Webes szoftverfejlesztés gyakorló feladat – Csokirendelés

Ebben a feladatban a képeken látható reszponzív viselkedésű weboldalt kell készítened Bootstrap keretrendszer segítségével, mely alkalmas csoki rendelések vezetésére. A szerver oldali REST API kiszolgálót Laravel, a frontend funkciókat Vue projektként kell létrehoznod!

Adatbázis-kezelés feladatrész (3 pont)

1. Hozz létre a lokális SQL szerveren csokirendeles_db néven adatbázist! Állítsd be az UTF8 kódolást alapértelmezettnek az adatbázis létrehozásánál! Az adatbázis alapértelmezett rendezési sorrendje a magyar szabályok szerinti legyen!
2. Az iménti adatbázisban készítsd el a **chocolates** nevű táblát a következő mezőkkel:
   - id: automatikusan növekvő egész érték, a tábla kulcsa
   - brand: szöveg
   - chocolate_name: szöveg
   - price: egész szám
   - expiry_date: dátum
3. Az iménti adatbázisban készítsd el az **orders** nevű táblát a következő mezőkkel:
   - id: automatikusan növekvő egész érték, a tábla kulcsa
   - email: e-mail
   - address: szöveg
   - chocolate_id: egész szám, idegenkulcs
   - count: egész szám
   - all_price: egész szám (számított mező)

Reszponzív viselkedésű weboldal feladatrész (10 pont)

Frontend feladatrész (15 pont)

Backend feladatrész (12 pont)

1. Hozz létre Eloquent Modelleket a csokirendeles_db tábláihoz a tanult Laravel konvenció alapján, határozd meg a Model attribútumait, és gondoskodj az adatbáziskapcsolat létrehozásáról!
2. Minden *csokival* kapcsolatos kérést a ChocolateController, minden *rendeléssel* kapcsolatos kérést az OrderController kezeljen!
3. Hozz létre két API végpontot, melyek az alábbi címeken legyenek elérhetők:
   http://localhost/api/chocolates
   http://localhost/api/orders
4. A chocolates végpontot úgy alakítsd ki, hogy képes legyen GET és POST típusú kérések kiszolgálására a létrehozott Controller metódusainak segítségével!
   POST kérés esetén:
   A rögzítendő adatok JSON formátumban érkeznek a következő minta szerint:

{"brand":"Milka", " chocolate_name ":"Oreo táblás csokoládé 100g", "price":460, ”expiry_date”: ”2025-11-10”}
A választ asszociatív tömbként küld vissza a minta alapján:
["uzenet" => "Csokoládé a rendszerben!"]
GET kérés esetén:
A teljes adattábla tartalmát JSON formátumú tömbként küldjük vissza eloquent model segítségével.

5. Az orders végpontot úgy alakítsd ki, hogy képes legyen GET, POST, PUT és DELETE típusú kérések kiszolgálására a létrehozott Controller metódusainak segítségével!

   POST kérés esetén:
   A rögzítendő adatok JSON formátumban érkeznek a következő minta szerint:
   {"email":" tanarur@nejanet.hu", ”address”:”3300, Eger, Rákóczi út 48”, "chocolate_id":"1", "count":"10"}
   Gondoskodj arról, hogy rögzítés előtt megvizsgáld:
   - a csoki létezik-e,
   - e-mail megfelelő formátumú
   - cím szöveges adat
   - a rendelt csokik száma legalább 1, maximum 40db

   A rendelés létrehozásánál tárold el az összárat a csoki ára alapján.

   Készíts/módosíts magyar nyelven íródó validációs hibaüzenetet!
   A választ asszociatív tömbként küld vissza a minta alapján:
   ["uzenet" => "Sikeres rendelés!"]

   GET kérés esetén:
   A teljes adattábla tartalmát JSON formátumú tömbként küldjük vissza, melynek elemei az adatbázis mezőneveivel megegyező adattagokkal rendelkező objektumok.

   PUT kérés esetén:
   A végpontnak paraméterezhetőnek kell lenni a módosítani kívánt rekord azonosítójával:
   http://localhost/api/orders/{id}
   A frissítendő adat JSON formátumban érkezik a következő minta szerint
   {„count”: x }
   Gondoskodj arról, hogy frissítés előtt megvizsgáld:
   - a rendelt csokik száma legalább 1, maximum 40db

   Számold újra összárat!
   Rögzítsd a változtatásokat, mjad adj üzenetet!
   ["uzenet" => "Rendelés mennyisége frissítve!"]

   DELETE kérés esetén:
   A végpontnak paraméterezhetőnek kell lenni a törölni kívánt rekord azonosítójával:
   http://localhost/api/orders/{id}
   A választ asszociatív tömbként küldd vissza a minta alapján:
   ["uzenet" => "Rendelés törölve"]

Minták: