

## **Hochschule Osnabrück**

University of Applied Sciences

### **Fakultät Ingenieurwissenschaften und Informatik**

Schriftliche Ausarbeitung zum Thema:

#### **CS GO Shop mit Hilfe von Quarkus Qute**

im Rahmen des Moduls  
Software-Architektur – Konzepte und Anwendungen,  
des Studiengangs Informatik-Medieninformatik

Autor: Maxim Zitnikowski  
Matr.-Nr.: 836059  
E-Mail: maxim.zitnikowski@hs-osnabruceck.de

Autor: Mathias Hölz  
Matr.-Nr.: 837095  
E-Mail: mathias.hoelz@hs-osnabruceck.de

Themensteller: Prof. Dr. Rainer Roosmann

Abgabedatum: 15.02.2021

## Inhaltsverzeichnis

<b>1</b>	<b>EINLEITUNG .....</b>	<b>7</b>
1.1	VORSTELLUNG DES THEMAS .....	8
1.2	ZIEL DER AUSARBEITUNG.....	8
1.3	AUFBAU DER HAUSARBEIT .....	9
<b>2</b>	<b>DARSTELLUNG DER GRUNDLAGEN .....</b>	<b>10</b>
2.1	QUARKUS .....	10
2.2	DATENBANK IM CONTAINER.....	11
2.3	QUARKUS QUTE .....	12
2.4	DAS TEAM .....	13
2.5	WERKZEUGE.....	13
<b>3</b>	<b>ANWENDUNG .....</b>	<b>14</b>
3.1	STARTSEITE .....	14
3.2	WAFFE IM DETAIL.....	16
3.3	EINKAUFSWAGEN .....	18
3.4	PROFILSEITE .....	20
3.5	WAFFEN VERKAUFEN.....	22
3.6	TESTEN DER ANWENDUNG.....	23
3.7	REGISTRIEREN.....	23
3.8	LOGIN .....	25
3.9	DISCOUNT SYSTEM .....	27
3.10	QUELLCODE DOKUMENTATION .....	28
3.11	E-MAIL-SYSTEM.....	29
<b>4</b>	<b>ZUSAMMENFASSUNG UND FAZIT.....</b>	<b>30</b>
<b>5</b>	<b>REFERENZEN .....</b>	<b>32</b>

## **Abbildungsverzeichnis**

<i>Abbildung 1: Kanban-Board im EDVSZ-Gitlab .....</i>	13
<i>Abbildung 2: Index-Seite des Shops .....</i>	14
<i>Abbildung 3: DropDown-Menü .....</i>	15
<i>Abbildung 4: Detail-Seite der Waffe .....</i>	16
<i>Abbildung 5: Waffenempfehlungsemail.....</i>	17
<i>Abbildung 6: Einkaufswagen-Seite .....</i>	18
<i>Abbildung 7: Successfull Payment E-mail.....</i>	19
<i>Abbildung 8: Payment Successfull Seite.....</i>	19
<i>Abbildung 9: Not Enough Balance Seite .....</i>	19
<i>Abbildung 10: Profilseite .....</i>	20
<i>Abbildung 11: Kaufbestätigung .....</i>	20
<i>Abbildung 12: Profil Bearbeiten Seite .....</i>	21
<i>Abbildung 13: Verkaufen-Seite .....</i>	22
<i>Abbildung 14: Swagger-UI GUI .....</i>	23
<i>Abbildung 15: Registrier-Seite .....</i>	24
<i>Abbildung 16: Erfolgreiche Registrierung Alert .....</i>	25
<i>Abbildung 17: Login-Seite .....</i>	25

## **Tabellenverzeichnis**

<i>Tabelle 1: Erläuterung der Login-Properties .....</i>	26
--	----

## **Source-Code Verzeichnis**

<i>Snippet 1: Hinzufügen der MySQL Abhängigkeit in der pom.xml-Datei.....</i>	11
<i>Snippet 2: Docker-Compose File .....</i>	11
<i>Snippet 3: Konfiguration der MySQL-Verbindung in der application.properties-Datei ..</i>	12
<i>Snippet 4: Hinzufügen der Qute Abhängigkeit in der pom.xml-Datei.....</i>	12
<i>Snippet 5: Hinzufügen der Qute Abhängigkeit in der pom.xml-Datei.....</i>	16
<i>Snippet 6: Logout Funktion .....</i>	21
<i>Snippet 7: Form Authentifizierung Properties .....</i>	26
<i>Snippet 8: Auszug aus Discount-Klasse .....</i>	27
<i>Snippet 9: Dokumentation einer API-Methode .....</i>	28
<i>Snippet 10: Kurzbeschreibung der Weapon Klasse.....</i>	28
<i>Snippet 11: Application Properties für den Mailer.....</i>	29

## **Abkürzungsverzeichnis**

CS: GO	Counter Strike: Global Offensive
JVM	Java Virtual Machine
API	Application Programming Interface
SMTP	Simple Mail Transfer Protocol
JDBC	Java Database Connectivity
IDE	Integrated Development Environment
CDI	Context and Dependency Injection for the Java EE Plattform
Java EE	Java Enterprise Edition, in der Version 7
ORM	Object-Relational Mapping
SWA	Software-Architektur

## 1 Einleitung

Skins sind einzigartige Waffen-Skins in einem Computerspiel namens Counter Strike: Global Offensive, welches aus dem Genre der Online-Taktik-Shooter kommt. Im Jahr 2013 wurden in einem der Updates Kisten mit Waffen-Skins eingeführt, die seitdem den Spielern für ein erfolgreiches Spiel als Belohnung verteilt werden. Zum Öffnen benötigen die Spieler jedoch Schlüssel, die sie ausschließlich über den Steam-Markt erwerben können. Welche Art von Skins der Spieler bekommt ist unbekannt und die Wahrscheinlichkeit einen seltenen Skin zu ziehen ist ziemlich gering. Eine andere Möglichkeit besteht darin, gebrauchte Skins im Steam-Markt oder bei einem der Drittanbieter zu kaufen [@CSGO].

Skins in CS: GO sind also nichts anderes als 3D-Modelle, die das Aussehen von Standardwaffen ersetzen. Es sollte beachtet werden, dass Skins oft auch als "Waffen-Skins" bezeichnet werden [@CSGO].

Skins bringen somit neue Farben ins Spiel und lassen es neu aussehen. Darüber hinaus sind Waffen-Skins Sammlerstücke, die eine Art In-Game-Wirtschaft schaffen, die Valve und den Skin-Erstellern zusätzliches Einkommen bringt. Neben anderen Merkmalen der neuen Skins sollte beachtet werden, dass sie das Interesse am Spiel erhöhen, was zu einer signifikanten Zunahme des Online-Angebots zum Zeitpunkt der Veröffentlichung neuer Ausrüstungssätze führt [@1MIO].

Jeder CS GO Spieler, der schon mal seine Skins über die Steam Distributionsplattform verkauft hat, kennt, dass bei jeder Transaktion 30% des Verkaufspreises von Steam als Gebühr behalten werden [@30PRO]. Bei geringen Preisen können die Gebühren recht überschaubar sein, allerdings nicht mehr, wenn es um Waffen-Skins geht, deren Preise mehrere zehntausend Euro betragen. An dieser Stelle kommen die Drittanbieter ins Spiel, die eine benutzerfreundlichere Gebührenpolitik anbieten ohne dem Steam-Markt eine seriöse Konkurrenz bieten zu können.

Da die Menschen in der Zeit der Technologien leben, wurde überlegt sich mit dem Problem näher zu befassen und dafür eine geschickte Lösung zu finden, die den Nutzen das Verhandeln erleichtert ohne große Gebühren zu zahlen. Im Rahmen dieser Hausarbeit wird ein CS GO Shop aufgebaut, der als Drittanbieter aufgefasst werden kann. Der Shop bietet seinen Nutzern die Möglichkeit ihre Skins über die Plattform zu kaufen oder zu verkaufen. Außerdem sollen die Nutzer die Möglichkeit haben untereinander die Preise zu verhandeln.

## 1.1 Vorstellung des Themas

Wie bereits im Kapitel zuvor angekündigt wurde, ist das Thema der Hausarbeit „CS GO Shop mit Hilfe von Quarkus Qute“. Ein wichtiger Aspekt des Shops ist das Verhandeln der User. Ein User soll in der Lage sein einen anderen Nutzer nach einem Rabatt zu fragen und nach einer erfolgreichen Verhandlung einen bereitgestellten Coupon-Code einzulösen, um einen Preisnachlass bekommen zu können. Im Mittelpunkt steht aber dennoch das Kaufen und Verkaufen der Waffenskins, die auch nicht angemeldeten Nutzern empfohlen werden können. Das Empfehlen der Waffen wird mit Hilfe von Quarkus unter Nutzung eines SMTP Servers realisiert. Das Senden der Emails kommt an mehreren Stellen zum Einsatz, welches die Erscheinung eines lebenden Systems beim User erweckt. Außerdem werden im Shop vereinfachtes Hibernate ORM mit Panache und die formularbasierte Authentifizierung eingesetzt und erfolgreich verwendet. Zum Realisieren des Front-ends wird Quarkus Qute verwendet, welches das Transportieren bestimmter Instanzen zu Front-end erlaubt. Im Back-End kommt eine mySQL Datenbank zum Einsatz, welche in einem Docker Container realisiert wird.

Außerdem werden in dem Shop weitere kleinere Technologien und Tools eingesetzt, die im Laufe der Dokumentation näher betrachtet und erläutert werden.

## 1.2 Ziel der Ausarbeitung

Das Ziel dieser Arbeit ist es einen Shop für Waffen-Skins aus dem Spiel CS GO zu entwickeln, welches das Handeln mit den Skins ohne große Gebühren ermöglicht. Im Fokus steht nicht die Vielfalt der angebotenen Funktionalitäten, sondern die Benutzerfreundlichkeit bei der Anwendung des CS GO Shops.

Ein weiteres Ziel ist die Realisierung der Benutzerverwaltung, die mithilfe von Panache und MySQL umgesetzt werden soll. Außerdem soll für das Aufsetzen einer MySQL-Datenbank ein Docker Container verwendet werden, der einen Port nach Außen freigeben soll, um eine Verbindung mit der Datenbank zu ermöglichen.

Aus der Sicht des Nutzers ist das Ziel seine Skins möglichst gebührensparend zu verkaufen, wobei die Benutzung der Website dem Nutzer keine Probleme und Unverständlichkeiten bereiten soll. Eine weitere Absicht ist es dem Nutzer die Möglichkeit zu geben sein Guthaben aufzuladen, damit beim Kauf keine Probleme auftreten. Soll es während der Kaufabwicklung zu Problemen kommen, so soll der User über die Ursache des Problems informiert werden.

Das Ziel der Entwickler ist es eine Community aufzubauen, die mit Skins handeln kann und die Skins an Freunde empfehlen kann, um die Reichweite des Shops zu erhöhen. Zum Erreichen dieses Ziels soll das Verhandlungssystem beitragen. So sollen die Nutzer die Möglichkeit haben untereinander die Preise klären zu können.

### 1.3 Aufbau der Hausarbeit

Zu Beginn der Arbeit wird zunächst auf die allgemeinen Grundlagen des CS GO Shops eingegangen und die Systeme, die zur Realisierung des Programms beigetragen haben, werden vorgestellt.

- Zuerst wird der grobe Aufbau der Anwendung vorgestellt.
- Im nächstens Schritt wird vorgestellt, welche Musskriterien der Shop erfüllen muss.
- Im Schritt darauf wird erklärt, wie diese Muskriterien in dem Shop darstellt werden und wie sie umgesetzt worden sind.
- Außerdem wird erzählt, welche Herausforderungen während der Implementierung aufgetreten sind und wie sie gelöst wurden.
- Zu einem wird die Aufteilung der Arbeit im Team angesprochen und die Vorgehensweise bei der Umsetzung wird erläutert.
- Zum anderen soll ein Überblick über das Programm geschaffen werden und die Technologien, die dahinterstehen, sollen erklärt werden.

## 2 Darstellung der Grundlagen

In dem kommenden Abschnitt wird erklärt, auf welchen Technologien der CS-GO-Shop basiert und an welchen Stellen diese eingesetzt werden.

### 2.1 Quarkus

Bei Quarkus geht es um ein Kubernetes-natives Java Framework für Java Virtual Machine (JVM) und die native Kompilierung mit dem vollständigen Stack [@QAURK]. Das Framework ist speziell für Container optimiert worden und soll Java in der Welt der Microservices, Serverless-Apps, Cloud und Containern zu einer der führenden Plattform machen.

Quarkus ist im Frühjahr 2019 von Red Hat unter dem Slogan „Supersonic Subatomic Java“ auf die Bühne der Java Frameworks gebracht worden und nutzt Oracles GraalVM um native Apps zu erstellen. Die Startzeiten der Apps betragen in der Regel Millisekunden. Betrachtet man die Dependency Injection-Lösung von Quarkus, so sieht man, dass diese auf der CDI basiert und ein Framework bietet, womit die Funktionalität und die Konfiguration erweitert werden kann. Die Erweiterungen können in Quarkus ebenso einfach hinzugefügt werden. Alternativ können an dieser Stelle auch die Quarkus Tools verwendet werden, wovon es eine große Vielzahl gibt.

Im Vergleich zu den Konkurrenten bietet Quarkus bis zu 300-mal schnellere Startzeit und benötigt bei der Anwendungsentwicklung nur ein Zehntel des Speichers. Beide der oben genannten Aspekte ermöglichen eine große Reduzierung der Gesamtkosten und helfen dabei diese niedrig zu halten [@QAURK1].

Damit die Anzahl der Nutzer immer steigt, verfolgt Quarkus die Vision, dass die Entwickler bei der Arbeit Spaß haben sollen. Aus diesem Grund hat das Entwickler-Team von Quarkus einen großen Wert daraufgelegt, dass Livecoding, Unified Configuration und Extensions gut funktionieren. Im Entwicklermodus, der mit dem Befehl „mvn compile quarkus: dev“ gestartet werden kann, wird im Quarkus das Livecoding unterstützt, in dem die Änderungen des Entwicklers nach jedem Speichern kompiliert und nach Bedarf dargestellt werden, wenn der Entwickler eine http-Anfrage erstellt [@QAURK2]. Außerdem bietet Quarkus eine Vielzahl der Extensions, die über die „pom.xml“-Datei eingebunden und über die „application.properties“-Datei konfiguriert werden können.

## 2.2 Datenbank im Container

Der CS GO Shop, wie viele andere Projekte, die Daten verwenden, erfordert eine Verbindung zu einer relationalen Datenbank. Die übliche Methode zum Herstellen von Verbindungen zu einer Datenbank besteht darin, eine Datenquelle zu verwenden und einen JDBC-Treiber zu konfigurieren. Im Rahmen dieses Projektes wurde eine MySQL Datenbank, die über die „pom.xml“-Datei als Abhängigkeit eingebunden.

```
<dependency>
    <groupId>io.quarkus</groupId>
    <artifactId>quarkus-jdbc-mysql</artifactId>
</dependency>
```

*Snippet 1: Hinzufügen der MySQL Abhängigkeit in der pom.xml-Datei*

Nachdem die MySQL Abhängigkeit hinzugefügt wurde, muss im nächsten Schritt die Verbindung zu der Datenbank hergestellt werden. Um die Datenbank zu erstellen und nachhinein zu verwenden, wird zuerst MySQL in einem Docker Container installiert. Um einen Docker Container zu erstellen, wird eine „docker-compose.yml“-Datei verwendet, die dafür sorgt, dass alle nötigen Einstellungen vorgenommen werden und der gewünschte Port nach außen freigegeben wird, um die Datenbank von außerhalb des Containers zugänglich zu machen.

Im nächsten Schritt wird das Quarkus Programm mit dem Docker Container verbunden. Dies

```
version: '3.3'

services:
  db:
    image: mysql:latest
    restart: always
    environment:
      MYSQL_ROOT_PASSWORD: root_pw
      MYSQL_DATABASE: mm_skins
      MYSQL_USER: root
      MYSQL_PASSWORD: root
    ports:
      - 3307:3306
```

*Snippet 2: Docker-Compose File*

geschieht durch die „application.properties“-Datei, in der die Informationen zum Aufbau einer Verbindung zu der benötigten Datenbank eingetragen werden.

```
quarkus.datasource.db-kind = mysql  
quarkus.datasource.username = root  
quarkus.datasource.password = root_pw  
quarkus.datasource.jdbc.url = jdbc:mysql://localhost:3307/mm_skins
```

*Snippet 3: Konfiguration der MySQL-Verbindung in der application.properties-Datei*

Nach diesem Schritt ist die Quarkus Anwendung mit der Datenbank verbunden und kann für die Speicherung bestimmter Daten verwendet werden. In diesem Zusammenhang spielt die Datei „import.sql“ keine unwichtige Rolle. Mit Hilfe der Datei können während der Entwicklungsphase beispielhafte Datenpersistiert werden, um den Prozess der Entwicklung zu vereinfachen.

## 2.3 Quarkus Qute

Qute ist eine Template-Engine, die speziell für die Anforderungen von Quarkus entwickelt wurde. Die API kombiniert sowohl den imperativen als auch den nicht blockierenden reaktiven Codierungsstil. Im Entwicklungsmodus werden alle Dateien unter `src/main/resources/templates` auf Änderungen überwacht und permanent aktualisiert [@QTE]. Ähnlich wie bei der MySQL muss auch hier die `quarkus-resteasy-qute` Erweiterung hinzugefügt werden, um Qute in Ihrer JAX-RS-Anwendung verwenden zu können. Dies geschieht ebenfalls in der „`pom.xml`“-Datei.

```
<dependency>  
    <groupId>io.quarkus</groupId>  
    <artifactId>quarkus-resteasy-qute</artifactId>  
</dependency>
```

*Snippet 4: Hinzufügen der Qute Abhängigkeit in der pom.xml-Datei*

Wenn in einem HTML-Template Parameter deklariert werden, versucht Qute, alle Ausdrücke zu validieren, die auf diese Parameter verweisen. Soll während der Erstellung einer der Ausdrücke falsch geschrieben sein oder sogar fehlen, so wird der Entwickler während der Erstellung unverzüglich auf die Konsole über die Fehler informiert.

## 2.4 Das Team

Um einen organisierten Projektablauf von Anfang an zu gewährleisten, wurde vor dem Start des Projektes ein Vorgehensmodell mit einem regelbasierten Vorgehen bestimmt. Da die Erfahrungen der beiden Teammitglieder ausschließlich auf agile Vorgehensmodelle beschränkt sind, standen auch nur diese zur Auswahl. Die Wahl des Vorgehensmodells fiel auf Kanban. Die Verwendung des Kanban-Boards und die regelmäßigen Dailies sorgen für ein strukturierteres Vorgehen und ein eigenverantwortliches Arbeiten.

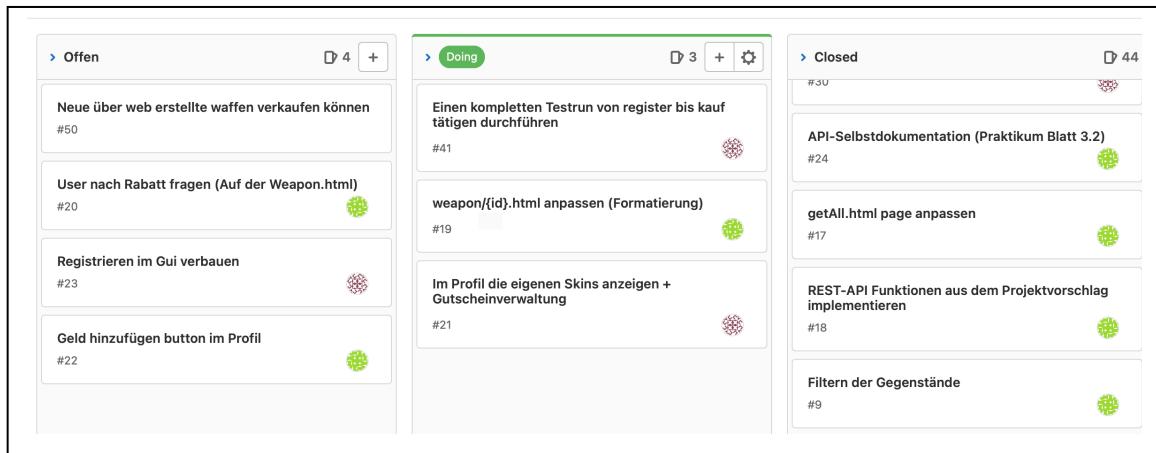


Abbildung 1: Kanban-Board im EDVSZ-Gitlab

## 2.5 Werkzeuge

Zur Versionsverwaltung der Anwendung wird ein verteiltes Versionsverwaltungssystem *Git* zusammen mit *EDVSZ-GitLab* verwendet. Der Vorteil ist dabei, dass jeder der beiden Entwickler auf eine bestimmte Version zurückgreifen kann, falls im Laufe des Projekts Schwierigkeiten bei der Entwicklung entstehen [@GIT].

Die Wahl der IDE fiel auf *Visual Studio Code*, worin das verteilte Versionsverwaltungssystem Git integriert werden kann, um den Workflow der Teammitglieder zu optimieren und die Nutzung von Git möglichst unproblematisch zu machen.

Für die täglichen Dailies wurde ein Besprechungstool *Microsoft Teams* genutzt, welches zum einen ein Treffen der Mitglieder zur Besprechung von abgeschlossen Tickets ermöglichte und zum anderen das Klären der offenen Fragen zum weiteren Entwicklungsverlauf der Anwendung erleichterte.

### 3 Anwendung

#### 3.1 Startseite

Die Anforderungen an den CS GO Shop setzen sich aus mehreren Punkten zusammen, die erfüllt werden müssen. Der Nutzer soll nicht nur die Möglichkeit haben sich die Skins anzugecken und zum Warenkorb hinzuzufügen, sondern auch diese zu bezahlen. Nach dem erfolgreichen Kauf soll der Nutzer über seine neuen Waffen-Skins per E-Mail informiert werden. Außerdem soll jeder unangemeldete Nutzer die Skins seinen Freunden empfehlen können. Darüber hinaus sollen die Nutzer die Möglichkeit haben zu verhandeln, um von dem Verkäufer ein Rabatt zu bekommen.

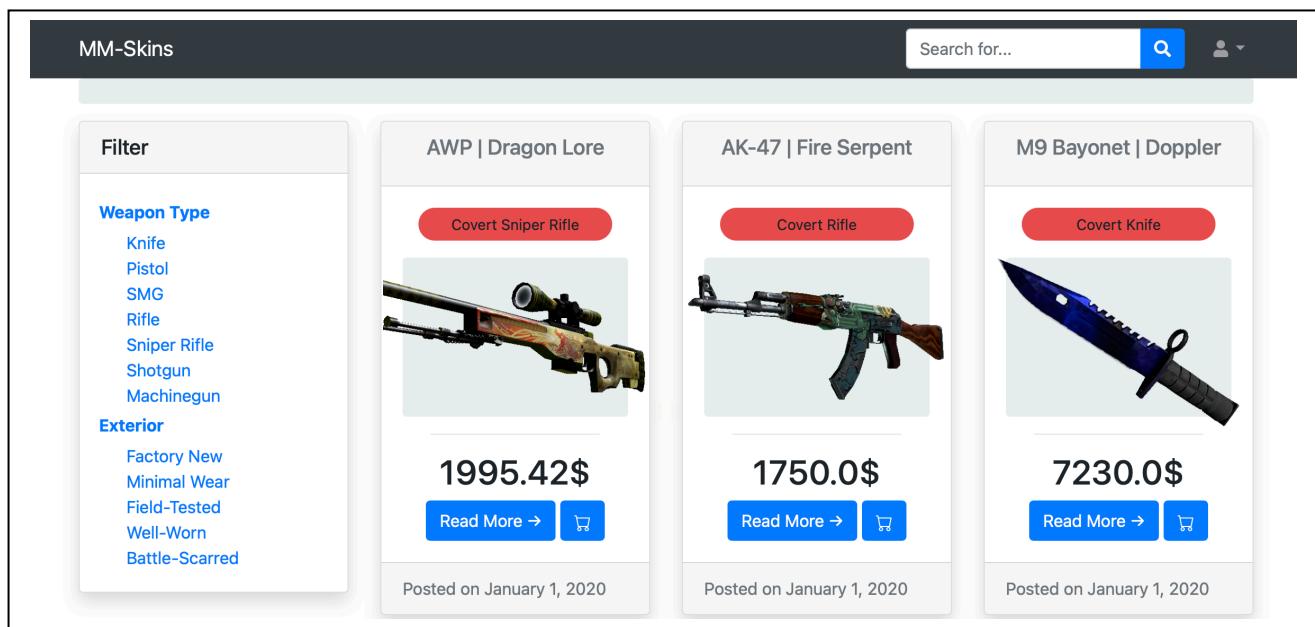


Abbildung 2: Index-Seite des Shops

Die Anwendung heißt den User mit einer übersichtlichen Startseite, auf der die Waffenskins reihenweise untereinander aufgelistet werden, willkommen. Am linken Seitenrand ist ein Filter-Div, welches eine Filterung nach dem Weapon Type und der Exterior, die in dem deutschsprachigen Raum unter dem Begriff „Abnutzung“ bekannt ist, ermöglicht. Wählt der User zum Beispiel „Knife“ als Filterkriterium, so werden alle Messer, die im Angebot sind, aufgelistet und somit wird die Suche nach einem bestimmten Waffen Typen erleichtert. Außerdem hat der Nutzer die Möglichkeit nach Skins zu suchen. Dies wurde mit Hilfe von Keyword-Matching erreicht, so kann der User im Eingabefeld in der oberen rechten Ecke einen Begriff, der exakt im Angebot vorkommt, eingeben und bekommt alle Waffen, die diesen Begriff in ihrem Namen beinhalten, zurück. Die gefundenen Waffen werden wiederum auf einer Seite aufgelistet, die einen ähnlichen Aufbau wie die Home-Seite des Shops hat. Ergibt die Suche keine Ergebnisse

so wird der User selbstverständlich darüber informiert, dass zu seinem Keyword keine Skins gefunden wurden.

In der rechten oberen Ecke sieht der Nutzer ein Männchen-Icon, welches in seinem inneren ein Dropdown-Menü und gleichzeitig eine Navigationskomponente der obersten Ebene ist. Das Dropdown-Menü stellt eine Verbindungskomponente zwischen der Indexseite und der

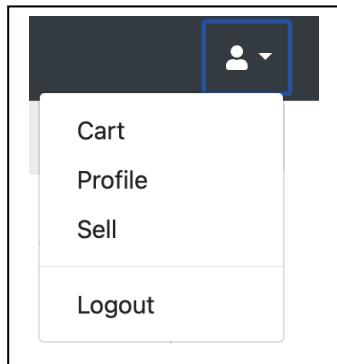


Abbildung 3: DropDown-Menü

Nutzerverwaltung dar. Zum einen gibt es eine Möglichkeit zum Einkaufswagen und der Profilseite zur gelangen. Und zum anderen kann der User seine eigenen Waffen zum Verkauf anbieten. Auf die Funktionalität von jedem einzelnen Punkt des Dropdown-Menüs wird im weiteren Verlauf der Dokumentation noch genauer eingegangen.

Betrachtet man die Dragon Lore aus der Abbildung 2, so fällt einem auf, dass jeder Skin einen Namen z.B. „AWP | Dragon Lore“, eine Art Seltenheit: „Covert Sniper Rifle“, ein Bild und einen Preis besitzt. Anhand von diesen Merkmalen kann ein erfahrener CS GO Trader leicht einschätzen, ob der Skin das verlangte Geld wert ist. Sind die Informationen nicht ausreichend, so besteht die Möglichkeit sich die Waffe genauer anzusehen, indem man auf den „Read More“-Button klickt. Außerdem kann man die Waffe gleich dem Einkaufswagen hinzufügen. Bei der Seltenheit der Waffe gibt es eine Besonderheit, abgesehen davon, dass je seltener eine Waffe ist, desto höher der Preis ist, hat jede Art der Seltenheit eine bestimmte Farbe, die dem User eine grobe Einschätzung des Preises der Waffe verschaffen soll. Hierbei sei aber gesagt, dass es die „Contraband Rifle“ mit dem Namen „M4A4 | Howl“ (Abbildung 4) als Waffentyp nur einmal in Spiel gibt, dies ist dem gestohlenen Waffendesign, welches Valve zur Veröffentlichung einer eigenen Seltenheitsstufe für gestohlene Designs „Contraband“ nötigte, geschuldet.

### 3.2 Waffe im Detail

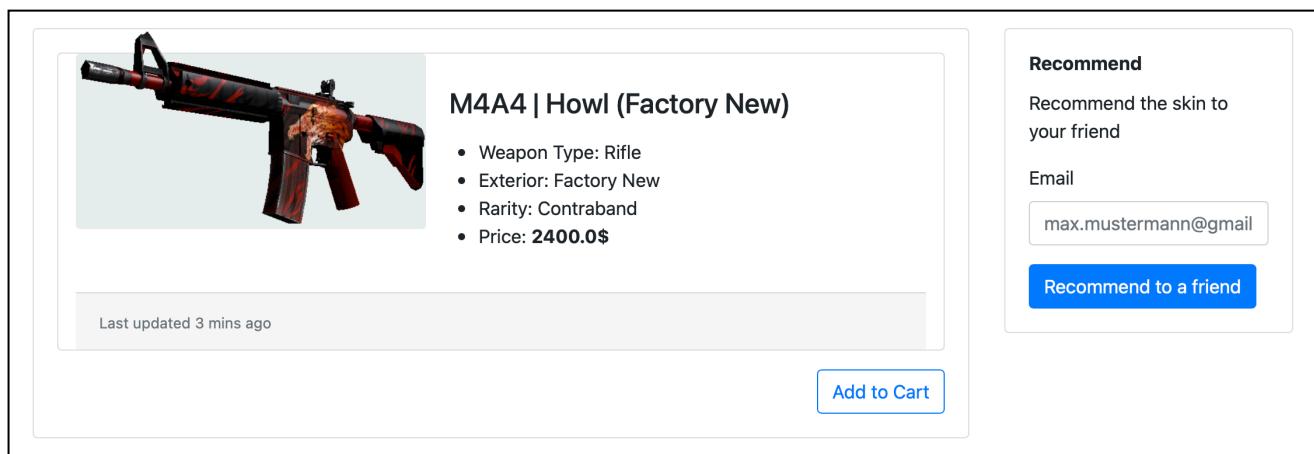


Abbildung 4: Detail-Seite der Waffe

Jede Waffe im Spiel CS GO besitzt viele Eigenschaften, die sie einzigartig machen. So wird am Beispiel der „M4A4 | Howl“, dessen Name aus dem `weapon_type` und dem `weapon_name` zusammengesetzt wird, gezeigt welche Informationen zu jeder einzelnen Waffe preisgegeben werden und wie die zugehörige Klasse definiert ist.

```
@Entity
public class Weapon extends PanacheEntity{
    public String weapon_name;
    public String icon_url;
    public String weapon_type;
    public String gun_type;
    public String exterior;
    public String rarity;
    public String rarity_color;
    public float price;
    ...
}
```

Snippet 5: Hinzufügen der Qute Abhängigkeit in der pom.xml-Datei

Aus der Übersicht der Waffe, besteht ebenfalls die Möglichkeit, die Waffe zum Warenkorb hinzuzufügen. An dieser Stelle wird überprüft, ob der User angemeldet ist, ist das nicht der Fall, so wird der User aufgefordert sich anzumelden. Hat die Authentifizierung bereits stattge-

funden und der User wurde erfolgreich identifiziert, so kann die Waffe zum Warenkorb hinzugefügt werden und der Nutzer wird darüber informiert. Es besteht keine Möglichkeit dieselbe Waffe mehrmals in den Einkaufswagen reinzutun.

Darüber hinaus besteht die Option eine Waffe einem Freund zu empfehlen, dazu ist die Card an rechten Seitenrand vorgesehen (Abbildung 4). Gibt der User eine gültige E-Mail-Adresse ein, so wird an die angegebene Adresse eine von den Entwicklern vordefinierte EmailTemplate geschickt. Das Verschicken der Emails wird durch die „quarkus-mailer“ Erweiterung möglich und erlaubt mit wenig Aufwand Emails über Quarkus zu versenden. Nach dem Installieren der Erweiterung muss eine E-Mail-Ressource angelegt werden, in der die „MailTemplateInstances“ vordefiniert werden. In der E-Mail wird ebenfalls das Quarkus-Qute-Core eingesetzt, um alle nutzerrelevanten Daten darzustellen. Die E-Mail, die der Empfänger zu sehen bekommt, sieht wie folgt aus (Abbildung 5). Die E-Mail enthält ein Button, womit der Empfänger nur mit einem Mausklick gleich zu der Seite der empfohlenen Waffe gelangen kann.

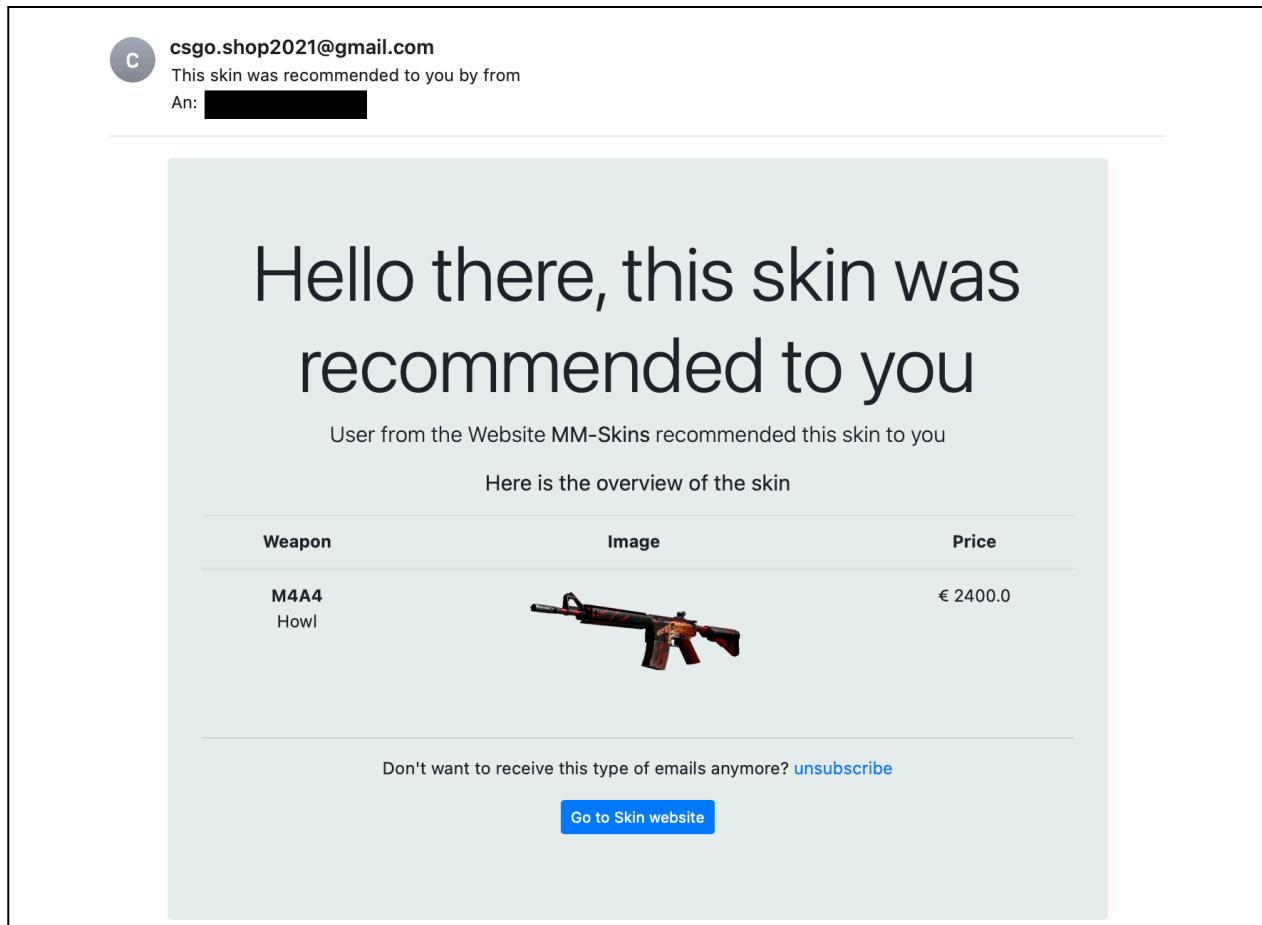


Abbildung 5: Waffenempfehlungsemail

### 3.3 Einkaufswagen

Nachdem eine Waffe erfolgreich zu dem Einkaufswagen hinzugefügt wurde, stehen dem angemeldeten User einige Funktionen zur Verfügung (Abbildung 6). Zuerst sieht man, dass die wichtigsten Informationen zu der Waffe weiterhin angezeigt werden, damit die eindeutige Zuweisung der Waffen bestehen bleibt und der User nicht durch unnötigen Text verwirrt wird. Neben dem Löschen der Waffe aus dem Warenkorb, besteht auch die Möglichkeit den Einkaufswagen mit einem Klick komplett zu entleeren. Nach dem Entleeren des Einkaufwagen wird dem Nutzer die entsprechende Meldung angezeigt, dass er noch keine Waffen im Einkaufswagen hat und nur mit einem Klick zu den auf der Seite angebotenen Waffen navigiert werden kann.

#	Image	Weapon	Ask for Discount	Coupon	Price	Action
•		M4A4 Howl	<a href="#">ask owner</a>	No coupon available	€ 2400.0	

Total    € 2400.0

[Continue Shopping](#) [Complete Order](#)

Abbildung 6: Einkaufswagen-Seite

Jeder angemeldete Nutzer kann den Verkäufer nach einem Rabatt fragen. Klickt der Nutzer auf den „ask owner“-Button, so wird der Verkäufer per E-Mail benachrichtigt, der sich wiederum mit dem potenziellen Kunden in Verbindung setzen kann, um den Preis zu verhandeln. Im inneren der Funktionalität wird auch an dieser Stelle die „quarkus-mailer“ Erweiterung verwendet.

Haben sich der Verkäufer und der potentielle Käufer auf einen Preis geeinigt, so erstellt der Verkäufer in seinem Profil einen Gutschein-Code für diejenige Waffe und teilt diesen dem Käufer mit. Nachdem ein Gutschein-Code zu einer Waffe erstellt wurde, wird die Möglichkeit der Eingabe des Gutschein-Codes im Einkaufswagen des Käufers freigegeben und dieser kann den Code eingeben, um den vereinbarten Rabatt zu bekommen. Der Preis der Waffe wird dementsprechend angepasst und die Bestellung kann zu einem niedrigeren Preis abgeschlossen werden. Abhängig davon, ob der Nutzer genug Guthaben hat, wird dementsprechend eine

Seite angezeigt, die entweder den Nutzer über den erfolgreichen Kauf (Abbildung 8) oder ihn darüber informiert, dass das zur Verfügung stehende Guthaben nicht ausreicht, um die Bestellung abzuschließen (Abbildung 9).

Im Falle einer erfolgreichen Bestellung wird der Käufer über einen erfolgreichen Kauf per E-Mail informiert (Abbildung 7) und die Waffe dem persönlichen Inventar des Käufers hinzugefügt, so dass sie erneut zum Verkauf angeboten werden kann und der jetzige Käufer wiederum die Rolle des Verkäufers einnehmen kann.

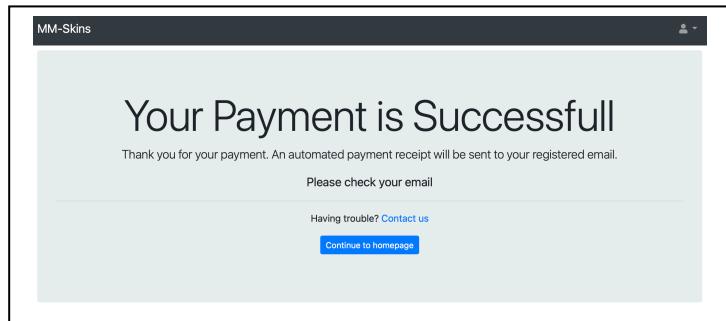


Abbildung 8: *Payment Successfull Seite*

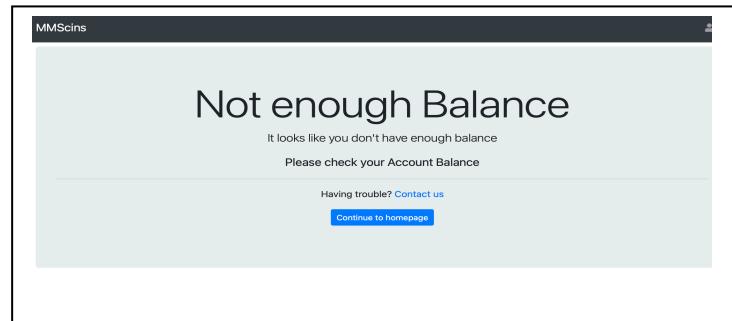


Abbildung 9: *Not Enough Balance Seite*

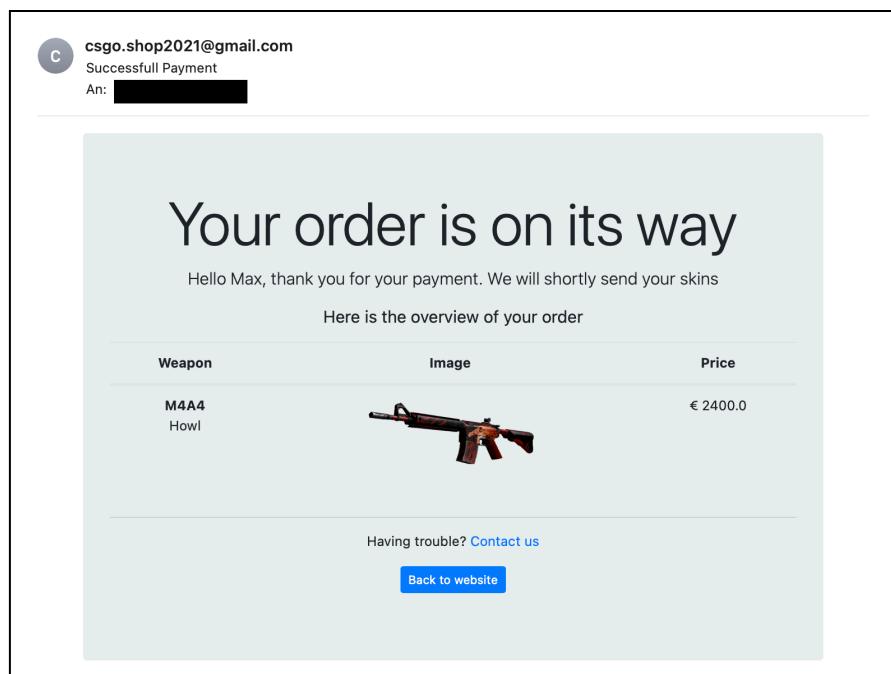


Abbildung 7: *Successfull Payment E-mail*

### 3.4 Profilseite

Im Profil werden persönliche Informationen angezeigt, die bearbeitet werden können. Wie im letzten Kapitel bereits erwähnt, werden die Waffen, die einem Nutzer gehören, in seinem Profil angezeigt. An dieser Stelle hat der Verkäufer die Möglichkeit einen Gutschein-Code zu erstellen falls er nach einem Rabatt von einem potentiellen Käufer gefragt wird. Ist so ein Code erstellt worden, so wird in der Spalte „Coupon“ die Codenummer und der dazugehörige Wert angezeigt.

The screenshot shows the 'MM-Skins' application's profile page. At the top, there is a navigation bar with 'MM-Skins' and a user icon. Below it, a breadcrumb navigation shows 'Home / User'. On the left, there is a user profile card for 'Max Mustermann' featuring a cartoon character icon, the name 'Max Mustermann', and a balance of '60.0 \$'. A blue button labeled 'Add Funds' is visible. To the right, a table displays personal information: Full Name (Max Mustermann), Email (csgo.shop2021@gmail.com), Username (kunde), and Balance (\$60.0). A blue 'Edit Profile' button is located at the bottom of this section. Below this, another table shows a purchase history with columns for Product, Description, Coupon, and Price. One entry shows an M4A4 Howl skin with a price of '\$ 2400.0' and a coupon input field containing 'Value' with a 'Create coupon' button below it. The entire interface has a clean, modern design with a light gray background and blue accents for buttons.

Abbildung 10: Profilseite

Hat der Nutzer nicht ausreichend Geld, um einen gewünschten Waffenskin zu kaufen, so kann das Guthaben im Profil aufgeladen werden. Dazu ist der Button „Add Funds“ vorgesehen. Klick der Nutzer auf den Button, wird ein Alert aufgerufen, in dem die Summe zum Hinzufügen eingegeben werden kann. Nach dem Bestätigen der Eingabe wird der Nutzer zu einer weiteren Seite navigiert, auf der der Kauf bestätigt werden muss (Abbildung 11).

The screenshot shows a 'PayPal' summary page. At the top, there is a header with the PayPal logo. Below it, a section titled 'Summary' displays the product as 'Add Funds' and the price as '\$250.0'. A note states: 'The safety of our users is very important to us, therefore it is only possible to add funds to your balance via paypal. If you've confirmed the amount of money to add to your balance press the "Purchase via PayPal" button to proceed.' At the bottom, there is a green button labeled 'Purchase via PayPal' with a small icon. The overall layout is simple and functional, typical of a payment confirmation screen.

Abbildung 11: Kaufbestätigung

Nach der Bestätigung wird der User automatisch auf die Profilseite zurückgeführt. Möchte man seine Profilinformationen anpassen, so drückt man auf den „Edit Profile“-Button.

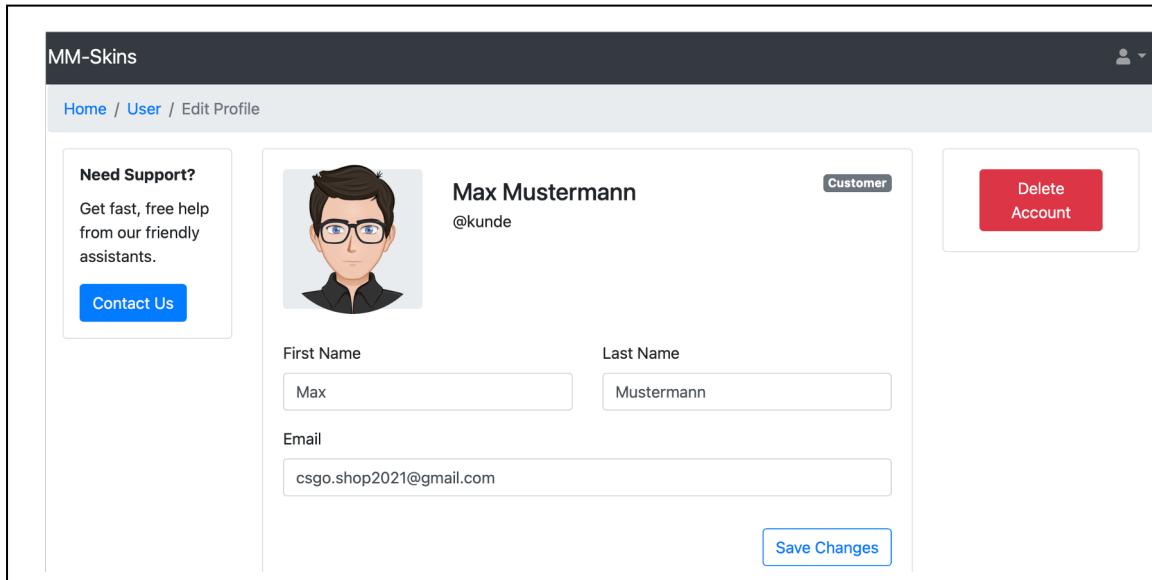


Abbildung 12: Profil Bearbeiten Seite

Hier kann der Nutzer seinen Vor- und Nachnamen anpassen, außerdem kann die E-Mail-Adresse geändert werden. Nach dem Bearbeiten müssen die Änderungen gespeichert werden, damit sie im System eingetragen werden können. Aus der Profil-Bearbeiten-Seite hat der Nutzer nicht nur die Möglichkeit sein Profil zu bearbeiten, sondern sogar komplett zu löschen. Dies wird mit dem „Delete Account“-Button realisiert, welcher an dem rechten oberen Rand positioniert ist. Nach dem Löschen wird der User aus der Datenbank und seine Session im Browser gelöscht. Zum Löschen der Session wird eine JavaScript Funktion verwendet.

```
const LOGGED_COOKIE = "quarkus-credential";
function logOut() {
    //console.log("logging out")
    document.cookie = LOGGED_COOKIE + '=; Max-Age=0; path=/';
    //console.log(document.cookie)
};
```

Snippet 6: Logout Funktion

### 3.5 Waffen verkaufen

In dem CS GO Shop kann jeder angemeldete User seine Skins zum Verkauf auf der Website anbieten. Dazu ist die Sell-Seite vorgesehen.

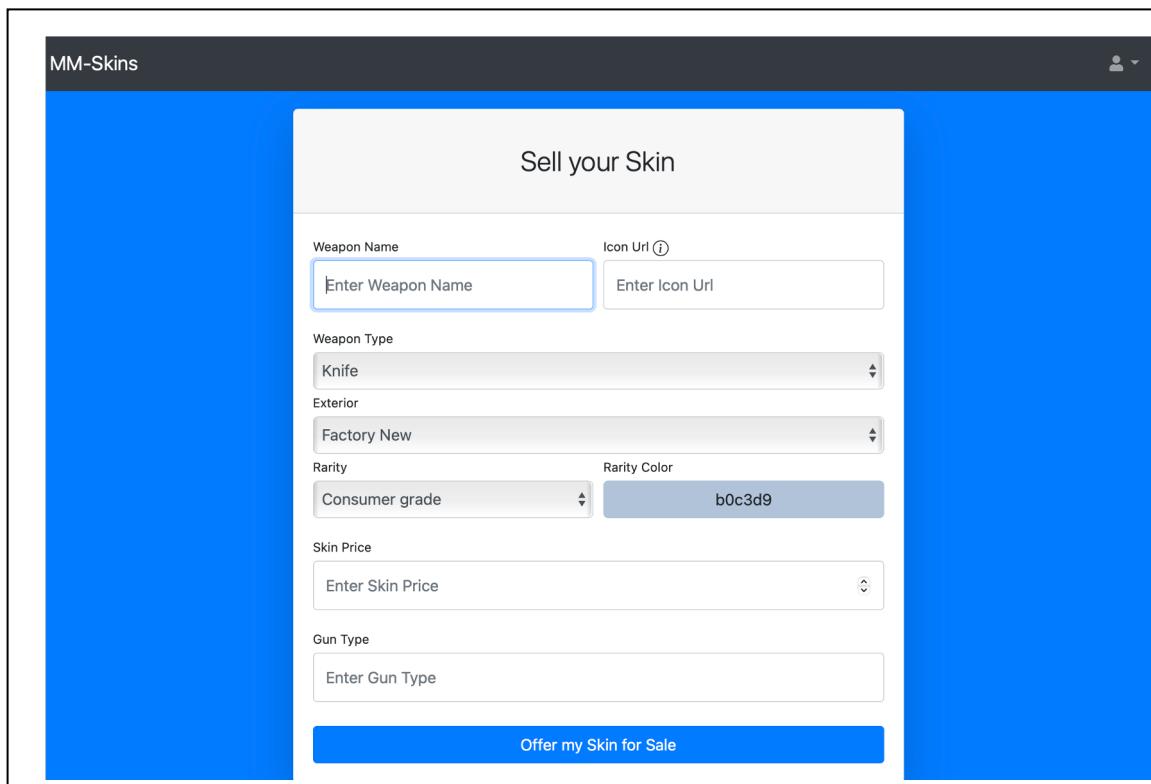


Abbildung 13: Verkaufen-Seite

Hier kann man alle benötigten Informationen zu der Waffe eingeben, um sie zum Verkauf anbieten zu können. Die Images, die auf der Seite benutzt werden, kommen von der Seite „<https://steamcommunity-a.akamaihd.net/economy/image/>“, da aber der Pfad nicht vollständig ist, muss noch die Url des Bildes eingetragen werden, die aus der API der Seite „<http://csgobackpack.net/api/GetItemsList/v2/>“ kopiert werden kann. Man müsste nach dem Feld „icon\_url\_large“ von einer benötigten Waffe suchen und in dem Icon-Url-Input-Feld einsetzen. Diese Information wird ebenfalls angezeigt, wenn der User über dem I-Button auf der Sell-Seite mit der Maus fährt. Die Rarity Color ist von der gewählten Rarity abhängig und wird nach dem Ändern der Rarity angepasst.

Die Seite ist nicht perfekt gestaltet worden und dadurch werden sinnlose Eingaben von Usern möglich. Das Problem könnte man lösen, indem man alle Informationen zu einer Waffe direkt aus dem Steam-Account ausliest. Allerdings wäre das Verbinden des Shops und Erstellung eines Steam-Bots mit einem extremen Aufwand verbunden, was die Rahmen der Hausarbeit sprengen würde.

### 3.6 Testen der Anwendung

Das Testen der Anwendung wurde über eine benutzerfreundliche Benutzeroberfläche mit dem Namen Swagger UI getestet, die ebenfalls als Quarkus Erweiterung installiert werden kann. Außerdem wird eine „*smallrye-openapi*“-Erweiterung verwendet, um die API OpenAPI v3-Spezifikation zu generieren. Um OpenAPI zu konfigurieren wurde die Klasse SwaggerConfig erstellt, die die OpenAPIDefinition beinhaltet. Gibt man im Browser „<http://localhost:9090/openapi>“ ein, so wird die OpenAPI v3-Spezifikation heruntergeladen. Die REST-API ist nach den Ressourcen aufgeteilt, um eine möglichst gute Übersicht zu gewährleisten. Die Methoden, die nur bestimmten Rollen zur Verfügung stehen, sind mit einem Schloss gegenzeichnet und können nur von diesen Rollen aufgerufen werden.

The screenshot shows the Swagger UI interface for a User API. At the top, it displays the title "User API with Quarkus by MM-Skins" with version 1.1.2 and OAS3 indicators. Below the title, there's a navigation bar with links for "/openapi", "Maxim Zitnikowski, Mathias Hölz - Website", "Send email to Maxim Zitnikowski, Mathias Hölz", and "MI". On the right side of the header, there's a green "Authorize" button with a lock icon. The main content area is organized into sections: "weapon" (All the weapon methods), "user" (All the user methods), "user profile" (All the user profile methods), "email" (All the user email methods), and "discount" (All the discount methods). Each section has a description and a right-pointing arrow indicating more details. The entire interface has a clean, modern design with a light blue background and white text.

Abbildung 14: Swagger-UI GUI

### 3.7 Registrieren

Damit sich neue Anwender in dem Shop registrieren können, wurde ebenfalls eine Seite dafür eingerichtet. Die Seite umfasst ein modernes Design, welche einem neuen User möglichst wenig Informationen abverlangt.

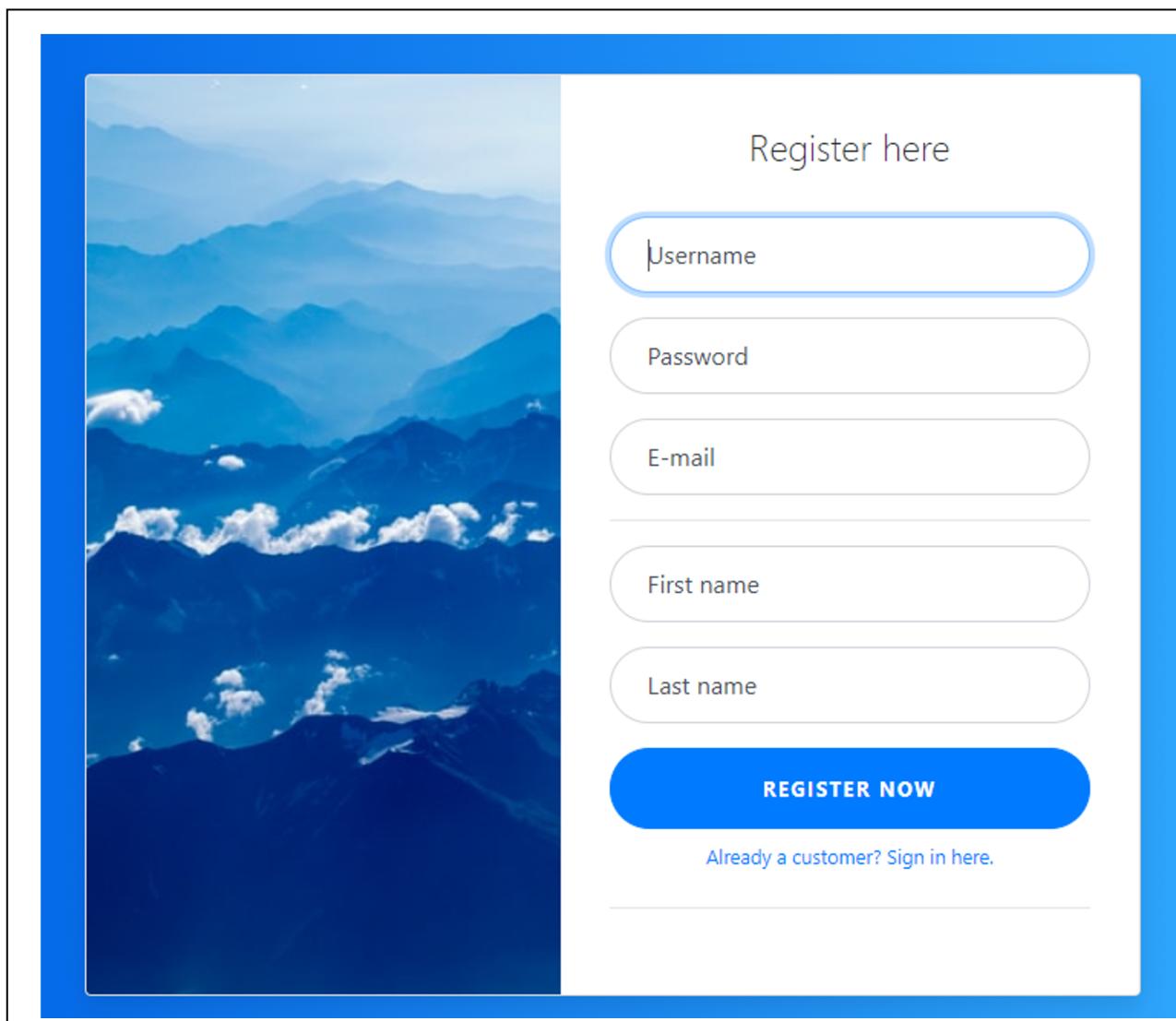


Abbildung 15: Registrier-Seite

Abgesehen von dem hauptsächlich benötigten Daten wie Username, Passwort und E-Mail-Adresse, werden noch der Vor- und Nachname eines Nutzers abgespeichert. Dadurch wird der Registrierungsprozess einfach und unkompliziert gehalten. Für den Fall, dass man sich bei der Auswahl der Login und Registrier-Seite vertan hat, besteht die Möglichkeit über einen Link auf die Login-Seite zu wechseln.

Bei einer erfolgreichen Registrierung wird der User über ein Alert informiert, und zurück auf die Index Seite geleitet.

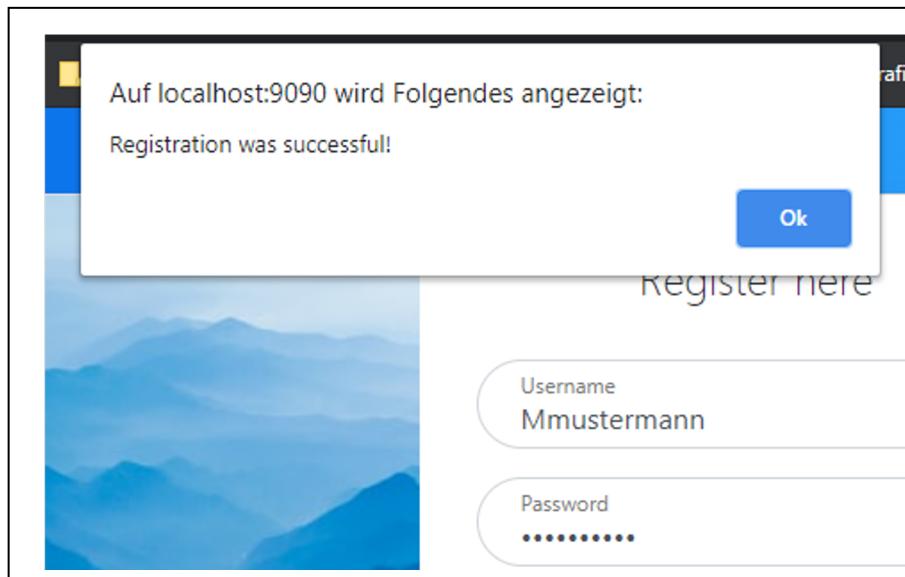


Abbildung 16: Erfolgreiche Registrierung Alert

Der Registrierungsprozess wurde nicht optimiert, aufgrund dessen ist es möglich, dass sich ein User zwei Mal den exakt gleichen Account erstellen könnte, da keine Abfrage auf bestehende E-Mail-Adressen und Usernamen angewendet wird. Auch bei der E-Mail-Adresse wurde auf eine gesonderte Fehlerüberprüfung verzichtet.

### 3.8 Login

Genauso wie die Seite zum Registrieren ist auch die Login-Seite modern und minimalistisch gehalten.

A screenshot of a login form titled "Login". It features two input fields: "Username" and "Password", both with placeholder text "Enter username" and "Enter password" respectively. Below the inputs is a blue "Login" button. At the bottom of the form, there is a link "Not a customer? Register here!".

Abbildung 17: Login-Seite

Sollte ein Anwender unbeabsichtigt auf der Login-Seite landen, besteht die Möglichkeit sich unten über den Link zum Formular der Registrierung zu begeben. Für den Login-Prozess werden, wie man es standardmäßig im Netz erwarten würde, der Username und das Passwort benötigt. Im Gegensatz zu anderen Webseiten besteht jedoch keine Möglichkeit, die Login-Daten über den Browser zu speichern oder das Passwort über eine angelegte E-Mail-Adresse zurückzusetzen.

Der Login verwendet die von Quarkus angebotene „Form Based Authentication“ nach der Anleitung von [@\[QAURK3\]](#). Zur Konfiguration wird eine Modifizierung der Applikation Properties benötigt. Im Folgenden werden die einzelnen Properties welche zum Einsatz kamen vorgestellt und erläutert.

```
# HTTP Form Based Authentication Settings
quarkus.http.auth.basic=true
quarkus.http.auth.form.enabled = true
quarkus.http.auth.form.redirect-after-login= true
quarkus.http.auth.form.login-page = login.html
quarkus.http.auth.form.error-page = error.html
quarkus.http.auth.form.timeout = PT45M
quarkus.http.auth.form.cookie-name = quarkus-credential
```

*Snippet 7: Form Authentifizierung Properties*

<b>Quarkus.http.auth.form.enabled</b>	Ein boolean Wert, welcher vorgibt, ob die Form Authentifizierung aktiviert ist.
<b>Quarkus.http.auth.form.redirect-after-login</b>	Boolean Wert, der ermöglicht den User nach erfolgreichem Login auf eine Landingpage zu verweisen.
<b>Quarkus.http.auth.form.login-page</b>	Die vorgegebene login-Webseite.
<b>Quarkus.http.auth.form.error-page</b>	Fehlerseite die im Fall eines erfolglosen Logins angezeigt wird.
<b>Quarkus.http.auth.form.timeout</b>	Ein Timer der festlegt, nach wie langer Inaktivität der bestehende Session-Cookie nicht erneuert wird, und dadurch der Nutzer abgemeldet wird.
<b>Quarkus.http.auth.form.cookie-name</b>	Der Name des Cookies, welcher die Session verwaltet.

*Tabelle 1: Erläuterung der Login-Properties*

Wie sich durch die vorgestellten Application Properties bereits erahnen lässt, verwaltet Quarkus authentifizierte Nutzer nicht in einer HTTP-Session, sondern in einem Cookie. Um also ein Logout zu vollziehen, muss der Session-Timer des Cookies auf *null* gesetzt werden.

### 3.9 Discount System

Da User im Shop die Chance erhalten sollen, miteinander zu verhandeln, wurde ein System eingeführt, welches ermöglicht Rabatte für eine Angebotene Waffe zu erstellen. Die Discount Klasse setzt sich aus drei Komponenten zusammen. Zum einen wäre da der *boolean* valid, welcher bestimmt, ob der Code bereits eingesetzt wurde oder nicht. Die anderen beiden Bestandteile sind der Hashcode welcher verwendet wird, wenn ein User den Code anwenden will, um seinen Rabatt einzulösen, und der eigentliche Wert des Gutscheins.

```
@Entity
@Table(name = "DISCOUNT")
public class Discount extends PanacheEntity {

    /** Whether the code has already been used or not. */
    public boolean valid;
    /** The hashcode to use the code. */
    public int code;
    /** The value which will be subtracted from a weapons price. */
    public float value;

    ...
}
```

Snippet 8: Auszug aus Discount-Klasse

Die größte Schwierigkeit beim Verwalten der Gutscheine, ist zu verhindern, dass mehrere oder unbestimmte Benutzer den Rabatt eines anderen einlösen. Ursprünglich war dafür der *boolean* valid gedacht, welcher verhindern soll, dass der Code öfter als einmal eingesetzt werden kann. Dies begrenzt zwar die Nutzung eines Gutscheines auf einmal, da aber die Gutscheine nur den Waffen zugewiesen werden, ist eine Waffe nach Verbrauch eines Gutscheines im Endeffekt für alle anderen Nutzer auch vergünstigt. Entschließt sich also ein Kunde eine Waffe doch nicht zu kaufen, direkt nachdem er seinen Gutschein eingelöst hat, so ist die Waffe für jemand anderen, der sie seinem Warenkorb hinzufügt, zum gleichen Preis erhältlich. Für diesen Sonderfall besteht aber für einen Verkäufer immer noch die Möglichkeit den bestehenden Rabattcoupon zu überschreiben, und den dadurch existierenden Rabattcode unwirksam zu machen. Die sicherste Variante wäre jedoch eine Beziehung zwischen Nutzer und Gutscheinen zu erstellen, damit wirklich nur der Nutzer für den ein Gutschein erstellt werden soll, diesen auch anwenden kann. Dies hätte das System jedoch weiter verkompliziert und noch mehr Zeit gekostet, weshalb die Entscheidung auf eine etwas leichtere wenn auch unsicherere Variante fiel.

### 3.10 Quellcode Dokumentation

Zur Dokumentation des Quellcodes wird JavaDoc verwendet. Laut [@JDOC] wird die Dokumentierung von Software öfter außer Acht gelassen. Damit das System diesen Fehler nicht begeht, wurde auf eine saubere Dokumentation aller Quellcode Dateien geachtet. Über den Klassen befindet sich eine kurze Beschreibung, was sie enthalten und wofür sie benötigt werden. Ein Teil dieser Beschreibung ist ebenfalls der Autor der Klasse.

```
/**  
 * This class models a weapon which can be sold at our shop  
 * @author Maxim Zitnikowski  
 */
```

Snippet 9: Kurzbeschreibung der Weapon Klasse

Fortgesetzt wird die Dokumentation, indem die einzelnen Variablen mit einem kurzen Satz erklärt werden. Zum Abschluss werden die Konstruktoren und Methoden nach dem gleichen Schema kommentiert. Eine kurze Beschreibung der Methode oder des Konstruktors, die enthaltenen Parameter mit einer Erläuterung wofür die Parameter eingesetzt werden und falls enthalten, den Wert, den die Methode zurückgibt.

```
/**  
 * API-method: Creates a new discount and  
 * assigns it to the weapon with the designated id  
 *  
 * @param id The id of the discount you're looking for  
 * @param value The value of the discount  
 *  
 * @return True when the function succeeded, false when it failed  
 * because the weapon wasn't found.  
 */
```

Snippet 10: Dokumentation einer API-Methode

Durch diese Art der Dokumentation wird der gesamte Aufbau der Quelldateien erläutert und dürfte keinen Raum für Fragen entstehen lassen.

### 3.11 E-Mail-System

Damit User miteinander verhandeln können, ist ein E-Mail-System in der Anwendung eingebunden. Gesteuert wird das System über den im Quarkus verbauten Mailer dessen Aufbau in [@QAURK4] erläutert wird. Konfiguriert wird der Mailer genau-so wie der Login Prozess über die Application Properties.

```
# Configuration for quarkus mailer
quarkus.mailer.auth-methods=DIGEST-MD5 CRAM-SHA256 CRAM-SHA1 CRAM-MD5 PLAIN
LOGIN
quarkus.mailer.from = csgo.shop2021@gmail.com
quarkus.mailer.host = smtp.gmail.com
quarkus.mailer.port=465
quarkus.mailer.ssl=true
quarkus.mailer.username = csgo.shop2021@gmail.com
quarkus.mailer.password = odacxjidsmajorxp
quarkus.mailer.mock=false
```

*Snippet 11: Application Properties für den Mailer*

Versendet werden E-Mails über den Gmail SMTP Server. Um dies zu ermöglichen, wurde ein E-Mail-Account speziell für den Shop angelegt. Danach muss in dem Account ein dediziertes Passwort für den Quarkus Mailer erstellt werden, welches dann unter *quarkus.mailer.password* hinterlegt wird (Snippet 9). Nach der erfolgreichen Konfiguration des Quarkus Mailers und des Gmail SMTP Servers, können E-Mails angenehm über API-Methoden versendet werden. Dabei hat man mehrere Möglichkeiten. Zum einen die Variante, bei der einfach nur Text als E-Mail-Inhalt versendet wird. Die andere ist dabei wesentlich interessanter. Zusätzlich zu gewöhnlichen E-Mails können auch solche erstellt werden, welche HTML-Seiten auf Basis von Qute Templates darstellen.

## 4 Zusammenfassung und Fazit

In dieser Hausarbeit wurden eine API, sowie eine Webapplikation mit Quarkus entwickelt, welche einen Onlineshop für das Handeln mit Skins aus dem Spiel Counter Strike: Global Offensive ermöglichen soll. Über die API können sämtliche Komponenten unseres Shops mit den entsprechenden Zugriffsrechten angesprochen werden. Diese sind unter anderem die Waffen, die Rabattgutscheine, und User. Über den Webshop können sich neue Nutzer registrieren, per E-Mail-Verkehr miteinander verhandeln und vorhandene sowie neue Waffen erwerben und verkaufen.

Das Modul Software-Architektur hat uns sehr gut auf die Hausarbeit vorbereitet, sodass wir mit unserem vorgegebenen Projekt zügig und stetig vorankamen. Viele der bereits in den Praktika erarbeiteten Konzepte konnten wir fließend in unsere Hausarbeit einarbeiten und hatten dadurch wenig mit Problemen zu kämpfen. Selbst einige neue Herausforderungen wie die An- und Abmeldung mit Quarkus über ein Formular konnten von uns meistens innerhalb eines Tages gelöst werden. Leider war es uns aufgrund mangelnder Zeit nicht möglich die Anwendung bis ins kleinste Detail zu verfeinern, dennoch sind wir mit dem Ergebnis, welches wir erreicht haben, mehr als zufrieden.

## Arbeitsteilung der Dokumentation

Kapitel	Seite [von - bis]	Verantwortlich
1 Einleitung	7	Maxim Zitnikowski
1.1 Vorstellung des Themas	8	Maxim Zitnikowski
1.2 Ziel der Ausarbeitung	8	Maxim Zitnikowski
1.3 Aufbau der Hausarbeit	9	Maxim Zitnikowski
2 Darstellung der Grundlagen	10	Maxim Zitnikowski
2.1 Quarkus	10	Maxim Zitnikowski
2.2 Datenbank im Container	11	Maxim Zitnikowski
2.3 Quarkus Qute	12	Maxim Zitnikowski
2.4 Das Team	13	Maxim Zitnikowski
2.5 Werkzeuge	13	Maxim Zitnikowski
3 Anwendung	14	Maxim Zitnikowski
3.1 Startseite	14-15	Maxim Zitnikowski
3.2 Waffe im Detail	16-17	Maxim Zitnikowski
3.3 Einkaufswagen	18-19	Maxim Zitnikowski, Mathias Hölz
3.4 Profilseite	20-21	Maxim Zitnikowski, Mathias Hölz
3.5 Waffen verkaufen	22	Maxim Zitnikowski, Mathias Hölz
3.6 Testen der Anwendung	23	Maxim Zitnikowski, Mathias Hölz
3.7 Registrieren	23-24	Mathias Hölz
3.8 Login	25-26	Mathias Hölz
3.9 Discount System	27	Mathias Hölz
3.10 Quellcode Dokumentation	28	Mathias Hölz
3.11 E-Mail-System	29	Mathias Hölz
4 Zusammenfassung und Fazit	30	Mathias Hölz
5 Referenzen	32	Mathias Hölz, Maxim Zitnikowski

## 5 Referenzen

- [@CSGO] Counter-Strike: Global Offensive, [https://de.wikipedia.org/wiki/Counter-Strike:\\_Global\\_Offensive](https://de.wikipedia.org/wiki/Counter-Strike:_Global_Offensive) (abgerufen am 06.02.2021)
- [@JDOC] Dokumentationskommentare mit JavaDoc [http://openbook.rheinwerk-verlag.de/javainsel9/javainsel\\_05\\_014.htm](http://openbook.rheinwerk-verlag.de/javainsel9/javainsel_05_014.htm) (abgerufen am 10.02.2021)
- [@1MIO] Steam: 20 Millionen gleichzeitige Spieler, Counter-Strike: Global Offensive bei einer Million, <https://www.pcgameshardware.de/Counter-Strike-Global-Offensive-Spiel-20444/News/Steam-CSGO-Rekord-1345536/> (abgerufen am 06.02.2021)
- [@30PRO] Steam: Umsatzbeteiligung von vielen Plattformen laut Paradox-Vorsitzendem unverschämt, <https://www.pcgameshardware.de/Steam-Software-69900/News/Umsatzbeteiligung-kritik-konkurrenzvergleich-1293623/> (abgerufen am 06.02.2021)
- [@QAURK1] Was ist Quarkus?, <https://www.redhat.com/de/topics/cloud-native-apps/what-is-quarkus> (abgerufen am 06.02.2021)
- [@QAURK2] Einstieg in Quarkus: Das Full-Stack Framework für Microservices und Serverless, <https://jaxenter.de/serverless/quarkus-full-stack-framework-87817> (abgerufen am 07.02.2021)
- [@QAURK3] QUARKUS – BUILT - IN AUTHENTICATION SUPPORT, <https://quarkus.io/guides/security-built-in-authentication#form-auth> (abgerufen am 09.02.2021)
- [@QAURK4] QUARKUS – SENDING EMAILS <https://quarkus.io/guides/mail#testing-email-sending> (abgerufen am 10.02.2021)
- [@QTE] QUTE TEMPLATING ENGINE, <https://quarkus.io/guides/qute> (abgerufen am 07.02.2021)
- [@GIT] 1.1 Erste Schritte - Was ist Versionsverwaltung?, <https://git-scm.com/book/de/v2/Erste-Schritte-Was-ist-Versionsverwaltung%3F> (abgerufen am 07.02.2021)