Факултет техничких наука
Универзитет у Новом Саду

Прокси

Елементи развоја софтвера
Предметни пројекат

Ауторка:
PR 83/2020 Бојана Михајловић

20. мај 2023.

Садржај

1. Увод

Пројекат симулира рад прокси компоненте у клијент-сервер апликацији. Прокси који у себи чува резултате претраге побољшава перформансе система јер смањује број приступа бази података.

У систему постоји и компонента уређај која периодично врши мерење и слање нових података у базу података.

2. Техничке спецификације

2.1. Коришћене технологије:

- C# (.Net)
- SQLite

2.2. Опис проблема

Апликације које користе базе података као складиште података приликом добављања података користе методе које су временски дуге и захтевне за ресурсе. Ако су захтеви за читање података из базе чести, апликација ће имати смањене перформансе.

2.3. Решење проблема

Како би се смањио број читања нових података из базе података, уводи се прокси компонента. Прокси компонента служи као посредник између базе података и клијента.

Прокси компонента прима захтеве од клијента и локално складишти резултате претраге. Следећи пут када клијент пошаље захтев за подацима, прокси ће проверити да ли тражене податке има локално, и ако их има, вратиће их клијенту. Ако прокси те податке нема локално, онда ће их затражити од базе података.

Овом компонентом се смањује број приступа бази података и побољшавају се перформансе апликације.

3. Циљеви пројекта и захтеви

3.1. Општи кориснички захтеви

- Могућност претраге мерења из базе података
- Могућност аутоматског генерисања и уноса нових мерења у базу података
- Могућност чувања резултата претраге у локалној меморији проксија
- Могућност аутоматског брисања мерења из локалне меморије проксија којима се није приступало одређени период
- Могућност уписа логова у текстуалну датотеку

3.2. Технички захтеви

Апликацију је потребно покренути на рачунару са инсталираним Windows 10 (или новијим) оперативним системом, као и .Net Framework 4.8 (или новијим).

4. Дизајн и архитектура система

4.1. Компоненте и дијаграми система

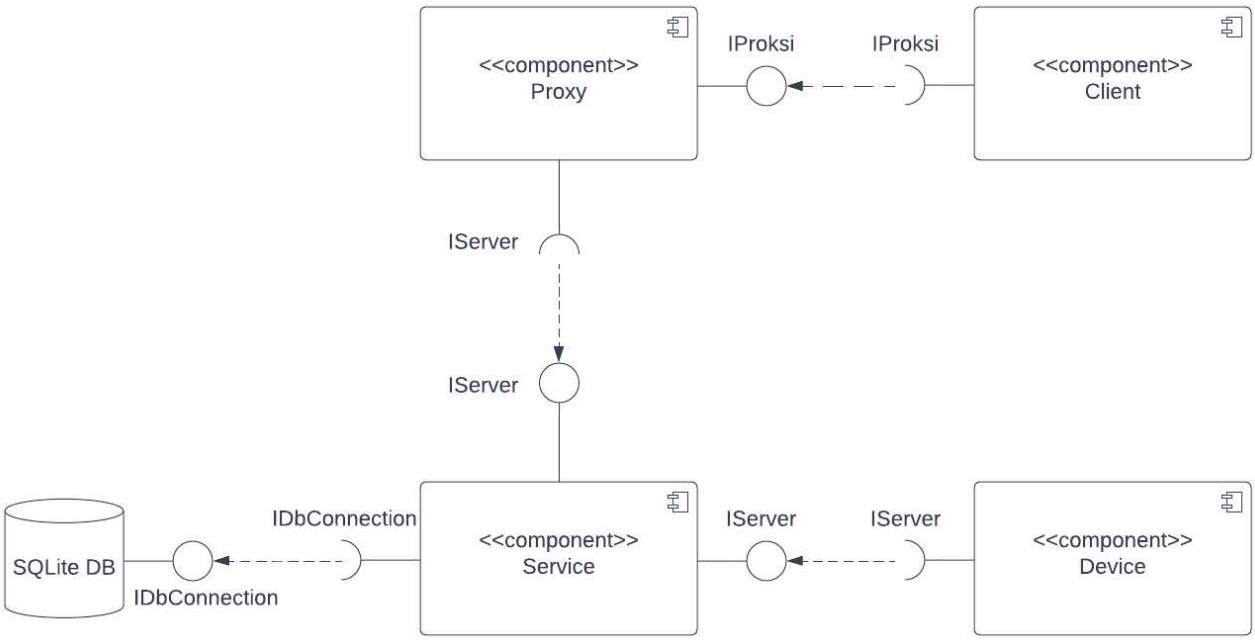
Овај систем се састоји од сервера који комуницира са SQLite базом података, проксија који комуницира са сервером, клијента који комуницира са проксијем, и уређаја који шаље измерене податке серверу.

Компоненте међусобно комуницирају преко WCF (Windows Communication Protocol) и потребних интерфејса.

Модел података који се шаље између компоненти је имплементиран у класи Мерење. Мерење садржи информацију о свом идентификатору, идентификатору уређаја који га је измерио, врсти мерења, вредности и времену када је измерено.

Интерфејси и модел података су имплементирани у Class Library под именом Common.

На слици 1 је приказан дијаграм компоненти система.



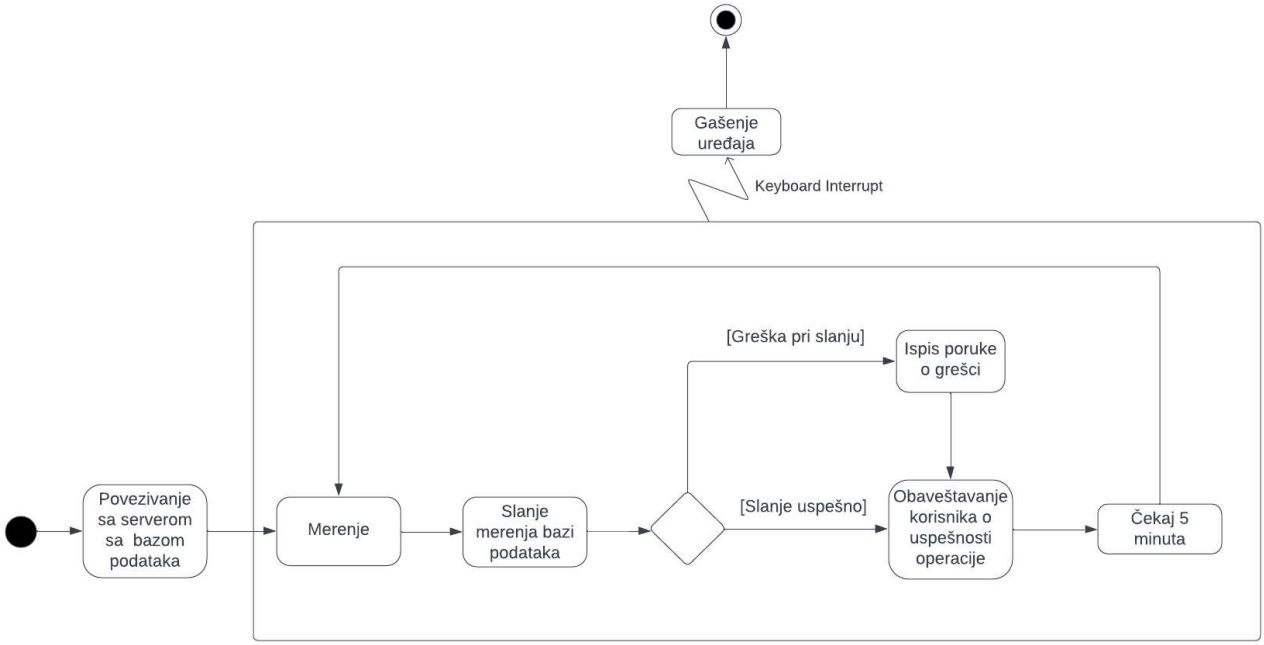
Слика 1. *Дијаīрам комīоненīīи*

4.1.1. Уређај

Компонента уређај није неопходна за рад система. Она симулира уређај који врши мерења, а затим та мерења шаље у базу података. Мерења се врше на сваких пет минута, могу бити или аналогна или дигитална, и имају свој јединствени идентификатор. Један уређај може измерити и дигитална и аналогна мерења.

После сваког мерења на конзоли исписује податке о мерењу, и поруку успешност слања података у базу података.

На слици 2 је приказан дијаграм активности компоненте уређај.



Слика 2. *Дијаграм активности за уређај*

4.1.2. Клијент

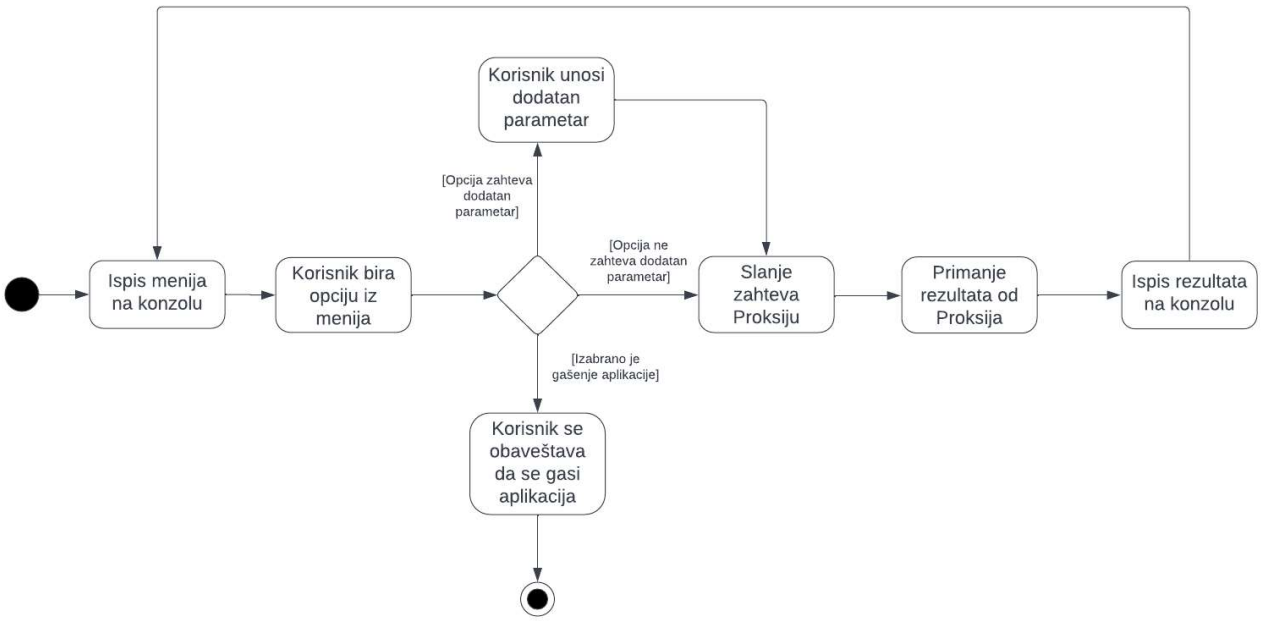
Клијентска компонента служи за интеракцију са корисником. Преко CLI (Console Line Interface) кориснику нуди мени са критеријумима по којима може да претражи сачувана мерења. Критеријуми претраге који су доступни кориснику су:

1. Пронађи сва мерења које је измерио уређај са траженим ИД-јем
2. Пронађи последње мерење које је измерио уређај са траженим ИД-јем
3. Пронађи последње мерење за све уређаје
4. Пронађи сва аналогна мерења
5. Пронађи сва дигитална мерења.

Такође, кориснику се нуди опција да угаси клијента.

Након што корисник одабере критеријум претраге, шаље се захтев проксију. Прима се резултат претраге, и исписује се на конзоли. Након тога се кориснику поново приказује мени.

На слици 3 је приказан дијаграм активности за компоненту клијент.



Слика 3: *Дијаграм активности за клијентску компоненту*

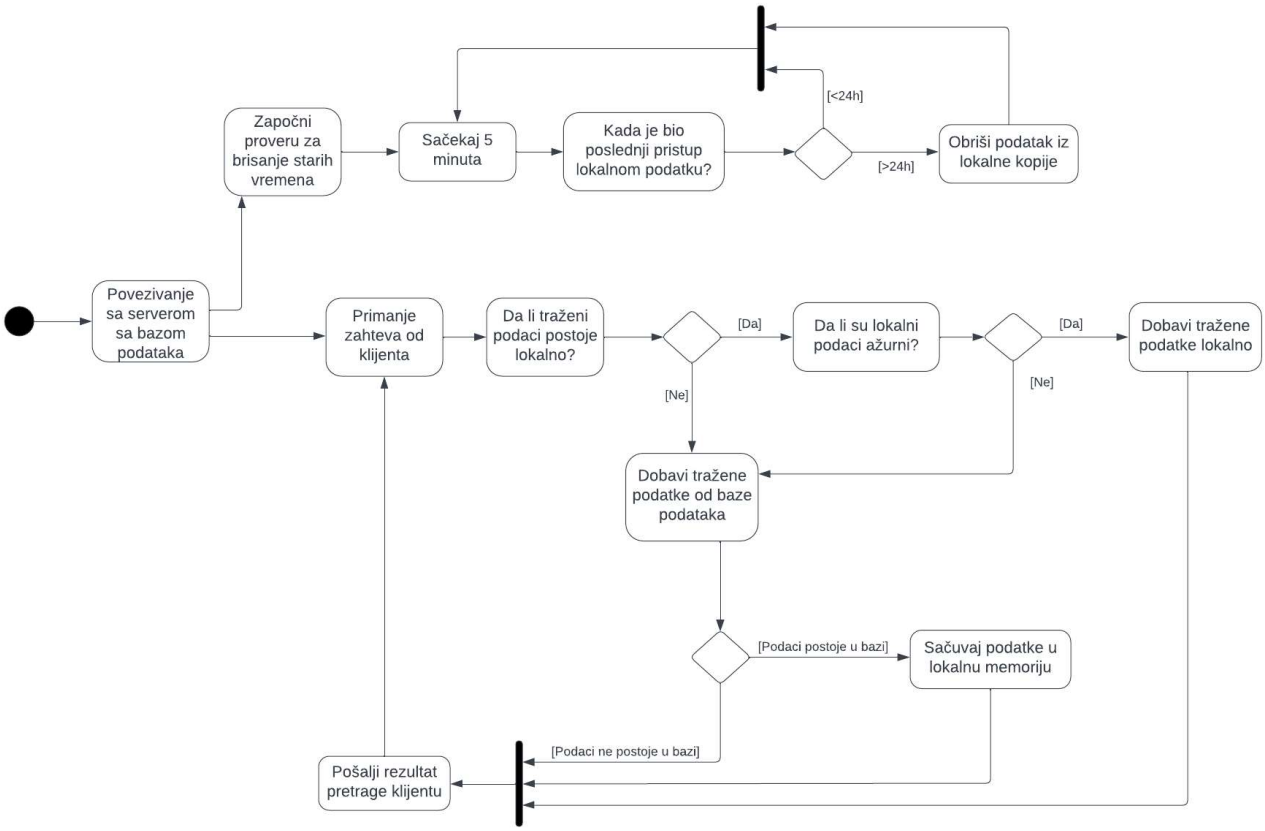
4.1.3. Прокси

Компонента прокси је централна компонента овог система. Њен циљ је побољшање перформанси апликације. Прокси врши двојаку улогу – као сервис ка клијентској апликацији, и као клијент ка серверу са базом података. Осим тога, садржи и локалну меморију у коју смешта податке које је добавио из базе.
Као сервис прима захтеве од клијентске апликације. Када прими захтев, проверава да ли тражене податке има локално. Ако их има у локалној меморији, бази података шаље захтев у ком проверава да ли су локални подаци ажурни. Ако су ажурни, податке које је пронашао локално враћа клијенту. Ако подаци нису ажурни, или ако не постоје у локалној меморији, бази података се шаље нови захтев – за добављање тражених података. Они се прво смештају у локалну меморију проксија, а затим враћају клијенту као резултат претраге.

Прокси такође памти времена добављања сваког податка, као и времена последњег приступа. Сваки пут када се податак добави од базе, ажурирају се и време последњег приступа и време добављања, а у случају читања података из локалне меморије, ажурира се само време последњег приступа. Ова времена служе да би прокси обрисао податке којима није приступано дуже од једног дана. Тиме прокси растерећује своју меморију.

Прокси такође уписује логове о свим дођајима у текстуалну датотеку.

На слици 4 је приказан дијаграм активности за компоненту прокси.



Слика 4: *Дијаграм активности за прокси*

4.1.4. Сервер

Компонента сервер комуницира са базом података и из ње чита податке које прокси тражи, или уписује податке које је уређај послао. При захтеву за упис се прослеђује објекат који желимо да упишемо у базу, а при захтеву за читање се шаље query за релациону базу.

База података која се користи у овом систему је SQLite. Како је она библиотека, за њену интеграцију у пројекат је било потребно само скинути одговарајући NuGet пакет.

Сервер такође уписује логове о свим дођајима у текстуалну датотеку.

5. Тестови

5.1. Ручни тестови

У току развоја апликације свака имплементирана функционалност се тестирала.
Текстуалне датотеке са логовима су олакшале праћење начина на који систем функционише, и евентуално, проналажење тачног места где је у току извршавања дошло до грешке.
Сви ручни тестови које је тестер покренуо су успешно положени.

5.2. Аутоматски тестови

Ручно тестирање не може да изазове неке специфичне случајеве који се ретко дешавају, али могу да изазову катастрофалне последице. Због тога се користе аутоматски тестови.

5.2.1. Коришћене технологије

- Moq 4.18.4
- NUnit 3.13.3
- NUnit3TestAdapter 4.4.2

5.2.2. Тестови који нису урађени

Неписано правило Унит тестирања је да тестирамо само код који смо ми написали. Сходно томе, не тестирају се базе података, фајл системи исл. Подразумева се да је особа која је написала код за њих тестирала њихову функционалност.
Иако постоје начини да се тестира SQLite база података која је коришћена у овом пројекту, ти начини захтевају коришћење Entity Framework Core-а, који нема подршку за .NET Framework. Зато обе методе СерверСервис класе нису тестиране, као и неке од метода из ПроксиСервис класе.
Због тога су те методе тестиране ручно, и прошле су са задовољавајућим резултатима.