УНИВЕРЗИТЕТ У БЕОГРАДУ
ФАКУЛТЕТ ОРГАНИЗАЦИОНИХ НАУКА

ЗАВРШНИ РАД

Тема: Развој софтверског система за управљање кућним буџетом употребом ASP.NET Core оквира

Ментор
проф. др Саша Лазаревић

Студент
Никола Абадић 105/17

Београд, 2021. године

Развој софтверског система за управљање кућним буџетом употребом ASP.NET Core оквира

Као последица константног развоја веб технологија последњих деценија, веб апликације постају значајно приступачније, сигурније, лаке за коришћење и практичне за корисника. Примећује се примена веб апликација и технологија у све више аспеката свакодневног живота и навика корисника, јер пружају брзу и непосредну интеракцију и приступ релевантним информацијама. Креирани софтверски систем представља веб апликацију у чијем развоју су коришћени ASP.NET Core MVC и Entity Framework Core оквири. Апликација својим корисницима нуди могућност евиденције и управљања личним финансијама и кућним буџетом. Развој ове веб апликације прати упрошћену Ларманову методу кроз пет фаза: идентификовање корисничких захтева, анализа, пројектовање, имплементација и тестирање. Приликом реализације имплементирани су Repository и Unit of Work патерни. Искоришћене су предности .NET окружења и његове интегрисаности са свим коришћеним технологијама која за резултат има целовито корисничко искуство

кључне речи: софтверски систем, веб апликација, Ларманова метода, .NET Core, ASP .NET Core MVC, Entity Framework Core, Repository патерн, Unit of Work патерн, Razor, кућни буџет

Садржај

Списак слика

Списак табела

1. Увод

Развојем веб технологија примећује се велики број решења који корисницима задовољавају потребе за брзим приступом информацијама и међусобном интеракцијом. Последњих година објављен је велики број оквира за веб апликације, како за развој корисничког интерфејса, тако и за серверске програме (енгл. backend). Један од оквира који је објавио Microsoft је ASP.NET Core, који омогућава развој веб апликација различитих архитектура, које је могуће лако надоградити и тестирати.

Креирана веб апликација омогућава својим регистрованим и пријављеним корисницима могућност евиденције и управљања својим кућним буџетом – помоћу рачуна и трансакција. Веб апликација прати MVC архитектуру, која је заједно са осталим коришћеним технологијама описана у наредном поглављу.

У оквиру овог рада прати се развој софтверског система за управљање кућним буџетом коришћењем упрошћене Ларманове методе. Након идентификовања корисничких захтева, дат је приказ фаза анализе, пројектовање и имплементације различитих делова софтверског система, као и опис даљег тестирања.

У оквиру студијског примера је идентификовано 10 случајева коришћења, за које се у фази анализе дефинише одговарајућа структура и понашање софтверског система – пословна логика. У наредној фази, пројектује се изглед корисничког интерфејса, апликациона логика и складиште података. У оквиру фазе имплементације, дат је приказ софтверског система имплементираног помоћу програмског језика C# и Microsoft SQL Server система за управљање базом података. У оквиру слоја приступа подацима имплементирани су Repository и Unit of Work патерни.

Коришћено је Microsoft Visual Studio 2019 развојно окружење.

2. Преглед коришћених технологија

Коришћени оквир за имплементацију софтверског система је ASP.NET Core којег извршава .NET Core оквир који је део .NET екосистема. Програмски језик коришћен у оквиру .NET екосистема је C#. Креирана веб апликација прати MVC архитектуру. Како би кориснички интерфејс изгледао савременије и богатије коришћене су технологије и у оквиру Razor страница – Bootstrap, JavaScript, JQuery и Ajax.

2.1 .NET екосистем

.NET је платформа отвореног кора (енгл. open source) за развој софтвера, која је креирана од стране Microsoft-а и намењена је креирању различитих врста апликација - веб, мобилних апликација, десктоп, видео игре и слично. Састоји се из различитих алата, програмских језика и библиотека. Током 90-их година, Microsoft почиње са радом на .NET стратегији, чији је циљ био обухватити све Microsoft-ове производе. Прва верзија .NET-а излази 13. фебруара 2002. године, и до данас је објављено више верзија са новим функционалностима [9].

.NET екосистем се састоји из више компонената у оквиру исте платформе. Обухвата следећа извршна окружења (имплементације):

- **.NET Framework** (WPF, Windows Forms, ASP.NET) – намењен Windows ОС.
- **.NET Core** (ASP.NET Core, Universal Windows Platform - UWP) – вишеплатформско окружење
- **.NET 5** (ASP.NET Core, WPF, Windows Forms, Blazor) – јединствена платформа за десктоп, веб, cloud, мобилне, IoT, AI апликације и видео игре
- **Mono for Xamarin** (IOS, OS X, Android) - вишеплатформско окружење

Наведена окружења имплементирају .NET Standard библиотеку, која представља спецификацију .NET API-ја, који имају различиту имплементацију за свако извршно окружење – различита окружења имплементирају различите верзије .NET Standard библиотеке (имплементирају одређене скупове API-ја) [9].

Извршна окружења користе алате и инфраструктуру да компајлирају и изврше код. Под овим се подразумевају програмски језици- C#, F#, Visual Basic, компајлери- Roslyn, системи за управљање отпадом (енгл. garbage collection), као и алати за извршавање кода попут MS Build-а, или CLR-а [9].

Постоје велики број библиотека које се могу користити у оквиру .NET екосистема и најчешће су доступни у облику NuGet пакета. NuGet је .NET-ов систем за управљање зависностима (енгл. package manager) који садржи више од 90 хиљада пакета [9].

2.2 .NET Core оквир

.NET Core је једно од извршних окружења у оквиру .NET екосистема. Објављен је 2016. године и отвореног је кода (енгл. open source). Не представља нову верзију .NET оквира и није замишљен као његова замена. Изграђен је како би омогућио вишеплатформску компатибилност развоја апликација. Сачињен је од App Host-а (dotnet.exe), који покреће Common language runtime (CoreCLR) и .NET Core библиотеке класа [9].

2.3 ASP.NET Core оквир

ASP.NET Core је вишеплатформски оквир отвореног кода, оријентисан је на перформансе и намењен је креирању апликација. ASP.NET Core оквир омогућава креирање веб апликација и сервиса, IoT и мобилних апликација, као и backend-а мобилних апликација на било ком оперативном систему. Извршава га .NET Core оквир [10].

Бенефити које овај оквир пружа су [11]:

- Већа сигурност
- Смањена зависност између компоненти
- Боље перформансе

Подразумевана архитектура је MVC (Model-View-Controller) архитектура која је описана у следећем поглављу.

Коришћењем овог оквира могуће је креирање и других врста веб апликација. За развој корисничког интерфејса на серверској страни користе се Razor странице, док се на корисничкој страни може користити Blazor. За креирање RESTful HTTP сервиса ASP.NET Core нуди могућност креирања Web API-ја.

2.3.1 ASP.NET Core MVC архитектура

MVC је патерн за развој софтвера који се често користи за развој апликација које имају кориснички интерфејс. Могу се пронаћи различите интерпретације оригиналног MVC патерна које се фокусирају на различите аспекте овог патерна. Оригинални патерн је специфициран за апликације са богатим корисничким графичким интерфејсом (GUI) и користи терминологију и парадигме везане за њихово окружење. Суштински, патерн покушава да одвоји управљање и манипулацију података од њихове визуелне репрезентације [12].

Компоненте које чине овај патерн су:

1. Модели – подаци који се приказују
2. Погледи – Шаблони који приказују податке садржане у оквиру модела
3. Контролери – Ажурирају модел и бирају одговарајући поглед

Свака од компоненти у оквиру MVC апликације је одговорна за један аспект целокупног системи, а заједно се користе за генерисање корисничког интерфејса.

Редослед одвијања догађаја приликом одговора апликације на интеракцију корисника је следећи [12]:

1. Контролер прима захтев
2. У зависности од захтева, контролер прикупља податке од пословне логике софтверског система или ажурира податке у оквиру модела
3. Контролер бира поглед који ће приказати и прослеђује му модел који садржи податке за приказ
4. Поглед користи податке из модела и генерише кориснички интерфејс

Приликом описивања MVC патерна у овом формату, контролер представља приступну тачку интеракције. Приликом иницирања интеракције, корисник комуницира са контролером. У веб апликацијама, ова комуникација се одвија у форми HTTP захтева, где контролер прихвата захтев ка одређеном URL-у. У зависности од захтева, контролер може преузети различите акције користећи модел [12].

Након ажурирања модела, контролер бира начин на који ће приказати податке. Једна од предности коришћења MVC патерна је да модел, који представља податке, одвојен од финалне репрезентације тих података, односно погледа. Ово одвајање ствара могућност да контролер на различите начине представи резултате својих акција. Уколико захтев потиче од стандардне веб апликације, контролер може приказати HTML поглед. Уколико захтев долази са друге апликације – контролер враћа резултат у формату који је та апликација разуме, попут JSON-а или XML-а [12].

Још једна предност одвајања модела од погледа је поједностављено тестирање. Код корисничког интерфејса је обично тежак за тестирање, јер зависи од окружења. Модел се онда тестира независно од коришћених зависности у конструктима корисничког интерфејса [12].

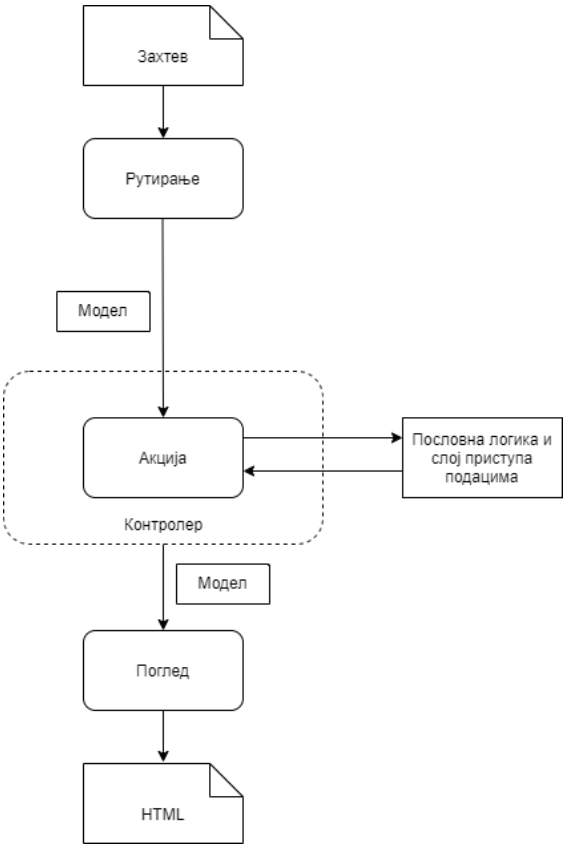
ASP.NET Core имплементира MVC користећи посебан посредни софтвер (енгл. middleware) који се најчешће налази на крају низа свих посредних софтвера (енгл. pipeline). Захтев се обрађује од стане сваког од посредних софтвера у оквиру pipeline-а, да би га на крају прихватио MVC middleware који обједињује цео MVC патерн. Први корак који MvcMiddleware предузима је рутирање захтева до одговарајућег контролера и акције у оквиру њега [12].

У ASP.NET Core оквиру контролер је заправо класа која се састоји од неколико логички повезаних акција. Акција је метода која се покреће као одговор на примљен захтев. Акција као параметар прима модел – објекат који се састоји од података садржаних у оквиру захтева који су неопходни за извршавање акције. У оквиру акције се најчешће врши [12]:

- Валидација прослеђеног модела
- Позивање одговарајућих метода на моделу пословне логике
- Селекција одговарајућег одговора на основу добијених података

Након што акција добије резултат од пословне логике, она може изабрати начин генерисања добијеног резултата. Резултат се до погледа може проследити помоћу модела – објекта који садржи податке који су погледу неопходни како би генерисао одговор. Одговор се шаље назад кроз pipeline посредног софтвера [12].

На наредној слици се може видети животни циклус захтева, од његовог пријема до генерисања одговора.



Слика 1 - Животни циклус захтева код ASP.NET MVC оквира

Кључна предност која се остварује овим патерном је подела компоненти на основу задужења и функције (енгл. separation of concerns):

- Поглед је одговоран само за прихватање података и генерисање HTML или JSON документа
- Пословна логика је одговорна за извршење захтеваних системских операција
- Контролер валидира долазећи захтев и селектује одговарајући поглед на основу добијених података од пословне логике

Постојање јасно дефинисаних граница поједноставило је независно ажурирање и тестирање сваке од компоненти. Ако се логика на корисничком интерфејсу промени, није неопходно и ажурирање пословне логике и самим тим се смањује појава грешака [12].

2.4 Entity Framework Core

Entity Framework Core (EF Core) је најновија верзија Entity Framework-а коју је објавио Microsoft. Дизајниран је на начин да буде лаган (енгл. lightweight), подложан допунама и пружа подршку вишеплатформском развоју софтвера и представља део Microsoft-овог .NET Core оквира. Једноставан је за коришћење и нуди побољшања у перформансама у односу на претходне верзије Entity Framework-а [20].

Entity Framework Core је објектно-релациони мапер који омогућава програмерима да раде са подацима на објектно-оријентисан начин тако што врши мапирање (превођење) објеката написаних у програмском језику апликације у податке који се чувају у релационим базама података. Објектно-релациони мапери су библиотеке кода омогућавају [20]:

- Мапирање доменског модела у табеле релационе базе података
- Креирање базе података и одржавање шема у складу са променама у моделу
- Генерисање SQL кода и његово извршавање над базом података
- Управљање трансакцијама

Entity Framework Core пружа подршку за више различитих система за управљање базом података.

Приступ подацима се одвија уз помоћ модела који је сачињен од доменских класа и контекста који представља сесију над базом податка. Контекст омогућава упите над подацима и чување података. Entity Framework Core подржава следеће приступе у развоју модела [4]:

- Генерисање модела на основу постојеће базе податка
- Ручно куцање модела који ће одговарати бази података
- Након креирања модела, коришћење EF миграција за креирање базе података на основу модела. Миграције омогућавају и измену и допуну модела и базе података.

Entity Framework Core користи скуп конвенција за креирање модела заснованог на креираним доменским класама. Могућа је додатна спецификација за додавање и/или измену модела који се креира на основу конвенције – реимплементацијом (енгл. override) OnModelCreating методе у оквиру изведеног контекста уз помоћ ModelBuilder API- ја. Ово је најмоћнији начин конфигурације који је омогућен и не захтева измену доменских класа. Конфигурација Fluent API-јем ће „прегазити“ конвенцију и анотације у оквиру доменских класа (које представља алтернативни начин конфигурације) [21].

Додавањем нових функционалности, доменски модели апликација се често мењају и одговарајуће шеме базе података се морају синхронизовати са доменским класама. Миграције, које су саставни део Entity Framework Core оквира, пружају могућност инкременталних измена шема базе података, истовремено задржавајући постојеће податке унутар базе података [22].

Када дође до промене у моделу, програмер користи алате EF Core оквира како би додао одговарајућу миграцију која описује промене које ће базу података ажурирати тако да остане синхронизована са моделом апликације. EF Core пореди постојећи модел са снимком (енгл. snapshot) претходног модела како би утврдио разлику и генерише одговарајући миграциони фајл који се може пронаћи у оквиру изворног кода пројекта. Након генерисања нове миграције, она се примењује на базу података. EF Core оквир чува све примењене миграције у посебну табелу и на тај начин зна које миграције су већ примењене и које тек треба да примени [22].

Инстанце објеката доменских класа се добијају као резултат упита над базом података коришћењем Language Integrated Query (LINQ) упита. LINQ омогућава коришћење C# -а (или других програмских језика у оквиру .NET екосистема) за писање упита. Користи изведен контекст и доменски модел за референцирање објеката базе података. EF Core оквир шаље репрезентацију LINQ упита до провајдера базе података који је преводи у SQL или неки други језик специфичан за упите над базом података. Упити се увек извршавају над базом података, чак иако су подаци који су резултат упита већ сачувани у оквиру контекста[23].

2.5 LINQ упити

LINQ, тј. Language Integrated Query, омогућава писање структурираних упита са сигурним типовима за локалне колекције објеката и удаљене изворе података. LINQ омогућава извршавање упита над сваком колекцијом која имплементира интерфејс IEnumerable <>, без обзира на то да ли је реч о низу, листи, XML DOM-у, или удаљеном извору података (као што је табела на SQL Серверу). Предности LINQ-а су и провера типова у време превођења и динамичко састављање упита[18].

У LINQ -у, основне јединице података су **секвенце** и **елементи**. Секвенца је сваки објекат који имплементира генерички интерфејс IEnumerable, а елемент је свака ставка у тој секвенци[18].

Оператор упита (енгл. query оператор) јесте метода која трансформише секвенцу. Типичан оператор упита прихвата улазну секвенцу и производи трансформисану излазну секвенцу. У класи Enumerable, у именском простору System.Linq има око 40 оператора упита и сви су имплементирани као статичке проширене методе. Они се зову стандардни оператори за упите. LINQ подржава и секвенце које се могу динамички проследити из удаљеног извора података као што је SQL Сервер. Те секвенце додатно имплементирају интерфејс IQueryable <> и подржава их одговарајући скуп стандардних оператора за упите у класи Queryable. У сврху креирања сложенијих упита, могуће је уланчавање оператора за упите[18].

Оператори упита најчешће као параметар примају анонимну функцију (ламбда израз) или делегат.

2.5.1 Ламбда изрази

Ламбда израз је анонимна метода написана уместо инстанце делегата. Компајлер одмах конвертује ламбда израз у једно од следећег [18]:

- инстанцу делегата
- стабло израза, типа Expression<TDelegate>, које представља кôд унутар ламбда израза у секвенцијалном објектном моделу. То омогућава да се ламбда израз интерпретира касније у време извршавања.

Ламбда израз декларише потпис и тело методе, али немају формални идентитет – анонимни су, осим ако нису додељени декларисаном делегату. За разлику од делегата, може им се директно приступити приликом доделе вредности догађајима и у оквиру различитих LINQ оператора и метода [19].

Приликом имплементације, ламбда изрази су често коришћени уместо креирања одговарајућих предикатских делегата (енгл. Predicate), који примају параметар, проверавају услов и враћају добијену логичку вредност.

2.6 C# програмски језик

C# је модеран, објектно-оријентисан програмски језик који омогућава програмерима да развијају различите врсте сигурних и комплексних апликација које се извршавају у .NET екосистему. C# вуче корене из C породице програмских језика [13].

C# програме извршавају .NET, виртуелни систем извршавања кода - Common Language Runtime (CLR) и скуп библиотека. CLR представља Microsoft-ову имплементацију CLI-a (common language infrastructure), међународног стандарда који представља основу за креирање извршних и развојних окружења у којима се програмски језици и библиотеке могу користити заједно [13].

Изворни код написан у језику C# се компајлира у IL (intermediate language) који се усклађује са CLI спецификацијом. IL и ресурси се чувају као assembly код, најчешће са екстензијом .dll. Овај фајл садржи манифест који обезбеђује информације о типу и верзији assembly-ја. Приликом извршавања C# програма, assembly се учитава у CLR који извршава Just-In-Time (JIT) превођење IL кода у изворни машински код. CLR обезбеђује и друге функционалности попут аутоматског garbage collection-a, управљање изузецима и ресурсима [13].

Програмски језик C# нуди функционалности које омогућавају развој комплексних и издржљивих апликација. **Garbage collection** аутоматски врши повраћај меморије окупиране од стране недоступних и некоришћених објеката. **Nullable** типови променљивих омогућавају да и вредносне променљиве имају null вредност. **Управљање изузецима** обезбеђује структуиран и допуњујући приступ детекције грешака и њиховом исправљању. **Lambda изрази** подржавају технике функционалног програмирања. **Language Integrated Query (LINQ**) синтакса пружа општи патерн за рад са подацима из било ког извора. C# такође нуди могућност **асинхроних операција** приликом развоја дистрибуираних система [13].

C# има јединствен систем типова променљивих. Све променљиве наслеђују тип object и сви типови деле заједнички скуп општих операција. Постоје две врсте променљивих у језику C# - референтни и вредносни. C# такође омогућава динамичку алокацију објеката (доделу вредности и одређивање типа) [13].

Основни концепти C#-а као објектно-оријентисаног програмског језика објашњени су у наредним поглављима.

2.6.1 Енкапсулација

Енкапсулација (учаурење) је први концепт објектно-оријентисаног програмирања који представља апстракцију којом се обједињују подаци и функције у јединствен ентитет- класу.

Са друге стране, овај концепт истиче да класа или структура може да одреди различите модификаторе приступа за своје атрибуте и методе ван те класе или структуре. Елементи класе чије коришћење није предвиђено изван класе или склопа (assembly) се могу сакрити како би се смањила могућност појаве грешака или злоупотреба [15].

Измена или приступ учауреним подацима је једино могућ преко дефинисаних метода класе са јавним модификатором приступа.

У програмском C# су дефинисани следећи модификатори приступа[16]:

1. Public – приступ је могућ из целог склопа или било ког другог склопа који на њега референцира
2. Private – приступ је могућ једино у оквиру класе/структуре
3. Protected – приступ је могућ у оквиру класе и у оквиру свих класа које је наслеђују
4. Internal – приступ је могућ у оквиру истог склопа
5. Protected internal - приступ је могућ из целог склопа и из свих класа које наслеђују ту класу (које се могу налазити и у другим склоповима)
6. Private protected – приступ је могућ у оквиру класе и у оквиру других класа које је наслеђују, а налазе се у оквиру истог склопа

Класе и структуре могу имати public и internal модификаторе приступа (internal је подразумевајући). Елементи структуре могу бити public, private или internal, док елементи класа могу имати свих шест модификатора приступа. Елементи класа и структура има private као подразумевајући модификатор приступа [16].

Најчешћа имплементација енкапсулације у C# програмском језику је креирање Property-ја за сваки од атрибута класе. На овај начин атрибути имају приватан модификатор приступа, а могуће му је приступити једино преко Property-ја који је јаван.

2.6.2 Наслеђивање

Наслеђивање је један од основних концепата објектно-оријентисаног програмирања који омогућава креирање нових класа који користе, проширују и мењају понашање дефинисано у оквиру других класа. Класа чије се методе и атрибути наслеђују се назива базна класа (изворна класа). Наслеђивање је једноструко – једна класа може имати само једну базну класу. Међутим, наслеђивање је преносно- ако класа Ц наслеђује класу Б, а класа Б наслеђује класу А, тада класа Ц наслеђује све методе и атрибуте класа А и Б [14].

Када су интерфејси у питању, они могу имати подразумевану имплементацију својих метода. Ове имплементације наслеђују други интерфејси и класе које имплементирају те интерфејсе [14].

Класе које наслеђују друге класе наслеђују све методе и атрибуте базне класе – осим конструктора и деструктора. Класа не мора да поново имплементира наслеђене методе – може их користити у неизмењеном облику [14].

Када базна класа прогласи своју методу **виртуелном** (virtual), класа која је наслеђује може да има сопствену имплементацију ове методе (енгл. override). Ако базна класа прогласи своју методу **апстрактном** (abstract), класа која је наслеђује мора имати своју имплементацију ове методе (осим и ако сама класа није апстрактна) [14].

Класа може спречити друге класе да је наследе тако што ће бити проглашена као **sealed** (могуће је и прогласити методу као sealed – тада класе које је наслеђују неће моћи да имају своју имплементацију).

2.6.3 Полиморфизам

Реч полиморфизам је грчког порекла и има значење „више облика“ . У програмирању, постоје два аспекта са којих се сагледава овај концепт[17]:

1. У време извршавања, објекти изведених класа (класе које наслеђују неку другу класу), се могу третирати као објекти базне класе у одређеним ситуацијама – када су ти објекти параметри неке методе или као чланови неке колекције или низа. Код појаве овог концепта, декларисан тип објекта не мора бити идентичан типу објекта приликом извршавања.
2. Базне класе могу дефинисати и имплементирати виртуелне методе, док их изведене класе могу override-овати, односно дати им сопствену имплементацију. У време извршавања, приликом позива методе, CLR тражи тип објекта у време извршавања, и позива одговарајућу имплементацију виртуелне методе (која може имати различите имплементације). Као последица полиморфизма, могуће је позвати методу над објектом базне класе која ће имати имплементацију објекта у време извршавања кода.

У C#-у, на све типове се може применити полиморфизам, укључујући и типове дефинисане од стране корисника, зато што сви типови наслеђују класу Object.

Када изведена класа наследи базну класу, она наслеђује све методе, поља, Property-је и догађаје базне класе. Дизајнер изведене класе има више опција за које се може одлучити приликом избора имплементација виртуелних метода[17]:

- Изведена класа може да реимплементира виртуелне методе и тако дефинише ново понашање
- Изведена класа може наследити понашање виртуелне методе дефинисане у оквиру „најближе“ базне класе и спречи њену даљу реимплементацију у оквиру својих изведених класа
- Изведена класа може да дефинише нову, невиртуелну имплементацију оних чланова који скривају имплементацију дату у базној класи

Виртуелна метода у базној класи мора имати **virtual** кључну реч приликом њеног дефинисања, док у изведеној класи њена реимпементација мора садржати кључну реч **overrire**. Поља (атрибути) класе не могу бити виртуелна[17].

Када изведена класа реимплементира виртуелну методу, та метода са новом имплементацијом се позива чак иако је објекат декларисан као објекат базне класе.

Уколико изведена класа жели да има методу са истим називом као и виртуелна метода у базној класи, могуће је користити кључну реч **new** приликом дефинисања нове методе и на тај начин „сакрити“ имплементацију дефинисану у оквиру базне класе. Уколико изведена класа жели да спречи даљу реимплементацију виртуелне методе, може у оквиру своје реимплементације садржати кључну реч **sealed**, чиме ће метода престати да буде виртуелна. Изведена класа која је креирала нову методу или реимплементирала виртуелну методу може приступити виртуелној методи и њеној имплементацији у оквиру базне класе користећи кључну реч **base**[17].

2.7 Технологије коришћене за израду корисничког интерфејса

Како би се унапредило корисниково искуство коришћени су Bootstrap – frontend оквир и JavaScript библиотеке (JQuery i Chart.js). За слање HTTP захтева са веб страница коришћен је Ajax технологија.

2.7.1 Bootstrap

Bootstrap је бесплатан оквир отвореног кода који је намењен frontend развоју веб сајтова и апликација. Написан је користећи HTML, CSS, и JavaScript-а и олакшава креирање респонзивних сајтова. Респонзиван дизајн омогућава да веб страница препозна корисникову величину екрана и оријентацију и да се аутоматски прилагоди. Bootstrap укључује компоненте корисничког интерфејса, различите распореде компоненти и JavaScript алате[24].

Креиран је у оквиру компаније Twitter 2010. године и прерастао је у један од највећих пројеката отвореног кода.

Предности коришћења Bootstrap-а су [25]:

- Респонзиван распоред компоненти (енгл. grid) и слике
- Велики број компоненти – навигациони менији, различите врсте поља, иконице и др.
- Документација и заједница

2.7.2 JavaScript

JavaScript је скриптни програмски језик који је креиран како би веб странице „оживеле“. Програми написани у овом језику се називају скрипте и могу се писати директно унутар HTML страница – на тај начин ће се покренути аутоматски, након учитавања странице. Скрипте се извршавају као обичан текст – не захтевају специјалну припрему или компајлирање пре извршавања[26].

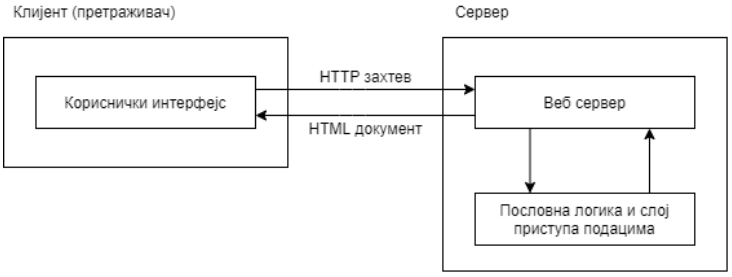
JavaScript је динамички програмски језик који се најчешће користи као део веб страница чија имплементација дозвољава интеракцију са корисником, стварајући динамичне веб странице. JavaScript се интерпретира и има објектно-оријентисане могућности. JavaScript на клијентској страни је један од најчешћих облика коришћења овог језика. Како би се претраживач интерпретирао скрипте, оне морају бити садржане у оквиру HTML документа или референциране од стране HTML документа[27].

JavaScript нуди велики број библиотека које пружају различите функционалности. У имплементацији је коришћена Chart.js библиотека која омогућава графички приказ различитих типова графикона и дијаграма.

2.7.3 JQuery и Ajax

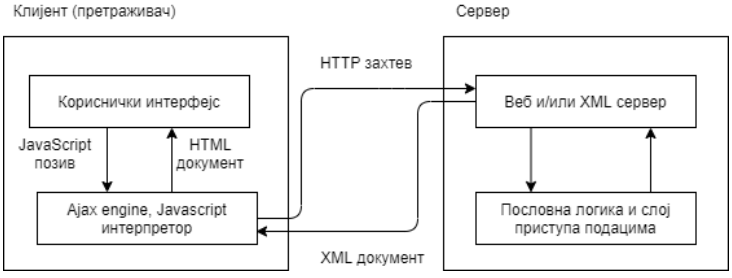
JQuery је JavaScript библиотека отвореног кода (енгл. open source). Методе ове библиотеке се могу позвати у једној линији кода и обухватају велики број општих задатака који су написани у JavaScript-у. Постојање ове библиотеке олакшава коришћење JavaScript-а на веб сајтовима. JQuery је концизна и брза библиотека која нуди велики број функционалности и подржана је од стране свих претраживача. Иако постоји велики број JavaScript библиотека, JQuery је једна од најпопуларнијих и широко коришћених од стане програмера. Флексибилна је, лака за коришћење и постоји велики број додатака (енгл. plugins) који нуде додатке функционалности. Ова библиотека се најчешће користи за манипулацију HTML DOM-а, за управљање догађајима, CSS анимације и слање Ajax позива. Коришћењем JQuery-а, веб странице постају интерактивније, једноставније и унапређују корисничко искуство. Карактер $ се користи за дефинисање и приступ библиотеци, и овај селектор проналази елементе и предузима одговарајућу акцију [28].

Асинхрони JavaScript и XML – Ajax представља скуп технологија које се користе за развој веб апликација. Комбинацијом ових технологија, веб странице постају одзивне (енгл. responsive), јер се мали пакети информација шаљу до сервера, чиме се веб странице не учитавају поново сваки пут када корисник унесе неку измену. На овај начин, корисник ступа у интеракцију са апликацијом без сталног поновног учитавања страница. Ова интеракција се одвија јако брзо и само се делови веб страница поново учитавају и освежавају. Код традиционалних веб апликација, HTTP захтеви које су иницирани од стране корисникове интеракције са корисничким интерфејсом, се шаљу на сервер. Сервер затим обрађује примљен захтев и враћа HTML страницу назад до клијента (веб претраживача). Током HTTP транспорта, корисник не може да ступи и другу интеракцију са веб апликацијом [29].



Слика 2 - Комуникација код класичних веб апликација

Код Ajax апликација, корисникова интеракција са веб апликацијом се не прекида. Ajax-ов engine или JavaScript интерпретор омогућава независан транспорт HTTP захтева и одговора од интеракције корисника и апликације тако што приказује интерфејс и управља комуникацијама са сервером [29].



Слика 3 - Комуникација коришћењем Ajax технологије

3. Студијски пример

На студијском примеру приказан је развој софтверског система за управљање кућним буџетом коришћењем упрошћене Ларманове методе развоја софтвера. Развој је представљен кроз пет фаза, које су уједно и наредна поглавља овог рада: прикупљање корисничких захтева, анализа, пројектовање, имплементација и тестирање.

4. Кориснички захтеви

У фази прикупљања захтева се дефинишу својства и услови које софтверски систем или шире гледајући пројекат треба да задовољи. Захтеви се често категоризују као функционални и нефункционални захтеви. Функционални захтеви дефинишу захтеване функције система, док нефункционални захтеви дефинишу све остале захтеве (Влајић, 2015).

4.1 Вербални опис модела

Софтверски систем за управљање кућним буџетом је замишљен као помоћ у вођењу евиденције личних финансија корисника, односно њихових банковних рачуна и трансакција. Основне функционалности које софтверски систем омогућава регистрованим и пријављеним корисницима су рад са рачунима – њихово креирање, измена и брисање. Један рачун може имати више платних картица.

Са сваког рачуна може се креирати трансакција, односно уплата или исплата на рачун. Поред креирања, корисник има могућност претраге, измене и брисања трансакција. На овај начин корисник има приступ евиденцији својих трошкова и прихода, али и укупном износу на рачуну и колико новца је утрошено или приходовано за сваку од категорија трансакција.

Трансакција, поред корисниковог рачуна, може бити повезана и са другим рачуном, односно рачуном примаоца или уплатиоца – уколико су регистровани у систему.

Како би се побољшало корисничко искуство приликом коришћења система, корисници имају могућност креирања, измене и брисања шаблона трансакција. На овај начин могу поновити претходне трансакције.

4.2 Спецификација захтева помоћу модела случајева коришћења

Захтеви се код упрошћене Ларманове методе описују помоћу UML Модела случаја коришћења (Use-Case Model). Модел случаја коришћења се састоји од скупа случаја коришћења (СК), актора (АК) и веза између случаја коришћења и актора. Случај коришћења описује скуп сценарија (use-case појављивања), односно скуп жељених коришћења система од стране актора. Актор (учесник) представља спољног корисника система (Влајић, 2015).

Код корисника, као једној од две могуће улоге актора у систему, идентификовани су следећи случајеви коришћења:

1. Креирање рачуна
2. Измена рачуна
3. Брисање рачуна
4. Креирање трансакције
5. Претрага трансакција
6. Измена трансакције
7. Брисање трансакције
8. Креирање шаблона трансакције
9. Измена шаблона трансакције
10. Брисање шаблона трансакције

Слика 4 - Дијаграм случајева коришћења

СК1: Случај коришћења – Креирање рачуна

Назив СК: Креирање рачуна

Актори СК: Корисник

Учесници СК: Корисник и систем (програм)

Предуслов: Систем је укључен и корисник је улогован под својом шифром. Систем приказује форму за рад са рачуном.

Основни сценарио СК

1. Корисник **уноси** податке о рачуну. (АПУСО)
2. Корисник **контролише** да ли је коректно унео податке о рачуну. (АНСО)
3. Корисник **позива** систем да запамти рачун. (АПСО)
4. Систем **памти** рачун. (СО)
5. Систем **приказује** кориснику запамћени рачун и поруку: "Систем је запамтио рачун". (ИА)

Алтернативна сценарија

5.1 Уколико систем не може да креира рачун он приказује кориснику поруку "Систем не може да запамти рачун". (ИА)

СК2: Случај коришћења – Измена рачуна

Назив СК: Измена рачуна

Актори СК: Корисник

Учесници СК: Корисник и систем (програм)

Предуслов: Систем је укључен и корисник је улогован под својом шифром. Систем приказује форму за рад са рачуном. Учитана је листа корисникових рачуна.

Основни сценарио СК

1. Корисник **уноси** вредност по којој претражује рачуне. (АПУСО)
2. Корисник **позива** систем да нађе рачуне по задатој вредности. (АПСО)
3. Систем **тражи** рачуне по задатој вредности. (СО)
4. Систем **приказује** кориснике рачуне и поруку: "Систем је нашао рачуне по задатој вредности". (ИА)
5. Корисник **бира** рачун. (АПУСО)
6. Корисник **позива** систем да учита рачун. (АПСО)
7. Систем **учитава** рачун. (СО)
8. Систем **показује** кориснику податке о рачуну и поруку "Систем је учитао рачун" (ИА)
9. Корисник **уноси** (мења) податке о рачуну. (АПУСО)
10. Корисник **контролише** да ли је коректно унео податке о рачуну. (АНСО)
11. Корисник **позива** систем да запамти податке о рачуну. (АПСО)
12. Систем **памти** податке о рачуну. (СО)
13. Систем **приказује** кориснику запамћени рачун и поруку: "Систем је запамтио рачун." (ИА)

Алтернативна сценарија

4.1 Уколико систем не може да нађе рачуне он приказује кориснику поруку: "Систем не може да нађе рачуне по задатој вредности". Прекида се извршење сценариа. (ИА)

8.1 Уколико систем не може да учита рачун он приказује кориснику поруку "Систем не може да учита рачун". Прекида се извршење сценариа. (ИА)

13.1 Уколико систем не може да запамти податке о рачуну он приказује кориснику поруку "Систем не може да запамти рачун". (ИА)

СК3: Случај коришћења – Брисање рачуна

Назив СК: Брисање рачуна

Актори СК: Корисник

Учесници СК: Корисник и систем (програм)

Предуслов: Систем је укључен и корисник је улогован под својом шифром. Систем приказује форму за рад са рачуном. Учитана је листа корисникових рачуна.

Основни сценарио СК

1. Корисник **уноси** вредност по којој претражује рачуне. (АПУСО)
2. Корисник **позива** систем да нађе рачуне по задатој вредности. (АПСО)
3. Систем **тражи** рачуне по задатој вредности. (СО)

4. Систем **приказује** кориснику рачуне и поруку: “Систем је нашао рачуне по задатој вредности”. (ИА)
5. Корисник **бира** рачун који жели да обрише. (АПУСО)
6. Корисник **позива** систем да учита податке о одабраном рачуну. (АПСО)
7. Систем **учитава** податке о одабраном рачуну. (СО)
8. Систем **обавештава** корисника о успешном учитавању података поруком “Систем је учитао одабран рачун“ и приказује податке о рачуну. (ИА)
9. Корисник **позива** систем да обрише рачун. (АПСО)
10. Систем **брише** рачун. (СО)
11. Систем **приказује** кориснику поруку: “Систем је обрисао рачун.” (ИА)

Алтернативна сценарија

4.1 Уколико систем не може да нађе рачун он приказује кориснику поруку: “Систем не може да нађе рачун по задатој вредности”. Прекида се извршење сценариа. (ИА)

8.1 Уколико систем не може да учита изабран рачун он приказује кориснику поруку “Систем не може да учита рачун“. Прекида се извршење сценариа (ИА)

11.1 Уколико систем не може да обрише рачун он приказује кориснику поруку “Систем не може да обрише рачун”. (ИА)

СК4: Случај коришћења – Креирање трансакције

Назив СК: Креирање трансакције

Актори СК: Корисник

Учесници СК: Корисник и систем (програм)

Предуслов: Систем је укључен и корисник је улогован под својом шифром. Систем приказује форму за рад са трансакцијом. Учитана је листа категорија.

Основни сценарио СК

1. Корисник **уноси** податке о трансакцији. (АПУСО)
2. Корисник **контролише** да ли је коректно унео податке о трансакцији. (АНСО)
3. Корисник **позива** систем да запамти трансакцију. (АПСО)
4. Систем **памти** трансакцију. (СО)
5. Систем **приказује** кориснику запамћену трансакцију и поруку: “Систем је запамтио трансакцију“. (ИА)

Алтернативна сценарија

5.1 Уколико систем не може да запамти податке о трансакцији он приказује кориснику поруку “Систем не може да запамти трансакцију”. (ИА)

СК5: Случај коришћења – Претрага трансакција

Назив СК: Претраживање трансакција

Актори СК: Корисник

Учесници СК: Корисник и систем (програм)

Предуслов: Систем је укључен и корисник је улогован под својом шифром. Систем приказује форму за рад са трансакцијама. Учитане су листе категорија и трансакција.

Основни сценарио СК

1. Корисник **уноси** вредност по којој претражује трансакције. (АПУСО)
2. Корисник **позива** систем да нађе трансакције по задатој вредности. (АПСО)
3. Систем **тражи** трансакције по задатој вредности. (СО)
4. Систем **приказује** кориснику податке о трансакцијама и поруку: “Систем је нашао трансакције по задатој вредности”. (ИА)
5. Корисник **бира** трансакцију. (АПУСО)
6. Корисник **позива** систем да учита трансакцију. (АПСО)
7. Систем **учитава** трансакцију. (СО)
8. Систем **приказује** кориснику податке о трансакцији и поруку: “Систем је учитао трансакцију”. (ИА)

Алтернативна сценарија

4.1 Уколико систем не може да нађе трансакције он приказује кориснику поруку: “Систем не може да нађе трансакције по задатој вредности”. Прекида се извршење сценариа. (ИА)

8.1 Уколико систем не може да учита трансакцију он приказује кориснику поруку “Систем не може да учита трансакцију”. (ИА)

СК6: Случај коришћења – Измена трансакција

Назив СК: Измена трансакција

Актори СК: Корисник

Учесници СК: Корисник и систем (програм)

Предуслов: Систем је укључен и корисник је улогован под својом шифром. Систем приказује форму за рад са трансакцијом. Учитане су листе трансакција и категорија.

Основни сценарио СК

1. Корисник **уноси** вредност по којој претражује трансакције. (АПУСО)
2. Корисник **позива** систем да нађе трансакције по задатој вредности. (АПСО)
3. Систем **тражи** трансакције по задатој вредности. (СО)
4. Систем **приказује** кориснику трансакције и поруку: "Систем је нашао трансакције по задатој вредности". (ИА)
5. Корисник **бира** трансакцију. (АПУСО)
6. Корисник **позива** систем да учита трансакцију. (АПСО)
7. Систем **учитава** трансакцију. (СО)
8. Систем **показује** кориснику податке о трансакцији и поруку "Систем је учитао трансакцију" (ИА)
9. Корисник **уноси** (мења) податке о трансакцији. (АПУСО)
10. Корисник **контролише** да ли је коректно унео податке о трансакцији. (АНСО)
11. Корисник **позива** систем да запамти податке о трансакцији. (АПСО)
12. Систем **памти** податке о трансакцији. (СО)
13. Систем **приказује** кориснику запамћену трансакцију и поруку: "Систем је запамтио трансакцију." (ИА)

Алтернативна сценарија

4.1 Уколико систем не може да нађе трансакције он приказује кориснику поруку: "Систем не може да нађе трансакције по задатој вредности". Прекида се извршење сценариа. (ИА)

8.1 Уколико систем не може да учита трансакцију он приказује кориснику поруку "Систем не може да учита трансакцију". Прекида се извршење сценариа. (ИА)

13.1 Уколико систем не може да запамти податке о трансакцији он приказује кориснику поруку "Систем не може да запамти трансакцију". (ИА)

СК7: Случај коришћења – Брисање трансакције

Назив СК: Брисање трансакције

Актори СК: Корисник

Учесници СК: Корисник и систем (програм)

Предуслов: Систем је укључен и корисник је улогован под својом шифром. Систем приказује форму за рад са трансакцијом. Учитане су листе категорија и трансакција.

Основни сценарио СК

1. Корисник **уноси** вредност по којој претражује трансакције. (АПУСО)
2. Корисник **позива** систем да нађе трансакције по задатој вредности. (АПСО)
3. Систем **тражи** трансакције по задатој вредности. (СО)
4. Систем **приказује** кориснику трансакције и поруку: "Систем је нашао трансакције по задатој вредности". (ИА)
5. Корисник **бира** трансакцију који жели да обрише. (АПУСО)
6. Корисник **позива** систем да учита податке о одабраној трансакцији. (АПСО)
7. Систем **учитава** податке о одабраном трансакцији. (СО)
8. Систем **обавештава** корисника о успешном учитавању података поруком "Систем је учитао одабран трансакцију" и приказује податке о трансакцији. (ИА)
9. Корисник **позива** систем да обрише трансакцију. (АПСО)
10. Систем **брише** трансакцију. (СО)
11. Систем **приказује** кориснику поруку: "Систем је обрисао трансакцију." (ИА)

Алтернативна сценарија

4.1 Уколико систем не може да нађе трансакције он приказује кориснику поруку: "Систем не може да нађе трансакције по задатој вредности". Прекида се извршење сценариа. (ИА)

8.1 Уколико систем не може да учита изабран трансакцију он приказује кориснику поруку "Систем не може да учита трансакцију". Прекида се извршење сценариа (ИА)

11.1 Уколико систем не може да обрише трансакцију он приказује кориснику поруку "Систем не може да обрише трансакцију". (ИА)

СК8: Случај коришћења – Креирање шаблона трансакције

Назив СК: Креирање шаблона трансакције

Актори СК: Корисник

Учесници СК: Корисник и систем (програм)

Предуслов: Систем је укључен и корисник је улогован под својом шифром. Систем приказује форму за рад са шаблоном. Учитана је листа категорија.

Основни сценарио СК

1. Корисник **уноси** податке о шаблону. (АПУСО)
2. Корисник **контролише** да ли је коректно унео податке о шаблону. (АНСО)
3. Корисник **позива** систем да запамти шаблон. (АПСО)
4. Систем **памти** шаблон. (СО)
5. Систем **приказује** кориснику запамћени шаблон и поруку: "Систем је запамтио шаблон". (ИА)

Алтернативна сценарија

5.1 Уколико систем не може да запамти податке о шаблону он приказује кориснику поруку "Систем не може да запамти шаблон". (ИА)

СК9: Случај коришћења – Измена шаблона трансакције

Назив СК: Измена шаблона трансакције

Актори СК: Корисник

Учесници СК: Корисник и систем (програм)

Предуслов: Систем је укључен и корисник је улогован под својом шифром. Систем приказује форму за рад са шаблоном. Учитане су листа шаблона и категорија.

Основни сценарио СК

1. Корисник **уноси** вредност по којој претражује шаблоне. (АПУСО)
2. Корисник **позива** систем да нађе шаблоне по задатој вредности. (АПСО)
3. Систем **тражи** шаблоне по задатој вредности. (СО)
4. Систем **приказује** кориснику шаблоне и поруку: "Систем је нашао шаблоне по задатој вредности". (ИА)
5. Корисник **бира** шаблон. (АПУСО)
6. Корисник **позива** систем да учита шаблон. (АПСО)
7. Систем **учитава** шаблон. (СО)
8. Систем **показује** кориснику податке о шаблону и поруку "Систем је учитао шаблон" (ИА)
9. Корисник **уноси** (мења) податке о шаблону. (АПУСО)
10. Корисник **контролише** да ли је коректно унео податке о шаблону. (АНСО)
11. Корисник **позива** систем да запамти податке о шаблону. (АПСО)
12. Систем **памти** податке о шаблону. (СО)
13. Систем **приказује** кориснику запамћени шаблон и поруку: "Систем је запамтио шаблон." (ИА)

Алтернативна сценарија

4.1 Уколико систем не може да нађе шаблоне он приказује кориснику поруку: "Систем не може да нађе шаблоне по задатој вредности". Прекида се извршење сценариа. (ИА)

8.1 Уколико систем не може да учита шаблон он приказује кориснику поруку "Систем не може да учита шаблон". Прекида се извршење сценариа. (ИА)

13.1 Уколико систем не може да запамти податке о шаблону он приказује кориснику поруку "Систем не може да запамти шаблон". (ИА)

СК10: Случај коришћења – Брисање шаблона трансакције

Назив СК: Брисање шаблона трансакције

Актори СК: Корисник

Учесници СК: Корисник и систем (програм)

Предуслов: Систем је укључен и корисник је улогован под својом шифром. Систем приказује форму за рад са шаблоном. Учитана је листа шаблона.

Основни сценарио СК

1. Корисник **уноси** вредност по којој претражује шаблоне. (АПУСО)
2. Корисник **позива** систем да нађе шаблоне по задатој вредности. (АПСО)
3. Систем **тражи** шаблоне по задатој вредности. (СО)
4. Систем **приказује** кориснику шаблоне и поруку: “Систем је нашао шаблоне по задатој вредности”. (ИА)
5. Корисник **бира** шаблон који жели да обрише. (АПУСО)
6. Корисник **позива** систем да учита податке о одабраном шаблону. (АПСО)
7. Систем **учитава** податке о одабраном шаблону. (СО)
8. Систем **обавештава** корисника о успешном учитавању података поруком “Систем је учитао одабран шаблон“ и приказује податке о шаблону. (ИА)
9. Корисник **позива** систем да обрише шаблон. (АПСО)
10. Систем **брише** шаблон. (СО)
11. Систем **приказује** кориснику поруку: “Систем је обрисао шаблон.” (ИА)

Алтернативна сценарија

4.1 Уколико систем не може да нађе шаблон он приказује кориснику поруку: “Систем не може да нађе шаблон по задатој вредности”. Прекида се извршење сценариа. (ИА)

8.1 Уколико систем не може да учита изабран шаблон он приказује кориснику поруку “Систем не може да учита шаблон“. Прекида се извршење сценариа (ИА)

11.1 Уколико систем не може да обрише шаблон он приказује кориснику поруку “Систем не може да обрише шаблон”. (ИА)

5. Фаза анализе

Фаза анализе описује логичку структуру и понашање софтверског система (пословну логику софтверског система). Понашање софтверског система је описано помоћу системских дијаграма секвенци и преко системских операција. Структура софтверског система се описује помоћу концептуалног и релационог модела (Влајић, 2015).

5.1 Понашање софтверског система – Системски дијаграми секвенци

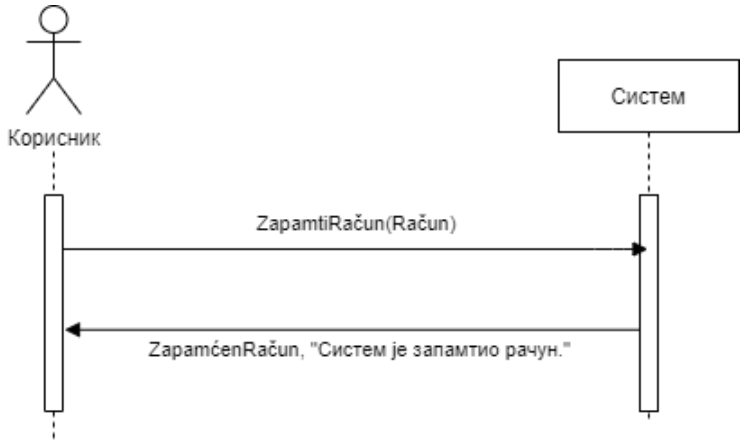
За сваки од сценариа идентификованих у оквиру случајева коришћења у фази прикупљања корисничких захтева се креира UML дијаграм секвенци.

Системски дијаграм секвенци приказује, за издвојени сценарио случаја коришћења, догађаје у одређеном редоследу, који успостављају интеракцију између актора и софтверског система. Догађаје праве актори (нпр. клик на дугме, које се налази на екранској форми), у оквиру APSO акција, над примаоцем догађаја (нпр. дугме). Прималац догађаја прихвата догађај и позива системску операцију која се налази на страни система. Након извршења системске операције систем враћа неки резултат као одговор на догађај - то може да буде сигнал о успешности извршења операције и/или неки податак (Влајић. 2015).

ДС1: Дијаграм секвенци случаја коришћења – Креирање рачуна

Основни сценарио

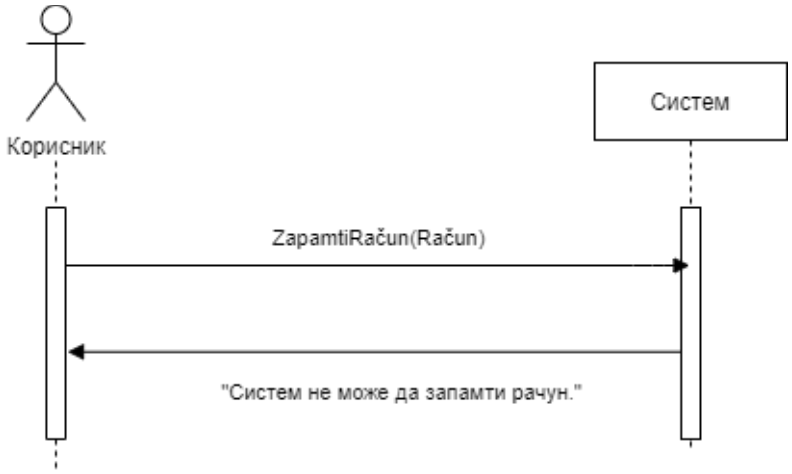
1. Корисник **позива** систем да запамти податке о рачуну. (АПСО)
2. Систем **приказује** кориснику запамћени рачун и поруку: “Систем је запамтио рачун“. (ИА)



Слика 5 - ДС Креирање рачуна

Алтернативна сценарија

2.1 Уколико систем не може да запамти податке о рачуну он приказује кориснику поруку “Систем не може да запамти рачун”. (ИА)



Слика 6 - ДС Неуспешно креирање рачуна

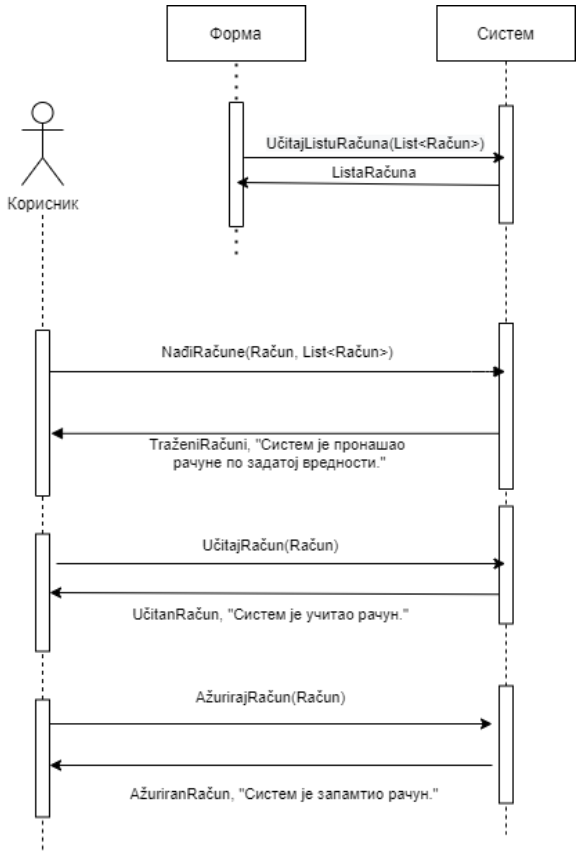
Са наведених секвенцијалних дијаграма уочена је једна системска операција:

1. Signal **ZapamtiRačun(Račun)**

ДС2: Дијаграм секвенци случаја коришћења – Измена рачуна

Основни сценарио

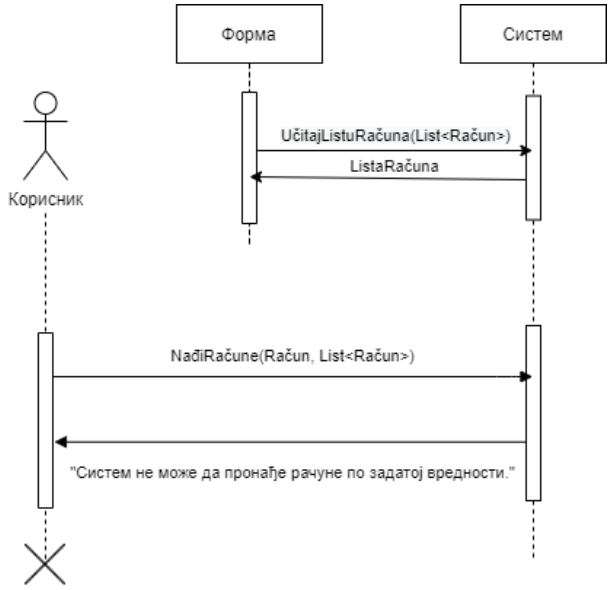
1. Форма **позива** систем да учита листу рачуна. (АПСО)
2. Систем **враћа** форми листу рачуна. (ИА)
3. Корисник **позива** систем да нађе рачуне по задатој вредности. (АПСО)
4. Систем **приказује** кориснику рачуне и поруку: “Систем је пронашао рачуне по задатој вредности”. (ИА)
5. Корисник **позива** систем да учита рачун. (АПСО)
6. Систем **показује** кориснику податке о рачуну и поруку “Систем је учитао рачун“ (ИА)
7. Корисник **позива** систем да запамти податке о рачуну. (АПСО)
8. Систем **приказује** кориснику запамћени рачун и поруку: “Систем је запамтио рачун.” (ИА)



Слика 7 - ДС Измена рачуна

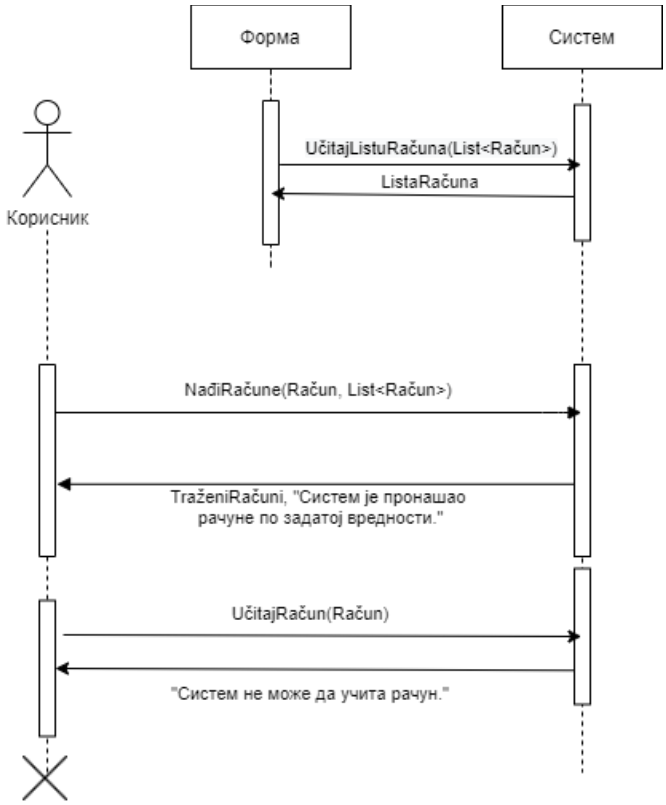
Алтернативна сценарија

4.1 Уколико систем не може да нађе рачуне он приказује кориснику поруку: “Систем не може да нађе рачуне по задатој вредности”. Прекида се извршење сценариа. (ИА)



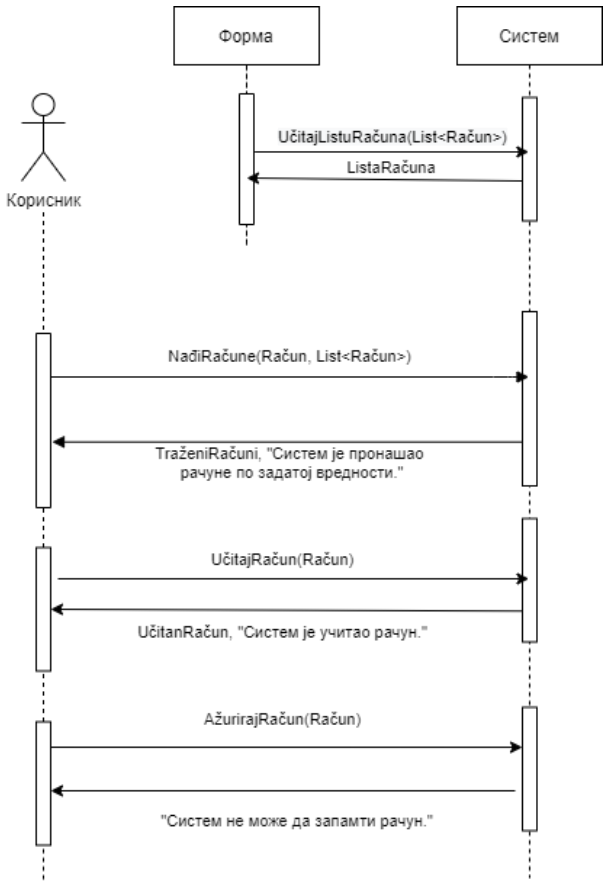
Слика 8 - ДС Неуспешна претрага рачуна

6.1 Уколико систем не може да учита рачун он приказује кориснику поруку “Систем не може да учита рачун”. Прекида се извршење сценариа. (ИА)



Слика 9 - ДС Неуспешно учитавање рачуна

8.1 Уколико систем не може да запамти податке о рачуну он приказује кориснику поруку “Систем не може да запамти рачун”. (ИА)



Слика 10 - ДС Неуспешна измена рачуна

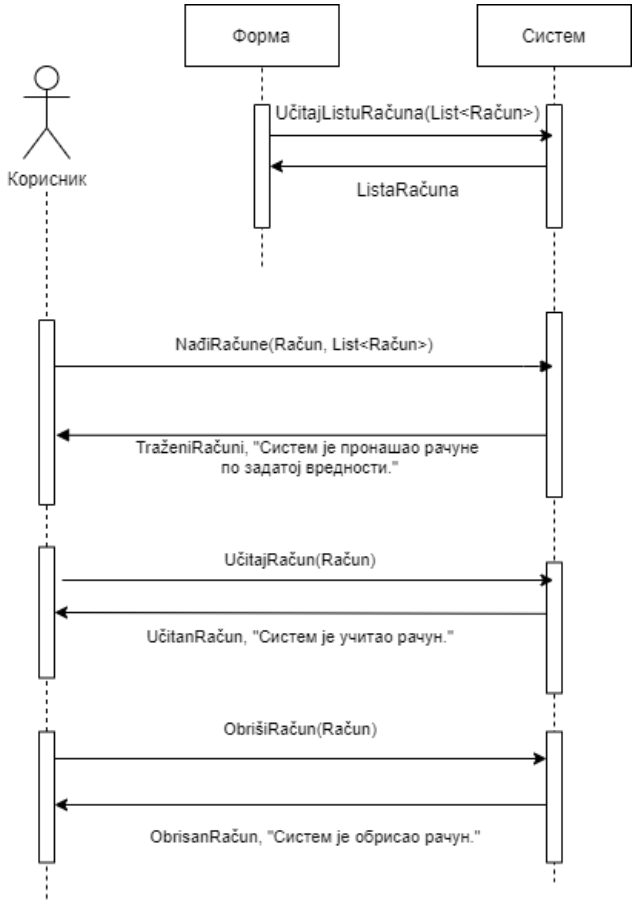
Са наведених секвенцијалних дијаграма уочено је четири системске операције:

1. Signal **УчитајЛистуРачуна(Лista<Рачун>)**
2. Signal **НађиРачуне(Рачун, Лista<Рачун>)**
3. Signal **УчитајРачун(Рачун)**
4. Signal **АжурирајРачун(Рачун)**

ДС3: Дијаграм секвенци случаја коришћења – Брисање рачуна

Основни сценарио

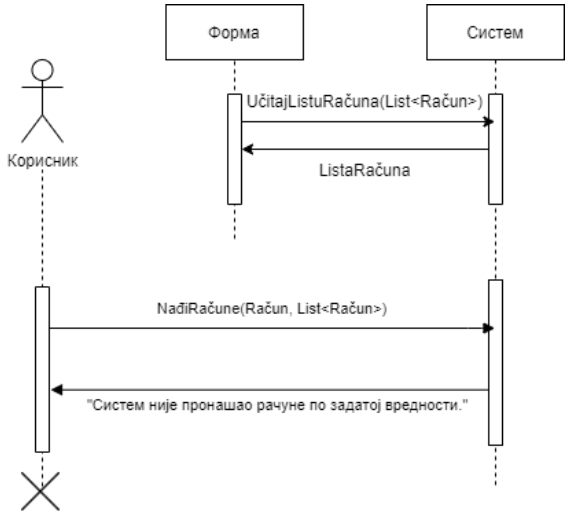
1. Форма **позива** систем да учита листу рачуна. (АПСО)
2. Систем **враћа** форми рачуна. (ИА)
3. Корисник **позива** систем да нађе рачуне по задатој вредности. (АПСО)
4. Систем **приказује** кориснику рачуне и поруку: “Систем је пронашао рачуне по задатој вредности”. (ИА)
5. Корисник **позива** систем да учита податке о одабраном рачуну. (АПСО)
6. Систем **обавештава** корисника о успешном учитавању података поруком “Систем је учитао рачун“ и приказује податке о рачуну. (ИА)
7. Корисник **позива** систем да обрише рачун. (АПСО)
8. Систем **приказује** кориснику поруку: “Систем је обрисао рачун.” (ИА)



Слика 11 - ДС Брисање рачуна

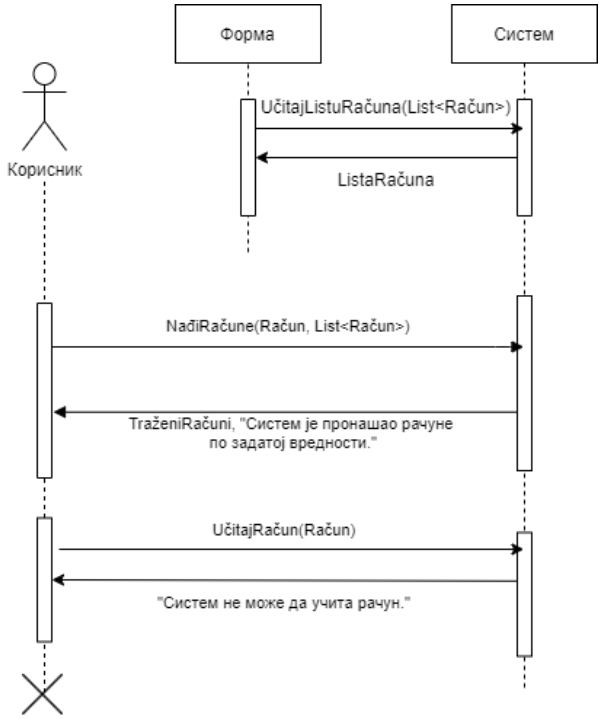
Алтернативна сценарија

4.1 Уколико систем не може да нађе рачуне он приказује кориснику поруку: “Систем није пронашао рачуне по задатој вредности”. Прекида се извршење сценариа. (ИА)



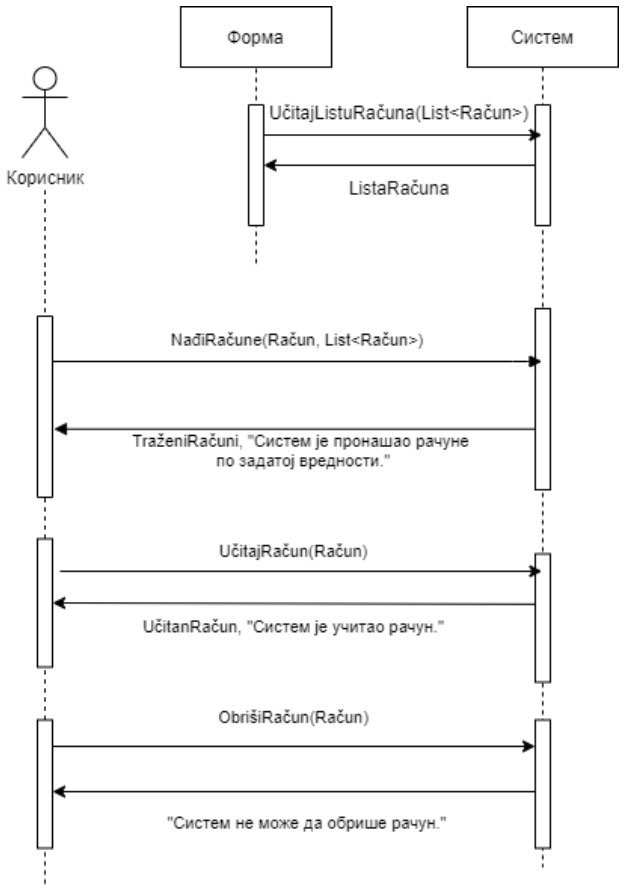
Слика 12 - ДС Неуспешна претрага приликом брисања рачуна

6.1 Уколико систем не може да учита изабран рачун он приказује кориснику поруку “Систем не може да учита рачун“. Прекида се извршење сценариа (ИА)



Слика 13 - ДС Неуспешно учитавање приликом брисања рачуна

8.1 Уколико систем не може да обрише рачун он приказује кориснику поруку “Систем не може да обрише рачун”. (ИА)



Слика 14 - ДС Неуспешно брисање рачуна

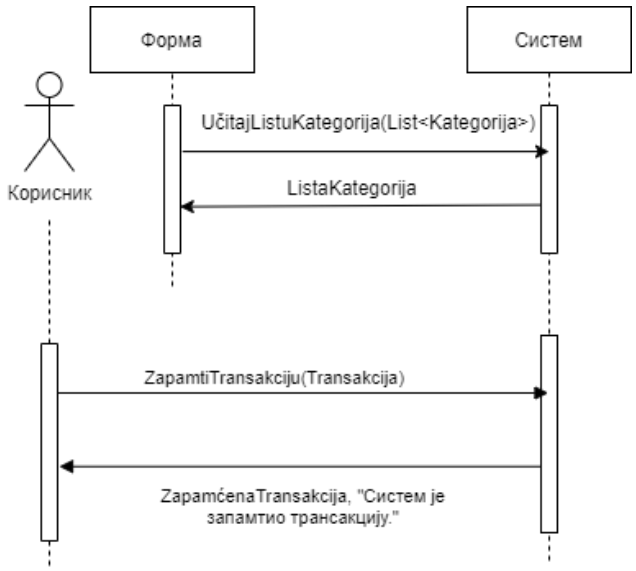
Са наведених секвенцијалних дијаграма уочене су четири системске операције:

1. Signal **УčitajListuRačuna(List<Račun>)**
2. Signal **NađiRačune(Račun, List<Račun>)**
3. Signal **УčitajRačun(Račun)**
4. Signal **ObrišiRačun(Račun)**

ДС4: Дијаграм секвенци случаја коришћења – Креирање трансакције

Основни сценарио

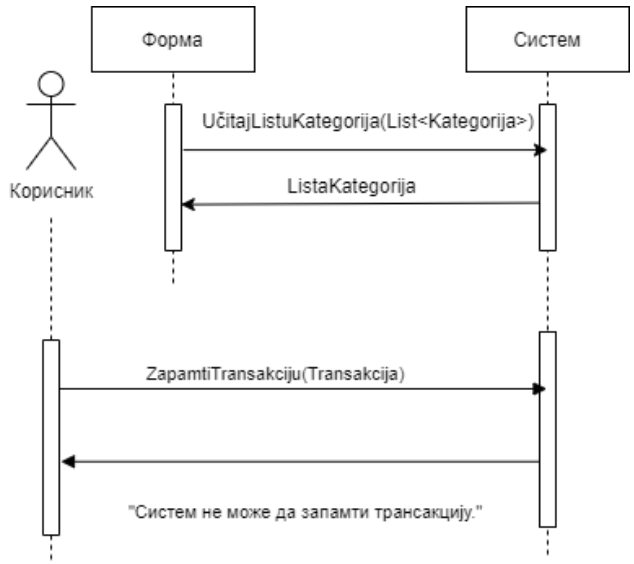
1. Форма **позива** систем да учита листу категорија. (АПСО)
2. Систем **враћа** форми листу категорија. (ИА)
3. Корисник **позива** систем да запамти податке о трансакцији. (АПСО)
4. Систем **приказује** кориснику запамћену трансакцију и поруку: “Систем је запамтио трансакцију“. (ИА)



Слика 15 - ДС Креирање трансакције

Алтернативна сценарија

4.1 Уколико систем не може да запамти податке о трансакцији он приказује кориснику поруку “Систем не може да запамти трансакцију”. (ИА)



Слика 16 - ДС Неуспешно креирање трансакције

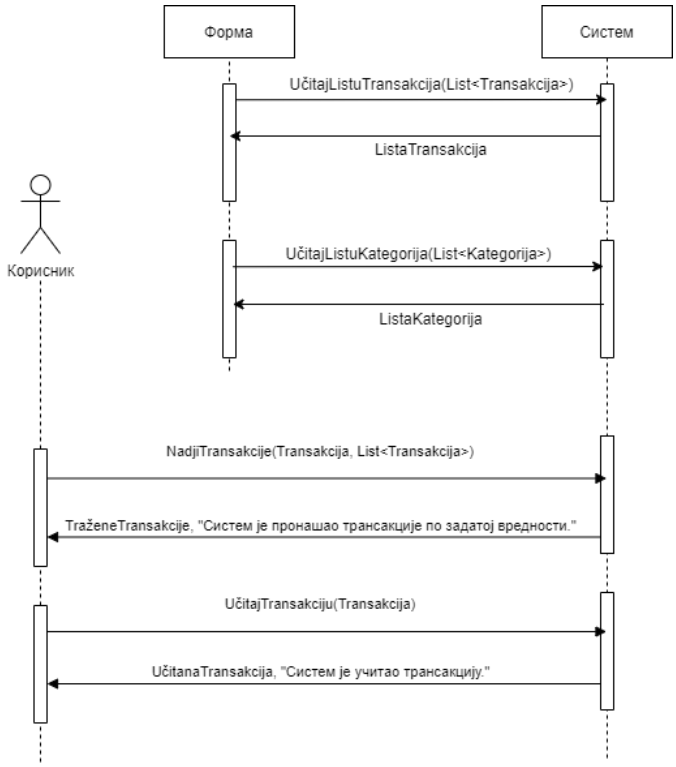
Са наведених секвенцијалних дијаграма уочене су две системске операције:

1. Signal **УčitajListuKategorija(List<Kategorija>)**
2. Signal **ZapamtiTransakciju(Transakcija)**

ДС5: Дијаграм секвенци случаја коришћења – Претрага трансакција

Основни сценарио

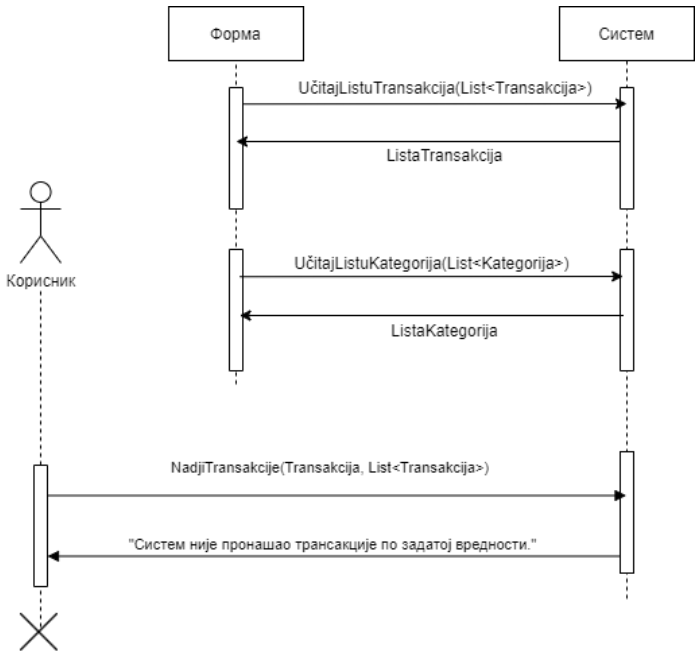
1. Форма **позива** систем да учита листу трансакција. (АПСО)
2. Систем **враћа** форми листу трансакција. (ИА)
3. Форма **позива** систем да учита листу категорија. (АПСО)
4. Систем **враћа** форми листу категорија. (ИА)
5. Корисник **позива** систем да нађе трансакције по задатој вредности. (АПСО)
6. Систем **приказује** кориснику податке о трансакцијама и поруку: “Систем је пронашао трансакције по задатој вредности”. (ИА)
7. Корисник **позива** систем да учита трансакцију. (АПСО)
8. Систем **приказује** кориснику податке о трансакцији и поруку: “Систем је учитао трансакцију”. (ИА)



Слика 17 - ДС Претрага трансакција

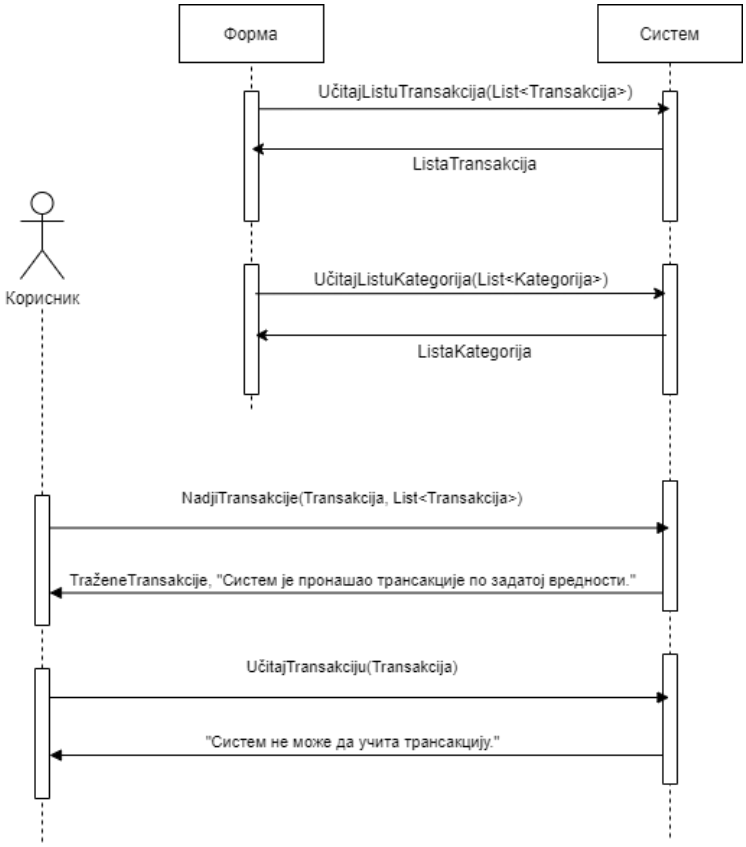
Алтернативна сценарија

6.1 Уколико систем не може да нађе трансакције он приказује кориснику поруку: “Систем није пронашао трансакције по задатој вредности”. Прекида се извршење сценариа. (ИА)



Слика 18 - ДС Неуспешна претрага трансакција

8.1 Уколико систем не може да учита трансакцију он приказује кориснику поруку “Систем не може да учита трансакцију.“ (ИА)



Слика 19 – ДС Неуспешно учитавање трансакције

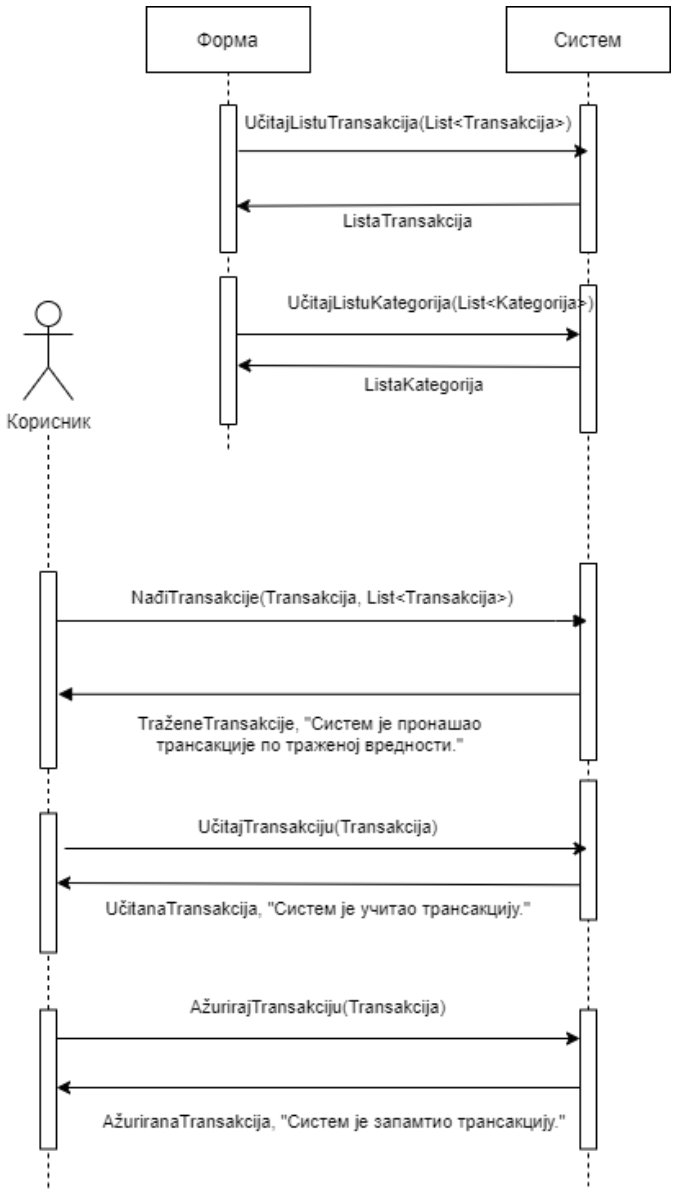
Са наведених секвенцијалних дијаграма уочене су четири системске операције:

1. Signal **УчитајЛистуТрансакција (List<Transakcija>)**
2. Signal **УчитајЛистуКатегорија(List<Kategorija>)**
3. Signal **НађиТрансакције(Transakcija List<Transakcija>)**
4. Signal **УчитајТрансакцију(Transakcija)**

ДС6: Дијаграм секвенци случаја коришћења – Измена трансакције

Основни сценарио

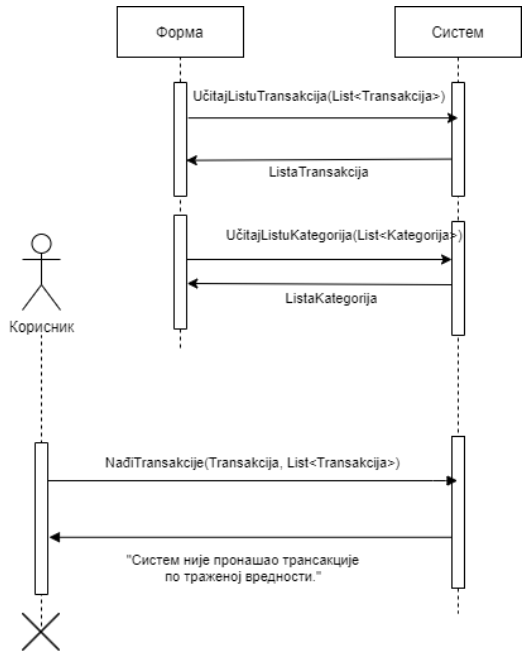
1. Форма **позива** систем да учита листу трансакција. (АПСО)
2. Систем **враћа** форми листу трансакција. (ИА)
3. Форма **позива** систем да учита листу категорија. (АПСО)
4. Систем **враћа** форми листу категорија. (ИА)
5. Корисник **позива** систем да нађе трансакције по задатој вредности. (АПСО)
6. Систем **приказује** кориснику трансакције и поруку: “Систем је нашао пронашао трансакције по задатој вредности”. (ИА)
7. Корисник **позива** систем да учита трансакцију. (АПСО)
8. Систем **показује** кориснику податке о трансакцији и поруку “Систем је учитао трансакцију.“ (ИА)
9. Корисник **позива** систем да запамти податке о трансакцији. (АПСО)
10. Систем **приказује** кориснику запамћену трансакцију и поруку: “Систем је запамтио трансакцију.” (ИА)



Слика 20 - ДС Измена трансакције

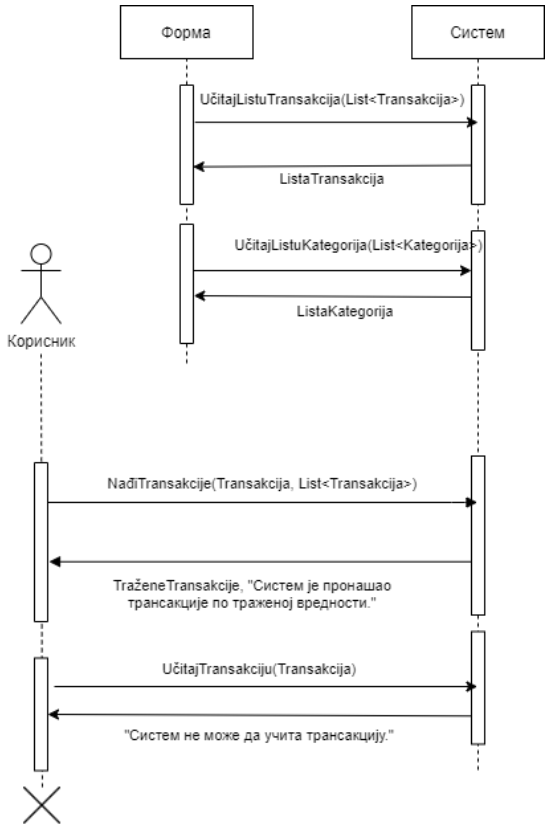
Алтернативна сценарија

6.1 Уколико систем не може да нађе трансакције он приказује кориснику поруку: “Систем није пронашао трансакције по задатој вредности”. Прекида се извршење сценариа. (ИА)



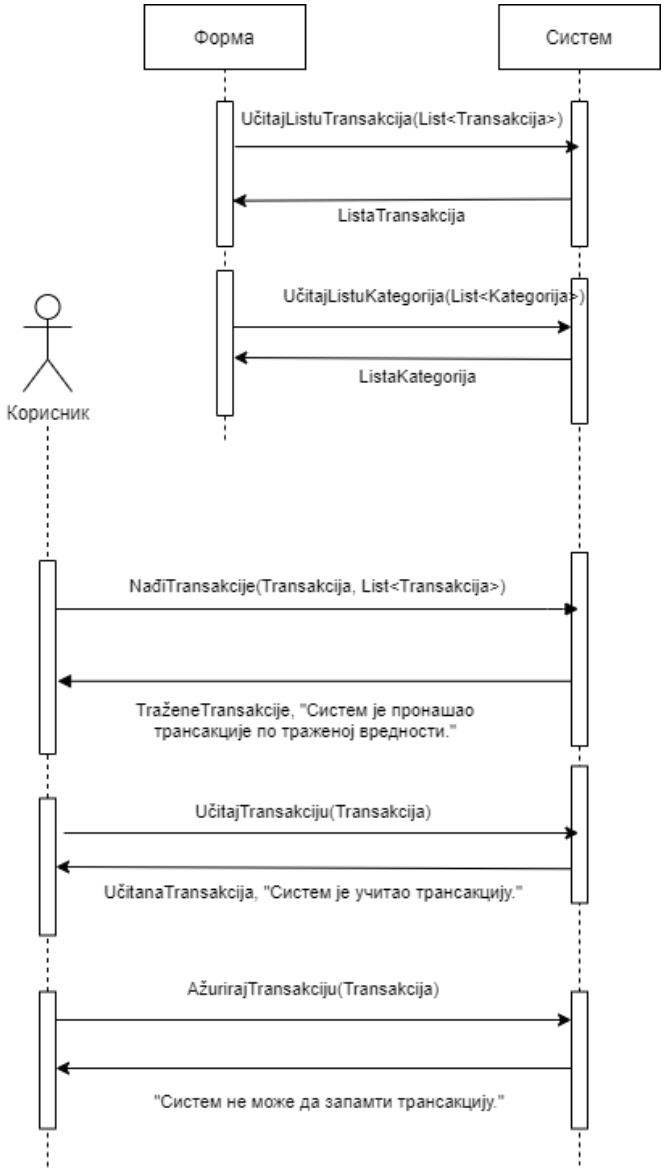
Слика 21 - ДС Неуспешна претрага приликом измене трансакције

8.1 Уколико систем не може да учита трансакцију он приказује кориснику поруку “Систем не може да учита трансакцију”. Прекида се извршење сценариа. (ИА)



Слика 22 - ДС Неуспешно учитавање приликом измене трансакције

10.1 Уколико систем не може да запамти податке о трансакцији он приказује кориснику поруку “Систем не може да запамти трансакцију”. (ИА)



Слика 23 - ДС Неуспешна измена трансакције

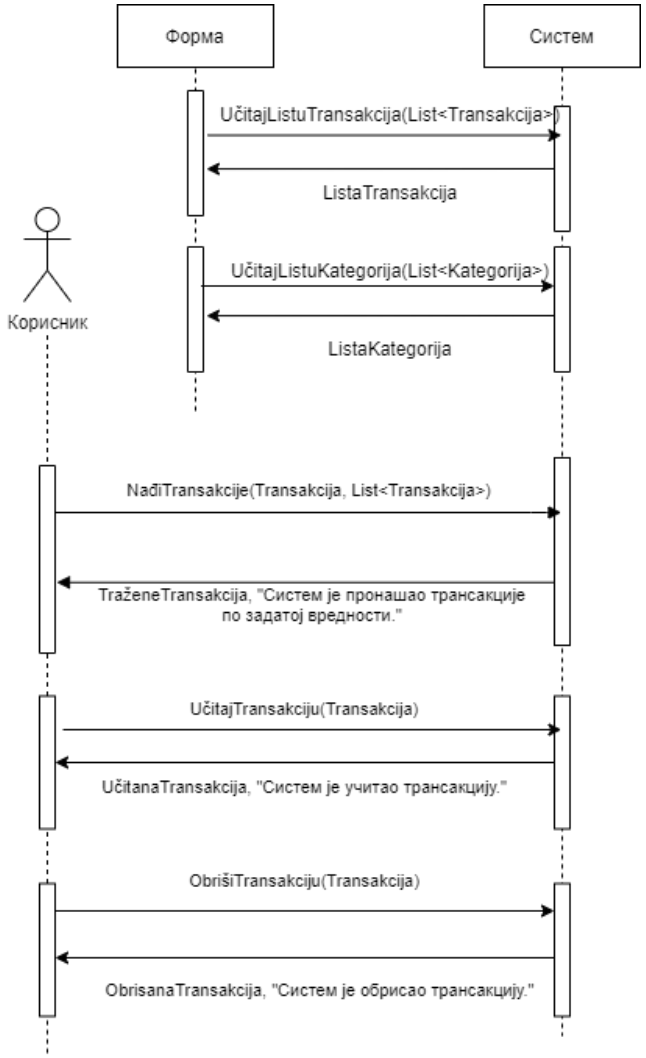
Са наведених секвенцијалних дијаграма уочено је пет системских операција:

1. Signal **УчитајЛистуТрансакција(List<Transakcija>)**
2. Signal **УчитајЛистуКатегорија(List<Kategorija>)**
3. Signal **НађиТрансакције (Transakcija, List< Transakcija >)**
4. Signal **УчитајТрансакцију(Transakcija)**
5. Signal **АжурирајТрансакцију(Transakcija)**

ДС7: Дијаграм секвенци случаја коришћења – Брисање трансакције

Основни сценарио

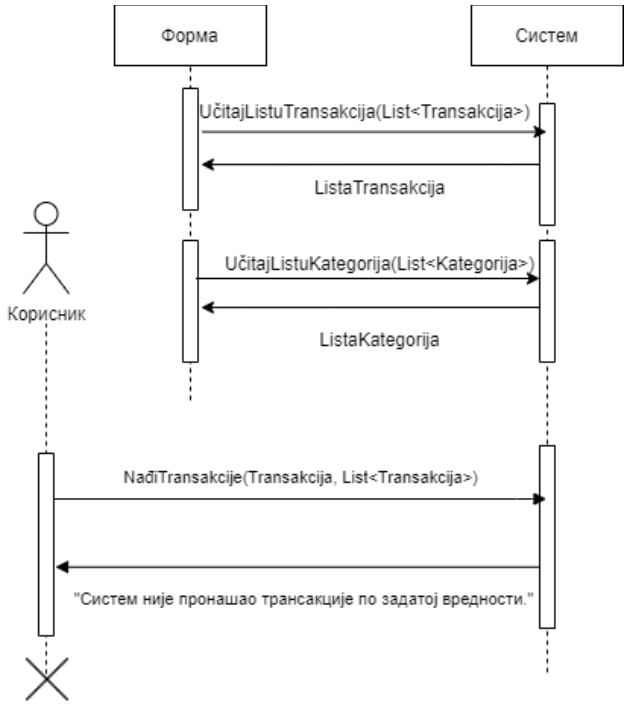
1. Форма **позива** систем да учита листу трансакција. (АПСО)
2. Систем **враћа** форми листу трансакција. (ИА)
3. Форма **позива** систем да учита листу категорија. (АПСО)
4. Систем **враћа** форми листу категорија. (ИА)
5. Корисник **позива** систем да нађе трансакције по задатој вредности. (АПСО)
6. Систем **приказује** кориснику трансакције и поруку: “Систем је пронашао трансакције по задатој вредности”. (ИА)
7. Корисник **позива** систем да учита податке о одабраном трансакцији. (АПСО)
8. Систем **обавештава** корисника о успешном учитавању података поруком “Систем је учитао трансакцију“ и приказује податке о трансакцији. (ИА)
9. Корисник **позива** систем да обрише трансакцију. (АПСО)
10. Систем **приказује** кориснику поруку: “Систем је обрисао трансакцију.” (ИА)



Слика 24 - ДС Брисање трансакције

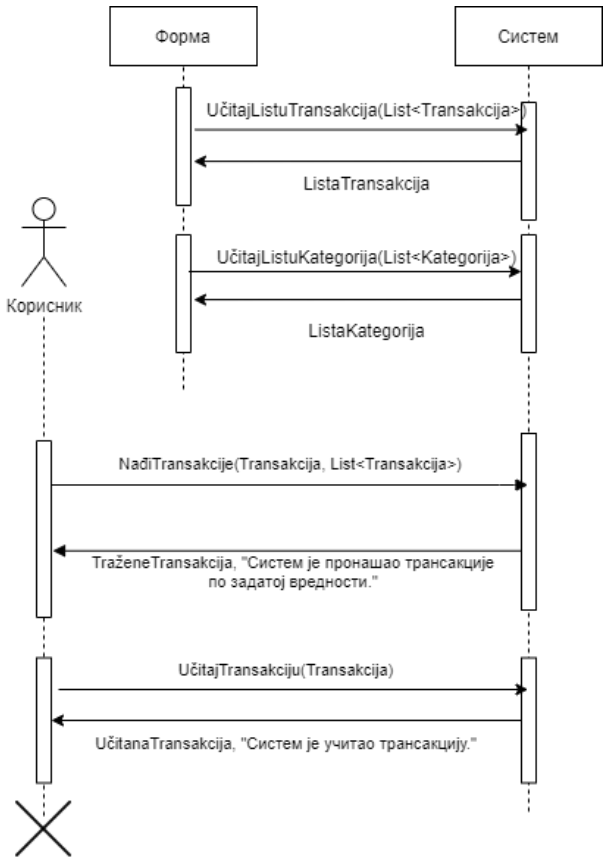
Алтернативна сценарија

6.1 Уколико систем не може да нађе трансакције он приказује кориснику поруку: “Систем не може да пронађе трансакције по задатој вредности”. Прекида се извршење сценариа. (ИА)



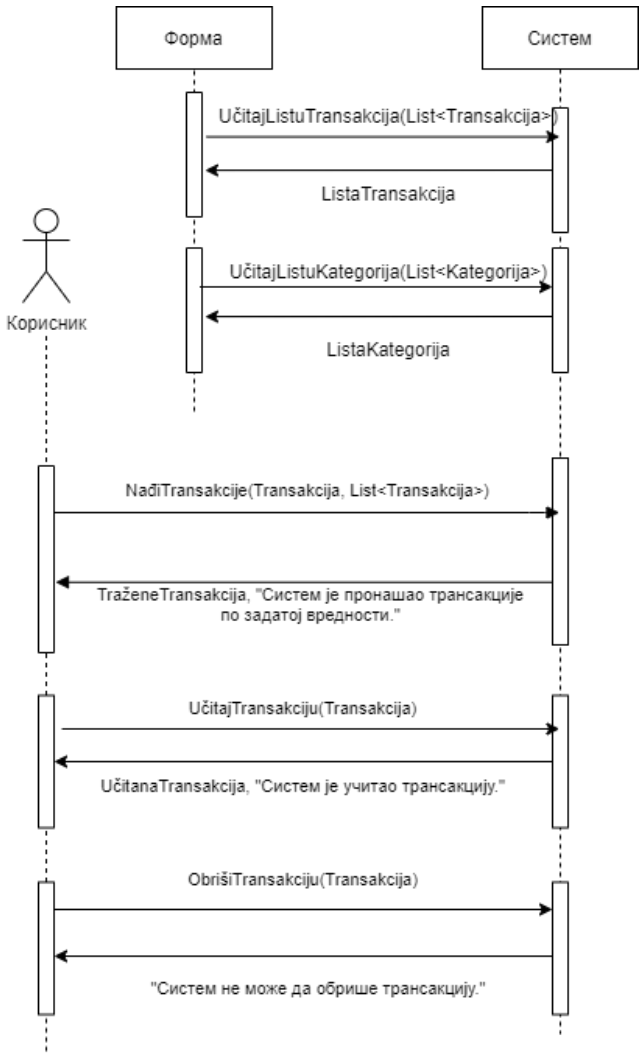
Слика 25 - ДС Неуспешна претрага приликом брисања трансакције

8.1 Уколико систем не може да учита изабран трансакцију, приказује кориснику поруку “Систем не може да учита трансакцију“. Прекида се извршење сценариа (ИА)



Слика 26 - ДС Неуспешно учитавање приликом брисања трансакције

10.1 Уколико систем не може да обрише трансакцију он приказује кориснику поруку “Систем не може да обрише трансакцију”. (ИА)



Слика 27 - ДС Неуспешно брисање трансакције

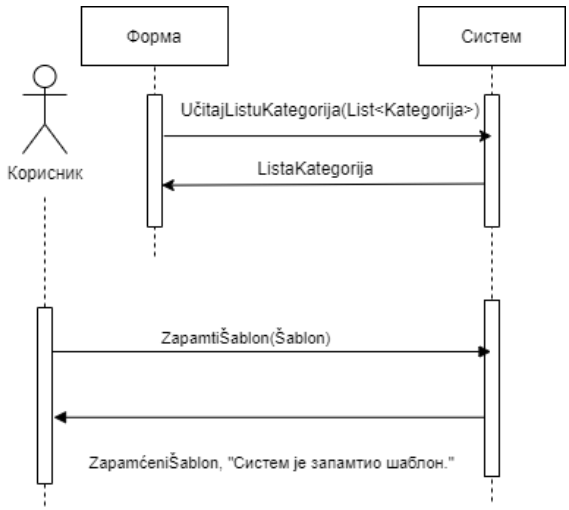
Са наведених секвенцијалних дијаграма уочене је пет системских операција:

1. Signal **УčitajListuTransakcija(List<Transakcija>)**
2. Signal **УčitajListuKategorija(List<Kategorija>)**
3. Signal **NađiTransakcije(Transakcija, List<Transakcija>)**
4. Signal **УčitajTransakciju(Transakcija)**
5. Signal **ObrišiTransakciju(Transakcija)**

ДС8: Дијаграм секвенци случаја коришћења – Креирање шаблона трансакција

Основни сценарио

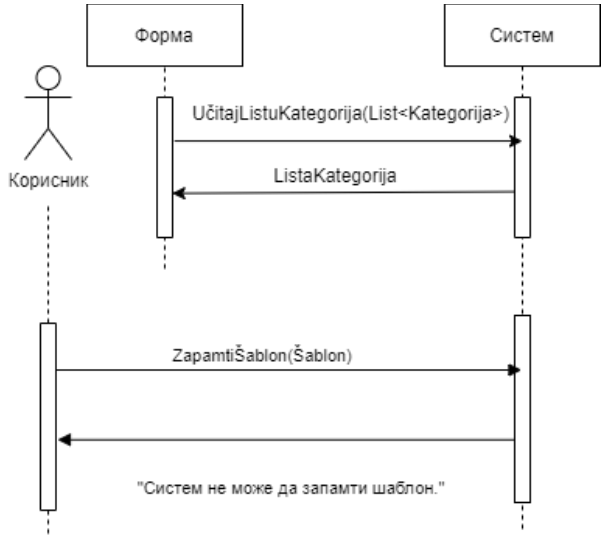
1. Форма **позива** систем да учита листу категорија. (АПСО)
2. Систем **враћа** форми листу категорија. (ИА)
3. Корисник **позива** систем да запамти податке о шаблону. (АПСО)
4. Систем **приказује** кориснику запамћен шаблон и поруку: “Систем је запамтио шаблон.“. (ИА)



Слика 28 - ДС Креирање шаблона трансакције

Алтернативна сценарија

4.1 Уколико систем не може да запамти податке о шаблону он приказује кориснику поруку “Систем не може да запамти шаблон”. (ИА)



Слика 29 - ДС Неуспешно креирање шаблона трансакције

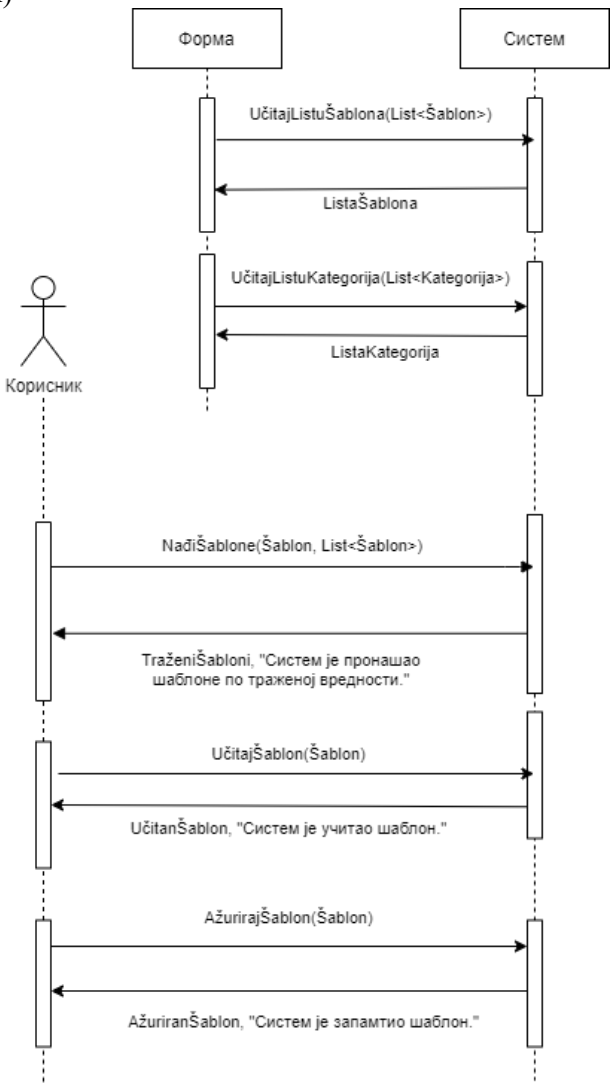
Са наведених секвенцијалних дијаграма уочене су две системске операције:

1. Signal **УчитајЛистуКатегорија(List<Kategorija>)**
2. Signal **ЗапамтиШаблон(Шаблон)**

ДС9: Дијаграм секвенци случаја коришћења – Измена шаблона трансакције

Основни сценарио:

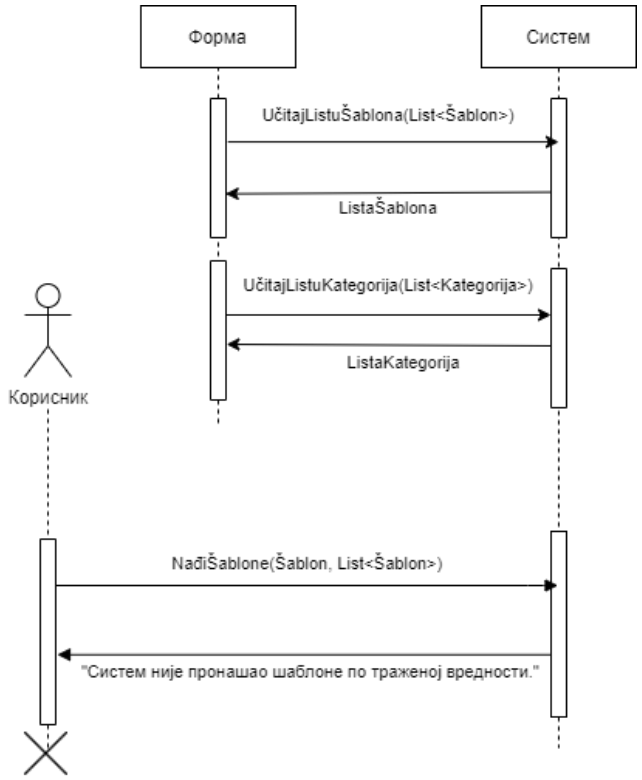
1. Форма **позива** систем да учита листу шаблона. (АПСО)
2. Систем **враћа** форми шаблона. (ИА)
3. Форма **позива** систем да учита листу категорија. (АПСО)
4. Систем **враћа** форми категорија. (ИА)
5. Корисник **позива** систем да нађе шаблоне по задатој вредности. (АПСО)
6. Систем **приказује** кориснику шаблоне и поруку: “Систем је пронашао шаблоне по задатој вредности”. (ИА)
7. Корисник **позива** систем да учита шаблон. (АПСО)
8. Систем **показује** кориснику податке о шаблону и поруку “Систем је учитао шаблон“ (ИА)
9. Корисник **позива** систем да запамти податке о шаблону. (АПСО)
10. Систем **приказује** кориснику запамћени шаблон и поруку: “Систем је запамтио шаблон.” (ИА)



Слика 30 - ДС Измена шаблона трансакције

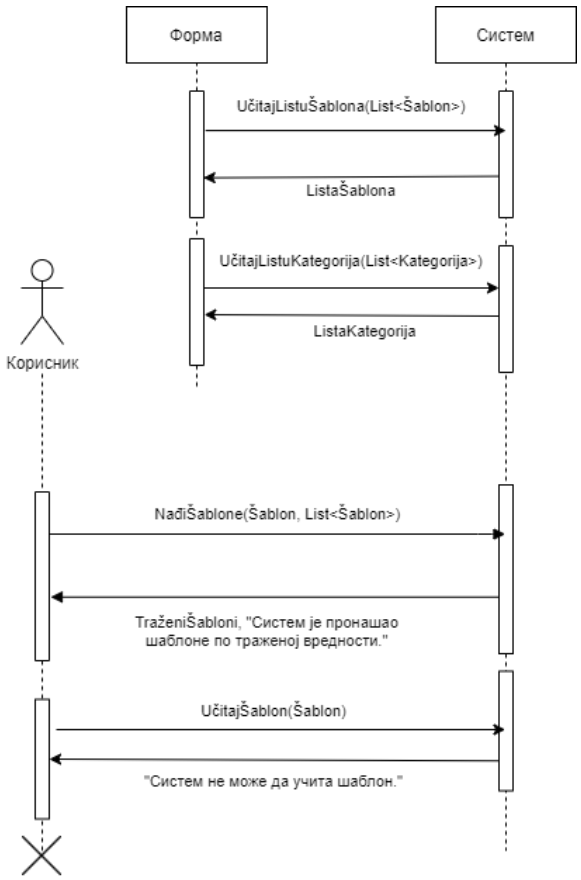
Алтернативна сценарија

6.1 Уколико систем не може да нађе шаблоне он приказује кориснику поруку: “Систем није пронашао шаблоне по задатој вредности”. Прекида се извршење сценариа. (ИА)



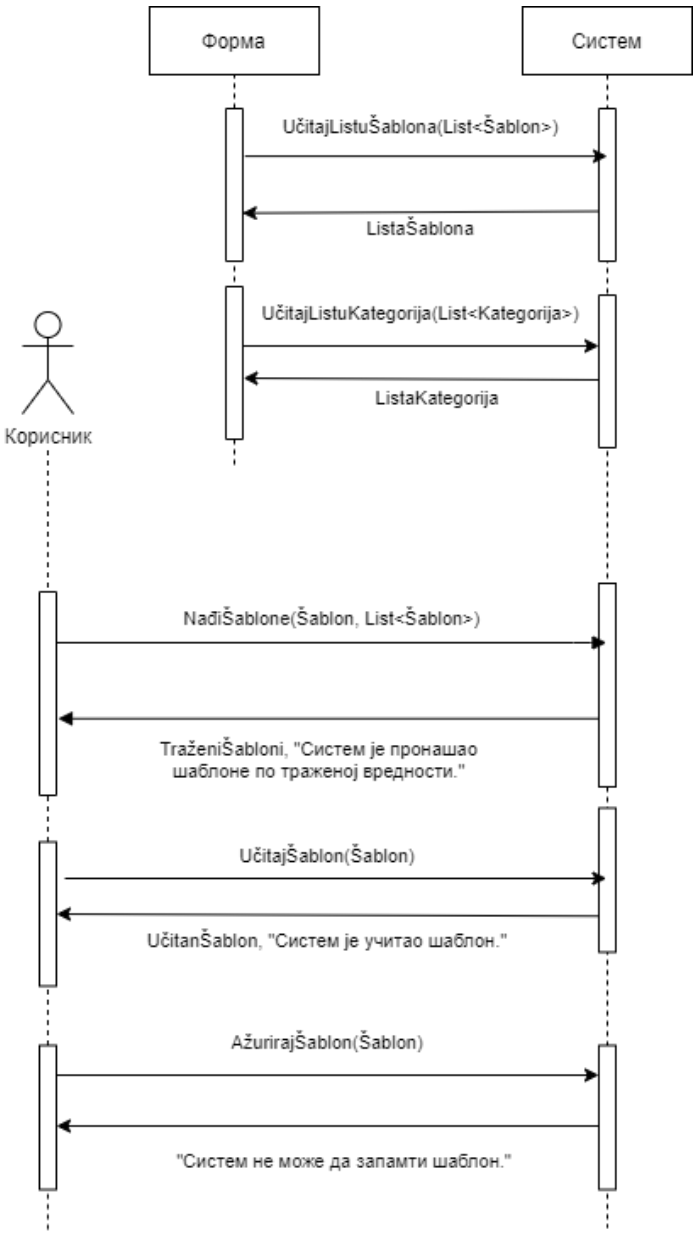
Слика 31 - ДС Неуспешна претрага приликом измене шаблона трансакције

8.1 Уколико систем не може да учита шаблон он приказује кориснику поруку “Систем не може да учита шаблон”. Прекида се извршење сценариа. (ИА)



Слика 32 - ДС Неуспешно учитавање приликом измене шаблона трансакције

10.1 Уколико систем не може да запамти податке о шаблону он приказује кориснику поруку “Систем не може да запамти шаблон”. (ИА)



Слика 33 - ДС Неуспешна измена шаблона трансакције

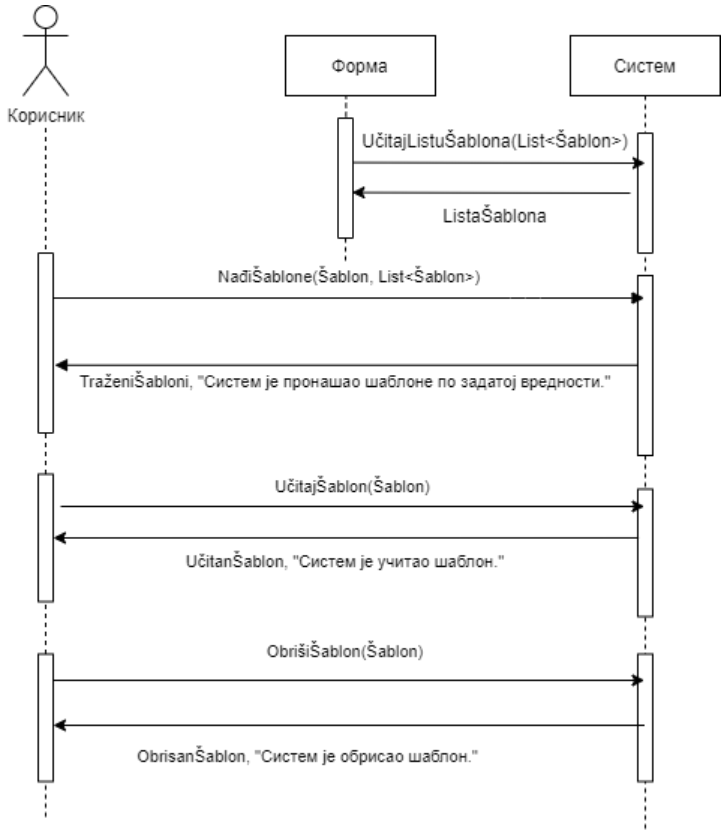
Са наведених секвенцијалних дијаграма уочено је пет системских операција:

1. Signal **УчитајЛистуШаблона(List<Шаблон>)**
2. Signal **УчитајЛистуКатегорија(List<Категорија>)**
3. Signal **НађиШаблоне(Шаблон, List<Шаблон>)**
4. Signal **УчитајШаблон(Шаблон)**
5. Signal **АжурирајШаблон(Шаблон)**

ДС10: Дијаграм секвенци случаја коришћења – Брисање шаблона трансакције

Основни сценарио

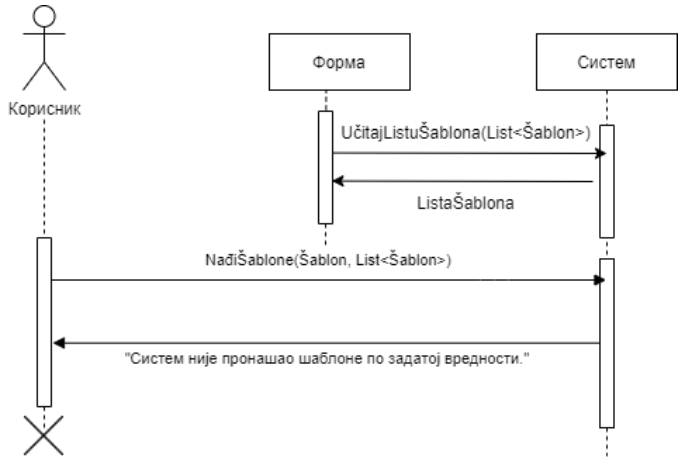
1. Форма **позива** систем да учита листу шаблона. (АПСО)
2. Систем **враћа** форми листу шаблона. (ИА)
3. Корисник **позива** систем да нађе шаблоне по задатој вредности. (АПСО)
4. Систем **приказује** кориснику шаблоне и поруку: “Систем је пронашао шаблоне по задатој вредности”. (ИА)
5. Корисник **позива** систем да учита податке о одабраном шаблону. (АПСО)
6. Систем **обавештава** корисника о успешном учитавању података поруком “Систем је учитао шаблон“ и приказује податке о шаблону. (ИА)
7. Корисник **позива** систем да обрише шаблон. (АПСО)
8. Систем **приказује** кориснику поруку: “Систем је обрисао шаблон.” (ИА)



Слика 34 - ДС Брисање шаблона

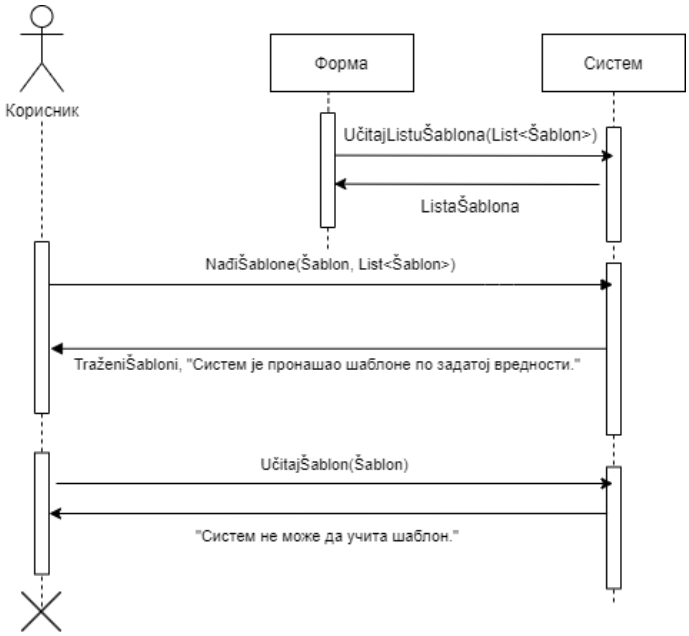
Алтернативна сценарија

4.1 Уколико систем не може да нађе шаблоне он приказује кориснику поруку: “Систем није пронашао шаблоне по задатој вредности”. Прекида се извршење сценариа. (ИА)



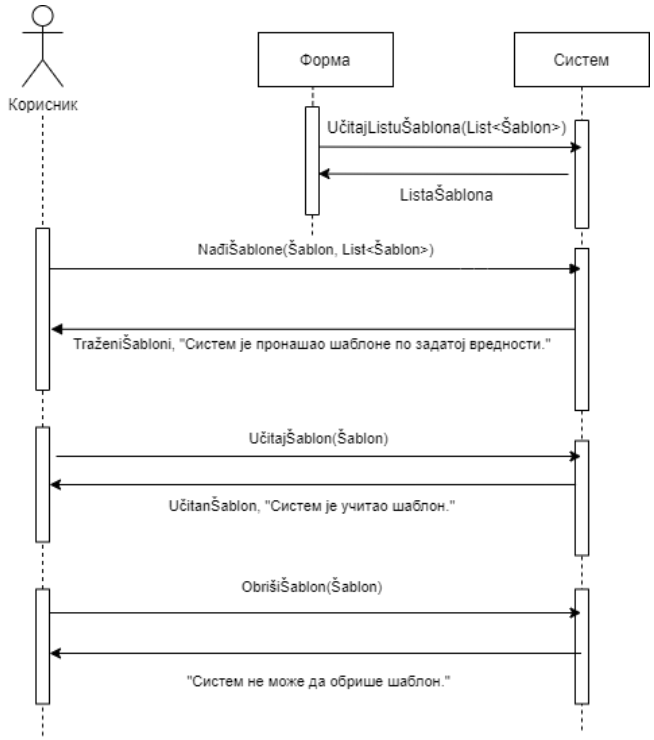
Слика 35 - ДС Неуспешна претрага приликом брисања шаблона трансакције

6.1 Уколико систем не може да учита изабран шаблон он приказује кориснику поруку “Систем не може да учита шаблон“. Прекида се извршење сценариа (ИА)



Слика 36 - ДС Неуспешно учитавање приликом брисања шаблона трансакције

8.1 Уколико систем не може да обрише шаблон он приказује кориснику поруку “Систем не може да обрише шаблон”. (ИА)



Слика 37 - ДС Неуспешно брисање шаблона трансакције

Са наведених секвенцијалних дијаграма уочене су четири системске операције:

1. Signal **УчитајЛистуШаблона(List<Шаблон>)**
2. Signal **НађиШаблоне(Шаблон, List< Шаблон >)**
3. Signal **УчитајШаблон(Шаблон)**
4. Signal **ОбришиШаблон(Шаблон)**

Списак системских операција

Као резултат анализе сценарија добијено је укупно 19 системских операција коју треба пројектовати:

1. Signal **ZapamtiRačun(Račun)**
2. Signal **UčitajListuRačuna(Lista<Račun>)**
3. Signal **NađiRačune(Račun, Lista<Račun>)**
4. Signal **UčitajRačun(Račun)**
5. Signal **AžurirajRačun(Račun)**
6. Signal **ObrišiRačun(Račun)**
7. Signal **UčitajListuKategorija(List<Kategorija>)**
8. Signal **ZapamtiTransakciju(Transakcija)**
9. Signal **UčitajListuTransakcija (List<Transakcija>)**
10. Signal **NađiTransakcije(Transakcija List<Transakcija>)**
11. Signal **UčitajTransakciju(Transakcija)**
12. Signal **AžurirajTransakciju(Transakcija)**
13. Signal **ObrišiTransakciju(Transakcija)**

14. Signal **ZapamtiŠablon(Šablon)**
15. Signal **UčitajListuŠablona(List<Šablon>)**
16. Signal **NađiŠablone(Šablon, List<Šablon>)**
17. Signal **UčitajŠablon(Šablon)**
18. Signal **AžurirajŠablon(Šablon)**
19. Signal **ObrišiŠablon(Šablon)**

5.2 Понашање софтверског система - Дефинисање уговора о системским операцијама

На основу системских дијаграма секвенци добијен је списак системских операција, где се у наставку анализе понашања софтверског система за сваку од њих формира уговор.

Креирани уговори описују понашање системских операција. Уговори описују шта операција треба да ради, без објашњења како ће то да ради. Један уговор је везан за једну системску операцију (Влајић, 2015).

Уговор УГ1: ZapamtiRačun(Račun):signal;

Веза са СК: СК1

Предуслови: Вредносна и структурна ограничења над објектом *Рачун* морају бити задовољена.

Постуслови: Подаци о рачуну су запамћени.

Уговор УГ2: UčitajListuRačuna(Lista<Račun>):signal;

Веза са СК: СК2, СК3

Предуслови:

Постуслови:

Уговор УГ3: NađiRačune(Račun, Lista<Račun>):signal;

Веза са СК: СК2, СК3

Предуслови:

Постуслови:

Уговор УГ4: UčitajRačun(Račun):signal;

Веза са СК: СК2, СК3

Предуслови:

Постуслови:

Уговор УГ5: АжурирајРачун(Рачун):signal;

Веза са СК: СК2

Предуслови: Вредносна и структурна ограничења над објектом *Рачун* морају бити задовољена.

Постуслови: Подаци о рачуну су запамћени.

Уговор УГ6: ОбришиРачун(Рачун):signal;

Веза са СК: СК3

Предуслови: Структурна ограничења над објектом *Рачун* морају бити задовољена.

Постуслови: Подаци о рачуну су обрисани.

Уговор УГ7: УчитајЛистуКатегорија(List<Категорија>):signal;

Веза са СК: СК4, СК5, СК6, СК7, СК8, СК9

Предуслови:

Постуслови:

Уговор УГ8: ЗапамтиТрансакцију(Трансакција):signal;

Веза са СК: СК4

Предуслови: Вредносна и структурна ограничења над објектом *Трансакција* морају бити задовољена.

Постуслови: Подаци о трансакцији су запамћени.

Уговор УГ9: УчитајЛистуТрансакција (List<Трансакција>):signal;

Веза са СК: СК5, СК6, СК7

Предуслови:

Постуслови:

Уговор УГ10: НађиТрансакције(Трансакција List<Трансакција>):signal;

Веза са СК: СК5, СК6, СК7

Предуслови:

Постуслови:

Уговор УГ11: УчитајТрансакцију(Трансакција):signal;

Веза са СК: СК5, СК6, СК7

Предуслови:

Постуслови:

Уговор УГ12: АжурирајTransakciju(Transakcija):signal;

Веза са СК: СК6

Предуслови: Вредносна и структурна ограничења над објектом *Трансакција* морају бити задовољена.

Постуслови: Подаци о трансакцији су запамћени.

Уговор УГ13: ОбришиTransakciju(Transakcija):signal;

Веза са СК: СК7

Предуслови: Структурна ограничења над објектом *Трансакција* морају бити задовољена.

Постуслови: Подаци о трансакцији су обрисани.

Уговор УГ14: ZapamtiŠablon(Šablon):signal;

Веза са СК: СК8

Предуслови: Вредносна и структурна ограничења над објектом *Шаблон* морају бити задовољена.

Постуслови: Подаци о шаблону су запамћени.

Уговор УГ15: УчитајListuŠablona(List<Šablon>):signal;

Веза са СК: СК9, СК10

Предуслови:

Постуслови:

Уговор УГ16: НађиШаблоне(Šablon, Lista<Šablon>):signal;

Веза са СК: СК9, СК10

Предуслови:

Постуслови:

Уговор УГ17: УчитајШablon(Šablon):signal;

Веза са СК: СК9, СК10

Предуслови:

Постуслови:

Уговор УГ18: АжурирајШablon(Šablon):signal;

Веза са СК: СК9

Предуслови: Вредносна и структурна ограничења над објектом *Шаблон* морају бити задовољена.

Постуслови: Подаци о шаблону су запамћени.

Уговор УГ19: ObrišiŠablon(Šablon):signal;

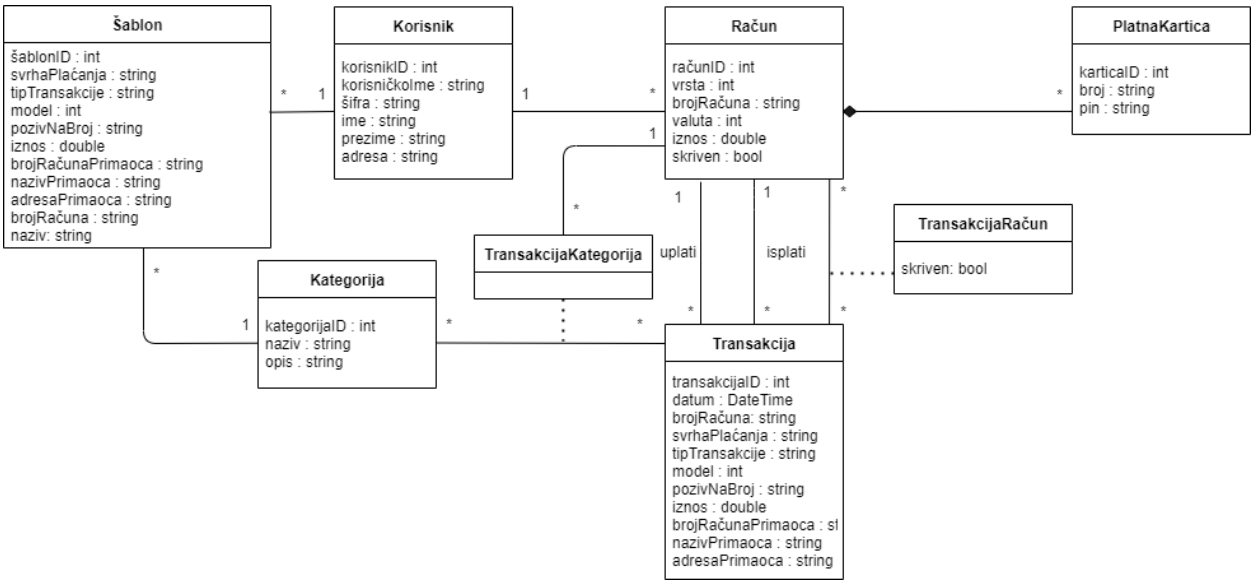
Веза са СК: СК10

Предуслови: Структурна ограничења над објектом *Шаблон* морају бити задовољена.

Постуслови: Подаци о шаблону су обрисани.

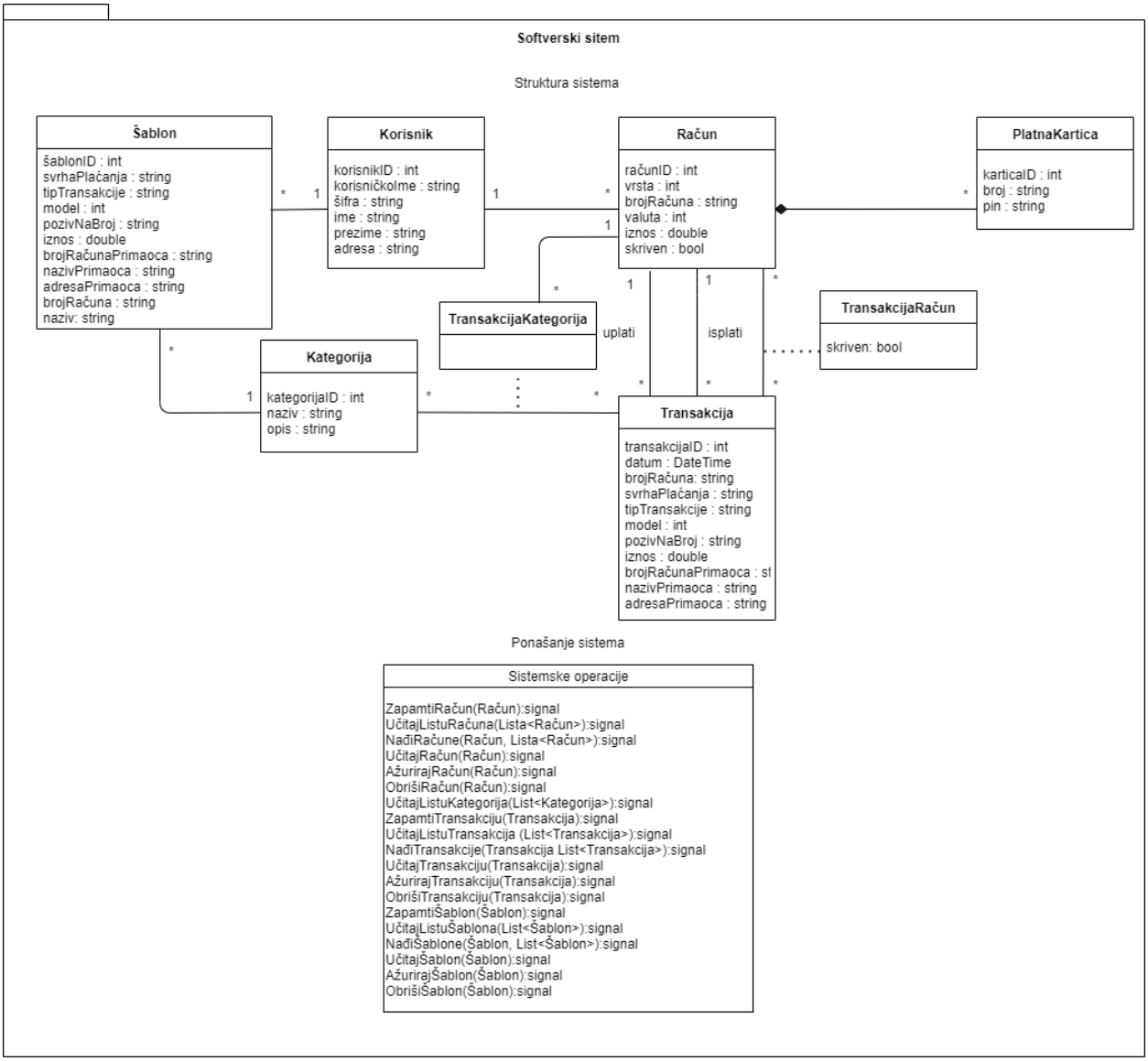
5.3 Структура софтверског система - Концептуални (доменски) модел

Структура софтверског система се описује помоћу концептуалног модела. Концептуални модел садржи концептуалне класе (доменске објекте) и асоцијације између концептуалних класа. Често се за концептуалне моделе каже да су то доменски модели или модели објектне анализе (Влајић, 2015).



Слика 38 - Концептуални (доменски) модел

Добијена логичка структура и понашање софтверског система (који чине пословну логику) резултат је фазе анализе:

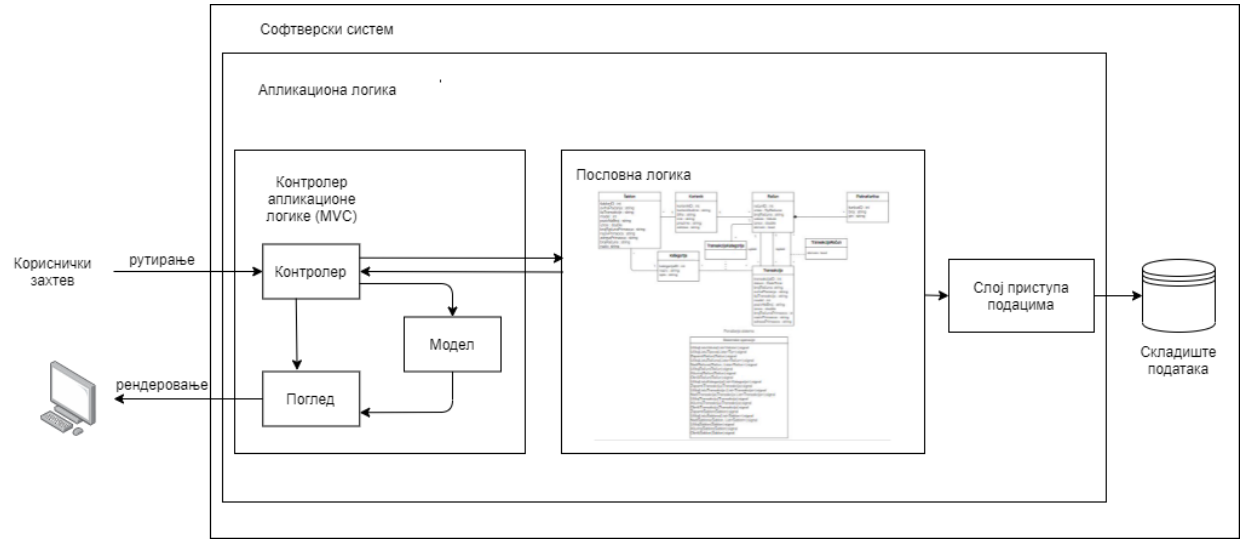


Слика 39 - Структура и понашање софтверског система

6. Фаза пројектовања

Фаза пројектовања описује физичку структуру и понашање софтверског система (архитектуру софтверског система). Архитектура софтверског система је у већини случајева тронивојска и састоји се из следећих слојева (Влајић, 2015):

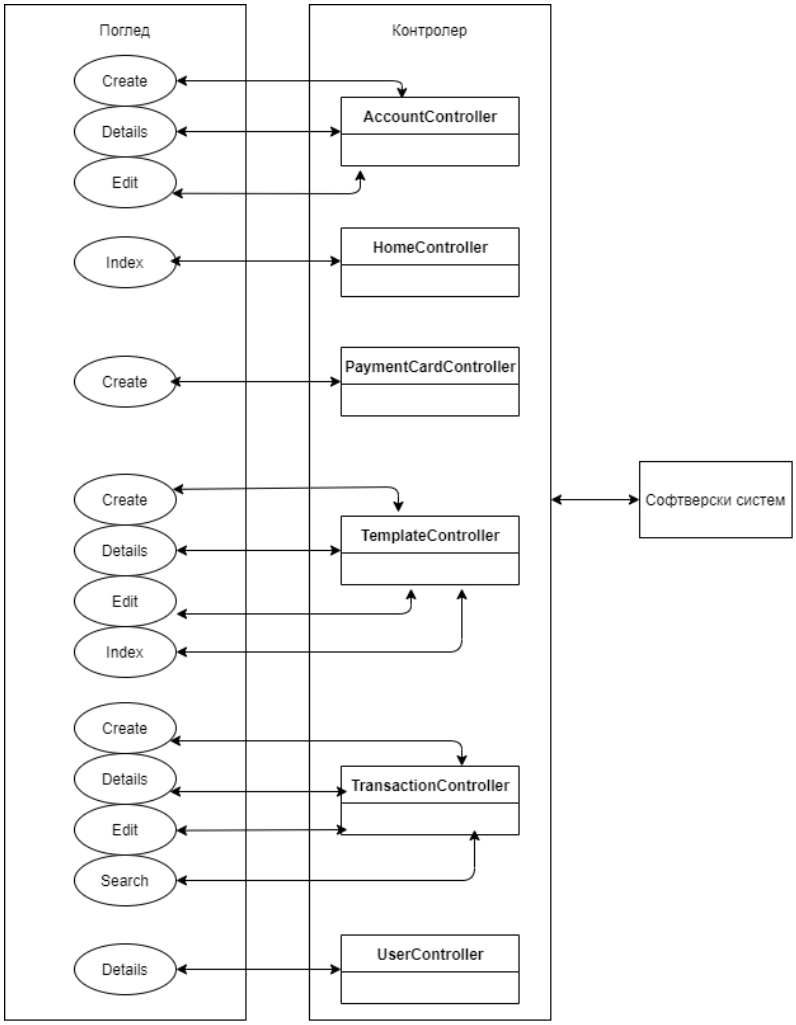
1. Кориснички интерфејс - који представља улазно–излазну репрезентацију софтверског система.
2. Апликациона логика - описује структуру и понашање софтверског система. Састоји се из контролера апликационе логике, пословне логике и слоја приступа подацима
3. Складиште података - који чува стање атрибута софтверског система



Слика 40 - Архитектура софтверског система

6.1 Пројектовање корисничког интерфејса

Кориснички интерфејс састоји се од скупа веб страница које прихватају податке које корисник уноси, прослеђују их до одговарајућег контролера који захтев прослеђује слоју приступа подацима. Након што контролер добије одговор, прослеђује захтеване податке (модел) до одговарајућег погледа.



Слика 41 - Повезаност контролера и корисничког интерфејса у архитектури софтверског система

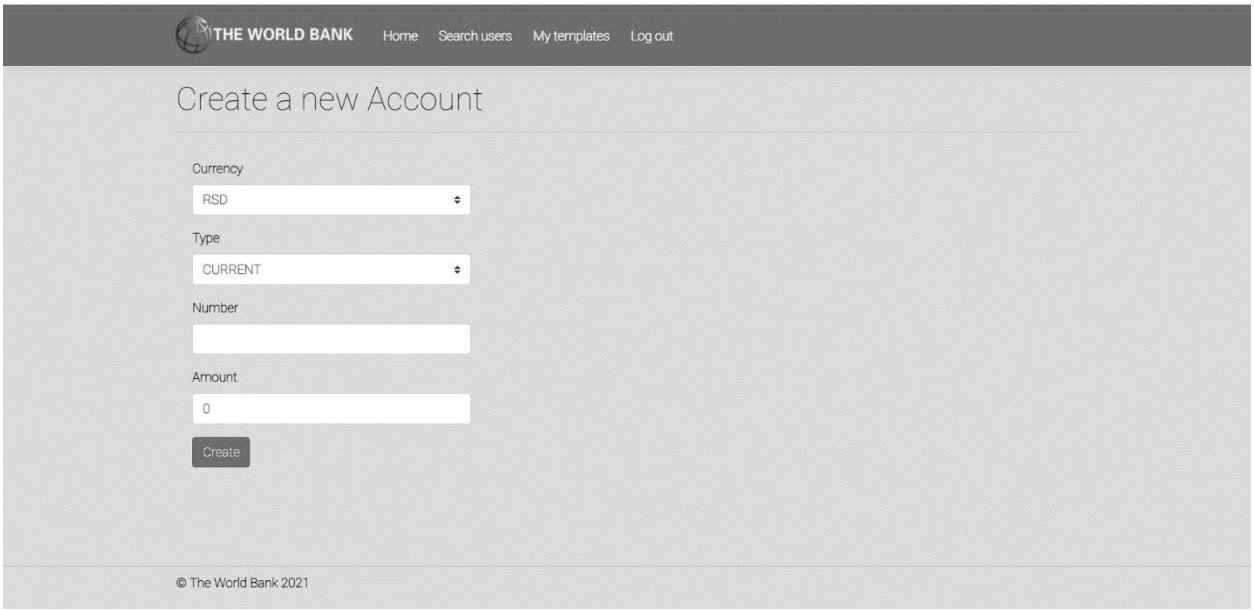
СК1: Случај коришћења – Креирање рачуна

Назив СК: Креирање рачуна

Актори СК: Корисник

Учесници СК: Корисник и систем (програм)

Предуслов: Систем је укључен и корисник је улогован под својом шифром. Систем приказује форму за рад са рачуном.



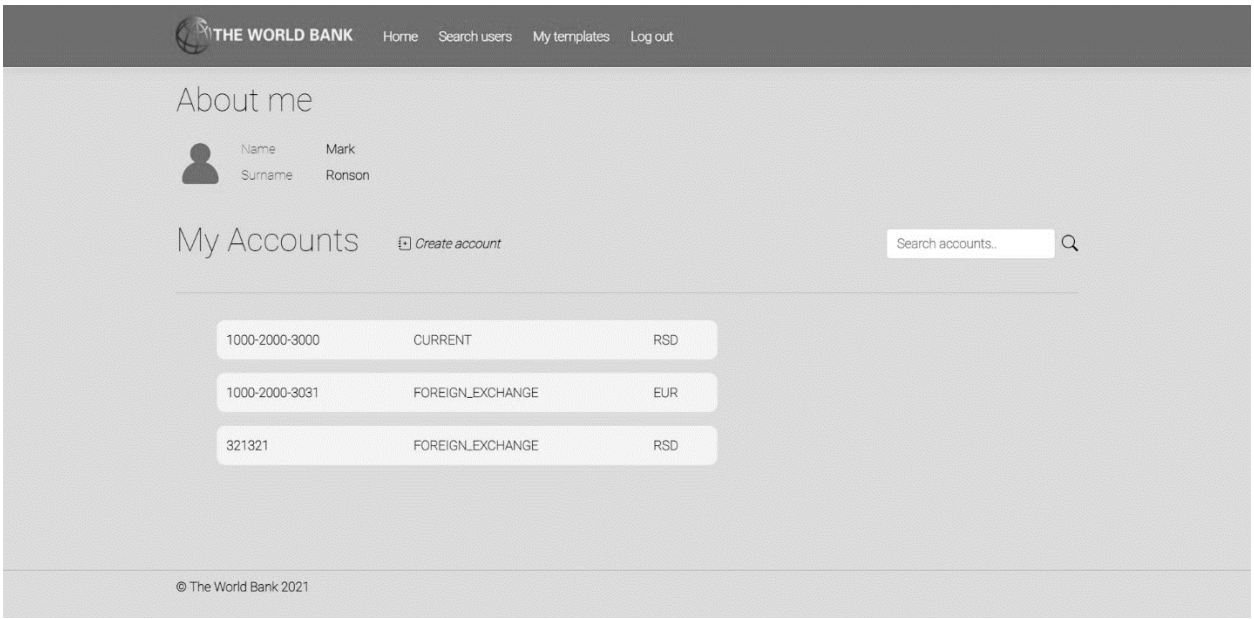
Слика 42 - Страница за креирање рачуна

Основни сценарио СК

1. Корисник **уноси** податке о рачуну. (АПУСО)
2. Корисник **контролише** да ли је коректно унео податке о рачуну. (АНСО)
3. Корисник **позива** систем да запамти рачун. (АПСО)

Опис акције: Корисник притиском на дугме Create позива системску операцију ZapamtiRačun.

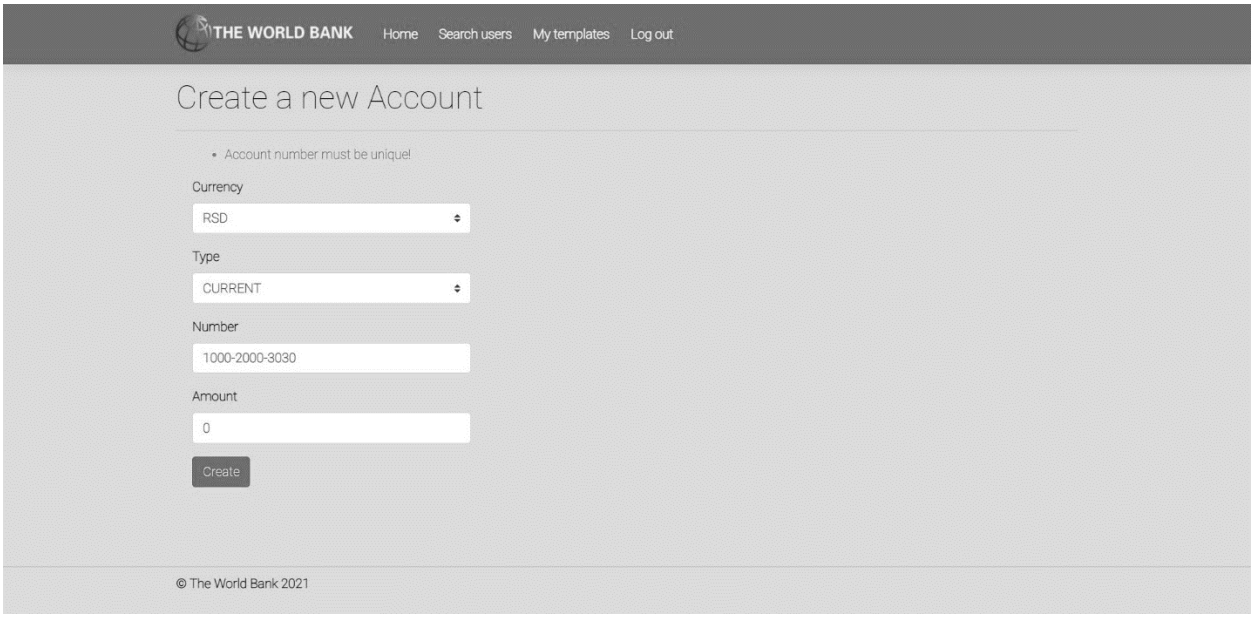
4. Систем **памти** рачун. (СО)
5. Систем **приказује** кориснику запамћени рачун и поруку: “Систем је запамтио рачун“. (ИА)



Слика 43 - Страница са успешно креираним рачунима

Алтернативна сценарија

5.1 Уколико систем не може да креира рачун он приказује кориснику поруку “Систем не може да запамти рачун”. (ИА)



Слика 44 - Страница са неуспешно креираним рачуном

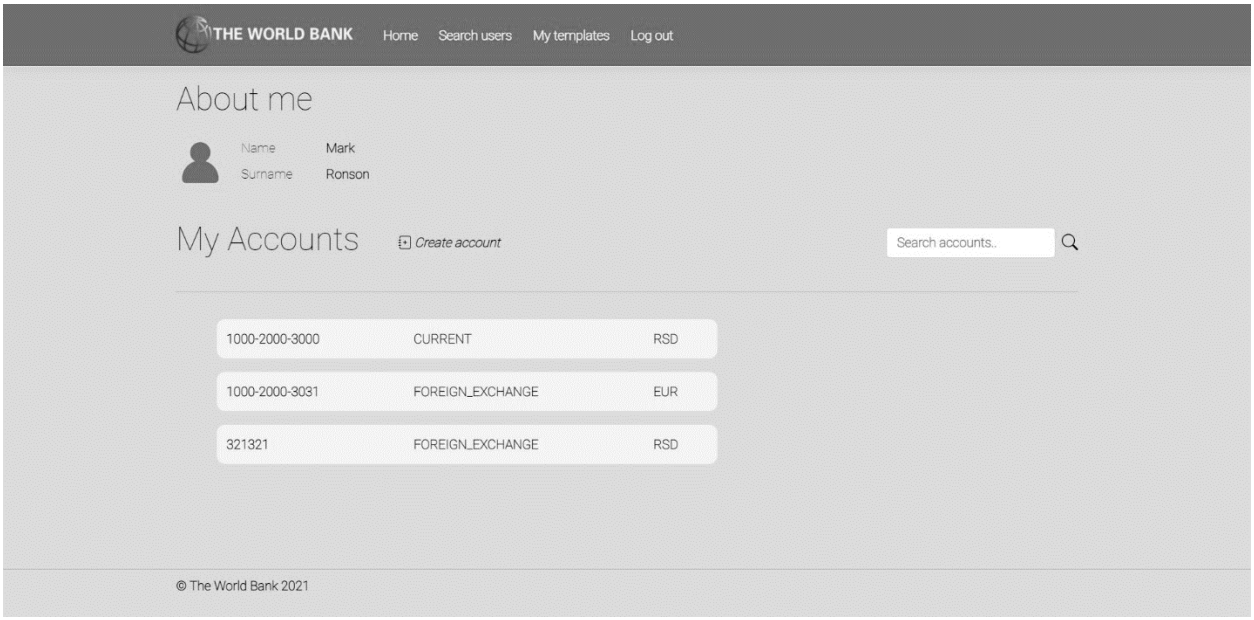
СК2: Случај коришћења – Измена рачуна

Назив СК: Измена рачуна

Актори СК: Корисник

Учесници СК: Корисник и систем (програм)

Предуслов: Систем је укључен и корисник је улогован под својом шифром. Систем приказује форму за рад са рачуном. Учитана су листе корисникових рачуна.



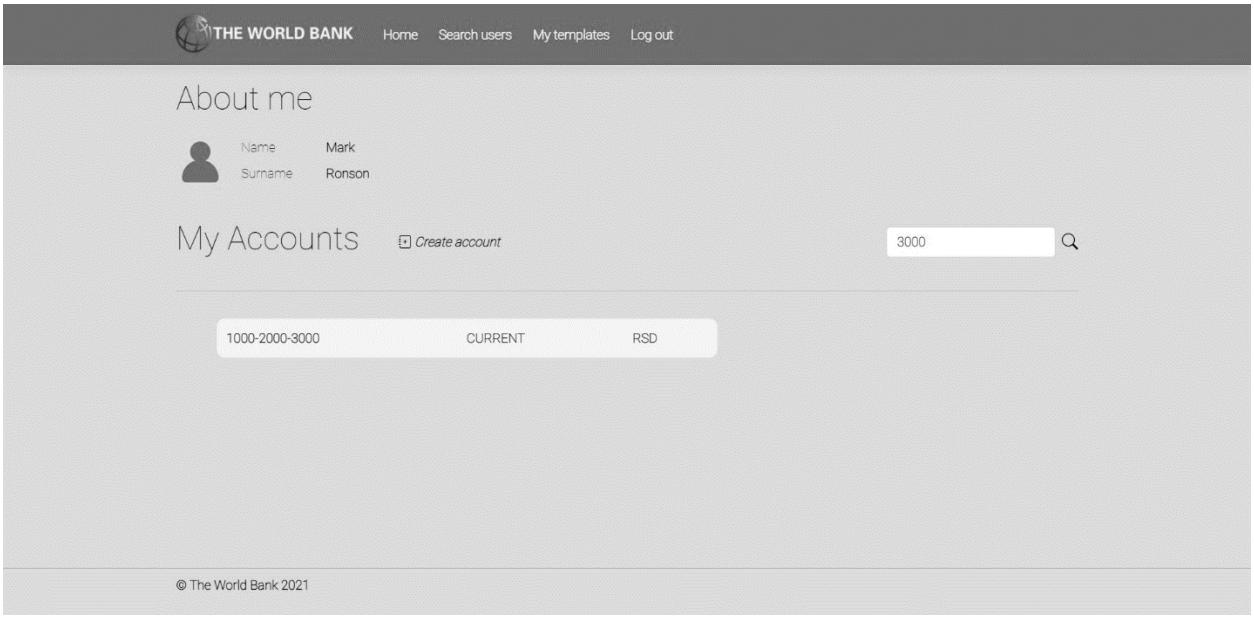
Слика 45 - Страница која приказује корисникове рачуне

Основни сценарио СК

1. Корисник **уноси** вредност по којој претражује рачуне. (АПУСО)
2. Корисник **позива** систем да нађе рачуне по задатој вредности. (АПСО)

Опис акције: Корисник притиском на иконицу лупе позива системску операцију НађиРачуне.

3. Систем **тражи** рачуне по задатој вредности. (СО)
4. Систем **приказује** кориснике рачуне и поруку: “Систем је нашао рачуне по задатој вредности”. (ИА)

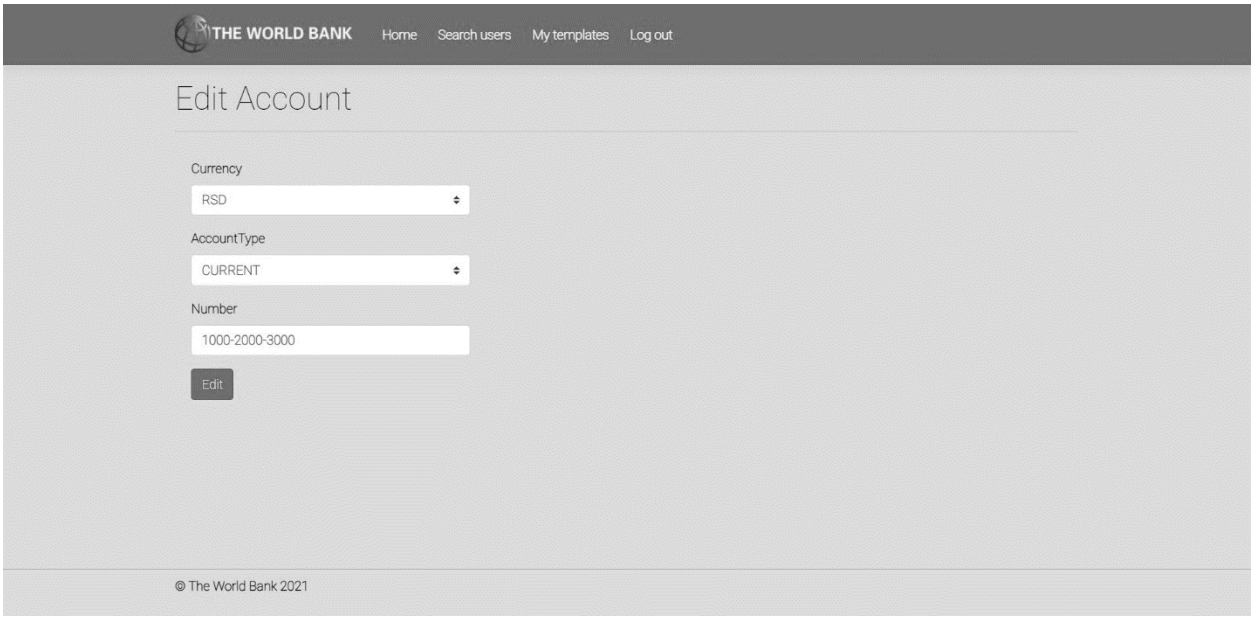


Слика 46 - Успешна претрага рачуна

5. Корисник **бира** рачун. (АПУСО)
6. Корисник **позива** систем да учита рачун. (АПСО)

Опис акције: Корисник притиском на број жељеног рачуна позива системску операцију УчитајРачун.

7. Систем **учитава** рачун. (СО)
8. Систем **показује** кориснику податке о рачуну и поруку "Систем је учитао рачун" (ИА)

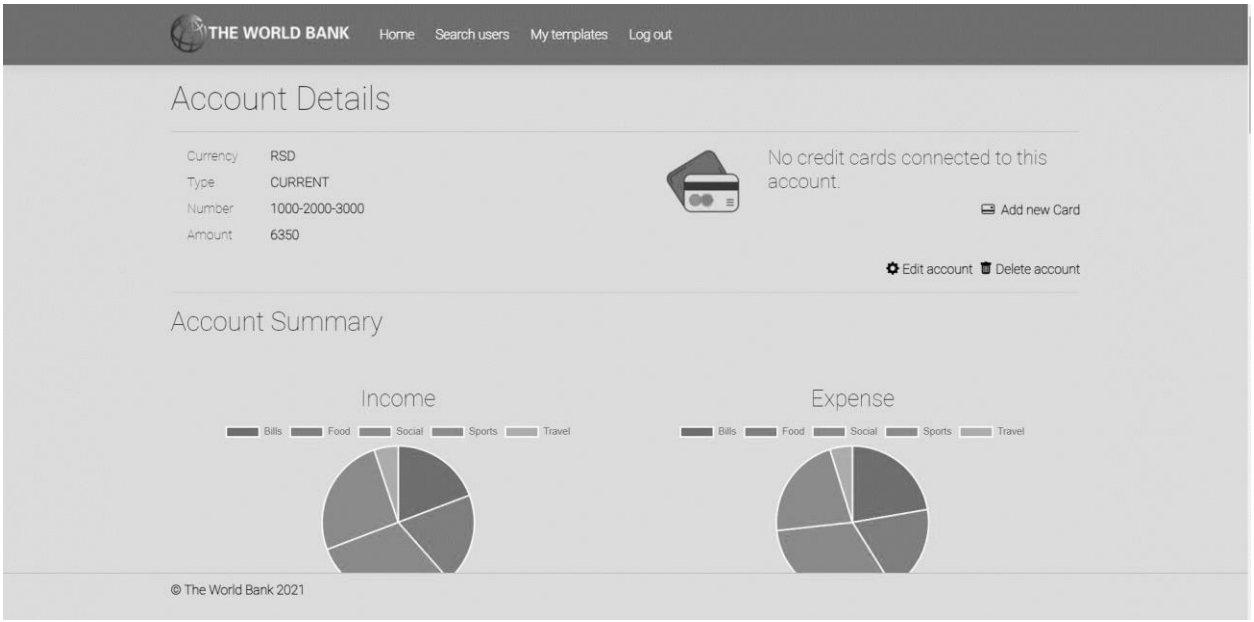


Слика 47 - Страница за уређивање рачуна

9. Корисник **уноси** (мења) податке о рачуну. (АПУСО)
10. Корисник **контролише** да ли је коректно унео податке о рачуну. (АНСО)
11. Корисник **позива** систем да запамти податке о рачуну. (АПСО)

Опис акције: Корисник притиском на дугме Edit позива системску операцију АжурирајРачун.

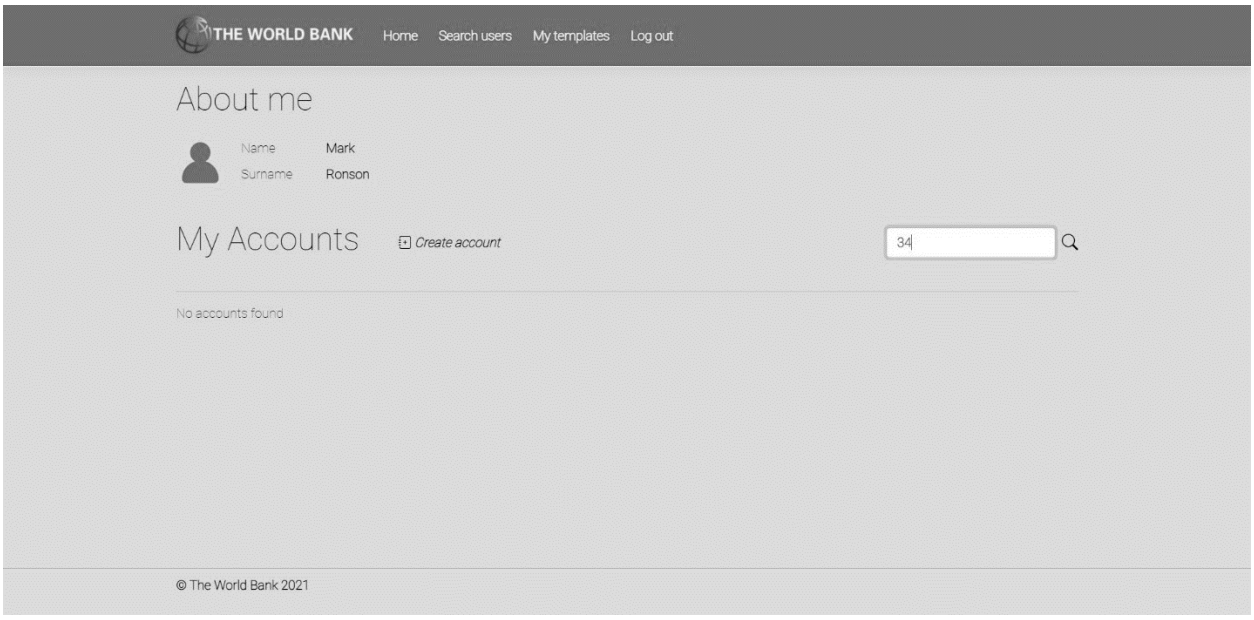
12. Систем **памти** податке о рачуну. (СО)
13. Систем **приказује** кориснику запамћени рачун и поруку: "Систем је запамтио рачун." (ИА)



Слика 48 - Страница о детаљима рачуна

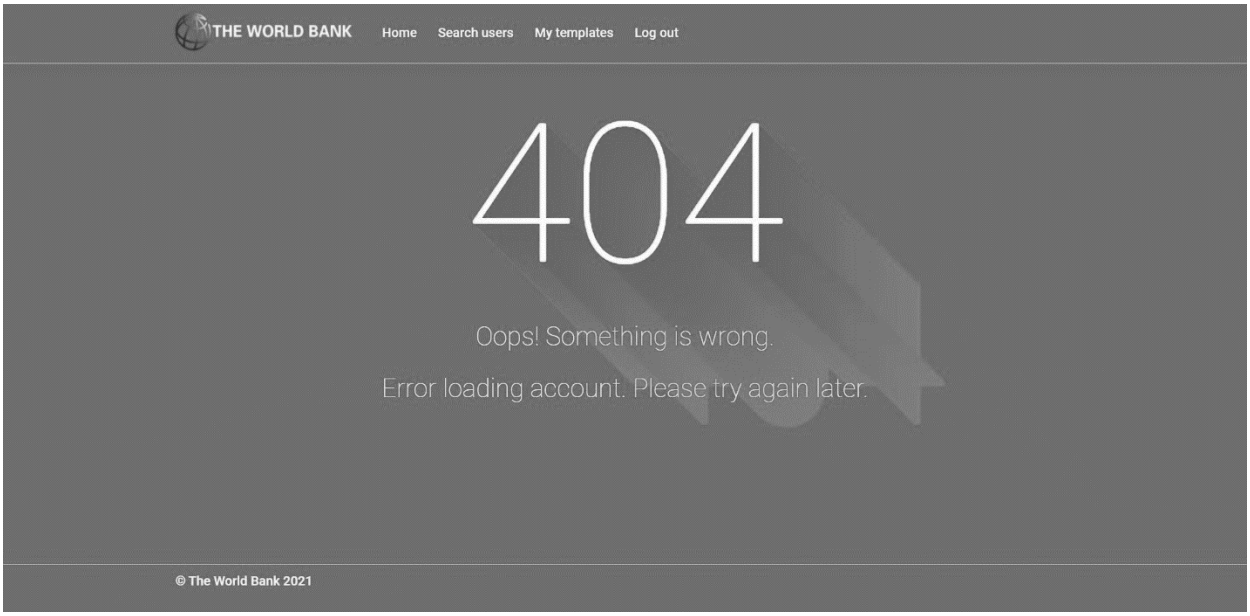
Алтернативна сценарија

4.1 Уколико систем не може да нађе рачуне он приказује кориснику поруку: “Систем не може да нађе рачуне по задатој вредности”. Прекида се извршење сценариа. (ИА)



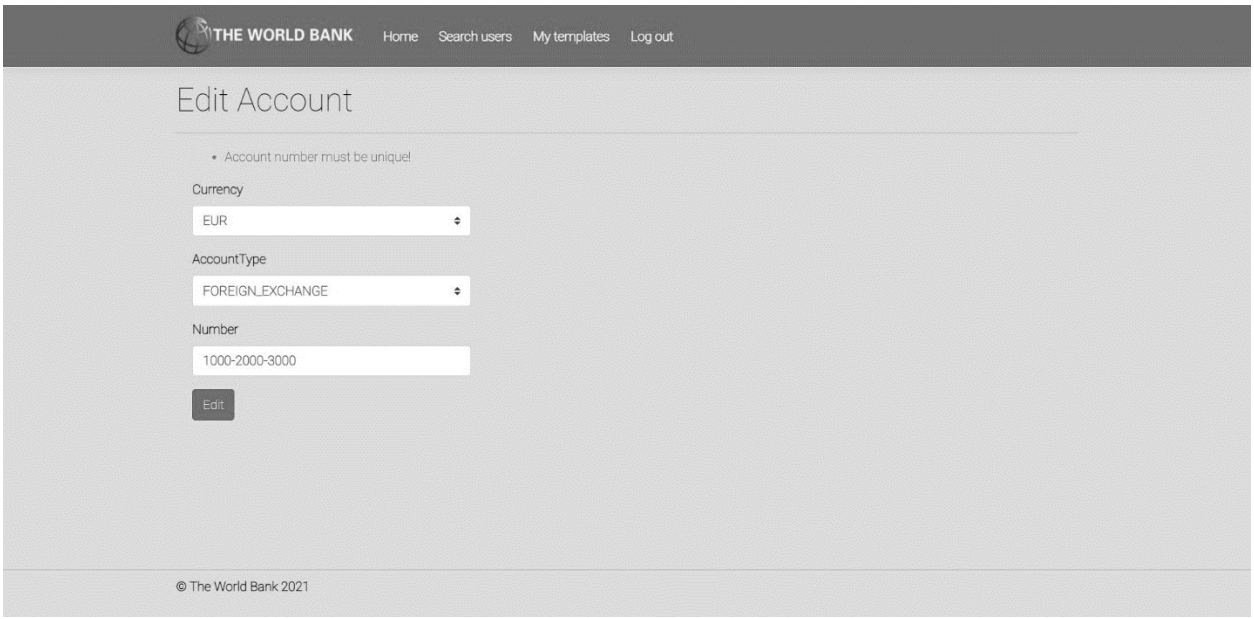
Слика 49 - Неуспешна претрага рачуна

8.1 Уколико систем не може да учита рачун он приказује кориснику поруку “Систем не може да учита рачун”. Прекида се извршење сценариа. (ИА)



Слика 50 - Страница која приказује неуспешно учитавање рачуна

13.1 Уколико систем не може да запамти податке о рачуну он приказује кориснику поруку “Систем не може да запамти рачун”. (ИА)



Слика 51 - Приказ неуспешне измене рачуна

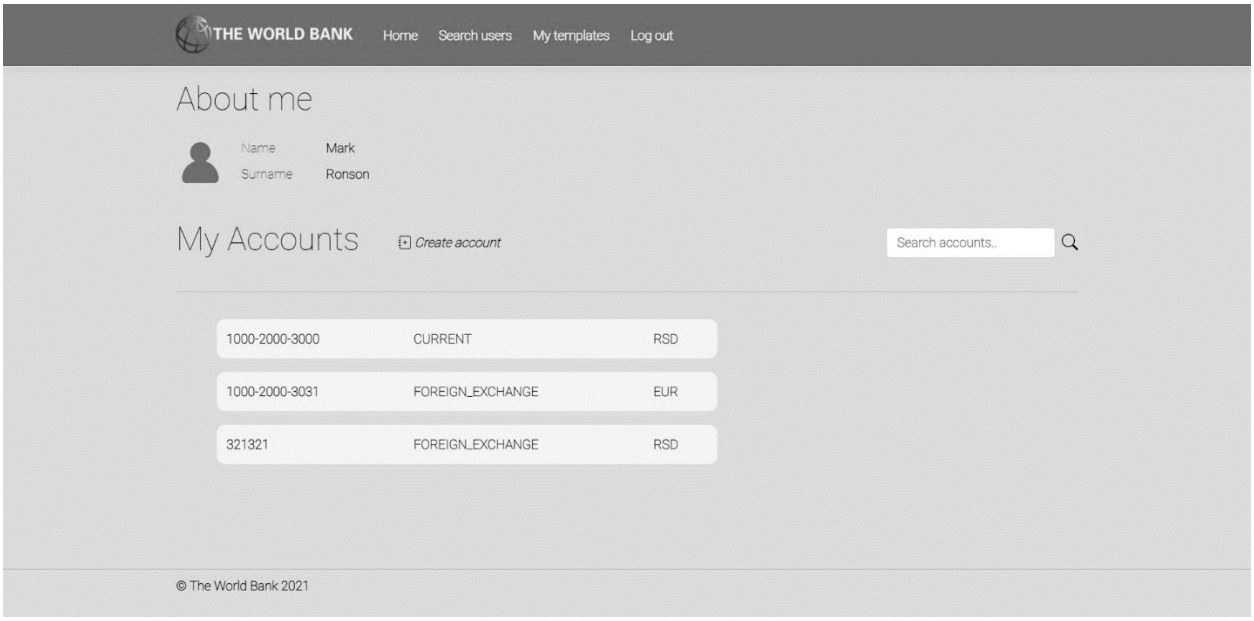
СК3: Случај коришћења – Брисање рачуна

Назив СК: Брисање рачуна

Актори СК: Корисник

Учесници СК: Корисник и систем (програм)

Предуслов: Систем је укључен и корисник је улогован под својом шифром. Систем приказује форму за рад са рачуном. Учитана је листа корисникових рачуна.



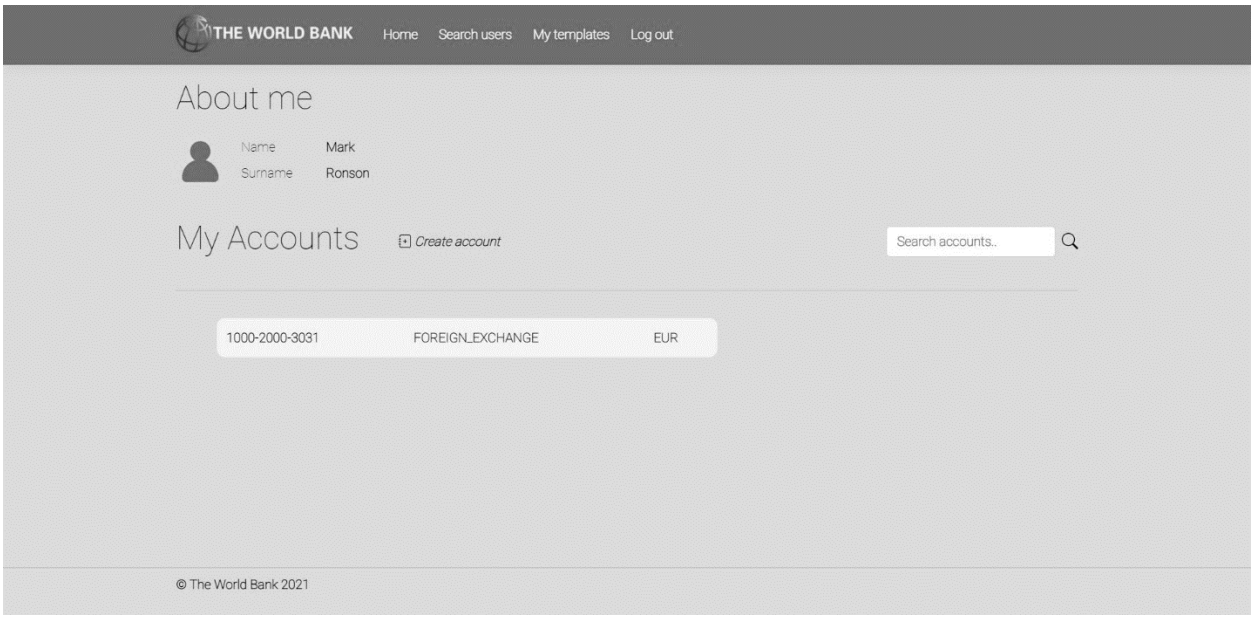
Слика 52 - Почетна страница која приказује корисникове рачуне

Основни сценарио СК

1. Корисник **уноси** вредност по којој претражује рачуне. (АПУСО)
2. Корисник **позива** систем да нађе рачуне по задатој вредности. (АПСО)

Опис акције: Корисник притиском на иконицу лупе позива системску операцију НађиРачуне.

3. Систем **тражи** рачуне по задатој вредности. (СО)
4. Систем **приказује** кориснику рачуне и поруку: “Систем је нашао рачуне по задатој вредности”. (ИА)

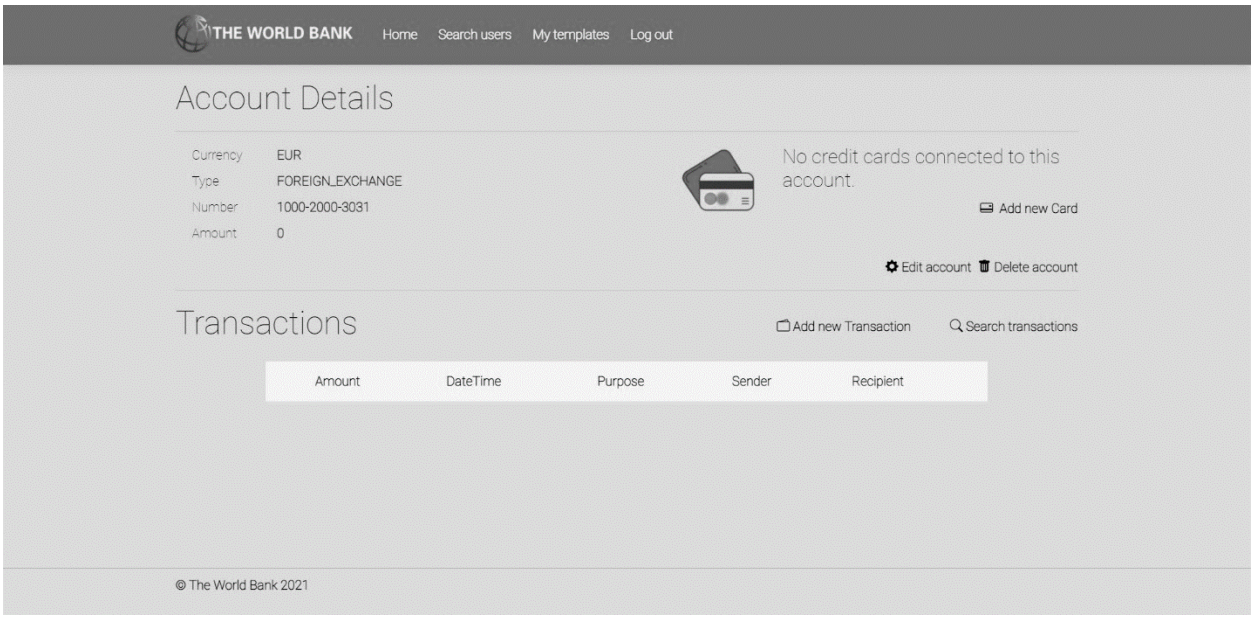


Слика 53 - Приказ резултата претраге рачуна

5. Корисник **бира** рачун који жели да обрише. (АПУСО)
6. Корисник **позива** систем да учита податке о одабраном рачуну. (АПСО)

Опис акције: Корисник притиском на број жељеног рачуна позива системску операцију УчитајРачун.

7. Систем **учитава** податke о одабраном рачуну. (СО)
8. Систем **обавештава** корисника о успешном учитавању података поруком "Систем је учитао одабран рачун" и приказује податке о рачуну. (ИА)

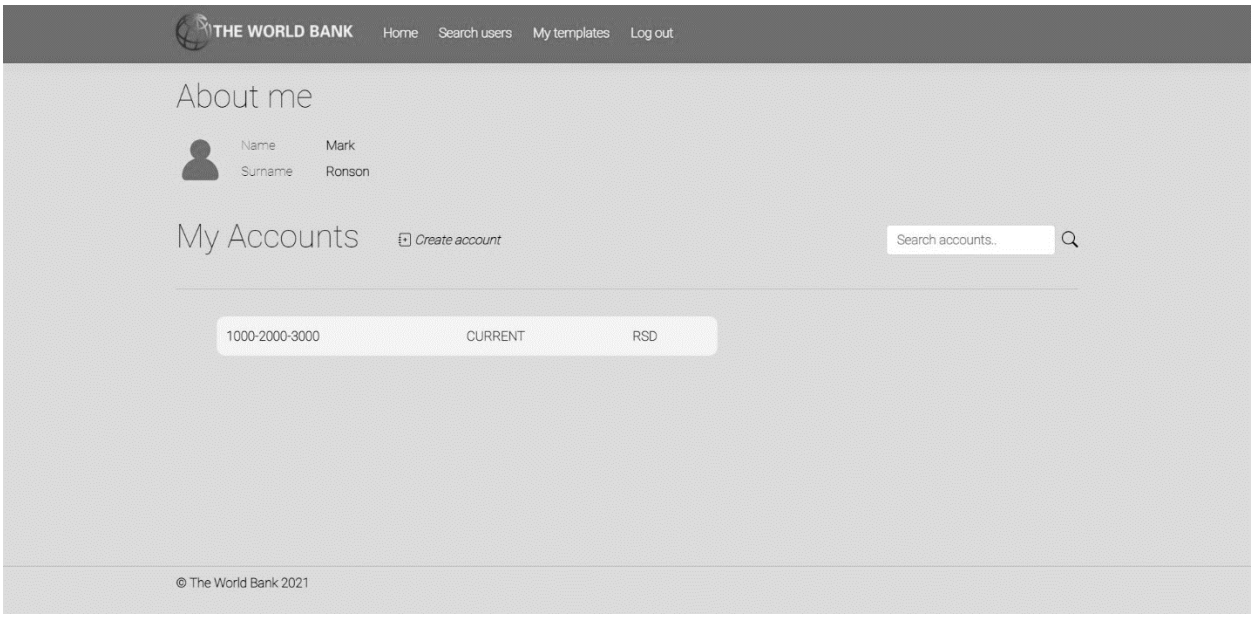


Слика 54 - Приказ детаља о рачуну са опцијом брисања

9. Корисник **позива** систем да обрише рачун. (АПСО)

Опис акције: Корисник притиском на дугме Delete account позива системску операцију ObrišiRačun.

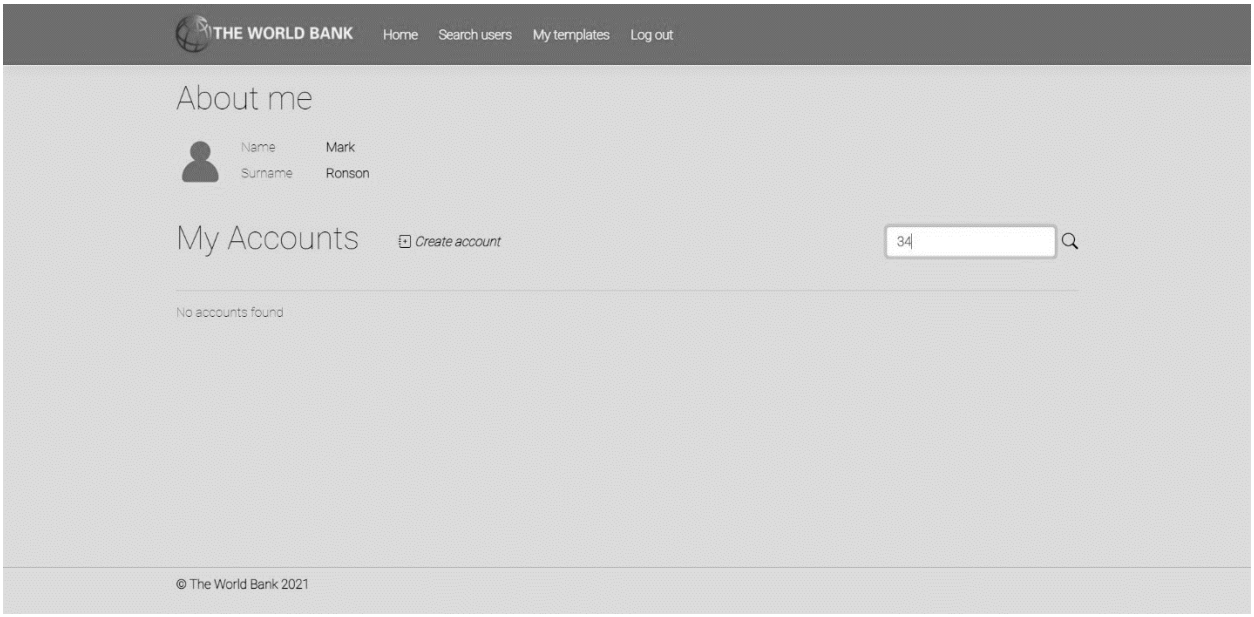
10. Систем **брише** рачун. (СО)
11. Систем **приказује** кориснику поруку: “Систем је обрисао рачун.” (ИА)



Слика 55 - Приказ корисникових рачуна након брисања рачуна

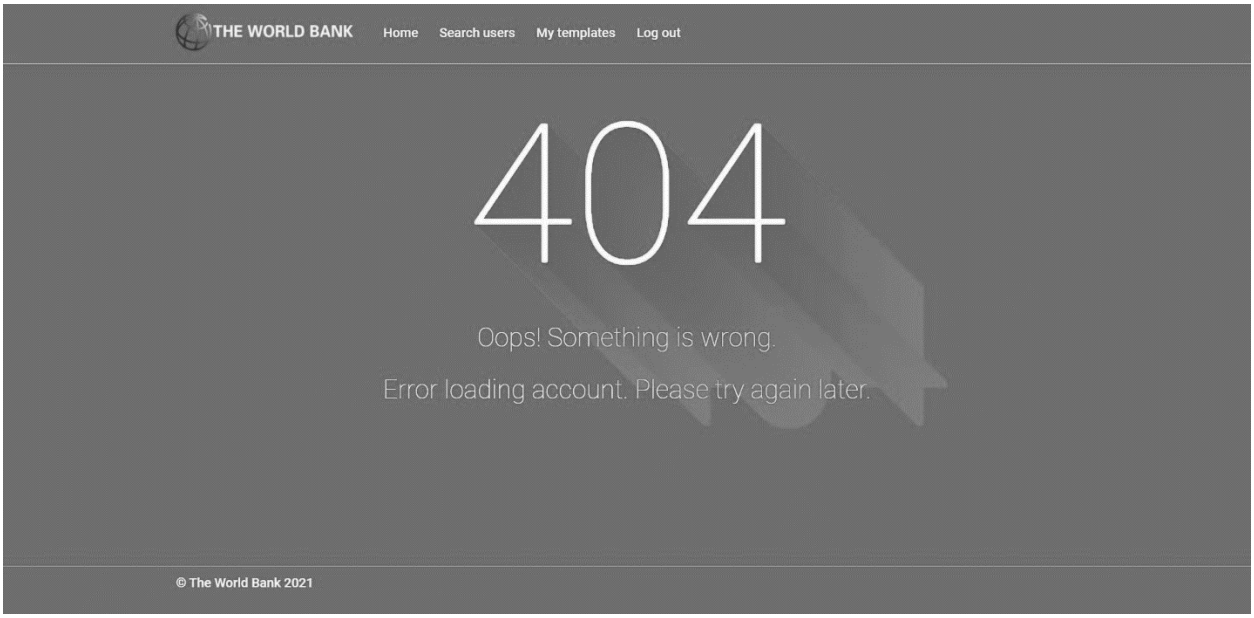
Алтернативна сценарија

4.1 Уколико систем не може да нађе рачун он приказује кориснику поруку: “Систем не може да нађе рачун по задатој вредности”. Прекида се извршење сценариа. (ИА)



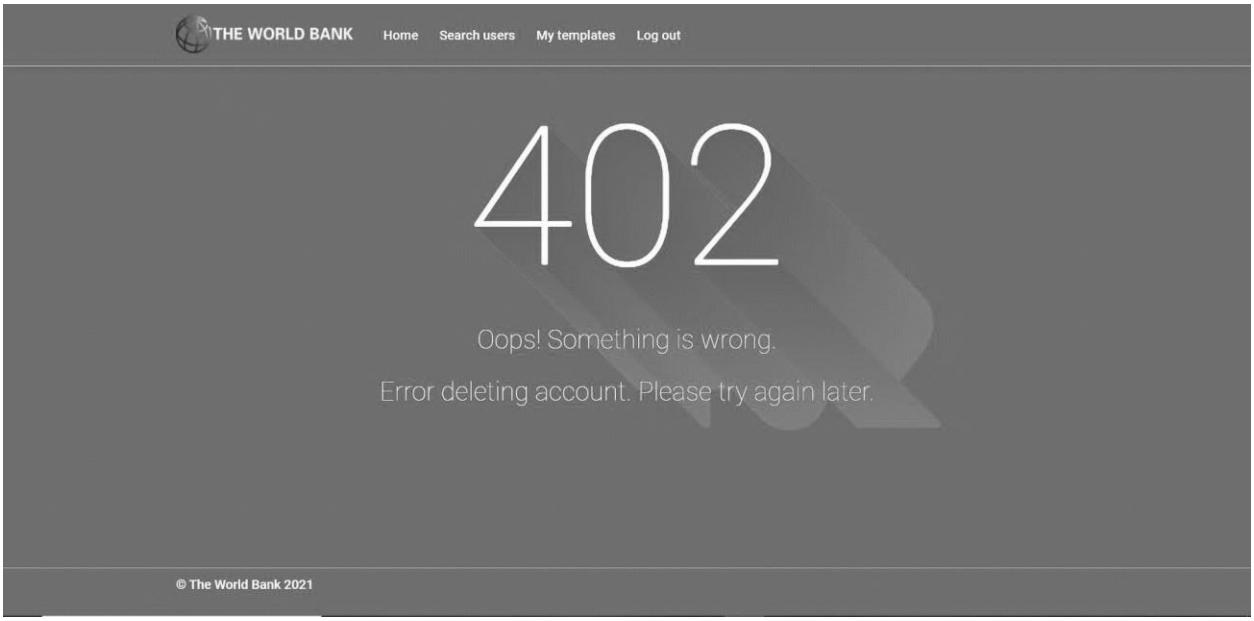
Слика 56 - Приказ неуспешне претраге рачуна приликом брисања рачуна

8.1 Уколико систем не може да учита изабран рачун он приказује кориснику поруку “Систем не може да учита рачун“. Прекида се извршење сценариа (ИА)



Слика 57 - Страница која приказује грешку приликом учитавања рачуна

11.1 Уколико систем не може да обрише рачун он приказује кориснику поруку “Систем не може да обрише рачун”. (ИА)



Слика 58 - Страница која приказује неуспешно брисање рачуна

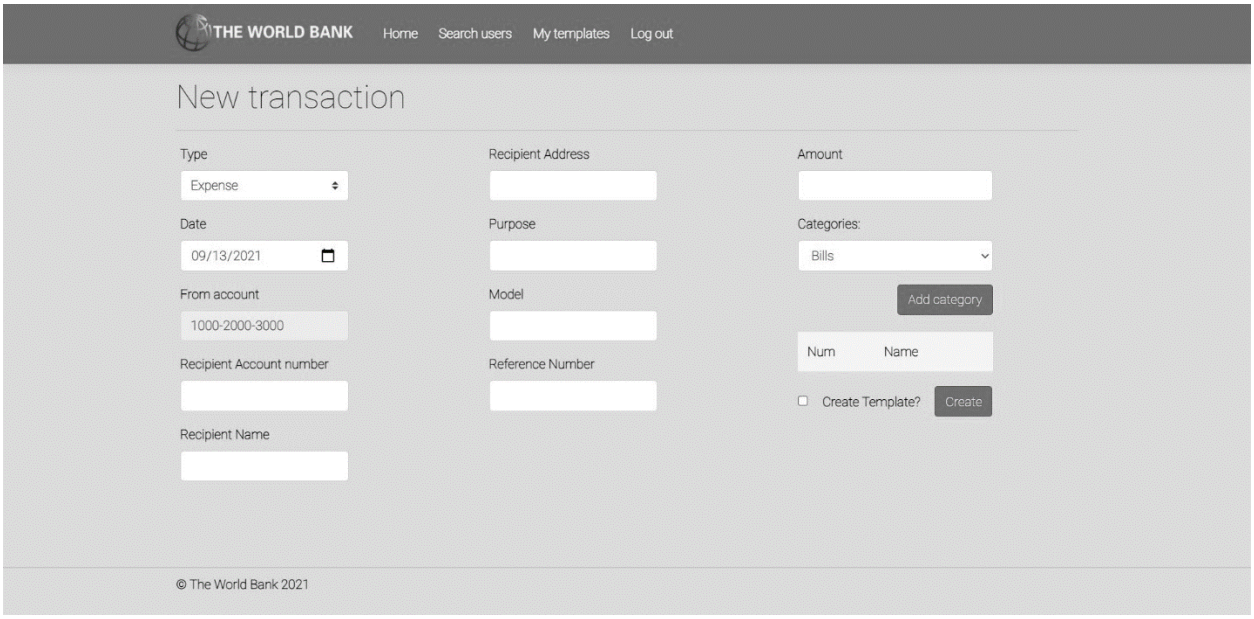
СК4: Случај коришћења – Креирање трансакције

Назив СК: Креирање трансакције

Актори СК: Корисник

Учесници СК: Корисник и систем (програм)

Предуслов: Систем је укључен и корисник је улогован под својом шифром. Систем приказује форму за рад са трансакцијом. Учитана је листа категорија.



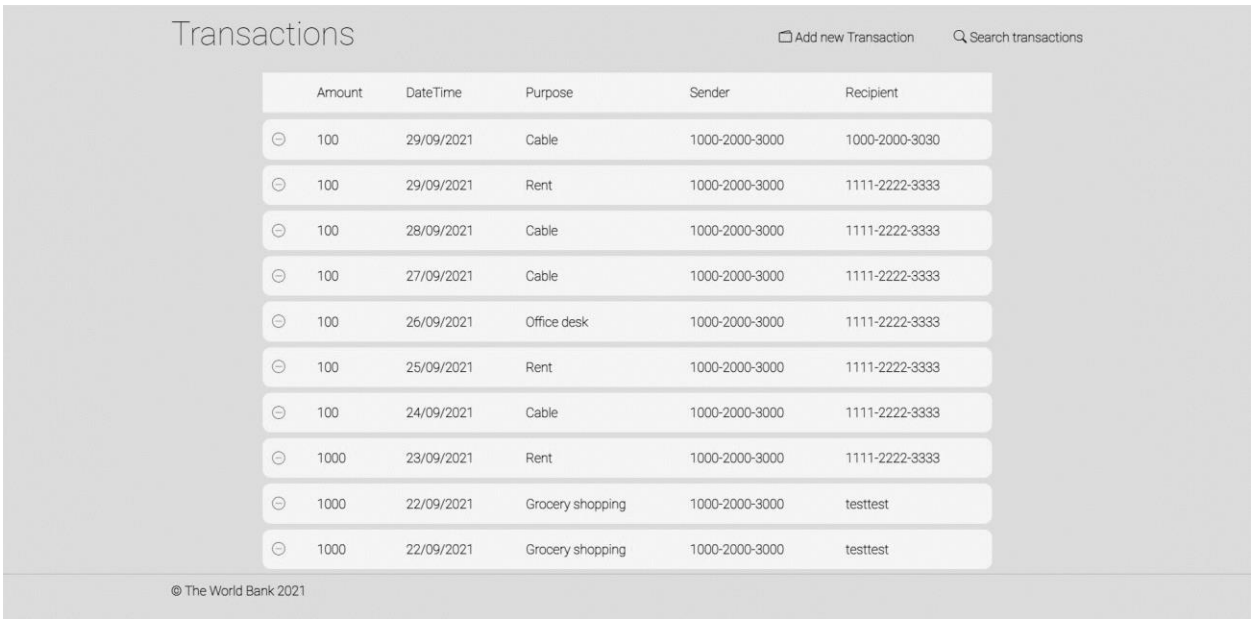
Слика 59 - Страница за унос нове трансакције

Основни сценарио СК

1. Корисник **уноси** податке о трансакцији. (АПУСО)
2. Корисник **контролише** да ли је коректно унео податке о трансакцији. (АНСО)
3. Корисник **позива** систем да запамти трансакцију. (АПСО)

Опис акције: Корисник притиском на дугме Create позива системску операцију ZapamtiTransakciju

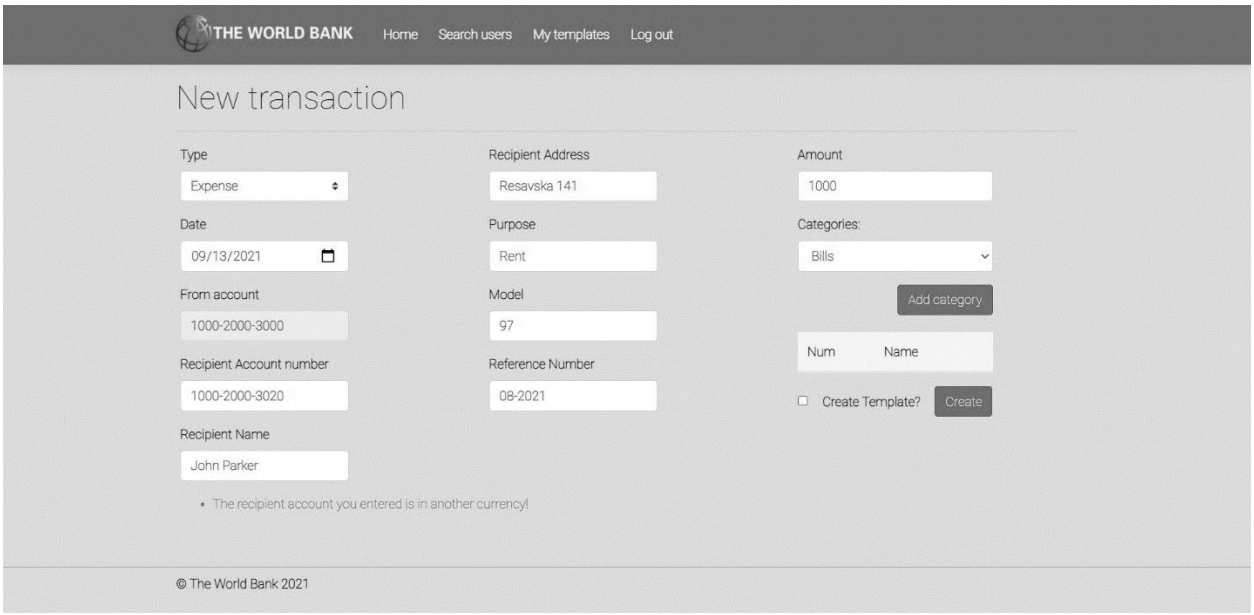
4. Систем **памти** трансакцију. (СО)
5. Систем **приказује** кориснику запамћену трансакцију и поруку: "Систем је запамтио трансакцију". (ИА)



Слика 60 - Приказ креираних трансакција у оквиру странице о детаљима рачуна

Алтернативна сценарија

5.1 Уколико систем не може да запамти податке о трансакцији он приказује кориснику поруку “Систем не може да запамти трансакцију”. (ИА)



Слика 61 - Приказ неуспешног креирања трансакције

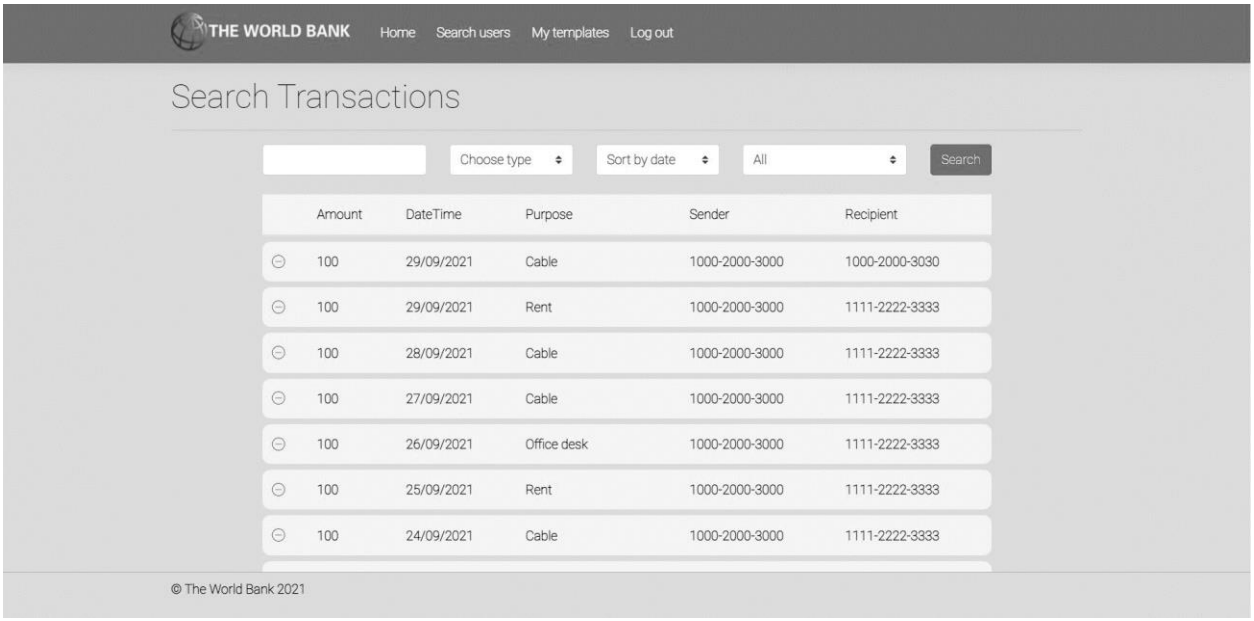
СК5: Случај коришћења – Претрага трансакција

Назив СК: Претраживање трансакција

Актори СК: Корисник

Учесници СК: Корисник и систем (програм)

Предуслов: Систем је укључен и корисник је улогован под својом шифром. Систем приказује форму за рад са трансакцијама. Учитана су листе категорија и трансакција.



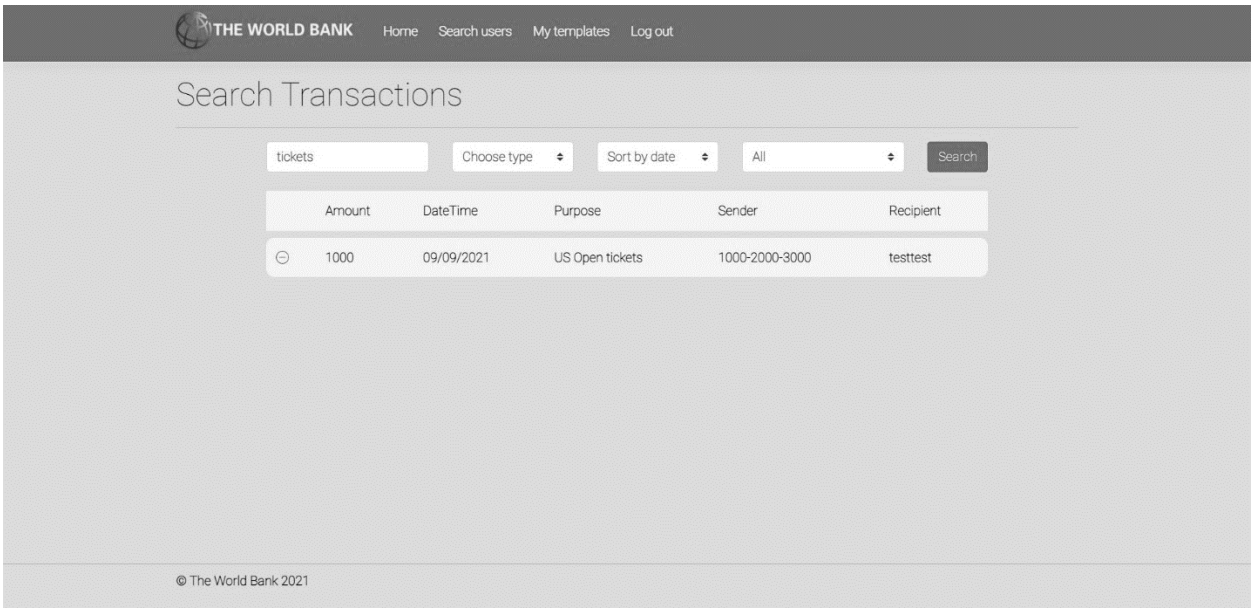
Слика 62 - Страница претраге трансакција

Основни сценарио СК

1. Корисник **уноси** вредност по којој претражује трансакције. (АПУСО)
2. Корисник **позива** систем да нађе трансакције по задатој вредности. (АПСО)

Опис акције: Корисник притиском на дугме Search позива системску операцију НађиTransakcije.

3. Систем **тражи** трансакције по задатој вредности. (СО)
4. Систем **приказује** кориснику податке о трансакцијама и поруку: “Систем је нашао трансакције по задатој вредности”. (ИА)

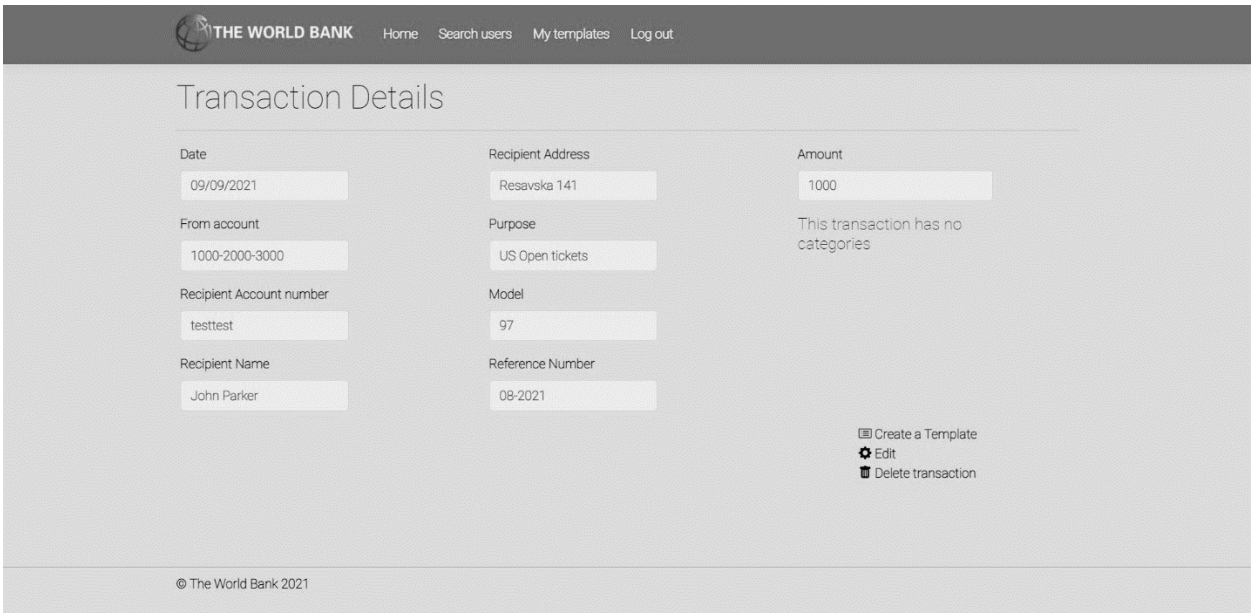


Слика 63 - Приказ резултата претраге трансакција

5. Корисник **бира** трансакцију. (АПУСО)
6. Корисник **позива** систем да учита трансакцију. (АПСО)

Опис акције: Корисник притиском на назив жељене трансакције позива системску операцију УčitajTransakciju.

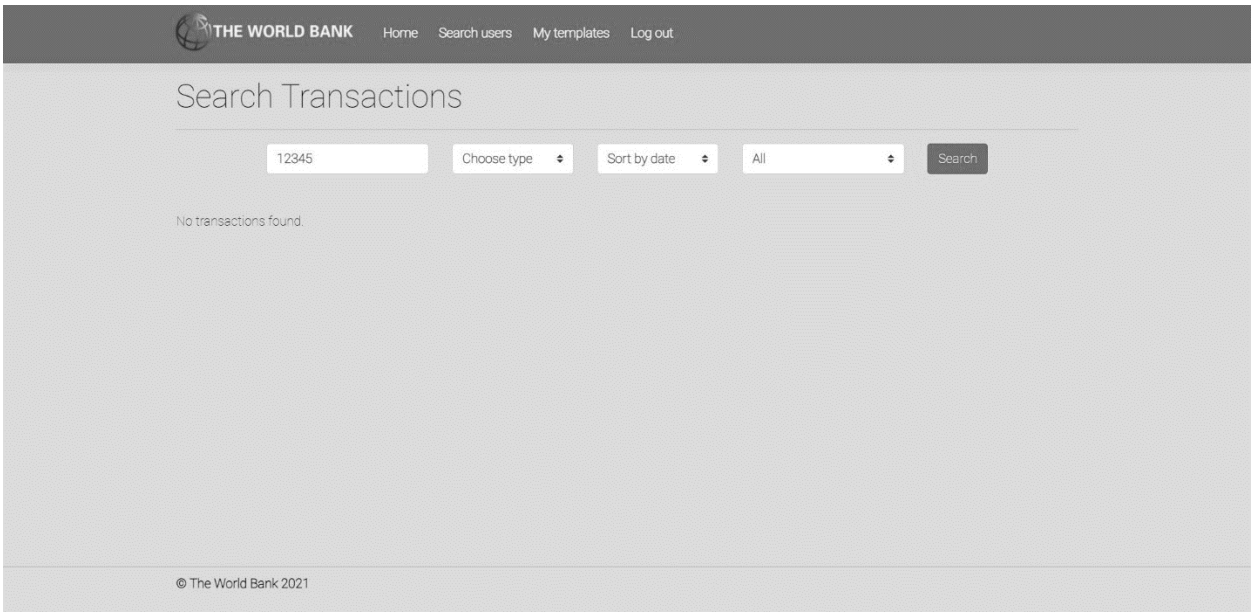
7. Систем **учитава** трансакцију. (СО)
8. Систем **приказује** кориснику податке о трансакцији и поруку: “Систем је учитао трансакцију”. (ИА)



Слика 64 - Приказ успешно учитане трансакције

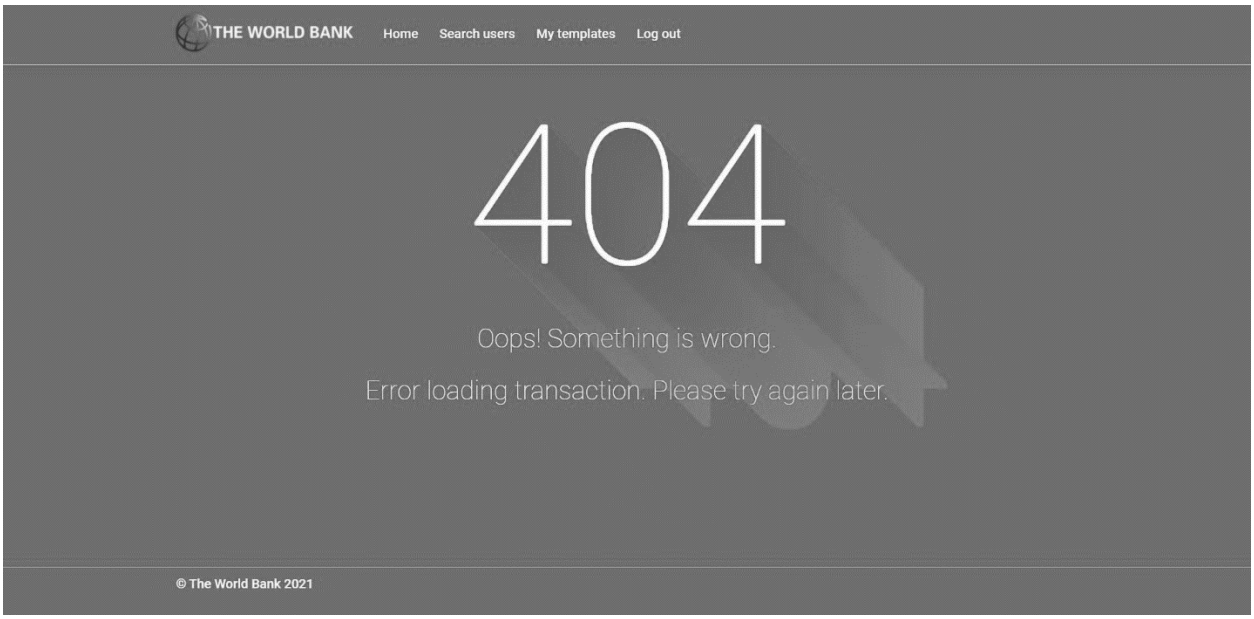
Алтернативна сценарија

4.1 Уколико систем не може да нађе трансакције он приказује кориснику поруку: “Систем не може да нађе трансакције по задатој вредности”. Прекида се извршење сценариа. (ИА)



Слика 65 - Приказ неуспешне претраге трансакција

8.1 Уколико систем не може да учита трансакцију он приказује кориснику поруку “Систем не може да учита трансакцију”. (ИА)



Слика 66 - Приказ грешке приликом учитавања трансакције

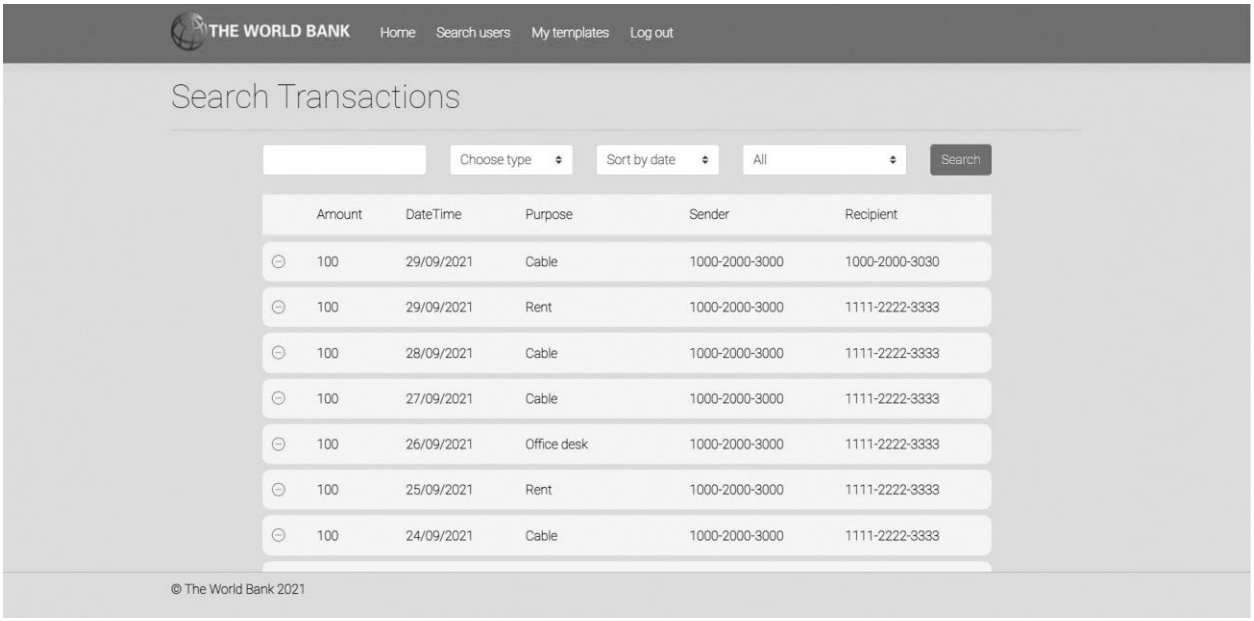
СК6: Случај коришћења – Измена трансакција

Назив СК: Измена трансакција

Актори СК: Корисник

Учесници СК: Корисник и систем (програм)

Предуслов: Систем је укључен и корисник је улогован под својом шифром. Систем приказује форму за рад са трансакцијом. Учитана су листе трансакција и категорија.



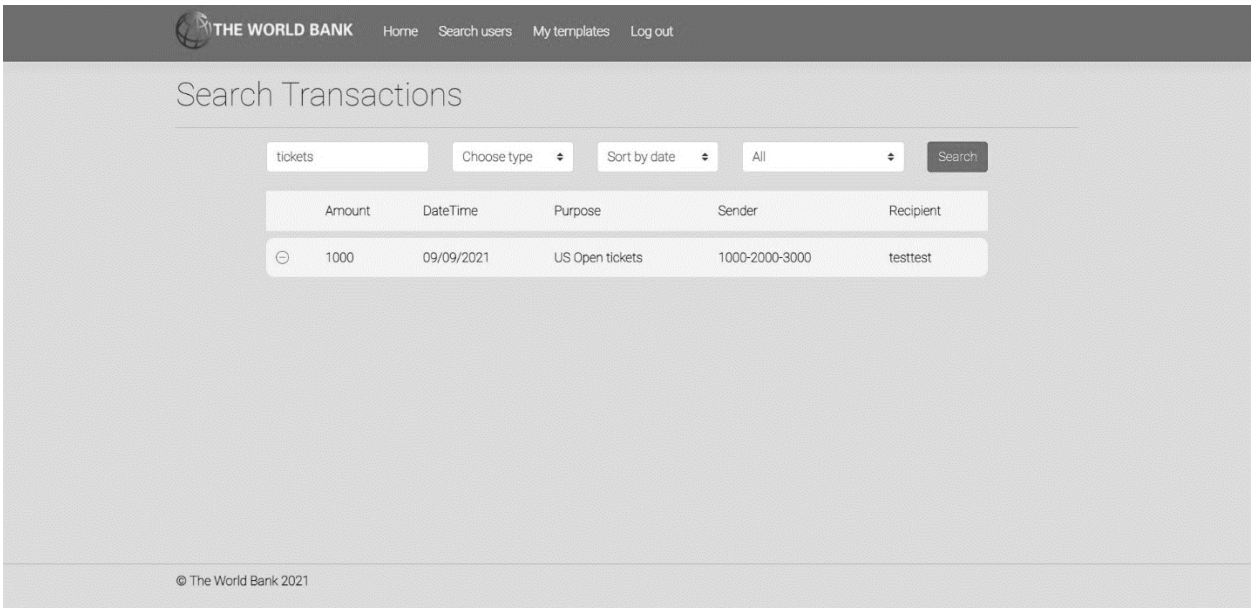
Слика 67 - Почетна страница за претрагу трансакција

Основни сценарио СК

1. Корисник **уноси** вредност по којој претражује трансакције. (АПУСО)
2. Корисник **позива** систем да нађе трансакције по задатој вредности. (АПСО)

Опис акције: Корисник притиском на дугме Search позива системску операцију НађиTransakcije.

3. Систем **тражи** трансакције по задатој вредности. (СО)
4. Систем **приказује** кориснику трансакције и поруку: “Систем је нашао трансакције по задатој вредности”. (ИА)

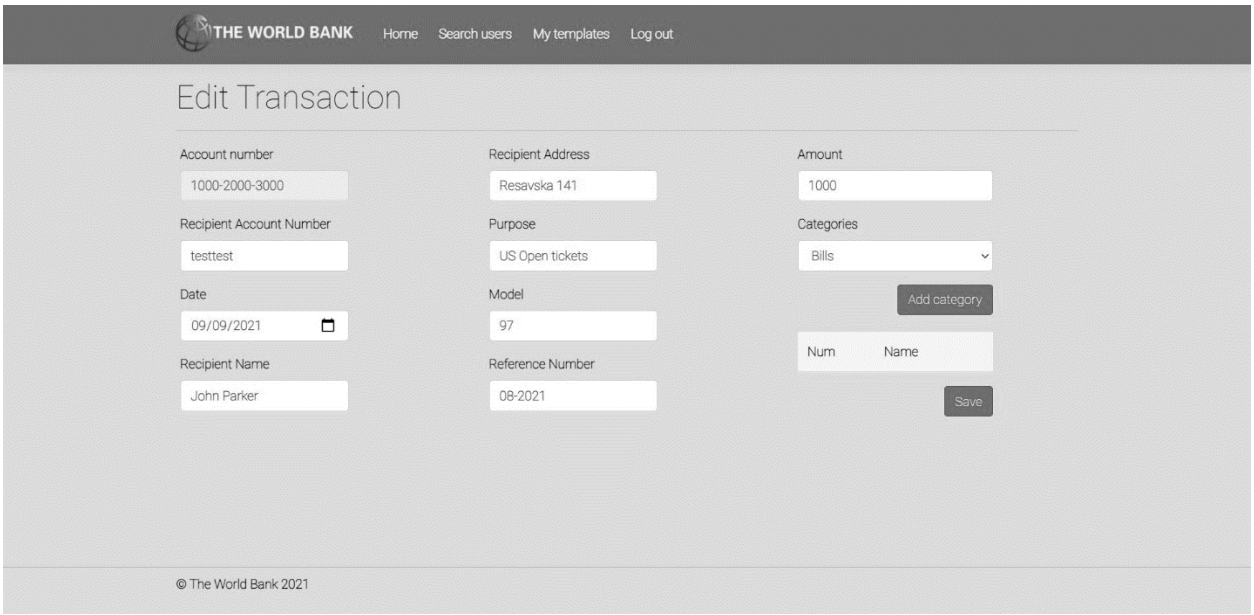


Слика 68 - Резултат претраге трансакција

5. Корисник **бира** трансакцију. (АПУСО)
6. Корисник **позива** систем да учита трансакцију. (АПСО)

Опис акције: Корисник притиском на назив жељене трансакције позива системску операцију УčitajTransakciju.

7. Систем **учитава** трансакцију. (СО)
8. Систем **показује** кориснику податке о трансакцији и поруку "Систем је учитао трансакцију" (ИА)

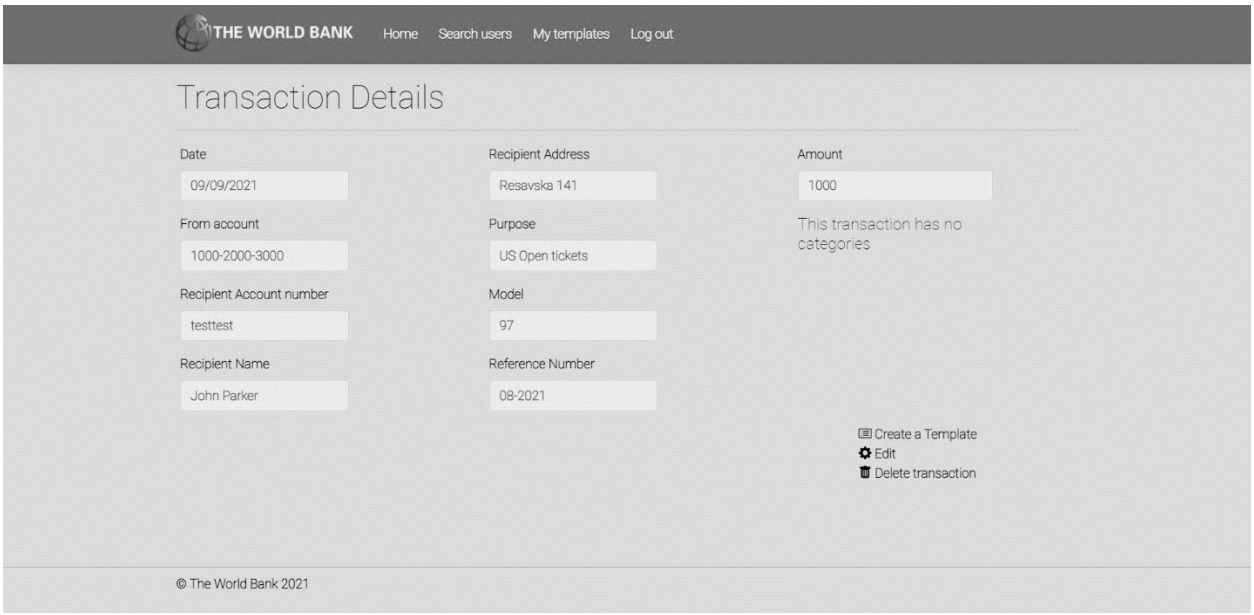


Слика 69 - Страница за измену трансакције

9. Корисник **уноси** (мења) податке о трансакцији. (АПУСО)
10. Корисник **контролише** да ли је коректно унео податке о трансакцији. (АНСО)
11. Корисник **позива** систем да запамти податке о трансакцији. (АПСО)

Опис акције: Корисник притиском на дугме Save позива системску операцију АžurirajTransakciju.

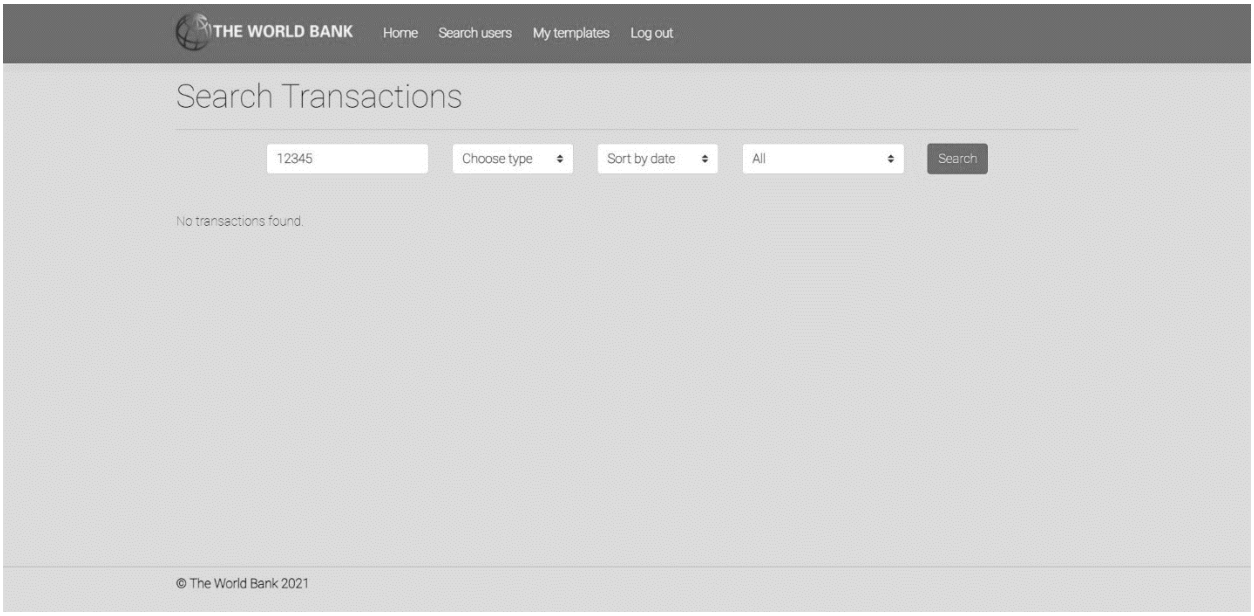
12. Систем **памти** податке о трансакцији. (СО)
13. Систем **приказује** кориснику запамћену трансакцију и поруку: "Систем је запамтио трансакцију." (ИА)



Слика 70 - Приказ сачуване трансакције након измене

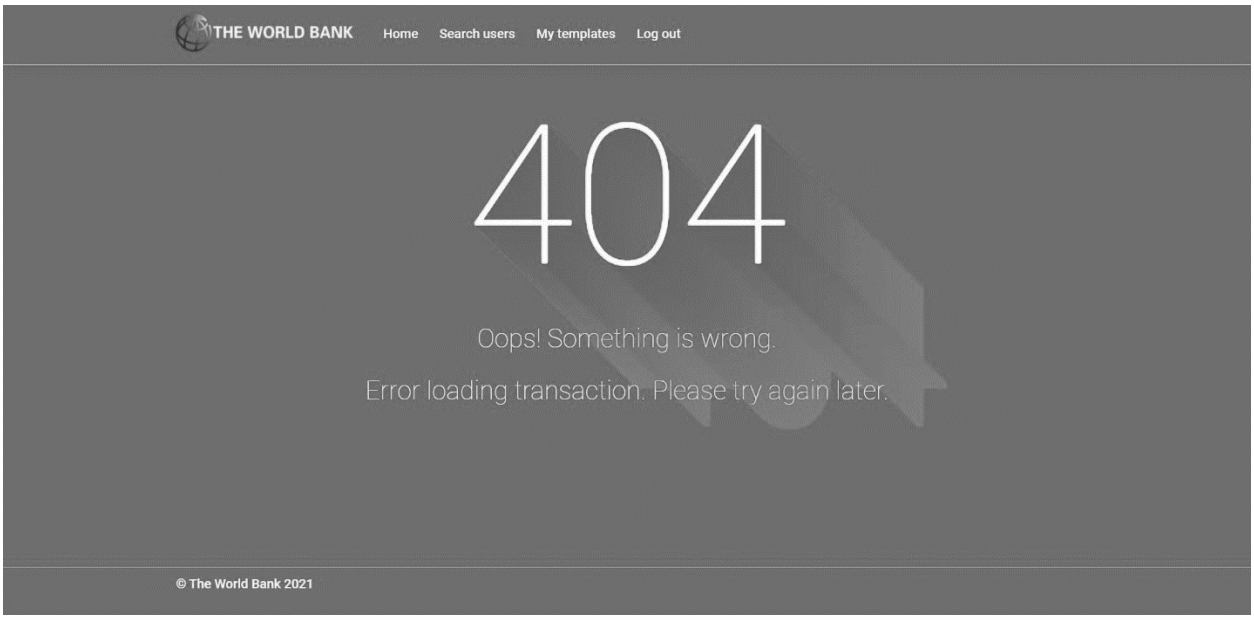
Алтернативна сценарија

4.1 Уколико систем не може да нађе трансакције он приказује кориснику поруку: "Систем не може да нађе трансакције по задатој вредности". Прекида се извршење сценариа. (ИА)



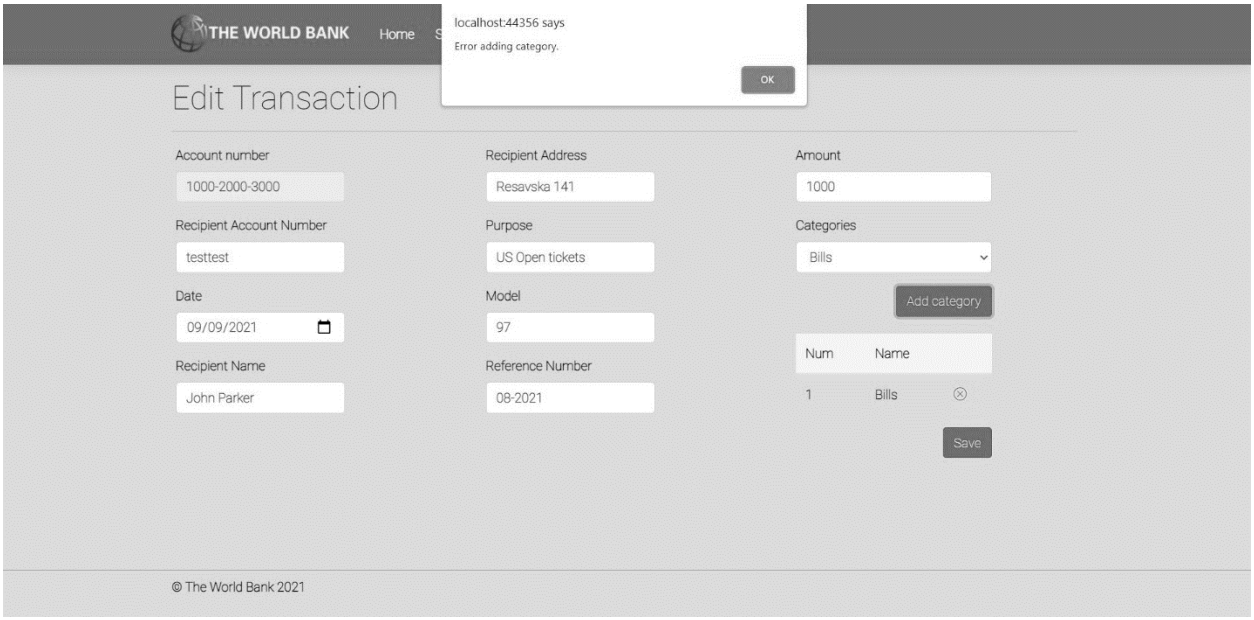
Слика 71 - Неуспешна претрага трансакција приликом измене

8.1 Уколико систем не може да учита трансакцију он приказује кориснику поруку “Систем не може да учита трансакцију”. Прекида се извршење сценариа. (ИА)



Слика 72 - Страница о грешци приликом учитавања трансакције

13.1 Уколико систем не може да запамти податке о трансакцији он приказује кориснику поруку “Систем не може да запамти трансакцију”. (ИА)



Слика 73 - Приказ грешке приликом измене трансакције

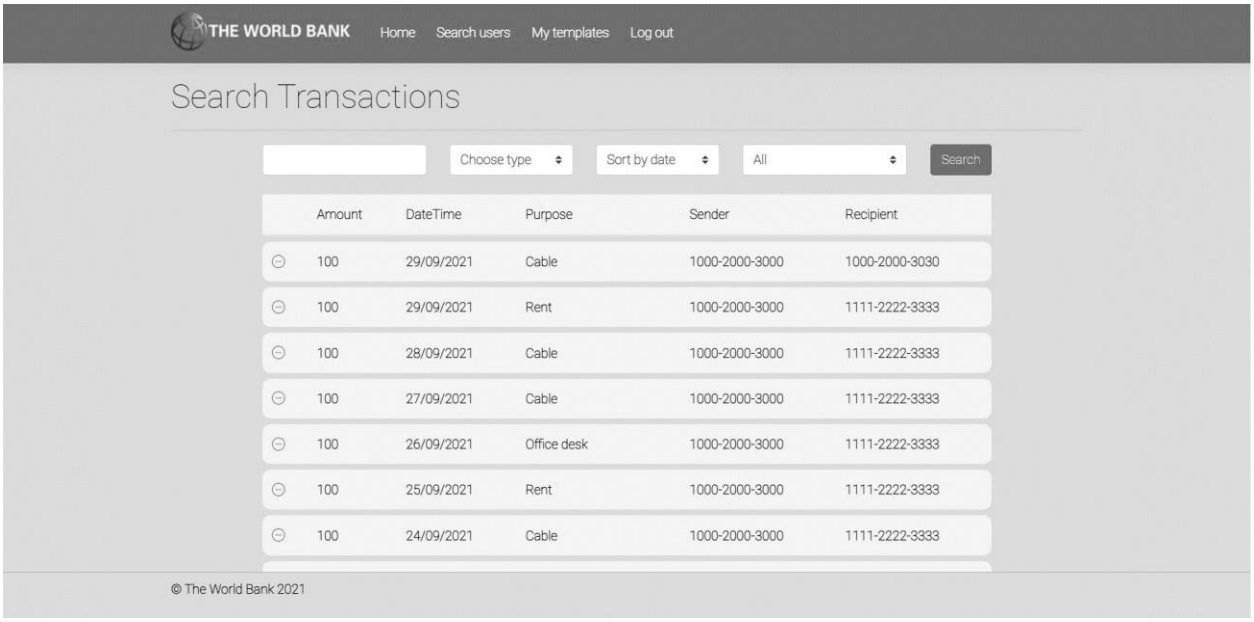
СК7: Случај коришћења – Брисање трансакције

Назив СК: Брисање трансакције

Актори СК: Корисник

Учесници СК: Корисник и систем (програм)

Предуслов: Систем је укључен и корисник је улогован под својом шифром. Систем приказује форму за рад са трансакцијом. Учитана су листе категорија и трансакција.



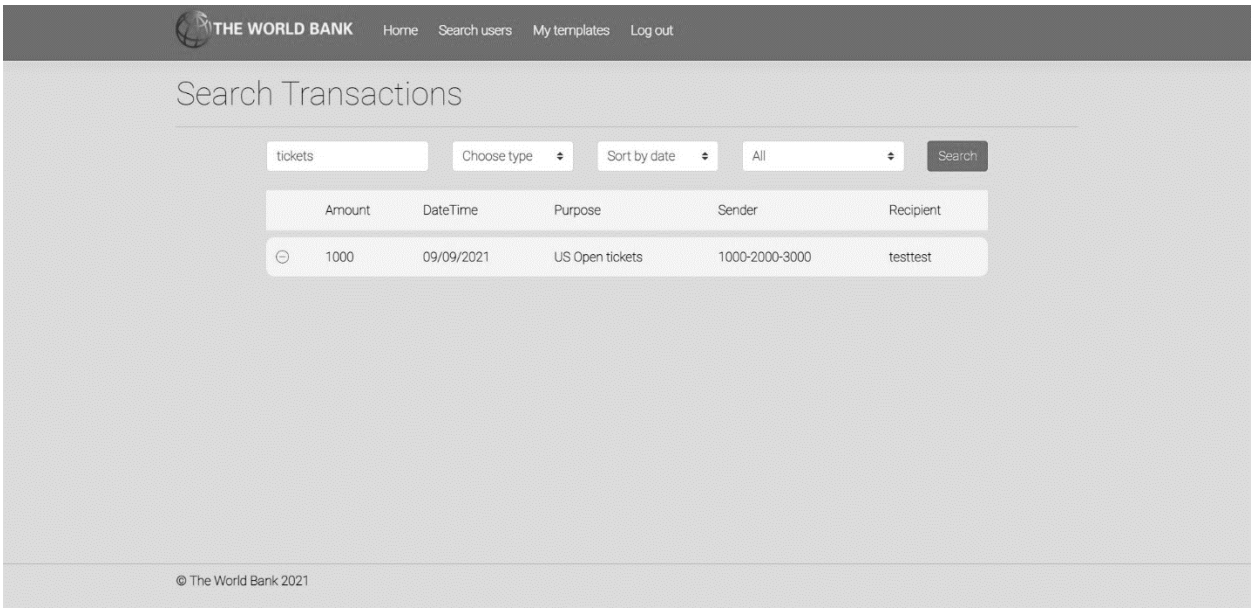
Слика 74 - Приказ свих трансакција са корисниковог рачуна

Основни сценарио СК

1. Корисник **уноси** вредност по којој претражује трансакције. (АПУСО)
2. Корисник **позива** систем да нађе трансакције по задатој вредности. (АПСО)

Опис акције: Корисник притиском на дугме Search позива системску операцију НађиTransakcije.

3. Систем **тражи** трансакције по задатој вредности. (СО)
4. Систем **приказује** кориснику трансакције и поруку: “Систем је нашао трансакције по задатој вредности”. (ИА)

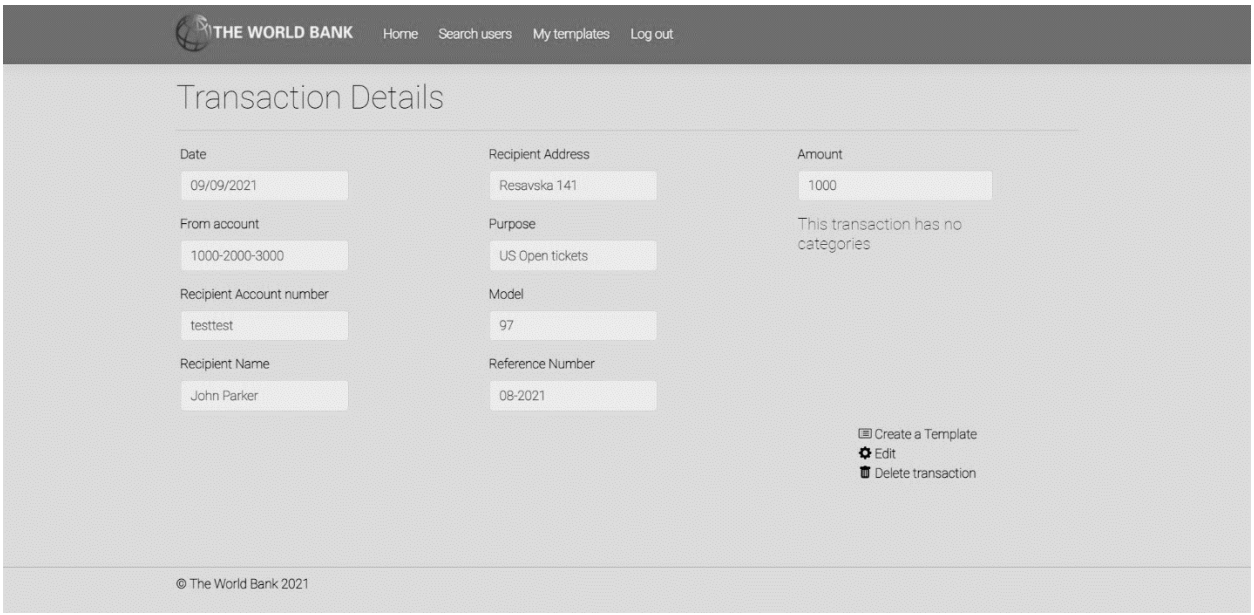


Слика 75 - Приказ резултата претраге трансакција

5. Корисник **бира** трансакцију који жели да обрише. (АПУСО)
6. Корисник **позива** систем да учита податке о одабраној трансакцији. (АПСО)

Опис акције: Корисник притиском на назив жељене трансакције позива системску операцију УчитајTransakciju.

7. Систем **учитава** податке о одабраном трансакцији. (СО)
8. Систем **обавештава** корисника о успешном учитавању података поруком "Систем је учитао одабран трансакцију" и приказује податке о трансакцији. (ИА)

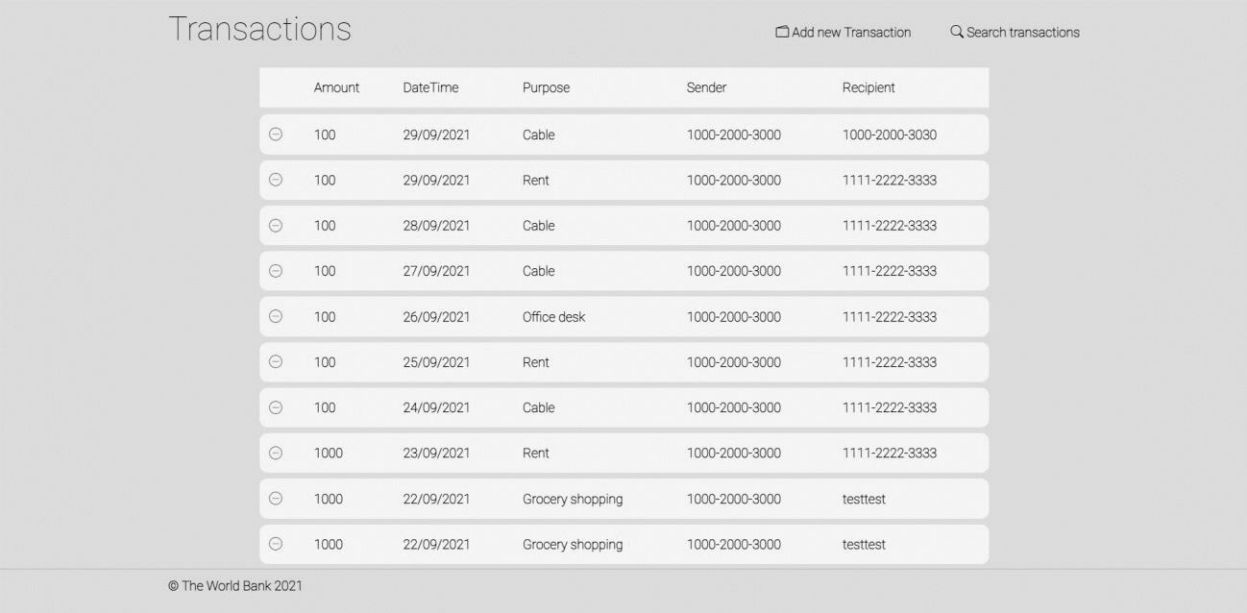


Слика 76 - Страница са детаљима трансакције

9. Корисник **позива** систем да обрише трансакцију. (АПСО)

Опис акције: Корисник притиском на дугме Delete transaction позива системску операцију ОбришиTransakciju.

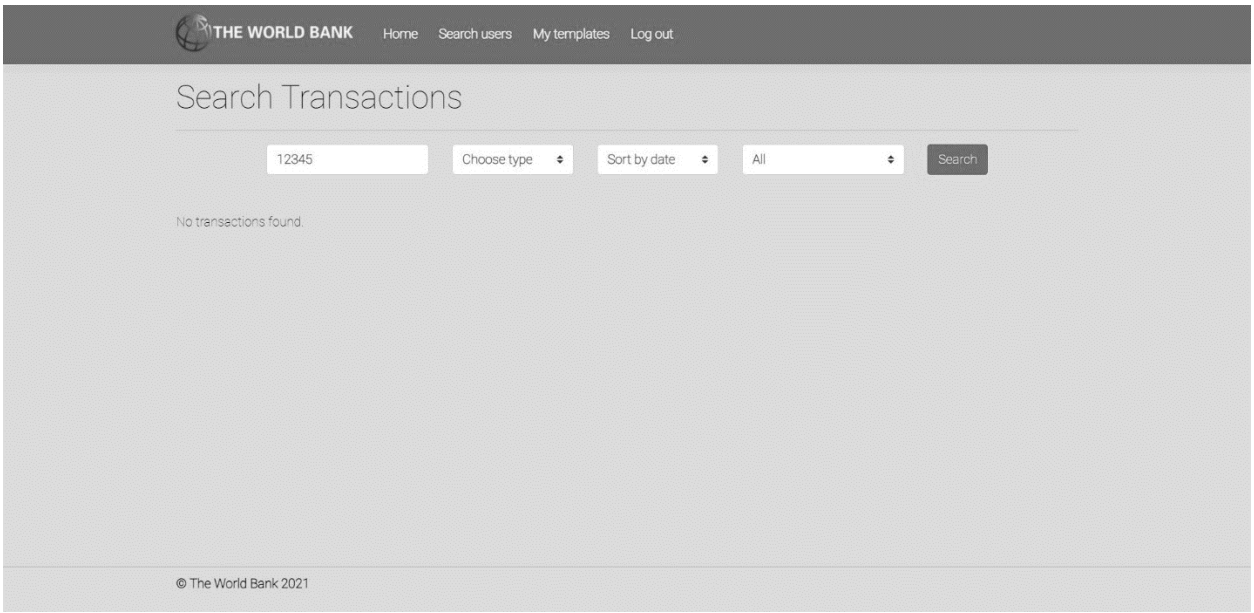
10. Систем **брише** трансакцију. (СО)
11. Систем **приказује** кориснику поруку: "Систем је обрисао трансакцију." (ИА)



Слика 77 - Приказ листе трансакција након брисања трансакције

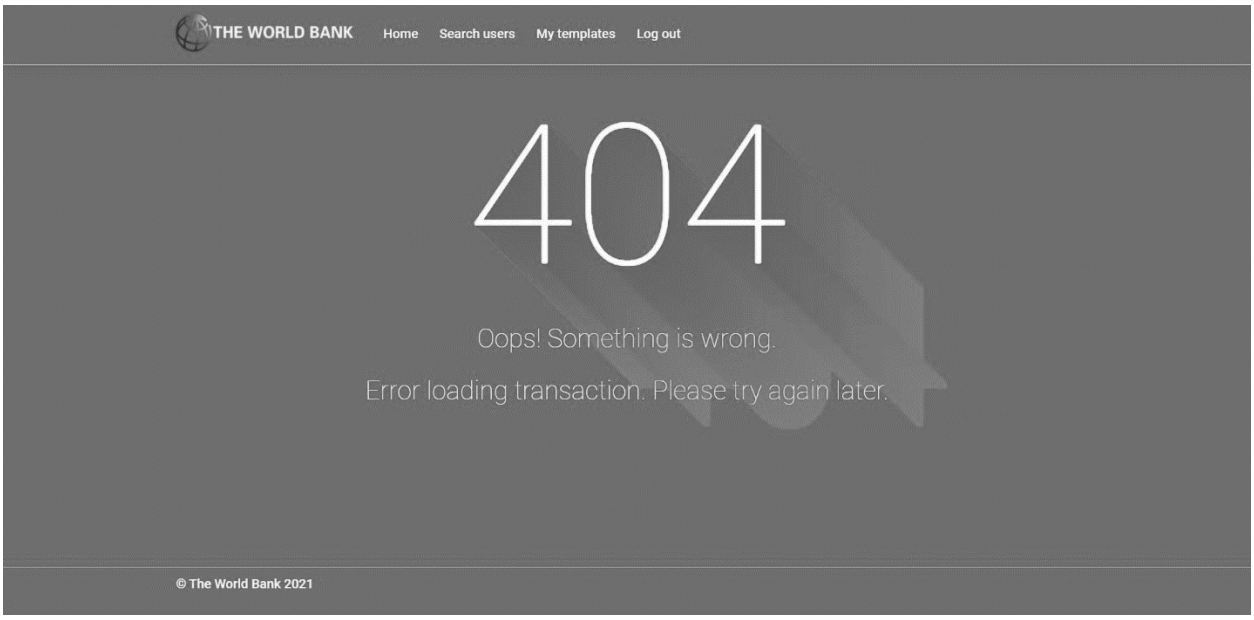
Алтернативна сценарија

4.1 Уколико систем не може да нађе трансакције он приказује кориснику поруку: “Систем не може да нађе трансакције по задатој вредности”. Прекида се извршење сценариа. (ИА)



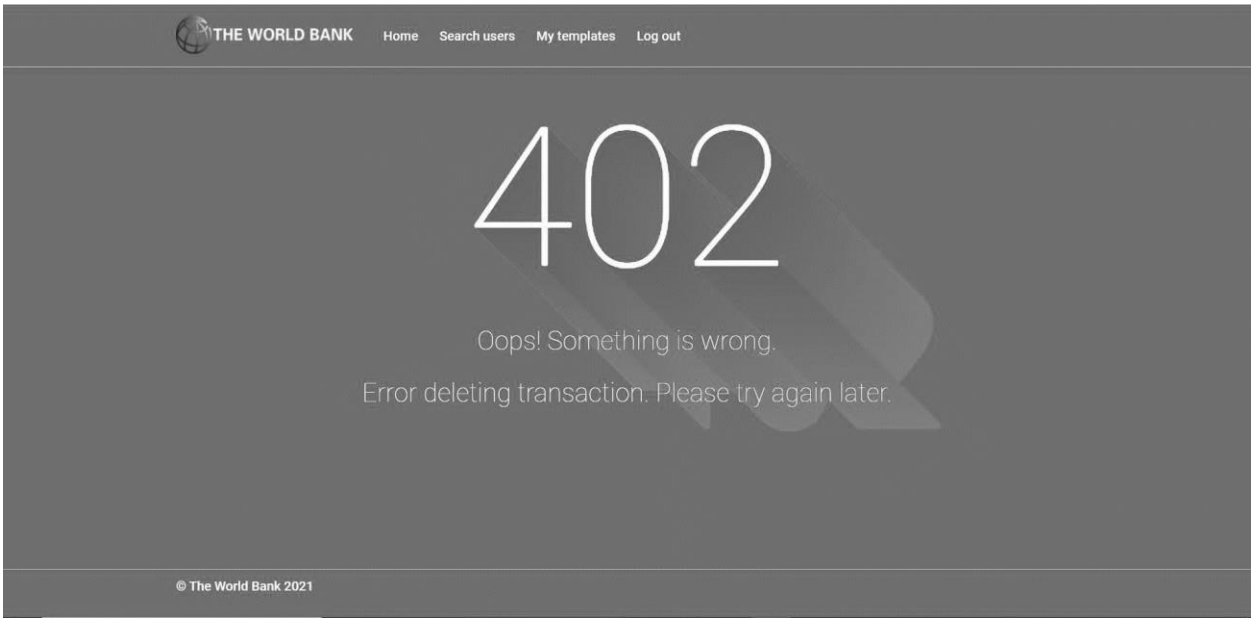
Слика 78 - Неуспешна претрага приликом брисања трансакције

8.1 Уколико систем не може да учита изабран трансакцију он приказује кориснику поруку “Систем не може да учита трансакцију“. Прекида се извршење сценариа (ИА)



Слика 79 - Порука о грешци приликом учитавања трансакције

11.1 Уколико систем не може да обрише трансакцију он приказује кориснику поруку “Систем не може да обрише трансакцију”. (ИА)



Слика 80 - Порука о грешци приликом брисања трансакције

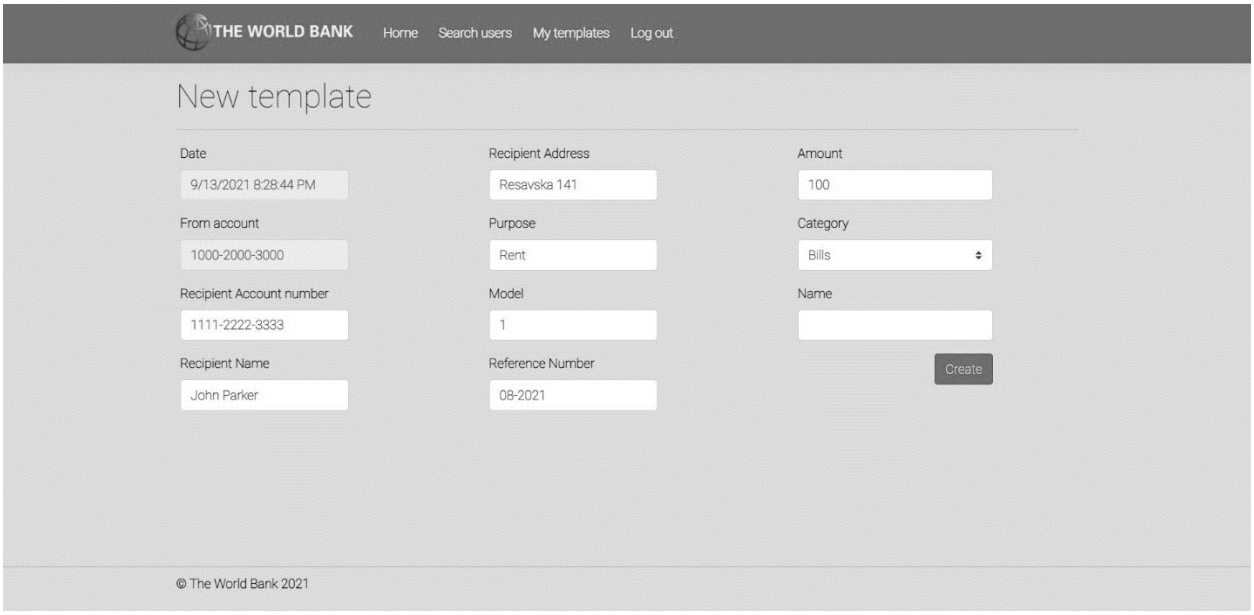
СК8: Случај коришћења – Креирање шаблона трансакције

Назив СК: Креирање шаблона трансакције

Актори СК: Корисник

Учесници СК: Корисник и систем (програм)

Предуслов: Систем је укључен и корисник је улогован под својом шифром. Систем приказује форму за рад са шаблоном. Учитана је листа категорија.



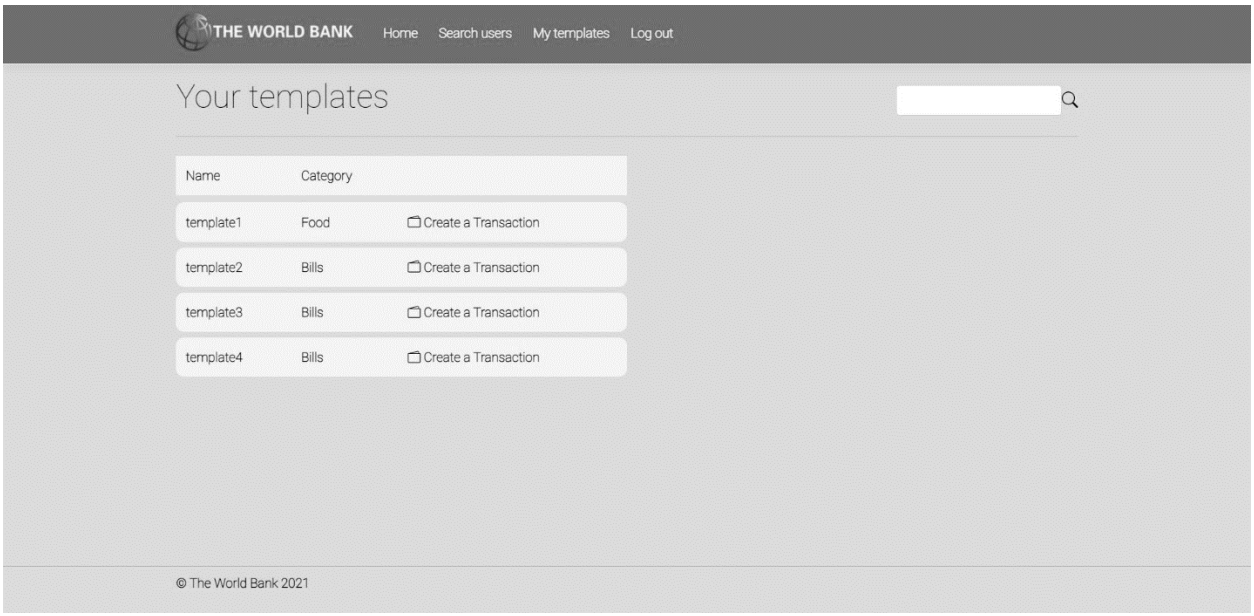
Слика 81 - Страница за креирање шаблона трансакције

Основни сценарио СК

1. Корисник **уноси** податке о шаблону. (АПУСО)
2. Корисник **контролише** да ли је коректно унео податке о шаблону. (АНСО)
3. Корисник **позива** систем да запамти шаблон. (АПСО)

Опис акције: Корисник притиском на дугме Create позива системску операцију ZapamtiŠablon.

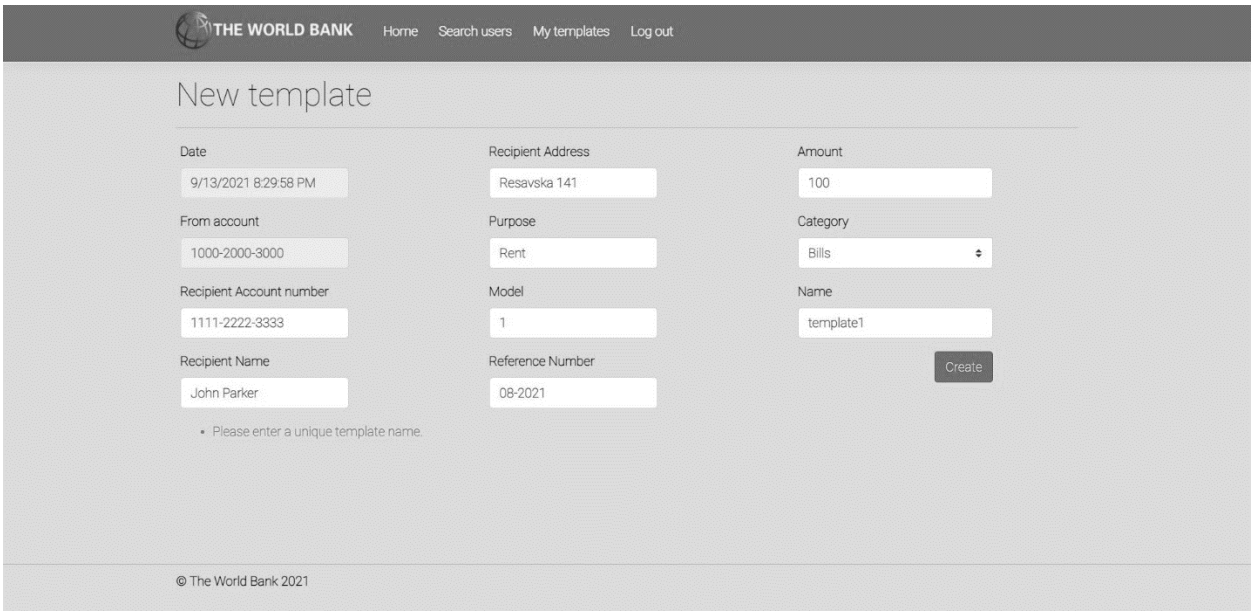
4. Систем **памти** шаблон. (СО)
5. Систем **приказује** кориснику запамћени шаблон и поруку: “Систем је запамтио шаблон“. (ИА)



Слика 82 - Приказ сачуваних шаблона

Алтернативна сценарија

5.1 Уколико систем не може да запамти податке о шаблону он приказује кориснику поруку “Систем не може да запамти шаблон”. (ИА)



Слика 83 - Приказ неуспешног креирања шаблона

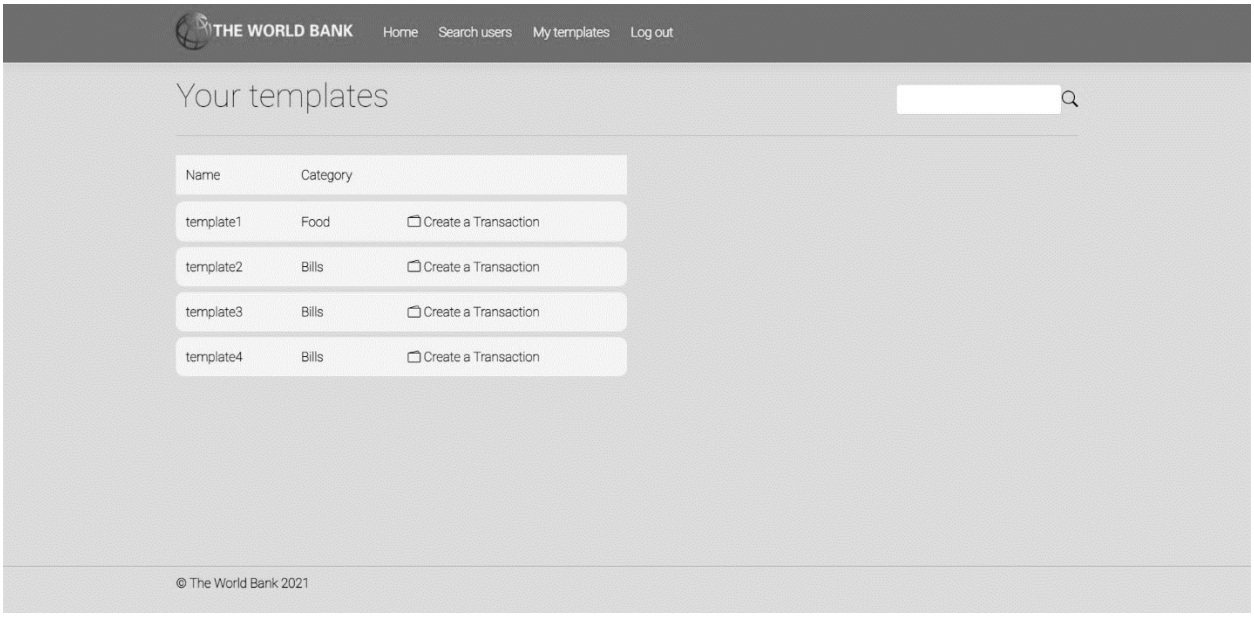
СК9: Случај коришћења – Измена шаблона трансакције

Назив СК: Измена шаблона трансакције

Актори СК: Корисник

Учесници СК: Корисник и систем (програм)

Предуслов: Систем је укључен и корисник је улогован под својом шифром. Систем приказује форму за рад са шаблоном. Учитана је листа шаблона и категорија.



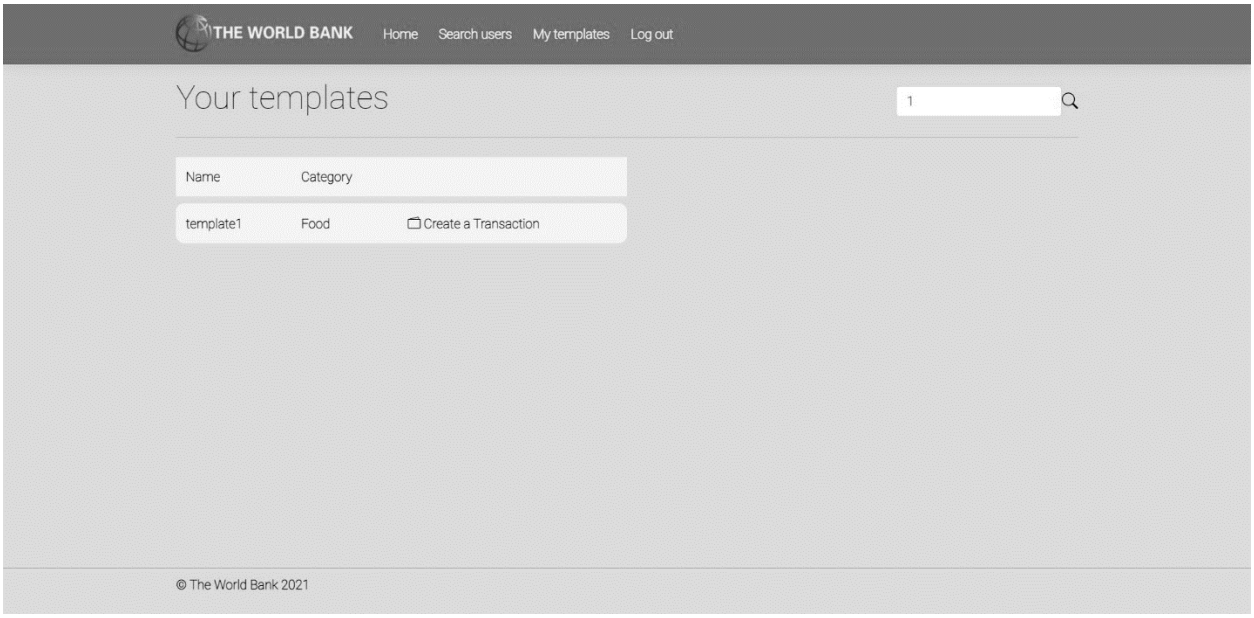
Слика 84 - Приказ корисникових шаблона

Основни сценарио СК

1. Корисник **уноси** вредност по којој претражује шаблоне. (АПУСО)
2. Корисник **позива** систем да нађе шаблоне по задатој вредности. (АПСО)

Опис акције: Корисник притиском на иконицу лупе позива системску операцију НађиШablone.

3. Систем **тражи** шаблоне по задатој вредности. (СО)
4. Систем **приказује** кориснику шаблоне и поруку: “Систем је нашао шаблоне по задатој вредности”. (ИА)

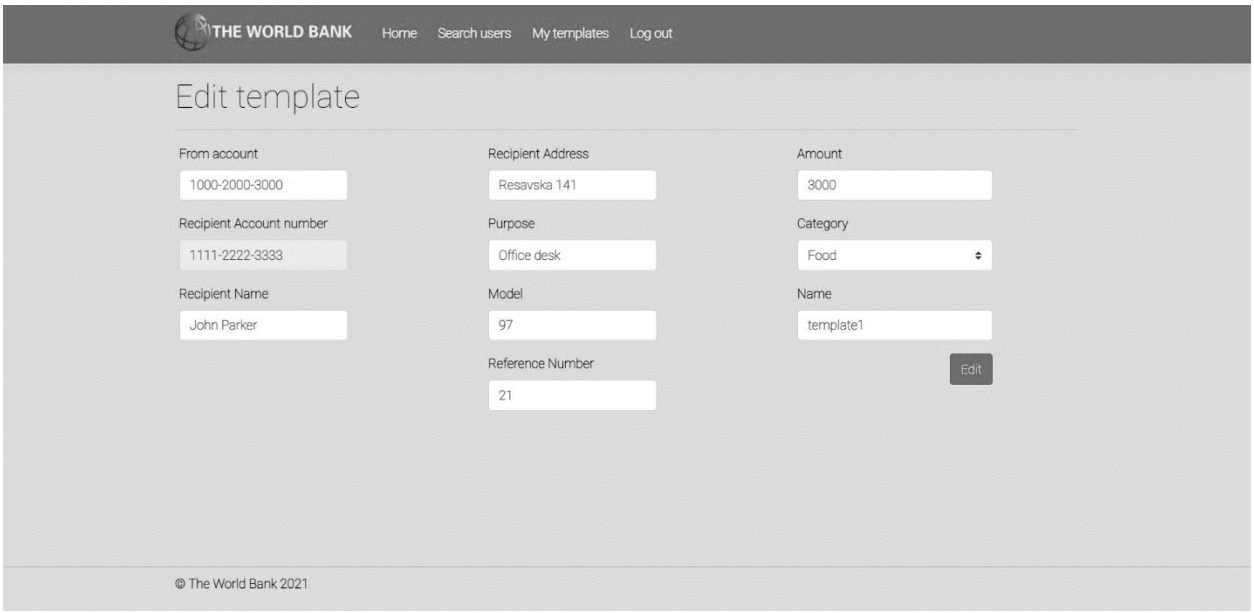


Слика 85 - Приказ успешне претраге шаблона

5. Корисник **бира** шаблон. (АПУСО)
6. Корисник **позива** систем да учита шаблон. (АПСО)

Опис акције: Корисник притиском на назив жељеног шаблона позива системску операцију УчитајШablon.

7. Систем **учитава** шаблон. (СО)
8. Систем **показује** кориснику податке о шаблону и поруку "Систем је учитао шаблон" (ИА)



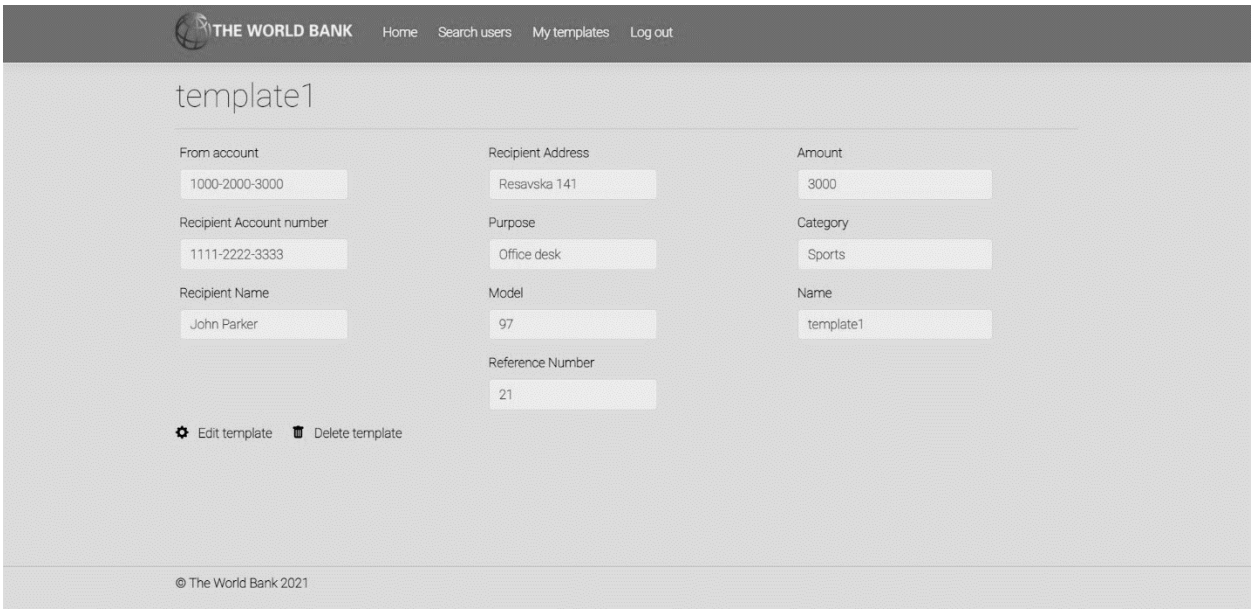
Слика 86 - Страница за измену шаблона трансакције

9. Корисник **уноси** (мења) податке о шаблону. (АПУСО)
10. Корисник **контролише** да ли је коректно унео податке о шаблону. (АНСО)

11. Корисник **позива** систем да запамти податке о шаблону. (АПСО)

Опис акције: Корисник притиском на дугме Edit позива системску операцију АжурирајШаблон.

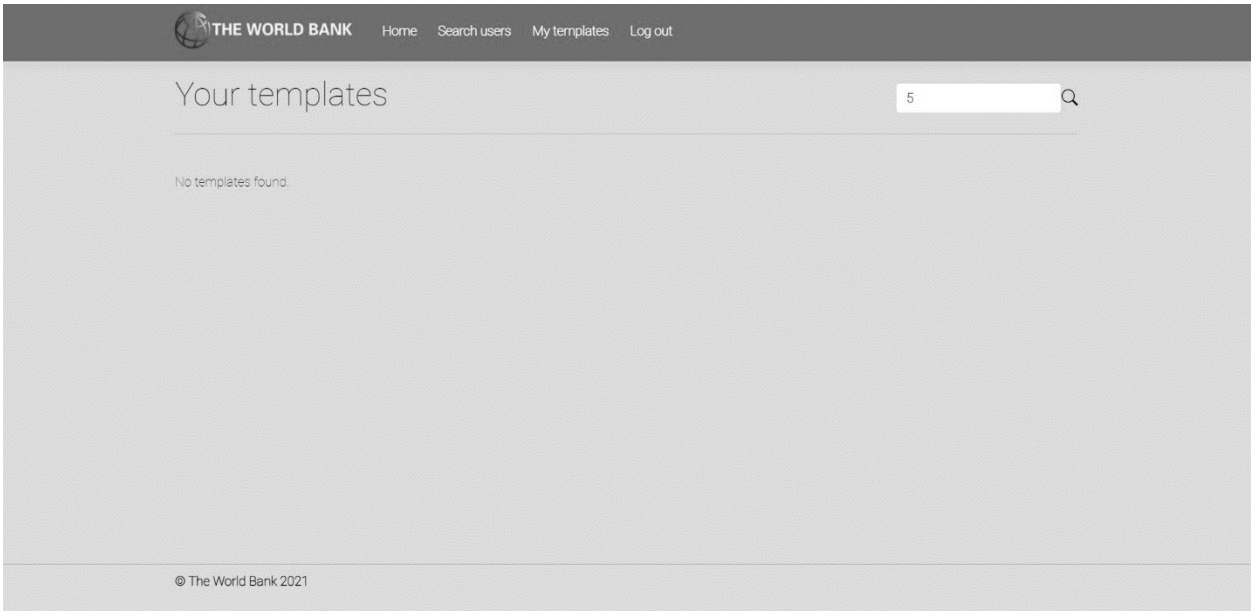
12. Систем **памти** податke о шаблону. (СО)
13. Систем **приказује** кориснику запамћени шаблон и поруку: “Систем је запамтио шаблон.” (ИА)



Слика 87 - Страница са детаљима сачуваног шаблона

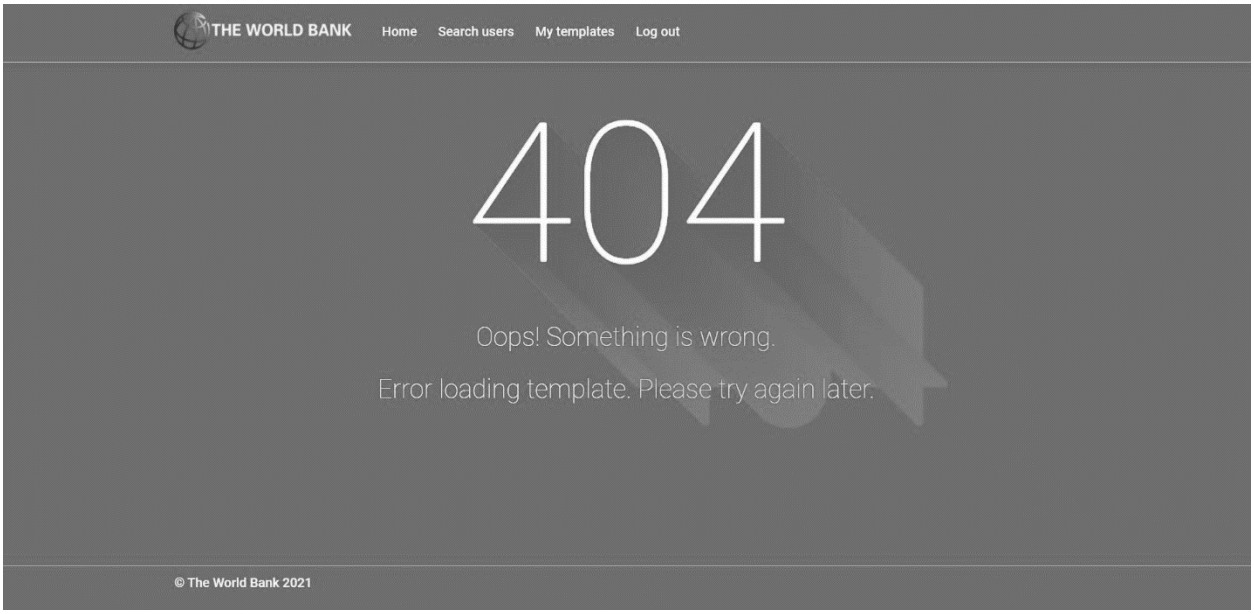
Алтернативна сценарија

4.1 Уколико систем не може да нађе шаблоне он приказује кориснику поруку: “Систем не може да нађе шаблоне по задатој вредности”. Прекида се извршење сценариа. (ИА)



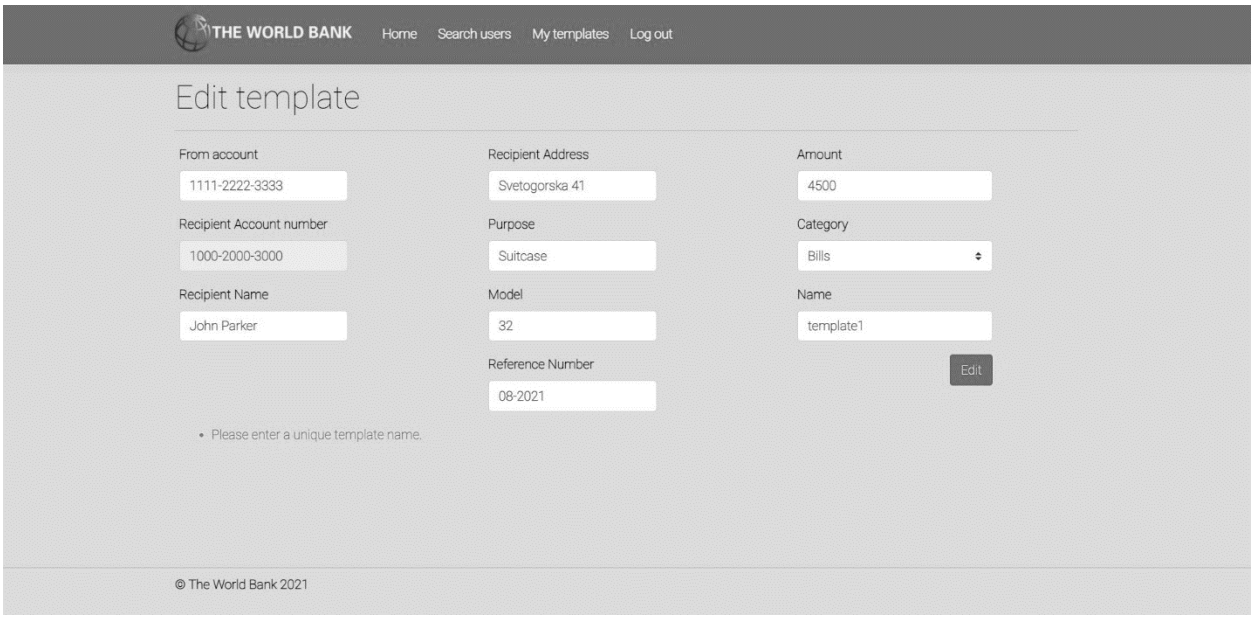
Слика 88 - Приказ неуспешне претраге шаблона

8.1 Уколико систем не може да учита шаблон он приказује кориснику поруку “Систем не може да учита шаблон”. Прекида се извршење сценариа. (ИА)



Слика 89 - Порука о грешци приликом учитавања шаблона

13.1 Уколико систем не може да запамти податке о шаблону он приказује кориснику поруку “Систем не може да запамти шаблон”. (ИА)



Слика 90 - Порука приликом неуспешне измене шаблона трансакције

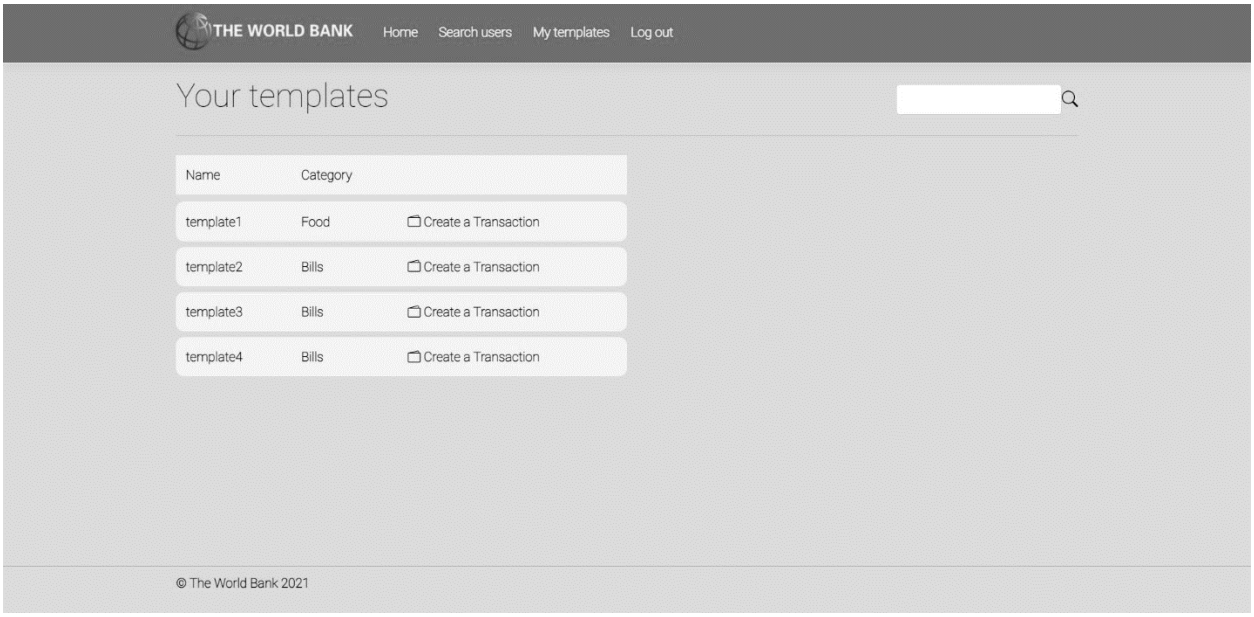
СК10: Случај коришћења – Брисање шаблона трансакције

Назив СК: Брисање шаблона трансакције

Актори СК: Корисник

Учесници СК: Корисник и систем (програм)

Предуслов: Систем је укључен и корисник је улогован под својом шифром. Систем приказује форму за рад са шаблоном. Учитана је листа шаблона.



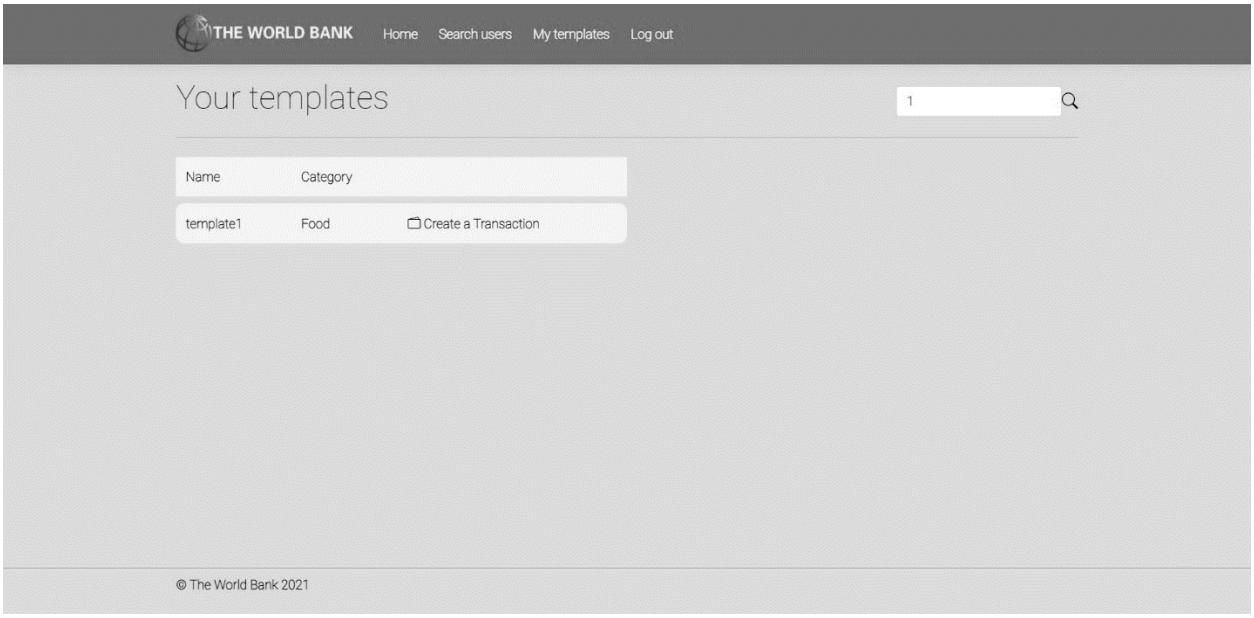
Слика 91 - Страница са корисниковим шаблонима

Основни сценарио СК

1. Корисник **уноси** вредност по којој претражује шаблоне. (АПУСО)
2. Корисник **позива** систем да нађе шаблоне по задатој вредности. (АПСО)

Опис акције: Корисник притиском на иконицу лупе позива системску операцију НађиШablone.

3. Систем **тражи** шаблоне по задатој вредности. (СО)
4. Систем **приказује** кориснику шаблоне и поруку: “Систем је нашао шаблоне по задатој вредности”. (ИА)

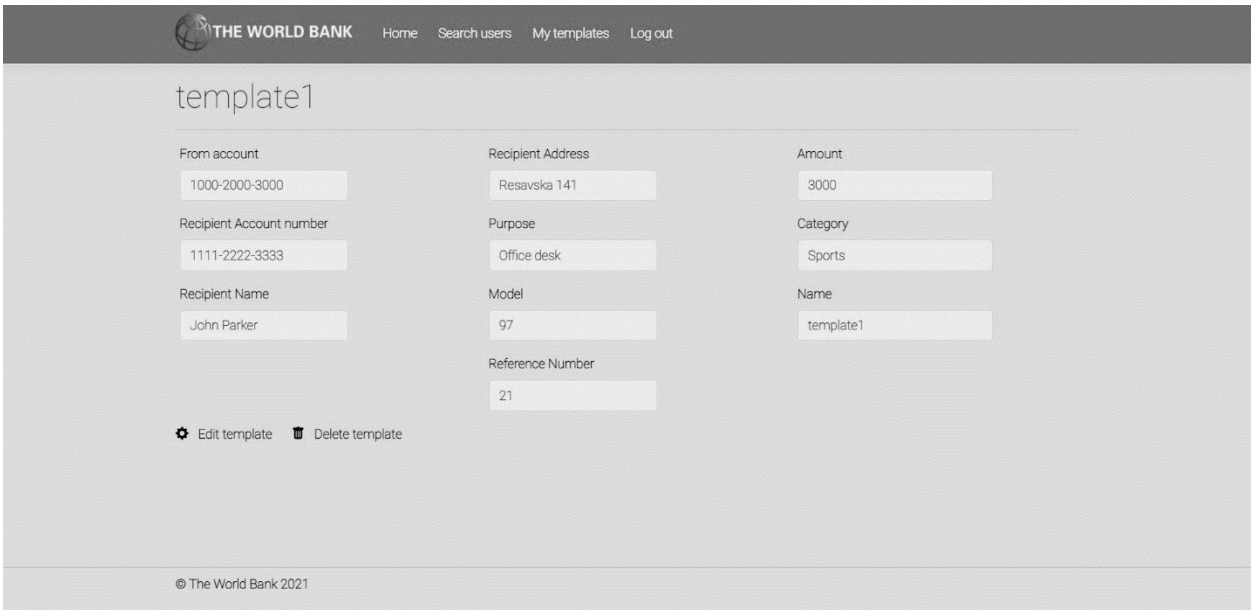


Слика 92 - Резултат претраге шаблона приликом брисања

5. Корисник **бира** шаблон који жели да обрише. (АПУСО)
6. Корисник **позива** систем да учита податке о одабраном шаблону. (АПСО)

Опис акције: Корисник притиском на назив жељеног шаблона позива системску операцију УчитајШablon.

7. Систем **учитава** податке о одабраном шаблону. (СО)
8. Систем **обавештава** корисника о успешном учитавању података поруком "Систем је учитао одабран шаблон" и приказује податке о шаблону. (ИА)

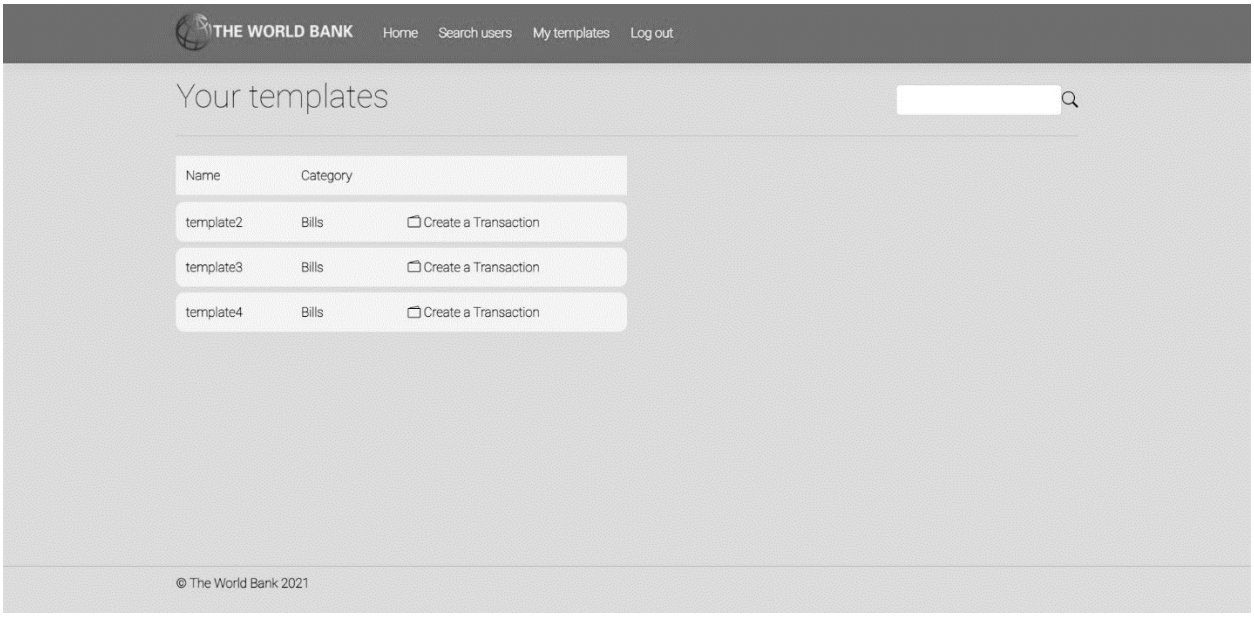


Слика 93 - Страница са детаљима шаблона трансакције

9. Корисник **позива** систем да обрише шаблон. (АПСО)

Опис акције: Корисник притиском на дугме Delete template позива системску операцију ОбришиШablon.

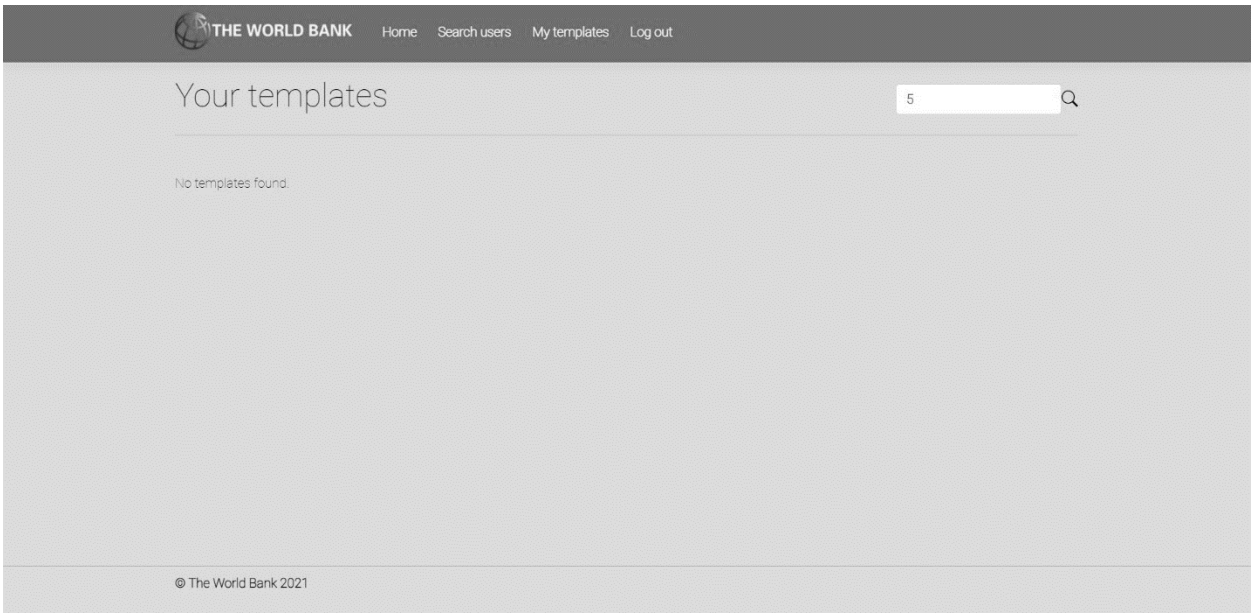
10. Систем **брише** шаблон. (СО)
11. Систем **приказује** кориснику поруку: “Систем је обрисао шаблон.” (ИА)



Слика 94 - Страница са корисниковим шаблонима након брисања

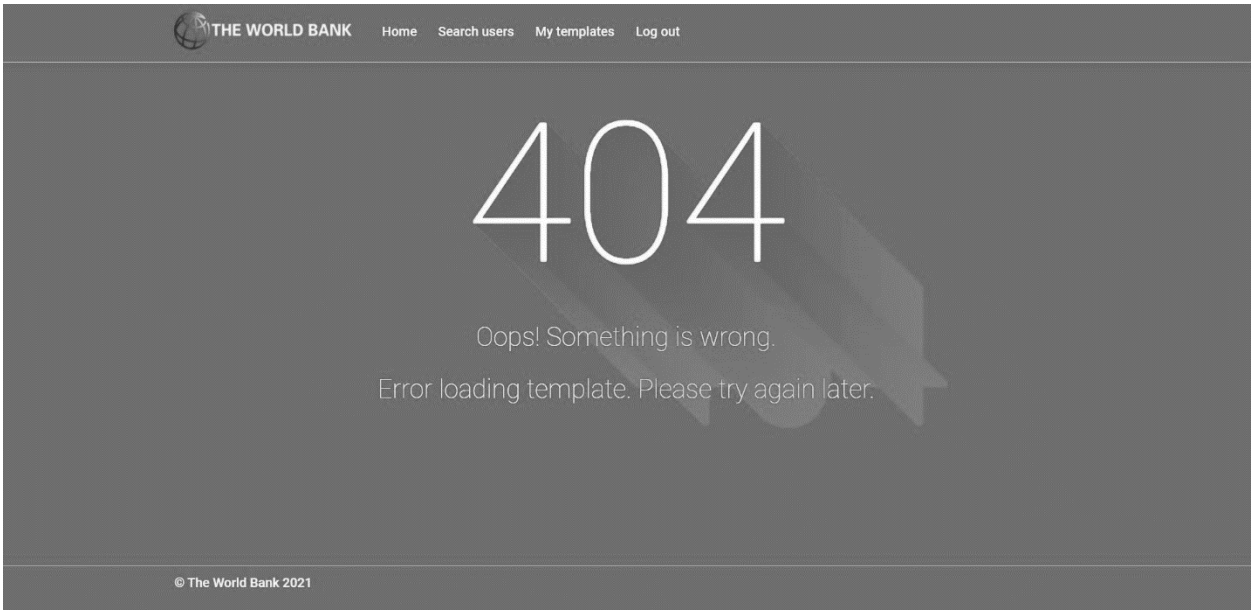
Алтернативна сценарија

4.1 Уколико систем не може да нађе шаблон он приказује кориснику поруку: “Систем не може да нађе шаблон по задатој вредности”. Прекида се извршење сценариа. (ИА)



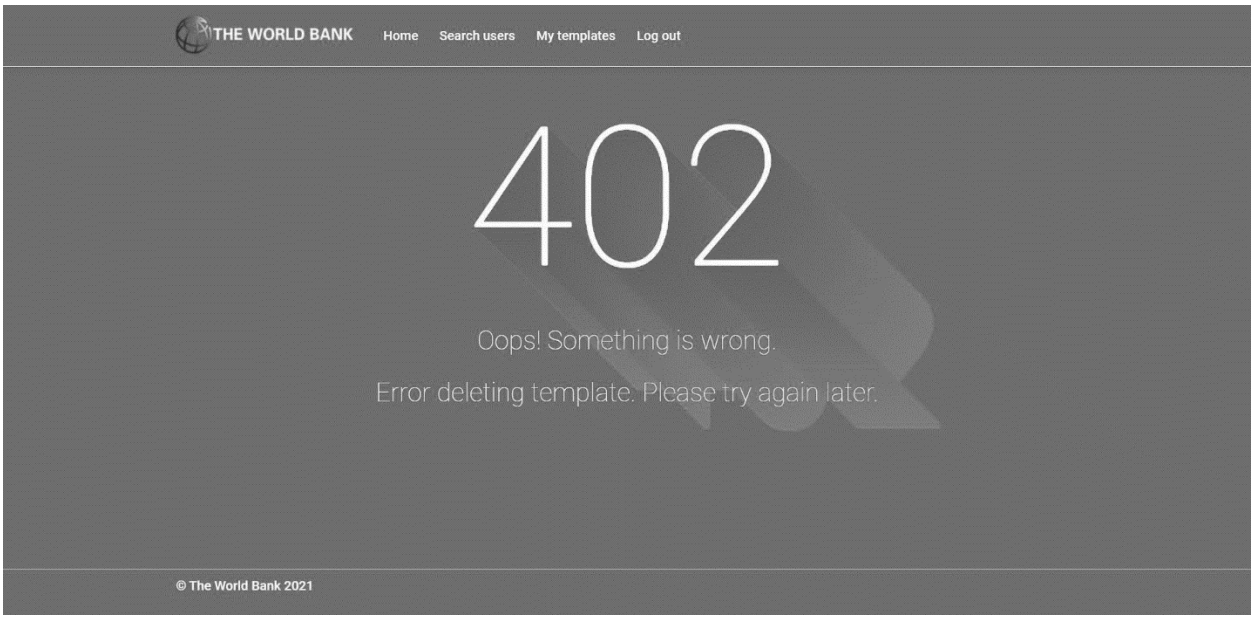
Слика 95 - Порука о неуспешној претрази шаблона трансакција

8.1 Уколико систем не може да учита изабран шаблон он приказује кориснику поруку “Систем не може да учита шаблон“. Прекида се извршење сценариа (ИА)



Слика 96 - Страница о грешци приликом учитавања шаблона трансакције

11.1 Уколико систем не може да обрише шаблон он приказује кориснику поруку “Систем не може да обрише шаблон”. (ИА)



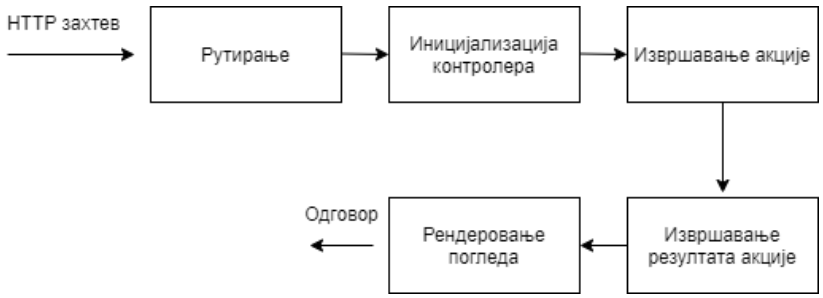
Слика 97 - Страница о грешци приликом брисања шаблона

6.2 Пројектовање апликационе логике

Под пројектовањем апликационе логике се подразумева пројектовање контролера апликационе логике, пројектовање пословне логике и пројектовање слоја приступа подацима.

6.2.1 Пројектовање контролера апликационе логике

Контролер прихвата захтеве који пристижу са погледа и потом позива одговарајуће операције слоја приступа подацима. Након што добије резултат, контролер податке попуњава у одговарајући модел и прослеђује заједно са погледом.



Слика 98 - Приказ животног циклуса MVC захтева

6.2.2 Пројектовање пословне логике

Пословна логика је описана структуром (доменским класама) и понашањем (системским операцијама). За сваки од уговора системских операција дефинисаних у фази анализе пројектује се концептуално решење (Влајић, 2015).

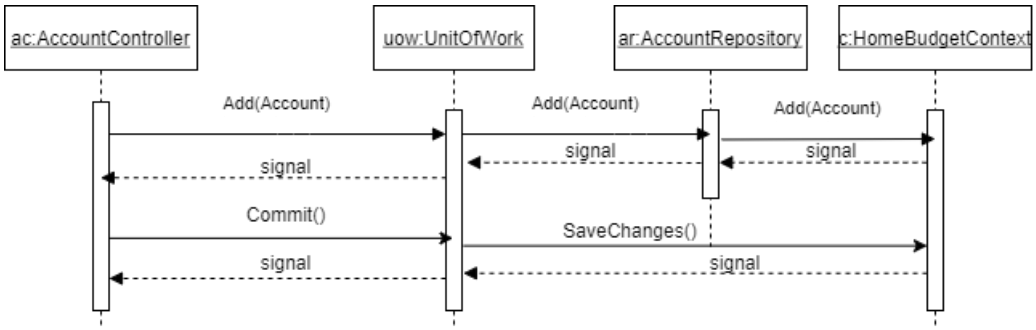
У наставку је за сваку идентификовану системску операцију (и одговарајући уговор) у фази анализе формиран одговарајући дијаграм секвенци који приказује комуникацију контролера апликационе логике и слоја приступа подацима.

Уговор УГ1: ZapamtiRačun(Račun):signal;

Веза са СК: СК1

Предуслови: Вредносна и структурна ограничења над објектом *Рачун* морају бити задовољена.

Постуслови: Подаци о рачуну су запамћени.



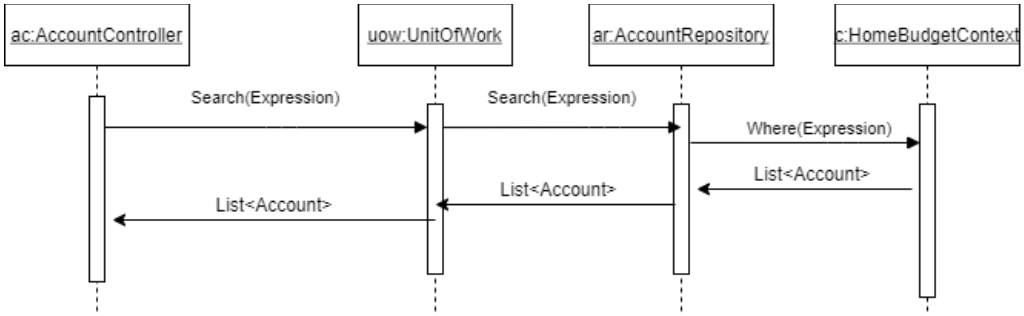
Слика 99 - Дијаграм секвенци - Уговор: ЗапамтиРачун

Уговор УГ2: UčitajListuRačuna(Lista<Račun>):signal;

Веза са СК: СК2, СК3

Предуслови:

Постуслови:



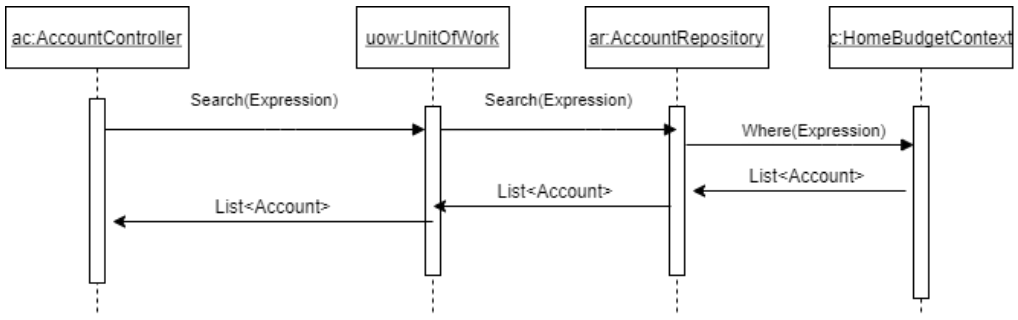
Слика 100 - Дијаграм секвенци - Уговор: УчитајЛистуРачуна

Уговор УГ3: NađiRačune(Račun, Lista<Račun>):signal;

Веза са СК: СК2, СК3

Предуслови:

Постуслови:



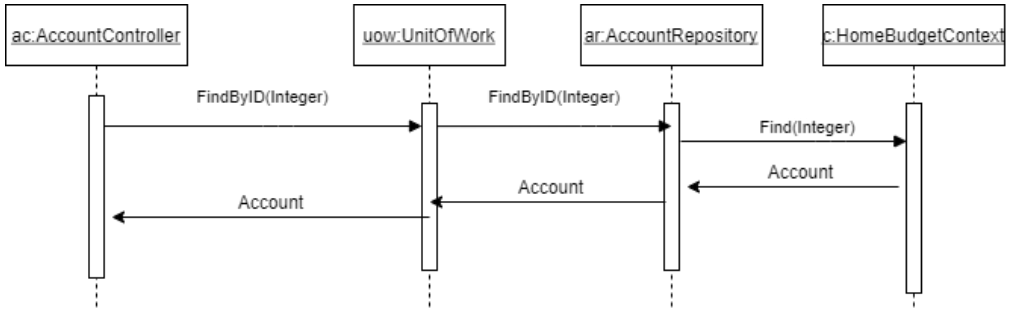
Слика 101 – Дијаграм секвенци - Уговор: НађиРачуне

Уговор УГ4: УčitajRačun(Račun):signal;

Веза са СК: СК2, СК3

Предуслови:

Постуслови:



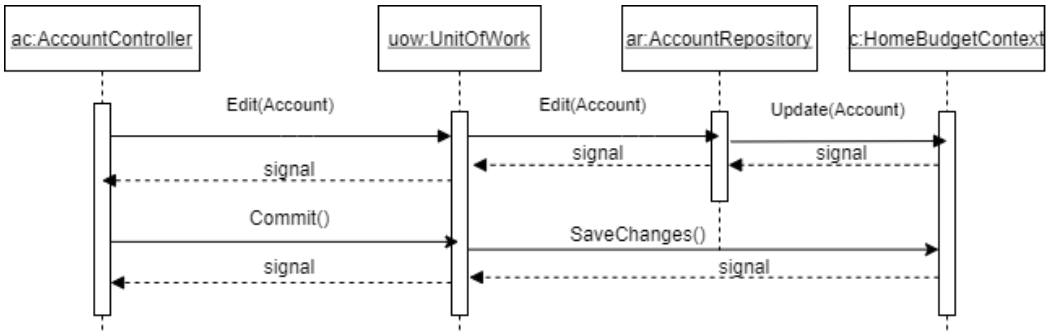
Слика 102 – Дијаграм секвенци - Уговор: УчитајРачун

Уговор УГ5: АžurirajRačun(Račun):signal;

Веза са СК: СК2

Предуслови: Вредносна и структурна ограничења над објектом *Рачун* морају бити задовољена.

Постуслови: Подаци о рачуну су запамћени.



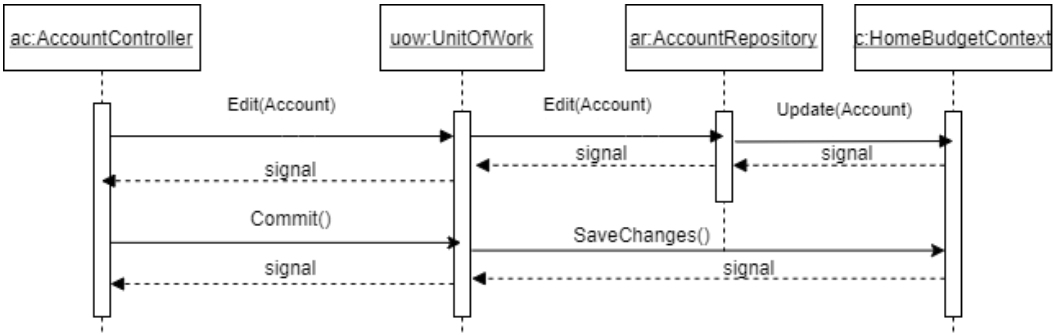
Слика 103 - Дијаграм секвенци - Уговор: АжурирајРачун

Уговор УГ6: ОбришиРачун(Račun):signal;

Веза са СК: СК3

Предуслови: Структурна ограничења над објектом *Рачун* морају бити задовољена.

Постуслови: Подаци о рачуну су обрисани.



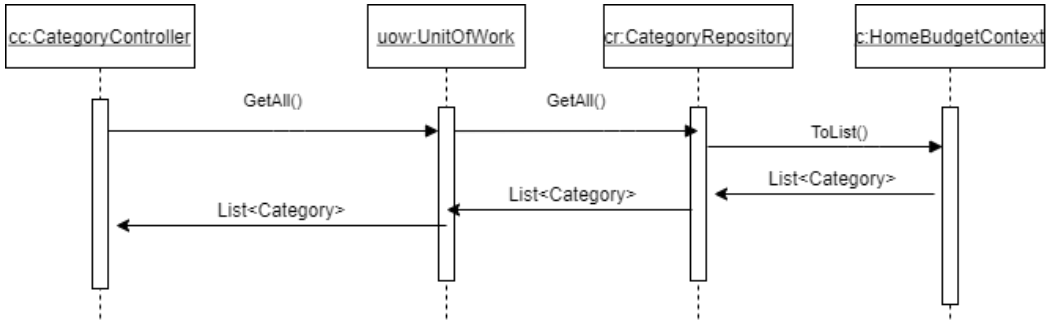
Слика 104 - Дијаграм секвенци - Уговор: ОбришиРачун

Уговор УГ7: UčitajListuKategorija(List<Kategorija>):signal;

Веза са СК: СК4, СК5, СК6, СК7, СК8, СК9

Предуслови:

Постуслови:



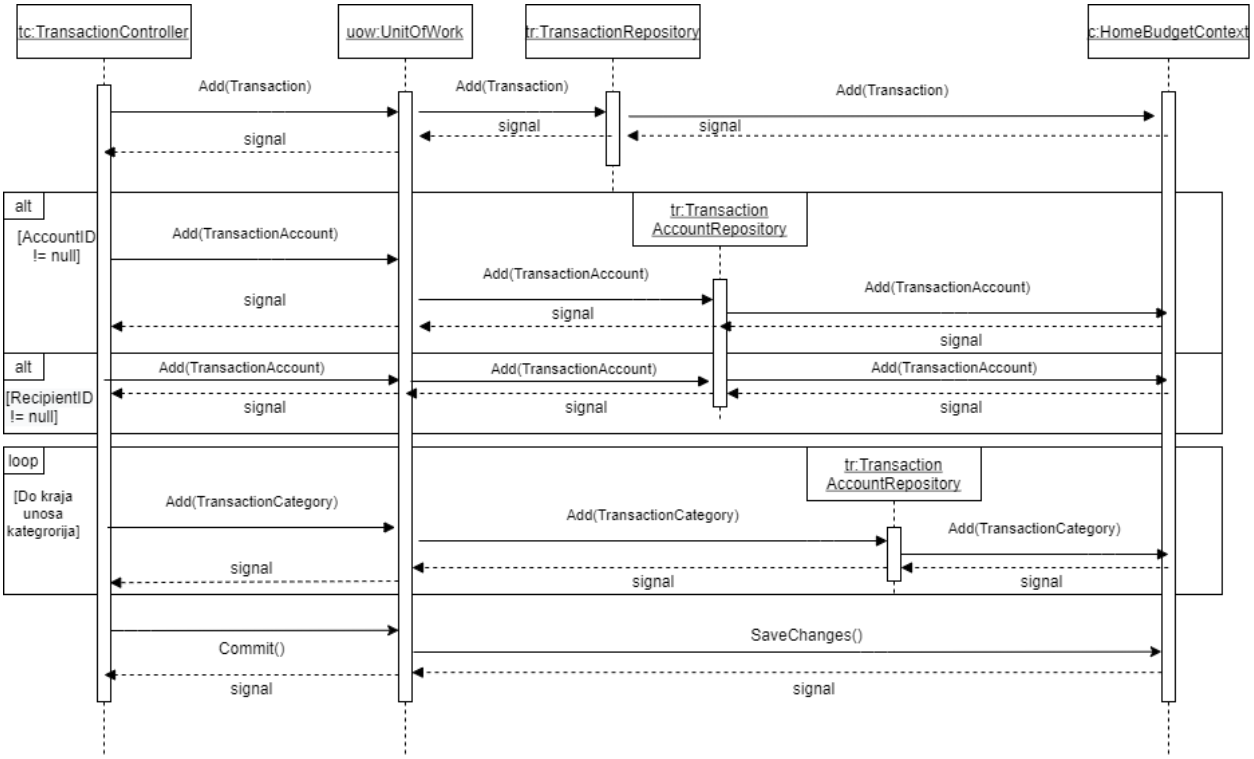
Слика 105 - Дијаграм секвенци – Уговор: УчитајЛистуКатегорија

Уговор УГ8: ZapamtiTransakciju(Transakcija):signal;

Веза са СК: СК4

Предуслови: Вредносна и структурна ограничења над објектом *Трансакција* морају бити задовољена.

Постуслови: Подаци о трансакцији су запамћени.



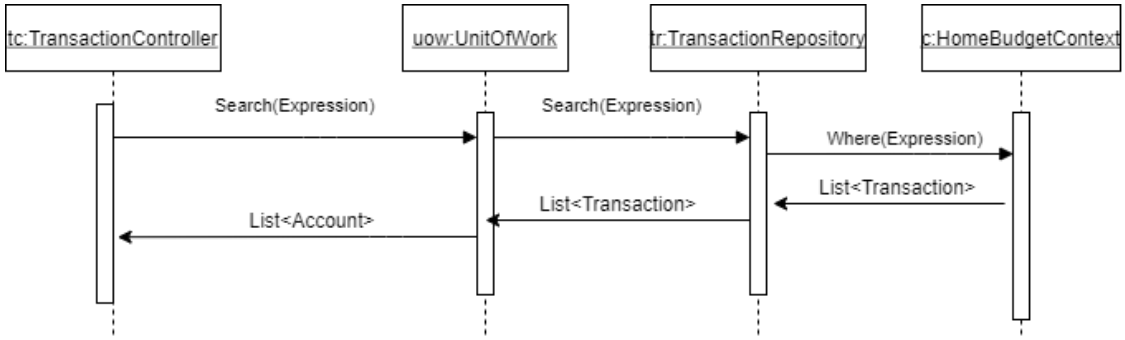
Слика 106 - Дијаграм секвенци - Уговор: ЗапамтиТрансакцију

Уговор УГ9: УčitajListuTransakcija (List<Transakcija>):signal;

Веза са СК: СК5, СК6, СК7

Предуслови:

Постуслови:



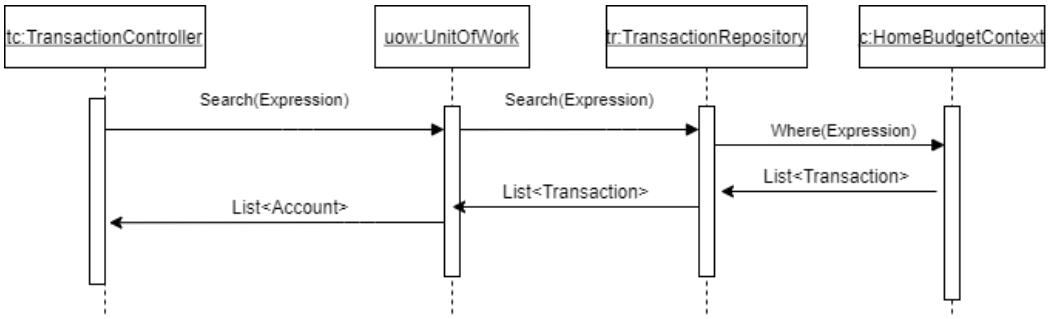
Слика 107 - Дијаграм секвенци - Уговор: УчитајЛистуТрансакција

Уговор УГ10: NađiTransakcije(Transakcija List<Transakcija>):signal;

Веза са СК: СК5, СК6, СК7

Предуслови:

Постуслови:



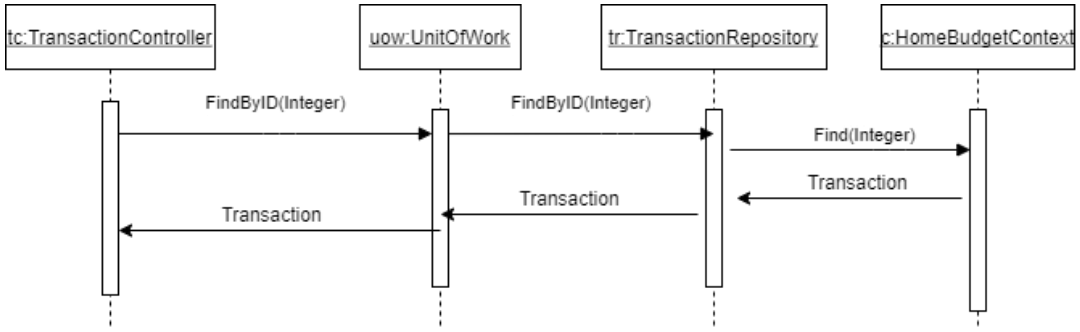
Слика 108 - Дијаграм секвенци - Уговор: НађиТрансакције

Уговор УГ11: UčitajTransakciju(Transakcija):signal;

Веза са СК: СК5, СК6, СК7

Предуслови:

Постуслови:



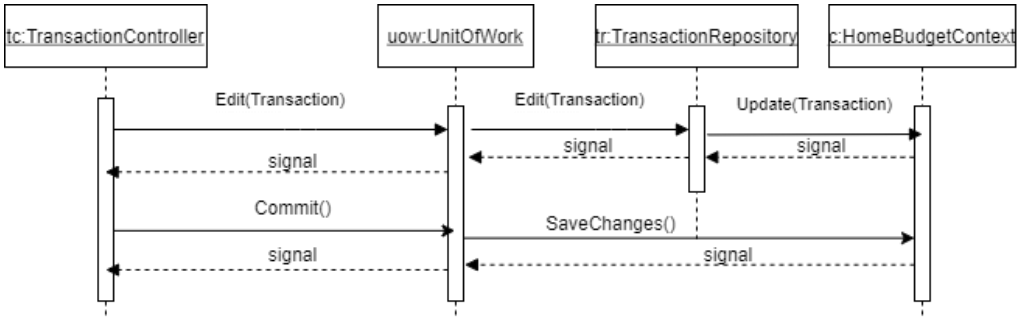
Слика 109 - Дијаграм секвенци - Уговор: УчитајТрансакцију

Уговор УГ12: АžurirajTransakciju(Transakcija):signal;

Веза са СК: СК6

Предуслови: Вредносна и структурна ограничења над објектом *Трансакција* морају бити задовољена.

Постуслови: Подаци о трансакцији су запамћени.



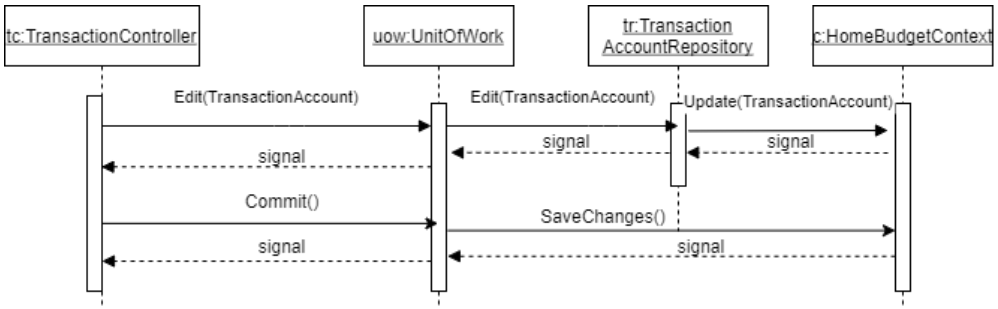
Слика 110 - Дијаграм секвенци - Уговор: АжурирајТранакцију

Уговор УГ13: ObrišiTransakciju(Transakcija):signal;

Веза са СК: СК7

Предуслови: Структурна ограничења над објектом *Трансакција* морају бити задовољена.

Постуслови: Подаци о трансакцији су обрисани.



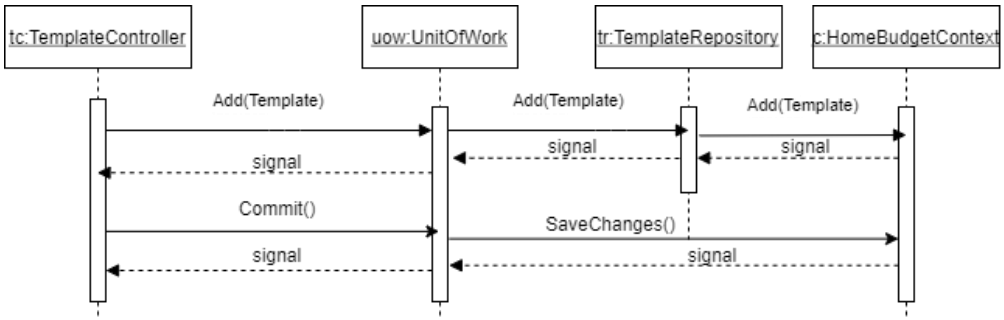
Слика 111 - Дијаграм секвенци - Уговор: ОбришиТрансакцију

Уговор УГ14: ZapamtiŠablon(Šablon):signal;

Веза са СК: СК8

Предуслови: Вредносна и структурна ограничења над објектом *Шаблон* морају бити задовољена.

Постуслови: Подаци о шаблону су запамћени.



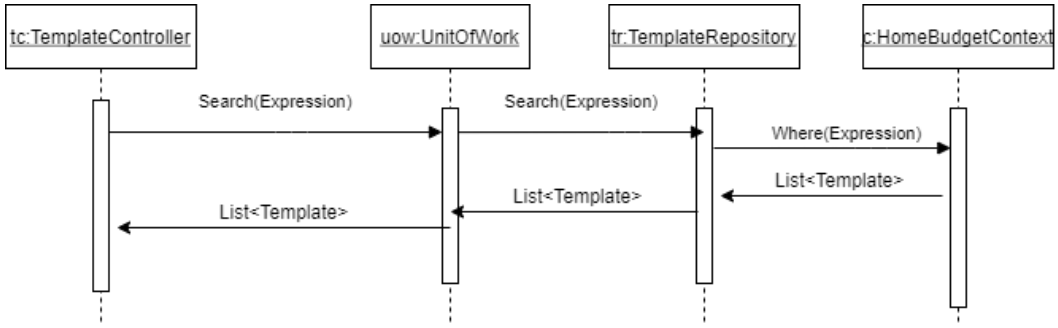
Слика 112 - Дијаграм секвенци - Уговор: ЗапамтиШаблон

Уговор УГ15: УčitajListuŠablona(List<Šablon>):signal;

Веза са СК: СК9, СК10

Предуслови:

Постуслови:



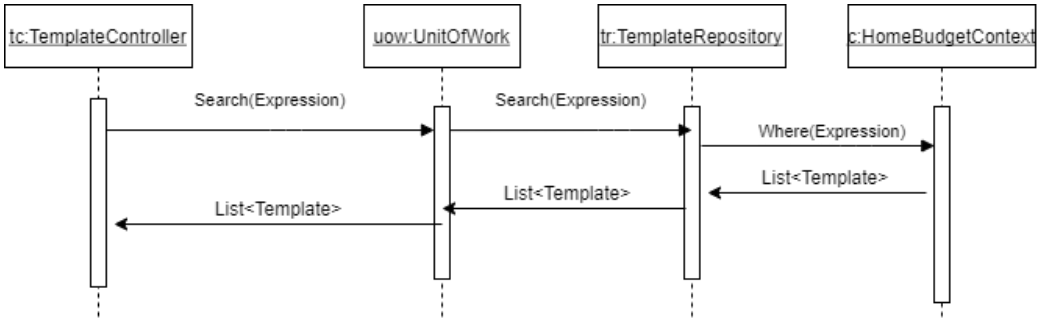
Слика 113 - Дијаграм секвенци - Уговор: УчитајЛистуШаблона

Уговор УГ16: NađiŠablone(Šablon, Lista<Šablon>):signal;

Веза са СК: СК9, СК10

Предуслови:

Постуслови:



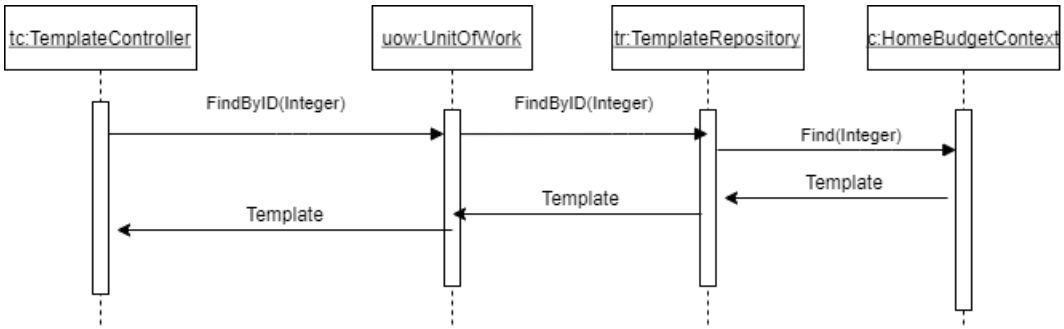
Слика 114 - Дијаграм секвенци - Уговор: НађиШаблоне

Уговор УГ17: УčitajŠablon(Šablon):signal;

Веза са СК: СК9, СК10

Предуслови:

Постуслови:



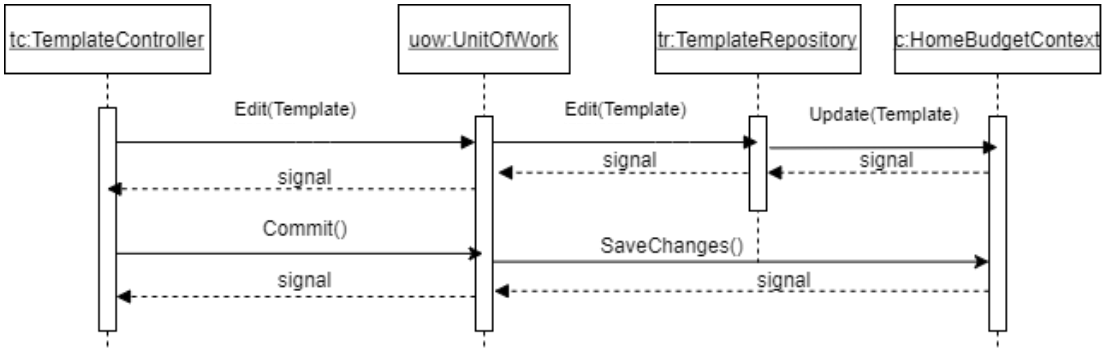
Слика 115 - Дијаграм секвенци - Уговор: УчитајШаблон

Уговор УГ18: АжурирајШablon(Šablon):signal;

Веза са СК: СК9

Предуслови: Вредносна и структурна ограничења над објектом *Шаблон* морају бити задовољена.

Постуслови: Подаци о шаблону су запамћени.



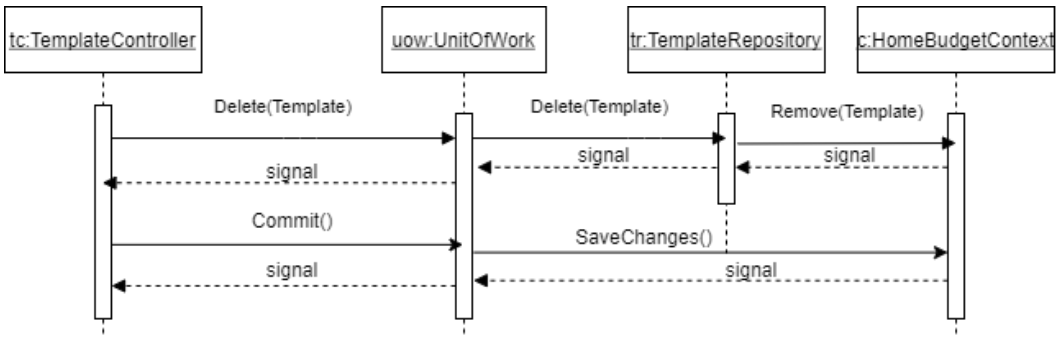
Слика 116 - Дијаграм секвенци - Уговор: АжурирајШаблон

Уговор УГ19: ОбришиШablon(Šablon):signal;

Веза са СК: СК10

Предуслови: Структурна ограничења над објектом *Шаблон* морају бити задовољена.

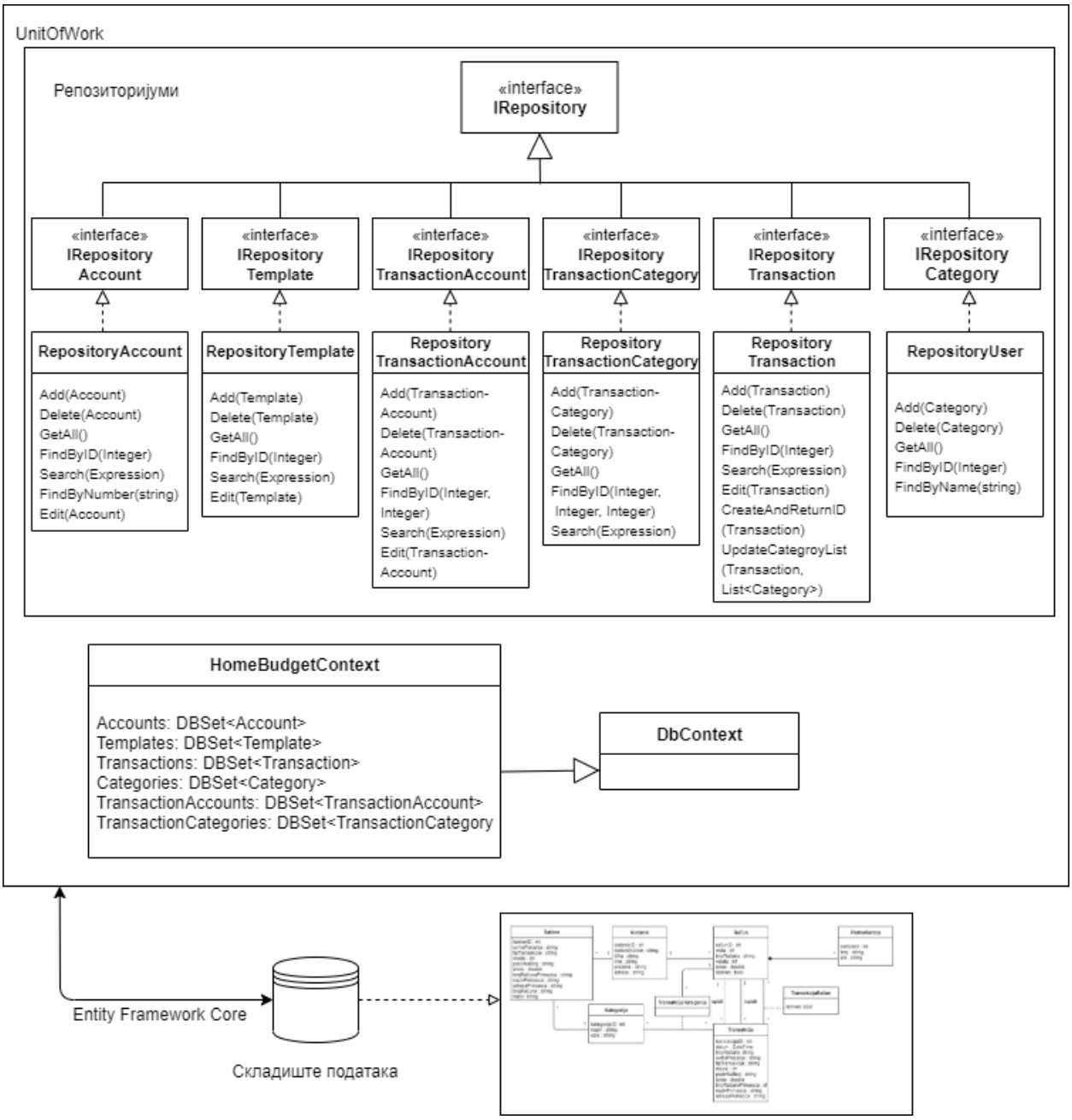
Постуслови: Подаци о шаблону су обрисани.



Слика 117 - Дијаграм секвенци - Уговор: ОбришиШаблон

6.2.3 Пројектовање слоја приступа подацима

Контролери шаљу захтеве до слоја приступа подацима позивајући одговарајуће методе Repository класа. Свака од Repository класа има приступ контексту који омогућава комуникацију са складиштем података. Како би се избегло стварање конфликта над базом података, реализацијом UnitOfWork патерна, формира се јединствена тачка приступа подацима и јединствена сесија над базом података.



Слика 118 - Пројектовање слоја приступа подацима

6.3 Пројектовање складишта података

На основу претходно креираног доменског модела, формиран је следећи релациони модел:

Korisnik(<u>KorisnikID</u>, KorisničkoIme, Ime, Prezime, Šifra, Adresa),

Račun(<u>RačunID</u>, Vrsta, BrojRačuna, Valuta, Iznos, Skriven, *KorisnikID*)

PlatnaKartica(*<u>RačunID</u>*, <u>KarticaID</u>, Broj, PIN)

Transakcija(TransakcijaID, Datum, BrojRačuna, SvrhaPlaćanja, TipTransakcije, Model, PozivNaBroj, Iznos, BrojRačunaPrimaoca, NazivPrimaoca, AdresaPrimaoca, *RačunID, RačunID'*)

TransakcijaRačun(*RačunID*, *TransakcijaID*, Skriven)

Kategorija(KategroijaID, Naziv, Opis)

TransakcijaKategrorija(*TransakcijaID*, *KategrorijaID*, *RačunID*)

Šablon(ŠablonID, SvrhaPlaćanja, TipTransakcije, Model, PozivNaBroj, Iznos, BrojRačunaPrimaoca, NazivPrimaoca, AdresaPrimaoca, BrojRačuna, Naziv, *KategorijaID, KorisnikID*)

Табела Корисник		Просто вредносно ограничење		Сложено вредносно ограничење		Структурно ограничење
Атрибути	Име	Тип атрибута	Вредност атрибута	Међузависност атрибута једне табеле	Међузависност атрибута више табела	INSERT UPDATE CASCADES Račun, Šablon DELETE RESTRICED Račun, Šablon
	KorisnikID	Integer	Not null and > 0			
	Ime	String	Not null			
	Prezime	String	Not null			
	KorisničkoIme	String	Not null			
	Šifra	String	Not null			
	Adresa	String	Not null			

Табела 1 – Корисник

Табела Рачун		Просто вредносно ограничење		Сложено вредносно ограничење		Структурно ограничење
Атрибути	Име	Тип атрибута	Вредност атрибута	Међузависност атрибута једне табеле	Међузависност атрибута више табела	INSERT RESTRICTED Korisik UPDATE RESTRICTED Korisnik UPDATE CASCADES Transakcija PlatnaKartica, TransakcijaKategroija TransakcijaRačun DELETE RESTRICED Transakcija PlatnaKartica, TransakcijaKategroija TransakcijaRačun
	RačunID	Integer	Not null and > 0			
	Vrsta	Integer	Not null			
	BrojRačuna	Integer	Not null			
	Valuta	Integer	Not null			
	Iznos	Integer	Not null			
	Skriven	Boolean	Not null			
	KorisnikID	Integer	Not null and > 0			

Табела 2 – Рачун

Табела Платна картица		Просто вредносно ограничење		Сложено вредносно ограничење		Структурно ограничење
Атрибути	Име	Тип атрибута	Вредност атрибута	Међузависност атрибута једне табеле	Међузависност атрибута више табела	INSERT RESTRICTED Račun UPDATE RESTRICTED Račun DELETE
	RačunID	Integer	Not null and > 0			
	KarticaID	Integer	Not null and > 0			
	Broj	String	Not null			
	PIN	Integer	Not null			

Табела 3 - Платна картица

Табела Трансакција		Просто вредносно ограничење		Сложено вредносно ограничење		Структурно ограничење
Атрибути	Име	Тип атрибута	Вредност атрибута	Међузависност атрибута једне табеле	Међузависност атрибута више табела	INSERT RESTRICTED Račun UPDATE RESTRICTED Račun UPDATE CASCADES Transakcija-Kategroija, Transackija-Račun DELETE RESTRICED Transakcija-Kategroija, Transackija-Račun
	TransakcijaID	Integer	Not null and > 0			
	Datum	DateTime	Not null			
	BrojRačuna	String	Not null			
	SvrhaPlaćanja	String	Not null			
	TipTransakcije	String	Not null			
	Model	Integer	Not null			
	PozivNaBroj	String	Not null			
	Iznos	Integer	Not null			
	BrojRačuna-Primaoca	String	Not null			
	NazivPrimaoca	String	Not null			
	AdresaPrimaoca	String	Not null			
	RačunID	Integer	Not null and > 0			
	RačunID'	Integer	Not null			

Табела 4 - Трансакција

Табела ТрансакцијаРачун		Просто вредносно ограничење		Сложено вредносно ограничење		Структурно ограничење
Атрибути	Име	Тип атрибута	Вредност атрибута	Међузависност атрибута једне табеле	Међузависност атрибута више табела	INSERT RESTRICTED Račun, Transakcija UPDATE RESTRICTED Račun, Transakcija DELETE
	RačunID	Integer	Not null and > 0			
	TransakcijaID	Integer	Not null and > 0			
	Skriven	Boolean	Not null			

Табела 5 – ТрансакцијаРачун

Табела Категорија		Просто вредносно ограничење		Сложено вредносно ограничење		Структурно ограничење
Атрибути	Име	Тип атрибута	Вредност атрибута	Међузависност атрибута једне табеле	Међузависност атрибута више табела	INSERT UPDATE CASCADES Transakcija-Kategroija, Šablon DELETE RESTRICED Transakcija-Kategroija, Šablon
	KategorijaID	Integer	Not null and > 0			
	Naziv	String	Not null			
	Opis	String	Not null			

Табела 6 - Категорија

Табела ТрансакцијаКатегорија		Просто вредносно ограничење		Сложено вредносно ограничење		Структурно ограничење
Атрибути	Име	Тип атрибута	Вредност атрибута	Међузависност атрибута једне табеле	Међузависност атрибута више табела	INSERT RESTRICTED Transakcija, Račun, Kategroija UPDATE RESTRICTED Transakcija, Račun, Kategroija DELETE
	TransakcijaID	Integer	Not null and > 0			
	KategroijaID	String	Not null and > 0			
	RačunID	String	Not null and > 0			

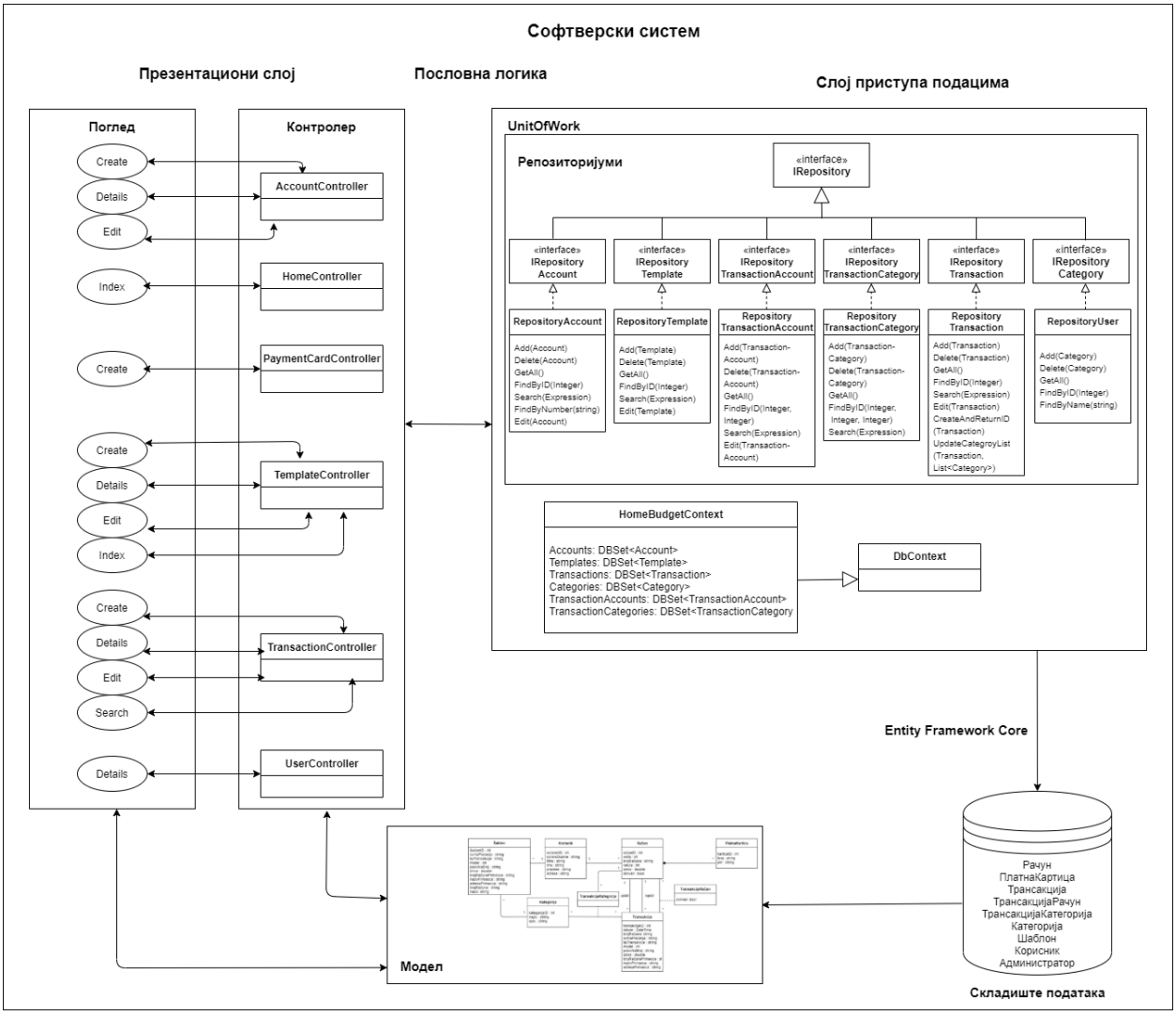
Табела 7 - ТрансакцијаКатегорија

Табела Шаблон		Просто вредносно ограничење		Сложено вредносно ограничење		Структурно ограничење
Атрибути	Име	Тип атрибута	Вредност атрибута	Међузависност атрибута једне табеле	Међузависност атрибута више табела	INSERT RESTRICTED Korisnik, Kategorija UPDATE RESTRICTED Korisnik, Kategorija DELETE
	ŠablonID	Integer	Not null and > 0			
	SvrhaPlaćanja	String	Not null			
	TipTransakcije	String	Not null			
	Model	Integer	Not null			
	PozivNaBroj	String	Not null			
	Iznos	Integer	Not null			
	BrojRačuna-Primaoca	String	Not null			
	NazivPrimaoca	String	Not null			
	AdresaPrimaoca	String	Not null			
	BrojRačuna	String	Not null			
	Naziv	String	Not null			
	KategorijaID	Integer	Not null and > 0			
	KorisnikID	Integer	Not null and > 0			

Табела 8 – Шаблон

6.4 Коначан изглед архитектуре софтверског система

На основу претходно пројектованих целина, добијена је коначна архитектура софтверског система као резултат фазе пројектовања.



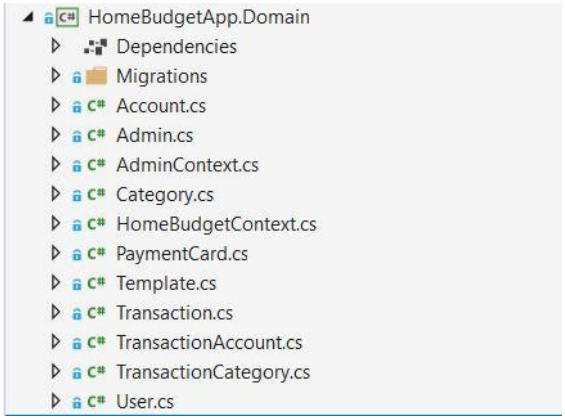
Слика 119 - Коначна архитектура софтверског система

7. Имплементација

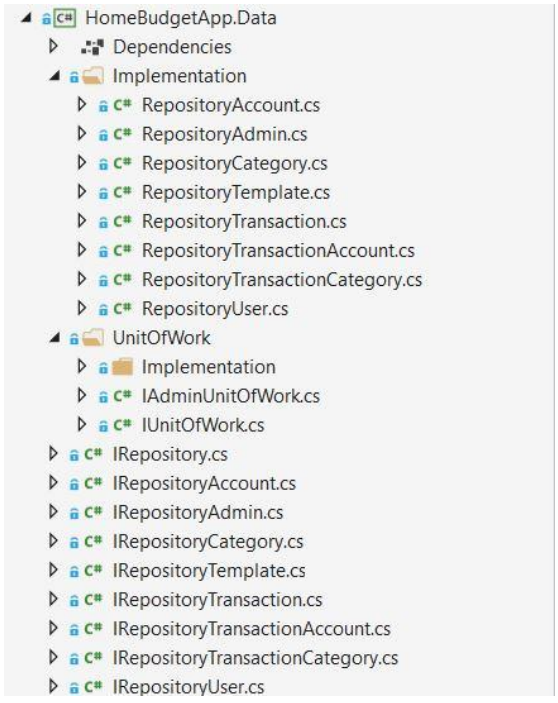
Креирана веб апликација је развијена у програмском језику C#, употребом ASP.NET Core и MVC оквира. Коришћено развојно окружење Microsoft Visual Studio 2019 и апликација се извршава на IIS серверу. Систем за управљање базом података је Microsoft SQL Server, док је кориснички интерфејс представљен Razor страницама.

7.1 Структура софтверског решења

Имплементиране су следеће класе:



Слика 120 – Библиотека класа Domain



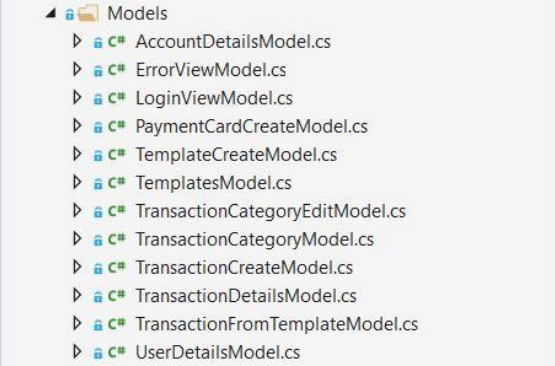
Слика 121 – Библиотека класа Data

Слика 122 - Пројекат WebApp

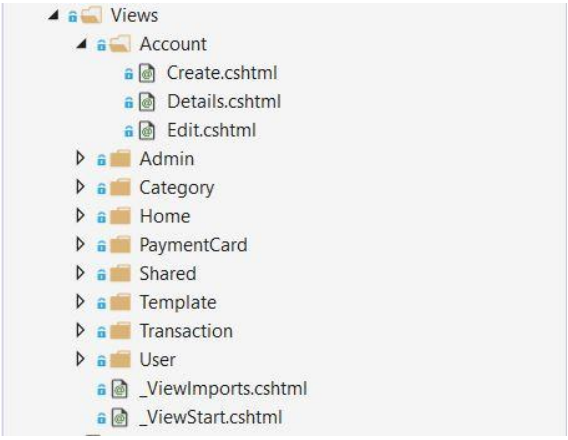
Слика 123 - Фолдер wwwroot

Слика 124 - Фолдер Controllers

Слика 125 - Фолдер Filters



Слика 126 - Фолдер Models



Слика 127 - Фолдер Views

7.2 Имплементација апликационе логике

У наставку је описана имплементација апликационе логике, која прима захтев од корисничког интерфејса, обрађује га користећи податке из складишта података и враћа одговор до корисника.

7.2.1 Имплементација комуникације са корисницима

Почетна тачка од које креће извршавање апликације је класа Program.cs која имплементира Main методу.

Ова метода позива CreateHostBuilder методу, која се од верзије 3.0 ASP.NET Core-а састоји из два корака, односно позива две методе - Host.CreateDefaultBuilder() која подешава основне концепте апликације као што су конфигурација апликације, logging и dependency injection container и IHost.ConfigureWebHostDefaults() која обезбеђује конфигурацију специфичну за апликацију, помоћу Startup.cs класе [2].

Имплементација Program.cs класе приказана је у наставку.

```
public class Program

    {

        public static void Main(string[] args)
        {
            CreateHostBuilder(args).Build().Run();
        }

        public static IHostBuilder CreateHostBuilder(string[] args) =>
            Host.CreateDefaultBuilder(args)
                .ConfigureWebHostDefaults(webBuilder =>
                {
                    webBuilder.UseStartup<Startup>();
                });
}
```

Унутар класе Startup.cs су имплементиране две јавне методе – Configure и ConfigureServices. Обе методе позива ASP.NET Core runtime приликом покретања апликације.

ConfigureServices метода конфигурише сервисе апликације. Сервис је компонента која се може користити више пута и обезбеђује функционалност апликације. Сервиси се региструју у оквиру ConfigureServices методе и могу се користити било где у оквиру апликације помоћу dependency injection-a или ApplicationServices својства. Метода Configure извршава конфигурацију pipeline-а извршавања HTTP захтева у апликацији [3].

Приликом имплементирања апликације, у оквиру методе ConfigureServices регистровани су контролери, погледи, филтери, параметри сесије, као и класе које омогућавају приступ подацима. Унутар методе Configure додато је коришћење сесије у pipeline. Након измена, Startup.cs класа изгледа:

```
public class Startup
{

    public void ConfigureServices(IServiceCollection services)
    {
     services.AddDistributedMemoryCache();
     services.AddSession(opt => opt.IdleTimeout = TimeSpan.FromMinutes(10));
     services.AddControllersWithViews();
     services.AddScoped<IUnitOfWork, HomeBudgetUnitOfWork>();
     services.AddScoped<IAdminUnitOfWork, AdminUnitOfWork>();
     services.AddScoped<LoggedInUser>();
     services.AddScoped<NotLoggedIn>();
     services.AddScoped<AdminLoggedIn>();
     services.AddScoped<AdminNotLoggedIn>();
     services.AddDbContext<HomeBudgetContext>();
     services.AddDbContext<AdminContext>();
    }
    public void Configure(IApplicationBuilder app, IWebHostEnvironment env)
    {
      if (env.IsDevelopment())
      {
        app.UseDeveloperExceptionPage();
      }
```

```
     else
     {
          app.UseExceptionHandler("/Home/Error");
          app.UseHsts();
       }
       app.UseHttpsRedirection();
       app.UseStaticFiles();
       app.UseRouting();
       app.UseSession();
       app.UseAuthorization();

       app.UseEndpoints(endpoints =>
       {
          endpoints.MapControllerRoute(
          name: "default",
          pattern: "{controller=Home}/{action=Index}/{id?}");
       });
    }
}
```

7.2.2 Имплементација пословне логике

Пословна логика је описана структуром коју чине доменске класе и понашањем које се састоји из системских операција (Влајић, 2015).

Имплементација доменских класа

На основу дефинисаних концептуалних класа у доменском моделу и њихових релација, пишу се одговарајуће класе у програмском језику C#. Свака класа се састоји из приватних атрибута и јавних метода које омогућавају њихов приступ и измену – за сваки атрибут је креиран Property. Доменске класе се налазе у оквиру библиотеке класа Domain. У наставку је дат пример једне доменске класе – класе Шаблон.

```
public class Template
    {
        public int TemplateID { get; set; }
        public int UserID { get; set; }
        public User User { get; set; }
        public string RecipientAccountNumber { get; set; }
        public string RecipientName { get; set; }
        public string RecipientAddress { get; set; }
        public string Purpose { get; set; }
        public int Model { get; set; }
        public string ReferenceNumber { get; set; }
        public double Amount { get; set; }
        public string AccountNumber { get; set; }
        public string Name { get; set; }
        public int CategoryID { get; set; }
        public Category Category { get; set; }
        public string Type { get; set; }
    }
}
```

Имплементација системских операција

Системске операције имплементиране су у оквиру одговарајућих Repository класа унутар библиотеке класа Data.

Уговор УГ1: ZapamtiRačun(Račun):signal;

Веза са СК: СК1

Предуслови: Вредносна и структурна ограничења над објектом *Рачун* морају бити задовољена.

Постуслови: Подаци о рачуну су запамћени.

```
public void Add(Account account)
{
    context.Accounts.Add(account);
}
```

Уговор УГ2: UčitajListuRačuna(Lista<Račun>):signal;

Веза са СК: СК2, СК3

Предуслови:

Постуслови:

```
public List<Account> Search(Expression<Func<Account, bool>> pred)
{
    return context.Accounts.Where(pred).ToList();
}
```

Уговор УГ3: NađiRačune(Račun, Lista<Račun>):signal;

Веза са СК: СК2, СК3

Предуслови:

Постуслови:

```
public List<Account> Search(Expression<Func<Account, bool>> pred)
{
    return context.Accounts.Where(pred).ToList();
}
```

Уговор УГ4: UčitajRačun(Račun):signal;

Веза са СК: СК2, СК3

Предуслови:

Постуслови:

```
public Account FindByID(int id, params int[] ids)
{
    return context.Accounts.Find(id);
}
```

Уговор УГ5: АжурирајРачун(Рачун):signal;

Веза са СК: СК2

Предуслови: Вредносна и структурна ограничења над објектом *Рачун* морају бити задовољена.

Постуслови: Подаци о рачуну су запамћени.

```
public void Edit(Account account)
{
    context.Accounts.Update(account);
}
```

Уговор УГ6: ОбришиРачун(Рачун):signal;

Веза са СК: СК3

Предуслови: Структурна ограничења над објектом *Рачун* морају бити задовољена.

Постуслови: Подаци о рачуну су обрисани.

```
public void Edit(Account account)
{
    context.Accounts.Update(account);
}
```

Уговор УГ7: УчитајЛистуКатегорија(List<Kategorija>):signal;

Веза са СК: СК4, СК5, СК6, СК7, СК8, СК9

Предуслови:

Постуслови:

```
public List<Category> GetAll()
{
    return context.Categories.ToList();
}
```

Уговор УГ8: ЗапамтиТрансакцију(Трансакција):signal;

Веза са СК: СК4

Предуслови: Вредносна и структурна ограничења над објектом *Трансакција* морају бити задовољена.

Постуслови: Подаци о трансакцији су запамћени.

```
public void Add(Transaction transaction)
{
    context.Transactions.Add(transaction);
}
public void Add(TransactionAccount param)
{
    context.TransactionAccounts.Add(param);
}
public void Add(TransactionCategory param)
{
    context.TransactionCategories.Add(param);
}
```

Уговор УГ9: UčitajListuTransakcija (List<Transakcija>):signal;

Веза са СК: СК5, СК6, СК7

Предуслови:

Постуслови:

```
public List<Transaction> Search(Expression<Func<Transaction, bool>> pred)
{
    return context.Transactions.Where(pred).ToList();
}
```

Уговор УГ10: NađiTransakcije(Transakcija List<Transakcija>):signal;

Веза са СК: СК5, СК6, СК7

Предуслови:

Постуслови:

```
public List<Transaction> Search(Expression<Func<Transaction, bool>> pred)
{
    return context.Transactions.Where(pred).ToList();
}
```

Уговор УГ11: UčitajTransakciju(Transakcija):signal;

Веза са СК: СК5, СК6, СК7

Предуслови:

Постуслови:

```
public Transaction FindByID(int id, params int[] ids)
{
    return context.Transactions.Find(id);
}
```

Уговор УГ12: АжурирајTransakciju(Transakcija):signal;

Веза са СК: СК6

Предуслови: Вредносна и структурна ограничења над објектом *Трансакција* морају бити задовољена.

Постуслови: Подаци о трансакцији су запамћени.

```
public void Edit(Transaction transaction)
{
    context.Transactions.Update(transaction);
}
```

Уговор УГ13: ОбришиTransakciju(Transakcija):signal;

Веза са СК: СК7

Предуслови: Структурна ограничења над објектом *Трансакција* морају бити задовољена.

Постуслови: Подаци о трансакцији су обрисани.

```
public void Edit(Transaction transaction)
{
    context.Transactions.Update(transaction);
}
```

Уговор УГ14: ZapamtiŠablon(Šablon):signal;

Веза са СК: СК8

Предуслови: Вредносна и структурна ограничења над објектом *Шаблон* морају бити задовољена.

Постуслови: Подаци о шаблону су запамћени.

```
public void Add(Template template)
{
    context.Templates.Add(template);
}
```

Уговор УГ15: УчитајListuŠablona(List<Šablon>):signal;

Веза са СК: СК9, СК10

Предуслови:

Постуслови:

```
public List<Template> Search(Expression<Func<Template, bool>> pred)
{
    return context.Templates.Where(pred).AsNoTracking().ToList();
}
```

Уговор УГ16: НађиШаблоне(Šablon, Lista<Šablon>):signal;

Веза са СК: СК9, СК10

Предуслови:

Постуслови:

```
public List<Template> Search(Expression<Func<Template, bool>> pred)
{
    return context.Templates.Where(pred).AsNoTracking().ToList();
}
```

Уговор УГ17: УчитајШаблон(Šablon):signal;

Веза са СК: СК9, СК10

Предуслови:

Постуслови:

```
public Template FindByID(int id, params int[] ids)
{
    return context.Templates.Find(id);
}
```

Уговор УГ18: АжурирајШаблон(Šablon):signal;

Веза са СК: СК9

Предуслови: Вредносна и структурна ограничења над објектом *Шаблон* морају бити задовољена.

Постуслови: Подаци о шаблону су запамћени.

```
public void Edit(Template template)
{
    context.Templates.Update(template);
}
```

Уговор УГ19: ОбришиШаблон(Šablon):signal;

Веза са СК: СК10

Предуслови: Структурна ограничења над објектом *Шаблон* морају бити задовољена.

Постуслови: Подаци о шаблону су обрисани.

```
public void Delete(Template template)
{
    context.Templates.Remove(template);
}
```

7.3 Имплементација слоја приступа подацима

Слој приступа подацима имплементиран је помоћу објектно-релационог мапера Entity Framework Core-а и применом Unit of Work i Repository патерна.

7.3.1 Класа Context

У сврху објектно-релационог мапирања коришћен је Entity Framework Core. Он омогућава приступ подацима помоћу модела и context објекта. Модел је сачињен од доменских класа, док context представља сесију на базом података и омогућава упите над базом и чување података [4].

Током имплементације, најпре су креиране доменске класе које су укључивале одговарајуће спољне кључеве који су дефинисани у фази анализе. Додатке релације, које Entity Framework Core није могао да закључи на основу доменских класа, дефинисане су у оквиру OnModelCreating методе у context класи.

Context класа као атрибуте име колекције – DbSet-ове који представљају будуће табеле у релационој бази података. У оквиру исте класе могуће је иницијално испунити базу података одређеним подацима (тзв. seeding), што се најчешће користи за базе података које су намењене тестирању. Креирањем и покретањем одговарајућих миграционих фајлова, добијене су табеле релационе базе података. Имплементирана Context класа је приказана у наставку:

```
public class HomeBudgetContext : DbContext

{
  public DbSet<User> Users { get; set; }
  public DbSet<Account> Accounts { get; set; }
  public DbSet<Category> Categories { get; set; }
  public DbSet<Transaction> Transactions { get; set; }
  public DbSet<TransactionCategory> TransactionCategories { get; set; }
  public DbSet<Template> Templates { get; set; }
  public DbSet<TransactionAccount> TransactionAccounts { get; set; }

  public static readonly ILoggerFactory MyLoggerFactory = LoggerFactory.Create(builder =>
  { builder.AddConsole(); });

  protected override void OnConfiguring(DbContextOptionsBuilder optionsBuilder)
  {
    optionsBuilder
      .UseLoggerFactory(MyLoggerFactory)
      .EnableSensitiveDataLogging()
      .UseSqlServer(@"Server = (localdb)\mssqllocaldb; Database = HomeBudget; ");
  }
  protected override void OnModelCreating(ModelBuilder modelBuilder)
  {
    modelBuilder.Entity<Account>().OwnsMany(a => a.PaymentCards);
    modelBuilder.Entity<Transaction>(t =>
       {
          t.HasOne(t => t.Account).WithMany(a => a.TransactionsTo).OnDelete
(DeleteBehavior.Restrict);
```

```
          t.HasOne(t => t.Recipient).WithMany(a => a.TransactionsFrom).OnDelete
(DeleteBehavior.Restrict);
        });

     modelBuilder.Entity<TransactionCategory>(c =>
       {
           c.HasKey(c => new { c.CategoryID, c.TransactionID, c.OwnerID });
           c.HasOne(b => b.Transaction).WithMany(p => p.Categories).OnDelete
(DeleteBehavior.Restrict);
        });

        modelBuilder.Entity<TransactionAccount>(c =>
         {
            c.HasKey(c => new { c.AccountID, c.TransactionID });
            c.HasOne(b => b.Transaction).WithMany(p => p.Accounts).OnDelete
(DeleteBehavior.Restrict);
         });

         modelBuilder.Entity<Transaction>().OwnsOne(t => t.PaymentCard);
         Seed(modelBuilder);
   }
   private void Seed(ModelBuilder modelBuilder)
   {
       modelBuilder.Entity<User>().HasData(
       new User { UserID = 1, Name = "Mark", Surname = "Ronson", Username = "markronson",
       Password = "1234" },
       new User { UserID = 2, Name = "John", Surname = "Parker", Username = "johnparker",
       Password = "1111" });
    }
}
```

7.3.2 Unit of Work и Repository патерни

Примена Repository патерна подразумева креирање класе за сваки ентитет која ће садржати имплементиране методе могућих операција над тим ентитетом (најчешће CRUD операције). Свака од Repository класа има атрибут који је референца на Context класу и која представља сесију над базом података.

Коришћењем Repository патерна постиже се:

- Централизована пословна логика за сваки ентитет
- Олакшава се тестирање операција
- Флексибилна архитектура коју је лако прилагодити изменама [5].

Слика 128 - Примена Repository патерна

Repository патерн је имплементиран кроз Irepository интерфејс, интерфејс за сваки од ентитета (нпр. IRepositoryTemplate) који додају методе за имплементацију карактеристичне за ентитет и класе које имплементирају интерфејсе (нпр. RepositoryTemplate).

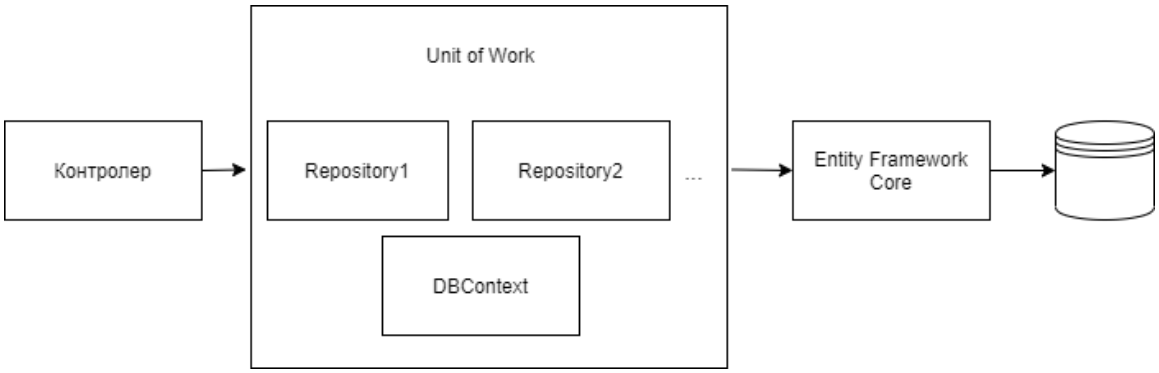
```
public interface IRepository<T>

{
    void Add(T param);
    List<T> GetAll();
    T FindByID(int id, params int[] ids);
    void Delete(T param);
}

public interface IRepositoryTemplate : IRepository<Template>
{
    public List<Template> Search(Expression<Func<Template, bool>> pred);
    public void Edit(Template template);
}

public class RepositoryTemplate : IRepositoryTemplate
{
    private readonly HomeBudgetContext context;
    public RepositoryTemplate(HomeBudgetContext context)
    {
        this.context = context;
    }
    public void Add(Template template)
    {
        context.Templates.Add(template);
    }

    public void Delete(Template template)
    {
        context.Templates.Remove(template);
    }

    public Template FindByID(int id, params int[] ids)
    {
        return context.Templates.Find(id);
    }

    public List<Template> GetAll()
    {
        return context.Templates.ToList();
    }

    public List<Template> Search(Expression<Func<Template, bool>> pred)
    {
        return context.Templates.Where(pred).AsNoTracking().ToList();
    }
    public void Edit(Template template)
    {
        context.Templates.Update(template);
    }
}
```

Заједно са Repository патерном, Unit of Work патерн ствара апстрактни слој између пословне логике и слоја приступа подацима [6].

Unit of Work патерн служи за груписање више операција у једну трансакцију. Такође, применом овог патерна се обезбеђује да више Repository класа које извршавају операције на базом

података деле референцу на заједнички Context објекат. На овај начин се смањује број конекција са базом података и укупан број трансакција [7].

У имплементацији овог патерна, креира се класа која ће садржати Commit методу и атрибуте свих Repository класа које позивају методе над заједничким Context објектом. На овај начин се унутар контролера ствара јединствена тачка приступа подацима.



Слика 129 - Примена Unit of Work и Repository патерна

```
public interface IUnitOfWork : IDisposable
{
   public IRepositoryUser User { get; set; }
   public IRepositoryAccount Account { get; set; }
   public IRepositoryTransaction Transaction { get; set; }
   public IRepositoryCategory Category { get; set; }
   public IRepositoryTransactionCategory TransactionCategory { get; set; }
   public IRepositoryTransactionAccount TransactionAccount { get; set; }
   public IRepositoryTemplate Template { get; set; }
   public void Commit();
}

public class HomeBudgetUnitOfWork : IUnitOfWork
{
   private readonly HomeBudgetContext context;

   public HomeBudgetUnitOfWork(HomeBudgetContext context)
   {
      this.context = context;
      User = new RepositoryUser(context);
      Account = new RepositoryAccount(context);
      Transaction = new RepositoryTransaction(context);
      Category = new RepositoryCategory(context);
      TransactionCategory = new RepositoryTransactionCategory(context);
      Template = new RepositoryTemplate(context);
      TransactionAccount = new RepositoryTransactionAccount(context);
    }

    public IRepositoryUser User { get; set; }
    public IRepositoryAccount Account { get; set; }
    public IRepositoryTransaction Transaction { get; set; }
    public IRepositoryCategory Category { get; set; }
    public IRepositoryTransactionCategory TransactionCategory { get; set; }
    public IRepositoryTransactionAccount TransactionAccount { get; set; }
    public IRepositoryTemplate Template { get; set; }
```

```
    public void Commit()
    {
      context.SaveChanges();
    }

    public void Dispose()
    {
      context.Dispose();
    }
}
```

7.4 Имплементација складишта података

На основу пројектованих доменских класа и релација између њих, идентификоване су следеће табеле релационе базе података:

	Name	Data Type	Allow Nulls
🔑	AccountID	int	☐
	Currency	int	☐
	AccountType	int	☐
	Number	nvarchar(MAX)	☐
	Amount	float	☐
	TotalIncome	float	☐
	TotalExpense	float	☐
	UserID	int	☐
	Hidden	bit	☐

Слика 130 - Приказ табеле Рачун

	Name	Data Type	Allow Nulls
🔑	PaymentCardID	int	☐
🔑	AccountID	int	☐
	PaymentCardNumber	nvarchar(MAX)	☐
	PINCode	nvarchar(MAX)	☐

Слика 131 - Приказ табеле Платна картица

	Name	Data Type	Allow Nulls
🔑	CategoryID	int	☐
	Name	nvarchar(MAX)	☐
	Description	nvarchar(MAX)	☐

Слика 132 - Приказ табеле Категорија

	Name	Data Type	Allow Nulls
🔑	TemplateID	int	☐
	UserID	int	☐
	RecipientAccountNumber	nvarchar(MAX)	☑
	RecipientName	nvarchar(MAX)	☑
	RecipientAddress	nvarchar(MAX)	☑
	Purpose	nvarchar(MAX)	☑
	Model	int	☐
	ReferenceNumber	nvarchar(MAX)	☑
	Amount	float	☐
	AccountNumber	nvarchar(MAX)	☑
	Name	nvarchar(MAX)	☑
	CategoryID	int	☐
	Type	nvarchar(MAX)	☑

Слика 133 - Приказ табеле Шаблон

	Name	Data Type	Allow Nulls
🔑	TransactionID	int	☐
	DateTime	datetime2(7)	☐
	AccountID	int	☑
	AccountNumber	nvarchar(MAX)	☑
	RecipientID	int	☑
	RecipientName	nvarchar(MAX)	☑
	RecipientAddress	nvarchar(MAX)	☑
	RecipientAccountNumber	nvarchar(MAX)	☑
	Purpose	nvarchar(MAX)	☑
	Model	int	☐
	ReferenceNumber	nvarchar(MAX)	☑
	Amount	float	☐
	Type	nvarchar(MAX)	☑

Слика 134 - Приказ табеле Трансакција

	Name	Data Type	Allow Nulls
🔑	AccountID	int	☐
🔑	TransactionID	int	☐
	Hidden	bit	☐

Слика 135 - Приказ табеле ТрансакцијаРачун

	Name	Data Type	Allow Nulls
🔑	CategoryID	int	☐
🔑	TransactionID	int	☐
🔑	OwnerID	int	☐

Слика 136 - Приказ табеле ТрансакцијаКатегорија

	Name	Data Type	Allow Nulls
🔑	UserID	int	☐
	Name	nvarchar(MAX)	☐
	Surname	nvarchar(MAX)	☐
	Username	nvarchar(MAX)	☐
	Password	nvarchar(MAX)	☐
	Address	nvarchar(MAX)	☑

Слика 137 - Приказ табеле Корисник

7.5 Имплементација презентационог слоја

Креирана веб апликација имплементира MVC (Model-View-Controller) патерн који апликацију дели на три групе компоненти: моделе, контролере и погледе. Овим патерном се постиже подела на основу задужења и функције компоненти (енгл. Separation of concerns). Кориснички захтев се рутира до контролера који користећи модел извршава акцију и/или враћа резултате упита. Контролер затим бира поглед који враћа кориснику, који садржи податке садржане у одговарајућем моделу. Имплементацијом овог патерна постиже се једноставније ширење апликације, кодирање, дебаговање и тестирање [8].

Модел у овом патерну представља стање апликације на одређеном погледу. Једноставније, модел је класа која се састоји од атрибута чије вредности добија од контролера или их корисник уноси на погледу. У наставку је дат пример модела:

```
public class UserDetailsModel
{
    public int UserID { get; set; }
    public string Name { get; set; }
    public string Surname { get; set; }
    public List<Account> Accounts { get; set; }
}
```

Контролер је компонента која прихвата кориснички захтев, ради са моделом и одлучује који поглед ће вратити. У MVC апликацији, поглед само приказује информације, док контролер управља улазним подацима које добија од корисника. Контролер представља иницијалну тачку приступа и задужен је за одабир модела и погледа са којим ће радити – по томе и добија свој назив, он контролише одговор апликације на добијен захтев [8].

У ASP.NET Core оквиру, контролери наслеђују апстрактну класу Controller и састоје се из низа метода (акција) које враћају ActionResult – поглед или редирекцију на другу акцију. Акције је могуће додатно одредити прецизирањем на коју методу HTTP захтева пружају одговор. У наставку је дат пример једне акције контролера.

```
public class UserController : Controller

{

    [HttpGet]
    [LoggedInUser]
    public ActionResult Details()
    {
        int? id = HttpContext.Session.GetInt32("userid");
        User user = null;
        UserDetailsModel model = new UserDetailsModel();
        if (id != null)
        {
            user = unitOfWork.User.FindByID((int)id);
        }
        if (user != null)
        {
            model.UserID = user.UserID;
            model.Name = user.Name;
            model.Surname = user.Surname;

            byte[] accountsByte = HttpContext.Session.Get("accounts");
            if (accountsByte == null)
            {
              model.Accounts = unitOfWork.Account.Search(a => a.UserID == user.UserID &&
              !a.Hidden);
            }
            else
            {
               model.Accounts = JsonSerializer.Deserialize<List<Account>>(accountsByte);
            }

        }
        return View(model);
    }

}
```

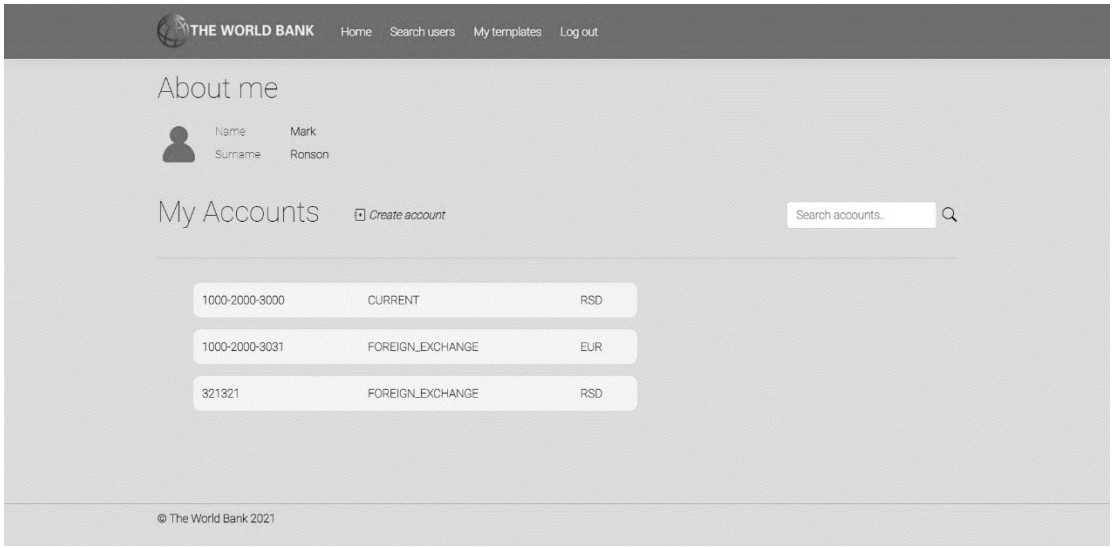
Једна од функционалности које ASP.NET Core оквир нуди су филтери. Филтери се могу односити на акције, целе контролере или глобално, на све контролере у оквиру апликације. Филтери су класе које извршавају одређену логику пре него што контролер извршава одређену акцију. На овај начин, у зависности од резултата методе OnActionExecuting у оквиру филтера, контролер зна да ли треба да изврши акцију. У наставку је дат пример филтера LoggedInUser, који у случају да корисник није улогован на систем, врши редирекцију на страницу за пријаву.

```
public class LoggedInUser : ActionFilterAttribute
{
    public override void OnActionExecuting(ActionExecutingContext context)
    {
        if (context.Filters.OfType<NotLoggedIn>().Any())
        {
            return;
        }

        if (context.HttpContext.Session.GetInt32("userid") == null)
        {
            context.Result = new RedirectResult("/Home/Index");
```

```
        }
        else
        {
            Controller controller = (Controller)context.Controller;
            controller.ViewBag.IsLoggedIn = true;
        }
    }
}
```

Погледи су задужени за презентацију садржају у оквиру корисничког интерфејса. Користе Razor View engine који омогућава коришћење .NET кода унутар HTML страница. Садрже минималну количину логике која је везана за приказивање садржаја [8].

У наставку је приказан један поглед (User.Details) након рендеровања:



Слика 138 - Приказ имплементације погледа

У наставку је дат приказ имплементираних екранских форми.

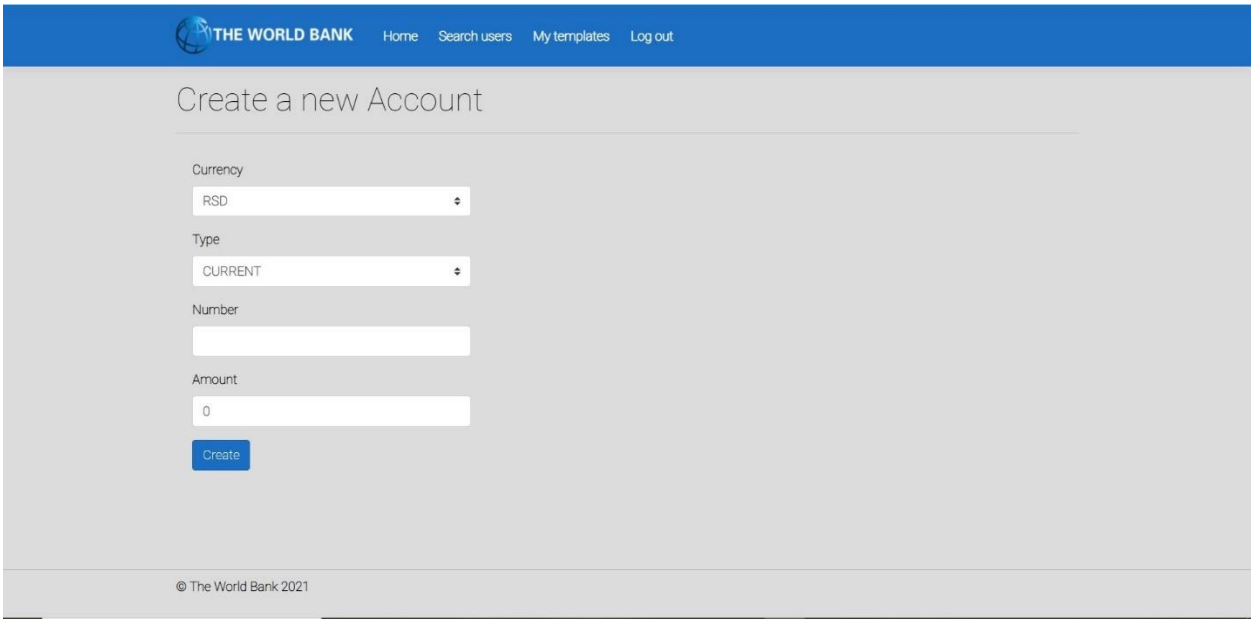
СК1: Случај коришћења – Креирање рачуна

Назив СК: Креирање рачуна

Актори СК: Корисник

Учесници СК: Корисник и систем (програм)

Предуслов: Систем је укључен и корисник је улогован под својом шифром. Систем приказује форму за рад са рачуном.



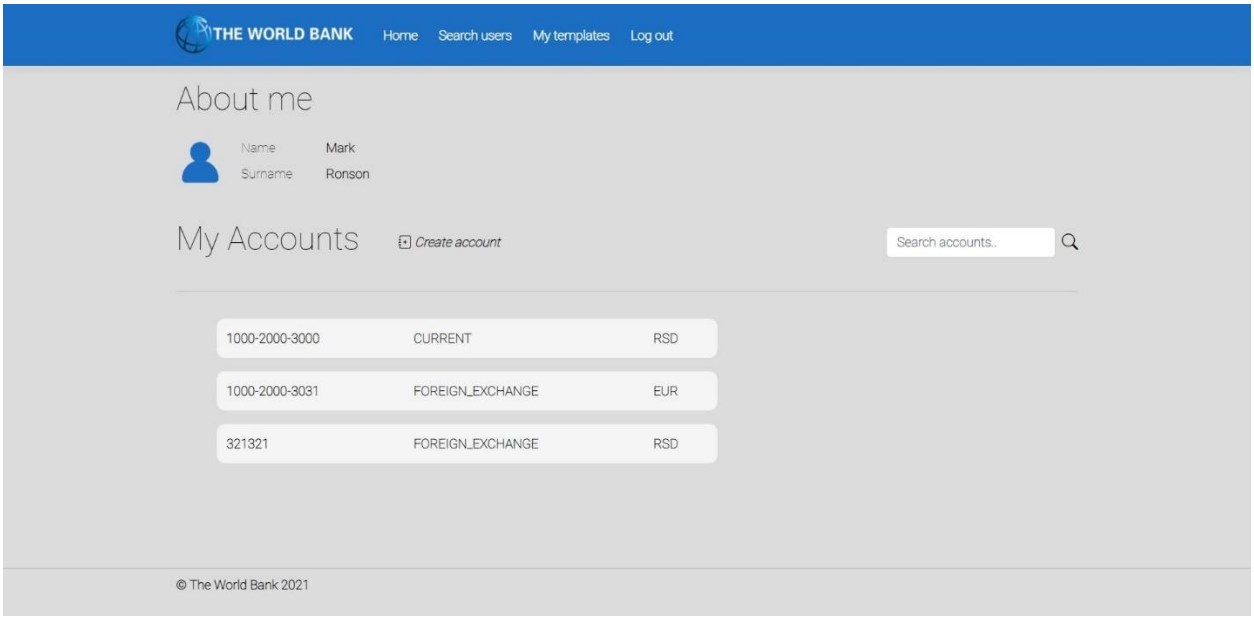
Слика 139 - Страница за креирање рачуна

Основни сценарио СК

1. Корисник **уноси** податке о рачуну. (АПУСО)
2. Корисник **контролише** да ли је коректно унео податке о рачуну. (АНСО)
3. Корисник **позива** систем да запамти рачун. (АПСО)

Опис акције: Корисник притиском на дугме Create позива системску операцију ZapamtiRačun.

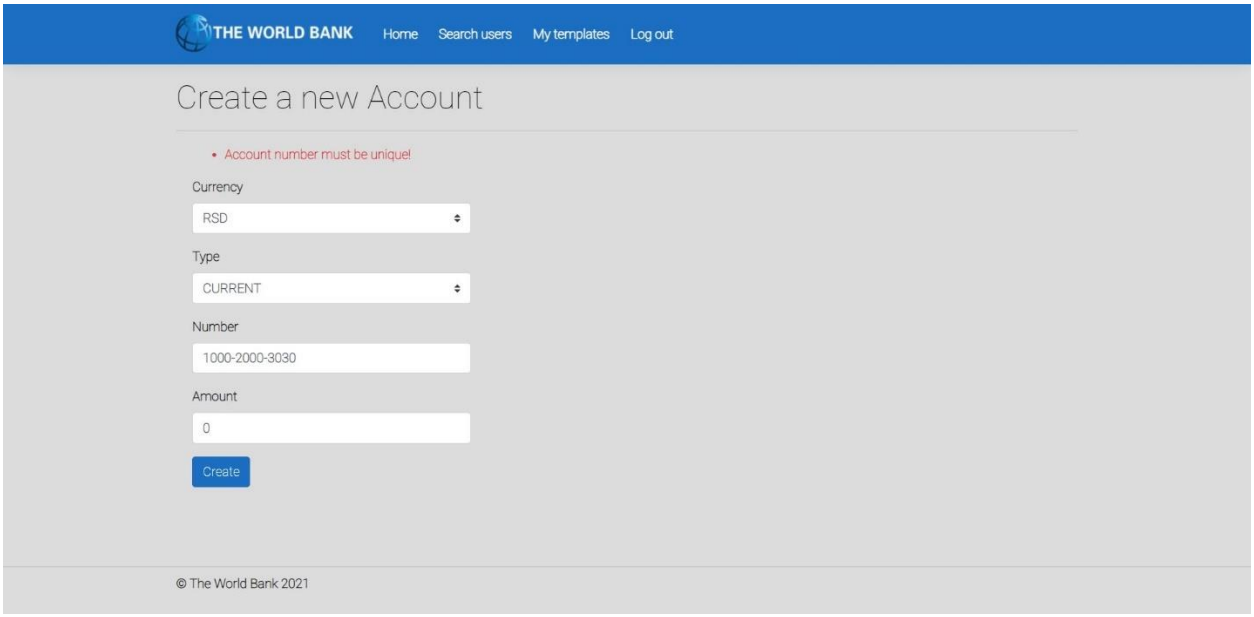
4. Систем **памти** рачун. (СО)
5. Систем **приказује** кориснику запамћени рачун и поруку: “Систем је запамтио рачун“. (ИА)



Слика 140 - Страница са успешно креираним рачунима

Алтернативна сценарија

5.1 Уколико систем не може да креира рачун он приказује кориснику поруку “Систем не може да запамти рачун”. (ИА)



Слика 141 - Страница са неуспешно креираним рачуном

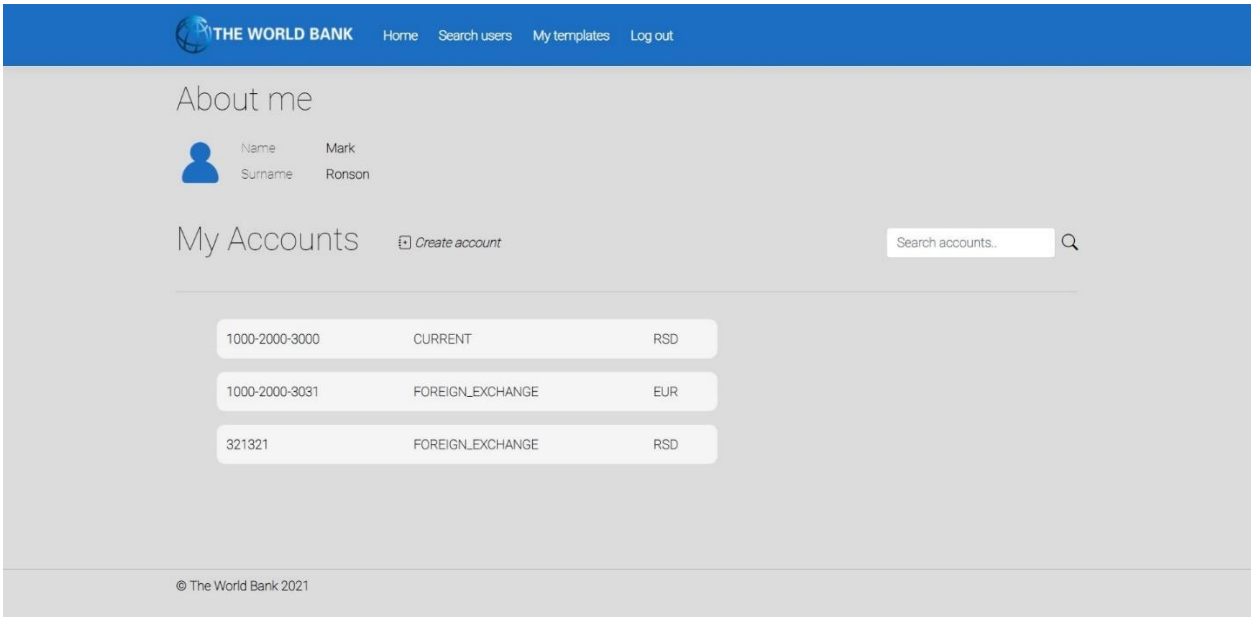
СК2: Случај коришћења – Измена рачуна

Назив СК: Измена рачуна

Актори СК: Корисник

Учесници СК: Корисник и систем (програм)

Предуслов: Систем је укључен и корисник је улогован под својом шифром. Систем приказује форму за рад са рачуном. Учитана су листе корисникових рачуна.



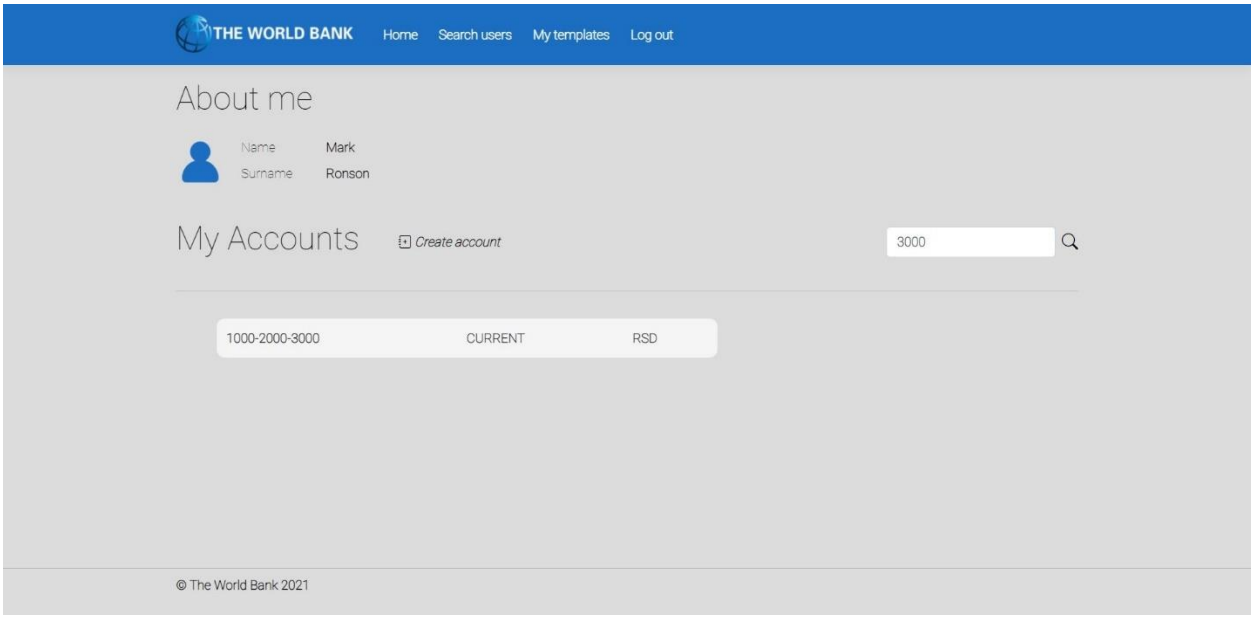
Слика 142 - Страница која приказује корисникове рачуне

Основни сценарио СК

1. Корисник **уноси** вредност по којој претражује рачуне. (АПУСО)
2. Корисник **позива** систем да нађе рачуне по задатој вредности. (АПСО)

Опис акције: Корисник притиском на иконицу лупе позива системску операцију НађиРачуне.

3. Систем **тражи** рачуне по задатој вредности. (СО)
4. Систем **приказује** кориснике рачуне и поруку: “Систем је нашао рачуне по задатој вредности”. (ИА)

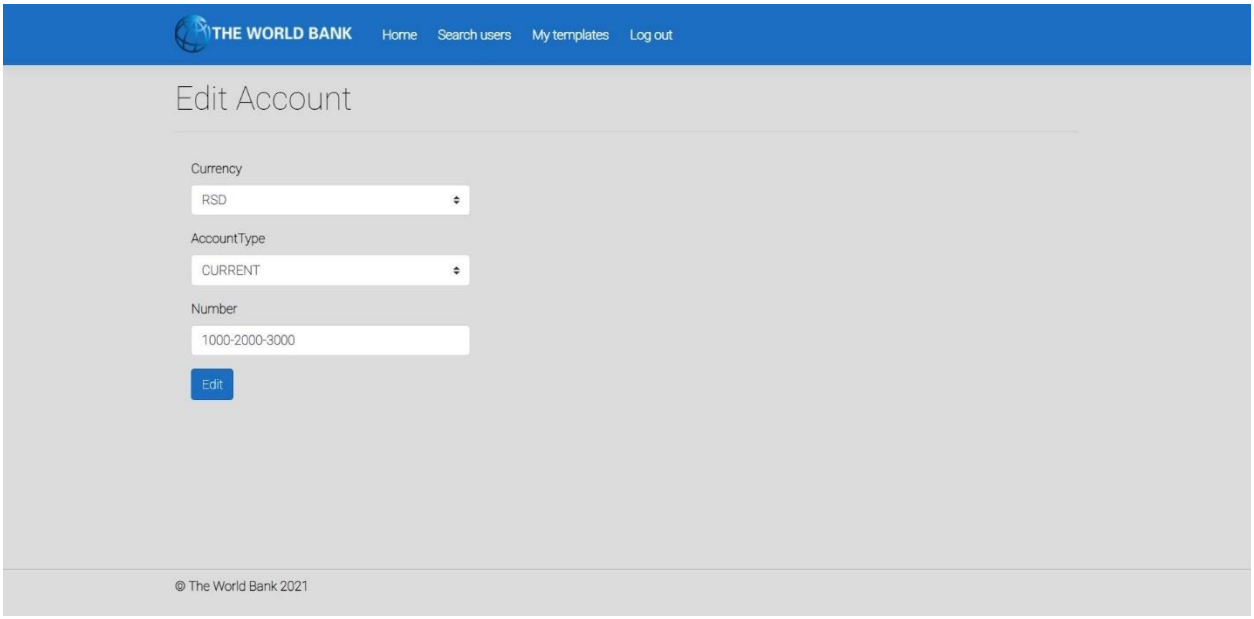


Слика 143 - Успешна претрага рачуна

5. Корисник **бира** рачун. (АПУСО)
6. Корисник **позива** систем да учита рачун. (АПСО)

Опис акције: Корисник притиском на број жељеног рачуна позива системску операцију УчитајРачун.

7. Систем **учитава** рачун. (СО)
8. Систем **показује** кориснику податке о рачуну и поруку “Систем је учитао рачун“ (ИА)

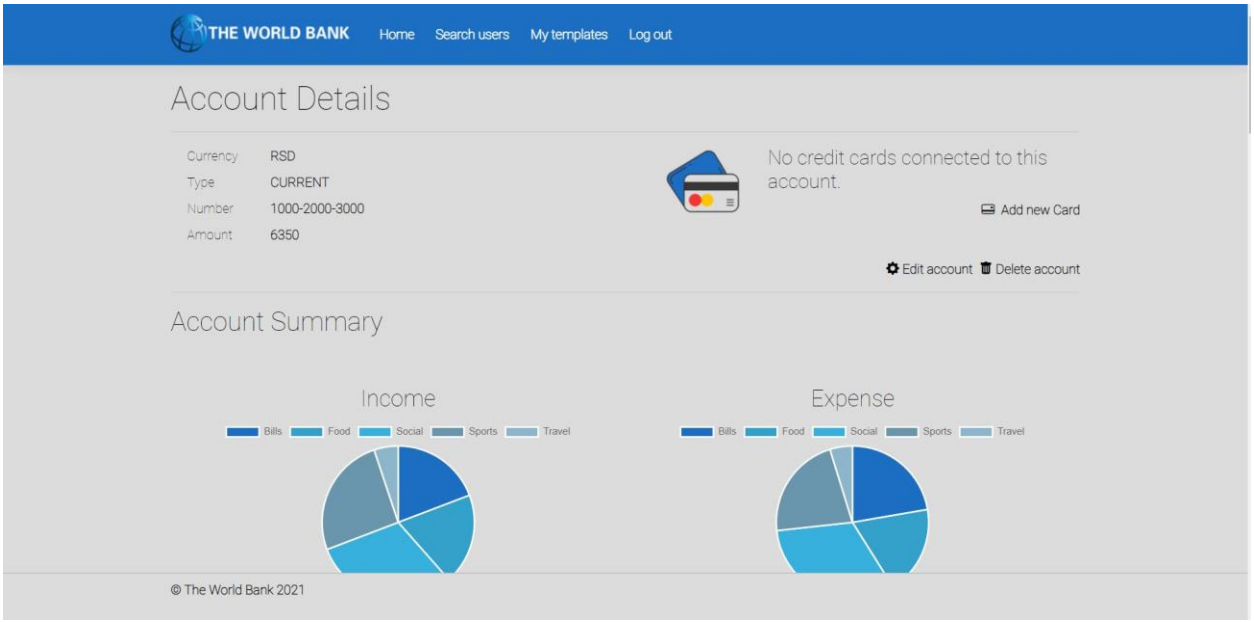


Слика 144 - Страница за уређивање рачуна

9. Корисник **уноси** (мења) податке о рачуну. (АПУСО)
10. Корисник **контролише** да ли је коректно унео податке о рачуну. (АНСО)
11. Корисник **позива** систем да запамти податке о рачуну. (АПСО)

Опис акције: Корисник притиском на дугме Edit позива системску операцију АжурирајРачун.

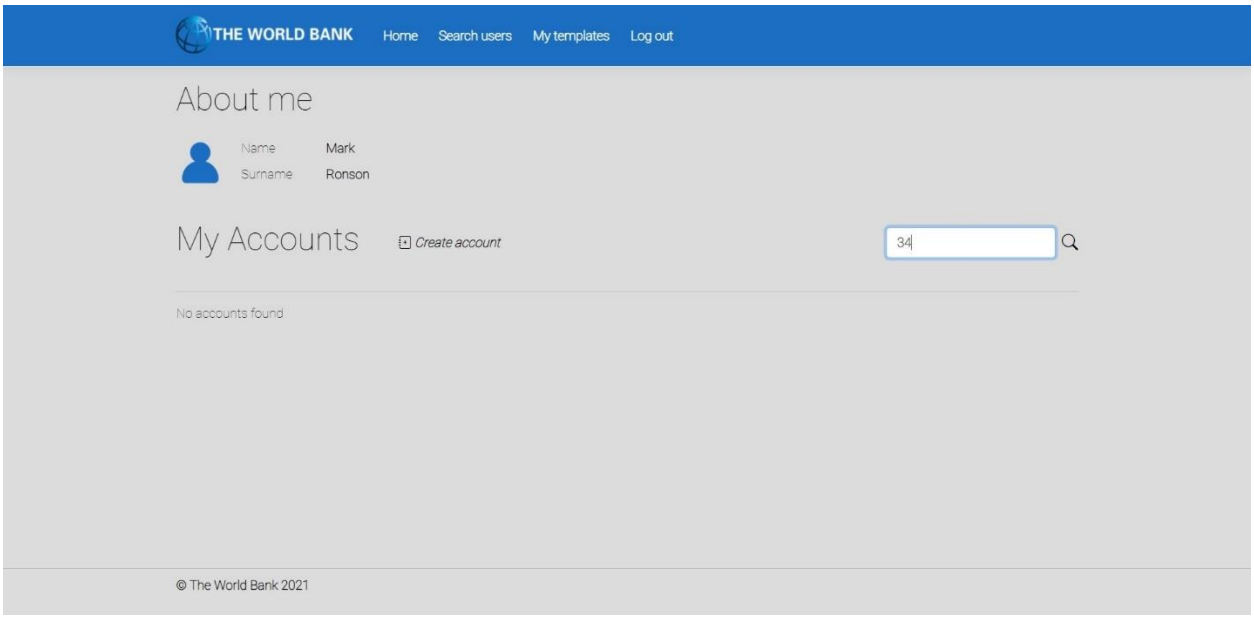
12. Систем **памти** податке о рачуну. (СО)
13. Систем **приказује** кориснику запамћени рачун и поруку: “Систем је запамтио рачун.” (ИА)



Слика 145 - Страница о детаљима рачуна

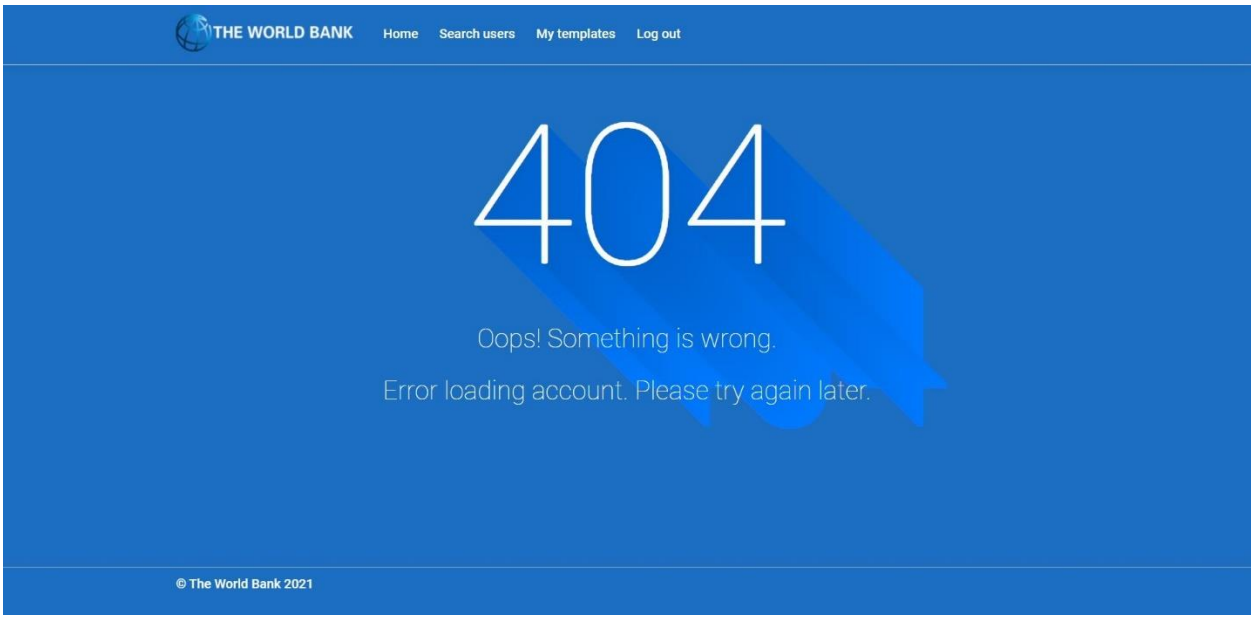
Алтернативна сценарија

4.1 Уколико систем не може да нађе рачуне он приказује кориснику поруку: “Систем не може да нађе рачуне по задатој вредности”. Прекида се извршење сценариа. (ИА)



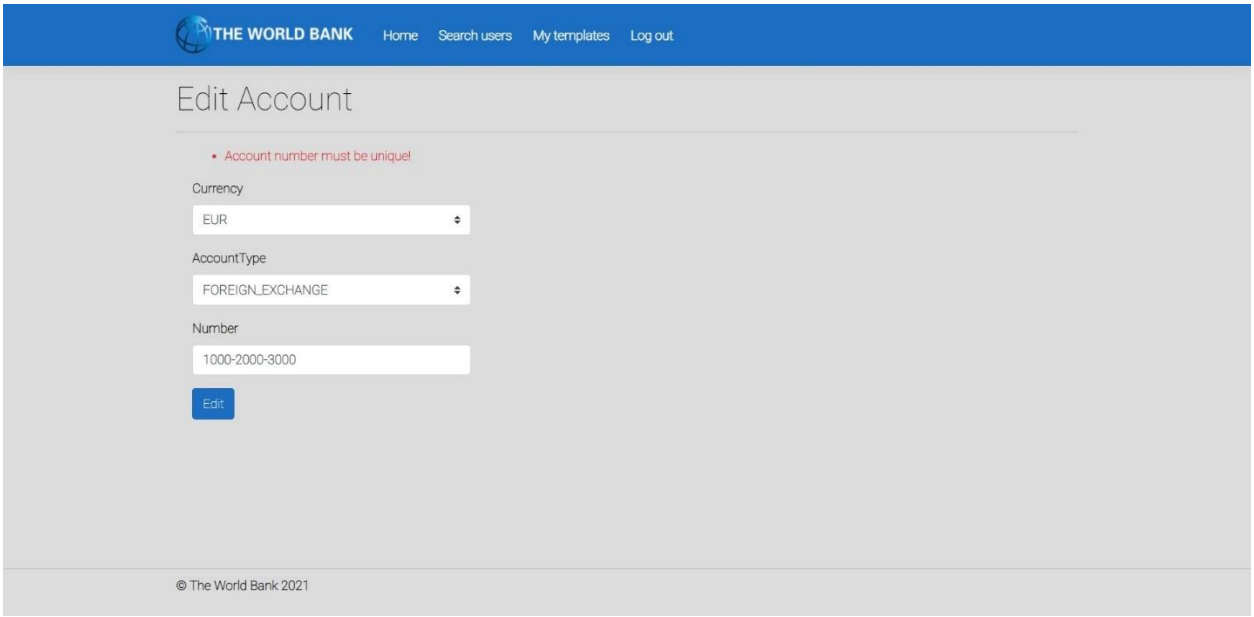
Слика 146 - Неуспешна претрага рачуна

8.1 Уколико систем не може да учита рачун он приказује кориснику поруку “Систем не може да учита рачун”. Прекида се извршење сценариа. (ИА)



Слика 147 - Страница која приказује неуспешно учитавање рачуна

13.1 Уколико систем не може да запамти податке о рачуну он приказује кориснику поруку “Систем не може да запамти рачун”. (ИА)



Слика 148 - Приказ неуспешне измене рачуна

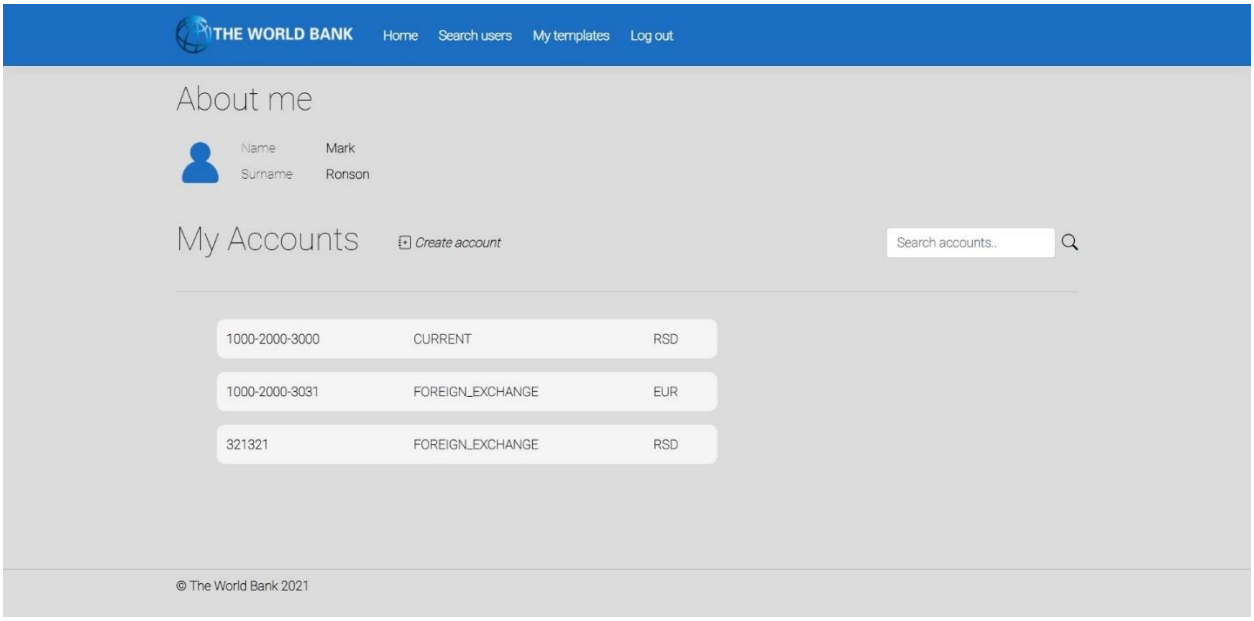
СК3: Случај коришћења – Брисање рачуна

Назив СК: Брисање рачуна

Актори СК: Корисник

Учесници СК: Корисник и систем (програм)

Предуслов: Систем је укључен и корисник је улогован под својом шифром. Систем приказује форму за рад са рачуном. Учитана је листа корисникових рачуна.



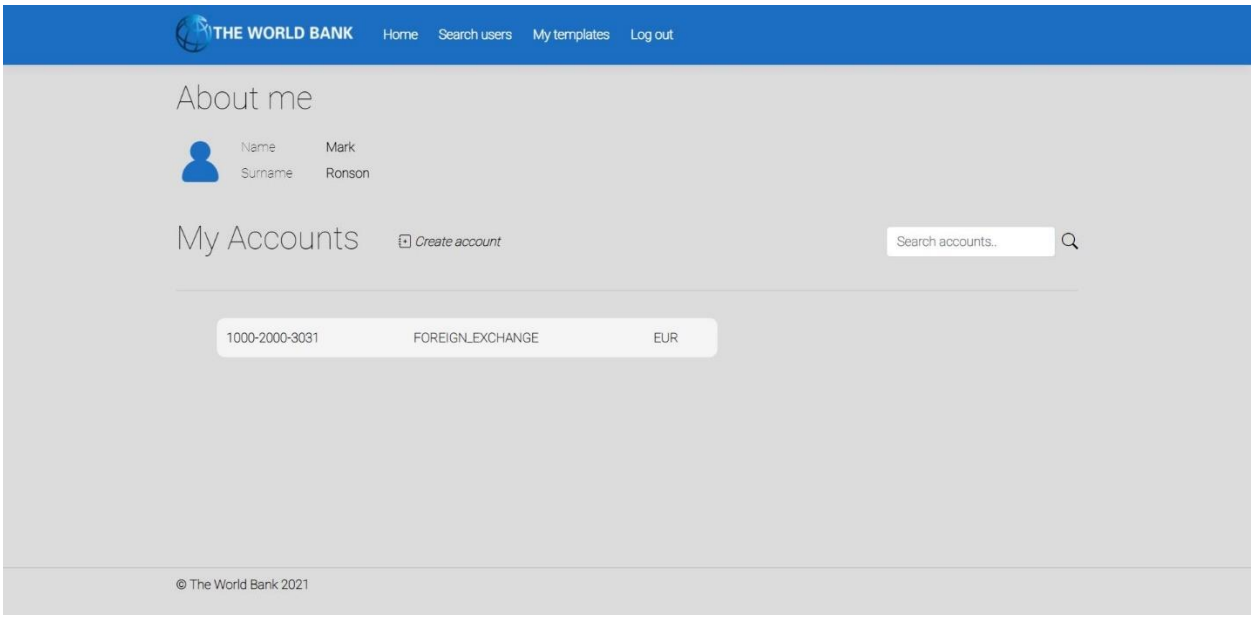
Слика 149 - Почетна страница која приказује корисникове рачуне

Основни сценарио СК

1. Корисник **уноси** вредност по којој претражује рачуне. (АПУСО)
2. Корисник **позива** систем да нађе рачуне по задатој вредности. (АПСО)

Опис акције: Корисник притиском на иконицу лупе позива системску операцију НађиРачуне.

3. Систем **тражи** рачуне по задатој вредности. (СО)
4. Систем **приказује** кориснику рачуне и поруку: “Систем је нашао рачуне по задатој вредности”. (ИА)

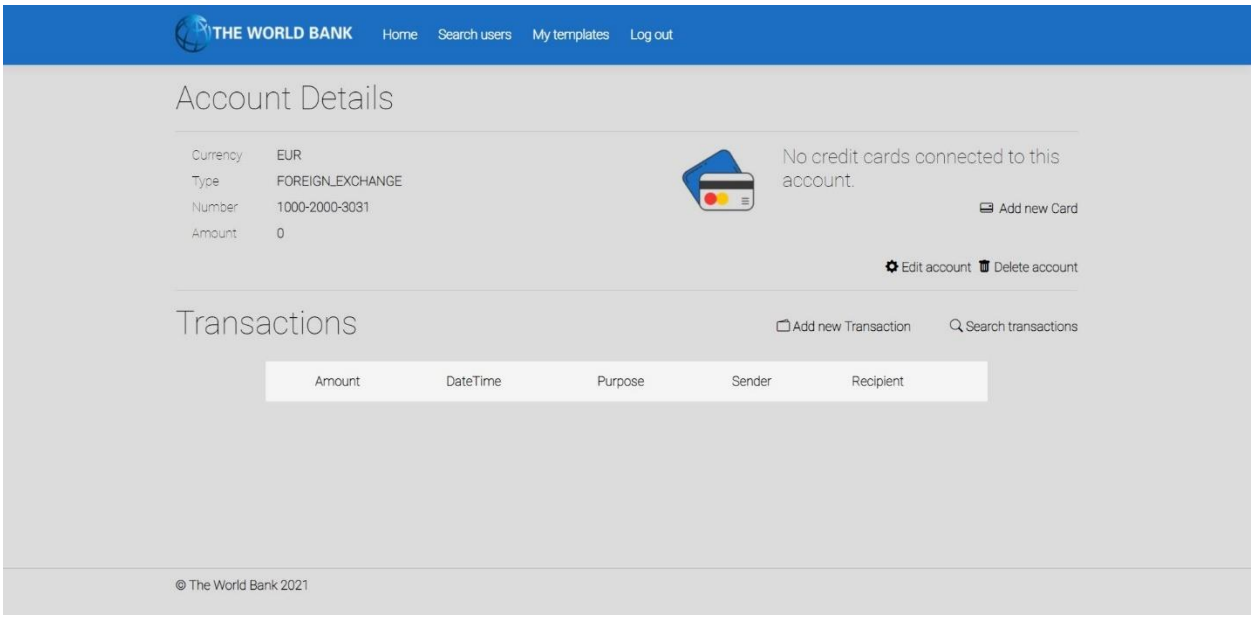


Слика 150 - Приказ резултата претраге рачуна

5. Корисник **бира** рачун који жели да обрише. (АПУСО)
6. Корисник **позива** систем да учита податке о одабраном рачуну. (АПСО)

Опис акције: Корисник притиском на број жељеног рачуна позива системску операцију УчитајРачун.

7. Систем **учитава** податke о одабраном рачуну. (СО)
8. Систем **обавештава** корисника о успешном учитавању података поруком "Систем је учитао одабран рачун" и приказује податке о рачуну. (ИА)

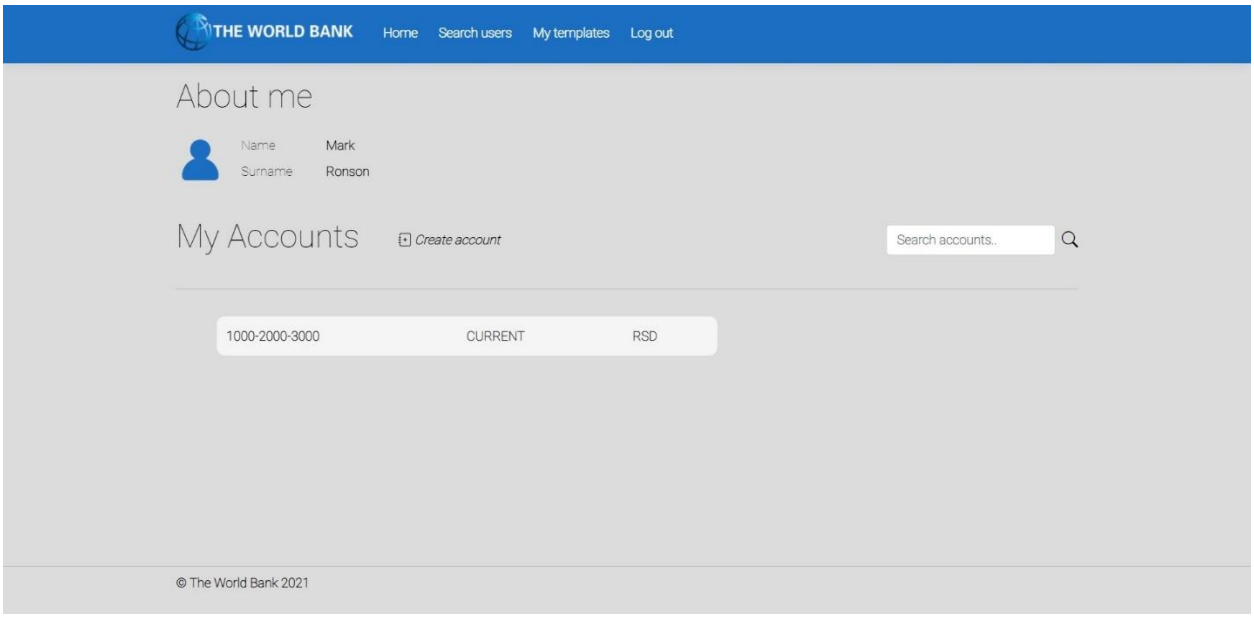


Слика 151 - Приказ детаља о рачуну са опцијом брисања

9. Корисник **позива** систем да обрише рачун. (АПСО)

Опис акције: Корисник притиском на дугме Delete account позива системску операцију ObrišiRačun.

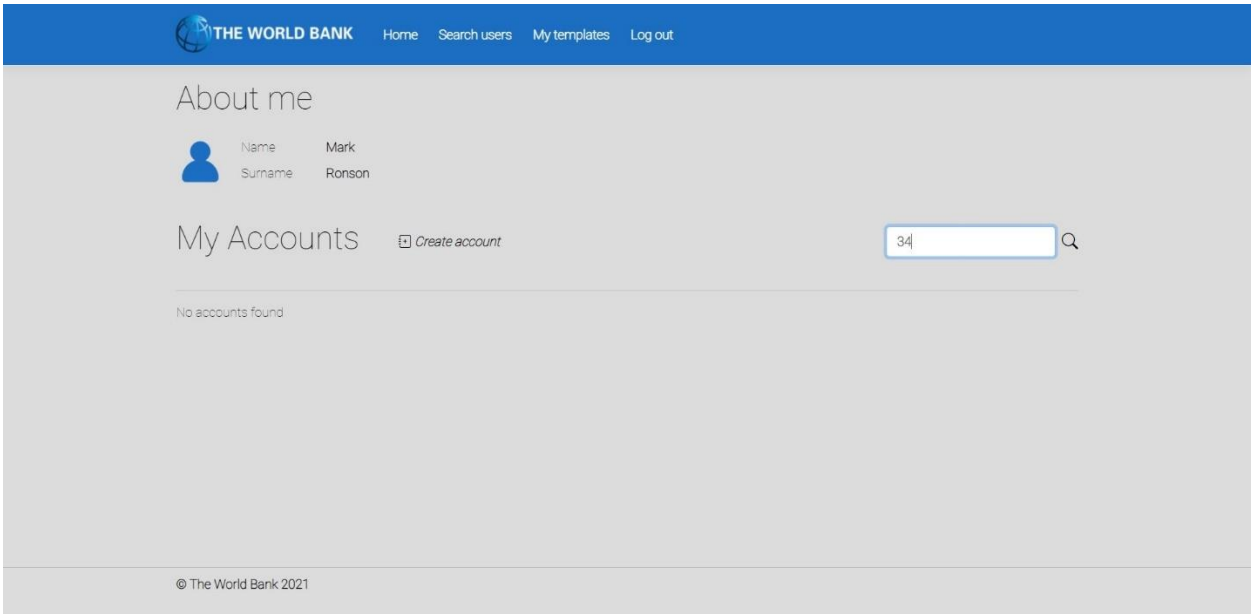
10. Систем **брише** рачун. (СО)
11. Систем **приказује** кориснику поруку: “Систем је обрисао рачун.” (ИА)



Слика 152 - Приказ корисникових рачуна након брисања рачуна

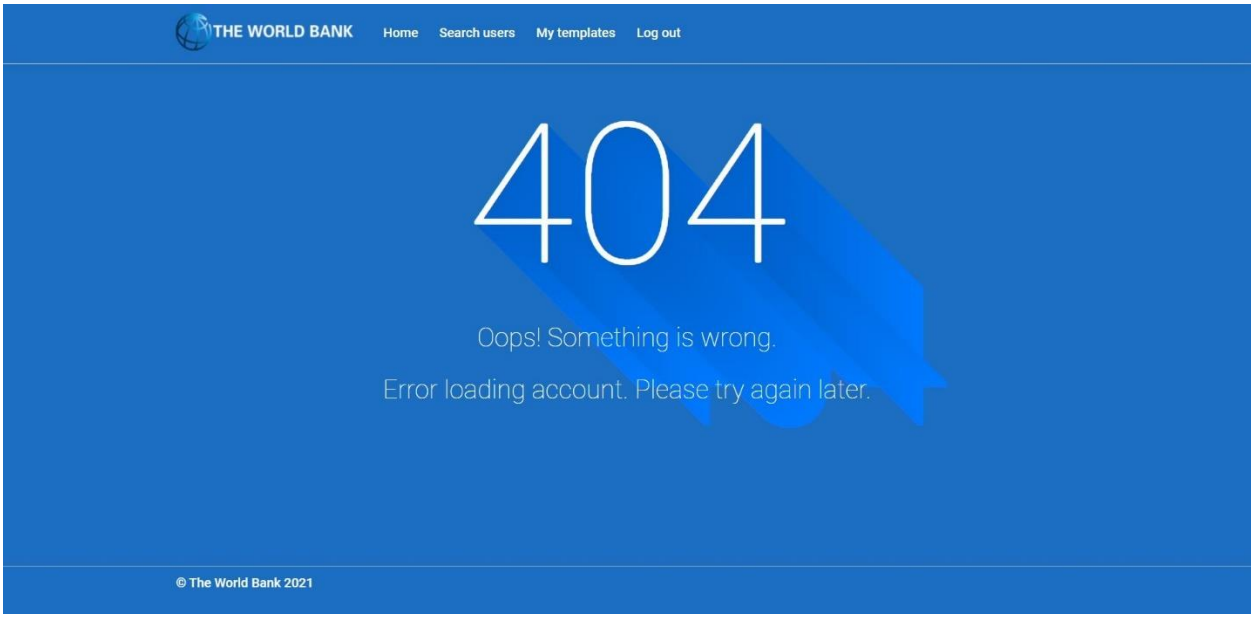
Алтернативна сценарија

4.1 Уколико систем не може да нађе рачун он приказује кориснику поруку: “Систем не може да нађе рачун по задатој вредности”. Прекида се извршење сценариа. (ИА)



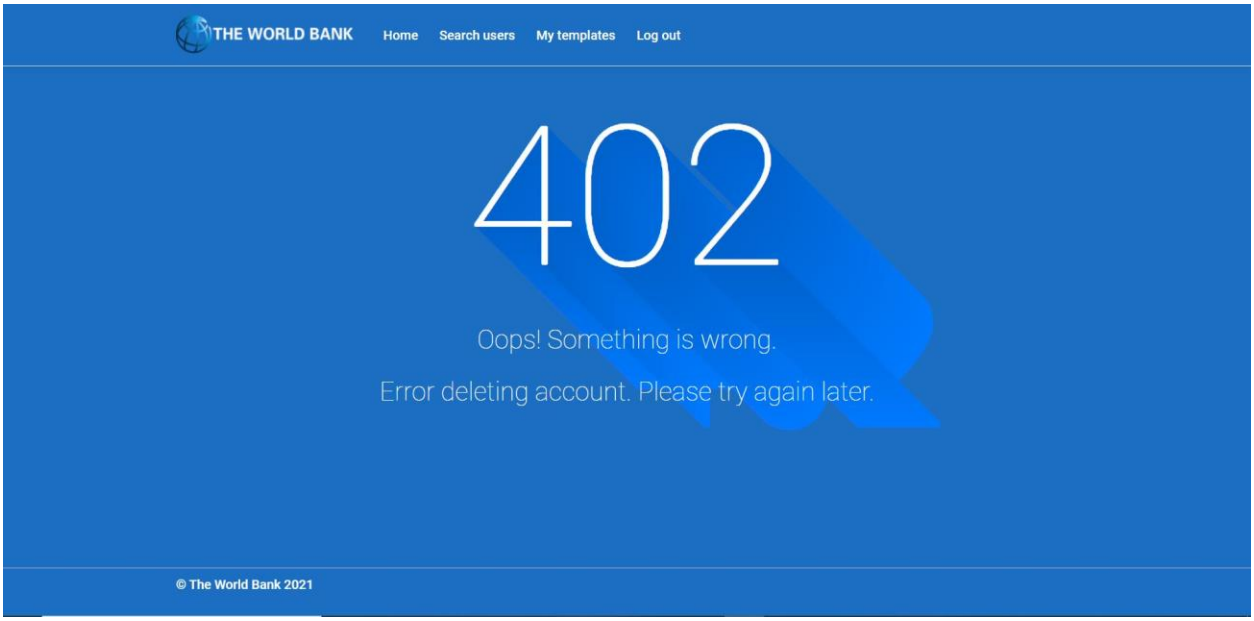
Слика 153 - Приказ неуспешне претраге рачуна приликом брисања рачуна

8.1 Уколико систем не може да учита изабран рачун он приказује кориснику поруку “Систем не може да учита рачун“. Прекида се извршење сценариа (ИА)



Слика 154 - Страница која приказује грешку приликом учитавања рачуна

11.1 Уколико систем не може да обрише рачун он приказује кориснику поруку “Систем не може да обрише рачун”. (ИА)



Слика 155 - Страница која приказује неуспешно брисање рачуна

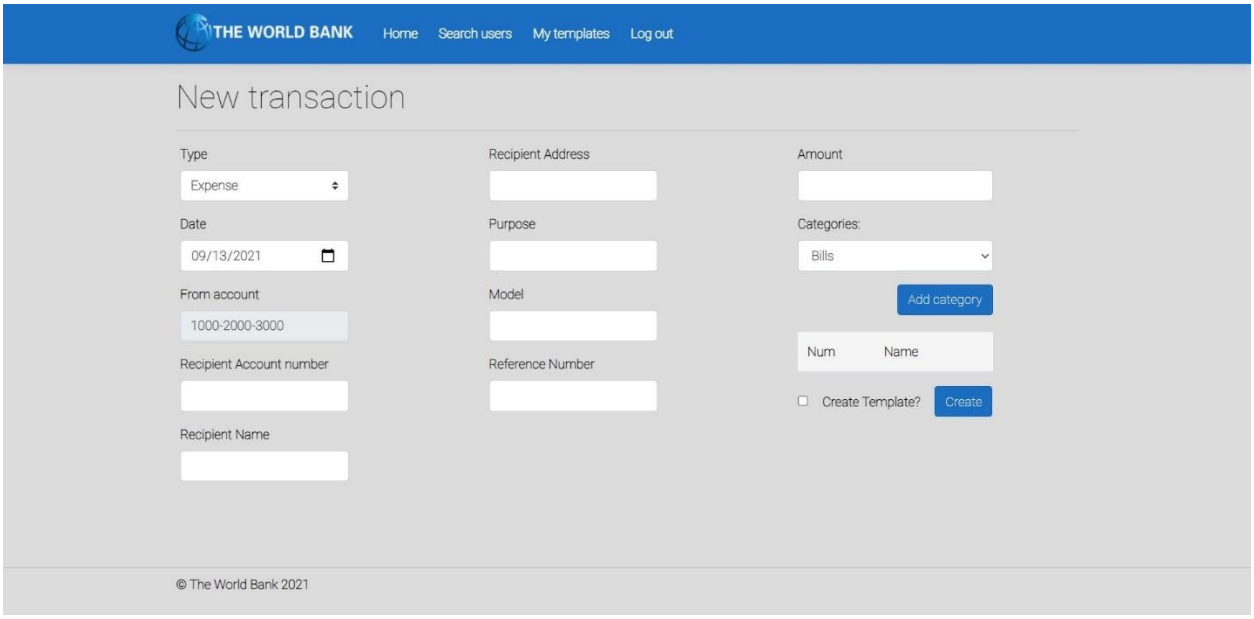
СК4: Случај коришћења – Креирање трансакције

Назив СК: Креирање трансакције

Актори СК: Корисник

Учесници СК: Корисник и систем (програм)

Предуслов: Систем је укључен и корисник је улогован под својом шифром. Систем приказује форму за рад са трансакцијом. Учитана је листа категорија.



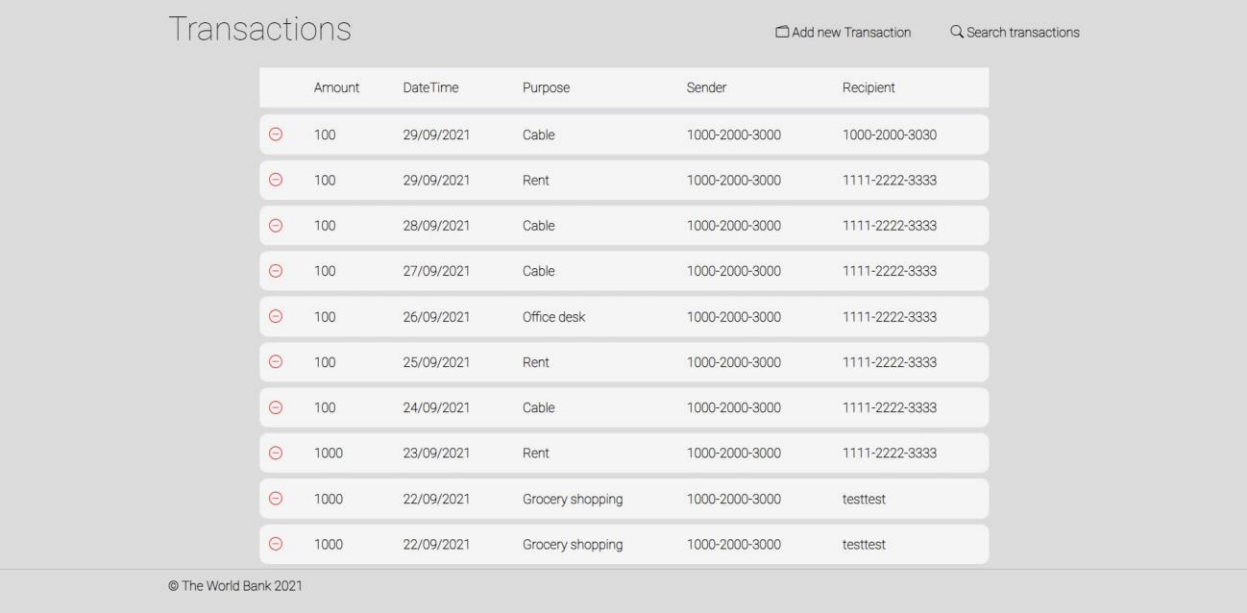
Слика 156 - Страница за унос нове трансакције

Основни сценарио СК

1. Корисник **уноси** податке о трансакцији. (АПУСО)
2. Корисник **контролише** да ли је коректно унео податке о трансакцији. (АНСО)
3. Корисник **позива** систем да запамти трансакцију. (АПСО)

Опис акције: Корисник притиском на дугме Create позива системску операцију ZapamtiTransakciju

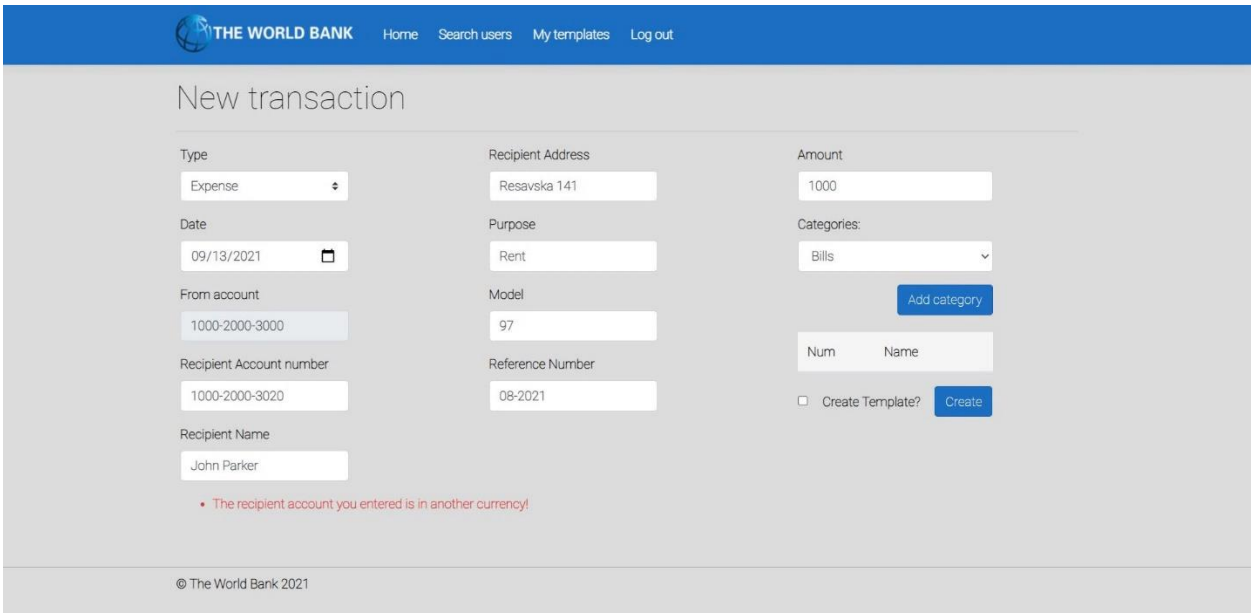
4. Систем **памти** трансакцију. (СО)
5. Систем **приказује** кориснику запамћену трансакцију и поруку: "Систем је запамтио трансакцију". (ИА)



Слика 157 - Приказ креираних трансакција у оквиру странице о детаљима рачуна

Алтернативна сценарија

5.1 Уколико систем не може да запамти податке о трансакцији он приказује кориснику поруку “Систем не може да запамти трансакцију”. (ИА)



Слика 158 - Приказ неуспешног креирања трансакције

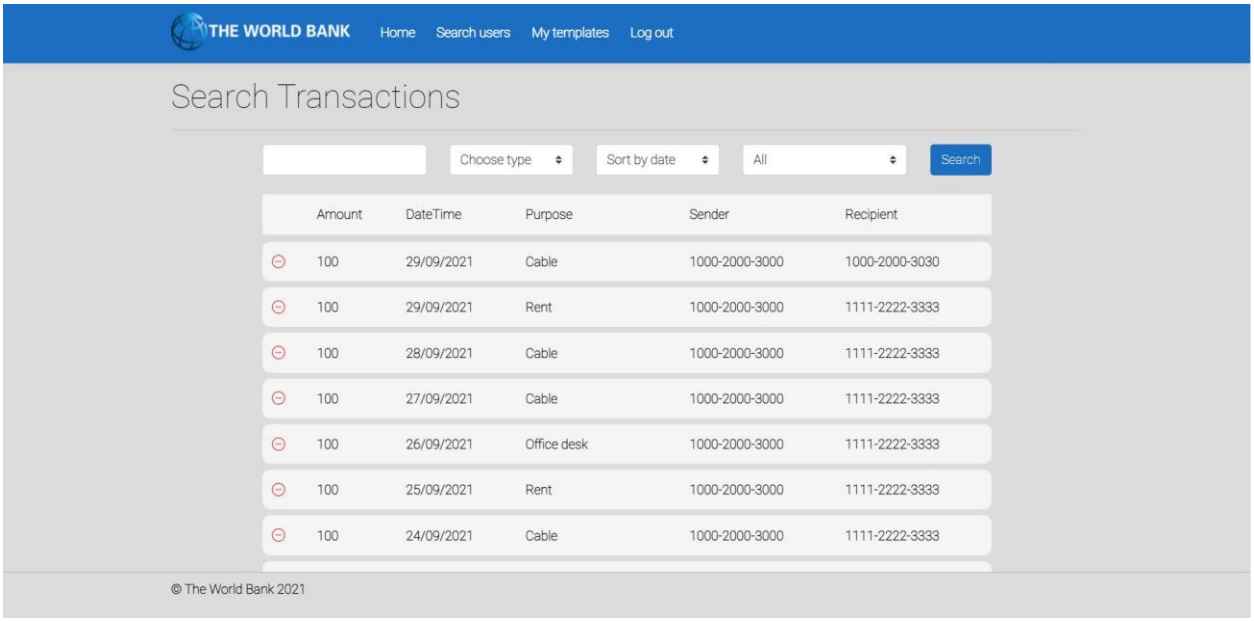
СК5: Случај коришћења – Претрага трансакција

Назив СК: Претраживање трансакција

Актори СК: Корисник

Учесници СК: Корисник и систем (програм)

Предуслов: Систем је укључен и корисник је улогован под својом шифром. Систем приказује форму за рад са трансакцијама. Учитана су листе категорија и трансакција.



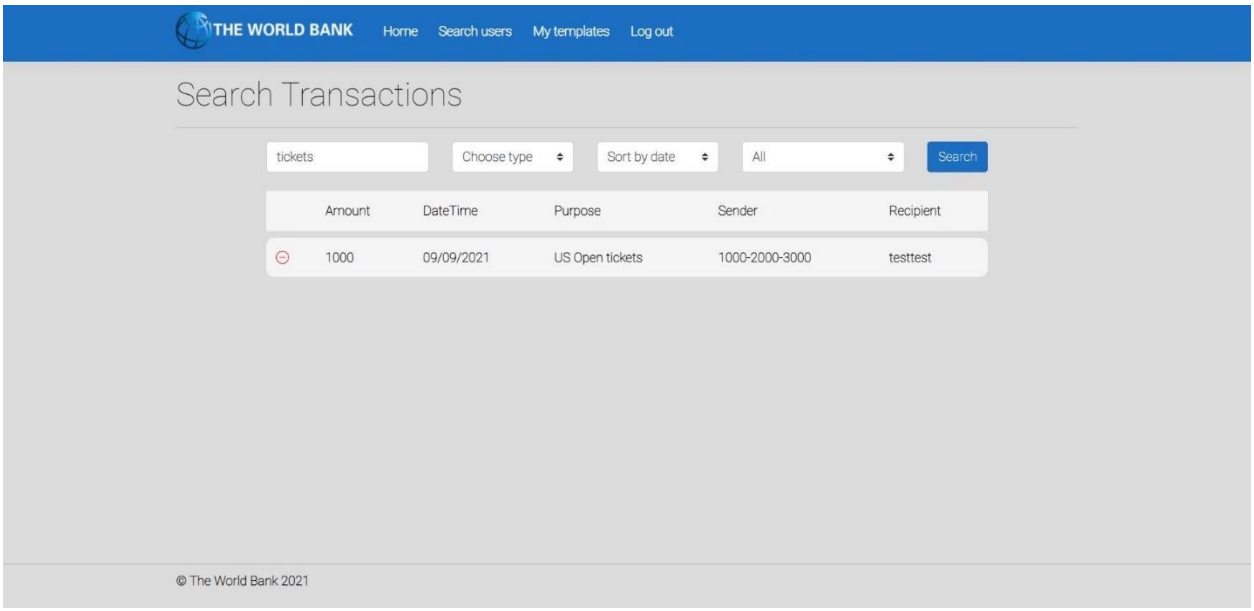
Слика 159 - Страница претраге трансакција

Основни сценарио СК

1. Корисник **уноси** вредност по којој претражује трансакције. (АПУСО)
2. Корисник **позива** систем да нађе трансакције по задатој вредности. (АПСО)

Опис акције: Корисник притиском на дугме Search позива системску операцију НађиTransakcije.

3. Систем **тражи** трансакције по задатој вредности. (СО)
4. Систем **приказује** кориснику податке о трансакцијама и поруку: “Систем је нашао трансакције по задатој вредности”. (ИА)

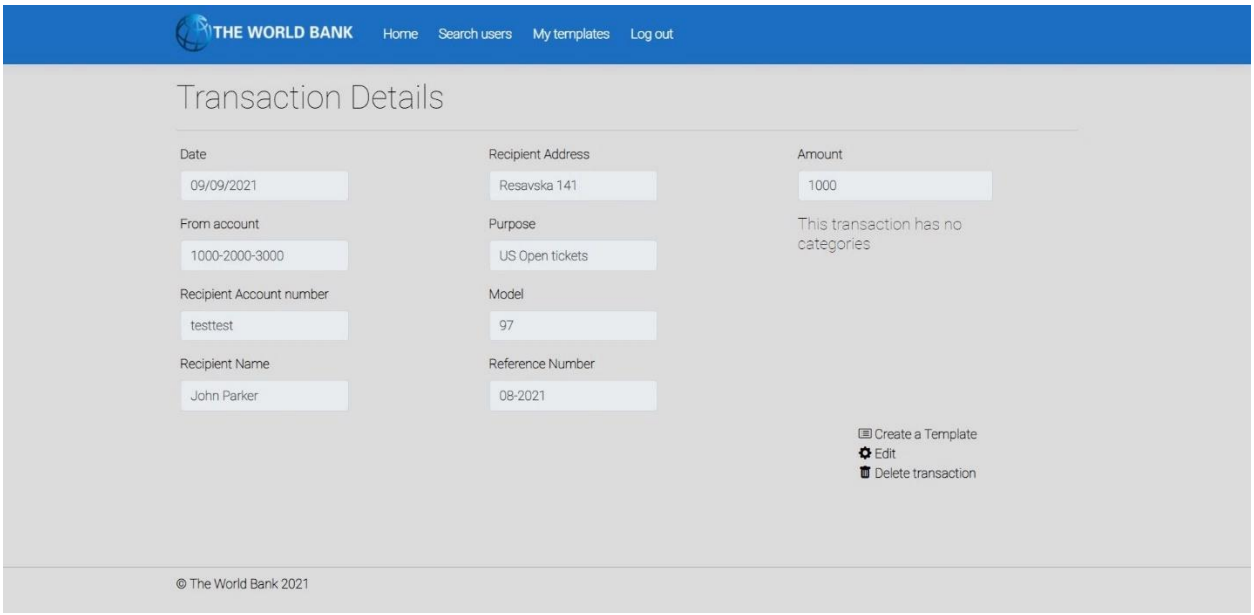


Слика 160 - Приказ резултата претраге трансакција

5. Корисник **бира** трансакцију. (АПУСО)
6. Корисник **позива** систем да учита трансакцију. (АПСО)

Опис акције: Корисник притиском на назив жељене трансакције позива системску операцију УчитајTransakciju.

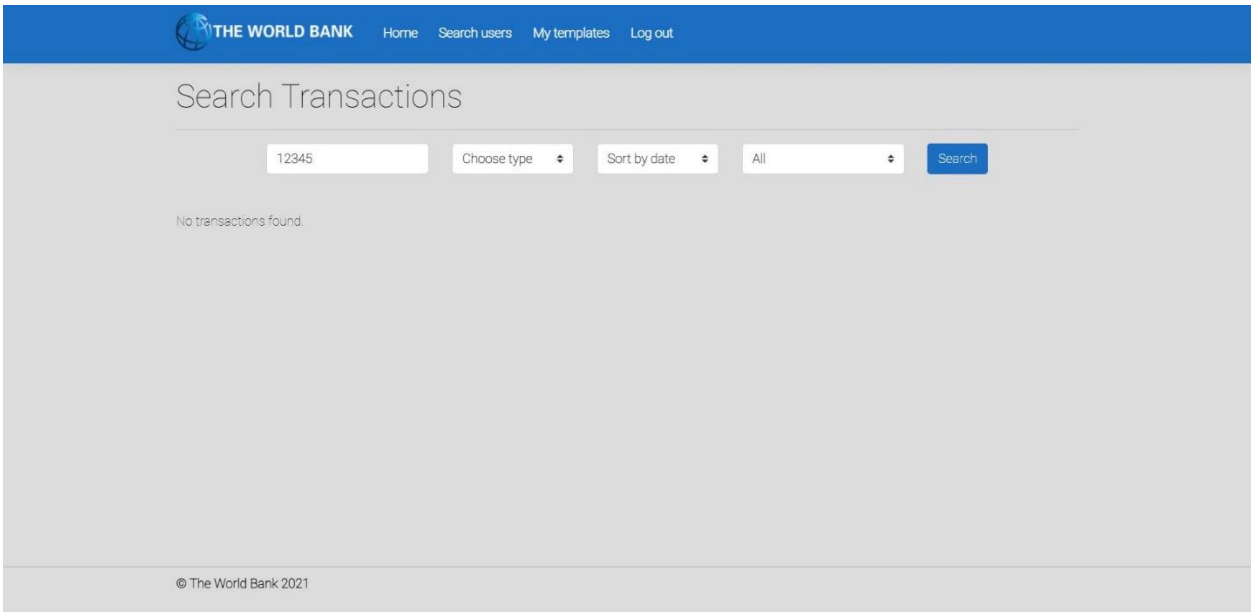
7. Систем **учитава** трансакцију. (СО)
8. Систем **приказује** кориснику податке о трансакцији и поруку: “Систем је учитао трансакцију”. (ИА)



Слика 161 - Приказ успешно учитане трансакције

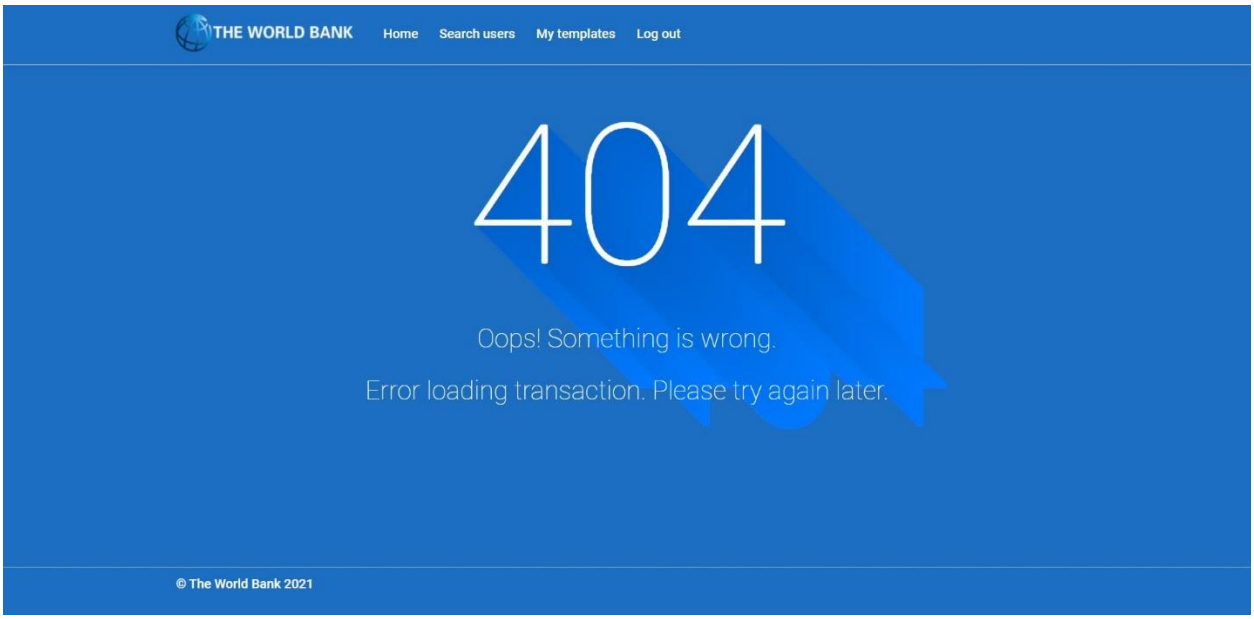
Алтернативна сценарија

4.1 Уколико систем не може да нађе трансакције он приказује кориснику поруку: “Систем не може да нађе трансакције по задатој вредности”. Прекида се извршење сценариа. (ИА)



Слика 162 - Приказ неуспешне претраге трансакција

8.1 Уколико систем не може да учита трансакцију он приказује кориснику поруку “Систем не може да учита трансакцију”. (ИА)



Слика 163 - Приказ грешке приликом учитавања трансакције

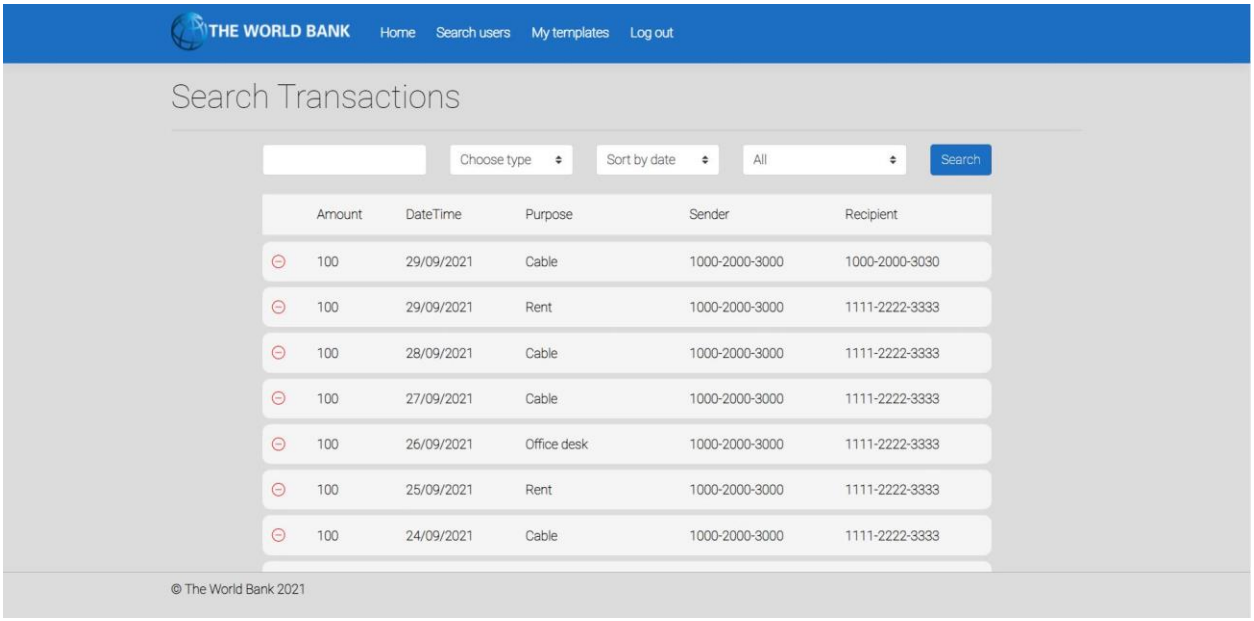
СК6: Случај коришћења – Измена трансакција

Назив СК: Измена трансакција

Актори СК: Корисник

Учесници СК: Корисник и систем (програм)

Предуслов: Систем је укључен и корисник је улогован под својом шифром. Систем приказује форму за рад са трансакцијом. Учитана су листе трансакција и категорија.



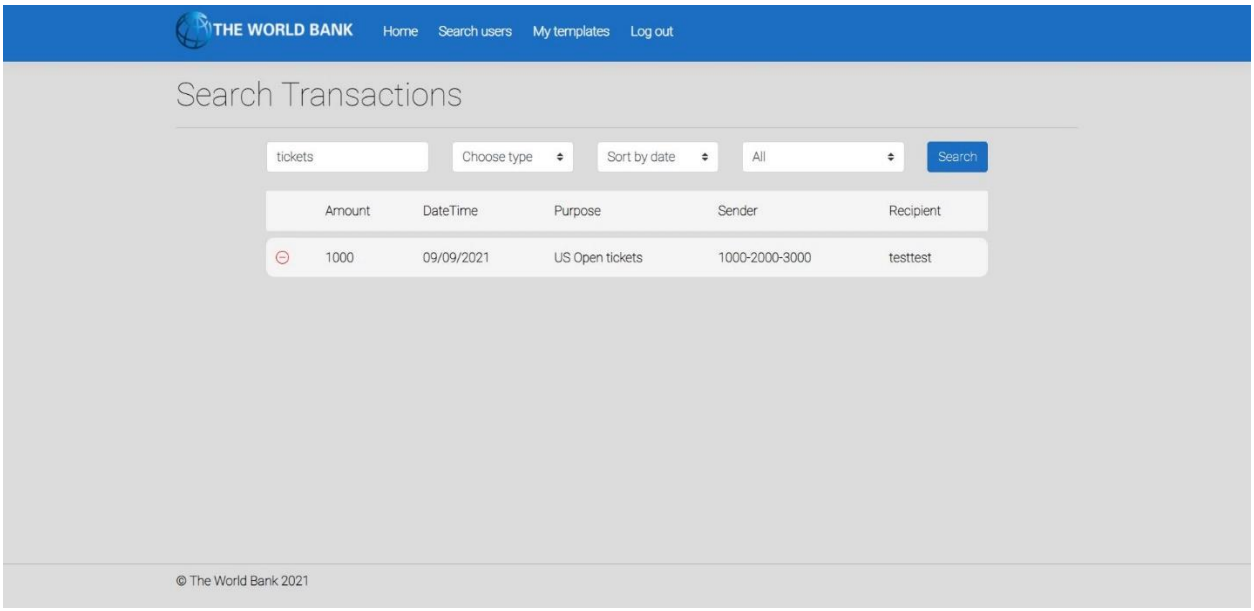
Слика 164 - Почетна страница за претрагу трансакција

Основни сценарио СК

1. Корисник **уноси** вредност по којој претражује трансакције. (АПУСО)
2. Корисник **позива** систем да нађе трансакције по задатој вредности. (АПСО)

Опис акције: Корисник притиском на дугме Search позива системску операцију НађиTransakcije.

3. Систем **тражи** трансакције по задатој вредности. (СО)
4. Систем **приказује** кориснику трансакције и поруку: “Систем је нашао трансакције по задатој вредности”. (ИА)

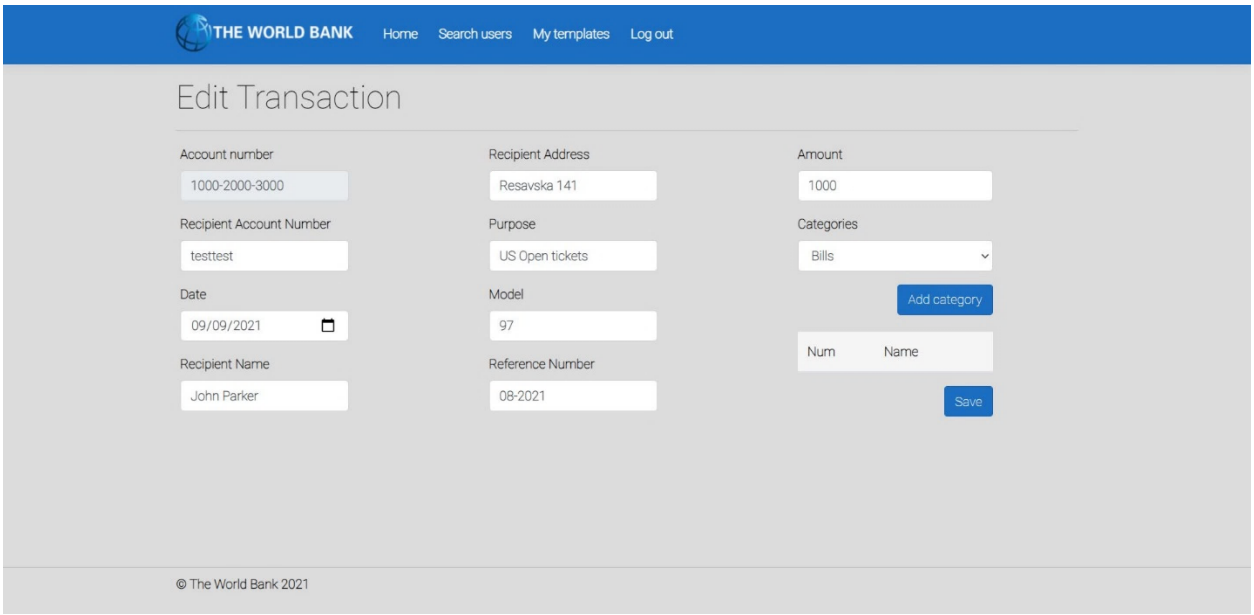


Слика 165 - Резултат претраге трансакција

5. Корисник **бира** трансакцију. (АПУСО)
6. Корисник **позива** систем да учита трансакцију. (АПСО)

Опис акције: Корисник притиском на назив жељене трансакције позива системску операцију УčitajTransakciju.

7. Систем **учитава** трансакцију. (СО)
8. Систем **показује** кориснику податке о трансакцији и поруку “Систем је учитао трансакцију“ (ИА)

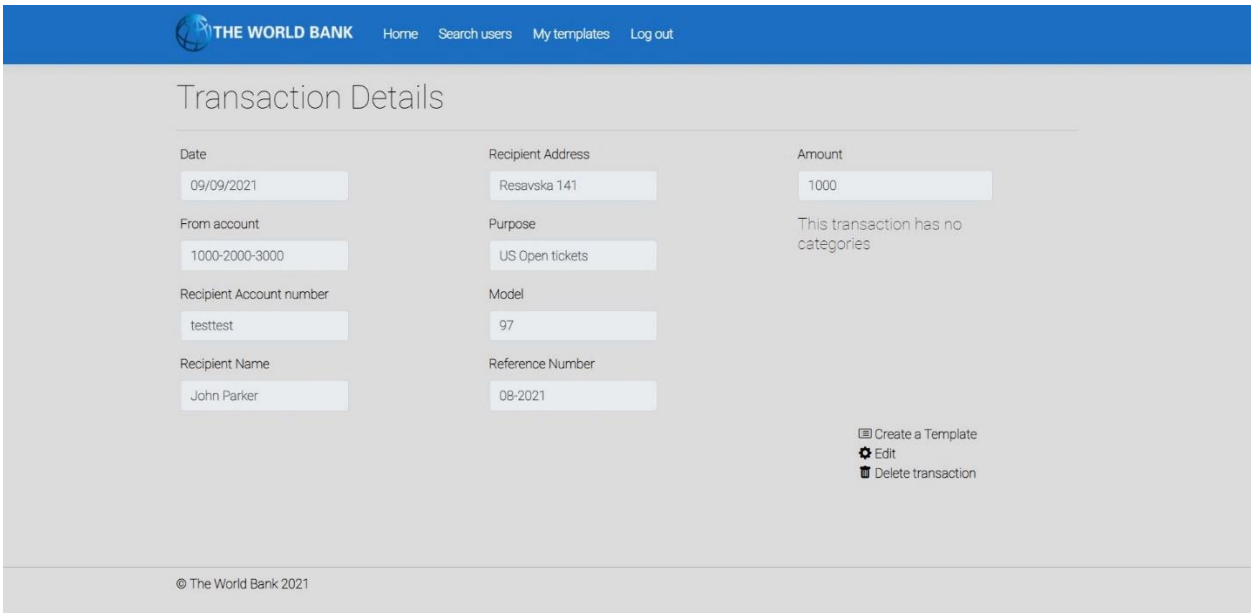


Слика 166 - Страница за измену трансакције

9. Корисник **уноси** (мења) податке о трансакцији. (АПУСО)
10. Корисник **контролише** да ли је коректно унео податке о трансакцији. (АНСО)
11. Корисник **позива** систем да запамти податке о трансакцији. (АПСО)

Опис акције: Корисник притиском на дугме Save позива системску операцију АжурирајТransakciju.

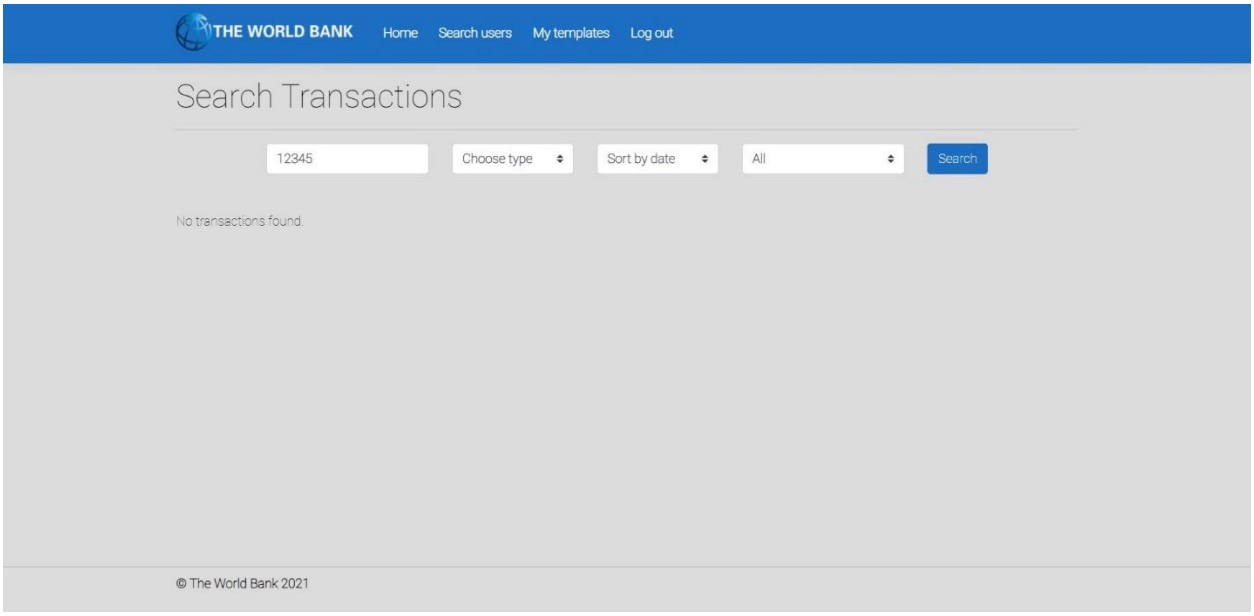
12. Систем **памти** податке о трансакцији. (СО)
13. Систем **приказује** кориснику запамћену трансакцију и поруку: "Систем је запамтио трансакцију." (ИА)



Слика 167 - Приказ сачуване трансакције након измене

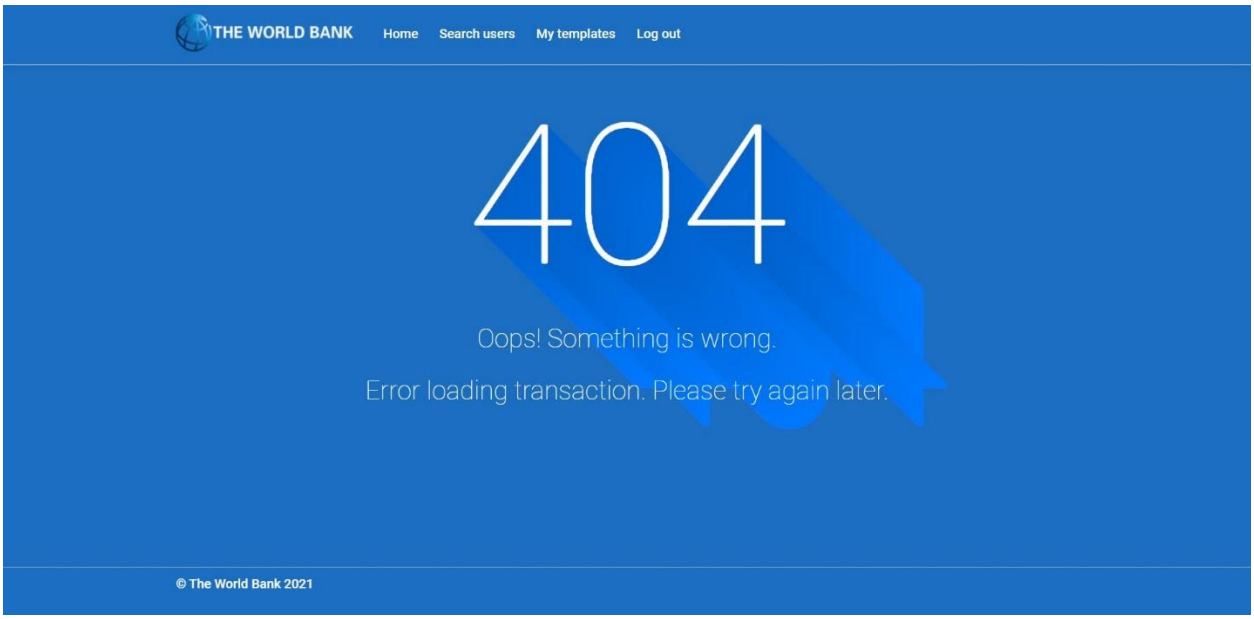
Алтернативна сценарија

4.1 Уколико систем не може да нађе трансакције он приказује кориснику поруку: "Систем не може да нађе трансакције по задатој вредности". Прекида се извршење сценариа. (ИА)



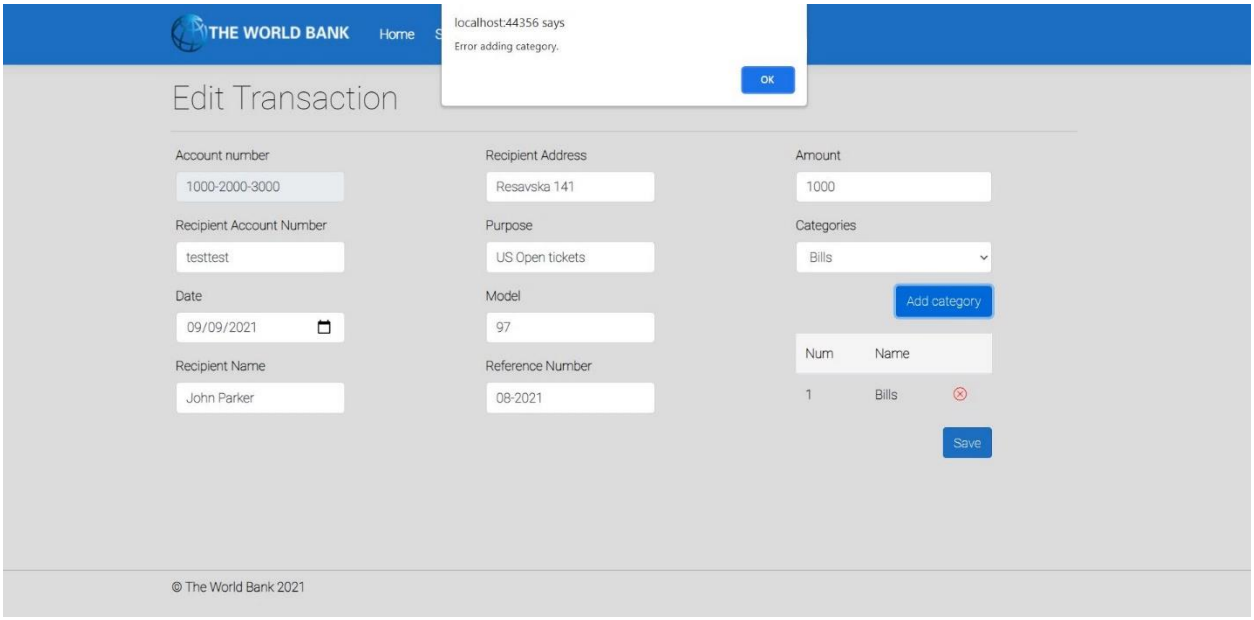
Слика 168 - Неуспешна претрага трансакција приликом измене

8.1 Уколико систем не може да учита трансакцију он приказује кориснику поруку “Систем не може да учита трансакцију”. Прекида се извршење сценариа. (ИА)



Слика 169 - Страница о грешци приликом учитавања трансакције

13.1 Уколико систем не може да запамти податке о трансакцији он приказује кориснику поруку “Систем не може да запамти трансакцију”. (ИА)



Слика 170 - Приказ грешке приликом измене трансакције

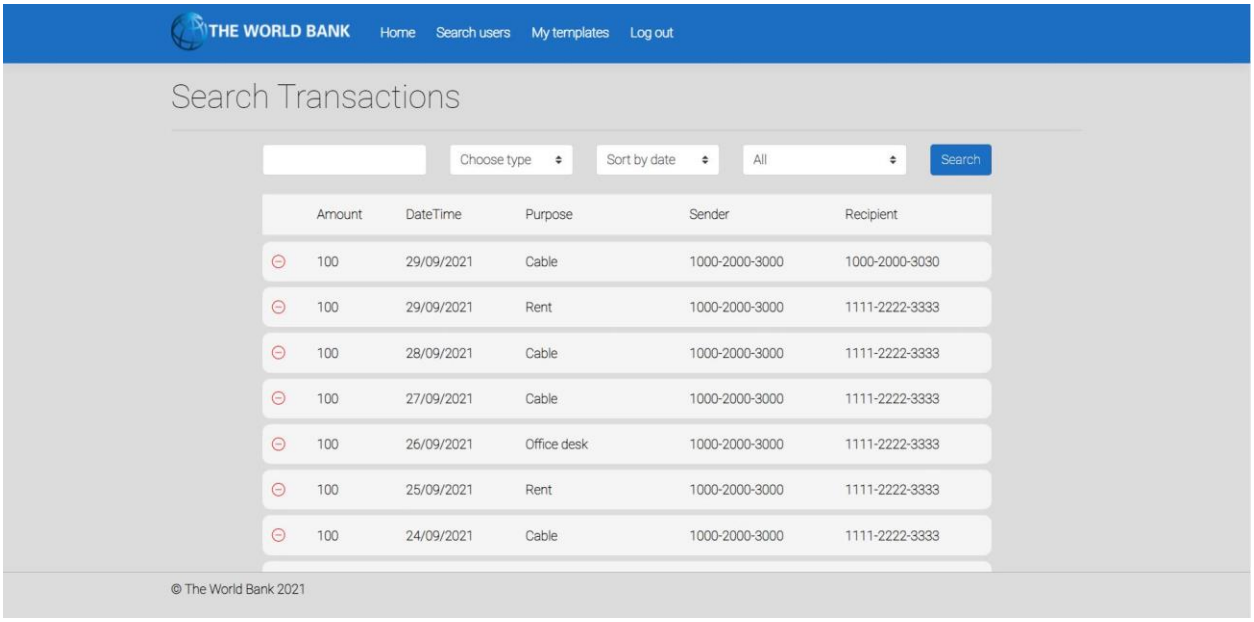
СК7: Случај коришћења – Брисање трансакције

Назив СК: Брисање трансакције

Актори СК: Корисник

Учесници СК: Корисник и систем (програм)

Предуслов: Систем је укључен и корисник је улогован под својом шифром. Систем приказује форму за рад са трансакцијом. Учитана су листе категорија и трансакција.



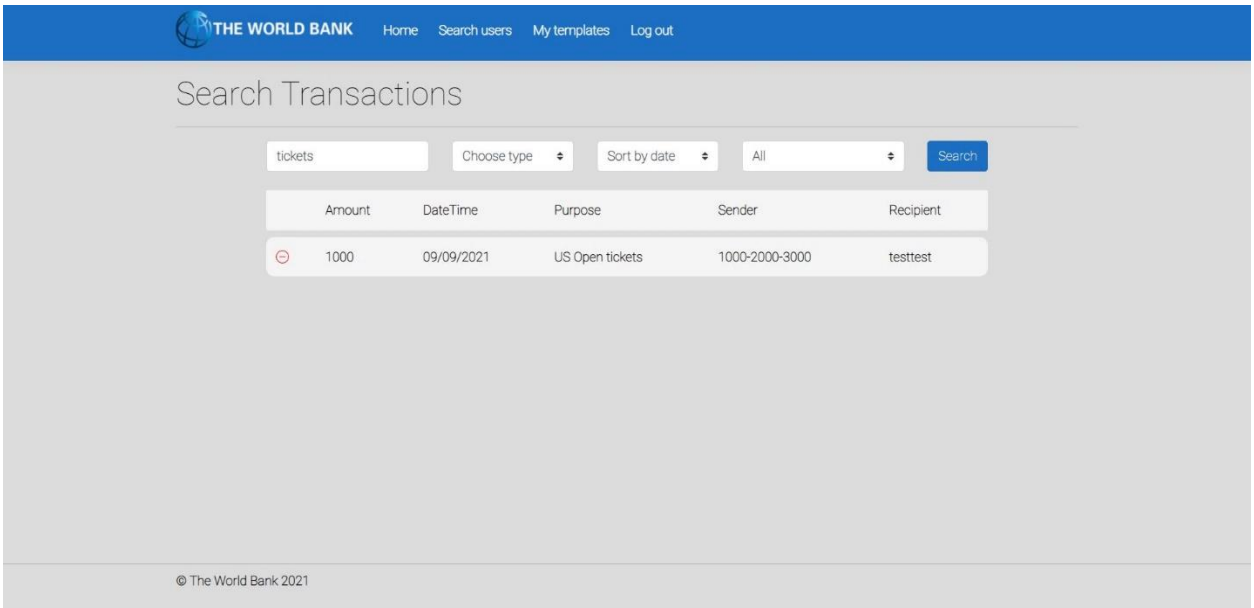
Слика 171 - Приказ свих трансакција са корисниковог рачуна

Основни сценарио СК

1. Корисник **уноси** вредност по којој претражује трансакције. (АПУСО)
2. Корисник **позива** систем да нађе трансакције по задатој вредности. (АПСО)

Опис акције: Корисник притиском на дугме Search позива системску операцију НађиTransakcije.

3. Систем **тражи** трансакције по задатој вредности. (СО)
4. Систем **приказује** кориснику трансакције и поруку: “Систем је нашао трансакције по задатој вредности”. (ИА)

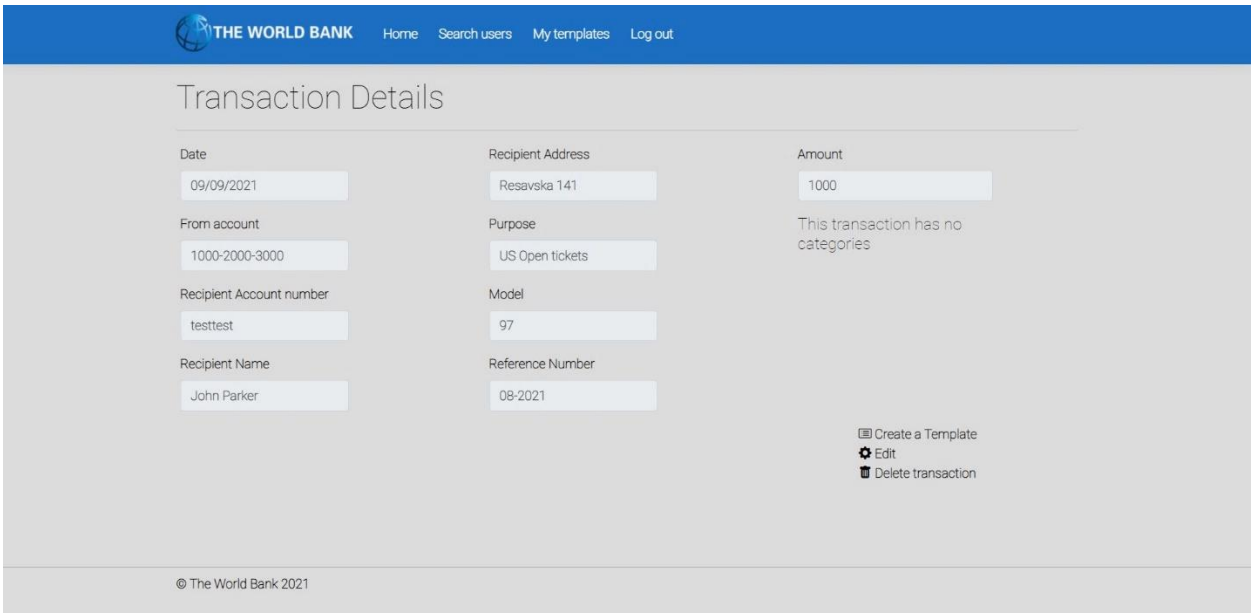


Слика 172 - Приказ резултата претраге трансакција

5. Корисник **бира** трансакцију који жели да обрише. (АПУСО)
6. Корисник **позива** систем да учита податке о одабраној трансакцији. (АПСО)

Опис акције: Корисник притиском на назив жељене трансакције позива системску операцију УчитајTransakciju.

7. Систем **учитава** податке о одабраном трансакцији. (СО)
8. Систем **обавештава** корисника о успешном учитавању података поруком "Систем је учитао одабран трансакцију" и приказује податке о трансакцији. (ИА)

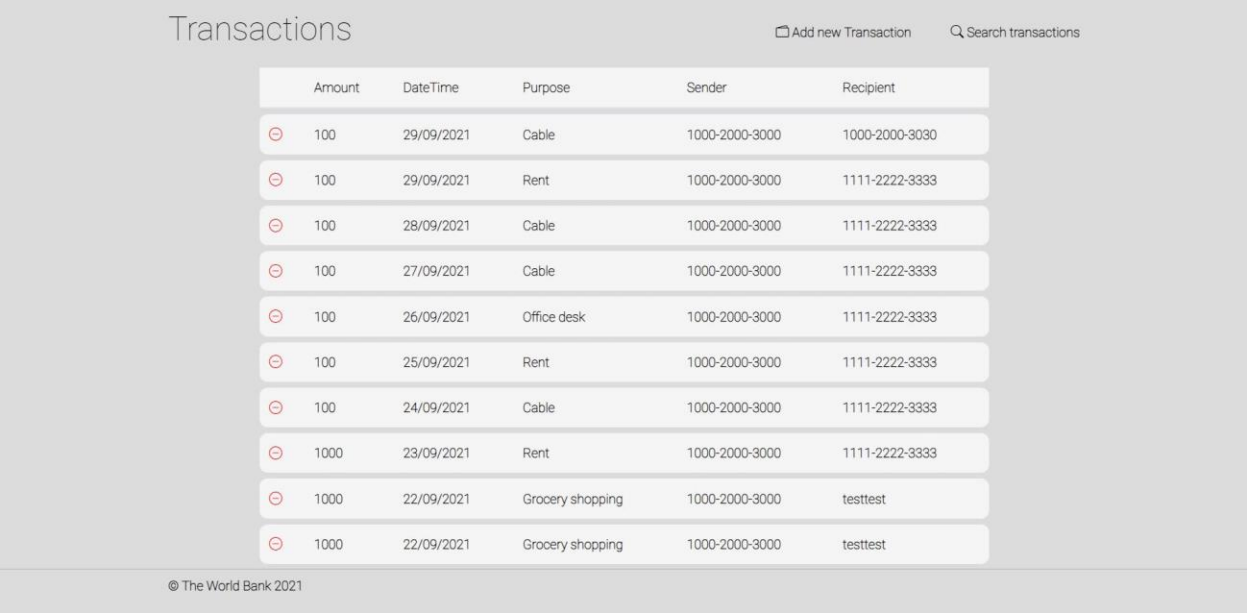


Слика 173 - Страница са детаљима трансакције

9. Корисник **позива** систем да обрише трансакцију. (АПСО)

Опис акције: Корисник притиском на дугме Delete transaction позива системску операцију ОбришиTransakciju.

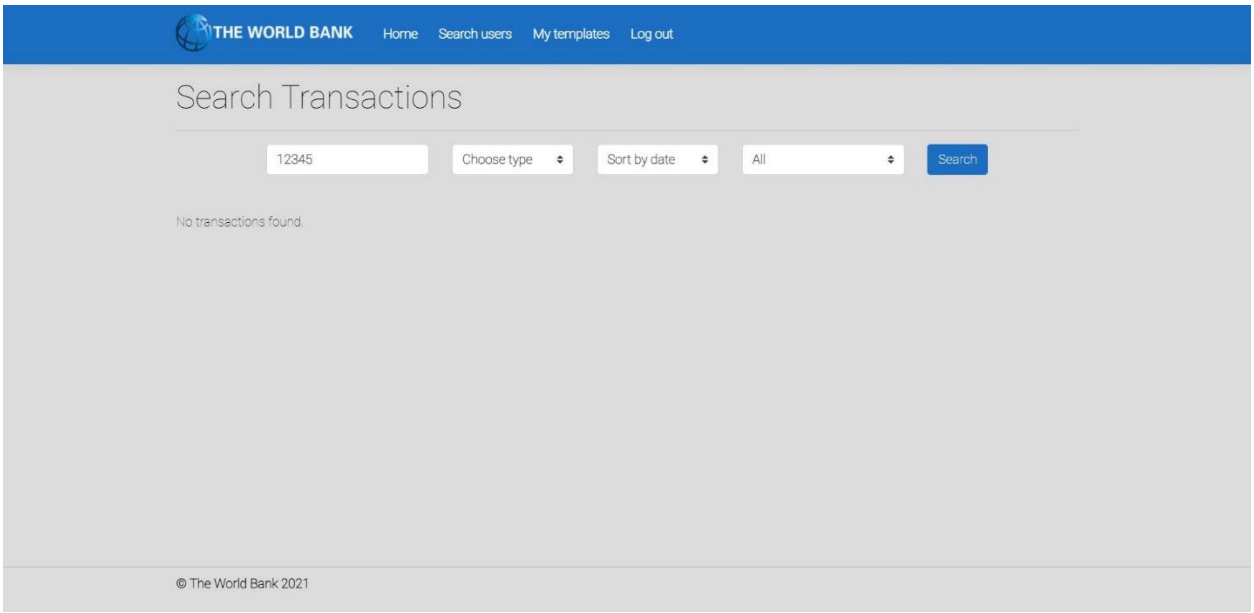
10. Систем **брише** трансакцију. (СО)
11. Систем **приказује** кориснику поруку: "Систем је обрисао трансакцију." (ИА)



Слика 174 - Приказ листе трансакција након брисања трансакције

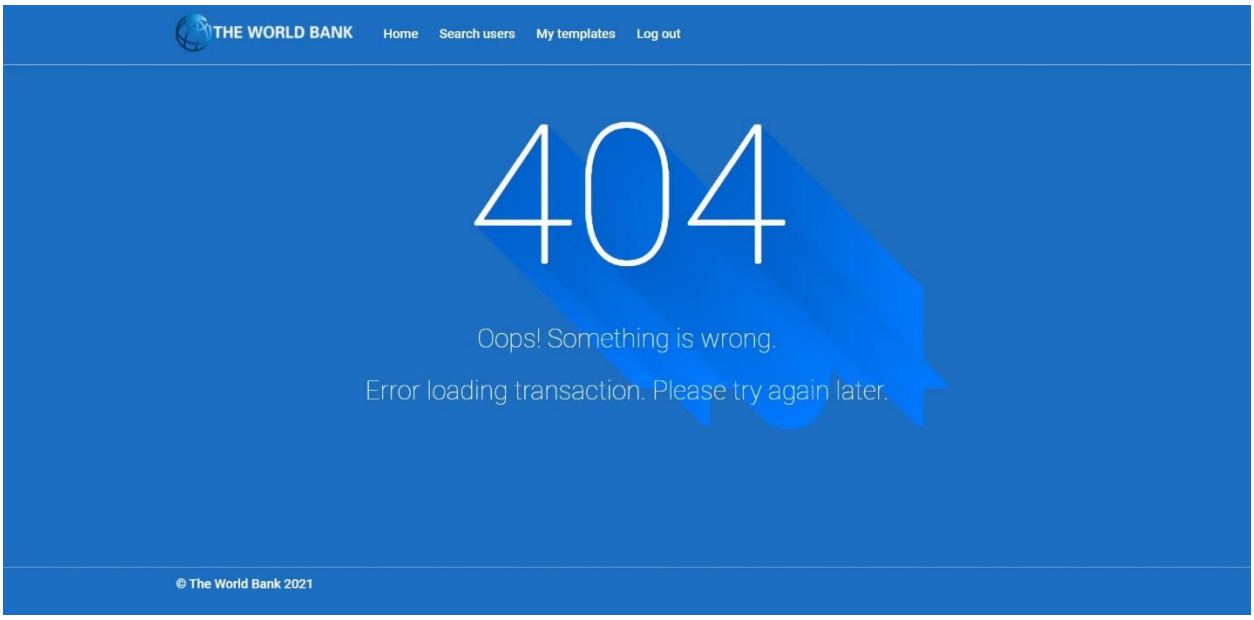
Алтернативна сценарија

4.1 Уколико систем не може да нађе трансакције он приказује кориснику поруку: “Систем не може да нађе трансакције по задатој вредности”. Прекида се извршење сценариа. (ИА)



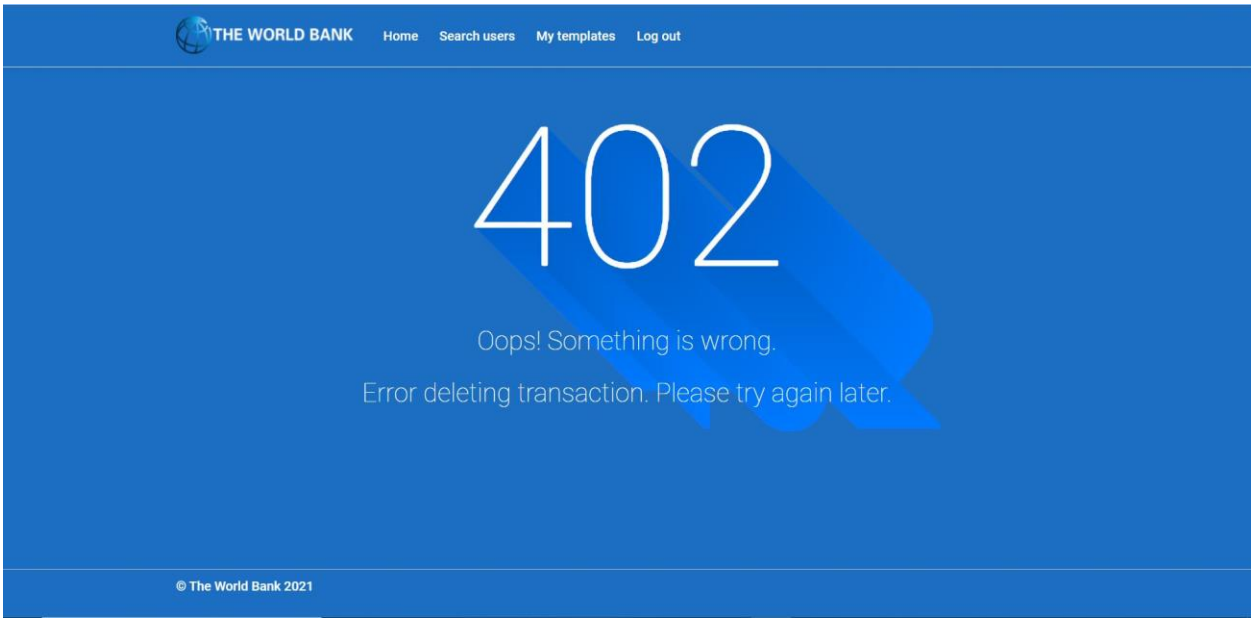
Слика 175 - Неуспешна претрага приликом брисања трансакције

8.1 Уколико систем не може да учита изабран трансакцију он приказује кориснику поруку “Систем не може да учита трансакцију“. Прекида се извршење сценариа (ИА)



Слика 176 - Порука о грешци приликом учитавања трансакције

11.1 Уколико систем не може да обрише трансакцију он приказује кориснику поруку “Систем не може да обрише трансакцију”. (ИА)



Слика 177 - Порука о грешци приликом брисања трансакције

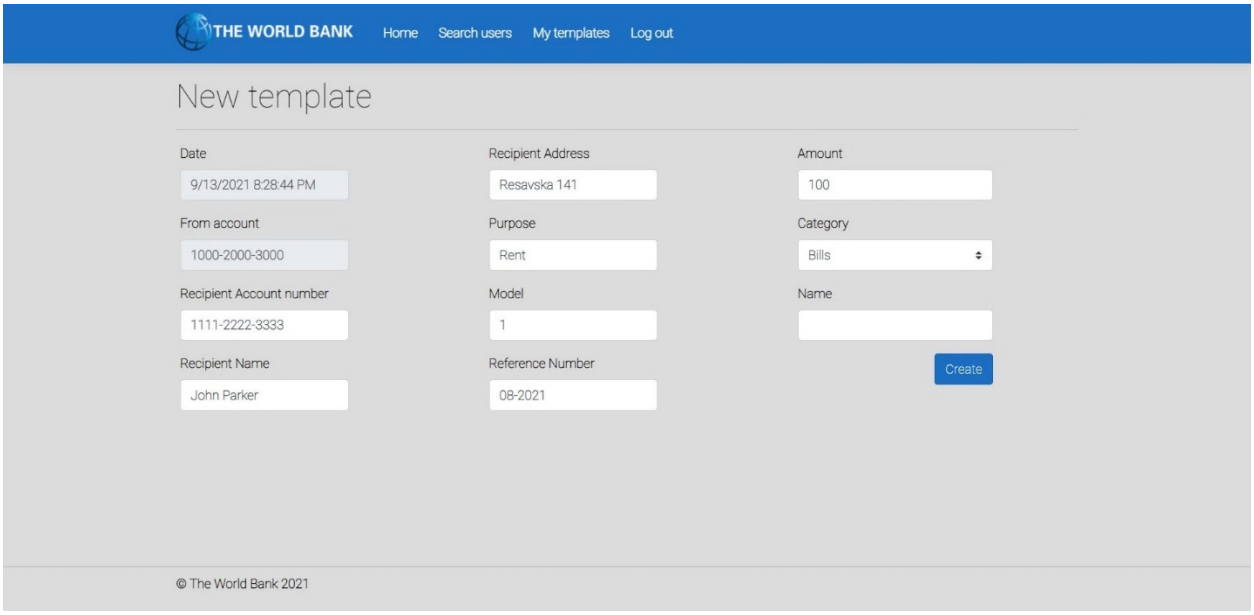
СК8: Случај коришћења – Креирање шаблона трансакције

Назив СК: Креирање шаблона трансакције

Актори СК: Корисник

Учесници СК: Корисник и систем (програм)

Предуслов: Систем је укључен и корисник је улогован под својом шифром. Систем приказује форму за рад са шаблоном. Учитана је листа категорија.



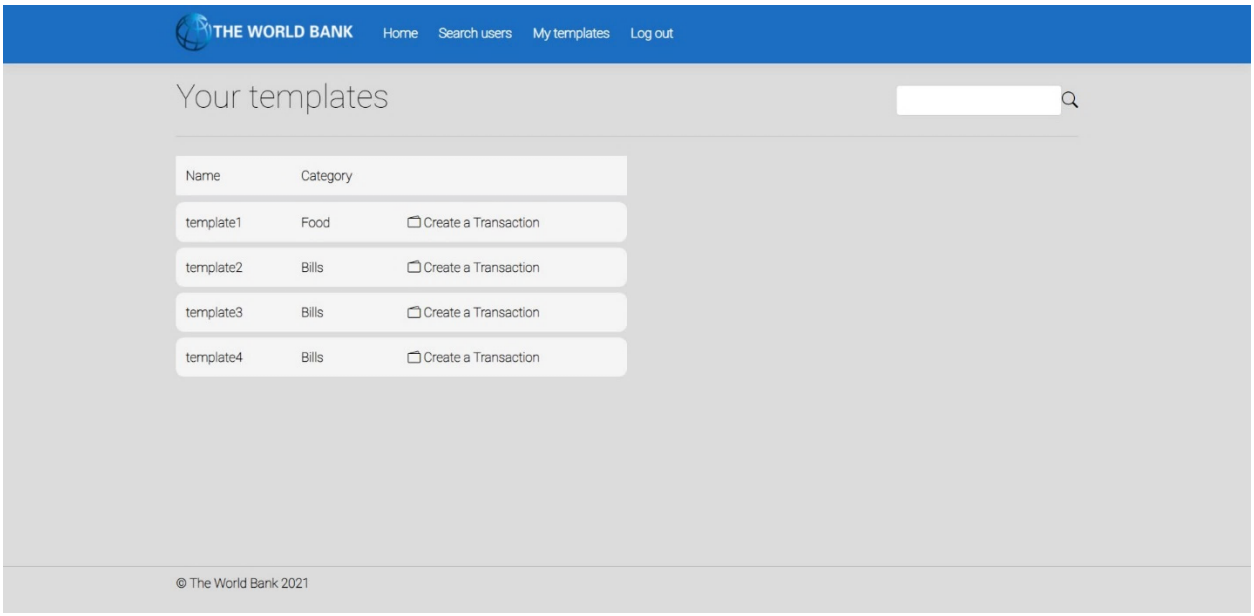
Слика 178 - Страница за креирање шаблона трансакције

Основни сценарио СК

1. Корисник **уноси** податке о шаблону. (АПУСО)
2. Корисник **контролише** да ли је коректно унео податке о шаблону. (АНСО)
3. Корисник **позива** систем да запамти шаблон. (АПСО)

Опис акције: Корисник притиском на дугме Create позива системску операцију ZapamtiŠablon.

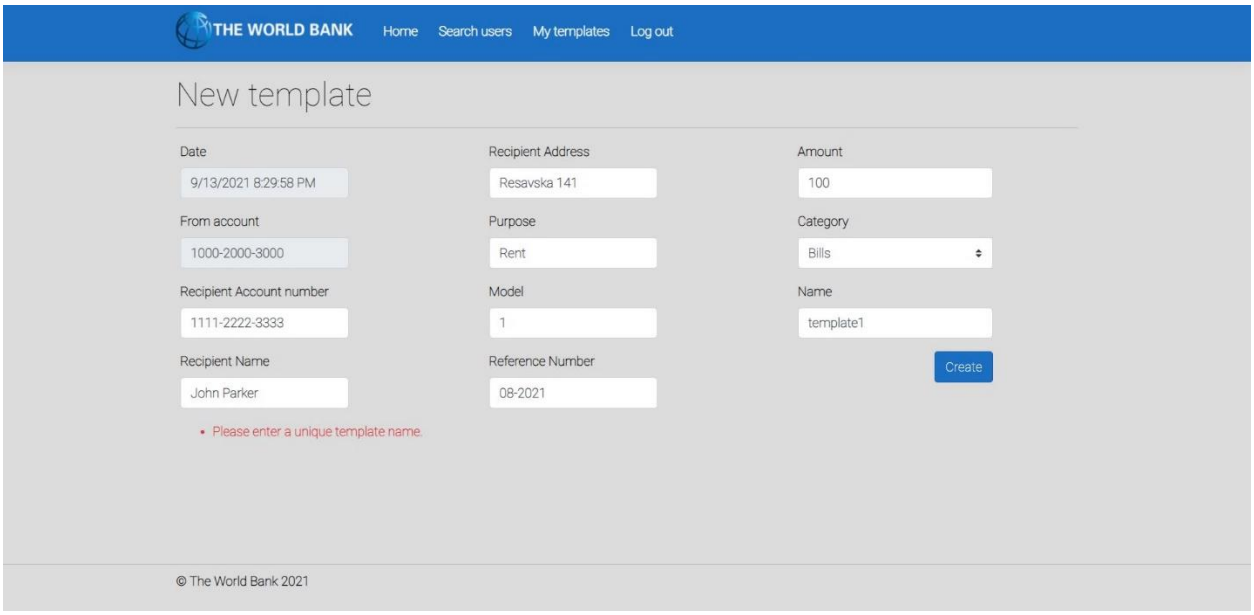
4. Систем **памти** шаблон. (СО)
5. Систем **приказује** кориснику запамћени шаблон и поруку: “Систем је запамтио шаблон“. (ИА)



Слика 179 - Приказ сачуваних шаблона

Алтернативна сценарија

5.1 Уколико систем не може да запамти податке о шаблону он приказује кориснику поруку “Систем не може да запамти шаблон”. (ИА)



Слика 180 - Приказ неуспешног креирања шаблона

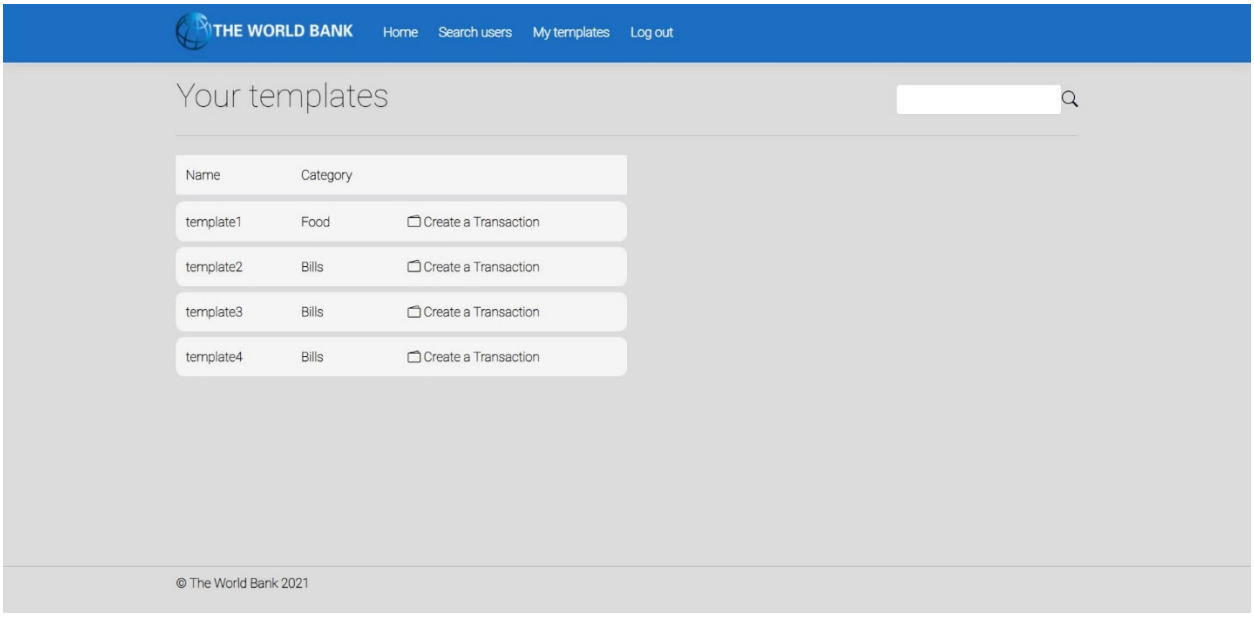
СК9: Случај коришћења – Измена шаблона трансакције

Назив СК: Измена шаблона трансакције

Актори СК: Корисник

Учесници СК: Корисник и систем (програм)

Предуслов: Систем је укључен и корисник је улогован под својом шифром. Систем приказује форму за рад са шаблоном. Учитана је листа шаблона и категорија.



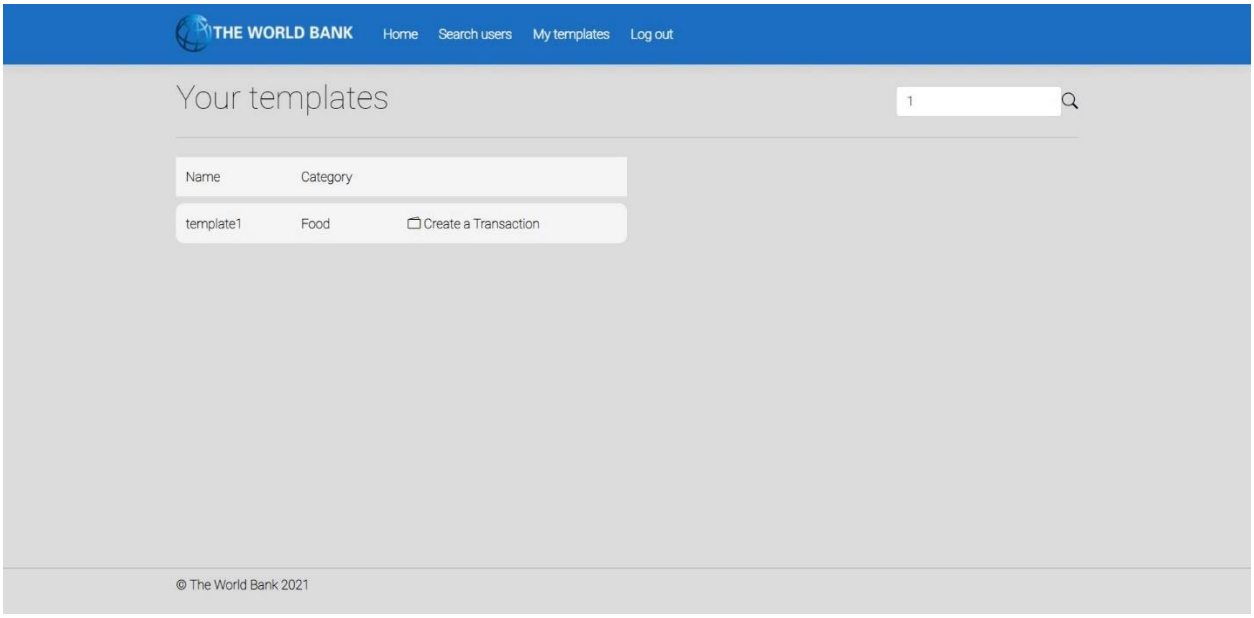
Слика 181 - Приказ корисникових шаблона

Основни сценарио СК

1. Корисник **уноси** вредност по којој претражује шаблоне. (АПУСО)
2. Корисник **позива** систем да нађе шаблоне по задатој вредности. (АПСО)

Опис акције: Корисник притиском на иконицу лупе позива системску операцију НађиШаblone.

3. Систем **тражи** шаблоне по задатој вредности. (СО)
4. Систем **приказује** кориснику шаблоне и поруку: “Систем је нашао шаблоне по задатој вредности”. (ИА)

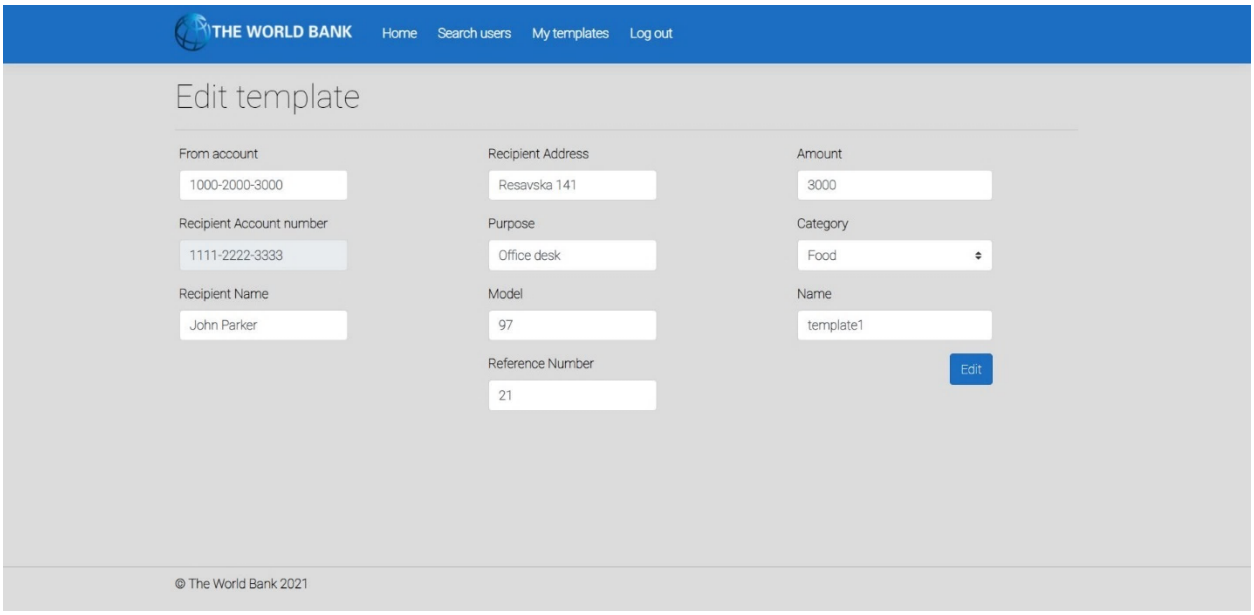


Слика 182 - Приказ успешне претраге шаблона

5. Корисник **бира** шаблон. (АПУСО)
6. Корисник **позива** систем да учита шаблон. (АПСО)

Опис акције: Корисник притиском на назив жељеног шаблона позива системску операцију УчитајШablon.

7. Систем **учитава** шаблон. (СО)
8. Систем **показује** кориснику податке о шаблону и поруку "Систем је учитао шаблон" (ИА)



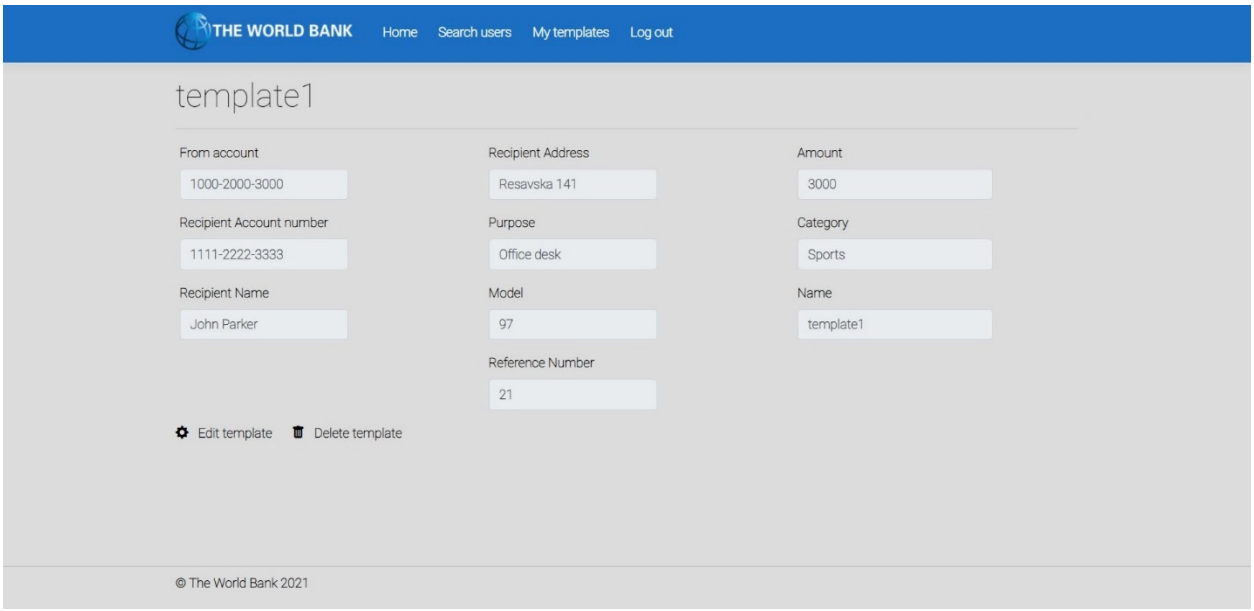
Слика 183 - Страница за измену шаблона трансакције

9. Корисник **уноси** (мења) податке о шаблону. (АПУСО)
10. Корисник **контролише** да ли је коректно унео податке о шаблону. (АНСО)

11. Корисник **позива** систем да запамти податке о шаблону. (АПСО)

Опис акције: Корисник притиском на дугме Edit позива системску операцију АжурирајШаблон.

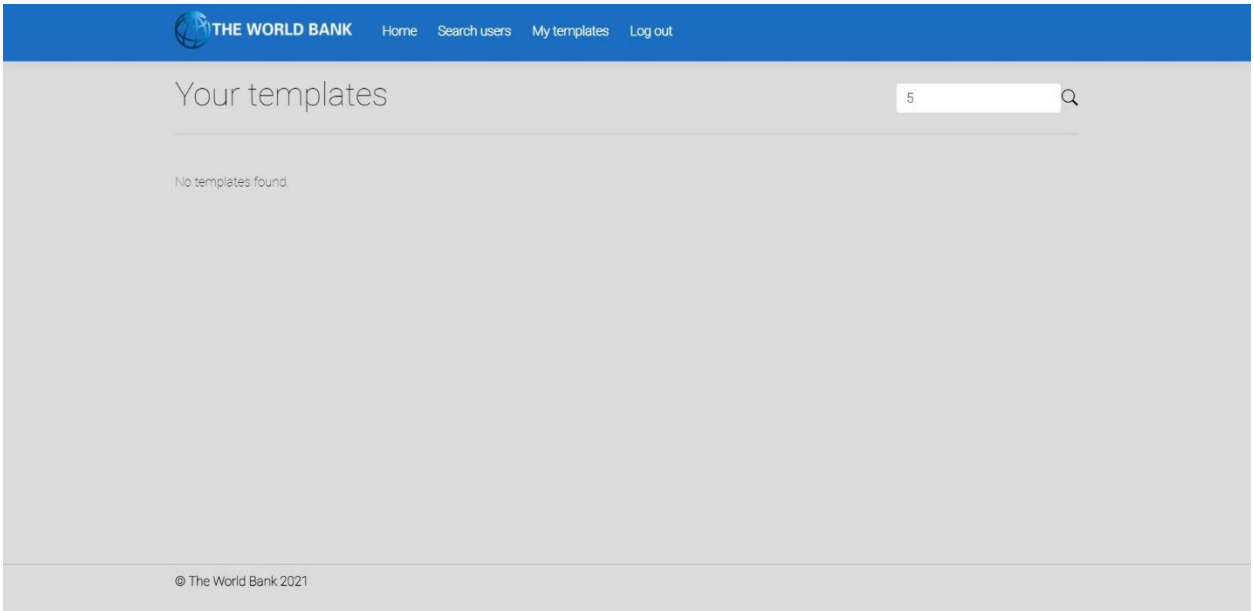
12. Систем **памти** податke о шаблону. (СО)
13. Систем **приказује** кориснику запамћени шаблон и поруку: “Систем је запамтио шаблон.” (ИА)



Слика 184 - Страница са детаљима сачуваног шаблона

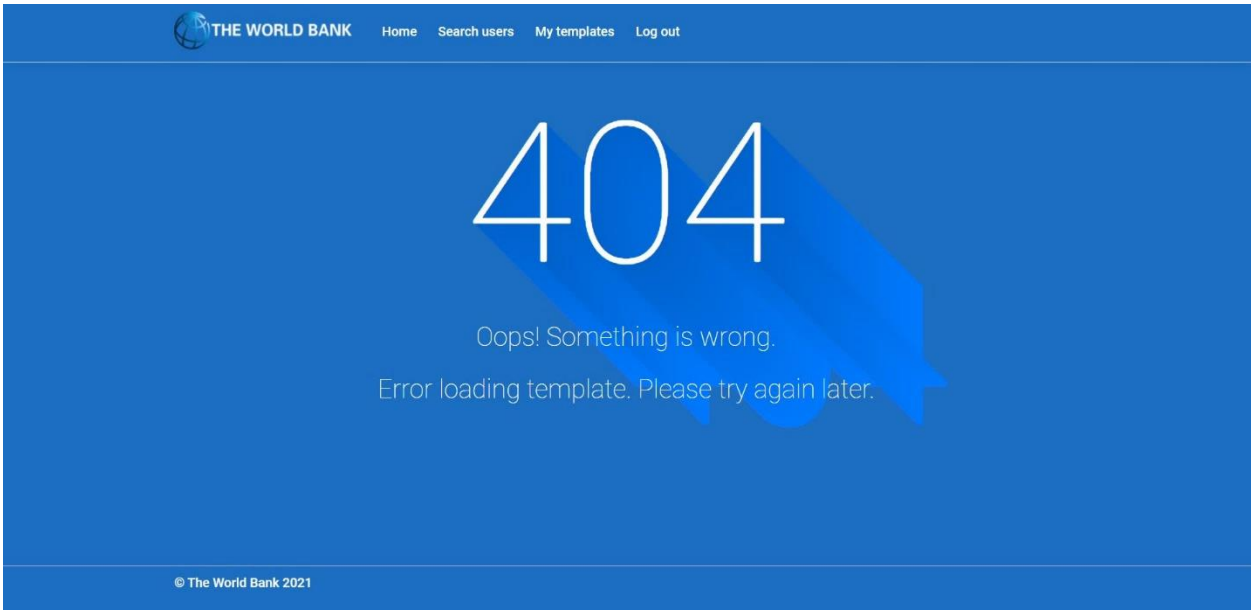
Алтернативна сценарија

4.1 Уколико систем не може да нађе шаблоне он приказује кориснику поруку: “Систем не може да нађе шаблоне по задатој вредности”. Прекида се извршење сценариа. (ИА)



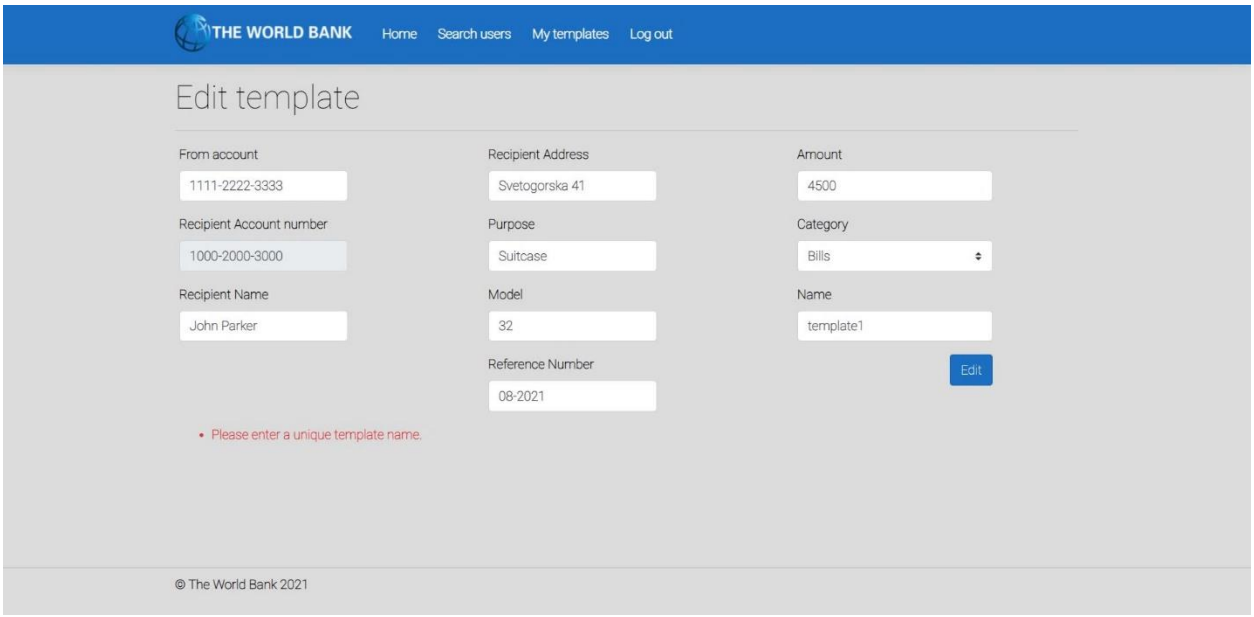
Слика 185 - Приказ неуспешне претраге шаблона

8.1 Уколико систем не може да учита шаблон он приказује кориснику поруку “Систем не може да учита шаблон”. Прекида се извршење сценариа. (ИА)



Слика 186 - Порука о грешци приликом учитавања шаблона

13.1 Уколико систем не може да запамти податке о шаблону он приказује кориснику поруку “Систем не може да запамти шаблон”. (ИА)



Слика 187 - Порука приликом неуспешне измене шаблона трансакције

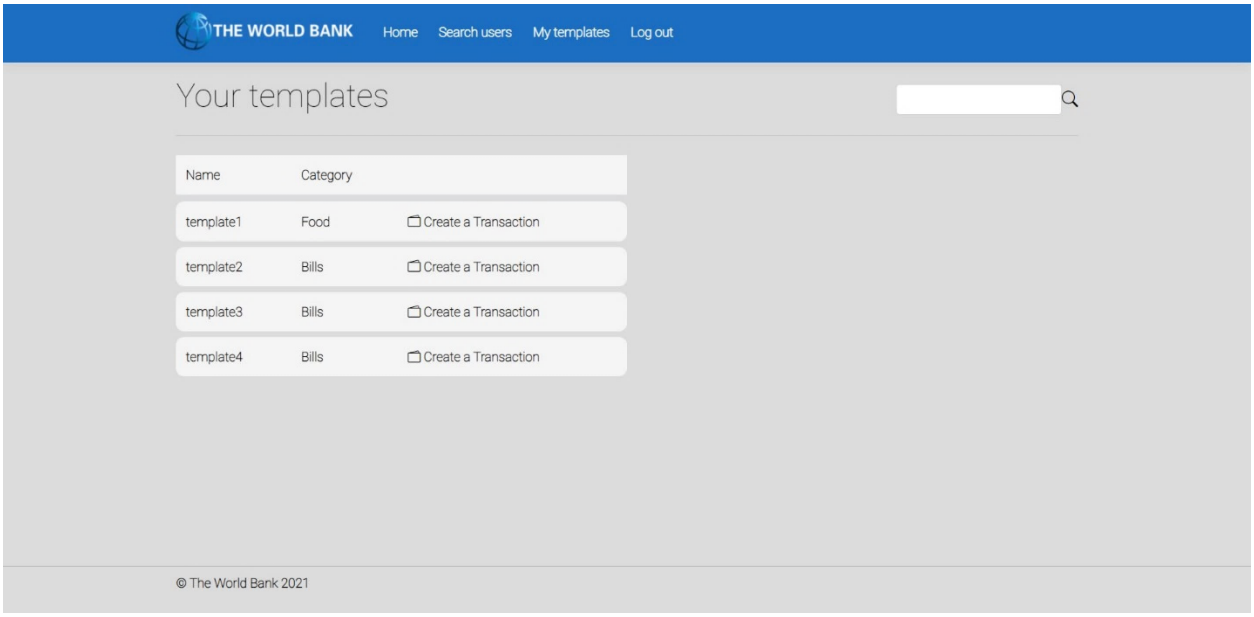
СК10: Случај коришћења – Брисање шаблона трансакције

Назив СК: Брисање шаблона трансакције

Актори СК: Корисник

Учесници СК: Корисник и систем (програм)

Предуслов: Систем је укључен и корисник је улогован под својом шифром. Систем приказује форму за рад са шаблоном. Учитана је листа шаблона.



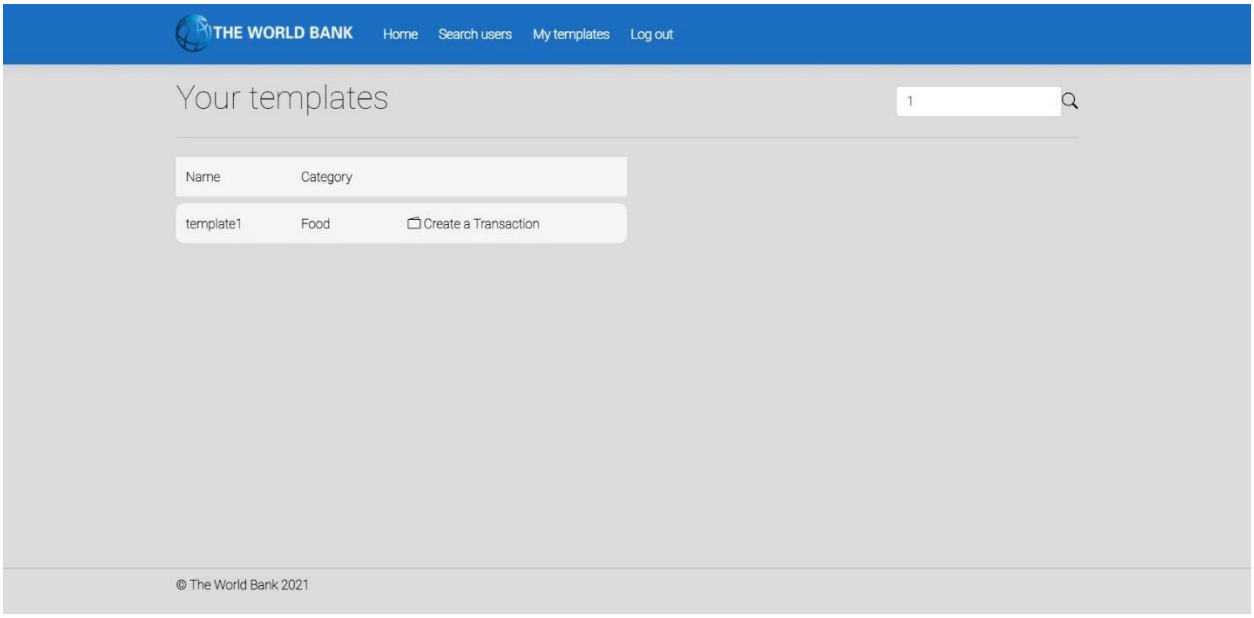
Слика 188 - Страница са корисниковим шаблонима

Основни сценарио СК

1. Корисник **уноси** вредност по којој претражује шаблоне. (АПУСО)
2. Корисник **позива** систем да нађе шаблоне по задатој вредности. (АПСО)

Опис акције: Корисник притиском на иконицу лупе позива системску операцију НађиШаblone.

3. Систем **тражи** шаблоне по задатој вредности. (СО)
4. Систем **приказује** кориснику шаблоне и поруку: “Систем је нашао шаблоне по задатој вредности”. (ИА)

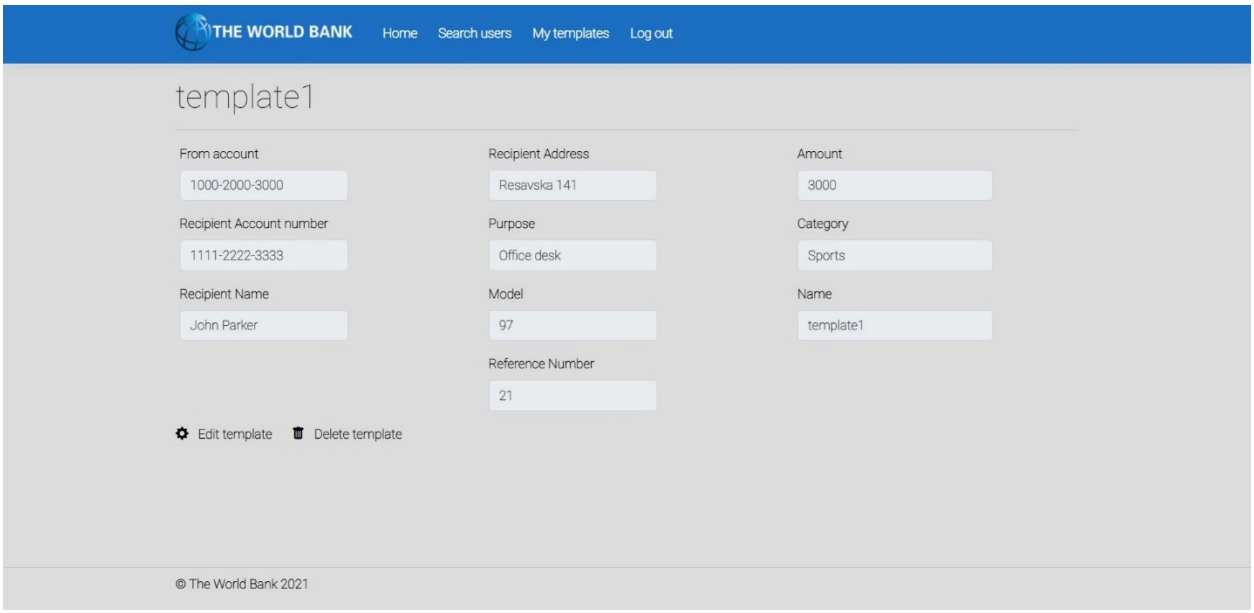


Слика 189 - Резултат претраге шаблона приликом брисања

5. Корисник **бира** шаблон који жели да обрише. (АПУСО)
6. Корисник **позива** систем да учита податке о одабраном шаблону. (АПСО)

Опис акције: Корисник притиском на назив жељеног шаблона позива системску операцију УчитајШablon.

7. Систем **учитава** податке о одабраном шаблону. (СО)
8. Систем **обавештава** корисника о успешном учитавању података поруком "Систем је учитао одабран шаблон" и приказује податке о шаблону. (ИА)

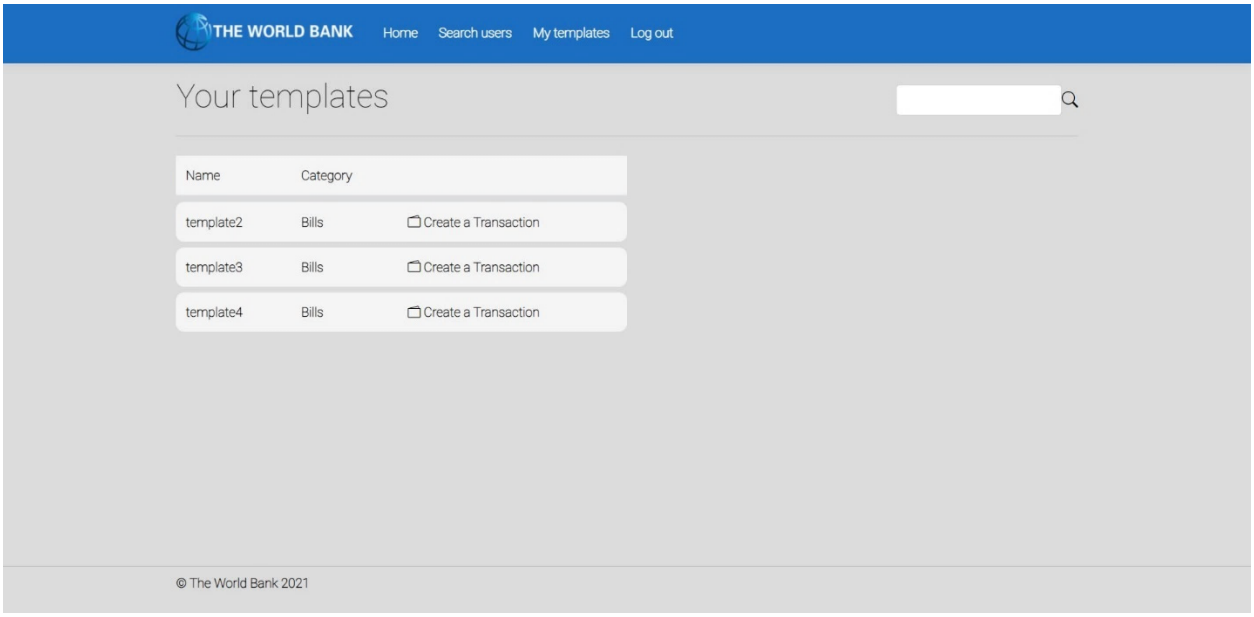


Слика 190 - Страница са детаљима шаблона трансакције

9. Корисник **позива** систем да обрише шаблон. (АПСО)

Опис акције: Корисник притиском на дугме Delete template позива системску операцију ОбришиШablon.

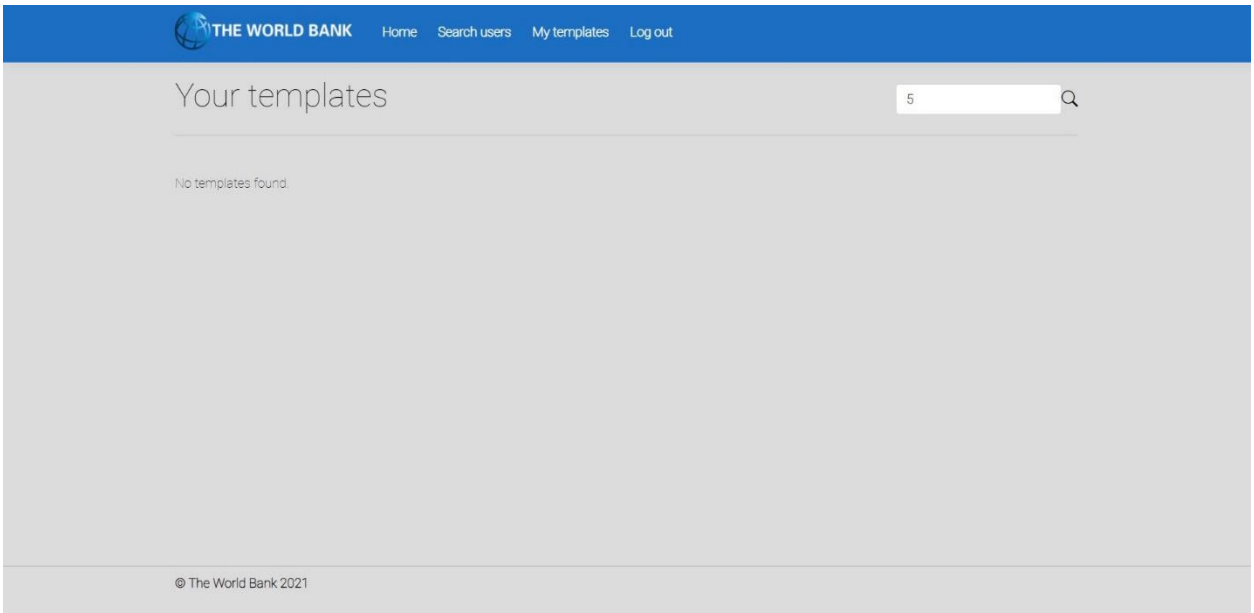
10. Систем **брише** шаблон. (СО)
11. Систем **приказује** кориснику поруку: “Систем је обрисао шаблон.” (ИА)



Слика 191 - Страница са корисниковим шаблонима након брисања

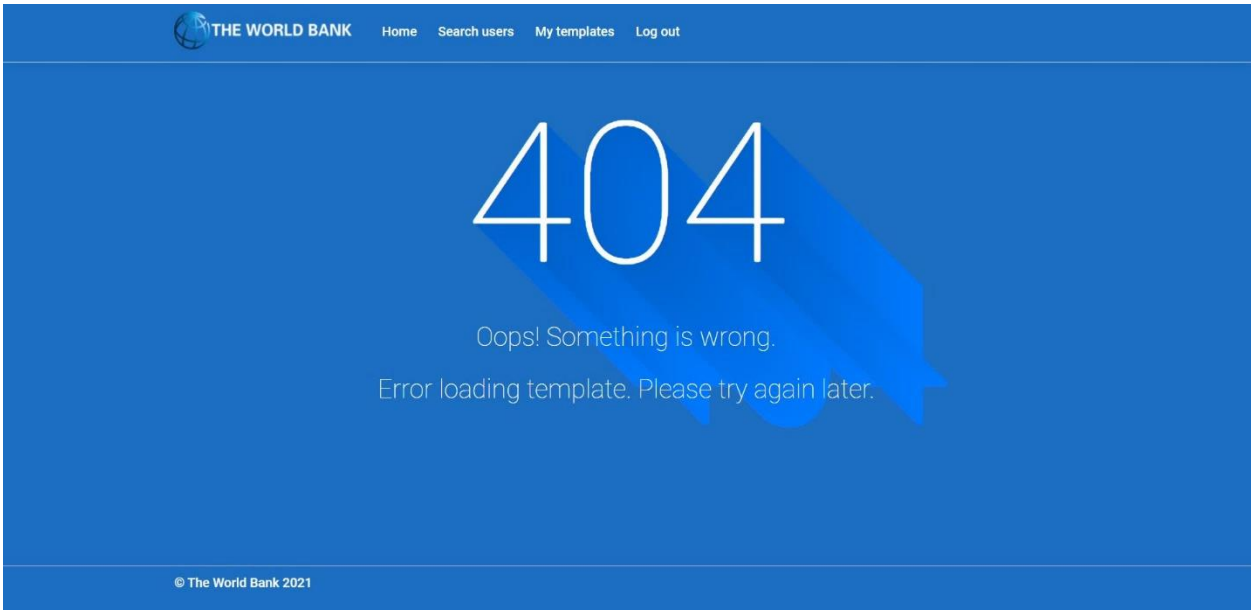
Алтернативна сценарија

4.1 Уколико систем не може да нађе шаблон он приказује кориснику поруку: “Систем не може да нађе шаблон по задатој вредности”. Прекида се извршење сценариа. (ИА)



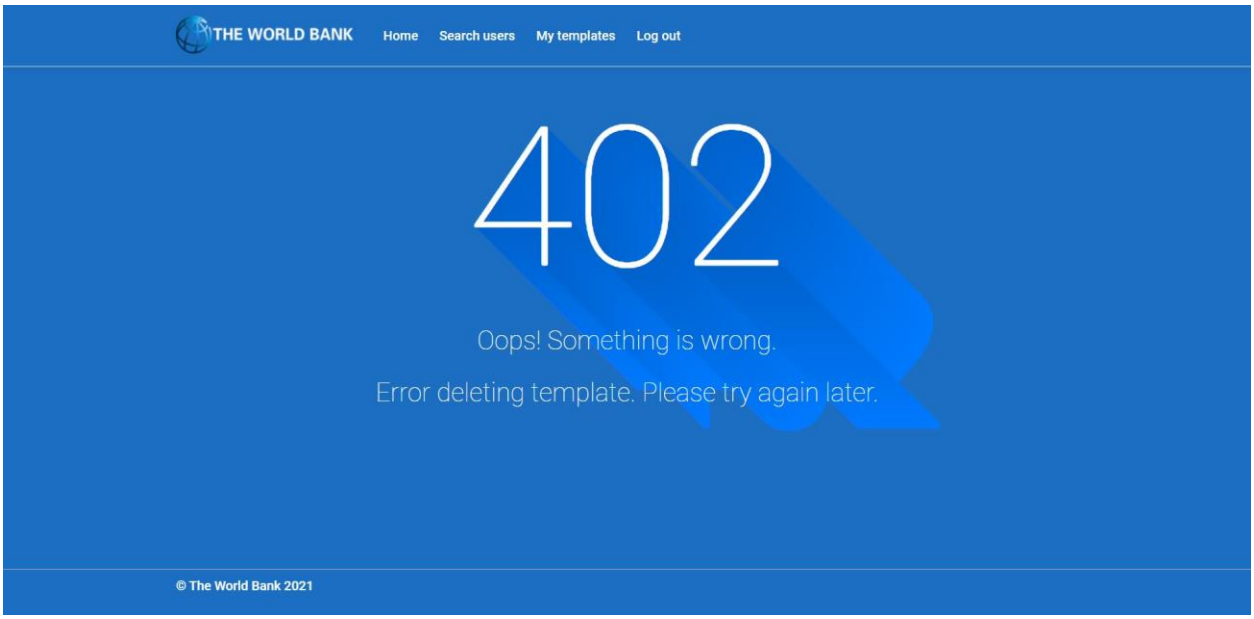
Слика 192 - Порука о неуспешној претрази шаблона трансакција

8.1 Уколико систем не може да учита изабран шаблон он приказује кориснику поруку “Систем не може да учита шаблон“. Прекида се извршење сценариа (ИА)



Слика 193 - Страница о грешци приликом учитавања шаблона трансакције

11.1 Уколико систем не може да обрише шаблон он приказује кориснику поруку “Систем не може да обрише шаблон”. (ИА)



Слика 194 - Страница о грешци приликом брисања шаблона

8. Тестирање

Извршено је ручно тестирање апликације. Њеним покретањем и уносом неодговарајућих и неисправних података, вршене су провере исправности имплементираних валидација. Такође, уношени су и валидни подаци, како би се тестирали сви приказани случајеви коришћења и утврдили њихови успешни исходи. Исправљени су сви откривени недостаци који су пронађени приликом тестирања.

Следећи корак у смеру тестирања апликације би било писање одговарајућих Unit тестова за све системске операције, као и за све методе доменских објеката. У ову сврху би било пожељно креирати још једну базу података исте структуре, која би пре и након покретања тестова имала идентичан скуп података.

9. Закључак

У овом раду приказана је заокружена целина развоја једне веб апликације коришћењем ASP.NET Core оквира. Након упознавања са коришћеним технологијама и .NET екосистемом, објашњене су фазе развоја софтверског система применом упрошћене Ларманове методе на студијском примеру апликације за евиденцију и управљање кућним буџетом. Развој најпре креће од идентификовања корисничких захтева који су приказани помоћу модела случајева коришћења. За уочене сценарије случајева коришћења се у фази анализе дефинише понашање и структура софтверског система – издваја се скуп системских операција и концептуални модел. Добијена пословна логика постаје део апликационе логике софтверског система, која заједно са складиштем података и корисничким интерфејсом чини класичну тронивојску архитектуру софтверског система. Апликација имплементира MVC патерн који се често користи код апликација са сложенијом пословном логиком.

Коришћењем оквира и патерна за развој софтвера, креирана је веб апликација која задовољава идентификоване корисничке захтеве и представља целину коју је могуће проширити. Коришћени оквири, објављени од стране Microsoft-a, интегрисани су и представљају чврсту основу веб апликација коју је могуће даље надоградити употребом различитих оквира и патерна. У овом раду објашњена је имплементација Repository и Unit of Work патерна..

Правци даљег развоја ове веб апликације било би тестирање идентификованих системских операција и унапређење респонзивности апликације, како би се пружило подједнако корисничко искуство и на мобилним уређајима. Такође, могуће је проширити скуп идентификованих захтева додавањем функционалности попут умрежавања корисника.

Писање овог рада и сам развој апликације пружио ми је значајно знање и искуство. Захтевао је разумевање основних концепта развоја веб апликација (енгл. web development) и коришћених технологија, пре свега програмског језика C#. Такође, рад на овој апликацији ми је омогућио спајање и повезивање више целина из различитих области информационих система и технологија које сам на теоријски и практичан начин прешао током основних студија на Факултету организационих наука. Од поменутих области издвојио бих: рад са базама података, пројектовање софтверског система, frontend технологије, коришћење git система за верзионисање кода и друго. Овај рад ме је мотивисао да наставим своје формално образовање на Факултету организационих наука, али и на даље усавршавање и примену свега наученог у пракси.

10. Литература

[1] Vlajić, S. (2015). Projektovanje softvera (skripta). Београд, Србија: Факултет организационих наука.

[2] Lock, A. (2019, август 27). Exploring the new project file, Program.cs, and the generic host. Преузето са https://andrewlock.net/exploring-the-new-project-file-program-and-the-generic-host-in-asp-net-core-3/ [Последњи приступ 14.9.2021]

[3] Anderson, R., Dykstra, T., Smith, S. (2019, децембар 5). App startup in ASP.NET Core. Преузето са https://docs.microsoft.com/en-us/aspnet/core/fundamentals/startup?view=aspnetcore-5.0 [Последњи приступ 14.9.2021]

[4] Entity Framework Core. (2020, септембар 20). Преузето са https://docs.microsoft.com/en-us/ef/core/ [Последњи приступ 14.9.2021]

[5] The Repository Pattern. (2010, април 27). Преузето са https://docs.microsoft.com/en-us/previous-versions/msp-n-p/ff649690(v=pandp.10)?redirectedfrom=MSDN [Последњи приступ 14.9.2021]

[6] Dykstra, T. (2013. јул 30) Implementing the Repository and Unit of Work Patterns in an ASP.NET MVC Application (9 of 10). Преузето са https://docs.microsoft.com/en-us/aspnet/mvc/overview/older-versions/getting-started-with-ef-5-using-mvc-4/implementing-the-repository-and-unit-of-work-patterns-in-an-asp-net-mvc-application [Последњи приступ 14.9.2021]

[7] Understanding and Implementing UnitOfWork pattern in ASP.NET Core. (2020, мај 28). Преузето са https://referbruv.com/blog/posts/understanding-and-implementing-unitofwork-pattern-in-aspnet-core [Последњи приступ 14.9.2021]

[8] Smith, S. (2020, фебруар 12). Overview of ASP.NET Core MVC. Преузето са https://docs.microsoft.com/en-us/aspnet/core/mvc/overview?view=aspnetcore-5.0 [Последњи приступ 14.9.2021]

[9] Milanovic, M. (2020, децембар 27). A Brief Walk Through .Net Ecosystem. Преузето са https://milan.milanovic.org/post/a-brief-walk-through-net-ecosystem/ [Последњи приступ 20.9.2021]

[10] Roth, D., Anderson, R., Luttin, S. (2020, април 17). Introduction to ASP.NET Core. Преузето са https://docs.microsoft.com/en-us/aspnet/core/introduction-to-aspnet-core?view=aspnetcore-5.0 [Последњи приступ 20.9.2021]

[11] De Oliveira, J. Bruchet, M. Learning ASP.NET Core 2.0. Бирмингем, Уједињено Краљевство: Pack Publishing Ltd.

[12] Metzgar, D. (2017). Exploring .NET Core with Microservices, ASP.NET Core and Entity Framework Core. Њујорк, Сједињене Америчке Државе: Manning Publications Co.

[13] A tour of the C# language. (2021, август 23). Преузето са https://docs.microsoft.com/en-us/dotnet/csharp/tour-of-csharp/ [Последњи приступ 20.9.2021]

[14] Inheritance - derive types to create more specialized behavior. (2021, мај 14). Преузето са https://docs.microsoft.com/en-us/dotnet/csharp/fundamentals/object-oriented/inheritance [Последњи приступ 20.9.2021]

[15] Object oriented programming. (2021, мај 14). Преузето са https://docs.microsoft.com/en-us/dotnet/csharp/fundamentals/object-oriented/ [Последњи приступ 20.9.2021]

[16] Access Modifiers (C# Programming Guide). (2020, август 3). Преузето са https://docs.microsoft.com/en-us/dotnet/csharp/programming-guide/classes-and-structs/access-modifiers [Последњи приступ 20.9.2021]

[17] Polymorphism. (2021, мај 14). Преузето са https://docs.microsoft.com/en-us/dotnet/csharp/fundamentals/object-oriented/polymorphism [Последњи приступ 21.9.2021]

[18] Albahari, J., & Albahari, B. (2014). C# 5.0 Mali referentni priručnik (1st ed.). Mikro knjiga.

[19] Delegates and lambdas. (2016, јун 20). Преузето са https://docs.microsoft.com/en-us/dotnet/standard/delegates-lambdas [Последњи приступ 21.9.2021]

[20] Brind, M. Welcome To Learn Entity Framework Core. Преузето са https://www.learnentityframeworkcore.com/ [Последњи приступ 21.9.2021]

[21] Creating and configuring a model. (2021, октобар 13). Преузето са https://docs.microsoft.com/en-us/ef/core/modeling/ [Последњи приступ 21.9.2021]

[22] Migrations Overview. (2021, октобар 28). Преузето са https://docs.microsoft.com/en-us/ef/core/managing-schemas/migrations/?tabs=vs [Последњи приступ 21.9.2021]

[23] Querying Data. (2019, октобар, 3). Преузето са https://docs.microsoft.com/en-us/ef/core/querying/ [Последњи приступ 21.9.2021]

[24] Bootstrap. (2017, јануар) Преузето са https://whatis.techtarget.com/definition/bootstrap [Последњи приступ 22.9.2021]

[25] Ouellette, A. (2021, јул 8). What is Bootstrap: A Beginner's Guide. Преузето са https://careerfoundry.com/en/blog/web-development/what-is-bootstrap-a-beginners-guide/ [Последњи приступ 22.9.2021]

[26] An Introduction to JavaScript (2021, јун 13). Преузето са https://javascript.info/intro [Последњи приступ 22.9.2021]

[27] JavaScript – Overview. Преузето са https://www.tutorialspoint.com/javascript/javascript_overview.htm [Последњи приступ 22.9.2021]

[28] Pandya, J. (2020, новембар 4). What Is jQuery Used For? Преузето са https://www.learningjquery.com/2020/11/what-is-jquery-used-for [Последњи приступ 22.9.2021]

[29] What is Ajax? Преузето са https://www.ibm.com/docs/en/rational-soft-arch/9.6.1?topic=page-asynchronous-javascript-xml-ajax-overview [Последњи приступ 22.9.2021]