Zbornik radova Fakulteta tehničkih nauka, Novi Sad

UDK: 004.4
DOI: https://doi.org/10.24867/29BE27Kajtez

РАЗВОЈ ВЕБ АПЛИКАЦИЈЕ ЗА ПРЕЗЕНТОВАЊЕ ПОДАТАКА О ФАКУЛТЕТИМА И ДАВАЊЕ ПРЕДЛОГА УПОТРЕБОМ "*CHAT GPT AI*" МОДЕЛА

DEVELOPMENT OF A WEB APPLICATION FOR PRESENTING DATA ABOUT FACULTIES AND GIVING SUGGESTIONS USING THE "*CHATGPT AI*" MODEL

Миле Кајтез, *Факултет техничких наука, Нови Сад*

Област – ЕЛЕКТРОТЕХНИЧКО И РАЧУНАРСКО ИНЖЕЊЕРСТВО

Кратак садржај – *Рад описује развој "Faculty Info" веб апликације која ће помоћи будућим студетима у процесу доношења одлуке који факултет уписати и којом професијом се бавити у будућности, употребом вештачке интелигенције (у овом случају коришћењем "Chat GPT AI" модела). Такође, корисник има могућност увида у информације о факултетима, департманима, курсевима и предметима, као и постављање питања о истим.*

Кључне речи: *систем за управљање знањем / ChatGPT / React / .NET CORE 6*

Abstract – *The work describes the development of an application that will help future students in the decision-making process of which faculty to enter and which profession to pursue in the future, using artificial intelligence (in this case using the "Chat GPT AI" model). Also, the user has the opportunity to view information about faculties, departments, courses and subjects, as well as ask questions about them.*

Keywords: *knowledge management system / ChatGPT / React / .NET CORE 6*

1. УВОД

У претходних десетак до двадесет година, све је већа употреба друштвених мрежа као главног извора информација у готово свим областима, почевши од дневних вести, временске прогнозе, најаве разних догађаја, до информација у областима као што су медицина, образовање и мноштво других друштвено корисних активности. Највећи број корисника друштвених мрежа су млади људи, пре свега они који су у средњим школама и на факултетима. Изложеност великој количини информација, често доводи до тога да млади не могу самостално и на прави начин извући правовремене и добре закључке. Иза тога стоје и многи други разлози, као што су презентовање информација на начин који није попуно разумљив, половичност, делимична тачност информација и слично.

НАПОМЕНА:

Овај рад проистекао је из мастер рада чији ментор је био др Александар Селаков, ванр. проф.

Једно од решења јесте коришћење вештачке интелигенције, која ће уместо човека прикупљати, селектирати, обрађивати и претварати информације у, за човека, разумљив и погодан облик. Конкретно "*Faculty Info*" апликација ће користити вештачку интелигенцију која ће будућим студентима, на основу њихових животних преференција и интересовања, помоћи у доношењу одлуке који факултет уписати и чиме се бавити у будућности.

2. КОРИШЋЕНЕ ТЕХНИКЕ И ТЕХНОЛОГИЈЕ

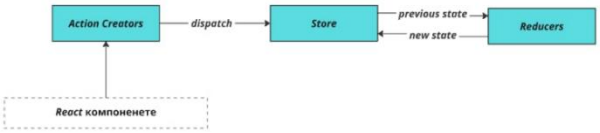
Анализом сличних решења чији је примарни задатак информасање студентата и оних који ће то тек постати, закључак је да свако решење има свој опсег употребе. Оно што је заједничко за све јесу њихова усмереност да корисници на ефикасан и тачан начин, у што краћем временском периоду, дођу до жељених информација. Поред самих перформанси, битно је нагласити да апликације могу бити коришћене од стране великог броја корисника. Некада, велики број функционалности може бити контрапродуктиван јер доводи до збуњености корисника по питању начина коришћења апликације. Као у свему, и у овом аспекту развоја је потребно пронаћи баланс између пружања услуга, једноставности и перформанси коришћења. Уколико се пронађе тај баланс, често те апликације доживљавају велику експанзију и употребу, па самим тим и успех. Као предуслов за испуњење претходно наведених услова, у самом процесу развоја се морају донети одлуке по питању које технологије и технике користити. Пошто је "*Faculty Info*" систем, по свом типу веб апликација, технике и технологије су подељене на два дела, тј. оне које се користе за израду клијентске апликације, као и технике и технологије за израду серверског дела и комуникацију и складиштење података у оквиру базе података.

2.1. Технологије клијентске апликације

Могућности за креирање клијентске апликације су данас велике због постојања великог броја развојних окружења који омогућавају флексибилност у развоју. Конкретно, у случају "*Faculty Info*" апликације, ради се о "*SPA (Single Page Application)*" [1], креираном употребом "*React JS*" библиотеке [2] , "*Redux*" механизма [3] и "*CSS (Cascading Style Sheets)*" [4] језика, који се користи за дефинисање самог изгледа апликације. За разлику од ранијих, данашње веб апликације су окренуте ка клијентској страни, тј.

тежи се ка томе да клијентска апликација буде што „тежа“ тј. да се део операција извршава одмах на клијентској страни, без беспотребног слања захтева ка серверу. Овај приступ омогућава лакше конструисање апликација које ће бити намењене за велики број корисника, као и њихов бржи и поузданији рад. У сврху тога, клијентска апликација, поред компоненти, садржи и “*Redux*” библиотеку, која омогућава чување дела података на клијентској страни, што скраћује време добављања података и самим тим резултује бржим радом и одзивом у реалном времену, док сам корисник стиче осећај правовременог и брзог одговора. “*Redux*” је state контејнер за “*JavaScript*” [5] апликације и састоји се из три дела (Слика 2.1.1), и то:

1. “*Store*” – објекат у оквиру којег се налазе сви подаци и уз помоћ којег се могу пратити све промене података у току рада апликације. Једини начин за промену *state* објекта, јесте емитовање акције. Такође, променом *state* објекта извршава се поновно учитавање свих компоненти.
2. “*Action*” – акција која позива промену *state* објекта (описује промену стања)
3. “*Reducer*” – врши промену *state* објекта (спроводи акцију)

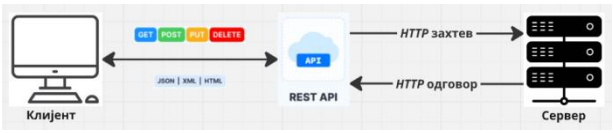


Слика 2.1.1. - “Redux” начин рада

Што се тиче имплементације самог дизајна клијентске апликације, *CSS* стилови су писани коришћењем стилизованих компоненти (“*Styled Components*”) [6], што се показало као једно од бољих решења за велике веб апликације. Употребом ове врсте компоненти омогућава се креирање многих дизајнерских функционалности попут промене теме, лакшег коришћења стилова на више места, што резултира лаку прилагодивост и проширивост већ постојећих стилова, итд.

2.2. Технологије серверске апликације

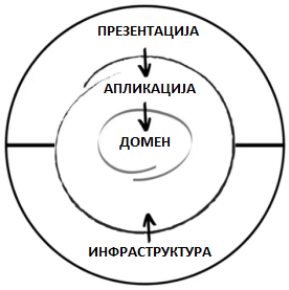
Да би систем, као целина могао да корисницима пружи тачне и правовремене информације, потребно је да исти има серверску апликацију која може да задовољи претходно наведене услове, да има стабилан и поуздан рад, могућност лаког опоравка од отказа, гаранцију да подаци који се обрађују буду валидни и заштићени од спољних утицаја, итд. Да би се дошло до одговора на питање који архитектурални стил користити у изради овог дела апликације, прво се морају сагледати идеје коришћења апликације, скуп функционалности које систем треба да задовољи, број корисника који ће потенцијално да користе систем, који тип корисника ће користити систем, итд. Што се саме архитектуре тиче, узимајући у обзир да је сам развој апликације у почетној фази, донета је одлука да се користи монолитна архитектура, која у почетним фазама производа нуди много већу брзину развоја и флексибилност измена. Што се тиче саме комуникације са клијентском страном, она је имплементирана употребом “*HTTP*” протокола [7], док сама серверска апликација ради по “*REST*” принципу [8] (Слика 2.2.1.).



Слика 2.2.1. - “REST” принцип комуникације клијентске и серверске апликације

Инфраструктура и организација логичких целина серверске апликације, дефинисана је коришћењем “*Onion*” архитектуре (Слика 2.2.2.) [9], која омогућава многе бенефите као што су:

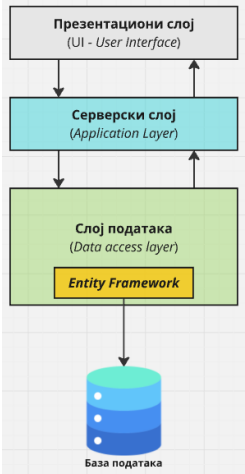
1. Лакши потенцијални прелазак са монолитне на микросервисну архитектуру
2. Лакше одржавање кода
3. Флексибилније креирање, модификација и тестирање компоненти и функционалности
4. Лакша и независнија комуникација са екстерним сервисима



Слика 2.2.2. - Илустрација “*Onion*” архитектуре

У склопу “*Onion*” архитектуре користи се и “*Command and Query Responsibility Segregation (CQRS)*” образац [10], који раздваја операције читања и ажурирања базе података. Пружа додатну скалабилност и сигурност због омогућавања лакшег развијања система током времена и спречавања да наредбе ажурирања узрокују конфликте на нивоу домена. Команде (“*Commands*”) се користе за ажурирање базе података, док се упити (“*Queries*”) користе за добављање података. Што се тиче технологије која се користи у имплементацији претходно наведене архитектуре, користи се “*C#*” програмски језик [11] и “*ASP .NET Core 6*” [12] који представља радно окружење за развијање серверских апликација. Употребом “*ASP .NET Core 6*” окружења пружа се могућност лаког креирања серверске апликације, логике за комуникацију са базом података, употребом објектно-релационог мапера, као што је "*EF (Entity Framework)*" [13], као и другим екстерним сервисима, али и употребом “*CRUD (Create, Read, Update, Delete)*” operacija [14]. "*Entity Framework*" је објектно-релациони мапер који пружа опцију лаког и

брзог начина дефинисања и коришћења базе података (Слика 2.2.3.).



Слика 2.2.3. Приказ позиције "*Entity Framework*" мапера

Сама база података је дефинисана употребом "*PostgreSQL*" [15], објектно-релационог система за управљање базама података који садржи моћан објектно-релациони модел података, богат избор врста података, лаку надоградивост, као и надограђени скуп наредби "*Structured Query Language (SQL)*" језика [16].

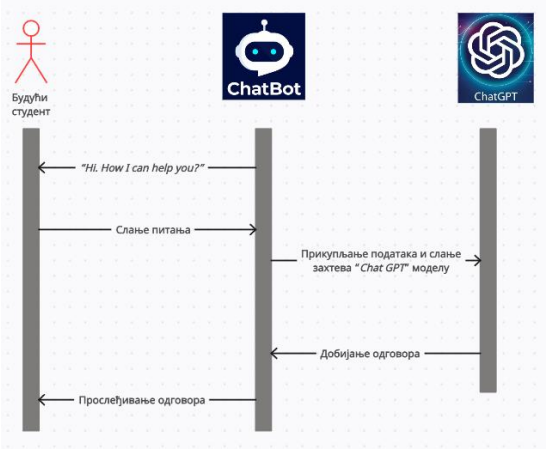
Поред наведених делова система, систем користи и екстерне сервисе, као што су "*Sendgrid*" [17] за слање електронске поште, "*Amazon Simple Storage Service (AWS S3)*" [18] за посебно складиштење ресурса попут слика и докумената, "*Geoapify*" [19] за одређивање географских локација, као и "*ChatGPT OpenAI API*" [20] који омогућава коришћење моћних модела језика као што су "*GPT-3.5*" и "*GPT-4*" путем једноставног "*API (Application Programming Interface)*" [21] позива.

3. ФУНКЦИОНАЛНОСТИ И ДЕМОНСТРАЦИЈА ИМПЛЕМЕНТАЦИЈЕ

Апликација је конципирана као дигитална платформа која омогућава приступање подацима о факултетима, курсевима и предметима, као и манипулисање подацима у циљу лакшег доношења одлука у вези будућег студирања и професионалне каријере. Систем подржава три врсте корисника:

1. *Главни администратор* који је одговоран за администрацију целокупног система и подешавања (манипулација факултетима и администраторима факултета).
2. *Администратор факултета* одговоран за администрацију информација везаних за свој факултет, одговарање на питања, дефинисање курсева, предмета и анкета, као и увид у изглед страница факултета ком је придодат.
3. *Будући студент* који има могућност прегледа факултета, курсева и предмета, манипулација профилних података, постављање питања, одговарање на питања из анкете, као и комуникација са *chatbot* механизмом.

Сваки од корисника, при отварању апликације, има приступ почетној страници, где се може извршити логовање или регистровање, у зависности од тога да ли је корисник већ регистрован или је пак будући студент. У наставку ће бити описан начин рада апликације у процесу комуникације будућег студента и *chatbot* механизма, у интеграцији са "*ChatGPT*" моделом (Слика 3.1.), што представља главну функционалност апликације.



Слика 3.1. Кораци комуникације будућег студента и *chatbot*-а

После успешног одговарања на сва или пак на већину питања из анкете, корисник може отићи на страницу (Слика 3.2.) на којој има могућност комуникације путем *chatbot* механизма, који се ослања на "*ChatGPT*" модел. Комуникацију започиње *chatbot*, док је задатак корисника да постави питања која жели. У року од пар секунди добиће одговор и на тај начин се формира комуникација. Како се комуникација развија, тако систем све више препознаје корисникова интересовања и чиме би потенцијално могао да се бави у будућности и на основу тих информација шаље будуће одговоре. Корисник у просеку на одговор чека до 3 секунде, што представља сасвим солидну брзину комуникације и добијања нових информација.

Слика 3.2. Приказ комуникације корисника са *chatbot*-ом

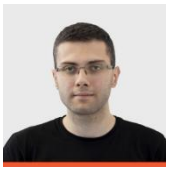
4. ЗАКЉУЧАК

Проласком кроз рад, описан је систем који се састоји из три целине, које функционишу и међусобно комуницирају и на тај начин чине један, јединствен систем. Показало се да сама веб апликација поседује особине које дају предуслов за брзо и ефикасно коришћење, као и одзив у реалном времену. Ова тврдња може се потврдити постојањем "*Redux*" логике на клијентској апликацији, која чува податке и омогућава смањење броја захтева ка серверској страни. Коришћењем "*Onion*" архитектуре на серверској страни, постоји могућност лаког проширења и измене делова логике. Што се тиче базе података, користи се релациона база података која омогућава лакше манипулисање односима међу ентитетима система. Сама апликација обезбеђује интуитивно и поуздано искуство, омогућавајући ефикасно и јасно управљање информацијама, као и интеракцију са "*ChatGPT*" моделом ради добијања персонализованих препорука. Очекује се да ће апликација пружити корисницима свеобухватне ресурсе и подршку у процесу доношења одлука о образовању. Иако апликација има и друге добре особине, као што су једноставан и прегледан дизајн, тачно и правовремено одговарање сервисне стране, висок ниво апстракције кода, свакако да постоји још простора за побољшања и увођење нових решења, као што су креирање мобилне верзије апликације која ће проширити број корисника који користи систем, претварање апликације у друштвену мрежу на којој ће корисници моћи међусобно делити информације и материјале који ће им бити од помоћи у даљем образовању, као и увођење нових функционалности попут увида у резултате испита, унос оцена, приказ статистике, постојање нотификација, могућности креирања анкета и других садржаја, итд.

5. ЛИТЕРАТУРА

[1] https://developer.mozilla.org/en-US/docs/Glossary/SPA, преузето Јуна 2024.
[2] https://reactjs.org/, преузето Јуна 2024.
[3] https://redux.js.org/, преузето Јуна 2024.
[4] https://www.tutorialrepublic.com/css-tutorial/, преузето Јуна 2024.
[5] https://www.javascript.com/, преузето Јуна 2024.
[6] https://styled-components.com/, преузето Јуна 2024.
[7] https://developer.mozilla.org/en-US/docs/Web/HTTP/Overview, преузето Јуна 2024.
[8] https://restfulapi.net/, преузето Јуна 2024.
[9] https://medium.com/expedia-group-tech/onion-architecture-deed8a554423, преузето Јуна 2024.
[10] https://www.techtarget.com/searchapparchitecture/definition/CQRS-command-query-responsibility-segregation, преузето Јуна 2024.
[11] https://docs.microsoft.com/en-us/dotnet/csharp/, преузето Јуна 2024.
[12] https://www.tutorialsteacher.com/core/dotnet-core, преузето Јуна 2024.
[13] https://docs.microsoft.com/en-us/ef/, преузето Јуна 2024.
[14] https://www.codecademy.com/article/what-is-crud, преузето Јуна 2024.
[15] https://www.postgresql.org/, преузето Јуна 2024.
[16] https://www.w3schools.com/sql/, преузето Јуна 2024.
[17] https://sendgrid.com/en-us, преузето Јуна 2024.
[18] https://aws.amazon.com/s3/, преузето Јуна 2024.
[19] https://www.geoapify.com/, преузето Јуна 2024.
[20] https://openai.com/index/chatgpt/, преузето Јуна 2024.
[21] https://aws.amazon.com/what-is/api/, преузето Јуна 2024.

Кратка биографија:



Миле Кајтез је рођен 30.06.1997. године у Новом Саду. Дипломирао је на Факултету техничих наука у Новом Саду, на смеру Примењено софтверско инжењерство 2022. године. Потом, те године, уписује мастер академске студије на истом смеру и завршава их 2024. године бранећи тему "*Развој веб апликације за презентовање података о факултетима и давање предлога употребом ChatGPT AI модела*".