Конкурентни приступ бази података

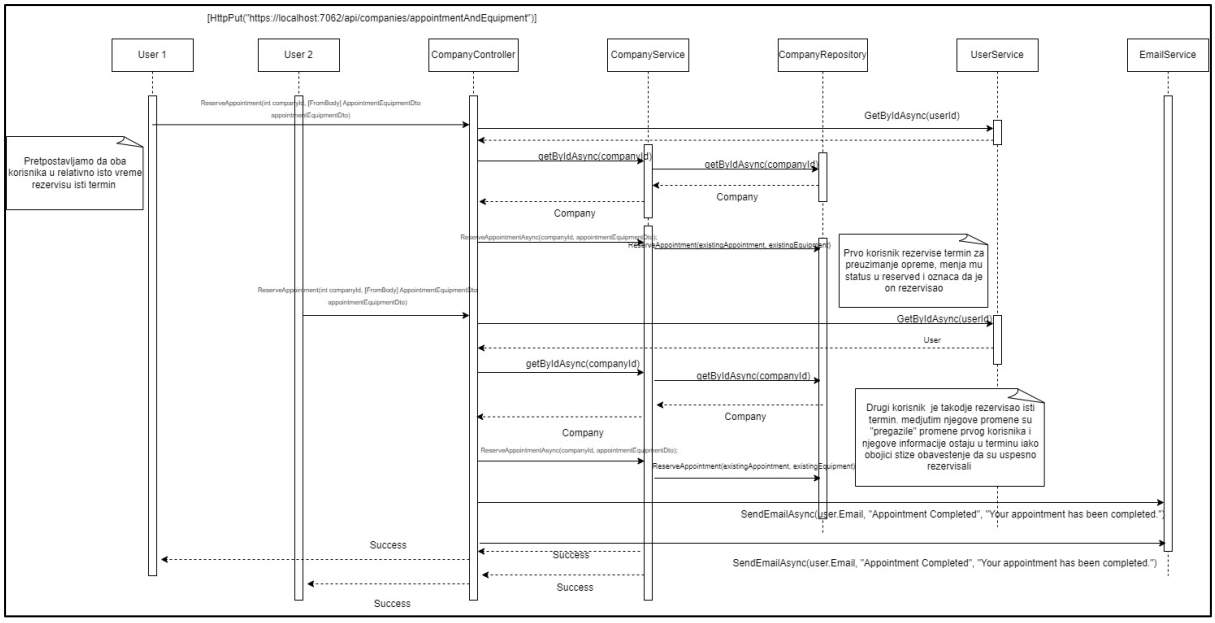
Лука Зеловић – РА 106/2020

Студент 4

Конфликтна ситуација 5 – Термини који су унапред дефинисане не смеју бити резервисани од стране више различитих корисника

Опис:

Корисници у апликацији могу прегледати доступне термине за резервацију и покушати да их резервишу. Међутим, будући да је термин доступан свим корисницима док није резервисан, може се десити да више корисника покуша да резервише исти термин истовремено. Ако не постоји механизам за руковањем тих захтева, може доћи до тога да привидно оба корисника успешне резервишу исти термин. Пошто је резервисање термина заправо метода ажурирања објекта „appointment“, доћи ће до гажења измена одређеног корисника а он ће свакако добити информацију да је успешно резервисао. У овој конфликтној ситуацији присутан је Lost Update.



Опис начина решавања конфликтне ситуације:

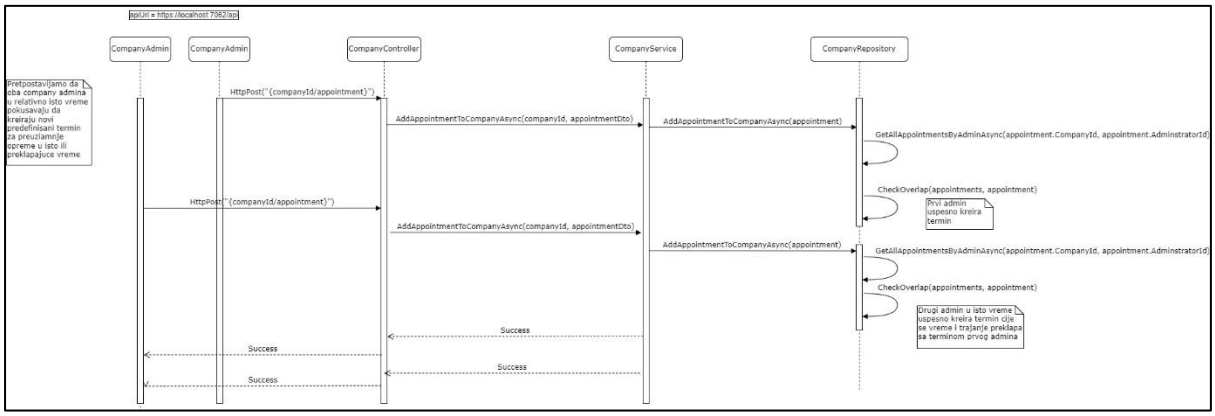
Пошто се приликом резервације термина врши заправо ажурирање објекта термина где се мења његов статус и нека друга поља, да би избегли ову конфликтну ситуацију ми користимо оптимистичко закључавање и у модел термина додајемо Shadow Property – Version. Уз помоћ овог поља сваком објекту (реду у табели) додајемо верзију која је у овом случају насумичан uint број који ће сваки пут мењати након ажурирања па ће тако сигнализирати да је дошло до промена. Што се тиче нивоа изолације, користим

ReadCommited и у репозиторијум започињем трансакцију „using (var transaction = await _context.Database.BeginTransactionAsync())“. Ако све прође како треба извршава се „await transaction.CommitAsync();“ а уколико дође до грешке „await transaction.RollbackAsync();“. Овај случај сам тестирао тако што сам направио две инстанце клијентске апликације и са два корисника покушао да истовремено резервишем термин. На почетку трансакције сам ставио breakpoint и тиме симулирао да су два корисника у релативно исто време покушали да резервишу исти термин.

Конфликтна ситуација 7 – Више администратора компаније не могу унапред дефинисати термине у исто или преклапајуће време

Опис:

У апликацији која подржава администрацију термина, више администратора компаније може покушати да унапред дефинише термине у календару. Конфликтна ситуација настаје када два или више администратора у исто време покушају да додају термине који почињу у исто време или се преклапају. На овај начин више администратора креира термине који заузимају исти временски период, што доводи до неконзистентности у систему.



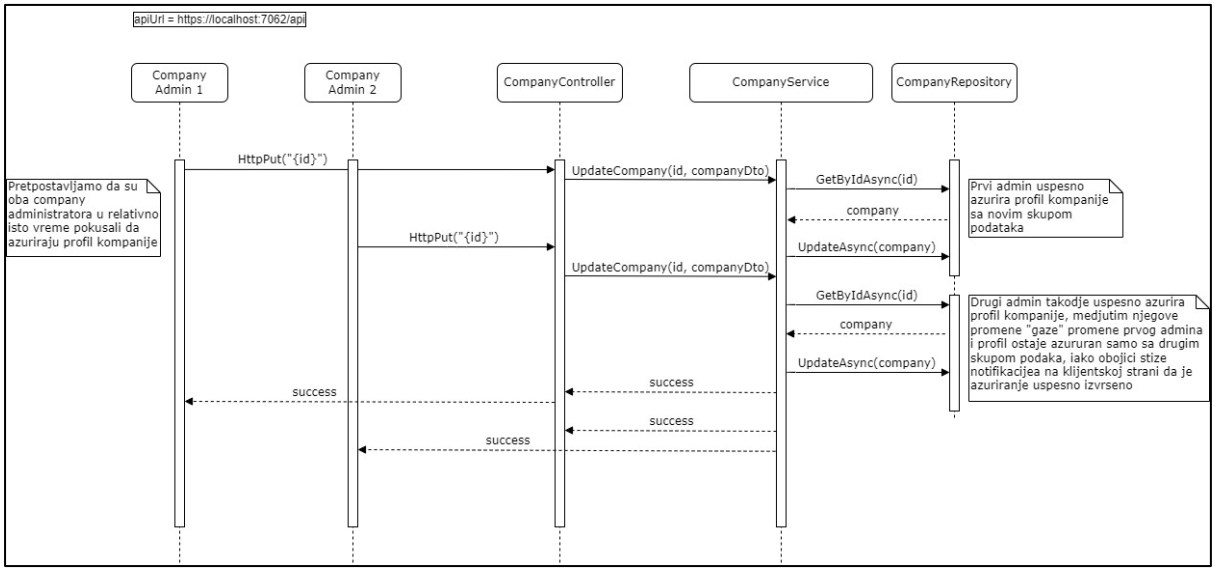
Опис начина решавања конфликтне ситуације:

Овај проблем решио сам тако што сам користио трансакције и у репозиторијуму додао „ using (var transaction = await _context.Database.BeginTransactionAsync(System.Data.IsolationLevel.Serializable))“. Ако све прође како треба извршава се „await transaction.CommitAsync();“ а уколико дође до грешке „await transaction.RollbackAsync();“. Пошто у овом случају имам креирање новог објекта за ниво изолације користио сам Serializable, јер се на тај начин провера да ли су у току трансакције додати нови редови у табелу. Овај случај је тестиран тако што сам на почетку трансакције у репозиторијуму поставио breakpoint и преко две инстанце клијентске апликације покушао да са два различита админа креира термин у исто време. Због поменутог нивоа изолације, долази до грешке и ниједан корисник не креира термин успешно.

Додатна конфликтна ситуација – Више различитих администратора компаније не могу истовремено ажурирати профил компаније

Опис:

У систему где више администратора компаније може управљати профилом компаније, конфликтна ситуација може настати када два или више администратора истовремено покушају да ажурирају податке профила компаније. Ово може довести до Lost Update проблема, где измене једног администратора „прегазе“ измене другог без обавештења.



Опис начина решавања конфликтне ситуације:

Слично првој поменутој ситуацији, овде је такође додато ново поље (Shadow Property) у модел компаније – Version и коришћено оптимистичко закључавање. Оно служи за праћење верзије инстанци компаније. У репозиторијуму додата је трансакција „using (var transaction = _context.Database.BeginTransaction())“ , и као и првој ситуацији ниво изолације је ReadCommited. ())“.Овај случај сам тестирао тако што сам у функцији у репозиторијуму поставио breakpoint и затим преко клијентске апликације ажурирао одређене податке компаније. Када је процес стао на breakpoint-у ,ручно у бази сам променио одређени податак компаније и затим када сам кликнуо continue да се првобитни процес заврши , дошло је до грешке и ажурирање преко клијентске апликације је било неуспешно.