


# 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**—GetraenkeIO ist eine webbasierte Software zur Verwaltung des Getränkelagers in Vereinen oder anderen Einrichtungen. Sie ermöglicht es registrierten Benutzern, Getränke eigenständig zu buchen, während Administratoren mit erweiterten Rechten Zugriff auf Bestandsstatistiken und Verwaltungsfunktionen haben. Das System unterstützt die effiziente Verwaltung des Lagers.

Technologisch basiert die Anwendung auf einem Python-Backend, das über eine REST-API mit einem React-basierten Frontend kommuniziert. Die Lagerbestände und andere relevante Daten werden in einer relationalen PostgreSQL-Datenbank gespeichert. Docker wird verwendet, um die Anwendung flexibel bereitzustellen und eine einfache Skalierbarkeit zu gewährleisten.

## I. HINTERGRUND UND MOTIVATION

In vielen Vereinsheimen stellt die Getränkeverwaltung nach wie vor eine organisatorische Herausforderung dar. Die Ausgabe und Bezahlung erfolgen häufig auf Vertrauensbasis, wodurch es schwierig ist, den Überblick über Lagerbestände und Einnahmen zu behalten. Auch gängige Hilfsmittel wie Strichlisten oder Excel-Tabellen sind fehleranfällig und erfordern eine regelmäßige, manuelle Pflege.

GetraenkeIO bietet hierfür eine digitale und einfach bedienbare Lösung. Die webbasierte Anwendung stellt eine benutzerfreundliche Oberfläche bereit, über die Vereinsmitglieder selbstständig Getränke auswählen und ihre Käufe erfassen können. Im Hintergrund verwaltet das System automatisch den Lagerbestand, dokumentiert Zahlungen und erstellt grundlegende Statistiken.

Verantwortliche erhalten so einen klaren Überblick über Bestände und Konsumverhalten, während die alltägliche Verwaltung spürbar vereinfacht wird. Als besonderes Merkmal integriert GetraenkeIO alle relevanten Funktionen in einer kompakten Anwendung, die speziell auf den Einsatz in kleinen Organisationen zugeschnitten ist.

## II. 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 trinkt und dieses im System buchen möchte, damit er seine Zeche bezahlen kann.

- 2) **Getränkeverwalter:** Der Getränkeverwalter/Getränkewart des Vereinsheims, der sich um den Getränkenachschub kümmert und regelmäßig das Geld der Trinker eintreibt.

## III. USER STORIES

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 einer Meldung an den Benutzer, dass er erfolgreich registriert wurde.
  - d) Registrierung nicht erfolgreich => Ausgabe einer Fehlermeldung
- USM2 **Einloggen:** Ich als Benutzer möchte mich einloggen können, 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, um zu sehen, welche Getränke ich kaufen kann.
- a) Vorbedingung: Trinker ist eingeloggt.
  - b) Trinker sieht alle verfügbaren Getränke.
  - c) Trinker sieht den Preis der verfügbaren Getränke.
  - d) Trinker 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: Trinker ist eingeloggt.
- b) Trinker kann auf ein Getränk klicken.
- c) Getränk verfügbar => Das Guthaben des Trinkers verringert sich um den Preis des Getränks.
- d) Getränk nicht verfügbar => Trinker kann das Getränk nicht auswählen 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) Trinker öffnet ein Menü in dem die Historie angezeigt wird.
- c) Alle von einem Trinker getrunkenen Getränke werden in einer Liste angezeigt.
- d) Der Trinker 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 sehen möchte, wann sie leer werden.

- a) Vorbedingung: Benutzer mit Berechtigung Getränkeverwalter ist eingeloggt.

- b) Getränkeverwalter 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.

## C. KANN-Anforderungen

USC1 **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) Getränkeverwalter öffnet Menü für den Umsatzverlauf
- c) Getränkeverwalter wählt einen Trinker und einen Zeitraum aus, für den er den Umsatz ansehen möchte.
- d) Webinterface zeigt den ausgewählten Umsatz für den ausgewählten Zeitraum an.

USC2 **PayPal Anbindung:** Ich als Trinker möchte mein Guthaben per PayPal aufladen, weil dies den Aufladeprozess einfacher und schneller gestaltet.

- a) Vorbedingung: Benutzer mit Berechtigung Trinker ist eingeloggt.
- b) Trinker öffnet Menü/Button um per PayPal zu bezahlen.
- c) Trinker 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.

## IV. TECHNOLOGIE-STACK

### A. Datenhaltung

Zur Datenhaltung soll die relationale Datenbank 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 FastAPI verwendet werden.

### C. Frontend

Für die Darstellung der Daten und die Benutzerinteraktion im Browser soll React verwendet werden.

### D. Containerisierung

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.