Retsensioon veebiteenusele WatchWinders

Retsenseerib BestInShow

Otstarbekus

Meeskond WatchWindersi on teinud veebiteenuse, mis aitab jälgida kellakarpide kaudu mehhaaniliste käekellade seisundit ja annab kliendile ülevaade tema kollektsioonis olevatest kelladest. Tegemist on kindlasti väga huvitava probleemiga ja teenuse kasutamisest võiksid huvitatud olla päris paljud kollektsionäärid. WatchWinders kirjutab ise, et tervikliku probleemi lahenduseks oleks ka kellakarbi püsivara loomine, mida käesoleva õppeaine skoobi tõttu ei implementeerita, mis on ka täiesti loogiline. Projekti raames valmis saanud lahendusega saab kasutaja sisse ja välja logida, kellakarpe lisada, uut kella kellakarpi lisada, kella muuta, kella ja kellakarpi kustutada, muuta kellakarbi seadeid ja näha ülevaadet oma karpidest. Lisaks sellele näeb admin autoriseeritult logisid ning töötaja saab lisada ja muuta kontaktandmeid. Võib öelda, et tehtud lahendus on piisav esmaseks kellade halduseks ja see kergendab kollektsionääride tööd oma kellade hooldamisel.

Ülesehitus

API ülesehitus järgib koolis näidatud struktuuri. Domeenikihis asuvad domeeniklassid, kusjuures klassid on struktureeritud teemade kaupa erinevatesse kaustadesse, mis teeb vajaliku domeeni ülesotsimise lihtsaks. Kõikides domeenimudelites on väljadel vajalikud annotatsioonid. Data access layeri interface'i kihist leiab andmekihtide teoreetilise kirjelduse ja IUnitOfWork'i interface'i. Järgmisest data access layeri kihist EF leiab implementatsiooni entity frameworki jaoks esimese kihi interface'ist. Kogemata on sealt jäänud kustutamata UnitOfWorki fail, mille sisu on väljakommenteeritud ja tegelikult üldse ei peakski seal kihis olema. Kihis DAL.App.Interfaces on konkreetse API andmekihtide teoreetiline kirjeldus ja repode loomiseks mõeldud factory ja provider. Selles kihis olevas repositooriumide kaustas oleks võinud järgida domeenikihis asuvat struktuuri, kus omavahel seotud domeeniklassid olid eraldi kaustades. Viimases data access layeri kiht on eelmises kihis asuvate interface'ide implementatsioon. Ka selles kihis olevas repositooriumide kaustas oleks võinud jagada klassid teemade kaupa kaustadesse. Sellest kaustast leidsin ka ühe väljakommenteeritud klassi, mis on ilmselt kustutamata jäänud. Bussiness logic kiht koosneb DTO, factory ja service kaustadest. Kõik need kaustad on struktureeritud teemade kaupa alamkaustadesse,

mis on nii suure hulga klasside olemasolul hädavajalik ja teeb konkreetse klassi otsimise lihtsamaks. Kõikidest factory'test ja service'itest on interface'id ja nende implementatsioonid. DTO-d on üldiselt hästi kirjeldatud, aga osadel stringi väljadel puuduvad maxlength piirangud, mis näiteks domeenis on olemas (nt PersonDTO). Nt Personi lisamisel (PersonController -> AddNewPerson) kontrollib ModelState PersonDTO-d, aga selle väljadel ei ole ühtegi piirangut. Factory meetodites toimub DTO-dest domeenimudeli loomine ja vastupidi. Mõningad meetodid tekitasid küsimusi: BoxFactor'is on sisuliselt kaks samasugust meetodit CreateBox ja CreateBoxDTO, mis mõlemad võtavad sisendiks Box mudeli ja saavad väljundiks sama DTO.

Andmebaasi ülesehitus

Andmemudel oli koostatud väga põhjalikult. Põhiolemid olid omavahel seotud abiolemite, olekute, kuuluvuste ja abstraktsioonidega. Põhiolemeid oli andmebaasis kokku 11, seega ületati õppejõu poolt antud nõue - minimaalselt 6 olemit - peaaegu poole võrra.

Andmetüüpide valikul köitis tähelepanu see, et olemis *Watch* oli väli *TimeProduced* String tüüpi. Kõrvaltvaataja pilgule tunduks loogilisem kasutada DateTime.

Ka olid olulistel väljadel olemas annotatsioonid. Küsimusi tekitas vaid see, miks oli kellakarbi viitenumbri maksimaalseks sümbolitearvuks pandud 3000.

Koodi loetavus

Kood oli väga hästi jägitav. Meetodide nimedest oli aru saada, mida üks või teine meetod teeb. Nimedes kajastusid nii tegevus kui ka parameetrid *(GetBoxById, AddNewPerson)* Selgelt ja arusaadavalt olid valitud ka muutujate nimed.

Koodi loetavusele tuli ka väga kasuks juba ülalpool mainitid klasside jagamine teemade kaupa kaustadesse.

Veahaldus

Retsensiooni kirjutamise käigus programmi kinni jooksutada ei õnnestunud ja ühtegi *nullreference exceptionit* ei antud - see annab põhjust oletada, et kriitiliste kohtade peal on veahaldus korralikult realiseeritud.

Küll aga oleks kasutajale abiks kui mõnel puhul antaks tagasidet, mis põhjusel *bad request* anti - kas kasutajat ei leitud, kas konkreetsel kasutajal sellist kellakarpi pole jne.

Kokkuvõte

Kokkuvõtvalt võib öelda, et hoolimata mõningatest pisivigadest oli veebiteenus väga hästi läbi mõeldud ja realiseeritud. Meeskonna idee oli huvitav ja veebiteenuse ülesehitus vastas koolis näidatud arendusmustrile. Andmebaas oli põhjalik ja sisaldas kõiki vajalikke põhiolemeid ja abiolemeid. Kood oli hästi struktureeritud ja kergesti loetav. Väikseid lisandusi võiks teha vaid veahalduse koha pealt.