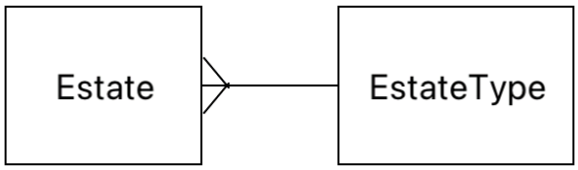
Informační systém realitního makléře

Makléř realitní kanceláře pracuje s nemovitostmi. Nemovitosti mají svoji polohu, adresu, zapisuje se k nim počet budov, počet obytných místností, počet pater, cena dané nemovitosti, dále to, zdali jde o novostavbu, a také údaj o tom, o jaký typ nemovitosti se jedná.

Naprogramujte aplikační, datovou, ORM a databázovou vrstvu aplikace, ve které by si realitní makléř mohl vést evidenci svých nemovitostí, o které se stará a snaží se je prodat nebo pronajmout. Místo prezentační vrstvy přidáte pouze testovací třídu s metodami, které ověří funkčnost aplikační vrstvy.

V programu budou dva business objekty – Estate (spravovaná nemovitost) a EstateType (typ nemovitosti). Vazba mezi nimi bude N : 1, tj. několik různých nemovitostí může být téhož typu, ale každá nemovitost má právě jeden typ nemovitosti.



Vlastnosti business objektů Estate a EstateType jsou popsány níže.

Estate

Vlastnost	Typ dat	Popis
GPS	text	GPS souřadnice místa, kde se nemovitost nachází
AddressStreet	text	Ulice a číslo z adresy, kde se nemovitost nachází
AddressCity	text	Město, kde se nemovitost nachází
AddressZip	celé číslo	PSČ z adresy, kde se nemovitost nachází
BuildingCount	celé číslo	Počet budov patřících k nemovitosti
RoomCount	celé číslo	Počet obytných místností
FloorCount	celé číslo	Počet pater
Price	desetinné číslo	Cena nemovitosti v CZK
IsNewBuild	ano/ne	True = novostavba, false = starší, používaná nemovitost
EstateType	EstateType	Typ nemovitosti (viz další business objekt)

EstateType

Vlastnost	Typ dat	Popis
Code	text	Kód typu nemovitosti
Description	text	Popis daného typu nemovitosti

Všechny vlastnosti business objektů musí být vždy před uložením do databáze vyplněny (všechny jsou povinné, není přípustné null, nula apod.)

Typ VS projektu

Doporučený typ projektu ve Visual Studiu je konzolový projekt v technologii .NET Core. Závazné je, aby byl celý projekt (až na SQL skripty a konfigurační soubory) naprogramován v C#.

Business objekty

Definujte v projektu třídy, které budou odpovídat výše uvedeným business objektům Estate a EstateType.

Databázová vrstva

Připravte SQL skripty pro databázové tabulky, které budou odpovídat business objektům Estate a EstateType. Doplňte sloupce potřebné pro primární (ID-čka) a cizí klíče.

Všechny sloupce v obou tabulkách jsou povinné (NOT NULL).

Poznámka: SQL skripty (a vlastně i příslušné tabulky) je možné i nechat vygenerovat (tzv. "code-first design"). V tom případě ale musíte mít správně připravené business objekty (včetně vlastností pro primární a cizí klíče) a správně nakonfigurovanou ORM vrstvu (viz dále).

ORM vrstva

Definujte v programu ORM vrstvu. Použijte EntityFramework (EF). Definujte třídu, která dědí ze třídy DbContext a má vlastnosti typu DbSet<T> pro každý business objekt.

Databázi můžete zvolit libovolně (doporučené jsou MS SQL Server nebo MySQL/MariaDB). Použijte anotace na business objektech (atributy Table, Column, Key, ForeignKey apod.). Definujte vlastnosti business objektů tak, aby EF použil při načítání dat z databáze tzv. "lazy loading".

Datová vrstva

Pro každý business objekt definujte jeho odpovídající repo-třídu. V každé repo-třídě budou minimálně metody pro tzv. CRUD operace:

- Metoda, která pro dané ID vrátí objekt s tímto ID se všemi jeho vlastnostmi načtenými z databáze (R = read)
- Metoda, která pro předaný objekt (bez nastaveného ID) přidá v databázi v příslušné tabulce nový řádek a použije hodnoty vlastností z předaného business objektu pro příslušné sloupce. Metoda vrátí objekt téhož typu s týmiž vlastnostmi a s nastaveným IDčkem, které bylo novému řádku přiděleno v databázi (C = create)
- Metoda, která pro předaný objekt (s nastaveným ID) provede v databázi aktualizaci řádku s daným ID podle hodnot vlastností v předaném business objektu (U = update)
- Metoda, která pro dané ID smaže z databáze příslušný řádek (D = delete)

Repo-třída bude mít navíc metodu, která vrátí všechny řádky databázové tabulky jako kolekci (IList<T>) příslušných business objektů.

Repo-třída může mít i další metody, které byste mohli potřebovat v aplikační vrstvě.

Aplikační vrstva

V aplikační vrstvě bude jediná servisní třída pro business objekt Estate. Bude mít metody podobné metodám v datové vrstvě:

- Metoda pro načtení dat z databáze o jednom objektu typu Estate podle předaného ID
- Metoda pro uložení dat o jednom objektu typu Estate do databáze. Podle toho, zda je v objektu nastaveno ID, či nikoli, se tato metoda rozhodne, zda provede operaci Create, tj. přidání nového řádku do databáze, nebo operaci Update, tj. aktualizaci existujícího řádku v databázi
- Metoda pro smazání jednoho řádku v databázi
- Metoda pro načtení dat o všech objektech typu Estate, které jsou v databázi. Vrátí kolekci těchto objektů

Servisní třída bude mít v sobě závislosti na obou repo-třídách datové vrstvy (pro business objekt Estate i pro EstateType).

Data pro typ nemovitosti

Databázové tabulce, která odpovídá business objektu EstateType, říkáme někdy "kódová tabulka" neboli "číselník". Číselníky jsou někdy neměnné (statické), jindy je možné do nich přidávat řádky, upravovat řádky a mazat řádky podobně jako v tabulkách normálních business objektů.

Číselník pro typ nemovitosti bude statický, data v něm se nemohou měnit. Abychom tam nějaká data použitelná pro naši aplikaci měli, je potřeba připravit SQL skript, který tabulku pro EstateType naplní. Je požadováno aspoň 6 typů nemovitostí. Doporučené jsou např. tyto:

Číselník pro EstateType

Kód typu nemovitosti	Popis
PP	Pouze pozemek
PZ	Pozemek se zahradou
D	Dům

Kód typu nemovitosti	Popis
DZ	Dům se zahradou
V	Vila
P	Palác
M	Mrakodrap
TB	Tovární budovy
SC	Shopping center
SO	Soukromý ostrov

Testovací třída

Testovací třída (napište buď obyčejnou třídu v konzolové aplikaci, nebo použijte testovací framework, např. NUnit) bude mít několik metod, které ověří, že výše popsané části aplikace (konkrétně její aplikační vrstva se servisní třídou) fungují správně.

Předpokládejme, že v servisní třídě **EstateService** budou tyto metody:

- Estate **GetEstate**(int id) – načtení dat o jedné nemovitosti
- Estate **SaveEstate**(Estate estate) – uložení dat o jedné nemovitosti (přidání nové, případně update existující)
- void **DeleteEstate**(int id) – smazání dat o jedné nemovitosti
- IList<Estate> **GetEstates**() – načtení dat o všech nemovitostech do kolekce objektů typu Estate

V testovací třídě **EstateServiceTest** budou metody, jejichž postupným zavoláním ověříme funkčnost třídy **EstateService**:

- Metoda **SaveEstateTest** otestuje servisní metodu SaveEstate tím, že uloží do databáze alespoň čtyři různé nemovitosti. Některé budou mít stejný typ nemovitosti, jiné se budou v typu nemovitosti lišit.
- Metoda **GetEstateTest** zkusí načíst první dvě nemovitosti pomocí servisní metody GetEstate (podle předpokládaných ID, případně si vypomůže tím, že napřed zavolá GetEstates) a ověří, že hodnoty vlastností u načtených nemovitostí odpovídají právě těm, které byly použité v metodě SaveEstateTest při vkládání řádků do databáze
- Metoda **DeleteEstateTest** zkusí pomocí DeleteEstate smazat třetí nemovitost a ověří, že celkový počet nemovitostí před smazáním a po smazání se o jednu liší (pro ověření použije servisní metodu GetEstates)
- Metoda **GetEstatesTest** otestuje servisní metodu GetEstates tím, že zkontroluje, že tato metoda vrací kolekci, v níž je předpokládaný počet prvků (odpovídající aktuálnímu počtu řádků v tabulce s nemovitostmi). Protože se pro celý test zavolají testovací metody jedna za druhou, musí tato metoda vzít v úvahu, že předchozí metoda (DeleteEstateTest) jednu nemovitost smazala.
- Metoda **CleanUpAfterTests** pouze uvede databázi do původního stavu (tj. smaže všechny nemovitosti, které tam ještě zbyly) kvůli tomu, aby následující spuštění sady testů dávalo stejné výsledky.

Všechny metody budou mít stejnou signaturu, pokud jde o parametry a návratovou hodnotu:
- nebudou mít žádné parametry
- nebudou nic vracet (void)

Pokud v některém z testů zjistíte, že předpokládaná data neodpovídají skutečným (např. u GetEstatesTest očekáváte, že počet nemovitostí v kolekci, kterou vám vrátí servisní metoda

GetEstates, je 3, ale z databáze se vám místo toho vrátí 4), musíte vyhodit výjimku. Do textové hlášky výjimky dejte nějaký krátký, ale výstižný text, který popisuje zjištěnou chybu, např. "Počet objektů vrácených z GetEstates neodpovídá očekávanému počtu".

V testovací třídě (pokud nepoužíváte framework pro unit testy) bude ještě jedna (public) metoda, která zavolá postupně jednu za druhou všechny ostatní testovací metody (viz výše). Tuto public metodu zavoláte z metody Main ve třídě Program (z hlavního vstupního bodu celé aplikace).