1.Einführung

Name des Projektes: Online Marketplace (Retail Marketplace)

Kurzbeschreibung: Eine Webanwendung zum Kaufen und Verkaufen von Produkten. Benutzer können sich registrieren, Produkte einstellen, Nachrichten an Verkäufer senden und nach Produkten suchen.

2.Funktionalitäten

2.1 Benutzerrollen:

Käufer: Produkte suchen, Produktinformationen anzeigen, mit Verkäufer kommunizieren

Verkäufer: Produkte hinzufügen, verwalten und löschen, mit Käufern kommunizieren.

2.2 Hauptfunktionen:

- Registrierung und Anmeldung von Benutzern.
- Hinzufügen, Bearbeiten und Löschen von Produkten.
- Filtern und Suchen von Produkten nach Kategorien.
- Kommunikation (Nachrichten) zwischen Käufern und Verkäufern.
- Verwaltung des Produktstatus (verfügbar, verkauft, reserviert).

3.Systemarchitektur

Frontend: Angular (TypeScript, HTML, CSS)

Backend: Node.js mit Express.js

Datenbank: PostgreSQL

Bilderspeicherung: Die Bilder werden direkt im Dateisystem des Servers gespeichert.
Speicherort: /public/

Authentifizierung: JWT (JSON Web Token)

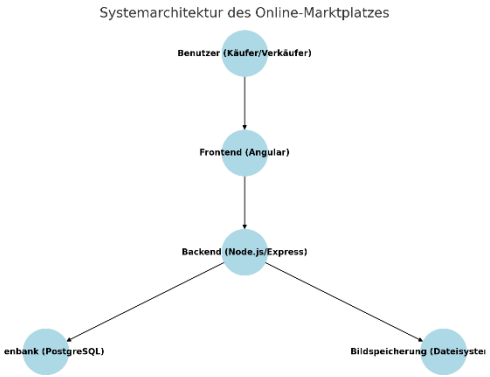

Abbildung 1.Systemarchitektur

4.Technische Umsetzung

4.1 Backend

Das Backend basiert auf Node.js und Express zur Verarbeitung der API-Anfragen.

API-Endpunkte:

POST /register	Benutzerregistrierung
POST /login	Anmeldung
GET /products	Produktliste abrufen
GET /products/:id	Benutzerprodukte abrufen
GET /categories	Produktkategorien abrufen
POST /messages	Nachrichten zwischen Benutzern senden
GET /messages/:userId	Nachrichten für das Benutzer abrufen
DELETE /products/:id	Produkt löschen
PATCH /products/:id/status	Produktstatus aktualisieren

4.2 Frontend

Das Frontend wurde mit Angular entwickelt, einer modernen Single Page Application (SPA)-Technologie. Es kommuniziert mit dem Backend über REST-APIs und bietet eine interaktive Benutzeroberfläche.

4.2.1 Modelle

Message

Das Message-Modell repräsentiert Nachrichten, die zwischen Benutzern im Online-Marktplatz ausgetauscht werden. Es wird für die Kommunikation zwischen Käufern und Verkäufern verwendet.

Product

Das Product-Modell beschreibt die Eigenschaften eines Produkts, das im Marktplatz angezeigt, hinzugefügt, bearbeitet oder gelöscht werden kann.

Category

Das Category-Modell beschreibt die verschiedenen Produktkategorien im Online-Marktplatz. Die Kategorien sind hierarchisch organisiert (z. B. Elektronik → Laptops → Gaming-Laptops).

4.2.2 Hauptkomponenten

RetailComponent

Die zentrale Komponente für den Marktplatz, die Navigation, Produktfilterung und Benutzerinteraktionen verwaltet.

Funktionen:

navigateToAddProduct() - leitet den Benutzer auf die Seite zum Hinzufügen eines neuen Produkts.

filterProducts() - aktualisiert den Suchbegriff im ProductfilterService um Produkte zu filtern.

backToMainPage() - bringt den Benutzer zurück zur Hauptseite und aktualisiert zuvor den Filter.

goMessages() - öffnet die Seite zur Anzeige gesendeter und empfangener Nachrichten.

logout() -meldet den aktuellen Benutzer ab und leitet zur Anmeldeseite weiter.

ProductListComponent

Lädt die Produkte aus API, entfernt die Produkte mit „verkauft“ Status aus dem Sicht, baut ein Hierarchiebaum für den Produktenkategorien, zeigt die Benutzerprodukten

ngOnInit() - initialisiert die Komponente und lädt beim Start die Produkt- und Kategoriedaten, abonniert den Suchbegriff im ProductfilterService, um dynamische Filterung zu ermöglichen.

loadProducts() - holt alle verfügbaren Produkte vom Server und filtert bereits verkaufte Produkte heraus.

loadCategories() - lädt alle Kategorien und erstellt daraus einen hierarchischen Kategoriebaum.

updateUserProducts() - filtert die Produktliste, sodass nur die Produkte des angemeldeten Benutzers angezeigt werden.

filterProducts(searchTerm: string) -filtert die Liste basierend auf dem Suchbegriff.

navigateToDetails(product: any) - leitet den Benutzer zur Detailansicht des ausgewählten Produkts weiter.

buildCategoryTree(categories: Category[]) - erstellt eine hierarchische Struktur aus einer flachen Kategorienliste.

filterProductsByCategory(categoryId: number) - filtert die Produkte nach der gewählten Kategorie und allen Unterkategorien.

getAllChildCategoryIds(categoryId: number) - gibt eine Liste aller IDs der Unterkategorien (rekursiv) zurück.

ProductDetailsComponent

Zeigt eine detaillierte Ansicht eines Produkts mit Bild, Preis und Beschreibung. Bietet eine Schaltfläche zum Senden von Nachrichten an Verkäufer. Gibt Möglichkeit einen Status (verfügbar/reserviert/verkauft) zu verändern.

ngOnInit() - lädt die Produktdetails, den aktuellen Benutzer und fügt eine Willkommensnachricht im Chat hinzu.

isProductOwner() - überprüft, ob der aktuelle Benutzer der Eigentümer des Produkts ist.

onStatusChange(event: Event) - ermöglicht dem Eigentümer, den Status des Produkts zu ändern.

deleteProduct() - löscht das aktuelle Produkt.

closeChat() - Schließt den Chat und leert die Nachrichtensammlung.

AddProductComponent

Fügt die neue Produkte vom Benutzer hinzu.

noOnInit()- wird beim Laden der Komponente aufgerufen und lädt die verfügbaren Kategorien.

loadCategories() - holt die verfügbaren Produktkategorien vom Server.

addProduct() -erstellt ein neues Produkt und sendet es an den Server.

onFileChange(event: any) - verarbeitet das Hochladen von Produktbildern.

isLeafCategory(categoryId: number | null) - prüft, ob die gewählte Kategorie keine Unterkategorien hat.

getCategoryPrefix(categoryId: number | null) - fügt einen Bindestrich (-) für jede Ebene der Kategorie hinzu, um hierarchische Abstände visuell anzuzeigen, geht iterativ durch die Elternkategorien und zählt die Ebenen.

onSubmit() - leitet das Hinzufügen des Produkts ein.

MessageListComponent

Zeigt eingehende und ausgehende Nachrichten für Benutzer

ngOnInit() - wird beim Laden der Komponente aufgerufen und lädt die Nachrichten des aktuellen Benutzers.

loadMessages() - lädt alle Nachrichten des aktuellen Benutzers vom Server.

selectMessage(message: Message) - ermöglicht dem Benutzer, eine Nachricht auszuwählen, um darauf zu antworten.

sendReply() - sendet eine Antwortnachricht an den Gesprächspartner.

closeReplyWindow() - schließt das Antwortfenster, indem die ausgewählte Nachricht gelöscht wird.

4.2.3 Services

AuthserviceService

Verwaltet die Authentifizierung, Token-Speicherung, Login und Logout-Funktionalitäten.

getToken() - ruft das gespeicherte Authentifizierungstoken aus dem LocalStorage ab.

setToken(token: string) - speichert ein neues Authentifizierungstoken im LocalStorage.

getUserId() - dekodiert das gespeicherte Token und extrahiert die Benutzer-ID.

register() - meldet einen neuen Benutzer mit username und password beim Server an.

decodeToken() - dekodiert das JWT-Token und gibt den Nutzerdatensatz als JSON-Objekt zurück.

checkTokenExpiration() - prüft, ob das gespeicherte Token bereits abgelaufen ist.

isAuthenticated() - überprüft, ob der Benutzer aktuell angemeldet ist.

logout() - beendet die Benutzersitzung.

ProductFilterService

Enthält BehaviourSubject aus RxJS Bibliotek, der immer den letzten gespeicherten Wert enthält und ihn an neue Abonnenten weitergibt. Hier wird verwendet um der Zustand zwischen Komponenten nämlich ProductListComponent und RetailComponent zu teilen.

updateSearchTerm(term: string) - aktualisiert den aktuellen Suchbegriff und benachrichtigt alle abonnierten Komponenten.

MessageService

Der MessageService verwaltet die Nachrichtenfunktionalität. Er ermöglicht es den Benutzern, Nachrichten zu senden und ihre empfangenen Nachrichten abzurufen.

sendMessage(message: Message) - sendet eine Nachricht vom aktuellen Benutzer an einen anderen Benutzer.

getUserMessages(userId: string) - holt alle Nachrichten für einen bestimmten Benutzer vom Server.

ProductService

Der ProductService verwaltet die Produktverwaltung im Online-Marktplatz. Er ermöglicht es Benutzern, Produkte abzurufen, hinzuzufügen, zu aktualisieren und zu löschen.

getProducts() - lädt die vollständige Liste der Produkte vom Server.

getProductById(productId: number) - lädt die Details eines bestimmten Produkts anhand seiner ID.

getCategories() - lädt die verfügbaren Kategorien vom Server.

addProduct(product: Product) - sendet die Daten eines neuen Produkts an den Server.

deleteProduct(productId: number) - löscht ein Produkt anhand seiner ID.

updateProductStatus(productId: number, status: string)- aktualisiert den Status eines Produkts (available, reserved, sold).

4.3 Routing

Das Routing ermöglicht eine nahtlose Navigation zwischen den Seiten.

```
export const routes: Routes = [
    {
        path: 'retail',
        canActivate: [AuthGuard],
        component: RetailComponent,
        children: [
            { path: '', component: ProductListComponent },
            { path: 'product/:id', component: ProductDetailsComponent },
            { path: 'add-product', component: AddProductComponent },
            { path: 'messages', component: MessageListComponent },

        ],
    },
    { path: 'register', component: RegisterComponent },
    { path: 'login', component: LoginComponent },
    { path: '', component: HomeComponent },
];
```

Abbildung 2. Routing

4.4 Database

PostgreSQL

Tabelle:

users (id, username, password)

id	username	password
[PK] integer	character varying (255)	character varying (255)

products (id, name, price, description, images, category_id, owner_id, status)

id	name	price	description	images	category_id	owner_id	status
[PK] integer	character varying (255)	numeric (10,2)	text	text[]	integer	integer	character varying (20)

categories (id, name, parent_id)

id	name	parent_id
[PK] integer	character varying (255)	integer

messages (id, product_id, content, created_at, sender_id, receiver_id)

id	product_id	content	created_at	sender_id	receiver_id
[PK] integer	integer	text	timestamp without time zone	integer	integer

5.Demonstration

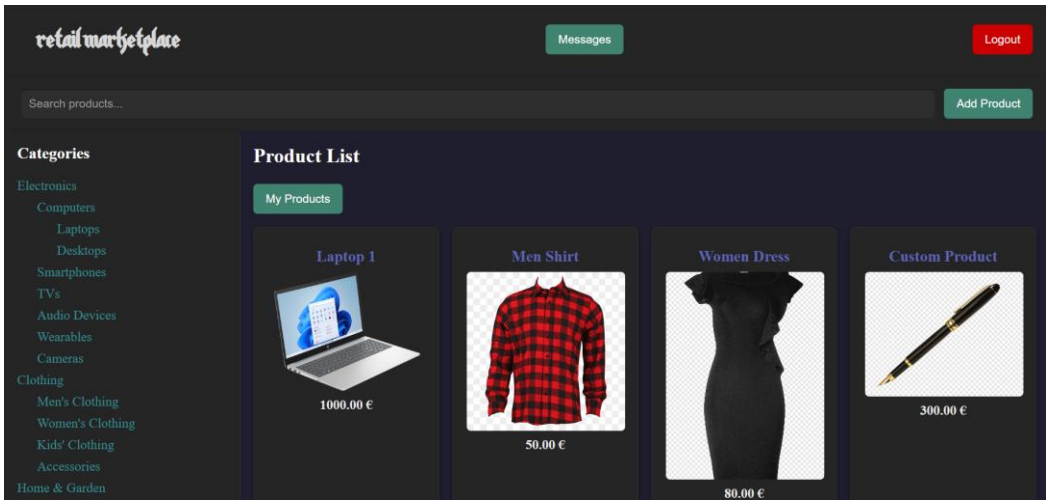

Abbildung 3.Hauptseite

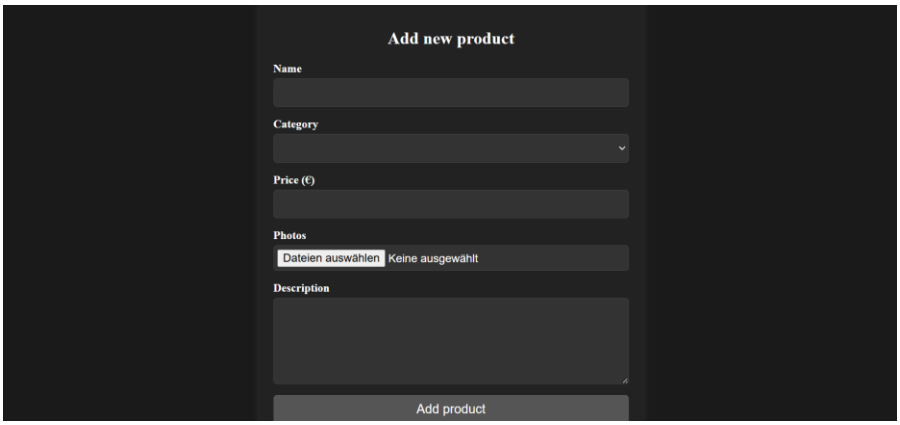

Abbildung 4. Neues Produkt erstellen

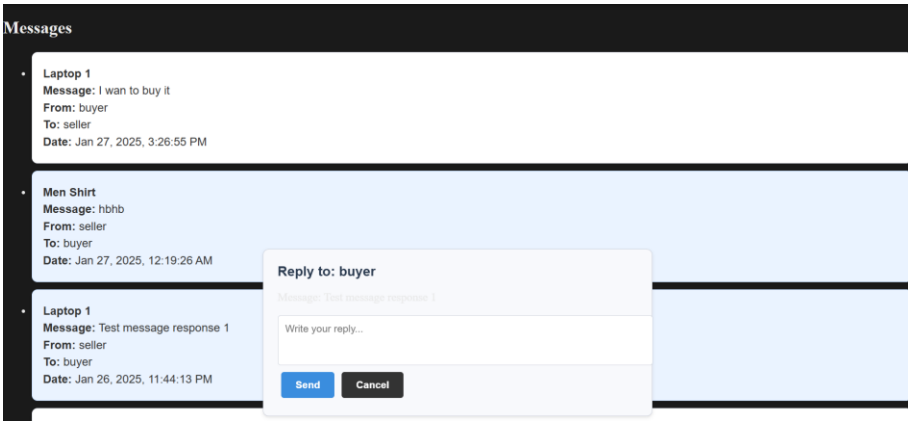

Abbildung 5. Nachrichten

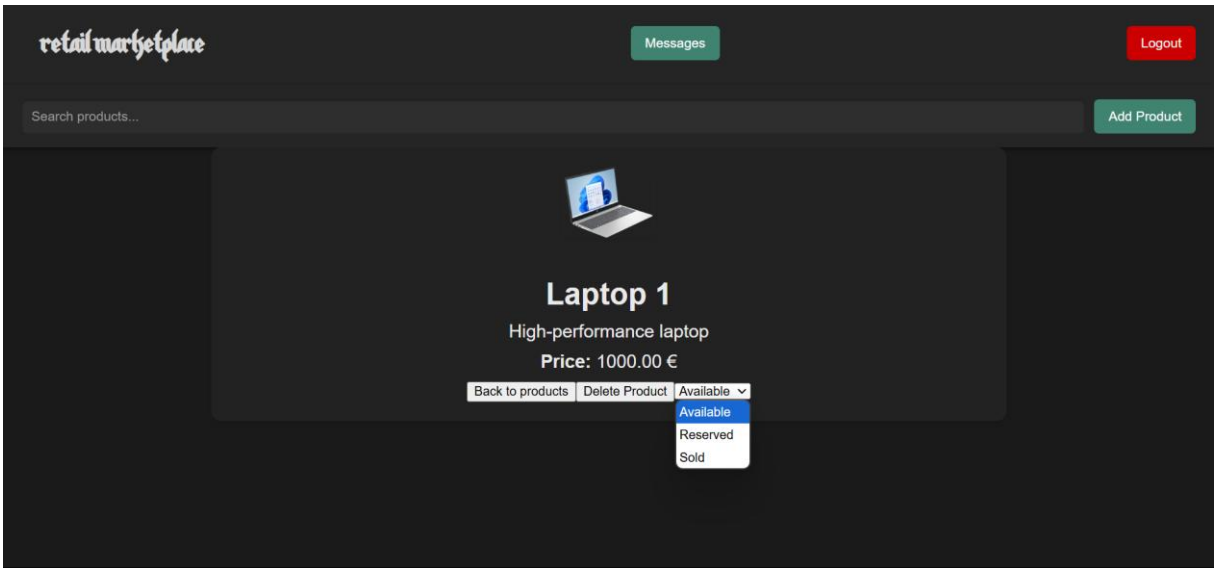

Abbildung 6. Produktdetails

6.Fazit

Das Projekt „Online-Marketpace“ wurde erfolgreich entwickelt und implementiert, um eine benutzerfreundliche Plattform für den Kauf und Verkauf von Produkten bereitzustellen.
Die Anwendung bietet eine intuitive Benutzeroberfläche, effiziente Produktverwaltung und eine einfache Kommunikationsmöglichkeit zwischen Käufern und Verkäufern.