Универзитет у Београду

Електротехнички факултет



Реализација квиза за два играча у савременим веб технологијама

Дипломски рад основних академских студија

|  |  |
| --- | --- |
| Ментор: | Кандидат: |
| Др Дражен Драшковић, доцент | Вељко Марковић 137/2013 |

Београд, септембар 2021.

Садржај

[Садржај i](#_Toc83225831)

[1. Увод 1](#_Toc83225832)

[2. Анализа проблема 2](#_Toc83225833)

[3. Захтеви за реализацију система 4](#_Toc83225834)

[3.1. Корисници 5](#_Toc83225835)

[3.1.1. Администратор 5](#_Toc83225836)

[3.1.2. Супервизор 6](#_Toc83225837)

[3.1.3. Учесник квиза 6](#_Toc83225838)

[3.1.4. Гост 6](#_Toc83225839)

[3.2. Игра 7](#_Toc83225840)

[3.2.1. Дневна игра 7](#_Toc83225841)

[3.2.2. Игра учесник против учесника 9](#_Toc83225842)

[3.3. Опис коришћених технологија 10](#_Toc83225843)

[4. Реализација система 12](#_Toc83225844)

[4.1. Опис базе података 12](#_Toc83225845)

[4.2. Преглед архитектуре система 14](#_Toc83225846)

[4.2.1. Серверска апликација 14](#_Toc83225847)

[4.2.2. Контролери 15](#_Toc83225848)

[4.2.3. Сервиси 17](#_Toc83225849)

[4.2.4. Контекст базе података, модели и миграције 17](#_Toc83225850)

[4.2.5. SignalR Hub 19](#_Toc83225851)

[4.2.6. Клијентска апликација 20](#_Toc83225852)

[4.3. Главни имплементациони изазови 21](#_Toc83225853)

[4.3.1. Аутентикација и ауторизација 21](#_Toc83225854)

[4.3.2. Додавање профилне слике корисника 24](#_Toc83225855)

[4.3.3. Временски интервали у игри учесник против учесника 24](#_Toc83225856)

[4.3.4. Чекање играча на потез 26](#_Toc83225857)

[4.3.5. Закључавање табеле 28](#_Toc83225858)

[5. Опис рада система 30](#_Toc83225859)

[5.1. Јавно доступан део система 30](#_Toc83225860)

[5.2. Администратор 33](#_Toc83225861)

[5.3. Супервизор 35](#_Toc83225862)

[5.4. Учесник квиза 37](#_Toc83225863)

[5.5. Дневна игра 40](#_Toc83225864)

[5.6. Игра учесник против учесника 45](#_Toc83225865)

[6. Закључак 53](#_Toc83225866)

[Литература 54](#_Toc83225867)

[Списак слика 55](#_Toc83225868)

1. Увод

Глобална експанзија интернета и све већи број корисника неминовно је наметнуо потребу за коришћењем интернет апликација у разне сврхе. Од разних облика пословања све до индустрије игара, све је већа потреба за реализацијом интернет апликација које за циљ имају да олакшају многе животне сегменте, као и да помогну корисницима да се забаве на разне начине.

Интернет апликације представљају апликације којима се приступа преко мреже. Интернет апликације су у данашње време све популарније збо свеприсутности интернет претраживача и једноставности коју они нуде корисницима. На почетку интернета, свака интернет страница достављала се клијенту као статички документ. Године 1995. компанија *Netscape* реализовала је скриптинг функционалност која се извршава на страни клијента, познатији као програмски језик *Javascript*. Дакле, уместо слања података на сервер да би генерисали целу страницу, уграђене скрипте са преузете странице су могле да обављају разне задатке, као што су валидација уноса у форме или приказивање или скривање неких делова страница. Напредак интернет технологија наставио се 1996. године када је имплементирана векторска графика која се може додати интернет претраживачима као додатак што је омогућило приказ разних анимација на интернет страницама. Потом је 1999. године концепт интернет апликација уведен у програмски језик *Јава*, а затим и у неке друге програмске језике и алате. Експанзија траје већ дуги низ година и константно напредује невероватном брзином.

Ради лакшег разумевања модерних интернет апликација и концепата квиза реализованих кроз исте те апликације, као и интерактивних игара генерално, циљ овог рада јесте имплементација квиза за два играча у савременим интернет технологијама. Корисник ће моћи блиско да се упозна са свим модерним концептима развоја овог типа интернет апликације. Биће детаљно објашњен развој клијентског и серверског дела система, као и базе података. Такође, комуникација између кључних софтверских компонената овог система ће бити детаљно анализирана.

У другом поглављу биће анализиран проблем интерактивних игара. Овај проблем биће сагледан кроз разне примере интернет апликација, као и неких мобилних апликација.

У трећем поглављу овог рада приближићемо саме захтеве за реализацију система. Биће описани главни ентитети у систему, као и сви типови корисника. Осим тога, биће дефинисан појам игре и који све режими рада игре постоје у систему.

У четвртом поглављу биће концизно објашњена реализација целог система. Биће анализирана софтверска архитектура система, као и њене основне компоненте као што су клијентска апликација, серверска апликација, база података итд. Осим тога, биће предочени главни имплементациони изазови са којима се требало суочити приликом реализације.

У петом поглављу биће достављен опис рада целог система кроз мноштво корисничких екрана и објашњења.

У последњем, шестом поглављу, дат је кратак закључак о имплементираном решењу, које су његове предности и могућности примене у пракси и раду.

1. Анализа проблема

Данашње време одликују брз проток велике количине информација и експоненцијални развој технологија. Дигитална трансформација дефинише процесе који омогућавају свим аспектима људског деловања у друштву да искористе предности савремених технологија за лакше, брже и ефикасније обављање својих активности и достизање циљева.

Многе ствари је данас могуће обавити брзо и лако онлајн. Поручивање хране, заказивање лекарских прегледа, плаћање рачуна, куповина различитих артикала путем интернета, представљају само неке од примера где се јасно може видети дигитализација на делу.

Форма квиза у оваквом систему је веома лако нашла своју примену. Конкретна питања и задаци наводе да се из гомиле информација исфилтрира суштина и помогне кориснику да издвоји садржај који треба да упамти и репродукује. Питања и задаци који се све чешће могу наћи на крају специфичних лекција и едукативних садржаја олакшавају кориснику да лакше провери да ли је усвојио садржај и усмеравају га на најбитније ставке из приложених информација. Овај формат се све чешће може срести и као званичан начин оцењивања знања у школама и на факултетима.

Интеракција јесте још једна ствар која чини квизове веома популарним форматом у разним областима деловања. Губитак концентрације и заинтересованости је данас честа појава код читања, гледања или слушања обимних садржаја и управо је позивање корисника на неки вид акције, гласањем, давањем одговора и слично, оно што кориснике форсира да остану фокусирани. На популарним друштвеним мрежама се поред сажетог садржаја често може срести и овај вид позива на интеракцију.

Постоје разне врсте и форме квизова као што су:

1. Квизови личности
2. Квизови знања
3. Музички квизови
4. Загонетни квизови
5. Групни квизови

О томе колико је овај формат популаран сведочи и број разних квизова у свету, као и то од када квизови датирају. Многобројни су телевизијски квизови, а све је више и онлајн квизова које могу играти сви корисници који имају приступ интернету и било ком електронском уређају који се може повезати на мрежу.

Као један од интересантнијих квизова на нашим просторима се издвоји квиз „*Слагалица*” који је и био инспирација за овај рад. „*Слагалица*” је квиз који се иницијално појавио као телевизијски квиз и емитује се већ 30 година на националној телевизији. Интересантан је свим генерацијама. Тајна дуговечности овог квиза би могла да лежи у разноликости игара и аспеката знања и вештина које се од самог такмичара очекују. Поред провере општег знања, на тесту су и логичко размишљање, вокабулар, спретност и брзина. Овај квиз подстиче и унапређује размишљање, меморију, комбиновање, закључивање, логику, такмичарски дух и општу социјализацију.

Игре као што су „*Слагалица*” и „*Мој Број*” преузете су из квиза „*Бројке и слова*” уз мање измене. У квизу „*Бројке и слова*” такође учествују два играча која су која склапају најдуже речи од понуђених слова и покушавају да израчунају најприближнију вредност задатом броју користећи основне математичке операције и насумично одабране бројеве.

Игра „*Скочко*” је направљена по угледу на друштвену игру под називом „*Mastermind*”. То је игра за два играча у којој је циљ да сваки играч погоди комбинацију боја (тајну шифру) коју је поставио други играч. Ова шифра се састоји од 4 обојене фигурице. Разлика код игре „*Слагалица*” у односу на ову игру је то што комбинацију „*фигурица*” одређује компјутер, а не други играч.

Игра „*Асоцијације*” преузета је у потпуности из квиза „*Квискотека*”. У овој игри два такмичара покушавају да погоде тачно решење отварајући поља иза којих се налазе речи, појмови или синтагме. У овој игри велику улогу игра опште знање такмичара, али и способност да се сагледа шира слика и пронађе повезаност из наизглед некад неспојивих појмова.

Било је питање времена када ће се овај квиз у општој интернет употреби на мобилним телефонима и рачунарима. Прва верзија овог квиза за мобилне уређаје омогућавала је такмичарима да се повежу преко „*bluetooth*” технологије и на тај начин одиграју дуел. Поред тога, постојала је и опција да играч игра партију сам са собом и на тај начин „вежба” за дуел.

Недуго потом, ова игрица се појавила и на друштвеној мрежи под називом Фејсбук (ориг. *Facebook*) где је постала потпуни феномен по броју корисника ове платформе који активно играју игру. И дан данас је овај вид играња игре „*Слагалица*” најпопуларнији међу млађом популацијом која велику количину времена проводи на интернету. Такмичари у овој форми игре могу да играју против било кога из листе пријатеља, а такође се могу и спојити са апсолутно случајним корисницима који нису у листи пријатеља. У првим фазама развоја ове игре на платформи Фејсбук било је могуће само на рачунарима који су представљали клијентски уређај. Као логичан след развоја технологија, појавила се иста апликација у форми за мобилне уређаје (мобилне телефоне и таблете). Прво је реализована игра за мобилне уређаје који раде на *Android* оперативном систему, а потом и за уређаје који раде на платформи *iOS.* Могуће је пријавити се на систем преко Фејсбук налога, а уколико корисник нема профил на поменутој друштвеној мрежи, омогућена је регистрација и пријава преко адресе електронске поште.

Ова тема пружа могућност да се стечено знање кроз учење и рад искористи за реализацију једне овакве игре. Неопходно је било савладати разне изазове и истражити могућа решења како би се један овакав концепт направио функционалним. Реч је о игри која је толико разнолика, у којој свако може увидети своје слабости, а тако и јаче стране, кроз коју се играчи и забављају, али и у којој је неопходно размишљање и учење. Ово је игрица у којој је време ограничено и не задржава корисника да мора да проведе сате слободног времена играјући игре и седети за рачунаром или гледајући у мобилни уређај, и при томе трошити своје нерве и штетити свом здрављу. Крајњи продукт имплементације овог система представља здрав и користан вид забаве за кориснике. Све ово је била мотивација и инспирација за реализацију једног оваквог пројекта.

1. Захтеви за реализацију система

У овом поглављу биће описане функционалности које је било неопходно реализовати у оквиру самог система. Реализована је веб апликација која представља систем за играње квиза *Слагалица*, који се састоји од пет игара. Називи игара су: „Слагалица“, „Мој број“, „Скочко“, „Спојнице“ и „Асоцијације“.

На почетној страни овог система омогућено је корисницима пријављивање на исти. Након исправно унетих података корисник има могућност да настави рад са остатком система. Пријава на систем је реализована путем форме где корисник након исправно унетих података (адреса електронске поште и лозинка) може наставити рад са остатком система. У случају било каквог проблема или грешке приликом пријаве на сам систем корисник добија одговарајућу поруку са описаним проблемом. Поред самог пријављивања на систем корисник има могућност промене лозинке и регистрацију новог корисника.

Регистрација новог корисника отвара форму за регистрацију која садржи следећа поља:

* 1. Име
  2. Презиме
  3. Адресу електронске поште
  4. Занимање
  5. Корисничко име
  6. Лозинку
  7. Потврду лозинке (односно исту вредност као код Лозинке)
  8. Пол
  9. Датум рођења
  10. Додавање профилне слике (максимално 300х300 пиксела)

Валидације које су имплементиране за претходно наведена поља: корисничко име мора бити јединствено у систему, лозинка корисника мора да испуњава следеће критеријуме: мора имати најмање 8 карактера, а највише 17, минималан број великих слова у лозинци је 1, минималан број малих слова је 3, минималан број нумерика је 1, минималан број специјалних знакова је 1, почетни карактер мора бити мало слово или велико слово или нумеричка вредонст, максималан број узастопних карактера је 3, име, презиме, корисничко име, пол, датум рођења, лозинка, занимање, адреса електронске поште представљају све неопходне параметре у форми. Додавање профилне слике представља опциони параметар у форми. Такође провера регуларности адресе електронске поште је укључена у валидацију.

Након исправно унетих параметара за регистрацију и слања захтева креира се нови регистрациони захтев који потом пристиже у панел администратора. Након што администратор прихвати регистрациони захтев, корисник постаје учесник система.

Осим стране за регистрацију на почетном екрану је могуће променити лозинку уколико корисник већ постоји у систему. Промена лозинке се такође врши путем форме. У форми за промену лозинке постоје три поља и то су: адреса електронске поште, стара лозинка и нова лозинка. Уколико корисник са унетом адресом електронске поште не постоји на екрану се исписује одговарајућа порука. Такође, уколико се унесе погрешна стара лозинка за корисника на екрану се исписује порука. Када су сви критеријуми за промену лозинке испуњени и корисник успешно промени лозинку, преусмерава се на страну за пријаву на систем.

Уколико корисник прескочи регистрацију или промену лозинке, могуће је приступити систему као нерегистровани корисник - гост.

* 1. Категорије корисника

У систему квиза који је реализован као веб апликација препознају се три категорије регистрованих корисника:

1. Администратор,
2. Супервизор,
3. Учесник квиза.

Осим ових категорија, постоји и нерегистровани корисник, односно гост.

* + 1. Администратор

Улога администратора јесте да администрира корисничке налоге, као и да води рачуна о дневним играма. Омогућено је да администратор представља корисника који не може да игра игре и не може да буде супервизор.

Када се пријави на систем, администратор види табелу свих корисника система. У том прегледу администратор види и све пристигле захтеве за регистрацију нових корисника који желе да учествују у квизу. Он може да управља пристиглим захтевима за регистрацију тако што има могућност да те захтеве одбије или прихвати. Уколико захтев за регистрацијом буде одбијен од стране администратора, учесник мора поново да пошаље нови захтев који ће администратор накнадно обрадити. Уколико администратор прихвати захтев за регистрацију, корисник који је послао захтев аутоматски постаје валидан учесник квиза и од тог момента може неометано да се пријави на систем и почне да игра игре.

Такође, администратор је уједно одговоран и за активирање и деактивирање корисничких налога. Када одобри захтев за регистрацију неког корисника, тај корисник аутоматски постаје активан корисник. Међутим, обезбеђено је и да администратор деактивира налоге, уколико из било ког разлога администратор сматра да одређени кориснички налог треба да буде деактивиран. Исто тако, деактивирани кориснички налог, администратор може поново активирати када се испуне сви критеријуми за то.

Осим управљања корисничким налозима, администратор је одговоран и за администрацију игре дана. Он може да одабере датум у будућности за који не постоји дефинисана игра дана и да за тај дан одабере игру за „Спојнице“ и за „Асоцијације“ које ће бити детаљно описане у наредним поглављима. У том контексту администратор треба да одабере групу већ дефинисаних појмова, који су претходно дефинисани од стране Супервизор корисника за игру „Спојнице“, као и једну групу већ дефинисаних појмова од стране Супервизора за игру „Асоцијације“. То значи да ће те игре бити игре које ће сви играчи играти тог дана у дневној игри. Што се тиче осталих игара, рачунар насумично генерише параметре (слова код игре „Слагалица“, бројеве код игре „Мој број“, комбинације симбола код игре „Скочко“).

* + 1. Супервизор

Други тип корисника јесте Супервизор. Супервизор је тип корисника који је задужен за рад са свим аспектима игара које сам учесник не види. Улога супервизора јесте да унесе базу речи које ће бити сматране за исправне и које ће бити бодоване учесницима приликом самог играња игре „Слагалица“. Супервизор може редно додавати речи за игру „Слагалица“ тако што је омогућено да отвори дијалог за додавање појединачне речи. Осим могућности уноса појединачне речи супервизор може унети скуп речи преко документа.

Друго задужење супервизора јесте да уноси нове игре за игру „Спојнице“. То подразумева унос описа саме игре (шта је потребно спојити) као и свих парова који ће бити директној вези у самој игри. И овде су могуће две опције додавања, као и код уноса речи. Појединачна игра се може додати кроз дијалог који садржи форму. Сва поља у форми су неопходна и ниједно поље не сме бити празно. Унос је и овде омогућен преко документа где је подржано додавање великог броја рекорда.

Треће и последње задужење супервизора је да уноси нове игре за игру „Асоцијације“. Код овог задужења подразумева се да супервизор унесе сва поља у свакој колони асоцијација, као и решење свих колона и коначно решење целе игре асоцијација. И овде је омогућено додавање појединачне игре и скупа игара преко документа. Све функционише на исти начин као претходно два описана случаја осим што је форма за додавање игре асоцијација другачија и садржи поља за колоне и решења колона, као и коначно решење целе игре асоцијација.

* + 1. Учесник квиза

Учесник квиза, када се пријави на систем, има четири опције:

1. Креира нову партију, као плави такмичар, и чека у лобију док се неки играч не пријави на његову игру (играч који се пријави на његову игру ће бити на позицији црвеног такмичара).
2. Може се пријавити на већ креирану партију и у том случају добија позицију црвеног такмичара, док је плави такмичар онај који је креирао партију.
3. Омогућено је, такође, да учесник квиза који се пријави на систем игра и самостално дневну игру која се у систему назива „Игра дана”. Ова игра је описана у одељку „Администратор” (3.1.1.). Сама игра као игра биће детаљно објашњена у поглављу „Игра” (3.2.).
4. Последња опција коју сваки учесник може користити јесте преглед резултата за игру текућег дана. Ранг листа текућег дана је реализована тако да садржи првих десет такмичара према највећем броју освојених поена. Уколико се тренутно пријављени учесник не налази међу најбољих десет такмичара, његов ранг се исписује на дну ранг листе. Такође, на истој страници, могуће је приступити приказу детаља свих партија које је такмичар играо у протеклих десет дана.
   * 1. Гост

Корисник који се није регистровао, односно нема кориснички налог у систему, може са почетне стране да уђе као гост и приступи систему као овај тип корисника. Тип корисника Гост може на јавно видљивој веб страници да види табелу коју представља ранг листу такмичара. Госту је омогућено да види табелу најбољих учесника у протеклих седам дана (обухватајући данашњи дан и претходних шест) као и табелу најбољих учесника у текућем месецу (обухватајући период од првог дана у текућем месецу до тренутног дана).

* 1. Игра

Игра представља суштински моменат реализације овог система и приказује „борбу” учесника један против другог или борбу учесника против рачунара у случају игре дана. Сам чин партије постоји у два различита случаја:

1. Дневна игра
2. Игра учесник против учесника
   * 1. Дневна игра

Дневна игра предстаља вид партије коју учесник може играти само једном у току једног дана. Игра дана је игра коју учесник игра искључиво против рачунара, односно, без противника. Циљ ове игре јесте да учесник скупи највећи могући број поена у свих пет игара укупно у временским интервалима који су задати за сваку појединачну игру.

Када се партија покрене, учеснику се на екрану отвара дијалог у којем ће се приказивати све игре. Свака игра је временски ограничена и сваку игру учесник мора самостално да покрене. Постоји пет игара које учесник игра и то су:

1. Слагалица
2. Мој број
3. Скочко
4. Спојнице
5. Асоцијације

У игри „Слагалица“, учеснику се на након одабраних слова на екрану приказује дванаест азбучних карактера који су насумично одабрани од стране рачунара. Учесник има 60 секунди да састави реч од понуђених слова и потврди састављену реч. Учесник саставља реч одабиром насумично изгенерисаних слова, која се приказују на екрану. Могуће је и обрисати последње одабрано слово или цео унос уколико корисник није задовољан. Сваки карактер који је учесник одабрао се обележава плавом позадином и онемогућује се поновно бирање истог карактера уколико је исти већ одабран. Након што учесник потврди реч или истекне време, игра се завршава и кориснику се при врху дијалога исписује број освојених поена у текућој игри. Поени у игри „Слагалица“ се рачунају тако што се у формираној речи израчуна број слова који се у њој налазе и тај број се помножи са два. У преводу, свако слово у формираној речи вреди два поена. Наравно, ово се све подразумева само у случају да је реч коју је учесник саставио исправна и да се налази у бази речи коју је претходно креирао Супервизор кроз свој панел. Након пар тренутака учесник се пребацује на следећу игру а то је игра „Мој број“.

„Мој број“ је игра у којој је циљ да такмичар направи такав математички израз да би добио што ближи број задатом броју. За ову игру учесник сме користити само основне математичке операције као што су множење, дељење, сабирање и одузимање, уз коришћење заграда ради лакшег формирања приоритета при рачунању самог израза. Игра почиње након што учесник сам покрене време. У том моменту почиње да тече време од 60 секунди и то је временски интервал у којем учесник мора да направи адекватан математички израз. Рачунар насумично бира 4 једноцифрена броја из опсега {1 - 9}, 1 двоцифрен број из скупа {10, 15, 20} и 1 двоцифрени или трофифрени број из скупа {25, 50, 75, 100}, и задаје циљани број из опсега {1 - 999} који треба да се добије коришћењем задатих бројева. Слично као и у игри „Слагалица“, и овде се израз формира притиском на саме бројеве са екрана као и притиском на дугмад која представљају аритметичке операције или заграде. Сваки број који је већ искоришћен у изразу се обележава плавом позадином и није могуће поновно бирање већ одабраног броја. Игра се завршава након што истекне време или након што учесник потврди формирани израз. Рачунање поена у игри „Мој број“ у форми дневне игре се своди на то уколико је учесник изразом погодио тачан број освојиће 10 поена, уколико у опсегу од 5 поена изнад или преко циљаног броја, такмичар ће освојити 5 поена, уколико ништа од претходна 2 случаја није обезбеђено, такмичар осваја 0 поена.

Игра „Скочко“ је такође ограничена временским интервалом од 60 секунди и почиње након што учесник покрене. У том моменту се формира комбинација од 4 насумично одабрана симбола коју је потребно погодити логичким покушајима појединачних комбинација. Поред сваке потврђене комбинације било је потребно приказати стање саме комбинације, број погођених симбола који су на правом месту, као и број погођених симбола који се не налазе на исправном месту. Било је неопходно имплементирати 6 покушаја за такмичара. Са десне стране дијалога се налазе симболи које је могуће одабрати за комбинацију. Симболи су: пик, херц, треф, каро, звезда и скочко, тим редоследом. Сваки симбол се бира притиском на исти и након сваког одабира знала,он се пресликава у леви део где се налази сама комбинација. Свака комбинација се састоји од 4 симбола и није могуће одабрати више од 4 симбола у једној комбинацији. Да би се приказало стање саме комбинације са бројем црвених и жутих поља, учесник мора да потврди комбинацију. Такође је омогућено да, уколико учесник није задовољан комбинацијом коју је унео, а још увек није потврдио, обрише задату комбинацију и почне испочетка погађање исте. Игра се завршава након што време истекне или након што учесник испроба свих 6 покушаја или након што погоди саму комбинацију кроз 6 или мање покушаја. Рачунање поена у овој игри је једноставно, уколико учесник погоди комбинацију, освојиће 10 поена, уколико не погоди комбинацију или време истекне, освојиће 0 поена у овој игри.

Следећа игра носи назив „Спојнице“. На дијалогу је било неопходно приказати по 10 парова „*Leva Spojnica*” - „*Desna Spojnica*” и сам опис шта је потребно упарити у овој игри. „Спојнице“ се попуњавају појмовима након што такмичар покрене игру. У том моменту се на екрану појављује опис саме игре спојнице и шта је потребно спојити као и сви појмови у левој и десној колони у насумичном поретку. Такмичар има 60 секунди за ову игру као и за претходне три игре. Сваки упарени појам кориснику доноси један поен.

Игра „Асоцијације“ јесте последња игра у низу. Ова игра има временски интервал од 240 секунди и у том интервалу такмичар треба погоди што више колона у асоцијацијама или у савршеном случају погоди коначно решење саме асоцијације. На дијалогу се могу видети поља 4 колоне, то су колона A, колона B, колона C и колона D. Свака колона садржи 4 појма на основу којег учесник треба да погоди решење колоне. Поља у свакој колони су нумерисана бројевима од 1 до 4, дакле А1, А2, А3, А4, B1, B2, B3, итд. Корисник може отварати поља докле год не добије идеју за коначно решење колоне или чак коначног решења целе асоцијације без икаквог ограничења пошто игра у режиму против рачунара, односно без противника. Свака погођена асоцијације за колону доноси такмичару 5 поена. Уколико такмичар погоди коначно решење асоцијације, он добија 10 поена за тај погодак као и додатних N\*5 поена где је N број неотворених коначних решења група, 1 ≤ N ≤ 4. Уколико време истекне, а такмичар не погоди коначно решење асоцијације, сва поља и коначна решења која није погодио се отварају. Уколико такмичар пак погоди коначно решење, игра се такође завршава и, уколико је било неотворених поља или коначних решења колона које није погодио, она се такође откривају такмичару. Након завршетка игре асоцијација, дијалог се затвара и игра дана је у том моменту званично завршена.

* + 1. Игра учесник против учесника

Игра „Учесник против учесника” представља вид игре који такмичари могу играти неограничен број пута у току дана. Једино што је потребно јесте да се такмичари споје у лобију и да игра крене. Овај тип игре је могуће играти када један од такмичара креира партију, а други такмичар приступи истој игри из дијалога за бирање игре. Било је потребно обезбедити креирање партије. Када такмичар направи партију, креатор резервише себи место на позицији плавог такмичара. На њему је само да мирно сачека док се неко од такмичара не пријави на његову игру. Када се то деси обојици такмичара се покреће дијалог за игру и игра може да почне.

Са друге стране, такмичар који се пријављује на игру код неког другог такмичара заузима позицију црвеног играча. Пријавом на игру се такмичару отвара дијалог у којем су излистане све креиране игре које чекају на свог противника. Свака игра за опис има корисничко име такмичара који је креирао саму игру, тако да такмичари знају са ким улазе у дуел. Избором неке од излистаних игара, такмичарима се покреће дијалог за игру.

Плави такмичар, односно онај који је креирао игру је први на потезу. Дијалог који приказује тренутно надметање у самој игри на врху садржи информацију о тренутном броју освојених поена, као и о тренутном броју освојених поена противника тако да оба такмичара буду константно обавештени о ситауацији. Остатак дијалога који приказује прву игру, игру „Слагалица“, је идентичан оном који је претходно описан у одељку „Дневна игра“ (3.2.1.). Црвени такмичар је блокиран све док плави такмичар не покрене игру. Након тога насумична слова се приказују на екрану и црвени такмичар се одмрзава тако да оба такмичара могу да погађају реч од понуђених слова. Први такмичар који потврди реч се замрзава. Тај такмичар је све време у том стању све док противник не потврди своју реч или док не истекне време. Када истекне време такмичарима се понаособ исписује број освојених поена и прелази се на другу рунду где игру започиње црвени такмичар, а плави такмичар је пасиван, односно замрзнут све док игра не крене. Након што игра почне, логика функционисања је иста као и у првој рунди. Након завршетка друге рунде прелази се на следећу игру „Мој број“.

Игра „Мој број“ функционише на истом принципу као и прва игра, плави такмичар започиње прву рунду, док је црвени такмичар замрзнут. У другој рунди је ситуација обрнута. Функционалност саме игре је иста као и у већ описаном одељку „Дневна игра“.

Игра „Скочко“ такође функционише на исти начин као претходне две игре где плави такмичар започиње рунду, док црвени чека замрзнут и обрнуто у другој рунди. Исто тако, логика и функционисање саме игре је иста као што је већ описано.

Четврта игра „Спојнице“ има 4 режима рада, први режим је када плави такмичар игра, док је црвени замрзнут, други режим је када црвени такмичар погађа спојнице које плави такмичар није погодио, трећи режим је почетак друге рунде, односно црвени такмичар погађа тотално нову игру спојница и четврти режим је када плави такмичар погађа спојнице које црвени такмичар није погодио у другој рунди. Кад год је један такмичар на потезу, други такмичар је замрзнут, односно може само да посматра шта се дешава на екрану док противник игра.

Последња игра је игра „Асоцијације“ која се игра наизменично. Такмичар који је на потезу може да отвори једно неотворено поље, након чега сме да погађа коначно решење неке од група или коначно решење целе асоцијације. Погођене групе плавог такмичара су плаве боје док су погођене групе црвеног такмичара црвене боје. Након сваке погођене групе обојици такмичара се аутоматски мењају број освојених поена, односно број освојених поена противника. Исто тако свако отворено поље било ког такмичара, противник је у могућности да види „уживо”.

* 1. Опис коришћених технологија

Приликом реализације овог система коришћено је неколико есенцијалних технологија. Као прва јесте .*NET* *Core*. .*NET* *Core* представља једну од најсавременијих технологија за серверску страну (енг. *backend*) која је развијена од стране компаније Мајкрософт (ориг. *Microsoft*). То је најновији продукт из .*NET* екосистема и представља платформу на којој се могу развијати апликације које не зависе од оперативног система. .*NET* апликације је могуће развијати и на *Linux* и на *Windows* оперативним системима, као и на другим мање познатим оперативним системима. На овој платформи могуће је развијати разне апликације као што су мобилне апликације, десктоп апликације, интернет апликације и друге. Разлог настанка ове платформе јесте био недостатак алата .*NET* *Framework* који је могао да ради само на *Windows* оперативном систему и био је уско повезан са самим уређајима који су користили сервисе као такве. .*NET* *Core* је такође специфичан по томе што програмски код извршава на исти начин било да је у питању *х64* или *х86* инструкцијска машина. Такође, могуће је развијати апликације у неколико различитих програмских језика у овом екосистему, а то су:

1. *C#*
2. *F#*
3. *Visual Basic*

Модел из .*NET* *Core* екосистема који је коришћен у изради овог пројекта je *Web* *API* (енг. *API* – *Application Programming Interface*). *Web* *API* јесте тип пројекта из .*NET* *Core* екосистема који представља модел за развијање web апликација базиран на *REST* архитектури. У овом моделу је имплементирана сва логика функционисања система што се тиче серверске стране апликације. Сва комуникације и сви захтеви који пристижу, пристижу на овај архитектурални модел. Улазна тачка у *Web* *API* пројекат јесу контролери који прихватају захтеве из клијентског дела (енг. *frontend*) апликације. Свим захтевима се проверава ауторизација, а затим врши даља обрада и враћа одговор.

За приступ бази података коришћен је алат *Entity* *Framework*, такође део .*NET* екосистема и представља алат за мапирање модела базе на одређене класе у програмском коду. Сама база података која је коришћена јесте *SQL* *Server* и у њој се налази цела шема базе, као и сами подаци. *SQL* *Server* јесте релациона база података која садржи табеле које су међусобно спојене преко одређених кључева. Све табеле и релације су креиране кроз *C#* програмски код у овој форми коришћења кроз алат *Entity* *Framework*. Приступ који је коришћен приликом реализације повезивања и мапирања кроз *Entity* *Framework* јесте *Code* *First* приступ. Овај приступ представља модел где креирање класе у програмском коду и извршавање одређених команди резултује у креирање табела у самој бази. Други приступ који је мало мање популаран јесте *Database* *First* приступ. Овај приступ је обрнуто пропорцијалан претходном. Он се базира на креирање самих табела и веза између табела у самој бази, а након покретања одређених команди, на основу формиране шеме базе, у програмском коду се генеришу класни модели који представљају креиране ентитете у бази.

Што се тиче специфичних технологија коришћених на клијентској страни апликације ту су *Angular* који предњачи и пропратне библиотеке које употпуњују функционисање и изглед саме апликације, а то су библиотека *PrimeNG*, *Nebular* и библиотека *RxJS*. *Angular* је радни оквир(енг. *framework*,представља сличан концепт библиотеци) за развој интернет апликација који је креиран од стране компаније Гугл (ориг. *Google*). Основна намена технологије *Angular* јесте развој једностраничних интернет апликација. Језик у којем се најчешће пишу *Angular* апликације јесте *TypeScript*. *TypeScript* је језик који представља омотач око језика *Javascript* и он јесте реализације вишег програмског језика уз сву подршку за објектно оријентисано програмирање. Овај језик се на крају преводи у *Javascript* команде које су познате интернет претраживачима. Свака *Angular* апликација се састоји из мноштва компоненти које сачињавају стабло. Скуп компоненти формира целину и та целина се групише у модул. Свака апликација развијена у овој технологији се мора састојати из бар једног модула, а тај модул мора садржати корену компоненту. Остале компоненте се угњеждавају у корену компоненту.

Екстерна библиотека *PrimeNG* је библиотека која која садржи имплементацију базичних компоненти сваке интернет апликације као што су елементи форме, дијалози, падајући менији итд. *PrimeNG* је развијен од стране компаније „*PrimeTek Informatics*” која је већ годинама позната у свету по развијању и унапређивању решења за корисничке интерфејсе. Ова библиотека је библиотека отвореног кода (енг. *open source*) што значи да је бесплатна за коришћење и нуди огроман спектар разних функционалности за сваку засебну компоненту. Инсталирање је могуће извршити преко alata *npm - node package manager* кроз пар једноставних команди. Након успешне инсталације довољно је само укључити модуле који су неопходни за коришћење у систем. Слично важи и за библиотеку *Nebular*.

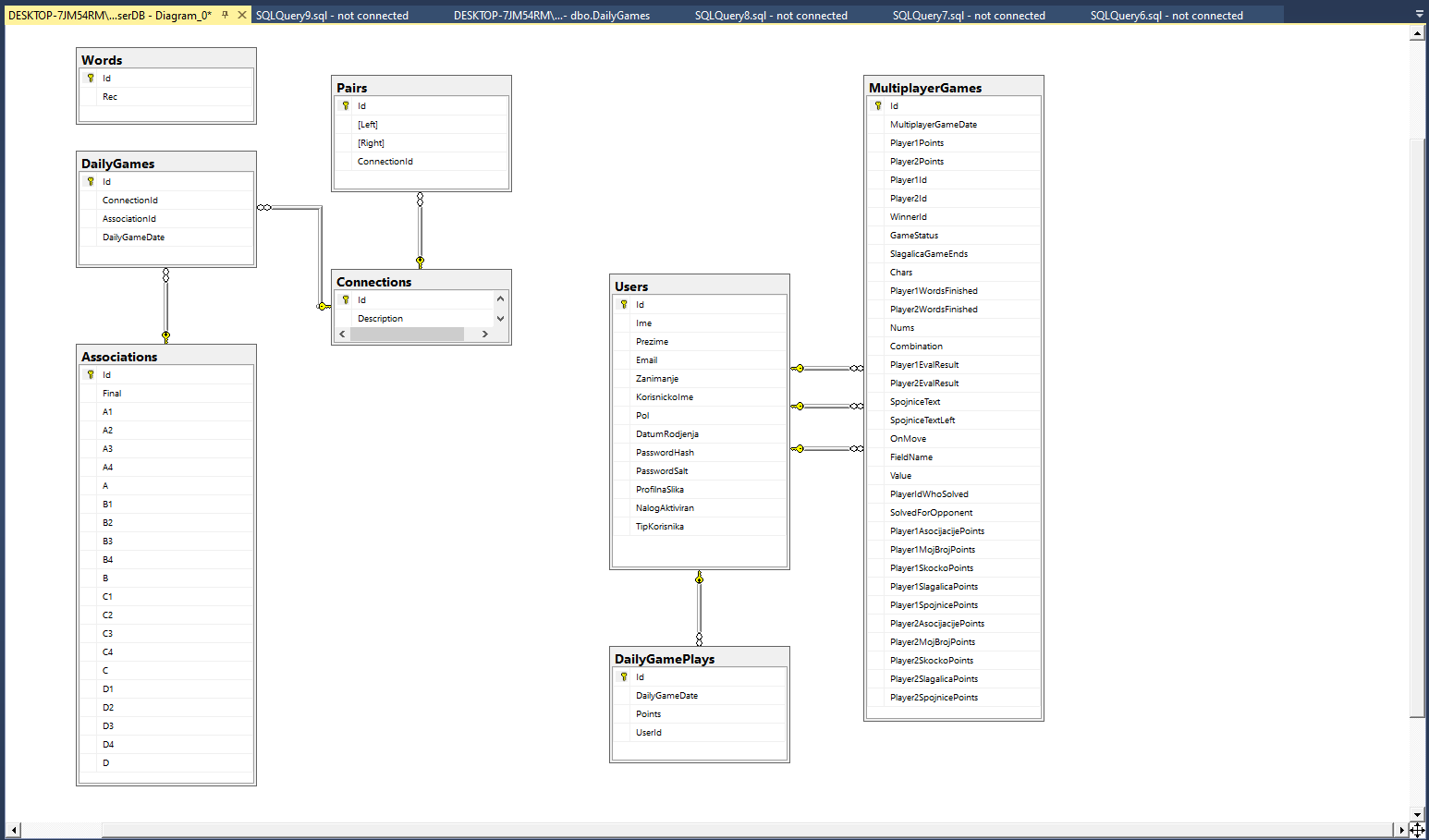
Библиотека *RxJS* која је претходно наведена представља библиотеку која омогућава реактивно програмирање. Она је у ствари *JavaScript* имплементација и подршка за асинхроно програмирање. Ова библиотека је одговорна за слагање асинхроних позива који су утемељени на догађајима. Функционише по принципу претплаћивања, на пример неки ентитет је претплаћен на промене приликом неких догађаја. Када се десе промене, нови ток података се шаље на претплаћене ентитете где потом ти ентитети имају могућност обраде нових података.

1. Реализација система

У овом поглављу биће детаљно објашњена реализација система квиза за два играча. Комплетна архитектура софтвера на серверској и клијентској страни биће анализирана у потпуности. Такође ће бити приказана шема базе података и основне појединости које представљају кључне ставке у њеној структури.

* 1. Опис базе података

База података која је коришћена при реализацији овог система јесте релациона база *SQL Server.* За реализацију система било је потребно неколико табела чији су модел и релације приказани на слици 4.1.1.



Слика 4.1.1. Шема базе података из алата *SQL Server.*

Прва табела од значаја јесте табела *Words.* У овој табели се чувају речи које се сматрају за исправне приликом провере исправности унетих речи у првој игри „Слагалица“. Ова табела је веома проста и садржи само две колоне, а то су колона која представља идентификатор и примарни кључ *Id* и друга колона која представља податак о самој речи назива се *Rec.* Ова колона је типа *string.* У ову табелу стижу подаци које може унети искључиво Супервизор кроз свој панел.

Табела *DailyGames* представља табелу у којој се чувају све информације везане за игре које ће бити подмеднуте играчима у дневној игри. Овде се налазе подаци који прецизирају која игра „Спојнице“, као и која игра „Асоцијације“ ће бити прослеђена играчима за одређени дан. Табела садржи четири колоне и то су колона идентификатор која је уједно и примарни кључ *Id,* колона *ConnectionId* која представља страни кључ и везу са табелом *Connections,* колона *AssociationId* која представља страни кључ такође и везу са табелом *Associations* и колона *DailyGameDate* која представља тачан датум за који ће конкретне игре бити везане у дневној игри. Све колоне које представљају јединствени идентификатор су типа *GUID (Globally Unique Identifier).*

Табела *Connections* је табела која садржи две колоне и то су колона јединственог идентификатора *Id* и колона у којој се чува сам опис за конкретну игру Спојнице, а то је колона *Description.* Овај ентитет је одговоран за чување информација о самој игри Спојнице. Табела која употпуњује ову функционалност јесте табела *Pairs.* У овој табели постоје четири колоне и то су колона јединственог идентификатора и примарни кључ *Id,* колона *Left* у којој се чувају појмови са леве стране пара*,* колона *Right* где се налазе појмови са десне стране пара*,* и колона *ConnectionId* која представља страни кључ односно везу са родитељском табелом *Connections.*

Слично томе, може се приметити табела *Associations.* У овој табели се чувају сви подаци везани за игру Асоцијације. Табела садржи колоне у којима се чувају појмови и синтагме који се крију иза самих поља у колонама игре Асоцијације. Осим тога, постоје и колоне у табели које садрже коначна решења самих колона, као и коначно решење целе асоцијације. Поред тога, ту је наравно и колона јединственог идентификатора и примарног кључа *Id.* Колоне у овој табели су *A1, A2, A3, A4, A, B1, B2, B3, B4, B, C1, C2, C3, C4, C, D1, D2, D3, D4, D, Final* и *Id.*

Једна од најважнијих табела свакако јесте табела *Users.* У овој табели се чувају сви подаци од важности за све кориснике у систему. Табела садржи колону *Id* која представља примарни кључ и јединствени идентификатор самог рекорда у табели, затим колона *Ime* која садржи име корисника, колона *Prezime* која садржи презиме корисника, колона *Email* која садржи адресу електронске поште корисника, колона *Zanimanje* у којој се чува податак о занимању корисника, колона *KorisnickoIme* у којој се чува јединствено корисничко име корисника система, колона *Pol* у којој се чува пол корисника (мушко, женско), колона *DatumRodjenja* у којој се чува датум рођења корисника, колона *PasswordHash* и колона *PasswordSalt* које служе као начин за безбедно управљање корисничким шифрама (биће детаљно објашњено у наредним поглављима), колона *ProfilnaSlika* у којој се чува путања до профилне слике самог корисника, колона *NalogAktiviran* где се чува информација о активацији корисничког налога и колона *TipKorisnika* која садржи информацију о томе да ли је корисник Администратор, Супервизор или Учесник квиза.

Табела *DailyGamePlays* јесте табела у којој се налазе подаци везани за одигране игре дана и кориснике који су играли саму игру. Овај ентитет садржи четири колоне и то су примарни кључ и јединствени идентификатор *Id,* колона *DailyGameDate* у којој се налази податак о датуму за који дан је везана конкретна игра дана, колона *Points* у којој се чува број освојених поена корисника за конкретну игру дана и колона *UserId* која представља страни кључ и информацију о томе који корисник је у питању.

Табела којој се највише приступа у току коришћења овог система свакако јесте табела *MultiplayerGames.* У овој табели се налазе сви подаци везани за игру учесник против учесника.Табела садржи колону која представља примарни кључ и јединствени идентификатор *Id,* колону *MultiplayerGameDate* која представља датум када је одиграна игра учесник против улесника, колону *Player1Points* и колону *Player2Points* које представљају укупан број освојених поена плавог и црвеног такмичара у игри, колону *Player1Id* и колону *Player2Id* које представљају јединствене идентификаторе за плавог и црвеног такмичара, колону *WinnerId* која представља идентификатор корисника који је победио у игри. Колоне *Player1AsocijacijePoints, Player1MojBrojPoints, Player1SkockoPoints, Player1SpojnicePoints, Player2AsocijacijePoints, Player2SpojnicePoints, Player2SlagalicaPoints, Player2MojBrojPoints* и *Player2SkockoPoints* садрже информацију о укупном броју освојених поена у појединачној игри за плавог, а потом и црвеног такмичара. Остале колоне (*GameStatus, SlagalicaGameEnds, Chars, Player1WordsFinished, Player2WordsFinished, Nums, Combination, Player1EvalResult, Player2EvalResult, SpojniceText, SpojniceTextLeft, OnMove, FieldName, Value, PlayerIdWhoSolved SolvedForOpponent*) служе за помоћ при имплементацији одређених логика функционисања у самој игри и улоге неких ће бити објашњене у наставку реализације система.

* 1. Преглед архитектуре система

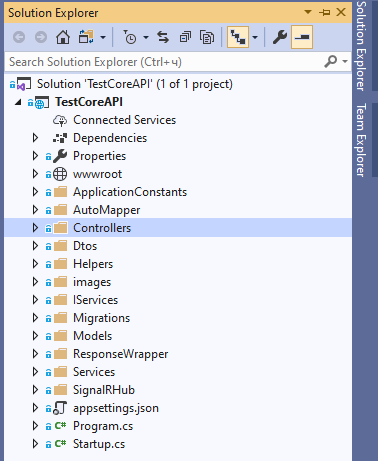
У овом потпоглављу биће описана архитектура самог система са техничког аспекта. Биће објашњене све главне софтверске компоненте система као и комуникација између истих. Такође, биће анализирани детаљи и функције кључних модула у систему. Сама архитектура овог система припада такозваној трослојној веб архитектури. Трослојна веб архитектура на највишем нивоу представља систем који има три главна нивоа и то су:

1. Клијентска апликација
2. Серверска апликација
3. База података

Сваки од ових наведених нивоа у себи може садржати поднивое где се, на пример, серверска апликација може разбити на неке комплексне архитектуралне принципе серверске стране.

* + 1. Серверска апликација

Структура серверске апликације се може видети на следећој слици:



Слика 4.2.1.1. Структура серверске апликације*.*

Серверска апликација је имплементирана, као што је већ раније наведено, у технологији *.NET Core Web API* и представља серверску страну овог система. *Web API* тип пројекта јесте тренутно најпопуларнији шаблон за писање интернет апликација првенствено због своје одличне структуре и архитектуре самих компонената. Све софтверске компоненте у овом шаблону су предвиђене да се користе кроз добро познати пројектни узорак који се назива *Dependency Injection. Dependency Injection* као пројектни узорак представља могућност динамичке имплементације одређених модула тако што се уместо инстанцирања конкретних објеката неке класе, у времену извршавања (енг. *run-time*)подмеће конкретна имплементација у зависности од потребе самог модула или софтверске компоненте. Тако се смањује велика зависност која се у софтверу ствара између компонената и отварају се врата за динамичку имплементацију и могућност лаких измена и промена у будућности.

Сви сервиси и модули, као и конфигурација се подмећу у *Startup.cs* класи која представља улазни део у серверску апликацију након што се изврше команде из *Program.cs* класе. У овом систему, може се приметити неколико битних конфигурационих ставки, као и регистрација појединих сервиса кроз пројектни узорак *Dependency Injection.* Конфигурација базе података, односно постављање за базу *SQL Server* се врши као једна од првих ставки у методи *ConfigureServices.* Потом се може приметити конфигурација за коришћење библиотеке, односно пакета *AutoMapper* који служи за аутоматско мапирање објеката различитих класа преко имена њихових класних поља. Затим следи регистрација неколицине сервиса из система као што су *IUserService* и *IOrganizationService.* Такође, ту се налази и конфигурација за *CORS (Cross Origin Resource Sharing). CORS* представља механизам који дозвољава (или блокира) одређеним ентитетима да приступе серверској апликацији са одређеног домена. У овом случају може се приметити да серверска апликација дозвољава захтеве само са домена <http://localhost:4300>. Последња команда јесте регистрација *SignalR* сервиса у *Dependency Injection.*



Слика 4.2.1.2. Део програмског кода из класе *Startup.cs.*

* + 1. Контролери

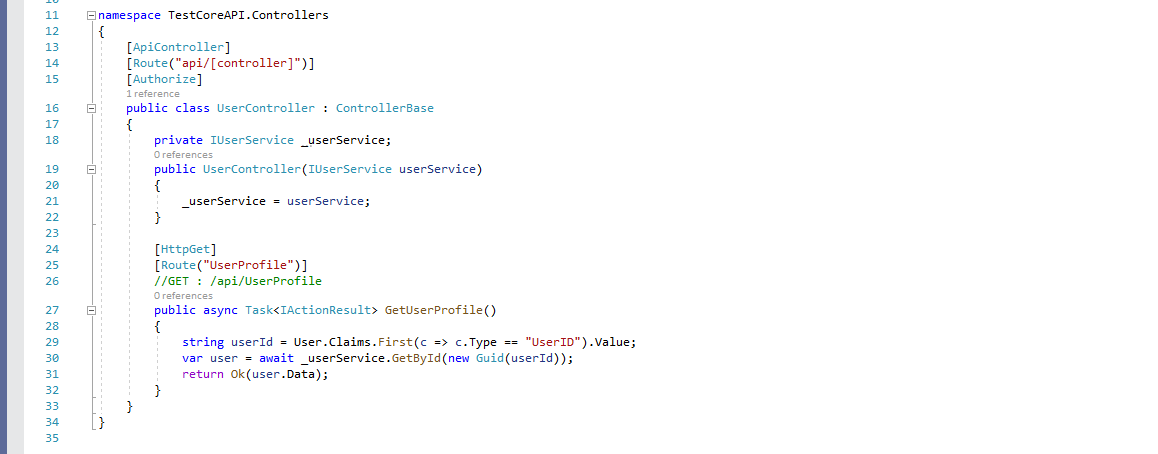
Сви захтеви који могу да пристигну на серверску страну апликације долазе до првог нивоа приступа у софтверску компоненту која се назива контролер. Контролер представља улазни део у *Web API* и то је полазна тачка где стижу сви захтеви и одакле се извршавање програмског кода делегира на даље. У овом систему постоји три контролер компоненте и то су *AuthenticationController, OrganizationController* и *UserController.*

*AuthenticationController* јесте контролер који је задужен за обраду захтева који су специфични за аутентикацију или за неку специфичну корисничку функционалност. На пример, у овом контролеру се налази обрада регистрације корисника, као и сама обрада захтева пријаве на систем. Оно што је специфично за овај контролер јесте да нема никакав ауторизациони ниво одбране, дакле овај контролер је могуће послати без икакве претходне ауторизације, што значи да су *API* методе отворене за спољашност.



Слика 4.2.2.1. Део програмског кода из класе *AuthenticationController.cs.*

Други контролер је *UserController. UserController* представља контролер који је у овом систему задужен само за једну ствар, а то је дохватање тренутно улогованог корисника и свих детаља везаних за тог корисника. Овај контролер је заштићен и захтева ауторизоване захтеве да би се подаци дохватили успешно. Ауторизација захтева је постављена на највишем нивоу у овом контролеру и то је на самом контролеру, што значи да све *API* методе у овом контролеру захтевају ауторизоване захтеве. Постоји и опција захтевања ауторизације по методи, али у том случају би ауторизација била сетована изнад самих метода које захтевају ауторизоване позиве, а не на самом контролеру.



Слика 4.2.2.2. Део програмског кода из класе *UserController.cs.*

Последњи контролер у систему јесте и највећи и најкомплекснији, а то је *OrganizationController.* У овом контролеру се налази сва могућа обрада захтева који су специфични за саму игру, конфигурисање, статистику итд. Методе у овом контролеру су заштићене ауторизацијом.



Слика 4.2.2.3. Део програмског кода из класе *OrganizationController.cs.*

* + 1. Сервиси

У другом нивоу серверске апликације налазе се сервиси. Сервиси представљају софтверску компоненту у којој се налази сва логика овог система. Контролери су задужени искључиво за позивање извршавања метода из сервиса и враћање резултата из тих метода. Сама логика и обрада свих података се налази у овој сервисном слоју серверске апликације. Инстанце сервиса се дохватају, углавном, у контролеру, а потом се преко те инстанце која је прослеђена кроз већ објашњени пројектни узорак *Dependency Injection* позивају методе сервиса. Сервиси у овом систему представљају имплементацију њиховог надређеног интерфејса. Ова имплементација је реализована на овај начин из разлога евентуалног проширења система и креирања евентуалних нових имплементација сервисног интерфејса који би био подметнут неком другом контролеру, сервису, или било којој компоненти. Пример сервисног интерфејса се налази на слици испод:



Слика 4.2.3.1. Део програмског кода из интерфејса *IUserInterface.cs.*

* + 1. Контекст базе података, модели и миграције

Као што је наведено раније, у класи *Startup.cs* се налази конфигурација за базу података као и постављање апликационог контекста базе података. У *Dependency Injection* је подметнута класа апликационог контекста базе *ApplicationDbContext.cs* што значи да се у тој класи налазе све дефиниције и мапирања везана за базу података.



Слика 4.2.4.1. Додавање апликационог контекста базе *ApplicationDbContext.*

У *ApplicationDbContext.cs* класи се налазе сви ентитети који су мапирани један-на-један са базом података и представљају моделе који су спона програмског кода и табела из базе података. Сама конекција је омогућена проширивањем *DbContext* класе из пакета *EntityFramework.* Све ове ентитетске класе су дефинисане унутар омотачке апстрактне класе *DbSet* која се налази у самом пакету *EntityFramework.* Овим је омогућено и експлицитно постављена особина да се кро програмски код шема базе података може мењати тако што се поља на мапираним класама мењају, додају или уклањају.



Слика 4.2.4.2. Проширивање *ApplicationDbContext* класе*.*

Веома битна ставка у имплементацији и развоју овог система, као и одржавању шеме базе података јесу миграције. Миграције представљају сет инструкција које се командом генеришу у програмском језику *C#* и оне су одговорне за саме промене над шемом базе података. Током развоја библиотека *EntityFramework* константно прати стање свих модела (да ли је додато неко поље, да ли је промењен тип неког поља итд...) и када програмер одлучи да изврши промене над шемом базе он ће извршити команду *add-migration {migration-name}* и тада ће библиотека наћи промене на моделима између претходног стања и тренутног стања и изгенерисаће сет инструкција у новој миграцији које ће бити задужене за промену шеме базе. Након што су инструкције успешно изгенерисане програмер је дужан да изврши инструкцију *update-database.* У том моменту ће се саме промене из програмског кода пресликати на шему базе и поново ће ентитетске класе и табеле из базе података бити мапиране. Пример миграције је приказан на слици испод:

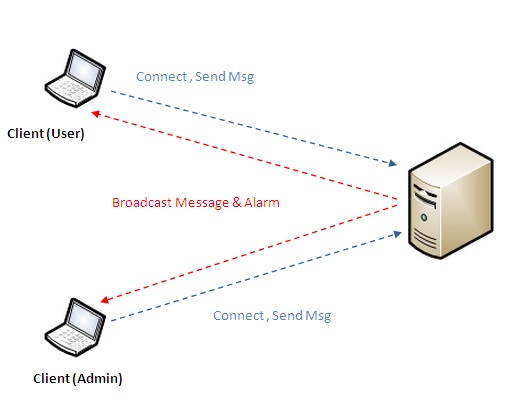


Слика 4.2.4.3. Пример миграције*.*

Ова миграција задужена је за додавање нове колоне *Player1EvalResult* типа *int* у табелу *MultiplayerGames*.

* + 1. SignalR Hub

Ова софтверска компонента јесте једна од кључних у имплементацији интерактивног квиза за два играча. *SignalR Hub* представља софтверску библиотеку у *.NET* екосистему која омогућава слање асинхроних обавештења клијентским апликацијама од стране серверске апликације. Ова библиотека садржи серверске компоненте као и клијентске компоненте које су имплементиране у језику *JavaScript.* Суштина ове библиотеке јесте да повеже клијенте са сервером и да одржава интеракцију између њих преко сервера чија је улога у овом случају да буде серверски комуникатор између самих клијената.



Слика 4.2.5.1. Пример комуникације преко *SignalR* алата*.*

* + 1. Клијентска апликација

Клијентска апликација у овом систему, као што је већ раније поменуто, реализована је кроз *Angular* радни оквир. Верзија која је коришћена је *Angular* 10. Клијентска апликација организована је у неколико модула и то су *core, theme, authentication* и *pages* модул. Наравно, они су сви део главног модула *app.* У *core* модулу су углавном имплементирани *Angular* сервиси који функционишу на принципу пројектног узорка *Singleton*. То значи да када се неки сервис убаци у *Dependency Injection* одређене компоненте, сваки позив и приступ инстанци тог сервиса из конкретне компоненте је омогућен кроз само једну инстанцу тог сервиса. Дакле, приликом приступања неком сервису и његовим функцијама, не инстанцира се константно нови објекат тог сервиса, већ се увек дохвата већ креирана инстанца и над њом се врше операције. То је генерално начин функционисања *Angular* сервиса. У *core* модулу су реализовани *HttpHandlerService*, *UserService* и *FrameService.*

*HttpHandlerService* је сервис у којем су имплементиране методе које се налазе у последњој линији приликом слања захтева на серверску апликацију. Ту се налазе сви *http* позиви ка серверу и одатле се након пристизања одговора, одговор прослеђује уназад ка компоненти која је позвала неку функцију сервиса. Сва комуникација између клијентске и серверске апликације се одвија преко *http* протокола у *REST* архитектури. Серијализација и десеријализација се одиграва у протоколу уз евентуалну додатну конфигурацију при пристизању података. Типови захтева који су реализовани у овом систему су *get* и *post.*

*UserService* представља сервис у којем се чувају есенцијални подаци који су од значаја за функционисање самог корисника. Ту се, на пример, налази објекат који представља корисника и све његове детаље, да би у сваком могућем тренутку, уколико је потребно, може извући конкретан податак о улогованом кориснику и проследити на неку обраду (на пример *Id* корисника, корисничко име или адреса електронске поште). Ту се такође чувају неки подаци који су битни за саму игру учесник против учесника као што је објекат класе *MultiplayerGame* који је у том моменту активан за играча који игра партију.

*FrameService* је сервис у ком се налазе сви подаци који су битни за само стање екрана у одређеним моментима. На пример, ту се чувају информације о томе да ли кориснику екран треба бити замрзнут или откључан (да ли је *loader* активан или не), ту се налази обрада и логика приказивања *toast* порука итд.

Модул *theme* је модул у којем су реализоване више естетске ствари везане за саму апликацију. Ту се налазе компоненте које су одговорне за оквир и распоред сегмената саме апликације, *header* и *footer* компоненте, имплементација глобалних стилова у језику *css*, *directives* и *pipes* имплементација.

*Authentication* модул садржи имплементацију свих спољних страна апликације које су јавно доступне свим корисницима као што су страница за пријаву на систем, страница за регистрацију, страница за ранг листе. Осим тога овај модул садржи веома битне делове одговорне за сигурност клијентске апликације, као што је *AuthService*, *AuthGuard* и *AuthInterceptor*. У сервису *AuthService* се налази имплементација провере типа корисника док се у сервису *AuthGuard* налази имплементација заштите нивоа приступа страници од стране одређеног корисника (Супервизор има приступ једној групи интернет страница у овом систему, Администратор другој групи, Учесник квиза трећој). Сервис *AuthInterceptor* представља пресретач свих захтева ка спољашности ван клијентске апликације. Овај сервис пресреће сваки захтев упућен ван и његова функција је да попуни неке ауторизационе параметре у самом захтеву пре слања захтева ка серверској апликацији. То значи да је његова улога да у сваки захтев ка серверској апликацији (у случају улогованог корисника) смести ауторизациони токен који је претходно добио од сервера када се пријавио на систем. Једино тако захтев ће бити ауторизован од стране сервера и сервер ће „веровати” тим захтевима. Више о овоме у наредном поглављу.

Последњи модул од значаја је *pages* модул. У овом модулу се налазе све компоненте које су имплементиране за странице којима приступа учесник и супервизор. Осим имплементација и стилизовања ових компонената, овај модул не садржи ништа комплексније од реализације самих страница и њихове интерне логике.

* 1. Главни имплементациони изазови

У току реализације система, искрсло је неколико већих изазова који су отежали саму имплементацију. У овом поглављу биће анализирани најинтересантнији изазови и њихова решења кроз примере и објашњења.

* + 1. Аутентификација и ауторизација

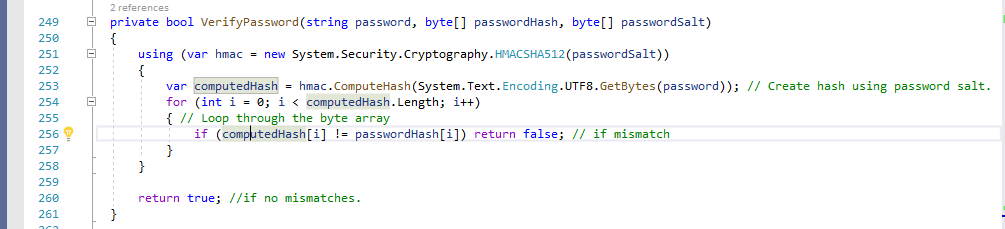
Сигурно један од најбитнијих момената у реализацији система јесте била имплементација аутентификације и ауторизације корисника. Без овога систем би био непотпун, а са овом функционалношћу систем је добио форму једног озбиљног софтвера који је заштићен сигурносним факторима. С обзиром на то да клијентска и серверска апликација константно комуницирају, било је потребно обезбедити сигурну комуникацију где серверска апликација „верује” свим захтевима који су упућени од стране клијентске апликације. Поред тога, било је неопходно обезбедити, да у сваком моменту буде јасно који корисник је пријављен на систем, као и памћење стања пријављеног корисника у систему.

Аутентификација је обезбеђења коришћењем парадигме *JWT* (*JSON Web Token*)*.* *JWT* представља токен у којем се налазе све битне информације везане за тренутно пријављеног корисника као што су, на пример, корисничко име, адреса електронске поште, јединствени идентификатор корисника и слично. Безбедан начин коришћења шифре корисника у систему је имплементиран преко *PasswordHash* и *PasswordSalt* логике. Овиме је омогућена безбедна пријава на систем јер се саме шифре корисника уопште не чувају ни на који начин у систему. *PasswordHash* и *PasswordSalt* се приликом регистрације формирају од текстуалне шифре корисника и представљају низ бајтова који су добијени криптографским функцијама из класе *HMACSHA512*.



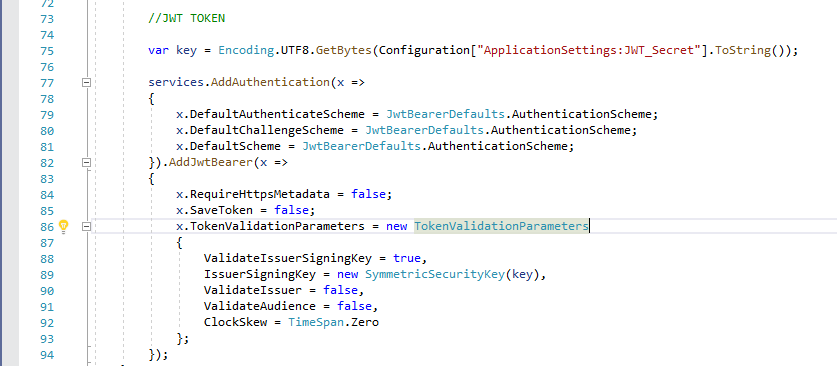
Слика 4.3.1.1. Формирање низа бајтова за *PasswordHash* и *PasswordSalt* од корисничке шифре*.*

Ове две вредности се потом чувају у бази података у табели *Users.* Приликом пријаве на систем, корисничка лозинка која се унесе и проследи на серверску апликацију се поново кодира у низ бајтова који се потом проверавају са већ унетим низом бајтова приликом регистрације корисника. Уколико се сви бајтови поклапају, значи да је унета лозинка исправна. Уколико било где постоји разлика у низу бајтова то сигнализира да лозинка није исправна и корисник неће моћи да се пријави на систем.



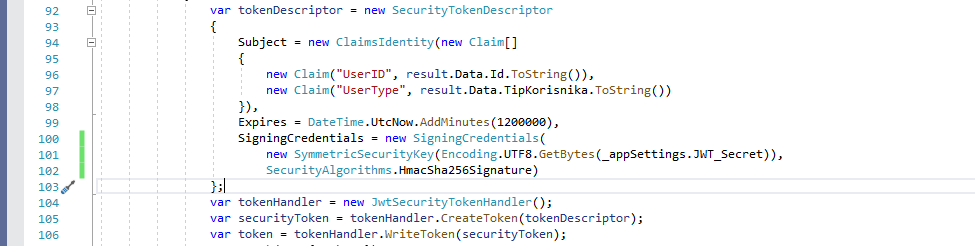
Слика 4.3.1.2. Провера исправности низа бајтова из корисничке лозинке*.*

Уколико је све у реду са самом лозинком и корисник се може пријавити на систем, серверска апликација издаје токен са свим неопходним информацијама о пријављеном кориснику. Од тог момента серверска апликација очекује од клијентских захтева да у свом заглављу садрже исти токен који је сервер издао кориснику приликом пријаве на систем. Овим је обезбеђен сигуран приступ серверској логици и увек приступачна информација о томе који корисник је направио захтев ка серверу. Серверска конфигурација која садржи информације о томе шта сервер очекује од пристиглих захтева се налази у *Startup.cs* класи. Ту је дефинисана аутентификација коју сервер очекује од клијентских захтева као и детаљи који се очекују од самог *JWT* токена.



Слика 4.3.1.3. Конфигурација аутентикације на серверској апликацији*.*

Издавање и генерисање самог токена се врши у *AuthenticationController* класи у *Login* методи. Овде се дефинишу појединости везане за сам токен као што су информације о томе шта ће се налазити у пољима *Claim* (у овом случају ту се налазе јединствени идентификатор корисника као и информација о типу корисника), колико дуго је токен валидан и којим алгоритмом шифровања ће бити потписан.



Слика 4.3.1.4. Део програмског кода који приказује имплементацију генерисања токена*.*

Што се тиче логике на клијентској страни, приликом захтева за пријаву на систем, у случају успешног покушаја пријаве, клијент у одговору добија токен који потом чува у локалној меморији интернет претраживача (*localStorage*)*.* Након што је токен трајно сачуван у меморији претраживача, имплементиран је пресретач свих захтева са клијентске стране ка серверу који је задужен да у заглавље сваког захтева налепи вредност токена из меморије претраживача. Ова имплементација је приказана на сликама 4.3.1.5. и 4.3.1.6.



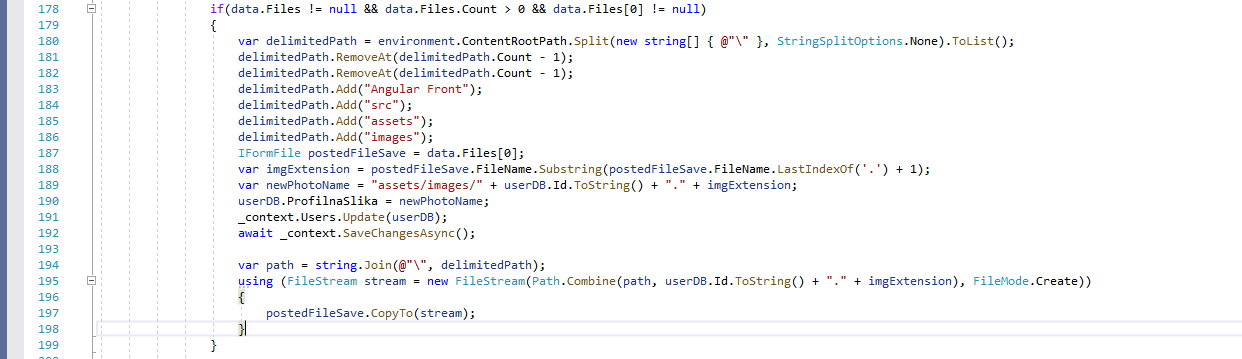
Слика 4.3.1.5. Део програмског кода у клијентској апликацији који приказује имплементацију обраде токена након захтева пријаве на систем*.*



Слика 4.3.1.6. Део програмског кода у клијентској апликацији који приказује имплементацију пресретача захтева ка серверу*.*

* + 1. Додавање профилне слике корисника

Једно од питања које се постављало приликом имплементације овог система јесте како на ефикасан начин обезбедити чување профилне слике корисника у систему тако да приступ и дохватање исте буде једноставан за клијентску апликацију. Да би клијентска апликација могла да приступи брзо и ефикасно самом документу слике, документ мора да се налази негде у структури директоријума клијентске апликације. Зато је имплементирано решење где се приликом регистрације новог корисника, документ слике која је закачена у захтеву обрађује на серверској страни тако што се у табели *Users* чува путања до слике које ће бити сачувана у */assets/images* директоријуму клијентске апликације. Сам документ који представља слику је сачуван у претходно наведеном директоријуму кроз серверску апликацију. Тако је обезбеђен једноставан приступ и референцирање слике из клијентске апликације, јер се путања до слике која се налази у објекту тренутно пријављеног корисника једноставно подметне у улазне (енг. *input*) компоненте.



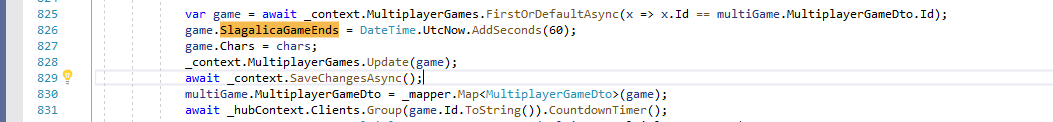
Слика 4.3.2.1. Део програмског кода који приказује чување слике*.*



Слика 4.3.2.2. Део програмског кода који приказује референцирање слике*.*

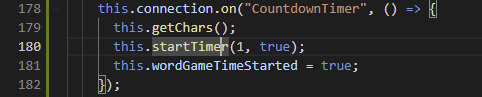
* + 1. Временски интервали у игри учесник против учесника

Један од првих изазова у имплементацији игре учесник против учесника био је подесити да оба такмичара виде исто или приближно исто време које откуцава за појединачну игру. Игра се мора завршити у исто време за оба такмичара и мора трајати исти временски интервал за оба такмичара. Пошто време почиње да откуцава оног момента када такмичар који је на потезу започне игру, наметало се питање како уз минималну разлику у времену започети откуцавање времена и другом играчу. Решење овог изазова је пронађено у коришћењу *SignalR* библиотеке. Логика решења јесте да, када такмичар који је на потезу покрене игру, са клијентске стране се пошаље захтев који се на серверу обрађује у неколико логичких целина. Прво се у табели *MultiplayerGames* пронађе игра коју тренутно играју два такмичара и у колону *SlagalicaGameEnds* се упише тачан датум завршетка тренутне игре а то је тренутни датум у универзалном координисаном времену (користи се *UTC* рачунање времена да серверско време клијената не би погрешно утицало на игру) на који је додат временски интервал који представља трајање тренутне игре. Након тога, преко *SignalR* библиотеке се свим корисницима који су у тренутној игри пошаље обавештење да је потребно да се почне са откуцавањем времена.

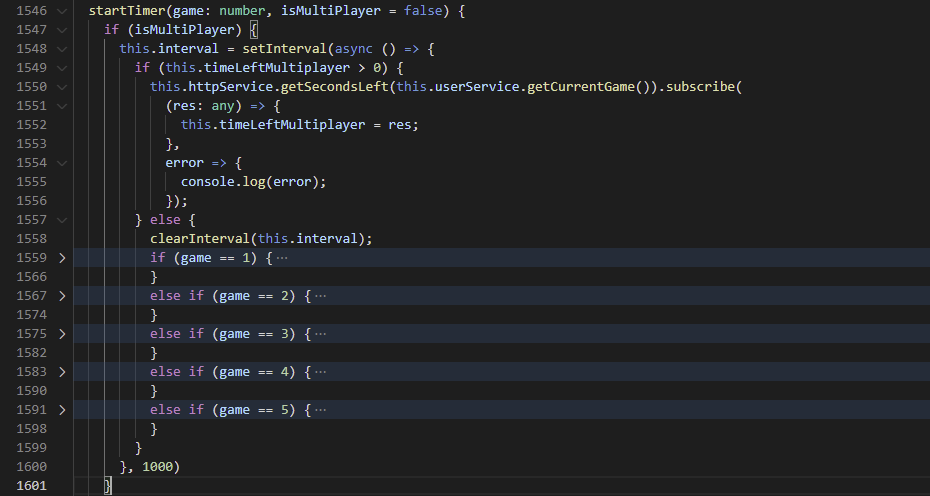


Слика 4.3.3.1. Део програмског кода који приказује чување датума краја игре као и обавештење свим клијентима у групи да време треба да почне да откуцава*.*

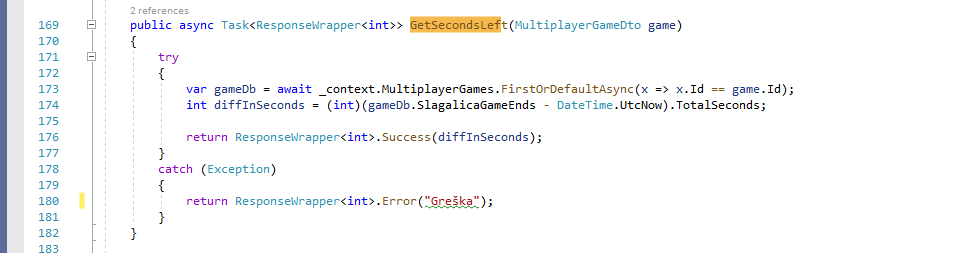
Након извршења овог блока програмског кода, сви клијенти из конкретне групе који су претплаћени на *CountdownTimer()* обавештајну методу извршавају логику са клијентске апликације која започиње слање захтева за рачунањем остатка времена у тренутној игри сваке секунде све док одговор од сервера не постане 0 секунди. Уколико време истекне пре него што такмичари заврше игру, даље се извршава код специфичан за саму игру и начин рачунања поена код одређене игре. Након тога прелази се на следећу игру или следећу рунду у игри у зависности од игре.



Слика 4.3.3.2. Део програмског кода који приказује позив за почетак откуцавања времена*.*



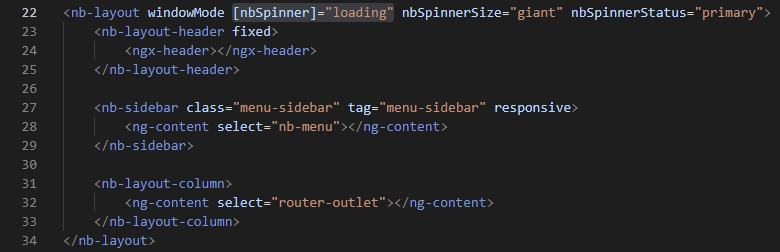
Слика 4.3.3.3. Део програмског кода који приказује клијентску имплементацију откуцавања времена*.*



Слика 4.3.3.4. Део програмског кода који приказује серверску имплементацију рачунања преосталог времена у игри*.*

* + 1. Чекање играча на потез

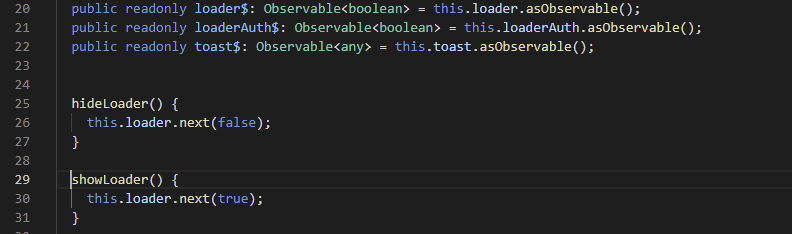
Још један од изазова са којим се требало суочити у самој игри учесник против учесника приликом имплементације јесте како онемогућити корисника да извршава било какву акцију док је противник на потезу у играма које се играју наизменично. Било је потребно неометано пустити играча који је на потезу да одигра свој део, док блокирани противник посматра ситуацију која се дешава на другој страни. Решење овог проблема је пронађено у комбинацији коришћења *SignalR* библиотеке и одређених модификација на клијентској страни. Након што такмичар заврши свој потез избором неке од акција специфичних за саму игру његов екран се замрзава и укључује се *loader* преко целог екрана који омогућава блокирање свих акција од стране конкретног корисника. Ово је обезбеђено коришћењем омотачке компоненте из библиотеке *Nebular*. Ова компонента се назива *nb*-*layout* и у себи садржи *nbSpinner* *input* који на екрану приказује *loader* уколико је вредност за *nbSpinner* прослеђена као *true*. Исто тако уколико се проследи вредност *false*, *loader* се уклања са екрана. Контролисање стања за *loader* је имплементирано у сервису *FrameService*.



Слика 4.3.4.1. Део програмског кода који приказује омотачку компоненту*.*



Слика 4.3.4.2. Део програмског кода који приказује доделу вредности за *loader* стање*.*



Слика 4.3.4.3. Део програмског кода из сервиса *FrameService* који приказује контролисање стања за *loader.*

Што се тиче серверске стране (на примеру игре Асоцијације) и сигнализације кроз *SignalR* библиотеку, сваки завршен потез неког од такмичара мења колону *OnMove* у табели *MultiplayerGames* за тренутну игру. У том пољу чува се информација који је играч следећи на потезу. Након промене вредности у поменутој колони позива се метода *NextPlayerAssoc()* на коју су претплаћени учесници у тренутној партији да би се клијентској страни поставило исправно стање на екрану (ко ће бити блокиран, а ко на потезу).



Слика 4.3.4.4. Део програмског кода који приказује промене за потезно поље*.*

Након овога претплаћени клијенти извршавају блок програмског кода из *NextPlayerAssoc()* обавештења где се извршава постављање исправног стања екрана информације ко је на потезу код сваког корисника.



Слика 4.3.4.5. Део програмског кода који приказује позив методе за постављање информације о томе који такмичар је на потезу*.*



Слика 4.3.4.6. Део програмског кода који приказује имплементацију методе за постављање информације о томе који такмичар је на потезу*.*

* + 1. Закључавање табеле

У току имплементације појавио се један вид препреке која је била неочекивана. С обзиром на то да се доста захтева у току игре учесник против учесника дешава у исто време, појавило се једно понашање у систему које је стварало конфузију. Примера ради, пуно захтева који на серверској страни извршавају промене у неким колонама над истом табелом (углавном *MultiplayerGames* табелом) су се најнормалније извршавали, међутим примећено је да увек један од два захтева заправо није направио промену у бази. Након дубоке анализе примећено је да када више од једног захтева покушава да изврши промене над истим рекордом у одређеној табели, само први који стигне заправо направи промене. Ово се дешавало због тога што *EntityFramework* као библиотека, у општем случају игнорише све остале захтеве који покушавају да изврше промене над истим рекордом у релативно слично време из сигурносних разлога. Решење овог проблема је било експлицитно закључавање табеле и извршавање апсолутно истих инструкција, али под трансакцијом. У том случају, ни један захтев за променама над базом неће бити прескочен, већ ће се сваки извршити након што се претходна трансакција заврши.



Слика 4.3.5.1. Део програмског кода који приказује извршавање промена над базом користећи трансакцију*.*

1. Опис рада система

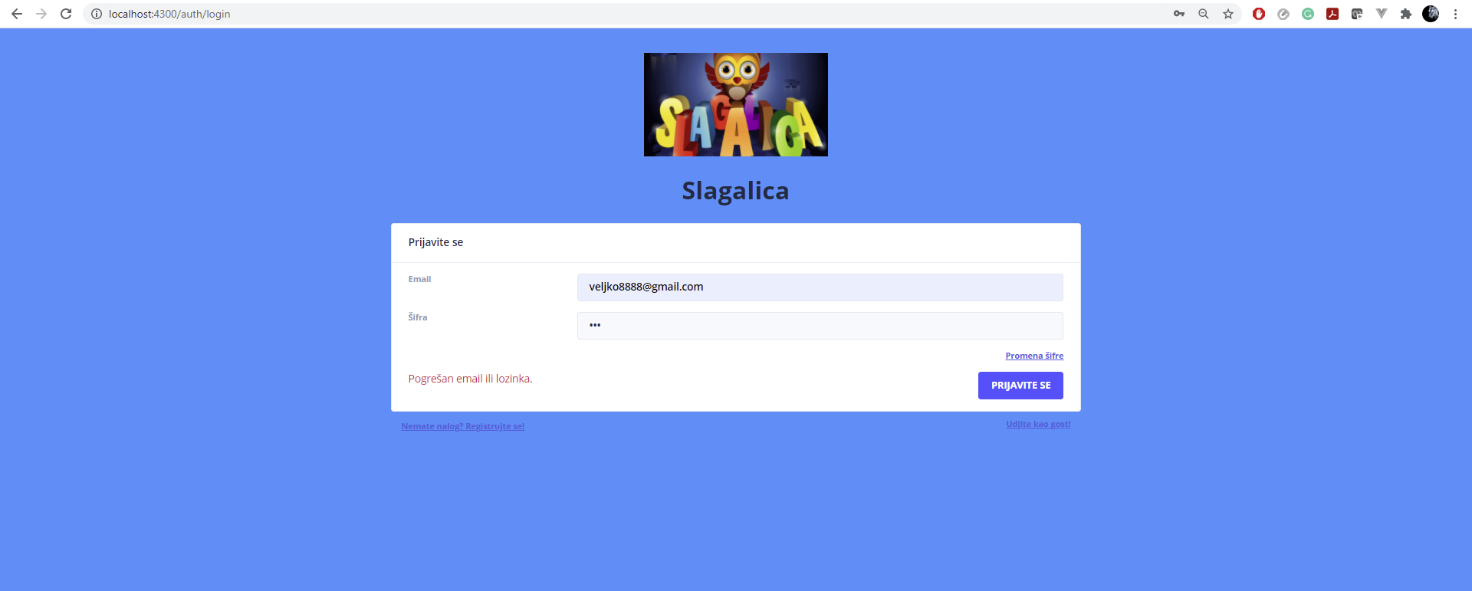
У овом поглављу биће анализиран целокупан рад система уз пратећа објашњења и неопходне детаље. Такође, биће приказана упутства за коришћење реализоване интернет апликације уз слике пропратних корисничких екрана.

* 1. Јавно доступан део система

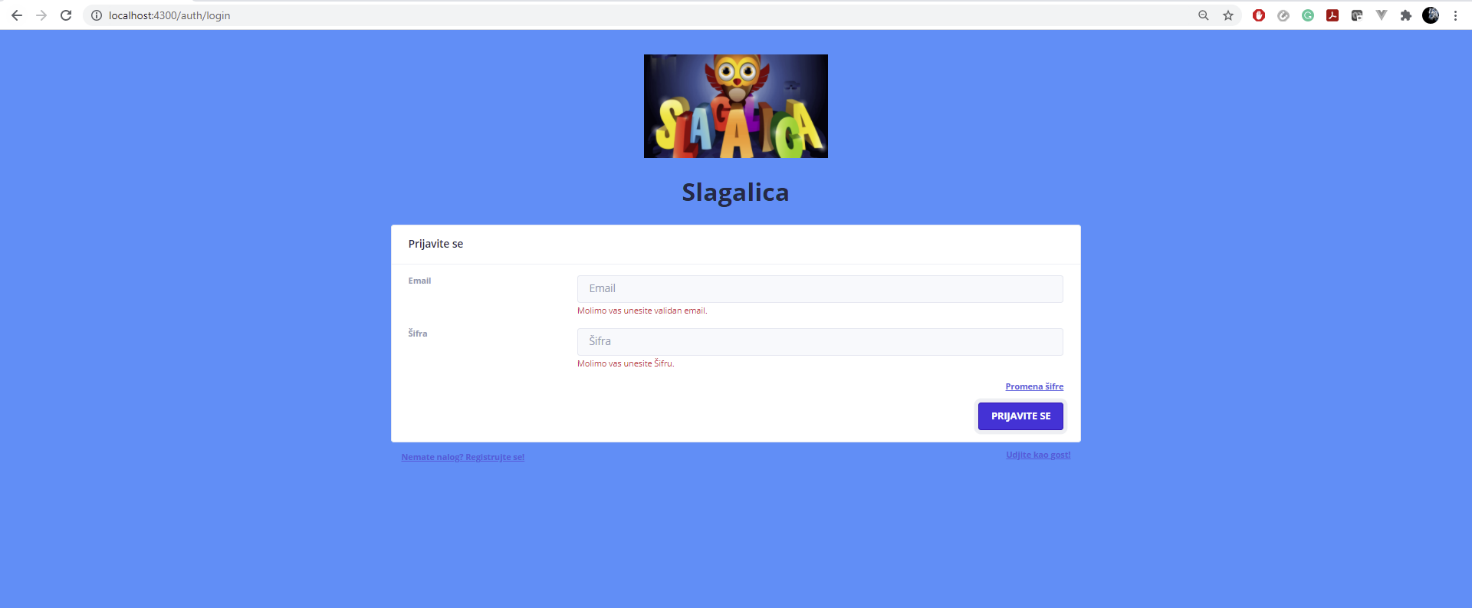
Кад корисник стигне на почетну страницу система на његовом екрану се налази форма за пријаву на систем. Корисник се пријављује на систем уношењем своје адресе електронске поште и лозинке. Уколико корисник унесе погрешне информације систем ће га обавестити о грешци. Такође, уколико корисник не унесе неки од података из форме, систем ће га обавестити о пољима које мора да попуни, или исправи унос.



Слика 5.1.1. Страница за пријаву на систем*.*

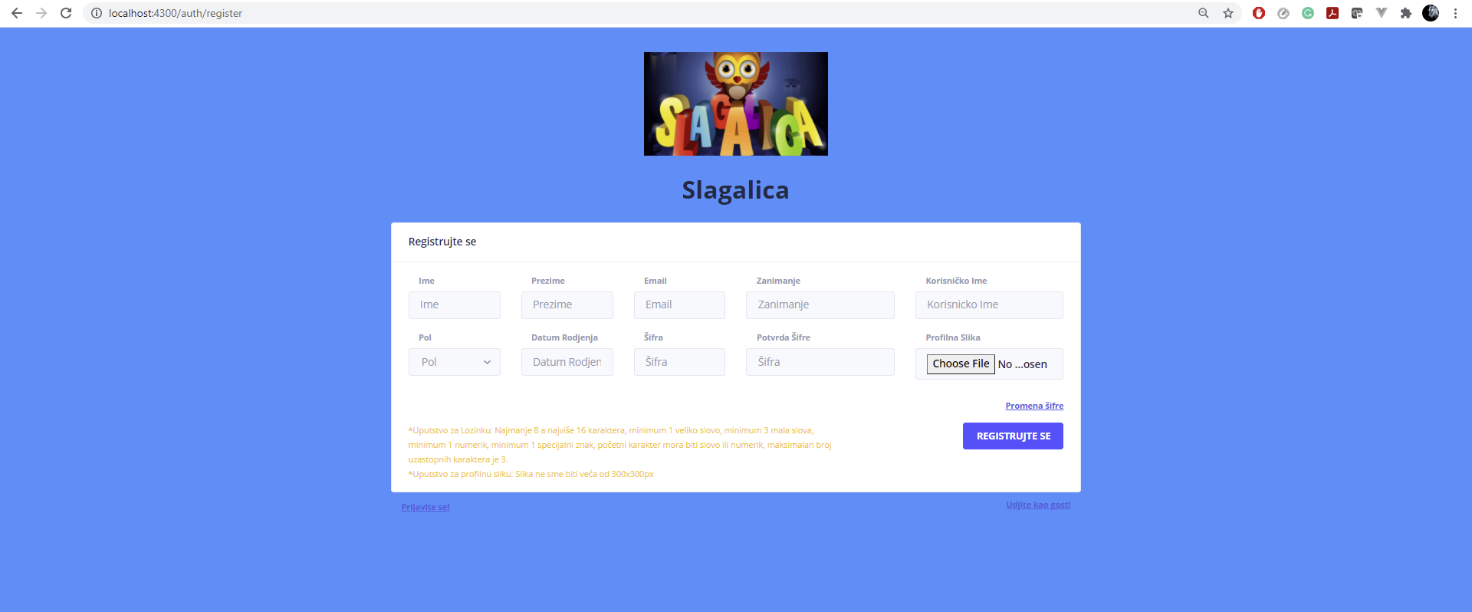


Слика 5.1.2. Страница за пријаву на систем – Погрешна лозинка или адреса електронске поште*.*

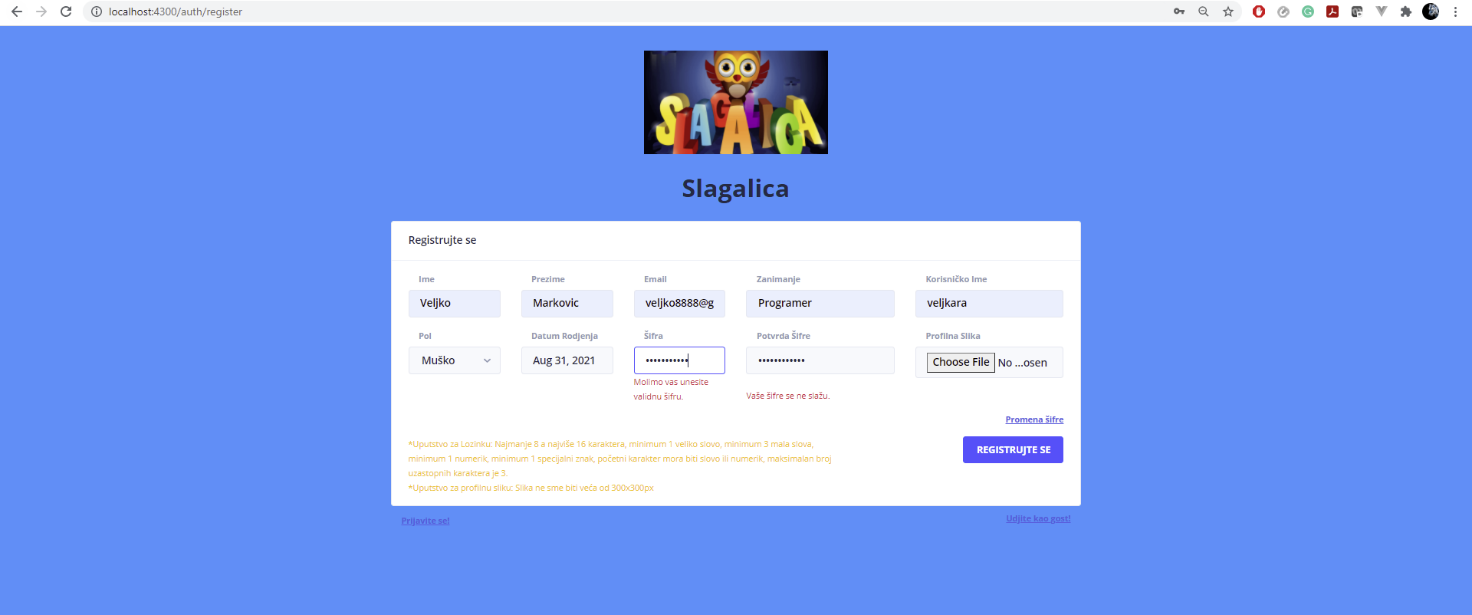


Слика 5.1.3. Страница за пријаву на систем – Валидација поља на форми*.*

Поред пријаве на систем, међу јавно доступним интернет страницама овог система налази се и страница за регистрацију корисника. Од корисника се очекује да унесе сва поља из форме осим профилне слике која није обавезна. Сваки евентуални проблем приликом уноса у форми ће бити исписан кориснику на екрану. На пример, уколико се лозинка и потврда лозинке не поклапају корисник ће бити обавештен.

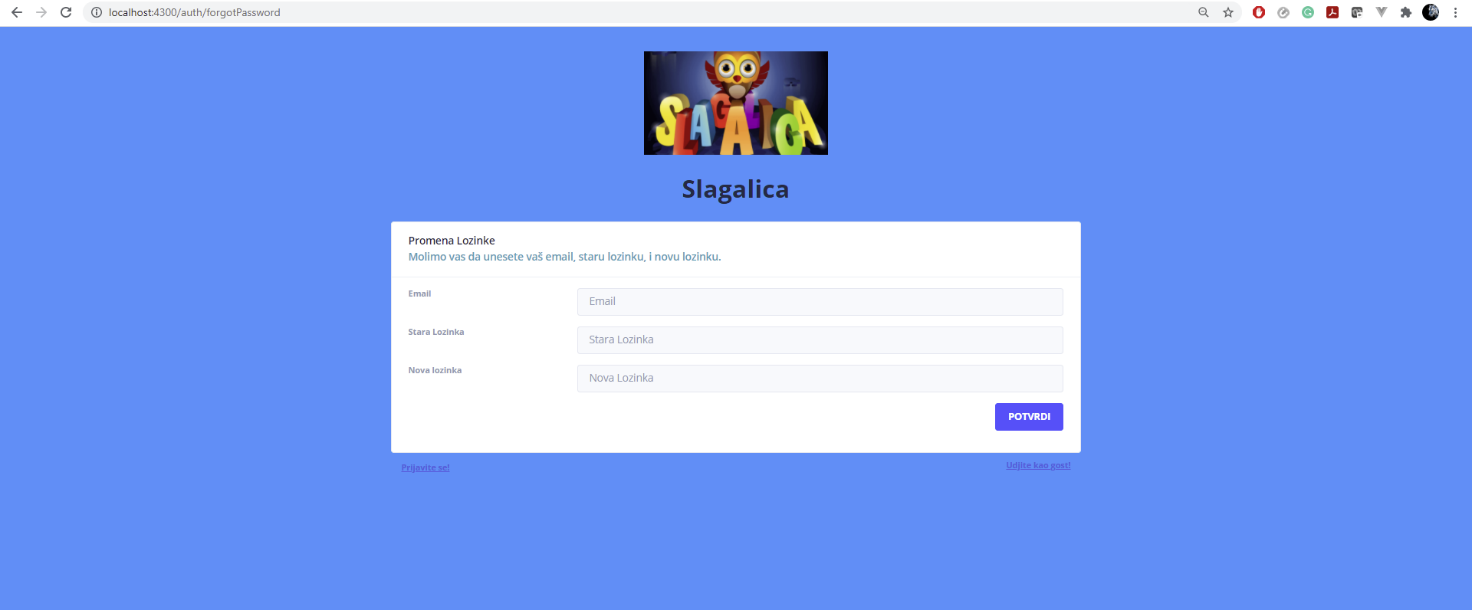


Слика 5.1.4. Страница за регистрацију*.*



Слика 5.1.5. Страница за регистрацију – Валидација лозинке*.*

Такође, могуће је променити лозинку преко јавно доступне странице за промену лозинке. Да би се променила лозинка потребно је унети адресу електронске поште, стару лозинку и нову лозинку. Уколико се унесе погрешна стара лозинка, грешка ће бити приказана на екрану. Такође, уколико не постоји адреса електронске поште за коју се мења лозинка, информација ће бити приказана на екрану.

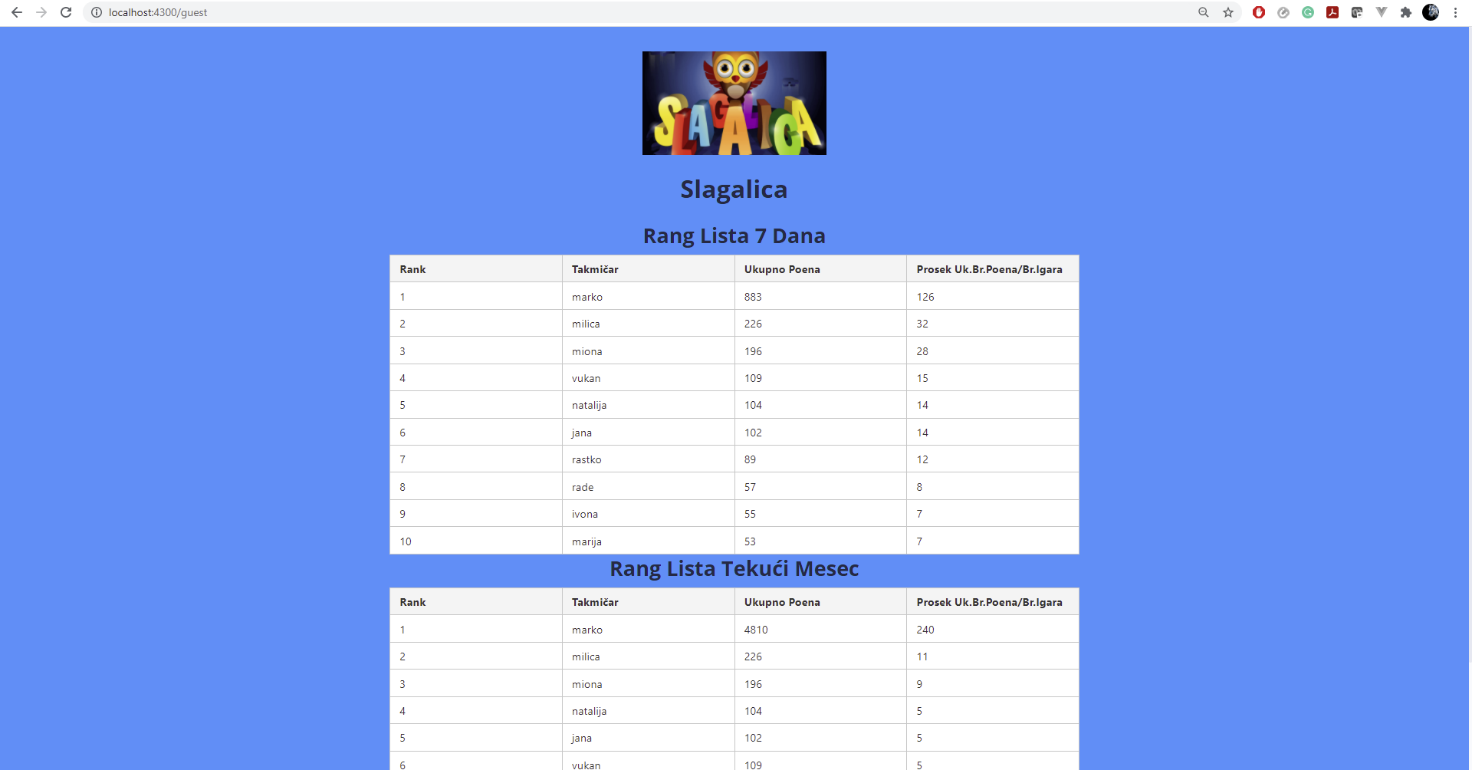


Слика 5.1.6. Страница за промену лозинке*.*



Слика 5.1.7. Страница за промену лозинке – Погрешна лозинка*.*

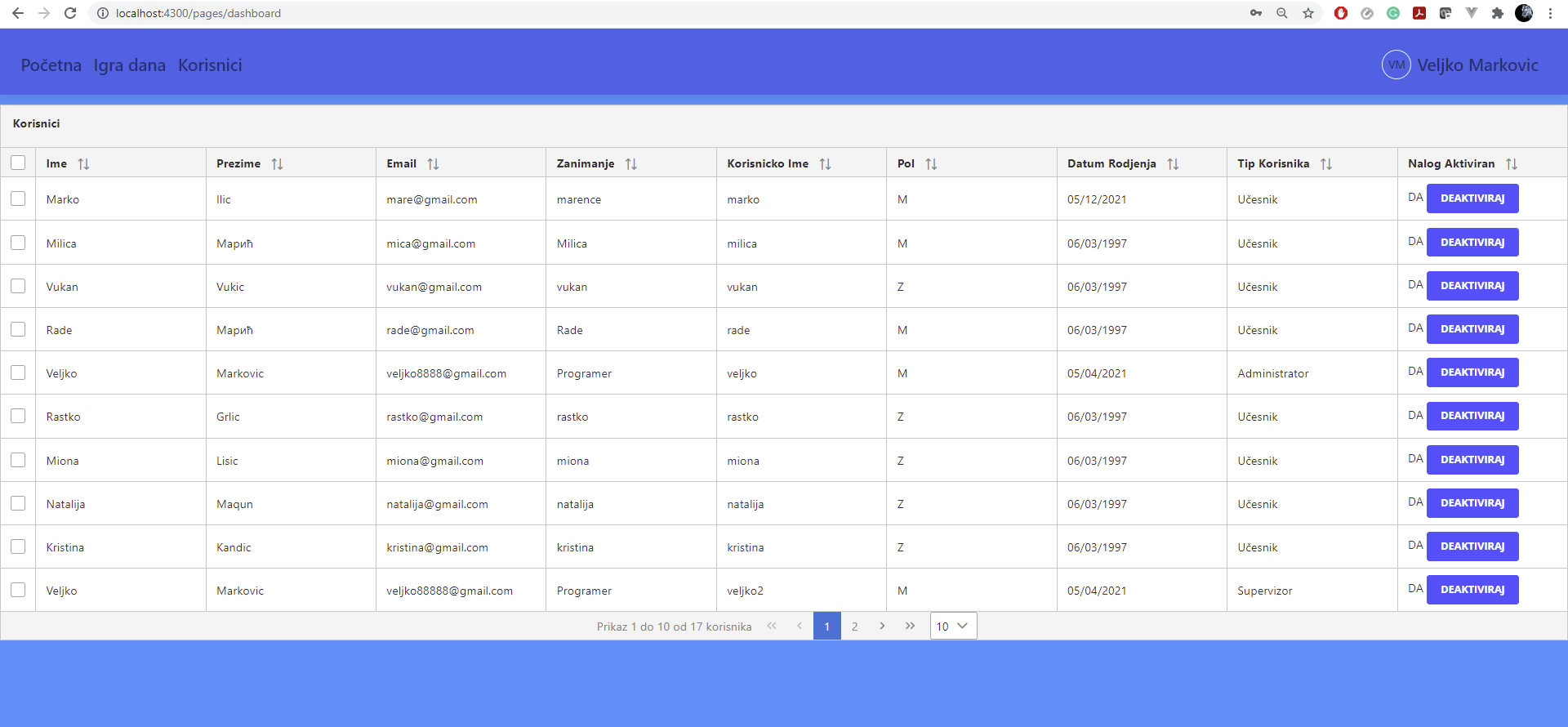
Поред ауторизационих интернет страница на овом систему, међу јавно доступним страницама налази се још и страница за ранг листе. Овој страници се приступа као гост и могуће је видети ранг листу учесника у квизу у претходних 7 дана, као и ранг листу учесника за текући месец. На листама се исписује 10 најбољих такмичара у оба случаја.



Слика 5.1.8. Ранг листе*.*

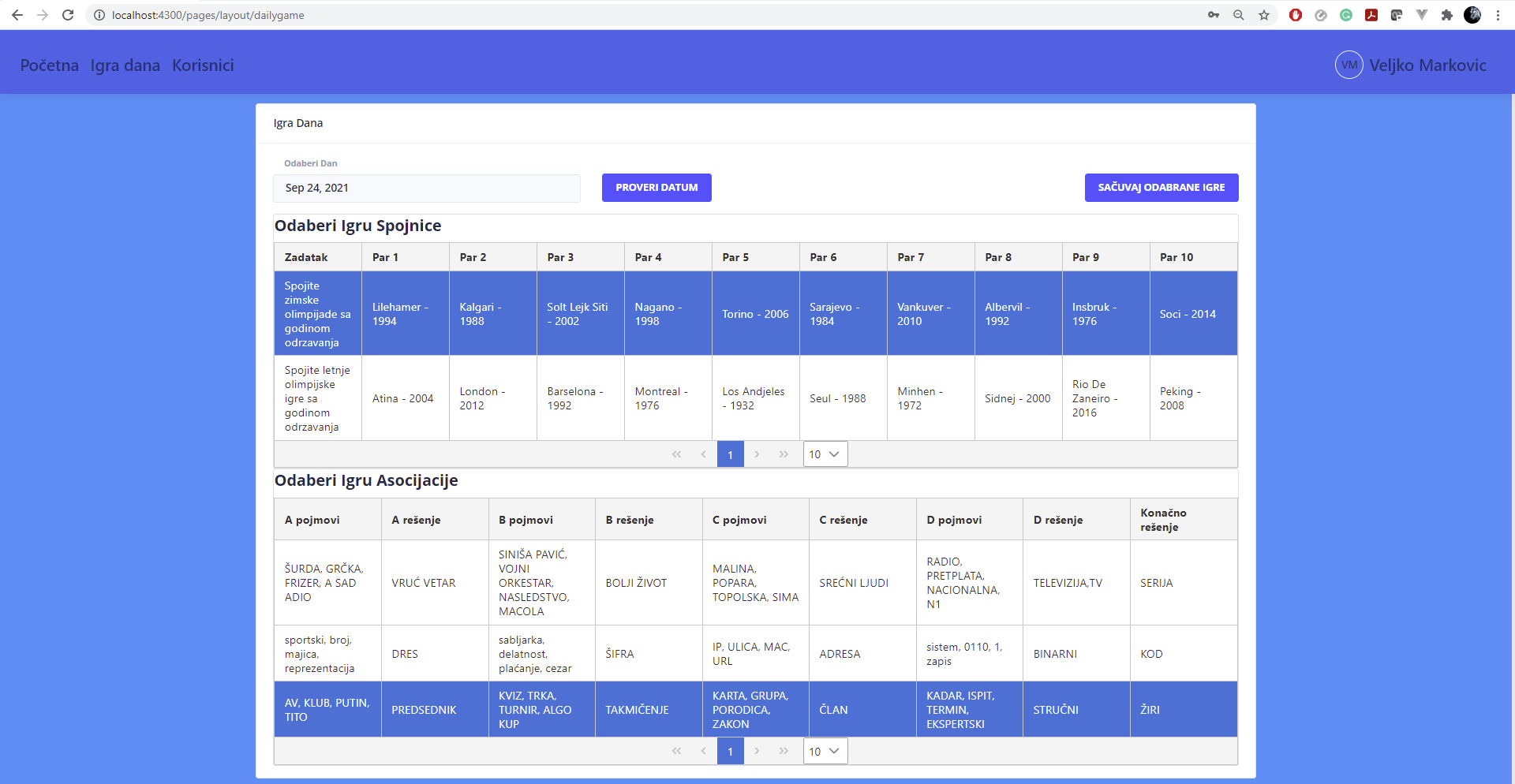
* 1. Администратор

Администратор када се пријави на систем, на свом екрану види табелу свих корисника система. Табела има детаљан приказ за сваког корисника и садржи све битне информације. Администратор је задужен за управљање корисничким налозима, тако што прихвата или игнорише пристигле захтеве за регистрацију. Он може да активира кориснички налог притиском на дугме „AKTIVIRAJ” и тада корисник постаје званично учесник квиза. Такође, администратор може и деактивирати кориснички налог уколико се јави потреба за тим притиском на дугме „DEAKTIVIRAJ”. Оба дугмета се налазе у самој табели у последњој колони. Уколико је корисник активан, поред њега стоји дугме за деактивацију, а уколико корисник није активан учесник квиза, поред њега стоји дугме за активацију. О статусу акције активирања или деактивације корисник је обавештен путем искачуће (енг. „*toast*”) поруке.



Слика 5.2.1. Администратор – Управљање корисничким налозима*.*

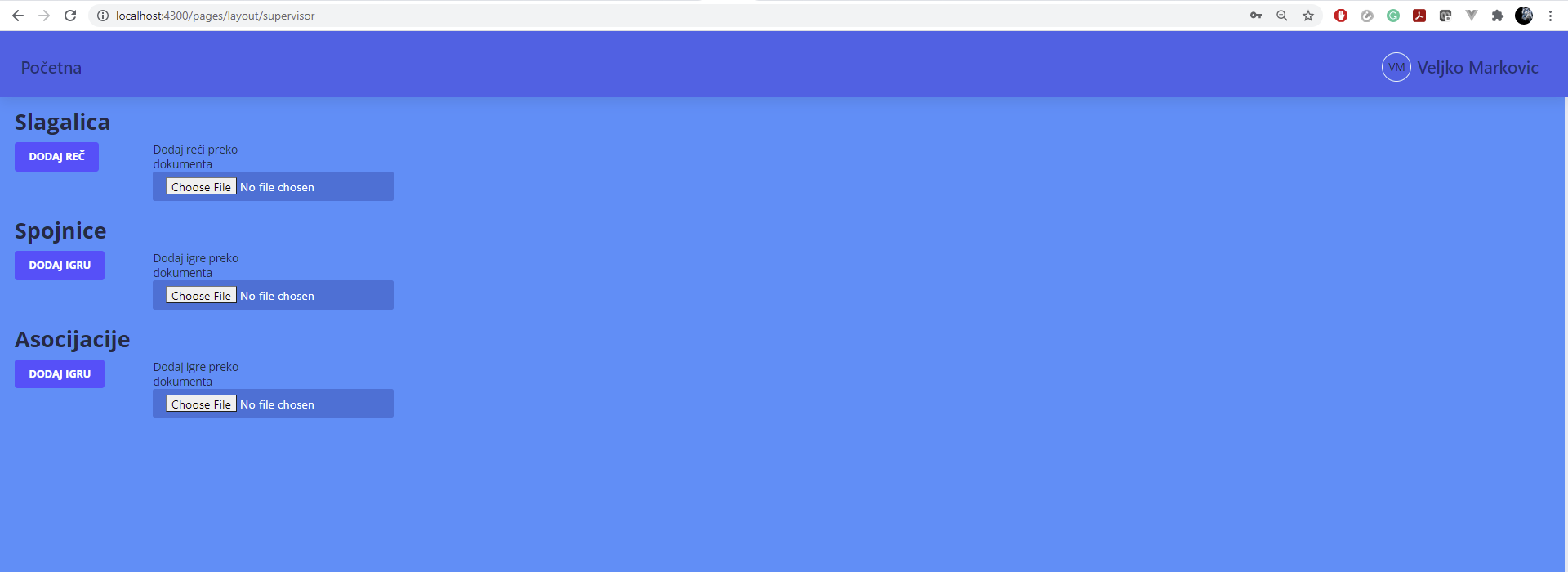
Поред улоге управљања корисничким налозима, администратор је задужен и за одабир игара дана. На страници за организацију игара дана, администратору се излиставају две табеле након што одабере датум у будућности. У првој табели налазе се све игре спојница које је могуће одабрати за игру дана. У другој табели се налазе све могуће игре асоцијација које је могуће одабрати за игру дана. Све ове игре претходно уноси супервизор у систем кроз свој панел. Бирање игара дана се врши једноставним избором жељене игре дана. Одабране игре дана се маркирају плавом позадином. Када администратор притисне дугме „SAČUVAJ ODABRANE IGRE”, у том моменту се за одабрани датум креирају везе са наведеним играма. О статусу акције чувања одабраних игара корисник је обавештен путем искачуће поруке.



Слика 5.2.2. Администратор – Бирање игара дана*.*

* 1. Супервизор

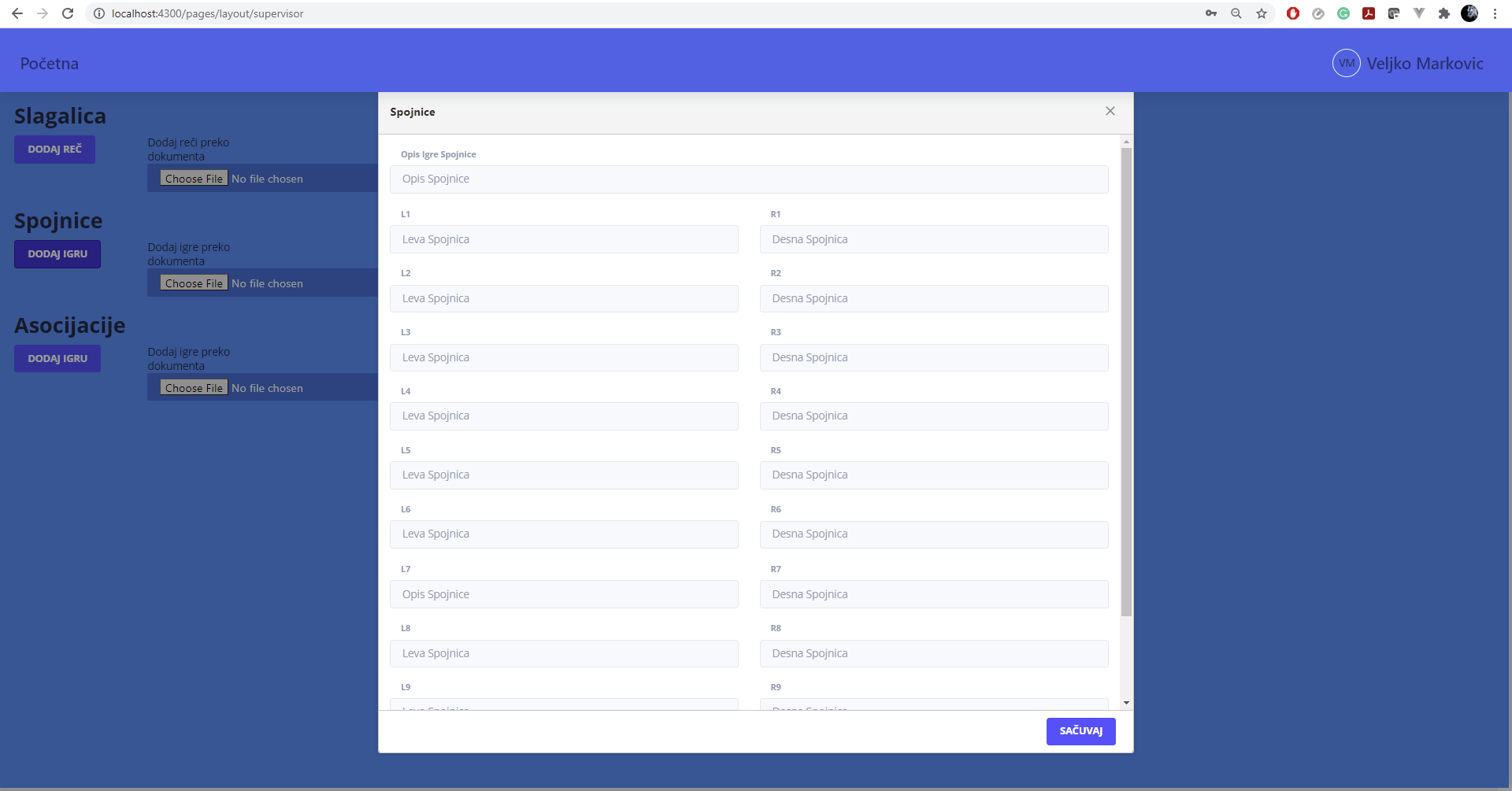
Супервизор када се пријави на систем има три опције. Он може додавати речи које ће се сматрати за исправне приликом такмичења у игри Слагалица, може додавати игре спојница и може додавати игре асоцијација. Што се тиче додавања речи, супервизору је омогућено додавање појединачне речи притиском на дугме „DODAJ REČ”. У том моменту му се отвара дијалог који садржи форму где је потребно унети реч и сачувати је у систему. Поред појединачног уноса омогућено је и уношење већег броја речи преко документа притиском на дугме „DODAJ REČI PREKO DOKUMENTA”. Унос игре спојница је такође омогућен појединачно и преко документа. Приликом појединачног уноса, супервизору се на екрану појављује дијалог који садржи форму у којој треба попунити све спојнице као и опис саме игре шта је циљ спојити за дате појмове. Осим спојница, супервизор је задужен за додавање и игара асоцијација, такође или појединачно или преко документа. Слично претходно описаним корацима, супервизор може додати појединачну реч кроз дијалог који садржи форму. Форма поседује сва поља која су неопходна за попуњавање једне игре асоцијације, као што су појмови у колонама, њихова решења и коначно решење асоцијације. У случају додавања игара преко документа, и за спојнице и за асоцијације, потребно је притиснути дугме „DODAJ IGRE PREKO DOKUMENTA” у конкретном нивоу за конкретну игру. Такође, корисник је адекватно обевештен о исходу сваке извршене акције путем искачуће поруке.



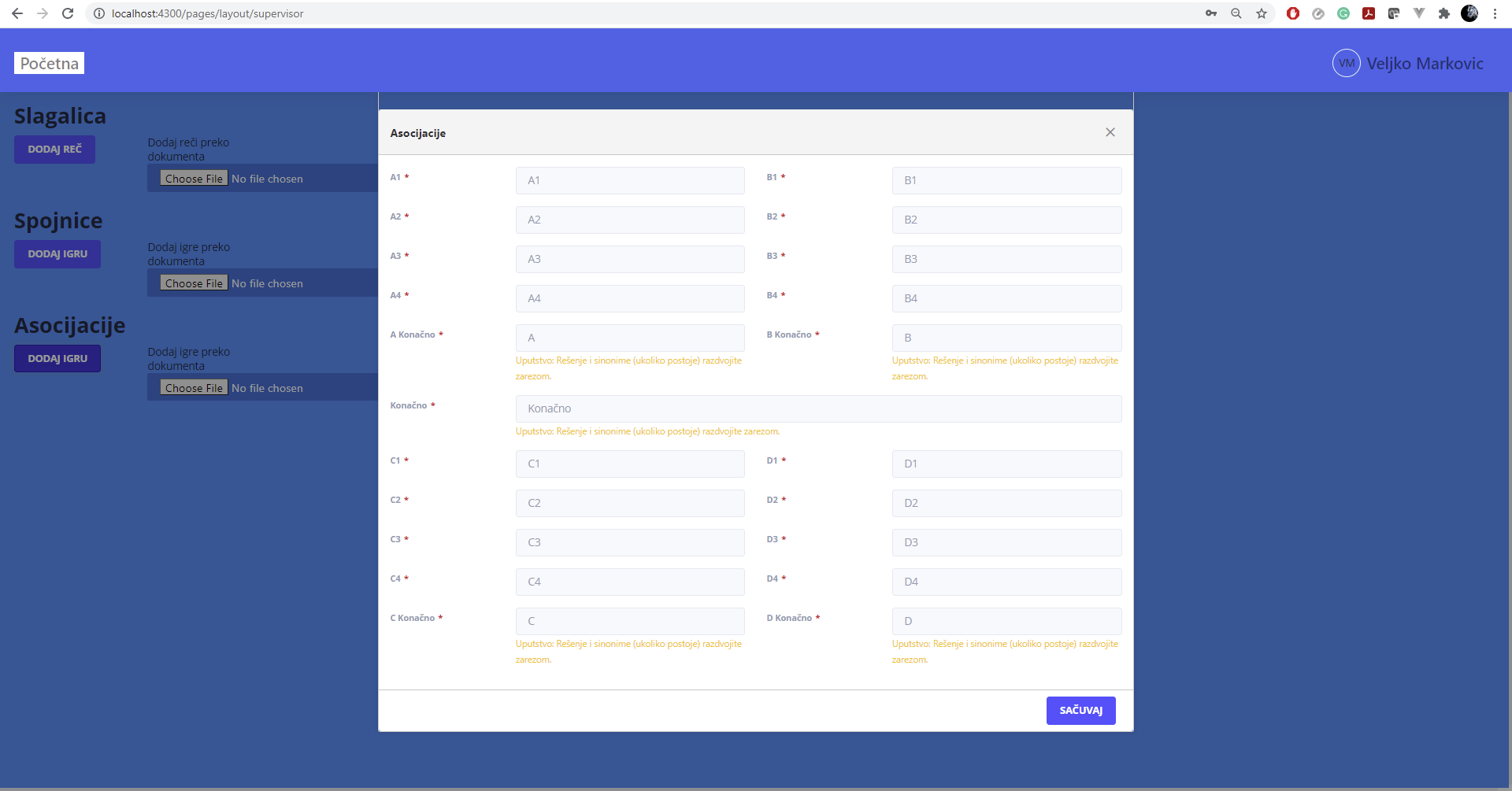
Слика 5.3.1. Супервизор приказ*.*



Слика 5.3.2. Супервизор – Додавање речи*.*



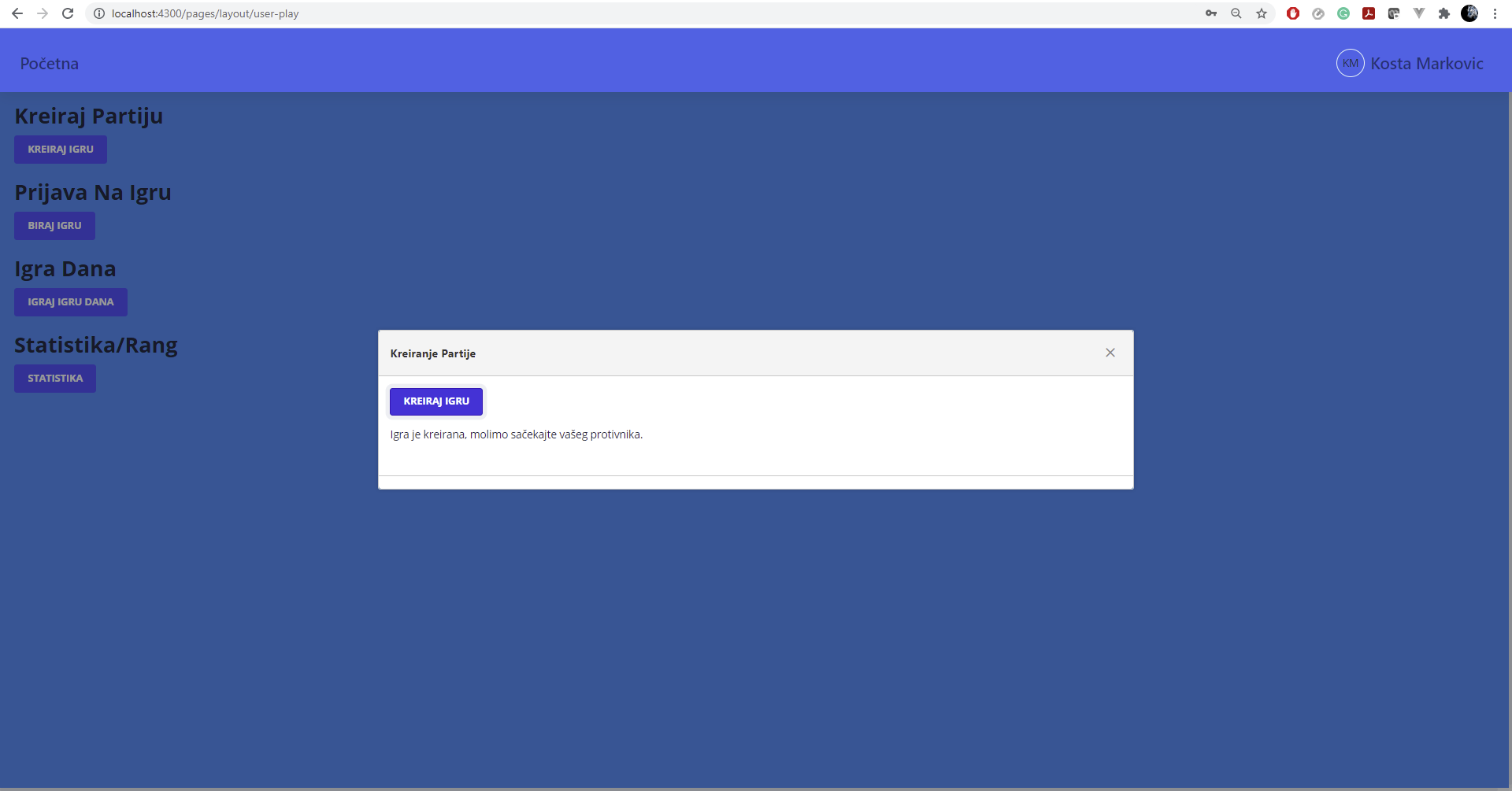
Слика 5.3.3. Супервизор – Додавање спојница*.*



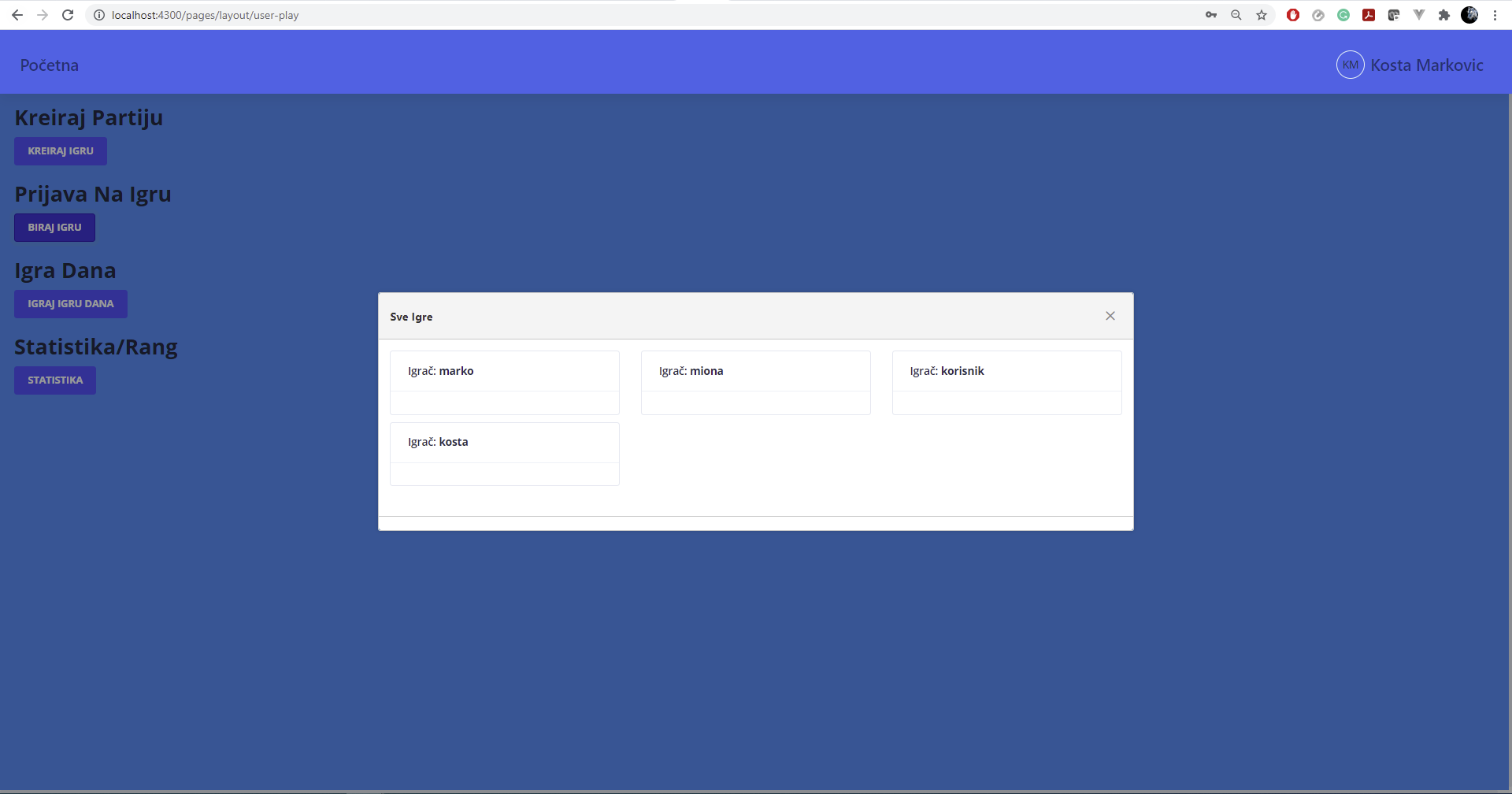
Слика 5.3.4. Супервизор - Додавање асоцијација*.*

* 1. Учесник квиза

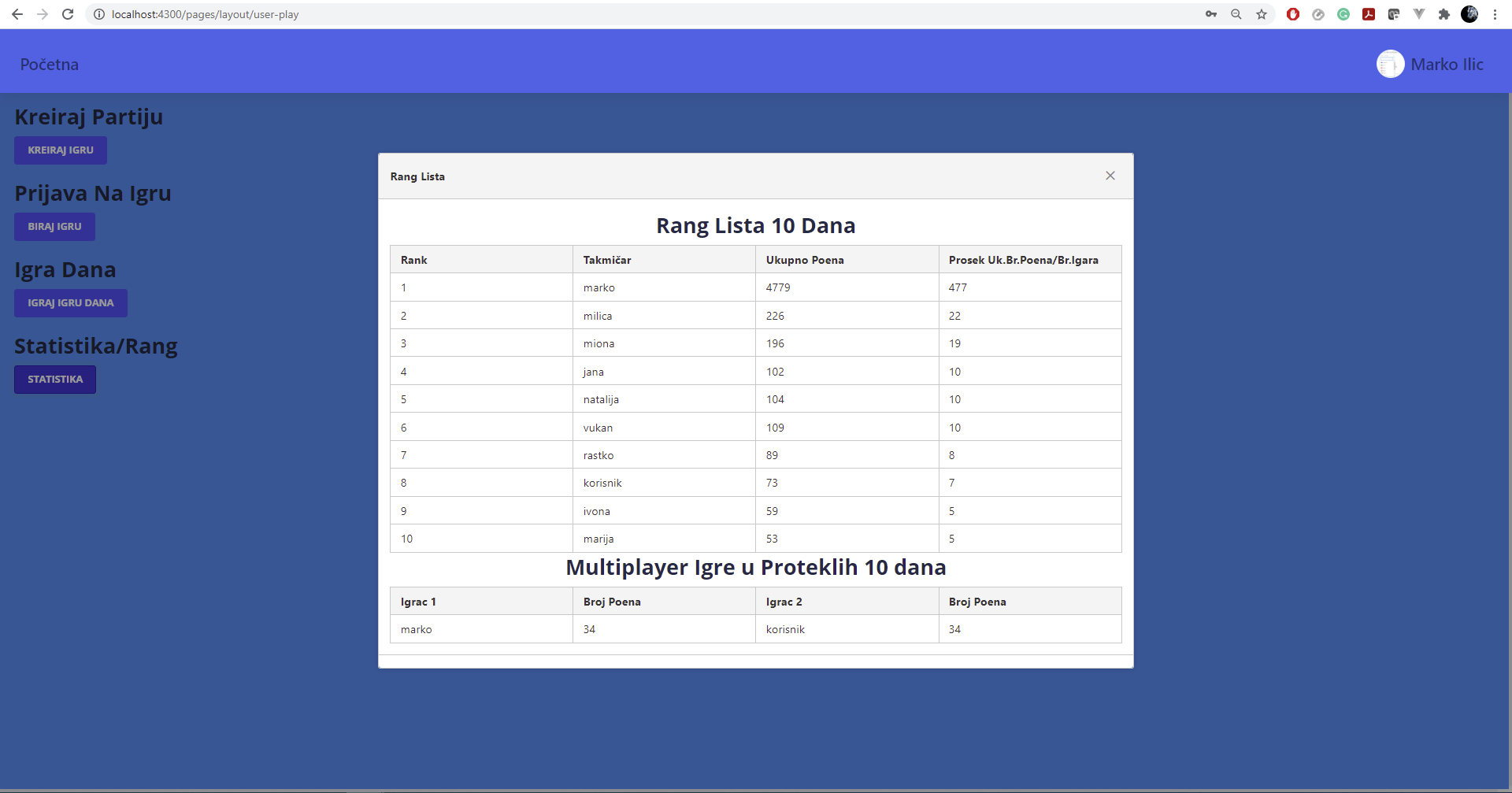
Учесник квиза, када се пријави на систем види екран где може да бира једну од 4 опције. Он може креирати игру коју жели да игра против неког противника, може да бира неку од већ креираних игара где противници чекају некога, може играти игру дана уколико већ није играо и може прегледати статистику. Када креира игру, корисник бира дугме „KREIRAJ IGRU” где му се отвара дијалог са једним дугметом на којем такође пише „KREIRAJ IGRU”. Када корисник притисне то дугме, креира се партија и он заузима место плавог такмичара у будућој игри. У дијалогу му се исписује информација да је игра креирана и да сачека свог противника. Код пријаве на игру код другог такмичара, корисник може отворити дијалог за бирање игре притиском на дугме „BIRAJ IGRU”. Тада му се приказују све игре које тренутно чекају на свог противника. Одабиром неке од њих корисник започиње партију играч против играча. Уколико учесник одабере опцију играња игре дана, отвара му се дијалог где игра дана аутоматски почиње. Последња опција коју корисник има јесте да прегледа статистику, односно ранг листе. То може учинити притиском на дугме „STATISTIKA”. Када одабере ту опцију, кориснику се отвара дијалог који садржи две табеле. Прва табела представља листу 10 најбољих такмичара у претходних 10 дана. Уколико корисник није у првих 10 такмичара, њему се испод прве табеле приказује његов ранг. Друга табела представља све игре корисника које је играо против других такмичара у последњих 10 дана и крајњи исход сваке партије.



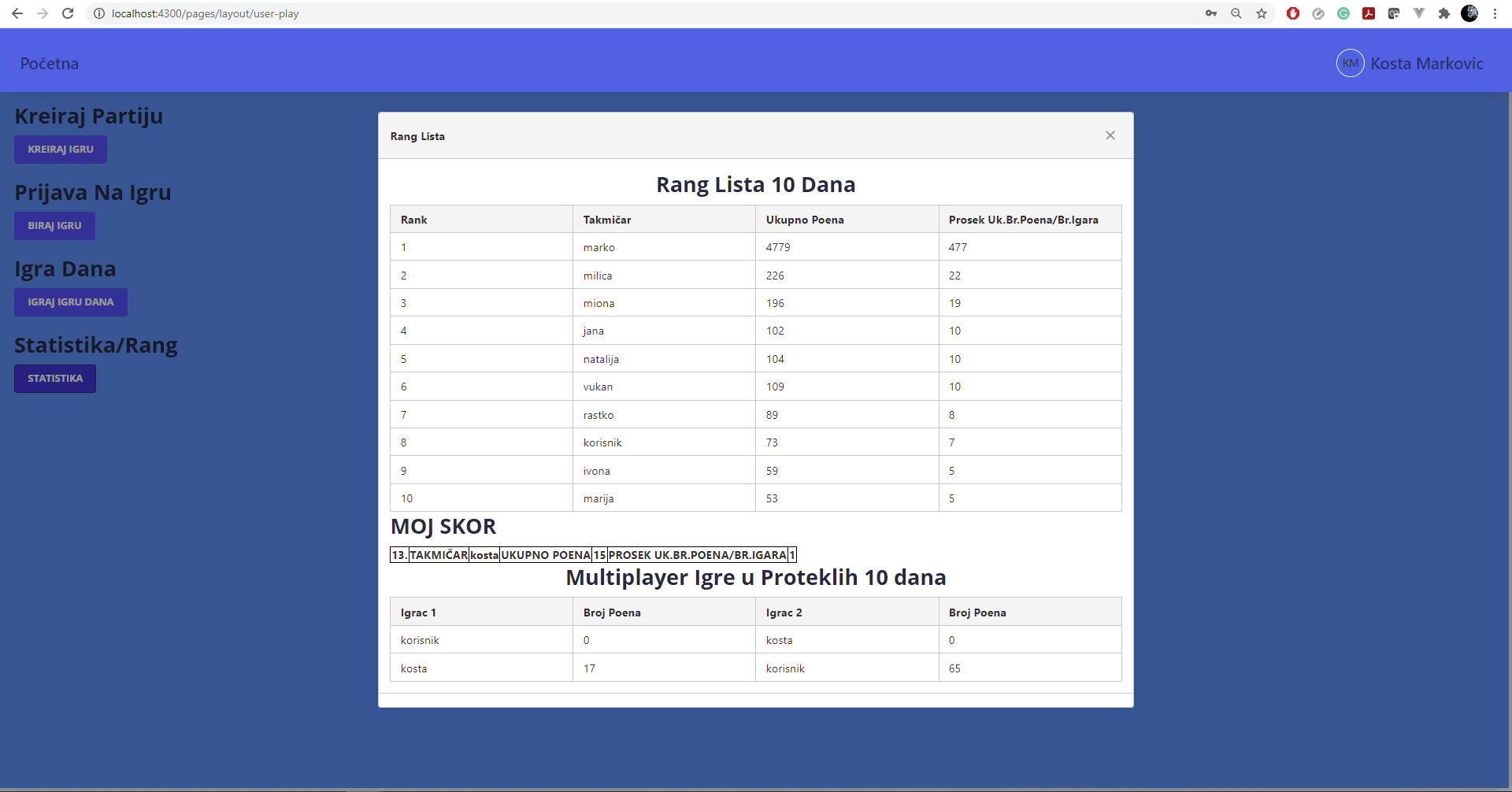
Слика 5.4.1. Учесник квиза - Креирање партије*.*



Слика 5.4.2. Учесник квиза - Бирање партије*.*



Слика 5.4.3. Учесник квиза - Статистика*.*

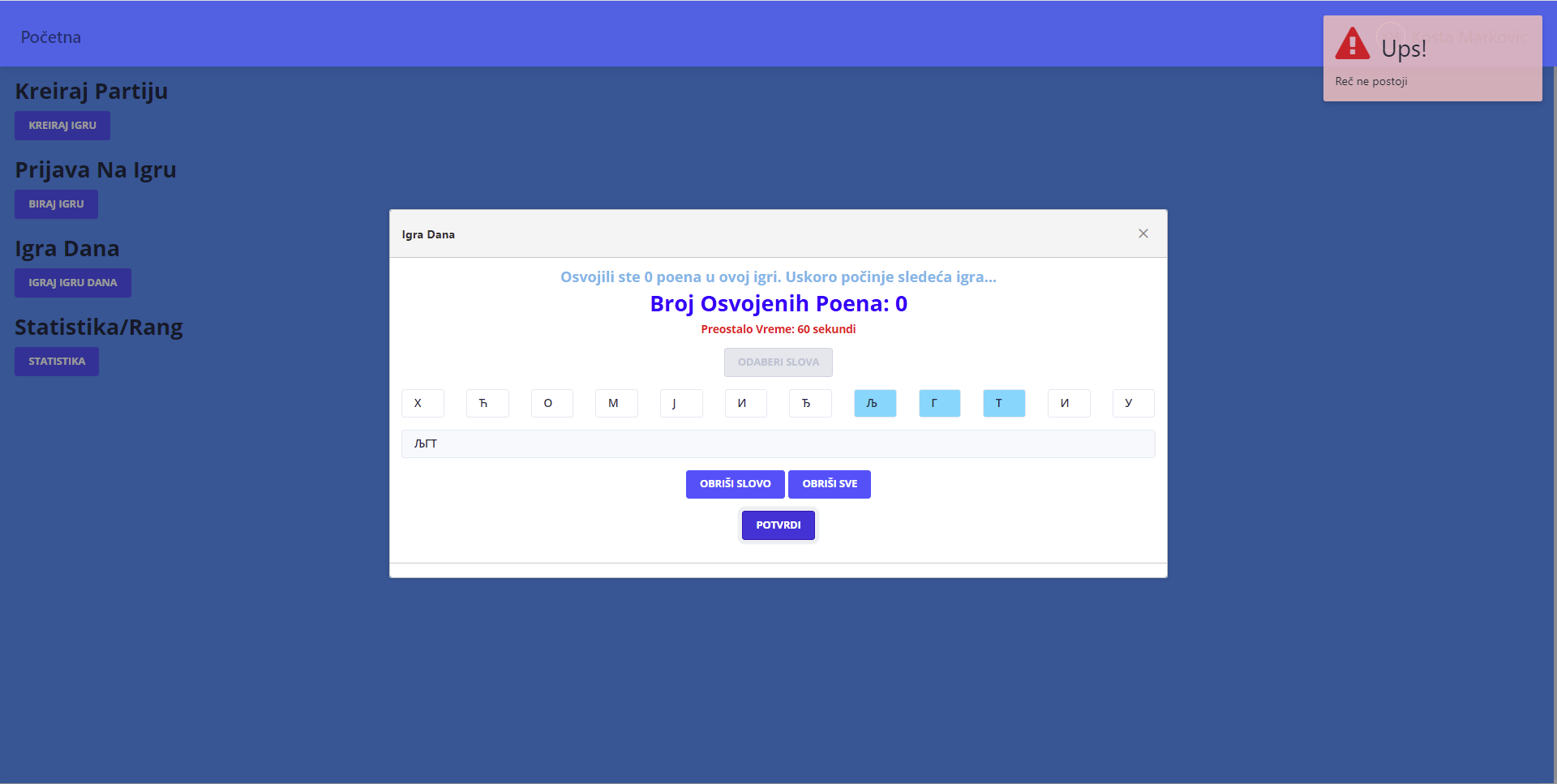


Слика 5.4.4. Учесник квиза – Статистика (ранг ван топ 10)*.*

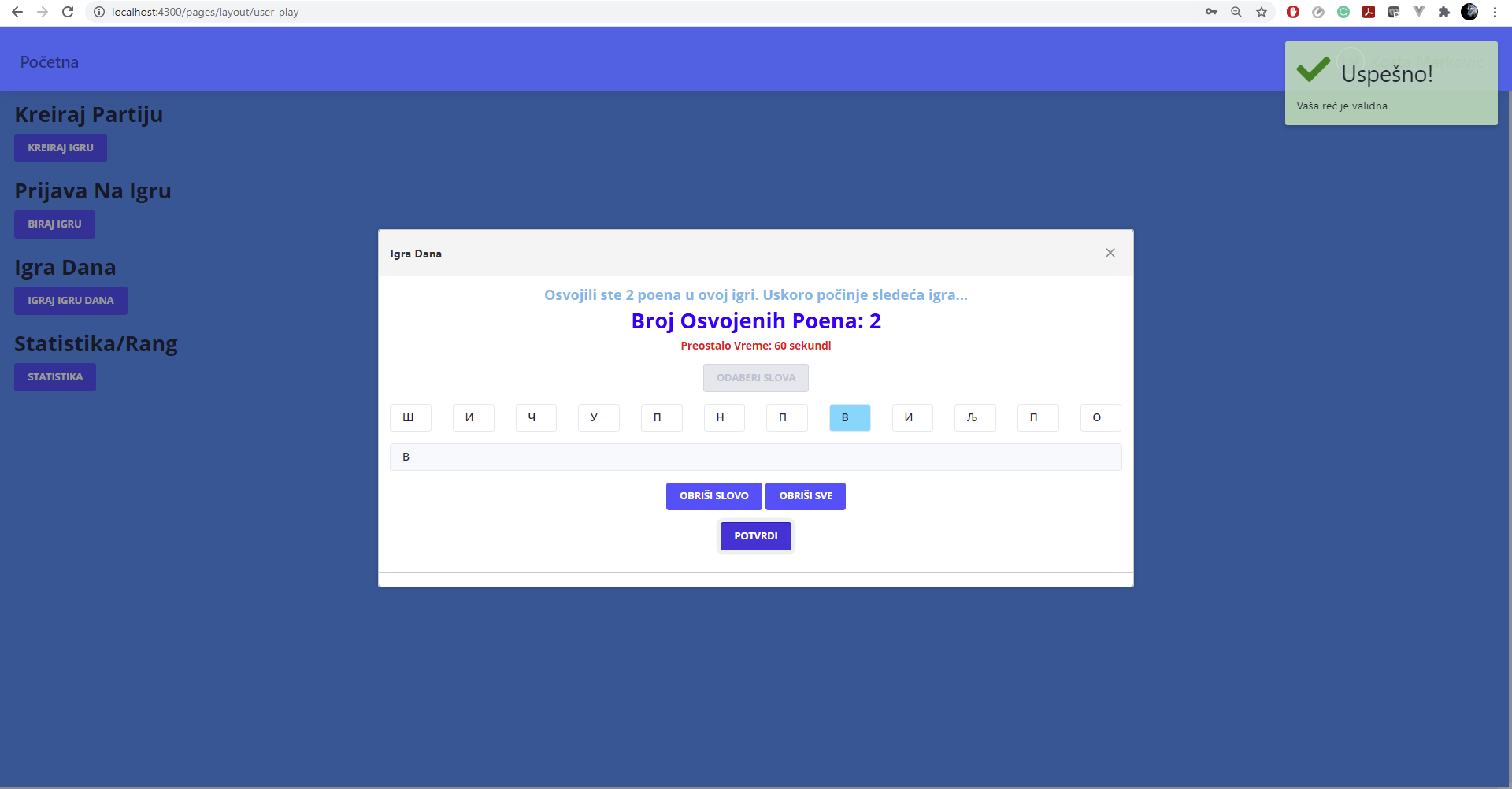
* 1. Дневна игра

Дневна игра представља игру коју учесник игра сам против рачунара, односно без суштински без противника. Игра се може играти једном дневно. Као и игра учесник против учесника, ова игра се састоји из 5 добро познатих игара. Дневна игра се започиње тако што учесник покреће партију притиском на дугме „IGRAJ IGRU DANA”. Када се отвори дијалог, званично почиње игра дана. Игре се засебно стартују притиском на одређено дугме које је везано за конкретну игру (у игри Слагалица дугме је „ODABERI SLOVA”, у игри Мој број дугме је „ODABERI BROJEVE” итд.).

За игру слагалица, такмичар има 60 секунди да споји исправну реч и потврди је. Слова се генеришу случајним одабиром. Бирање слова функционише простим притиском на дугмад са словима која се потом маркирају плавом позадином као знак да је слово одабрано и да није могуће поново бирати исти карактер. Могуће је и обрисати последње одабрано слово или цео унос уколико корисник није задовољан, притиском на дугме „OBRIŠI SLOVO”, односно „OBRIŠI SVE”. Када такмичар потврди реч или уколико време истекне, игра се завршава и исписују се освојени поени у тренутној игри. О исправности и регуларности саме формиране речи корисник је обавештен преко искачуће поруке.

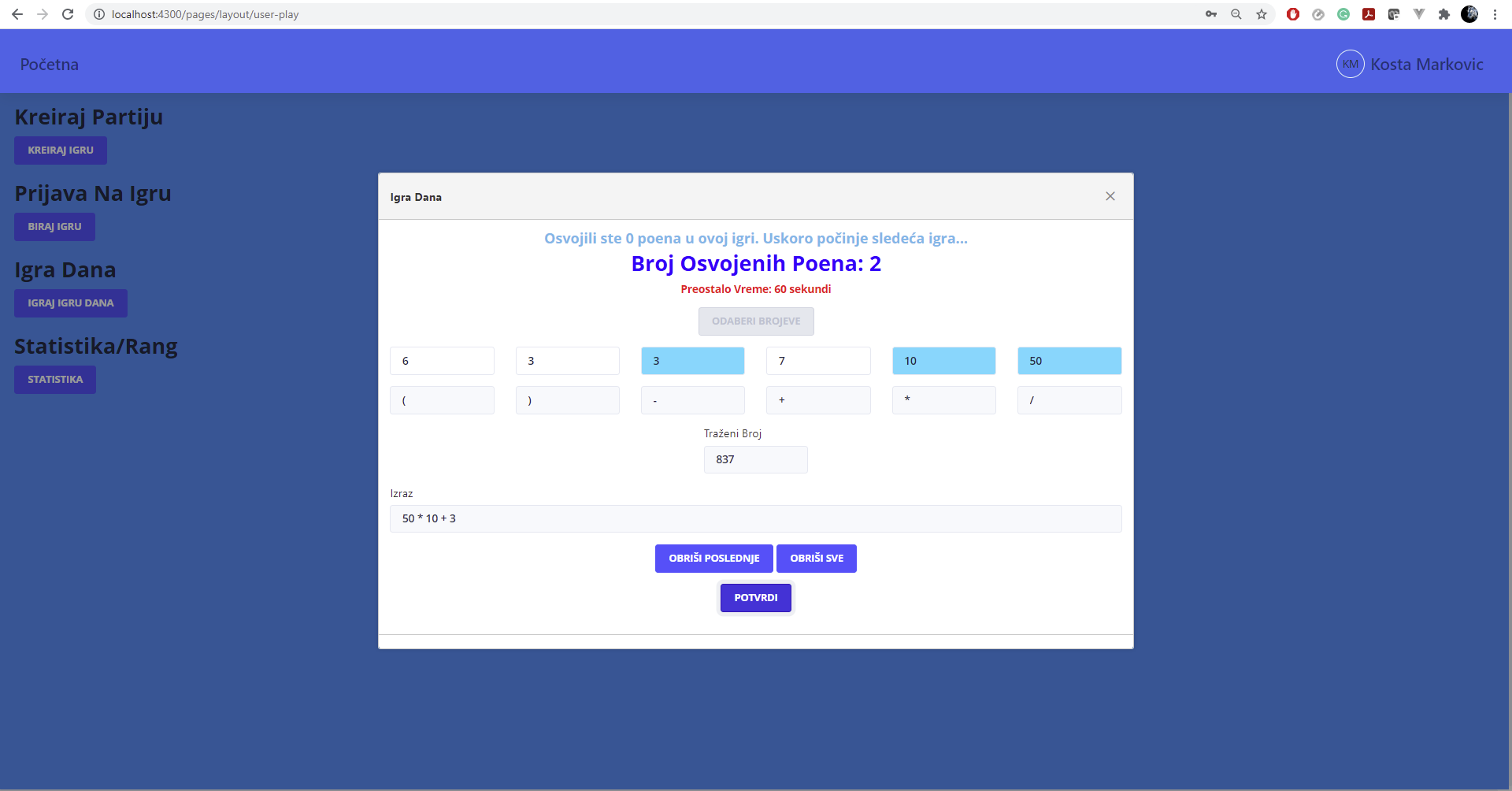


Слика 5.5.1. Дневна игра – Слагалица – Неисправна реч*.*



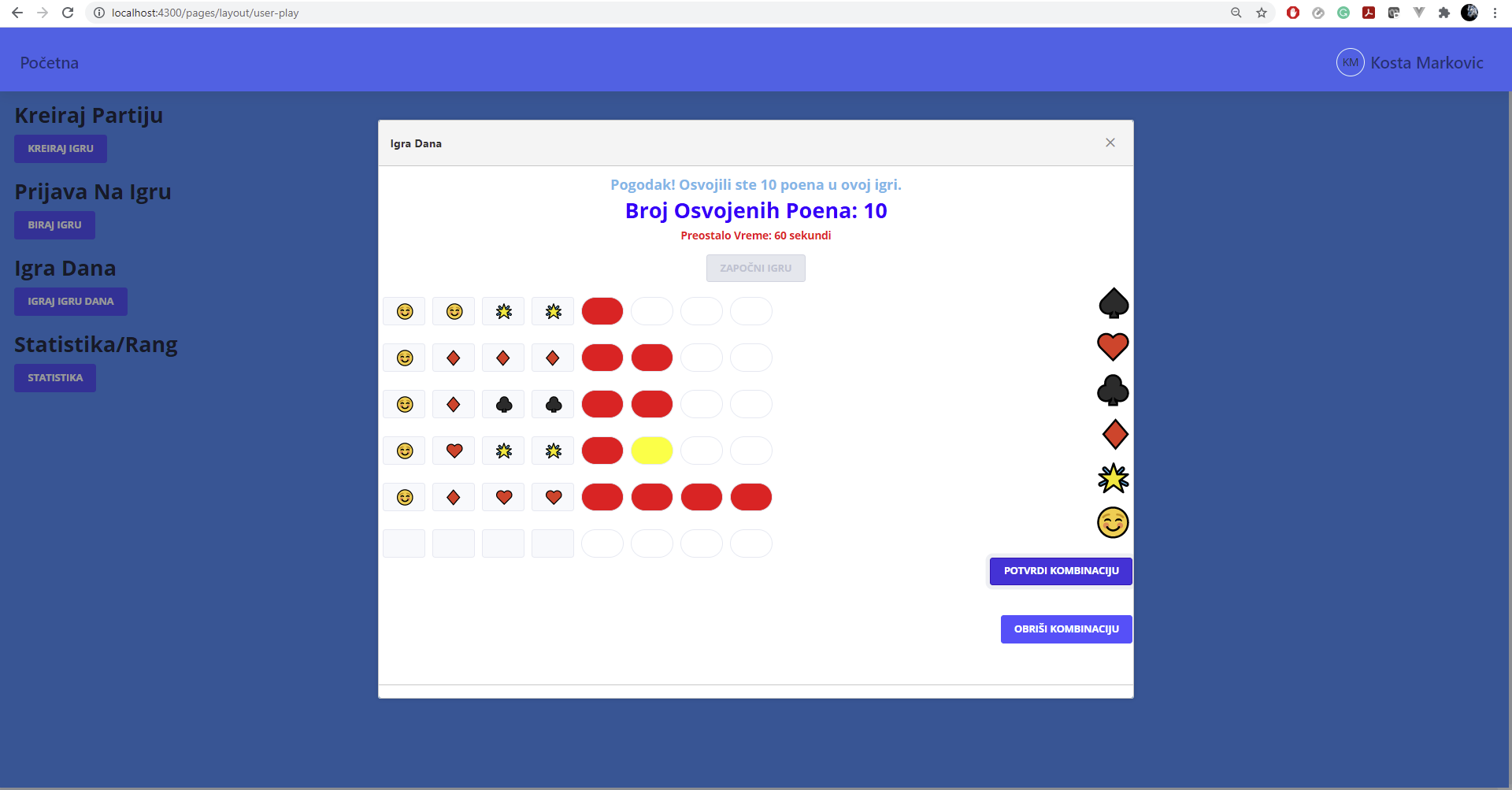
Слика 5.5.2. Дневна игра – Слагалица – Исправна реч*.*

Следећа игра је игра „Мој број“. У овој игри такмичар треба да састави идеално тачан број од понуђених бројева. Када стартује игру, на екрану му се приказују циљани број и бројеви које сме искористити да би дошао до жељеног броја уз најједноставније математичке операције. Притиском на дугме „ODABERI BROJEVE” игра се стартује и временско ограничење је 60 секунди. Простим одабиром бројева формира се израз, одабрани бројеви су потом маркирани плавом позадином и не могу се поново бирати. Када се игра заврши, корисник је обавештен о броју освојених поена.



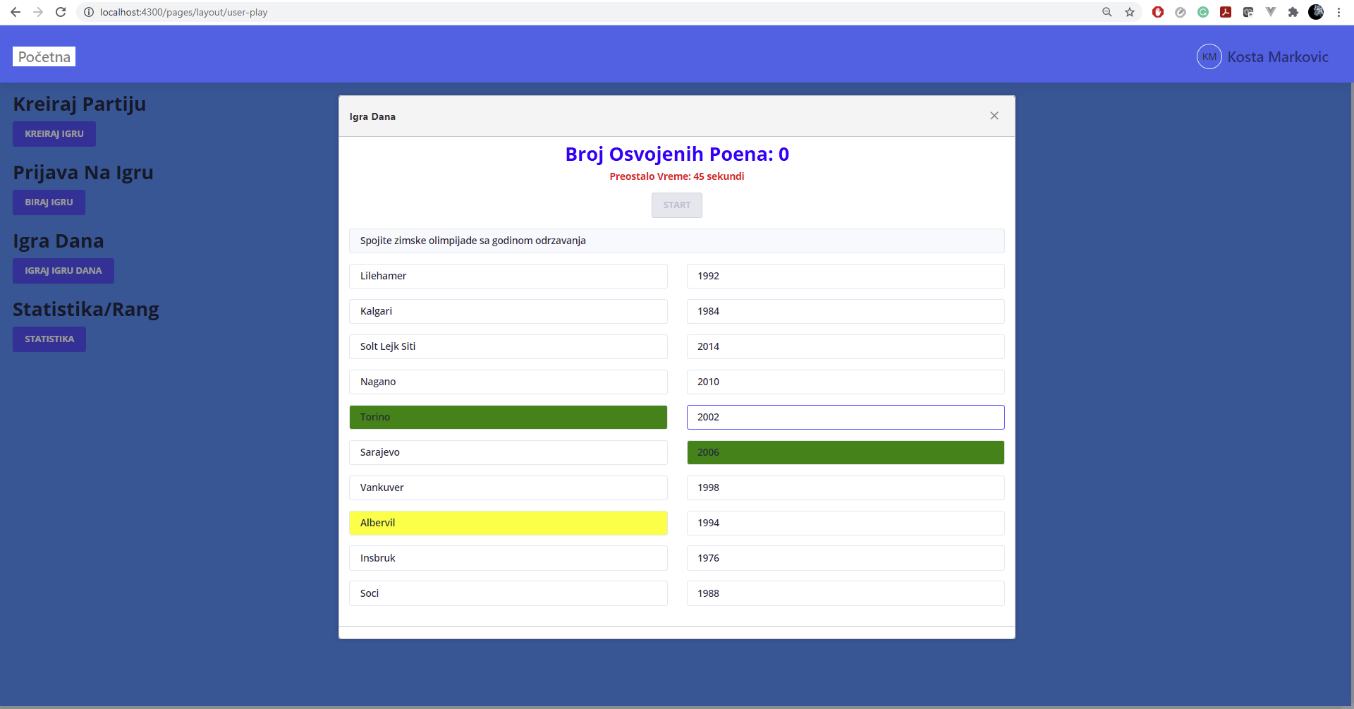
Слика 5.5.3. Дневна игра – Мој Број*.*

Трећа игра звана Скочко почиње притиском на дугме „ZAPOČNI IGRU”. Временски интервал од 60 секунди почиње да тече у том тренутку и учесник има 6 прилика за погађањем случајно изгенерисане комбинације. Са леве стране дијалога се налазе све комбинације које учесник бира у току саме игре. Поред сваке потврђене комбинације исписује се и стање саме комбинације, црвена и жута поља, која представљају број погођених симбола на правом месту и број погођених симбола који нису на правом месту. Са десне стране дијалога се налазе симболи које је могуће одабрати за комбинацију. Бирање појединачних комбинација се врши притискањем симбола и на крају потврдом комбинације. Играч може брисати комбинацију која није потврђена уколико није задовољан. На крају игре, играч је обавештен о броју освојених поена.



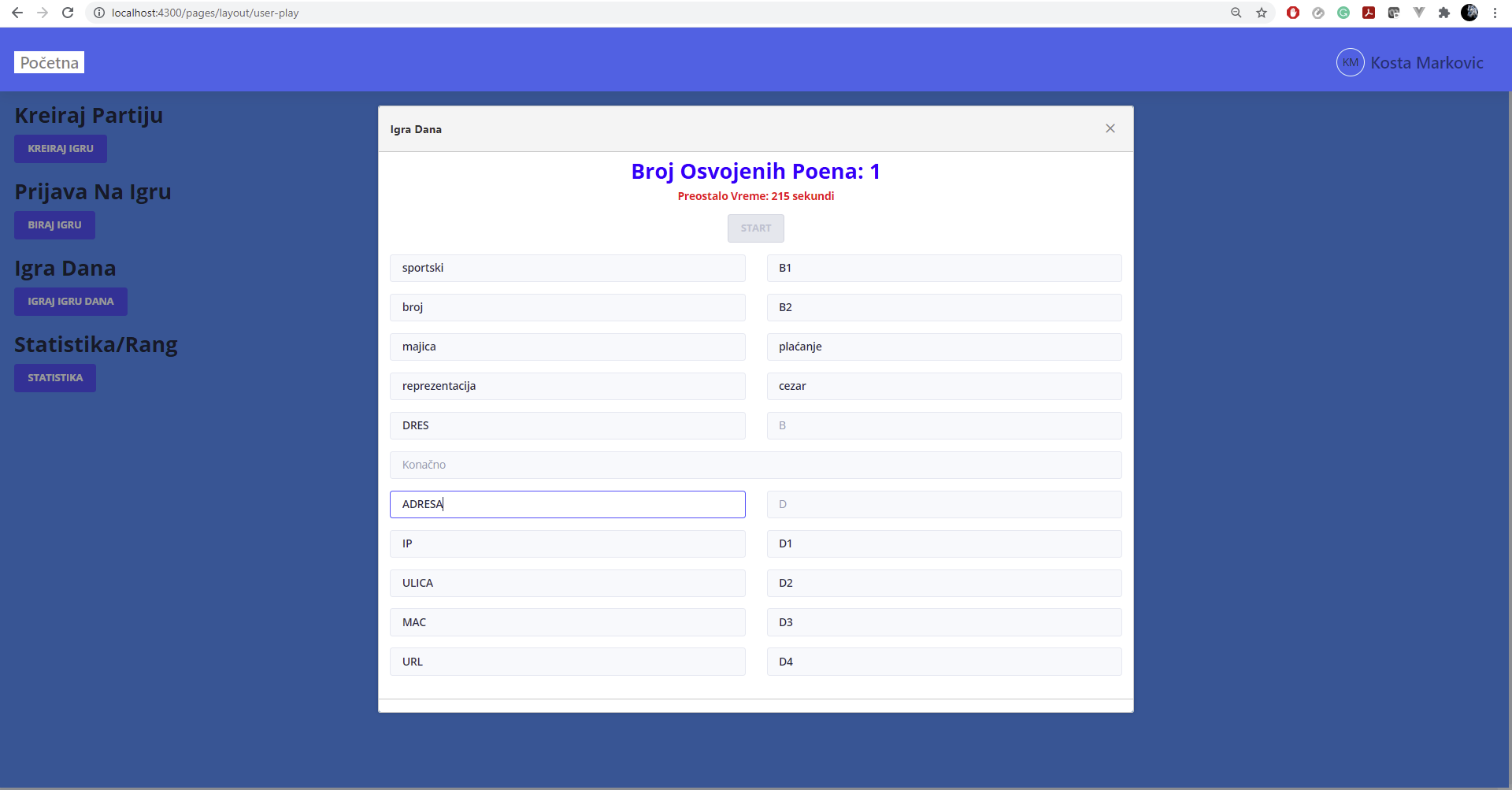
Слика 5.5.4. Дневна игра – Скочко*.*

Спојнице се попуњавају појмовима након што такмичар покрене игру притиском на дугме „START”. У том моменту се на екрану појављује опис саме игре спојнице и шта је потребно спојити као и сви појмови у левој и десној колони у насумичном поретку. Игра траје 60 секунди. Игра се игра тако што се тренутни појам са леве стране обележи жутом позадином и од такмичара се очекује да притиском на неки од појмова са десне стране упари тренутни појам. Уколико такмичар погоди појам, упарени појмови на левој и десној страни се обележавају зеленом позадином и прелази се на следећи појам у низу са леве стране. Ова логика се извршава све док се не дође до краја леве колоне или док не истекне време за решавање спојница. На крају игре играч је обавештен о броју освојених поена.



Слика 5.5.5. Дневна игра – Спојнице*.*

Последња игра у низу јесте игра Асоцијације. Игра се игра тако што се одабиром поља отварају појмови сакривени иза самих поља. Такмичар може отварати поља без престанка због тога што се налази у режиму играња против рачунара. Циљ игре јесте погодити решење коначне асоцијације на основу коначних решења појединачних колона. Игра траје 240 секунди. Када се игра заврши кориснику се исписује број освојених поена, а потом и укупан број освојених поена у игри дана.



Слика 5.5.6. Дневна игра – Асоцијације*.*

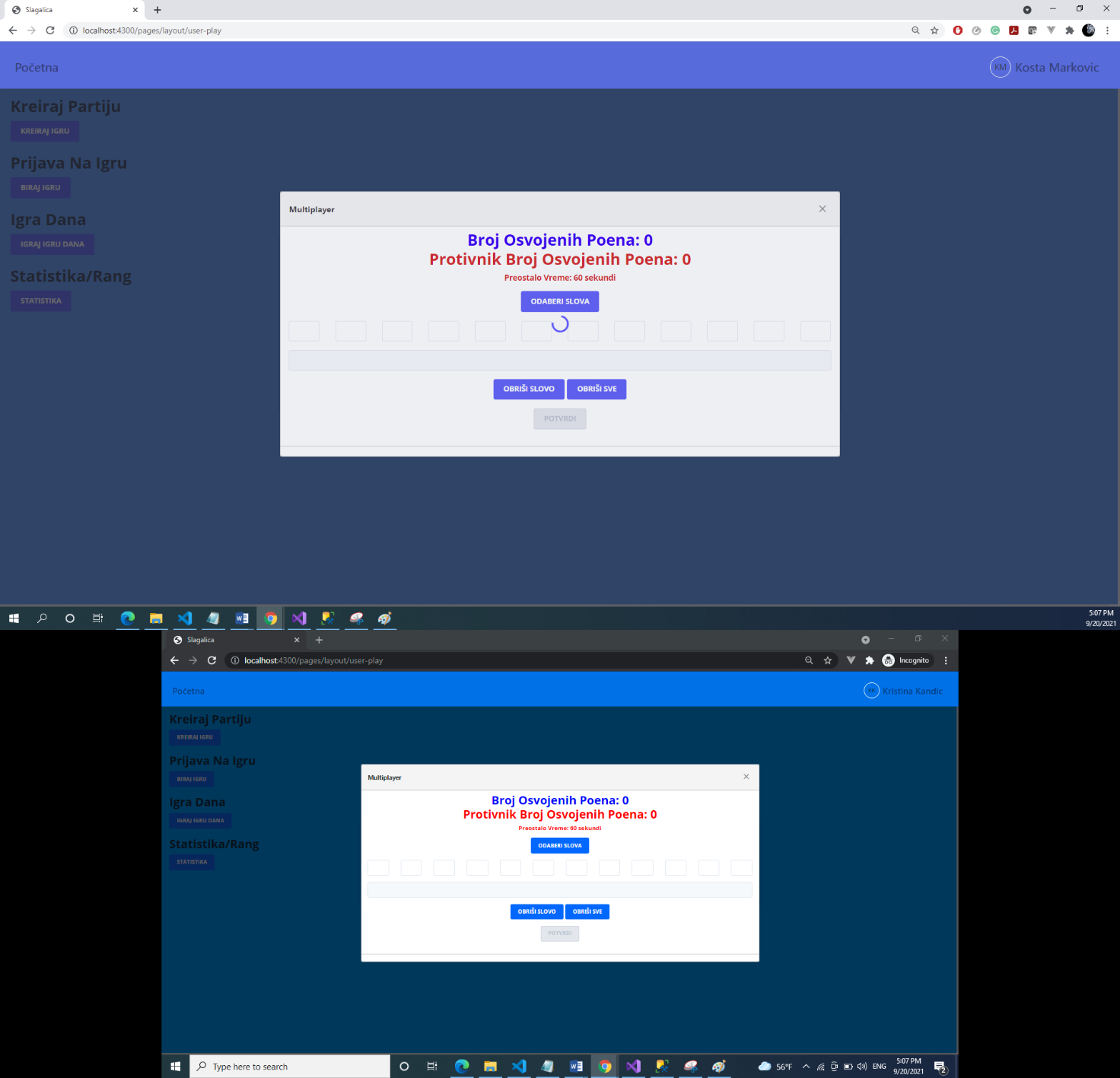


Слика 5.5.7. Дневна игра – Крај игре*.*

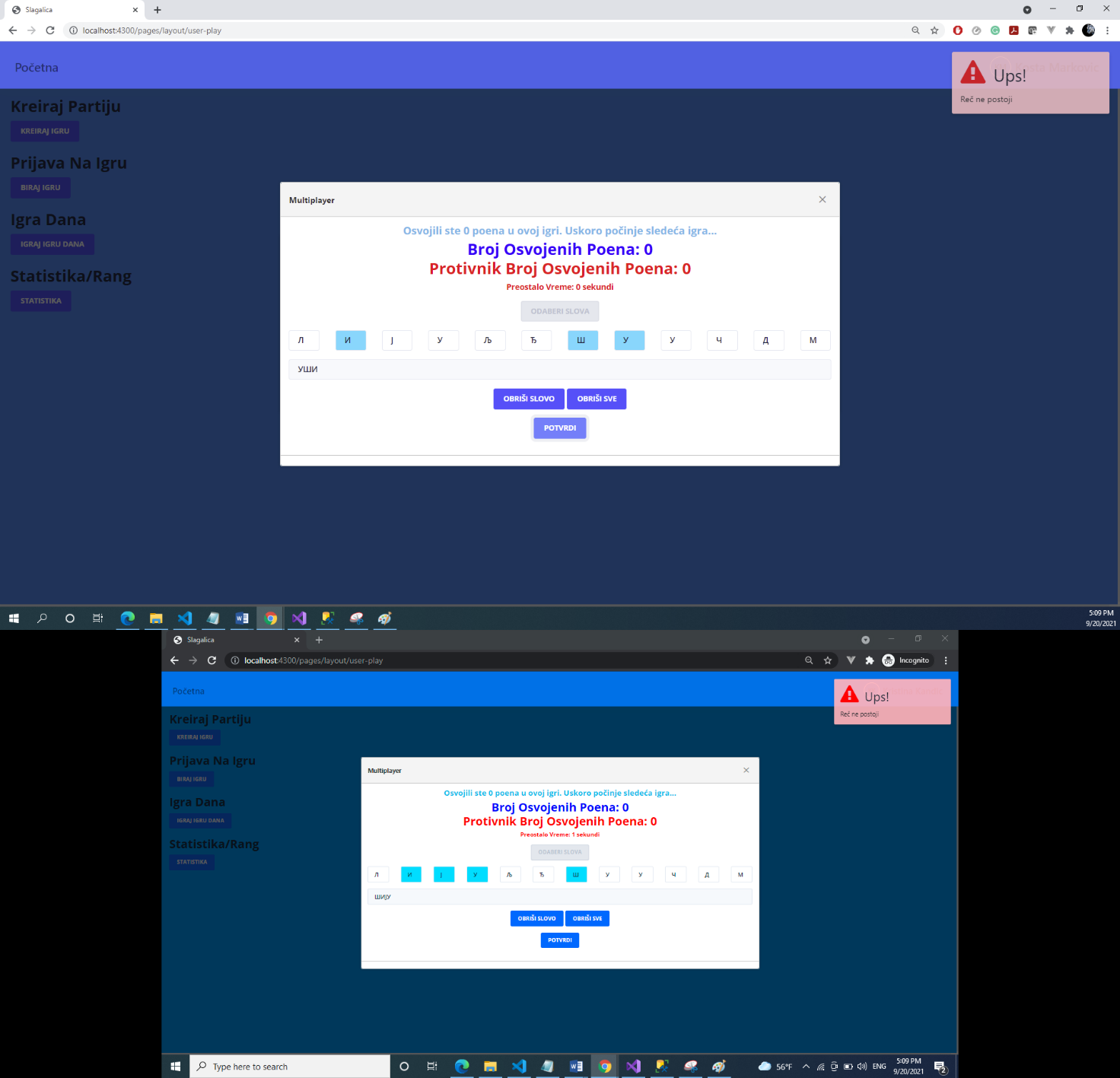
* 1. Игра учесник против учесника

Игра учесник против учесника почиње када се корисник који је креирао игру и корисник који се пријавио на исту аутоматски споје. Они се такмиче у пет већ поменутих игара. Логика и циљ сваке игре је исти као и дневној игри, уз неке промене у функционисању и предавању контроле приликом промене потеза такмичара.

Игра слагалица почиње тако што плави такмичар бира слова. Када одабере слова, оба такмичара добијају исте карактере од којих треба да формирају најдужу могућу исправну реч. Када један такмичар потврди реч, њему се екран замрзава док чека другог такмичара да потврди реч или док не истекне време. Када се игра заврши такмичари су обавештени о броју освојених поена, након чега се прелази на следећу игру и глобални скор обојици такмичара се потом освежава.

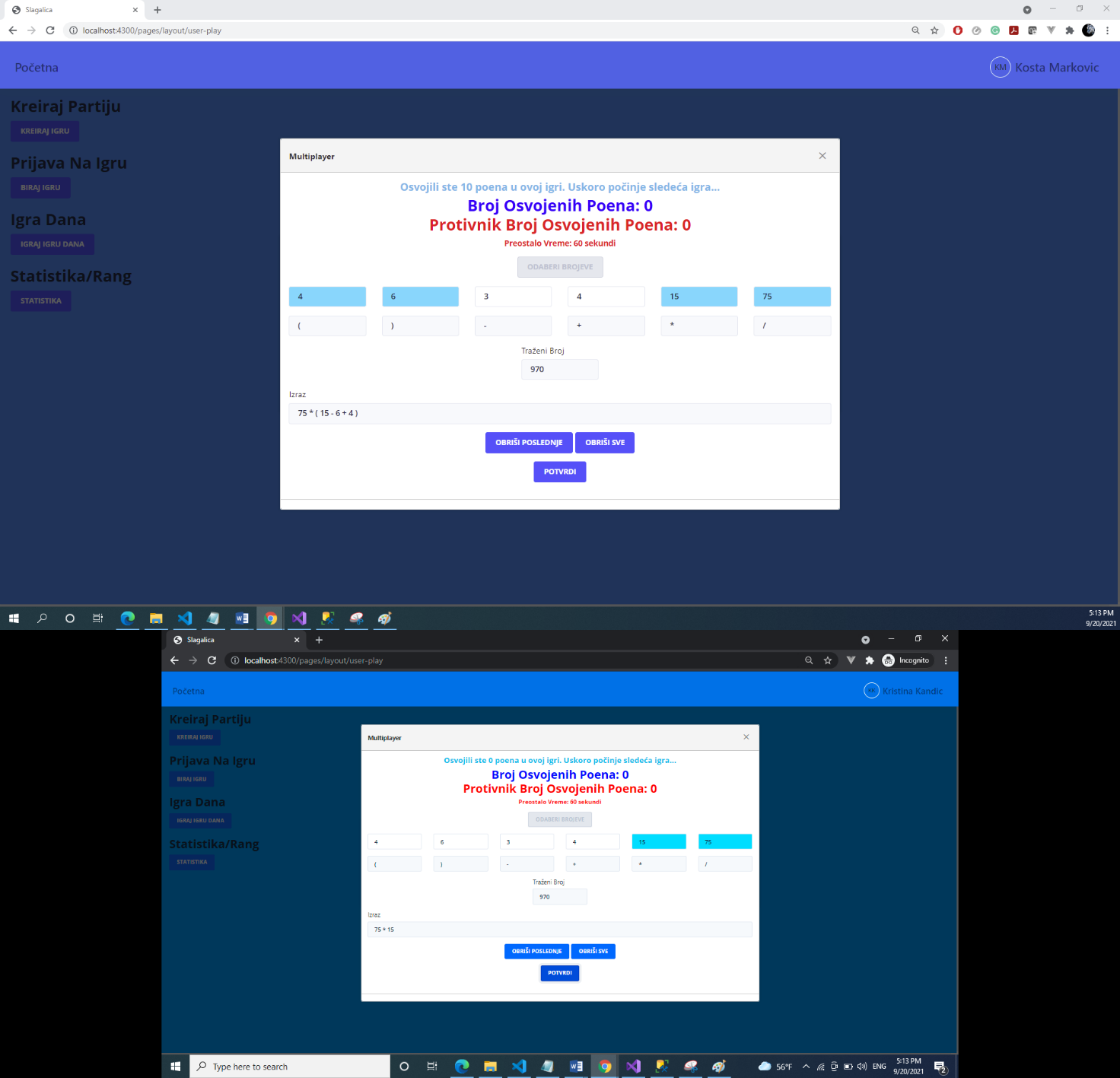


Слика 5.6.1. Учесник против учесника – Слагалица почетак*.*



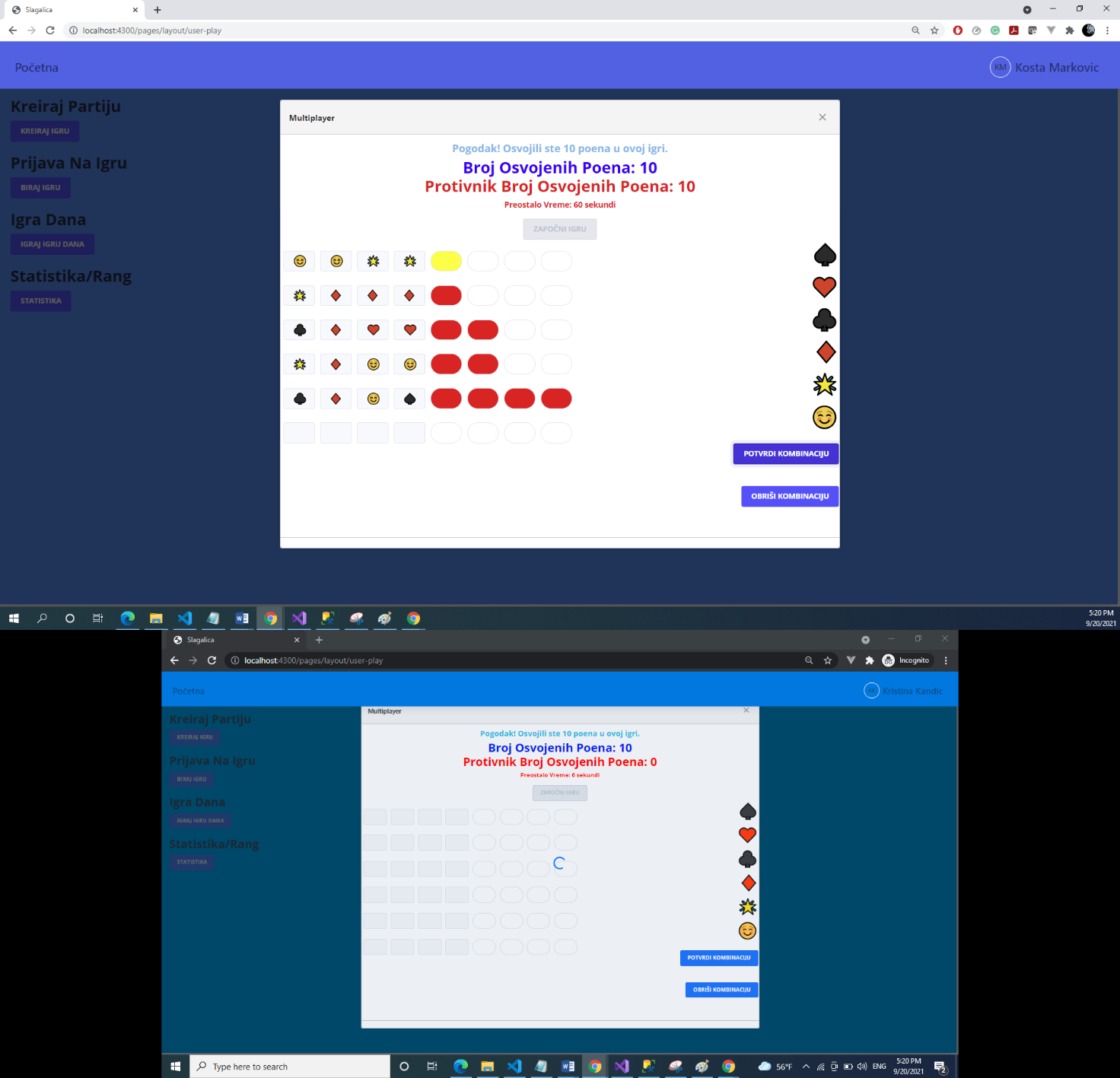
Слика 5.6.2. Учесник против учесника – Слагалица крај*.*

Следећа игра је игра Мој број. Игра функционише по истом принципу као и претходна игра. На потезу је прво плави такмичар, а то значи да он бира бројеве и стартује игру. Када неко од такмичара потврди број, екран се замрзава и тај такмичар чека другог такмичара на његову потврду израза или евентуални истек времена. Када се игра заврши, такмичарима се исписује број освојених поена у игри и прелази се на екран за следећу игру. Ту се освежава глобални скор обојици такмичара.



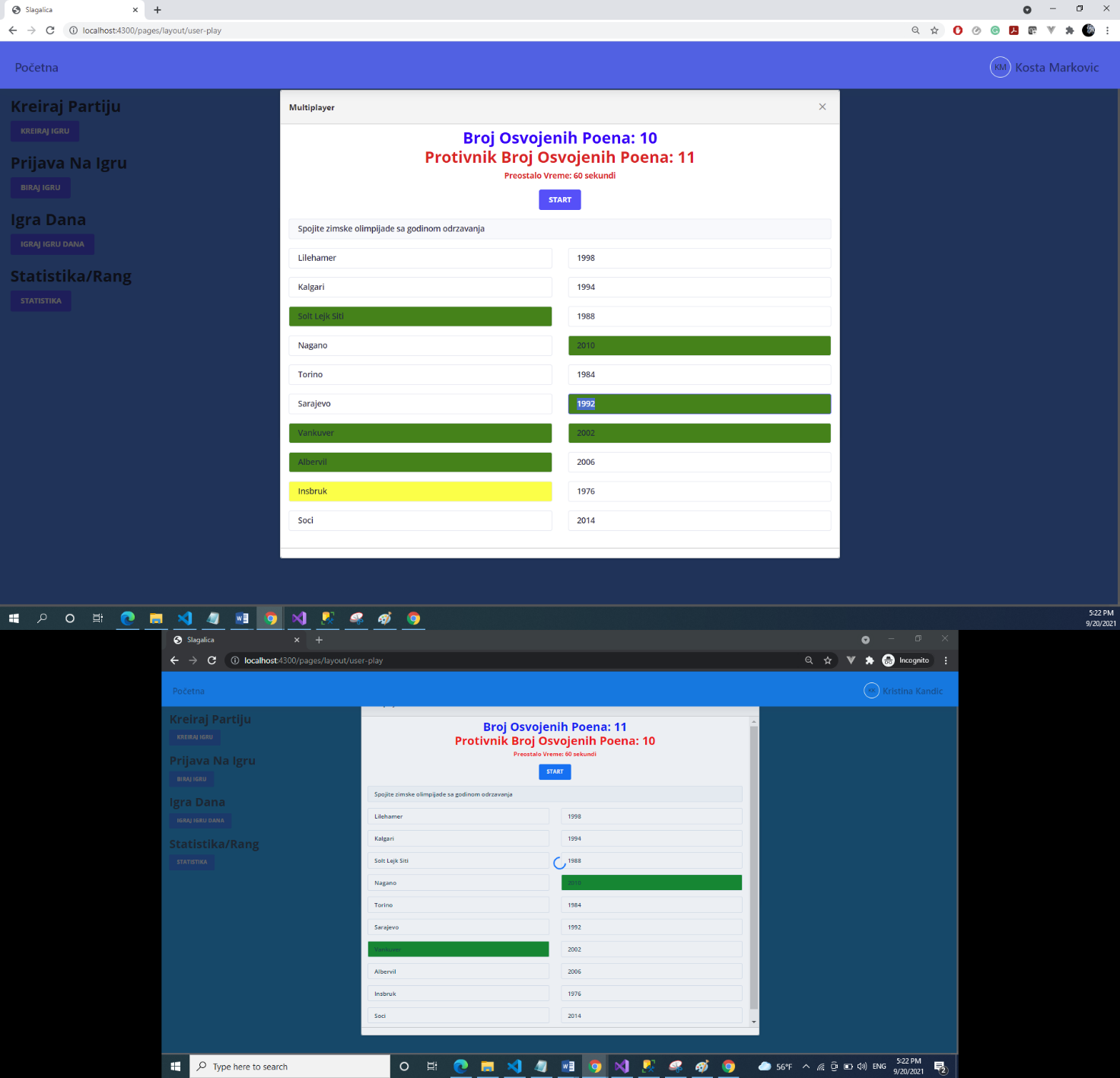
Слика 5.6.3. Учесник против учесника – Мој Број*.*

Скочко је назив игре која следи после игре Мој број. Прво игра плави такмичар који покреће игру док црвени такмичар чека блокиран. Када плави такмичар заврши погађање своје комбинације или истекне време, прелази се на црвеног такмичара. Тада плави такмичар постаје блокиран док црвени такмичар преузима контролу и игра своју рунду. Када се заврше обе рунде освежава се глобални скор и прелази се на следећу игру.



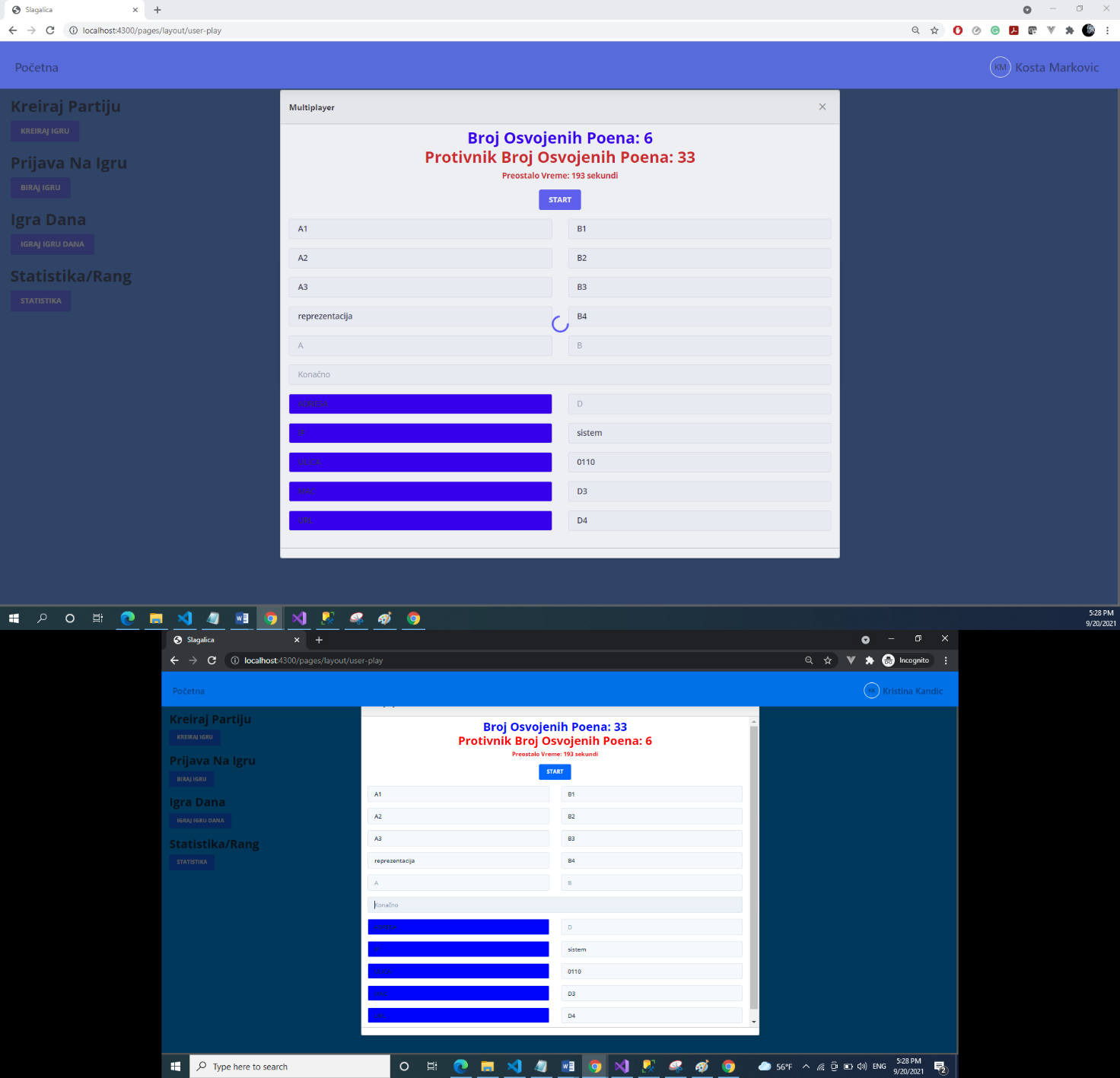
Слика 5.6.4. Учесник против учесника – Скочко*.*

Четврта игра по реду јесу Спојнице. Прво игра плави такмичар који спаја појмове. Уколико није спојио све појмове, контрола се пребацује на црвеног такмичара који покушава да споји појмове који су преостали. Након што се и то заврши, прелази се на другу рунду, где црвени такмичар има свој потез. Након што црвени такмичар одигра рунду, уколико није спојио све појмове, ради се слично као и у претходном случају, контрола се пребацује на противника, односно у овом случају на плавог такмичара, који добија прилику да споји појмове које црвени такмичар није успео. Након што се и тај последњи режим игре заврши, глобални скор се освежава обојици такмичара и прелази се на последњу игру.

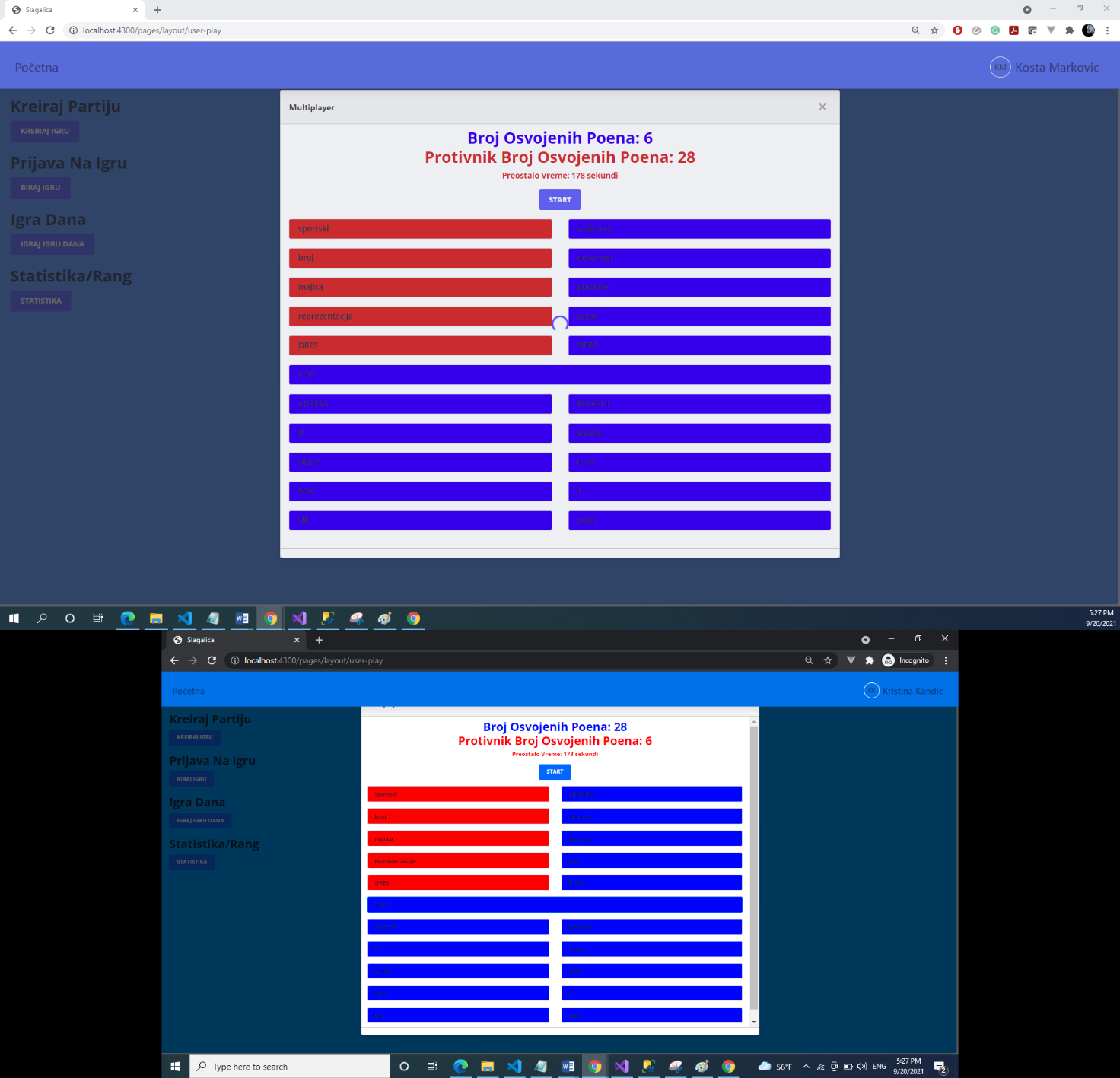


Слика 5.6.5. Учесник против учесника – Спојнице*.*

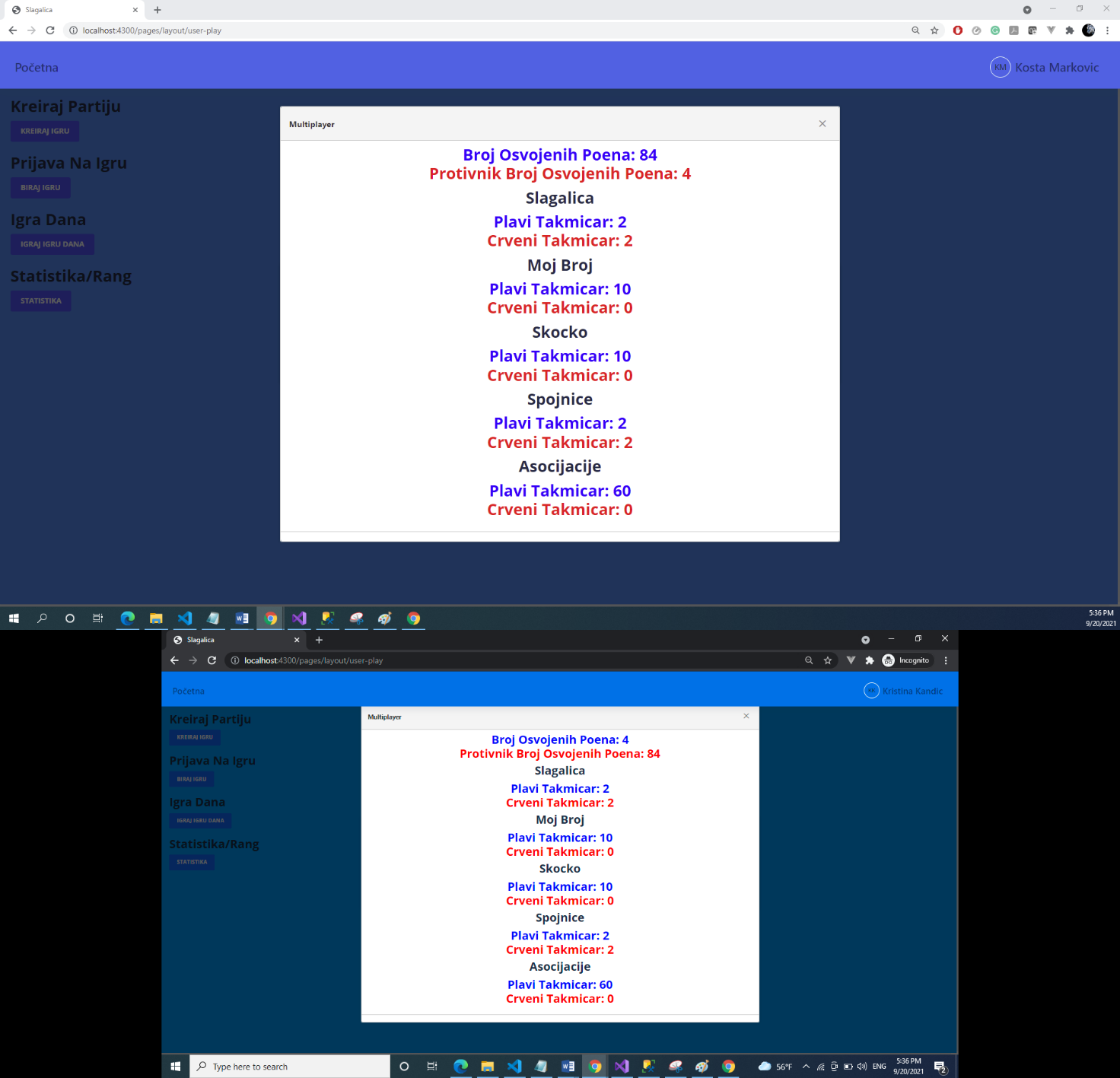
Последња игра у борби јесте игра Асоцијације. Ова игра се игра наизменично. Прву рунду стартује плави такмичар који је на потезу. Он може отворити највише једно неотворено поље и потом погађати коначно решење неке од колона или коначно решење целе асоцијације. Свако отварање поља се аутоматски приказује и другом такмичару, као и сваки погодак колоне или евентуалног поготка коначног решења. Игра се игра такође у две рунде, где другу рунду стартује црвени такмичар у новој игри асоцијација. Глобални скор се у овој игри освежава обојици такмичара након сваког поготка колоне или коначног решења. Када се обе рунде заврше, такмичарима се приказује коначан скор и број освојених поена у свакој игри.



Слика 5.6.6. Учесник против учесника – Асоцијације – Игра у току*.*



Слика 5.6.7. Учесник против учесника – Асоцијације – Завршена рунда*.*



Слика 5.6.8. Учесник против учесника – Завршена игра*.*

1. Закључак

Кроз овај рад може се упознати са базичним, као и напредним концептима развијања модерних интернет апликација. Интерактивне игре и њихове подкатегорије у којима је омогућено да такмичари „уживо” играју један против другог постају све популарније у данашње време. Велико интересовање корисника за играњем ове врсте игара довела је до експанзије самих технологија које омогућавају развој оваквих апликација. На тржишту се непрестано појављују нове технологије и библиотеке које помажу у решавању специфичних изазова који су везани за проблематику интерактивних игара. За реализацију овог рада одабране су тренутно једне од најпопуларнијих интернет технологија које иза себе имају велику заједницу, стабилан и константан напредак.

Имплементација серверске и клијентске апликације у овом раду може помоћи многима да лакше разумеју како данашње модерне интернет апликације функционишу као целина. По угледу на овај рад, моћи ће да се имплементирају разна слична решења у зависности од потреба корисничких захтева. Такође, овај рад може помоћи другима који желе да издвоје неке посебне делове и имплементацију неког од три слоја система. Кроз овај рад приказани су сви главни изазови са којим се суочавају креатори интерактивних игара и квизова.

На основу приложене имплементације решења могуће је надоградити многе друге функционалности. Један од примера јесте да се увек на једноставан начин могу додати нове игре у систем, где је потребно имплементирати саму логику нове игре, док се за остале концепте може угледати на већ реализовано решење. Такође, могуће је додати нове типове корисника, или додати нове могућности за већ постојеће типове корисника. На једноставан начин се могу вршити замене одређених софтверских компонената и модула у имплементацији јер је већина кључних аспеката реализована преко пројектног узорка *Dependency Injection* који помаже у динамичној имплементацији решења.

Пошто је серверска страна и њена логика имплементирана у технологији *.NET Core*, *API* је могуће искористити на било којој платформи и било ком оперативном систему. Овај део имплементације није неопходно користити само за интернет апликације. Могућа је интеграција било ког типа клијентске апликације која подржава *REST* комуникацију. На пример уместо клијентске интернет апликације, може се искористити клијентска мобилна апликација која би се евентуално повезала са имплементираним серверским делом.

Литература

1. Michael David, *.NET Programming for Beginners*, 2020.
2. Dimitris Loukas, *C# 8 and .NET Core 3.1 Recipes*, 2020.
3. Milo Tomašević, *Algoritmi i structure podataka*, 2008.
4. Adam Freeman, *Pro Angular*, 2017.
5. Kyle Simpson, *You don’t know JS,* 2014.
6. Dražen Drašković, Materijali iz predmeta Programiranje Internet Aplikacija (materijali sa predavanja) - (<https://rti.etf.bg.ac.rs/rti/ir4pia/materijali/predavanja/si4pia_Angular7-P1-P3.pdf>, 09.08.2021.)

Списак слика

[Слика 4.1.1. Шема базе података из алата *SQL Server.* 12](#_Toc83225869)

[Слика 4.2.1.1. Структура серверске апликације*.* 14](#_Toc83225870)

[Слика 4.2.1.2. Део програмског кода из класе *Startup.cs.* 15](#_Toc83225871)

[Слика 4.2.2.1. Део програмског кода из класе *AuthenticationController.cs.* 16](#_Toc83225872)

[Слика 4.2.2.2. Део програмског кода из класе *UserController.cs.* 16](#_Toc83225873)

[Слика 4.2.2.3. Део програмског кода из класе *OrganizationController.cs.* 17](#_Toc83225874)

[Слика 4.2.3.1. Део програмског кода из интерфејса *IUserInterface.cs.* 17](#_Toc83225875)

[Слика 4.2.4.1. Додавање апликационог контекста базе *ApplicationDbContext.* 18](#_Toc83225876)

[Слика 4.2.4.2. Проширивање *ApplicationDbContext* класе*.* 18](#_Toc83225877)

[Слика 4.2.4.3. Пример миграције*.* 19](#_Toc83225878)

[Слика 4.2.5.1. Пример комуникације преко *SignalR* алата*.* 19](#_Toc83225879)

[Слика 4.3.1.1. Формирање низа бајтова за *PasswordHash* и *PasswordSalt* од корисничке шифре*.* 21](#_Toc83225880)

[Слика 4.3.1.2. Провера исправности низа бајтова из корисничке шифре*.* 22](#_Toc83225881)

[Слика 4.3.1.3. Конфигурација аутентикације на серверској апликацији*.* 22](#_Toc83225882)

[Слика 4.3.1.4. Део програмског кода који приказује имплементацију генерисања токена*.* 23](#_Toc83225883)

[Слика 4.3.1.5. Део програмског кода у клијентској апликацији који приказује имплементацију обраде токена након захтева пријаве на систем*.* 23](#_Toc83225884)

[Слика 4.3.1.6. Део програмског кода у клијентској апликацији који приказује имплементацију пресретача захтева ка серверу*.* 23](#_Toc83225885)

[Слика 4.3.2.1. Део програмског кода који приказује чување слике*.* 24](#_Toc83225886)

[Слика 4.3.2.2. Део програмског кода који приказује референцирање слике*.* 24](#_Toc83225887)

[Слика 4.3.3.1. Део програмског кода који приказује чување датума краја игре као и обавештење свим клијентима у групи да време треба да почне да откуцава*.* 25](#_Toc83225888)

[Слика 4.3.3.2. Део програмског кода који приказује позив за почетак откуцавања времена*.* 25](#_Toc83225889)

[Слика 4.3.3.3. Део програмског кода који приказује клијентску имплементацију откуцавања времена*.* 25](#_Toc83225890)

[Слика 4.3.3.4. Део програмског кода који приказује серверску имплементацију рачунања преосталог времена у игри*.* 26](#_Toc83225891)

[Слика 4.3.4.1. Део програмског кода који приказује омотачку компоненту*.* 26](#_Toc83225892)

[Слика 4.3.4.2. Део програмског кода који приказује доделу вредности за *loader* стање*.* 26](#_Toc83225893)

[Слика 4.3.4.3. Део програмског кода из сервиса *FrameService* који приказује контролисање стања за *loader.* 27](#_Toc83225894)

[Слика 4.3.4.4. Део програмског кода који приказује промене за потезно поље*.* 27](#_Toc83225895)

[Слика 4.3.4.5. Део програмског кода који приказује позив методе за постављање информације о томе који такмичар је на потезу*.* 27](#_Toc83225896)

[Слика 4.3.4.6. Део програмског кода који приказује имплементацију методе за постављање информације о томе који такмичар је на потезу*.* 28](#_Toc83225897)

[Слика 4.3.5.1. Део програмског кода који приказује извршавање промена над базом користећи трансакцију*.* 29](#_Toc83225898)

[Слика 5.1.1. Страница за пријаву на систем*.* 30](#_Toc83225899)

[Слика 5.1.2. Страница за пријаву на систем – Погрешна лозинка или адреса електронске поште*.* 30](#_Toc83225900)

[Слика 5.1.3. Страница за пријаву на систем – Валидација поља на форми*.* 31](#_Toc83225901)

[Слика 5.1.4. Страница за регистрацију*.* 31](#_Toc83225902)

[Слика 5.1.5. Страница за регистрацију – Валидација шифре*.* 32](#_Toc83225903)

[Слика 5.1.6. Страница за промену шифре*.* 32](#_Toc83225904)

[Слика 5.1.7. Страница за промену шифре – Погрешна лозинка*.* 33](#_Toc83225905)

[Слика 5.1.8. Ранг листе*.* 33](#_Toc83225906)

[Слика 5.2.1. Администратор – Управљање корисничким налозима*.* 34](#_Toc83225907)

[Слика 5.2.2. Администратор – Бирање игара дана*.* 35](#_Toc83225908)

[Слика 5.3.1. Супервизор приказ*.* 36](#_Toc83225909)

[Слика 5.3.2. Супервизор – Додавање речи*.* 36](#_Toc83225910)

[Слика 5.3.3. Супервизор – Додавање спојница*.* 37](#_Toc83225911)

[Слика 5.3.4. Супервизор - Додавање асоцијација*.* 37](#_Toc83225912)

[Слика 5.4.1. Учесник квиза - Креирање партије*.* 38](#_Toc83225913)

[Слика 5.4.2. Учесник квиза - Бирање партије*.* 39](#_Toc83225914)

[Слика 5.4.3. Учесник квиза - Статистика*.* 39](#_Toc83225915)

[Слика 5.4.4. Учесник квиза – Статистика (ранг ван топ 10)*.* 40](#_Toc83225916)

[Слика 5.5.1. Дневна игра – Слагалица – Неисправна реч*.* 41](#_Toc83225917)

[Слика 5.5.2. Дневна игра – Слагалица – Исправна реч*.* 41](#_Toc83225918)

[Слика 5.5.3. Дневна игра – Мој Број*.* 42](#_Toc83225919)

[Слика 5.5.4. Дневна игра – Скочко*.* 43](#_Toc83225920)

[Слика 5.5.5. Дневна игра – Спојнице*.* 44](#_Toc83225921)

[Слика 5.5.6. Дневна игра – Асоцијације*.* 44](#_Toc83225922)

[Слика 5.5.7. Дневна игра – Крај игре*.* 44](#_Toc83225923)

[Слика 5.6.1. Учесник против учесника – Слагалица почетак*.* 45](#_Toc83225924)

[Слика 5.6.2. Учесник против учесника – Слагалица крај*.* 46](#_Toc83225925)

[Слика 5.6.3. Учесник против учесника – Мој Број*.* 47](#_Toc83225926)

[Слика 5.6.4. Учесник против учесника – Скочко*.* 48](#_Toc83225927)

[Слика 5.6.5. Учесник против учесника – Спојнице*.* 49](#_Toc83225928)

[Слика 5.6.6. Учесник против учесника – Асоцијације – Игра у току*.* 50](#_Toc83225929)

[Слика 5.6.7. Учесник против учесника – Асоцијације – Завршена рунда*.* 51](#_Toc83225930)

[Слика 5.6.8. Учесник против учесника – Завршена игра*.* 52](#_Toc83225931)