**Веб апликација за преглед и управљање подацима о нотара**

Јањић Дориан

Факултет техничких наука

Универзитет у Новом Саду

Трг Доситеја Обрадовића 6, 21000 Нови Сад

janjic.sr31.2020@uns.ac.rs

**Сажетак – У овом раду описана је веб апликација**[1] **за нотара која омогућава ефикасно управљње подацима нотара, резервације термина и овере уговора. За имплементацију су коришћене технологије као што су Spring Boot** [2]**, Angular**[3] **и MySQL**[4]**. Предности примене овог решења су олакшано заказивање и управљање терминима, аутоматско ажурирање података, могућност оверавања уговора и интероперабилност са другим информационим системима. Кроз процес развијања прототипа апликације, дубље су описане њене функционалности и практична примена.**

**Кључне речи - нотар; термини; резервације; уговори;**

* **УВОД**

Нотар или јавни бележник има важну улогу у правној сфери, обично се ангажују да врше верификацију и потврду разних правних докумената и трансакција. Ово укључује потврду потписа, изјава и других правних актова како би се обезбедила њихова валидност.

Традиционални систем нотара, иако има своје предности, такође може имати и неке проблеме и изазове. Неки од њих укључују:

1. **Време и Трошкови:** Процес ангажовања нотара може бити спор и скуп. Некада може бити потребно чекати дуже време да бисте добили термин код нотара. Такође, нотари обично захтевају плаћање за своје услуге, што може довести до додатних трошкова за кориснике.
2. **Локација и Доступност:** Нотари су обично локализовани на одређеним местима, па то може представљати проблем ако нисте близу њихове канцеларије. Ово ограничава доступност особама које живе на већим удљеним подручјима или онима који нису у могућности да се лично појаве на месту нотара.
3. **Конкуренција:** Због тога што нотари обично налазе на ограниченим подручјима и њихов број може бити ограничен, то може довести до недостатка у избору. Ово ограничава конкуренцију и може утицати на квалитет и цене њихових услуга.
4. **Папирологија и Бирократија:** Традиционални систем нотара обично захтева обимну папирну документацију. Ово може успорити поступак верификације и потврде документа.
5. **Технолошка Ограничења:** Традиционални систем нотара може бити мање прилагођен новим технологијама и модерним начинима рада. Ово може одразити недостатак флексибилности у понуди услуга.

Иако постоје ови проблеми, важно је напоменути да се многи сектори у свету труде да се савременизују и унапреде систем нотарације како би се ови проблеми решавали. Нове технологије и дигитални начини комуникације могу помоћи у олакшавању многих ових изазова.

Циљ овог семинарског рада је развој информационог система нотара који ће решити ове проблеме. Нови систем има за циљ побољшање ефикасности у нотарским услугама, да олакша процес заказивања термина. Као и да се повећа ниво аутоматизација процеса тако што нотар има комуникацију са информационим системима матичара и катастра приликом преноса власништва над непокретношћу и при другим услугама које пружа.

Нови систем нуди низ предности, укључујући:

1. **Потврда Идентитета:** Нотар је овлашћен да верификује и потврди идентитет особа које потписују документе, осигуравајући да су потписи даваоаца важећи.
2. **Обезбеђивање Аутентичности:** Нотар потврђује аутентичност документа и акатова које потписују странке, што помаже спречавању фалсификата и превара.
3. **Сведочење О изјавама:** Нотар може бити присутан када особа даје изјаву или потписује важан документ, потврђујући да је оно што је наведено тачно.
4. **Олакшавање Правних Трансакција:** Нотари олакшавају различите правне трансакције као што су купопродаја имовине, пренос права итд.
5. **Формирање и оверавање Уговора:** Нотари помажу при сачињавању правних докумената као што су наслеђа, уговори и изјаве.

* **Сродна решења и коришћење технологије**

У овом поглављу дат је преглед постојећих решења за развој информационог система нотара и наведене су технологије које омогућавају решавање проблема које су наведене у поглављу Увод овог семинарског рада.

* + **Сродна решења**

На основу мојих истраживања и сазнања, нисам успео да пронађем слична решења која се односе на ову специфичну тему. Међутим, постоји могућност да таква решења већ постоје, али можда нису доступна на интернету, или нису доступна јавности. Занимљиво је приметити да су нотари у Немачкој већ кренули корак даље, те ће ускоро бити уведена апликација “E-notar” [5] у неколико јавнобележничких канцеларија. Ова апликација се тренутно налази у тестној фази и требала би да функционише до усвајања закона који ће регулисати област архивирања у дигиталном формату. Овакав корак напред указује на потребу за модернизацијом и прилагођавањем дигиталним технологијама у домену нотарских услуга.

* + **Коришћење технологије**

За развој серверског дела апликације определили смо се за Spring Boot, моћну и употребљиву платформу за израду Java-базираних веб апликација. Ово нам је омогућило брзо креирање и успешно покретање стабилних серверских апликација. Конкретно, Spring Boot је омогућио једноставну конфигурацију, интеграцију са базом података и другим спољним библиотекама, као и развој безбедних и ефикасних **API**-ја[6].

За израду клијентског дела наше апликације, одлучили смо се за употребу Angular-a**, TypeScript**-а[7], **HTML**-а[8] и **Bootstrap**-a[9]. Angular, осим што представља модерну развојну платформу која се заснива на TypeScript-у, истиче се изобиљем компоненти које су нам омогућиле да креирамо скалабилне веб апликације с интерактивним корисничким интерфејсима. Додатно, помогао нам је ефикасно управљати формама, лако обављати рутинске задатке као што су усмеравање страна, а такође је послужио у процесу клијент-сервер комуникације.

За аспект дизајна и стила, определили смо се за коришћење HTML и Bootstrap-а. HTML, као основни језик за обележавање веб страница, послужио нам је за структурирање садржаја апликације. Уз помоћ Bootstrap-а, широко примењивог CSS[10] фрејмворка, брзо смо додали стилове, компоненте и респонзивни дизајн, знатно засиливши визуелни аспект нашег решења.

За потребе чувања података, одлучили смо се за MySQL базу података. Као популарна релациона база отвореног кода, MySQL нам је обезбедила скалабилност, поузданост и лак доступ до података. Његова једноставност у коришћењу и широк опсег функционалности га чине идеалним избором за нашу апликацију.

Са коришћењем ових технологија, успешно смо изградили и клијентски и серверски дел нашег решења. Као резултат, створили смо синхронизовану инфраструктуру која омогућава креирање прилагођених корисничких интерфејса, сачување и управљање подацима, као и брзу и безбедну комуникацију између клијента и сервера.

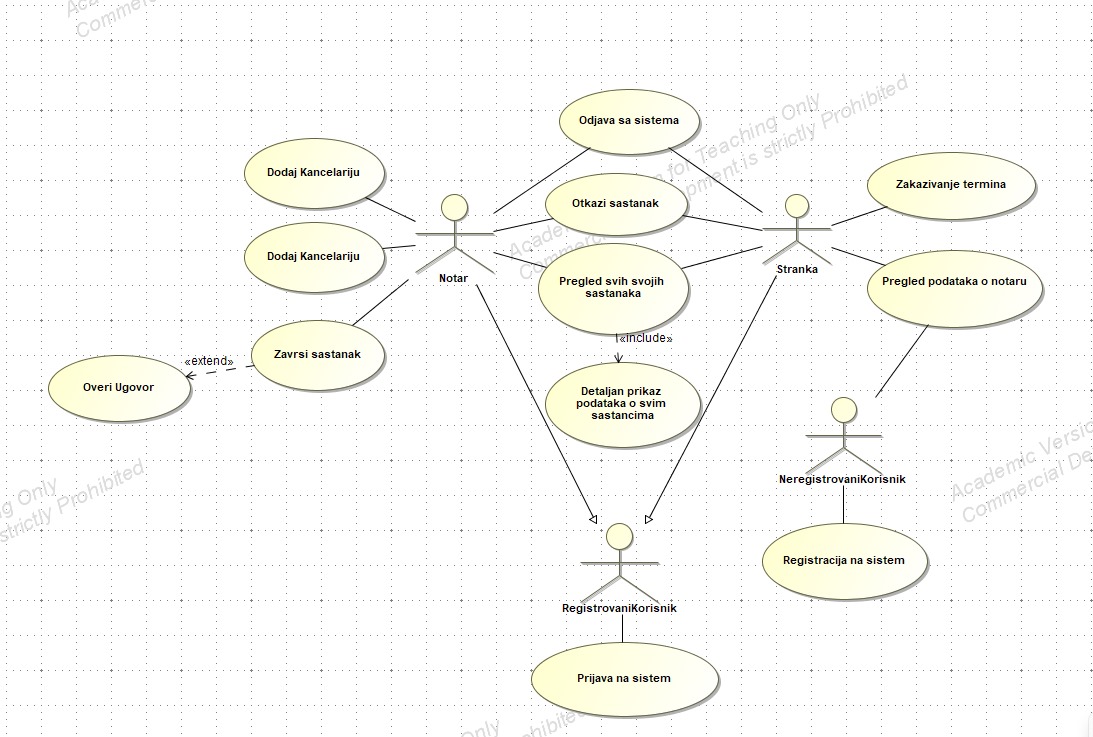
* **Спецификација захтева**

У овом поглављу су објашњени функционални и нефункционални захтеви софтверског решења представљеног у овом раду.

* + ***Спецификација функционалних захтева***

У овом одељку су описани функционални захтеви које је потребно да испуњава софтверско решење информационог система нотара.

У дијаграму случаја коришћења, имамо следеће актере: ***регистровани корисници који су нотар и странка*** *и* ***нерегистровани корисник****.* Функционални захтеви овог софтверског решења су представљени *UML* [11] дијаграмом случајева коришћења, као што је приказано на слици 1.



Слика 1 – Приказ *UML* дијаграма случајева коришћења

**Нерегистровани корисник** има следећи случај коришћења:

* Регистрација на систем: Омогућава нерегистрованм кориснику да се региструје на систем
* Преглед података о нотару: Могућност да има увид у основне податке о нотару

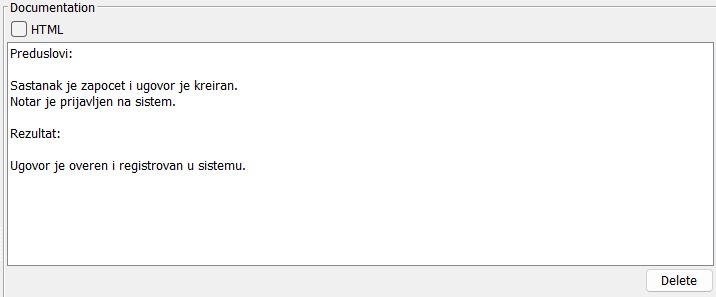
**Странка** има следећи случај коришћења:

* Преглед свог профила: Регистровани корисник може да прегледа информације о свом профилу, укључујући контактне податке и личне информације.
* Преглед података о нотару: Има увид у све доступне податке о нотару.
* Преглед свих података о слободним и заказаним терминима.
* Заказивање и oтказивање заказаног термина.
* Пријава и одјава са система

**Нотар** има следећи случај коришћења:

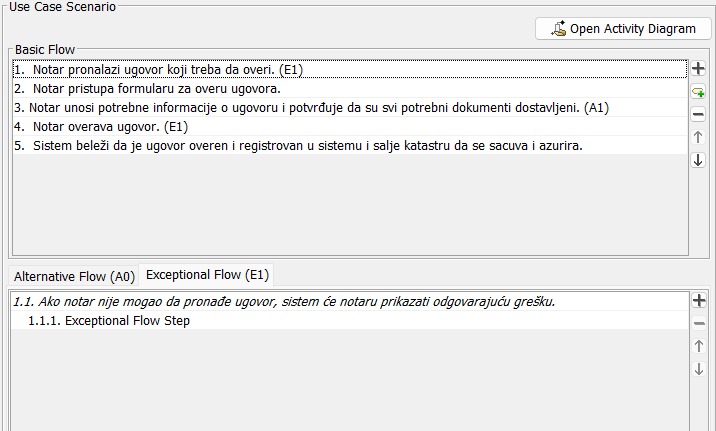
* Преглед свог профила: Нотар може да прегледа информације о свом профилу, укључујући контактне податке и личне информације.
* Преглед свих термина(састанака): Нотар има увид у све слободне, заказане и отказане термине.
* Преглед свих уговора: Нотар има увид у све уговоре.
* Додавање и отказивање састанка: Нотар има опцију да дода нови термин или да оброше већ заказан термин.
* Додавање и оверавање уговора: Нотар може да направи нови уговор и да га овери.
* Додавање канцеларије: Нотар има могућност да заведе нову канцеларију у систем.

Слика 2 представља предуслове случајева коришћења “Овера уговора“.



Слика 2 – Приказ предуслова дијаграма случаја коришћења “Овера уговора“.

Слика 3 представља опис случајева коришћења “Овера уговора“.



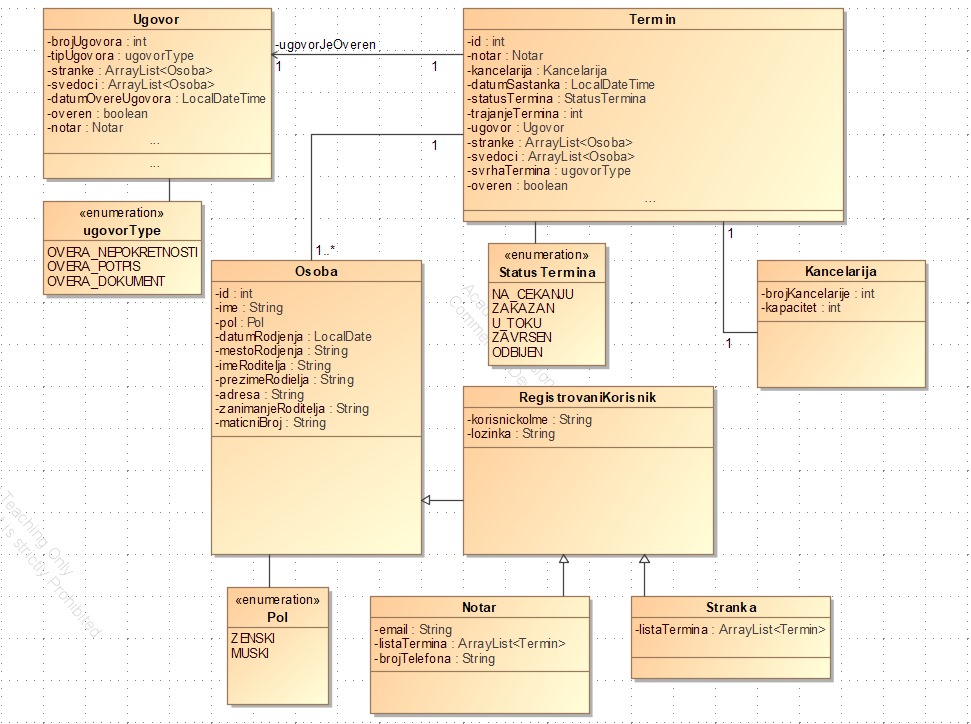
Слика 3 – Приказ описа случаја коришћења “Овера уговора“.

* + ***Спецификација нефункционалних захтева***

У следећем делу су наведени нефункционални захтеви које треба да задовољи софтверско решење информационог система нотара. Апликација укључује следеће нефункционалне захтеве:

* **Доступност и Корисничко Искуство:** Апликација треба да буде прилагођена корисницима свих нивоа знања, омогућавајући им лак и разумљив приступ свим функционалностима. Такође, корисничко искуство треба да буде интуитивно и прилагођено.
* **Безбедност и Заштита Података:** Апликација мора да има ниво безбедности који ће сачувати податке од неовлашћеног приступа и злонамерних активности.
* **Сигурност и Приватност:** Апликација треба да осигура висок ниво сигурности за заштиту осетљивих података и приватности корисника. То укључује аутентификацију, ауторизацију, шифровање података и заштиту од напада.
* **Интеграција:** Апликација треба да буде способна за интеграцију са другим системима које користе нотари, као и са регулаторним или другим организацијама.
* **Одржаваност и Скалабилност:** Код апликације треба да буде лак за одржавање и проширивост, омогућавајући да се нове функционалности додавају и постојеће подешавају без великих компликација.

На слици 4 је помоћу *UML* дијаграма класа представљен објектни модел система.



Слика 4 – Дијаграм класа

Класни дијаграм који смо израдили у UML-у садржи следеће класе и везе:

* **Термин:**
  + Класа која представља термин када је заказан састанак.
* **Статус термина (енумерациона класа):**
  + Представља у каквом стању се налази термин.
  + Користи се у класи термин.
* **Уговор:**
  + Класа која представлја уговор који је састављен на састанку.
* **Тип уговора (енумерациона класа):**
  + Класа која представлја ког типа је уговор.
  + Користи се у класи уговор.
* **Особа (апстрактна класа):** 
  + Класа која представља сваког корисника система.
  + Служи као прва основна класа
* **Регистрован корисник (апстрактна класа):**
  + Везана је за класу особа преко асоцијације
  + Служи као друга основна класа за конкретне класе: Нотар и Странка
* **Нотар:**
  + Представаља конкретну класу која описује нотара.
  + Везана је за класу регистрованог корисника преко асоцијације.
  + Нотар је корисник система који управља терминима и уговорима.
* **Странка:**
  + Представаља конкретну класу која описује странку.
  + Везана је за класу регистрованог корисника преко асоцијације.
* **Пол (енумерациона класа):**
  + Класа која представља пол особе.
  + Користи се у класи особа.
* **Канцеларија:**
  + Класа која предтавља у којој канцеларији се одвија састанак

Дати класни дијаграм у UML-у приказује структуру и везе различитих класа у оквиру апликације. Укључене су класе које описују различите аспекте и улоге у систему. Присутни су апстрактни и конкретни корисници, као што су Нотар и Странка, сваки са јасно дефинисаним карактеристикама. Такође, дијаграм садржи енумерационе класе које омогућавају детаљно одређивање стања, као што су Пол и Тип уговора. Кроз асоцијације и наслеђивање, приказане су везе између класа, као и њихова хијерархијска организација. Ова структура доприноси бољем разумевању организације и односа између различитих ентитета у апликацији која се односи на рад нотара.

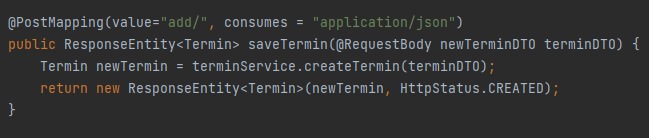
* ***Имплементација***

У овом делу је разматран начин на који су реализоване одређене функционалности у информационом систему нотара. За пример ћемо узети појединачну имплементацију додавања новог слободног термина у систем. Након тога, биће приказано ажурирање већ постојећих података термина. Додатно, биће испитана и методологија за преузимање података из базе података уговора. Целокупна анализа подразумева истраживање обе стране апликације, и то и клијентског и серверског дела, што ће допринети дубоком разумевању процеса имплементације у сложеном окружењу информационог система нотара.

* + ***Имплементација серверског дела апликације***

Овај одељак садржи начин помоћу ког су имплементиране неке од функција информационог система нотара на серверском делу апликације.

За додавање података новог термина имплементирана је метода у једном контролеру приказана на слици 5.



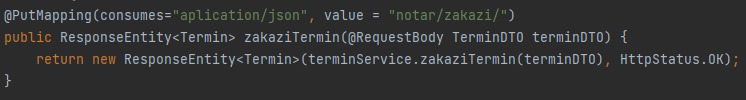
Слика 5 – Додавање података о новом термину (серверски код)

Ова функција, под именом saveTermin(), је обележена са **@PostMapping** анотацијом која указује на **HTTP POST** захтев. У ову функцију се прослеђује објекат који представља одређени термин, односно **DTO (Data Transfer Object)** за термин.

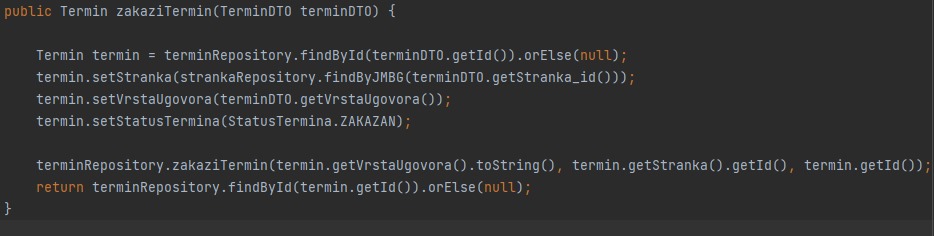
Ако термин са наведеним истим временом догађања и истом канцеларијом већ постоји, тада се генерише HTTP одговор са статусом **400 (Bad request)**, придружен поруком која детаљно описује грешку. У случају да термин са унесеним подацима не постоји, направи се нова инстанца класе Termin, и сви њени атрибути се попуњавају вредностима из DTO објекта на основу унесених података (у нашем случају, информације о датуму, времену и другим детаљима термина).

На крају се извршава createTermin() метода из TerminService која податке о новом термину уписује у базу података. Као завршни корак, формира се објекат класе **ResponseEntity<>** који обухвата одговор са статусом који указује на успешно сачуване информације о новом термину.

За ажурирање података о термина имплементирана је метода у контролеру и сервису приказана на сликама 6. и 7.



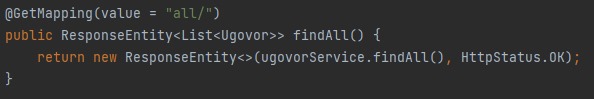
Слика 6 – Ажурирање података термина (серверски код у контролеру)



Слика 7 – Ажурирање података термина (серверски код у сервису)

Ова метода са називом *zakazi()* има *@PutMapping* анотацију која указује да је у питању ***HTTP PUT*** захтев. Овој методи се прослеђује објекат који репрезентује конкретан термин, то јест *DTO (Data Transfer Object)* за конретан термин. Затим се позива метода zakaziTermin() која се налази у TerminService-u. У телу методе се прво добавља постојећи термин по идентификационом броју позивом методе *findById()* чија имплементација се налази у класи *TerminServiceImplementation*. Ако се не пронађе ни један термин, функција ће вратити null, то значи да termin са унетим идентификационim бројem не постоји у бази података и враћа се објекат класе *ResponseEntity* са статусом 400 *(Bad Request)*. Ако добављени подаци о terminu постоје у бази података, ажурирају се атрибути Straka, VrstaUgovora и StatusTermina на основу вредности из објекта класе *terminDTO*. Затим се позива zakaziTermin*()* метода из класе *TerminRepository* да би се сачувале промене у бази података и на крају се објекат класе *ResponseEntity<>* враћа са одговором са статусом 200 о успешном ажурирању података о термину.

За преузимање података из базе података термина имплементирана је метода у једном контролеру приказана на слици 8.

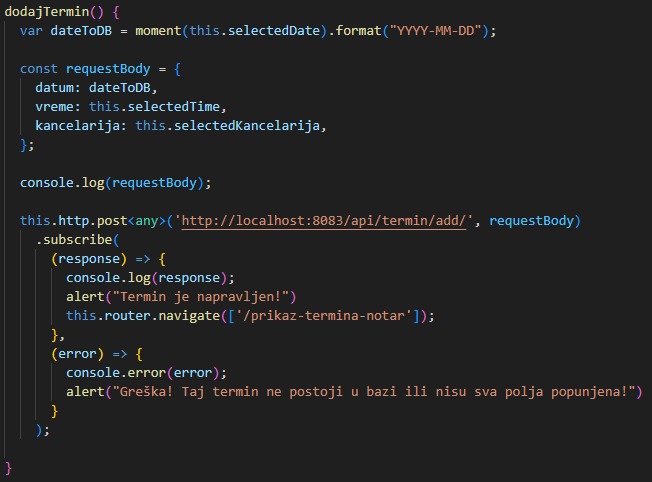


Слика 8 – Преузимање из базе података термина (серверски код у контролеру)

Ова метода са називом *findAll()* има *@GetMapping* анотацију која указује да је у питању ***HTTP GET*** захтев. Прво се позива метода *findAll()* чија се имплементација налази у класи *UgovorService* и која враћа листу свих уговора који постоје у бази података. Затим се креира празна листа која садржи објекте класе *UgovorDTO* и они репрезентују ***DTO (Data Transfer Object)***за конкретне уговоре. На крају се креира објекат класе *ResponseEntity<>* који враћа одговор са статусом 200 *(OK)*.

* ***Имплементација клијентског дела апликације***

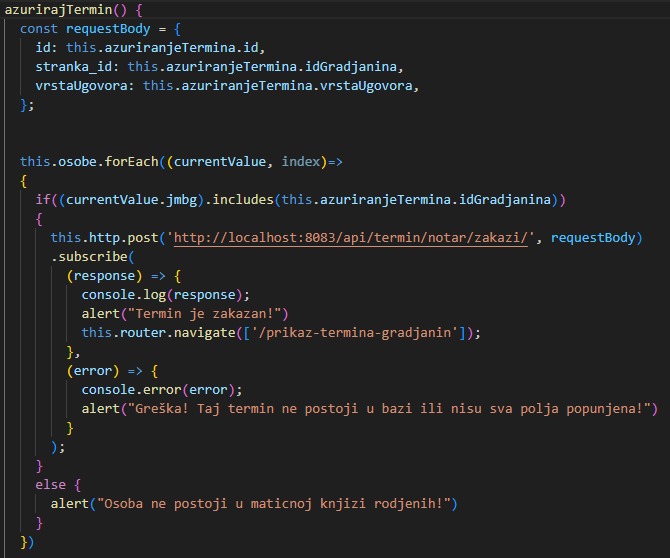
За додавање података о новом термину, на основу кода у серверском делу апликације који је приказан на слици 5, имплементирана је метода у једном *TypeScript* фајлу приказана на слици 9.



Слика 9 – Додавање података термина (клијентски код)

У овој функцији под називом *dodajTermin()* се прво одређује формат датума термина и декларише константа *requestBody* са неопходним подацима које треба додати у базу. Ти подаци се преузимају из глобалних променљивих. Након тога користи се метода за слање ***HTTP POST*** захтева. Путања захтева је *"*[*http://localhost:8083/api/notar/termin/add*](http://localhost:8083/api/notar/termin/add)*/"*, уз коју се шаље и *requestBody* као тело тог захтева серверу. Када сервер одговори на захтев онда се користи метода ***subscribe()*** помоћу које се региструје одговор сервера. Ако је одговор успешан, извршава се *callback* функција која је прослеђена као први аргумент методе *subscribe()*, што значи да ће се приказати порука *"Термин је направљен!"* као *alert* порука, а затим се преусмери страница на „/prikaz-termina-notar“помоћу *Angular* рутирања кроз компоненте. У супротном, ако се деси грешка приликом захтева онда се извршава ***callback*** функција која је прослеђена као други аргумент *subscribe()* методи, то јест приказује се порука о грешци и у конзоли и као ***alert*** порука.

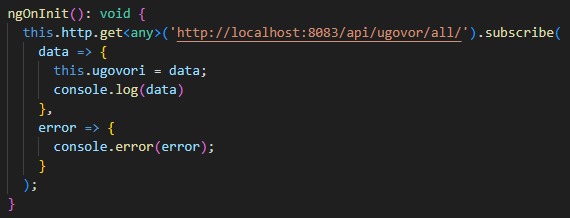
За ажурирање података термина, на основу кода у серверском делу апликације који је приказан на сликама 6. и 7, имплементирана је метода у једном *TypeScript* фајлу приказана на слици 10.



Слика 10 – Ажурирање података термина (клијентски код)

У овој функцији под називом *azurirajTermin()* се прво декларише константа *requestBody* која садржи податке о термину које треба ажурирати. Ти подаци се преузимају из глобалних променљивих. Затим се проверава особа, која је попунила податке, да ли постоји у систему матичара, тако што упоређује њен унети ЈМБГ са ЈМБГом из базе података матичне кљиге рођених. Ако је услов испунјен користи се метода за ажурирање *HTTP PUT* захтев. Путања захтева је *"http://localhost:8083/api/termin/notar/zakazi/"*, уз коју се шаље и *requestBody* као тело тог захтева серверу. Када сервер одговори на захтев онда се користи метода *subscribe()* помоћу које се региструје одговор сервера. У случају успешног одговора од сервера, тај добијени одговор се исписује у конзоли и приказује се порука да су подаци о термину успешно ажурирани, тј. да је термин заказан користећи *alert*, а онда се кориснику пребацује на страницу */prikaz-termina-gradjanin* помоћу *Angular* рутера. Ако дође до неке грешке она се исписује у конзоли претраживача и као alert порука.

За преузимање података из базе података термина, на основу кода у серверском делу апликације који је приказан на слици 8, имплементирана је метода у једном *TypeScript* фајлу приказана на слици 11.



Слика 10 – Добијање података свих термина (клијентски код)

Функција која се зове ngOnInit(), ona се извршава аутоматски приликом учитавања странице, имплементација је одрађена тако да чим корисник отвори прозор да има увид у жељене податке, што су у овом случају уговори. Затим позива захтев HTTP GET који служи за придобијање података са серверске стране. Ова метода враћа објекат класе ***Observable*** у ком се налази листа уговора. Путања захтева је *"http://localhost:8083/api/ugovor/all/"*. Када сервер одговори на захтев и пошаље податке, и захтев буде валидан онда се користи метода *subscribe()* помоћу које се региструје одговор сервера и чува у глобалну листу променљивих „ugovori“. А ако захтев није валидан, исписаће грешку у конзоли претраживача.

* ***Демонстрација***

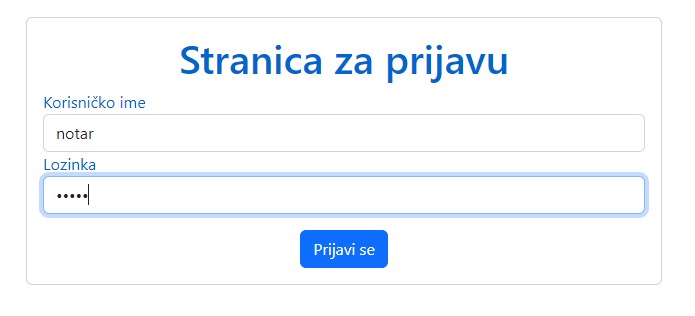
У овом поглављу се налази демонстрација софтверског решења за информациони систем нотара.

Први корак приликом покретања апликације кориснику се приказује прозор са слике 11.

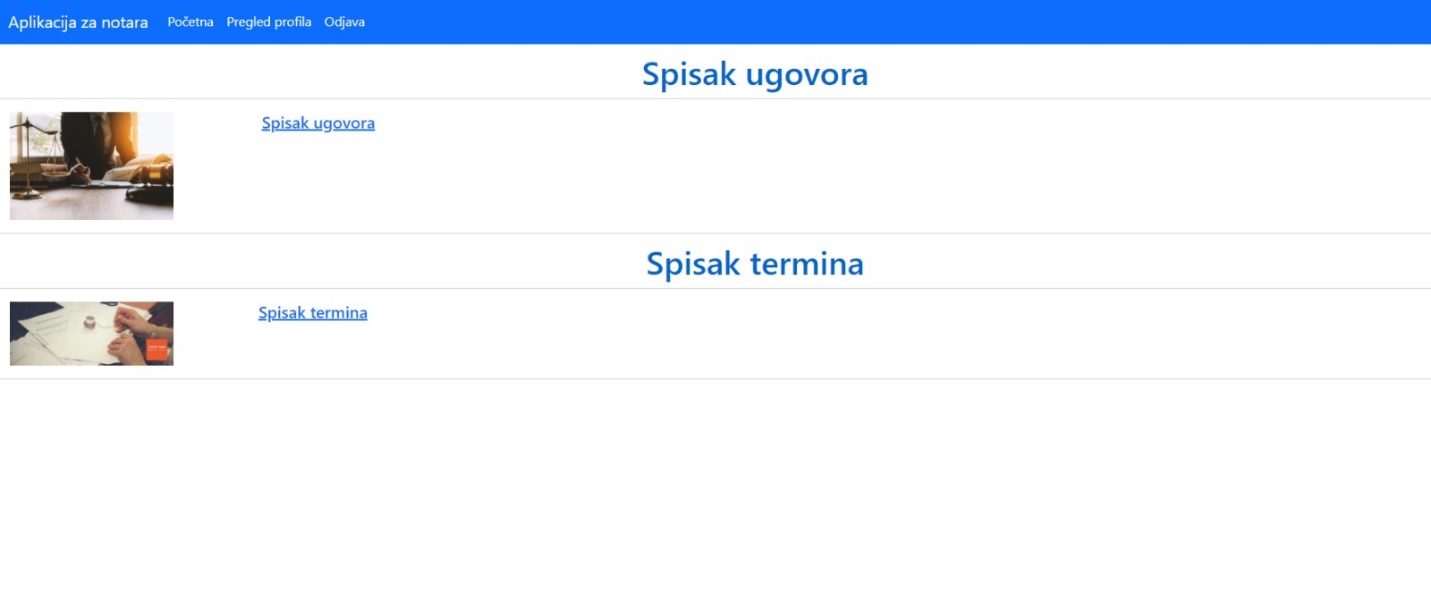


Слика 11 – Изглед прозора који се приказује приликом покретања апликације

На слици 12. је приказана страница са формом за пријаву на апликацију. Корисник има могућност да попуни поља креденцијала за аутентификацију и притисне на дугме „Prijavi se“, и ако су креденцијали валидни биће преусмерен на одговарајућу страницу спрам улоге у систему, која може бити Нотар или Грађанин.

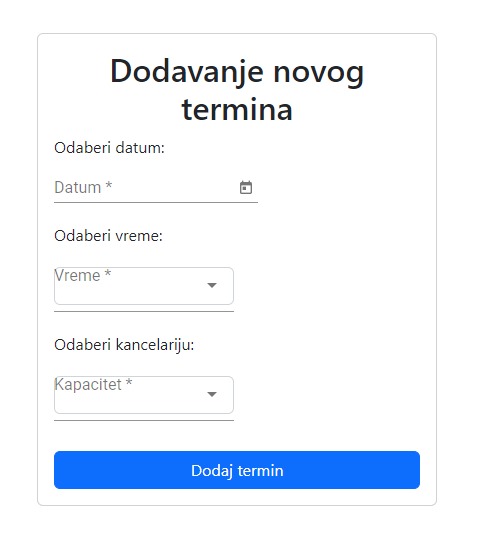


Слика 12 – Изглед прозора са формом са пријаву на систем

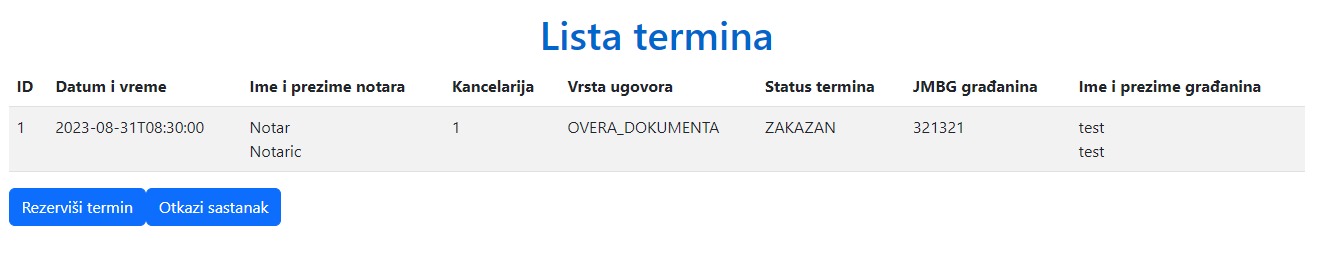
Уколико се корисник пријави као нотар биће приказана страница са слике 13. У горњем делу станице се налази трака за навигацију са опцијама за приказ почетне странице, преглед профиле и одјаву. Остатак странице садржи неке опције које нотар може да изврши у систему у складу са UML диаграмом коришћења са слике 1.

Слика 13 – Изглед прозора са неким функционалностима нотара и навигацијом

На слици 14. је приказана страница са формом за додавање новог термина. Одабирањем датума, времена и капацитета и притиском на дугме „Dodaj termin“. Нотар додаје нови термин у систем и биће преусмерен на страницу са листом термина која је приказана на слици 15.

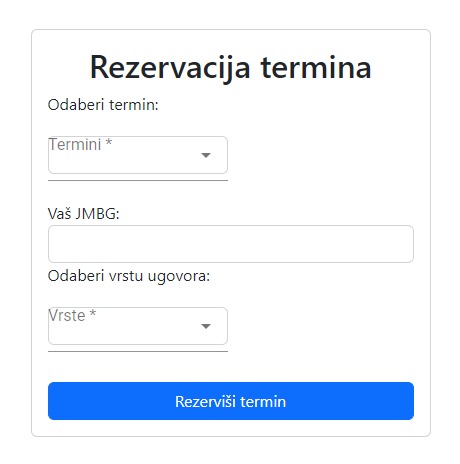


Слика 14 – Изглед прозора са формом за додаваље новог термина



Слика 15 – Изглед прозора са табелом термина

На слици 16. је приказана страница са формом за резервацију термина, која ажурира већ постојећи термин. Одабирањем термина, врсте уговоре и уносом ЈМБГ-а особе која резервише термин и притиском на дугме „Rezervisi termin“, уколико је ЈМБГ валидан, термин је резервисан и корисник је преусмерен на страницу са листом термина као сто је приказана на слици 15.



Слика 16 – Изглед прозора са формом за резервацију термина

На слици 17. је приказана страница са листом Уговора, који су резултат читања из базе података.



Слика 17 – Изглед прозора са табелом уговора, који су резултат читања из базе

* **ЗАКЛЉУЧАК**

У овом раду је представљена апликација за информациони систем нотара и демонстриран рад апликације. Развој модерне веб апликације за нотаре представља значајан корак ка унапређењу нотарског пословања. Ово решење се показало као ефикасан начин да се суоче изазови који се тичу времена, трошкова, доступности и технолошке ограничености у традиционалном нотарском систему.

Увођењем овакве апликације, нотари добијају већу оперативну флексибилност, а корисници имају бржи и лакши приступ нотарским услугама. Оптимизација процеса заказивања термина, аутоматско ажурирање података и могућност оверавања уговора, доприносе повећању ефикасности и смањењу бирократије у већини нотарских активности.

Интероперабилност новог система са другим информационим системима доприноси већем степену погодности и повезаности, допуштајући бољу комуникацију и обраду података између различитих служби и организација.

Укратко, ова веб апликација за нотаре не само што решава проблеме који се појављују у традиционалном нотарском пословању, већ и отвара нове могућности и путеве за подизање квалитета и брзине пружања правних услуга. Примена модерних технологија, као што су Spring Boot, Angular и MySQL, омогућава да се савремени изазови суоче на ефикасан начин и допринесе подизању стандарда у области нотарског пословања.

* **Литература**
* [1] "Увод у веб апликације: дефиниција и основни концепти." TechTarget.com

<https://www.techtarget.com/searchsoftwarequality/definition/Web-application-Web-app>

* [2] "Шта је Java Spring Boot?" Microsoft Azure. Приступљено са

https://azure.microsoft.com/en-ca/resources/cloud-computing-dictionary/what-is-java-spring-boot

* [3] "Angular архитектура и структура пројекта." Angular.io. Приступљено са

<https://angular.io/guide/architecture>

* [4] "Шта је MySQL?" Oracle.com. Приступљено са

<https://www.oracle.com/mysql/what-is-mysql/>

* [5] “E-Notar” Paragrafco.co.rs. Приступљено са

https://www.paragrafco.co.rs/dnevne-vesti/300323/300323-vest4.html

* [6] "Шта је API” Resilako.com. Приступљено са

https://resilako.com/racunari/sta-je-api-gde-se-koristi-i-kako-funkcionise/

* [7] "Шта је Type-Script" Startit.rs. Приступљено са

https://startit.rs/sve-popularniji-superset-javascript-a-dobio-novu-verziju-zasto-treba-da-naucite-typescript/

* [8] "Шта је HTML?" Skolasvilajnac.edu.rs. Приступљено са

http://www.skolasvilajnac.edu.rs/mreze/osnove\_html.htm

* [9] "Шта је Bootstrap?" TechTarget.com. Приступљено са

<https://www.techtarget.com/whatis/definition/bootstrap>

* [10] "Шта је HTML?” Oxfordwebstudio.com. Приступљено са

https://www.oxfordwebstudio.com/da-li-znate/sta-je-css.html

* [11] "UML Дијаграми." SmartDraw.com. Приступљено са [www.smartdraw.com/uml-diagram/](https://www.smartdraw.com/uml-diagram/)