Technische Dokumentation

zur Bedienung der AllesTaco Website

Inhalt

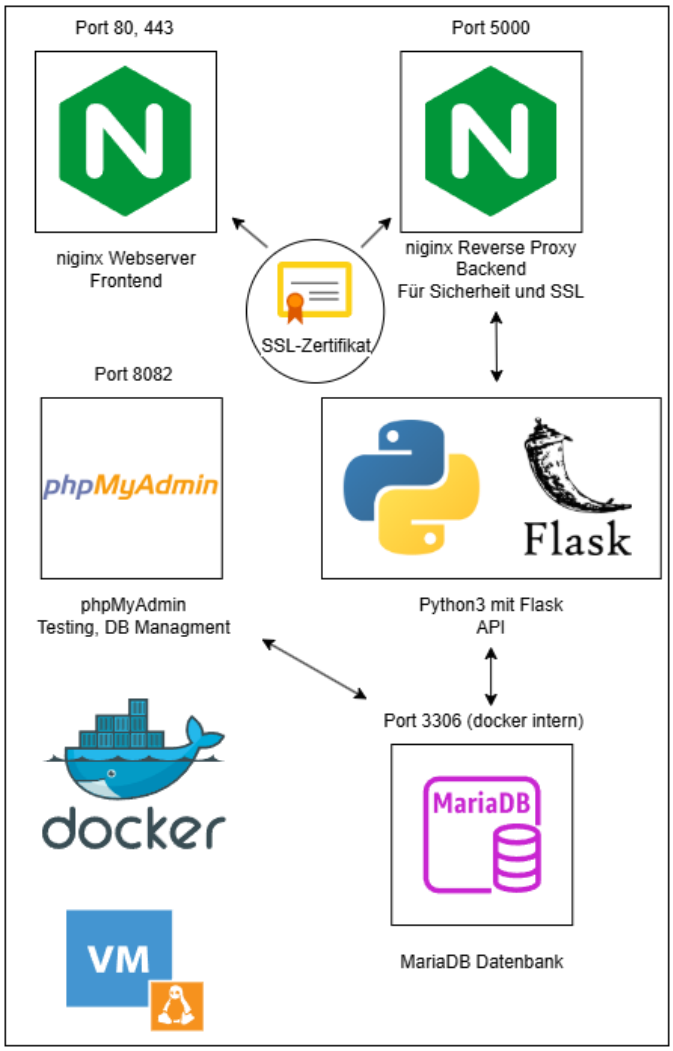
Projektziel

Im Rahmen des Studienprojekts wird eine Verkaufsplattform entwickelt, deren Ziel es ist, den Absatz von Produkten und Dienstleistungen zu fördern. Vorbild dafür sind Plattformen wie ebay, wirkaufendeinauto.de. Es soll der direkte Austausch von Produkten zwischen Privatpersonen ermöglicht werden. Ziel ist es den Verkaufsablauf auf die wichtigsten Elemente zu beschränken und auf komplizierte Prozesse zu verzichten. Die wichtigsten Elemente dabei sind die Benutzerverwaltung, Artikelerstellung, Kommentarfunktion und Statistikauswertung. Zusätzlich soll die Barrierefreiheit mit betrachtet werden.

Systemarchitektur

Schicht	**Tools**	**Beschreibung**
Frontend	HTML, CSS, JavaScript	Klassischer Aufbau einer Website
	Fetch API	Für die API-Kommunikation
	SessionStorage	Temporäre Datenhaltung von Nutzerinformationen
Backend	Python mit Flask	REST-API für Datenverarbeitung
	MariaDB	Dauerhafte Datenspeicherung
	Session-Cookies	Authentifizierung
Tools	GitHub	Versionierung
	Swagger	API-Dokumentation
Deployment	Docker	Containerisierung
	NGINX	Webserver

Technischer Unterbau



Um größtmögliche Flexibilität zu haben und eine einfache und reproduzierbare Möglichkeit der Installation zu gewährleisten wurde sich für eine Microservice-Architektur mit Docker entschieden. Alle Services laufen in unterschiedlichen Containern. Die API läuft in einem Python3 Container in den die notwendigen Abhängigkeiten beim Build-Prozess installiert werden. Alle anderen Container, sind Standard-Images der jeweiligen Applikationen, welche über eingebundene Konfigurationsdateien oder über Umgebungsvariablen in der Docker-Compose Konfiguration konfiguriert wurden.

Um Sicherheit zu gewährleisten, wurde sich für einen Reverse Proxy vor der API entschieden. Dieser sorgt für die SSL-Verschlüsselung und für bessere Sicherheit, beim Hosting im Internet. Hier wurde sich an den best practises von Flask orientiert.

Ebenso wurde, zum Testen und zum Verwalten der Datenbank ein phpMyAdmin Continaer deployt. Dieser hilft beim Administrieren der Datenbank und zum Testen verschiedener Szenarien. In einem Produktiven Deployment, sollte dieser nicht besonders gesichert werden. Im konkreten Fall wurde dieser per Firewall vom Internet abgeschottet.

Verzeichnisstruktur

Dieses Projekt ist eine Webanwendung, die eine Verkaufsvermittlungsplattform bereitstellt. Die Struktur ist grob in drei Teile unterteilt. Einerseits gibt es den Bereich des Käufers. Dieser kann sich Produkte ansehen, dem Warenkorb hinzufügen und den Warenkorb kaufen. Des Weiteren kann der Käufer seine Bestellhistorie ansehen und bereits angeschlossene Käufe bewerten.

Auf der anderen Seite gibt es den Verkäufer. Dieser kann neue Produkte hochladen und zum Verkauf anbieten. Falls es dabei zu einem Fehler können die Produkte noch bearbeitet werden. Daneben sieht der Verkäufer alle seine zurzeit eingestellten Produkte. Darüber hinaus sieht der Verkäufer all seine Verkäufe. Auch eine grafische Aufarbeitung der Verkäufe kann der Verkäufer sich anzeigen lassen.

Zuletzt gibt ein paar Konfigurationsdateien. Dazu zählen die Bilder, die Gestaltung und Funktion der Kopf- und Fußzeile und die Startseite mit der Anmeldung. Im Folgenden gibt es einen genaueren Überblick über die einzelnen Ordner.

Dieses Projekt ist eine Webanwendung, die eine Plattform zur Vermittlung von Verkäufen bereitstellt. Die Struktur ist in drei Hauptbereiche unterteilt:

Käufer

Datei/Ordner	Beschreibung
Kaeufer/	Hauptansicht und Dashboard des Käufers. Zeigt eine standardmäßig eine Auswahl an Produkten bzw. die Ergebnisse einer Suche.
Warenkorb/	Zeigt die zum Warenkorb hinzugefügte Produkte. Nur hier ist die Auswahl der Menge möglich. Hier können die Produkte gekauft werden. Eine konkrete kaufen Funktion gibt es nicht. Kaufen leert nur den Warenkorb.
orderHistory/	Zeigt eine Bestellübersicht des Käufers. Nachdem die Artikel in aus einem Warenkorb gekauft wurde, werden die gekauften Produkte hier angezeigt. Jedes einzelne, gekaufte Produkt kann hier bewertet werden.
Bewerten/	Hier kann der Käufer ein gekauftes Produkt bewertet werden. Eine Bewertung wird dem Verkäufer zugeordnet. Eine Zuordnung zu Produkt ist zwar über die Datenbank möglich, jedoch nicht im Frontend zu sehen.
verkaeuferProfil/	Käufer können hier sehen welche Bewertungen ein Verkäufer hat.

Verkäufer

Datei/Ordner	Beschreibung
Verkaeufer/	Die Startseite des Verkäufers. Hier werden die Produkte angezeigt, die er eingestellt hat. Es gibt ebenfalls die Möglichkeit neue Produkte einzustellen bzw. diese zu löschen.
newProduct/	Hier kann ein neues Produkt hinzugefügt werden. Jedes Feld ist verpflichtend.
editProduct/	Bereits hochgeladene Produkte können hier bearbeitet werden. Es muss dabei immer ein neues Bild eingestellt werden.
verkaeuferProfil/	Die Anzeige der Kommentare, die ein Verkäufer bereits erhalten hat.
verkaeuferVerkaufe/	Der Verkäufer kann hier sehen welche Produkte er bereits verkauft hat.
verkaeuferChart/	Eine grafische Aufarbeitung in Form eines Balkendiagramms.

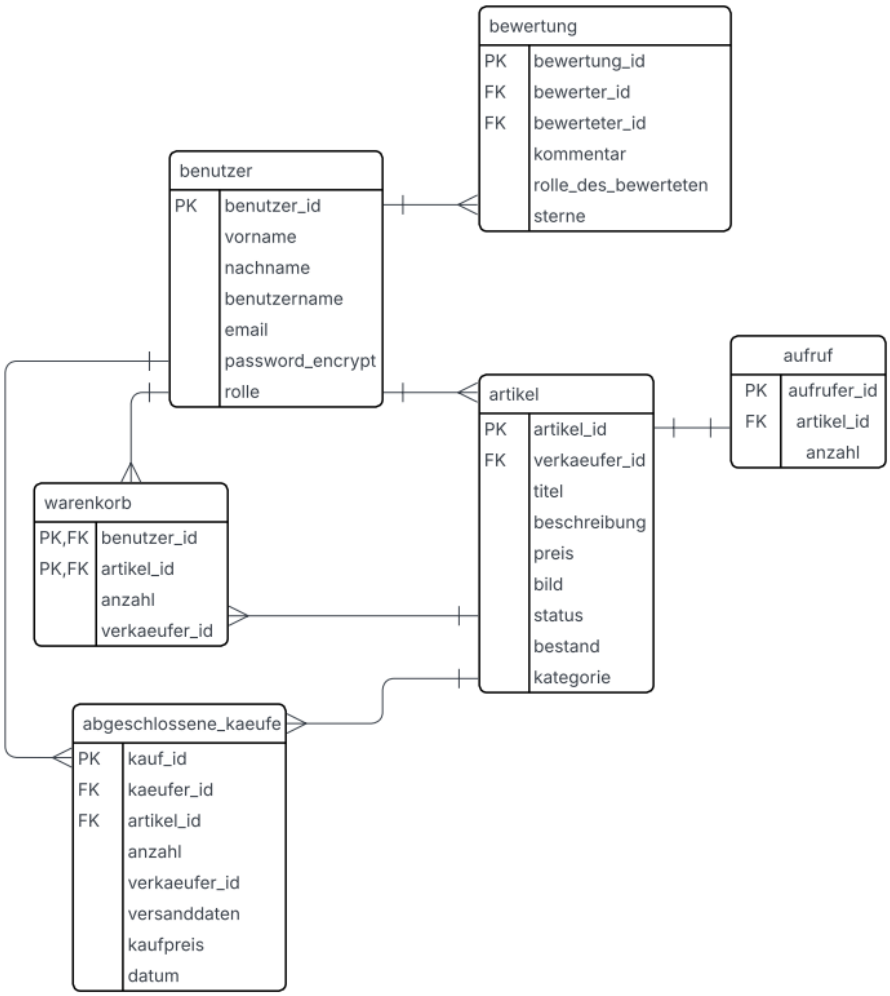
Konfigurationsdateien

Datei/Ordner	Beschreibung
Bilder/	Speicherort für statische Bilder
Start/	Die Landingpage. Diese verwaltet die Cookies und die Auswahl des Kontostyps. Des Weiteren können hier neue User sich registrieren und bekannte User sich anmelden.
all.js/css	Steuert die Funktionen der Kopf- und Fußzeile. Dazu zählen die Suche, das Dropdown Menü und der Impressums- und Datenschutzbutton
config.js/	Verwaltet die URL für die API-Aufrufe

Weitere Informationen zum Aufbau und zu den konkreten Funktionen finden sich in der README auf GitHub sowie direkt im Code in Form von Kommentaren.

Datenbankmodell

Die verwendete Datenbank besteht aus insgesamt sechs Tabellen. Den genauen Aufbau und die Beziehungen zwischen den Tabellen können folgendem Diagramm entnommen werden.



Deployment

Auch der Deployment Prozess ist in detaillierter Form in der README auf GitHub zu finden. Aus diesem Grund wird auf eine ausführliche Erklärung hier verzichtet.

API-Dokumentation

Eine separate API-Dokumentation ist nicht erforderlich, da alle verfügbaren Endpunkte übersichtlich über Swagger abrufbar sind. Die API-Dokumentation ist unter folgender URL erreichbar: https://allestaco.niclas-sieveneck.de:5000/apidocs/

Bekannte Probleme

1. Verkäufer können ihre eigenen Produkte kaufen und diese hinterher auch bewerten
2. Es ist keine Bearbeitung der Accountdaten möglich
3. Die grafische Übersicht über abgeschlossene Verkäufe filtert nicht nach Jahren
4. Weiter und zurück funktioniert nicht bei der Suche
5. Suche funktioniert nur beim Käufer
6. Ausverkaufte Produkte werden nicht gekennzeichnet

Zukunftsaussichten

Innerhalb dieses Proof of Concept konnten leider nicht alle im Konzept beschriebenen Element umgesetzt werden. Folgende Element sind noch nicht Teil der Plattform:

Käufer

- Filter, um die Suche zu verfeinern

Verkäufer

- Bewerten von Käufern
- Verkaufsstatus anpassen
- Suche nach eigenen Produkten
    - Es werden immer alle angezeigt
- Graphische Aufbereitung der Verkäufe nach Kategorien

In einer zukünftigen Version müssen diese Funktionen zur Vollständigkeit mit implementiert werden.

Grundsätzlich bietet dieses Projekt diverse Möglichkeiten zur Weiterentwicklung. So könnte eine Chat Funktion zwischen Käufer und Verkäufer entwickelt werden. Dadurch ist es nicht mehr von Nöten einen festen Preis anzugeben, sondern kann der Preis durch Verhandlungen ermittelt werden.

Außerdem ist eine weitere administrative Rolle sinnvoll. Diese sollte mit erweiterten Rechten ausgestattet werden, um eine überwachende und moderierende Funktion zu übernehmen. Dazu könnte eine Benutzerverwaltung zählen, die Kommentar- und Bewertungsmoderation oder die Produktkontrolle. Dieser bräuchte dann ein eigenes Dashboard und es müssten weitere API-Schnittstellen geschaffen werden, um die gesamte Funktionalität abzubilden.

Ebenso könnte die Sicherheit des Projekts noch weiter erhöht werden. Es könnte eine bessere Passwort-Richtlinie eingeführt werden als nur die Beschränkung auf mindestens 8 Zeichen. Dabei könnten Sonderzeichen, Groß- und Kleinbuchstaben verlangt werden.

Als weiter Verbesserung könnten, die Datentypen in der Datenbank noch verbessert werden. Da hier, zum Beispiel sind bei einigen Daten noch zu große VARCHAR-Werte eingetragen.

Kommentiert [NS1]: Hier habe ich noch erweitert. Hoffe passt so?

Anmerkungen

Dieses Projekt ist mit der Unterstützung von chatgpt.com erstellt worden.