Univerza v Ljubljani
Fakulteta za računalništvo in informatiko

Luka Horvat

Elektronsko naročanje v restavraciji

diplomsko delo

visokošolski študijski program
prve stopnje
računalništvo in informatika

Mentorica: doc. dr. Mira Trebar
Somentor: as. dr. David Jelenc

Ljubljana, 2021

Besedilo je oblikovano z urejevalnikom besedil LaTeX.

Fakulteta za računalništvo in informatiko izdaja naslednjo nalogo:

Tematika naloge: Elektronsko naročanje v restavracij

Besedilo teme diplomskega dela študent prepiše iz študijskega informacijskega sistema, kamor ga je vnesel mentor. V nekaj stavkih bo opisal, kaj pričakuje od kandidatovega diplomskega dela. Kaj so cilji, kakšne metode uporabiti, morda bo zapisal tudi ključno literaturo.

Na tem mestu zapišite, komu se zahvaljujete za izdelavo diplomske naloge. Pazite, da ne boste koga pozabili. Utegnil vam bo zameriti. Temu se da izogniti tako, da celotno zahvalo izpustite.

Kazalo

Seznam uporabljenih kratic

kratica	angleško	slovensko
CA	classification accuracy	klasifikacijska točnost
DBMS	database management system	sistem za upravljanje podatkovnih baz
SVM	support vector machine	metoda podpornih vektorjev

Povzetek

Naslov: Elektronsko naročanje v restavraciji

Avtor: Luka Horvat

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. Obstajali so projekti, ko so naredili aplikacije, ki bi/so vpeljale nekaj tehnologije v restavracije vendar v Sloveniji niso uspeli. Eden ključnih razlogov zakaj niso uspeli je bilo sabotiranje s strani natakarjev, saj so misli da bo tehnologija zamenjala njegove službe. V zavedanju njihovih problemov sem se odločil narediti diplomski nalogo na to temo. Rezultat je aplikacija, ki jo je mogoče uporabiti samo kot pregled hrane in pijače, ki jo ponuja restavracija ali pa kot sistem za naročanje. V diplomski nalogi najprej opišem kako sem prišel do ideje, sam razvoj aplikacije in njeno končno delovanje. Predstavim vse težave na katere sem naletel med samim razvojem. Na koncu sem pripravil tudi primerjavo s konkurenco na trenutnem trgu oziroma z aplikacijo, ki vem da ni uspela. Seveda sem pripravil tudi seznam izboljšav, ki bi mojo aplikacijo lahko pripeljalo med eno izmed boljših.

Ključne besede: Slovenija, naročanje, propadel projekt, spletna aplikacija.

Abstract

Title: Diploma thesis sample

Author: Luka Horvat

This sample document presents an approach to typesetting your BSc thesis using LaTeX. A proper abstract should contain around 100 words which makes this one way too short.

Keywords: computer, computer, computer.

Poglavje 1

Uvod

Tehnologije v sedanjem času predstavljajo velik napredek na vseh področij. Na strani gostinstva ni drastičnih napredkov, saj sem to ugotovil kar na osnovi lastnih izkušen. Kot študent fakultete za računalništvo in informatiko sem opravljal delo v eni izmed restavracij, saj biti študent z dodatnim zaslužkom ni dilema. Opravljal sem delo dostavljavca hrane v eni izmed bližnji restavraciji, hkrati pa opazoval potek dela v kuhinji in strežbi. Kasneje sem napredoval v strežbo in kot programer hitro opazil stvari, ki bi se jih dalo izboljšati. Najbolj me je motilo nezadovoljstvo strank ob veliki zasedenosti restavracije. Preprosto en človek ne uspe postreči več kot eno mizo na enkrat. Zato sem si zamislil sistem za naročanje hrane, ki ne bi bil namenjen zamenjavi ljudi v strežbi, vendar samo kot pregledovalnik (angl. Menu) ali pa naročanju jedi in pijače. Stranka bi bila tista, ki bi se odločila ali želi pri naročanju uporabiti stik z osebo v strežbi ali bi preprosto naročila z uporabo aplikacije na tablici, ki bi bila vedno dosegljiva na vsaki mizi.

Poglavje 2

Uporabljene tehnologije in programske opreme

V diplomski nalogi sem se srečeval predvsem z naslednjimi programskimi jeziki: JavaScript, Python, SQL. Vse podrobnosti pa so opisane v spodnjih poglavjih.

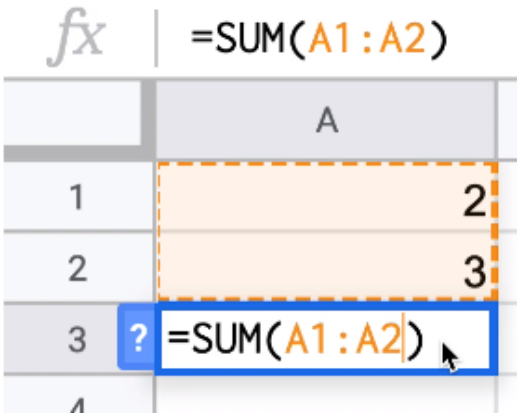
2.1 Vue.js

JavaScript je programski jezik, zaradi katerega so spletne strani postale dinamične in bolj zmogljive. Z njem smo programsko kodo, ki je bila na strežniškem delu, preselili v brskalnik. Tako smo na začetku dobili veliko JavaScript programske kode povezane z različnimi HTML in CSS datotekami brez kakršne koli formalne organizacije (poznano kot «script»). Zaradi tega smo razvijalci začeli uporabljati JavaScript Frameworks, saj poenostavijo izdelavo aplikacij.

Vue je eden izmed mnogih kot npr. Angular, Ember, React,... poznan pa je predvsem zaradi enostavnosti za upravljanje in izvajanje testov. Vsem pa je skupna točka reaktivnost, vendar v drugačnem pomenu besede. Gre za to, da se aplikacija postopno prilagaja glede na vrednosti podatkov.

2.1.1 Kaj je reaktivnost?

Reaktivnost [2], je programska paradigma, ki nam omogoča, da se na deklarativni način prilagodimo spremembam. Dober primer reaktivnosti je npr. funkcija SUM, ki jo uporabljamo v Excelu. Slika 2.1 prikazuje primer v Excelu.



Slika 2.1: Primer fukcije SUM v programu Excel

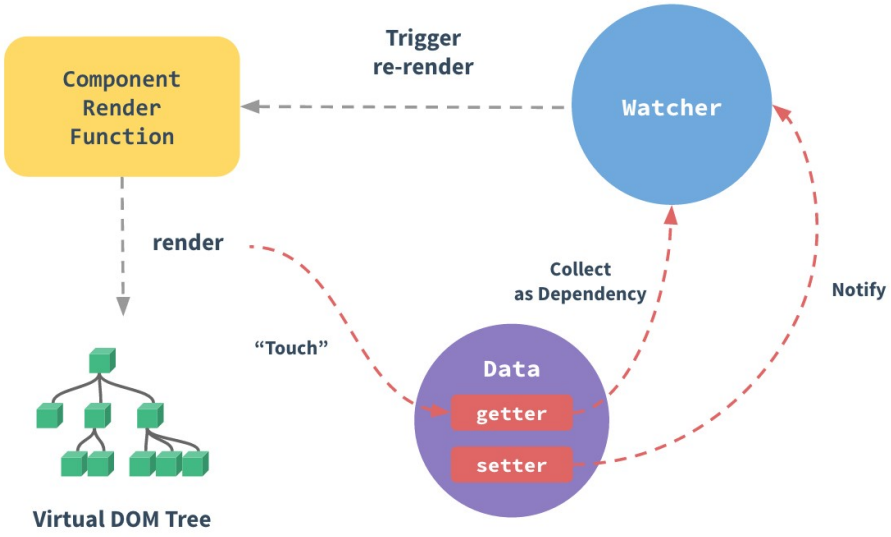
Če vstavimo številko 2 v prvo celico in številko 3 v drugo celico ter izberemo funkcijo SUM teh dveh celic. Kot rezultat dobimo vsoto obeh številk skupaj, kar ni nič posebnega. Vendar če bomo spremenili vrednost prve celice, bo funkcija SUM avtomatsko posodobila skupno vrednost. Tako deluje tudi reaktivnost v aplikacijah za razliko, da je na nek podatek lahko vezanih več funkciji oziroma delov programske kode, ki se ob spremembi njegove vrednosti posodobijo.

2.1.2 Delovanje reaktivnost v Vue

Vue se torej v primerjavi z navadnim JavaScriptom sprehodi skozi podatke in njihove lastnosti (angl. Properties) pretvori v fukciji Getter in Setter, ki sta nevidni uporabniku [1]. Poglejte si sliko 2.2 za lažjo predstavo.

Torej funkcija Getter pokliče instanco Watcher z namenom odvisnosti do

drugih komponent. To pomeni, če je podatek označen kot odvisen (angl. Dependency) bodo nekateri deli programske kode oziroma funkcije poklicane vsakič, ko se spremeni vrednost podatka. Funkcija Setter pa obvesti instanco Watcher, vsakič ko se podatku spremeni lastnost. Ta poskrbi, da se pokliče funkcijo upodabljanja (angl. Render) tiste komponente, ki potem prikaže spremembe v samem pogledu aplikacije.



Slika 2.2: Kateri dialekt uporabljati?

2.1.3 Knjižnice in dodatki

Tako kot vsak programski jezik ima tudi Vue svoje dodatke, ki pomagajo pri razvoju aplikaciji. Spodnje sem uporabljal sam in so vredni omembe.

Vue CLI velja za standardno orodje za ekosistem Vue. Zagotavlja, da že pri gradnji novega projekta povezuje različne dodatke med seboj. To omogoča razviljacu, da se bolj osredotoči na programiranje in ne na povezovanje njih v celoto. Zadeva izgleda nekako tako, da preko CLI

vmesnika izbereš kakšen projekt želiš. Imaš seveda že privzete nastavitve, vendar omogoča tudi nastavljanje po meri. Sam sem uporabil Vuex, Vue-Router, ESLint in Vuetify.

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 SPA aplikacij. Usmerjevalki je mišljen v smislu usmerjanja na druge komponente (angl. Component), ki v Vue.js predstavljajo druge poglede, lahko bi rekli podobno kot podstrani.

ESLint je orodje za prepoznavanje in priporočanje o popravkih v programski kodi. Cilje je narediti kodo bolj usklajeno in urejeno, kar pripomore k izogibanju napak.

Vuetify je eden izmed mnogih uporabniških vmesnikov, ki je zgrajen na vrhu Vue.js. Za razliko od drugih vmesnikov je Vuetiy enostaven za učenje z več stotimi komponentami izdelanih po specifikacijah Material Design.

Vue-devtools je zgolj dodatek v brskalniku, ki omogoča lažje sledenje delovanja aplikacije in odpravljanju napak.

2.2 Python/Flask

Flask je eno izmed najbolj popularnih spletno aplikacijski vmesnik (angl. Freamwork). Zasnovan je tako, da omogoča hiter in enostaven začetek z možnostjo razširitve na zapletene aplikacije. Uporabil sem ga za komunikacijo med aplikacijo in podatkovno bazo. Je enostaven za uporabo z veliko podporo na spletu.

2.3 MySQL/Toad DataModler

2.4 Viusal Studio

2.5 XAMPP

2.6 Git Hub

Poglavje 3

Struktura in delovanje aplikacije

Poglavje 4

Test

. Prvi koristen nasvet v zvezi uporabo LaTeXa je, da ta dokument preberete v celoti!

Datoteka `vzorec_dip_Seminar.tex` na kratko opisuje, kako se pisanja diplomskega dela lotimo z uporabo programskega okolja LaTeX [**?**, **?**]. V tem dokumentu bomo predstavili nekaj njegovih prednosti in hib. Kar se slednjih tiče, nam pride na misel ena sama. Ko se srečamo z njim prvič nam izgleda morda kot kislo jabolko, nismo prepričani, ali bi želeli vanj ugrizniti. Toda prav iz kislih jabolk lahko pripravimo odličen jabolčni zavitek in s praktičnim preizkusom LaTeXa najlažje pridemo na njegov pravi okus.

LaTeX omogoča logično urejanje besedil, ki ima v primerjavi z vizualnim urejanjem številne prednosti, saj se problema urejanja besedil loti s programerskega stališča. Logično urejanje besedil omogoča večjo konsistentnost, uniformnost in prenosljivost besedil. Vsebinska struktura nekega besedila pa se odraža v strukturiranem LaTeXovem kodiranju besedila.

V 5. poglavju bomo spoznali osnovne gradnike LaTeXa. V 6. poglavju bomo na hitro spoznali besedilne konstrukte kot so izreki, enačbe in dokazi. Naučili se bomo, kako se na njih sklicujemo. 7. poglavje bo predstavilo vključevanje plovk: slik in tabel. Poglavje 8 na kratko predstavi tipične sestavne dele strokovnega besedila. V 9. poglavju omenjamo nekaj najpogostejših slovničnih napak, ki jih delamo v slovenščini. V 10. poglavju je še

nekaj koristnih nasvetov v zvezi z uporabo LaTeXa. V 11. poglavju se bomo srečali s sklicevanjem na literaturo, 12. poglavje pa govori o formatu PDF/A, v katerem morate svojo diplomo oddati v sistemu STUDIS. Sledil bo samo še zaključek.

4.1 Vizija

Poglavje 5

Osnovni gradniki LaTeXa

LaTeX bi lahko najbolj preprosto opisali kot programski jezik namenjen oblikovanju besedil. Tako kot vsak visokonivojski programski jezik ima tudi LaTeX številne ukaze za oblikovanje besedila in okolja, ki omogočajo strukturiranje besedila.

Vsi LaTeXovi ukazi se začnejo z levo poševnico `\`, okolja pa definiramo bodisi s parom zavitih oklepajev { in } ali z ukazoma `\begin{ }` in `\end{ }`. Ukazi imajo lahko tudi argumente, obvezni argumenti so podani v zavitih oklepajih, opcijski argumenti pa v oglatih oklepajih.

Z ukazi torej definiramo naslov in imena avtorjev besedila, poglavja in podpoglavja in po potrebi bolj podrobno strukturiramo besedila na spiske, navedke itd. Posebna okolja so namenjena zapisu matematičnih izrazov, kratki primeri so v naslednjem poglavju.

Vse besedilne konstrukte lahko poimenujemo in se s pomočjo teh imen nato kjerkoli v besedilu na njih tudi sklicujemo.

LaTeX sam razporeja besede v odstavke tako, da optimizira razmike med besedami v celotnem odstavku. Nov odstavek začnemo tako, da izpustimo v izvornem besedilu prazno vrstico. Da besedilo skoči v novo vrstico pa ukažemo z dvema levima poševnicama. Število presledkov med besedami v izvornem besedilo ni pomembno.

Poglavje 6

Matematično okolje in sklicevanje na besedilne konstrukte

Matematična ali popolna indukcija je eno prvih orodij, ki jih spoznamo za dokazovanje trditev pri matematičnih predmetih.

Izrek 6.1 *Za vsako naravno število n velja*

$$n < 2^n. \tag{6.1}$$

Dokaz. Dokazovanje z indukcijo zahteva, da neenakost (6.1) najprej preverimo za najmanjše naravno število – 0. Res, ker je $0 < 1 = 2^0$, je neenačba (6.1) za $n = 0$ izpolnjena.

Sledi indukcijski korak. S predpostavko, da je neenakost (6.1) veljavna pri nekem naravnem številu n, je potrebno pokazati, da je ista neenakost v veljavi tudi pri njegovem nasledniku – naravnem številu $n + 1$. Računajmo.

$$\begin{aligned} n + 1 &< 2^n + 1 && (6.2)\\ &\leq 2^n + 2^n && (6.3)\\ &= 2^{n+1} \end{aligned}$$

Neenakost (6.2) je posledica indukcijske predpostavke, neenakost (6.3) pa enostavno dejstvo, da je za vsako naravno število n izraz 2^n vsaj tako velik kot 1. S tem je dokaz Izreka 6.1 zaključen. □

Opazimo, da je LaTeX številko izreka podredil številki poglavja. Na podoben način se lahko sklicujemo tudi na druge besedilne konstrukte, kot so med drugim poglavja, podpoglavja in plovke, ki jih bomo spoznali v naslednjem poglavju.

Poglavje 7

Plovke: slike in tabele

Slike in daljše tabele praviloma vključujemo v dokument kot plovke. Pozicija plovke v končnem izdelku ni pogojena s tekom besedila, temveč z izgledom strani. LaTeX bo skušal plovko postaviti samostojno, praviloma na mestu, kjer se pojavi v izvornem besedilu, sicer pa na vrhu strani, na kateri se na takšno plovko prvič sklicujemo. Pri tem pa bo na vsako stran končnega izdelka želel postaviti tudi sorazmerno velik del besedila. V skrajnem primeru, če imamo res preveč plovk na enem mestu besedila, ali če je plovka previsoka, se bo LaTeX odločil za stran popolnoma zapolnjeno s plovkami.

Poleg tega, da na položaj plovke vplivamo s tem, kam jo umestimo v izvorno besedilo, lahko na položaj plovke na posamezni strani prevedenega besedila dodatno vplivamo z opcijami `here, top` in `bottom`. Zelo velike slike je najbolje postaviti na posebno stran z opcijo `page`. Skaliranje slik po njihovi širini lahko prilagodimo širini strani tako, da kot enoto za dolžino uporabimo kar širino strani, npr. `0.5\textwidth` bo raztegnilo sliko na polovico širine strani.

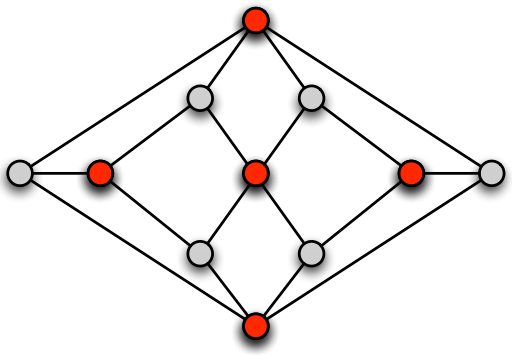
Na vse plovke se moramo v besedilu sklicevati, saj kot beseda pove, plovke plujejo po besedilu in se ne pojavijo točno tam, kjer nastopajo v izvornem besedilu. Sklic na plovko v besedilu in sama plovka naj bosta čimbližje skupaj, tako da bralcu ne bo potrebno listati po diplomi. Upoštevajte pa, da se naloge tiska dvostransko in da se hkrati vidi dve strani v dokumentu!

Na to, kje se bo slika ali druga plovka pojavila v postavljenem besedilu torej najbolj vplivamo tako, da v izvorni kodi plovko premikamo po besedilu nazaj ali naprej!

Tabele ja najbolje oblikovati kar neposredno v LaTeXu, saj za oblikovanje tabel obstaja zelo fleksibilno okolje `tabular` (glej tabelo 7.1). Slike po drugi strani pa je bolje oblikovati oziroma izdelati z drugimi orodji in programi in se v LaTeXu le sklicevati na ustrezno slikovno datoteko.

7.1 Formati slik

Bitne slike, vektorske slike, kakršnekoli slike, z LaTeXom lahko vključimo vse. Slika 7.1 je v `.pdf` formatu. Pa res lahko vključimo slike katerihkoli forma-



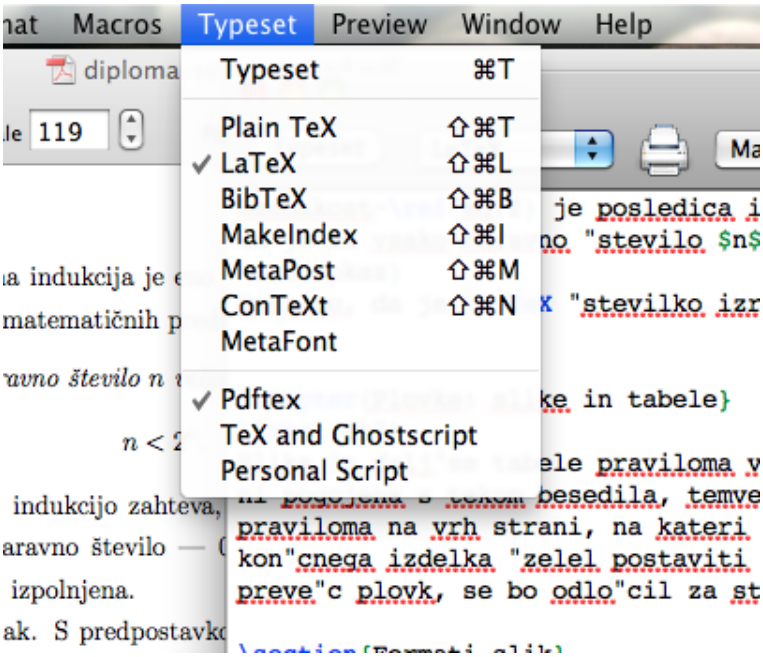
Slika 7.1: Herschelov graf, vektorska grafika.

tov? Žal ne. Programski paket LaTeX lahko uporabljamo v več dialektih. Ukaz `latex` ne mara vključenih slik v formatu Portable Document Format `.pdf`, ukaz `pdflatex` pa ne prebavi slik v Encapsulated Postscript Formatu `.eps`. Strnjeno je vključevanje različnih vrst slikovnih datotek prikazano v tabeli 7.1.

Nasvet? Odločite se za uporabo ukaza `pdflatex`. Vaš izdelek bo brez vmesnih stopenj na voljo v .pdf formatu in ga lahko odnesete v vsako tiskarno.

ukaz/format	`.pdf`	`.eps`	ostali formati
`pdflatex`	da	ne	da
`latex`	ne	da	da

Tabela 7.1:



Slika 7.2: Kateri dialekt uporabljati?

Če morate na vsak način vključiti sliko, ki jo imate v `.eps` formatu, jo vnaprej pretvorite v alternativni format, denimo `.pdf`.

Včasih se da v okolju za uporabo programskega paketa LaTeX nastaviti na kakšen način bomo prebavljali vhodne dokumente. Spustni meni na Sliki 7.2 odkriva uporabo LaTeXa v njegovi pdf inkarnaciji — `pdflatex`. Vključena slika 7.2 je seveda bitna.

Na vse tabele se moramo v besedilu, podobno kot na slike, tudi sklicevati, saj kot plovke v oblikovanem besedilo niso nujno na istem mestu kot v izvornem besedilu.

7.1.1 Podnapisi k slikam in tabelam

Vsaki sliki ali tabeli moramo dodati podnapis, ki na kratko pojasnuje, kaj je na sliki ali tabeli. Če nekdo le prelista diplomsko delo, naj bi že iz slik in njihovih podnapisov lahko na grobo razbral, kakšno temo naloga obravnava.

Če slike povzamemo iz drugih virov, potem se moramo v podnapisu k taki sliki sklicevati na ta vir!

Poglavje 8

Struktura strokovnih besedil

Strokovna besedila imajo ustaljeno strukturo, da bi lahko hitreje in lažje brali in predvsem razumeli taka besedila, saj načeloma vemo vnaprej, kje v besedilu se naj bi nahajale določene informacije.

Najbolj osnovna struktura strokovnega besedila je:

naslov besedila, ki naj bo sicer kratek, a kljub temu dovolj poveden o vsebini besedila,

imena avtorjev so običajno navedena po teži prispevka, prvi avtor je tisti, ki je besedilo dejansko pisal, zadnji pa tisti, ki je raziskavo vodil,

kontaktni podatki – poleg imena in naslova institucije je potreben vsaj naslov elektronske pošte,

povzetek je kratko besedilo, ki povsem samostojno povzame vsebino in izpostavi predvsem glavne rezultate ali zaključke,

ključne besede so tudi namenjene iskanju vsebin med množico člankov,

uvodno poglavje uvede bralca v tematiko besedila, razloži kaj je namen besedila, predstavi področje o katerem besedilo piše (če temu ni namenjeno v celoti posebno poglavje) ter na kratko predstavi strukturo celotnega besedila,

poglavja tvorijo zaokrožene celote, ki se po potrebi še nadalje členijo na podpoglavja, namenjena so recimo opisu orodij, ki smo jih uporabili pri delu, teoretičnim rezultatom ali predstavitvi rezultatov, ki smo jih dosegli,

zaključek še enkrat izpostavi glavne rezultate ali ugotovitve, jih primerja z dosedanjimi in morebiti poda tudi ideje za nadaljne delo,

literatura je seznam vseh virov, na katere smo se pri svojem delu opirali, oziroma smo se na njih sklicevali v svojem besedilu.

Strokovna besedila običajno pišemo v prvi osebi množine, v nevtralnem in umirjenem tonu. Uporaba sopomenk ni zaželjena, saj želimo zaradi lažjega razumevanja za iste pojme vseskozi uporabljati iste besede. Najpomenbnejše ugotovitve je smiselno večkrat zapisati, na primer v povzetku, uvodu, glavnem delu in zaključku. Vse trditve naj bi temeljile bodisi na lastnih ugotovitvah (izpeljavah, preizkusih, testiranjih) ali pa z navajanjem ustreznih virov.

Največ se lahko naučimo s skrbnim branjem dobrih zgledov takih besedil.

Poglavje 9

Pogoste napake pri pisanju v slovenščini

V slovenščini moramo paziti pri uporabi pridevnikov, ki se ne sklanjajo kot so npr. kratice. Pravilno pišemo model CAD in **ne** CAD model!

Pri sklanjanju tujih imen ne uporabljamo vezajev, pravilno je Applov operacijski sistem in **ne** Apple-ov.

Pika, klicaj in vprašaj so levostični: pred njimi ni presledka, za njimi pa. Klicajev in vprašajev se v strokovnih besedilih načeloma izogibamo. Oklepaji so desnostični in zaklepaji levostični (takole).

V slovenščini pišemo narekovaje drugače kot v angleščini! Običajno uporabljamo dvojne spodnje-zgornje narekovaje: „slovenski narekovaji“. Za slovenske narekovaje je v tej LaTeXovi predlogi definiran nov ukaz `\sn{ ... }`.

Vezaj je levo in desno stičen: `slovensko-angleški slovar` in ga pišemo z enim pomišljajem.

V slovenščini je pred in po pomišljaju presledek, ki ga v LaTeXu pišemo z dvema pomišljajema: `Pozor -- hud pes!` V angleščini pa je za razliko pomišljaj levo in desno stičen in se v LaTeXu piše s tremi pomišljaji: `---`. S stičnim pomišljajem pa lahko nadomeščamo predlog od ... do, denimo pri navajanju strani, npr. preberite strani 7–11 (`7--11`).

„Pred ki, ko, ker, da, če vejica skače“. To osnovnošolsko pravilo smo v

življenju po potrebi uporabljali, dopolnili, morda celo pozabili. Pravilo sicer drži, ampak samo če je izpolnjenih kar nekaj pogojev (npr. da so ti vezniki samostojni, enobesedni, ne gre za vrivek itd.). Povedki so med seboj ločeni z vejicami, razen če so zvezani z in, pa, ter, ne–ne, niti–niti, ali, bodisi, oziroma. Sicer pa je bolje pisati kratke stavke kot pretirano dolge.

V računalništvu se stalno pojavljajo novi pojmi in nove besede, za katere pogosto še ne obstajajo uveljavljeni slovenski izrazi. Kadar smo v dvomih, kateri slovenski izraz je primeren, si lahko pomagamo z Računalniškim slovarčkom [**?**].

Poglavje 10

Koristni nasveti pri pisanju v LaTeXu

Programski paket LaTeX je bil prvotno predstavljen v priročniku [**?**] in je v resnici nadgradnja sistema TeX avtorja Donalda Knutha [**?**], znanega po svojih knjigah o umetnosti programiranja, ter Knuth-Bendixovem algoritmu [**?**].

Različnih implementacij LaTeXa je cela vrsta. Za OS X priporočamo TeXShop, za Windows PC pa MikTeX. Spletna verzija, ki poenostavi sodelovanje pri pisanju, je npr. ShareLaTeX.

Včasih smo si pri pisanju v LaTeXu pomagali predvsem s tiskanimi priročniki, danes pa je enostavneje in hitreje, da ob vsakem problemu za pomoč enostavno povprašamo Google, saj je na spletu cela vrsta forumov za pomoč pri TeXiranju.

LaTeX včasih ne zna deliti slovenskih besed, ki vsebujejo črke s strešicami. Če taka beseda štrli preko desnega roba, LaTeXu pokažemo, kje lahko tako besedo deli, takole: `ra\-ču\-nal\-ni\-štvo`. Katere vrstice štrlijo preko desnega roba, se lahko prepričamo tako, da dokument prevedemo s vključeno opcijo `draft`: `\documentclass[a4paper, 12pt, draft]{book}`.

Predlagamo, da v izvornem besedilu začenjate vsak stavek v novi vrstici, saj LaTeX sam razporeja besede po vrsticah postavljenega besedila. Bo pa zato iskanje po izvornem besedilu in popravljanje veliko hitrejše. Večina

sistemov za TEXiranje sicer omogoča s klikanjem enostavno prestopanje iz prevedenega besedila na ustrezno mesto v izvornem besedilu in obratno.

Boljšo preglednost dosežemo, tako kot pri pisanju programske kode, tudi z izpuščanjem praznih vrstic za boljšo preglednost strukture izvornega besedila.

S pomočjo okolja `\begin{comment} ... \end{comment}` lahko hkrati zakomentiramo več vrstic izvornega besedila.

Pri spreminjanju in dodajanju izvornega besedila je najbolje pogosto prevajati, da se sproti prepričamo, če so naši nameni izpolnjeni pravilno.

Kadar besedilo, ki je že bilo napisano z nekim vizualnim urejevalnikom (npr. z Wordom), želimo prenesti v LATEX, je tudi najbolje to delati postopoma s posameznimi bloki besedila, tako da lahko morebitne napake hitro identificiramo in odpravimo. Za prevajanje Wordovih datotek v LATEX sicer obstajajo prevajalniki, ki pa običajno ne generirajo tako čisto logično strukturo besedila, kot jo LATEX omogoča. Hiter in enostaven način prevedbe besedila, ki zahteva sicer ročne dopolnitve, poteka tako, da besedilo urejeno z vizualnim urejevalnikom najprej shranimo v formatu pdf, nato pa to besedilo uvozimo v urejevalnik, kjer urejamo izvorno besedilo v formatu LATEX.

10.1 Pisave v LATEXu

V LATEXovem okolju lahko načeloma uporabljamo poljubne pisave. Izbira poljubne pisave pa ni tako enostavna kot v vizualnih urejevalnikih besedil. Posamezne oblikovno medseboj usklajene pisave so običajno združene v družine pisav. V LATEXu se privzeta družina pisav imenuje Computer Modern, kjer so poleg navadnih črk (roman v LATEXu) na voljo tudi kurzivne črke (*italic* v LATEXu), krepke (**bold** v LATEXu), kapitelke (SMALL CAPS v LATEXu), linearne črke (sans serif v LATEXu) in druge pisave. V istem dokumentu zaradi skladnega izleda uporabljamo običajno le pisave ene družine.

Ko začenjamo uporabljati LATEX, je zato najbolj smiselno uporabljati kar privzete pisave, s katerimi je napisan tudi ta dokument. Z ustreznimi ukazi

lahko nato preklapljamo med navadnimi, kurzivnimi, krepkimi in drugimi pisavami. Zelo enostavna je tudi izbira velikosti črk. LaTeX odlično podpira večjezičnost, tudi v sklopu istega dokumenta, saj obstajajo pisave za praktično vse jezike, tudi take, ki ne uporabljajo latinskih črk.

Za prikaz programske kode se pogosto uporablja pisava, kjer imajo vse črke enako širino, kot so črke na mehanskem pisalnem stroju (`typewriter` v LaTeXu).

Najbolj priročno okolje za pisanje kratkih izsekov programske kode je okolje `verbatim`, saj ta ohranja vizualno organizacijo izvornega besedila in ima privzeto pisavo pisalnega stroja.

```
for (i = 0; i < 100; i++)
   for (j = i; j < 10; j++)
      some_function(i, j);
```

Poglavje 11

Kaj pa literatura?

Kot smo omenili že v uvodu, je pravi način za citiranje literature uporaba BibTeXa [?]. BibTeX zagotovi, da nobene obvezne informacije pri določeni vrsti literature ne izpustimo in da vse informacije o določeni vrsti vira dosledno navajamo na enak način.

Osnovna ideja BibTeXa je, da vse informacije o literaturi zapisujemo v posebno datoteko, v našem primeru je to `literaturbbbbba.bib`. Vsakemu viru v tej datoteki določimo simbolično ime. V našem primeru je v tej datoteki nekaj najbolj značilnih zvrsti literature, kot so knjige [?], članki v revijah [?] in zbornikih konferenc [?], spletni viri [?, ?, ?], tehnično poročilo [?], diplome [?] itd. Diploma [?] iz leta 1990 je bila prva diploma na Fakulteti za elektrotehniko in računalništvo, ki je bila oblikovana z LaTeXom!

Po vsaki spremembi pri sklicu na literaturo moramo najprej prevesti izvorno besedilo s prevajalnikom LaTeX, nato s prevajalnikom BibTeX, ki ustvari datoteko `vzorec_dip_Seminar.bbl`, in nato še dvakrat s prevajalnikom LaTeX.

Kako natančno se spisek literature nato izpiše (ali po vrstnem redu sklicevanja, ali po abecedi priimkov prvih avtorjev, ali se imena avtorjev pišejo pred priimki itd.) je odvisno od stilske datoteke. V diplomi bomo uporabili osnovno stilsko datoteko `plain`, ki vire razporedi po abecedi. Zato je potrebno pri določenih zvrsteh literature, ki nima avtorjev, dodati polje `key`,

ki določi vrstni red vira po abecedi.

Z uporabo BIBTEXa v slovenščini je še nekaj nedoslednosti, saj so pomožne besede, ki jih BIBTEX sam doda, kot so *editor*, *pages* in besedica *and* pred zadnjim avtorjem, če ima vir več avtorjev [**?**], zapisane v angleščini, čeprav smo izbrali opcijo `slovene` pri paketu `babel`. To nedoslednost je možno popraviti z ročnim urejanjem datoteke `vzorec_dip_Seminar.bbl`, kar pa je smiselno šele potem, ko bibliografije v datoteki `literatura.bib` ne bomo več spreminjali, oziroma ne bomo več dodajali novih sklicev na literaturo v izvornem besedilu. Vsebino datoteke `vzorec_dip_Seminar.bbl` lahko na koncu urejanja tudi vključimo kar v izvorno besedilo diplome, tako da je vso besedilo, vključno z literaturo, zajeto le v eni datoteki.

Ko začenjamo uporabljati BIBTEX je lažje, če za urejanje datoteke .bib uporabljamo kar isti urejevalnik kot za urejanje datotek .tex, čeprav obstajajo tudi posebni urejevalniki oziroma programi za delo z BIBTEXom.

Le če se bomo na določen vir v besedilu tudi sklicevali, se bo pojavil tudi v spisku literature. Tako je avtomatično zagotovljeno, da se na vsak vir v seznamu literature tudi sklicujemo v besedilu diplome. V datoteki `.bib` imamo sicer lahko veliko več virov za literaturo, kot jih bomo uporabili v diplomi.

Vire v formatu BIBTEX lahko enostavno poiščemo in prekopiramo iz akademskih spletnih portalov za iskanje znanstvene literature v našo datoteko `.bib`, na primer v Google učenjaku. Izvoz v Google učenjaku še dodatno poenostavimo, če v nastavitvah izberemo BIBTEX kot želeni format za izvoz navedb. Navedbe, ki jih na tak način prekopiramo, pa moramo pred uporabo vseeno preveriti, saj so taki navedki običajno generirani povsem avtomatično.

Pri sklicevanju na literaturo na koncu stavka moramo paziti, da je pika po ukazu `\cite{ }`. Da LATEX ne bi delil vrstico ravno tako, da bi sklic na literaturo v oglatih oklepajih začel novo vrstico, lahko pred sklicem na literaturo dodamo nedeljiv presledek: `~\cite{ }`.

11.1 Izbiranje virov za spisek literature

Dandanes se skoraj vsi pri iskanju informacij vedno najprej lotimo iskanja preko svetovnega spleta. Rezultati takega iskanja pa so pogosto spletne strani, ki danes obstajajo, jutri pa jih morda ne bo več, ali pa vsaj ne v taki obliki, kot smo jo prebrali. Smisel navajanja literature pa je, da tudi po dolgih letih nekdo, ki bo bral vašo diplomo, lahko poišče vire, ki jih navajate v diplomi. Taki viri pa so predvsem članki v znanstvenih revijah, ki se arhivirajo v knjižnicah, založniki teh revij pa večinoma omogočajo tudi elektronski dostop do arhiva vseh njihovih člankov.

Znanstveni rezultati, ki so objavljeni v obliki recenziranih člankov, bodisi v konferenčnih zbornikih, še bolje pa v znanstvenih revijah, so veliko bolj izčiščen in zanesljiv vir informacij, saj so taki članki šli skozi recenzijski postopek. Zato na svetovnem spletu začenjamo iskati vire za strokovna besedila predvsem preko akademskih spletnih portalov, kot so npr. Google učenjak, Research Gate ali Academia, saj so na teh portalih rezultati iskanja le akademske publikacije. Če je za dostop do nekega članka potrebno plačati, se obrnemo za pomoč in dodatne informacije na našo knjižnico.

Če res ne gre drugače, pa je pomembno, da pri sklicevanju na spletni vir, vedno navedemo tudi datum, kdaj smo dostopali do tega vira.

Poglavje 12

Sistem STUDIS in PDF/A

Elektronsko verzijo diplome moramo oddati preko sistema STUDIS v formatu PDF/A [**?**]. Natančneje v formatu PDF/A-1b.

LaTeX in omenjeni format imata še nekaj težav s sobivanjem. Paket `pdfx.sty`, ki naj bi LaTeXu omogočal podporo formatu PDF/A ne deluje v skladu s pričakovanji. Ta predloga delno ustreza formatu, vsekakor dovolj, da jo študentski informacijski sistem sprejme. Znaten del rešitve je prispeval Damjan Cvetan.

V predlogi, poleg izvornega dokumenta `.tex` in vloženih slik `pic1.pdf` in `pic2.png`, potrebujemo še predlogo datoteke z metapodatki `pdfa-1b.xmp` in datoteko z barvnim profilom `sRGBIEC1966-2.1.icm`.

Poglavje 13

Sklepne ugotovitve

Uporaba LaTeXa in BibTeXa je v okviru Diplomskega seminarja **obvezna**! Izbira LaTeX ali ne LaTeX pri pisanju dejanske diplomske naloge pa je prepuščena dogovoru med vami in vašim mentorjem.

Res je, da so prvi koraki v LaTeXu težavni. Ta dokument naj vam služi kot začetna opora pri hoji. Pri kakršnihkoli nadaljnih vprašanjih ali napakah pa svetujem uporabo Googla, saj je spletnih strani za pomoč pri odpravljanju težav pri uporabi LaTeXa ogromno.

Literatura

[1] Evan You (ured.). Vue how changes are tracked. Dosegljivo: https://vuejs.org/v2/guide/reactivity.html#How-Changes-Are-Tracked. [Dostopano: 31. 12. 2020].

[2] Evan You (ured.). Vue reactivity in depth. Dosegljivo: https://v3.vuejs.org/guide/reactivity.html#what-is-reactivity. [Dostopano: 31. 12. 2020].