Univerza v Ljubljani
Fakulteta za računalništvo in informatiko

Edi Medvešček

Časovno omejevanje dostopa do interneta

diplomsko delo

visokošolski strokovni študijski program prve stopnje računalništvo in informatika

Mentor: doc. dr. Peter Peer

Ljubljana, 2013

Besedilo je oblikovano z urejevalnikom besedil LaTeX.

Univerza *v Ljubljani*
Fakulteta *za računalništvo in informatiko*

Tržaška 25
1001 Ljubljana, Slovenija
telefon: 01 476 84 11
01 476 83 87
faks: 01 426 46 47
01 476 87 11
www.fri.uni-lj.si
e-mail: dekanat@fri.uni-lj.si

Št. naloge: 00348/2012
Datum: 05.09.2012

Univerza v Ljubljani, Fakulteta za računalništvo in informatiko izdaja naslednjo nalogo:

Kandidat: **EDI MEDVEŠČEK**

Naslov: **ČASOVNO OMEJEVANJE DOSTOPA DO INTERNETA**
TIME LIMITING OF ACCESS TO THE INTERNET

Vrsta naloge: Diplomsko delo visokošolskega strokovnega študija prve stopnje

Tematika naloge:

V diplomski nalogi preučite obstoječe rešitve za časovno omejevanje dostopa do interneta, zasnujte lastno rešitev, predstavite uporabljena orodja in tehnologije za razvoj ter implementirajte in testirajte rešitev.

Mentor:
doc. dr. Peter Peer

Dekan:
prof. dr. Nikolaj Zimic

Fakulteta za računalništvo in informatiko • UNIVERZA V LJUBLJANI • 2

Izjava o avtorstvu diplomskega dela

Spodaj podpisani Edi Medvešček, z vpisno številko **63020252**, sem avtor diplomskega dela z naslovom:

Časovno omejevanje dostopa do interneta

S svojim podpisom zagotavljam, da:

- sem diplomsko delo izdelal samostojno pod mentorstvom doc. dr. Petra Peera.
- so elektronska oblika diplomskega dela, naslov (slov., angl.), povzetek (slov., angl.) ter ključne besede (slov., angl.) identični s tiskano obliko diplomskega dela
- soglašam z javno objavo elektronske oblike diplomskega dela v zbirki "Dela FRI".

V Ljubljani, dne 15. januar 2013 Podpis avtorja:

Iskreno se zahvaljujem mentorju dr. Petru Peeru za vso podporo, nasvete in praktično pomoč. Za potrpežljivost in vzpodbudo se zahvaljujem staršem, ki so mi omogočili študij in me vsa leta podpirali. Prav tako se zahvaljujem sodelavcem na TŠČ Nova Gorica in vsem ostalim, ki so mi pomagali pri študiju.

Kazalo

Seznam kratic

API	Application Programming Interface (Aplikacijski programski vmesnik)
CIM	Common Information Model (Skupni informacijski model)
COM	Component Object Model (Vmesniški standard za programske komponente)
CVS	Concurrent Versions System (Sistem za preverjanje verzije programa)
DMI	Desktop Management Interface (Okvir za upravljanje in sledenje komponentam)
DMTF	Distributed Management Task Force (Organizacija za razvoj, vzdrževanje in promocijo standardov)
GPL	General Public License (Licenca za prosto programje)
GUI	Graphic user interface (Uporabniški grafični vmesnik)
IDE	Integrated Development Environment (Integrirano razvojno okolje)
JAR	Java Archive (Javanski paketi)

Java EE	Java Platform, Enterprise Edition (Poslovna različica Jave)
Java ME	Java Platform, Mobile Edition (Različica Jave za mini naprave, kot so mobiteli, pametni televizorji ipd.)
Java SE	Java Platform, Standard Edition (Standardna različica Jave za osebne računalnike)
JDK	Java Development Kit (Javansko razvojno okolje)
JNI	Java Native Interface (Vmesnik med Javo in gostiteljskim sistemom)
JVM	Java Virtual Machine (Javanski navidezni stroj)
LAN	Local Area Network (Lokalno omrežje)
MD5	Message–Digest Algorithm (Zgoščevalni algoritem)
NTP	Network Time Protocol (Protokol za pridobitev točnega časa)
OLE	Object Linking and Embedding (Tehnologija za vključevanje in povezovanje dokumentov in objektov)
OOP	Object Oriented Programming (Objektno orientirano programiranje)
PHP	Hypertext Preprocessor (Skriptni jezik za izdelavo spletnih strani)
SDK	Software Development Kit (Programsko razvojno okolje)

SHA-1	Secure Hash Alghorithm (Zgoščevalni algoritem)
SNMP	Simple Network Management Protocol (Protokol za upravljanje naprav v omrežjih)
SQL	Structured Querry Languague (Relacijski poizvedovalni jezik)
Visual Basic	Programski jezik Visual Basic
Visual C++	Programski jezik Visual C++
WBEM	Web–Based Enterprise Management (Skupek sistemov upravljalnih tehnologij, razvit za poenotenje upravljanja v porazdeljenih računalniških okoljih)
WMI	Windows Management Instrumentation (Orodja za upravljanje v okolju Windows)
WMIC	Windows Management Instrumentation Command–line (Orodja za upravljanje v okolju Windows s pomočjo ukazne vrstice)

Povzetek

Prvi del diplomske naloge predstavlja pregled in analizo že obstoječih rešitev časovnega omejevanja dostopa do interneta. V drugem delu smo razvili lastno aplikacijo časovnega omejevanja. V aplikacijo smo vgradili različne funkcionalnosti, ki uporabniku omogočajo različne možnosti omejevanja dostopa do interneta. Pri razvoju smo se osredotočili na varnost in zanesljivost, poseben poudarek pa smo dali na morebitno zaobhajanje varnostnega mehanizma. Aplikacijo smo zasnovali v programskem jeziku Java, grafični vmesnik pa smo razvili s pomočjo NetBeans IDE okolja. Aplikacija je preprosta za uporabo, tako da jo lahko uporablja tudi računalniško nevešča oseba. Aplikacija učinkovito preprečuje dostopanje do interneta, podpira tudi večuporabniški način.

Ključne besede: Starševski nadzor, Java, JAR, JDK, NetBeans

Abstract

The thesis deals with time limiting of access to the Internet. In the first part of the thesis we present and analyse already existing solutions of such restricted Internet usage, whereas in the second part, our own application for time limiting is developed. We built into the application various functionalities, allowing the user a number of possibilities for limiting the access to the Internet. We mainly focused on security and reliability, with special attention paid to the potential avoidance of the security system. Interface was designed in the Java program, whereas the graphic interface was developed using NetBeans IDE environment. The application is simple to use, therefore even those who are not so computer literate can use it. Application effectively limits access to the Internet and also supports multi–user mode.

Keywords:: Parental control, Java, JAR, JDK, NetBeans

Poglavje 1

Uvod

V današnji dobi moderne tehnologije sta računalnik in internet kljub številnim nevarnostim, ki jih lahko prinašata, uporabni in nepogrešljivi orodji tako za odrasle kot tudi za otroke in mladostnike. Današnji šolarji dobivajo številne domače naloge, za izpolnitev katerih se morajo doma usesti za računalnik in pobrskati po internetu, za starše pa je zelo nepraktično sedeti več ur ob otroku in gledati, kaj le-ta počne, medtem ko naj bi pisal domačo nalogo. Nekateri otroci so kljub skrbnemu nadzoru staršev tako iznajdljivi, da se nadzoru izmuznejo in počnejo stvari, ki niso povezane s šolskim delom. Na računalniku *visijo* tudi pozno ponoči; igrajo igre ali klepetajo s svojimi prijatelji. Torej, kaj je rešitev v takih primerih? Programska rešitev za starševski nadzor bo pripomogla k temu, da bodo otroci uporabljali internet varno, ob času, ki jim ga bodo namenili starši, in za naloge, o katerih bodo starši obveščeni.

Programska rešitev za starševski nadzor opravlja dve funkciji. Nadzoruje, koliko časa nekdo preživi ob računalniku; ob brskanju po internetu ali igranju igric. Obenem nadzira, kaj uporabniki počnejo pri uporabi računalnika. Starši, knjižnice, internetne kavarne, šole in uradi, povsod tam, kjer je pomembno, za kakšen namen uporabniki uporabljajo računalnik, imajo lahko velike koristi od uporabe tovrstne opreme na svojih računalnikih.

V večini tovrstnih programov lahko določimo, kako dolgo lahko uporabnik

uporablja računalnik in kdaj. Na primer, program lahko omogoči uporabo računalnika dve uri dnevno in obenem blokira ure, ko so starši še v službi, in nočne ure – takrat je uporaba računalnika onemogočena, tako da otrok ne more nenadzorovano in brez vedenja staršev uporabljati računalnika. Mnogi programi omogočajo, da nastavimo dnevno, tedensko ali mesečno omejitev. Časovni roki se lahko nastavijo različno za različne programe, tako lahko na primer omejimo uporabo računalniških igric, obenem pa ostane npr. uporaba Microsoft Worda neomejena.

Starši lahko izbirajo, kateri programi so dovoljeni za vsakega posameznega uporabnika. Ta funkcija omejuje posebne spletne strani, prepoveduje nalaganje in blokira programe, kot so npr. igre. Prav tako prepove oziroma omeji dostop do nadzorne plošče. Starši lahko otroku vnaprej določijo, katere strani lahko obišče in katere so zanj prepovedane. Tako mu lahko v času pisanja domačih nalog blokirajo igre, klepete, Facebook ipd.

Programi za starševski nadzor nam omogočajo omejevanje dostopa do določenih tipov internetnih strani. Tako lahko doma našim otrokom, oziroma v podjetjih uporabnikom računalnika, blokiramo dostop do neželenih vsebin. Uporabnikom računalnika lahko blokiramo možnost iskanja neželenih strani (npr. erotične vsebine). V podjetjih lahko preverimo, na katerih straneh uporabniki računalnikov izgubljajo čas med delom, in jim dostop do teh strani blokiramo. V podjetjih lahko blokiramo dostop do vsega, razen do strani, ki jih zaposleni potrebujejo za vsakdanje delo. Omejimo lahko tudi uporabo tako imenovanih programov za sporočanje, kot so IRC, MSN ipd. Torej, programi za starševski nadzor omogočajo pregled nad tem, do katerih vsebin imajo uporabniki računalnikov dostop, in omogočajo, da lahko dostop do določenih neželenih vsebin blokiramo.

Če povzamemo zgoraj navedeno, lahko rečemo, da pod pojem starševski nadzor spada tako časovno kot vsebinsko omejevanje dostopa do interneta. V diplomski nalogi smo se omejili na časovno omejevanje.

1.1 Cilj

V prvem delu naloge smo si zadali cilj izdelati pregled in analizo že obstoječih rešitev časovnega omejevanja. V drugem delu pa smo si zadali cilj izdelave aplikacije za časovno omejevanje dostopa do interneta. Pri delu smo upoštevali kriterije, ki jih TopTenREVIEWS [1] uporablja za oceno starševske nadzorne programske opreme:

- Nabor funkcij
  Gledamo na različne funkcije, ki so na voljo, pa tudi njihovo uporabnost. Najboljši izdelki ne le nadzorujejo dostop in čas, ampak tudi podpirajo omejitve glede dodatne opreme, kot so programi, igre, klepet in prenosi ter zagotavljajo poročanje.

- Enostavnost uporabe
  Pomembna je čim bolj enostavna uporaba.

- Preprosta namestitev
  Najboljši programi se hitro nalagajo in nimajo težav med procesom prenosa ali namestitve.

- Časovi nadzor učinkovitosti
  Starševski časovni nadzor mora omogočati časovni nadzor nad uporabo računalnika, interneta in ostalih aplikacij. Omenjene omejitve določi uporabnik, v našem primeru starš.

- Tehnična podpora, pomoč
  Pomembno je, ali je proizvod opremljen s priročniki za uporabo in ali obstajajo kontaktni podatki (e–poštni naslov, telefon) ali zapiski (spletna dokumentacija), kamor se lahko uporabniki obrnejo ob morebitnih težavah pri uporabi.

Poglavje 2

Pregled obstoječih rešitev

V tem poglavju bomo opisali nekaj najpogostejših tehnik časovnega omejevanja dostopa do interneta. Pri vsaki rešitvi bomo pogledali, kaj so njene specifične lastnosti, prednosti in slabosti.

2.1 Omejevanje v Windows 7

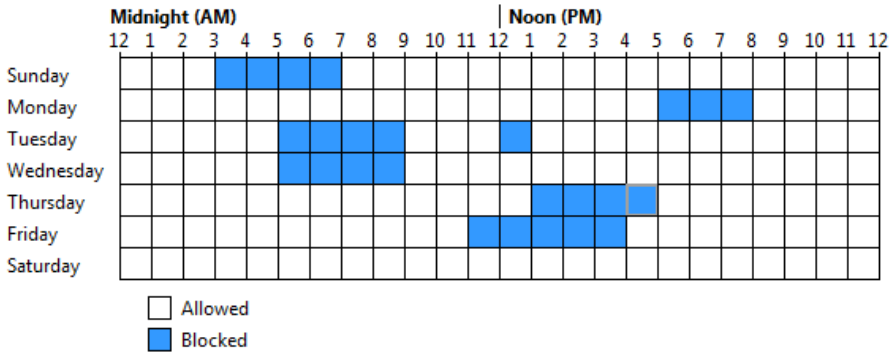
Ko vklopimo starševski nadzor za standardni uporabniški račun otroka, imamo na voljo naslednje nadzorne naloge:

- Časovne omejitve
  Nastavimo lahko časovne omejitve, s katerimi nadziramo, kdaj se lahko otroci prijavijo v računalnik. Z njimi preprečimo, da bi se otroci ob določenem času prijavili v računalnik. Za vsak dan v tednu lahko določimo različen čas prijave. Če so otroci ob poteku dodeljenega časa še vedno prijavljeni, jih računalnik samodejno odjavi.

- Igre
  Nadziramo lahko dostop do iger in vrsto vsebine, ki jo želimo blokirati, ter določimo, ali želimo določene igre dovoliti ali blokirati.

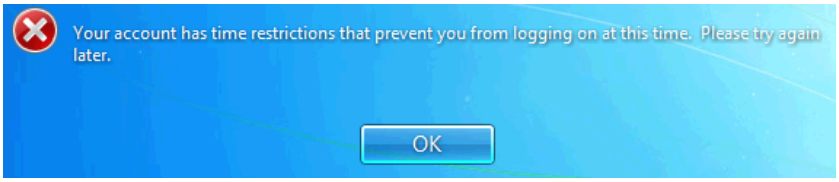
- Dovoljevanje ali blokiranje določenih programov
  Otrokom lahko preprečimo, da bi zaganjali neželene programe.

Za določanje dovoljenega in prepovedanega časa imamo tabelo, v kateri določimo dneve in ure, ko uporabnik lahko oziroma ne sme dostopati do računalnika (slika 2.1). Če želimo preprečiti dostopanje do interneta, imamo možnost, da uporabniku preprečimo dostop do poljubnega programa na primer Internet Explorerja.



Slika 2.1: Omejevanje s časovno tabelo v Windows 7

V kolikor poskuša uporabnik dostopati do računalnika izven dovoljenega časa, se mu na zaslonu prikaže obvestilo na sliki 2.2.



Slika 2.2: Obvestilo o omejitvi uporabe v Windows 7

Omejitve lahko določimo vsakemu uporabniku posebej. Nimamo pa podprte funkcije, kjer bi lahko poljubno določili dovoljen ali prepovedan čas uporabe, na primer dovoljena uporaba je dve uri in pol. Omejeni smo zgolj na celourne časovne intervale.

2.2 Omejevanje s pomočjo dodatkov za spletne brskalnike

Glede na to, da je brskalnik FireFox pri nas eden izmed najbolj priljubljenih spletnih brskalnikov, se osredotočimo nanj. Zanj je sicer na voljo nekaj dodatkov, ki jih lahko uporabimo za namene preprečevanja dostopa in filtriranja neprimernih vsebin v spletu. Pomanjkljivost takih dodatkov je, da so omejeni zgolj na delovanje v spletu. Zato nam ne bodo nič koristili v aplikacijah za neposredno sporočanje (MSN, Skype ipd.). Pomanjkljivost omenjenih dodatkov je tudi ta, da večinoma niso prevedeni v slovenski jezik. Za angleško govoreči del spleta pa so koristni.

2.2.1 ParentalControlBar

ParentalControlBar je bolj vsestranska rešitev, ki zajame tako uporabnike FireFoxa, kot tudi tiste, ki prisegajo na Internet Explorer. ParentalControlBar se namesti kot orodna vrstica. Ena večjih nerodnosti, ki smo jih opazili, je ta, da ne omogoča "tihega" preprečevanja dostopa, marveč se vedno, ko uporabnik naleti na onemogočeno spletno stran, pokaže okno, ki nas opozarja, da je bila stran blokirana, in sprašuje uporabnika, kaj storiti.

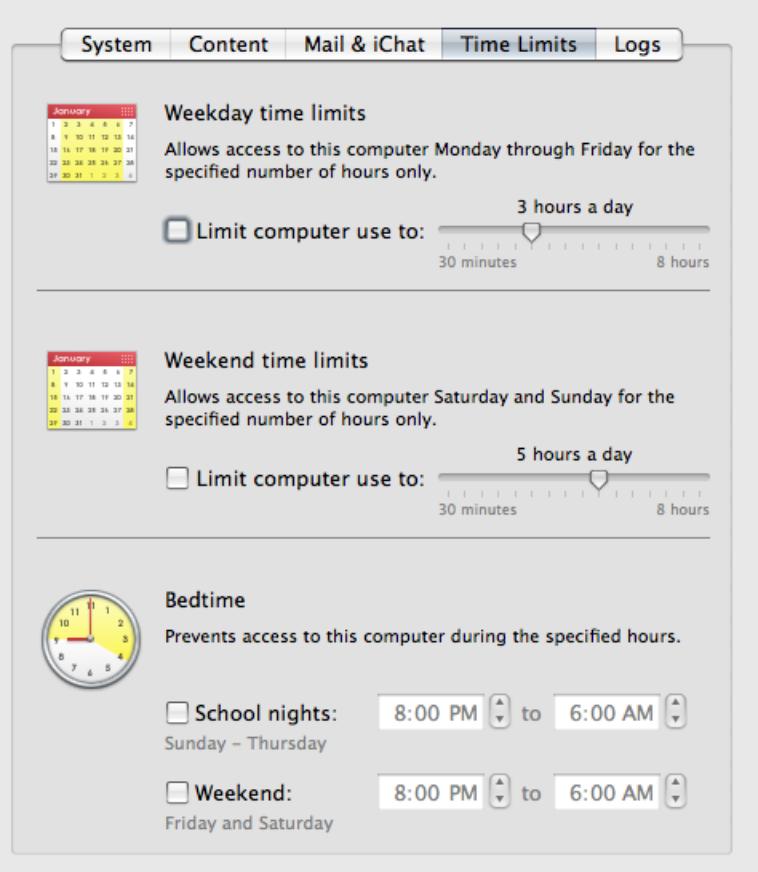
2.3 Omejevanje v Mac OS X

Starševski nadzor v Mac OS X je bil predstavljen v različici 10.5 (Leopard), kjer lahko nastavimo omejitve dejavnosti uporabnikom.

Če želimo nastaviti omejevanje (slika 2.3), moramo najprej uporabnikom kreirati uporabniške račune, za katere želimo nastaviti določene omejitve.

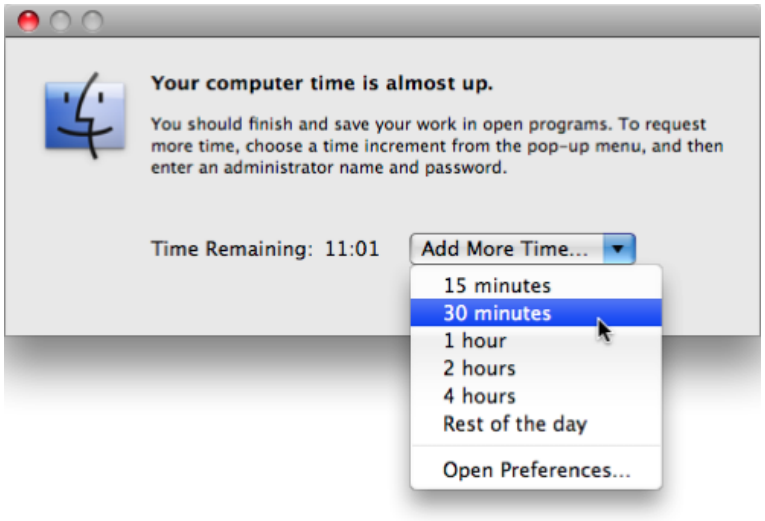
Nastavimo lahko omejevanje po dnevih, ki ima nastavitev, koliko ur na dan želimo, da uporabnik lahko uporablja računalnik. Na voljo imamo tudi dve možnosti omejevanja: omejevanje med tednom in omejevanje med vikendom.

Nastavimo lahko tudi čas, ko uporabnik ne sme uporabljati računalnika. Za vikende lahko nastavimo drugačna pravila kot veljajo med tednom, na primer uporabniku lahko damo med vikendom več časa.



Slika 2.3: Nastavitve časovnega omejevanja v Mac OS X

15 minut preden uporabniku poteče dovoljen čas uporabe računalnika, se na ekranu pojavi okno z možnostjo podaljšanja časa (slika 2.4). Seveda pa mora to podaljšanje odobriti administrator z vnosom svojega gesla in vnesenim podaljšanim časom. Če administrator tega ne stori, bo uporabnik avtomatsko odjavljen iz sistema.



Slika 2.4: Podaljšanje časa uporabe v Mac OS X

Starševski nadzor na Mac OS X ima poleg časovnega omejevanja še nekaj močnih orodij [2]:

- omejevanje vsebin,
- omejevanje aplikacij,
- nadzor nad dostopom do elektronske pošte in klepetalnic,
- nadzor nad dostopom do CD in DVD enot.

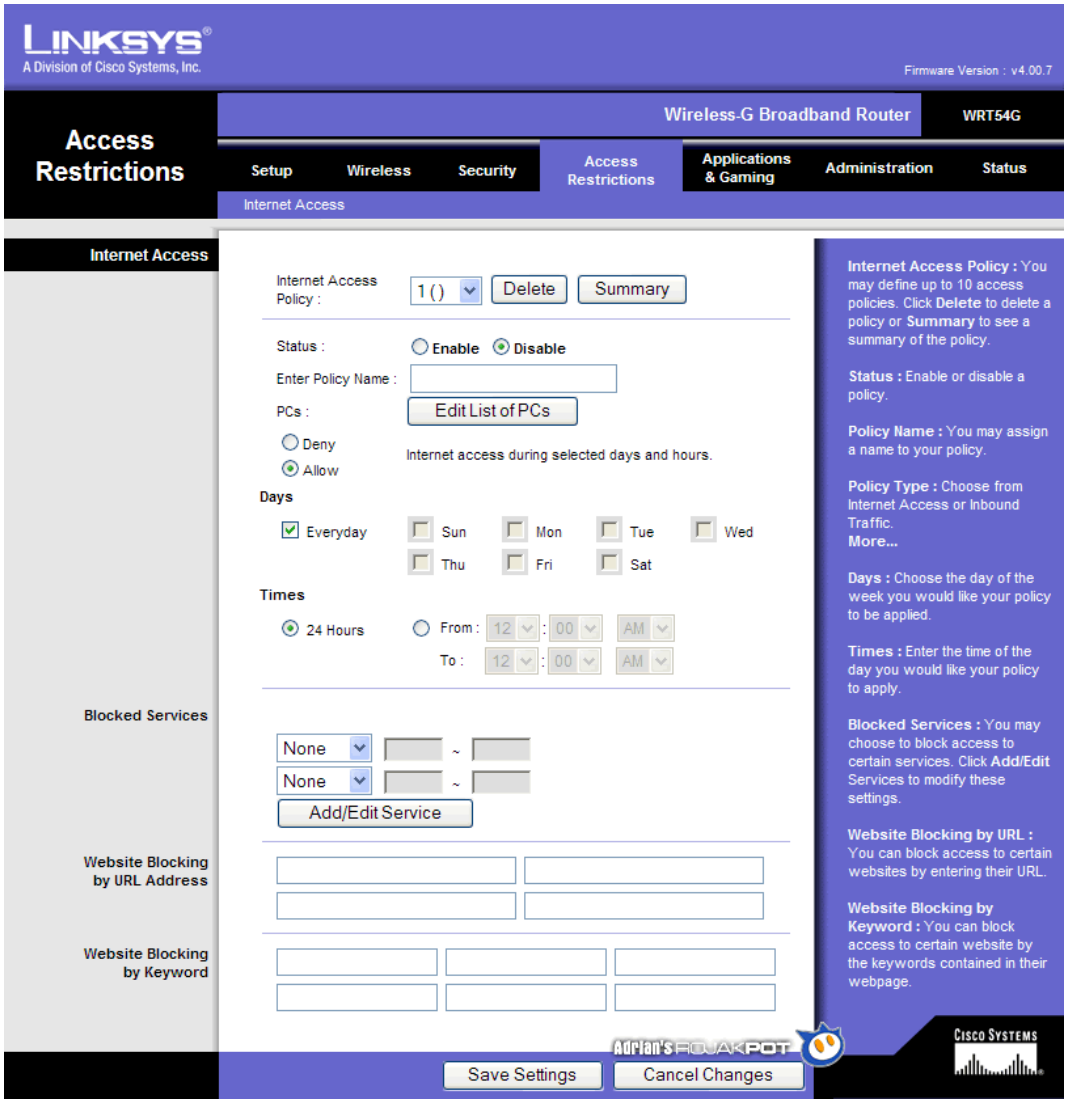
2.4 Omejevanje na usmerjevalniku

Pri omejevanju na usmerjevalniku smo se osredotočili na usmerjevalnik Linksys WRT54GL. Usmerjevalnik omogoča skromno časovno omejevanje (slika 2.5). Izberemo dneve v tednu, za katere nastavimo, od kdaj do kdaj lahko uporabnik dostopa oziroma ne sme dostopati do interneta. Za vsak dan v tednu bodo tako veljala enaka časovna pravila.

Usmerjevalnik omogoča tudi blokiranje vsebin s pomočjo URL naslova ali s pomočjo ključnih besed. Vnesemo lahko zgolj štiri naslove in zgolj šest ključnih besed, po katerih usmerjevalnik filtrira vsebino.

Prednost omejevanja na usmerjevalniku je, da lahko z enim pravilom določimo omejitve za celotno LAN skupino računalnikov. Prednost omejevanja na usmerjevalniku je tudi ta, da lahko blokiramo samo določena vrata.

Usmerjevalnik ne zna blokirati Proxy strežnikov, ActiveX komponent in Jave. Tudi ne omogoča kreiranja super uporabnika, s katerim bi bilo mogoče zaobiti omejitve.



Slika 2.5: Možnosti časovnega omejevanja na usmerjevalniku

2.5 Omejevanje s pomočjo Proxy strežnika

Proxy strežnik je strežniški program, ki ima vlogo posredovanja zahtev in odgovorov med internetom in računalniki v lokalnem omrežju. Torej, je nekakšen posrednik. Poleg tega, da usmerja zahteve, ima Proxy strežnik še nekaj pomembnih nalog:

- shranjevanje spletnih vsebin v medpomnilnik,
- skrivanje in varovanje krajevnega omrežja pred nezaželjenimi posegi z interneta,
- povezovanje lokalnih omrežij v internet,
- omejevanje dostopa,
- omejevanje storitev,
- omejevane pravic uporabnikom za dostop v internet,
- omejevanje dostopa do neprimernih spletnih vsebin,
- filtriranje paketov TCP/IP,
- skrivanje IP naslova odjemalca.

Proxy strežnik je program, ki odgovarja na prispele zahteve po dostopu do spletnih strani, ki jih pošiljajo uporabniki iz krajevnega omrežja. Uporabnikom odgovarja z vsebino, ki jo ima že shranjeno v svojem medpomnilniku, če pa vsebin še nima shranjenih, posreduje zahtevo do naslovljenega spletnega strežnika. Odgovor z interneta posreduje uporabniku, ki je poslal zahtevo po določeni spletni strani. Ob tem se spletna stran shrani v medpomnilnik na disk Proxy strežnika, kjer je na voljo za primer, če bi še kdo povpraševal po tej vsebini.

Na ta način se lahko precej zmanjša promet med lokalnim omrežjem in internetom, kar v končni fazi pomeni navidezno povečanje hitrosti dostopa do spletnih strani ali drugih vsebin na internetu.

Nekateri Proxy strežniki uporabljajo dinamično shranjevanje. Pri tem načinu Proxy preverja pogostost dostopa do določenih strani in bolj obiskane strani obnavlja sam, tudi ko ni zahtev uporabnikov. S tem so te strani bolj sveže.

Današnji Proxy strežniki poznajo nov protokol Cache Array Routing Protocol (CARP), ki omogoča distribuirano shranjevanje preko več Proxy strežnikov, povezanih v verigo (angl. chain) ali v polje (angl. array). Povezanost računalnikov lokalnega omrežja preko Proxy strežnika omogoča skrivanje računalnikov pred pogledi z interneta. To je ena bolj preprostih in učinkovitih metod varovanja lokalnih računalnikov pred vdori z interneta. Posamezni podatkovni paketki TCP/IP protokola, ki potujejo med računalniki, nosijo s seboj tudi naslov računalnika, s katerega prihajajo. Ker gredo ti paketki preko Proxy strežnika, tam prestopijo in privzamejo IP naslov strežnika. Nekdo, ki opazuje omrežje z interneta, vidi samo Proxy strežnik, ostalih računalnikov v lokalnem omrežju pa ne.

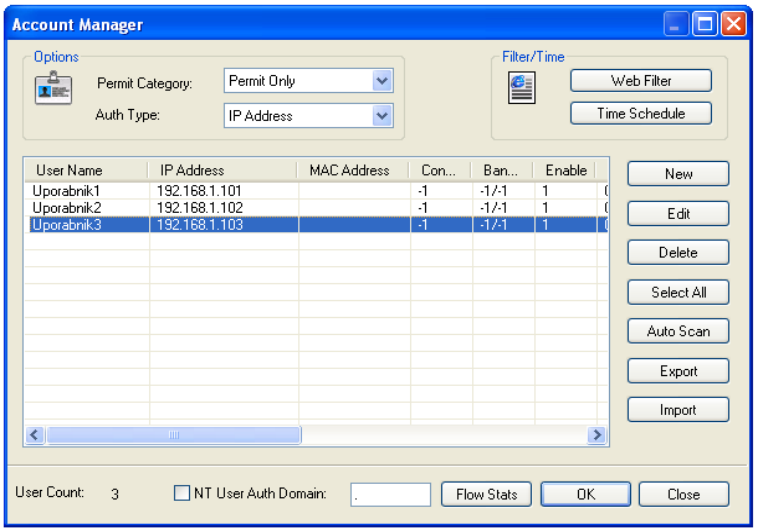
Ena od osnovnih nalog Proxy strežnika je tudi ta, da ga lahko uporabimo za povezovanje manjšega lokalnega omrežja v internet. V tem primeru deluje Proxy strežnik kot TCP/IP usmerjevalnik. Omejevanje servisov je zmožnost Proxy strežnika, da posreduje zahteve samo določenih internet storitev. Običajne storitve so WWW, FTP, E-mail, Telnet, DNS in drugi. Zaradi varnosti ali pa zaradi drugih kriterijev določamo, kaj sme skozi Proxy strežnik.

V krajevnih omrežjih imajo uporabniki pri dostopu do omrežnih virov omejitve, ki so posledica njihovih pravic, ki jim jih nastavi upravitelj omrežja. Tudi dostop do interneta je lahko reguliran, kar nam omogoči sam Proxy strežnik, ki ima možnost teh nastavitev. V sklopu uporabnikovih omejitev je tudi časovna omejitev, kar pomeni, da lahko določen uporabnik dostopa do interneta samo ob določenih urah. Nekateri boljši Proxy strežniki imajo tudi možnost omejevanja storitev. Primer: nek uporabnik lahko samo brska po spletnih straneh, drug uporabnik pa lahko uporablja vse internetne storitve. Večina sodobnih Proxy strežnikov omogoča tudi bolj ali manj uspešno ome-

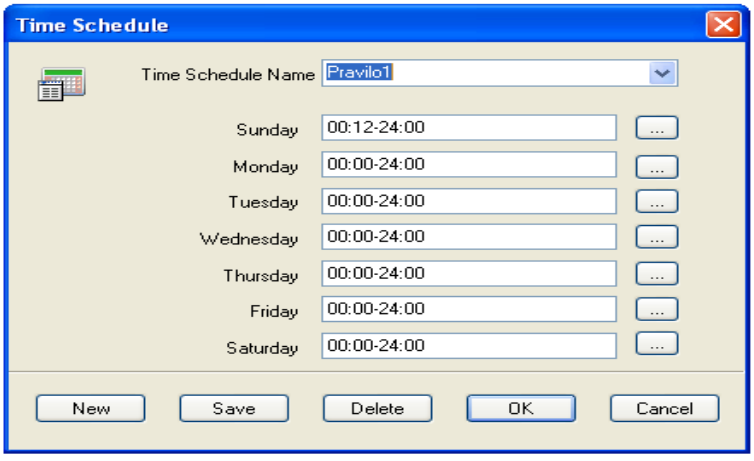
jevanje dostopa do določenih spletnih strani, do katerih bi si želeli, da uporabniki krajevnega omrežja nimajo dostopa. Običajno so to spletne strani, ki vsebinsko niso primerne. Naslove spletnih strežnikov ali kar celih domen (npr: www.sex.com ali porno.com) se da ročno vpisati v seznam internetnih naslovov, do katerih omejujemo dostop. Ker je teh naslovov preveč in bi bil ročni vnos nepraktičen, obstajajo posebni dodatni programi, ki imajo že spisek spletnih strani po kategorijah, katere lahko dovolimo ali ne. Filtriranje na nivoju TCP/IP paketov in vrat mrežnega protokola omogoča najvišjo stopnjo nadzora in varnosti, vendar zahteva precej dobro poznavanje samega TCP/IP protokola. Nadzor prometa preko Proxy strežnika se vrši s pomočjo log datotek. V te se zapisujejo razni podatki, ki govorijo o tem, kateri računalnik in/ali uporabnik iz omrežja hoče dostopati v internet, iz katerih naslovov v internetu bi rad dobil spletne vsebine, kdaj se je to zgodilo in druge podatke, ki pomagajo skrbniku mreže pri nadzoru. Proxy strežnik ima tudi mehanizme, ki sproti opozarjajo skrbnika mreže o morebitnih poizkusih vdora ali kakšni drugi nepravilnosti v delovanju strežnika. Opozorila dobi skrbnik prek elektronske pošte, telefonskega klica ali prek zvočnega alarma [3].

2.5.1 CCProxy

CCProxy [4] poleg standardnih opcij, ki jih Proxy strežniki omogočajo, omogoča tudi časovno omejevanje. Vstavimo lahko različne uporabnike, glej sliko 2.6, za vsakega posameznega uporabnika pa lahko določimo lastna pravila oziroma omejitve (slika 2.7).



Slika 2.6: Vstavljanje uporabnikov v CCProxy

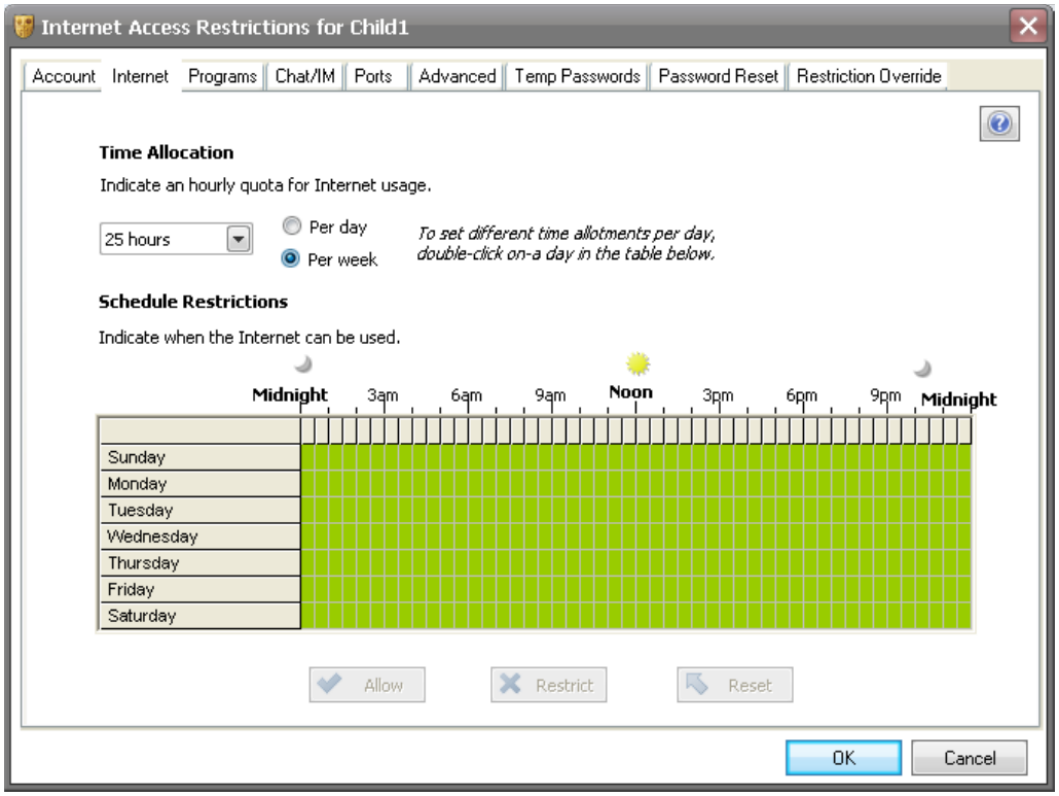


Slika 2.7: Časovne nastavitve na CCProxy

2.6 KidsWatch Time Management 6.5

Program časovnega omejevanja po testih kotira med bolje ocenjenimi. Uporabniku lahko omejimo čas dostopa do računalnika ali interneta na dan, teden

ali na časovno dolžino dneva. Vse omejitve lahko kombiniramo v različne scenarije. Npr.: uporabniku lahko prepovemo uporabo vsak dan od ponedeljka do petka, od 17.30 do 19.30, dovolimo mu 15-minutno igranje igric, 15 minut za internetno klepetanje in 90 minut za domačo nalogo. Z vnosom začasnega gesla lahko tudi podaljšamo čas uporabe, na primer, če je uporabnik ravno sredi svojega dela in potrebuje še nekaj časa (slika 2.8).



Slika 2.8: KidsWatch

Blokiramo lahko tudi določene internetne strani, iskanje po ključnih besedah in zagon spletnega brskalnika.

KidsWatch lahko tudi prepove dodatno nameščanje programske opreme, spreminjanje nastavitev na nadzorni plošči, kot so: uporabniški računi, datum in ura, omrežne nastavitve.

Lahko si nastavimo tudi sprotno beleženje dogodkov, kamor lahko beležimo:

obiskane spletne strani, vsebino pogovorov, uporabo računalnika in uporabo programov. Podprto je tudi pošiljanje obvestil na elektronsko pošto, v primeru, da bi uporabnik poskušal na primer obiskati prepovedano spletno stran ali pa bi med svojim delom uporabljal katero izmed ključnih besed iz črne liste [5].

Poglavje 3

Uporabljene tehnologije

V tem poglavju bomo opisali tehnologije in orodja, ki smo jih uporabljali za načrtovanje in razvoj aplikacije.

3.1 Razvojno okolje

Aplikacijo smo razvijali v program NetBeans, to je integrirano razvojno orodje, ki zagotavlja celovite rešitve za razvoj programske opreme. V tem poglavju bomo opisali:

- Javo,
- NetBeans IDE.

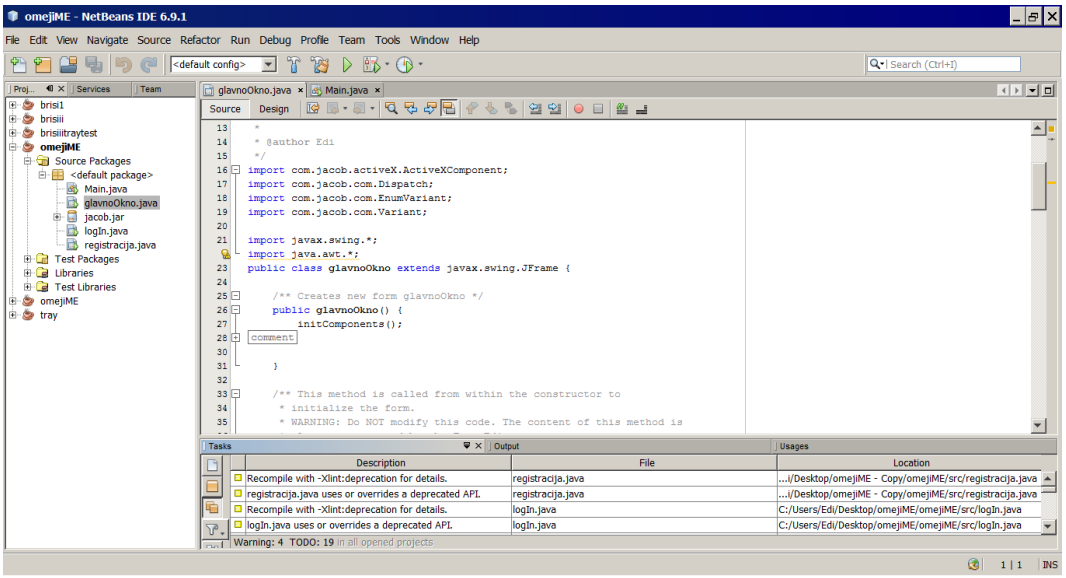
3.1.1 Java

Java je objektno usmerjen programski jezik. Sintaksa je podobna C in C++. Prvotno je bila razvita za izdelavo programske opreme v mrežnih napravah, kar omogoča arhitekturo z več gostitelji in varen prenos programskih komponent in prevedene kode. To omogoča, da Jave ni potrebno prilagajati na posamezne odjemalce, ampak je koda za vse odjemalce enaka (angl. write once, run anywhere).

JVM ali navidezni javanski stroj je temelj vseh javanskih okolij in je komponenta, ki je odgovorna za neodvisnost strojne opreme in neodvisnost med različnimi operacijskimi sistemi. JVM je navidezni računalniški stroj, ki vsebuje nabor ukazov in manipulira s spominom ob izvajanju aplikacij. JVM ne pozna programskega jezika Java, razume le prevedeno javansko kodo, ki jo imenujemo vmesna koda in vsebuje JVM navodila in ostale pomožne informacije. Kljub temu, da zaradi varnosti JVM določa striktne strukturne omejitve v vmesni kodi, se lahko vsak programski jezik, ki zadošča tem pogojem, izvaja na JVM. Tako je JVM uporabljen kot dostavno vozilo za ostale programske jezike, vseeno pa zagotavlja veliko robustnost, varnost in hitrost za razvoj [6].

3.1.2 NetBeans IDE

NetBeans IDE (slika 3.1) je odprtokodno integrirano razvojno okolje za razvoj aplikacij v javanskem okolju in C/C++. Na voljo je za operacijske sisteme Windows, Mac, Linux in Solaris. Nudi dobro podporo za razvoj poslovnih in mobilnih aplikacij, prav tako podpira PHP, JavaScript, Ajay, Groovy in Grails. Z razširitvami in s knjižnicami pa še dodatno povečamo podporo. Druge funkcije vključujejo podporo gradnji GUI in podporo CVS. Je zelo dobro dokumentirano in ima podporo skupnosti. Verzija 7.0.1 je prinesla izboljšano integracijo z Oracle Web Logic Server in GlassFish 3.1 ter novosti Maven 3 in podporo za urejanje HTML5. Napisano je v celoti v programskem jeziku Java.



Slika 3.1: Razvojno orodje NetBeans IDE

NetBeans IDE omogoča uporabnikom široko paleto funkcij za:

- izdelavo Java namiznih aplikacij – razvijanje konzolnih aplikacij in aplikacij z grafičnim uporabniškim vmesnikom,
- izdelavo Java EE in spletnih aplikacij – celotno zbirko orodij za razvoj Java EE aplikacij, izdelava spletnih aplikacij z uporabo Ajax, JavaScript in CSS tehnologij,
- vizualni mobilni razvoj – ustvarjanje, preizkušanje in razhroščevanje GUI aplikacij, ki tečejo na mobilnih telefonih in dlančnikih.

NetBeans IDE vsebuje različne elemente, ki jih potrebujemo za hiter in učinkovit razvoj aplikacij [7]:

- preoblikovanje kode (angl. code refactoring),
- razhroščevalnik,
- kontrolo izvirnih datotek in verzij,

- testiranje enot (angl. unit testing),
- dopolnjevanje kode,
- vključitev različnih podatkovnih baz,
- določanje vira za aplikacijski strežnik, tako da lahko med razvojem poženemo aplikacijo neposredno iz delovnega okolja.

NetBeans IDE je zelo zmogljivo orodje ravno zato, ker zanj obstaja množica dodatkov (angl. plugins) različnih razvijalcev, s katerimi si olajšamo delo.

3.2 Windows Management Instrumentation (WMI)

Windows Management Instrumentation je niz razširitev Windows Driver modela, ki je vključen v vmesnik operacijskega sistema, prek katerega komponente podajajo informacije in opozorila. WMI je Microsoftova implementacija na podlagi Web–Based Enterprise Management (WBEM) in Common Information Model (CIM) standardov iz Distributed Management Task Force (DMTF).

WMI podpira skriptne jezike, kot so VBScript ali Windows PowerShell. Z njimi je mogoče, tako na lokalni ravni kot na daljavo, upravljati osebne računalnike in strežnike z nameščenim operacijskim sistemom Microsoft Windows. WMI je nameščen v Windows 2000 in na novejše operacijske sisteme. Na voljo je kot dodaten prenos za starejše operacijske sisteme Windows.

Microsoft ponuja tudi vmesnik z ukazno vrstico WMI, imenovano Windows Management Instrumentation Command-line (WMIC).

WMI definira okolje neodvisnih specifikacij, ki omogočajo upravljanje podatkov, ki se uporabljajo za upravljanje aplikacij. WMI predpisuje standarde upravljanja in s tem povezanih tehnologij, ki delujejo z obstoječimi standardi upravljanja, kot so upravljanje namizja DMI in SNMP. WMI dopolnjuje tudi druge standarde, ki so vključeni v skupni enotni model. Ta model predstavlja

upravljalno okolje, s katerim se lahko upravlja podatke iz kateregakoli vira, ki so dostopni na enak način [8].

3.3 ActiveX

ActiveX je okvir, s pomočjo katerega je na neodvisen način definirana ponovna uporaba programskih komponent programske opreme v programskem jeziku. Za zagotavljanje različnih funkcionalnosti programskih aplikacij je lahko sestavljen iz ene ali več teh komponent.

Predstavil jo je Microsoft leta 1996 kot razvoj svojega Component Object Modela (COM) in Object Linking and Embedding (OLE) tehnologije, ki se običajno uporablja v operacijskem sistemu Windows, čeprav tehnologija sama po sebi ni vezana nanj.

Veliko Microsoftovih aplikacij, kot so na primer Internet Explorer, Microsoft Office, Microsoft Visual Studio in Windows Media Player, uporablja kontrolnike ActiveX za vzpostavljanje svojih naborov funkcij in tudi kot enkapsulacijo svojih funkcionalnosti. Kontrolniki ActiveX so lahko vgrajeni tudi v druge aplikacije. Internet Explorer podpira tudi kontrolnike ActiveX, ki so vgrajeni v spletne strani.

ActiveX Controls Mini Program Building Blocks služijo za ustvarjanje porazdeljenih aplikacij, ki delujejo prek interneta s pomočjo spletnih brskalnikov. Primeri tako prilagojene aplikacije so na primer aplikacija za ogled določene vrste datotek, aplikacija za prikaz animacije, aplikacija za delo s podatki.

ActiveX Controls je primerljiv z Java Applet. Programerji zasnujejo mehanizme, ki omogočajo spletnim brskalnikom prenos in izvajanje le-teh. Vendar pa lahko Java Applet deluje na skoraj katerikoli platformi, hkrati pa ActiveX Controls uradno deluje samo s spletnega brskalnika Microsoft Explorer in operacijskega sistema Microsoft Windows. Slabost AciveX Controls je, da omogočajo enostavno namestitev t. i. Malware programske opreme, torej računalniških virusov in vohunskih (angl. spy) programov, ki se lahko po ne-

sreči namestijo iz zlonamerne spletne strani, ki uporablja ActiveX Controls.

Pisanje ActiveX Controls je mogoče v kateremkoli jeziku, ki podpira razvoj COM komponente. Spodaj je naštetih nekaj programskih jezikov, ki podpirajo COM komponente [9].

- C + +
- Borland Delphi
- Visual Basic
- .NET Framework (C# / VB.NET)

3.4 JACOB – Java COM Bridge

JACOB je JAVA–COM Bridge, ki omogoča klice komponent avtomatizacije COM direktno iz javanskega okolja. Uporablja JNI za izvedbo native klicev na COM knjižnice. JACOB deluje v x86 in x64 okoljih, podpira 32-bitni in 64-bitni JVM. Izdan je pod GNU Library or Lesser General Public licenco (LGPL).

S pomočjo JACOB knjižice lahko razvijalci kličejo COM/ActiveX komponente, napisane v poljubnem programskem jeziku. Razvijalci lahko tudi kličejo javansko kodo iz programskih jezikov, ki podpirajo COM, kot so C#, Visual Basic ali Visual C++.

JACOB omogoča na platformah Windows dostop do COM/ActiveX komponent na lokalni ravni. Lahko pa se uporablja tudi iz ne-Windows platforme (npr. Unix, Linux, Mac ali mobilnih naprav) s pomočjo vključenega Remote Acess servisa.

COM/ActiveX komponente z vizualnimi vmesniki se lahko enostavno uporabljajo v Javi s pomočjo Swing ali SWT knjižnic.

Za tiste, ki potrebujejo napredni nadzor nad COM/ActiveX ali nad katero kompleksno COM/ActiveX komponento, JACOB vključuje tudi podporne kompleksne podatkovne tipe, kot so evidence, različice, polja, polja

z evidencami, večdimenzionalne nize, povratne klice do predmetov COM v Javi za ravnanje z dogodki [10].

3.5 Microsoft COM

Microsoft COM (Component Object Model) tehnologija v operacijskem sistemu Microsoft Windows omogoča komunikacijo komponentam programske opreme. COM uporabljajo razvijalci programske opreme za izdelavo ponovno uporabljivih komponent, povezljivih komponent in skupnih gradnikov aplikacij. COM objekte je mogoče izdelati z različnimi programskimi jeziki oziroma s programski mehanizmi, ki omogočajo implementacijo COM objektov. Družina tehnologij COM vključuje COM+, Distributed COM (DCOM) in ActiveX ® Controls.

Microsoft ponuja COM vmesnike za številne aplikacije Windows programskih vmesnikov, kot so: Direct Show, Media Foundation, Packaging API, Windows Animation Manager, Windows Portable Devices in Microsoft Active Directory (AD).

COM se uporablja v aplikacijah družine Microsoft izdelkov. Na primer COM OLE tehnologija omogoča dinamično povezovanje Word dokumentov do podatkov v Excelovih preglednicah. Avtomatizacija COM uporabnikom omogoča izgradnjo scenarijev za opravljanje ponavljajočih se nalog in nadzor medsebojnih relacij.

COM + je ime COM storitev in tehnologij, prvič predstavljeno v operacijskem sistemu Windows 2000. COM + je združil tehnologijo COM komponente in za vlogo gostitelja Microsoft Transaction Server (MTS). COM + samodejno upravlja programske naloge, kot so: združevanje virov nepovezanih aplikacij, objava dogodkov in vpisovanje porazdeljenih transakcij [11].

3.6 Zgostitveni algoritmi

Zgostitveni algoritmi (angl. hash algorithms, včasih tudi hash values, hash codes, hash sums, checksums, message digests ali fingerprints) poljubno dolg niz znakov preslikajo v število fiksne dolžine.

Izračunajo t. i. prstni odtis (angl. fingerprint) oz. kontrolno vsoto (angl. checksum) tega niza znakov, kar je osnova za digitalni podpis oziroma za zagotovilo, da so podatki ohranili integriteto.

Zgostitveni algoritmi morajo biti:

- enosmerni (iz kontrolne vsote ni mogoče nazaj izračunati originalnih podatkov),
- (v praksi) ne sme priti do kolizije (ne smeta obstajati dva različna niza podatkov, ki bi vrnila isto kontrolno vsoto; to je sicer odvisno od bitnosti izvlečka).

Dobri zgostitveni algoritmi imajo t. i. efekt plazu – če se vhod malenkost spremeni, se bo izhod drastično spremenil [12].

3.6.1 MD5

MD5 (Message–Digest Algorithm 5) je znan kot pogosto uporabljana kodirna funkcija s 128-bitnim izhodom. Po internetnem standardu (RFC 1321) je bil MD5 priznan in uporabljen v velikem številu aplikacij za izboljšanje varnosti. Pogosto se uporablja tudi za preverjanje datotek. MD5 si je zamislil Ronald Rivest leta 1991 z namenom, da bi zamenjal zgodnejšo funkcijo MD4. Leta 1996 so našli napako v zasnovi algoritma MD5. Kljub temu, da to ni bila velika napaka, so strokovnjaki za kodiranje začeli priporočati uporabo drugih varnostnih algoritmov, kot na primer SHA–1. Leta 2004 so odkrili dodatne pomanjkljivosti v algoritmu, nadaljnje odkrivanje napak pa je sledilo do leta 2007. Urad US–CERT je izjavil: “MD5 moramo obravnavati kot algoritemsko zlomljivega in kot neprimernega za nadaljnjo uporabo.”

Pomanjkljivost MD5 je, da ni odporen na kolizije, ki so sicer zelo redke, ampak nedopustne za pomembne posle. S pomočjo MD5 kolizij (katere je možno, v primeru, da obstajajo, odkriti v nekaj sekundah) se namreč lahko ustvari vmesne certifikate, katere MD5 zgoščevalna funkcija prepozna kot legitimne, čeravno gre za ponaredek. Pojav je znan pod pojmom napad kolizij (angl. collision attack). MD5 zato ni primeren v aplikacijah za SSL certifikate ali digitalne podpise na osnovi SSL [13].

3.7 Java Beans

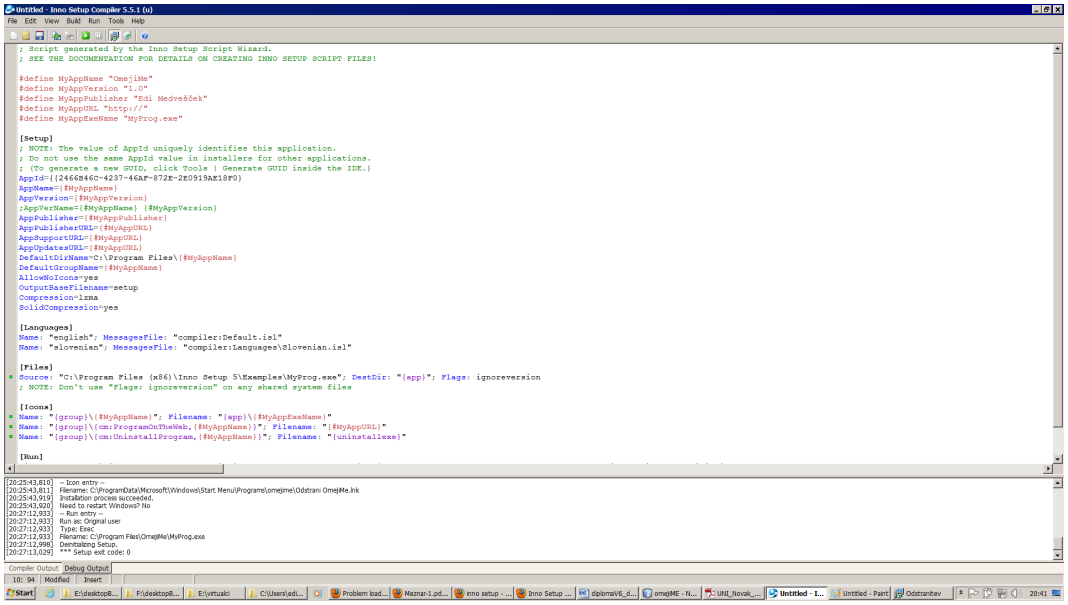
JavaBeans oziroma javanska zrna so ponovno uporabljive programske komponente v programskem jeziku Java. Razredi so napisani v programskem jeziku Java, ki ustrezajo posebnim dogovorom. Uporabljajo se pri enkapsulaciji več objektov v en objekt oziroma zrno. Tako jih lahko tretiramo kot eno javanski tip, ne pa kot več posameznih objektov. JavaBean je javanski objekt, ki je seriliziran, ima ničelni konstruktor in podpira dostop do lastnosti zrn preko getter in setter metod [14].

Prednosti zrn so:

- Imajo vse prednosti Javine "pisati write once, run anywhere" paradigme.
- Lastnosti, dogodke in metode zrn, ki so izpostavljene drugim aplikacijam, je mogoče nadzorovati.
- Lahko ponujajo pomožno programsko opremo, s katero lažje nastavimo zrno.
- Nastavitve zrna lahko shranimo v trajno hrambo, do katere lahko dostopamo v kasnejšem času.
- Lahko jih registriramo za prejemanje dogodkov iz drugih objektov, lahko pa tudi generiramo dogodke, ki se pošiljajo nanje.

3.8 Inno Setup

Za izdelavo namestitvenega in odstranitvenega programa smo uporabili orodje Inno Setup (slika 3.2).



Slika 3.2: Inno Setup okolje

Značilnosti [15]:

- Podpora za vse različice operacijskega sistema Windows do sedaj: 7, 2008 R2, Vista, XP, 2008, 2003, 2000, Me, 98, 95, and NT 4.0 (brez dodatnih servisnih paketov).
- Razširjena podpora za namestitev 64-bitnih aplikacij na 64-bitne različice operacijskega sistema Windows. Tako x64 kot Itanium arhitekture so podprte. (Na Windows Server 2003 x64, Itanium arhitektura, je potrebno predhodno namestiti Service Pack 1 ali novejšega.)
- Podpira vzpostavitev enega EXE namestitvenega programa za enostavno spletno distribucijo.

- Standardni Windows 2000/XP čarovniški vmesnik.
- Prilagodljive nastavitve vrste namestitve, npr. polno, minimalno ali po meri.
- Celotna odstranitvena zmogljivost.
- Vključuje vgrajeno podporo za redke bzip2 in 7–ZIP LZMA/LZMA2 kompresije.
- Ustvarjanje bližnjic kjerkoli, tudi v Start meniju in na namizju.
- Zapisovanje v register in INI datoteke.
- Zagon drugih programov pred, med ali po namestitvi.
- Podpora večjezični namestitvi.
- Podpora zaščitenim in kriptiranim namestitvam.
- Podpora digitalno podpisanemu nameščanju in odstranjevanju.
- Tiho nameščanje in odstranjevanje.
- Unicode namestitev (Windows 2000 ali novejši).
- Integrirani Pascal Script za napredno prilagajanje nameščanja in odstranjevanja.
- Dostopna celotna izvorna koda (Borland Delphi 2.0–5.0 in 2009).

3.9 Network Time Protocol – NTP

NTP protokol je med uporabniki internetnih storitev precej nepoznan. Dobro pa ga poznajo vsi vzdrževalci omrežne opreme, ponudniki internetnih storitev in znanstveniki. Točen čas je za obstoj interneta ključnega pomena, saj je v resnici skoraj vsak paket, ki potuje po medmrežju, časovno opredeljen. V primeru časovnega razhajanja med aktivno omrežno opremo bi

se kaj kmalu znašli v precejšnji zmedi in pospešenem razkroju medmrežja. NTP protokol se je začel razvijati v začetku osemdesetih let prejšnjega stoletja. Danes smo večinoma prešli na četrto generacijo NTP strežnika, torej NTPv4. NTP strežniki so hierarhično urejeni, tako imamo v Sloveniji nekaj javno dostopnih hierarhično visokih (angl. Stratum–2) strežnikov ter nekaj najnatančnejših (angl. Stratum–1) strežnikov, ki pa niso javno dostopni. Dostopni so ponudnikom internetnih storitev ter večjim akademskim organizacijam, ki posredujejo čas končnim uporabnikom [16].

3.10 SQLite

SQLite je relacijski sistem za upravljanje s podatkovnimi bazami. Za razliko od drugih sistemov za upravljanje s podatkovnimi bazami ni realiziran kot ločen proces, do katerega lahko dostopamo preko odjemalca, ampak je realiziran kot ena sama datoteka z integriranimi tabelami, indeksi, prožilci in pogledi. Baza je prenosljiva med različnimi platformami, lahko jo prosto prenašamo tudi med 32 in 64-bitnimi platformami. Vse te lastnosti so jo naredile priljubljeno pri aplikacijah, kjer je potrebno hraniti podatke v bazi. Koda SQLite je brezplačna tako za zasebno kot za javno rabo.

SQLite je kompaktna knjižnica. Velikost knjižnice je lahko manjša od 350 KB, odvisno od ciljne platforme in nastavitve optimizacije prevajalnika. Če onemogočimo nekatere možnosti, je lahko velikost knjižnice SQLite celo manjša od 300 KB. SQLite lahko teče tudi z minimalno količino pomnilnika 4 KB in zelo majhno kopico, zaradi česar je SQLite priljubljena izbira na pomnilniško omejenih napravah, kot so mobili telefoni, dlančniki in MP3 predvajalniki. SQLite na splošno deluje hitreje z več pomnilnika. Kljub temu pa deluje zadovoljivo tudi v okoljih z manjšo količino pomnilnika.

SQLite je zelo natančno preizkušena pred vsako izdajo in ima zelo dober sloves. Večina izvorne kode SQLite je namenjena izključno za testiranje in preverjanje. SQLite elegantno rokuje z napakami pri dodeljevanju pomnilnika in diskovnimi V/I napakami. Transakcije so ACID (atomicity,

consistency, isolation, and durability) obvladovane, tudi če se zruši sistem ali pride do izpada. Vse to se sproti preverja z avtomatskimi testi, ki simulirajo sistemske napake. Seveda tudi z vsem tem testiranjem še vedno obstajajo napake. Toda za razliko od nekaterih podobnih projektov (predvsem komercialnih konkurentov) je SQLite odprt. Določa tudi hroščovne sezname vključno s seznami kritičnih napak in kodnih sprememb.

Kodo SQLite baze razvija mednarodna ekipa razvijalcev, ki delajo na SQLite s polnim delovnim časom. Zmogljivost in zanesljivost SQLite se krepi, hkrati pa se ohranja povratna združljivost s specifičnim vmesnikom, SQL sintakso in obliko datoteke zbirke podatkov. Izvorna koda je popolnoma brezplačna. Na voljo je tudi strokovna pomoč [17].

Poglavje 4

Razvoj aplikacije

4.1 Analiza problema

Pri analizi problema smo zaznali naslednje zahteve:

- Nameščanje in odstranjevanje aplikacije mora biti enostavno (potrebujemo namestitveni in odstranitveni program s preprostim grafičnim vmesnikom).
- Delovanje mora biti samostojno.
- Uporabnik določa nastavitve delovanja programa.
- Aplikacija mora učinkovito blokirati dostop do interneta izven dovoljenega časa.
- Delovanje mora biti dovolj robustno, da uporabnik ne more na enostaven način obiti zaščite (aplikacija mora imeti nadzorne mehanizme, ki preprečujejo poizkuse uporabnika, da bi obšel zaščito).
- Potrebujemo nadzorni program, ki periodično preverja, ali se aplikacija izvaja.
- Aplikacija se avtomatsko zažene z zagonom operacijskega sistema Windows.

- Aplikacija mora delovati neprekinjeno in ne sme ovirati normalnega dela operacijskega sistema (da aplikacija teče, bo skrbel nadzorni program; poraba sistemskih sredstev obeh programov mora biti nizka).

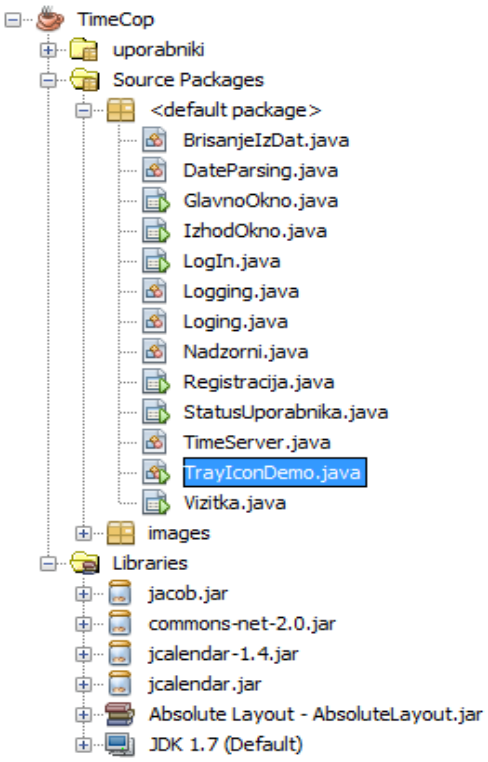
- Delovanje nadzornega programa mora biti skrito (nadzorni program nima grafičnega vmesnika).

- Za razvoj aplikacije smo si izbrali programski jezik Java, predvsem zato, ker nam je najbolje poznan, pa tudi ker je platformsko neodvisen.

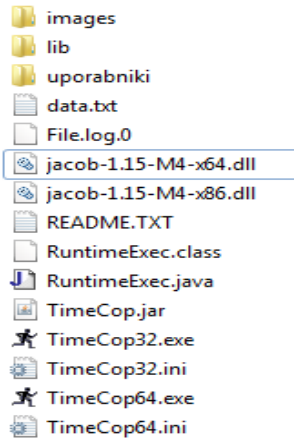
- Aplikacija mora podpirati več jezikov, slovenskega in angleškega.

4.2 Načrt arhitekturne rešitve

Programsko kodo smo razdelili na več javanskih razredov. S tem je programska koda postala preglednejša in tudi lažja za razumevanje. Na ta način smo si nekoliko olajšali delo, saj smo se tako hitreje znašli v programski kodi. Program smo razdelili na smiselne celote. Na sliki 4.1 so podani vsi javanski razredi, ki smo jih napisali. Slika 4.2 prikazuje vsebino mape naše aplikacije *TimeCop*.



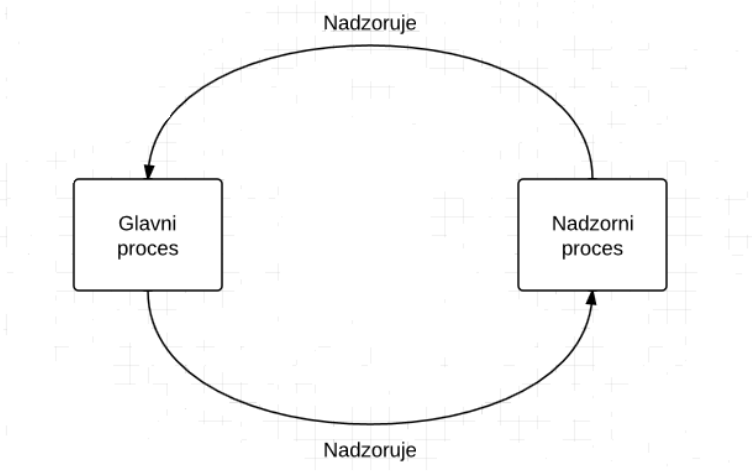
Slika 4.1: Javanski razredi našega izdelka *TimeCop*



Slika 4.2: Vsebina mape naše aplikacije *TimeCop*

4.3 Načrtovanje nadzornega programa

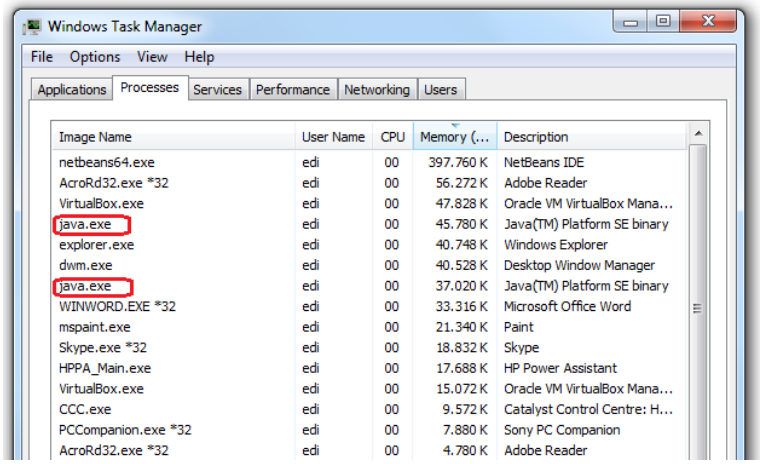
Namen nadzornega programa je stalno preverjanje, ali je glavni program v izvajanju (slika 4.3). V kolikor bi prišlo do zaustavitve glavnega programa zaradi namerne prekinitve s strani uporabnika ali pa do nezaželenega sesutja glavnega programa, mora nadzorni program ponovno zagnati glavni program. Prav tako bi ob morebitni zaustavitvi nadzornega programa glavni program poskrbel, da bi se nadzorni program ponovno zagnal. Nadzorni program se zažene skupaj z glavnim programom, njegovo delovanje je skrito, kar pomeni, da nima grafičnega vmesnika, ampak se izvaja v ozadju kot proces. Program s periodo 5 sekund preverja, ali se glavni program izvaja. Periodo 5 sekund smo izbrali, da ne bi po nepotrebnem bremenili sistema, hkrati pa je to še dovolj majhen čas, da se glavni program ponovno zažene, v kolikor bi le-ta bil onemogočen.



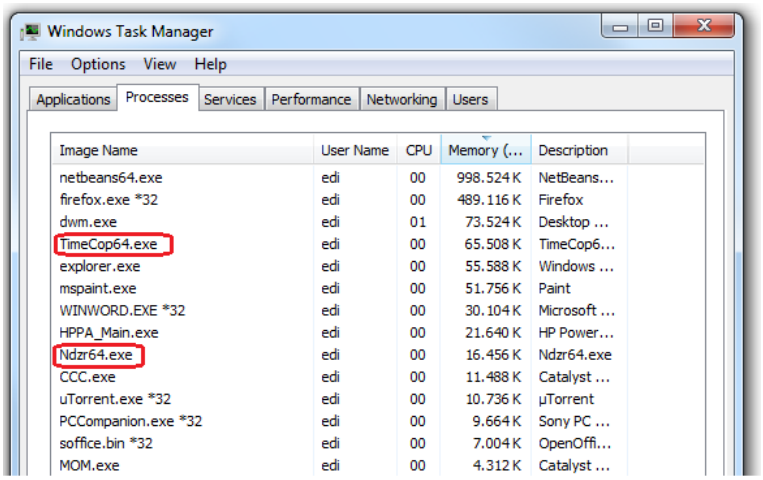
Slika 4.3: Nadzorovanje procesov

Javanski procesi (slika 4.4) se avtomatsko poimenujejo *java.exe* ali *javaw.exe*. Tukaj pa nastopi problem, ker želimo imeti dva procesa z različnimi imeni, tako da v kolikor en ni v teku, drugi proces takoj zažene prvega. Omenjeni problem smo rešili s t. i. java ovijanjem (angl. wrapping), ki nam je

proces *java.exe* preimenoval v *TimeCop.exe* (slika 4.5).



Slika 4.4: Ime javanskega procesa pred preimenovanjem



Slika 4.5: Preimenovanje javanskih procesov

4.4 Izdelava nadzornega programa

Koda 4.1 naredi poizvedbo vseh trenutnih procesov in jih zapiše v množico tipa set. Omenjeni javanski tip smo izbrali, ker število procesov v izvajanju

ni v naprej znano, zato nismo uporabili tabele, kateri moramo v naprej definirati velikost, ampak smo uporabili dinamično podatkovno zbirko set, ki se po potrebi avtomatsko povečuje. Zelo uporabna je tudi metoda *contains()*, s katero preverimo, ali se nek element nahaja v zbirki. Na ta način smo prihranili nekaj vrstic kode.

```
public static void zazeniTimerNadzorni() {
   Date startDate = new Date();
   startDate.geteDate();
   timer.scheduleAtFixedRate(new TimerTask() {
      public void run() {
         try {
            najdiProcese();
            if (!set1.contains("TimeCop"))
         } catch (Exception e) { //IOE
            e.printStackTrace();
         }
      }
   }, startDate, period);
}
```

Koda 4.1: Metoda za zagon časovnika

S pomočjo časovnika, ki se proži vsakih nekaj sekund, preverimo, ali je trenutno v izvajanju proces *TimeCop*, če tega procesa ni, pomeni, da leta ni zagnan in tako na novo poženemo omenjeni proces s klicom metode *zazeniProgram()*. Podana koda 4.2 skrbi za zagon aplikacije *TimeCop*.

```
public static void zazeniProgram(){
   try {
      Runtime runTime = Runtime.getRuntime();
      Process process = runTime.exec("TimeCop.exe");
   } catch (Exception e) {
      e.printStackTrace();
   }
```

```
}
```

Koda 4.2: Metoda za zagon aplikacije *TimeCop*

Zaradi specifičnosti Jave in JVM-ja le-ta ne omogoča izdelave novega javanskega procesa (angl. native fork) znotraj JVM-ja. Tako smo imeli veliko težavo, kako izdelati nov proces znotraj javanskega okolja. Problem smo rešili tako, da smo izdelali nov primerek JVM-ja in v novem JVM zagnali nov proces. Razred za kreiranje novega primerka JVM je prikazan v kodi 4.3.

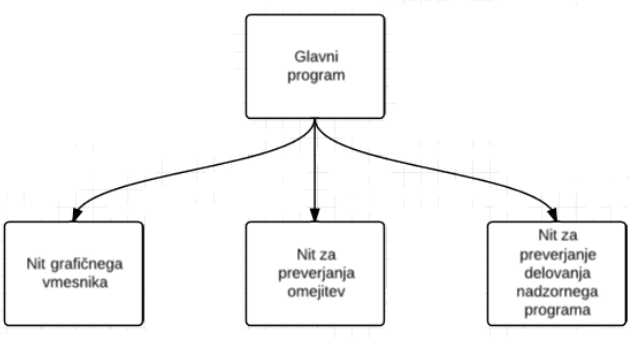
```
private JavaProcess1() {}
public static int exec(Class klass) throws IOException,
                                    InterruptedException {
   String javaHome = System.getProperty("java.home");
   String javaBin = javaHome +
         File.separator + "bin" +
         File.separator + "java";
   String classpath =
      System.getProperty("java.class.path");
   String className = klass.getCanonicalName();
   ProcessBuilder builder = new ProcessBuilder(
         javaBin, "-cp", classpath, className);
   Process process = builder.start();
   process.waitFor();
   return process.exitValue();
}
```

Koda 4.3: Izdelava novega primerka JVM

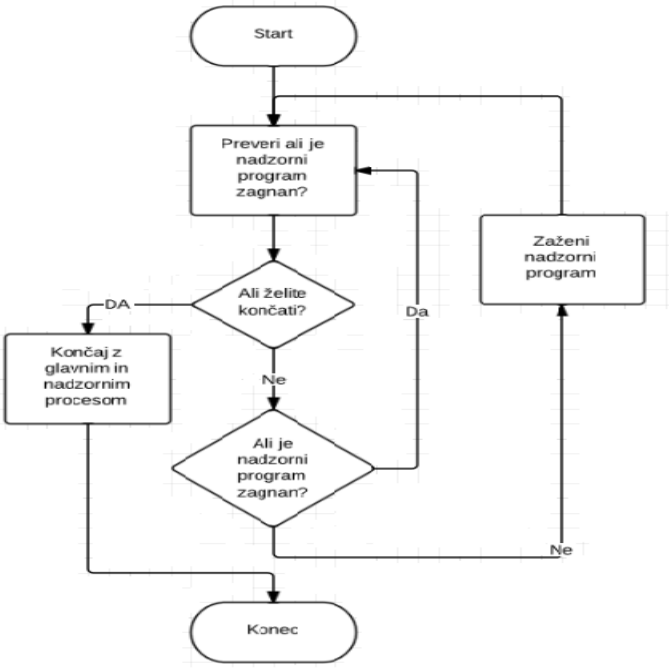
4.5 Načrtovanje glavnega programa

4.5.1 Načrtovanje niti in procesov

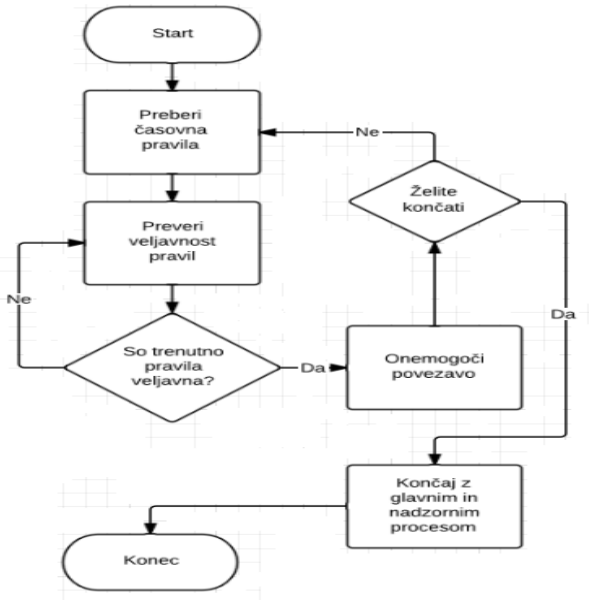
Glavni program se zažene skupaj z operacijskim sistemom. Ob njegovem zagonu se hkrati zažene tudi nadzorni program. Glavni program smo optimizirali, tako da smo glavne dele razdelili v različne niti, kot prikazuje slika 4.6. Na sliki 4.7 je prikazana nit preverjanja delovanja nadzornega programa. Slika 4.8 prikazuje nit preverjanja omejitev.



Slika 4.6: Niti glavnega programa



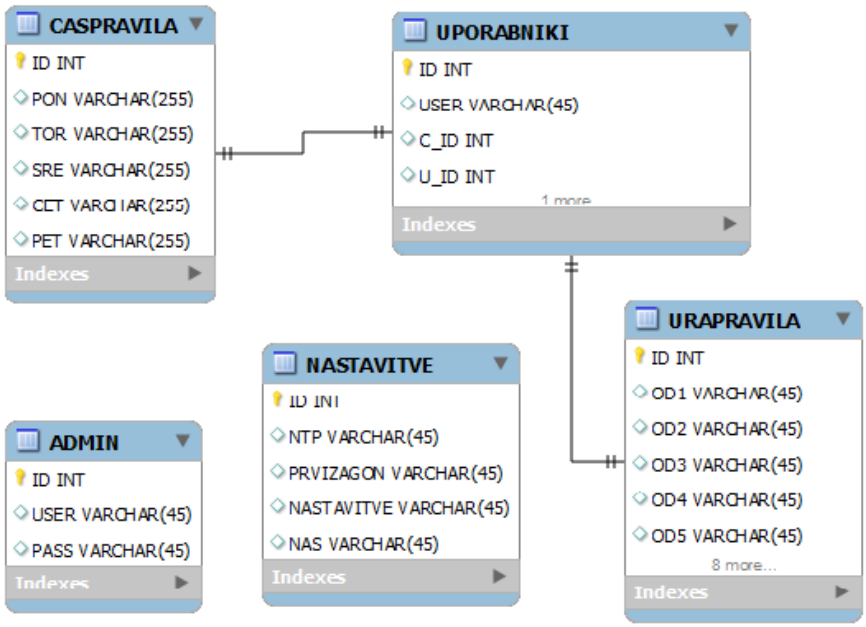
Slika 4.7: Nit za preverjanje delovanja nadzornega programa



Slika 4.8: Nit za preverjanje omejitev

4.5.2 Načrtovanje podatkovne baze

Za delovanje aplikacije je potrebno hraniti relativno veliko podatkov. Sprva smo podatke hranili v datotekah, kar pa se je pokazalo za dokaj nepraktično in predvsem precej težavno za rokovanje s podatki. Z razvojem aplikacije so se odpirale tudi vedno nove potrebe po hranjenju večje količine podatkov, tako smo se odločili, da preidemo na SQLite podatkovno bazo. S pomočjo omenjene baze smo predvsem olajšali rokovanje s podatki, izboljšale pa so se tudi performance aplikacije. Na sliki 4.9 je podan E-R diagram podatkovne baze.



Slika 4.9: E–R diagram baze

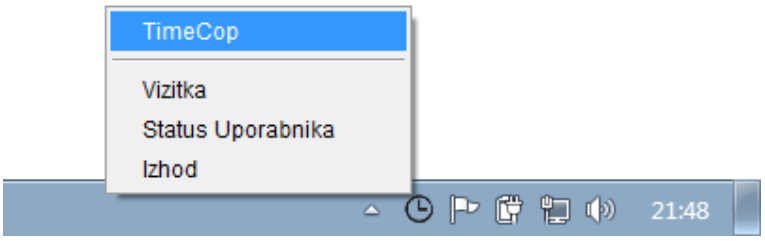
4.6 Izdelava glavnega programa

Ob zagonu operacijskega sistema se program avtomatsko zažene in minimizira med ikone v glavni orodni vrstici (angl. tray icon). Ob kliku na *TimeCop* ikono lahko izbiramo med:

- zagonom aplikacije *TimeCop*,
- vizitko,
- statusom uporabnika,
- izhodom.

4.6.1 Pladenj ikon v glavni orodni vrstici

Ob zagonu operacijskega sistema se hkrati zažene tudi aplikacija *TimeCop*. Aplikacija se minimizira v t. i. pladenj ikon na glavni orodni vrstici operacijskega sistema, glej sliko 4.10. Koda 4.4 prikazuje dodajanje aplikacije med pladenj ikon. Ob prijavi v administrativni del aplikacije nas aplikacija vpraša za uporabniško ime in geslo. Po geslu nas vpraša tudi, če želimo zapreti aplikacijo, tako se izognemo temu, da bi uporabnik brez dovoljenja zaprl program.



Slika 4.10: Pladenj ikon v glavni orodni vrstici

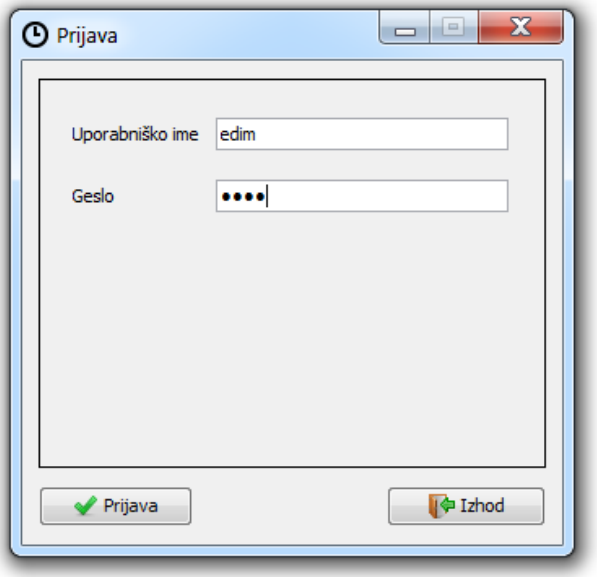
```
if (!SystemTray.isSupported()) {
        System.out.println("SystemTray ni podprt");
        return;}
    final PopupMenu popup = new PopupMenu();
    final TrayIcon trayIcon =
    new TrayIcon(createImage("images/clock.png", "tray
        icon"));
```

```
        trayIcon.setImageAutoSize(true); //popravimo
            velikost ikone
        final SystemTray tray = SystemTray.getSystemTray();
}
```

Koda 4.4: Dodajanje ikone *TimeCop* v orodno vrstico

4.6.2 Prijava

Slika 4.11 prikazuje prijavo v administrativni del aplikacije. Za prijavo potrebujemo uporabniško ime in geslo. Tako smo aplikacijo zaščitili, da nepooblaščeni uporabniki ne morejo brez pooblastil zaobiti zaščite. Geslo in uporabniško ime je poznano samo administratorju. Aplikacija podpira več administratorskih računov, tako lahko na primer do nastavitev aplikacije dostopa več uporabnikov z različnimi računi.

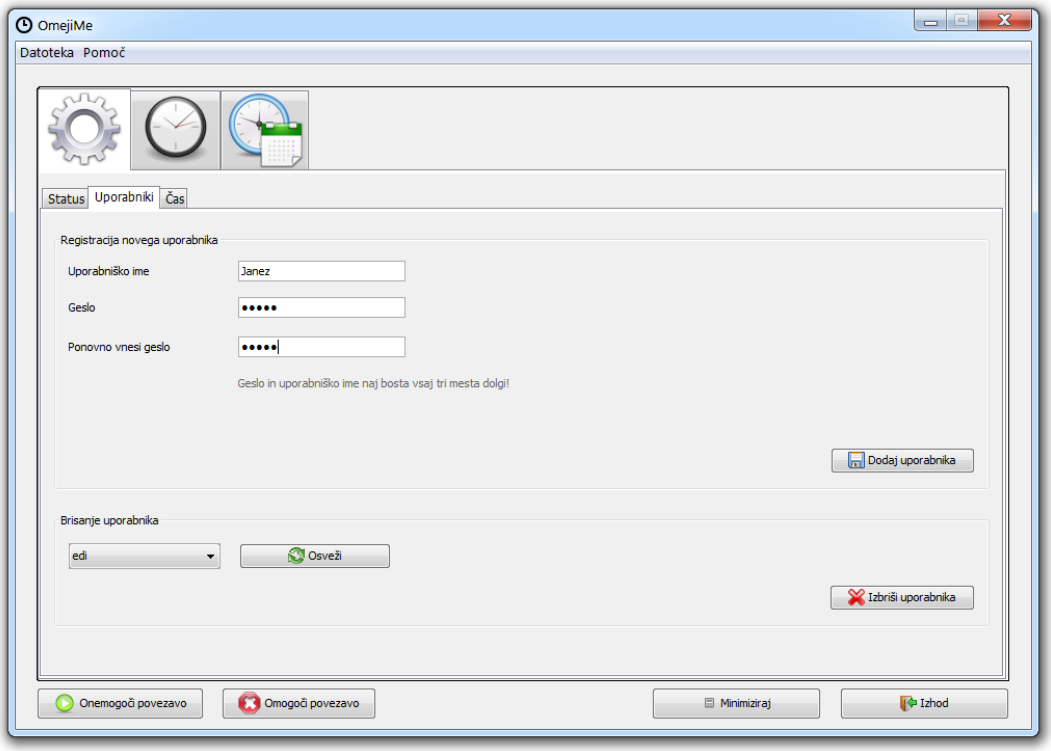


Slika 4.11: Prijava v glavni program

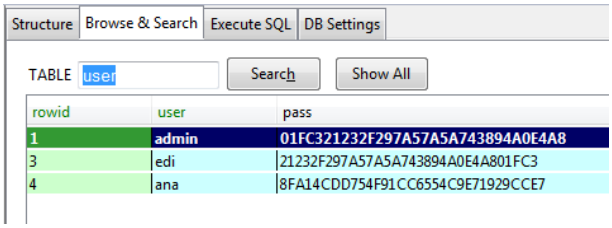
V kolikor bi uporabnik želel odjaviti ali zapreti aplikacijo, mora seveda vpisati svoje uporabniško ime in geslo.

4.6.3 Registracija

Aplikacija omogoča kreiranje več administrativnih uporabniških računov. Slika 4.12 prikazuje registracijo novega uporabnika. Ob registraciji se geslo in uporabniško ime zapišeta v podatkovno bazo. Geslo smo kriptirali z MD5 algoritmom (koda 4.5), tako da tudi ob morebitnem vpogledu v podatkovno bazo uporabnik ne more prebrati gesla, ampak vidi samo t. i. izvleček gesla (slika 4.13).



Slika 4.12: Registracija uporabnikov



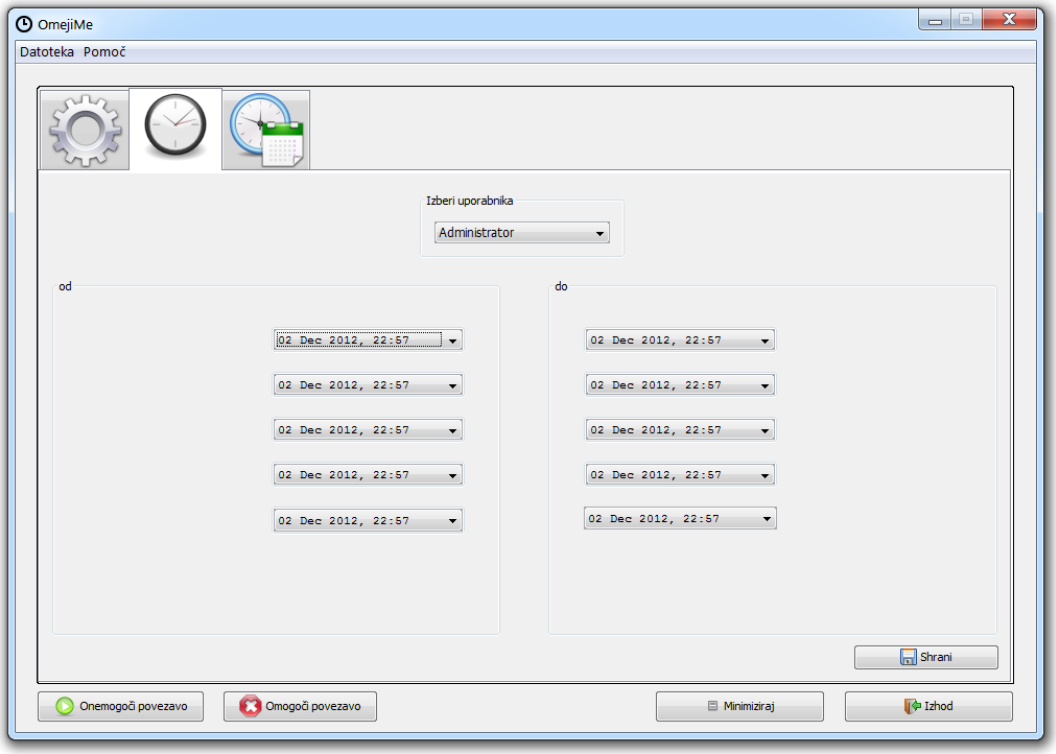
Slika 4.13: Vsebina tabele *user*

```
public static String kodiraj(String pass) {
  try {
    MessageDigest md = MessageDigest.getInstance("MD5");
    String input = pass;
    md.update(input.getBytes());
    byte[] output = md.digest();
    outputTemp=output.clone();
  } catch (Exception e) {
    Logging.logiraj(e.toString());}
 return bytesToHex(outputTemp);
}
```

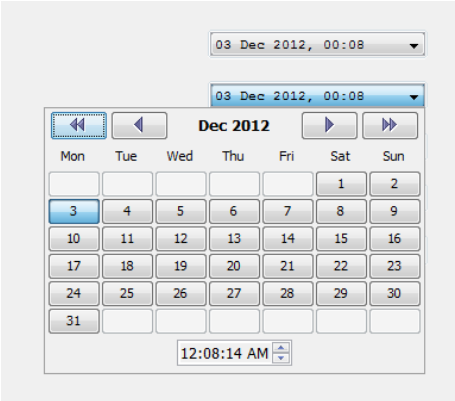
Koda 4.5: Metoda za kodiranje gesla

4.6.4 Omejevanje

Aplikacija omogoča dva načina časovnega omejevanja. Pri prvem načinu (slika 4.14) določimo dovoljen oziroma prepovedan čas dostopa za neko točno določeno časovno obdobje. S pomočjo t. i. izbirnika datumov (slika 4.14) določimo točen dan in uro začetka in konca omejevanja. Na voljo imamo pet različnih definicij pravil. Pri vnosu datumov aplikacija avtomatsko preveri, da ni končni datum morda pred začetnim datumom. Tako se izognemo morebitni napaki uporabnika in s tem nepravilnemu delovanju aplikacije.



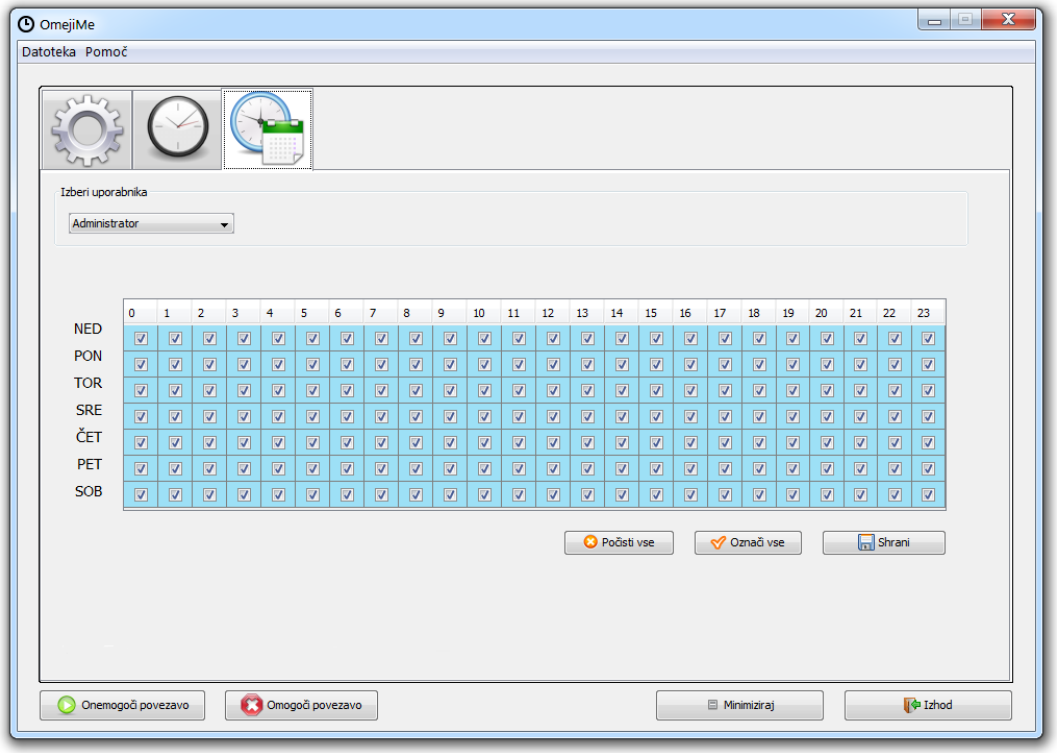
Slika 4.14: Časovno omejevanje – prvič



Slika 4.15: Izbirnik datumov

Pri drugem načinu pa lahko prepovemo oziroma dovolimo dostop do interneta za vsako uro in dan v tednu. To storimo tako, da obkljukamo časovne

intervale med polnimi urami. Glej sliko 4.16.



Slika 4.16: Časovno omejevanje – drugič

Aplikacija je zasnovana tako, da hkrati preverja pravila obeh načinov in v kolikor je v vsaj enem načinu dostop prepovedan, aplikacija prepove dostop do interneta. Koda 4.6 prikazuje preverjanje časovnih pravil.

```
if (tabelaObjektov==true && current_hour == j &&
   current_dan == i) {
      onemogoci = true;
}

if ((koledar1.getdate().before(sedaj) &&
   koledat2.getdate().after(sedaj))
            || (koledar3.getdate().before(sedaj) &&
               koledat4.getdate().after(sedaj)
```

```
            || (koledar5.getdate().before(sedaj) &&
               koledat6.getdate().after(sedaj)
            || (koledar7.getdate().before(sedaj) &&
               koledat7.getdate().after(sedaj)
            || (koledar9.getdate().before(sedaj) &&
               koledat8.getdate().after(sedaj) {
      onemogoci = true;
}
```

Koda 4.6: Koda preverjanja časovnih pravil

Če velja vsaj eden izmed zgoraj naštetih pogojev, prekinemo internetno povezavo s klicem metode *onemogocPovezavo()* (koda 4.7). Povezavo prekinemo tako, da naredimo poizvedbo mrežnih adapterjev, nato pa le-te nastavimo na onemogočeno. Pri operacijskem sistemu Windows upravljanje z napravami spada pod WMI vmesnik. Do WMI vmesnika iz Jave lahko dostopamo s pomočjo ActiveX komponente.

```
private void onemogociPovezavo() {
      ActiveXComponent com = new ActiveXComponent
      ("winmgmts:\\\\localhost\\root\\CIMV2");
      Variant svc = com.invoke("ExecQuery", new Variant(
      "Select * From Win32_NetworkAdapter
      WHERE NetConnectionID IS NOT NULL"));
      EnumVariant enumVariant = new
         EnumVariant(svc.toDispatch());
      Dispatch item = null;
      while (enumVariant.hasMoreElements()) {
         item = enumVariant.nextElement().toDispatch();
         try {
            System.out.println(
            Dispatch.call(item, "Caption").toString() + "
               : "
            + Dispatch.call(item,
```

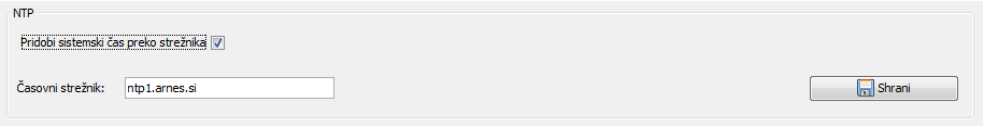
```
                "NetConnectionID").toString());

            Dispatch.call(item, "Disable");
        } catch (Exception e) {
            System.out.println(e.getLocalizedMessage());
            Logging.logiraj(e.toString());
        }
    }
}
```

Koda 4.7: Metoda *onemogociPovezavo()*

4.6.5 Network Time Protocol

Aplikacija pridobi sistemski čas s pomočjo protokola NTP (koda 4.8). Na ta način preprečimo, da bi uporabnik spremenil sistemski čas in tako zaobšel varnostne mehanizme. Privzeti strežnik je *ntp1.arnes.si*, uporabnik pa lahko vnese poljubnega. Glej sliko 4.17.



Slika 4.17: NTP nastavitve

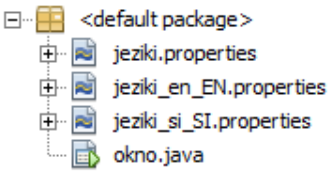
```
public static final void timeTCP(String host) throws
    IOException {
        TimeTCPClient client = new TimeTCPClient();
        try {
            // We want to timeout
            client.setDefaultTimeout(60000);
            client.connect(host);
            System.out.println(client.getDate());
```

```
            date1_fromServer = client.getDate();
        } finally {
            client.disconnect();
        }
}
```

Koda 4.8: Metoda za pridobitev sistemskega časa preko NTP strežnika

4.6.6 Lokalizacija

Aplikacija podpira slovenski in angleški jezik. Uporabnik nastavi željen jezik. Prevodi so shranjeni v *.ini* datotekah (slika 4.18).



Slika 4.18: Lokalizacija

Lokalizacija nam je prav prišla tudi pri izpisovanju datumov in časa v slovenski obliki. Primer klica slovenskih lokalnih nastavitev (koda 4.9):

```
Locale.setDefault(new Locale("si","SI"));
```

Koda 4.9: Nastavljanje slovenskih lokalnih nastavitev

4.6.7 Pisanje dnevnika

Vsi dogodki, do katerih lahko pride pri izvajanju aplikacije, se tudi ustrezno zapisujejo v *.log* datoteko (koda 4.10). Zapisujemo dogodke, kot so: *zagon aplikacije, zaustavitev aplikacije, pričetek blokiranja povezave, konec blokiranja povezave.* Poleg zgoraj omenjenih dogodkov pa beležimo tudi vse

morebitne napake oz. izjeme, do katerih bi lahko prišlo ob delovanju. Poleg vsakega dogodka se v datoteko zapiše tudi čas in datum dogodka, tako vemo, kdaj je prišlo do nekega dogodka. Poskrbeli smo tudi, da *.log* datoteka ne bi zrasla preveč, tako smo omejili maksimalno velikost datoteke na 1 MB. Ko je omenjena datoteka polna, se kreira nova datoteka, v katero se ponovno začnejo beležiti dogodki, ko je tudi ta polna, se ponovno odpre in sprazni prva datoteka in prične se zapisovanje v prvo. Koda 4.10 prikazuje zapisovanja neke izjeme v datoteko.

```
try {
    datoteka.write(s);
}
catch (IOException e) {
    fh = new FileHandler("File.log",1024000,2,true);
    logger.addHandler(fh);
    SimpleFormatter formatter = new SimpleFormatter();
    fh.setFormatter(formatter);
    logger.setLevel(Level.ALL);
    e.printStackTrace();
    logger.info(e.toString());
}
```

Koda 4.10: Pisanje dnevnika v datoteko

4.7 Načrtovanje namestitvenega programa

Namestitveni program mora:

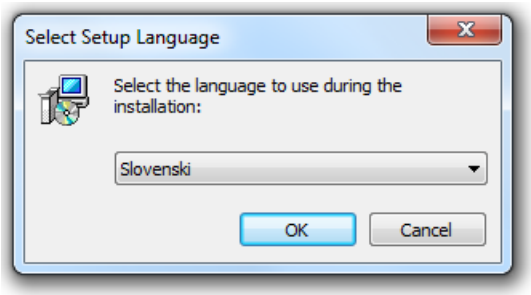
- biti preprost,
- namestiti vse potrebne datoteke,
- podpirati možnosti uporabe zagona programa ob zagonu operacijskega sistema,

- podpirati večjezičnost,

- podpirati možnost odstranitve programa.

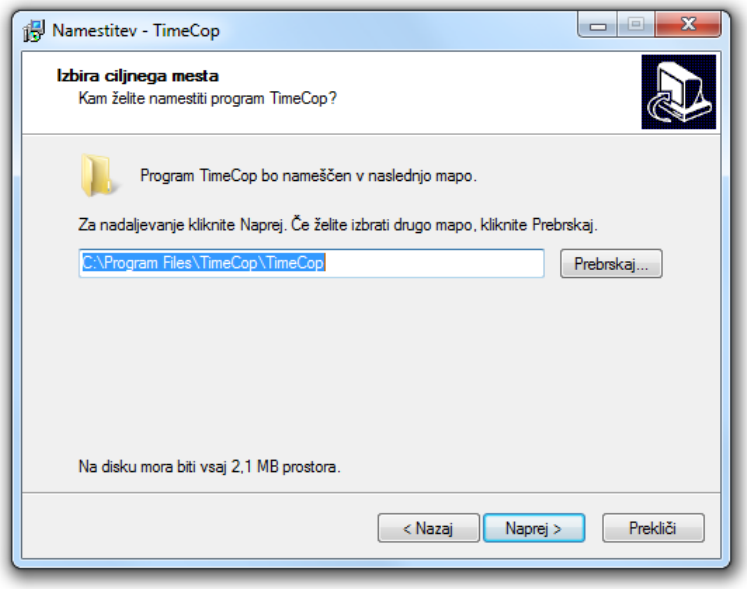
4.8 Izdelava namestitvenega programa

Za namestitev aplikacije smo uporabili program Inno Setup. Ob zagonu namestitvenega programa lahko izbiramo, v katerem jeziku bo potekala namestitev. Na voljo imamo slovenski in angleški jezik (slika 4.19). Slika 4.21 prikazuje dodajanje aplikacije v Start meni. Na slikah 4.22 in 4.23 pa je prikazan zaključek namestitve.

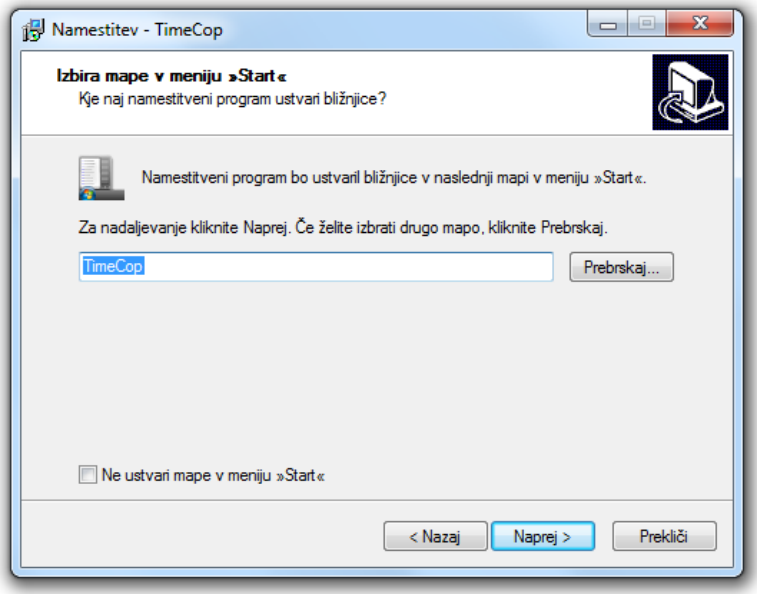


Slika 4.19: Izbira jezikov pri namestitvi

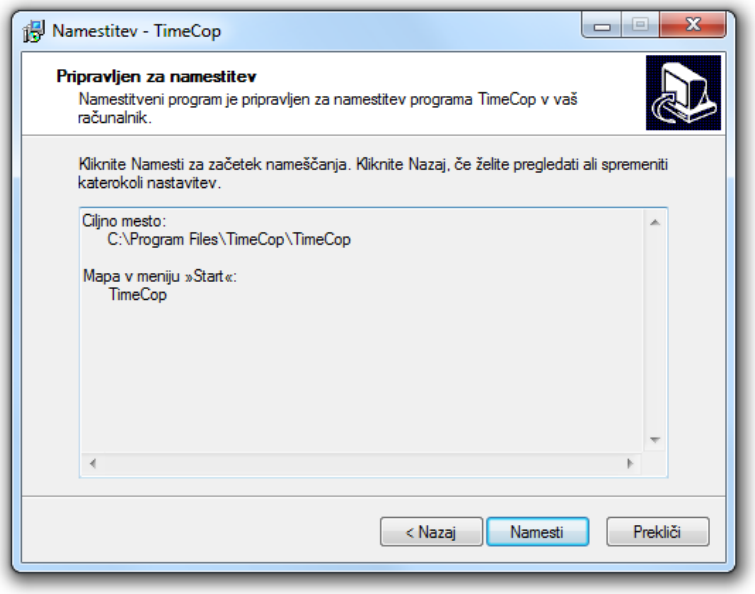
Namestitveni program nas vpraša, kam želimo namestiti aplikacijo (slika 4.20). Privzeta pot za namestitev je *C:\Program Files\TimeCop*. Če z lokacijo nismo zadovoljni, lahko izberemo drugo.



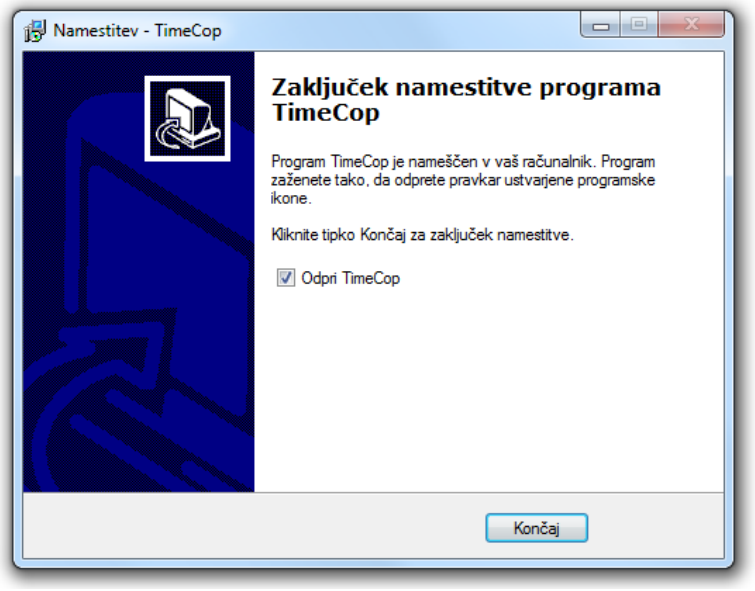
Slika 4.20: Namestitev – izbira poti



Slika 4.21: Namestitev – dodajanje v Start meni



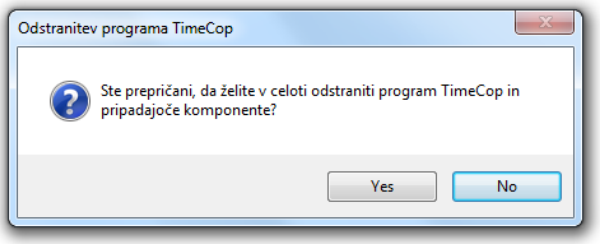
Slika 4.22: Namestitev – potrditev izbranih opcij



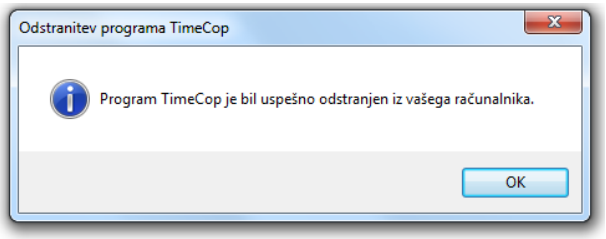
Slika 4.23: Namestitev – zaključno okno

4.9 Odstranitveni program

Aplikacija omogoča tudi preprosto odstranjevanje z zagonom *uninstall TimeCop* programa. Aplikacijo je mogoče odstraniti tudi preko nadzorne plošče, in sicer iz *Dodaj ali odstrani* programske opcije. Sliki 4.24 in 4.25 prikazujeta postopek odstranitve aplikacije.



Slika 4.24: Odstranitev programa



Slika 4.25: Obvestilo o odstranitvi

4.10 Ocena uporabniške izkušnje

Aplikacija je preprosta za uporabo. Z lahkoto jo lahko uporablja tudi računalniško nevešča oseba. Namestitev smo kar se da poenostavili, tako da nas namestitveni program sam popelje skozi namestitev. Aplikacija se avtomatsko zažene ob zagonu operacijskega sistema. Aplikacija ni pretirano požrešna glede virov. V pomnilniku skupaj z nadzornim programom povprečno zaseda 45 MB prostora. Poraba procesorskega časa pa je skorajda zanemarljiva. Grafični

vmesnik je minimalen in enostaven za razumevanje. Aplikacija omogoča tudi preprosto in udobno odstranitev iz sistema.

Poglavje 5

Zaključek

Namen diplomske naloge je bil narediti pregled možnih načinov časovnega omejevanja dostopa do interneta ter razvoj lastne aplikacije.

V teoretičnem delu smo predstavili razvojna orodja in tehnologije, ki so povezane z razvojem aplikacije na javanski platformi. V praktičnemu delu diplomskega dela smo podrobno predstavili vse faze razvoja aplikacije, katerih končni rezultat je delujoča aplikacija *TimeCop*. Pri razvoju nismo imeli večjih težav, saj je na temo razvoja javanskih aplikacij na voljo veliko literature in spletnih forumov, kjer smo našli vse odgovore na naša vprašanja. Implementirane so bile vse funkcionalnosti in lastnosti aplikacije, ki so bile zahtevane in načrtovane. Trenutno imamo v načrtu razvoj nove verzije aplikacije, ki bo vključevala posodobitve grafičnega vmesnika in razširjene možnosti omejevanja ter podporo delovanja v operacijskem sistemu Linux, kar bo aplikacijo naredilo še bolj zanimivo in uporabno.

Aplikacijo smo testirali en mesec in v tem času smo našli nekaj napak, ki pa smo jih sproti odpravili. Z delovanjem aplikacije smo zadovoljni. Zadovoljni smo tudi s samo hitrostjo delovanja aplikacije.

Preko podrobne predstavitve razvoja aplikacije na javanski platformi v praktičnem delu diplomske naloge smo se v prvi vrsti podrobno seznanili z novimi orodji in tehnikami razvoja javanskih aplikacij, prav tako pa smo s tem nadgradili svoje znanje programiranja v programskem jeziku Java. Za-

hvaljujoč se široki zbirki različne literature in spletnim forumom na temo programiranja javanskih aplikacij ter tudi predhodnim izkušnjam programiranja v programskem jeziku Java, smo orodja in tehniko razvoja aplikacij spoznali in usvojili hitro. Programski jezik Java smo si izbrali, ker je neodvisen od platforme. Sprva smo hoteli namreč napisati aplikacijo, ki bi delovala na poljubni platformi, vendar smo se kasneje zaradi specifičnosti operacijskih sistemov osredotočili na operacijski sistem Windows. Aplikacija je dobra osnova za morebitno razširitev na ostale operacijske sisteme.

Seznam slik

Literatura

[1] (2013) Parental Time Control Software Review. Dostopno na: http://parental–time–control–software–review.toptenreviews.com

[2] (2013) Parental Controls in Mac OS X 10.5. Dostopno na: http://cjrtools.org/mac/tutorials/mac–parental–controls.html

[3] (2013) Proxy strežnik. Dostopno na: http://www.sts–koper.si/arhiv/proxy

[4] (2013) CCProxy. Dostopno na: http://www.youngzsoft.net/ccproxy

[5] (2013) KidsWatch Time Management 6.5. Dostopno na: http://www.kidswatch.com/

[6] T. Lindholm, F. Yellin, The Java Virtual Machine Specification, Second Edition, Addison–Wesley, Palo Alto, California, 1999, pogl. 1

[7] (2013) NetBeans IDE features. Dostopno na: http://www.netbeans.com/features

[8] (2013) Windows Management Instrumentation. Dostopno na: http://en.wikipedia.org/wiki/Windows_Management_Instrumentation

[9] (2013) ActiveX. Dostopno na: http://en.wikipedia.org/wiki/ActiveX

[10] (2013) JACOB – Java COM Bridge. Dostopno na:
http://sourceforge.net/projects/jacob–project

[11] (2013) Microsoft COM. Dostopno na:
http://www.microsoft.com/com/default.mspx

[12] (2013) M. Kovačič, G. Žagar, F. Šteharnik, MD5 na zatožni klopi, Dostopno na:
http://matej.owca.info/predavanja/MD5

[13] (2013) MD5. Dostopno na:
http://en.wikipedia.org/wiki/MD5

[14] (2013) JavanBeans. Dostopno na:
http://en.wikipedia.org/wiki/JavaBeans

[15] (2013) Inno Setup. Dostopno na:
http://www.jrsoftware.org/isinfo.php

[16] L. Manjolovič, Vzpostavitev NTP strežnika, diplomska naloga, Višja strokovna šola na Tehniškem šolskem centru Nova Gorica, Slovenija, 2009. Dostopno na:
http://luka.manojlovic.net/projects/ntp–server–ntpmostovnacom

[17] (2013) SQLite. Dostopno na:
http://www.sqlite.org/about.html