Naloga 4 - Spletne storitve s podatkovno bazo

Nalogo 3 nadgradite tako da vključite delo s **podatkovno bazo (Entity Framework - Code First)**. Vsi podatki, ki ste jih do sedaj statično hranili, se sedaj naj nahajajo v podatkovni bazi. Prav tako dodajte naslednje funkcionalnosti za upravljanje s podatki:

- Dodajanje novih primerkov vseh entitet (npr. uporabnikov, klubov, igralcev)
- Urejanje primerkov vseh entitet
- Brisanje primerkov vseh entitet
- Dodajanje obstoječih primerkov ene entitete k drugi (npr. dodajanje obstoječega igralca klubu)
- Odstranjevanje povezav med entitetami (npr. odstranjevanje igralca iz kluba)

Vse metode naj vračajo uspešnost izvedene storitve. Prav tako je za vse funkcionalnosti potrebno nadgraditi odjemalca.

Pri uporabi EntityFramework in spletnih storitev, se zaradi virtualnih spremenljivk pojavlja tudi napaka "The underlying connection was closed: The connection was closed unexpectedly.", rešitev najdete tukaj.

Pri implementaciji si lahko pomagata se sledečim primerom.

Podrobnosti o entityframeworku najdete na naslednji povezavi.

Naloga 6 - Aplikacija z ogrodjem WPF (Windows Presentation Foundation)

Z ogrodjem Windows Presentation Foundation (WPF) izdelajte aplikacijo, ki bo klicala spletne storitve, definirane v prejšnjih nalogah. V kolikor le-teh niste implementirali, lahko poslovno logiko aplikacije implementirate tudi znotraj projekta (brez klicanja storitev) (2/3 točk).

Tudi tukaj veljajo enaka pravila glede same izvedbe dodajanja, brisanj, urejanja posameznih entitet (pravilno osveževanje) ter funkcionalnosti uporabniškega vmesnika. Še posebej bodite pozorni, da brisanje, urejanje itd. ni realizirano z vnašanjem ID-jev oz. ključnih besed v vnosna polja.

Vsi, ki ste imeli težave s podatkovno bazo, lahko podatke hranite v tekstovni datoteki, xml datoteki, drugem podatkovnem viru ali v statičnih seznamih (brisanje in urejanje se v času izvajanja mora obnašati pravilno).

Poskrbite, da bodo v vašem programu uporabljene tudi komponente:

- WrapPanel
- lastni Window
- UserControl
- GroupBox
- ToolBar
- CheckBox

Naloga 7 - Nadgrajevanje uporabniških vmesnikov (WPF ali Forms)

Nadgradite uporabniške vmesnike, ki ste jih razvijali v okviru 5. ali 6. naloge, ali bi raje nadgrajevali aplikacijo WIndows Forms ali WPF se odločite sami. V kvizu bosta po **1-2 vprašanji iz vsakih zahtev, tako, da si pred kvizom pomagajte**.

Skupne zahteve (implementirate obojni):

- Omogočite označevanje posameznih elementov entitet (nujno, da deluje za vsaj eno tabelo v bazi) in njihov izvoz v CSV, v poljubno mesto na disku. Za shranjevanje (dialog) uporabite (v primeru WPF - Microsoft.Win32, v primeru Windows Forms - System.Windows.Forms) in poljuben način pisanja v datoteko.
- Omogočite prilagajanje velikosti oken. Ob večanju in manjšanju okna se naj spremenijo tudi velikosti in razporeditev gumbov in gradnikov (*responsive design*). Poskrbite, da so spremembe smiselne.
- V oknu ali kontroli za dodajanje novega primerka (za vse entitete) dodajte kontrolo Progress Bar, ki se bo polnila proporcionalno glede na število izpolnjenih polij v obrazcu, ko bodo vsi podatki izpolnjeni 100%, ko je izpolnjenih polovica podatkov 50%, ...
- Zamenjajte privzete pisave in barve (na nek relativno estetski način, v kolikor še tega niste storili)
- Nad elementi implementirajte vsaj 2 dogodka, ki nista tipa (Click)

Zahteve WPF:

- Smiselno uporabite Slider, Tab Control in StackPanel* (samo če niste uporabili template, ki je na Eštudiju **OsebaEditor**, če ste ga uporabite ScrollBar ali ScrollView)

Zahteve Windows Forms:

- Smiselno uporabite ToolTip, RadioButton in TabControl

Naloga 8 - ASP.NET Web API

Izdelajte spletne storitve v ASP.NET, (ASP.NET Core web application -> API) ki bo temeljila na vaših entitetah in bo omogočala

Storitve morajo (razen kjer je specifično zapisano) biti razvite za vse entitete.

Storitve entitet (implementirate za vsako entiteto posebej):

- Pridobivanje vseh objektov (GET entitete)
- Pridobivanje objekta po id (GET entitete/id)
- Dodajanje novega objekta (POST entitete)
- Odstranjevanje objekta po id (DELETE entitete)
- Posodabljanje entitete po id (PUT entitete/id)

Dodatne storitve (implementirajte na 1 entiteti, ki ni mnogo mnogo):

- iskanje objektov entitet tako, da storitvi pošljete objekt entitete v katerem so manjkajoči podatki
  - npr. entiteta Država (id, imeDrzave, kontinent, clanicaNato)
  - v storitev bi poslali primerek entitete Država, ki bi imel vnešen le kontinent vi pa bi vrnili vse primerke držav, ki se nahajajo na kontinentu
  - isto storitev bi lahko uporabili tudi tako, da bi vanjo poslali zgolj podatek ali je članicaNato ali ne...
  - Storitev če vanjo pošljete entiteto, ki nima definiranega ničesar vrača vse primerke
  - Lahko pa tudi pošljete zgolj 2 atributa, ... itd.
- in še 2 storitvi, ki si jih izmislite **sami!,** in nista **pre-enostavni** in sta **smiselni**

Poskrbite, da boste uporabljali dobre prakse razvoja REST storitev predvsem:

- Pravilna uporaba metod GET, POST, PUT, DELETE *(PATCH ni potrebno implementirati!)*
- Pravilno poimenovanje končnih točk
- Konsistentnost rabe in poimenovanja vaših storitev

Pripravite tudi demonstracijo **vseh** vaših storitev z orodjem Postman.

Orodje Postman se uporablja za izvajanje in hranjenje REST zahtevkov in olajša prototipiranje APIjev. Vodič za uporabo orodja Postman si lahko ogledate na **povezavi** in **povezavi**.

Vse operacijami nad entitetami morejo trajno hraniti spremembe. Tehnologija, s katero boste **trajno** shranjevali podatke je lahko poljubna (Code First, LINQ to SQL, XML serializacija, tekstovna datoteka,...).

Naloga 9 - Dokumentiranje REST vmesnikov z ogrodjem Swagger

Swagger je odprtokodno programsko ogrodje, ki pomaga razvijalcem oblikovati, graditi, dokumentirati in uporabljati RESTful spletne storitve.

Glavni del ogrodja Swagger je orodje Swagger UI, ki vključuje podporo za avtomatizirano dokumentacijo, ustvarjanje kode in generiranje testnih primerov. Generirani dokumenti JSON temeljijo na OpenAPI dokumentaciji.

Specifikacija OpenAPI (prej Swagger Specification) je format za opisovanje REST APIjev. JSON datoteka OpenAPI omogoča, opis celotnega APIja:

- Endpoints - Razpoložljive končne točke (/ uporabniki) in operacije na vsaki končni točki (GET / uporabniki, POST / uporabniki)
- Parametri operacij vhodi in izhodi vsake operacije
- Načini avtentikacije
- Kontaktni podatki, licenca, pogoji uporabe in druge informacije upravljalca APIja

Naloga

Izdelajte dokumentacijo predhodno izdelanih WEB API spletnih vmesnikov z uporabo orodja Swagger.

Za avtomatsko generiranje kode uporabite NuGet paket Swashbuckle.

Poskrbite, da bodo vaši vmesniki vsebovali kratke opise za vse metode, ki jih vaš vmesnik ponuja:

- <summary> za vsako metodo
- <remarks> za vsaj 1 metodo
- generiran kontakt in opis APIja (c.SwaggerDoc("v1", new OpenApiInfo { ...)
- <response> za vsaj eno metodo