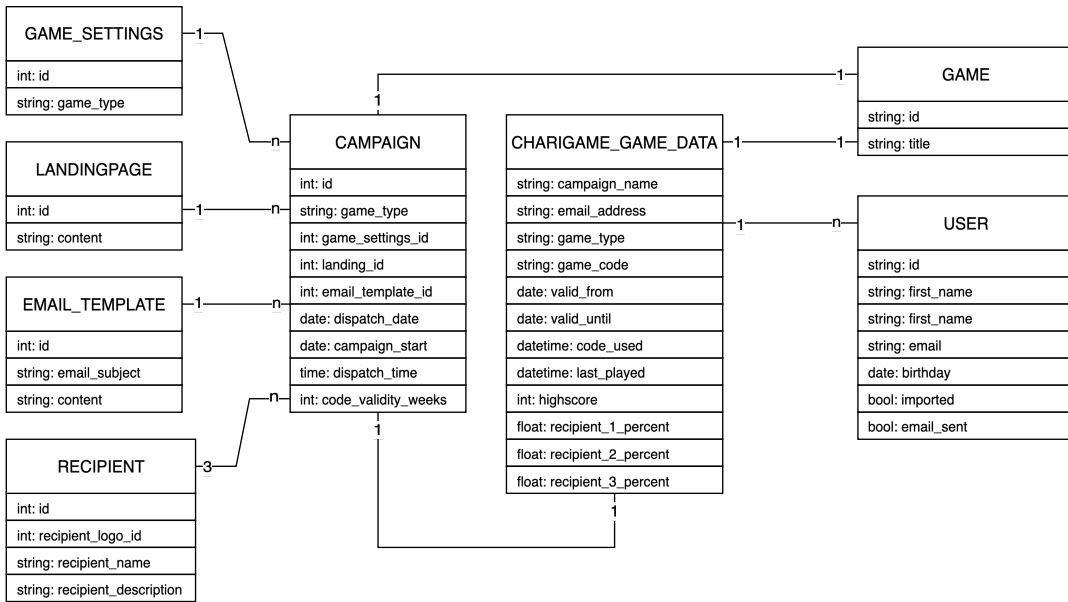


Abbildung 4.2: Relation der Custom Post Types (eigene Darstellung)

4.4.4 Verwaltungs- und Darstellungsebene

Für den Administrationsbereich ist die Bereitstellung eines eigenen Dashboards geplant, das als zentrales Menü im WordPress-Backend integriert wird. Dieses ersetzt die bisherige Lösung, die auf Data Tables basierte. Für die Darstellung sind Balkendiagramme, prozentuale Werte und Statistiken vorgesehen. Darüber hinaus gilt es Metriken direkt im Dashboard verfügbar zu machen. Sie stellt eine intuitivere Verwaltung der Kampagneninhalte bereit und adressiert die Anforderung [F4] und [F5].

Im Frontend-Bereich wird ein Template Loader eingesetzt, der die Darstellung der Campaign- und Landingpage-Inhalte übernimmt. Zusätzlich steuert ein Asset Manager das Einbinden und Entfernen aller benötigten Skripte und Styles, wie beispielsweise `backend.js`.

Ergänzend zur Trennung von Frontend und Backend ist ein dedizierter Helper-Layer vorgesehen. Dieser beinhaltet eine Helper-Komponente, die verschiedene serverseitige Funktionen bereitstellt. Dazu gehören unter anderem die Registrierung von AJAX-Actions sowie öffentliche Methoden zur Abfrage von Spendenverteilungen oder zur Verwaltung aggregierter Kampagnenergebnisse. Die Trennung in einen dedizierten Layer ermöglicht es, wiederkehrende Aufgaben zentral zu verwalten.

5 Implementierung

Auf Basis der entwickelten Konzeption folgt in diesem Kapitel die praktische Umsetzung. Dabei werden die einzelnen Komponenten des Systems beschrieben und ihre Implementierung erläutert. Neben technischen Aspekten wie der Strukturierung von Backend und Frontend sowie der Integration des Gutenberg-Editors wird auch auf die Realisierung der gesetzten Anforderungen eingegangen.

5.1 Entwicklungsumgebung und Tools

Für die Umsetzung des Projekts wurden verschiedene Tools eingesetzt. Als Entwicklungsumgebung kam PHPStorm zum Einsatz, ergänzt durch PHP-CS zur Einhaltung der WordPress-Codestandards. Google Chrome diente als Browser, während die lokale Entwicklung mit Local WP eingerichtet wurde. Zur Verwaltung von Paketen und Abhängigkeiten wurde npm verwendet und die Versionierung des Projekts erfolgte über GitHub.

Projektsetup

Im ersten Schritt der Implementierung wurde eine lokale Entwicklungsumgebung mittels Local WP eingerichtet. Die Entwicklungsumgebung basiert auf einem nginx-Webserver mit PHP Version 8.4.10 und einer MySQL-Datenbank in Version 8.0.35. WordPress wurde in der Version 6.8.2 installiert und mit den Standardkonfigurationen eingesetzt. Es wurden keine zusätzlichen Anpassungen an der Grundkonfiguration vorgenommen und keine weiteren Plugins hinzugefügt, um eine möglichst neutrale Ausgangsbasis für die Umsetzung zu gewährleisten. Zwecks Versionierung wurde Git auf der GitHub-Instanz der elancer-team GmbH verwendet und ein neuer Branch im bestehenden Repository angelegt. Der bestehende Stand des Plugins wurde abschließend in WordPress importiert und die eigentliche Implementierung begonnen.

5.2 Plugin-Struktur und Architektur

5.2.1 Ausgangsbasis und Bereinigung

Als Ausgangsbasis für die Implementierung wurde die WordPress Plugin Vorlage von DevinVinson¹ verwendet. Diese Vorlage bietet eine standardisierte Struktur für WordPress-Plugins und folgt den offiziellen WordPress-Entwicklungsrichtlinien im Hinblick auf die Best Practices [47]. Durch den Einsatz der Vorlage und unter Berücksichtigung weiterer Aspekte in der Implementierung kann die gesetzte Anforderung [T2] sichergestellt werden.

Im ersten Schritt wurden die zuvor importierten Daten aus dem bestehenden Plugin-Stand in die Vorlage eingesetzt und eine umfassende Bereinigung durchgeführt. Dabei wurden nicht verwendete Dateien identifiziert und entfernt. Zu diesen Altlasten zählten diverse Assets und Abhängigkeiten, die bereits zum Zeitpunkt des Transfers keine Funktion mehr erfüllten oder redundant wurden.

5.2.2 Migration und Anpassung der Komponenten

Migration zu Carbon Fields

Nachdem die grundlegende Bereinigung der Daten abgeschlossen war, wurde mit der Migration von ACF Pro zu Carbon Fields begonnen. Wie in Kapitel 4 konzipiert wurde Carbon Fields als Ersatz für ACF Pro implementiert. Hierzu wurde eine dedizierte Klasse für den Boot-Prozess von Carbon Fields erstellt und über Composer als Abhängigkeit in die neue Projektstruktur eingebunden. Die Initialisierung erfolgt über einen Klassenaufruf, der das Framework zentral verfügbar macht, die in Abbildung 5.1 veranschaulicht ist.

¹<https://github.com/DevinVinson/WordPress-Plugin-Boilerplate>

```

use Carbon_Fields\Carbon_Fields;

/**
 * Carbon Fields Class
 *
 * Initializes and bootstraps the Carbon Fields library.
 */
1 usage  & Christian Krenn
class ChariGame_Carbon_Fields {
    /**
     * Register hooks and actions.
     *
     * @return void
     */
    & Christian Krenn
    public function register(): void {
        add_action( 'after_setup_theme', array( $this, 'boot_carbon_fields' ) );
    }

    /**
     * Boot Carbon Fields library.
     *
     * Loads the autoloader and initializes Carbon Fields.
     *
     * @return void
     */
    1 usage  & Christian Krenn
    public function boot_carbon_fields(): void {
        require_once plugin_dir_path( dirname( path: __FILE__ ) ) . 'vendor/autoload.php';
        Carbon_Fields::boot();
    }
}

```

Abbildung 5.1: Aufbau der Klasse Charigame_Carbon_Fields (eigene Darstellung)

Alle bestehenden `get_field()`-Aufrufe des ursprünglich verwendeten ACF Plugins wurden daraufhin identifiziert und durch die entsprechenden Carbon Fields-Pendants ersetzt. Diese Migration gewährleistet die Kompatibilität mit der neuen Architektur und entfernt die externe Abhängigkeit von ACF Pro vollständig. Im Zuge der Migration der `carbon_get_post_meta()` Aufrufe mussten im nächsten Schritt die verwendeten Custom Post Types auf die neu eingesetzte Carbon Field Struktur angepasst werden. Diese Migration stellt die Umsetzung der Anforderung [T1] sicher.

Custom Post Types und REST API

Alle Custom Post Types wurden neu aufgebaut, um einen einheitlichen Duktus innerhalb des *Charigame*-Plugins zu realisieren und die Funktionalität mit Carbon Fields zu gewährleisten. Hier wurden die Namespace-Bezeichnungen angepasst und Klassen- sowie Datenbankbezeichnungen vereinheitlicht. Darüber hinaus sind die Felder, welche dem Redakteur im Backend zur Verfügung gestellt werden, durch Carbon Fields ersetzt worden. Die Umsetzung des Custom Post Types der Recipients ist in Abbildung 5.2 dargestellt.

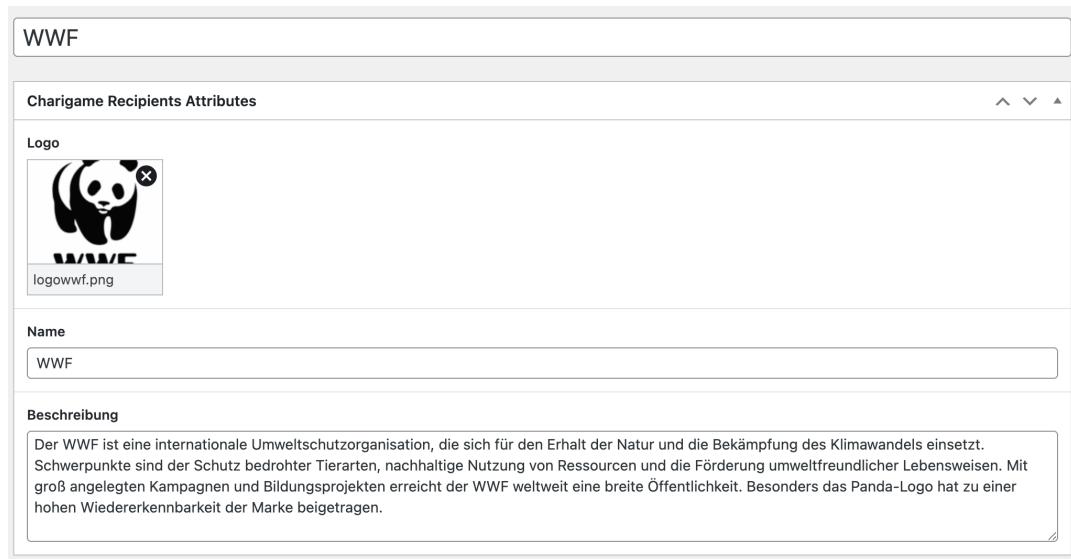


Abbildung 5.2: Donation Recipients im WordPress Backend
Enthaltenes Logos © WWF [41] (eigene Darstellung)

Bei der Neudeinition der CPTs wurden wichtige Eigenschaften der Carbon Fields für die REST API freigegeben, um diese im Backend-Editor über React abrufen zu können. Dies ist für die anschließende Umsetzung der Gutenberg-Blöcke und deren dynamische Eigenschaften erforderlich, da diese auf Kampagnendaten und verknüpfte Inhalte zugreifen. Dazu wurden unter anderem Attribute wie `game_type`, `game_settings`, `linked_landing_page` und `recipients` innerhalb der internen REST-Schnittstelle `wp-json` verfügbar gemacht.

Verzeichnisstruktur und Design-System

Die Verzeichnisstruktur wurde anhand der Plugin-Vorlage von DevinVinson erweitert und modular aufgebaut. Der Aufbau besteht aus verschiedenen Verzeichnissen, wobei die wichtigsten Verzeichnisse folgend kurz erläutert werden:

- **includes/**: Kernfunktionalitäten und Klassen
- **admin/**: Backend-spezifische Komponenten, shadcn/UI-Styles
- **frontend/**: Frontend-Darstellung und Templates
- **blocks/**: Gutenberg-Block-Definitionen
- **assets/**: Statische Ressourcen (CSS, JS, Bilder, Icons)
- **src/**: Quellcode der Block-Entwicklung und Spiele
- **templates/**: Template-Dateien für die Darstellung

Design-System

Für das Design im Front- und Backend wurde TailwindCSS verwendet. Hierzu wurden zwei unabhängig voneinander bestehende Konfiguration erstellt.

Die erste Konfiguration erfasst alle Tailwind-Klassen aus dem gesamten Projektkontext und generiert diese sowohl für das Frontend als auch das Backend. Die zweite Konfiguration ist spezifisch auf den Administrationsbereich und im Speziellen auf die shadcn/UI Komponenten abgestimmt.

Diese Implementierung im Administrationsbereich nutzt die folgenden Bausteine:

- Eine PHP-portierte shadcn/UI-Bibliothek
- Einen shadcn-Prefix zur Vermeidung von Konflikten mit WordPress-Styles
- Vorgebaute CSS-Dateien im Verzeichnis admin/css/
- Selektives Laden der Styles ausschließlich auf Charigame-Admin-Seiten

Diese Trennung gewährleistet, dass sich die unterschiedlichen Styling-Systeme nicht gegenseitig beeinträchtigen und eine saubere Kapselung der Admin-Oberfläche ermöglicht wird.

Asset Manager und Template Loader

Ein Asset Manager wurde implementiert, der selektiv die passenden Assets für den Admin- und Frontendbereich lädt. Im Adminbereich werden die zuvor erwähnten shadcn/Tailwind-Styles aus dem Design-System geladen. Im Frontend bindet der Asset Manager die Game-Styles/-Skripte ein. Diese werden nur auf den vom *Charigame* Plugin erstellten Seiten eingefügt.

Der Template Loader stellt wiederum sicher, dass Campaign-Inhalte das korrekte Template beziehen. Für den Custom Post Type der Landingpages wurde festgelegt, dass dieser nicht eigenständig betrachtet werden soll. Der Content soll ausschließlich in den Campaign-Templates ausgegeben werden. In diesem Fall führt der Template Loader eine Weiterleitung auf die referenzierte Kampagne durch.

Login-System und Sicherheit

Die Login-Logik wurde aus dem Frontend separiert und in einen dedizierten Login Manager überführt. Der Login Manager ist als Klasse `ChariGame_Login_Handler` implementiert und übernimmt zentral die Funktionen des Code-Login, des Session-Handling und der Auto-Login-Weiterleitungen. Ein Shortcode-System ermöglicht die Übertragung von Landingpage-Inhalten in Campaign-Templates, sodass eine flexible Wiederverwendung der Inhalte gewährleistet ist.

Die Code-Validierung erfolgt über die `validate_key()` Methode, die in der Datenbanktabelle `charigame_game_data` nach dem entsprechenden `game_code` sucht. Zur Performance-Optimierung wird ein mehrstufiges Caching-System mit den Cache-Gruppen `charigame_keys`, `charigame_users` und `charigame_codes` implementiert, jeweils mit einer Laufzeit von einer Stunde.

Das System implementiert mehrere Sicherheitsmaßnahmen, die im Kapitel 4 beschrieben wurden, darunter:

- **Nonce-Validierung:** `wp_verify_nonce()` schützt AJAX-Endpoints
- **Input-Sanitizing:** `sanitize_text_field()`, `intval()` für Benutzereingaben
- **Session-Management:** Sessions werden früh gestartet und konsistent genutzt
- **Caching:** Datenbank- und User-Lookups werden im Object Cache gepuffert

Im Detail wird die Nonce-Validierung bei einem Zugriff auf alle Ajax-Endpunkte verwendet. Eingaben werden früh und kontextgerecht bereinigt, bevor sie in Queries, Sessions oder Ausgaben genutzt werden.

Sessions werden früh initialisiert und konsistent genutzt, sodass Login-Zustände und Kampagnenkontakte zuverlässig und ohne Header-Probleme verfügbar sind.

Durch das angewendete Objekt-Caching werden wiederholte DB-Lookups (z. B. Code- und User-Validierung) reduziert, was die Performance erhöht und die Angriffsflächen durch unnötige Queries minimiert.

Insgesamt orientiert sich die Implementierung damit an etablierten Best Practices der WordPress-Entwicklung, indem sowohl Angriffspunkte gemindert werden und die Stabilität des Systems gewährleistet wird. Durch die implementierten Sicherheitsmechaniken konnte die gesetzte Anforderung [T3] erfüllt werden.

Color Manager

Der Color Manager verwaltet Kampagnenfarben und stellt diese innerhalb des Plugins zur Verfügung. Die Implementierung erfolgt als Singleton-Klasse, damit sichergestellt wird, dass nur eine einzige Instanz des Managers bestehen kann. Hier werden folgende Funktionalitäten bereitgestellt:

- Automatische Kampagnen-Erkennung über URL-Kontext zwecks Zuordnung der Kampagnenfarben
- CSS-Variablen für den Frontend- und Adminbereich (`-color-primary`, etc.)
- JavaScript-Objekt `charigameColors` für den Block-Editor
- Fallback auf Standard-Farbpalette

5.2.3 Backend-Integration und Dashboard

Das bestehende Dashboard wurde vollständig überarbeitet und mit den shadcn/UI-Komponenten des neuen Design-Systems integriert. Die Lösung bietet ein Responsive Design mit modernen UI-Komponenten, erweiterte Kampagnen-Metriken und eine verbesserte Benutzererfahrung. Für die Admin-Oberfläche wurden wiederverwendbare PHP-Komponenten entwickelt, welche im Styling auf der Bibliothek Shadcn/UI basieren. Dazu gehören verschiedene Komponenten wie u. a.:

- Button-Komponenten für konsistente Darstellung
- Card-Layouts für strukturierte Inhaltsblöcke
- Progress-Anzeigen für Kampagnen-Status
- Table-Komponenten für die tabellarische Datendarstellung mit Sortierung

5 Implementierung

Das neu entwickelte Dashboard ist folgend in Abbildung 5.3 zu sehen:

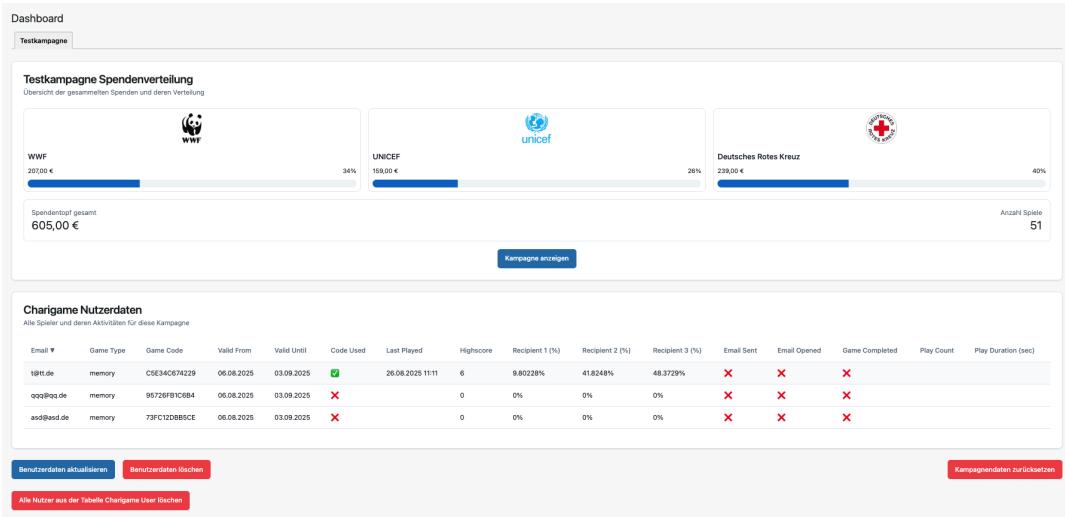


Abbildung 5.3: Neu aufgebautes Dashbord im Backend (eigene Darstellung)

Mit dem entwickelten Dashboard wurden die Anforderungen [F4] und [F5] realisiert.

E-Mail-Template System

Ein umfassendes E-Mail-Template-System wurde implementiert, das HTML-Templates als Custom Post Type im Block-Editor verfügbar macht. Dies bietet dem Redakteur die Möglichkeit, dass der Nutzer das Template der Einladungs-Mail direkt im Live-Editor anpassen kann.

Es wurden diverse Variablen wie `name`, `first_name`, `game_code`, `valid_from`, `valid_until`, `campaign_url` erstellt, welche der Redakteur innerhalb des Block-Editors verwenden kann. Optional wird der Game-Code automatisch als `?code=` Parameter an die CTA-URL angehängt, sodass die Teilnehmer mit dem Direktlink an der Spendenaktion teilnehmen können.

Die Backendansicht des neuen Template-Systems ist in Abbildung 5.4 dargestellt:

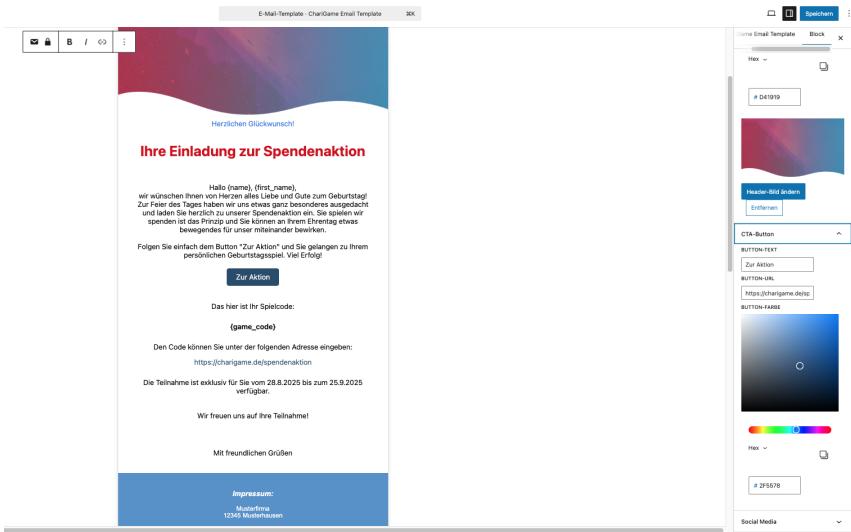


Abbildung 5.4: E-Mail-Template-System im Gutenberg Editor (eigene Darstellung)

Helper Funktionen

Während der Implementation wurden kontinuierlich verschiedene Getter- und Setter-Funktionen für die Plugin-Komponenten, die AJAX-Action-Registrierung sowie für die Frontend-Backend-Kommunikation benötigt. Damit diese zentral und über die verschiedenen Klassen hinaus im Projekt nutzbar sind, wurde die `class-charigame-helper.php` Datei als zentrale Klasse erstellt. Diese bietet im Projekt eine konsistente Schnittstelle für Datenabruf und -manipulation sowie ein Cache-Management zwecks Performance-Optimierung.

Spielintegration und Einstellungen

Die bestehenden Spiele wurden in die neue Architektur übernommen und an die eingesetzten Bibliotheken wie Carbon Fields gemünzt. Eine wesentliche Weiterentwicklung ist die Möglichkeit, individuelle Einstellungsmöglichkeiten separat zu erstellen und verwalten zu können.

Diese Trennung bringt den Vorteil mit sich, dass Spieleinstellungen nun pro Spieltyp erstellt werden können, ohne Änderungen am bestehenden Kampagnen-CPT vornehmen zu müssen. Zusätzlich besteht die Option, die hinterlegten Spieleinstellungen dank der modularen Struktur mehreren Kampagnen zuzuordnen.

Qualitätssicherung und Standards

Während der gesamten Implementierung wurden stets die WordPress-Best-Practices befolgt. Durch die Integration von PHPCS wurde eine automatisierte Prüfung der Code-Standards gewährleistet. Es wurden diverse Prinzipien der Codesicherung verwendet und auf die Best Practices der WordPress Coding Resources geachtet. Die objektorientierte Programmierung mit klarer Verantwortungstrennung und Modularität trägt zu einer erhöhten Testbarkeit und Wartbarkeit des Systems bei.

5.3 Gutenberg-Block-Entwicklung

Ein Kernbereich der Implementierung war der Aufbau der Gutenberg-Blöcke. Die Registrierung erfolgt im Projekt automatisiert über die Hauptklasse `class-charigame-blocks.php`, die alle Blöcke in `src/blocks/` mit vorhandener `block.json` registriert (vgl. Abbildung 5.5).

```
/*
 * Register all blocks from the src/blocks directory.
 *
 * @return void
 */
1 usage  ↗ Christian Krenn
public function register_blocks(): void {
    $blocks_dir = plugin_dir_path( file: __DIR__ ) . 'src/blocks/';

    if ( is_dir( $blocks_dir ) ) {
        $block_folders = glob( pattern: $blocks_dir . '*', flags: GLOB_ONLYDIR );

        foreach ( $block_folders as $block_folder ) {
            $block_json = $block_folder . '/block.json';
            if ( file_exists( $block_json ) ) {
                register_block_type( $block_folder );
            }
        }
    }
}
```

Abbildung 5.5: Automatisierte Gutenberg Blockregistrierung (eigene Darstellung)

Für das *Charigame* Plugin wurde darüber hinaus ein Block-Whitelisting implementiert. Hier werden im Gutenberg Block Editor im Charigame Plugin nur die Blöcke zuglassen, die speziell für das Plugin konzipiert wurden. Dies gewährleistet eine konsistente Benutzererfahrung und verhindert Layout-Inkonsistenzen.

Charigame Gutenberg Blöcke

Die implementierten Blöcke spiegeln den Stand der Weiterentwicklung wider und sind vollständig über den Gutenberg-Editor anpassbar. Durch den Einsatz der Blöcke im Gutenberg Editor sind die Anforderungen [F1],[F2] und [F3] realisiert. Innerhalb der Blöcke wurden unterschiedliche Attribute verwendet, welche jeweils pro Block im Anhang tabellarisch im Detail aufgelistet sind. Ferner ist in der folgenden Auflistung ein Verweis der Blöcke zu den korrespondierenden Backend Ansichten im Anhang gesetzt.

- **Intro Section:** Hero-Bereich mit konfigurierbaren Farben und Typografien
Block Attribute (vgl. Tabelle 6), Backend Darstellung (vgl. Abbildung 5)
- **Recipient Section:** Darstellung der Spendenempfänger
Block Attribute (vgl. Tabelle 7), Backend Darstellung (vgl. Abbildung 6)
- **Donation Section:** Dynamische Spendenverteilungs-Visualisierung
Block Attribute (vgl. Tabelle 8), Backend Darstellung (vgl. Abbildung 7)
- **How-to-Play Section:** Spielanleitung mit HeroIcons Integration
Block Attribute (vgl. Tabelle 9), Backend Darstellung (vgl. Abbildung 8)
- **Game Section:** Templatebasierte Spielintegration
Block Attribute (vgl. Tabelle 10), Backend Darstellung (vgl. Abbildung 9)
- **E-Mail Template:** Anpassbarer E-Mail Inhalt
Block Attribute (vgl. Tabelle 11), Backend Darstellung (vgl. Abbildung 10)

Jeder Block implementiert serverseitiges Rendering über die zugehörigen `render.php` Dateien des Blocks, um dynamische Inhalte und erhöhte Performance zu gewährleisten.

Die Blöcke selbst folgen der in Kapitel 4 geplanten Struktur und besitzen über die `render.php` hinaus eine `block.json` mit den verwendeten Attributen und eine `index.js` für die Anpassungen im Backend.

Zur Unterstützung der Block-Entwicklung wurden verschiedene Utilities entwickelt, damit Codedoppelungen vermieden werden. Zu den erstellten Utilities gehören u. a.

- **Colorpalette:** Stellt ein einheitliches Farbauswahl-Interface bereit
- **Iconlist:** Integriert Heroicons mit SVG-Inline-Rendering
- **Useimage:** Behandelt Medien-Handling für Block-Attribute und
- **Usecampaigndata:** Ermöglicht Kampagnendaten-Integration für dynamische Blöcke.

Die gesamte Landingpage, wie sie im Backend von den Redakteuren verwendet werden kann, ist in Abbildung 11 im Anhang zu erkennen.

5.4 Spendenverteilung und Schlussfolgerung

Während der Implementierung wurde festgestellt, dass die bestehende Spendenverteilungs-Logik nicht mehr funktionierte. Ein neuer Donation Manager wurde entwickelt, der Kampagnen-Statistiken berechnet. Die Spendenverteilung wird auf Basis des Donation Managers nach konfigurierbaren Regeln, die im Campaign CPT gesetzt werden können, durchgeführt. Zusätzlich sind Echtzeit-Updates für die Donation Section bereitgestellt und Rundungs- und Formatierungslogiken implementiert.

Mit der Implementierung wurden sowohl die technischen [T1–T3] als auch die funktionalen [F1–F5] Anforderungen umgesetzt. Dadurch entsteht ein System, das sich an den Best Practices orientiert, sicher und erweiterbar ist und die für Redakteure vorgesehenen Funktionalitäten bereitstellt. Auf dieser Basis können in Zukunft weitere Entwicklungen aufbauen, wie im Kapitel 6 thematisiert wird.

6 Fazit und Ausblick

Ziel dieser Arbeit war die technische Weiterentwicklung des WordPress-Plugins *Charigame*. Im Detail galt es das Plugin zu einer modularen, wartbaren und redaktionell effizient nutzbaren Lösung durch den Einsatz des Gutenberg-Editors auszubauen. Ausgangspunkt war eine historisch gewachsene Codebasis mit starker Abhängigkeit von ACF Pro sowie einer eingeschränkten Editierbarkeit von Inhalten. Aufbauend auf den theoretischen Grundlagen der Analyse des Projektkontexts und der konzeptionellen Ausarbeitung wurden in der Implementierung konkrete Maßnahmen umgesetzt. Diese Maßnahmen haben die Architektur weiterentwickelt und zentrale Funktionalitäten überarbeitet.

6.1 Zusammenfassung der Ergebnisse

Die Weiterentwicklung führte zu Verbesserungen in mehreren Bereichen. Im Aspekt der Architektur und Wartbarkeit wurde eine klare, modulare Plugin-Architektur mit objektorientierten Komponenten eingeführt. Die Komponenten, darunter Login-Handler, Color Manager und Donation Manager, haben Funktionalitäten, die zuvor mit dem Frontend gekoppelt waren, getrennt. Dadurch konnten klare Verantwortlichkeiten gesetzt werden. Darüber hinaus ist die zukünftige Erweiterbarkeit durch die klare Zuordnung der Klassen vereinfacht.

Das Entfernen externer Lizenzkosten erfolgte durch die Migration von ACF Pro zu Carbon Fields. Bei der Gutenberg-Integration wurden eigene Gutenberg-Blöcke für die Sektionen der Landingpage entwickelt.

Standardkonformität und Sicherheit wurden durch systematische Anwendung der WordPress Coding Standards (PHPCS) gewährleistet. Hier wurde konsequentes Input-Sanitizing und Output-Escaping sowie Nonce-Validierung relevanter AJAX-Endpunkte eingesetzt.

Der Datenzugriff und die Integration sind durch Freigabe relevanter Eigenschaften in der REST-API zur Editor-Integration vereinheitlicht. Ferner wurde ein Caching von Kampagnenstatistiken eingeführt und die bestehenden Custom Post Types überarbeitet.

Im Bereich Administration und UX wurde ein modernisiertes Dashboard mit wieder-verwendbaren Admin-Komponenten aufgebaut.

6.2 Einordnung der Zielerreichung

Die formulierten Zielsetzungen wurden erreicht und die konzipierten Anforderungen erfüllt. Durch die Umsetzung der modularen Architektur, klar definierte Klassen und einer einheitlichen Namensgebung ist der Code übersichtlicher und nachhaltiger wartbar. Mit PHPCS und der Ausrichtung an den WordPress-Guidelines wurden die Best Practices konsistent angewendet. Die Gutenberg-Blöcke erlauben die Pflege aller Landingpage-Bereiche ohne Codeanpassungen.

6.3 Reflexion und Grenzen

Die Arbeit zeigt, dass die Verbindung aus WordPress-Standards, Block-Editor und modularer Architektur ein stabiles Fundament für WordPress Plugins bildet. Die neue Codebasis reduziert Altlasten und schafft klarere Verantwortlichkeiten. Die Redaktionserfahrung verbessert sich durch direkte Bearbeitung im Editor und eine geringere Abhängigkeit von Entwicklern bei Content-Änderungen. Durchgängiges Sanitizing und Escaping, Nonces und Caching erhöhen Sicherheit und Performance.

Gleichzeitig bestehen Grenzen.

Das Dashboard wurde modernisiert, bietet aber Potenzial für tiefere Auswertungen wie Visualisierungen und Exporte. Unit- und Integrationstests sowie CI/CD-Pipelines sind auszubauen, um eine Release-Sicherheit zu gewährleisten. Die UI ist konsistent, doch ein systematisch dokumentiertes Designsystem kann helfen, diese Konsistenz aufrechtzuerhalten.

6.4 Ausblick

Auf Basis der Ergebnisse ergeben sich folgende mögliche Weiterentwicklungen:

Dashboard und Analytics: KPI-Visualisierungen mit tiefergehenden Auswertungsmöglichkeiten. Ebenso ein CSV/Excel-Export der Datengrundlage.

Qualitätssicherung: Unit- und Integrationstests für PHP und JavaScript. Die Einrichtung einer CI/CD-Pipeline zwecks Verbesserung der Release-Sicherheit.

Internationalisierung: Systematische i18n der Blöcke, Admin-Views und E-Mail-Templates ermöglichen die Mehrsprachigkeit.

Performance: Erweiterte Objekt- und Transient-Caches, Asset-Bündelung und Lazy Loading steigern die Performance.

Barrierefreiheit: WCAG-orientierte Überarbeitung, Komponentenbibliothek mit dokumentierten Patterns verbessern die Zugänglichkeit.

Funktionalität: Weitere Spieltypen, erweiterte Game-Settings als wiederverwendbare Presets erweitern den Funktionsumfang.

Integration: Datenschutzkonforme Tracking-Konzepte, optionale Anbindung an externe CRMs.

6.5 Schlussbemerkung

Mit der vorliegenden Weiterentwicklung wurde *Charigame* auf eine solide, erweiterbare und standardkonforme Basis gestellt. Die redaktionelle Arbeit im Gutenberg-Editor, die weiterentwickelte Architektur sowie sicherheits- und qualitätsorientierte Maßnahmen schaffen einen messbaren Mehrwert.

Literatur

Im folgenden Literaturverzeichnis sind alle Quellen aufgeführt, die für diese Arbeit genutzt wurden. Besonders wichtig waren dabei die WordPress Developer Resources, die als Primärquelle für technische Details, Best Practices und Sicherheitsrichtlinien dienten. Ergänzt wird dies durch Fachartikel, Statistiken und weitere Dokumentationen, um die Arbeit auf eine solide Grundlage zu stellen.

- [1] Statista GmbH. „Nutzungsanteile der Content-Management-Systeme (CMS) weltweit im April 2025.“ Statista Digital Market Outlook. (2025), Adresse: <https://de.statista.com/statistik/daten/studie/320685/umfrage/nutzungsanteil-der-content-management-systeme-cms-weltweit/> (besucht am 20.07.2025).
- [2] T. Patel, S. Mittal und N. K. Awadhiya, „A Review on Content Management Systems of Web Development,“ *International Journal of Computer Science Trends and Technology*, Jg. 7, Nr. 2, S. 101–104, März 2019, ISSN: 2347-8578. Adresse: <https://www.ijcstjournal.org/volume-7/issue-2/IJCST-V7I2P22.pdf> (besucht am 20.07.2025).
- [3] WordPress Foundation. „Requirements.“ WordPress.org Official Documentation. (2024), Adresse: <https://wordpress.org/about/requirements/> (besucht am 20.07.2025).
- [4] WordPress Foundation, *WordPress Coding Standards*, WordPress Developer Documentation, 2024. Adresse: <https://developer.wordpress.org/coding-standards/wordpress-coding-standards/> (besucht am 23.07.2025).
- [5] WordPress Foundation, *What is a Plugin?* WordPress Developer Documentation, 2024. Adresse: <https://developer.wordpress.org/plugins/intro/what-is-a-plugin/> (besucht am 20.07.2025).
- [6] WordPress Developer Resources, *Plugin Best Practices – Folder Structure*, Zugegriffen am 20. Juli 2025, n.d. Adresse: <https://developer.wordpress.org/plugins/plugin-basics/best-practices/>.
- [7] WordPress Developer Resources, *Plugin Best Practices – Folder Structure*, Zugegriffen am 23. Juli 2025, n.d. Adresse: <https://developer.wordpress.org/plugins/plugin-basics/best-practices/#folder-structure>.

- [8] WordPress Developer Resources, *Plugin Best Practices – Header Requirements*, Zugegriffen am 23. Juli 2025, n.d. Adresse: <https://developer.wordpress.org/plugins/plugin-basics/header-requirements/>.
- [9] WordPress Developer Resources, *Introduction to Plugin Development*, Zugegriffen am 23. Juli 2025, n.d. Adresse: <https://developer.wordpress.org/plugins/intro/>.
- [10] WordPress Developer Resources, *Plugin Best Practices – Activation / Deactivation Hooks*, Zugegriffen am 23. Juli 2025, n.d. Adresse: <https://developer.wordpress.org/plugins/plugin-basics/activation-deactivation-hooks/>.
- [11] WordPress Developer Resources, *Plugin Best Practices – Avoid Naming Collisions*, Zugegriffen am 23. Juli 2025, n.d. Adresse: <https://developer.wordpress.org/plugins/plugin-basics/best-practices/#avoid-naming-collisions>.
- [12] WordPress Developer Resources, *Plugin Best Practices – Procedural Coding Method*, Zugegriffen am 23. Juli 2025, n.d. Adresse: <https://developer.wordpress.org/plugins/plugin-basics/best-practices/#procedural-coding-method>.
- [13] WordPress Developer Resources, *Plugin Best Practices – Check for Existing Implementations*, Zugegriffen am 23. Juli 2025, n.d. Adresse: <https://developer.wordpress.org/plugins/plugin-basics/best-practices/#check-for-existing-implementations>.
- [14] WordPress Foundation, *Validating Data*, WordPress Developer Documentation, 2024. Adresse: <https://developer.wordpress.org/apis/security/data-validation/> (besucht am 17.08.2025).
- [15] WordPress Foundation, *Sanitizing Data*, WordPress Developer Documentation, 2024. Adresse: <https://developer.wordpress.org/apis/security/sanitizing/> (besucht am 17.08.2025).
- [16] WordPress Foundation, *Escaping Functions*, WordPress Developer Documentation, 2024. Adresse: <https://developer.wordpress.org/apis/security/escaping/#escaping-functions> (besucht am 17.08.2025).
- [17] WordPress Foundation, ... *Except when you can't*, WordPress Developer Documentation, 2024. Adresse: https://developer.wordpress.org/apis/security/escaping/#toc_4 (besucht am 17.08.2025).
- [18] WordPress Foundation, *Nonces*, WordPress Developer Documentation, 2024. Adresse: <https://developer.wordpress.org/apis/security/nonces/> (besucht am 23.08.2025).

- [19] WordPress Foundation, *What Is a Theme?* WordPress Developer Documentation, 2024. Adresse: <https://developer.wordpress.org/themes/getting-started/what-is-a-theme/> (besucht am 23.08.2025).
- [20] WordPress Foundation, *Metadata*, WordPress Developer Documentation, 2024. Adresse: <https://developer.wordpress.org/plugins/metadata/> (besucht am 23.08.2025).
- [21] WordPress Foundation, *Reference*, WordPress Developer Documentation, 2024. Adresse: <https://developer.wordpress.org/rest-api/reference/> (besucht am 23.08.2025).
- [22] WordPress Foundation, *Ajax*, WordPress Developer Documentation, 2024. Adresse: <https://developer.wordpress.org/plugins/javascript/ajax/> (besucht am 23.08.2025).
- [23] WordPress Foundation, *Cron*, WordPress Developer Documentation, 2024. Adresse: <https://developer.wordpress.org/plugins/cron/> (besucht am 23.08.2025).
- [24] WordPress Foundation, *Block Editor Handbook*, WordPress Developer Documentation, 2024. Adresse: <https://developer.wordpress.org/block-editor/> (besucht am 19.08.2025).
- [25] Gutenstats, *Gutenstats: Statistische Analyse des Gutenberg-Editors*, 2025. Adresse: <https://gutenstats.blog/>.
- [26] WordPress Foundation, *Block Editor Handbook*, WordPress Developer Documentation, 2024. Adresse: <https://developer.wordpress.org/block-editor/explanations/architecture/key-concepts/> (besucht am 19.08.2025).
- [27] WordPress Foundation, *Block Editor Handbook*, WordPress Developer Documentation, 2024. Adresse: <https://developer.wordpress.org/block-editor/explanations/architecture/data-flow/> (besucht am 19.08.2025).
- [28] WordPress Foundation, *Block Editor Handbook*, WordPress Developer Documentation, 2024. Adresse: <https://developer.wordpress.org/block-editor/reference-guides/block-api/block-patterns/> (besucht am 19.08.2025).
- [29] WordPress Developer Resources, *File Structure of a Block*, Zugegriffen: 19.08.2025, 2023. Adresse: <https://developer.wordpress.org/block-editor/getting-started/fundamentals/file-structure-of-a-block/>.
- [30] WordPress Developer Resources, *Key Concepts – Templates*, Zuletzt aktualisiert am 8. Juli 2025; Erstveröffentlicht am 9. März 2021, WordPress Foundation, 2025.

-
- [31] S. Deterding, D. Dixon, R. Khaled und L. Nacke, „From Game Design Elements to Gamefulness: Defining "Gamification",“ in *Proceedings of the 15th International Academic MindTrek Conference: Envisioning Future Media Environments*, Tampere, Finland: ACM, 2011, S. 9–15. doi: 10.1145/2181037.2181040.
 - [32] Bundeszentrale für politische Bildung. „Gamification: Grundbegriffe, Chancen und Risiken.“ Online verfügbar. (2023), Adresse: <https://www.bpb.de/themen/kultur/digitale-spiele/504558/gamification-grundbegriffe-chancen-und-risiken/> (besucht am 19.08.2025).
 - [33] M. Sailer. „Die Wirkung von Gamification auf Motivation und Leistung: Empirische Studien im Kontext manueller Arbeitsprozesse.“ (2016), Adresse: <https://link.springer.com/book/10.1007/978-3-658-14309-1>.
 - [34] Europäische Kommission, *Mitteilung der Kommission an das Europäische Parlament, den Rat, den Europäischen Wirtschafts- und Sozialausschuss und den Ausschuss der Regionen: Eine neue EU-Strategie (2011-14) für die soziale Verantwortung der Unternehmen (CSR)*, KOM(2011) 681 endgültig, Brüssel, 2011. Adresse: <https://dip.bundestag.de/vorgang/mitteilung-der-kommission-an-das-europ%C3%A4ische-parlament-den-rat-den/39701>.
 - [35] H. Golrang und E. Safari, „Applying gamification design to a donation-based crowdfunding platform for improving user engagement,“ *Entertainment Computing*, Jg. 38, Article 100227, 2021. Adresse: <https://www.sciencedirect.com/science/article/abs/pii/S1875952121000227>.
 - [36] Bertelsmann-Stiftung, „Die gesellschaftliche Verantwortung von Unternehmen,“ Bertelsmann-Stiftung, Gütersloh, Germany, Techn. Ber., 2006, [Online; accessed 20-Aug-2025]. Adresse: https://www.bertelsmann-stiftung.de/fileadmin/files/BSt/Publikationen/GrauePublikationen/GP_Gesellschaftliche_Verantwortung_von_Unternehmen.pdf.
 - [37] CSR-Kompetenzzentrum OWL, *CSR 4.0 – digital und verantwortungsvoll*, Projekt im Auftrag des Wirtschaftsministeriums Nordrhein-Westfalen, GILDE-Wirtschaftsförderung Detmold und Initiative für BeschäftigungOWL e.V. Bielefeld, 2020. Adresse: https://www csr-kompetenz.de/fileadmin/bilder/Aktionen/CSR_4.0_BOX/CSR-Box_5._CSR_4.0.pdf (besucht am 23.08.2025).
 - [38] Freepik, *Hände machen Herzform (Video)*, Online, n.d. Adresse: https://de.freepik.com/gratis-video/haende-machen-herzform-cu_28749 (besucht am 20.08.2025).
 - [39] A. Romansa, *Person im roten Pullover hält Babyhand*, Online, 2016. Adresse: <https://unsplash.com/de/fotos/person-im-roten-pullover-halt-babyhand-Zyx1bK9mqmA> (besucht am 20.08.2025).

- [40] W. Foundation, *WordPress Logo*, Online, Offizielle WordPress-Medienressourcen, 2025. Adresse: <https://wordpress.org/about/logos/> (besucht am 20.08.2025).
- [41] UNICEF, Deutsches Rotes Kreuz und WWF, *Offizielle Logos, verwendet in Abbildung 3.6*, Logos © jeweilige Rechteinhaber, Verwendet in Abbildung 3.6 der Arbeit, 2025.
- [42] Bundesministerium für Arbeit und Soziales (BMAS). „CSR-Grundlagen: Was ist Corporate Social Responsibility (CSR)?“ Zugriffen: 22.08.2025. (2022), Adresse: <https://www.csr-in-deutschland.de/DE/CSR-Allgemein/CSR-Grundlagen/csr-grundlagen.html>.
- [43] Bundesministerium für Arbeit und Soziales (BMAS). „CSR – Tipps für Einsteiger.“ (o.J.), Adresse: <https://www.csr-in-deutschland.de/DE/CSR-Allgemein/CSR-in-der-Praxis/CSR-Management/Tipps-fuer-Einsteiger/tipps-fuer-einsteiger.html> (besucht am 28.08.2025).
- [44] U. Kessler. „Wettbewerbsvorteil: Alleinstellungsmerkmal (USP) und Kundennutzen.“ Zugriffen: 28.08.2025. (2025), Adresse: <https://www.franchiseportal.de/definition/wettbewerbsvorteil-a-31169>.
- [45] WordPress Developer Resources, *Custom Post Types*, Zuletzt aktualisiert am 25. August 2025, WordPress Foundation, 2025. Adresse: <https://learn.wordpress.org/lesson/custom-post-types/>.
- [46] W. Zhang, Y. Xie, J. Sun und C. Liang, „Exploring Users’ Continued Intention to Participate in Gamified Virtual CSR Co-Creation: A Gamification Affordance Perspective,“ *International Journal of Human-Computer Interaction*, Jg. 41, Nr. 8, S. 755–770, 2025. DOI: 10.1080/10447318.2025.2531275.
- [47] WordPress Developer Resources, *Boilerplate Starting Points*, Zuletzt aktualisiert am 27. August 2025, WordPress Foundation, 2025. Adresse: <https://developer.wordpress.org/plugins/plugin-basics/best-practices/#boilerplate-starting-points>.

Anhang

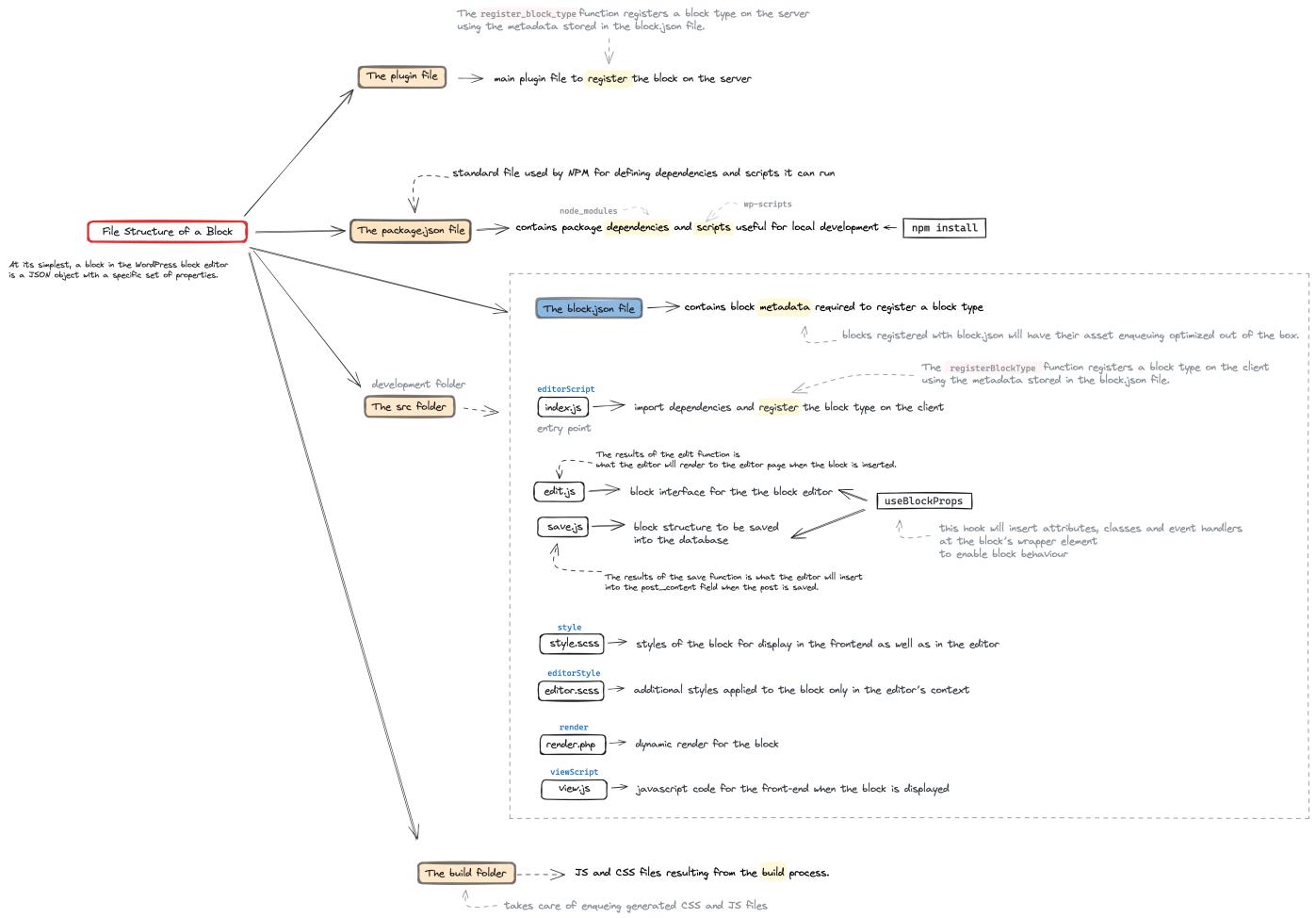


Abbildung 1: Dateistruktur eines WordPress-Blocks [29]

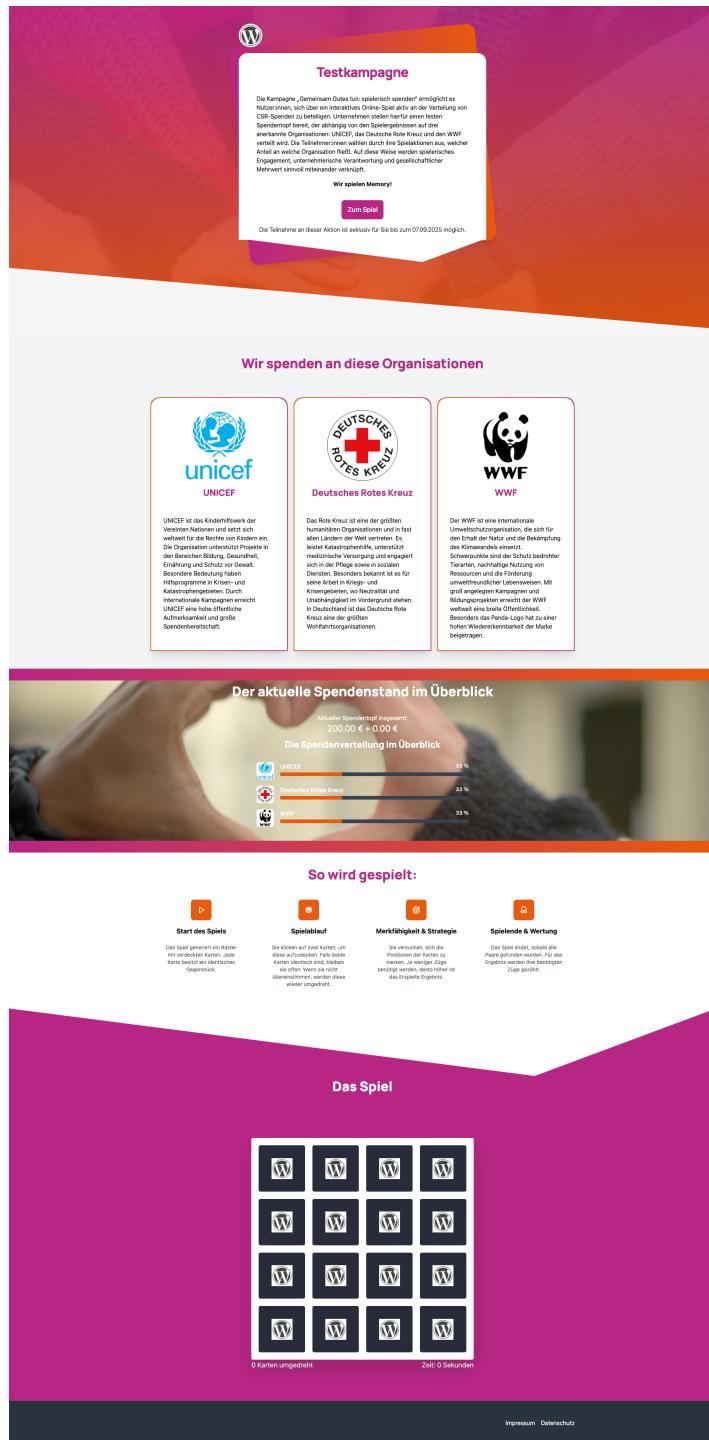


Abbildung 2: Gesamte Landingpage von Charigame im WordPress Frontend Enthaltene Logos © UNICEF, Deutsches Rotes Kreuz, WWF [41] (eigene Darstellung)

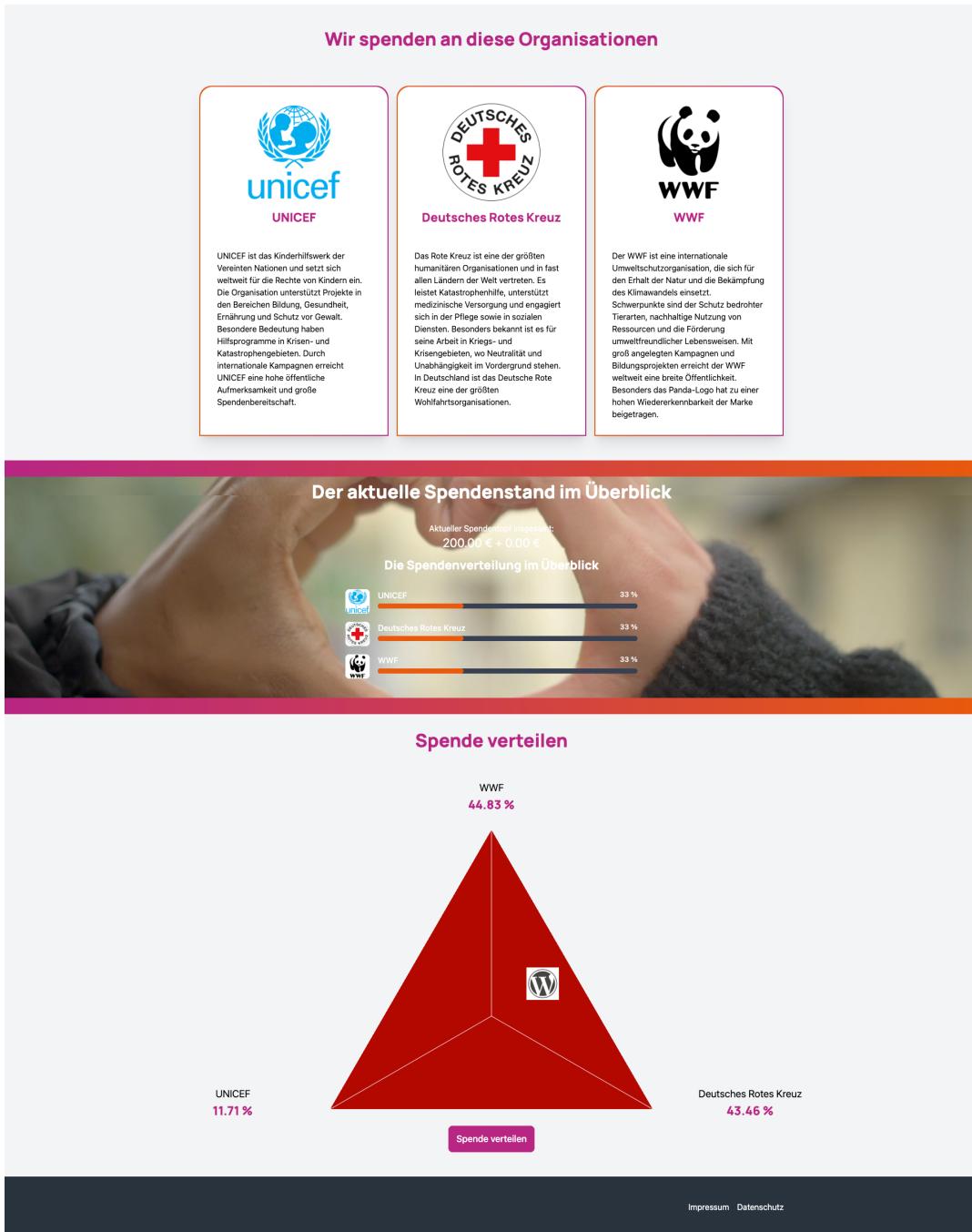


Abbildung 3: Spendenverteilungsseite von Charigame im WordPress Frontend
Enthaltene Logos © UNICEF, Deutsches Rotes Kreuz, WWF [41] (eigene Darstellung)



Abbildung 4: Dankesseite von Charigame im WordPress Frontend
Enthaltene Logos © UNICEF, Deutsches Rotes Kreuz, WWF [41]
(eigene Darstellung)

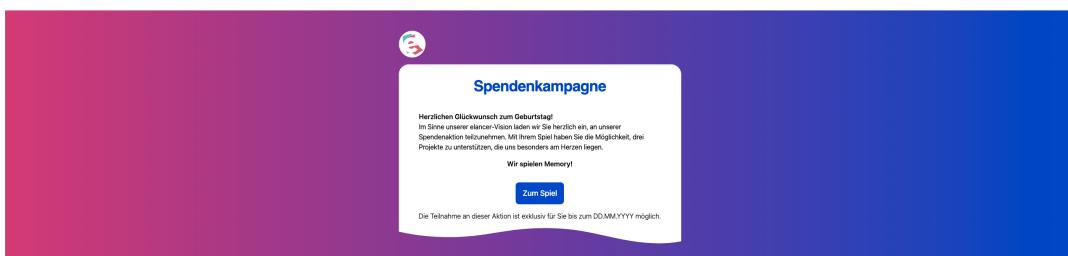


Abbildung 5: Backendansicht der Intro-Section (eigene Darstellung)



Abbildung 6: Backendansicht der Recipient-Section
Enthaltene Logos © UNICEF, Deutsches Rotes Kreuz, WWF [41] (eigene Darstellung)



Abbildung 7: Backendansicht der Donation-Section (eigene Darstellung)



Abbildung 8: Backendansicht der How-To-Play-Section
Enthaltene Logos © UNICEF, Deutsches Rotes Kreuz, WWF [41] (eigene Darstellung)



Abbildung 9: Backendansicht der Game-Section (eigene Darstellung)

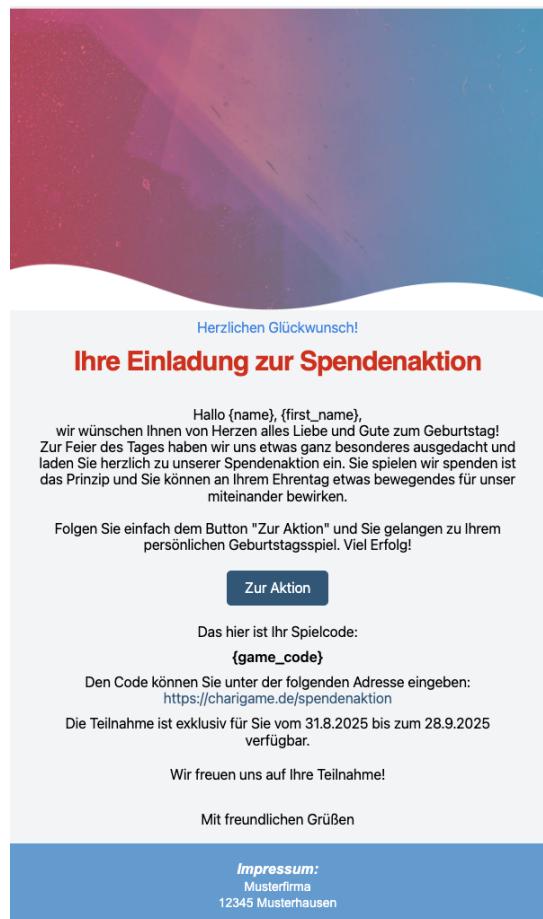


Abbildung 10: Backendansicht der Email-Edit-Section (eigene Darstellung)

Anhang

Abbildung 11: Gesamte Landingpage in der Backendansicht

Enthaltene Logos © UNICEF, Deutsches Rotes Kreuz, WWF [41] (eigene Darstellung)

Eingabefeld	Beschreibung	Eingabetyp
Login-Form Überschrift	Definiert den Headline-Text der Login-Form	Texteingabe
Background Image	Hinterlegen eines Hintergrundbildes für die Login-Form	Bild-Upload
Company Logo	Eingabe für das Unternehmenslogo	Bild-Upload
Company Name	Name des Unternehmens	Texteingabe
Company Street	Straße des Unternehmens	Texteingabe
Company City	Stadt des Unternehmens	Texteingabe
Impressum Page	Referenz auf die Impressumsseite der Website	Seitenreferenz
Datenschutz Page	Referenz auf die Datenschutzseite der Website	Seitenreferenz
AGB Page	Referenz auf die AGB-Seite der Website	Seitenreferenz
Primary Color	Definition der primären Corporate Identity (CI)-Farbe	Farbeingabe (Hex-Wert)
Secondary Color	Definition der sekundären CI-Farbe	Farbeingabe (Hex-Wert)
Tertiary Color	Definition der tertiären CI-Farbe	Farbeingabe (Hex-Wert)
Section Primary	Festlegung eines primären Abschnittstrenners	Bild-Upload
Section Secondary	Festlegung eines sekundären Abschnittstrenners	Bild-Upload
Video Moving Picture Experts Group 4 Part 14 (MP4)	Hinterlegen eines Videos für die Landingpage	Medien-Upload
Main-Font	Definition der Schriftart für die Landingpage (experimentell)	Medien-Upload

Tabelle 1: Übersicht über konfigurierbare Eingabefelder der General Settings (eigene Darstellung)

Eingabefeld	Beschreibung	Eingabetyp
Logo	Hinterlegen eines Bildes oder Logos für den Spendenempfänger	Bild-Upload
Name	Eingabe für den angezeigten Namen im Front-End	Texteingabe
Description	Festlegung des Beschreibungstextes für den Spendenempfänger	Texteingabe

Tabelle 2: Übersicht über konfigurierbare Eingabefelder der Donation Recipients
(eigene Darstellung)

Eingabefeld	Beschreibung	Eingabetyp
How To Play Headline	Definition der Überschrift der Frontend Sektion	Texteingabe
Step Icon	Hinterlegen eines Icons oder Bildes für den Spielschritt	Media-Upload
Step Headline	Definition der Überschrift des Spiel-schritts	Texteingabe
Step Text	Festlegung des Beschreibungstextes für den Spielschritt	Texteingabe
Step Color	Auswahl der Farbgebung (Primär-, Sekundär- oder Tertiärfarbe)	Auswahl

Tabelle 3: Übersicht über konfigurierbare Eingabefelder der Game Types (eigene Darstellung)

Eingabefeld	Beschreibung	Eingabetyp
First Name	Definition des Vornamens des Teilnehmers	Texteingabe
Last Name	Definition des Nachnamens des Teilnehmers	Media-Upload
E-Mail	Hinterlegen der E-Mail-Adresse des Teilnehmers	Texteingabe
Next Birthday	Festlegung des Geburtstags für des Teilnehmers	Datums-Auswahl
Imported	Auswahl, ob der Nutzer über das System importiert wurde	Checkbox
Email sent	Auswahl, ob eine Einladung an die Mail-Adresse gesendet wurde	Checkbox

Tabelle 4: Übersicht über konfigurierbare Eingabefelder der User (eigene Darstellung)

Eingabefeld	Beschreibung
Email	Zeigt die Email-Adresse des Teilnehmers
Game Type	Gibt den Spieltyp an, der für die Kampagne festgelegt wurde
Game Code	Zeigt den Code an, mit dem der Teilnehmer sich im Spiel einloggen kann
Valid From	Stellt das Datum dar, ab welchem der Game Code gültig ist
Valid Until	Stellt das Datum dar, bis welchem der Game Code gültig ist
Code Used	Gibt einen Zeitstempel an, wann der Game Code genutzt wurde um sich anzumelden
Last Played	Gibt einen Zeitstempel an, wann das Spiel zuletzt beendet wurde
Highscore	Zeigt den erspielten Punktestand des Teilnehmers an
Recipient 1	Stellt die Verteilung in Prozent dar, die der Teilnehmer für den Spendenempfänger 1 gewählt hat
Recipient 2	Stellt die Verteilung in Prozent dar, die der Teilnehmer für den Spendenempfänger 2 gewählt hat
Recipient 3	Stellt die Verteilung in Prozent dar, die der Teilnehmer für den Spendenempfänger 3 gewählt hat
E-Mail Sent	Zeigt durch 0 oder 1 an, ob die E-Mail an den Teilnehmer gesendet wurde

Tabelle 5: Übersicht über angezeigte Metriken der Data Table (eigene Darstellung)

Eingabefeld	Beschreibung
background_type	CSS-Background-Klassen für den Intro-Abschnitt
logo	Logo-Bild-URL (kleines rundes Logo oben links)
campaign_title	Hauptheadline (H1) des Intros
campaign_desc_text	Beschreibungstext der Kampagne
game_type_title	Inline-Hervorhebung im Satz „Wir spielen . . . !“
textColor	CSS-Textfarbklasse für Headline/Texte
borderStyle	Randstil des weißen Inhalts (z. B. zerrissen/wavy)

Tabelle 6: Übersicht über die Eingabefelder der Intro Section (eigene Darstellung)

Eingabefeld	Beschreibung
background_type	CSS-Background-Klassen für die Sektion
logo	Allgemeines Bildfeld (aktuell nicht genutzt)
headline	Sektions-Headline (H2)
campaign_desc_text	Allgemeiner Beschreibungstext (aktuell nicht angezeigt)
game_type_title	Allgemeines Feld (aktuell nicht verwendet)
valid_until	Allgemeines Feld (aktuell nicht verwendet)
imageID	Medien-ID (aktuell nicht verwendet)
recipients	Empfänger-Daten (Titel, Logo, Beschreibung), per API geladen
textColor	CSS-Textfarbklasse für Headline/Texte
direction	Layout-Steuerung: 'lg:flex-row' oder 'lg:flex-col'

Tabelle 7: Übersicht über die Eingabefelder der Recipient Section (eigene Darstellung)

Eingabefeld	Beschreibung
headline	H2-Überschrift der Sektion (RichText, mit Klasse via textColor)
background_type	CSS-Utility-Klassen für den Hintergrund (z. B. Gradient/Grid)
barColor	CSS-Utility-Klasse für die Progress-Bar-Farbe
donationDescText	Beschreibungstext über der Verteilungsübersicht
donationTitle	Titel/Label über dem Spendensummenwert
textColor	CSS-Textfarbkklasse für Überschriften/Texte
borderStyle	Oberer/unterer Randstil (z. B. zerrissen/wavy) des Inhaltsbereichs
recipients	Per API geladene Empfängerliste, in der Übersicht dargestellt

Tabelle 8: Übersicht über die Eingabefelder der Donation Section (eigene Darstellung)

Eingabefeld	Beschreibung
icon	Icon-Name (lädt SVG aus assets/icons/outline)
headline	Titelzeile des Items
body	Beschreibungstext des Items
headlineColor	CSS-Textfarbe der Headline
textColor	CSS-Textfarbe des Body
iconColor	Textfarbe des Icons (per currentColor auf SVG angewendet)
iconBgColor	Hintergrundfarbe des Icon-Containers
backgroundColor	Hintergrundfarbe des gesamten Items

Tabelle 9: Übersicht über die Eingabefelder eines How-to-Play-Items (eigene Darstellung)

Eingabefeld	Beschreibung
gameType	Spieltyp-Bezeichnung, per API gesetzt
headline	H2-Überschrift des Spielabschnitts
textColor	CSS-Textfarbklaasse
background_type	CSS-Background-Klassen für die Sektion

Tabelle 10: Übersicht über die Eingabefelder der Game Section (eigene Darstellung)

Eingabefeld	Beschreibung
header_image	Bild-URL für den E-Mail-Header
header_claim	Intro-Text unter dem Header-Bild
header_claim_color	Textfarbe des Header-Claims
headline	Hauptheadline (zentriert)
headline_color	Textfarbe der Headline
content	Hauptinhalt (RichText) der E-Mail
cta_text	Beschriftung des CTA-Buttons
cta_color	Hintergrundfarbe des CTA-Buttons und Linkfarbe
cta_url	Ziel-URL des CTA
info	Optionaler Info-Absatz unterhalb des CTA
signature	Optionaler Signatur-Absatz
social_media	Liste von Social-Items (Name, Link, Icon-URL)
imprint_title	Titel des Impressum-Fußers
imprint_background_color	Hintergrundfarbe des Impressums
imprint_text_color	Textfarbe des Impressums
imprint_content	Inhalt des Impressums (RichText)
game_code_text	Text vor Code-/URL-Preview
validity_text	Gültigkeitshinweis mit Variablen {valid_from}/{valid_until}
closing_text	Abschlusstext der E-Mail
add_code_parameter	Fügt optional code=GAMECODE zur URL hinzu

Tabelle 11: Übersicht über die Eingabefelder des Email Templates (eigene Darstellung)

Erklärung

Ich versichere, die von mir vorgelegte Arbeit selbstständig verfasst zu haben. Alle Stellen, die wörtlich oder sinngemäß aus veröffentlichten oder nicht veröffentlichten Arbeiten anderer oder der Verfasserin/des Verfassers selbst entnommen sind, habe ich als entnommen kenntlich gemacht. Sämtliche Quellen und Hilfsmittel, die ich für die Arbeit benutzt habe, sind angegeben. Die Arbeit hat mit gleichem Inhalt bzw. in wesentlichen Teilen noch keiner anderen Prüfungsbehörde vorgelegen.

Gummersbach, 01.09.2025

Ort, Datum

C. Krem

Unterschrift