Figyelem!

Forkold a SpringModuleTest nevű projektet, majd clone-ozd a saját felhasználóneved alatt létrejött projektet! A megoldásodat commit-old és push-old!
Amikor elkészültél, küldd el a GitHub-os projekted linkjét nekünk Slack-en (Balázsnak és Riának is)!
Bár nem elvárás, azonban érdemes munka közben gyakran commit-olnod és push-olnod, mert így biztosan nem fog elveszni a munkád.

Megjegyzés: Megoldásokat nem fogadunk el a fentiektől eltérő formában és/vagy módon. Tehát a megoldást nem fogadjuk el tömörített mappaként, Google Drive-on megosztott fájlokként, képformátumban, pdf-formátumban, emailben küldve stb.

Kérjük, a megoldás írása előtt szánj időt a feladatok átolvasására, továbbá a már készen kapott programkód értelmezésére!
A programkód tartalmaz dokumentációt, amelyet erősen ajánlott elolvasni; továbbá olyan elemeket, amelyek szabadon felhasználhatóak. Azonban a módosításuk (törlésük, átírásuk) problémákat okozhat. Ezért a már megkapott programkód módosítása nem ajánlott - kivéve abban az esetben, amikor a feladat kifejezetten kér bizonyos módosításokat.

A programkód írásakor vedd figyelembe az alábbiakat:

A modulzáró eredménye automatikusan 0 pont (0%), ha a leírtak közül bármelyik teljesül:
- a modulzárót nem az elvárt formában és módon oldod meg, a leadásakor nem felelsz meg az elvárásoknak
- a programkód nem tekinthető meg (így nem ellenőrizhető)
- a programkód bármilyen személyes adatot (nevet, lakcímet, bankkártyaadatokat stb.) vagy belépési adatokat (felhasználónevet, jelszót) tartalmaz (ide értendők az adatbázis-szerverhez való csatlakozásnál használatos belépési adatok)
- a programkód tartalmazza az adatbázis elérési útját
- a programkódban szintaktikai hiba található, és emiatt nem futtatható (a pom.xml esetleges “pirossága” nem probléma, amennyiben nem befolyásolja a program működését)
- az alkalmazás az indítása után azonnal le is áll (vagyis valamilyen Error vagy Exception miatt nem indítható)

Pontlevonás vagy a feladatokra részpontszám jár az alábbiak miatt:
- a programkód átláthatatlan, nem következetes, nem rendezett
- a programkód nem angol nyelvű elemeket tartalmaz (legyen az akár elnevezés, akár komment) - kivéve az endpointok és a HTML-fájlok, amelyekben megengedett a magyar szöveg
- a megoldás nem követi a feladat leírását, nem tartja be a feladat kéréseit
- a feladat néhány része vagy egésze nem teljesített
- a programkód és a feladatok leírása tartalmaz olykor figyelmeztetéseket - a figyelmen kívül hagyásuk miatt okozott esetleges programhibákért (pl. Exception) szintén pontlevonás jár

Kérlek, a modulzáró megoldása során vedd figyelembe az alábbiakat:

- a már meglévő programkód szabadon felhasználható, azonban módosítása (törlése, átírása) erősen ellenjavalt - természetesen kivétel ez alól az az eset, amikor a feladat kifejezetten kér bizonyos módosításokat
- amennyiben a feladat leírása nem egyértelmű, vagy nem tartalmazza konkrétan a megoldás módját, akkor bármilyen eszköz (szabadon) választható a megoldáshoz
- a megoldás szabadon választott eszköze legyen átgondolt, egyértelmű és logikus
- a programkód legyen formázott, egységes, továbbá feleljen meg a tanult konvencióknak (rendezés, elnevezés stb.)
- a feladatok többsége önállóan és szabadon választott sorrendben megvalósítható
- a tesztelés nem része a feladatnak, azonban legalább manuálisan mindenképpen teszteld a megoldásaidat
- a kinézet (CSS-fájl) nem része a feladatnak, azonban az alkalmazásban elhelyezett *style.css* szabadon felhasználható és módosítható

Feladatok

1. Tedd lehetővé az összes űrhajó adatainak a megtekintését!
   Hozz létre egy endpoint-ot, amelyre HTTP-request-et küldve megjelenik az összes űrhajó adata, amely az adatbázisban található.
   Az endpoint-ot a SpaceShipController osztályba írd. Továbbá hozz létre egy SpaceShipService nevű osztályt a megfelelő package-ben, valamint a többi szükséges fájlt (HTML-fájl, Repository interface stb.)

   Az űrhajókról az alábbi adatok jelenjenek meg a weboldalon, méghozzá ebben a sorrendben:
   - regisztrációs kód,
   - név,
   - űrhajó osztálya,
   - aktív-e

   Az adatok megjelenítésének módja szabadon választható (táblázat, lista, bekezdés stb.). A lényeg, hogy az egy űrhajóhoz tartozó adatok különüljenek el a többi űrhajó adataitól.

   Megjegyzés: összesen 15 darab SpaceShip van az adatbázisban.

2. Tedd lehetővé csak az aktív űrhajók kilistázását!
   Adj lehetőséget az alkalmazás felhasználójának arra, hogy csak azoknak az űrhajóknak az adatait tekinthesse meg, amelyek aktívak.
   Ez a szűrési lehetőség legyen plusz funkció, és továbbra is legyen megtekinthető az összes űrhajó adata. Vagyis ne módosítsd az előző feladatot, hanem egészítsd ki!

   A weboldalon adj lehetőséget a felhasználónak, hogy megszűrje az űrhajókat. Ez a lehetőség szabadon választható (lehet gomb, link, checkbox stb.) A lényeg, hogy a felhasználó számára legyen egyértelmű, hogy van ilyen szűrési lehetőség.

3. Tedd lehetővé adott űrhajó legénységének megtekintését!
   Hozz létre egy endpoint-ot a SpaceShipController osztályban, amely megjeleníti egy adott űrhajó legénységének az összes adatát, amely az adatbázisban található.
   Az endpoint PathVariable-ként kapja az űrhajó regisztrációs kódját vagy nevét (szabadon választható), amely alapján megkeresi az adatbázisból az űrhajót és a hozzá tartozó legénységet.

   A weboldalon jelenjen meg az űrhajó neve - de csak egyszer. A legénységének adatai közül pedig az alábbiak jelenjenek meg, méghozzá ebben a sorrendben:
   - név,
   - rang

   Az adatok megjelenésének módja ezúttal is szabadon választható.

Megjegyzés: Összesen 66 Crew van az adatbázisban. Van olyan űrhajó, amelynek nincs legénysége (a *Baby Doll* nevű), a legtöbb legénységszám egy hajón pedig 10 (*Wave Rider*, *The Cube*).

4. Alakítsd át az Officer osztályt *Entity*-vé és *UserDetails*-zé!
   Legyen az Officer osztályból is egy adatbázis-tábla! Az osztályban lévő field-eket értelemszerűen kezeld! (Hozzáadhatsz további field-eket, amennyiben szükséges.) Továbbá legyen ez az osztály az alkalmazás “regisztrált felhasználója”. Ehhez használd a Spring Security *UserDetails* interface-ét!

   Ügyelj arra, hogy a username field értéke legyen egyedi az adatbázisban!

   Az Officer authority-ja szabadon választható. Bármi lehet (ADMIN, USER, STB.) - akár egy, akár több is.

   Megjegyzés: Vedd észre, hogy az osztály már tartalmaz olyan elemeket, amelyek megkönnyítik a *UserDetails* megvalósítását! Ezeket használd nyugodtan.

5. Alakítsd át az OfficerService osztályt *UserDetailsService*-zé!
   Legyen az OfficerService egy *UserDetailsService*!
   A *loadUserByUsername(String username)* metódus az adatbázisból keresse meg a regisztrált Officer-t.

   Megjegyzés: mivel az Officer csak most lett adatbázis-tábla, ezért nem fog adatokat tartalmazni - ugye.

6. Valósítsd meg a regisztrálás folyamatát!
   Tedd lehetővé az Officer regisztrálását! A felhasználó adatbázisba való írását az OfficerService-ben valósítsd meg. Ügyelj arra, hogy ne lehessen olyan felhasználónévvel regisztrálni, amellyel már regisztráltak!

   A regisztrációhoz hozd létre a megfelelő endpoint-ot és HTML-fájlt is a lenti leírásnak megfelelően!

   Az endpoint sikeres regisztráció esetén irányítson át a már létező */login* endpoint-ra. Sikertelen regisztráció esetén (például a felhasználónév foglalt) adja vissza a regisztrációs HTML-fájlt. (A regisztrációs oldalon érdemes ebben az esetben megjeleníteni egy hibaüzenet is, azonban ennek a megvalósítása nem kötelező.)

   A regisztrációs HTML-fájl tartalmazzon egy form-ot, amely a felhasználónevet és a jelszót kéri a felhasználótól.

   Figyelem! Jelszó nem kerülhet encode-olás nélkül az adatbázisba! Ha úgy valósítod meg a feladatot, hogy nem gondoskodsz a felhasználó jelszavának a védelméről, akkor ***nem jár pont a feladatért****.*

7. Fejezd be az új hajó regisztrálásának megvalósítását!
   A SpaceShipController tartalmaz egy GET endpoint-ot (/spaceship), amely azzal a céllal készült, hogy rögzíthessük egy új űrhajó adatait az adatbázisba. Ehhez tartozik a spaceship_new.html.

   Egészítsd ki a HTML-fájlban található form-ot, hogy az új űrhajó adatainak értéket lehessen adni - kivéve a regisztrációs kódot, elvégre az GeneratedValue. (A form-ban már szereplő select egy lenyíló menü, amely tartalmazza a lehetséges űrhajó-osztályokat. Így ezt a beviteli mezőt már nem kell neked létrehoznod.)
   Ne felejtsd el a form minden hiányzó elemét hozzáadni!

   Hozz létre egy megfelelő endpoint-ot a SpaceShipController-ben, amely kezeli az új űrhajó regisztrálását!
   Az új űrhajó adatainak adatbázisba történő írását a SpaceShipService osztályban valósítsd meg.

   Figyelem! Az űrhajók nevének egyedinek kell lennie. Jelen esetben nem kell kezelned azt, hogy a felhasználó esetleg olyan űrhajó nevet ad meg, amely már létezik - azonban nem árt, ha tudsz erről.

8. Állítsd be az alkalmazás védelmét!
   Jelenleg bárki tud új űrhajót regisztrálni. Módosítsd a WebSecConfig osztályban található beállításokat úgy, hogy csak bejelentkezett felhasználók tudják megtekinteni az űrhajó regisztrációs oldalát - és természetesen csak ők tudjanak új űrhajót hozzáadni az adatbázishoz.

—

Bónusz feladatok

1. Tedd lehetővé az összes legénység megjelenítését!
   Valósítsd meg azt, hogy az adatbázisban szereplő összes legénységi tag adatai megjelenjenek egy endpoint-on keresztül - hasonlóan az űrhajók adataihoz.
   A legénységi tagokról az alábbi adatok jelenjenek meg, méghozzá ebben a sorrendben:
   - név,
   - rang,
   - szolgálati hajó neve

   Az adatok megjelenítése ezúttal is szabadon választható formában történik - azonban most is fontos, hogy az adatok legyenek egymástól elkülönítve.

2. Tedd lehetővé a legénységi tagok nevében való keresést!
   Lehessen szűrni a legénységi tagok listáját a nevük szerint! Vagyis lehessen keresni a nevükben.
   Ügyelj arra, hogy a keresés ne pontos név-egyezésre adjon találatokat, hanem a nevekben keressen.
   Például: csak azokat a legénységi tagokat jeleníti meg, akiknek a nevében szerepel az “a” betű; vagy a “soldier” szó.

Értelemszerűen: akárcsak az űrhajók szűrésénél, itt is legyen egyértelmű a felhasználó számára, hogy van ilyen keresési lehetőség.

3. Tedd lehetővé új legénységi tag felvételét egy adott űrhajóra!
   Valósítsd meg azt, hogy egy bizonyos űrhajóhoz új legénységi tagot lehessen regisztrálni - vagyis menteni az adatbázisba.
   Ügyelj arra, hogy a legénységi tagok rangja csak adott érték lehet (Rank enum).
   Továbbá ügyelj arra is, hogy csak és kizárólag már létező űrhajóhoz lehessen új legénységi tagot hozzáadni.

   A feladat részeként módosítsd a WebSecConfig osztály beállításait úgy, hogy csak bejelentkezett felhasználó tudjon új legénységi tagot menteni az adatbázisba.