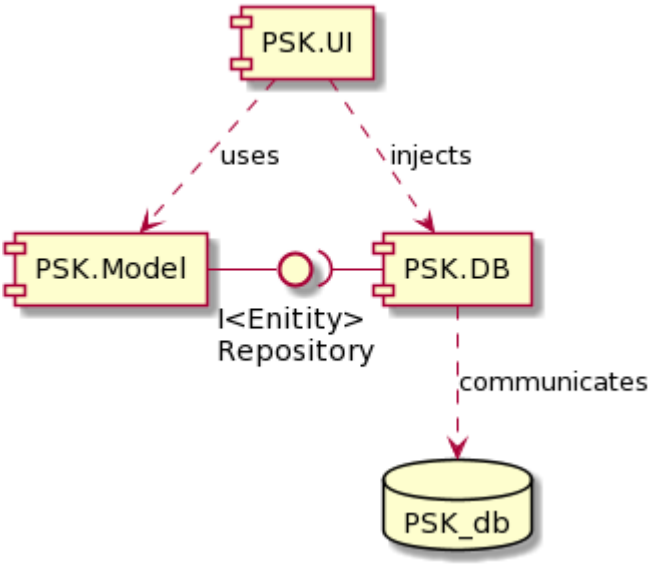
PSK Techninė ataskaita

Sistemos struktūra:



Sistema padalinta į 3 architektūrinius sluoksnius, kurie išdėstyti 3 atskiruose projektuose: PSK.UI (prezentacijos), PSK.Model (verslo logikos), PSK.DB (duomenų pasiekimo).

PSK.UI:

Pirminis sistemos .NET Core 3.1 WEB.AP projektas. Jo pagrindas parašytas C# kalba (Startup, Controllers), tačiau taip pat jame paleidžiamas React karkasas (Vartotojo sąsaja), kuriame naudojamos JavaScript ir HTML kalbos. PSK.UI konfigūracija valdoma iš app.settings.json ir app.config failų. PSK.UI projektas priklauso nuo kitų dviejų, nes jis naudoja PSK.Model klases ir joms įdiegia objektus iš PSK.DB projekto.

PSK.Model:

Antrinis sistemos .NET Core 3.1 Class Library projektas. Visas parašytas C# kalba. Jame laikomi verslo logiką apdorojantys servisai, tai pat įvairios pagalbinės klasės. Šis projektas yra naudojamas iš PSK.UI esančių kontrolerių.

PSK.DB:

Tretinis sistemos .NET Core 3.1 Class Library projektas. Visas parašytas C# kalba. Jame laikomos repositorijos, atsakingos už darbą su Mysql duomenų baze. Šiam darbui naudoja Entity Framework ORM. Dauguma kreipinių į duomenų bazę aprašyti naudojantis LINQ, tačiau yra ir tiesioginių SQL sakinių. Taip pat šiame projekte saugomos ir valdomos duomenų bazės migracijos. Šis projektas priklauso nuo PSK.Model, nes įgyvendina PSK.Model užduotus I<Entity>Repository interface‘us.

Kokybiniai (nefunkciniai) reikalavimai sistemai:

Concurrency: naudojame token login'ą ir autorizaciją. Token'as sukuriamas DB.Model, saugomas sesijoje ir duomenų bazėje ir sutikrinamas React'ui siunčiant requestą. Autorizacija yra pritaikyta globaliai visiems endpoint'ams (išskyrus registration ir login).

https://github.com/jurciusmantas/PSK/blob/master/PSK.Model/Authorization/TokenValidator.cs

https://github.com/jurciusmantas/PSK/blob/master/PSK.UII/Startup.cs 40-49 eil.

Security: sistemoje daugiausia duomenims gauti naudojami „Entity Framework" siūlomi LINQ sakiniai. Tačiau naudojamos ir kelios vietos, kur yra „Raw Sql" užklausos. Jose naudojami parametrai nėra „string" tipo, todėl negali sukelti Sql Injection. „Raw Sql" užklausas galima rasti:

https://github.com/jurciusmantas/PSK/blob/master/PSK.DB/SqlRepository/EmployeeSqlRepository.cs 86-94 eil.

https://github.com/jurciusmantas/PSK/blob/master/PSK.DB/SqlRepository/RestrictionSqlRepository.cs 74-81 eil.

Data Access: dauguma užklausų, kurios nereikalauja join'ų yra vykdomos naudojantis LINQ, o kur reikia naudojam SQL (žiūr. Security). Šios su duomenų baze bendraujančios klasės visos yra Request Scoped tipo, todėl visos transakcijos yra daugiausia request'o ilgio.

https://github.com/jurciusmantas/PSK/blob/master/PSK.UII/Startup.cs 126-133 eil.

Data consistency; Optimistic locking: update sakiniuose tikriname ar atitinka versija. Vartotojui iškyla pranešimas, kad bandytų vėl.

https://github.com/jurciusmantas/PSK/blob/master/PSK.Model/Services/TopicService.cs 165-178 eil.

Memory management: Use Case komponentai (servisai) yra Request scope tipo. Visi jų naudojami objektai taip pat request scope arba ne ilgesnio tipo, nes kitaip sukeltų LifeStyle Mismatch exception'ą (https://simpleinjector.readthedocs.io/en/latest/LifestyleMismatches.html).

Pavyzdys: https://github.com/jurciusmantas/PSK/blob/master/PSK.Model/ObjectContainer.cs visa klasė.

Reactive programming, asynchronous/non-blocking communication: vartotojo sąsajoje naudojamas React karkasas užtikrina šio reikalavimo įgyvendinimą.

Cross-cutting functionality/Interceptors: mūsų pasirinktas .NET Core karkasas nesuteikia Interceptoriaus (kuris yra Java EE) funkcionalumo, todėl mes pritaikėme Dekoratorių. Pavyzdys tokio dekoratoriaus:

https://github.com/jurciusmantas/PSK/blob/master/PSK.Model/Logging/LoginLoggingDecorator.cs

Išjungimas:

https://github.com/jurciusmantas/PSK/blob/master/PSK.UII/appsettings.json 2-5 eil.

Extensibility/Glass-box extensibility: servisam ir repositorijai mūsų pasirinktas SimpleInjector nepalaikė konfigūracijos iš XML ar Json, tačiau radome apėjimą. Tam, kad būtų pakeisti visi servisai arba repositorijos, reikia appsettings.json faile nurodyti aplanko, kuriame yra sukurtos naujos servisų arba repositorijų implementacijų bibliotekos, kelius ir nauja implementacija bus pritaikyta neperkopijuojant kodo.

Konfigūracijos keitimas

https://github.com/jurciusmantas/PSK/blob/master/PSK.UII/appsettings.json 6-9 eil.

Pakeitimai įdiegiami:

https://github.com/jurciusmantas/PSK/blob/master/PSK.Model/ObjectContainer.cs 47-84 eil.