




# GetraenkeIO: Eine Cloud-Native Getränkeverwaltungssoftware mit Bestandsaufnahme


Konzeptpapier: CL-CP-2025-42, selfref 2025


Eduard Taach, Justin Weinhut, Martin Dotzler, Nikolas Wegerer, Andreas Ehrles, Christoph P. Neumann   
CyberLytics-Lab an der Fakultät Elektrotechnik, Medien und Informatik  
Ostbayerische Technische Hochschule Amberg-Weiden  
Amberg, Deutschland


*Zusammenfassung*—Bla Bla Bla TODO!!! {The abstract does neither mention a teaching module nor a team/project, it is a summary of the content, thus, the functional objective – and maybe the intended technology stack. Do NOT remove the abstract , this section is mandatory. You should consider comparing your self-written abstract with the result of a generative AI that summarizes your content after you have written a nearly stable draft version. However, do not use a verbatim copy to replace your abstract, just use generative AI for inspirational purposes.}


*Index Terms*—template; lorem ipsum.


## I. EINLEITUNG | BACKGROUND AND MOTIVATION | MISSION STATEMENT | ELEVATOR PITCH

The cs-techrep formatting is adopted both from IEEE [ieee2018formattingrules] and IARIA [aria2014formattingrules] styles. The cs-techrep L<sup>A</sup>T<sub>E</sub>X class is based on IEEEtran class [ieee2015howto]. In addition, be aware of the supplementary IARIA editorial rules [aria2009editorialrules]  that provide a beginner-friendly set of further advices. It is recommended to use a grammar tool, e. g., the LanguageTool [languagetool] browser plugin in combination with Overleaf [overleaf].

The title of your paper should not exceed two lines . In exceptional cases, three lines might be allowed. A four-line-title is absolutely forbidden (hint: use the longer form in the abstract).

For capitalization of titles and section headings, use a web tool like Capitalize My Title  with the option Chicago for capitalization rules by Chicago Manual of Style (CMOS).

The pipe symbol „|“ in the section headings represents alternatives! Choose one and remove the others . The selectively provided quoted terms are special German alternatives. You may deviate from the structure of this example document and its exemplary section headings.

The introduction needs to be written from perspective of a subject-matter expert  and NOT from a technical perspective. Provide the USPs of your intended software product (in German: „fachliche Alleinstellungsmerkmale“).

## II. OPTIONAL: RELATED WORK

TODO: löschen???

## III. FUNKTIONELLES DESIGN

### A. Verfügbare Rollen

Die User Stories basieren auf zwei verschiedenen Rollen:

- 1) **Trinker:** Der Reguläre Benutzer des Systems, der in einem Vereinsheim o.ä. ab und zu ein Bierchen trink und dieses im System buchen möchte, damit er seine Zeche bezahlen kann.
- 2) **Getränkeverwalter:** Der Getränkeverwalter/Getränkewart des Vereinsheims, der sich um den Getränkenachschub und die regelmäßig das Geld der Trinker eintreibt.

## IV. FUNCTIONAL REQUIREMENTS | USER STORIES | USE CASES

Die Anforderungen an das Projekt werden im Folgenden als User-Stories formuliert, wobei die MUSS-Anforderung am Ende des Sprints das MVP bilden. Falls noch Zeit bleibt sollen im weiteren die SOLL-Anforderungen und kann-Anforderungen umgesetzt werden.

### A. MUSS-Anforderungen

USM1 **Registrieren:** Ich als Benutzer möchte mich registrieren, damit ich im System bekannt bin.

- a) Eingabe von Benutzername und Passwort durch den Benutzer.
- b) Benutzer Interagiert mit Web-Oberfläche
- c) Ausgabe Meldung an den Benutzer das es geklappt hat.
- d) Wenn nicht erfolgreich => Ausgabe einer Fehlermeldung

USM2 **Einloggen:** Ich als Benutzer möchte mich einloggen können, weil ich die Seite nutzen möchte um Getränke zu buchen.

- a) Vorbedingung: Benutzer ist registriert.
- b) Eingabe von Benutzername und Passwort durch den Benutzer.
- c) Benutzer Interagiert mit Web-Oberfläche
- d) Wenn der Benutzername und das Passwort mit einem registriertem Benutzer übereinstimmen => weiterleiten auf Übersichtsseite
- e) Wenn nicht erfolgreich => Ausgabe einer Fehlermeldung

USM3 **Übersicht:** Ich als Trinker möchte eine Übersicht über alle verfügbare Getränke, weil ich sehen möchte, welches Getränk ich kaufen kann.

- a) Vorbedingung: Benutzer ist eingeloggt.
- b) Benutzer sieht alle verfügbaren Getränke.
- c) Benutzer sieht den Preis der verfügbaren Getränke.
- d) Benutzer Interagiert mit Web-Oberfläche

USM4 **Getränk buchen:** Ich als Trinker möchte ein verfügbares Getränk buchen, damit dieses als von mir Getrunken im System hinterlegt wird.

- a) Vorbedingung: Benutzer ist eingeloggt.
- b) Benutzer kann auf ein Getränk klicken.
- c) Benutzer Getränke verfügbar => Das Guthaben des Benutzers verringert sich um den Preis des Getränks.
- d) Getränk nicht verfügbar => Benutzer kann das Getränk nicht drücken bzw. wird durch eine Meldung darauf hingewiesen.

USM5 **Guthaben aufladen:** Ich als Getränkeverwalter möchte das Guthaben der registrierten Mitglieder verändern/auf-laden können, damit sie bei mir für ihre Getränke bezahlen können.

- a) Vorbedingung: Benutzer mit Berechtigung Getränkeverwalter ist eingeloggt.
- b) Getränkeverwalter kann ein Menü zur Guthabenverwaltung öffnen
- c) Getränkeverwalter kann das Guthaben von Trinkern anpassen
- d) Nach der Anpassung des Guthabens ist das veränderte Guthaben beim Betroffenen Trinker sichtbar und verwendbar.

USM6 **Getränkedaten Verwalten:** Ich als Getränkeverwalter möchte einstellen können, welche Getränke zu welchen Preisen verfügbar sind, damit diese von Trinkern gekauft werden können.

- a) Vorbedingung: Benutzer mit Berechtigung Getränkeverwalter ist eingeloggt.
- b) Getränkeverwalter kann ein Menü zur Getränkeverwaltung öffnen
- c) Getränkeverwalter kann Getränke und den Zugehörigen Preis anpassen.
- d) Nach der Anpassung der Getränke sind diese bei Trinkern sichtbar und buchbar.

#### B. SOLL-Anforderungen

USS1 **Getränkeverlauf anzeigen:** Ich als Trinker möchte sehen können, welche Getränke ich wann gekauft habe, weil ich das auch später noch nachvollziehen möchte.

- a) Vorbedingung: Benutzer mit Berechtigung Trinker ist eingeloggt.
- b) Benutzer öffnet ein Menü in dem die Historie angezeigt wird.
- c) Alle von einem Benutzer getrunkenen Getränke werden in einer Liste angezeigt.
- d) Der Benutzer kann den jeweiligen Preis zum Kaufdatum der Getränke in der Historie sehen

USS2 **Bestandsverwaltung:** Ich als Getränkeverwalter möchte den verfügbaren Bestand von Getränken in der Anwendung hinterlegen können, weil ich sehe möchte, wann sie leer werden.

- a) Vorbedingung: Benutzer mit Berechtigung Getränkeverwalter ist eingeloggt.
- b) Benutzer öffnet ein Menü in dem er die Anzahl der verfügbaren Getränke einstellen kann.
- c) Nach speichern/hinzufügen der Anzahl der verfügbaren Getränke wird diese in die Datenbank übernommen.
- d) Wenn Trinker ein Getränk trinken, wird der Bestand des jeweiligen Getränks um 1 reduziert.

USS3 **Umsatzübersicht:** Ich als Getränkeverwalter möchte den Umsatz den jedes Mitglied generiert hat einsehen können, damit ich auswerten kann wer in einem bestimmten Zeitraum am meisten Getrunken hat.

- a) Vorbedingung: Benutzer mit Berechtigung Getränkeverwalter ist eingeloggt.
- b) Benutzer öffnet Menü für den Umsatzverlauf
- c) Benutzer wählt einen Trinker und einen Zeitraum auf, für den er den Umsatz ansehen möchte.
- d) Webinterface zeigt den ausgewählten Umsatz für den ausgewählten Zeitraum an.

#### C. KANN-Anforderungen

USC1 **PayPal Anbindung:** Ich als Trinker möchte mein Guthaben per PayPal aufladen, weil ich nicht gern Bargeld mit mir Rumtrage und den Getränkeverwalter nicht nerven möchte.

- a) Vorbedingung: Benutzer mit Berechtigung Trinker ist eingeloggt.
- b) Benutzer öffnet Menü/Button um per PayPal zu bezahlen.
- c) Benutzer wählt Betrag, der aufgeladen werden soll und bezahlt diesen per PayPal
- d) Der aufgeladene Betrag wird dem Benutzer zugeschrieben und ist auf seinem Konto sichtbar.

#### V. OPTIONAL: NON-FUNCTIONAL REQUIREMENTS

TODO: Löschen???

#### VI. TECHNOLOGIE-STACK

##### A. Datenhaltung

Zur Datenhaltung soll die relationale Datenbank (TODO: PostgreSQL?) genutzt werden.

##### B. Backend

Das Backend soll aus einer REST-API bestehen, welche die benötigten Daten im JSON-Format bereitstellt. Dafür soll Python mit dem Framework (TODO: FastAPI?) verwendet werden.

##### C. Frontend

Für die Darstellung der Daten und die Benutzerinteraktion im Browser soll das TODO: ReactJS? mit dem Framework TODO: Next.js? verwendet werden.

#### *D. Conatinerisierung*

TODO: Um die Anwendung Cloud-Nativ zu gestalten, sollen Front- und Backend in einem Docker-Container laufen. Alle Docker-Container sollen mithilfe von Docker-Compose zu einer Anwendung zusammengefasst werden können.