MunchMunch: Eine MERN-basierte kulinarische Web-Anwendung für verbessertes User Engagement beim Entdecken neuer Gerichte und Rezepte

Technical Report: CL-2024-42, Juni 2024

Juliana Kühn, Nikolett Rácz, Raffael Friedl, Maximilian Lippmann,
Christoph P. Neumann
CyberLytics-Lab at the Department of Electrical Engineering, Media, and Computer Science
Ostbayerische Technische Hochschule Amberg-Weiden
Amberg, Germany

***Zusammenfassung*—MunchMunch realisiert eine MERN-basierte Applikation zur Förderung der kulinarischen Bildung und Optimierung des Kocherlebnisses. Die Anwendung bietet Funktionen zur Entdeckung, Speicherung und Personalisierung von Gerichten. Nutzer können bevorzugte Speisen speichern und auf zugängliche Rezepte mit detaillierten Zutatenlisten und schrittweisen Anleitungen zugreifen, die den Kochprozess vereinfachen. Die Motivation hinter dieser Anwendung liegt in der zunehmenden Nachfrage nach digitalen Hilfsmitteln für das Kochen und die kulinarische Bildung. Die Architekturziele umfassen eine benutzerfreundliche und intuitive Oberfläche, sowie eine robuste und skalierbare Anwendungs-Infrastruktur. Die Lösungsstrategie basiert auf dem MERN-Stack (MongoDB, Express.js, React, Node.js), wobei das Frontend für die Anzeige und Interaktion zuständig ist und das Backend die Geschäftslogik verwaltet. Die Kommunikation zwischen Frontend und Backend erfolgt über eine RESTful API.**

***Index Terms*—Web technologies; MERN; Web server; User interfaces; React; Node.js; Express.js; RESTful API.**

I. Problemstellung

Die MERN-basierte Anwendung „MunchMunch“ wird entwickelt, um die kulinarische Bildung zu fördern und das Kocherlebnis zu optimieren. Nutzer können maßgeschneiderte und geschmackvolle Gerichte entdecken, Lieblingsspeisen speichern und auf zugängliche Rezepte mit detaillierten Zutatenlisten und schrittweisen Anleitungen zugreifen. Das MVP (Minimum Viable Product) umfasst folgende Punkte:

1) Registrierung und Anmeldung von Nutzern
2) Anzeige von Essensvorschlägen aus verschiedenen Kategorien
3) Suche- und Filterfunktion für Rezepte
4) Detaillierte Rezepte mit Zutatenlisten und Anleitungen
5) Favoritenfunktion für Rezepte

Sind alle diese Punkte erfüllt, ist der User in der Lage die Applikation zu nutzen. Nachdem diese abgeschlossen sind, kann die Anwendung erweitert werden. Geplante, aber noch nicht umgesetzte Erweiterungen, sind im Abschnitt „Fazit und Ausblick“ beschrieben. Netflix[1] und EAT CLUB[2] dienen als Inspiration für die aktuelle Entwicklung. Der Quellcode von MunchMunch ist im Open-Source Format unter der MIT Lizenz veröffentlicht.

A. Mission Statment

Eine gesunde Ernährung ist essenziell für das menschliche Wohlbefinden. MunchMunch macht es sich zur Aufgabe, dem Nutzer einen möglichst einfachen Zugang zu gesunden Rezepten zur Verfügung zu stellen. Durch detaillierte Rezepte soll der User in der Lage sein, jedes Rezept zu kochen. Such- und Filterfunktionen ermöglichen es dem User, Rezepte nach jeders Geschmack zu finden. Als User möchte man außerdem in der Lage sein, seine Lieblingsrezepte zu speichern, um diese später schnell und einfach zur Hand zu haben.

B. Stakeholder

Es gibt verschiedene Stakeholder, für die MunchMunch interessant ist.

1) **Singles**: Singles leben oft alleine und suchen nach einfachen, aber leckeren Rezepten, die für eine Person geeignet sind. Sie schätzen schnelle und unkomplizierte Kochanleitungen. MunchMunch kann in diesem Fall helfen, personalisierte Rezepte vorzuschlagen und einfache Schritt-für-Schritt-Anleitungen bereitzustellen.
2) **Hobbyköche:** Hobbyköche haben Spaß am Kochen und experimentieren gerne mit neuen Rezepten und Techniken. Sie suchen oft nach Inspiration und möchten ihre Fähigkeiten erweitern. Sie suchen nach vielfältigen Rezepten, detaillierten Anleitungen und Informationen zu fortgeschrittenen Kochtechniken. MunchMunch stellt eine große Auswahl an Rezepten, welche die Möglichkeit bieten, in den eigenen Lieblingsgerichten gespeichert und geteilt zu werden.
3) **Feinschmecker:** Feinschmecker legen großen Wert auf hochwertige Zutaten und raffinierte Gerichte. Sie sind oft auf der Suche nach neuen Geschmackserlebnissen und Trends in der Kulinarik. Sie wollen hochwertige Rezepte, ausgefallene Zutaten und Informationen zu neuen kulinarischen Trends. MunchMunch bietet Zugang zu exklusiven und ausgefallenen Rezepten und Empfehlungen für hochwertige Zutaten
4) **Professionelle Köche:** Professionelle Köche haben eine fundierte Ausbildung und arbeiten in der Gastronomie. Sie

Please cite as: Juliana Kühn, Nikolett Rácz, Raffael Friedl, Maximilian Lippmann und Christoph P. Neumann. *MunchMunch: Eine MERN-basierte kulinarische Web-Anwendung für verbessertes User Engagement beim Entdecken neuer Gerichte und Rezepte*. Techn. Ber. CL-2024-42. Ostbayerische Technische Hochschule Amberg-Weiden, CyberLytics-Lab at the Department of Electrical Engineering, Media, and Computer Science, Juni 2024.

suchen nach Plattformen, um neue Rezepte zu entdecken. Sie können MunchMunch nutzten um einzigartige Rezepte kennenzulernen.

5) **Hausmänner und Hausfrauen:** Hausmänner und Hausfrauen kochen regelmäßig für ihre Familie und suchen nach ausgewogenen, gesunden und schmackhaften Rezepten, die für die ganze Familie geeignet sind und allen schmecken. MunchMunch kann dabei helfen die Lieblingsrezepte zu speichern und bietet eine große Auswahl an verschiedenen Rezepten.
MunchMunch bietet somit für jeden dieser Stakeholder spezifische Vorteile, die das Kocherlebnis bereichern und an die individuellen Bedürfnisse und Vorlieben angepasst sind.

II. Vorgehensweise

A. Kontextabgrenzung

Die kulinarische Plattform greift auf eine manuell gepflegte Datenbank mit Rezepten zu. Diese Rezepte sind in verschiedene Kategorien unterteilt, die im Dashboard übersichtlich dargestellt werden. Nutzer haben die Möglichkeit, ein Konto zu erstellen, Rezepte zu favorisieren und zu speichern. Alle Nutzerinformationen werden sicher in der Datenbank verwaltet.

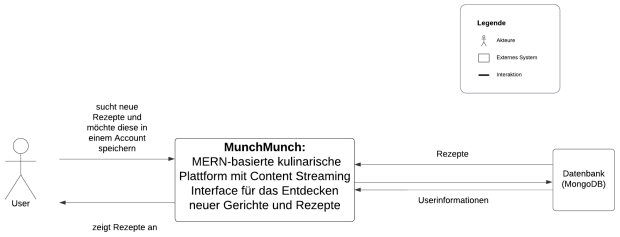

Abbildung 1. Kontextabgrenzung des MunchMunch Gesamtsystems

B. Qualitätsziele

Qualitätsziel	Erklärung
Übersichtlichkeit	Klare, nicht mit Informationen überflutete Übersicht der Rezepte unterteilt in verschiedene Kategorien
Intuitive Bedienung	User weiß sofort wie Plattform zu bedienen ist
Responsiveness	Plattform soll auf jedem Endgerät bedienbar sein
Sicherheit	Authentifizierung erfolgt über einen Token und das Passwort wird gehasht
Performance	Daten und Bilder werden vom Backend an das Frontend übermittelt

III. Randbedingungen

Bevor die Entwicklung der Plattform beginnen konnte, mussten einige Randbedingungen festgelegt werden, damit die Plattform auf jedem Rechner eines Entwicklermitglieds läuft, und später für die breiteste Masse der Zielgruppe verfügbar sein kann.

A. Technische Randbedingungen

Wichtig war, dass die Anwendung möglichst viele Nutzer erreichen kann. Deshalb wurde als Plattform der Applikation eine Web-Anwendung gewählt, welche mittels Responsive Design auf Heimcomputern, Tablets und Smartphones gleichermaßen verfügbar sein soll. Die genutzte Entwicklungsumgebung ist WebStorms von JetBrains und als Versionskontrolle wurde ein Repository auf GitHub angelegt. Dort wurden auch die Issues festgehalten und dem jeweiligen Entwickler zugewiesen. Für Icons wurde die Bibliothek Fontawesome verwendet.

B. Organisatorische Randbedingungen

Es wurde jeden Montag ein Weekly-Meeting vor der Feedback-Session abgehalten, in dem besprochen wurde, was seit dem letzten Meeting passiert ist und was für diese Woche zu tun ist. Dabei sollte jedes Teammitglied anwesend sein, sofern es nicht aus anderen Gründen verhindert war. Ein paar Aufgabenstellungen wurde gemeinsam im Team erledigt, da die Struktur dieses Projektes für jeden neu war und man so gemeinsam eine Lösung finden konnte.

C. Codier-Richtlinien

Es wurden folgende Richtlinien im Team festgelegt, damit am Ende der Programm-Code ein einheitliches Bild abgibt:

- Neue Branches müssen folgende Nennung haben: „iss(Issue-Nummer)_dev_(Kürzel des Teammitglieds)"
- Programm-Code und Kommentare werden auf Englisch geschrieben.
- Methoden, Variablen und Ordnernamen werden in „camelCase" geschrieben.
- Typen, Klassen und React-Komponenten werden hingegen in „PascalCase" notiert.

IV. Lösungsstrategien

Für den Technologie-Stack wurde der MERN-Stack gewählt. Dieser umfasst React.js[3] für das Frontend, Node.js[4] mit Express.js[5] für das Backend sowie MongoDB[6] als Datenbank. Zudem wird sowohl im Frontend als auch im Backend TypeScript[7] verwendet, um die Vorteile statischer Typisierung und verbesserter Code-Qualität zu nutzen. Vite[8] wird als Build-Tool und Entwicklungsserver verwendet, um schnelle Entwicklungszyklen und eine optimale Performance zu gewährleisten. Die Anwendung wird mittels Docker[9] containerisiert und als Pipeline eingesetzt, um das Projekt automatisch von GitHub auf eine Amazon EC2-Instanz zu pushen und auf die Liveadresse zu deployen. Mit React.js werden die interaktiven Benutzeroberflächen entwickelt, während Node.js und Express.js robuste APIs ermöglichen. Die Datenbank MongoDB bietet Flexibilität und Skalierbarkeit für die Datenverwaltung. TypeScript trägt dazu bei, die Wartbarkeit und Lesbarkeit des Codes zu verbessern und frühzeitig Fehler zu erkennen. Für die Gestaltung der Benutzeroberfläche wurden

kontinuierlich mit dem UI/UX Design Tool Figma Prototypen erstellt, damit alle Entwickler für den Aufbau des Systems eine einheitliche Grundlage haben und die gemeinsame Vision zu verbildlichen.

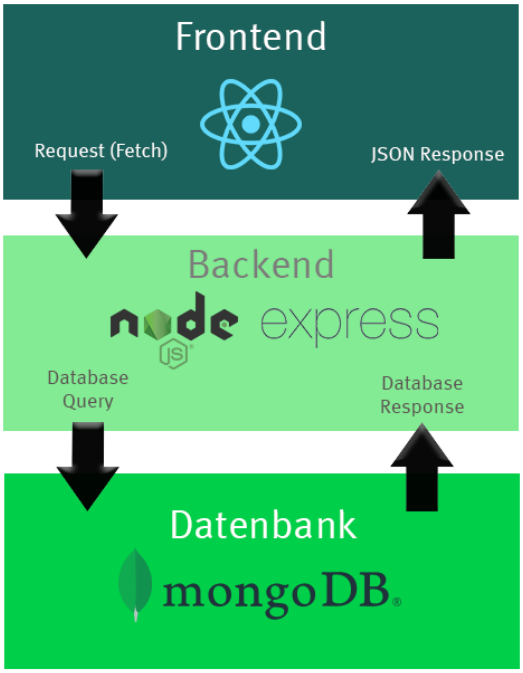

Abbildung 2. Systemüberblick visuell dargestellt

A. Datenbankarchitektur

Für die Speicherung von Daten wurde eine NoSQL, dokumentbasierte, Datenbank MongoDB verwendet. In der Datenbank werden die Speisen in der Collection Meals mit dem dazugehörigen Rezept in Recipe gespeichert. Die Entscheidung wurde getroffen, um später die Gerichte mit weiteren Rezepte erweitern zu können. Außerdem kann der Benutzer sich registrieren. Mit dem angelegten Konto hat der Benutzer die Möglichkeit, die Gerichte und ihre dazugehörigen Rezepte zu speichern.

B. Backend-Architektur

Das Backend wurde als Node.js-Server implementiert, wobei Express.js für das Routing genutzt wird. TypeScript dient als primäre Programmiersprache. Es setzt sich aus folgenden Komponenten zusammen:

1) **Serverstart** : Der Serverstart finden in src/index.ts statt.
2) **Konfiguration**: Dateien server/src/config/config.ts, server-/src/config/.env, server/src/config/.env.development, und server/src/config/.env.production enthalten die Konfiguration der Applikation. Sie beinhaltet die Integration von Middlewares, Routeneinbindung und Fehlerbehandlung.
3) **Datenbankverknüpfung**: Die Datenbankverknüpfung ist in server/src/db/Connections.ts definiert
4) **Datenmodelle**: Die Datenmodelle sind in server/src/models/datamodels definiert.
5) **Routen**: Alle Routen sind in server/src/routes definiert. Beispielsweise auth.ts, wo die Login- und Registrierungsfunktionen implementiert sind.
6) **TypeScript**-Typen: Alle TypeScript-Typen/-Enum/-Klassen/-Interfaces usw. sind im Frontend unter client/src/models/datamodels/enums und im Backend unter server/src/models/datamodels/enums definiert.
7) **Security**: Zur Sicherheit der Daten wird ein JWT Token benutzt und in der server/src/config/.env, Datei unter JWTSECRET gespeichert. Desweiteren wird in der Auth.ts eine bcrypt Funtkion benutzt um das Passwort zu hashen.

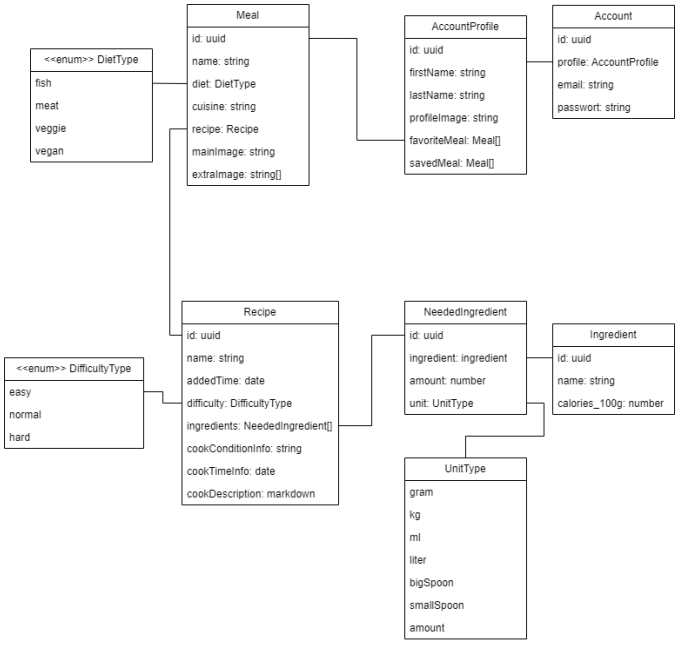

Abbildung 3. Datenmodell visuell dargestellt

C. Frontend-Architektur

Das Frontend wurde mit der UI-Bibliothek React und TypeScript umgesetzt. Es setzt sich aus den folgenden Hauptkomponenten zusammen:

1) Routen für die Navigation
2) Styling Elementen mit css und TailWind css
3) Views und Komponenten
4) Axios als http request client für Kommunikation mit dem Backend
5) Die Business Logic der Applikation ist in client/src/services implementiert.

D. CI/CD Pipeline

Im Projekt MunchMunch wird eine Continuous Integration und Continuous Deployment (CI/CD) Pipeline verwendet, um eine effiziente und automatisierte Umgebung für das Testen

und Bereitstellen von Anwendungen zu schaffen. Der Prozess beginnt mit einem Git Push auf das Hauptrepository auf GitHub, woraufhin GitHub Actions die neuesten Änderungen übernehmen und folgende Schritte durchführen:

1) **Build und Push von Docker Images:** Docker-Container werden mittels Docker Compose gebaut und die fertigen Images auf Docker Hub gepusht. Dies wird durch die im Repository hinterlegten GitHub Secrets für die Docker-Anmeldedaten und die jeweiligen Dockerfile-Dateien des Client-, sowie Server-Projekts ermöglicht.
2) **Deployment auf EC2:** Nach dem Push der Docker-Images werden diese durch einen Self-hosted Runner auf einer EC2-Instanz von Amazon[10] ausgeführt. Dieser wurde vorab durch Interaktion mit der Amazon Linux EC2 Shell via SSH und MobaXTerm als Service auf der Instanz konfiguriert. Das Deployment verwendet ein spezifisches docker.compose.yml, das im Produktionsumfeld konfiguriert ist.

Die Datei ‘docker-deploy.yml‘ steuert den Workflow von GitHub Actions und umfasst Schritte zum Einloggen in Docker Hub, Build und Push der Docker Images, sowie den Pull und Start der Container auf dem EC2-Server.

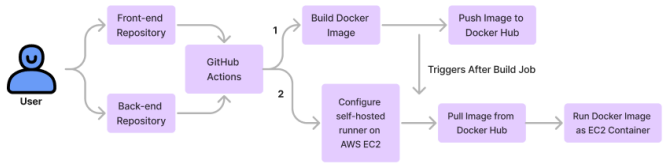

Abbildung 4. Schematische Visualisierung des Deploymentprozesses

E. Cloud Deployment

Das Projekt nutzt AWS für die Bereitstellung im Cloud-Umfeld. Folgende AWS-Dienste werden verwendet:

- **AWS Route 53:** Konfiguriert, um die Domain ‘munch-munch.tech‘ zu verwalten.
- **EC2 Instanzen:** Hosting der Anwendung auf Amazon Linux 2 EC2-Instanzen.
- **IAM Identity Center:** Verwaltet die Berechtigungen und Zugänge für die AWS-Ressourcen, sowie Zugänge zur AWS Konsole für alle Teammitglieder unter einer Root-Domäne.

Ein ‘.env production file‘ wird verwendet, um die Umgebungsvariablen im Produktionsmodus sicher zu handhaben.

V. Entwicklungswerkzeuge

A. Express.js

Express.js ist ein node-basiertes HTTP-Framework. Es wird für die Kommunikation zwischen dem Frontend und der Datenbank verwendet. Dazu nutzt es eine Verbindung zur Datenbank über die MongoDB Bibliothek. Mithilfe einer RESTful API werden die Daten über verschiedene Routen mit GET-Methoden geliefert. Um neue Daten zu der Datenbank hinzufügen zu können, oder bestehende Daten zu aktualisieren, wird die HTTP POST-Methode verwendet.

B. Nodemon

Nodemon überwacht die Dateien im Server-Verzeichnis auf Änderungen. Falls die Änderungen gespeichert werden, wird der Backend-Server neu gestartet. Der Hot-Reload-Service von Nodemon unterstützt den Entwicklungsprozess und das Debugging durch detaillierte Fehlerausgaben bei Neukompillierung.

C. Bcrypt

Bcrypt.js ist eine Krypto-Bibliothek, welche unter Anderem die bcrypt Hashfunktion implementiert. Es wird im Authentifizierungsmodul verwendet, um Passwörter der Benutzer in einem geheimen Text umwandeln zu können, um einen Standard von Sicherheit zu gewährleisten.

D. Infrastruktur und TypeScript

TypeScript wird im gesamten Projekt verwendet, um die Typsicherheit zu verbessern und die Entwicklung zu vereinfachen. Es hilft besonders bei der Definition von Datenmodellen und der sicheren Handhabung von Datentransaktionen zwischen Frontend und Backend, via der HTTP-Aufrufe innerhalb der Frontend Service-Klassen auf die RESTful API des Backends, welche innerhalb der Backend Routing-Middlewares implementiert sind.

E. Tailwind.css und Vite

Für das Styling der Benutzeroberfläche wird Tailwind.css[11] verwendet, welches effizientes und responsives Design durch Utility-first CSS-Klassen ermöglicht. Vite wird als Build-Tool und Entwicklungsserver eingesetzt, um schnelle Wiederaufbauprozesse und eine optimierte Entwicklungsperformance zu gewährleisten.

F. Axios

Axios wird als HTTP-Client genutzt, um Kommunikation zwischen dem Frontend und dem Backend zu ermöglichen. Es unterstützt das Projekt durch die einfache Handhabung von Netzwerkanfragen und das Verarbeiten von JSON-Daten.

VI. Fazit und Ausblick

Mit der erfolgreichen Implementierung des MVPs von Munch-Munch wurde eine solide Grundlage geschaffen, um die kulinarische Bildung zu fördern und das Kocherlebnis zu optimieren. Die Anwendung ermöglicht es den Nutzern, mühelos neue Gerichte zu entdecken, Rezepte zu speichern und personalisierte Kochanleitungen zu nutzen. Dank der Verwendung des MERN-Stacks konnten wir eine robuste und skalierbare Plattform entwickeln, die sowohl auf Heimcomputern als auch auf mobilen Geräten nutzbar ist.

Für die Zukunft sind folgende Erweiterungen geplant, um das Nutzererlebnis weiter zu verbessern:

1) Einkaufsliste: Als Nutzer möchte ich die Zutaten aus einem Rezept in einer Einkaufsliste speichern können, damit ich beim Einkaufen direkt die benötigten Zutaten sehen kann.
   a) Ein Button in einem Rezept zum Speichern der Zutaten wird angezeigt.
   b) Die Zutaten werden in einer Einkaufsliste, die ein weiterer Menüpunkt ist, gespeichert.

c) Die Einkaufsliste mit den gespeicherten Zutaten kann jederzeit angesehen werden.

2) Alternativer Entdeckermodus: Als Nutzer möchte ich einen spielerischen Modus haben, in dem mir jeweils nur ein einzelnes Rezept vorgeschlagen wird.
   a) Beim Wischen nach rechts wird ein Match angezeigt und das Gericht in den Favoriten gespeichert.
   b) Beim Wischen nach links wird das Nichtgefallen eines Rezepts ausgedrückt und das Gericht wird in Zukunft weniger angezeigt.
   c) In den alternativen Modus kann bei Bedarf auf der Übersicht gewechselt werden.
3) Anzeige der bisher gekochten Rezepte: Als Nutzer möchte ich sehen, welche Rezepte ich bereits gekocht habe, damit ich auch mal Neues ausprobieren kann, aber trotzdem in der Lage bin, diese nochmals zu kochen.
   a) Rezepte können mit "Gekocht"gekennzeichnet werden.
   b) Gekochte Rezepte werden nicht mehr bei den normalen Vorschlägen angezeigt.
   c) Es gibt eine eigene Kategorie, in der die bereits gekochten Rezepte angezeigt werden.
4) KI-Chatbot: Als Nutzer möchte ich einen integrierten KI-Chatbot haben, dem ich verschiedene Dinge schreiben kann, damit er mir dafür ein passendes Rezept heraussucht.
   a) Der KI-Chatbot wird als weiterer Menüpunkt angezeigt.
   b) Der KI-Chatbot hat ein Eingabefeld, um ihm Nachrichten zu schreiben.
   c) Der KI-Chatbot zeigt das passende Rezept dazu an.
   d) Das Rezept kann in die Favoritenliste gespeichert werden.

LITERATUR

[1] Netflix,Inc. *Netflix*. URL: https://www.netflix.com/de/.

[2] EAT CLUB. *EAT CLUB*. URL: https://www.eatclub.de/rezepte.

[3] Facebook. *React.js*. URL: https://react.dev/.

[4] Ryan Dahl. *Node.js*. URL: https://nodejs.org/en.

[5] Douglas Christopher Wilson. *Express*. URL: https://expressjs.com/.

[6] Eliot Horowitz, Dwight Merriman, Kevin P.Ryan. *MongoDB, Inc.* URL: https://www.mongodb.com/.

[7] Microsoft. *TypeScript: JavaScript with syntax for types*. URL: https://www.typescriptlang.org/.

[8] Evan You. *Vite: Next Generation Frontend Tooling*. URL: https://github.com/vitejs/vite.

[9] Docker. *Docker: Accelerated Container Application Development*. URL: https://www.docker.com/.

[10] Amazon. *Amazon Web Services*. URL: https://aws.amazon.com/de/.

[11] Tailwind Labs. *Tailwind CSS*. URL: https://tailwindcss.com/.

[12] Paul Brandl, Manuel Kalla, Dominik Panzer, Kevin Paulus, Manuel Pickl, Franziska Rubenbauer, Berkay Yurdaguel und Christoph P. Neumann. *Neunerln: Eine MEVN-basierte Webanwendung zum kompetitiven Kartenspielen*. Techn. Ber. CL-2023-11. Ostbayerische Technische Hochschule Amberg-Weiden, CyberLytics-Lab an der Fakultät Elektrotechnik, Medien und Informatik, Juli 2023. DOI: 10.13140/RG.2.2.33933.31209.

[13] André Kestler, Antonio Vidos, Marcus Haberl, Tobias Dobmeier, Tobias Lettner, Tobias Weiß und Christoph P. Neumann. *Computer Vision Pipeline: Eine React- und Flask-basierte Webanwendung zur No-Code-Bildverarbeitung mit Cloud-Deployment*. Techn. Ber. CL-2023-08. Ostbayerische Technische Hochschule Amberg-Weiden, CyberLytics-Lab an der Fakultät Elektrotechnik, Medien und Informatik, Juli 2023. DOI: 10.13140/RG.2.2.23866.98248.

[14] Jakob Götz, Uwe Kölbel, Maximilian Schlosser, Oliver Schmidts, Jan Schuster, Philipp Seufert, Fabian Wagner und Christoph P. Neumann. *Nautical Nonsense: Eine Phaser3- und FastAPI-basierte Webanwendung für Schiffe-Versenken mit Cloud-Deployment*. Techn. Ber. CL-2023-07. Ostbayerische Technische Hochschule Amberg-Weiden, CyberLytics-Lab an der Fakultät Elektrotechnik, Medien und Informatik, Juli 2023. DOI: 10.13140/RG.2.2.17156.09601.

[15] Lukas Feil, Stefan Reger, Timon Spichtinger, Manuel Pickl, Gian Piero Cecchetti, Alexander Hammer, Berkay Yurdagül und Christoph P. Neumann. *Torpedo Tactics: Eine MEVN-basierte Webanwendung für Schiffe-Versenken mit Cloud-Deployment*. Techn. Ber. CL-2023-06. Ostbayerische Technische Hochschule Amberg-Weiden, CyberLytics-Lab an der Fakultät Elektrotechnik, Medien und Informatik, Juli 2023. DOI: 10.13140/RG.2.2.22608.69120.

[16] Rebecca Kietzer, Baran Baygin, Carl Küschall, Jonathan Okorafor, Luca Käsmann, Michael Zimmet, Michael Ippisch und Christoph P. Neumann. *Stockbird: Eine React-basierte Webanwendung mit serverless Cloud-Deployment zur Analyse des Einfluss von Tweets auf Aktienkurs-Schwankungen*. Techn. Ber. CL-2023-04. Ostbayerische Technische Hochschule Amberg-Weiden, CyberLytics-Lab an der Fakultät Elektrotechnik, Medien und Informatik, Juli 2023. DOI: 10.13140/RG.2.2.32675.02083.

[17] Christian Rute, Alex Müller, Alexander Rudolf Wittmann, Arthur Zimmermann, David Nestmeyer, Julian Tischlak, Matthias Wolfinger und Christoph P. Neumann. *FancyChess: Eine Next.js-basierte Cloud-Anwendung zum Schachspielen*. Techn. Ber. CL-2023-03. Ostbayerische Technische Hochschule Amberg-Weiden, CyberLytics-Lab an der Fakultät Elektrotechnik, Medien und Informatik, Juli 2023. DOI: 10.13140/RG.2.2.19253.24802.

[18] Anastasia Chernysheva, Jakob Götz, Ardian Imeraj, Patrice Korinth, Philipp Stangl und Christoph P. Neumann. *SGDb Semantic Video Game Database: Svelte- und Ontotext-basierte Webanwendung mit einer Graphen-Suche für Videospiele*. Techn. Ber. CL-2023-02. Ostbayerische Technische Hochschule Amberg-Weiden, CyberLytics-Lab an der Fakultät Elektrotechnik, Medien und Informatik, März 2023. DOI: 10.13140/RG.2.2.11272.60160.

[19] Johannes Horst, Manuel Zimmermann, Patrick Sabau, Saniye Ogul, Stefan Ries, Tobias Schotter und Christoph P. Neumann. *OPCUA-Netzwerk: Angular- und FastAPI-basierte Entwicklung eines OPC-UA Sensor-Netzwerks für den Heimbereich*. Techn. Ber. CL-2023-01. Ostbayerische Technische Hochschule Amberg-Weiden, CyberLytics-Lab an der Fakultät Elektrotechnik, Medien und Informatik, März 2023. DOI: 10.13140/RG.2.2.22177.79209.

[20] Alexander Ziebell, Anja Stricker, Annika Stadelmann, Leo Schurrer, Philip Bartmann, Ronja Bäumel, Ulrich Stark und Christoph P. Neumann. *Wo ist mein Geld: Eine MERN-basierte Webanwendung für gemeinsame Ausgaben mit Freunden oder Kollegen*. Techn. Ber. CL-2022-11. Ostbayerische Technische Hochschule Amberg-Weiden, CyberLytics-Lab an der Fakultät Elektrotechnik, Medien und Informatik, Juli 2022. DOI: 10.13140/RG.2.2.28888.67847.

[21] Bastian Hahn, Martin Kleber, Andreas Klier, Lukas Kreussel, Felix Paris, Andreas Ziegler und Christoph P. Neumann. *Twitter-Dash: React- und .NET-basierte Trend- und Sentiment-Analysen*. Techn. Ber. CL-2022-07. Ostbayerische Technische Hochschule Amberg-Weiden, CyberLytics-Lab an der Fakultät

Elektrotechnik, Medien und Informatik, Juli 2022. DOI: 10.13140/RG.2.2.15466.90564.

[22] Tobias Bauer, Fabian Beer, Daniel Holl, Ardian Imeraj, Konrad Schweiger, Philipp Stangl, Wolfgang Weigl und Christoph P. Neumann. *Reddiment: Eine SvelteKit- und ElasticSearch-basierte Reddit Sentiment-Analyse*. Techn. Ber. CL-2022-06. Ostbayerische Technische Hochschule Amberg-Weiden, CyberLytics-Lab an der Fakultät Elektrotechnik, Medien und Informatik, Juli 2022. DOI: 10.13140/RG.2.2.32244.12161.

[23] Florian Bösl, Helge Kohl, Anastasia Chernysheva, Patrice Korinth, Philipp Porsch und Christoph P. Neumann. *Explosion Guy: Cloud-basiertes Matchmaking für einen graphischen Bombenspaß*. Techn. Ber. CL-2022-05. Ostbayerische Technische Hochschule Amberg-Weiden, CyberLytics-Lab an der Fakultät Elektrotechnik, Medien und Informatik, Juli 2022. DOI: 10.13140/RG.2.2.18822.34882.

[24] Dominik Smrekar, Johannes Horst, Patrick Sabau, Saniye Ogul, Tobias Schotter und Christoph P. Neumann. *OTH-Wiki: Ein Angular- und FastAPI-basiertes Wiki für Studierende*. Techn. Ber. CL-2022-04. Ostbayerische Technische Hochschule Amberg-Weiden, CyberLytics-Lab an der Fakultät Elektrotechnik, Medien und Informatik, Juli 2022. DOI: 10.13140/RG.2.2.25533.23526.

[25] Johannes Halbritter, Helge Kohl, Lukas Kreussel, Stephan Prettner, Andreas Ziegler und Christoph P. Neumann. *Graphvio: Eine Graphdatenbank-Webanwendung für integrierte Datensätze von Streaminganbietern*. Techn. Ber. CL-2022-01. Ostbayerische Technische Hochschule Amberg-Weiden, CyberLytics-Lab an der Fakultät Elektrotechnik, Medien und Informatik, März 2022. DOI: 10.13140/RG.2.2.12111.46244.

[26] Tobias Bauer, Albert Hahn, Lukas Kleinlein, Nicolas Proske, Leonard Wöllmer, Andrei Trukhin und Christoph P. Neumann. *Covidash: Eine MEAN-Variation-basierte Webanwendung für Inzidenz-Zahlen und Impffortschritt in Deutschland*. Techn. Ber. CL-2021-06. Ostbayerische Technische Hochschule Amberg-Weiden, CyberLytics-Lab an der Fakultät Elektrotechnik, Medien und Informatik, Juli 2021. DOI: 10.13140/RG.2.2.33921.84321.

[27] Cameron Barbee, Tim Hoffmann, Christian Piffel, Tobias Schotter, Sebastian Schuscha, Philipp Stangl, Thomas Stangl und Christoph P. Neumann. *FireForceDefense: Graphisches Tower-Defense-Spiel mit Kubernetes-Deployment*. Techn. Ber. CL-2021-05. Ostbayerische Technische Hochschule Amberg-Weiden, CyberLytics-Lab an der Fakultät Elektrotechnik, Medien und Informatik, Juli 2021. DOI: 10.13140/RG.2.2.20500.07048.

[28] Egidia Cenko, Madina Kamalova, Matthias Schön, Christoph Schuster, Andrei Trukhin und Christoph P. Neumann. *MedPlanner: Eine Angular- und Django-basierte Webanwendung um ärztliche Termine übersichtlich zu verwalten*. Techn. Ber. CL-2021-04. Ostbayerische Technische Hochschule Amberg-Weiden, CyberLytics-Lab an der Fakultät Elektrotechnik, Medien und Informatik, Juli 2021. DOI: 10.13140/RG.2.2.19409.71528.