Fiori App Entwicklung II

Aufgabe 8 (40 Punkte):

In dieser Aufgabe soll die in Aufgabe 7 erstellte FitReality App weiterentwickelt werden. Hierzu soll auf dem Code aus Aufgabe 7 aufgebaut werden. Ist die Aufgabe abgeschlossen, soll der Entwicklungsstand entsprechend getagged und in das gleiche Repository wie in Aufgabe 7 gepushed werden.

FitReality Shop entwickeln

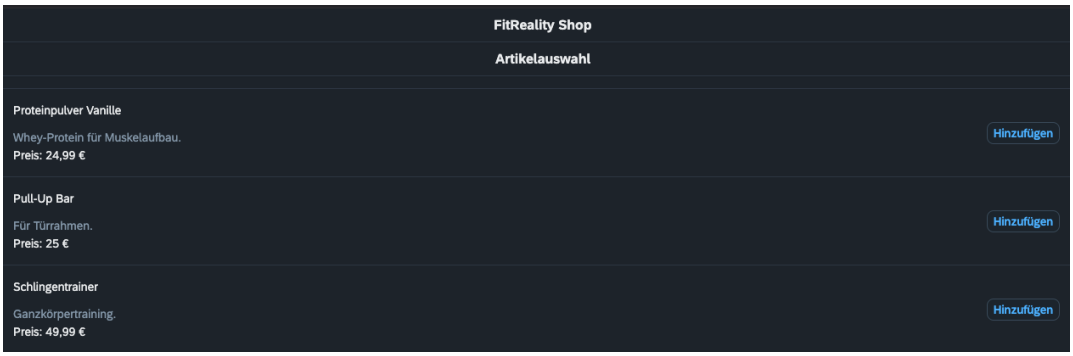
Analog zu Aufgabe 7, soll der in Aufgabe 8 zu erstellende Shop an das gleiche Backend angebunden werden. Wenn das Backend lokal gestartet wird, ist dies standardmäßig unter http://localhost:4004 erreichbar. Die für diese Aufgabe relevante Entity ist

- http://localhost:4004/odata/v4/fit-reality/Shop

1. Implementiert den Kleine-Helden Konfigurator so weit, wie in der Vorlesung gezeigt

    a. Bei Klick auf den Button “Zum FitReality Shop” soll auf die FitReality Shop umgeleitet werden.

    b. Die FitReality Shop Seite soll alle Artikel anzeigen, die der Shop derzeit anbieten:



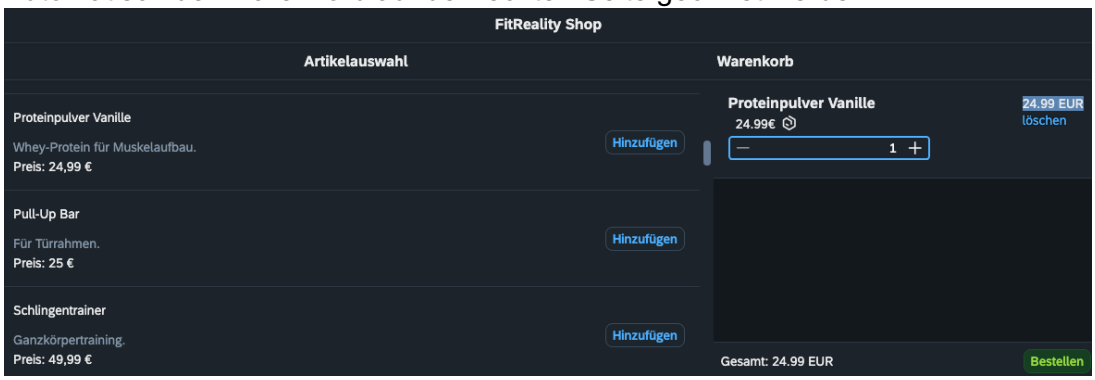
Hierzu soll die Entity http://localhost:4004/odata/v4/fit-reality/Shop benutzt werden. Diese Entity liefert eine Liste von Shopartikeln zurück. Jeder Artikel hat folgenden Aufbau:


```
{
      createdAt: "2025-01-22T09:21:43.323Z",
      createdBy: "anonymous",
      modifiedAt: "2025-01-22T09:21:43.323Z",
      modifiedBy: "anonymous",
      ID: "0b7b0919-9392-4eb5-8009-7892c7c95c26",
      name: "Deuserband",
      description: "Elastisches Trainingsband.",
      price: 13.5,
      amount: 55
}
```


Jeder Artikel soll wie folgt dargestellt werden:

- Titel: <name> des Artikels
- Beschreibung: <description> des Artikels
- Preis: <price> des Artikels
- Button „Hinzufügen“

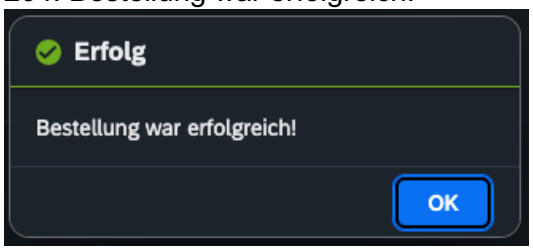
c. Beim Klicken auf den „Hinzufügen“ Button soll:
   i. Der aktuelle Artikel dem Warenkorb hinzugefügt werden
   ii. Automatisch der Warenkorb auf der rechten Seite geöffnet werden



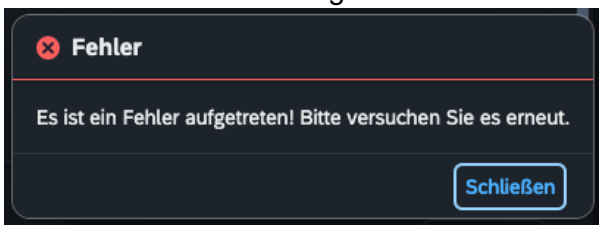
d. Warenkorb implementieren
    i. Der Warenkorb zeigt alle im Warenkorb befindlichen Elemente in einer Liste an
    ii. Für jedes Element wird folgendes angezeigt:
        1. <Name>
        2. Ein UI Element welches es erlaubt die Anzahl der Elemente im Bereich von 1-9 zu ändern.
        3. Einen „Löschen“ Button um das jeweilige Element aus dem Warenkorb zu entfernen.
        4. Der Gesamtpreis des jeweiligen Elements am rechten Rand.
        5. Die Gesamtsumme des Warenkorbs ganz unten im Warenkorb (links neben dem „Bestellen“ Button) angezeigt werden. Die Berechnung soll im Frontend stattfinden. Werden Artikel dem Warenkorb hinzugefügt oder gelöscht, soll sich die Summe automatisch korrekt aktualisieren.

e. Bestellen Funktion implementieren
   i. Im Warenkorb soll ein „Bestellen“ Button angezeigt werden (siehe Screenshots oben)
   ii. Beim Klicken auf den „Bestellen“ Button soll abhängig von der Antwort des Servers eine passende Meldung angezeigt werden:

204: Bestellung war erfolgreich!



500: Es ist ein Fehler aufgetreten! Bitte versuchen Sie es erneut.



2. Stellt sicher, dass sich die Anwendung korrekt starten lässt und die Daten wie in der Vorlesung gezeigt darstellt.

3. Checkt die Lösung für eure Aufgabe in euer Git-Repository ein:
   a. Fügt den Tag „aufgabe-8“ zu dem Abgabecommit hinzu:

```
git tag aufgabe-8
```

   b. Push alle Commits und Tags:

```
git push origin --all
```

Wichtig: Wir kontrollieren diesen Tag. Falls er falsch gesetzt ist (z.B. weil ihr danach noch Commits gemacht habt), werden die Änderungen nicht berücksichtigt. Falls der Tag nicht gesetzt ist, gibt es Punktabzug und wir kontrollieren HEAD.

Abgabeformat: Checkt eure Abgaben im OctaVIA GIT ein

Deadline:

Abgabe: Donnerstag, 30.01.2025, 14:00 Uhr