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 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:

- Dotnet 5.0.x projectek
- DoxyGen segítségével generált HTML/CHM/PDF formátumú fejlesztői dokumentáció
- A kód legyen StyleCop+NetAnalyzers-helyes, "zero warnings"
- Adatbázis használata + Entity Framework (Code-First/Sql-First) 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 5 projekt:
  Console App + Business Logic + Repository + Data + 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, az e-mail címe: **nikprog@iar.nik.uni-obuda.hu**).

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

Felhasználandó segédanyagok:

- Prog_tools.pdf ⇒ https://users.nik.uni-obuda.hu/prog3/
- hivatalos követelményrendszer dokumentum ⇒ https://users.nik.uni-obuda.hu/prog3/
- NikGitStats weboldal ⇒ https://users.nik.uni-obuda.hu/gitstats/

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 a code-first/sql-first megközelítéssel. 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 (join, nem lazy load) ÉS csoportosítás (group by)** szükséges. A módosításoknak nem kell generikusnak/flexibilisnek lennie, de mindhárom entitásnak módosíthatónak kell lennie valamennyire.

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

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.

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 hasonló funkciók implementálása elvárt (mindegyik táblához minden CRUD funkció és minimum 3, a listázáson túlmutató extra non-CRUD feladat).

A félév végén Git-en keresztül leadandó a teljes, szabályoknak megfelelő forráskód; szintén a Git-en belül a Doxygen fejlesztői dokumentációval; valamint a Moodle-n keresztül egy

rövid, 1-2 perces videóval, amin belül a hallgató futás közben bemutatja a program használatát és az összes menüelem működését.

Dátum	**Megnevezés**	**Eddigre kész...**
Már.06. 23:55	Git repo létrehozása	- Bitbucket regisztráció, új repository (OENIK_PROG3_ÉV_FÉLÉV_NEPTUN), oe_nik_prog ⇒ admin jog, SourceTree install. - Projekt név kitalálása (space/ékezet mentesen, pl. MySongShop); téma és funkciók kitalálása - .gitignore és readme.md file a repository rootban (readme.md tartalom: projekt neve, valamint a console app funkciólistája) - A .gitignore file-nak a VisualStudio.gitignore tartalmával kell rendelkeznie, az SQL Server bejegyzések kikommentelve - A file-ok szerkeszthetőek a Bitbucket webfelületéről
Már.20. 23:55	Solution, projektek	A projekt nevek a korábban kitalált projekt címet használják. A solution neve LEGYEN AZONOS a git repo nevével. Projektek létrehozása a git repository-n belül (Dotnet Core projektek) - MySongShop.Data - Class Library - MySongShop.Repository - Class Library - MySongShop.Logic - Class Library - MySongShop.Logic.Tests - Class Library - MySongShop.Program - Console App NuGet/Projektekhez csomagok hozzáadása - Mindegyik projekthez StyleCop és FxCopAnalyzers hozzáadása - *.Test-hez nUnit3 (v3), nUnit3TestAdapter, Moq hozzáadása - Program-hoz ConsoleMenu-simple hozzáadása **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 félévben még elfogadható) - **Figyeljünk rá, hogy kivételesen az MDF/LDF állomány is legyen a git repository része (.gitignore szerkesztése!)** - Követni kell a SOLID elveket és az alapvető réteg-elválasztási szabályokat! Kerüljük el a "god object"-ek használatát: a Repository osztályokat az entitások szerint daraboljuk fel - Nincs konkrétan leírt elvárás a Logic osztályok szétválasztására, de a józan ész szabályai szerint vágjuk fel az logikai réteget is több osztályra, ahogy az a feladatnak megfelel (ez ne legyen feltétlenül entitás-központú, a DDD elvek szerint a Logic osztályok legyenek mindig üzleti logika központúak). - Elvárt a nulla fordítási warning és error**.**

Ápr.10. 23:55	Menü + Minden lista művelet	MySongShop.Data - Service-based database, kódból vagy SQL-ből feltöltve adattal - ADO.NET EF Core data model, OnModelCreating() metódussal MySongShop.Repository - A CRUD műveleteket kiszervezzük egy külön IRepository interfészbe - Javasolt: IRepository<T>, és ehhez kapcsolódó entity-specifikus leszármazottak - Generikus repository implementáció a CRUD műveletekhez MySongShop.Logic - Több ILogic interfész létrehozása, amiben definiáljuk a konzol által később hívható üzleti logikai műveletek listáját (ez az összes CRUD és nem-CRUD művelet, amit majd a Console App el tud érni) - Néhány Logic osztály implementálása; jelenleg legyen kész az összes lista művelet, ami belül Repository metódust hív meg - A többi műveletet hagyjuk üresen - A Logic réteg NEM férhet hozzá DbContext leszármazottakhoz, konstruktor-paraméterként egy (vagy több) repository-t vár, interfész típusú paraméterekként MySongShop.Program - Valamilyen egyszerű menü vezérelje - Jelenleg csak "Ez még nincs kész" üzenet bármely nem-kész menüelemet kiválasztva - Az összes tábla listázása legyen elérhető - Minden DbContext/Repository/Logic példányosítás itt történik, **kötelezően** egy Factory osztályon keresztül
Ápr.24. 23:55	CRUD + non-CRUD funkciók	MySongShop.Logic - Minden CRUD és nem-CRUD művelet elkészítése - A műveletek eredménye és paramétere típusos érték / gyűjtemény! - A lekérdezések nem használhatják a DbContext metódusait, csak a repository adatlekérő metódusait - A repository-tól visszakapott IQueryable adatokat fűzi tovább bonyolultabb lekérdezéssé - Non-Crud művelet NEM LEHET a repository layerben, a Logic több repository példány is kaphat konstruktor paraméterként - A Logic osztályokat NEM SZABAD funkciócsoport szerint darabolni (NINCS CrudLogic, NonCrudLogic, CreateLogic, ReaderLogic). A Logic osztályokat NEM SZABAD entity-k szerint darabolni (NINCS ProductLogic, CarLogic). Próbáljunk valamilyen üzleti központú feldarabolást megtalálni! MySongShop.Program - Mindegyik entity listázása, egyszerű hozzáadás, törlés, és valamilyen egyszerű módosítás (**teljes CRUD**) - Legyen minimum három olyan adatlekérés is, ami több táblát/entitást használ fel, és közben valamilyen tábla-csatolást (és esetleg csoportosítást is) használ (**minimum 3 nem-CRUD**

		metódus)
Máj.08. 23:55	Féléves feladat vége	MySongShop.Logic - A nem-CRUD műveletekből kell egy hétköznapi xxxDoSomething() és egy xxxDoSomethingAsync() változat is. A hétköznapi metódus valami gyűjteményt adjon vissza (pl. List<SomeResult>), a másik metódus pedig egy Task-ot (pl. Task<List<SomeResult>>). - A programban lévő menüből mindkettő változat legyen elérhető, és ugyanúgy jelenítsen meg eredményt. - Az "xxxDoSomethingAsync()" verziója a non-crud műveleteknek nem kell az async/await kulcsszavakat használniuk. Egyszerűen "return Task.Run(xxxx)" vagy "return Task.Factory.StartNew(xxxx)", segítségével az xxxx helyére kell behelyezni a szokványos non-crud művelet meghívását. Ezt követően a Console App-ban .Wait() és/vagy .Result használata kell a Task eredményének feldolgozásához MySongShop.Logic.Tests - CRUD műveletek tesztelése mockolt repository-val, Assert-ek nélkül, csak Moq.Verify() segítségével - Nem-CRUD műveletek tesztelése mockolt repository-val, nUnit Assert-ek ÉS Moq.Verify() segítségével - Mock.Object.XXX() közvetlen hívása TILOS Repository Freeze: Május 8. 23:55 - Git-en keresztül leadandó a teljes, szabályoknak megfelelő forráskód; szintén a Git-en belül a Doxygen fejlesztői dokumentációval; valamint a Moodle-n keresztül egy rövid, 1-2 perces videóval, amin belül a hallgató futás közben bemutatja a program használatát és az összes menüelem működését. - Elvárás a Gitstat: true - Szóbeli védés az ezt követő héten