Siemens Energy - BETI 2024

Realisierung eines Webauftritts

Socialmedia RPG

Echo

Ogulcan Kuecuek

Leon Woenckhaus

Nick Hildebrandt

Aaron Turyabahika

Andre Seiler

5. Februar 2025

Inhaltsverzeichnis

Abbildungsverzeichnis

Abkürzungsverzeichnis

CRUD engl. ”Create, Read, Update, Delete”, die Funktionen, die ein Nutzer benötigt, um Daten anzulegen und zu verwalten. 2

RPG engl. ”role-playing games”, sind ein Genre von Spielen, in denen die Spieler in die Rolle eines imaginären Charakters schlüpfen. 1, 2

SSL engl. ”Secure Sockets Layer”, ein Protokoll, das die Sicherheit der Kommunikation über das Internet gewährleistet. 3

SSR engl. ”Server Side Rendering”, ist Technik, Webseiten auf dem Server zu rendern, bevor sie an den Client-Browser gesendet werden. 2

XP engl. ”experience points”, sind ein Konzept in Spielen, das den Fortschritt eines Charakters oder Spielers zeigt. 1

1 Projektbeschreibung

Das Social Media Projekt „Echo“ ist ein innovatives, kompetitives Social Media Netzwerk, das speziell für Gamer, Digital Natives und Content Creator entwickelt wurde. Echo kombiniert die Elemente traditioneller Social Media Plattformen wie Reddit mit Rollenspiel-Mechaniken (engl. ”role-playing games”, sind ein Genre von Spielen, in denen die Spieler in die Rolle eines imaginären Charakters schlüpfen (RPG)), um ein dynamisches und interaktives Nutzererlebnis zu schaffen.

Nutzer sammeln Erfahrungspunkte (engl. ”experience points”, sind ein Konzept in Spielen, das den Fortschritt eines Charakters oder Spielers zeigt (XP)), indem sie Likes und Kommentare auf ihre Posts von anderen Nutzern erhalten. Diese XP sind ein Maß für die Aktivität und Beliebtheit eines Nutzers innerhalb der Plattform. Zusätzlich zu den XP können Nutzer durch sogenannte „Streaks“ weitere Erfahrungspunkte sammeln. Ein Streak entsteht, wenn ein Nutzer über mehrere aufeinanderfolgende Tage hinweg aktiv ist und regelmäßig Inhalte postet oder mit anderen interagiert. Je länger der Streak, desto höher die Belohnung.

Ebenfalls können Benutzer anderen Accounts folgen, um aus diesen personalisierte Inhalte zu bekommen und keine Neuigkeiten mehr zu verpassen. Dieser Wert an sogenannten Followern wird ebenfalls gezählt.

Die gesammelten Erfahrungspunkte ermöglichen es den Nutzern, ihr Profil und das Design der Website individuell anzupassen. Dies umfasst personalisierte Themes, exklusive Avatare und spezielle Layouts, die das Profil einzigartig machen. Ab einer gewissen Anzahl an XP können Nutzer ihr Profilbild, Bannerbild und Hintergrundbild selbst wählen, was zusätzliche Individualisierungsmöglichkeiten bietet.

Ein weiteres zentrales Element von Echo sind die sogenannten Communities, auf denen sich Nutzer sammeln können (Beitreten) und diese Awards durch andere Benutzer verliehen bekommen können, die im Profil angezeigt werden. Auf Communities gibt es genauso wie bei Benutzern die Möglichkeit, Posts mit Kommentaren und Likes zu versehen. Dies fördert die Interaktion und das Gemeinschaftsgefühl innerhalb der Plattform.

Im Mittelpunkt von Echo stehen Gamification-Elemente, Gruppenzugehörigkeit und das Belohnungsgefühl. Nutzer werden durch kontinuierliche Belohnungen und Fortschritte motiviert, aktiv zu bleiben und sich in der Community zu engagieren.

1.1 Projektbegründung

Das Social Media Projekt „Echo“ zeichnet sich durch seine einzigartigen Gamification- und Rollenspiel-Elemente (RPG) aus, die es zu einem besonderen Sammelpunkt für Gamer und die restliche Internetkultur machen. Diese Elemente fördern nicht nur die Interaktivität und das Engagement der Nutzer, sondern schaffen auch ein dynamisches und wettbewerbsorientiertes Umfeld, das die Nutzer motiviert, aktiv zu bleiben und sich kontinuierlich weiterzuentwickeln.

Ein weiteres technisches Highlight von Echo ist der innovative Caching-Algorithmus, der sicherstellt, dass Bilder nicht doppelt gespeichert werden. Beim Hochladen von Bildern werden doppelte Dateien erkannt und durch einen Verweis auf das bereits vorhandene Bild ersetzt. Diese Methode spart nicht nur wertvolle Speicherressourcen, sondern trägt auch zur Schonung der Umwelt bei. Durch die Reduzierung des Speicherbedarfs wird der Energieverbrauch der Server gesenkt, was wiederum den CO2-Ausstoß verringert und somit einen positiven Beitrag zum Umweltschutz leistet. Laut einer Studie des Borderstep Instituts für Innovation und Nachhaltigkeit verursachen Rechenzentren in Deutschland jährlich etwa 13 Millionen Tonnen CO2-Emissionen [1], was die Bedeutung ressourcenschonender Technologien unterstreicht.

Die Motivation für das Projekt war es, eine minimalistische und schnelle Plattform speziell für Gamer zu entwickeln, um die Gaming- und Internetkultur in einen diversifizierten multimedialen Austausch zu stellen. Echo kombiniert technologische Innovation mit einer klaren Zielgruppenausrichtung, um eine Plattform zu schaffen, die sowohl funktional als auch nachhaltig ist. Die Integration von Gamification- und RPG-Elementen macht Echo zu einem einzigartigen Erlebnis für Nutzer, während die ressourcenschonende Technologie die Umweltbelastung minimiert. Diese Kombination aus Innovation und Zielgruppenfokussierung macht Echo zu einer herausragenden Plattform im Bereich der sozialen Medien.

1.2 Projektabgrenzung

Das Social Media Projekt „Echo“ wird durch ein serverseitig gerendertes Frontend realisiert (engl. ”Server Side Rendering”, ist Technik, Webseiten auf dem Server zu rendern, bevor sie an den Client-Browser gesendet werden (SSR)), das eine nahtlose und schnelle Benutzererfahrung gewährleistet. Dazu wird eine API entwickelt, die Daten aus einer relationalen Datenbank über die CRUD-Operationen (engl. ”Create, Read, Update, Delete”, die Funktionen, die ein Nutzer benötigt, um Daten anzulegen und zu verwalten (CRUD)) zur Verfügung stellt. Diese Architektur ermöglicht eine effiziente und skalierbare Datenverwaltung, die den Anforderungen einer dynamischen Social Media Plattform gerecht wird.

Das Projekt wird umfassend dokumentiert, wie in diesem Dokument beschrieben, und zusätzlich durch eine Abschlusspräsentation ergänzt. Diese Präsentation wird die wichtigsten Aspekte und Ergebnisse des Projekts zusammenfassen und visuell ansprechend darstellen.

Für die Vorstellung des Projekts wird „Echo“ auf einem Server im Internet bereitgestellt und über eine eigene Domain mit einem entsprechenden SSL-Zertifikat ((engl. ”Secure Sockets Layer”, ein Protokoll, das die Sicherheit der Kommunikation über das Internet gewährleistet (SSL))) erreichbar gemacht. Dies stellt sicher, dass die Plattform sicher und zuverlässig zugänglich ist und den modernen Sicherheitsstandards entspricht.

2 Projektplanung

Zu Beginn des Projekts haben wir uns als Gruppe zusammengefunden und die grundlegenden Ideen und Funktionalitäten auf mehreren Flipchartblättern diskutiert. In intensiven Debatten haben wir die verschiedenen Aspekte des Projekts durchgesprochen, um am Ende einen sehr abstrakten Mockup zu erstellen, der zeigte, wie unser Projekt aussehen sollte und welche Funktionen bzw. RPG-Elemente es enthalten sollte.

Anschließend haben wir diese Funktionen in kleinere Issues aufgeteilt, die wie folgt beschrieben und aufgebaut sind: Eine kurze, prägnante Beschreibung des vorgeschlagenen Features, die erklärt, was es neu macht und warum es sinnvoll ist. Eine Liste der konkreten Funktionen, die das Feature umfassen wird. Eine Beschreibung des idealen Nutzerflusses für dieses Feature, die erklärt, wie der Nutzer mit dem Feature interagiert. Eine Auflistung der Technologien, die für die Frontend-Implementierung verwendet werden, sowie spezielle Designanforderungen, wie z.B. die Verwendung von Icons. Eine Erklärung, wie die Benutzereinstellungen in der Datenbank gespeichert werden, z.B. durch ein neues Dokument pro Benutzer mit den entsprechenden Feldern, sowie spezifische Anforderungen an die Datenverarbeitung oder Sicherheit.

Diese Issues dienen dabei gleichzeitig auch als Basisdokumentation der Funktionalität, aus der wir dieses Dokument technisch ableiten. Die Dokumentation selbst sowie die Präsentation werden über die Git-Versionskontrolle verwaltet und über Issues getrackt. Diese Issues wurden dank Kategorien wie Kernfunktionalität, optionale Funktionalität, Backend-API und Frontend zugewiesen. Zudem konnten einige Issues andere voraussetzen, wie z.B. dass ein Login die Registrierung voraussetzt. Mehrere Issues bildeten dann Meilensteine, die wir datieren konnten.

Da wir eine agile Arbeitsweise nach Scrum verwenden, gibt es tägliche Standups, in denen jeder sagt, was er gerade getan hat, welche Probleme es dabei gab und was er als nächstes machen wird. Die Entwicklungsarbeit und Versionsverwaltung wird dann durch Git realisiert, mit der Cloud-Implementierung von GitHub, um den Entwicklungsstand aus allen Computern zu synchronisieren. Git ist ein verteiltes Versionskontrollsystem, das es uns ermöglicht, Änderungen am Code effizient zu verfolgen und zusammenzuführen. Damit es zu keinen Konflikten in der gleichzeitigen Bearbeitung von ein und derselben Datei durch mehrere Leute kommt, wurde für jede Aufgabe ein eigener Entwicklungszweig erstellt. Diese wurden nach Beendigung wieder in den Main-Zweig zurückimplementiert, um eine lauffähige Version unseres Projekts stets im Main zu behalten.

Um einen Überblick über laufende und abgeschlossene Aufgaben zu haben, wurde auf GitHub ein Kanban-Board eingerichtet (3 Spalten: Offen, In Bearbeitung und Fertig), bei dem erledigte Aufgaben nach gemeinsamer Bewertung und Verbesserung auf „Fertig“

gestellt wurden. Nachdem wir zunächst die Backend-API gemeinsam mit dem Frontend bearbeitet hatten, mussten wir aufgrund der verschiedenen domänenspezifischen Anforderungen unseren Entwicklungsprozess umstellen und das Frontend und Backend parallel, aber getrennt durch verschiedene Teammitglieder entwickeln, um das jeweilige Können optimal zu allocieren.

2.1 Teamaufbau und Rollen

1. Projektmanagement
    - Nick Hildebrandt
2. Dokumentation
    - Leon Woenckhaus
3. Präsentation
    - Aaron Turyabahika
4. Backend-API
    - Nick Hildebrandt
    - Leon Woenckhaus
5. Frontend
    - Andre Seiler
    - Ogulcan Kuecuek
    - Aaron Turyabahika
6. Deploayment und integration
    - Nick Hildebrandt

2.2 Ressourcenplanung

Detaillierte Planung der benötigten Ressourcen (Hard-/Software, Räumlichkeiten usw.).

Ggfs. sind auch personelle Ressourcen einzuplanen (z.B. unterstützende Mitarbeiter).

Hinweis: Häufig werden hier Ressourcen vergessen, die als selbstverständlich ange- sehen werden (z.B. PC, Büro).

2.3 Kostenplanung

2.4 Zeitplanung

First plan: Cold freeze: 10.02.25 / Hard Freeze: 13.02.25 Präsentation 17.02.25 (to be reviewed) Präsentationsdatum korrigiert: 20.02.25, Cold Freeze: 13.02.25/Hard Freeze 16.02.25

Das Projekt findet innerhalb eines festgelegten Zeitraums statt, wobei die tägliche Arbeitszeit auf 8 Stunden pro Person festgelegt ist. Es steht jedem Teammitglied frei, zusätzliche Zeit zu investieren, wenn es dies wünscht. Der Projektumfang wurde so geplant, dass die reguläre Arbeitszeit von 8 Stunden pro Tag pro Person ausreicht, um das Projekt zufriedenstellend abzuschließen. Sollten Teammitglieder bereit sein, mehr Zeit zu investieren, können zusätzliche Features und Verbesserungen implementiert werden, die über die ursprünglichen Anforderungen hinausgehen.

Mit der Ideenfindung für das Projekt wurde noch vor den Weihnachtsferien begonnen. Im Januar wurde viel Zeit mit der genauen Planung des Projekts und des Projektumfangs verbracht. Der offizielle Startschuss für die Umsetzung fiel am 20. Januar (*siehe Gantt-Diagramm? Nochmal nachschlagen wann Planung aufhört und Arbeit anfängt*). Abgesehen von ein paar Meilensteinen, die von vornherein feststanden, wird ab diesem Zeitpunkt mit dem SCRUM-Modell gearbeitet: In kurzen Sprints werden Issues erstellt und abgearbeitet. Dadurch kann flexibel reagiert werden, wenn eine Aufgabe mehr oder weniger Zeit als erwartet benötigt. Feste Zeiträume von Anfang an für jede Aufgabe festzulegen, ist aufgrund der Komplexität einiger Aufgaben unmöglich. Es kann nicht immer genau eingeschätzt werden, wie schnell eine Aufgabe bearbeitet werden kann. Es ist jedoch möglich, die Komplexität einer Aufgabe einzuschätzen und dementsprechend Ressourcen zuzuweisen.

Verfeinerung der Zeitplanung, die bereits im Projektantrag vorgestellt wurde.

Gant Diagramm

3 Zielplattform und Implementierung

Beschreibung der Kriterien zur Auswahl der Zielplattform (u.a. Programmiersprache, Datenbank, Client/Server, Hardware). Zielplatform: Linux Webbrowser gecko, webkit, v8 serverside rendered nodejs Linux backend was ist node node goldstandard (quelle hier) node.js statt deno: Zuvor mit deno kompalitbitätsproblem. Node allerdings besser etabliert, mehr Dokumentation aufgrund längerem bestehen. Node.js ist ausserdem besser kompatibel mit Nuxt zum Zeitpunkt des Projektbeginns.

3.1 Architekturdesign

Für unser Projekt haben wir uns für die Nutzung der Nuxt.js-Architektur entschieden, die auf dem Model-View-Controller (MVC)-Prinzip basiert. Diese Architektur ermöglicht eine klare Trennung der Datenlogik (Model), der Benutzeroberfläche (View) und der Anwendungslogik (Controller). Durch diese Trennung wird die Wartbarkeit und Erweiterbarkeit der Anwendung erheblich verbessert. Das ist für uns von großen Interesse, da wir unser Projekt so aufsetzen wollen, das wir es um weitere Features in der Zukunft erweitern können. So können wir zunächst den Social Media Aspekt des Projekts umsetzen, um darauf die RPG Elemente langsam aufzubauen. Dadurch können wir nach der Fertigstellung des Social Media Grundgerüsts jederzeit einen funktionierenden Prototypen präsentieren. Unsere Wahl für das Framework viel auf Nuxt.js, da Nuxt.js ist ein leistungsstarkes Framework ist. Es baut auf Vue.js auf und vereinfacht die Entwicklung von serverseitig gerenderten (SSR) und statisch generierten Anwendungen. Ein zentrales Konzept in Nuxt.js sind die Komponenten. Vue-Komponenten ermöglichen es, die Anwendung in wiederverwendbare und isolierte Module zu unterteilen. Diese Module können mehrfach verwendet und für verschiedene Anwendungsfälle angepasst werden, was die Entwicklung effizienter und die Codebasis übersichtlicher macht. Die Komponenten erlauben uns auch das Projekt effizient zu erweitern. Daher eignet sich Nuxt Ideal als Framework für unsere Anforderungen ans Projekt.

Unsere Wahl viel auf Nuxt über andere ähnlich aufgebaute Frameworks wie Next aufgrund folgender Auswahlkriterien: Nuxt.js bietet umfassende Unterstützung für serverseitiges Rendering und statische Seitengenerierung, was es zu einer idealen Wahl für Fullstack-Anwendungen macht. Für Nuxt existiert eine ausführliche Dokumentation und eine Vielzahl an Tutorials. Der Einstieg und die kontinuierliche Weiterentwicklung der Anwendung wird dadurch vereinfacht und Gruppenmitglieder mit weniger Programmier- Erfahrung können schneller in den Workflow eingebunden werden. Nuxt.js ist kompatibel mit den bestehenden Technologien und Umgebungen, die wir nutzen möchten, was eine nahtlose Integration und Migration ermöglicht.

Ggfs. Bewertung und Auswahl von verwendeten Frameworks sowie ggfs. eine kurze Einführung in die Funktionsweise des verwendeten Frameworks.

Prisma/SQLite Backend

Für die Umsetzung des Backends ist eine Datenbank unverzichtbar. SQLite ist für unser Projekt besonders gut geeignet, da es auch bei einer großen Anzahl von Einträgen und Abfragen eine hohe Geschwindigkeit und Effizienz bietet. Allerdings kann die direkte Einbindung von SQLite in PHP-Anwendungen die Anwendung anfällig für SQL-Injections machen. Um dieses Sicherheitsrisiko zu minimieren, haben wir uns entschieden, ein Object-Relational Mapping (ORM) zu verwenden.*(ORM Abkürzung in document handeln)* Aufgrund der vorhandenen Kompatibilität mit Nuxt.js und Node.js haben wir uns für Prisma als ORM entschieden. Prisma fungiert als eine Schicht zwischen der Datenbank und der Anwendung, die es ermöglicht, Datenbankabfragen sicher und effizient durchzuführen. Es bietet eine typsichere API, die die Entwicklung vereinfacht und gleichzeitig die Sicherheit erhöht, indem es SQL-Injections verhindert. Prisma unterstützt zudem die Migration und Verwaltung der Datenbankstruktur, was die Wartung und Weiterentwicklung der Anwendung erleichtert.

Im Rahmen des Deployments wird Nginx als Reverse HTTPS Proxy eingesetzt. Nginx übernimmt dabei die Aufgabe, eingehende Anfragen an die entsprechenden Backend-Server weiterzuleiten und sorgt so für eine effiziente Lastverteilung und erhöhte Sicherheit. Darüber hinaus wird Nginx auch für die Verwaltung der SSL-Zertifikate zuständig sein, um eine sichere HTTPS-Verbindung zu gewährleisten.*(was ist nginx quelle:https://nginx.org/en/)*

Für die automatische Verwaltung und Erneuerung der SSL-Zertifikate nutzen wir das ACME-Protokoll (Automated Certificate Management Environment). ACME ist ein Protokoll, das von der Internet Security Research Group (ISRG) entwickelt wurde und es ermöglicht, SSL/TLS-Zertifikate automatisch zu beziehen und zu erneuern. Dies reduziert den administrativen Aufwand und stellt sicher, dass unsere Zertifikate stets aktuell und sicher sind.(*acme acronym in document handeln, absatz review Nick*)

3.2 Benutzeroberfläche

Entscheidung für die gewählte Benutzeroberfläche (z.B. GUI, Webinterface).

Beschreibung des visuellen Entwurfs der konkreten Oberfläche (z.B. Mockups, Menüführung). Inspirationen: Instagram, Bluesky, Steam, Discord

Ggfs. Erläuterung von angewendeten Richtlinien zur Usability und Verweis auf Corporate Design.

3.3 Datenmodell

Entwurf/Beschreibung der Datenstrukturen (z.B. ERM und/oder Tabellenmodell, XML-Schemas) mit kurzer Beschreibung der wichtigsten (!) verwendeten Entitäten. ERM einfügen

Beschreibung der angelegten Datenbank (z.B. Generierung von SQL aus Modellie- rungswerkzeug oder händisches Anlegen), XML-Schemas usw.

3.4 Datenzugriff und Backend-API Routen

(to be reworked) Die Backend-API dient als zentrale Schnittstelle zwischen dem Frontend und dem Backend unserer Anwendung. Ihre Hauptaufgabe besteht darin, Daten wie Profile, Communities und Posts aus der Datenbank abzurufen und diese dem Frontend zur Verfügung zu stellen. Die API definiert dabei klar strukturierte Routen, die festlegen, wie diese Datenabfragen erfolgen und welche Informationen abgerufen werden können. Wir verwenden eine RESTful API (Representational State Transfer), eine Art von Web-API, die auf den Prinzipien des REST-Architekturstils basiert. RESTful APIs nutzen HTTP-Anfragen, um auf Ressourcen zuzugreifen und diese zu manipulieren. Diese Ressourcen werden durch eindeutige URLs identifiziert und können in verschiedenen Formaten wie JSON oder XML dargestellt werden. Unsere API übergibt JSON Objekte.

Durch die Nutzung der API-Routen wird die Integration ins Frontend erheblich vereinfacht. Entwickler müssen keine direkten SQL-Abfragen schreiben oder tiefgehende technische Kenntnisse besitzen, um auf die benötigten Daten zuzugreifen. Stattdessen können sie die vorgegebenen API-Endpunkte nutzen, um benutzerfreundlich und effizient die gewünschten Informationen zu erhalten und anzuzeigen.

Ein wesentliches Merkmal einer RESTful API ist ihre Statelessness, was bedeutet, dass jede Anfrage vom Client an den Server alle notwendigen Informationen enthält, um sie zu verstehen und zu verarbeiten. Dies erleichtert die Skalierbarkeit und Zuverlässigkeit der API. Darüber hinaus verwenden RESTful APIs standardisierte HTTP-Methoden wie GET, POST, PUT und DELETE, um CRUD-Operationen (Create, Read, Update, Delete) auf den Ressourcen durchzuführen.

(Jeuses chreisis umsortieren und aufdröseln (up)) Systemkontext: Darstellung, wie das Backend in die Gesamtarchitektur eingebunden ist (z. B. Diagramm mit Datenbank, Frontend, API-Gateway).

Technologiestack: Beschreibung der eingesetzten Technologien (z. B. Programmiersprache, Frameworks, Datenbank).

Designmuster: Falls zutreffend, z. B. REST, Microservices, etc.

Endpunkte: Liste der API-Endpunkte (z. B. GET /users, POST /orders). Beschreibung

des Zwecks jedes Endpunkts.

Request/Response-Formate: HTTP-Methoden (GET, POST, PUT, DELETE). Beispielanfragen und -antworten (JSON, XML, etc.). Fehlercodes und Fehlermeldungen.

Authentifizierung und Autorisierung: Beschreibung des Sicherheitskonzepts (z. B. OAuth, API-Keys). Zugriffsbeschränkungen und Rollen.

4 Abnahmephase

Welche Tests (z.B. Unit-, Integrations-, Systemtests) wurden durchgeführt und welche Ergebnisse haben sie geliefert (z.B. Logs von Unit Tests, Testprotokolle der Anwen- der)?

4.1 Bereitstellung

Welche Schritte waren zum Deployment der Anwendung nötig und wie wurden sie durchgeführt (automatisiert/manuell)?

4.2 Fazit

Wurde das Projektziel erreicht und wenn nein, warum nicht?

Ist der Auftraggeber mit dem Projektergebnis zufrieden und wenn nein, warum nicht?

Wurde die Projektplanung (Zeit, Kosten, Personal, Sachmittel) eingehalten oder haben sich Abweichungen ergeben und wenn ja, warum?

Hinweis: Die Projektplanung muss nicht strikt eingehalten werden. Vielmehr sind Abweichungen sogar als normal anzusehen. Sie müssen nur vernünftig begründet wer- den (z.B. durch Änderungen an den Anforderungen, unter-/überschätzter Aufwand).

Literatur

[1] B. Institut, „Rechenzentren in Deutschland: Eine Studie zu Energiebedarf und CO2-Emissionen,“ Borderstep Institut für Innovation und Nachhaltigkeit, 2020, Zugriff am 05. Februar 2025.