ŠOLSKI CENTER VELENJE

ELEKTRO IN RAČUNALNIŠKA ŠOLA

Trg mladosti 3, 3320 Velenje

MLADI RAZISKOVALCI ZA RAZVOJ ŠALEŠKE DOLINE

RAZISKOVALNA NALOGA

HACKINTOSH

Tematsko področje: TEHNIŠKE VEDE

Avtorja:

Edah Terzić, 2. letnik

Tobija Žuntar, 2. letnik

Mentorja:

Samo Železnik

Uroš Remenih

Velenje, 2021

Raziskovalna naloga je bila opravljena na Šolskem centru Velenje, Elektro in računalniški šoli, leta 2021.

Mentorja: Samo Železnik, Uroš Remenih

Datum predstavitve: april 2021

KLJUČNA DOKUMENTACIJSKA INFORMACIJA

ŠD ŠC Velenje, šolsko leto 2020/2021

KG Hackintosh

AV TERZIĆ, Edah / ŽUNTAR, Tobija

SA ŽELEZNIK, Samo / REMENIH, Uroš

KZ 3320 Velenje, SLO, Trg mladosti 3

ZA ŠC Velenje, Elektro in računalniška šola, 2021

LI 2021

IN **HACKINTOSH**

TD Raziskovalna naloga

OP 54 str., 36 sl., 33 vir.

IJ SL

JI sl / en

AI Učenje razvoja aplikacij za Applove platforme je v veliki meri oteženo z omejenim dostopom do programske opreme, ki je na voljo samo na precej dragih Applovih napravah. Da bi omogočili razvoj tudi tistim, ki tovrstne opreme nimajo, smo se odločili, da poskusimo operacijski sistem macOS in pripadajočo programsko opremo naložiti na vsakdanje računalnike z operacijskim sistemom Windows. Zanimalo nas je predvsem, če je to mogoče doseči kar z uporabo trenutnih šolskih računalnikov; prav tako smo pregledali, kakšne računalnike bi šola morala kupiti za dosego skoraj popolne kompatibilnosti. Preverili smo tudi, kako se ta rešitev obnese v primerjavi s sicer bolj popularno uporabo virtualnih naprav in ali je uporaba takšnih sistemov, četudi v izobraževalne namene, sploh dovoljena.

KEYWORD DOCUMENTATION

ND ŠC Velenje, Elektro in računalniška šola, 2021

CX Hackintosh

AU TERZIĆ, Edah / ŽUNTAR, Tobija

AA ŽELEZNIK, Samo / REMENIH, Uroš

PP 3320 Velenje, SLO, Trg mladosti 3

PB ŠC Velenje, Elektro in računalniška šola, 2021

PY 2021

TI **HACKINTOSH**

DT Research work

NO 54 p., 36 fig., 33 ref.

LA SL

AL sl / en

AB Teaching software development for Apple platforms is constrained by the hard requirement of software which runs exclusively on Apple's costly hardware. We have therefore decided to attempt to install the macOS operating system and encompassing software on usual Windows-based computers to give all students the ability to try out development for Apple platforms. We were particularly interested in getting this to work with the currently used school computers and have researched what hardware the school would have to use to ensure nearly complete compatibility when building new computers. We have also checked whether this solution outperforms usage of virtual machines. We have investigated whether the usage of such systems is legal for educational purposes.

KAZALO VSEBINE

KAZALO SLIK

1. UVOD

Naš cilj je omogočiti razvoj programov in aplikacij za operacijski sistem Apple macOS, pri čemer bomo uporabljali osebne računalnike, ki niso znamke Apple. Dijaki bodo tako imeli priložnost, da se podrobneje spoznajo z razvojem aplikacij za platforme macOS, iOS in iPadOS. Za uporabo tega načina smo se odločili, ker so se pri razvoju aplikacij v preteklosti virtualne naprave izkazale za premalo zanesljive in prepočasne za uporabo, prav tako imajo številne omejitve glede povezljivosti naprav in uporabe določenih funkcij.

V raziskovalni nalogi smo se osredotočili na delo s šolskimi računalniki ter na uporabo sistemov Hackintosh na splošno. Poglobili smo se v:

- tehnično delovanje sistemov Hackintosh sistemov,
- poskus namestitve sistema Hackintosh na šolske računalnike,
- uporabo programske opreme za razvoj aplikacij,
- legalnost uporabe sistemov Hackintosh na splošno in v izobraževalne namene.

Učinkovitost delovanja in zanesljivost naše rešitve smo primerjali tudi z najpopularnejšo alternativo, virtualnimi napravami. Tu smo ugotavljali predvsem, kakšna je razlika v hitrosti delovanja za naše potrebe (razvoj aplikacij).

Trudili smo se vzpostaviti najbolj združljiv sistem s uporabljenimi šolskimi računalniki. Pri tem smo preizkusili tudi, če je možno vzpostaviti sistem dvojnega zagona (ang. dual boot) s sistemom Windows. To bi vse skupaj olajšalo in ne bi potrebovali stoodstotne zanesljivosti računalnikov, saj bi se vsa navadna opravila izvedlo v operacijskem sistemu Windows, v sistemu macOS pa tako samo tista, ki jih je možno opraviti izključno v njem.

1.1 HIPOTEZE

V okviru raziskovalne naloge smo si zastavili naslednje hipoteze:

1. Na šolske računalnike lahko namestimo macOS in ga uporabimo za razvoj aplikacij.

2. Ob prenovi strojne opreme v računalniških učilnicah bi bilo smiselno kupiti strojno opremo, ki omogoča optimalno uporabo sistema macOS.

3. Uporaba sistemov Hackintosh je v izobraževalne namene dovoljena.

2. PREGLED OBJAV

2.1 MAC OS

Mac OS (tudi macOS in OS X) je Applov operacijski sistem za računalnike Macintosh. [1] Temelji na jedru UNIX (bolj natančno Darwin in XNU) ter je tudi certificiran kot operacijski sistem UNIX. Darwin temelji na jedru Mach in spada v skupino operacijskih sistemov BSD UNIX, kamor spadajo tudi operacijski sistemi FreeBSD, OpenBSD in NetBSD [2]. Prva različica se je imenovala Mac OS X 10.0 Cheetah in je bila izdana leta 2001. Predhodnik se je imenoval Mac OS Classic in je temeljil na lastni arhitekturi.

Mac OS je čez leta doživel kar dve spremembi arhitekture procesorjev. V osnovi je bil zasnovan za delovanje na procesorjih tipa PowerPC, ki so jih so ob izdaji prve različice uporabljali takratni Maci. Leta 2007 je Apple napovedal prehod na arhitekturo Intel in ob tem izdal Mac OS X 10.5 Leopard. [3] Ta je med drugim vseboval komponento, imenovano Rosetta, ki je ob izvajanju programov prevajala navodila. Mac OS X 10.5 je bila tudi zadnja različica, ki je imela podporo za Mace s procesorji PowerPC. [4] Rosetta je bila komponenta sistema do različice 10.6 Snow Leopard, ko se je prehod na Intelove procesorje dokončal.

Prehod na Intelove procesorje je tako tudi spodbudil eksperimentiranje z nameščanjem na navadne računalnike in tako pomenil začetek Hackintosha. Pri tem je bilo potrebno najti obhode za najrazličnejše prepreke.

Leta 2020 je Apple predstavil prehod na lastne procesorje Apple Silicon in hkrati operacijski sistem macOS 11 Big Sur, ki je prvi s podporo zanje [5]. Ta vsebuje komponento z imenom Rosetta 2, ki je zelo podobna svoji prvotni različici, vendar se razlikuje v tem, da prevaja iz navodil za Intelove procesorje na navodila za procesorje ARM. Poleg tega pa imajo vse vgrajene aplikacije Intel in Apple Silicon različice izvedljivih datotek. [6]

Apple tako načrtuje prehod celotne ponudbe računalnikov Mac na procesorje Apple Silicon, kar bi pomenilo konec sistemov Hackintosh na računalnikih s procesorji Intel in AMD.

2.2 HACKINTOSH

Pod ime Hackintosh po definiciji spada vsak računalnik, ki ni Applov in ki poganja operacijski sistem macOS. Beseda Hackintosh je skovanka iz besed »hack« in »Macintosh«. Sistemi Hackintosh obstajajo že odkar Apple v svojih računalnikih uporablja Intelove procesorje, saj so ti prisotni tudi na »normalnih« računalnikih. Veliko ostalih delov strojne opreme je identične tistim v PC-jih, edina večja razlika so posamezni čipi, ki jih Apple uporablja za razne napredne funkcije. Večina izmed teh je sicer za zagon in delovanje samega sistema zelo velikega pomena, zato se jih v sistemih Hackintosh emulira z uporabo posebnih gonilnikov. Podobni gonilniki se uporabljajo tudi za zagotavljanje podpore strojni opremi, ki podpore nima vključene v sam operacijski sistem, ker je Apple preprosto ne uporablja v svojih računalnikih. [7]

Razlogov za uporabo sistema Hackintosh je več:

- slaba razširljivost in nadgradljivost Applovih računalnikov,
- visoka cena Applovih računalnikov,
- uporaba dvojnega zagona (Dual Boot) med sistemi macOS, Windows ali Linux,
- razvoj in testiranje aplikacij za Applove platforme
- eksperimentiranje in učenje o delovanju operacijskega sistema macOS.

Ker gre pri sistemih Hackintosh za uporabo sicer precej ekskluzivne programske opreme na poceni in prosto dostopnih računalniki, je to seveda za veliko ljudi zelo privlačno. Apple tega seveda ne odobrava, več o tem pa smo raziskovali tudi sami. [8]

2.3 VIRTUALNE NAPRAVE

Virtualne naprave so navidezne naprave, ki tečejo na vrhu operacijskega sistema in imajo dodeljen del sistemskih virov računalnika. Poznamo več načinov izvajanja virtualnih naprav. Virtualne naprave se uporabljajo predvsem za razdeljevanje sistemskih virov med aplikacijami na strežniških sistemih ter poganjanje različnih operacijskih sistemov na gostiteljskem računalniku. Nas je zanimal predvsem slednji način uporabe, saj se z njim lahko na računalnike namesti tudi nekatere operacijske sisteme, s katerimi računalniki sami po sebi niso združljivi, oziroma je zagotavljanje združljivosti zelo težavno. [9]

Pri tem je pomembno predvsem dejstvo, da virtualizatorji, ki se pogosto uporabljajo za poganjanje operacijskega sistema macOS, emulirajo večino negeneričnih delov strojne opreme. Tu gre predvsem za to, da je pri uporabi virtualnih naprav na sistemu Windows zelo težko zagotoviti direkten dostop do grafične kartice v računalniku gostiteljskega sistema. To je bolje izvedeno na operacijskih sistemih Linux in *BSD, ki pa za nas ne pridejo v poštev (šolski računalniki po večini uporabljajo Windows). [10]

Popularni virtualizacijski programi, kot sta VirtualBox in VMware Workstation, skrijejo, oziroma zaklenejo možnost namestitve sistema macOS. [11] [12] Težava je v tem, da Apple v licenci vse svoje programske opreme navaja, da je to dovoljeno namestiti samo na njihovo strojno opremo. Razvojna podjetja se problemom najlažje izognejo tako, da namestitev sistema macOS na računalnikih, ki tega že v osnovi ne poganjajo, preprosto onemogočijo. Izjema tukaj je VirtualBox, ki možnosti ne skrije, je pa izrecno povedano, da takšna uporaba ni podprta. Za ostale virtualizacijske programe pa obstajajo razni “patchi”, ki namestitev sistema macOS omogočajo na vseh gostiteljskih operacijskih sistemih. [13]

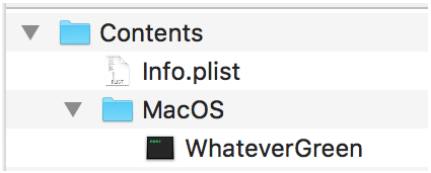
2.4 VSTAVKI KEXT

Kernel **EXT**ensions so vsestranski vstavki, ki jih macOS uporablja za zagotavljanje dodatne funkcionalnosti jedru sistema. Dejansko gre za mape s končnico *.kext*, ki vsebujejo različne pomožne datoteke. [14] Ti vstavki se najpogosteje uporabljajo za razširitev podpore strojni opremi. Za nas so pomembni predvsem, ker so eden izmed najpomembnejših gradnikov sistemov Hackintosh. V sistemu se nahajajo na različnih lokacijah:

- vstavki, ki so del sistema, so v mapi */System/Library/Extensions*;
- vstavki, ki jih namesti orodje MultiBeast, se nahajajo v mapi */Library/Extensions*;
- vstavki, ki morajo biti prisotni že pred zagonom sistema (za predstavitev tega kot Mac računalnika v našem primeru), se nahajajo v posebni mapi zagonskega programa (npr. za Clover: */EFI/CLOVER/kexts*).

Nekateri izmed vstavkov, ki jih v našem primeru uporabljamo, so dejansko zakrpane različice Applovih, tak primer je AppleHDA.kext, ki se uporablja za zagotavljanje zvoka.

Osnovna struktura vstavkov KEXT je sestavljena iz mape *Contents*, v kateri se nahajata datoteka *Info.plist* in mapa *MacOS*. V slednji je ena izvedljiva datoteka, ki vsebuje celotno kodo vstavka.



Slika 1: Struktura vstavkov KEXT

Datoteka *info.plist* sistemu pove različne podatke o vstavku, npr. podprto različico sistema in zahtevan tip procesorja. Datoteka sama je datoteka XML, ki vsebuje gnezdene kombinacije zapisov Key-Type-Value.

Key	Type	Value
Information Property List	Dictionary	(23 items)
BuildMachineOSBuild	String	17D102
Localization native development re...	String	en
Executable file	String	WhateverGreen
Bundle identifier	String	as.vit9696.WhateverGreen
InfoDictionary version	String	6.0
Bundle name	String	WhateverGreen
Bundle OS Type code	String	KEXT
Bundle versions string, short	String	1.2.9
Bundle creator OS Type code	String	????
CFBundleSupportedPlatforms	Array	(1 item)
Bundle version	String	1.2.9
DTCompiler	String	com.apple.compilers.llvm.clang.1_0
DTPlatformBuild	String	9F2000
DTPlatformVersion	String	GM
DTSDKBuild	String	17E189
DTSDKName	String	macosx10.13
DTXcode	String	0941
DTXcodeBuild	String	9F2000

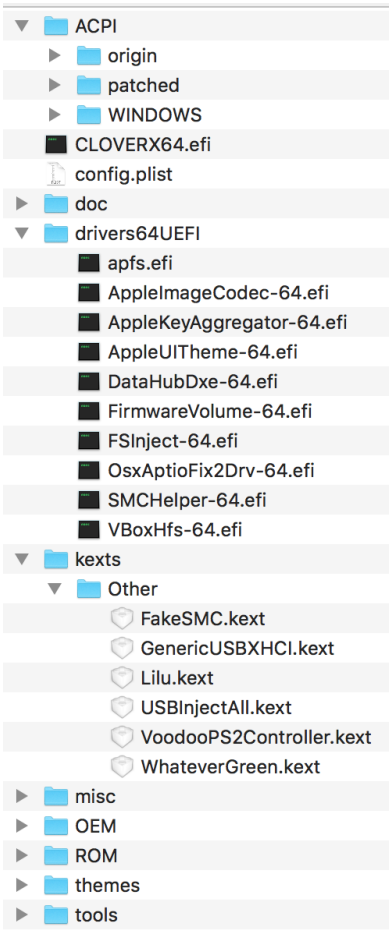
Slika 2: Datoteka info.plist

2.5 CLOVER

Clover je zagonski nalagalnik (angl. bootloader), ki omogoča zaganjanje različnih različic sistema macOS na navadnih PC-računalnikih. Ker omogoča tudi zagon drugih operacijskih sistemov, se ga lahko uporablja za postavitev sistema z dvojnim zagonom, kjer pri zagonu računalnika izberemo med sistemoma macOS in Windows oz. Linux. [15]

Clover pa za razliko od drugih zagonskih nalagalnikov, kot je npr. GRUB, vsebuje funkcije, namenjene nalaganju sistemskih vstavkov KEXT med začetno fazo zagona. Nekateri vstavki se morajo namreč naložiti še pred začetkom zagona (takšna primera sta FakeSMC in njegova novejša alternativa VirtualSMC), saj poskrbijo za to, da se računalnik operacijskemu sistemu predstavi kot pravi Mac. Clover je mogoče namestiti ročno s prenosom in zagonom namestitvenega programa ali z uporabo vnaprej pripravljenih celostnih rešitev, ki poleg namestitve še uredijo različne konfiguracijske datoteke z namenom olajšanja namestitve. Primer takega orodja je program UniBeast. [16]

Ob namestitvi na EFI particijo diska Clover ustvari strukturo map, ki je podobna tej na sliki:



Slika 3: Cloverjeva struktura map

Večina izmed teh map je ob namestitvi prazna, oziroma vsebujejo "placeholder" datoteke. Za nas so največjega pomena mape *ACPI*, *drivers64UEFI* in seveda *kexts/Other*. Pomembna je tudi datoteka *config.plist*, saj vsebuje Cloverjevo celotno konfiguracijo. [17]

V mapi *ACPI* se v podmapah *origin* in *patched* nahajajo originalne in zakrpane datoteke s tabelami ACPI. Te lahko kasneje uporabimo za odpravilo raznovrstnih težav s konfiguracijo opreme na naši matični plošči ter zagotovitev ustrezne podpore za upravljanje z računalnikovo porabo energije. Izvorne datoteke se v mapi *origin* ne ustvarijo same od sebe, lahko pa jih ustvarimo s pritiskom na tipko F4, medtem ko je prikazan Cloverjev zagonski meni. [17]

Mapa *drivers64UEFI* vsebuje gonilnike UEFI za strojno opremo. Ti so zahtevani zato, da lahko Clover zazna različne naprave za zagon ter med drugim tudi, da lahko na zagonskem meniju uporabljamo miško.

Mapa *kext/Other* vsebuje vse *.kext* datoteke, ki morajo biti naložene in prisotne že v zgodnji fazi zagona sistema. V njej se mora nahajati vsaj datoteka *FakeSMC.kext* ali *VirtualSMC.kext*, pogosto pa v to mapo dodamo še *Lilu.kext*, *WhateverGreen.kext* in *AppleHDA.kext*. Ti namreč poskrbijo za grafično pospeševanje in podporo zvočni kartici. V mapi *kext* so sicer prisotne še mape s številkami različic sistema macOS (*10.6, 10.7, 10.8, 10.9 ...*). Te se načeloma ne uporabljajo, namenjene pa so temu, da lahko v primeru dvojnega zagona različnih različic macOS-a zaradi združljivostnih razlogov uporabljamo različne različice vstavkov KEXT.

Poleg tega pa je tukaj še datoteka *clover.plist*, ki je datotekaXML s Key-Type-Value strukturo in vsebuje vse nastavitve samega zaganjalnika. Med drugim je tu notri definiran tip računalnika, ki bo prikazan v macOS-u (npr. *iMac 27in late 2013*), popravki za poti naprav APCI, razni grafični popravki, nastavitve vmesnika Cloverja itd. Za urejanje te datoteke obstajajo tudi namenski pripomočki, kot je npr. Clover Configurator. Ti poskrbijo tudi, da v nobenem primeru ni mogoče shraniti neveljavnih kombinacij možnosti. [18]

Clover je združljiv tako s sistemom BIOS kot z UEFI, vendar je na slednjem uporaba veliko lažja. Gre samo za to, da je patricija EFI na sistemih UEFI navadna patricija FAT32, do katere lahko brez večjih težav dostopamo. Tako lahko naredimo varnostne kopije naše konfiguracije Cloverja, prav tako lahko brez težav dodajamo in odstranjujemo vstavke KEXT. Clover podpira zagon vseh sistemov macOS od različice 10.6 dalje.

Trenutno je v razvoju in nekaterih primerih tudi že v aktivni uporabi novejša alternativa Cloverju, imenovana OpenCore. Prednost te je predvsem v tem, da je veliko čisteje zasnovana. Tako nima večine zastarelih in pogosto težavnih popravkov, ki so prisotni v Cloverju. OpenCore pa kljub svoji moderni zasnovi ohranja podporo za starejše različice macOS-a in jo celo izboljšuje — podpira namreč vse različice macOS od prve, ki je dobila podporo za Intelove procesorje (10.4 Tiger). Kljub svojim prednostim pa OpenCore še vedno ni v široki uporabi, saj ji še vedno manjkajo nekatere funkcije Cloverja, kar otežuje nadgradnjo iz njega.

2.6 XCODE

XCode je Applovo integrirano razvojno okolje (IDE) za razvoj programske opreme za Applove operacijske sisteme (macOS, iOS, watchOS, tvOS in iPadOS). Na voljo je izključno na operacijskem sistemu macOS, kar je tudi razlog, da je programsko opremo za Applove naprave možno dokončati samo na njem. [19]

Razvojno okolje samo po sebi ni nič posebnega, ima vse pričakovane funkcije, kot npr. samodokončanje kode, sprotni prikaz napak, predlaganje popravkov itd. Od ostalih razvojnih okolij pa se razlikuje v tem, da so vanj vgrajeni preizkušanje, nameščanje in objavljanje aplikacij. Za razvoj programske opreme za Applove sisteme obstaja veliko drugih razvojnih okolij (eno takšnih je AppCode), nekatera celo omogočajo razvoj aplikacij z uporabo drugih programskih jezikov (npr. Xamarin za C#). [20] Še vedno pa je pri vseh teh razvojnih orodjih zadnja faza izgradnje aplikacij izvedena s pomočjo XCode. Tako na primer ni mogoče aplikacije za iOS dokončno zgraditi na operacijskem sistemu Windows in jo namestiti na telefone iPhone.

Nekateri se v tem primeru, kot že prej omenjeno, poslužujejo uporabe virtualnih naprav, vendar je s temi vrsta težav. Za vsako novo različico aplikacije je namreč potrebno prekopirati vse datoteke, jih prevesti ter overiti z digitalnim potrdilom. To je zelo zamudno, še posebej, če to počnemo z uporabo virtualne naprave, ki je sama po sebi zelo počasna. Še večja zmeda nastane, kadar želimo aplikacije testirati na fizičnih napravah, saj povezave USB v virtualnih napravah s sistemom macOS niso dovolj zanesljive. [21]

V okviru naše raziskovalne naloge smo tako preizkusili tudi, kako zanesljivo in učinkovito deluje XCode na sistemih Hackintosh.

3. POTEK IZVAJANJA

3.1 PRIPRAVA NAMESTITVENEGA MEDIJA

Za namestitev operacijskega sistema macOS na računalnike, ki niso Applovi, potrebujemo poseben namestitveni medij, ki ob zagonu obide navadne gonilnike in jih zamenja s svojimi, prav tako pa naš računalnik sistemu predstavi kot Applovega.

Vsak tak namestitveni medij ima poleg slike sistema tudi svojo particijo EFI (oziroma zagonski zapis MBR na BIOS sistemih), ki vsebuje prilagojen zaganjalnik sistema (angl. bootloader). [22]

Trenutno sta po večini v uporabi naslednja dva zaganjalnika:

- Clover, ki je starejši in poleg načina EFI zelo dobro podpira tudi ti. Legacy oziroma BIOS način (originalno je bil napisan kot zamenjava za starejši Chimera zaganjalnik za BIOS sisteme).
- OpenCore, ki je novejša alternativa Cloverju, ki se uporablja predvsem v načinu EFI in odpravlja veliko omejitev Cloverja. Je še v razvoju, vendar je že dovolj stabilen za normalno uporabo. Sicer podpira tudi način BIOS, vendar ne tako dobro kot Clover.

Ker smo želeli sistem namestiti na šolske računalnike, ki imajo po večini še BIOS in ne UEFI, smo najprej poskusili s Cloverjem. Pri sami ustvaritvi zagonskega medija si lahko pomagamo z orodjem UniBeast (tonymacx86.com), ki že sam vključi nekatere osnovne zahtevane gonilnike; lahko pa vse skupaj naredimo tudi ročno, pri čemer ne smemo pozabiti vključiti vseh gonilnikov.

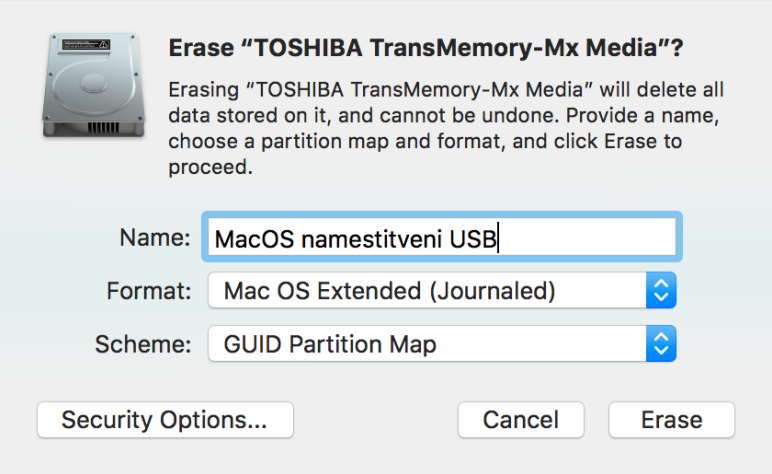
Za namestitev potrebujemo tudi celotno kopijo operacijskega sistema macOS. To lahko sicer dobimo na različne načine, vendar je najbolje (kar smo tudi storili), da z uporabo dejanskega Mac računalnika prenesemo macOS kar iz App Stora. Pri tem gre poudariti tudi, da čeprav so lahko neuradne oziroma modificirane različice sistema macOS v nekaterih primerih uporabne, z njihovim prenašanjem, nameščanjem in uporabo tvegamo okužbo z zlonamerno programsko opremo. Prav tako so modificirane različice macOSa težavne z vidika legalnosti, kompatibilnosti z gonilniki ter pogosto povzročijo blokado (angl. ban) Apple ID računa ob vpisu v App Store in druge Applove storitve. [23]

Po prenosu celotne kopije macOSa (torej aplikacije *Install macOS Mojave.app*) smo pričeli z ustvarjanjem namestitvenega medija. Za to smo potrebovali prazen ključek USB velikosti vsaj 16 GB. Uporabili smo orodje UniBeast, ki smo ga prenesli z uradne spletne strani [24]:

Slika 4: Spletna stran orodja UniBeast

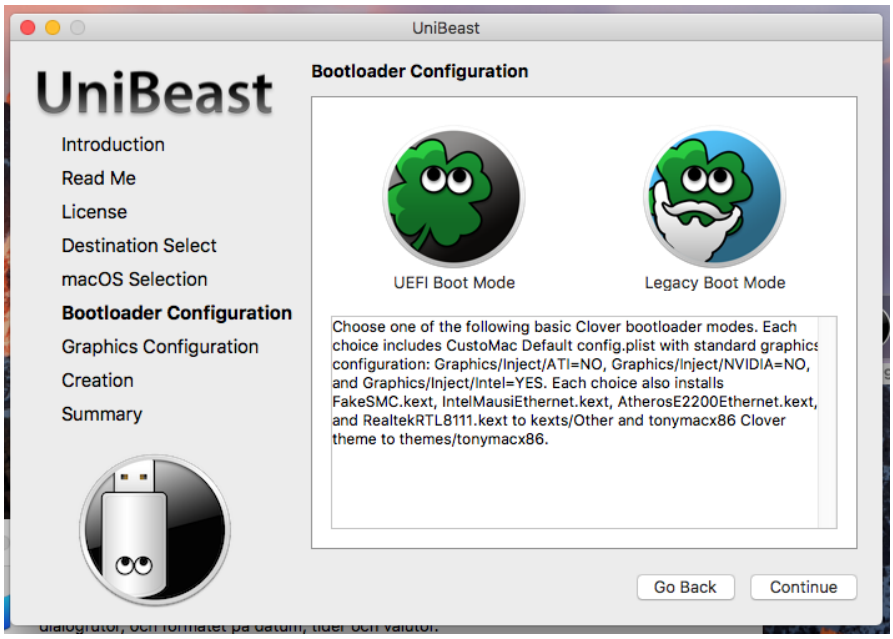
Preden smo začeli z ustvarjanjem zagonskega medija, smo morali na ključku USB pripraviti particije. To smo storili s programom Disk Utility, ki je del operacijskega sistema macOS. Na seznamu naprav smo izbrali naš ključek USB in v orodni vrstici kliknili možnost *Erase* (Izbriši). Uporabili smo naslednje nastavitve:

- število particij: 1
- ime particije: *poljubno*
- format: *Mac OS Extended (Journaled)* oz. *JHFS+,* če delamo s terminalom,
- shemo: *GUID Partition Map*.



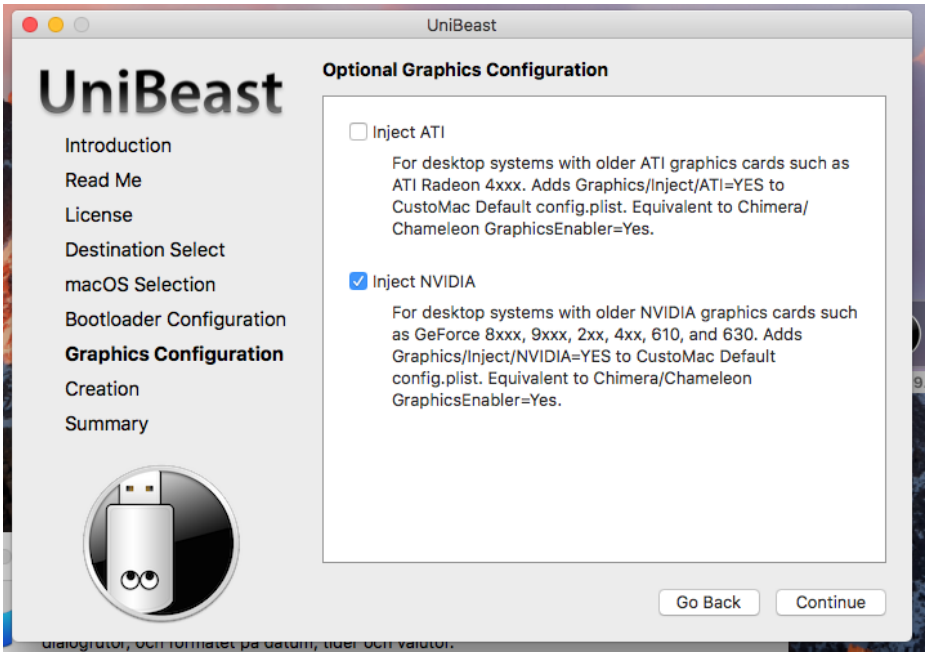
Slika 5: Formatiranje namestitvenega USB-ja

Nato smo z orodjem UniBeast prekopirali namestitvene datoteke na naš USB. V orodju imamo možnost, da izberemo, na katero napravo USB želimo prekopirati datoteke, katero različico sistema macOS želimo uporabiti ter kakšen način zagona podpira naš ciljni sistem (torej BIOS/Legacy način ali UEFI).



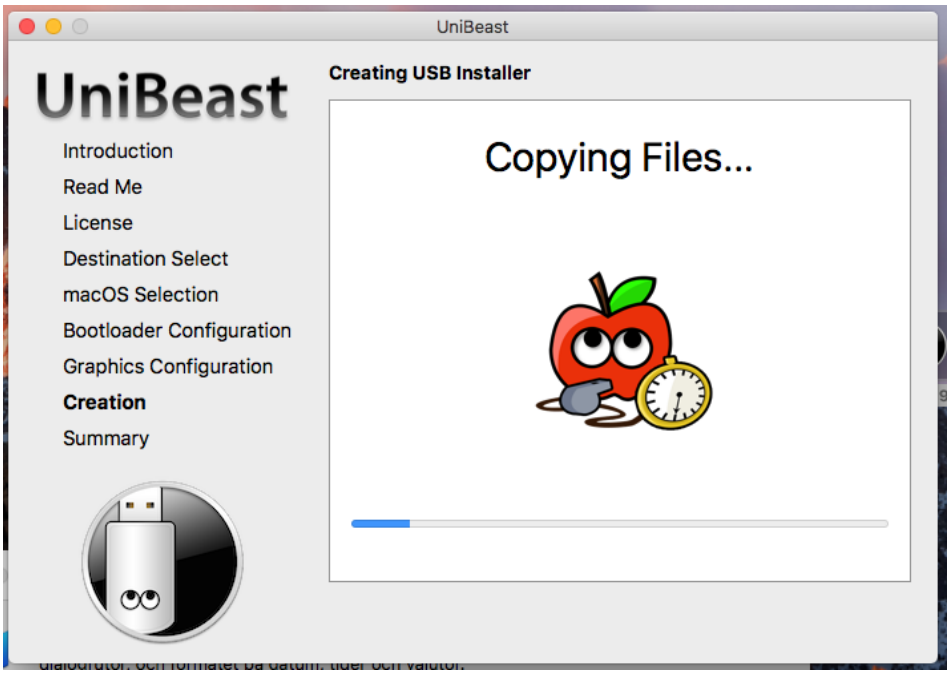
Slika 6: Izbira zagonskega načina orodja UniBeast

Pri uporabi nekaterih grafičnih kartic NVIDIA in ATI lahko že ob izdelavi namestitvenega medija izberemo možnosti za vključitev gonilnikov (da se izognemo težavam ob prvem zagonu). Mi tega nismo potrebovali, zato smo ti dve možnosti pustili izklopljeni.



Slika 7: Izbira za grafične kartice ATI in NVIDIA

Nato smo potrdili izbiro, vnesli skrbniško geslo in počakali, da se vse datoteke prekopirajo – to je trajalo približno eno uro, saj sta bila naš disk in ključek USB zelo počasna.

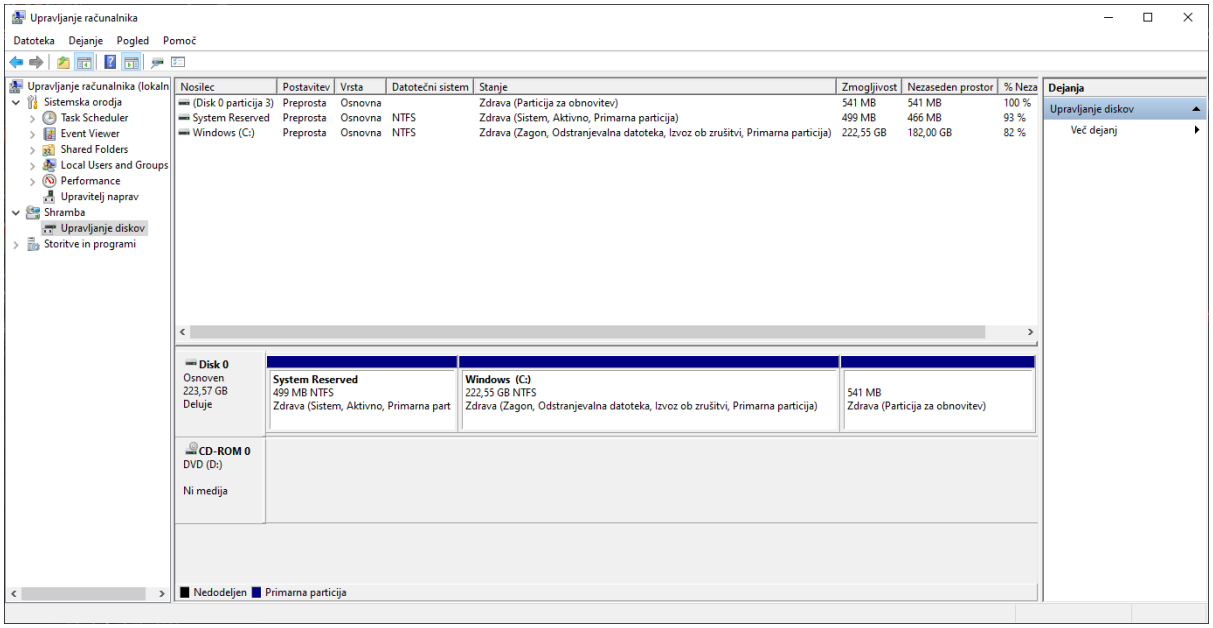


Slika 8: Kopiranje datotek orodja UniBeast

Za tem je bila priprava namestitvenega medija načeloma zaključena, mi pa smo nanj prekopirali še program MultiBeast, ki nam po namestitvi sistema omogoča zelo preprosto namestitev gonilnikov.

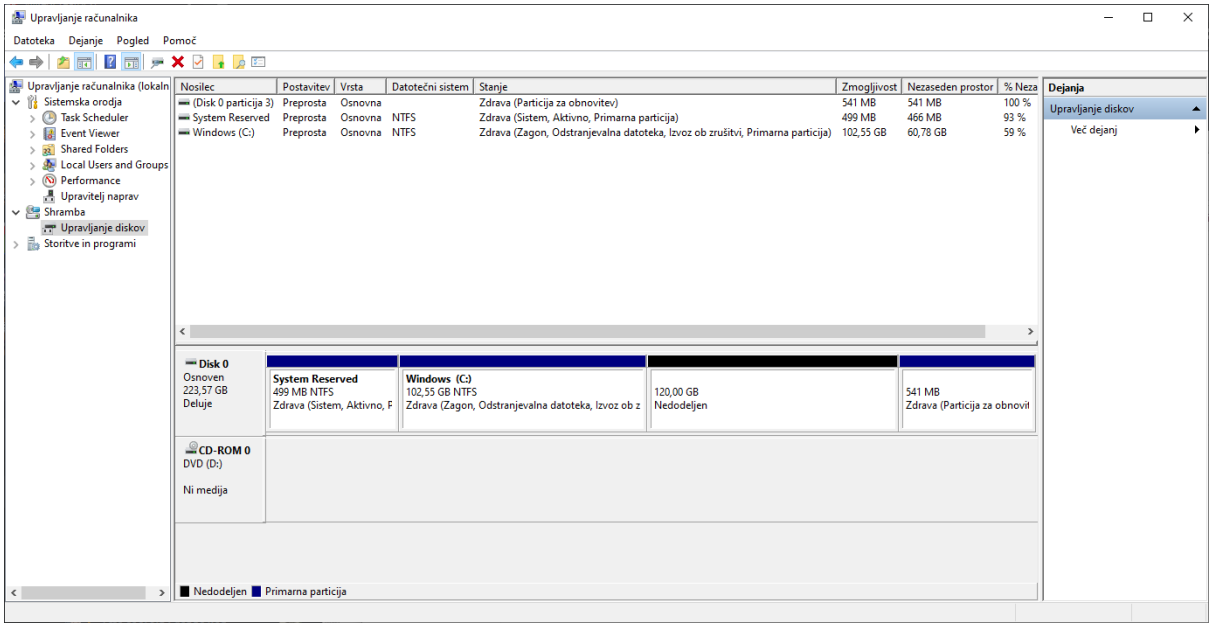
3.2 ZAGON IN NAMESTITEV

Pred zagonom sistema smo skrčili obstoječe particije sistema Windows, da smo naredili prostor na disku za macOS particijo. [25] To smo storili prek vmesnika *Upravljanje računalnika* (tega smo zagnali z ukazom *compmgmt.msc*). Na levi strani smo izbrali *Upravljanje diskov*.



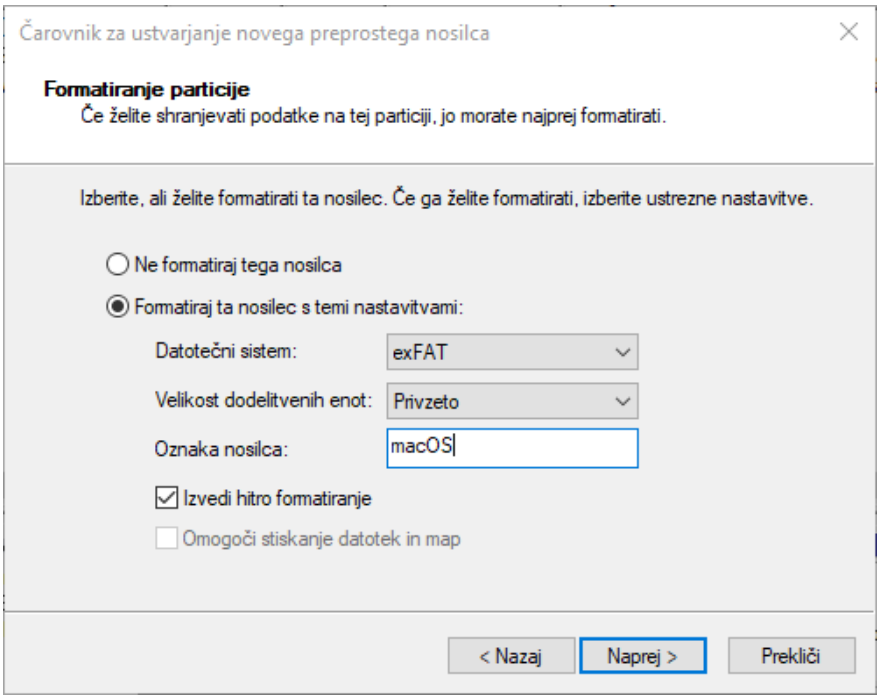
Slika 9: Particije pred spremembami

Na naši particijiWindows smo nato uporabili možnost *skrči*. V pogovornem oknu smo vnesli velikost, torej za koliko megabajtov smo želeli skrčiti particijo. Priporočena velikost macOS particije je 64 GB in več, čeprav je sistem možno namestiti tudi na particijo velikosti okoli 32 GB. Mi smo particijo skrčili za 120 GB:



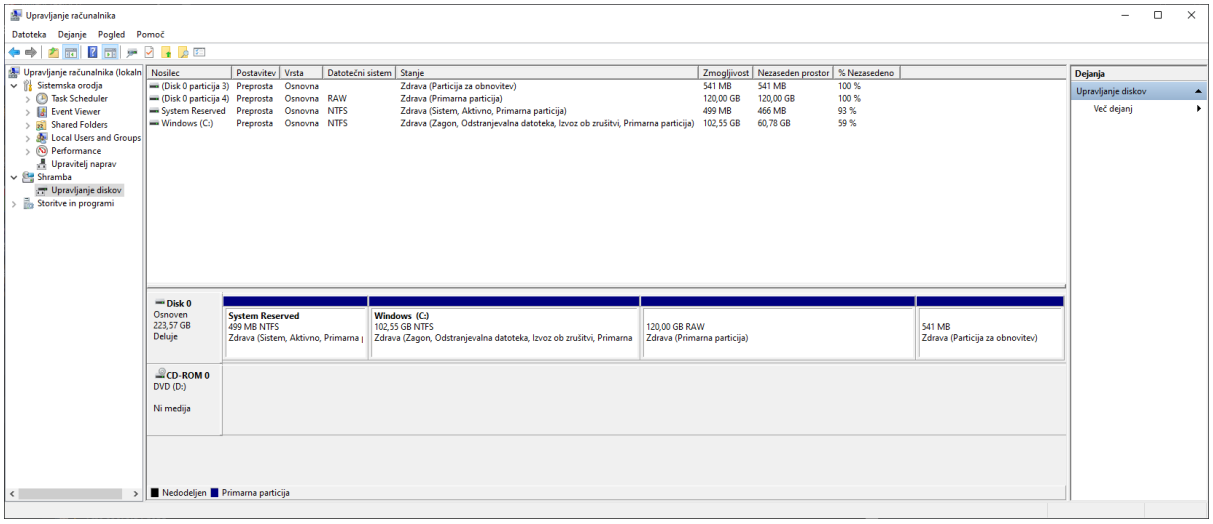
Slika 10: Priprava prostora za particijo

D bi se izognili različnim težavam, smo v praznem prostoru ustvarili *"placeholder"* particijo, ki smo jo kasneje pobrisali in na njenem mestu ustvarili particijo s sistemom macOS (iz sistema macOS samega). Priporočeno je, da pri formatiranju uporabi macOSu poznan datotečni sistem (FAT32, exFAT ...). Particijo lahko sicer pustimo tudi neformatirano.



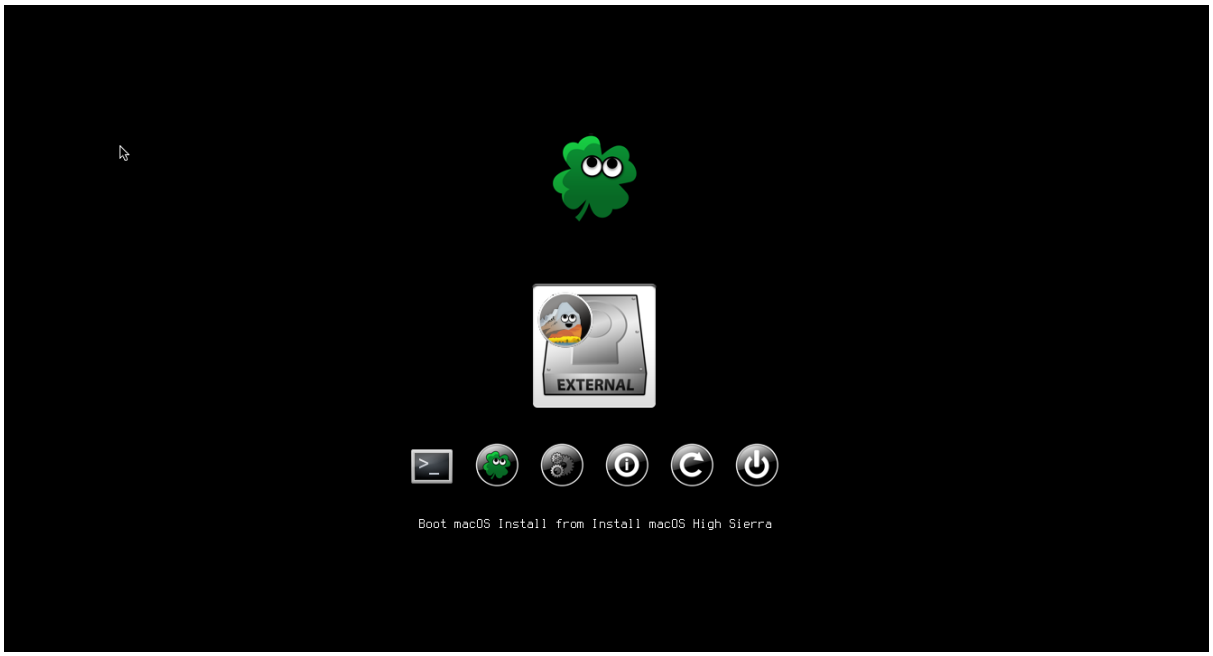
Slika 11: Formatiranje particije placeholder

Naša particijska tabela je za tem izgledala takole:



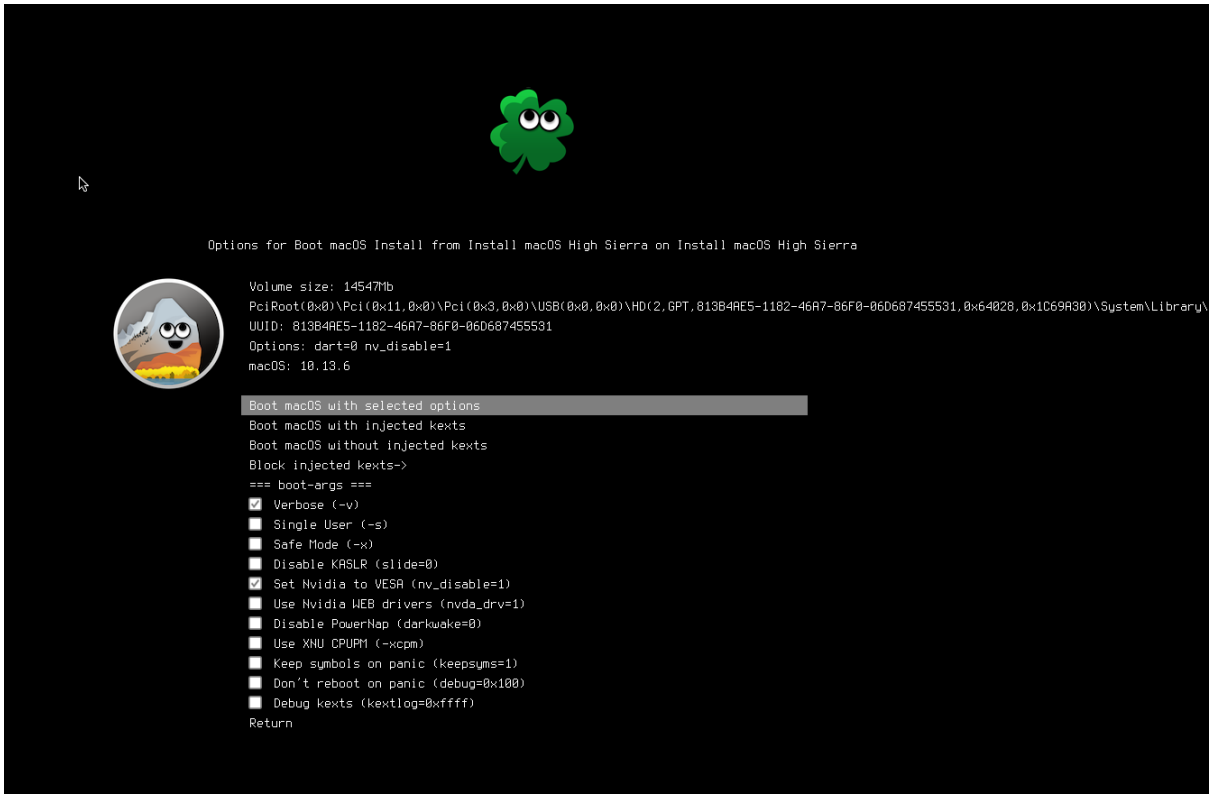
Slika 12: Dokončane particije

Nadaljevali smo z zagonom računalnika, pri katerem smo v zagonskem (angl. boot) meniju izbrali zagon z naprave USB. Čez nekaj časa se je pokazal bootloaderjev meni, v katerem smo izbrali zagonsko napravo (ključek USB).



Slika 13: Cloverjev zagonski meni

Izbrali smo napravo *Install macOS* in pritisnili *presledek*. Pojavil se nam je meni z zagonskimi možnostmi, izmed katerih smo omogočili možnost *Verbose*, ki nam ob zagonu sistema vklopi izpis raznih diagnostičnih sporočil. Ta so nam prišla prav ob napakah, saj smo zaradi njih lahko odkrili vir problema. Na voljo so tudi številne druge možnosti, kot na primer *varni način* oziroma *safe mode*, ki začasno onemogoči vse nenujne gonilnike. [26]

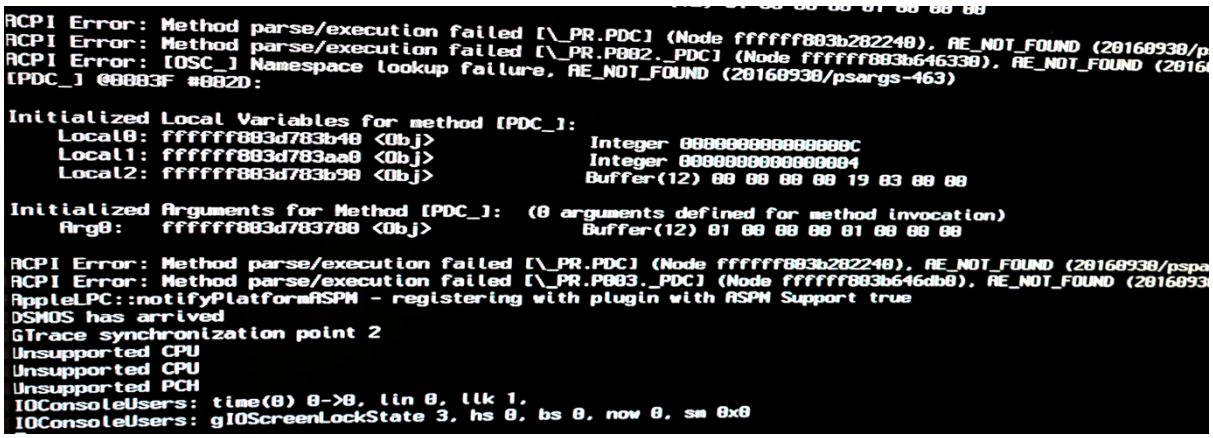


Slika 14: Zagonske možnosti

Med zagonom *Verbose* je sistem izpisal tudi ogromno število sporočil o napakah, ki pa niso bile usodne. Večino teh smo preprosto prezrli — številne izmed njih se izpišejo tudi ob zagonu sistema na pravih Applovih računalnikih.

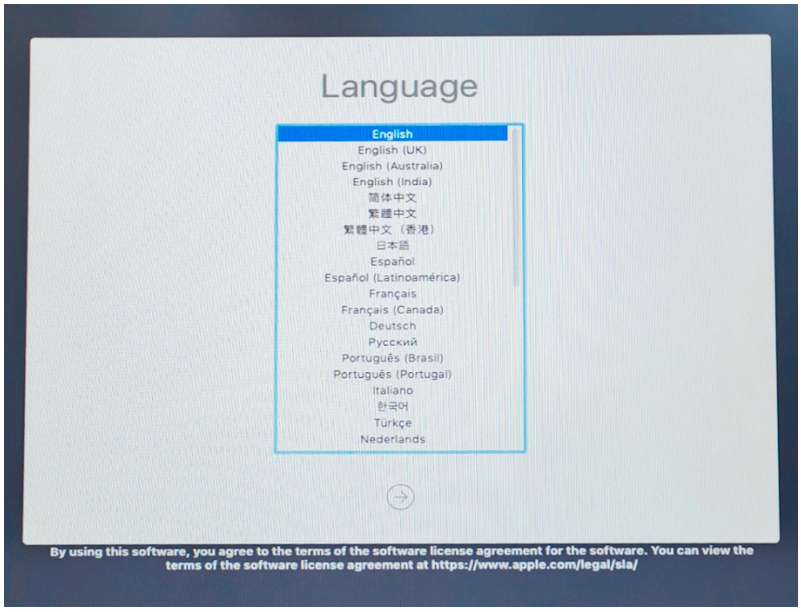
Slika 15: Zagon Verbose

V našem primeru so se pojavile tudi na videz resne napake, kot so npr. "Unsupported CPU" (nepodprt procesor), vendar to sistema pri zagonu sploh ni oviralo.



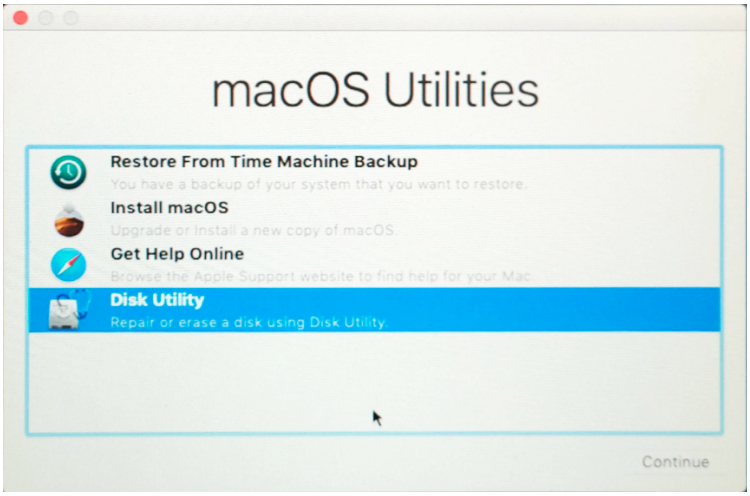
Slika 16: Napake ob zagonu

Čez nekaj časa (okoli deset minut) se nam je na zaslonu za hip pojavil Applov logotip, za tem pa nas je pozdravil grafični vmesnik sistema macOS. V sledečem meniju smo izbrali jezik sistema. Izbira jezikov je sicer bolj slaba, slovenščine med njimi ni.



Slika 17: Izbira jezika

Po izbiri jezika se nam je prikazalo majhno okno z različnimi možnosti:



Slika 18: Izbira orodja

Te možnosti se uporabljajo za naslednje namene:

- *Restore From Time Machine Backup*

  Omogoča obnovitev sistema iz varnostne kopije programa Time Machine. Ta možnost presenetljivo dobro deluje tudi na sistemih Hackintosh.
- *Install macOS*

  Namesti sistem macOS.

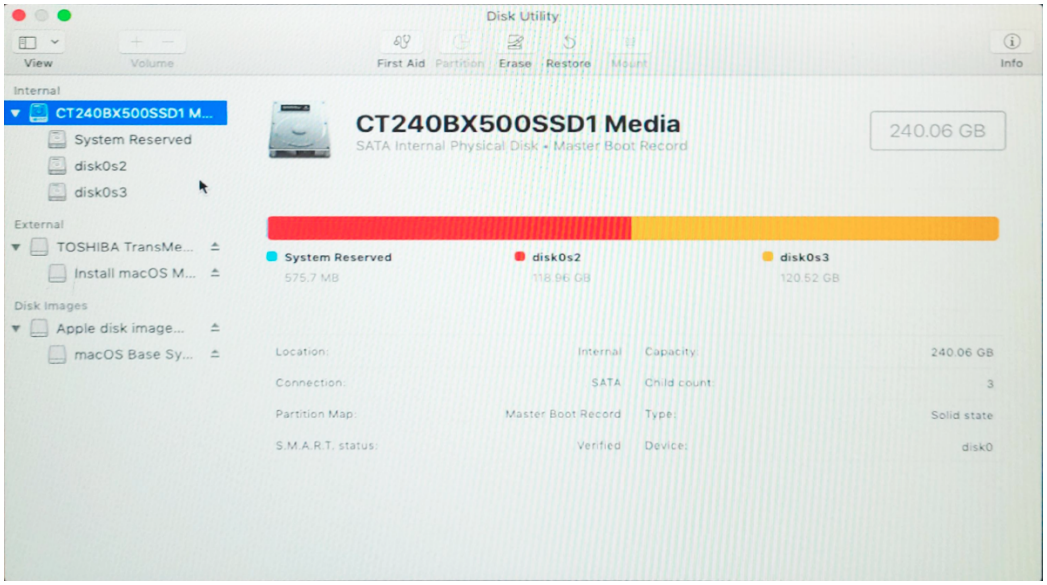
- *Get Help Online*

  Odpre brskalnik Safari in v njem Applovo stran za pomoč in podporo.
- *Disk Utility*

  Omogoča izdelavo in upravljanje s particijami, odpiranje in ustvarjanje slik diska ter obnovitev particij iz teh.

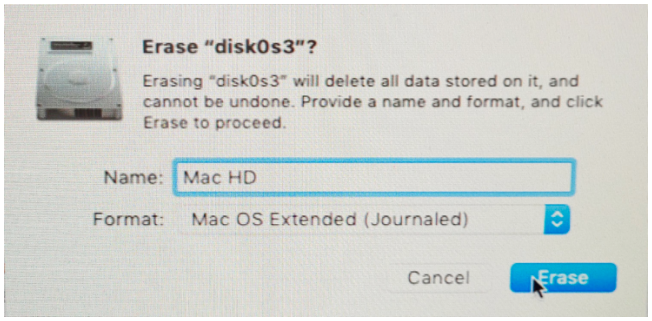
Na voljo sta nam še *Terminal* in *Network Profiler*, do katerih dostopamo prek menija *Utilities* na vrhu zaslona.

Preden smo sploh začeli s postopkom namestitve, smo morali formatirati particijo diska, ki smo jo prej ustvarili s sistemom Windows. V meniju smo izbrali *Disk Utility*:



Slika 19: Pregled particij

Priporočeno je, da jo formatiramo v datotečni sistem *JHFS+*, lahko pa poskusimo tudi novejši *APFS*. Ime particije je lahko poljubno, iz tradicije pa jo lahko poimenujemo kar »Mac HD« oziroma »Macintosh HD«.



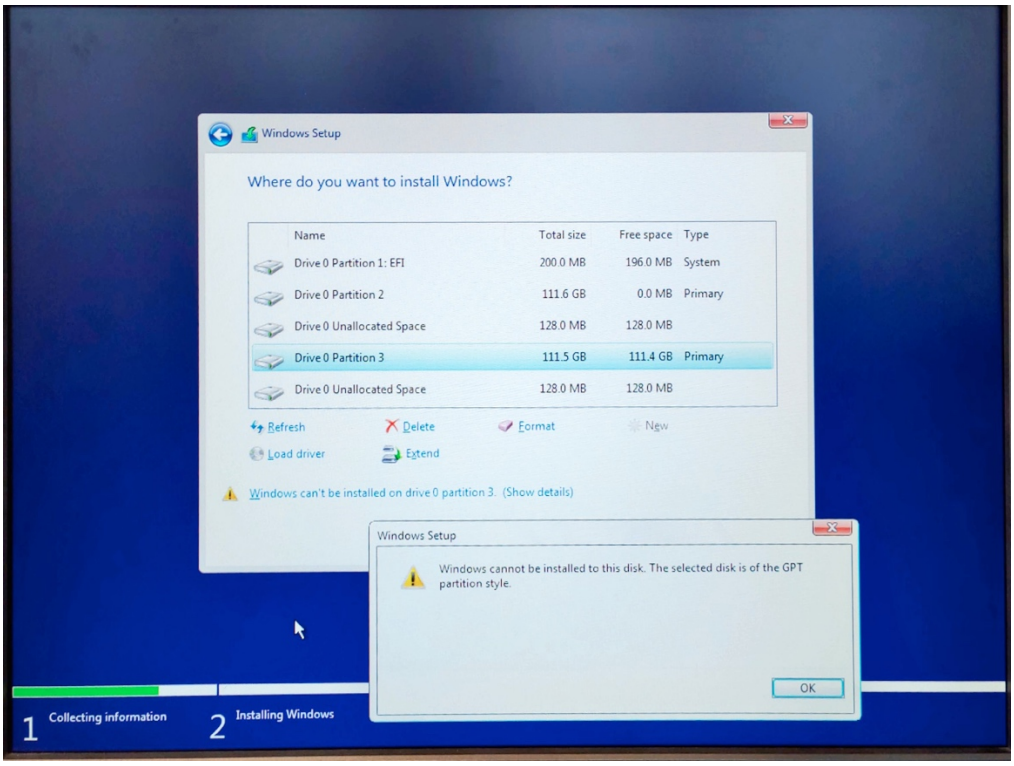
Slika 20: Formatiranje particije za namestitev

Po dokončanju formatiranja smo zaprli *Disk utility* in v glavnem meniju izbrali možnost *Install macOS*.

Od tam naprej se je namestitev nadaljevala samodejno, mi pa smo samo počakali, da se je dokončala. Računalnik se je med tem postopkom večkrat ponovno zagnal (na sistemih BIOS samo 2-krat, na UEFI sistemih 3–4-krat). Po končani namestitvi nas je sistem še pozval, da si ustvarimo uporabniški račun, oziroma se prijavimo z računom Apple ID. Na tej točki je bila namestitev sistema samega zaključena, vendar smo za učinkovito delovanje sistema Hackintosh namestili še zaganjalnik (angl. bootloader) in različne gonilnike.

3.3 PROBLEMI PRI NAMESTITVI

Že pred začetkom namestitve macOS-a smo naleteli na probleme. Disk je bil namreč formatiran v načinu MBR in ne GPT, macOS pa namestitve na diske s sistemom MBR ne podpira. Ob ponovnem zagonu se nam je izpisalo sporočilo *Missing Operating System*, zaradi česar nismo mogli pretvoriti diska z orodjem *diskpart*, saj nismo imeli dostopa do operacijskega sistema. Nadaljevali smo tako, da smo celoten disk pobrisali in začeli znova. Najprej smo namestili macOS, nato pa Windows. Dvojnega zagona nam v tem primeru ni uspelo vzpostaviti. Windows namreč v načinu BIOS ne podpira namestitve na diske GPT.



Slika 21: Napaka pri poskusu namestitve na disk GPT

Imeli smo probleme tudi z grafičnim pospeševanjem. Naša različica sistema macOS (10.14 Mojave) ne vključuje več podpore za starejše integrirane grafične kartice Intel HD, med katerimi je Intel HD 2000 v šolskih računalnikih. Najprej smo poskusili namestitev ponoviti z različico macOS 10.13 High Sierra, ki sicer nove različice programov in aplikacij ne podpirajo.

Poskusili smo namestiti naslednje različice sistema macOS:

- macOS 10.12 Sierra
- macOS 10.13 High Sierra

- macOS 10.14 Mojave

Izmed teh različic starejši dve (Sierra in High Sierra) nista niti dokončali zagona, zaradi delno delujoče podpore za integrirano grafično kartico Intel HD 2000. Mojave te sploh ne podpira in tako deluje le s privzetimi gonilniki VESA. Na koncu smo uporabili Mojave brez strojnega pospeševanja, kar sicer ni najbolj optimalno, ampak vseeno dovolj stabilno za potrebe testiranja in potencialno kompilacijo programske opreme.

3.4 IZBIRA KOMPATIBILNE STROJNE OPREME

Ker je različic macOS-a veliko in se kompatibilnost strojne opreme med njimi zelo razlikuje, smo se v tem delu osredotočili predvsem na različico 10.14 Mojave, saj smo to uspeli namestiti. [27]

Procesorji

MacOS podpira dokaj širok nabor procesorjev, med katerimi so Intelovi procesorji 4. do 9. generacije in procesorji AMD Ryzen. V sistemih Hackintosh se uporabljajo pretežno Intelovi procesorji zaradi boljšega posnemanja prave Applove strojne opreme (skoraj vsi Applovi računalniki imajo namreč vgrajene Intelove procesorje).

Nekatere starejše Ryzen serije so znano nekompatibilne (npr. Ryzen 1000, 2000 in 3000), prav tako je nekompatibilnih veliko Intel procesorjev, ki ne izhajajo iz družine Core. Tu gre predvsem za procesorje iz družin Intel Atom, Intel Celeron in Intel Pentium.

Matične plošče

Večina današnjih matičnih plošč v sistemih Hackintosh nima večjih težav pri uporabi, popolnoma drugače je seveda pri matičnih ploščah za prenosne računalnike.

Pozorni moramo biti predvsem na:

- zvočno kartico
- mrežno kartico
- integrirano grafično kartico (če je ta prisotna)

Večina sodobnih matičnih plošč proizvajalcev ASUS, Gigabyte in ASRock združljivostnih težav nima.

Pomnilnik

Kar se tiče delovnega pomnilnika oz. RAM-a, načeloma ni večjih problemov. Praktično ves RAM, ki deluje na sistemih Windows in Linux, deluje tudi na sistemu macOS.

Shramba

Pri pomnilniku ni kaj dosti za dodati, razen tega, da nekateri novejše SSD kartice tipa NVMe niso podprte. Je pa pomembno, da pred namestitvijo v BIOSu/UEFI-ju način delovanja SATA diskov nastavimo na AHCI, drugače nam macOS diska sploh ne bo prepoznal.

Kar se tiče SSD-jev je potrebno tudi preveriti, če je podprta funkcija TRIM (večinoma je, vendar obstajajo izjeme). Ta funkcija namreč skrbi za enakomerno izrabljanje pomnilniških celic v SSD-pogonu in tako drži daljšo življenjsko dobo.

Omrežje

Omrežja smo se dotaknili že pri razdelku Matična plošča, kjer je bilo omenjeno, da moramo biti pozorni na mrežno kartico na naši matični plošči. Z omrežjem je lahko kar nekaj težav, saj nam v primeru, da naša mrežna kartica ni podprta, večinoma ne preostane nič drugega, kot da jo zamenjamo.

Kar se tiče Ethernet mrežnih kartic jih dosti podpira že sistem sam, za naslednje družine pa obstajajo gonilniki v obliki modulov (KEXT):

- Realtek 2.5Gb Ethernet
- Realtek Gigabit Ethernet
- večina Atheros mrežnih kartic
- večina Intel mrežnih kartic
- I211-AT (dokaj pogoste na AMDmatičnih ploščah)

Izogibati pa se moramo mrežnih kartic za strežniške sisteme (Intel, HP, Dell, Mellanox ...), saj te načeloma niso podprte in zanje gonilnikov ni na voljo.

Glede brezžičnih (WiFi) mrežnih kartic pa je izbira bolj skopa, saj so v sistemu samem trenutno podprte samo Broadcomove BCM in Atherosove AR mrežne kartice. Neuradno je na voljo podpora tudi za Intelove AC-mrežne kartice, vendar je tukaj nekaj težav. Za izbiro med omrežji moramo namreč uporabiti poseben program, imenovan HeliPort, ki pa je na čase zelo nezanesljiv, prav tako ne podpira povezave na omrežja tipa WPA2 Enterprise (primer takšnega je omrežje Radius na naši šoli). HeliPort sam po sebi je narejen tako, da zamenja ikono WiFi v menijski vrstici sistema macOS, v meniju pa lahko izbiramo med omrežji:

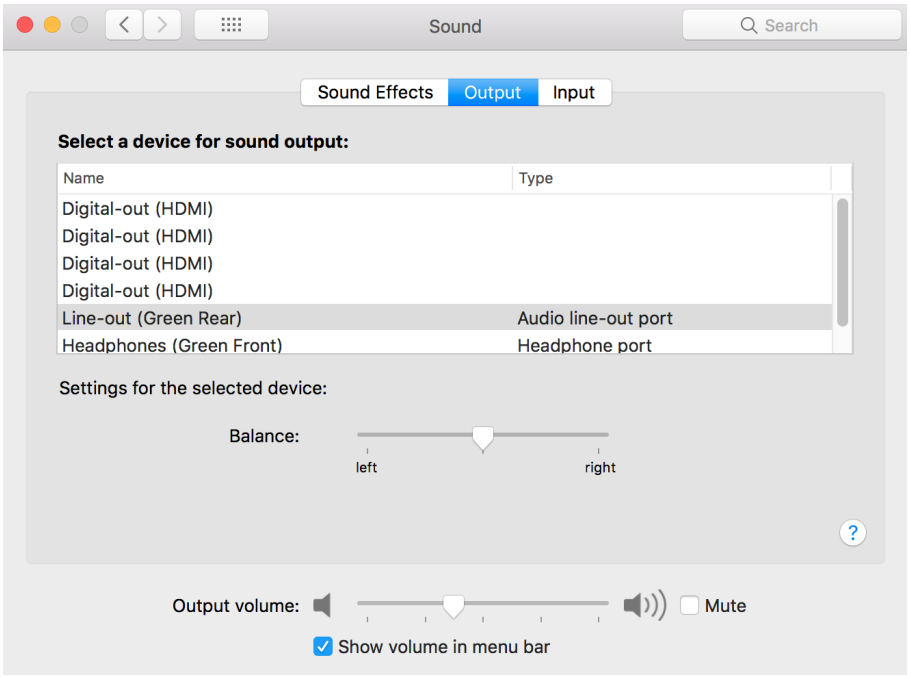
Slika 22:Pripomoček HeliPort

Mrežne kartice, ki so podprte s strani sistema samega, imajo samodejno tudi podporo tehnologije Bluetooth, za ostale mrežne kartice moramo to opraviti sami.

Zvok

V sistem macOS so vgrajeni tudi gonilniki AppleALC, ki podpirajo dokaj širok nabor zvočnih kartic. Za večino nepodprtih kartic je možno te gonilnike zakrpati, vendar lahko v tem primeru nastanejo težave. Primer takih težav je 3,5-mm zvočni vhod/izhod na prenosniku ThinkPad E590. Ta namreč deluje kot zvočni izhod, ob priključitvi slušalk z vgrajenim mikrofonom pa mikrofon ne deluje (čeprav ga sistem zazna).

Za nekatere vseeno nepodprte zvočne kartice obstajajo alternativni gonilniki VoodooHDA, ki pa sicer niso priporočeni. Težave lahko nastanejo tudi pri prenosi zvoka preko kablov HDMI in DisplayPort, saj vse grafične kartice nimajo enake podpore.

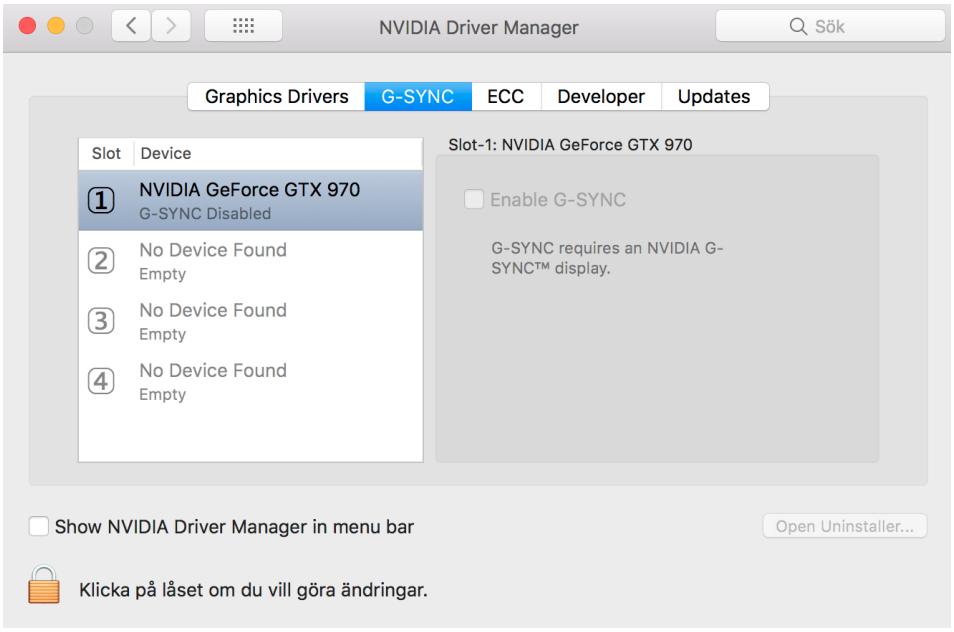


Slika 23: Izbira zvočnega izhoda

Grafične kartice:

Izbira grafične kartice je v sistemu Hackintosh ena izmed najpomembnejših, saj ob napačni izbiri sistem ne bo deloval popolnoma ali pa ne bo imel grafičnega pospeševanja (kar je za sistem macOS izredno pomembno, saj se nanj zanaša večji del grafičnega vmesnika).

Med popolnoma kompatibilne grafične kartice spadajo predvsem Intelove integrirane grafične kartice (večina jih je podprta že samodejno oz. z minimalnimi popravki) ter AMD-grafične kartice iz družin RX, Radeon Pro, Navi in Vega. Dolgo časa je veljalo, da so grafične kartice Nvidia dobra izbira, vendar ne več, saj je Nvidia pred kratkim nehala razvijati gonilnike za sistem macOS. Zadnja različica macOS-a, ki podpira grafične kartice Nvidia, je tako 10.13 High Sierra.



Slika 24: Nvidiini gonilniki za macOS

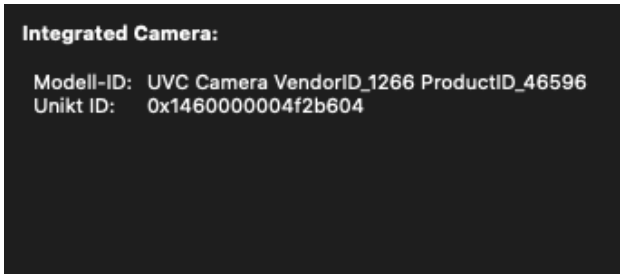
Pri uporabi proizvajalčevih gonilnikov imamo na voljo tudi več nadstandardnih funkcij grafične kartice, kot npr. G-SYNC pri grafičnih karticah Nvidia.

Moramo pa upoštevati, da nekatere starejše grafične kartice v novejših različicah niso podprte. Primer tega je naša integrirana grafična kartica Intel HD 2000, ki sploh ni podprta. Prav tako se je priporočljivo izogibati grafičnim karticam brezimenskih podjetij (HIS, VisionTek ...), saj zaradi njihove redkosti in pomanjkljivosti dokumentacije ni gonilnikov.

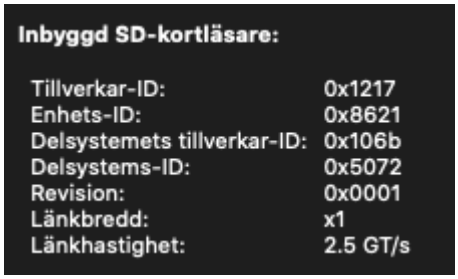
MacOS načeloma tudi ne podpira priklopa monitorja prek priključka VGA, razen v primerih res starih grafičnih kartic.

Ostala strojna oprema

Kar se tiče ostale strojne opreme, se lahko zanašamo predvsem na krpanje tabel ACPI in iskanje gonilnikov (modulov KEXT). Primer take strojne opreme sta vgrajena spletna kamera in čitalec kartic SD v prenosnih računalnikih. Naslednja dva posnetka zaslona sta s prenosnega računalnika Lenovo ThinkPad E590, kjer obe napravi delujeta po krpanju tabel ACPI:



Slika 25: Podatki o integrirani kameri



Slika 26: Podatki o čitalcu kartic SD

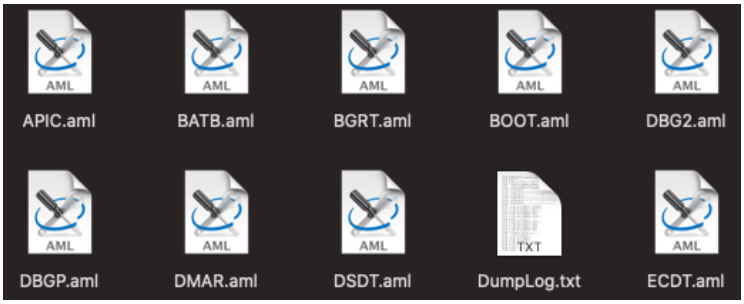
3.4.1 POPRAVLJANJE TABEL ACPIZA VEČJO KOMPATIBILNOST

Problemi se pogosto pojavljajo pri upravljanju z energijo in ACPI-jem. Ta namreč skrbi za zaznavo strojne opreme, učinkovito porabo z električno energijo, upravljanje s funkcijo spanja, hitrostjo vrtenja ventilatorjev itd. To je pomembno predvsem pri prenosnih računalnikih, saj brez podpore ACPI baterija ne bo zaznana, spanec ob zaprtju pokrova ne bo deloval, ob zbujanju računalnika iz spanja zaslon ne bo deloval ... Vse te stvari naredijo uporabo prenosnika zelo nerodno. V nekaterih (sicer redkih) primerih bo macOS našo konfiguracijo ACPI zaznal samodejno, drugače pa ga lahko do tega prisilimo sami s krpanjem tabel ACPI. [28]

Vsi naslednji primeri so za prenosni računalnik Lenovo ThinkPad E590, na katerem je bilo krpanje potrebno za zaznavo večine naprav v sistemu in učinkovito delovanje.

Samo krpanje tabel lahko vzame precej časa, saj moramo vse skupaj velikokrat preizkusiti, poznati moramo veliko podatkov o sistemu, seznaniti se moramo tudi s samim programskim jezikom, ki se za to uporablja. Grob opis postopka je sledeč.

1. Ob zagonu sistema v bootloaderju Clover ali OpenCore pridobimo originalne datoteke ACPI. To storimo z izbiro možnosti v meniju oziroma s pritiskom ustrezne tipke (npr. F4 v Cloverju). Tabele se nam nato shranijo v mapo na particiji EFI , kjer lahko kasneje do njih dostopamo. Najpomembnejša izmed njih je datoteka DSDT.aml, ostalih načeloma ne potrebujemo, lahko pa jih uporabimo za odpravo napak na dodatni strojni opremi (npr. sledilna ploščica na prenosnem računalniku).



Slika 27: Originalne datoteke ACPI

2. Z uporabo proizvajalčevega programa (npr. pri Intelu je to iASL, ki je na voljo iz njihove spletne strani) jih prevedemo iz strojne oblike v izvorno kodo. Ta je napisana v programskem jeziku ASL, ki je podoben jeziku C, vendar med seboj nista

kompatibilna. [29] Pri tem si lahko pomagamo s programom MaciASL, ki deluje kot:

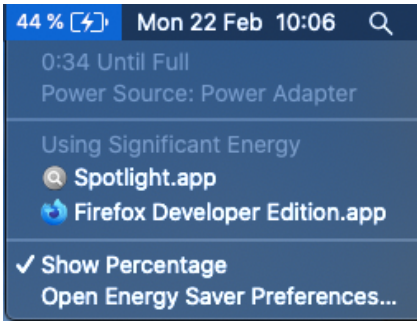
```
Device (BAT0)
{
    Name (_HID, EisaId ("PNP0C0A"))  // _HID: Hardware ID
    Name (_UID, Zero)  // _UID: Unique ID
    Name (_PCL, Package (0x01)  // _PCL: Power Consumer List
    {
        _SB
    })
    Name (B0ST, Zero)
    Name (BT0I, Package (0x0D)
    {
        Zero,
        0xFFFFFFFF,
        0xFFFFFFFF,
        One,
        0x2A30,
        Zero,
        Zero,
        One,
        One,
        "",
        "",
        "",
        ""
    })
```

```
Device (BAT0)
{
    Name (_HID, EisaId ("PNP0C0A"))  // _HID: Hardware ID
    Name (_UID, Zero)  // _UID: Unique ID
    Name (_PCL, Package (0x01)  // _PCL: Power Consumer List
    {
        _SB
    })
    Name (B0ST, Zero)
    Name (BT0I, Package (0x0D)
    {
        Zero,
        0xFFFFFFFF,
        0xFFFFFFFF,
        One,
        0x2A30,
        Zero,
        Zero,
        One,
        One,
        "",
        "",
        "",
        ""
    })
```

Slika 28:Jjezik ASL

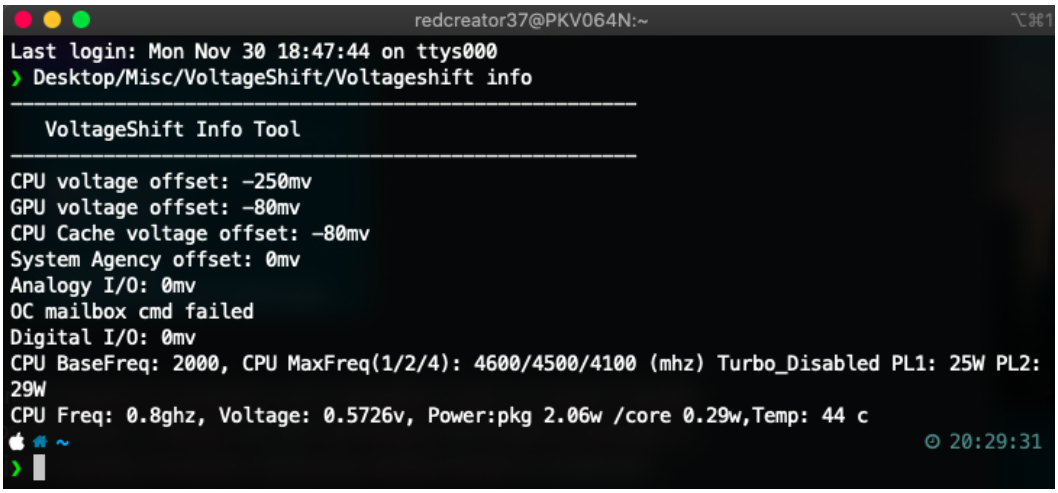
3. Datoteke zakrpamo ročno ali s pomočjo popravkov iz interneta. Če imamo srečo, ti že obstajajo, drugače se moramo lotiti ročnega krpanja. Tukaj nam lahko pomaga tudi, če na internetu najdemo datoteke DSDT za modele matičnih plošč/ prenosnih računalnikov, podobnih našemu, saj tako vemo, kateri deli (sicer zelo obsežnih) datotek so za nas pomembni.
4. Datoteke prevedemo nazaj v strojno obliko in jih vstavimo v zaganjalnik. Ta jih ob ponovnem zagonu potem naloži.

Če smo vse storili pravilno, bomo ob ponovnem zagonu med drugim imeli na voljo status baterije v menijski vrstici:



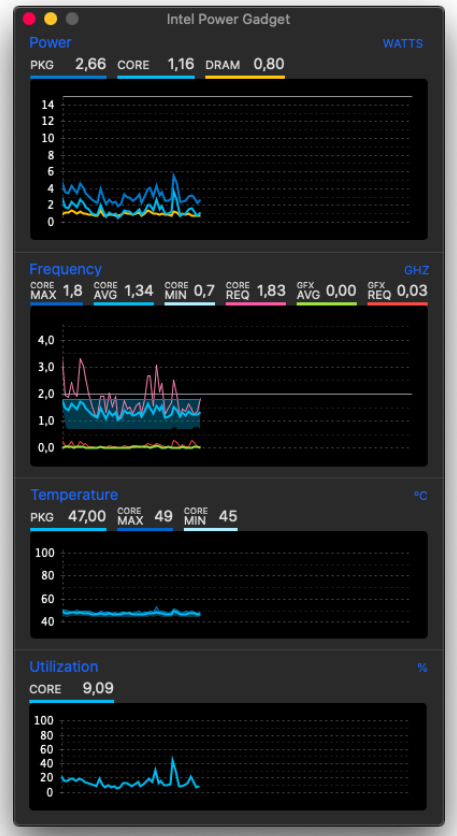
Slika 29: Stanje porabe baterije

Ena izmed funkcij je tudi upravljanje s frekvenco procesorja, saj ta posledično vpliva na porabo energije na prenosnih računalnikih in hitrost delovanja. V primeru da nam tudi krpanje tabel ACPI ne pomaga, si lahko pomagamo z "undervoltanjem" samega procesorja, njegovega predpomnilnika in integrirane grafične kartice. [30]



Slika 30: Orodje VoltageShift

Porabo energije, napetost in frekvenco procesorja lahko v sistemih s procesorji Intel preverimo z uporabo pripomočka Intel Power Gadget.



Slika 31: Orodje Intel Power Gadget

Vodič po programskem jeziku ASL je na voljo na tem naslovu:

acpica.org/sites/acpica/files/asl_tutorial_v20190625.pdf

3.4.2 PRIMER KOMPATIBILNEGA RAČUNALNIKA

Preverili smo, katere dele bi potrebovali, če bi šola želela kupiti kompatibilne računalnike za uporabo sistemov Hackintosh. Ugotovili smo, da za računalnike Hackintosh ne potrebujemo nujno zelo drage strojne opreme (na kar smo na začetku pomislili, saj so Applovi računalniki znani po visokih cenah). Najbolj pomembno je, da so kompatibilne komponente (kolikor toliko). Veliko denarja lahko prihranimo prav pri grafični kartici, saj ima macOS zelo dobro podporo prav za Intelove grafične kartice, ki so že del samega procesorja. Prav tako lahko prihranimo na komponentah, kjer kompatibilnostnih težav skoraj ni (napajalnik, pomnilnik RAM, disk / SSD ...).

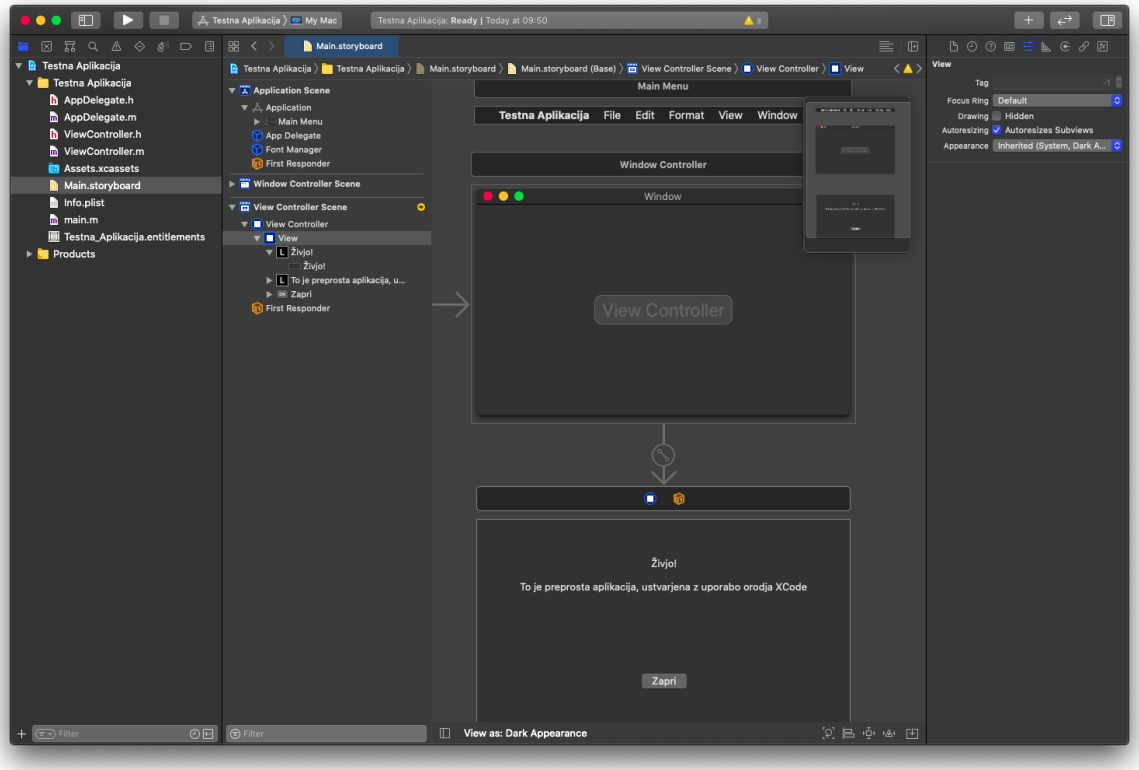
Primer poceni macOS-kompatibilnega računalnika:

- CPU: Intel Core i3 8100 Quad Core – 90 €
- matična plošča: GIGABYTE Z370 HD3 ATX – 86,70 €
- pomnilnik RAM: 2x G SKILL Ripjaws V Series 4GB DDR4 – skupaj 60,25 €
- SSD: PNY CS1311 128GB – 41,20 €
- napajalnik: EVGA 500 BQ 80+ 500W Bronze – 41,20 €

Skupna cena takšnega sistema je tako okoli 320 €, kar je za računalnik Hackintosh zelo poceni in primerljivo s cenami ostalih šolskih računalnikov.

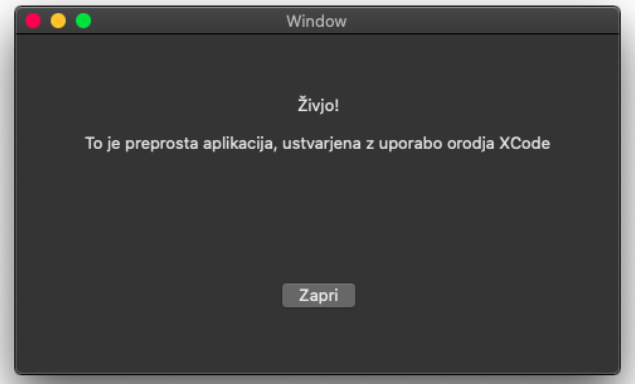
3.5 VZPOSTAVITEV RAZVOJNEGA OKOLJA

Že od samega začetka raziskovalne naloge smo si prizadevali vzpostaviti delujoč razvojni sistem za Applove platforme. Pri tem smo se osredotočili na Applovo razvojno orodje XCode, saj to (kot že omenjeno) predstavlja glavno blokado razvoja programske opreme za Applove platforme na vseh drugih. Tega smo tudi preizkusili:



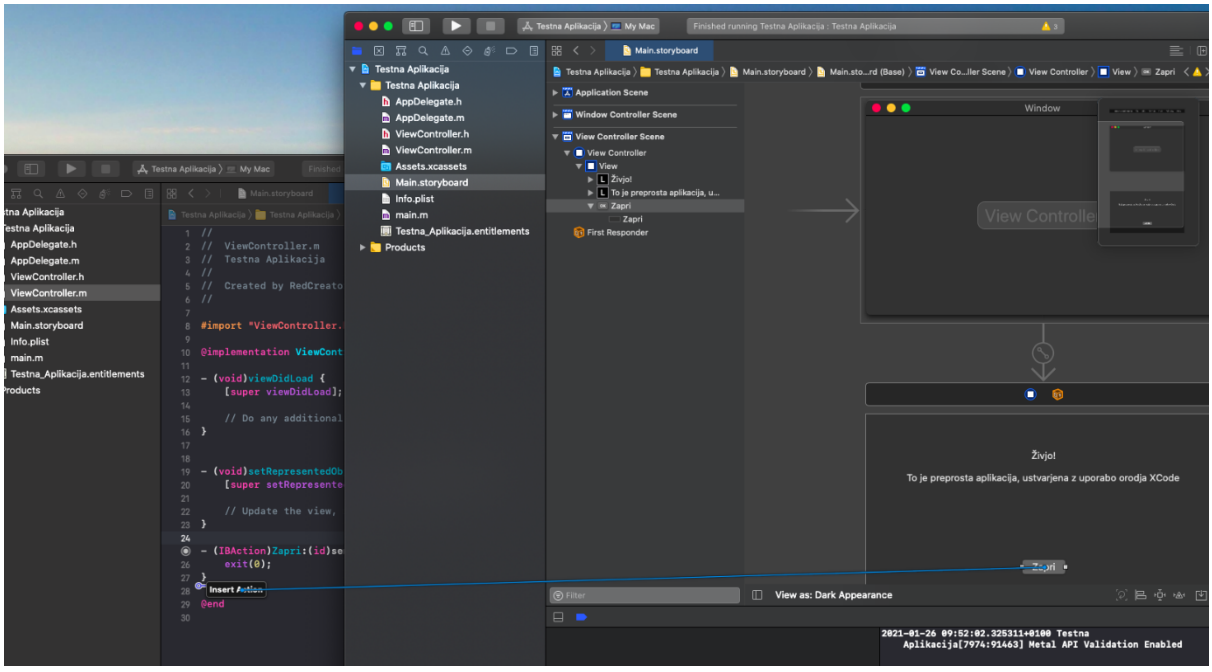
Slika 32: Načrtovanje uporabniškega vmesnika

Xcode je preprosto razvojno orodje, ki pa ima zelo napredne funkcije za razvoj uporabniških vmesnikov aplikacij. Kontrolnike preprosto dodamo v okno in nastavimo njihove lastnosti. Tako lahko z zgornjim primerom naredimo aplikacijo, ki izgleda približno takole:



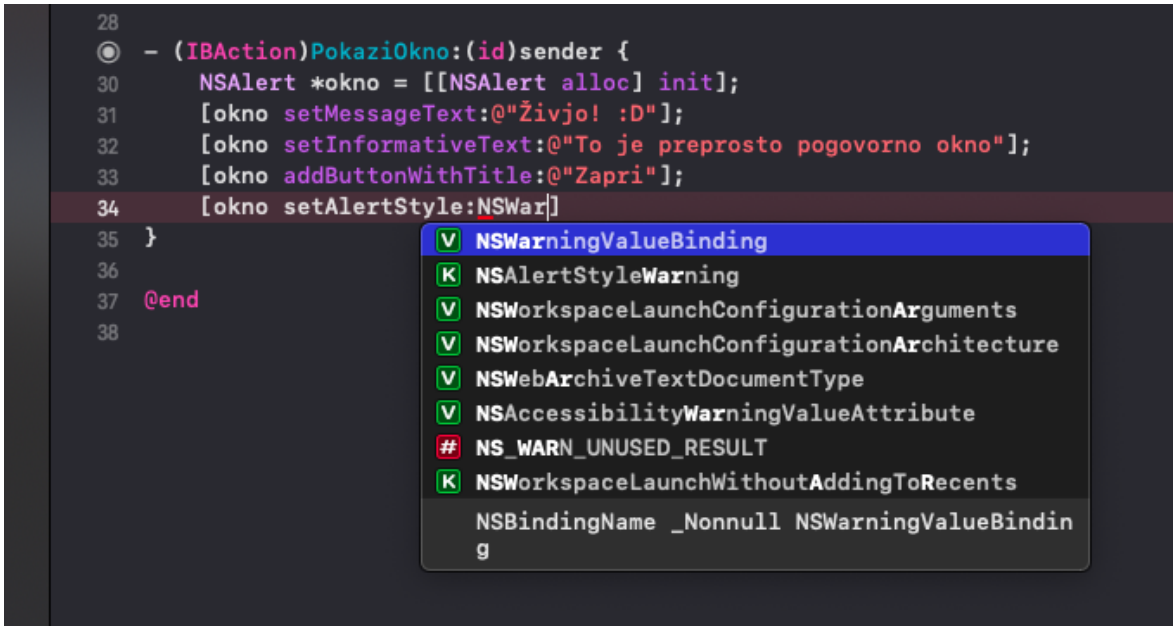
Slika 33: Okno aplikacije

Da dodamo kontrolam akcije, jih samo povlečemo z desnim klikom v datoteko Objective-C. Pri tem navedemo ime funkcije, da nam XCode lahko vstavi ogrodno (angl. boilerplate) kodo. Princip je sicer zanimiv, vendar lahko hitro postane nepraktičen, če imamo vmesnike z več kontrolami (predvsem zaradi tega, ker ni čistega pregleda nad vsemi že dodanimi akcijami za vsako kontrolo). Začetnikom proces sicer v večini primerov proces ni najbolj jasen, izgleda pa približno takole:



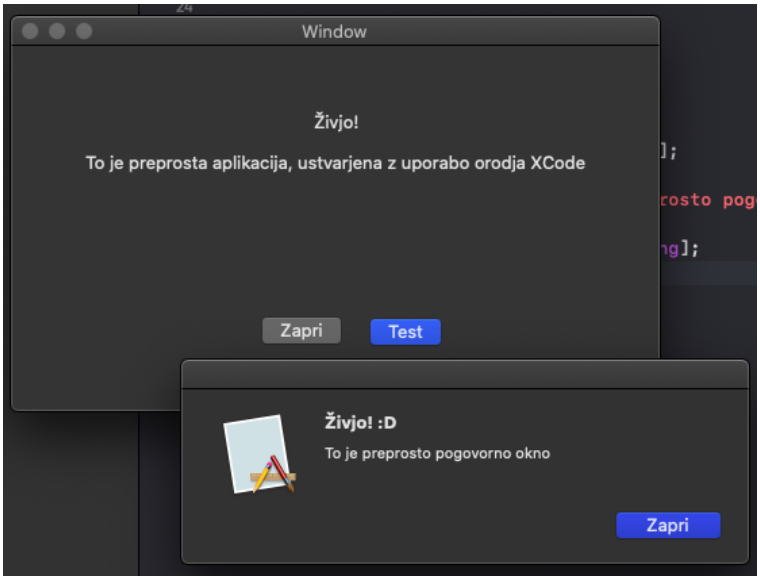
Slika 34: Povezovanje uporabniškega vmesnika s kodo

Nato sledi samo še to, da dodamo kodo za akcije. Pri tem lahko uporabimo programska jezika Swift oziroma (v našem primeru) Objective-C.



Slika 35: Pisanje kode v programu XCode

Po zagonu končna aplikacija izgleda takole:



Slika 36: Delujoča aplikacija

Ta postopek deluje na naših sistemih Hackintosh identično kot na originalnih Mac-ih. Prav tako je mogoče razvijati tudi aplikacije za ostale Applove platforme (iOS, iPadOS, watchOS, tvOS) brez kakršnihkoli problemov. Povrh vsega brez problemov deluje tudi simulator iOS, ki se uporablja za testiranje aplikacij iOS brez uporabe iPhonov.

Tako nam je uspelo vzpostaviti razvojni sistem, vendar uporaba macOS-a v tem stanju ni optimalna. Tak sistem je kljub vsemu še vedno zelo uporaben, saj ga lahko uporabljamo za prevajanje kode na koncu razvoja (samo pisanje pa izvajamo na sistemu Windows).

3.6 LEGALNOST HACKINTOSHA

Tehnično gledano kršimo pogoje uporabe programske opreme macOS, če jo naložimo na strojno opremo, ki ni Applova. Apple se nikoli ni potrudil, da bi skupnost Hackintosh zasledoval s pravnega vidika; raje se zanašajo na tehnične ovire, kot so šifrirana strojna in programska oprema ter lastna proizvodnja. S tem poskušajo nameščanje sistemov Hackintosh naredili dovolj težavno, da se s tem ne bi ukvarjali sleherniki.

Pri uporabi sistemov Hackintosh obstajajo tveganja, ki lahko povzročijo nestabilnost sistema. Posodobitve operacijskega sistema lahko zlahka povzročijo, da ta preneha delovati pravilno. Nekateri programi prav tako ne bodo delovali pravilno. Povezave z nekaterimi Applovimi storitvami, kot so iMessage, FaceTime itd., v nekaterih primerih iz pravnih in drugih razlogov ne delujejo. Obstajajo sicer rešitve in popravki, vendar ti s skoraj vsako posodobitvijo prenehajo delovati.

Apple vas s pravnega vidika lahko toži, če prodajate prednameščene sisteme Hackintosh, vendar tega najverjetneje ne bodo storili [31]. Takšni sistemi namreč ne vplivajo na njihovo trženje in dejansko pritegnejo nove stranke, saj se ljudje lahko navadijo uporabljati Applovo programsko opremo. V preteklosti so sicer že tožili podjetje, ki se je ukvarjalo s prodajo sistemov Hackintosh in tožbo tudi dobili [32]. Po navedbah Applove licence za končne uporabnike (EULA) je namestitev macOS-a v navadne računalnike izrecno prepovedana [33]. Tukaj se lahko sklicujemo na izjemo v ameriškem zakonu DMCA, da gre za »pravično uporabo« (angl. »fair use«), pri čemer se takšne sisteme uporablja samo v izobraževalne namene, prav tako je uporabljena legalno pridobljena programska oprema (namestitveni medij je namreč potrebno ustvariti s pravim Mac-om, s kopijo macOS-a, prenešeno iz Applove trgovine App Store). Pri tem smo seveda omejeni, da takšnih sistemov ne prodajamo naprej, oziroma jih ne uporabljamo v kakršnekoli druge poslovne namene. Želja veliko uporabnikov sistemov Hackintosh je sicer ta, da bi Apple svojo programsko opremo začel prodajati za uporabo na strojni opremi, ki ni njegova, vendar se to najverjetneje ne bo zgodilo.

Kot prej omenjeno pa lahko Applovo EULO v redkih primerih prezrete, če se odločite samo za eksperimentiranje in učenje razvoja na lastni strojni opremi za osebno rabo. Kljub nekaj potencialnim nevšečnostim ima Hackintosh veliko pozitivnih lastnosti, kot so

široka izbira komponent, nadgradljivost, povezljivost, boljše hlajenje in najpomembnejša – cena. Iz strojnega vidika gre lahko za enake komponente, le da imamo v primeru Hackintosha več fleksibilnosti pri njihovi izbiri. MacOS pa je avtorsko zaščitena programska oprema, za katere kopiranje in nameščanje potrebujete dovoljenje imetnika avtorskih pravic. Apple na splošno izda licenco za namestitev programske opreme samo za lastno strojno opremo v skladu z licenčno pogodbo za maloprodajnega končnega uporabnika. Če se ne strinjate s pogoji EULA, programske opreme ne smete uporabljati.

Če namestite programsko opremo brez dovoljenja licence imetnika avtorskih pravic, kršite njegove avtorske pravice. Kršitev avtorskih pravic se v večini primerov obravnava bolj kot prekršek, ki pa sicer vodi do pravne odgovornosti. Običajno mora imetnik avtorskih pravic vložiti tožbo zaradi kršitve. V ZDA je predpisanih 150.000 ameriških dolarjev odškodnine zaradi namerne kršitve dejanske škode, ki jo je ocenilo sodišče (v tem primeru verjetno maloprodajna cena macOS, približno 50 ameriških dolarjev). Apple je do danes vložil samo tožbe proti ljudem, ki se ukvarjajo z distribucijo in prodajo računalnikov Hackintosh, ne pa proti posameznikom, zato je dejansko tveganje zelo majhno. Ironično je, da bi lahko za teh 150.000 dolarjev kupili kar 250 računalnikov Mac, vsakega s primerno licencirano kopijo macOS-a.

4. REZULTATI IN RAZPRAVA

Prvo hipotezo o uporabi macOS-a na šolskih računalnikih za razvoj aplikacij smo delno potrdili, saj nam takšnega sistema ni uspelo popolnoma vzpostaviti. Namestili smo namreč sistem macOS ter vsa potrebna razvojna orodja, vendar nam ob tem ni uspelo vzpostaviti grafičnega pospeševanja, kar skrajno otežuje delo s sistemom. Prav tako nam ni uspelo vzpostaviti dvojnega zagona s sistemom Windows.

Ker je takšen sistem delno uporaben, ga lahko še vedno uporabimo za razvoj aplikacij. V tem primeru bi aplikacije kodirali na računalnikih s sistemom Windows, kodo pa bi nato prevedli na sistemu macOS. Aplikacije bi lahko na tem sistemu tudi testirali, razen v primeru, da te zahtevajo uporabo grafičnega pospeševanja za osnovno delovanje.

Drugo hipotezo, da bi lahko ob prenovi računalniških učilnic s pazljivo izbiro ustreznih komponent vzpostavili popolnoma delujoč sistem z dvojnim zagonom, smo v celoti potrdili. Vzpostavitev takšnega sistema bi bila ob uporabi popolnoma združljivih komponent v celoti mogoča. Izgradnja takšnih sistemov bi bila za šolo v primerjavi z nabavo pravih Applovih računalnikov cenovno zelo ugodna, saj smo dokazali, da takšni sistemi stanejo tudi manj kot 500 €.

Takšna rešitev pa bo v praksi uporabna samo za nekaj let, saj se pri Applu pripravljajo na popoln preklop na lastno arhitekturo procesorjev, ki temelji na tehnologiji ARM (ti. Apple Silicon). Hackintosh tako v nekaj (po naši oceni okoli pet) letih ne bo več popolnoma mogoč. Namreč, čeprav lahko kupimo procesorje, podobne Applovim, imajo slednji vgrajeno veliko število lastno razvitih tehnologij, ki so zelo skopo dokumentirane. Tudi v primeru, da nam s pomočjo ti. obratnega inženiringa uspe emulirati večino lastniške tehnologije v Applovih ARM čipih, bo učinkovitost takšnih sistemov pod ravnjo uporabnega.

Pri tem je vredno poudariti, da bodo obstoječi sistemi še vedno delovali, vendar bo podpora novim aplikacijam (in s tem zahtevanim razvojnim orodjem za razvoj za novejše Applove sisteme) zelo hitro začela izginjati. Applove naprave so na splošno znane za redno posodabljanje, programska oprema pa hitro izgubi podporo. To se zgodi povprečno v okoli

petih letih, kar je v primerjavi s sistemom Windows, pri katerih so nekatere različice do zdaj imele podporo tudi za do trinajst let, zelo kratek čas. Trenutni šolski računalniki so prav tako starejši od petih let, zaradi česar je smiselnost vzpostavljanja takšnih sistemov ponovno vprašljiva.

Tretje hipoteze, ki se nanaša na legalnost takšnih sistemov, ne moremo niti potrditi niti ovreči, saj gre za legalno »sivo cono«. Pri tem se zanašamo na to, da se naš namen Applu ne zdi sporen. V vsakem primeru pa ima Apple zadostno legalno podlago za vložitev tožbe, kot so to tudi pokazali.

Glede na to, da bi mi takšne sisteme uporabljali samo v izobraževalne namene in še to v trenutnem stanju zgolj v zelo omejeni obliki, je to pogojno opravičljivo; vendar nikoli ne moramo biti povsem prepričani, saj smo tukaj popolnoma prepuščeni Applovi volji.

5. ZAKLJUČEK

V okviru te raziskovalne naloge smo si poskušali pridobiti celotno razvojno platformo za Applove naprave, ki bi bila tako na voljo dijakom za učenje razvoja programske opreme. Pri tem so nas ovirali številni programski in strojni problemi. Večina izmed njih je nenamernih, saj Apple ne podpira številne strojne opreme, ki je prisotna v ostalih računalnikih. So pa med njimi tudi varovalke, ki onemogočajo preveč preprosto namestitev sistema macOS. Za večino programskih problemov smo tako našli preproste in popolnoma delujoče rešitve, medtem ko smo bili pri strojni opremi bolj omejeni.

Vsekakor pa se nam je raziskovalna naloga zdela zelo zanimiva, saj si nismo predstavljali, da je mogoče doseči tako dobro stopnjo uporabnosti sistema macOS brez uporabe kakršenkoli Applove strojne opreme. Polega tega smo imeli tudi priložnost, da na hitro preizkusimo razvoj programske opreme za Applove platforme.

6. ZAHVALE

Za pomoč pri izdelavi raziskovalne naloge se zahvaljujemo:

- Šolskemu centru Velenje, Elektro in računalniški šoli za uporabo računalnikov,
- Samu Železniku in Urošu Remenihu za mentorstvo in pomoč pri izvedbi raziskovalne naloge,
- Sonji Lubej za lektoriranje besedila.

7. VIRI IN LITERATURA

[1] https://www.apple.com/macos/

[2] https://www.nextcomputers.org/NeXTfiles/Docs/NeXTStep/3.3/nd/Concepts/Pre3.0_Concepts/01_SysOver.htmld/index.html

[3] https://www.apple.com/newsroom/2008/06/09Apple-Previews-Mac-OS-X-Snow-Leopard-to-Developers/

[4] https://www.apple.com/macosx/features/300.html

[5] https://www.apple.com/newsroom/2020/06/apple-introduces-macos-big-sur-with-a-beautiful-new-design/

[6] https://www.cnet.com/news/apples-jobs-pa-semi-to-design-iphone-chips/ https://arstechnica.com/gadgets/2020/06/this-is-apples-roadmap-for-moving-the-first-macs-away-from-intel/

[7] https://en.wikipedia.org/wiki/Hackintosh/

[8] http://benchmarkreviews.com/index.php?option=com_content&task=view&id=623&Itemid=38

[9] https://lifehacker.com/5583650/run-mac-os-x-in-virtualbox-on-windows

[10] https://en.wikipedia.org/wiki/Comparison_of_platform_virtualization_software

[11] http://www.redmondpie.com/how-to-install-mac-os-x-snow-leopard-in-virtualbox-on-windows-7/

[12] https://www.techjunkie.com/mac-windows-10-vmware-unlocker/

[13] https://kb.vmware.com/s/article/1000131

[14] http://www.macbreaker.com/2012/01/what-are-kexts.html

[15] https://en.wikipedia.org/wiki/Hackintosh#Clover

[16] https://www.tonymacx86.com/forums/unibeast.120/

[17] https://www.tonymacx86.com/threads/guide-patching-laptop-dsdt-ssdts.152573/

[18] https://www.easy2boot.com/configuring-e2b/configure-clover/

[19] https://en.wikipedia.org/wiki/Xcode

[20] https://en.wikibooks.org/wiki/Computer_Programming/MacOS_Programming

[21] https://www.vmware.com/products/fusion.html

[22] https://en.wikipedia.org/wiki/Hackintosh#Boot_loaders_and_emulators

[23] https://support.apple.com/apple-id

[24] https://www.tonymacx86.com/forums/unibeast.120/

[25] https://en.wikipedia.org/wiki/Disk_partitioning

[26] https://www.easy2boot.com/configuring-e2b/configure-clover/

[27] https://www.apple.com/macos/how-to-upgrade/

[28] https://wiki.osx86project.org/wiki/index.php/DSDT

[29] https://asl.readthedocs.io/en/latest/

[30] https://hackintoshpro.com/undervolt-mac

[31] http://www.computerworld.com/s/article/9121798/Apple_adds_DMCA_charge_to_lawsuit_against_Psystar

[32] http://www.pcworld.com/article/182218/Apple_Wins_Court_Victory_Over_Mac_Clone_Maker_Psystar.html

[33] https://images.apple.com/legal/sla/docs/macosx106.pdf

PRILOGA A: SLOVAR KRATIC

BIOS – Basic Input Output System

EFI – Extensible Firmware Interface

UEFI – Unified Extensible Firmware Interface

FAT – File Allocation Table (tudi **FAT32** in **exFAT**)

APFS – Apple File System

JHFS+ – Mac OS X Extended Journaled

MBR – Master Boot Record

GUID – Globally Unique Identifier

GPT – GUID Partition Table

KEXT – Kernel EXTension

EULA – End User License Agreement

DMCA – Digital Millenium Copyright Act

AMD – Advanced Micro Devices, Inc.

BSD – Berkeley Software Destribution

IDE – Integrated Development Environment

XML – eXtensible Markup Language

SMC – System Management Controller