Distribucija softvera na Debian GNU/Linux platformi

Benutić, Ivan

SVEUČILIŠTE U ZAGREBU

FAKULTET ORGANIZACIJE I INFORMATIKE

V A R A Ž D I N

Ivan Benutić

Distribucija softvera na Debian GNU/Linux platformi

ZAVRŠNI RAD

Varaždin, 2022.

SVEUČILIŠTE U ZAGREBU

FAKULTET ORGANIZACIJE I INFORMATIKE

V A R A Ž D I N

Ivan Benutić

Studij: Poslovni sustavi

Distribucija softvera na Debian GNU/Linux platformi

ZAVRŠNI RAD

Mentor:

Doc. dr. sc. Marcel Maretić

Varaždin, rujan 2022.

Ivan Benutić

Izjava o izvornosti

Izjavljujem da je moj završni/diplomski rad izvorni rezultat mojeg rada te da se u izradi istoga nisam koristio drugim izvorima osim onima koji su u njemu navedeni. Za izradu rada su korištene etički prikladne i prihvatljive metode i tehnike rada.

Autor/Autorica potvrdio/potvrdila prihvaćanjem odredbi u sustavu FOI-radovi

Sažetak

Cilj rada je prikazati razlike među sustavima za pakiranje i distribuciju aplikacijskih paketa na GNU/Linux sustavima. Od tradicionalnih rješenja poput deb i rpm paketa do najmodernijih sustava i standarda poput Snap, Flatpak i AppImage paketa. Nakon okvirnog pregleda glavnih sustava za pakiranje i distribuciju aplikacija slijedi prikaz izrade aplikacije izrađene u kombinaciji ReactJS i ElectronJS programskih okvira te pakiranje aplikacije u deb, rpm i Flatpak formate. Zatim je prikazan proces izrade lokalnih Yum i APT repozitorija kao i mogući procesi distribucije Flatpak paketa. Na kraju u novonastale repozitorije dodaju se izrađeni aplikacijski paketi te se prikazuje proces dodavanja novog repozitorija te instalacije aplikacije.

Ključne riječi: Linux; Flatpak; Snap; AppImage; Aplikacija; ElectronJS; Pakiranje; Distribucija;

Sadržaj

1. Uvod

Ovaj završni rad bavi se problematikom pakiranja i distribucije aplikacija na GNU/Linux operativnim sustavima poput Ubuntua, Debiana, Fedore, Archa i sličnih. Linux zajednica neprestano pokušava pojednostaviti i poboljšati svaki aspekt navedenih operativnih sustava i njihovih elemenata, a proteklih nekoliko godina obilježeno je naglim prihvaćanjem novih načina pakiranja i distribucije aplikacija koji rješavaju probleme prisutne već 30 godina.

Moderni načini pakiranja, održavanja, distribucije i kontrole aplikacijskih paketa uvelike su pomogli projektima poput igraće konzole Steam Deck i njenog operativnog sustava SteamOS baziranog na Arch Linuxu. Zajednica koja inače sporo prihvaća tako fundamentalne promjene poput upravitelja aplikacijskih paketa nadglasana je novom generacijom korisnika i Linux influencera željnih novih tehnologija koje pojednostavljuju prelazak novih korisnika na Linux operativne sustave.

Ne može se zanemariti ni utjecaj Covid-19 pandemije u proteklim godinama koja je potakla ljude da eksperimentiraju s Linux operativnim sustavima te odmah prihvate nove standarde poput Snap i Flatpak paketa, ali i novih standarda nevezanih za aplikacije poput Waylanda i Pipewirea. Udio Linuxa na tržištu desktop operativnih sustava je u trenutku pisanja 2.75% („Statcounter", bez dat.**)** u odnosu na udio od 1.6% u prosincu 2019. Godine („Netmarketshare", bez dat.).

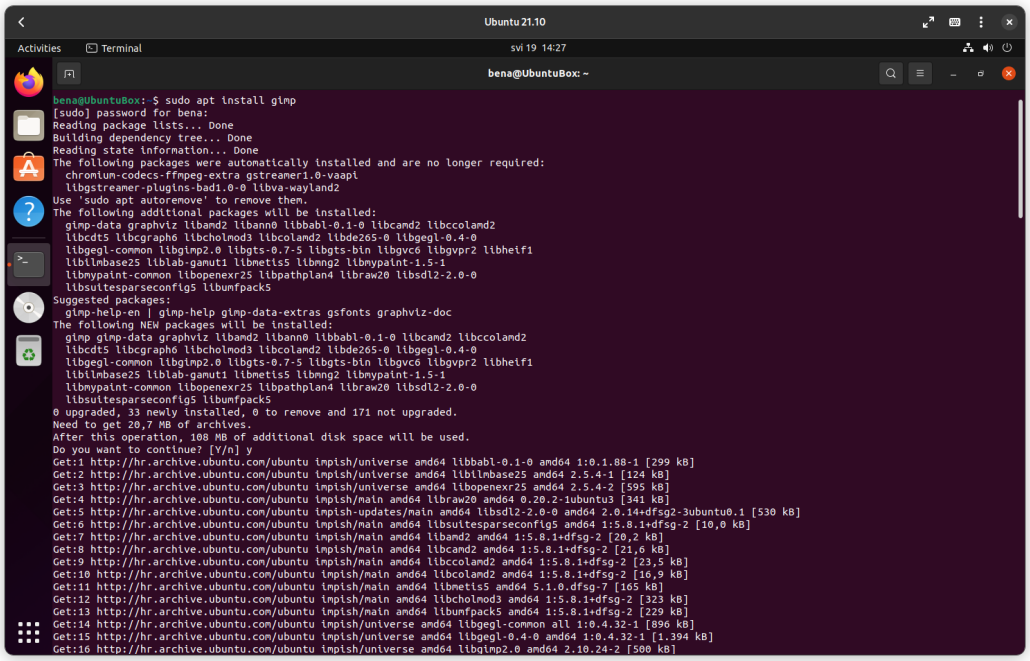
Ovaj rad će objasniti razlike među različitim tradicionalnim i modernim rješenjima za distribuciju aplikacija i aplikacijskih paketa. U sklopu rada također je izrađena i aplikacija na čijem primjeru će se prikazati postupak pakiranja i distribucije iste.

2. Problemi distribucije aplikacija na GNU/Linuxu

Prvi sustavi za distribuciju aplikacija na GNU/Linux sustavima stvoreni su još 1993. godine. Mnogi upravitelji paketima koje koristimo danas također su nastali 90-ih godina prošlog stoljeća kao na primjer RedHat Package Manager (RPM) korišten na RedHat Linuxu, CentOSu i Fedori te Advanced Package Manager korišten na Debianu i distribucijama Linuxa baziranim na Debianu poput Ubuntua.

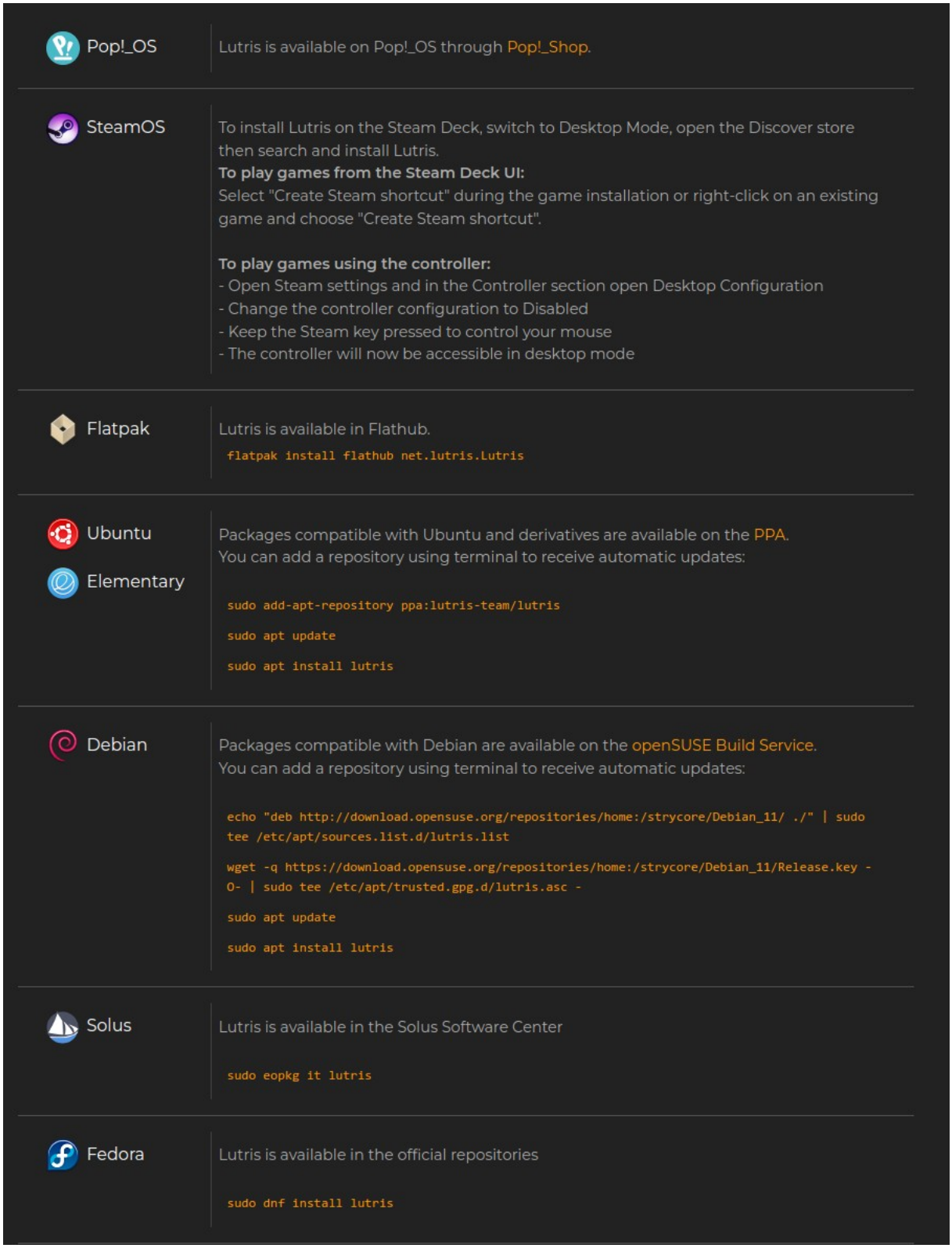
Tradicionalne načine pakiranja koji su vezani za pojedinu distribuciju Linuxa također nazivamo i izvorni (engl. native) načini pakiranja. Izvorni upravitelji paketima poput APT-a oslanjaju se na repozitorije stvorene za pojedine distribucije. Tako da svaki paket mora biti upakiran, testiran i dostupan za svaku verziju svake distribucije na svakoj procesorskoj arhitekturi zasebno. Na primjer, paket dostupan za Ubuntu 16.04 nije dostupan na Ubuntu 21.10. S obzirom na to da paketi ovise o drugim paketima i kompleksne aplikacije ovise o mnogo drugih manjih paketa, nedostatak bilo kojeg od tih manjih paketa onemogućuje rad aplikacije na drugim verzijama distribucije.

Na Slici 1 vidljiv je proces instalacije aplikacije GIMP na Ubuntu sustavu. Osim paketa „gimp“ APT nam kaže da će instalirati još 32 paketa koji su potrebni za rad aplikacije Gimp. Ti paketi moraju biti dostupni na repozitorijima kako bi aplikacija Gimp mogla funkcionirati.



Slika 1: instalacija GIMP-a (autorski rad)

Određene je aplikacije zbog toga vrlo teško instalirati na određene distribucije kao što je vidljivo na primjeru aplikacije Lutris na Slici 2. Moderni načini distribucije aplikacija rješavaju mnoge od ovih problema o čemu će biti riječi kasnije u ovom radu.



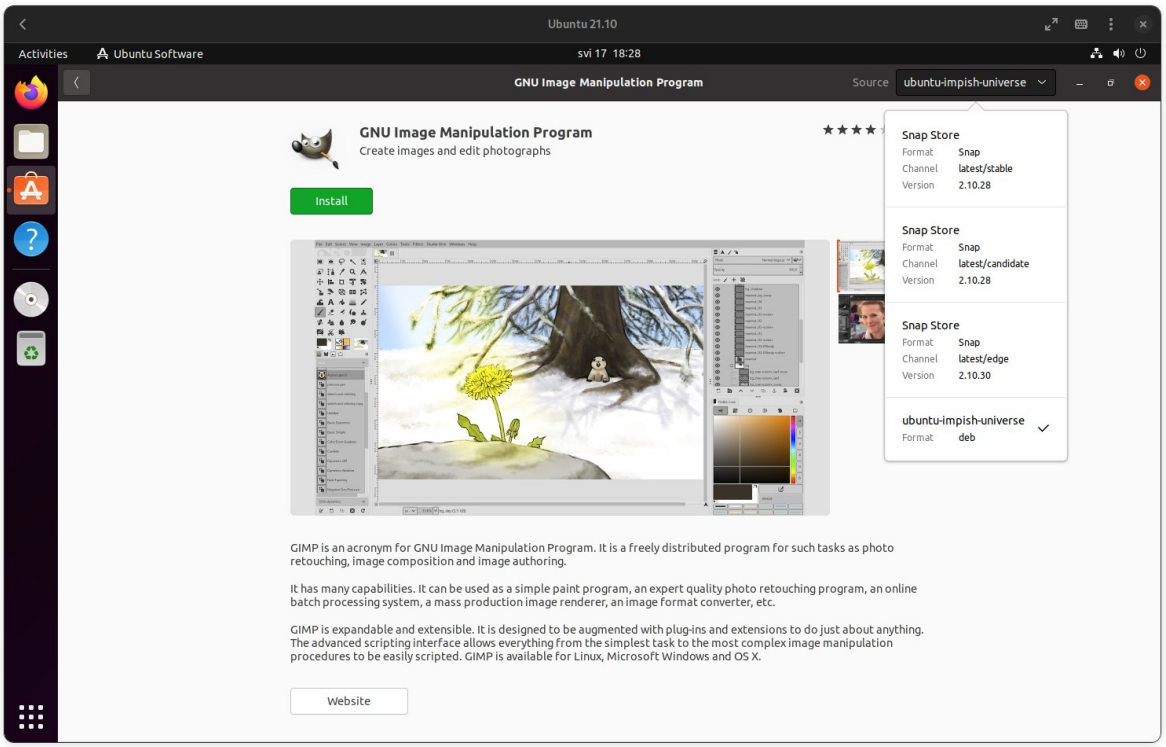
Slika 2: Upute za instalaciju aplikacije Lutris (lutris.net)

3. Nativni načini distribucije aplikacija

Problemi izvornih načina distribucije pokriveni su u prijašnjem poglavlju ovog rada, a u ovom ćemo poglavlju pobliže objasniti način rada nekih od najčešćih izvornih načina distribucije.

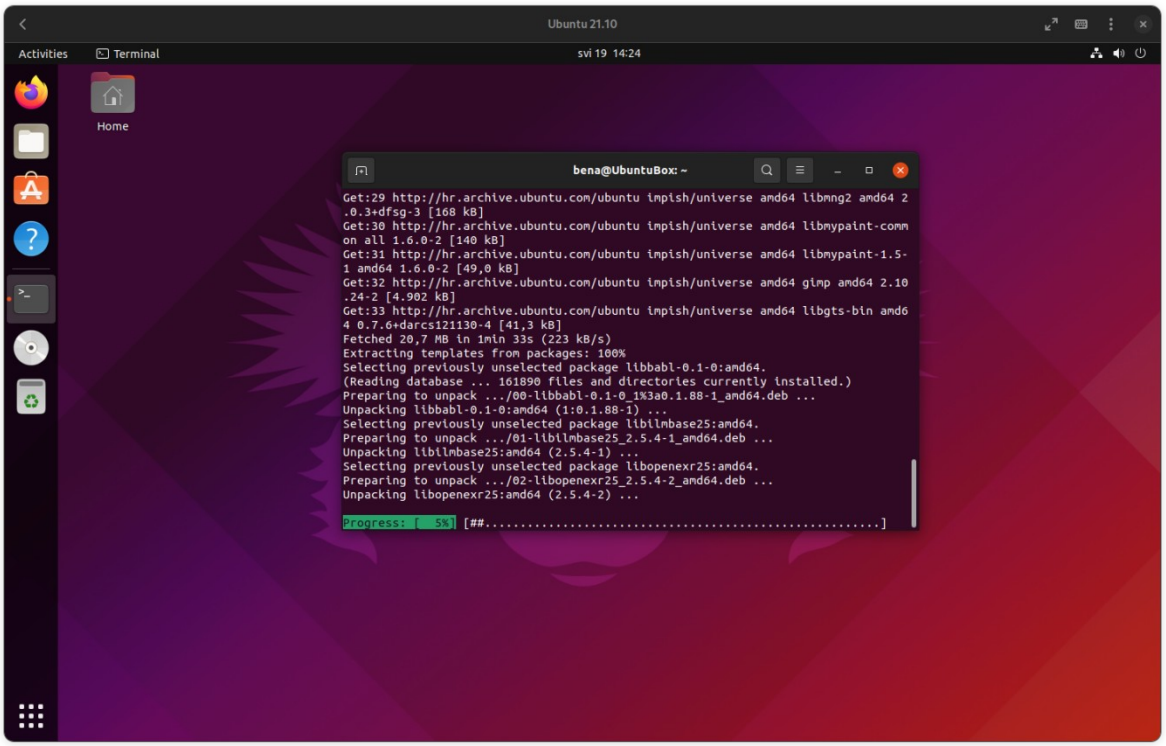
3.1. Advanced Package Manager (APT) i Debian package (DEB)

Advanced Package Manager (APT) alat je za instalaciju, ažuriranje i deinstalaciju Debian Software Package (DEB) datoteka. Koristi se na distribucijama Linuxa baziranim na Debian Linuxu poput Ubuntua. Na Ubuntu APT se može koristiti kroz grafičko sučelje putem aplikacija poput Ubuntu Software-a.



Slika 3: Ubuntu Software (autorski rad)

APT se također može koristiti i kroz naredbeno sučelje (engl. *Command line*).



Slika 4: APT u naredbenom sučelju (autorski rad)

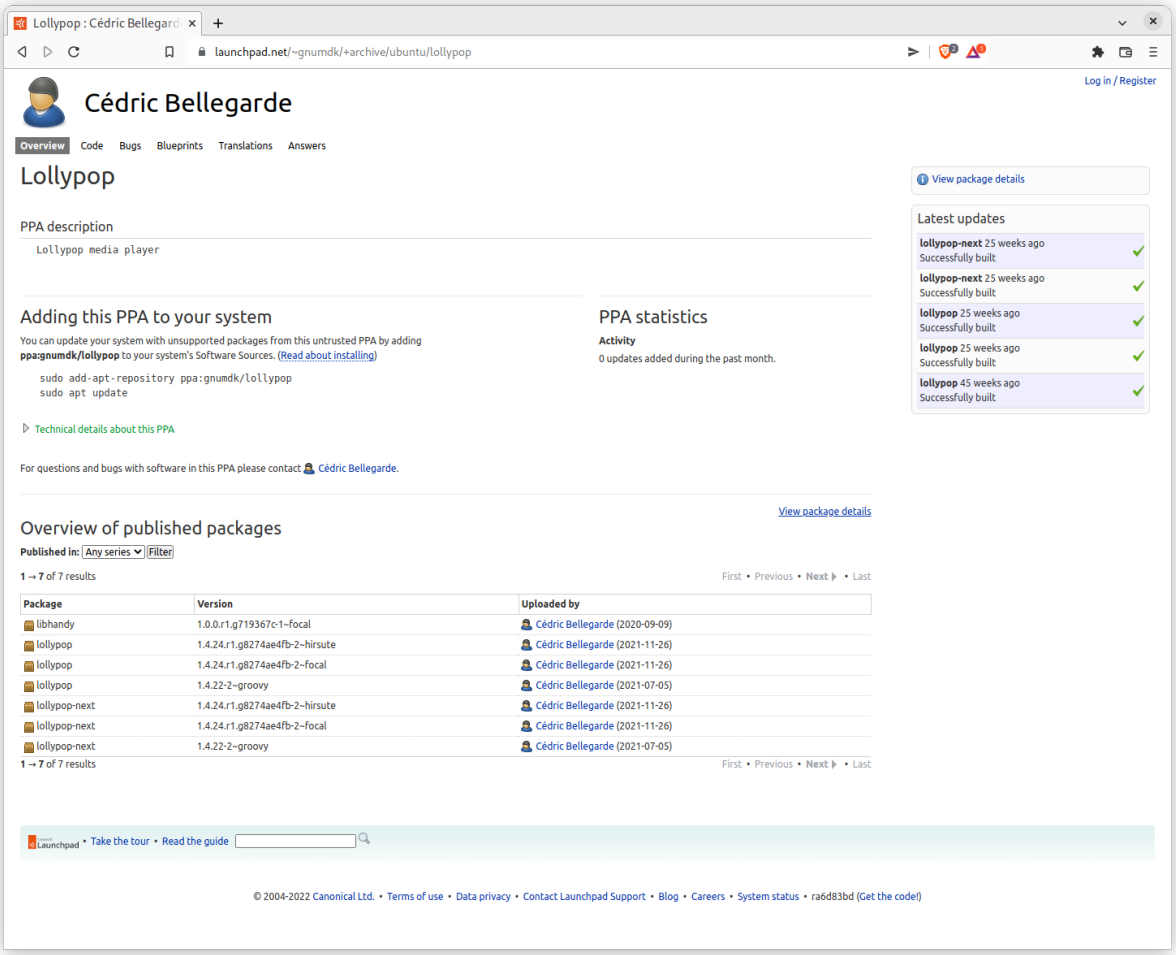
Ubuntu koristi nekoliko kategorija repozitorija.

- Main – Službeno podržani paketi otvorenog koda
- Universe – Paketi otvorenog koda koje održava zajednica
- Restricted – paketi zatvorenog koda, najčešće pokretački programi (engl. *drivers*) za komponente
- Multiverse – paketi koji trenutno imaju legalnih problema

Osim službenih repozitorija koje održavaju developeri distribucije, servisi poput Launchpad.net omogućuju izradu osobnih arhiva paketa (engl. Personal package archive, PPA). PPA omogućuje developerima aplikacija da neovisno o distribucijskim repozitorijima održavaju svoju aplikaciju. Na taj je način često moguće doći do novije verzije aplikacije nego što je dostupna na distribucijskom repozitoriju ili do potpuno novih aplikacija.

Postoje službeni i neslužbeni PPA. Službenima upravlja glavni developer aplikacije, dok na neslužbenima bilo tko može držati tuđu aplikaciju otvorenog koda. Neslužbeni PPA

najčešće se koriste kada developer neke aplikacije aplikaciju nije upakirao za, u ovom slučaju, Debianu srodne sustave (Prakash, 2021).



Slika 5: službeni PPA za aplikaciju Lollypop (launchpad.net)

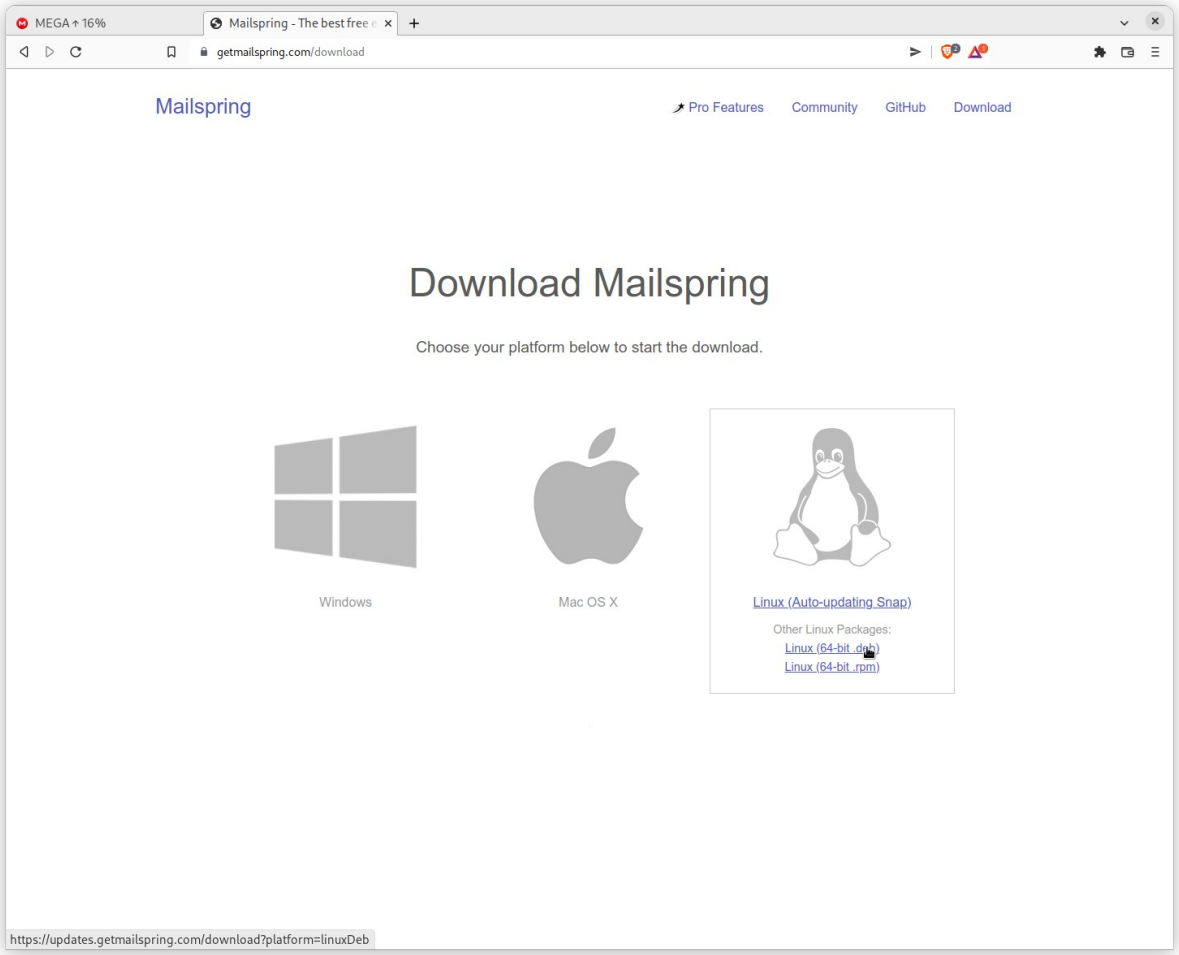
Kako bismo dodali PPA u popis repozitorija možemo pratiti upute na Launchpadu. U slučaju PPA za aplikaciju Lollypop naredba za dodavanje PPA je:

```
sudo add-apt-repository ppa:gnumdk/lollypop && sudo apt update
```

Nakon dodavanja PPA moguće je instalirati paket lollypop naredbom:

```
sudo apt install lollypop –y
```

Osim kroz repozitorije, aplikacije se na Debianu srodnim distribucijama mogu instalirati i direktno putem DEB datoteka. Takvu praksu najčešće vidimo na aplikacijama zatvorenog koda te je vrlo usporediva s načinom na koji se instaliraju aplikacije na Windows operativnom sustavu.



Slika 6: DEB datoteka dostupna na internetu (getmailspring.com)

Ručna instalacija DEB paketa ne pruža benefite poput instalacije drugih potrebnih paketa ili jednostavnog ažuriranja, što je također jedna od sličnosti s instalacijom programa na Windows operativnom sustavu.

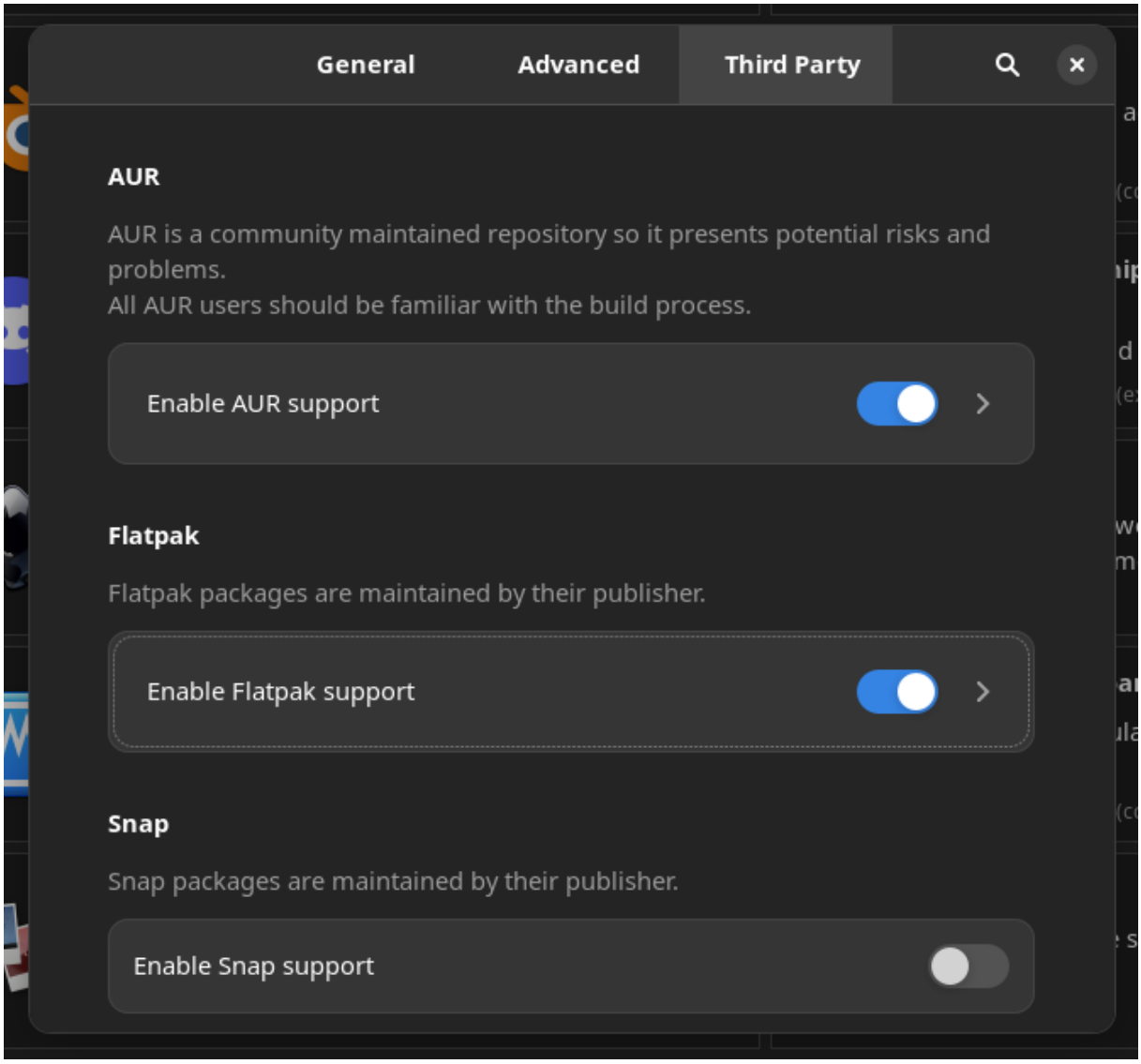
3.2. DNF i RedHat paketi

Dandified Yum poznatiji kao DNF je upravitelj izvornim paketima na svim modernim distribucijama Linuxa baziranima na RedHat Linuxu poput Fedore, Oracle Linuxa, CentOS-a i Rocky Linuxa. Koristi sličan sustav repozitorija kao i APT osim što koristi RedHat Package manager (RPM) pakete umjesto DEB datoteka ("Package management system", bez dat.).

3.3. Arch user repository (AUR)

Arch je Linux distribucija poznata po mnogočemu, a Arch User Repository (AUR) jedna je od najpoznatijih značajki Archa. AUR je jedinstven repozitorij koji sadrži upute za izgradnju aplikacijskih paketa (PKGBUILD), te upute povezuje s izvornim kodom aplikacijskog paketa (engl. *Source code*).

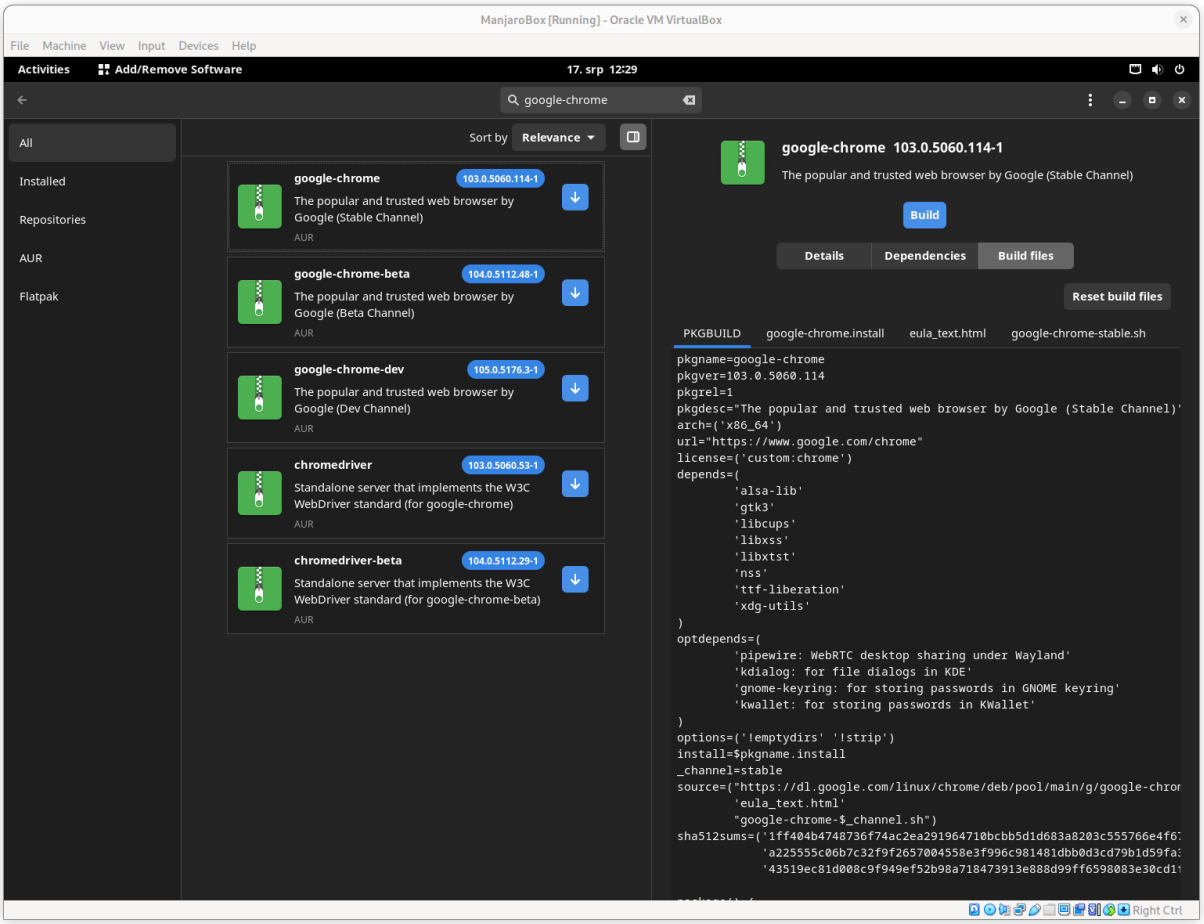
Sadržaj na AUR postavljaju i kontroliraju volonteri i sami korisnici. Iako neslužben, AUR je vrlo bitan dio Arch ekosustava jer (u vrijeme pisanja) sadrži preko 84 000 PKGBUILD paketa ("Arch user repository", bez dat.).



Slika 7: AUR podrška u upravitelju paketima na Manjaro Linuxu (autorski rad)

AUR je skup Git repozitorija na kojima se nalaze upute za izgradnju i poveznice na izvorni kod raznih aplikacija. Oslanja se na Arch Building System (ABS) kako bi interpretirao upute za izgradnju (PKGBUILD).

Ključ fleksibilnosti AUR-a u tome je što repozitoriji ne sadržavaju ni izvorni kod ni izgrađene pakete (engl. *binaries*). Tako se na AUR-u mogu pronaći i programi zatvorenog koda poput Spotify aplikacije i Google Chromea.



Slika 8: AUR PKGBUILD za Google Chrome (autorski rad)

U PKGBUILD datoteci nalazi se opis aplikacije, poveznica na izvorni kod, popis potrebnih paketa te potrebne funkcije i putanje do skripti za izgradnju aplikacije. Izgradnja se zatim odvija na korisničkom računalu pomoću ABS-a ili Pacmana (Robertson, 2021).

4. Moderni načini pakiranja i distribucije paketa

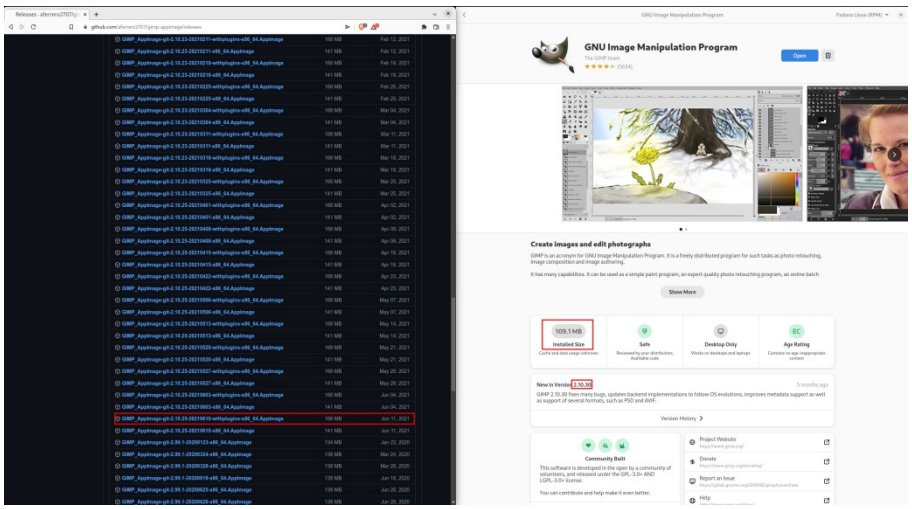
Moderni načini pakiranja i distribucije aplikacijskih paketa nastali su kako bi se riješili već prije spomenuti problemi kompatibilnosti i međuovisnosti paketa o kojima ovisi aplikacija. Svi moderni načini pakiranja i distribucije aplikacijskih paketa imaju jednu zajedničku značajku, a to je međuplatformnost (engl. Cross-platform). Međuplatformne metode rješavaju problem različitih distribucija Linuxa. Zajednica se slaže da su međuplatformni načini distribucije paketa budućnost Linuxa kao osobnog operativnog sustava (Stouff, 2022).

4.1. AppImage

AppImage je format za pakiranje aplikacija kojem je cilj maksimalna kompatibilnost i stabilnost. AppImage paketi uz glavnu aplikaciju pakiraju i sve potrebne pakete o kojima aplikacija ovisi, to AppImage pakete ponekad čini vrlo velikima u odnosu na DEB i RPM ekvivalente.

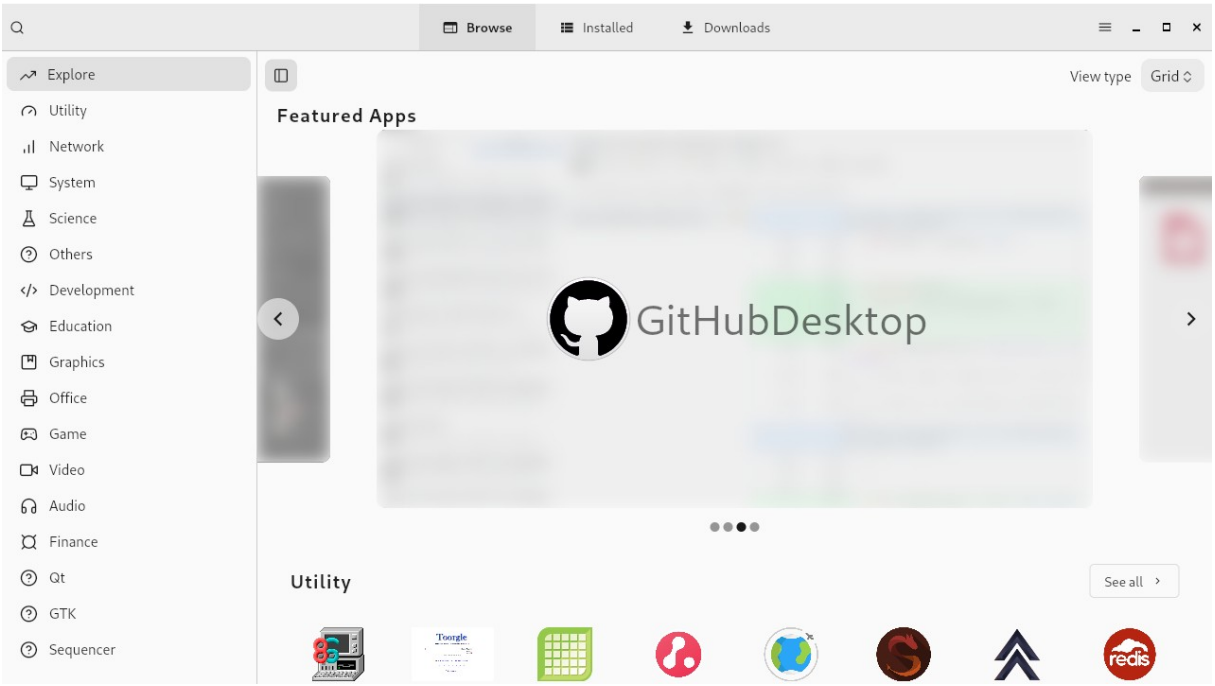
AppImage nema repozitorij ili sustav ažuriranja i samoodržavanja, već se AppImage datoteke skida s web stranica slično kao i portabilne aplikacije na Windows sustavima. Također ne postoji instalacija već se cijela AppImage datoteka pokrene kako bi se koristila aplikacija.

Na primjeru već spomenute aplikacije Gimp možemo vidjeti još jedan čest problem kod AppImage-a, ali i ostalih načina distribucije. AppImage verzija sedam je mjeseci starija od one koja se nudi na Fedorinom RPM repozitoriju. Na Slici 9 također vidimo da je AppImage paket 35% veći od tradicionalnog RPM paketa. Razlika u verziji aplikacije u ovom je slučaju zanemariva.



Slika 9: Razlika u veličini instalacije AppImage - RPM (github.com)

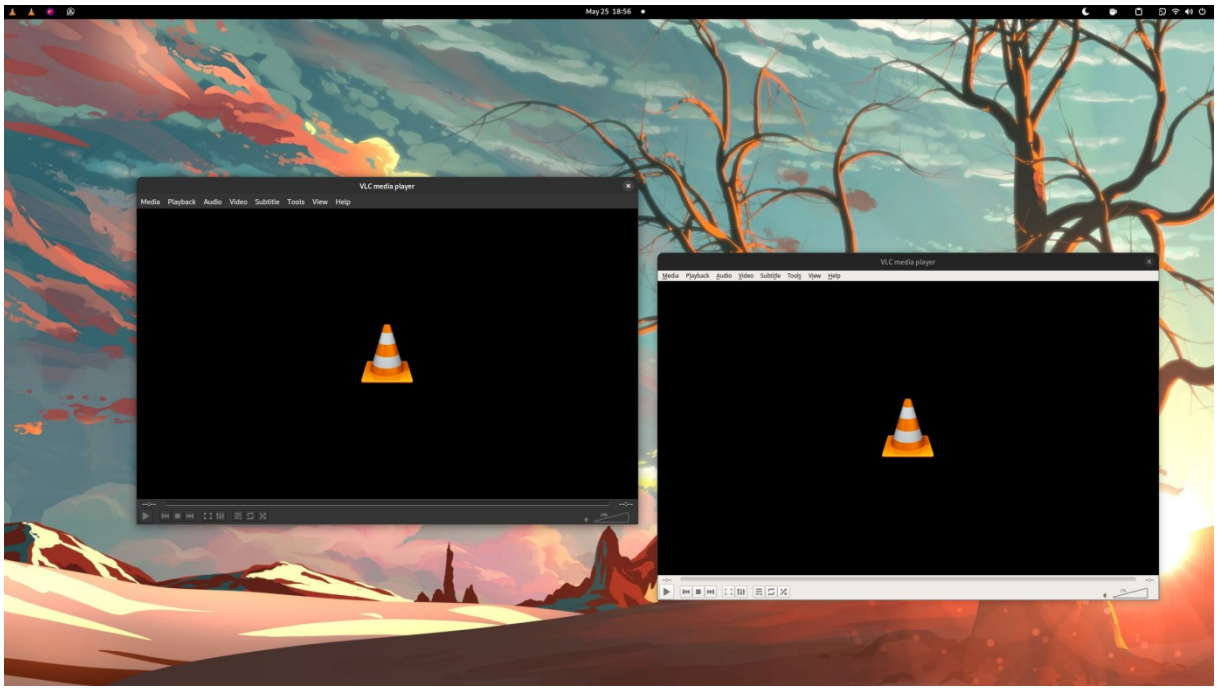
AppImage pakete moguće je pronaći i putem web stranice AppImageHub.com na kojoj se nalaze linkovi i reference za preko 1100 aplikacija u trenutku pisanja (appimagehub.com, bez dat.). Zahvaljujući postojanju AppImageHub-a stvoren je zaseban alat za pronalazak i preuzimanje AppImage paketa AppImage Pool.



Slika 10: AppImage Pool (autorski rad)

AppImage je odličan distribucijski format za držanje aplikacija na prijenosnim medijima poput USB-a ili pokretanje aplikacija u živom okruženju (engl. *Live environment*) jer ne zahtijevaju instalaciju dodatnih paketa. S obzirom na to da je AppImage format potpuno otvorenog tipa (open source) jednako je prihvaćen na svim velikim Linux desktop i server sustavima.

Nažalost AppImage nije općenito prihvaćen kao zadana opcija ni na jednoj od velikih distribucija. AppImage datoteke su velike i jako loše se integriraju u korisničko sučelje kako vizualno tako i procesom ažuriranja i održavanja same aplikacije. Primjer vizualne nekonzistencije vidljiv je na primjeru nepoštivanja korisničke teme i tamnog načina rada na Slici 11.



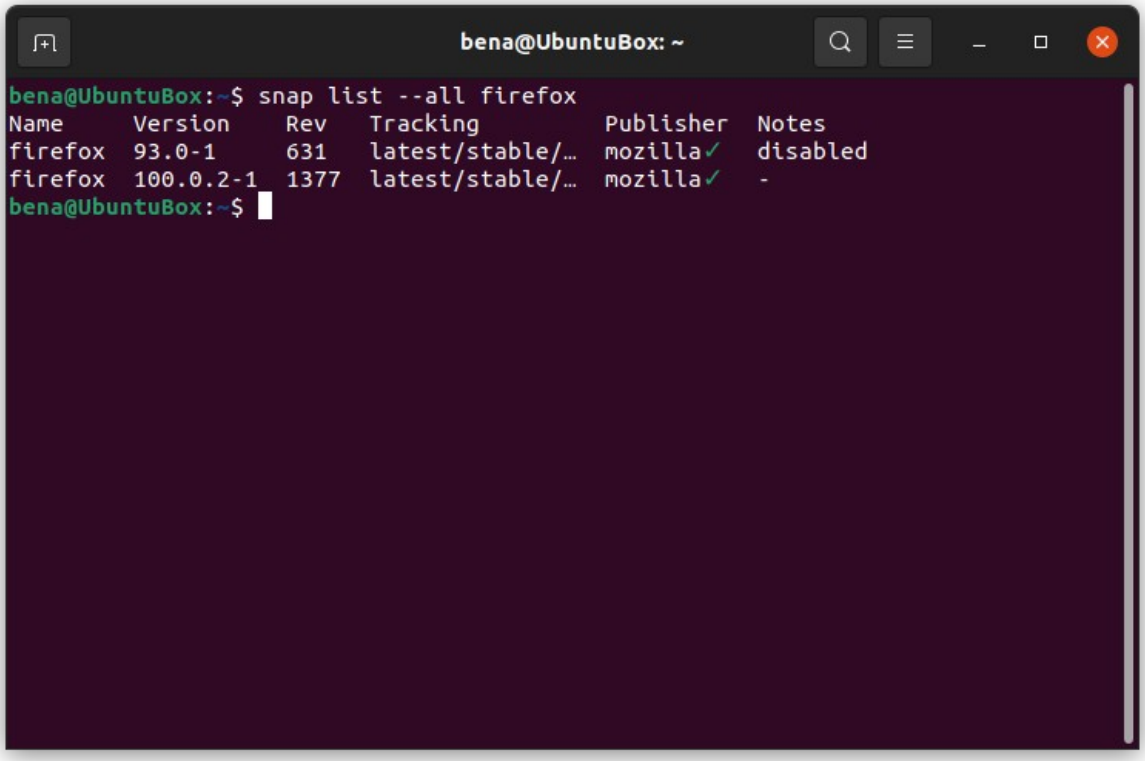
Slika 11: VLC AppImage (desno) ne poštuje tamni način rada dok Flatpak (lijevo) poštuje (autorski rad)

4.2. Snap

Snap je moderan distribucijski format iza kojeg stoji kompanija Canonical, ista kompanija koja vodi Ubuntu projekte. Snap paketi trenutno su podržani na Ubuntu i njemu srodnim distribucijama, ali dostupni su i na svim većim distribucijama poput Fedore ili Archa, osim distribucije Linux Mint čiji se održavatelji protive Snapovima iz razloga o kojima će biti riječi malo kasnije.

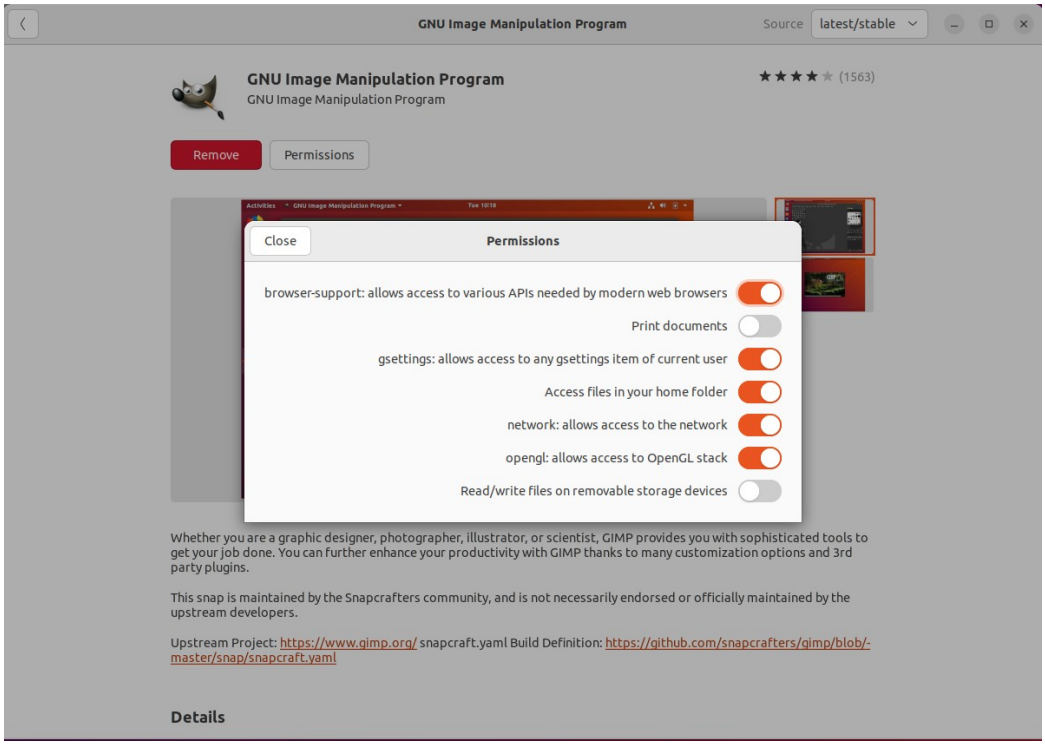
Snap paketi, kao i ostali moderni načini pakiranja, funkcioniraju na principu „1 aplikacija, 1 okolina“. Pakiranje Linux aplikacija u snap pakete jednostavno je i dobro dokumentirano. („Getting started", bez dat.).

Za razliku od tradicionalnih sustava pakiranja i distribucije te Flatpaka, Snap automatski ažurira aplikacijske pakete te vraća na prošlu verziju samo ukoliko ažuriranje nije uspješno izvršeno ili na eksplicitni zahtjev korisnika.



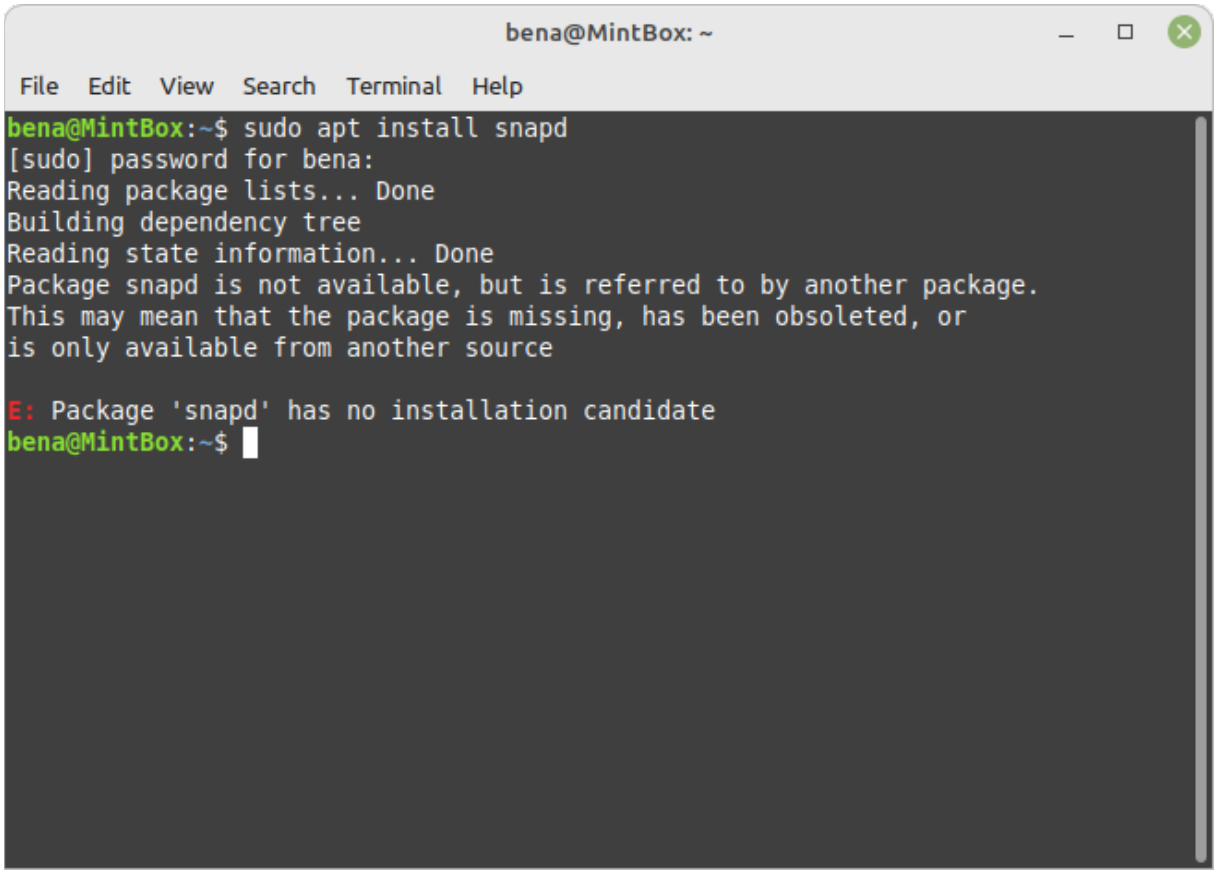
Slika 12: Dostupne verzije snap paketa Firefox (autorski rad)

Snap paketi dopuštaju korisniku kontrolu nad pravima aplikacije slično kako to danas rade Android i iOS pametni telefoni.



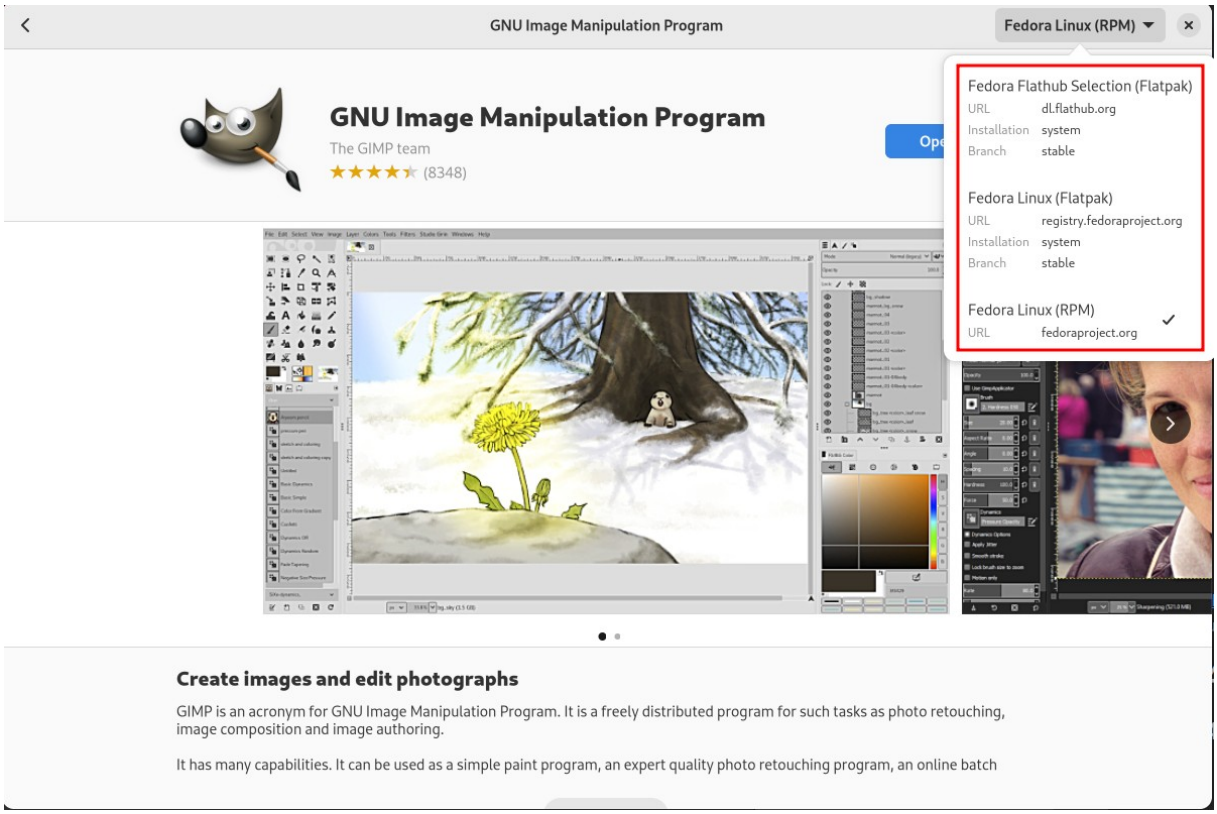
Slika 13: Manipulacija pravima na snap paketu Gimp (autorski rad)

Na Slici 13 vidljivo je sučelje Snap Store-a koji je po mnogima najkontroverznija komponenta distribucije Snap paketa. Snap store je softver zatvorenog koda pod potpunom kontrolom Canonicala, dakle Canonical kao kompanija odlučuje što, kada i kako će biti dostupno na jedinom centraliziranom repozitoriju Snap paketa. Upravo iz tog razloga distribucija Linux Mint blokirala je instalaciju alata „snapd“ koji je nužan za rad Snap paketa.



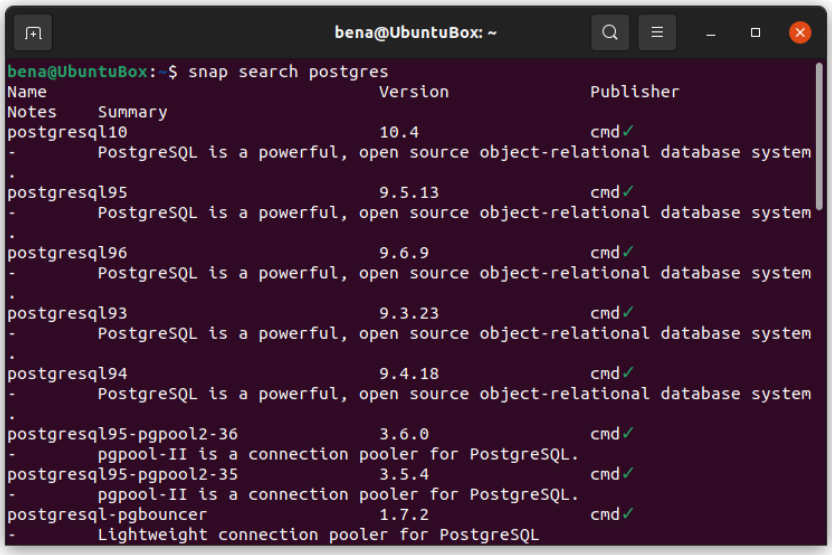
Slika 14: Pokušaj instalacije paketa snapd na Linux Mintu (autorski rad)

Činjenica da su Snap paketi dostupni samo kroz SnapStore, aplikaciju zatvorenog koda, znači da aplikacije poput GNOME software i Elementary AppCenter ne mogu pretraživati, instalirati ni održavati Snap pakete.



Slika 15: GNOME Software ne pronalazi Snap verziju (autorski rad)

Snap paketi najviše su prihvaćeni u serverskom i IoT prostoru zahvaljujući trudu Canonicala. Tako postoji Snap paket koji instalira i postavlja NextCloud sustav kao i mnoge administratorske alate i sustave koji nemaju grafičko sučelje poput PostgresSQL-a i sličnih.



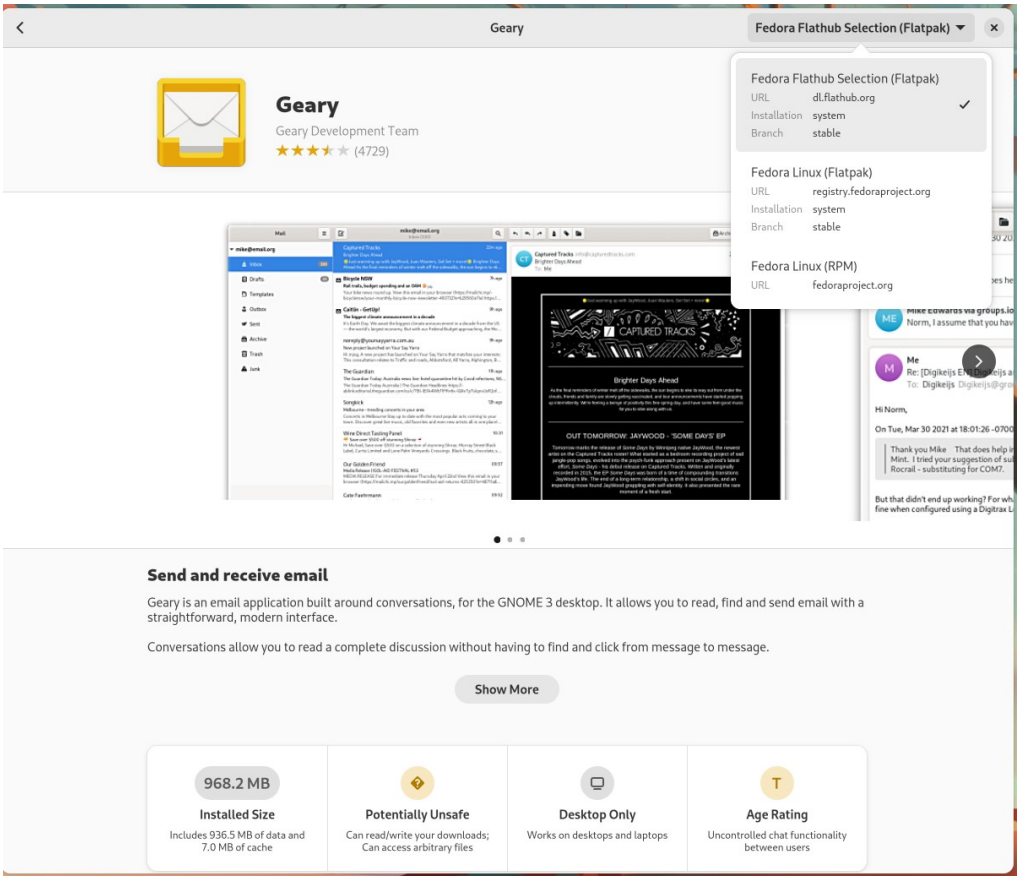
Slika 16: Dostupne verzije Snap paketa Postgres (autorski rad)

4.3. Flatpak

Flatpak je moderan međuplatformni standard pakiranja aplikacija za GNU/Linux sustave. Cilj Flatpaka je stvoriti standard otvorenog tipa koji omogućuje aplikacijama da rade neovisno o operativnom sustavu u pozadini, a također poštuju sigurnosne postavke nad kojima korisnik ima kontrolu.

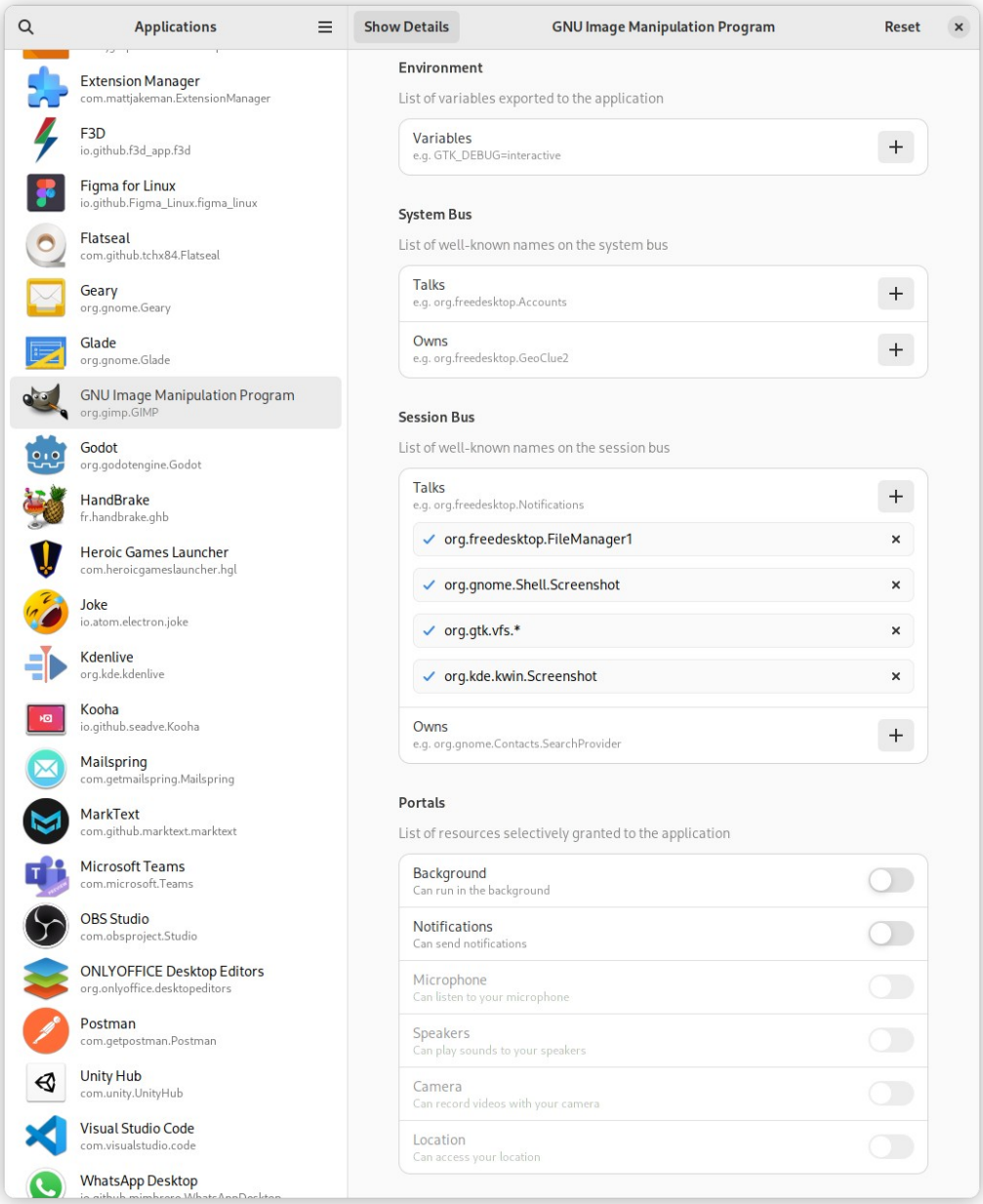
Poput Snap paketa, Flatpak paketi koriste model pješčanika (engl. *sandbox*), ali za razliku od svih ostalih formata za pakiranje i distribuciju Flatpak ne zahtijeva administratorske ovlasti pri instalaciji (engl. *Root access*). Upravo zbog toga se Flatpak našao kao jedino prihvatljivo rješenje na operativnim sustavima namijenjenim djeci poput Endless OS-a ili igraćim konzolama, npr. SteamOS na konzoli Steam Deck.

Za razliku od Snapa, distribucija Flatpak paketa potpuno je slobodna i otvorena tako da svatko može imati svoj Flatpak repozitorij, npr. Fedora Flatpak Repo, ali većina programa može se pronaći na najvećem repozitoriju Flatpak paketa, Flathubu.



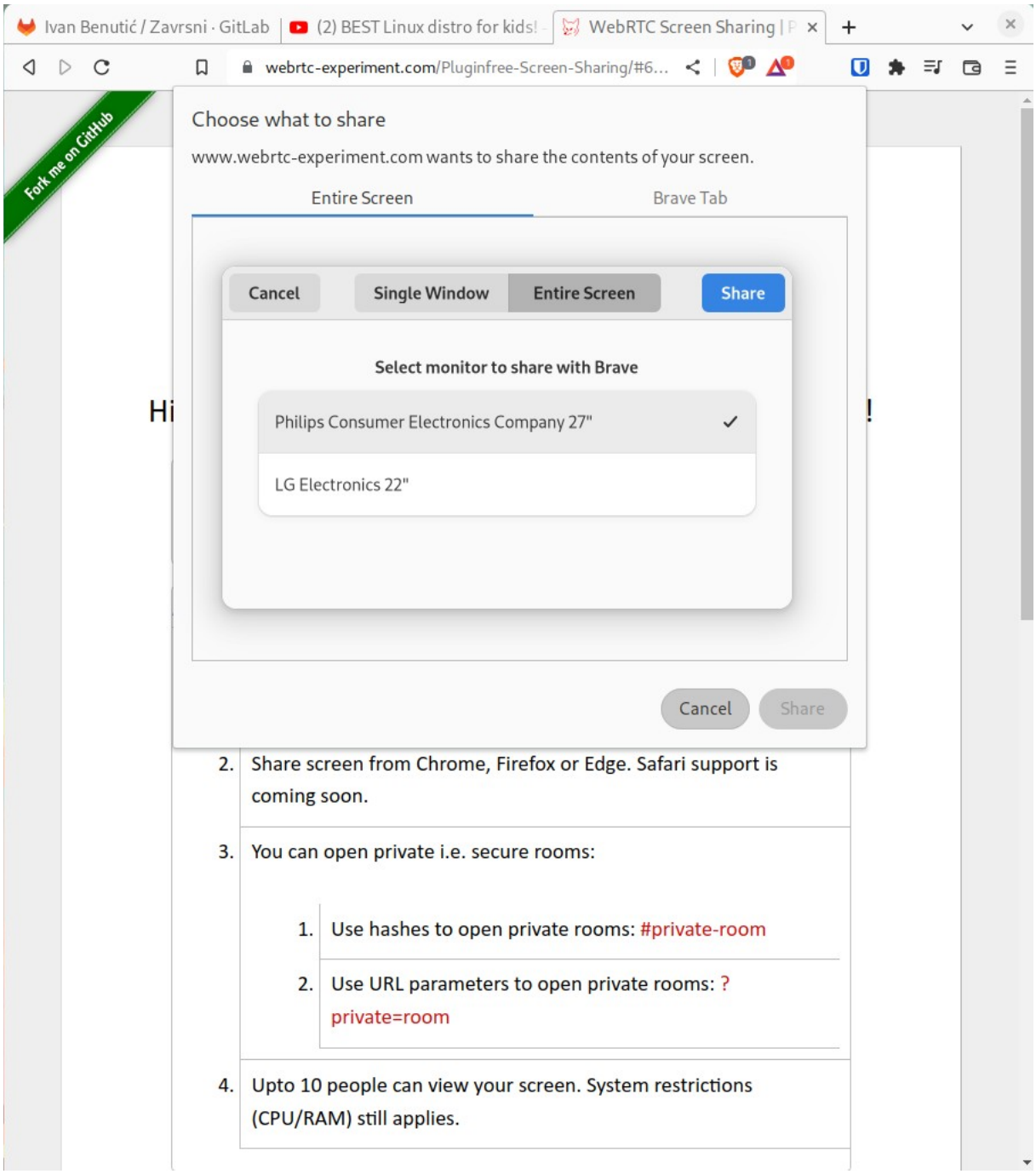
Slika 17: Integracija različitih Flatpak repozitorija na Fedora sustavu (autorski rad)

Pomoću aplikacije FlatSeal Flatpak paketi daju potpunu kontrolu nad dostupnim komponentama i datotekama na svom računalu za svaku aplikaciju posebno. Sličan sustav koristi se na Android uređajima. Pomoću Flatseala moguće je aplikacijama ograničiti pristup kameri, mikrofonu, notifikacijama, datotekama, snimanju zaslona i slično („Flatseal“, 2022).



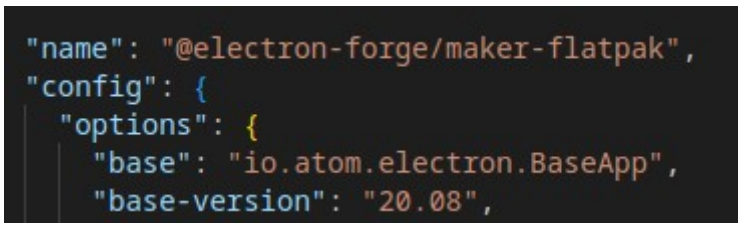
Slika 18: Flatseal sučelje (autorski rad)

Za rad u pozadini i slanje notifikacija Flatpak koristi poseban Portal protokol koji omogućuje još granularniju kontrolu nad pravima aplikacije nad sustavom (McLasen, 2018).



Slika 19: Portal dijalog traži dopuštenje za dijeljenje ekrana (autorski rad)

Flatpak funkcionira na temelju osnovnih paketa za određenu verziju operativnog sustava, potrebnih biblioteka i upravljačkih programa. Programeri te parametre postavljaju u procesu pakiranja aplikacije, a aplikacije koje koriste iste osnovne pakete ih dijele (engl. *Shared runtime*) kako bi se uštedio prostor na disku (Stouff, 2022)



Slika 20: postavljanje Flatpak baze u Electronu (autorski rad)

Flatpak je, barem za sada, riješio gotovo sve probleme pakiranja i distribucije aplikacijskih paketa na GNU/Linux sustavima.

5. Izrada i distribucija aplikacije

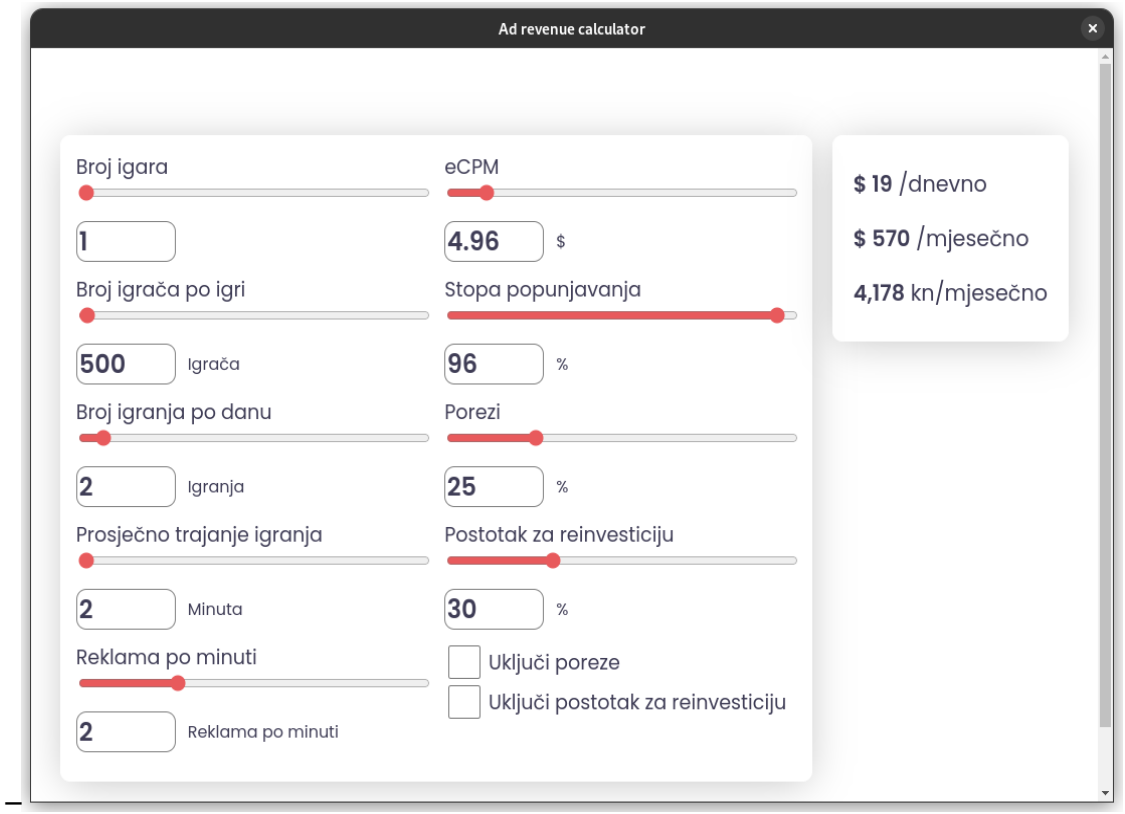
U sklopu ovog rada napravljena je aplikacija bazirana na ElectronJS programskom okviru. Bit će prikazan proces izrade (engl. *build*) .deb .rpm i .flatpak paketa, instalacija istih, kao i proces postavljanja aplikacije na privatne repozitorije.

5.1. ElectronJS

ElectronJS je programski okvir baziran na Chromiumovom V8 web pokretaču i NodeJS programskom okruženju namijenjen izradi međuplatformskih aplikacija za Windows, Linux i MacOS (ElectronJS.org, bez dat.).

5.2. O aplikaciji

Aplikacija je izrađena u programskom jeziku JavaScript uz uporabu programskih biblioteka ElectronJS i React. Aplikacija služi kao kalkulator potencijalne zarade od prikaza reklama u mobilnim igrama.



Slika 21: Aplikacija (autorski rad)

Izrada same aplikacije gotovo je ista kao i izrada web aplikacije u Reactu. Svaki klizač je zasebna komponenta koja sadrži stanje deklarirano React hookom useState, a zatim putem React hooka useEffect izračunavamo potencijalnu dnevnu zaradu. Ispod je prikazan kod osnovne logike aplikacije:

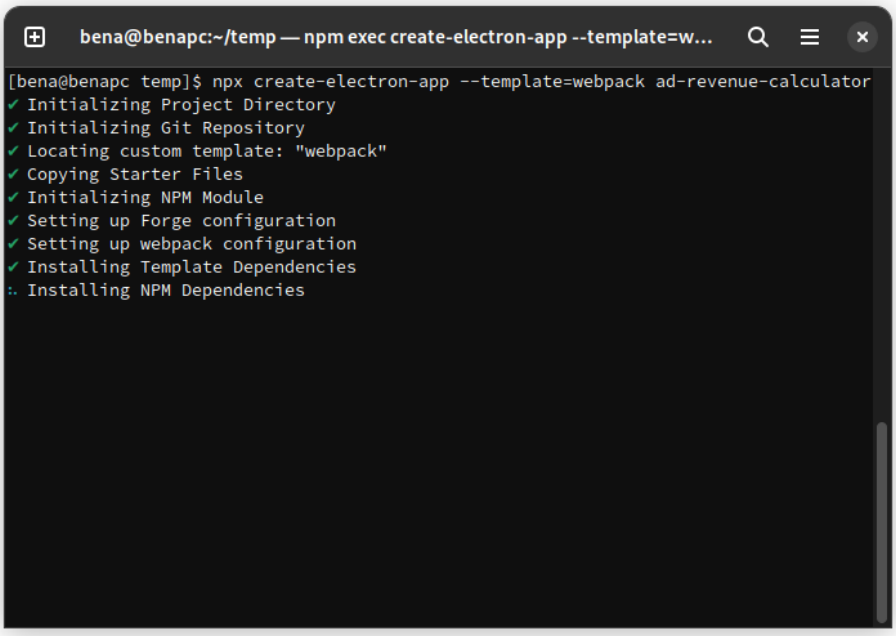
```
const [games, setGames] = useState(0);
const [avgPlayTime, setAvgPlayTime] = useState(0);
const [numberOfPlayers, setNumberOfPlayers] = useState(0);
const [cpm, setCpm] = useState(0);
const [sessionsPerDay, setSessionsPerDay] = useState(0);
const [adsPerMin, setAdsPerMin] = useState(0);
const [fillRate, setFillRate] = useState(0);
const [includeTaxes, setIncludeTaxes] = useState(false);
const [includeGoogleCut, setIncludeGoogleCut] = useState(false);
const [earnings, setEarnings] = useState(0);
const [taxes, setTaxes] = useState(0);
const [googleCut, setGoogleCut] = useState(0);

useEffect(() => {
  const impressions = numberOfPlayers * sessionsPerDay * games;
  const impressions_cpm = (impressions / 1000) * cpm;
  const adRev = avgPlayTime * adsPerMin * (fillRate / 100);
  const gC = includeGoogleCut ? 1 - googleCut / 100 : 1
  const tax = includeTaxes ? 1 - taxes / 100 : 1
  const finalDayly = impressions_cpm * adRev * tax * gC;
  setEarnings(parseInt(finalDayly));
}, [
  games,
  avgPlayTime,
  numberOfPlayers,
  cpm,
  sessionsPerDay,
  adsPerMin,
  fillRate,
  includeGoogleCut,
  includeTaxes,
  googleCut,
  taxes
])
```

Ispis korisničkog sučelja odvija se pomoću JSX sintakse. Primjer koda za ispis rezultata kalkulacije:

```
<div className="result">
<p>
  <b>$ {earnings.toLocaleString()}</b> /day
</p>
<p>
  <b>$ {(parseInt(earnings) * 30).toLocaleString()}</b> /month
</p>
<p>
  <b>{parseInt(earnings * 30 * 7.33).toLocaleString()}</b> kn/month
</p>
</div>
```

5.3. ElectronJS konfiguracija

Prateći dokumentaciju na ElectronForge stranicama, ElectronJS instanca konfigurirana je za rad s ReactJS-om (ElectronForge.io, bez dat.).

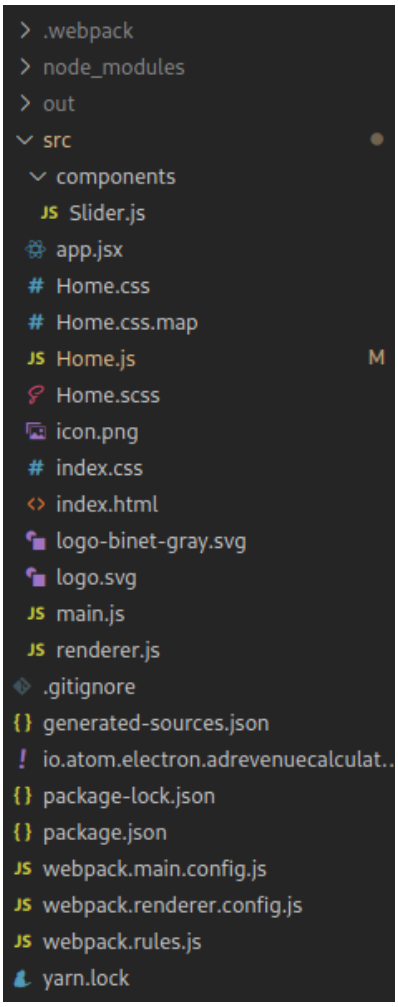


Slika 22: Kreiranje ElectronJS predloška pomoću NPM paketa (autorski rad)

Nakon instalacije i konfiguracije potrebnih NPM paketa za izradu aplikacije bilo je potrebno dodati Webpack pravilo za JSX datoteke. Kod dodan u datoteku webpack.rules.js izgleda ovako:

```
module.exports = [
    {
      test: /\.jsx?$/,
      use: {
        loader: 'babel-loader',
        options: {
          exclude: /node_modules/,
          presets: ['@babel/preset-react'],
        }
      }
    },
  ]
```

Nakon odrađivanja svih potrebnih radnji i konfiguracija struktura datoteka prikazana je na Slici 23.



Slika 23: Struktura aplikacije (autorski rad)

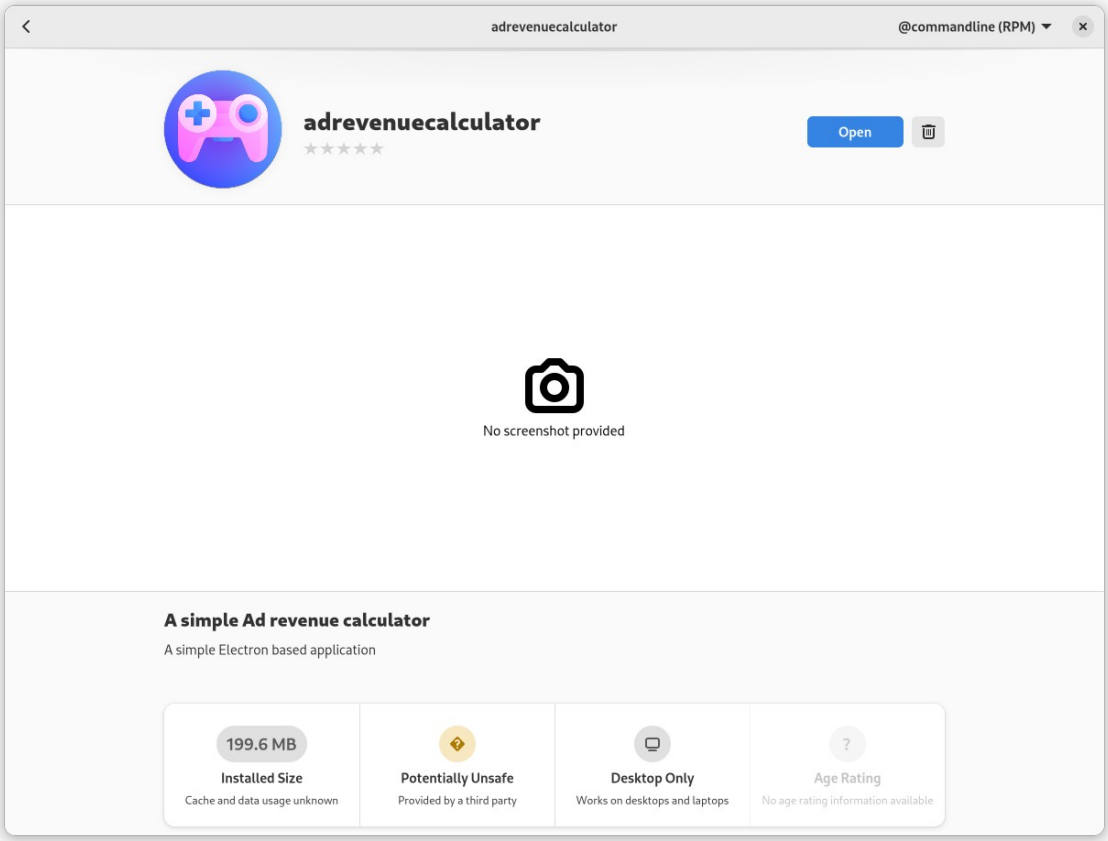
Kako bismo postigli cilj generiranja deb, Flatpak i rpm paketa potrebno je dodati za to predviđene ElectronForge pakete. Zato dio package.json datoteke za ovaj projekt izgleda ovako:

```
"devDependencies": {
    "@babel/core": "^7.17.9",
    "@babel/preset-react": "^7.16.7",
    "@electron-forge/cli": "^6.0.0-beta.63",
    "@electron-forge/maker-deb": "^6.0.0-beta.63",
    "@electron-forge/maker-flatpak": "^6.0.0-beta.63",
    "@electron-forge/maker-rpm": "^6.0.0-beta.63",
    "@electron-forge/maker-squirrel": "^6.0.0-beta.63",
    "@electron-forge/maker-zip": "^6.0.0-beta.63",
    "@electron-forge/plugin-webpack": "6.0.0-beta.63",
    "@vercel/webpack-asset-relocator-loader": "1.7.0",
    "babel-loader": "^8.2.4",
    "css-loader": "^6.0.0",
    "electron": "18.0.4",
    "node-loader": "^2.0.0",
    "style-loader": "^3.0.0"
  },
```

Svaki od @electron-forge/maker paketa vezan je za `electron-forge make` naredbu te svaki zahtjeva konfiguraciju unutar polja `makers[]`. Prikaz konfiguracije RPM makera:

```
{
    "name": "@electron-forge/maker-rpm",
    "config": {
      "options": {
        "icon": {
          "512x512": "src/icon.png"
        },
        "categories": [
          "Utility"
        ],
        "description": "A simple Ad revenue calculator",
        "productDescription": "A simple Electron based application"
      }
    }
  },
```

Parametri postavljeni u kodu iznad korišteni su u izradi manifesta i uputa za izgradnju određenog paketa, a rezultat se može vidjeti u Software Centru (Slika 24).



Slika 24: Prikaz ikone i opisa aplikacije u Software centru (autorski rad)

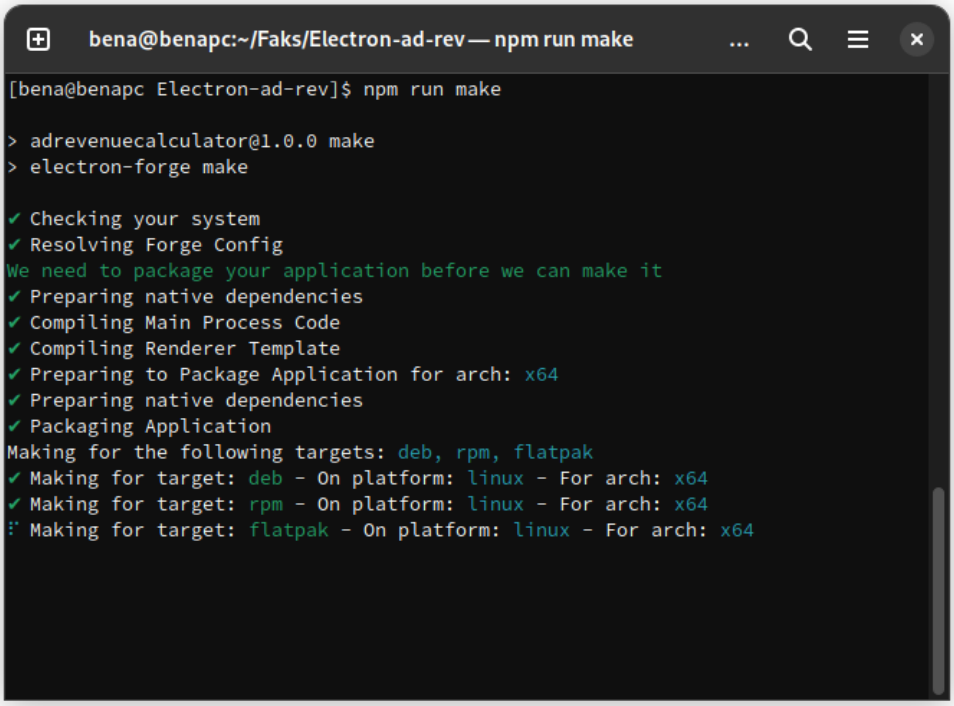
Dodavanjem informacija u package.json datoteku moguće je popuniti i druge informacije o aplikaciji poput snimki zaslona, kontakt informacija i sl. Parametar vezan za kategoriju služi upravitelju paketa za sortiranje i bolje pretraživanje aplikacija.

Modul za Flatpak zahtijevao je dodatne informacije o verziji Zypak modula za renderiranje aplikacije. Manualno deklariranje modula izgleda ovako:

```
{
    "name": "@electron-forge/maker-flatpak",
    "config": {
      "options": {
        "base": "io.atom.electron.BaseApp",
        "base-version": "20.08",
        "icon": {
          "512x512": "src/icon.png"
        },
        "categories": [
          "Utility"
        ],
        "description": "A simple Ad revenue calculator",
        "productDescription": "A simple Electron based application"
      },
      "modules": [
        {
          "name": "zypak",
          "sources": [
            {
              "type": "git",
              "url": "https://github.com/refi64/zypak",
              "tag": "v2022.03"
            }
          ]
        }
      ]
    }
  }
```

Tijekom izrade aplikacije ispostavilo se da je ovo nedokumentirani problem, te sam povodom toga otvorio problem na službenim GitHub stranicama.

Sada kada je konfiguracija spremna, a aplikacija gotova, vrijeme je da izradimo deb, rpm i flatpak datoteke. Zahvaljujući biblioteci ElectronForge to se može postići jednostavnom naredbom `npm run make` (Slika 25).



Slika 25: Izrada datoteka pomoću biblioteke Electron Forge (autorski rad)

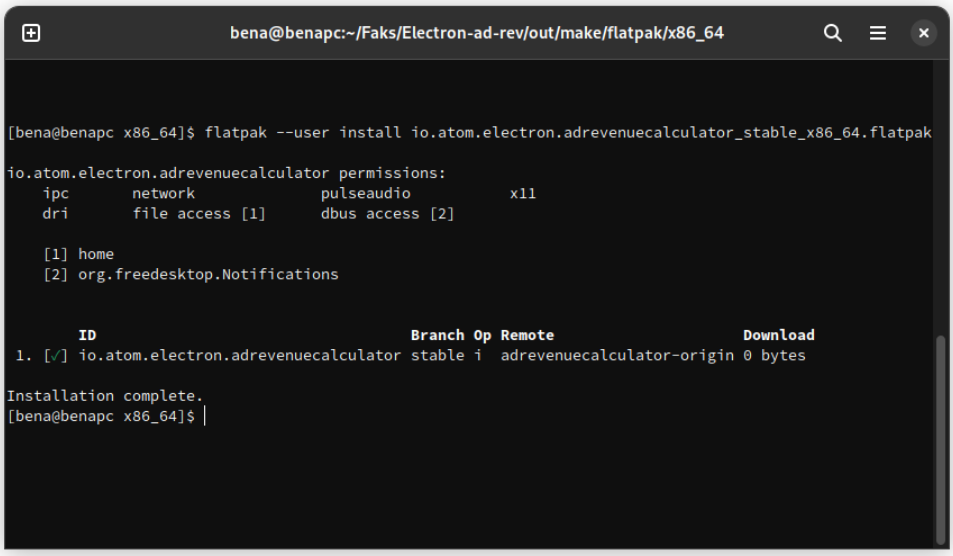
5.4. Instalacija i testiranje paketa

Electron forge stvorio je tri instalacijske datoteke po uputama danim u package.json datoteci.

S obzirom na to da smo na Slici 23 prikazali instalaciju putem Software Centra, ovaj put ćemo za instalaciju Flatpak paketa koristiti naredbeni redak. Naredba za instalaciju izgleda ovako:

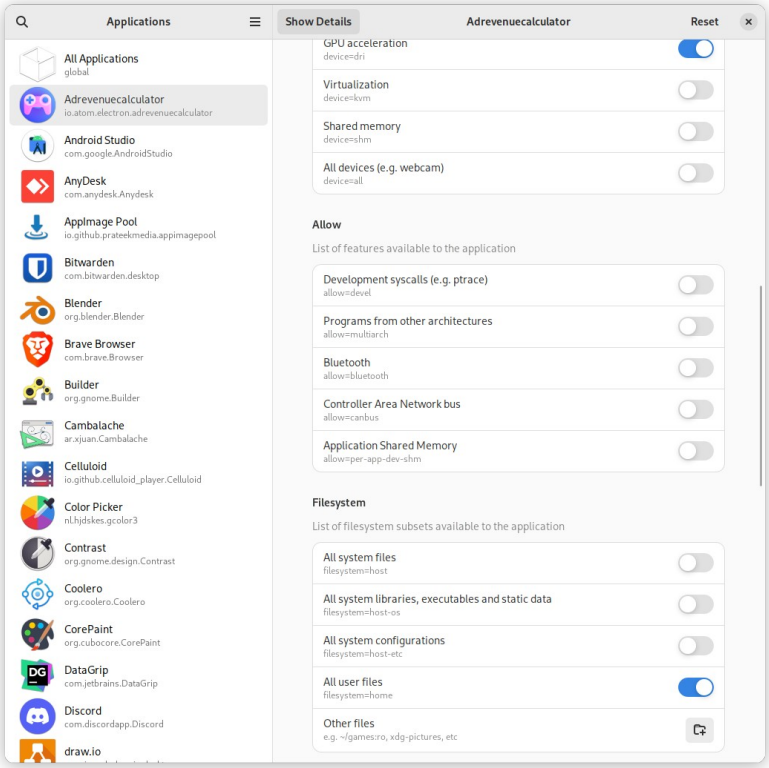
```
flatpak --user install
io.atom.electron.adrevenuecalculator_stable_x86_64.flatpak
```

Nakon vrlo kratke instalacije prikazane na Slici 26 aplikacija je spremna za korištenje.



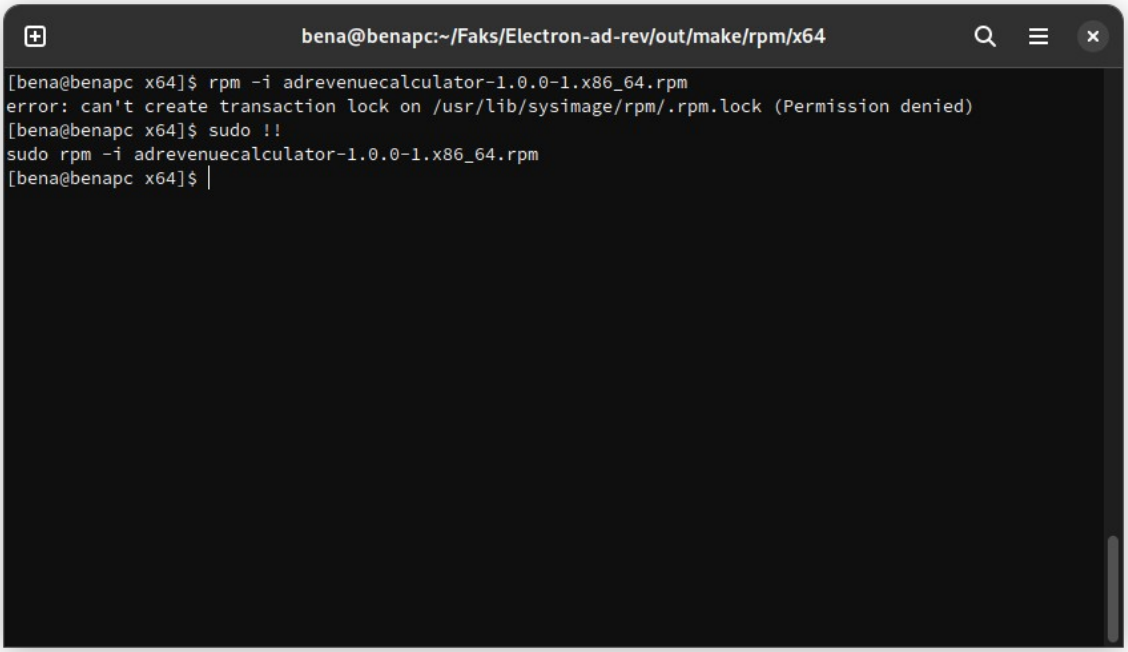
Slika 26: Instalacija flatpak paketa aplikacije u naredbenom retku (autorski rad)

Iz instalacije je moguće vidjeti da nigdje nije zatražen administratorski (root) pristup. A sigurnosne značajke Flatpak paketa vide se i kroz već spomenutu aplikaciju FlatSeal na Slici 27.



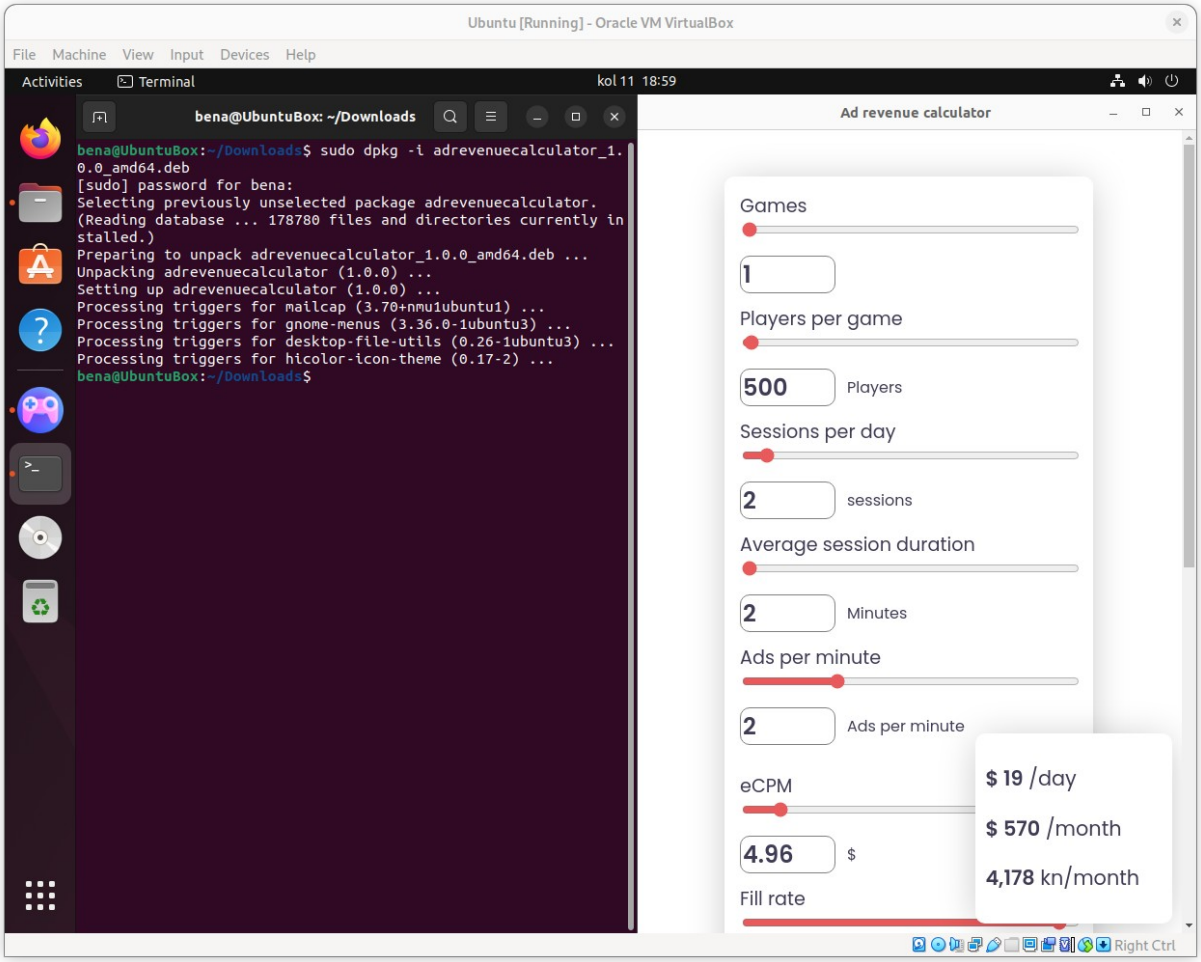
Slika 27: Prava aplikacije u Flatpak sandbox sučelju (autorski rad)

Prilikom instalacije RPM paketa zatražen je administratorski (root) pristup direktoriju vezanom za RPM (/usr/lib/sysimage/rpm/.rpm.lock) jer se nalazi u početnom (root) direktoriju označenom znakom „ / “ dok se flatpak instalira u korisnički direktoriji označen znakom „ ~ “. Proces instalacije također je manje intuitivan jer nedostaje povratna informacija o uspješnoj instalaciji paketa kao što je vidljivo na Slici 28.



Slika 28: Instalacija RPM paketa kroz naredbeni redak (autorski rad)

Proces instalacije deb paketa na Ubuntu sustavu vrlo je sličan procesu instalacije RPM paketa na Fedori prikazanom iznad. Instalaciju ovog puta vršimo kroz već prije spomenuti alat dpkg. naredbom `dpkg -i [ime paketa]` vrši se instalacija deb paketa. Na Slici 29 vidimo ispis vezan za instalaciju kao i pokrenutu aplikaciju nakon instalacije. Kao i kod RPM paketa, instalacija zahtjeva administratorske ovlasti, stoga je instalacija deb paketa manje sigurna od flatpak i AppImage paketa.



Slika 29: Instalacija i pokretanje deb paketa na Ubuntu sustavu (autorski rad)

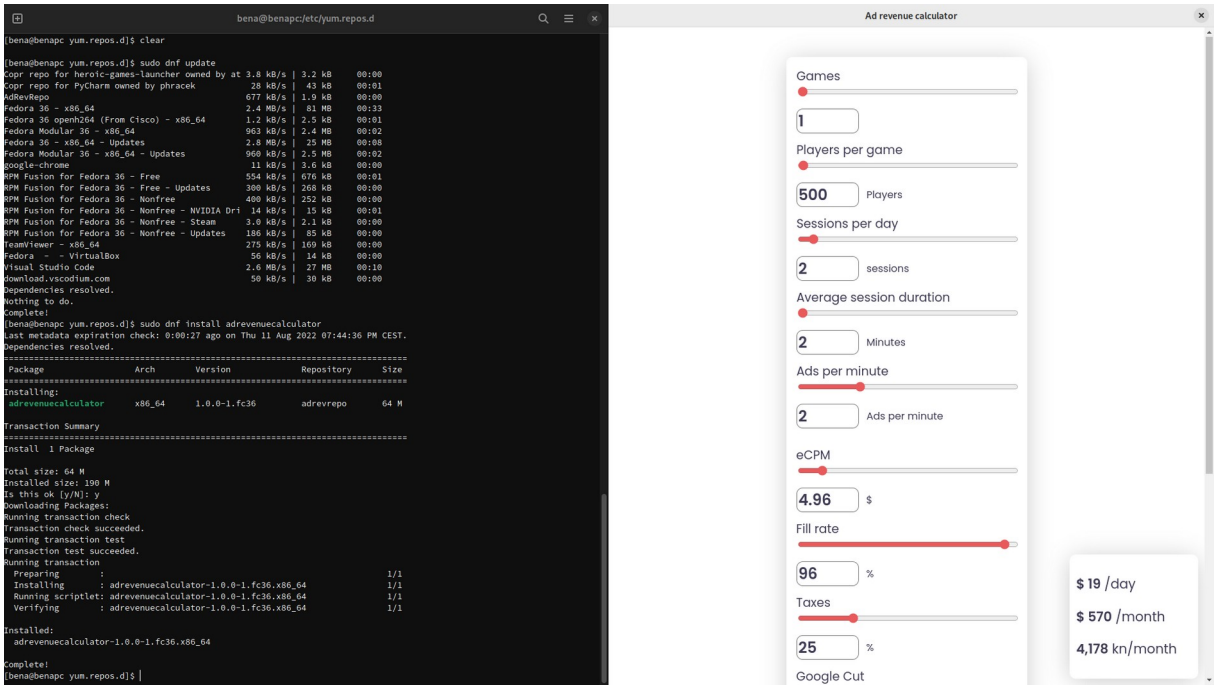
5.5. Izrada privatnog Yum repozitorija za RPM pakete

Privatni repozitoriji vrlo su korisni kada je potrebno podijeliti aplikaciju bez obzira na službene distribucijske repozitorije, na primjer takvu praksu imaju programerski timovi iza Brave pretraživača i Visual Studio Codiuma, programa na kojem se bazira Microsoftov alat Visual Studio Code („VSCodium“, bez dat.).

Prvi korak pri kreiranju Yum repozitorija je instalacija paketa createrepo. Zatim se u novi prazni direktorij postavljaju RPM paketi koje ćemo postaviti na repozitorij te se pokreće naredba `createrepo [putanja do direktorija]`.Nakon toga potrebno je stvoriti novu datoteku u direktoriju /etc/yum.repos.d. Na traženoj putanji stvorili smo datoteku adrev.repo. U datoteci se nalaze osnovne informacije o repozitoriju poput URL-a, aktivnosti i provjere autentičnosti GPG ključevima koja je u našem slučaju isključena (Patlan, 2020).
Sadržaj datoteke adrev.repo detaljnije je prikazan ovdje:

```
[adrevrepo]
name=AdRevRepo
baseurl=file:///home/bena/Faks/yumrepo
enabled=1
gpgcheck=0
```

Nakon ažuriranja DNF liste paketa naredbom `sudo dnf clean all && sudo dnf update` vidimo naš repozitorij pod imenom AdRevRepo kao što je navedeno u datoteci adrev.repo. Zatim jednostavno instaliramo našu aplikaciju naredbom `sudo dnf install adrevenuecalculator -y` (Slika 30).



Slika 30: Ažuriranje repozitorija, instalacija aplikacije i pokrenuta aplikacija putem Yum/DNF repozitorija (autorski rad)

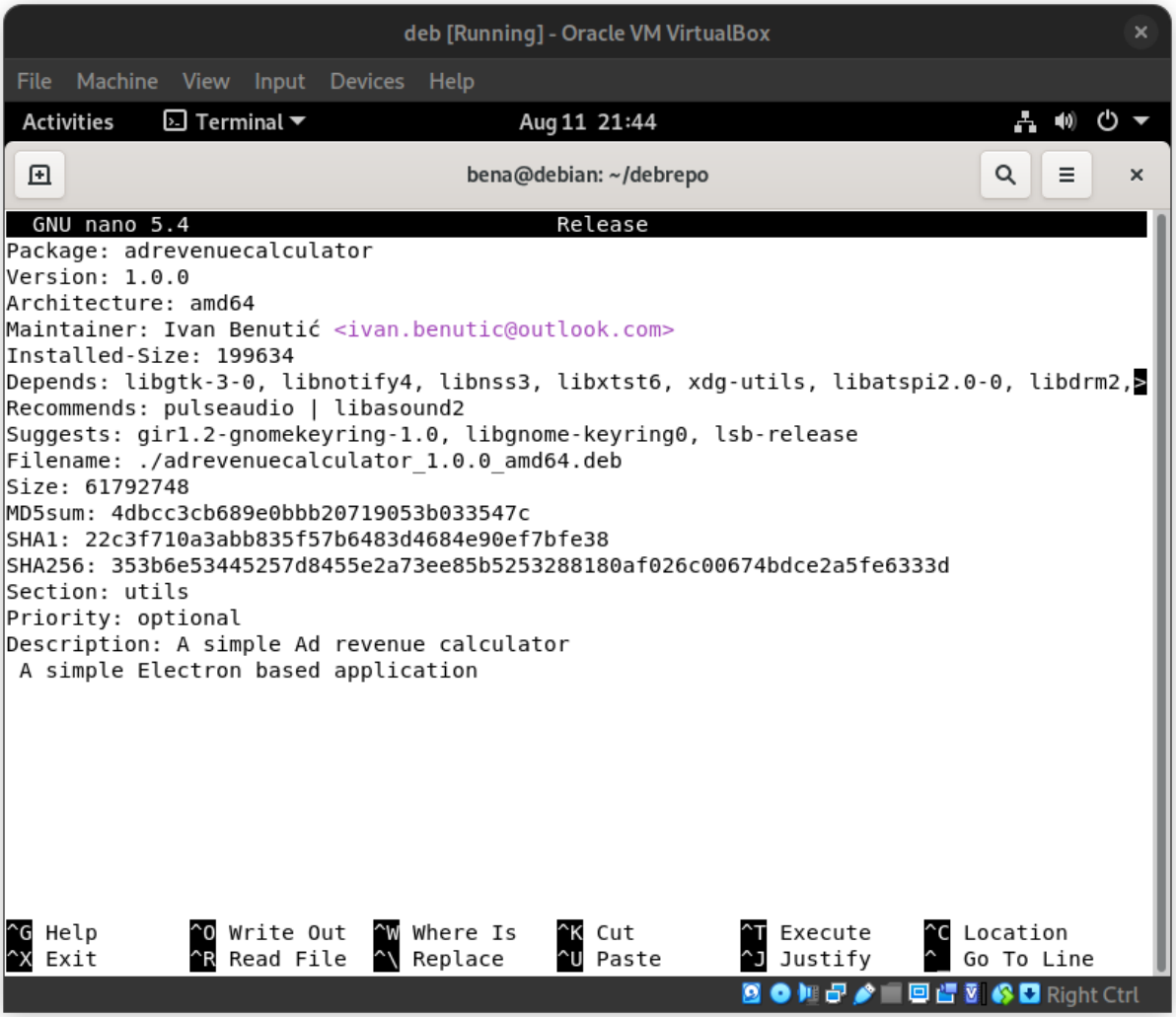
Iako je ovo primjer lokalnog repozitorija, ovakav repozitorij moguće je potpuno otvoriti prema Internetu te ga koristiti za distribuciju aplikacijskih paketa. Lokalni repozitoriji također imaju svrhu u poslužiteljskim okruženjima gdje je preuzimanje datoteka zabranjeno iz sigurnosnih razloga, stoga se često stvaraju repozitoriji kako bi upravitelj paketima poput RPM-a ili APT-a proveo instalaciju i razriješio eventualne probleme međuovisnosti paketa (engl. *Dependency*). Takav način rada čest je pri postavljanju Oracle poslužiteljskih sustava.

5.6. Izrada privatnog repozitorija na Debianu

Proces izrade repozitorija za deb pakete na Debian i njemu srodnim sustavima sličan je već prikazanom procesu za izradu privatnih Yum repozitorija. Nakon što smo postavili deb paket u novi direktorij koristit ćemo alat dpkg-scanpackages iz paketa dpkg-dev kako bismo naredbama:

```
dpkg-scanpackages . /dev/null > Release
dpkg-scanpackages . /dev/null | gzip -9c > Packages.gz
```

izradili Release i Packages.gz datoteke potrebne za deb repozitorij (Slika 31). Navedene datoteke sadrže informacije koje koristi upravitelj paketima za opis, kontrolu verzija i autentifikaciju paketa ("How to Create Your Own Repository for Packages on Debian", 2022).



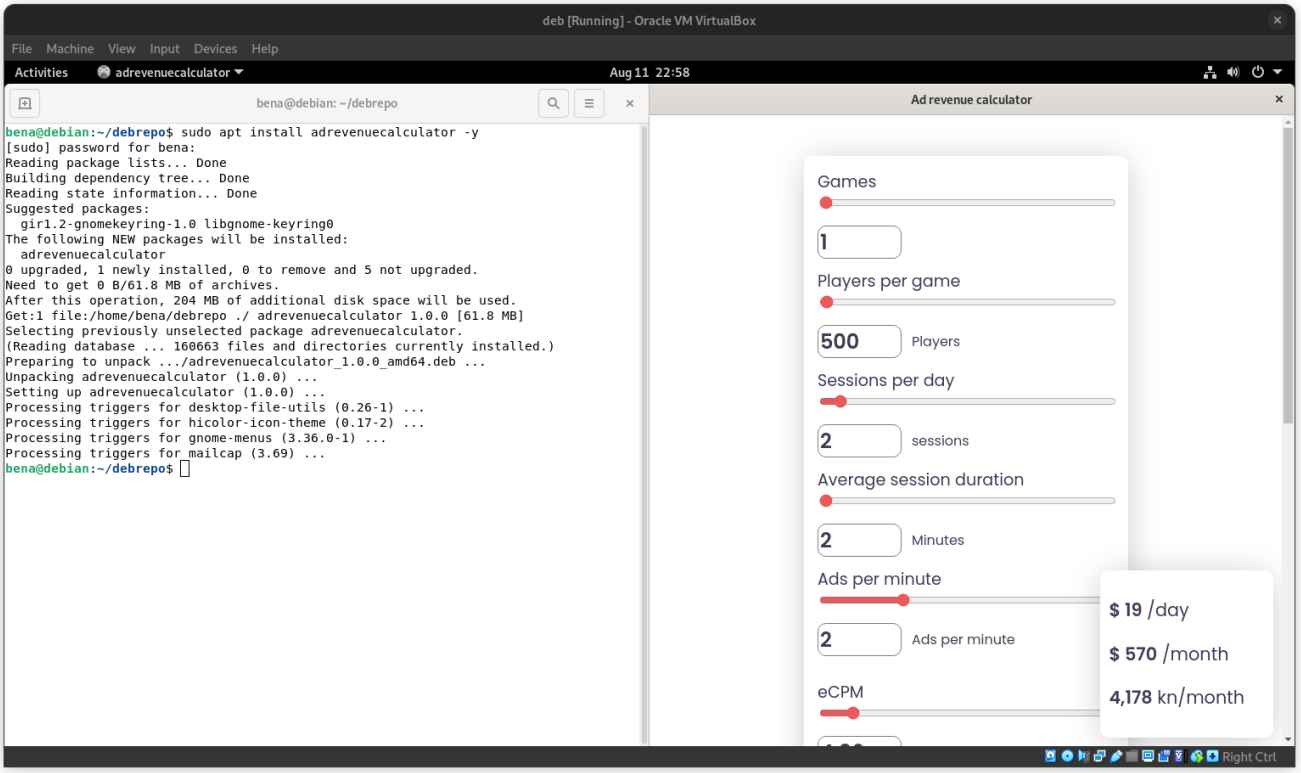
Slika 31: Sadržaj datoteke Release za Debian repozitorij (autorski rad)

Nakon izrade repozitorija potrebno ga je dodati u datoteku sources.list linijom:

`deb [trusted=yes] file:/home/bena/debrepo ./`

Ključna riječ `deb` označava da je riječ o repozitoriju koji na sebi ima deb datoteke, dok ključna riječ `deb-src` označava repozitorije koji na sebi drže izvorni (engl. Source) kod. Dio koda `[trusted=yes]` odnosi se na „povjerenje“ prema repozitoriju bez certifikata.

Potom naredbom `sudo apt update` ažuriramo listu dostupnih paketa te instaliramo novi paket naredbom `sudo apt install adrevenuecalculator -y`. Proces instalacije kao i instalirana aplikacija vidljivi su na Slici 32.



Slika 32: Instalacija paketa s lokalnog deb repozitorija na Debian sustavu (autorski rad)

Naravno i ovaj lokalni repozitorij može se postaviti na Internet i koristiti kao distribucijski kanal.

Kompanija koja stoji iza Ubuntua koji je baziran na Debianu pokrenula je servis Launchpad.net kojem je cilj pojednostaviti pokretanje i održavanje deb repozitorija kao i izvornog koda aplikacija kroz sustav osobnih arhiva paketa (PPA, engl. *Personal package archive*).

5.7. Distribucija aplikacije kao Flatpak paketa

Flatpak kao najčešće korišteni ekvivalent lokalnom repozitoriju nudi sustav naredbi `flatpak create-usb` koji u praksi može biti ili na stvarnom USB disku ili na mrežnom mjestu (engl. *Network share*). Tada se taj USB disk ili mrežno mjesto ponaša kao privatni repozitorij koji se u Flatpak sustav ubacuje naredbom:
`flatpak install –sideload-repo=<putanja do repozitorija> <ime paketa>`
("Publishing ", bez dat.). Ovakva praksa odlična je za korporativna okruženja i za grupni razvoj aplikacija.

Najčešći način distribucije flatpak paketa je kroz Flathub repozitorij. Sama objava aplikacija odvija se kroz oskudno dokumentirani proces na servisu GitHub te se developeri redovito oslanjaju na pomoć održavatelja Flathuba kao i na druge developere u zajednici koristeći GitHub, Reddit i Matrix kanale komunikacije ("App submission", bez dat.).
Flatpak također ima zasebni odjeljak dokumentacije za najpopularnije programske okvire, a među njima i Electron koji je korišten za izradu aplikacije u ovom radu.
S obzirom na činjenicu da Flathub funkcionira na način da se sama flatpak datoteka gradi (engl. *build*) na Flathubovom serveru, potrebno je tom serveru dati detaljne upute o aplikaciji. U slučaju Electrona potrebno je stvoriti 2 nove datoteke, a to su takozvana manifest datoteka u kojoj se nalaze informacije o aplikaciji koje koristi pretraživač aplikacija poput *Software Centera* te generated-sources.json datoteka koja daje dodatne upute o JavaScript bibliotekama za izgradnju dijela aplikacije koji se oslanja na NodeJS.

Manifest datoteka u našem slučaju izgleda ovako:

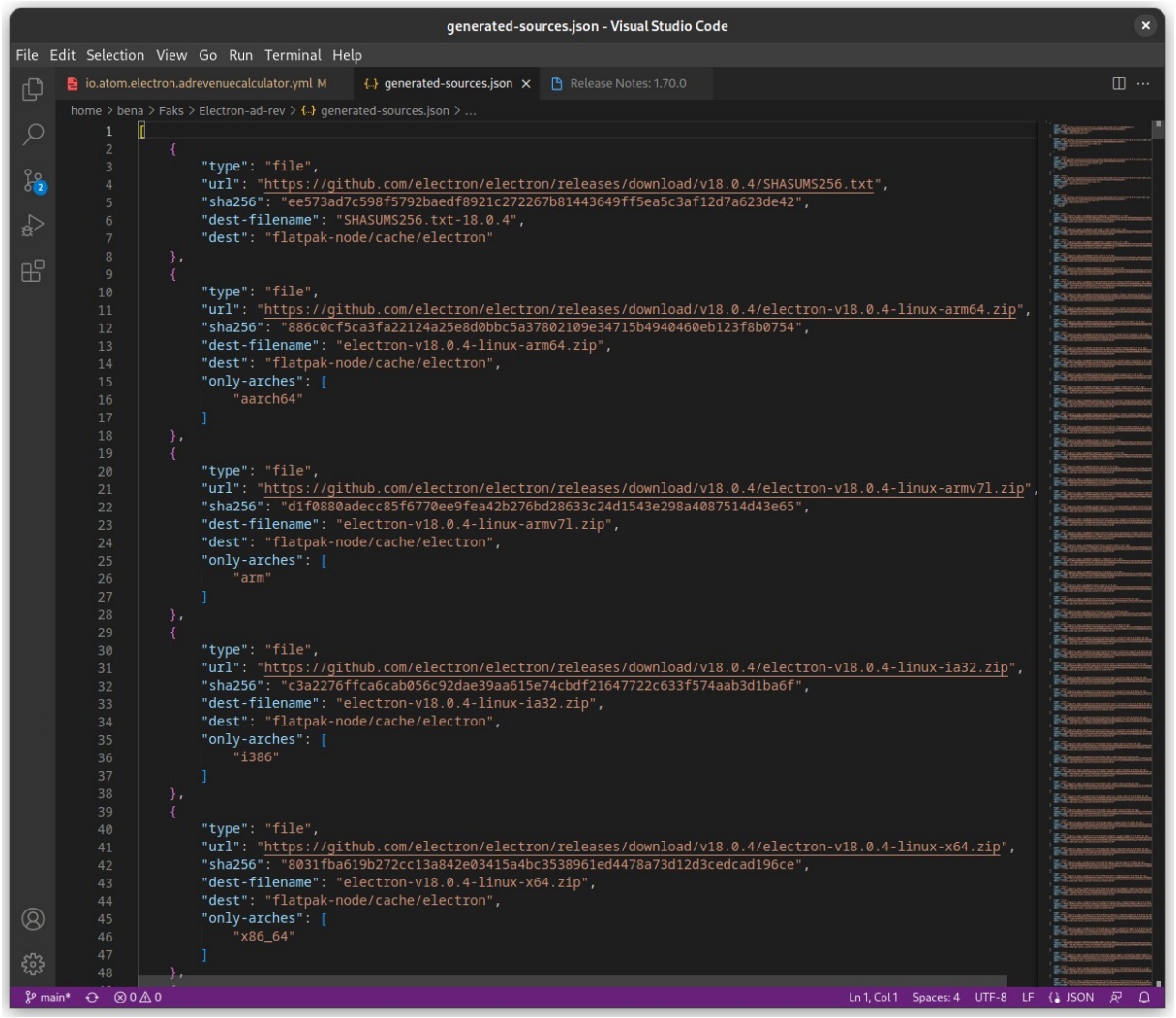
```
app-id: io.atom.electron.adrevenuecalculator
runtime: org.freedesktop.Platform
runtime-version: '20.08'
sdk: org.freedesktop.Sdk
base: io.atom.electron.BaseApp
base-version: '20.08'
sdk-extensions:
  - org.freedesktop.Sdk.Extension.node16
command: run.sh
separate-locales: false
finish-args:
  - --socket=x11
  - --socket=wayland
  - --socket=pulseaudio
```

```
  - --share=ipc
  - --share=network
build-options:
  append-path: /usr/lib/sdk/node16/bin
  env:
    NPM_CONFIG_LOGLEVEL: info
  modules:
    - name: ad-revenue-calculator
      buildsystem: simple
      build-options:
      env:
                                                          XDG_CACHE_HOME:
/run/build/ad-revenue-calculator/flatpak-node/cache
            npm_config_cache: /run/build/ad-revenue-calculator/flatpak-
node/npm-cache
        npm_config_offline: 'true'
    build-commands:
      # Install npm dependencies
      - npm install --offline
      # Build the app; in this example the `dist` script
      # in package.json runs electron-builder
      - |
        . ../flatpak-node/electron-builder-arch-args.sh
        npm run dist -- $ELECTRON_BUILDER_ARCH_ARGS  --linux --dir
      # Bundle app and dependencies
      - cp -a dist/linux*unpacked /app/main
      # Install app wrapper
      - install -Dm755 -t /app/bin/ ../run.sh
    subdir: main
    sources:
                    url:https://gitlab.com/ibenutic/electron-ad-revenue-
calculator.git
      - type: dir
        path: ..
        dest: main
      - generated-sources.json
      # Wrapper to launch the app
      - type: script
        dest-filename: run.sh
        commands:
          - zypak-wrapper.sh /app/main/ad-revenue-calculator "$@"
```

Kod je pisan u Yaml skriptnom jeziku te sadrži sve potrebne informacije o aplikaciji. Za stvaranje generated-sources.json datoteke postoji skripta pisana u programskom jeziku Python. Nakon instalacije Python pipx paketa skripta se poziva naredbom:

`python3 flatpak-node-generator npm  package-lock.json`

Datoteka koju stvori Python skripta u slučaju naše aplikacije prikazana je na Slici 33.



Slika 33: generated-sources.json (autorski rad)

U skripti se nalazi popis svih potrebnih paketa kako bi se mogla izgraditi aplikacija. ("Publishing ", bez dat.). Zatim je potrebno stvoriti takozvani *pull request* nove grane na Flathubovom GitHub repozitoriju te čekati na odaziv i daljnje upute od strane održavatelja Flathuba.

Može se pretpostaviti da će dokumentacija i procesi za moderne načine pakiranja i distribucije aplikacija s vremenom evoluirati zahvaljujući volonterima u zajednici, ali i kompanijama poput RedHata, glavnog pokrovitelja Fedora projekta, kao i bivšeg CentOS-a, osnove za brojne sustave poput Oracle Linuxa i drugih. Kao što je rekao Linux publicist Nick Stouff: „Flatpak je budućnost“ (Stouff, 2022).

6. Zaključak

U ovom radu iznesene su mnoge pozitivne strane modernih načina pakiranja aplikacija poput automatskog ažuriranja i rada u modelu pješčanika koji za sobom povlače mnoge kako sigurnosne, tako i operativne benefite. Naravno, nativni paketi još će dugo tvoriti osnovu GNU/Linux operativnih sustava zbog benefita poput brzine, otvorenog pristupa kernelu itd. (Stouff, 2022).
Flatpak je već uvelike prihvaćen na Desktopu od strane SteamOS-a, EndlessOS-a, Fedore i mnogih drugih.

Snap paketi su zahvaljujući dobroj podršci za serverske pakete vrlo dobro prihvaćeni među poslužiteljskim administratorima. Dio zajednice nesretan je zbog fragmentacije koju donosi postojanje tako sličnih načina pakiranja i distribucije aplikacija, ali situacija je svakako bolja nego što je bila prije nastanka modernih načina distribucije i pakiranja aplikacija.

Za očekivati je da će se sustavi distribucije, kao i procesi dijeljenja aplikacija ubrzano razvijati i evoluirati u robusna rješenja koja će biti opće prihvaćena. Distribucije poput EndlessOS-a i SteamOS-a te njihovo korištenje primarno Flatpak paketa zbog jednostavnosti instalacije i sigurnosti korištenja ukazuju na nove načine korištenja Linuxa namijenjene djeci i sve široj grupi ljudi.

Napredak u ovom aspektu GNU/Linux sustava bio je nužan za rast Linux desktopa kao ozbiljne opće prihvaćene platforme. Na temelju vlastitog istraživanja za potrebe ovoga rada i iskustva korištenja Linux OS-a dolazim do zaključka da su moderni načini pakiranja i distribucija aplikacija ubrzali razvoj ostalih aspekata GNU/Linux operativnih sustava i učinili GNU/Linux sustave pristupačnijima novim korisnicima.

7. Literatura

App submission. (bez dat.). U Flathub wiki. Preuzeto 10.09.2022. s https://github.com/flathub/flathub/wiki/App-Submission

AppImagehub.com (bez dat.) Preuzeto 10.09.2022. s https://appimagehub.com
Arch user repository (bez dat.). U ArchWiki. Preuzeto 10.9.2022. s https://wiki.archlinux.org/title/Arch_User_Repository

Dandrea, E. (21.01.2019). An Introduction To Snaps [Video file]. Preuzeto 10.09.2022. s https://www.youtube.com/watch?v=R6wrP-QEI3k

Electronforge.io (bez dat.) Preuzeto 10.09.2022. s https://www.electronforge.io/

ElectronJS.org (bez dat.). Preuzeto 10.09.2022. s https://electronjs.org

Flatseal (2022). Preuzeto 10.09.2022. s https://flathub.org/apps/details/com.github.tchx84.Flatseal

Getting started. (bez dat.). U Snapcraft documentation. Preuzeto 10.09.2022. s https://snapcraft.io/docs/getting-started

How to Create Your Own Repository for Packages on Debian (2022). Preuzeto 10.09.2022. s https://linuxopsys.com/topics/create-your-own-repository-for-packages-on-debian

McLasen (2018). Flatpak - a look behind the portal [Blog post]. Preuzeto 10.09.2022. s https://blogs.gnome.org/mclasen/2018/07/19/flatpak-a-look-behind-the-portal/

Netmarketshare (bez dat.) Operating System Market Share. Preuzeto 10.9.2022. s https://www.netmarketshare.com/

Package management stystem (bez dat.). U Fedora Docs. Preuzeto 10.9.2022. s https://docs.fedoraproject.org/en-US/quick-docs/package-management/

Patlan, E. (2020). How to create your own repositories for packages [Blog post]. Preuzeto 10.09.2022. s https://www.percona.com/blog/2020/01/02/how-to-create-your-own-repositories-for-packages/

Prakash, A. (2021). Using PPA in Ubuntu Linux [Complete Guide] Preuzeto 10.9.2022. s https://itsfoss.com/ppa-guide/

Publishing. (bez dat.). U Flatpak documentation. Preuzeto 10.09.2022. s https://docs.flatpak.org/en/latest/usb-drives.html

Robertson, B., (01.03.2021.). Arch User Repository: How Does It Really Work [Video file]. Preuzeto 10.09.2022. s https://www.youtube.com/watch?v=VbsmQFNx5n4

Statcounter (bez dat.) OS Marketshare. Preuzeto 10.9.2022. s https://gs.statcounter.com/os-market-share/desktop/worldwide/

Stouff, N. (19.01.2022). FLATPAK is the FUTURE of LINUX application distribution [Video file]. Preuzeto 10.09.2022. s https://www.youtube.com/watch?v=zs9QpPKDw74&

Stouff, N. (23.06.2020). Flatpak vs Snaps vs AppImage vs Packages - Linux packaging formats compared [Video file]. Preuzeto 10.09.2022. s https://www.youtube.com/watch?v=9HuExVD56Bo

VSCodium (bez dat.) VSCodium. Preuzeto 10.09.2022. s https://vscodium.com/#install

8. Popis Slika