BBIT iestājpārbaudījums: Full-stack programmētājs

Šie ir standarta uzdevumi visiem kandidātiem, lai pārbaudītu programmēšanas iemaņas un lai Jūs varētu iepazīties ar tehnoloģijām ar kurām mēs ikdienā strādājām. Uzdevumiem ir vairāki etapi, no kuriem pirmie divi ir obligāti. Uzdevumu izpildei būs nepieciešams instalēt:

1. Visual Studio Community Edition (https://visualstudio.microsoft.com/vs/community/)

1. uzdevums

Būs jāizveido *Console Application*, kas dara sekojošo:

- izveido ciparu masīvu 20x20
- aizpilda to ar gadījuma cipariem
- izvada masīvu uz ekrāna (ievērojot atstarpes starp elementiem)
- pēc tām atrod masīvā mazāko un lielāko ciparu koordinātes un izvada tos ar koordinātēm uz ekrāna
- tālāk atlasa masīvu augošā secībā izvada atlasītu masīvu uz ekrāna

2. uzdevums

Nepieciešams izveidot REST API, izmantojot ASP.NET Core (https://docs.microsoft.com/en-us/aspnet/core/tutorials/first-web-api). Šajā etapā ir pieļaujams izmantot *in-memory database*.

Ir trīs datu tipi: Māja, Dzīvoklis un Iedzīvotājs.

Tipam "**Māja**" ir sekojošas īpašības:

- Numurs
- Iela
- Pilsēta
- Valsts
- Pasta indekss

Tipam "**Dzīvoklis**" ir sekojošas īpašības:

- Numurs
- Stāvs
- Istabu skaits
- Iedzīvotāju skaits
- Pilna platība
- Dzīvojama platība
- Saikne ar māju, kur atrodas konkrēts dzīvoklis

Tipam "**Iedzīvotājs"** ir sekojošas īpašības:

- Vārds
- Uzvārds
- Personas kods
- Dzimšanas datums
- Telefons

- E-mail
- Saikne ar dzīvokli, kur dzīvo iedzīvotājs

API jāatbalsta sekojošas operācijas:
- Izveidoto objektu iegūšana
- Objektu izveidošana
- Objektu rediģēšana
- Objektu dzešana

3. uzdevums

Izmantojot 2.uzdevumā radītās CRUD operācijas, pielāgot risinājumu reālās dzīves scenārijiem.

1. Nepieciešams izņemt datu apstrādes loģiku no kontrolieriem un pārvietot to uz servisiem.
2. Nepieciešams pieslēgt SQL datubāzi
3. Nepieciešams izveidot noklusējuma objektu datubāzē (*database seed*).
    a. Izveidot divas mājas
    b. Izveidot 5 dzīvokļus (kopā)
    c. Izveidot 4 iedzīvotājus (kopā)
4. Katram modelim uzrakstīt datubāzes konfigurāciju.
5. Paplašināt attiecības starp modeļiem, ievērojot zemāk aprakstītos nosacījumus:
    a. Vienam iedzīvotājam var būt vairāki dzīvokļi.
    b. Vienam dzīvoklim var būt vairāki iedzīvotāji.
    c. Vienai mājai ir vairāki dzīvokļi, bet katram dzīvoklim ir tikai viena māja.
6. Pievienot “Iedzīvotāja” modelim jaunu *bool* lauku “IsOwner”
7. Migrēt datubāzi.
8. Radīt DTO modeļus
9. Pieslēgt AutoMapper.

4. uzdevums

Izmantojot 3.uzdevumā radīto risinājumu, izveidot *client-side* aplikāciju, kas ļauj redzēt un pārvaldīt datubāzes objektus caur lietotāja saskarni (*user interface*). Šī uzdevuma izpildei iesākam izmantot Visual Studio Code (https://code.visualstudio.com).

Izmantojot *Angular* un *Bootstrap 5*, izveidot skatus:

/all-houses
- Šajā skatā attēlot tabulu ar visām mājām, noklikšķinot uz māju, lietotājs tiek novirzīts uz detalizētu mājas skatu

/house/{id}
- Detalizēts mājas skats, kurā var veikt mājas rediģēšanu
- Var redzēt dzīvokļu sarakstu
- Noklikšķinot uz dzīvokli, lietotājs tiek pārvirzīts uz detalizētu dzīvokļa skatu

/apartment/{id}
- Detalizēts dzīvokļa skats, kurā var veikt dzīvokļa rediģēšanu
- Ir redzams iedzīvotāju saraksts
- Noklikšķinot uz iedzīvotāju, atveras modālais logs, lai rediģētu lietotāja datus

5. uzdevums

4.uzdevumam pievienot autentifikācijas metodi *OAuth 2.0*, kas ļauj lietotājiem ielogoties sistēmā, izmantojot lietotājvārdu un paroli. Sistēmai ir jāatbalsta divas lietotāju lomas ar atšķirīgām atļaujām. Tas nozīmē, ka ielogojoties ar dažādiem lietotājiem, būs redzama atšķirīga informācija.

	Manager				**Resident**			
	Create	Update	Read	Delete	Create	Update	Read	Delete
House	+	+	+	+	-	-	+	-
Apartment	+	+	+	+	-	-	+	-
Resident	+	+	+	+	-	+	+	-