Féléves Feladat Követelmények

A hallgatóknak egyedül egy saját ötleten alapuló beadandó feladatot kell megvalósítaniuk. Az elkészült program a tanult rétegzési elvek szerint legyen felépítve, a program kódjában pedig kötelesek használni a technológiákat, amit a programozási részen megismertek. Beadandó feladatot mindenkinek készítenie kell, azoknak is, akik a tárgy egyik felét már teljesítették.

A pontos határidőket gyakorlatvezetők hirdetik ki.

A program valamilyen (minimum három, egymással kapcsolatban álló táblából álló) adatbázis kezelésén keresztül valósítson meg tetszőleges üzleti funkciókat.

A projekt elkészítése során a következő követelményeknek kell megfelelni:

- DoxyGen segítségével generált HTML/CHM/PDF formátumú fejlesztői dokumentáció
- A kód legyen FXCop/StyleCop-helyes, az **oenik.ruleset** rulesetet kell használni
- Adatbázis használata + Entity Framework az eléréshez
- LINQ használata
- Unit tesztekkel ellátott kód (tipikusan az üzleti logika osztályaira)
- Rétegezett architektúra (a felhasználói felülettől leválasztott műveletvégző és adatbázis-kezelő DLL-ekkel, tehát tipikusan minimum 4 projekt:
  Console App + Business Logic + Entities + Tests)
- Egy felhasználós, egyetlen (master) branch-et használó GIT repository

A projekthez szükséges a bitbucket.org weboldalon egy GIT repositoryt készíteni, az alábbi elnevezési konvenciót követve: OENIK_PROG3_ÉV_FÉLÉV_NEPTUN, ahol az ÉV a félév kezdetekor az évszám; a FÉLÉV pedig 1 = tavaszi félév, 2 = őszi félév. A projekt repohoz admin jogú hozzáférést kell adni az **oe_nik_prog** nevű bitbucket felhasználónak (ezt a felhasználót mindegyik oktató el tudja érni).

A **prog_tools.pdf** dokumentumban található egy rövid leírás a félév során használandó eszközökről, mint például a Git, Doxygen, StyleCop, stb...

Felhasználandó segédanyagok:

- oenik.ruleset
- prog_tools.pdf
- hivatalos követelményrendszer dokumentum

Féléves Feladat Specifikáció

A hallgatóknak egyedül egy saját ötleten alapuló beadandó feladatot kell megvalósítaniuk. Az elkészült program kódjában pedig kötelesek használni a technológiákat, amit a programozási részen megismertek. Beadandó feladatot mindenkinek készítenie kell, azoknak is, akik a tárgy egyik felét már teljesítették.

Szükséges a projektfeladathoz egy adatbázist elkészíteni. Az adatbázisban lennie kell minimum 3 táblának, amik idegen kulcsokkal egymásra hivatkoznak. Mindegyik táblában kell minimum 5 nem-kulcs mező. Kapcsolótábla használata esetén a kapcsolótábla nem számít bele a 3 kötelező táblába.

A feladat: ezen 3 táblának teljes körű kezelése (lista + hozzáadás + módosítás + törlés), és ezen kívül néhány (minimum 3) olyan funkció megírása, amik az egyszerű listázáson túlmutatnak, és a megoldáshoz több tábla összecsatolása és/vagy csoportosítás szükséges. Ezen kívül, legyen lehetőségünk egy Java végponttól adatot lekérni, és azt megjeleníteni.

Például: **autómárkák** (id, név, országnév, url, alapítás éve, éves forgalom) + **modellek** (id, márka_id, név, megjelenés napja, motortérfogat, lóerő, alapár) + **extrák** (id, kategórianév, név, ár, szín, többször_használható_e) + **modellExtraKapcsoló** (id, modell_id, extra_id).

Példa funkciólista:

- Márkák listázása / hozzáadása / módosítása / törlése
- Modellek listázása / hozzáadása / módosítása / törlése
- Extrák listázása / hozzáadása / módosítása / törlése
- Modell-extra összerendelések listázása / hozzáadása / törlése
- Legyen lehetőségünk kiírni minden autóhoz az autó TELJES árát is: alapár + a rajta lévő extrák összára
- Legyen lehetőségünk kiírni márkánként az autók átlagos alapárát
- Legyen lehetőségünk kiírni az extrák kategórianeveként csoportosítva azt, hogy melyik kategória hányszor van autókhoz kapcsolva
- Legyen lehetőségünk egy Java webes végponttól "árajánlatokat" kérni: GET-ben adjuk át a Java végpontnak az autó teljes nevét és teljes árát; a Java servlet generáljon véletlenszerűen 5 árat a teljes ár körül, és azokat adja vissza valamilyen XML/JSON formában (5 darab "autónév + vásárló neve + javasolt ár" hármas, valahogy kódolva).

Az autós példa használata NEM engedélyezett. Találjunk ki valamilyen hasonlóan egyszerű, de izgalmasabb témájú struktúrát (az adatbázisok féléves feladattal azonos struktúra használata is engedélyezett). A fentihez képest azonos funkciók implementálása elvárt (mindegyik táblához lista/hozzáadás/módosítás/törlés és minimum 3, a listázáson túlmutató extra feladat, valamint a JAVA végpont használata).

A későbbiekben a projekt menetrendje található. Ahol a CRUD rövidítés szerepel, az a Create,Read,Update,Delete funkciókra, vagyis az alap író/olvasó funkciókra utal.

Dátum	**Megnevezés**	**Eddigre kész...**
Március 16.	Féléves feladat kezdete	Bitbucket regisztráció, VS config, SourceTree install. Név és téma kitalálása VS Solution-kompatibilis név kitalálása (pl. MySongShop) (Ez a projektnevekben legyen használva. A solution neve KÖTELEZŐEN legyen azonos a git repo nevével, OENIK_PROG3_ÉV_FÉLÉV_NEPTUN)
Április 5.	1. iteráció vége	Projektstruktúra létrehozása a git repository-n belül - MySongShop.Data - Class Library - MySongShop.Repository - Class Library - MySongShop.Logic - Class Library - MySongShop.Logic.Tests - Class Library - MySongShop.Program - Console App - Java/Netbeans projekt külön könyvtárban NuGet/Projektek beállítása - Mindegyik projekthez StyleCop és Code Analysis - *.Test-hez nUnit3 (v3), nUnit3TestAdapter, Moq MySongShop.Data - Service-based database, feltöltve adattal - Figyeljünk rá, hogy most kivételesen az MDF/LDF állomány is legyen a git repository része - ADO.NET Entity data model, sql-first - A létrehozó SQL állomány is legyen ennek a projektnek a része MySongShop.Program - A Data projekt App.config file-jából a connection stringet másoljuk át ide, mert csak ez az App.config fog betöltődni - Valamilyen egyszerű menü vezérelje - Jelenleg csak "Ez még nincs kész" üzenet bármely menüelemet kiválasztva - **Kötelező menülemek/funkciók:** - mindegyik entity listázása, bármelyik egyszerű hozzáadása, törlése, módosítása (**teljes CRUD**) - legyen olyan adatlekérés is, ami több táblát/entitást használ fel, és közben valamilyen feltételeket/csoportosítást használ (**minimum 3 nem-CRUD metódus**) - A JAVA végponttól a jövőben adatot lekérő menüelem is legyen (**Java/web interakció**)
Április 26.	2. iteráció vége	MySongShop.Repository - A CRUD műveleteket kiszervezzük egy külön IRepository interfészbe - Opcionális: IRepository<T>, és ehhez kapcsolódó entity-specifikus leszármazottak - Repository implementáció a CRUD műveletekhez MySongShop.Logic - ILogic interfész létrehozása, amiben definiáljuk az üzleti logikai (BL) műveletek listáját (ez az összes

		CRUD és nem-CRUD művelet, amit majd a Console App el tud érni) - Jelenleg legyen kész az összes CRUD művelet, ami belül Repository metódust hív meg - A nem-CRUD műveleteket hagyjuk üresen MySongShop.Logic.Tests - CRUD műveletek tesztelése mockolt repository-val MySongShop.Program - A CRUD műveletek legyenek elérhetőek **Rétegzési szabályok:** - A Console App csak logic műveletet hív, a Logic a CRUD műveleteket továbbítja a Repo felé, és a Repo hívja a DbContext metódusokat. - A Console App-on kívül más project **NEM HASZNÁLHAT** Console.Read/Write műveleteket - A Logic és a Console App **NEM HASZNÁLHAT** dbContext metódusokat, ez egyedül a Repository-nak engedélyezett - Minden réteg **CSAK** az alatta lévő réteggel kommunikálhat (esetleges felfele kommunikáció: eseményekkel - jelenleg nem szükséges) - Az Entity típusok jelenleg mindegyik rétegben használhatóak (ez nem a legjobb, de ebben a féléveben még elfogadható) MySongShop.JavaWeb - Legyen kész a végpont, ami véletlen adatot generál a kapott paraméterek alapján - Böngészőből elérhető (javasolt: postman tesztek)
Május 10.	Féléves feladat vége	MySongShop.Logic - nem-CRUD műveletek implementálása - A lekérdezések nem használhatják a DbContext metódusait, csak a repository adatlekérő metódusait - A repository-tól visszakapott IQueryable adatokat fűzi tovább bonyolultabb lekérdezéssé MySongShop.Logic.Tests - Nem-CRUD műveletek tesztelése mockolt repository-val MySongShop.Program - A nem-CRUD műveletek is legyenek elérhetőek a menün keresztül - Legyen olyan menüpont, ami a Java végpontot használja fel (az adatlekérés refactorálható a Logic rétegbe is akár), és megjeleníti a kapott eredményt