embarcadero
reseller partner

Embarcadero akademija: RAD Studio in Linux

Primož Gabrijelčič
http://primoz.gabrijelcic.org

v1.1

MARAND
Napredna računalniška hiša

Kazalo

Vsi programi, omenjeni v tem dokumentu, so na voljo na naslovu http://17slon.com/EA/EA-Linux.zip.

RAD Studio in Linux

Razvojno okolje RAD Studio je v zadnji različici (10.2 Tokyo) dobilo možnost izdelave programov za okolje Linux. Trenutno lahko programiramo konzolne (tekstovne) programe, ne pa tudi grafičnih uporabniških vmesnikov (za izdelavo le-teh se ozrite po zunanjih dodatkih CrossVCL in FMXLinux). A tudi s tekstovnim programom lahko storimo veliko – na primer izdelamo podatkovni strežnik, ki ga nato poganjamo na strojno nezahtevnih računalnikih ali pa v virtualnih računalnikih v oblaku. Na splošno velja, da so »oblačni« računalniki z Linuxom cenejši od enako zmogljivih Windows sistemov. (Nič čudnega – če ima virtualka nameščen Linux, zanj ni treba plačati licence.) Z nekaj triki pa lahko konzolni program predelamo tudi v čisto pravi Linux servis, ki se zažene ob zagonu sistema.

Za razvoj programov za Linux potrebujete RAD Studio 10.2 Enterprise (ali Delphi 10.2 Enterprise). Programe lahko ta trenutek pišete le v jeziku Delphi (podpora za C++Builder bo na voljo v naslednji različici), v celoti pa se prevedejo na računalniku Windows. Sistem Linux potrebujete le za preizkušanje. Na voljo je tudi (dokaj omejeno) razhroščevanje, pri katerem na sistemu Linux poženete program Platform Assistant (PAServer), ki skrbi za poganjanje programa in komunikacijo z razhroščevalnikom. Povsem enako, kot pri razvoju za OS X, torej.

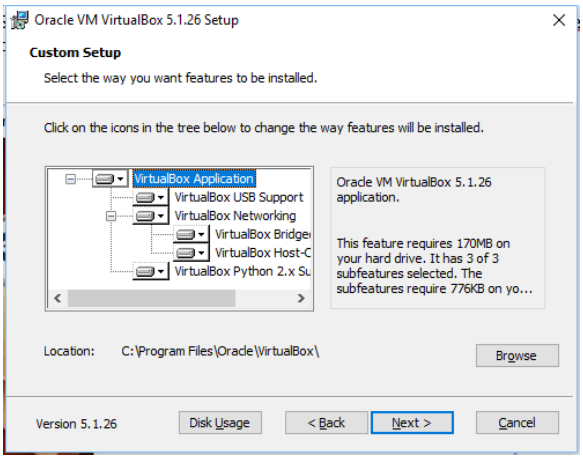
Uradno sta za poganjanje programov podprti distribuciji Ubuntu Server 16.04 LTS in Red Hat Enterprise 7, z veliko verjetnostjo bodo programi delovali v njunih naslednikih (preizkusili smo Ubuntu 17.04 in ni povzročal nobenih težav), delovanje na drugih distribucijah pa ni zagotovljeno. Linux lahko teče na ločenem računalniku, ali pa kar v virtualnem okolju. Pripravo take možnosti si bomo ogledali v tem priročniku.

Namestitev VirtualBoxa

Najprej moramo namestiti virtualno izvajalno okolje. Uporabimo lahko popularni VMWare, odprtokodni VirtualBox, ali katerokoli drugo okolje, ki zmore poganjati Ubuntu. V naši delavnici bomo uporabili VirtualBox.

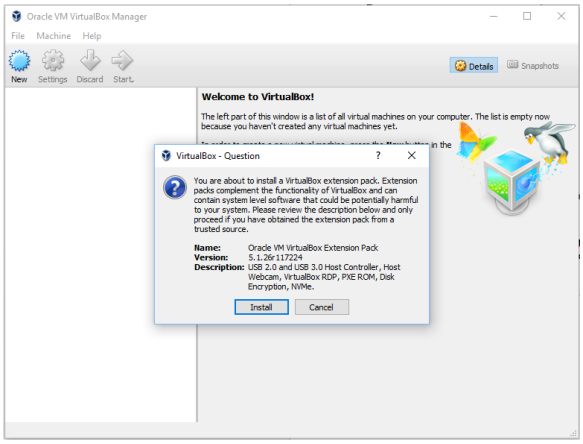
Najprej prenesemo namestitveni program. Preklikamo se skozi namestitvenega čarovnika in namestimo vse komponente. Če želite, lahko odstranite podporo za upravljanje VirtualBox iz Pythona, ki pa ne zaseda veliko prostora.

Med namestitvijo bo morda potrebno potrditi nameščanje Oraclovih gonilnikov za USB in omrežne kartice.

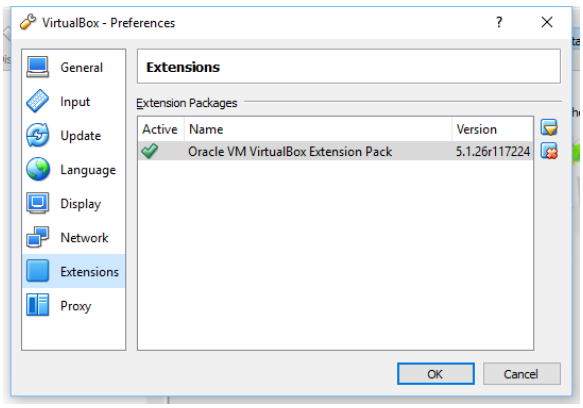


Če želite namestiti paket z dodatki ("Extension Pack"), ki razširi VirtualBox s podporo USB 2.0 in 3.0, možnostjo dostopa RDP in še nekaj "cukrčki", ga prenesite in poženite. *Extension Pack* je prost za domačo in izobraževalno, ne pa za poslovno rabo.

Namestitvena datoteka ima končnico .vbox-extpack, ki je asociirana s programom VirtualBox. Dvoklik na datoteko bo pognal VirtualBox in namestil razširitve.



Ali so razširitve pravilno nameščene, lahko preverimo v menuju *File, Preferences*.

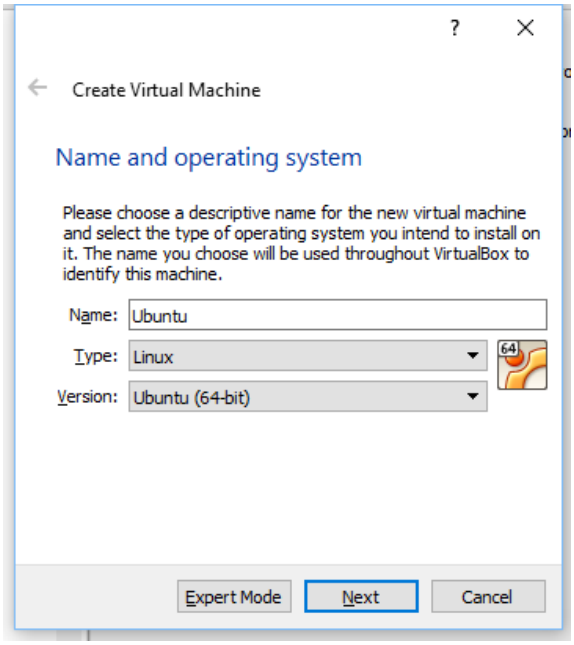


Priprava virtualnega računalnika

Postopek nameščanja operacijskega sistema Ubuntu v VirtualBox začnemo tako, da prenesemo namestitveno sliko (.iso). Uporabimo lahko sliko namiznega okolja (Desktop), ki vsebuje grafični uporabniški vmesnik, ali pa sliko strežniškega okolja (Server), ki ponuja le konzolni način dela, a zato na disku zasede bistveno manj prostora. Ker RAD Studio za Linux zaenkrat ne ponuja možnosti izdelave programov z grafičnim uporabniškim vmesnikom, smo za to predstavitev izbrali možnost *Server*.

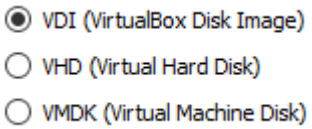
Postopek začnemo s klikom ikone New.

Vnesemo ime virtualnega stroja (kakršno koli), ter izberemo tip *Linux* in verzijo *Ubuntu (64-bit)*.

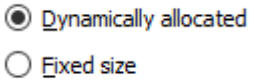


Nato določimo količino pomnilnika, ki ga bo imel virtualni stroj na voljo. Priporočamo 4 GB za različico *Desktop*, medtem ko *Server* čisto dobro deluje tudi z manj (recimo 1 GB).

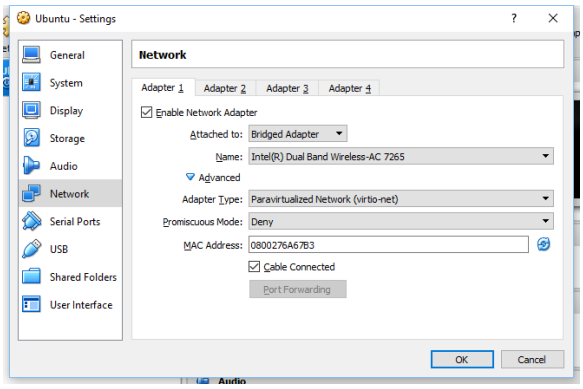
Določiti moramo tip in velikost virtualnega diska. Priporočamo naslednje izbire:



- Tip diska: *VDI*
- Način dodeljevanja prostora: *Dynamically allocated*
- Velikost diska: 20 GB je za strežnik Ubuntu popolnoma dovolj, lahko pa izberete večjo številko, ker bo dinamično dodeljen disk porabil na (pravem) disku le toliko prostora, kolikor bo vseboval podatkov.

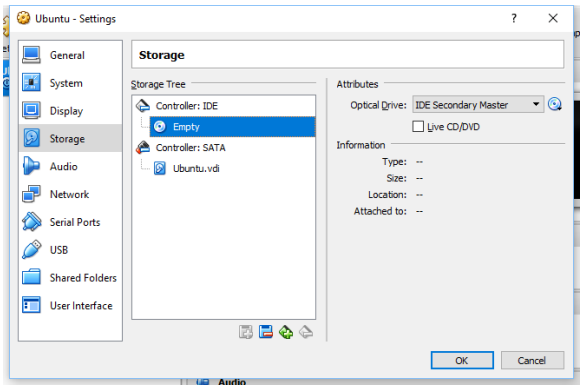


V VirtualBoxu se bo pojavil nov, prazen virtualni računalnik. Izberemo ga, kliknemo gumb *Settings* in na levi strani okna izberemo *Networks*.



Vključimo možnost *Enable Network Adapter* in ga nastavimo na tip *Bridged Adapter*. Odpremo vejo nastavitev *Advanced* in tip omrežnega vmesnika nastavimo na *Paravirtualized Network*. Zdaj bo virtualni računalnik dobil IP naslov iz našega omrežja.

Kliknemo še stran z nastavitvami *Storage* in nastavimo, da bo virtualni CD/DVD priključen na namestitveno datoteko ISO. Datoteko izberemo s klikom modre ikone CD na desni strani okna.



S tem je zaključena konfiguracija virtualnega računalnika in lahko začnemo z nameščanjem operacijskega sistema Ubuntu.

Namestitev Ubuntuja

Preverimo, da je v VirtualBoxu izbran pravi virtualni računalnik in kliknemo ikono *Start*. Virtualni računalnik se bo pognal iz navideznega pogona DVD, ki je priključen na namestitveno sliko .iso in začel z namestitvijo.

Najprej moramo izbrati jezik. Priporočam angleščino, ker je na internetu bistveno lažje najti angleške opise napak in problemov, kot slovenske.

Namig: Ko kliknete v okno virtualnega računalnika, ta »pograbi« miško. Sprostite jo tako, da pritisnete in spustite desno tipko Ctrl. Kombinacijo tipk se da nastaviti v File, Preferences, Input, Host Key Combination.

Pritisnemo Enter na izbiri *Install Ubuntu Server* in še enkrat izberemo jezik.

Sistem nato vpraša za lokacijo, a nas ni na prvem izbirnem zaslonu, zato izberemo *Other*.

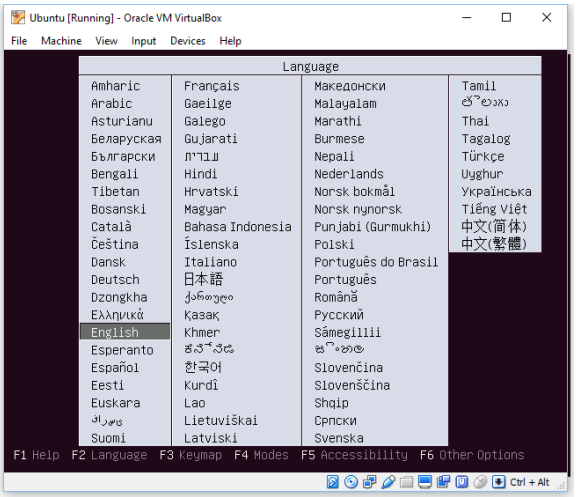
Na naslednjem zaslonu izberemo *Europe*, nato poiščemo Slovenijo.

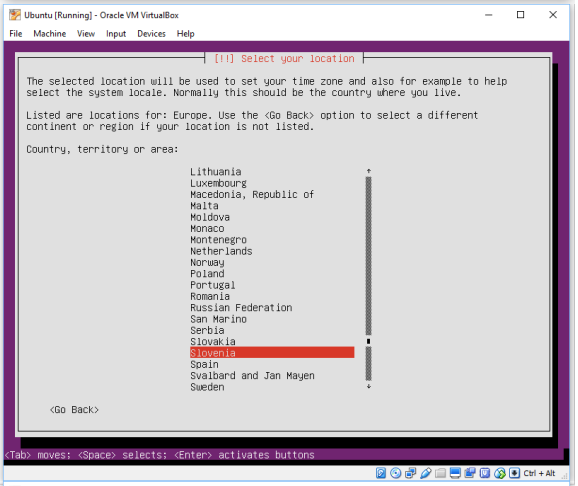
Namestitveni program nato ugotovi, da ne ve, kaj bi z nami, in priporoča nastavitev *United States – en_US.UTF-8.* ☹

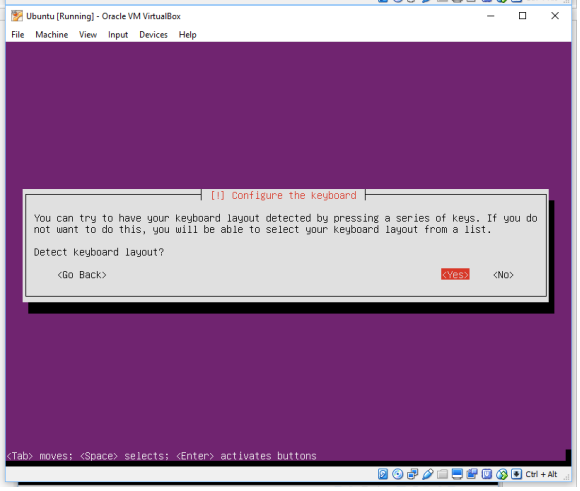
Upamo, da se bo to obnašanje v prihodnih različicah kaj izboljšalo …

Nato poskusi namestitveni program samodejno zaznati tipkovnico. To mu gre kar dobro od rok.

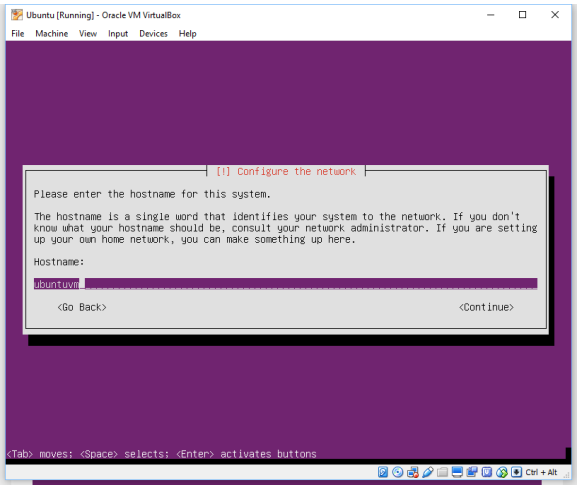
Nasvet: Med »gumbi« se premikate s tipko <Tab>. Izbirate s preslednico in potrdite s tipko <Enter>.





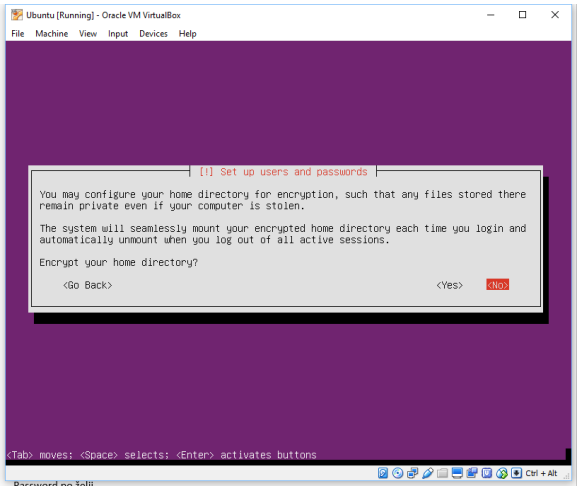


Ko je tipkovnica izbrana, namestitveni program nekaj časa dela brez vprašanja, nato moramo vnesti ime, pod katerim se bo ta računalnik predstavljal na omrežju.



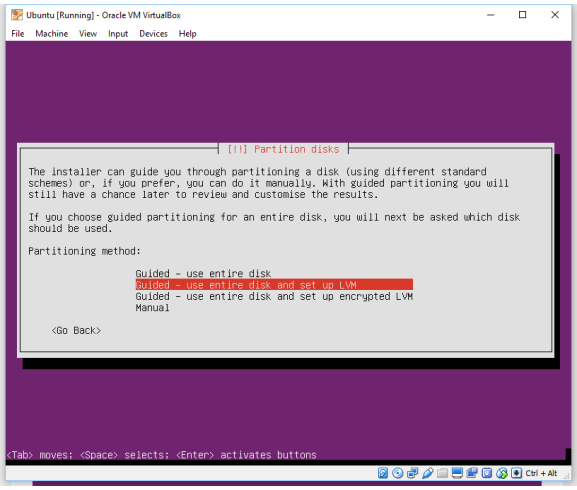
Nato vnesemo podatke (ime, uporabniško ime, geslo) za privzeti uporabniški račun. Prvo pravilo rabe Linuxa (in Unixov na splošno) je pač, da se nikoli ne prijavljamo v korenski račun (root). Privzeti račun bo imel možnost poganjanja ukazov z administratorskimi pravicami (sudo), zato bomo imeli v njem poln dostop do nastavitev operacijskega sistema.

Odločiti se moramo, ali bomo zašifrirali imenike z datotekami uporabnikov. Takšne zaščite za razvojno rabo najverjetneje ne potrebujete.



Nato potrdimo časovni pas. Običajno namestitveni program pravilno ugane, kje se nahajamo.

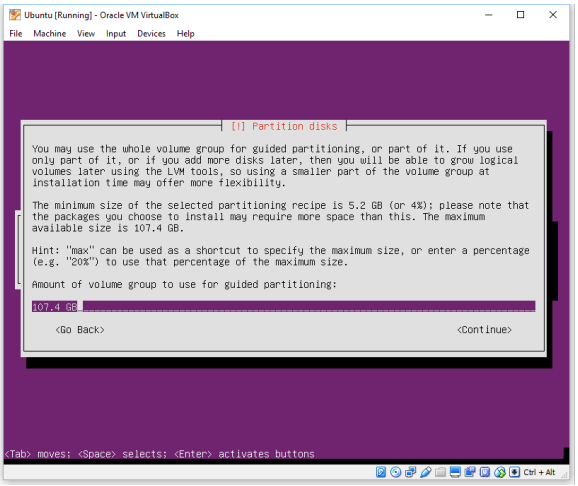
Določiti moramo, kako bo razdeljen disk. Ker se s podrobnim razdeljevanjem diska na razdelke (particije) raje ne bi ukvarjali, izberemo možnost »Use entire disk and set up LVM«.



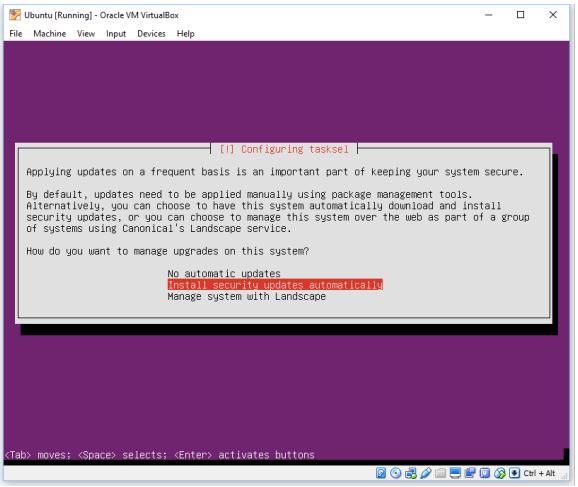
Nato izberemo (edini) disk in potrdimo, da smo res prepričani, da hočemo prepisati ta disk.

Izberemo celo velikost diska (pravi podatek je že vpisan).

Na naslednjem zaslonu še enkrat potrdimo izbire in glavni del nameščanja se končno začne.

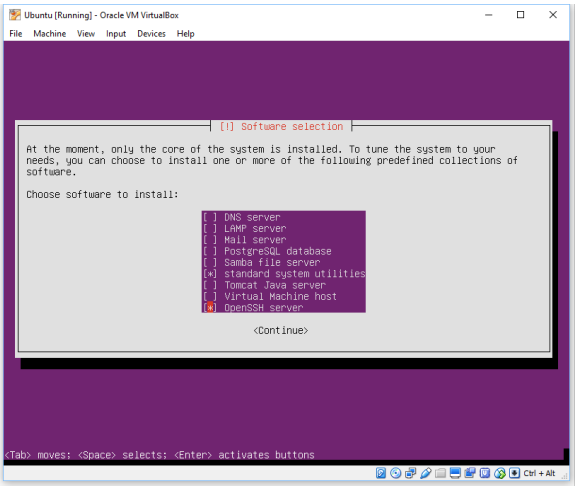


Naslednja točka je nastavljanje sistema za prenos posodobitev. Ker gre za virtualni računalnik, lahko preprečite samodejne namestitve, ali pa dovolite, da se pomembne namestitve nameščajo samodejno.

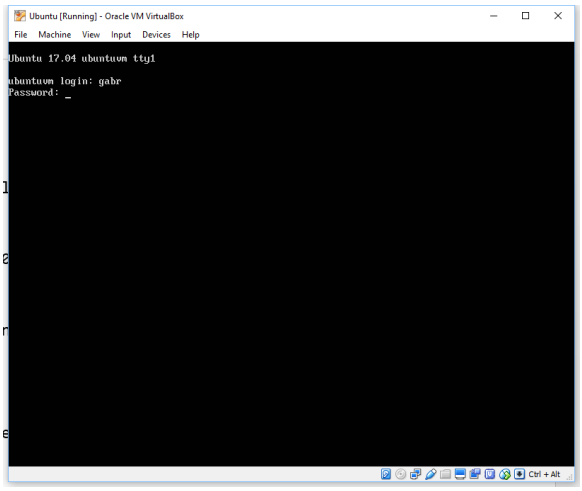


Sistem nas nato vpraša, katere programe naj namesti. Nujno izberite »OpenSSH server«, ker ga potrebujete za prenos datotek iz Windows na Ubuntu in nazaj.

Zdaj se začne najdaljši del instalacije. Bodite potrpežljivi, saj se včasih kar dolgo zdi, da se ne dogaja nič.



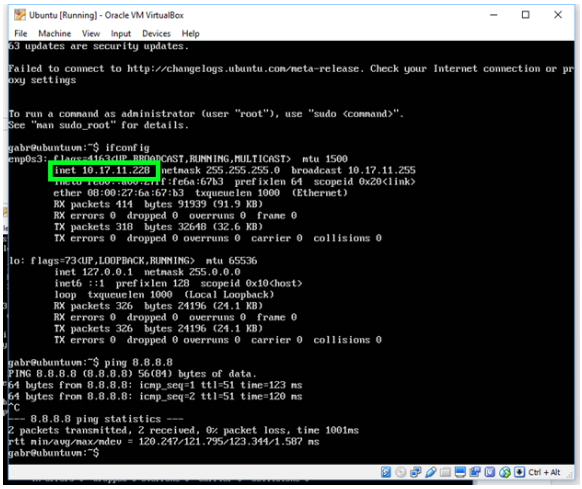
Na koncu potrdimo še nekaj izbir in se (končno!) znajdemo pred črnim konzolnim zaslonom. Uporabimo uporabniško ime in geslo, ki smo ga nastavili med namestitvijo in se prijavimo v sistem.



Virtualni računalnik najlažje ugasnemo tako, da kliknemo ikono X v konzolnem (»črnem«) oknu. VirtualBox bo virtualko pospravil na disk in ob naslednjem zagonu jo bomo dobili v natančno takem stanju, kot smo jo zapustili.

Prvi koraki

Po namestitvi najprej preverimo, če deluje omrežna povezava. V okno Ubuntu vpišemo ukaz *ifconfig* in preverimo, kakšen omrežni naslov je dobil sistem. V primeru na sliki je to *10.17.11.228.*

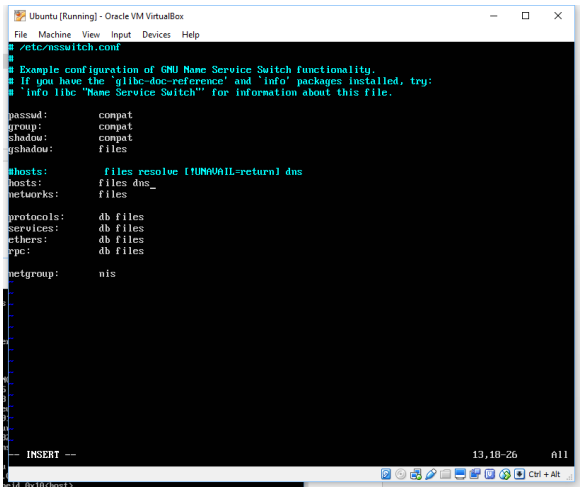


Nato preverimo dostop do interneta z ukazim *ping 8.8.8.8* (naslov Googlovega strežnika DNS) in preverimo, ali dobimo odziv.

Preverimo še delovanje sistema DNS, tako da »pingnemo« nek računalnik z imenom, namesto s številko. Na primer: *ping* www.ijs.si.

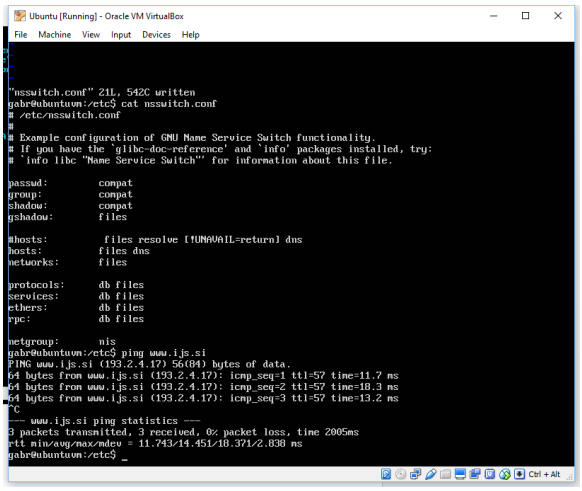
Če *ping* na številko deluje, v drugem primeru pa dobite nazaj »Name or service not known«, morate verjetno popraviti datoteko */etc/nsswitch.conf.*

Uporabite ukaz *sudo nano /etc/nsswitch.conf* da odprete datoteko v minimalističnem urejevalniku *nano.* Predpona *sudo* je potrebna, ker spreminjate sistemsko datoteko, do katere običajen uporabnik nima dostopa.



Pred obstoječo vrstico, ki se začne s *hosts:* vpišite znak # (komentar). Nato pod njo dodajte vrstico: *hosts: files dns*. (Presledke lahko vnesete s preslednico ali tabulatorko.)

Datoteko shranite z ukazom *Ctrl+X, Y, <Enter>.* Z ukazom *cat /etc/nsswitch.conf* izpišete vsebino datoteke na zaslon (za kontrolo). Nato še enkrat preizkusite *ping*. Če ne deluje, se obrnite na internet. ☹



Ko dostop do interneta deluje, posodobite računalnik z zaporedjem ukazov:

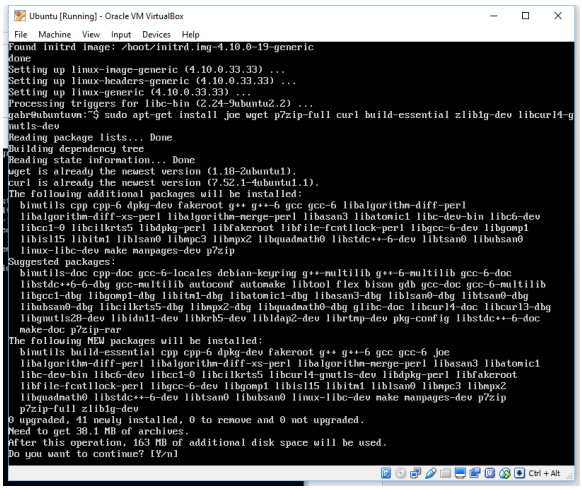
- sudo apt-get update
- sudo apt-get upgrade
- sudo apt-get dist-upgrade

Namestimo še vse pakete, ki jih potrebujemo za sodelovanje s programom Platform Assistant (PAServer), ki skrbi za komunikacijo z RAD Studiom.

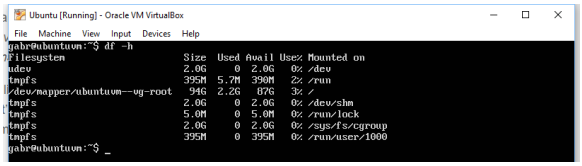
- sudo apt-get install joe wget p7zip-full curl build-essential zlib1g-dev libcurl4-gnutls-dev

Nato počistimo nepotrebne ostanke:

- sudo apt-get autoremove
- sudo apt-get autoclean



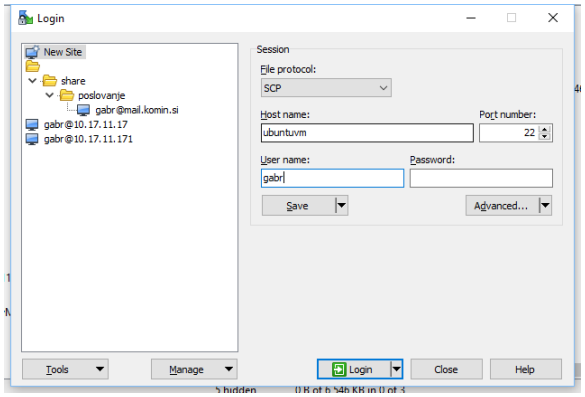
Z ukazom *df -h* lahko preverimo količino porabljenega diska.



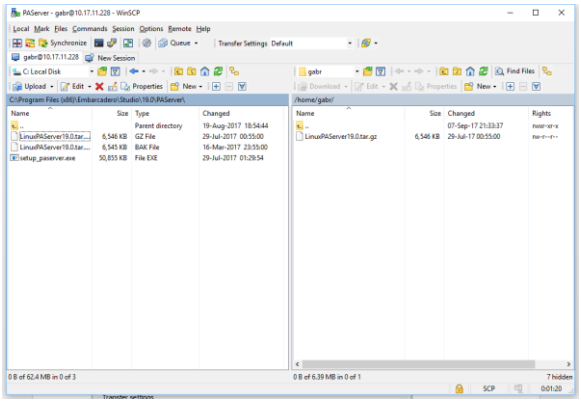
Namestitev razvojnega okolja

Na gostiteljskem računalniku morate imeti nameščen RAD Studio 10.2 Tokyo Enterprise ali Delphi 10.2 Tokyo Enterprise. Priporočamo, da namestite tudi vse nadgradnje. (V času pisanja te skripte je to nadgradnja 10.2.1.)

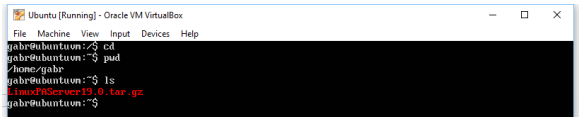
Nato namestite program WinSCP, s katerim lahko kopiramo datoteke na Ubuntu. Za kopiranje uporabimo protokol *SCP*, v polje *Host name* pa vpišite IP številko virtualnega računalnika (tisto, ki jo je izpisal ukaz *ifconfig*) ali njegovo ime (nastavljeno med namestitvijo; izpiše ga tudi ukaz *hostname*).



V imenik /home/<uporabnik> (kjer je <uporabnik> uporabniško ime, ki ste ga naredili med namestitvijo; v tem primeru je <uporabnik> *gabr*) skopirajte datoteko "c:\Program Files (x86)\Embarcadero\Studio\19.0\PAServer\LinuxPAServer19.0.tar.gz".



V Ubuntuju se postavite v domači imenik (ukaz *cd*) in preverite, kaj je v njem (*ls*).

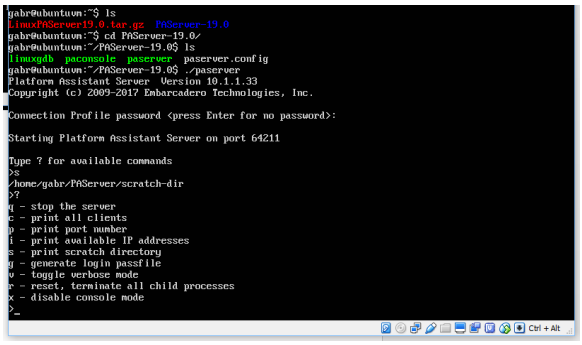


Datoteko LinuxPAServer19.0.tar.gz odpakiramo z ukazom *tar xzvf LinuxPAServer19.0.tar.gz.* Bodite pozorni na velike in male črke, ker jih Linux strogo loči. [Magična inkantacija *xzvf* pomeni: odpakiraj (e**x**tract), uporabi g**z**ip (ker je končnica .gz), izpisuj dogajanje (**v**erbose), vhod je v datoteki (**f**ile).]

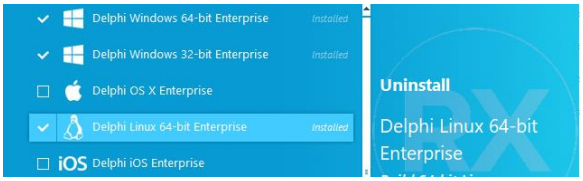
Nasvet: Vnos imen datotek in imenikov si olajšate tako, da začnete tipkati ime in po nekaj znakih pritisnete tipko <Tab>. Ubuntu bo vnesel polno ime datoteke/imenika. Če se to ne zgodi, obstaja več datotek/imenikov z enakim začetkom. Vnesite še nekaj znakov in poskusite znova.

Dva zaporedna pritiska na <Tab> izpišeta imena vseh datotek/imenikov z enakim začetkom.

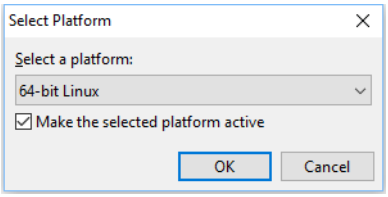
Postavimo se v novo mapo PAServer-19.0 (*cd PAServer-19.0*) in poženemo program *paserver* (*./paserver*). Navesti moramo relativno pot do programa (*./*), ker Linux programov ob zagonu ne išče v trenutnem direktoriju.



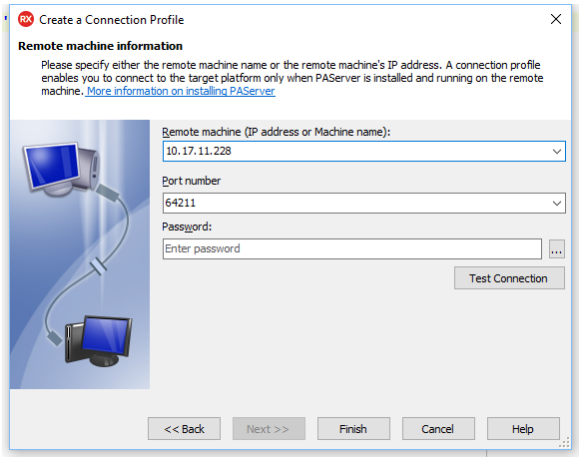
V RAD Studiu morate imeti nameščeno podporo za Linux. To lahko preverite tako, da v RAD Studiu izberete menu *Tools, Manage Platforms* ter preverite, da je v seznamu izbrana možnost *Delphi Linux 64-bit Enterprise*.



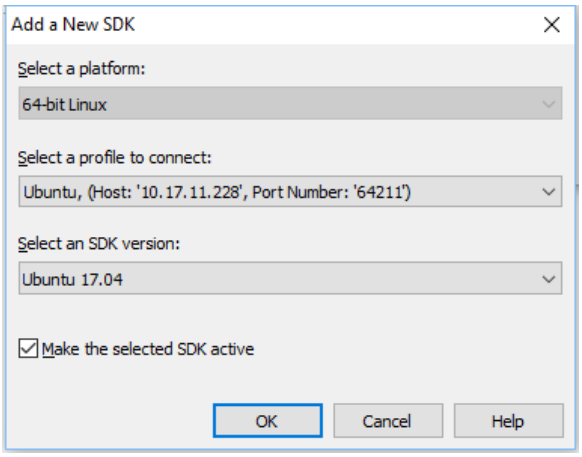
Nato naredimo novo konzolno aplikacijo v jeziku Delphi (C++Builder ta trenutek še ne podpira prevajanja za Linux). V oknu *Project Manager* z desno mišjo tipko kliknemo *Target Platforms*, izberemo *64-bit Linux* in določimo, da naj bo platforma aktivna. V program dodamo enostaven izpis (denimo: *Writeln("Hello, Linux!");*) in pritisnemo F9.

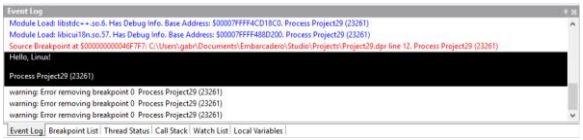


RAD Studio bo preskočil v način za izdelavo omrežnega profila. Profilu moramo dati ime (na primer *ubuntu*), in omrežno ime ali številko IP. S klikom gumba *Test Connection* preverimo, ali gostiteljski računalnik »vidi« PAServer na virtualnem računalniku. Ubuntu se privzeto namesti z ugasnjenim vdorobranom, zato s povezavo ne bi smelo biti težav.



Nato bo RAD Studio pripravil SDK za prevajanje na izbrani ciljni sistem. Vse nastavitve pustite na privzetih vrednosti in kliknite OK. Sledi nekaj sekund/minut kopiranja datotek, nato pa se bo program prevedel in pognal na virtualnem računalniku. (Če se to ne zgodi, pritisnite F9 še enkrat.)

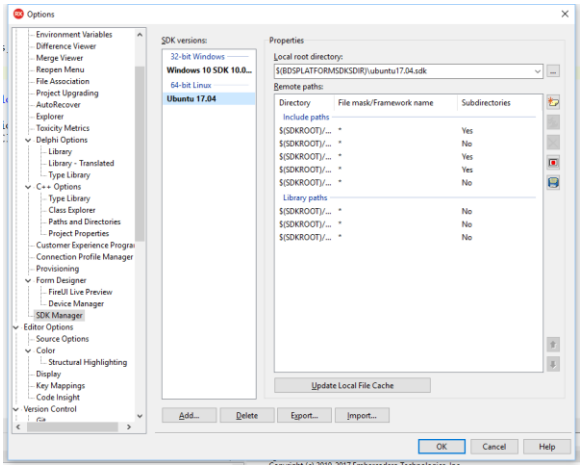




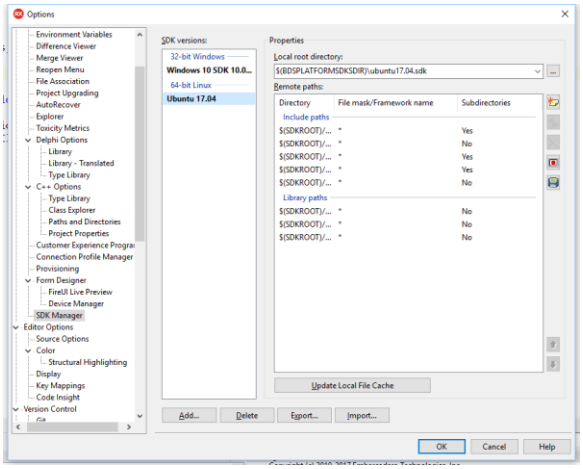
Če pogledate v navidezni računalnik, boste videli, da v njem ni videti izpisa »Hello, Linux!«. Ta se pojavi v RAD Studiu v oknu *Event Log*. Še bolj nerodno je, da program, ko ga razhroščujemo, ne bere iz standardnega vhoda (če karkoli natipkamo v Ubuntu, se to interpretira kot ukaz za PAServer, ne kot vhod našega programa), kar je neprijetna omejitev razhroščevalnega okolja.

Program bo deloval tako, kot pričakujemo, le, če ga poženemo brez razhroščevalnika (Ctrl+Shift+F9), v PAServerju pa uporabimo ukaz X (*disable console mode*).

Program lahko poženemo tudi tako, da se postavimo v mapo, kamor ga je shranil PAServer, ter ga poženemo »ročno«.



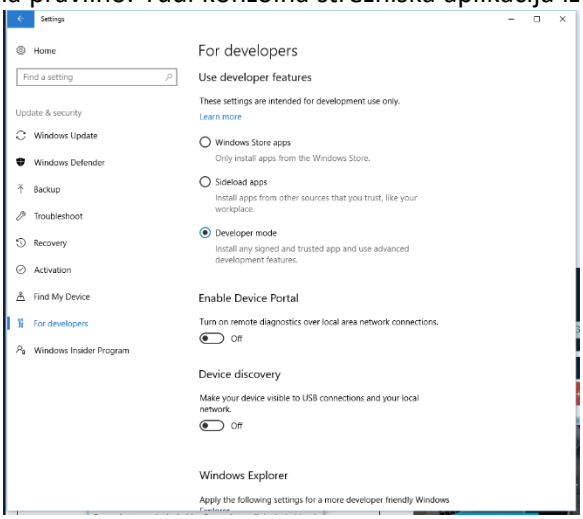
Nameščene SDK-je lahko upravljamo v *Tools, Options, SDK Manager.*



Povezave upravljamo v *Tools, Options, Connection Profile Manager.*

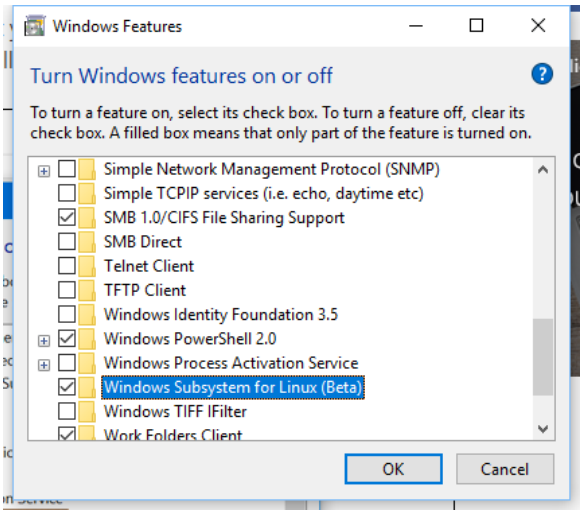
Windows Subsystem for Linux

V Windows 10 lahko namestite Ubuntu (le s tekstovnim vmesnikom). V slednjem lahko poganjate programe, razvite v RAD Studiu. No, vsaj poskusite lahko, saj Ubuntu na Windows uradno še ni podprt, pa tudi kaj hitro naletite na funkcijo, ki v njem ne dela pravilno. Tudi konzolna strežniška aplikacija iz tega predavanja v Ubuntu na Windows ne deluje. Prav tako (še) ne deluje program PAServer. Vseeno pa smo se odločili opisati namestitev tega podsistema, saj obstajajo velike možnosti, da bo v prihodnosti podprt tudi v RAD Studiu. Če ne drugega, bo omogočal Ubuntu na Windows zelo enostavno preizkušanje programov.

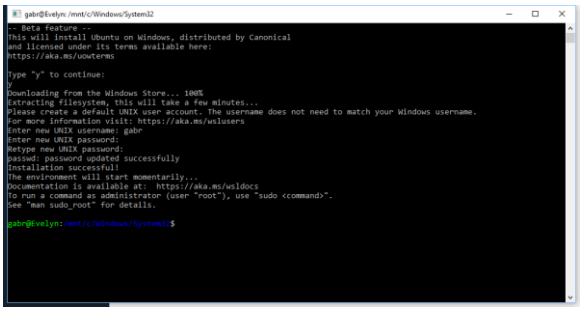


Najprej morate vključiti način za razvijalce. Odprite nastavitve (*Settings*) in v njih skupino nastavitev *Update & security* ter v njej podskupino *For developers*. Označite možnost *Developer mode*.

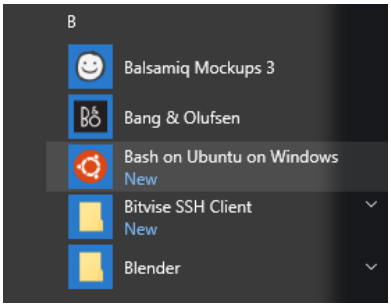
Zaprite nastavitve in odprite nadzorno ploščo (*Control panel*) ter v skupini *Programs and Features* kliknite *Turn Windows features on or off*. V seznamu poiščite *Windows Subsystem for Linux (Beta)* in jo označite.



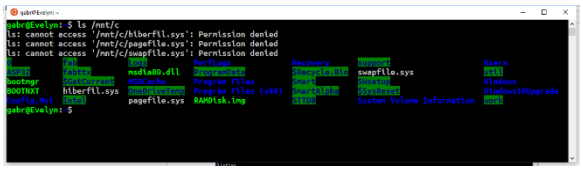
V zagonski menu ste dobili novo možnost *Bash*. Izberite jo in pognalo se bo konzolno okno, v katerem se bo začela namestitev Ubuntuja. Med namestitvijo boste kreirali tudi uporabniško ime, ki bo specifično za Ubuntu. (Lahko pa brez težav uporabite ime obstoječega Windows računa.)



Po končani namestitvi lahko Ubuntu poženete z izbiro *Bash on Ubuntu on Windows.*



V podsistem lahko brez težav nameščate pakete za Ubuntu z ukazom *apt-get*. Če bi radi izmenjevali datoteke z datotečnim sistemom gostiteljskega računalnika Windows, najdete datoteke gostiteljskega računalnika v mapi /mnt/c.



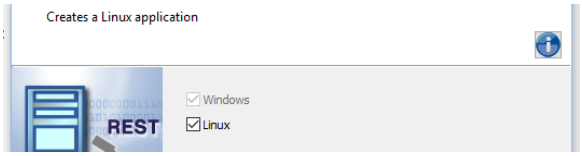
Programska oprema za spremljanje porabe goriva

Za prikaz razvoja programske opreme za Linux bomo napisali zelo enostavno kombinacijo grafičnega večplatformnega odjemalca in tekstovnega strežnika. Strežnik bo hranil bazo podatkov o porabi goriva, odjemalec pa bo omogočal (zelo enostavno) urejanje podatkov. Za hranjenje podatkov bomo uporabili komponente FireDAC, za prenos ukazov in podatkov med odjemalcem in strežnikom pa ogrodje DataSnap REST.

Strežnik

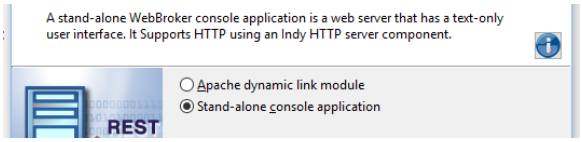
Razvoj strežniškega programa bomo začeli z izdelavo nove konzolne aplikacije (*File, New, Other, Delphi Projects, Console Application*). Vanjo dodamo strežnik DataSnap REST (*File, New, Other, DataSnap Server, DataSnap REST Application*).

Pojavi se čarovnik, v katerem moramo najprej označiti, da naj program deluje tudi na Linuxu.

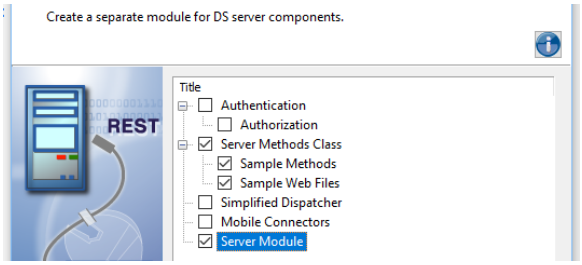


Nato izberemo, ali bomo naredili dodatek za spletni strežnik Apache, ali samostojno konzolno aplikacijo.

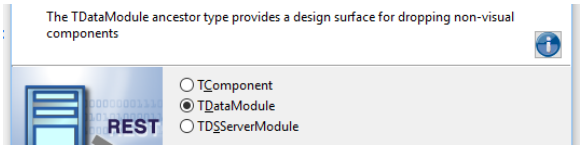
Nato izberemo vrata. Privzeta vrednost 8080 bo zadoščala.



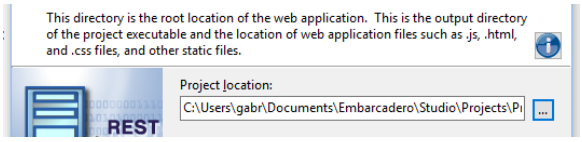
Pri izbiri komponent, ki naj jih naredi čarovnik, izberemo še *Server Module*. Nanj bomo postavili tiste komponente za delo z bazo, ki jih bomo potrebovali le v enem izvodu. Pri izdelavi resnične aplikacije bi verjetno ugasnili *Sample Methods* in *Sample Web Files*, v testnem primeru pa nam bodo prišli prav.



Osnovna komponenta za izdelavo modulov naj bo TDataModule.



Na koncu moramo izbrati mapo, kamor bo čarovnik shranil datoteke, povezane s projektom.



V množici projektnih datotek, ki jih naredi čarovnik, poiščemo enoto *ServerMethodsUnit1*, ki vsebuje data modul *TServerMethods1*. Ime tega modula se pojavi tudi v URL-ju (primer bomo videli kasneje), s katerim dostopamo do strežnika REST, zato data modul preimenujemo v nekaj bolj projektu primernega, denimo *MileageAPI*.

Program shranimo, prevedemo in poženemo (zaenkrat kar na Windows). Pojavi se enostaven menu, v katerem z ukazom *start* poženemo strežnik.

```
H:\clanki\predavanja\Marand\Embarcadero akademija\Delphi and Linux\koda
Enter a Command:
   - "start" to start the server
   - "stop" to stop the server
   - "set port" to change the default port
   - "status" for Server status
   - "help" to show commands
   - "exit" to close the application
->start
- Starting HTTP Server on port 8080
->
```

Za preizkus odpremo brskalnik in se odpravimo na naslov http://localhost:8080. Pojavi se testna spletna stran.

S klikom na *Server Functions* si lahko ogledamo vmesnike REST, implementirane v strežniku.

Testiranje strežnika bo hitrejše in bolj enostavno (sploh na Linuxu, kjer v razhroščevalniku ne moremo komunicirati s programom), če predelamo strežnik, tako da bo ob zagonu samodejno pognal strežniški del (kot da bi uporabnik natipkal *start*). Zato v glavnem programu (.dpr) tik pred zanko, ki komunicira z uporabnikom, pokličemo podprogram *StartServer*.

```
Page | ServerMethodsUnit1 | WebModuleUnit1 | ServerConst1 | koda2
begin
  WriteCommands;
  LServer := TIdHTTPWebBrokerBridge.Create(nil);
  try
    LServer.DefaultPort := APort;
    StartServer(LServer);
    while True do
    begin
      Readln(LResponse);
      LResponse := LowerCase(LResponse);
      if LResponse.StartsWith(cCommandSetPort) then
        SetPort(LServer, LResponse)
      else if sametext(LResponse, cCommandStart) then
```

Linux bi moral biti že na voljo med platformami, le aktiviran ni. Aktivirajmo ga. Nato z desno tipko kliknemo na profil, izberemo *Edit Connection* in nato *Test connection*, da preverimo povezavo do PAServer.

Strežnik spet preizkusimo v brskalniku. Povežemo se na ime virtualnega računalnika ali njegov IP naslov in vrata 8080.

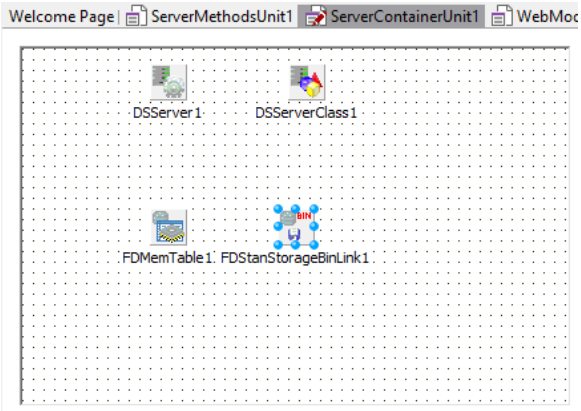
Opazimo lahko, da manjka možnost *Server function invoker*. Razlog za to je metoda *TWebModule1.AllowServerFunctionInvoker* iz enote *WebModuleUnit1*, ki to izbiro prikaže le, če se na strežnik povežemo iz lokalnega naslova.

```
function TWebModule1.AllowServerFunctionInvoker: Boolean;
begin
  Result := (Request.RemoteAddr = '127.0.0.1') or
    (Request.RemoteAddr = '0:0:0:0:0:0:0:1') or (Request.RemoteAddr = '::1');
end;
```

Zelo enostavno jo »popravimo«, tako da njeno vsebino spremenimo v *Result := true;*.

Baza

Za delo z bazo bomo zaradi enostavnosti uporabili kar pomnilniško tabelo *TFDMemTable*. Postavimo jo na data modul v enoti *ServerContainerUnit1*, ker bo v pognanem programu obstajala le ena instanca (en primerek) tega modula.



V tabeli definiramo naslednja polja:

- Timestamp: ftDateTime
- Mileage: ftInteger
- Volume: ftFloat

V prvem bomo shranili datum in čas zapisa, v drugem trenutno kilometrino avtomobila in v tretjem porabo goriva od zadnjega zapisa.

Na isti data modul damo še komponento *TFDStanStorageBinLink*, ki nam bo omogočila branje in shranjevanje baze na disk.

Napišemo enostavna podprograma za branje in shranjevanje baze.

```
procedure TServerContainer1.LoadMemTable;
begin
  if FileExists(FDataFile) then
    FDMemTable1.LoadFromFile(FDataFile);
  FDMemTable1.Open;
end;

procedure TServerContainer1.SaveMemTable;
begin
  ForceDirectories(ExtractFilePath(FDataFile));
  FDMemTable1.SaveToFile(FDataFile);
end;
```

Pot do baze imamo shranjeno v polju *FDataFile*, ki jo v konstruktorju nastavimo na operacijskemu sistemu prilagojen niz *CDataFile*.

```
const
  CDataFile = {$IFDEF LINUX}'data/mileage.fds'{$ELSE}
                            'data\mileage.fds'{$ENDIF};

constructor TServerContainer1.Create(AOwner: TComponent);
begin
  inherited;
  FDSServer := DSServer1;
  FDataFile := CDataFile;
  {$IFNDEF Linux}
  FDataFile := ExpandFileName(FDataFile);
  {$ENDIF}
  LoadMemTable;
end;
```

Na Linuxu moramo spremeniti relativno potko v absolutno, sicer podprogram *ForceDirectories* ne dela pravilno.

Na koncu konstruktorja naložimo začetno stanje baze iz diska.

Baza je sicer skupna, metode, ki bodo izpostavljene čez vmesnik REST, pa se bodo izvajale v ločenih nitih. Lahko se jih bo pognalo tudi več hkrati. Ker imamo primitivno »bazo«, ki ne omogoča sočasnih povezav, nam ne ostane drugega, kot da vsak REST klic bazo začasno zaklene za lastno uporabo.

Potrebujemo dve javni funkciji. Prva zaklene kritično sekcijo, ki ureja dostop do baze, in vrne interno komponento. Druga shrani trenutno stanje baze na disk in odklene kritično sekcijo.

```
var
  FMemTableLock: TCriticalSection;

function TServerContainer1.GetMemTable: TFDMemTable;
begin
  FMemTableLock.Acquire;
  Result := FDMemTable1;
end;

procedure TServerContainer1.ReleaseMemTable;
begin
  SaveMemTable;
  FMemTableLock.Release;
end;
```

V seznam uporabljenih enot moramo dodati *System.SyncObjs.* Kritično sekcijo moramo še kreirati (v konstruktorju *TServerContainer1*) …

```
FMemTableLock := TCriticalSection.Create;
```

… in uničiti (v destruktorju).

```
FreeAndNil(FMemTableLock);
```

API

Sedaj napišemo REST vmesnik za delo z bazo. Te funkcije dodamo v *ServerMethodsUnit1*, iz katerega bomo morali nekako priti do instance razreda *TServerContainer1*. Zato v *ServerContainerUnit1* dodamo globalno funkcijo, ki vrne (edino) instanco.

```
function ServerDM: TServerContainer1;
begin
  Result := FModule as TServerContainer1;
end;
```

V *ServerMethodsUnit1* dodamo metodo *AddDataPoint*, ki v bazo zapiše novo meritev.

```
function TMileageAPI.AddDataPoint(Timestamp: TDateTime; Mileage: integer;
  Volume: double): string;
var
  memTable: TFDMemTable;
begin
  memTable := ServerDM.GetMemTable;
  try
    memTable.Append;
    try
      memTable.FieldByName('Timestamp').AsDateTime := Timestamp;
      memTable.FieldByName('Mileage').AsInteger := Mileage;
      memTable.FieldByName('Volume').AsFloat := Volume;
    finally
      memTable.CommitUpdates;
    end;
  finally ServerDM.ReleaseMemTable; end;
end;
```

V *uses* moramo dodati *FireDAC.Comp.Client* in *ServerContainerUnit1*.

Bodite pozorni na to, da mora biti tip realnih podatkov *double* in ne *real*. Če parameter označimo kot *real*, strežnik funkcije (brez kakršnekoli pritožbe) ne bo naredil dostopne čez vmesnik REST.

Za preizkus poženemo strežnik v okolju Windows, se z brskalnikom povežemo na *localhost:8080* in v izbiri *Server Functions* preizkusimo klic metode *AddDataPoint*. Lahko pa tudi celotno zahtevo zapakiramo v en sam URL:

http://localhost:8080/datasnap/rest/TMileageAPI/AddDataPoint/2017-09-17/234/345

Potrebujemo še funkcijo, ki bo vrnila vsebino tabele. Prenesli jo bomo kar kot JSON data set.

Na data modul v *ServerContainerUnit1* dodamo komponento *TFDStanStorageJSONLink*. Dodali bi jo lahko kamorkoli, saj je pomembno le, da se v projekt doda njena enota, ki v inicializaciji registrira komponento.

Sedaj lahko v enoti *ServerMethodsUnit1* definiramo funkcijo *GetDataPoints*.

```
function TMileageAPI.GetDataPoints: TFDJSONDataSets;
var
  memTable: TFDMemTable;
begin
  memTable := ServerDM.GetMemTable;
  try
    Result := TFDJSONDataSets.Create;
    TFDJSONDataSetsWriter.ListAdd(Result, memTable);
  finally ServerDM.ReleaseMemTable; end;
end;
```

V seznam *uses* moramo dodati še enoto *Data.FireDACJSONReflect*.

Program lahko sedaj prevedemo za Windows, če pa platformo spremenimo v Linux, se prevajalnik pritoži.

[DCC Fatal Error] ServerMethodsUnit1.pas(7): F2613 Unit 'Data.FireDACJSONReflect' not found.

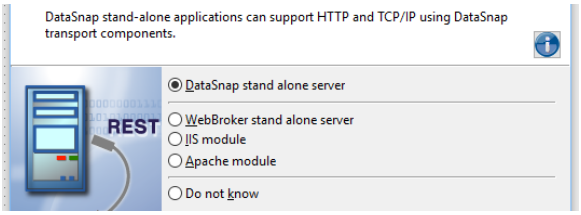
Prevajalnik iz neznanega razloga ne najde enot iz imenskega prostora *Data.* Na dokaj grd način lahko problem zaobidemo, tako da v glavnem programu .dpr navedemo polne poti do uporabljenih enot.

```
  {$IFDEF Linux}
  Data.FireDACJSONConsts in 'c:\Program Files
(x86)\Embarcadero\Studio\19.0\source\data\datasnap\Data.FireDACJSONConsts.p
as',
  Data.FireDACJSONReflect in 'c:\Program Files
(x86)\Embarcadero\Studio\19.0\source\data\datasnap\Data.FireDACJSONReflect.
pas',
  {$ENDIF Linux}
```

Odjemalec

Potrebujemo še odjemalski program. Najprej izdelamo multiplatformno aplikacijo (*File, New, Multi-Device Application, Blank Application*). Nato v njo dodamo DataSnap REST Client (*File, New, Other, DataSnap Server, DataSnap REST Client Module*).

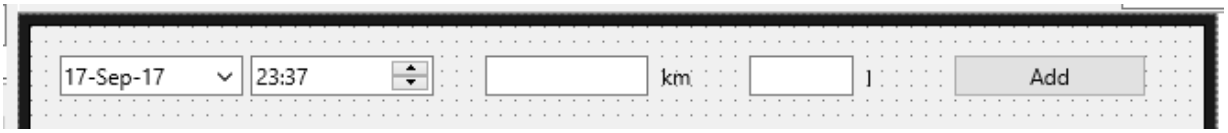
Pojavi se čarovnik, v katerem izberemo, da je naš strežnik samostoječ. Vse ostale možnosti pustimo na privzetih vrednostih.



Sedaj poženemo naš strežnik. Potrebujemo ga, da bo lahko odjemalec samodejno izdelal REST vmesnike. V odjemalcu odpremo enoto *ClientModuleUnit1*, z desno tipko kliknemo komponento *DSRestConnection1* in iz menuja izberemo *Generate DataSnap client classes*. S tem dobimo v projekt enoto *ClientClassesUnit1*, ki vsebuje (med drugim) funkcije *AddDataPoint* in *GetDataPoints*.

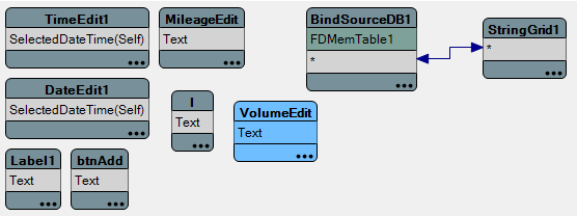
```
type
  TMileageAPIClient = class(TDSAdminRestClient)
  public
    function EchoString(Value: string;
      const ARequestFilter: string = ''): string;
    function ReverseString(Value: string;
      const ARequestFilter: string = ''): string;
    function AddDataPoint(Timestamp: TDateTime; Mileage: Integer;
      Volume: Double; const ARequestFilter: string = ''): string;
    function GetDataPoints(const ARequestFilter: string = ''):
      TFDJSONDataSets;
  end;
```

Na glavno formo postavimo nekaj komponent in napišemo grdo kodo, ki brez vsakega preverjanja pravilnosti podatkov vpisuje nove meritve v bazo.



```
procedure TfrmMileageClient.btnAddClick(Sender: TObject);
begin
  ClientModule1.MileageAPIClient.AddDataPoint(
    DateEdit1.Date + TimeEdit1.Time,
    StrToInt(MileageEdit.Text), StrToFloat(VolumeEdit.Text));
end;
```

V seznam *uses* moramo dodati *ClientModuleUnit1.*



Za prikaz podatkov bomo uporabili kar *TStringGrid*, povezan na *TFDMemTable.* Oba gradnika postavimo na formo, odpremo *LiveBindings Designer* in ju povežemo.

Na formo moramo dodati še komponenti *TFDStanStorageBinLink* in *TFDStanStorageJSONLink*, ki bosta omogočili branje podatkov iz JSON data seta, ki ga vrne *GetDataPoints*.

Napišemo metodo *RefreshData*, ki pokliče REST API funkcijo *GetDataPoints*, iz vrnjenega JSON data seta pobere prvo (in edino) tabelo in jo skopira v *FDMemTable1*.

```
procedure TfrmMileageClient.RefreshData;
var
  ds: TFDJSONDataSets;
begin
  FDMemTable1.Close;
  ds := ClientModule1.MileageAPIClient.GetDataPoints;
  FDMemTable1.AppendData(TFDJSONDataSetsReader.GetListValue(ds, 0));
  FDMemTable1.Open;
end;
```

Pokličemo jo iz metode *btnAddClick*, da po dodajanju meritve posodobi stanje, in na začetku, da prebere začetno stanje. Ne moremo je poklicati iz dogodka *OnCreate*, ker takrat odjemalec DataSnap še ni dokončno inicializiran, zato uporabimo *OnShow*.

```
procedure TfrmMain.btnAddClick(Sender: TObject);
begin
  ClientModule1.MileageAPIClient.AddDataPoint(
    DateEdit1.Date + TimeEdit1.Time,
    StrToInt(MileageEdit.Text), StrToFloat(VolumeEdit.Text));
  RefreshData;
end;

procedure TfrmMain.FormShow(Sender: TObject);
begin
  OnShow := nil;
  RefreshData;
end;
```

V seznam *uses* moramo dodati še *Data.FireDACJSONReflect*.

Program lahko sedaj preizkusimo s strežnikom na gostiteljskem računalniku Windows. Če ga bi radi povezali na virtualni Ubuntu, pa moramo v lastnost *Host* komponente *DSRestConnection1* v enoti *ClientModuleUnit1* vpisati naslov (ali ime) virtualnega računalnika.

Viri

WinSCP

https://winscp.net/eng/download.php

VirtualBox

Download VirtualBox

https://www.virtualbox.org/wiki/Downloads

Installing VirtualBox and extension packs

https://www.virtualbox.org/manual/ch01.html#intro-installing

Licensing: Frequently Asked Questions

https://www.virtualbox.org/wiki/Licensing_FAQ

Ubuntu

Download Ubuntu Server

https://www.ubuntu.com/download/server

Download Ubuntu Desktop

https://www.ubuntu.com/download/desktop

Configure Delphi and RedHat or Ubuntu for Linux development

http://chapmanworld.com/2016/12/29/configure-delphi-and-redhat-or-ubuntu-for-linux-development/

Installing Ubuntu 16.04 LTS on VMWare Fusion for Linux Development with Upcoming Delphi 10.2 - Part 1

https://community.embarcadero.com/blogs/entry/installing-ubuntu-16-04-on-vmware-for-delphi-dev

Installing Ubuntu 16.04 LTS on VMWare Fusion for Linux Development with Upcoming Delphi 10.2 - Part 2

https://community.embarcadero.com/blogs/entry/preparing-ubuntu-16-04-lts-for-linux-development-with-upcoming-delphi-10-2-part-2

How To Install Ubuntu Linux Alongside Windows 10 (UEFI)

http://www.everydaylinuxuser.com/2015/11/how-to-install-ubuntu-linux-alongside.html

How do I modify or disable the HUD's use of the Alt key?

https://askubuntu.com/questions/122209/how-do-i-modify-or-disable-the-huds-use-of-the-alt-key

How do I change global hotkeys / keyboard shortcuts?

https://askubuntu.com/questions/85591/how-do-i-change-global-hotkeys-keyboard-shortcuts

How to Add and Delete Users on Ubuntu 16.04

`https://www.digitalocean.com/community/tutorials/how-to-add-and-delete-users-on-ubuntu-16-04`

How to Install and Configure VNC on Ubuntu 16.04

`https://www.digitalocean.com/community/tutorials/how-to-install-and-configure-vnc-on-ubuntu-16-04`

How To Setup A Ubuntu Remote Desktop

`https://www.lifewire.com/setup-ubuntu-remote-desktop-4129666`

How to launch application as root from Unity Launcher?

`https://askubuntu.com/questions/118822/how-to-launch-application-as-root-from-unity-launcher`

How do I change the color for directories with ls in the console?

https://askubuntu.com/questions/466198/how-do-i-change-the-color-for-directories-with-ls-in-the-console

What do the different colors mean in ls?

`https://askubuntu.com/questions/17299/what-do-the-different-colors-mean-in-ls`

Samba Server installation on Ubuntu 16.04 LTS

`https://www.howtoforge.com/tutorial/samba-server-ubuntu-16-04/`

Websites Not Loading (dns issue) After Installing Ubuntu 17.04? [fix]

https://www.hecticgeek.com/2017/04/ubuntu-17-04-systemd-dns-issues/

Ubuntu 17.04 systemd-resolved DNS lookups randomly fail

https://superuser.com/questions/1153203/ubuntu-17-04-systemd-resolved-dns-lookups-randomly-fail

How to disable systemd-resolved in Ubuntu?

`https://askubuntu.com/questions/907246/how-to-disable-systemd-resolved-in-ubuntu`

Windows Subsystem for Linux

Windows Subsystem for Linux Documentation

`https://msdn.microsoft.com/en-us/commandline/wsl/about`

How to Install and Use the Linux Bash Shell on Windows 10

`https://www.howtogeek.com/249966/how-to-install-and-use-the-linux-bash-shell-on-windows-10/`

Where is the Ubuntu file system root directory in Windows NT subsystem and vice versa?

`https://askubuntu.com/questions/759880/where-is-the-ubuntu-file-system-root-directory-in-windows-nt-subsystem-and-vice`

Delphi in Linux

Linux Application Development

http://docwiki.embarcadero.com/RADStudio/en/Linux_Application_Development

Supported Target Platforms

http://docwiki.embarcadero.com/RADStudio/Tokyo/en/Supported_Target_Platforms

DataSnap Server Wizard for Linux

http://docwiki.embarcadero.com/RADStudio/en/DataSnap_Server_Wizard_for_Linux

DataSnap REST Application Wizard for Linux

http://docwiki.embarcadero.com/RADStudio/en/DataSnap_REST_Application_Wizard_for_Linux

Delphi for Linux Database and Web Development

http://blog.marcocantu.com/blog/2017-march-delphi-linux-database-web.html

Deploying on OS X (FireDAC)

http://docwiki.embarcadero.com/RADStudio/en/Deploying_on_OS_X_(FireDAC)

Building a (real) Linux daemon with Delphi - Part 1

http://blog.paolorossi.net/2017/07/11/building-a-real-linux-daemon-with-delphi-part-1-2/

Building a (real) Linux daemon with Delphi - Part 2

http://blog.paolorossi.net/2017/09/04/building-a-real-linux-daemon-with-delphi-part-2/

FireMonkey implementation for Linux

http://fmxlinux.com/

Build VCL application for macOS and Linux

http://www.crossvcl.com/

FireDAC

Tutorial: Using a REST DataSnap Server with an Application and FireDAC

http://docwiki.embarcadero.com/RADStudio/en/Tutorial:_Using_a_REST_DataSnap_Server_with_an_Application_and_FireDAC

Databases (FireDAC)

http://docwiki.embarcadero.com/RADStudio/en/Databases_(FireDAC)

YouTube

Delphi Linux Server Support in RAD Studio 10.2

https://www.youtube.com/watch?v=2wiD3F-nGZ8

Ubuntu Install Guide for Delphi on Linux Beta

https://www.youtube.com/watch?v=QZ7aGlkat64

Configuring Delphi for Linux Deployment

https://www.youtube.com/watch?v=l-nnF5o3bhU

Size of a Pointer on a 64-Bit Linux Console App with Delphi

https://www.youtube.com/watch?v=e5LzQvw9EF4

Data Access with a Linux Console App in Delphi

https://www.youtube.com/watch?v=fk1K8bPtzk0

Creating a DataSnap REST Application on Linux with Delphi

https://www.youtube.com/watch?v=dmC99I315uc

Building a Standalone WebBroker Server for Linux

https://www.youtube.com/watch?v=9QW72sNUFAA

Building and Deploying a Linux Apache Module with Delphi WebBroker

https://www.youtube.com/watch?v=z0nNRNQSxKI

Accessing Linux APIs and Command-Line Interface from Delphi

https://www.youtube.com/watch?v=4gDPqq8H-xw