**Povzetek**

**Abstract**

1. **Uvod**

Slovenija velja za državo z veliko število restavracij, vendar le malo iz med njih uporablja napredne sisteme naročanja kot npr. ena izmed večjih verig s hitro prehrano, McDonalds. V Sloveniji je bilo nekaj projektov s podobnimi idejami, vendar z napačnimi cilji zaradi katerih so bili neuspešni. Eden izmed razlogov da jim ni uspelo je bilo sabotiranje sistemov s strani natakarjev, saj so misli da bo tehnologija zamenjala njegove službe. V zavedanju teh problematik smo se odločil narediti diplomski nalogo na to temo.

Zamislili smo si sistem za oddajanje naročil v restavracijah, ki ne bi bil namenjen zamenjavi ljudi v strežbi, temveč kot pregledovalnik (angl. Menu) oziroma naročanju hrane in pijače. Aplikacija za stranke bi bila na tablicah, ki bi bile locirane na vsaki mizi restavracije. Stranka bi bila tista, ki bi se odločila ali želi pri naročanju uporabiti stik z osebo v strežbi ali bi naročila z uporabo aplikacije na tablici. Natakarju bi s tem omogoči več časa, katerega bi lahko posvetil pripravi pijače, kvaliteti postrežbe in ostalih dolžnosti. Tudi stranke, katere sedaj veljajo za bolj zahtevne in neučakane na vseh področij, bi bile hitreje in bolj kvalitetno postrežene. Tako bi imeli poleg restavracij z hitro prehrano tudi restavracije s hitro postrežbo.

Aplikacija bi bila sestavljena iz treh osnovnih pogledov, ki bi bili namenjeni gostu, natakarju in kuharju.

1. **Struktura in razvoj aplikacije**

Aplikacija je sestavljena iz treh pogledov, in sicer uporabnik oziroma gost, natakar in kuhar. Gost ima imel na voljo pregled vseh pijač in jedi, ki jih restavracija ponuja. Natakar ima popoln nadzor nad naročili, to pomeni sprejemanje, urejanje in zaključevanje naročil. Kuhar ima nadzor nad naročili, ki vsebujejo hrano.

Za delovanje sistema sem uporabil podatkovno bazo, strežnik in spletno aplikacijo, katero bi lahko zamenjali tudi za mobilno oziroma katero koli. Gre za koncept, ki ga je moč prilagajati predvsem uporabniškega vidika. Podatkovna baza je namenjena shranjevanju vseh podatkov in katera struktura je bolj podrobno opisana v poglavju 2.1. Za prikazovanje njenih podatkov v končni aplikaciji smo potrebovali še strežnik. Njegov namen je prikazati podatke iz podatkovne baze na način, ki je standardiziran in ga je mogoče uporabiti povsod in to je JSON format.

Za lažjo predstavo tudi diagram delovanja, vendar bi predlaganega zamenjal z manj podrobnim opisom. Tistega bi uporabil v 4. poglavju. Da bi si bralec lažje predstavljal primere delovanja.

Opis osnovnega delovanja za lažjo predstavo, kaj čemu služi. Podatkovna baza->strežnik->odjemalec sestavljen iz dveh delov (kuhar, natakar/kuhar). RESTful vmesnik, ki servira podatke odjemalcu. Kaj je to RESTful? Slika za lažjo predstavo.

Omejitve: en naročilo na mizo, kar pomeni da vsi za mizo naročajo skupaj. Neomajno število miz, neomejeno natakarjev (100 je omejitve podatkovne baze. 100 poizvedb na enkrat – mogoče bi bilo bolje napisat samo neomejeno, ker ne vem če obstaja restavracija z več kot 100 natakarji), en glavni kuhar.

* 1. ***Podatkovna baza***

Struktura podatkovne baze. Tabele in povezave. Kako je nastala nova tabela zaradi M:N povezave. Čemu je namenjena vsaka tabela in kako je sestavljana.

* + 1. ***MySQL***

Opis MySQL in zakaj ravno MySQL.

* 1. **Strežnik**

Predstavlja RESTful vmesnik, napisan v Python-Flask bolj podrobno opisano v spodnjem poglavju. Kako se strežnik povezuje s podatkovno bazo in podatke prikazuje v JSON formatu na spletnem vmesniku. Websocket rešitev, ki ob vsakem novem zapisu v bazo opozori odjemalca o novih podatkih v bazi. Težave z večkratnimi zahtevki na enkrat. Rešitev Flask-MySQL knjižnica.

* + 1. ***Python/Flask***

Opis Python in uporaba knjižnice Flask ter njenih ostalih knjižnic.

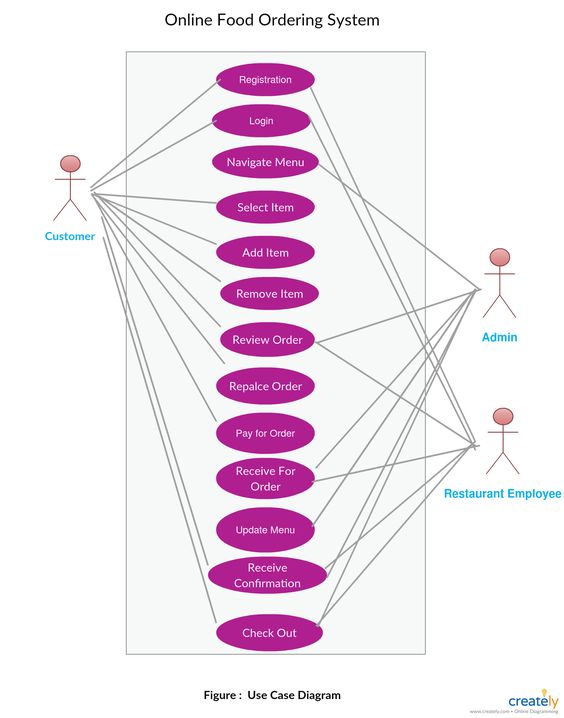
* 1. ***Odjemalec***

Aplikacija napisana v programskem jeziku JavaScript in Frame Vue. Gre za moderen, odziven in reaktiven vmesnik, ki deluje v realnem času. Bolj podroben opis v spodnjih poglavjih.

* + 1. ***Vue***
    2. ***Kaj je reaktivnost?***
    3. ***Delovanje reaktivnost v Vue***
    4. ***Knjižnice in dodatki***

Samo delovanje aplikacije in njenih pogledov (gost, natakar/kuhar skupaj kateri se loči glede na prijavo). Branje in prikazovanje podatkov pridobljenih iz strežnika/baze (axios). Rešitev osveževanje podatkov s pomočjo Websocket-ov s katerimi smo omogočili samodejno sinhronizacijo vseh pogledov.

1. **Delovanje aplikacije**

****

* 1. ***Vmesnik za gosta***

Slikovni prikaz vmesnika in vse njegove funkcionalnosti oziroma pogledi. Primeri uporabe:

1.) naročanje hrane in pijače

2.) oddajanje novih naročil, kar pomeni urejanje obstoječega, stanje se spremeni samo natakarju o dodatnem naročilu pijače in hrane

3.) izbris pijače in hrano katera mu je že bila servirana – ni mogoče

4.) način zaključevanje naročila

5.) naročanje hrane in pijače brez aplikacije – tipka »call waiter«

* 1. ***Vmesnik za natakarja***

Slikovni prikaz vmesnika in vse njegove funkcionalnosti oziroma pogledi.

1.) vpis v pogled natakarja

2.) pregled vseh naročil

3.) sprejemanje naročil – obvestilo na strani gosta

4.) spreminjanje naročil in kako aplikacija obvesti kuharja in gosta

5.) zaključevanje naročila in možnosti prenosa v druge aplikacije za izdajo računa

* 1. ***Vmesnik za kuharja***

Slikovni prikaz vmesnika in vse njegove funkcionalnosti oziroma pogledi.

1.) vpis v pogled kuharja

3.) pregled vseh naročil

3.) potrjevanje naročil oziroma zavračanje v primeru, da kuhar nima na voljo vseh sestavin

1. **Diskusija**
   1. ***Implementacija programa v realnosti***

Dve rešitvi za restavracije.

1. Cenejša različica bi bile QR kode na mizah, v katerih bi bilo zapisano ime restavracije in številka mize. To kodo bi gost skeniral z mobilnim telefon in bil preusmerjen v aplikacijo. Celoten sistem bi bil na javnem Internetu tudi za kuharje in natakarje. Potrebno bi bilo zagotoviti, da ne bi prihajalo do »fantomskih« naročil, kjer bi nepridipravi oddaljeno oddajali neveljavna naročila.
2. Dražja različica je, da bi imela vsaka miza svojo tablico. Sistem bi bil dostopen samo lokalno, znotraj restavracije, kar bi bilo iz vidika varnosti precej bolj varno. Restavracija poleg tablic potrebuje še lokalni spletni strežnik.

V prvem primeri bi bil strežnik postavljen za celotno Slovenijo, in bi omogočal storitev vsem restavracija po Sloveniji. V primeru QR kod bi morala biti dodatna varnost, saj bi lahko prišlo do napadov izven restavracije. Rešitev bi bilo geslo, ki se spreminja za vsako naročilo – izda ga natakar.

* 1. ***Izboljšave***

Izboljšav za to aplikacijo je veliko, ker obstaja veliko zadev, ki bi aplikacijo dvignilo na naslednji nivo.

Par primerov izboljšav.

Statistika za lastnika restavracije, ki bi poleg vseh podatkov lahko računala oceno nabave za prihodnji mesec. Ta bi imel svoj ločen uporabniški račun, ki bi omogočal tudi nastavitve v pogledu za gosta –lahko bi spreminjal številko mize.

Dodatno spremljanje hrane in pijače na strani kuharja/natakarja – katera pijača in hrana je že bila postrežena.

Brezstično plačevanje s kartico direktno na strani gosta.

Več kuharjev, da bi lahko vsak kuhar vedel kaj more delati – tako kot je sedaj implementirano za natakarje.

Če bi aplikacija delovala na centralnem strežniku, bi lahko za vsako restavracijo omogočali tudi dostavo hrane z enakim pogledom, ki bi bil vedno dostopen na skupni spletni strani – npr. kot Ehrana in Wolt.

Vpeljava sistema v restavracije brez potreb po natakarju. Podobne želje so imeli v Lars&Sven.

* 1. ***Konkurenca***

McDonalds, ki rešitev uporablja že nekaj let. Prednosti in slabosti – naročile se lahko izvede elektronsko samo na vhodu, ni mogoče ponovnega hitrega naročila,…

Spletne aplikacije, ki ponujajo dostavo vseh restavracij po Sloveniji npr. ehrana in wolt. To bi bilo mogoče narediti tudi za mojo rešitev. Prednost bi bila, da bi mi lahko to restavracijam ponujal kot paket.

1. **Sklepne ugotovitve**

**OPOMBE:**

* Opis ostalih aplikacij sem izpustil – VisualStudio in GitHub
* V aplikaciji sem popravil »REQUEST RECEIPT«