ICS projekt

Důležité upozornění

Pro hodnocení projektu (a úspěšné absolvování předmětu) je nutno dokončit **všechny 3 fáze projektu** a projekt **obhájit**. Pokud projekt nebude při obhajobě obsahovat základní funkcionalitu uvedenou v zadání, bude hodnocen 0 body. **Nespokojíme se tedy s nedokončeným projektem**. Tuhle poznámku sem dáváme proto, že se v předchozích ročnících vyskytly týmy, které po dosáhnutí součtu 50 bodů za předmět po 2. fázi rozhodly nedokončit projekt a poté byly nemile překvapeni, když se po nich vyžadovala plná funkcionalita při obhajobě. Dejte si na to tedy prosím pozor.

Cíl

Cílem je vytvořit použitelnou a snadno rozšiřitelnou aplikaci, která splňuje požadavky zadání. Aplikace nemá padat nebo zamrzávat, pokud uživatel vyplní něco špatně, upozorní ho validační hláškou.
Zadání úmyslně není striktní, je Vám ponechána volnost, pro vlastní realizaci. Při hodnocení je kladen důraz na technické zpracování a kvalitu kódu, ale hodnotíme i použitelnost a grafické zpracování aplikace. Pokud Vám přijde, že v zadání chybí nějaká funkcionalita, neváhejte ji doplnit. Pište aplikaci tak, aby jste ji sami chtěli používat.

Zadání - Aplikace pro správu filmů

Výsledná aplikace má sloužit pro správu filmové kolekce. Pro zjednodušení si můžete představit, že vytváříte jednodušší desktopovou verzi aplikace k webu jako IMDB nebo ČSFD.

Data

V rámci dat, se kterými se bude pracovat budeme požadovat minimálně následující data.

Film

- Originální název
- Název česky
- Žánr
- Titulní fotografie
- Země původu
- Délka filmu
- Textový popis filmu
- Seznam režisérů
- Seznam herců
- Seznam hodnocení filmu

Osoba (herec/režisér)

- Jméno
- Příjmení
- Věk
- Fotografie
- Seznam filmů ve kterých hrál
- Seznam filmů, které režíroval

Hodnocení filmu

- Číselné hodnocení
- Textové hodnocení

Funkcionalita

Aplikace bude obsahovat několik pohledů pro zobrazování a zadávání dat.
Je požadováno **perzistentní** uložení dat. To znamená, že když se aplikace restartuje, tak nesmí o data přijít. Je nutno data ukládat za běhu aplikace, aby bylo možno demonstrovat, že

když se například pomocí aplikace přidá nový film, tak se tento film zobrazí v seznamu filmů (a podobně pro ostatní data).
Pro uložení zvolte (SQL databázi), kterou zpřístupníte pomocí Entity Framework Core 3.1.
Minimální rozsah, který je požadován v rámci projektu je popsán v této kapitole.

Seznam filmů

Seznam bude obsahovat všechny filmy dostupné v aplikaci. Bude možno se z něj překliknout na detail filmu a na pohled pro přidání nového filmu.

Detail filmu

Pohled zobrazuje detail jednotlivého filmu se všemi informacemi o filmu (viz kapitolu Data). Na stránce se také dá přidávat nové hodnocení filmu a zobrazuje se průměrná číselná hodnota hodnocení a textové popisy jednotlivých existujících hodnocení.

Editace filmu

Pohled, který slouží na editaci filmu. Může se využít na vytvoření nového filmu nebo na editaci existujícího filmu. Bude obsahovat všechny informace o filmu včetně výběru herců a režisérů (viz kapitola Data).

Seznam osob (herců/režisérů)

Pohled obsahuje všechny osoby. Bude možno se z něj překliknout na detail osoby a na pohled pro přidání nové osoby

Detail osoby

Detail osoby - stránka zobrazuje všechny informace o konkrétní osobě včetně seznamu filmů, ve kterých hrála a které režírovala (viz kapitola Data).

Editace osoby

Pohled, který slouží na editaci osoby. Může se využít na vytvoření nové osoby nebo na editaci existující osoby. Bude obsahovat všechny informace o osobě včetně filmů, ve kterých hrála a které režírovala (viz kapitola Data).

"Vyhledávání"

Pohled, na kterém můžete použít textové vyhledávání napříč záznamy v aplikaci. Seznam všech nalezených záznamů se zobrazí na stránce a bude se dát překlikem dostat na detail daného záznamu (v případě hodnocení se odnaviguje na detail filmu, který k hodnocení přislouchá). Textově se vyhledává minimálně v těchto atributech:

- Film
  - Originální název
  - Název česky
  - Země původu
  - Textový obsah filmu
- Osoba
  - Jméno
  - Příjmení
- Hodnocení
  - Textové hodnocení

Správa projektu - Azure DevOps

Při řešení projektu týmy využívají Azure DevOps a využívají GIT na sdílení kódu. Do svého projektu přidělte přístup vyučujícím (způsob bude vysvětlen v rámci 1. cvičení); tj. do Vašeho týmového projektu si v části Members přidejte účet **uciteliw5@vutbr.cz**
Účet **uciteliw5@vutbr.cz** budou používat vyučující pro přístup k odevzdávaným souborům. Bez přidání tohoto účtu není možné přistoupit k vašemu projektu a tedy není možné jej ze strany vyučujících hodnotit.
Návod na přidání člena projektu můžete najít zde: *https://docs.microsoft.com/en-us/vsts/accounts/add-team-members-vs*

Z GITu *musí být viditelná postupná práce na projektu a spolupráce týmu*. Pokud uvidíme, že existuje malé množství nelogických a nepřeložitelných commitů tak nás bude zajímat, jak jste spolupracovali a může to vést na snížení bodového hodnocení. Organizaci pojmenujte **ics-2020-team<0000>** dle Vašeho čísla týmu a projekt **project** tak, že výsledné URL pro přístup pro tento imaginární tým by bylo https://dev.azure.com/ics-2020-team0000/project. Nezapomeňte nastavit **Work item process** template na **Scrum**.

Doporučení - za bonusové body při závěrečné obhajobě

- Pro řízení projektu využijte metodologii **Scrum**.
- Plánujte sprinty na jednotlivé fáze odevzdání. Práci rozdělte minimálně na **Product Backlog Item (PBI), Tasks a Bugs**. Vyžijete záložky **Boards** pro vzájemnou synchronizaci a **Burndown chart** bude na konci každého sprintu, tj. při každém odevzdání, reflektovat reálný stav projektu.
- Využijte možnost automatizovaných buildů spojených s otestováním Vámi provedených změn. Nastavte **Pipelines->Builds** tak, že při pushnutí do libovolné větve projektu se provede *build a spustí se veškeré přítomné testy*. Více informací na Automate all things with Azure Pipelines - THR2101

Odevzdávání

Odevzdávání projektu má **3 fáze**. V každé fázi se hodnotí jiné vlastnosti projektu. Nicméně fáze na sebe navzájem následují a studenti pokračují v práci na svém kódu i po jeho odevzdání v rámci následující fáze.
Kontroluje se kód, který je nahrán v GIT ve větvi master. Vždy se kontroluje **poslední commit před časem odevzdávání** dané fáze projektu. Na commity nahrány po času odevzdávání nebo v jiných větvích nebude brán zřetel. Pokud commit, který máme hodnotit otagujete, např. v1, v2, v3, usnadníte nám orietaci při hodnocení.
Je silně doporučováno projekty v průběhu semestru konzultovat po přednášce/cvičení, předejdete tak případným komplikacím při odevzdání.

Fáze 1 – objektový návrh

V téhle fázi se zaměříme na *datový návrh*. Vyžaduje se po Vás, aby datový návrh splňoval zadání a nevynechal žádnou podstatnou část. Zamyslete se nad vazbami mezi jednotlivými entitami v datovém modelu. V následující fázi budete entity nahrávat do databáze, takže myslete na jejich propojení již nyní. V této fázi budeme chtít, abyste **odevzdali kód**, kde budete mít *entitní třídy*, které budou obsahovat všechny vlastnosti, které budete dále potřebovat a vazby mezi třídami. **Nestačí tedy odevzdat diagram tříd, nebo nějakou jinou reprezentaci.** Budeme požadovat kód v jazyce C#.
Hodnotíme:

- logický návrh tříd
- využití abstrakce, zapouzdření, polymorfismu - kde to bude dávat smysl a eliminovat duplicity
- verzování v GITu po logických částech
- logické rozšíření datového návrhu nad rámec zadání (bonusové body)

Fáze 2 – databáze, repozitáře a mapování

Vytvořte napojení datových tříd pomocí Entity Frameworku na databázi.
Vytvořte tedy repozitářovou (Repository) vrstvu, která zapouzdří databázové entity a zpřístupní pouze data přemapovaná do modelů/DTO.
Pokud chcete Vaši aplikaci rozšířit, můžete nechat repozitářovou vrstvu pracovat s entitami a postavit nad ní ještě fasádu (Facade), která bude využívat repositář a entity přemapovávat do modelů/DTO v podobě listového model (obsahuje pouze data, která se hodí pro zobrazení v seznamu) nebo detailu.

Protože nemáte zatím UI, funkčnost aplikace ověřte automatizovanými testy! Kde to dává logický smysl tvořte **UnitTesty**, pro propojení s databází vytvářejte **Integrační testy**. Pro všechny typy testů využijte libovolný framework, doporučujeme **xUnit**.
Dbejte také kvality Vašeho kódu. Od této fáze se hodnotí i tenhle atribut. Opravte si tedy předchozí kód dle zásad Clean Code a S.O.L.I.D. probíraných na přednášce/cvičení a důsledně je dodržujte. Můžete si dopomoct např. rozšířením **Code Metrices**.
Hodnotíme:

- využití **Entity Framework Core (EF) Code First** na vytvoření databáze z tříd navržených ve fázi 1
- návrh a funkčnost repozitářů
- čistotu kódu
- pokrytí aplikace testy - ukážete tím, že repozitáře opravdu fungují
- dejte pozor na zapouzdření databázových entit pod vrstvou repozitáře (fasády), který je nepropaguje výše, ale přemapovává na modely/DTO

Fáze 3 – WPF frontend, data binding

V této fázi se od Vás již požaduje vytvoření WPF aplikace. Napište backend aplikace (vytvoření View-Modelů), která bude napojena na Vámi navržené datové modely z 2. fáze, které jsou zapouzdřeny v repozitáři/fasádě. A dále frontend (View), která bude zobrazovat data předpřipravená ve view-modelech. Zamyslete se nad tím, jakým způsobem je vhodné jednotlivá data zobrazovat.
Využijte *binding* v XAML kódu (vyvarujte se code-behind). Účelem není jenom udělat aplikaci, která funguje, ale také aplikaci, která je správně navržena a může být dále jednoduše upravitelná a rozšiřitelná. Dbejte tedy zásad probíraných ve cvičeních.
Za aplikace, jejichž vizuální návrh bude proveden dobře, a zároveň budou plně funkční, budeme udělovat také bonusové body.
Hodnotíme:

- funkčnost celé výsledné aplikace
- vytvoření View, View-Modely
- zobrazení jednotlivých informací dle zadání – seznam, detail…
- správné využití data-bindingu v XAML
- čistotu kódu

Doporučujeme (bonusové body):

- pokrytí view-modlů testy
- vytvoření dobře vypadající a plně funkční aplikace
- plánování projektu (logická struktura rozložení práce)
- nastavení automatizovaného buildu (kód je přeložitelný, spouští se automatizované testy, pipeline nekončí chybou)

Obhajoba

Obhajoby projektů budou probíhat v **posledních 2 týdnech** semestru. Termíny obhajob budou vyhlášeny v průběhu semestru.
Na obhajobu se dostaví **celý tým**. Z členů týmu bude cvičícími vybrán 1 student, který obhajobu povede. Na obhajobu **není nutné** mít prezentaci (Powerpoint nebo pdf). Budete nám muset ukázat, jak funguje váš kód, že je správně navržen. Připravte se na naše otázky k funkcionalitě jednotlivých tříd a k důvodům jejich členění. Na obhajobu bude mít tým 10-15 minut.