Progtech warehouse alkalmazás

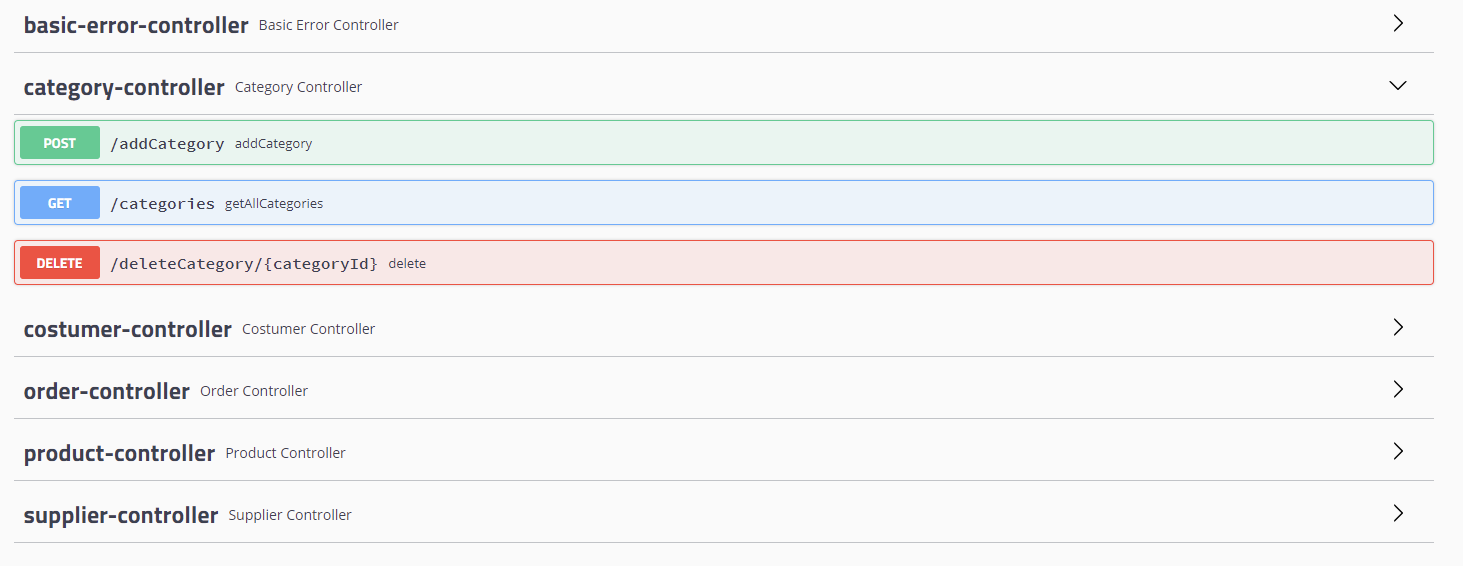
Cél:

Egy rakták Manager program létrehozása, amely képes kapcsolatot megvalósítani több különböző inhomogén raktárrendszer között. Lehetőséget biztosít az áruk és rendelések kezelésére, illetve az beszállítók és vásárlók adatainak adminisztrációjára.

Technológia:

Az alkalmazást spring boot keretrendszerben fogom megvalósítani. A projekt alapját a spring initializr nevű weboldalon generáltam.

A kezelőfelületen az összes táblához hozzá lehet adni, törölni, vagy kilistázni az adatbázisban lévő elemeket.

A kezelőfelület megvalósítására a swagger plugint alkmalazok. A felület POST, GET, és DELETE metódusokkal kommunikál a szerverrel. Minden táblához tartozik egy.

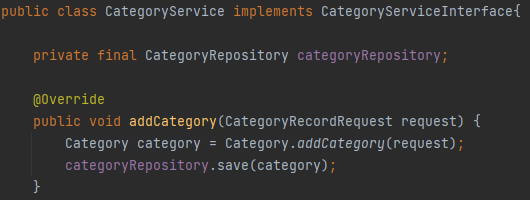
A felület POST metódusokkal képes requesteket küldeni a szervernek az adatok táblába való rögzítésére. A POST metódus alkalmazásakor a példány minden mezőjét meg kell adni, kivéve az elsődleges kulcsot, amit a szerver automatikusan rendel a példányhoz.

A GET metódus az összes példányt megjeleníti, elsődleges kulcsokkal együtt. Az elsődleges kulcsot beírva a DELETE requestbe törölni lehet a példányt az adatbázisból.

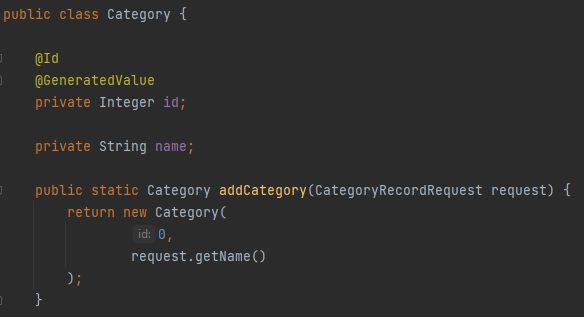
A szerver POST utasításokat requestekként kapja meg. Minden táblához tartozik egy RecordRequest osztály ami leírja hogy milyen mezőket kap a szerver egy adott POST request esetén.

A beérkezett RecordRequestet controller fogadja, amiből minden táblához készítek egyet.

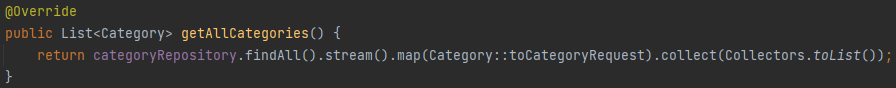
A controller meghívja a megfelelő „add{valami}” metódust ami a {valami}ServiceInterface interfészben van definiálva, de implementálva a {valami}Service osztályban van. Minden Service osztály példányosít egy Repositoryt ami a Dao mappában van.

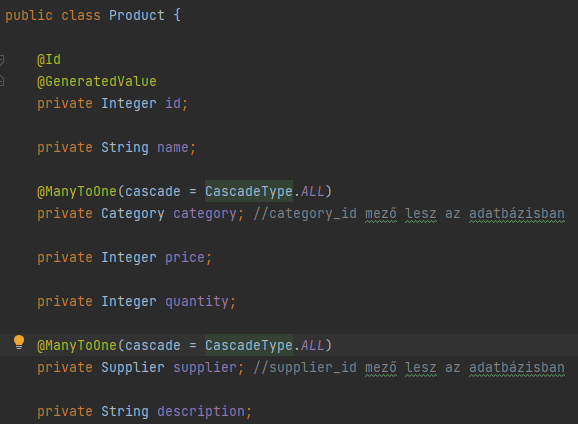


Az „add{valami}” metóduson belül történik egy átalakítás. A paraméterben megkapott RecordRequest-ből entitást csinálunk. Az entitás fog eltárolódni az adatbázisban, ezért neki már rendelkeznie kell egy azonosítóval, ami egyértelműen azonosítja őt táblában.



Az entity osztályon belül egy „@Id” lombok annotációval ellátjuk az id mezőt, ami így minden táblához hozzáadott entitást eltérő azonosítóval lát el. A képen látszódik a konstruktor amivel létrehozzuk az entitást.

Az táblában lévő adatokat egy egyszerű „getAll{valami}” metódussal tudjuk lekérdezni, ami visszaad egy listát a repositoryban lévő elemekkel. 

Az adatbázist a Hibernate-szoftver építi fel az entity osztályok, és az azokban lévő annotációk alapján. Minden entityre létrehoz egy adattáblát, az annotációkkal pedig táblák közötti kapcsolatokat, idegen kulcsokat tudjuk meghatározni. 

A Hibernate működéséhez szükséges adatok az application.properties állományban találhatóak a resources mappában. Ilyen például az ip cím és a port. Emellett be lehet állítani, hogy törölje a létező adatbázist és utána hozzon létre újat, vagy csak ellenőrizze, hogy megfelelő-e.

A projekthez adatbázisához Wampservert használtam. A szerver induláskor létrehozza az adatbázist. Amit a localhost PHPMyAdminban el lehet érni.

Az adatbázis végül így néz ki:

