**PREDSTAVITEV:**

**SLIDE 1**

Pozdravljeni moje ime je Luka in danes vam bom predstavil diplomsko nalogo z naslovom Elektronsko naročanje v restavraciji, ki smo ga izdelali pod mentorstvom Mire Trebar in semenarstvom Davida Jelenca

**SLIDE 2**

Slovenija velja za državo z veliko restavracijami, vendar le redke med njimi uporabljajo napredne sisteme naročanja kot npr. ena izmed večjih verig s hitro prehrano, McDonald's. V Sloveniji so obstajali projekti s podobnimi idejami, vendar nismo zasledili takšnih rešitev. Problematiko smo vzeli kot motivacijo ter si zadali cilj, da izdelamo aplikacijo kot pripomoček za optimizacijo dela natakarjev oziroma bolj kakovostno postrežbo.

Zamislili smo si sistem za oddajanje naročil v restavracijah, ki ne bi nadomestil ljudi v strežbi, temveč bi služil kot pregledovalnik ponudbe z možnostjo naročanja hrane in pijače. Aplikacija bi bila na tablicah, ki bi bile nameščene na vsaki mizi restavracije. Stranka bi bila tista, ki bi se odločala, ali želi pri naročanju uporabiti stik z osebo v strežbi ali bi naročila z uporabo aplikacije na tablici.

**SLIDE 3**

Načrtovanje aplikacije je potekalo v treh delih, in sicer izdelava diagrama primerov uporabe, razvoj arhitekture in izbiro tehnologij. Rešitev je sestavljena iz dveh spletnih aplikacij. Ena aplikacija je namenjena gostom, druga pa natakarjem in kuharjem.

Uporabili smo trenutno najbolj razširjeno arhitekturo, ki jo sestavljajo podatkovna baza, strežnik in aplikacija oziroma odjemalec. To je koncept, ki ga je mogoče prilagajati predvsem z uporabniškega vidika. Podatkovna baza je namenjena shranjevanju vseh podatkov za posamezno restavracijo. Strežnik implementira vmesnik RESTful, ki odjemalcu oziroma aplikaciji posreduje podatke iz podatkovne baze.

**SLIDE 4**

Na podlagi predvidenih zahtev oziroma funkcionalnosti smo izdelali logičen podatkovni model. Podatkovna baza je sestavljena iz šestih tabel, ki so: ProductType, Product, Order, ProductOrder, Table, User.

Tabela ProductType je namenjana zapisom za vrste jedi (predjed, glavna jed, sladica) in pijač (sokovi, piva, vina, itd.).

Tabela Product je namenjena opisu hrane in pijače ter je sestavljena iz atributov.

Tabela Order je namenjena zapisovanju naročil in njihovih podrobnosti.

Tabela ProductOrder je namenjena količini in končni ceni hrane in pijače v naročilu.

Tabela Table je namenjena označevanju miz v restavracijah.

Tabela User je namenjena le natakarjem in kuharjem.

Strukturo podatkovne baze smo definirali s pomočjo programa Toad Data Modeler.

Fizični podatkovni model pa smo implementirali v sistemu za upravljanje podatkovne baze (angl. Database Management System, DBMS) MySQL.

**SLIDE 5**

Strežnik predstavlja vmesnik med podatkovno bazo in odjemalcem. Najpomembneje je, da je sistem zanesljiv, saj odjemalec brez strežnika ne more delovati. Izbrali smo arhitekturo REST. Sama arhitektura omogoča, da odjemalec z zahtevami pridobiva podatke od strežnika, ki jih s pomočjo povezav URI oglašuje na relativnih povezavah. Strežnik smo napisali v programskem jeziku Python z vključitvijo knjižnic Flask, MySQL, SocketIO in CORS. Načela arhitekture REST so sledeča:

1. Odjemalec-strežnik zahteva ločitev odjemalca od strežnika, kar odjemalcu onemogoča neposredno povezljivost s podatkovno bazo in s tem poenostavi razširljivost uporabniškega dela. Tako se lahko uporabniški in strežniški del razvijata ali zamenjujeta neodvisno.
2. Brez stanja morajo biti vse interakcije med strežnikom in odjemalcem. Strežnik ne sme shranjevati nobenih stanj oziroma mora vsako zahtevo odjemalca obravnavati kot popolnoma novo.
3. Predpomnjenje prinaša izboljšanje zmogljivosti za odjemalca in omogoča razširljivost strežnika, ker se obremenitev zmanjša. V naši aplikaciji tega nismo uporabili, saj nimamo tako zahtevnih virov.
4. Večslojni sistem je sestavljen iz hierarhičnih slojev, kjer npr. za vmesnik API uporabimo strežnik A, za shranjevanje podatkov strežnik B ter strežnik C za avtenticiranje zahtev. S tem odjemalec ne more ugotoviti, ali komunicira s končnim strežnikom ali posrednikom.
5. Izvajanje programske kode na zahtevo je opcijsko načelo. Strežnik na zahtevo odjemalca pošlje oziroma izvede programsko kodo za odjemalca. To smo vpeljali s pomočjo spletnih vtičnikov SocketIO (potrebe pridobivanja podatkov v realnem času). Implementirali smo ga na strežniku in odjemalcu. Zagotavlja dvosmerno komunikacijo oziroma komunikacijo na podlagi dogodkov. Deluje na vseh platformah, brskalnikih ali napravah. Uporabili smo ga zaradi medsebojnega obveščanja odjemalcev o spremembah v podatkovni bazi. Na primer: ob gostovi spremembi naročila se te razlike preverijo na strežniku in vpišejo v podatkovno bazo, natakarja pa se o tem obvesti
6. Enotni vmesnik med strežnikom in odjemalcem. Vsak vir mora vsebovati povezavo, ki kaže na svoj relativni URI. Podatki so odjemalcu na voljo v podatkovnem formatu JSON. Strežnik s pomočjo knjižnice MySQL najprej prebere podatke iz podatkovne baze in jih predstavi na določeni relativni povezavi URI, ki jo določimo mi. Slika prikazuje uporabo relativne poti. Strežnik vsebuje 34 relativnih povezav

Aplikacija omogoča urejanje določenih podatkov, zato smo morali zagotoviti ustrezno varnost za njihovo urejanje. Prva raven varnosti je avtorizacija odjemalca. Zahtevki, ki se pošiljajo na strežnik, morajo biti omogočeni samo avtoriziranim odjemalcem. Zato smo uporabili piškotke HTTP, ki so izdelani za spletne brskalnike in so namenjeni za sledenje, prilagajanje in shranjevanje informacij o posamezni seji uporabnika. Vsi piškotki so shranjeni pri odjemalcu in so kriptografsko zaščiteni pred morebitnimi nepooblaščenimi dostopi. S tem odjemalcu preprečujemo spreminjanje podatkov v piškotkih oz. lahko nedovoljene spremembe podatkov na strežniku zaznamo. Uporabili smo Flask-Login, ki omogoča vse funkcionalnosti za upravljanje uporabniških sej.

Na sliki je prikazano delovanje piškotkov HTTP. Strežnik ob prvem zahtevku, torej ob uspešni prijavi, odjemalcu vrne piškotek. Odjemalec ob vsakem nadaljnjem zahtevku doda piškotek, s katerim strežnik overi zahtevo odjemalca.

**SLIDE 6**

Odjemalca bi lahko implementirali v spletnih tehnologijah (HTML, CSS in JavaScript) ali pa v namenski mobilni aplikaciji. Odločili smo se za spletne tehnologije, kjer je glavnina odjemalca implementrana v programskem jeziku JavaScript s pomočjo ogrodja Vue. Naredili smo odziven in reaktiven vmesnik, ki deluje v stvarnem času. Reaktivnost je programska paradigma, ki nam omogoča, da se na deklarativni način prilagodimo spremembam. Tako deluje tudi reaktivnost v aplikacijah, kjer je podatek lahko vezan na več funkcij oziroma delov programske kode, ki se ob spremembi vrednosti posodobijo. Vue je namenjen izdelavi enostranskih aplikacij (angl. Single-Page Application, SPA), saj vsebuje samo eno datoteko HTML. To prednost smo izkoristili s pomočjo ostalih knjižnic, ki so nam omogočale enostavnejšo izdelavo aplikacije. Uporabili smo naslednje:

Vue CLI velja kot standardno orodje za ekosistem Vue. Zagotavlja, da že pri gradnji novega projekta poveže različne dodatke med seboj.

Vuex je knjižnica za shranjevanje vrednosti v aplikacijah Vue.js. Služi kot centralizirana baza podatkov za vse komponente v aplikaciji.

Vue-Router je uradni usmerjevalnik za Vue.js. Integrira se globoko z jedrom Vue.js, tako da poenostavi izdelavo aplikacij SPA. Usmerjevalnik je mišljen v smislu usmerjanja na druge komponente, ki v Vue.js predstavljajo druge poglede oziroma podstrani.

Vuetify je eden izmed mnogih uporabniških vmesnikov, ki je zgrajen na vrhu Vue.js.

Programsko izvedbo za odjemalca smo razdelili v tri vloge oziroma dve aplikaciji. Ena aplikacija je namenjena natakarjem in kuharjem, ločuje se s prijavnim oknom in videzom vmesnika. Nismo naredili ločene aplikacije, saj ni bilo potrebe, funkcije za oba uporabnika so namreč zelo podobne. Druga aplikacija je namenjena samo gostom in je sestavljena iz več pogledov. Ločili smo jih zaradi varnosti, lažjega razvijanja in preglednosti, saj gre za dve popolnoma različni aplikaciji. Vse funkcionalnosti in delovanje ene in druge aplikacije so opisani v naslednjem poglavju.

Za potrebe pridobivanje podatkov smo uporabili Axios, ki je namenjen procesiranju zahtevkov HTTP. To pomeni, da podatke, ki jih oglašuje strežnik, s pomočjo te knjižnice pridobimo za odjemalca

Enako kot za strežnik smo tudi za odjemalca uporabili spletni vtičnik SocketIO. Vključili smo ga v obeh aplikacijah, in sicer za oddajanje naročil, posodabljanje naročil, obveščanje gosta o stanju naročila itd. Najprej smo hoteli uporabiti samodejno osveževanje na določen časovni interval, vendar je uporaba spletnih vtičnikov hitrejša in učinkovitejša.

**SLIDE 7**

Začetni pogled vmesnika za gosta vsebuje napis za dobrodošlico, ki bi ga lahko zamenjali oglaševanje, predstavitev restavracije ali karkoli bi si potencialni kupec zaželel imeti. V zgornjem desnem kotu se nahajat gumb call waiter za priklic natakarja in števec skupne cene artiklov v nakupovalni košarici. Gumb call waiter je namenjen gostom, ki aplikacije ne želijo uporabljati, ali v primeru pomoči, če ima gost kakšna vprašanja ali pride do kakršnihkoli težav. V spodnjem levem kotu se nahajata status in identifikacijska številka naročila.

Zavihki na levi strani predstavljajo seznam vseh vrst hrane in pijače, ki kažejo na podstrani ponudbe restavracije. Gostu to predstavlja ponudbo, med katero izbira pri dodajanju hrane in pijače v nakupovalno košarico.

Vsaka hrana ali pijača vsebuje sliko, ime, opis in ceno. Poleg tega vsebuje še gumb add to cart, ki ob kliku izgine in prikaže tri nove gumbe + (povečaj količino), – (zmanjšaj količino) in remove (odstrani). Nakupovalna košarica oziroma je skupno mesto vse hrane in pijače v seznamu za naročilo. Vse slike hrane in pijače so shranjene v datotečnemu sistemu spletnega strežnika. Naročilo se odda s klikom na gumb place order, ki gosta preusmeri na prvo stran in obvesti s pojavnim sporočilom, prikazanim. Ko je naročilo oddano, lahko gost ponovno dodaja hrano in pijačo v košarico, vendar je že oddani hrani in pijači onemogočeno zmanjšanje količine ali brisanje. Naročilo je sprejeto, ko ga natakar potrdi, kar spremeni status naročila in prikaže pojavno sporočilo. Če natakar uredi naročilo, ga mora gost pregledati in ponovno oddati. Tudi v tem primeru je gost obveščen s statusom in pojavnim sporočilom. Naročilo se zaključi s klikom na gumb request receipt, ki odpre pojavno okno za izbiro načina plačila.

**SLIDE 8**

Začetni pogled vmesnika je enak za natakarja in kuharja, saj gre za skupno aplikacijo, kjer se pogledi razlikujejo glede na vlogo uporabnika, ki je določena v podatkovni bazi. Slika~\ref{NatKuh\_zac} prikazuje prijavno okno. Po prijavi natakarja ali kuharja se uporabniško ime izpiše v levem spodnjem kotu. Na zgornji levi strani se prikaže zavihek z dvema podstranema in odjavnim gumbom \textit{sign out}. Prva podstran, imenovana \textit{dashboard}, ali prva stran ob uspešni prijavi prikazuje napis za hitrejšo razlikovanje med vlogami. Natakar ima v desnem zgornjem kotu še števec čakajočih in zaključenih naročil.

Natakarju se v zavihku all orders pojavijo vsa nezaključena. Vsako naročilo vsebuje naslednje podatke: identifikacijska številka naročila, čas oddaje naročila, status naročila, status kuharja, način plačila in številka mize. Izvaja lahko popoln nadzor nad naročili, vendar ob vsaki izvedeni akciji obvesti gosta (slike pojavnih sporočil iz prejšnjega poglavja). Novo naročilo mora natakar najprej potrditi z gumbom confirm ali urediti z gumbom check detail. Natakar lahko ureja celotno naročilo, kar pomeni, da lahko spreminja količino hrane in pijače ali jo briše iz naročila. Ko zaključi urejanje, mora klikniti na gumb update order. Celotno naročilo lahko potrdi šele, ko dobi kuharjevo potrditev o hrani. Ko je naročilo potrjeno, natakar čaka na kuharjevo potrditev o pripravi hrane, da jo lahko postreže. Natakar postreženo naročilo označi s klikom na gumb served. Če gost zahteva račun, se natakarju izpiše način plačila v zadevi payment. Naročilo se zaključi, ko natakar natisne račun s klikom na gumb invoice.

**SLIDE 9**

Kuharju se v zavihku food orders pojavijo vsa nezaključena naročila, ki vsebujejo hrano. Kuhar mora naročilo najprej pregledati z gumbom check details ter ga sprejeti ali zavrniti z gumboma confirm in decline. Naročilo lahko kuhar zavrne, če mu zmanjka sestavin. Ko kuhar zaključi pripravo hrane, o tem obvesti natakarja s klikom na gumb done, ki izbriše naročilo s seznama. Ob vsaki izvedeni akciji se pri vsakem naročilu spremeni status kuharja, ki je lahko: updated, confirmed, decline in done. Status updated se uporabi, če gost k naročilu doda izdelke.