VIŠJA STROKOVNA ŠOLA ACADEMIA

MARIBOR

AVTOMATIZACIJA PRIPRAVE PROMOCIJSKIH OZNAČEVALCEV Z ADOBE IN DESIGN EXTENDSCRIPT

Kandidat: Matic Trogrlič

Vrsta študija: študent izrednega študija

Študijski program: Informatika

Mentor predavatelj: mag. Matjaž Prtenjak

Mentor v podjetju: Aleš Pongračič, dipl. inž. itk.

Lektorica: Sabina Pačnik, prof. slov.

Maribor, 2026

IZJAVA O AVTORSTVU DIPLOMSKEGA DELA

Podpisani Matic Trogrlič sem avtor/ica diplomskega dela z naslovom Avtomatizacija priprave promocijskih označevalcev z Adobe InDesign ExtendScript, ki sem ga napisal pod mentorstvom mag. Matjaža Prtenjaka.

S svojim podpisom zagotavljam, da:

- je predloženo delo izključno rezultat mojega dela,
- sem poskrbel, da so dela in mnenja drugih avtorjev, ki jih uporabljam v predloženi nalogi, navedena oz. citirana skladno s pravili Višje strokovne šole Academia Maribor,
- se zavedam, da je plagiatorstvo – predstavljanje tujih del oz. misli, kot moje lastne kaznivo po Zakonu o avtorski in sorodnih pravicah (Uradni list RS, št. 16/07 – uradno prečiščeno besedilo, 68/08, 110/13, 56/15 in 63/16 – ZKUASP); prekršek pa podleže tudi ukrepom Višje strokovne šole Academia Maribor skladno z njenimi pravili,
- skladno z 32.a členom ZASP dovoljujem Višji strokovni šoli Academia Maribor objavo diplomskega dela na spletnem portalu šole.

Maribor, september 2026 Podpis študenta:

ZAHVALA

Iskreno se zahvaljujem mag. Matjažu Prtenjaku za strokovno pomoč, usmerjanje in koristne nasvete pri nastajanju diplomskega dela.

Zahvaljujem se mentorju v podjetju, dipl. inž. itk. Alešu Pongračiču, za tehnične usmeritve in praktične nasvete pri implementaciji sistema.

Hvala podjetju Jagros, d. o. o., da so mi omogočili pridobitev potrebnih podatkov in mi zaupali razvoj rešitve za avtomatizacijo priprave promocijskih označevalcev.

Vsem udeležencem testiranja uporabniške izkušnje se zahvaljujem za njihov čas in sodelovanje.

Posebna zahvala gre mojim bližnjim za podporo, potrpežljivost in spodbudo med študijem in pisanjem diplomskega dela.

POVZETEK

Diplomsko delo obravnava avtomatizacijo priprave promocijskih označevalcev v podjetju Jagros, d. o. o. (Trgovine Jager), kjer je bil navedeni proces pred razvojem sistema Oznacevalec v celoti ročen.

V okviru diplomskega dela sem razvil sistem Oznacevalec, ExtendScript aplikacijo za Adobe InDesign, ki ta proces avtomatizira. Sistem deluje prek tipkovničnih bližnjic, in to v dveh korakih: uporabnik najprej aktivira sistem in v letaku označi elemente artikla v določenem vrstnem redu (sliko, naziv, ceno in ikone), nato zajem potrdi in izbere format predloge, sistem pa samodejno generira končne datoteke. Sistem podpira dvanajst formatov predlog in posebno logiko za format Mali poličnik.

Pri razvoju sem se soočil s tehničnimi omejitvami okolja ExtendScript, ki podpira le starejši standard ES3, nima pa vgrajenega mehanizma za serializacijo podatkov JSON in trajnega spomina med zagoni skripte. Te omejitve sem rešil z lastno implementacijo serializacije JSON in s shranjevanjem stanja sistema v zunanjo datoteko state.json.

Sistem sem preizkusil na treh ravneh: s funkcionalnim testiranjem na 25 testnih primerih, z meritvijo časa priprave pri ročnem in avtomatiziranem postopku ter s testiranjem uporabniške izkušnje na osmih udeležencih, petih nestrokovnjakih brez predhodnega znanja InDesigna in treh izkušenih grafikih. Meritve so pokazale, da sistem v povprečju skrajša čas priprave za približno 16,5 odstotka. Vsi udeleženci testiranja, vključno z nestrokovnjaki, so nalogo uspešno zaključili izključno na podlagi pisnih uporabniških navodil, kar potrjuje, da je sistem razumljiv tudi uporabnikom brez tehničnega predznanja.

Diplomsko delo je pokazalo, da je mogoče s skriptiranjem v okolju Adobe InDesign ExtendScript, kljub njegovim tehničnim omejitvam, zgraditi zanesljiv sistem, ki skrajša čas priprave promocijskih označevalcev in je razumljiv uporabnikom brez predhodnega tehničnega znanja. Hkrati se je pokazalo, da zaradi velikih razlik med formati predlog enotne arhitekture brez podvajanja kode ni bilo mogoče v celoti doseči, kar ostaja priložnost za nadaljnji razvoj sistema.

Ključne besede: *Adobe InDesign, ExtendScript, avtomatizacija procesov, promocijski označevalci, skriptiranje*

ABSTRACT

»Automation of promotional price tag preparation using Adobe InDesign ExtendScript«

This thesis addresses the automation of promotional price tag preparation at Jagros d.o.o. (Trgovine Jager), where the process was, prior to the development of the Oznacevalec system, entirely manual.

As part of this thesis, I developed Oznacevalec, an ExtendScript application for Adobe InDesign that automates this process. The system operates in two steps triggered by keyboard shortcuts: the user first activates the system and selects the article's elements in the flyer in a defined order (image, name, price, icons), then confirms the selection, chooses a template format, and the system automatically generates the final files. The system supports 12 template formats as well as separate logic for Mali poličnik format.

During development, I faced technical limitations of the ExtendScript environment, which supports only the older ES3 standard, lacks a built-in JSON serialization mechanism, and does not retain memory between script runs. I resolved these limitations with a custom JSON serialization implementation and by storing the system's state in an external state.json file.

The system was tested on three levels: functional testing across 25 test cases, a time comparison between the manual and automated processes, and a usability test involving eight participants, five with no prior InDesign experience and three experienced graphic designers. The measurements showed that the system reduces preparation time by approximately 16.5% on average.

The thesis shows that, despite its technical limitations, the Adobe InDesign ExtendScript environment can be used to build a reliable system that shortens the preparation time of promotional price tags and is understandable to users without prior technical knowledge. At the same time, the large differences between template formats meant that a fully unified, duplication-free architecture could not be achieved, which remains an opportunity for future development of the system.

Keywords: *Adobe InDesign, ExtendScript, process automation, promotional price tags, scripting*

KAZALO VSEBINE

1 UVOD 9

1.1 Opis področja in opredelitev problema 9
1.2 Namen, cilji in osnovne trditve 9
1.3 Predpostavke in omejitve 10
1.4 Uporabljene raziskovalne metode 11

2 ANALIZA OBSTOJEČEGA PROCESA 12

3 NAČRTOVANJE SISTEMA 14

3.1 Arhitektura sistema 14
3.2 Podatkovni model 16
3.3 Dialogi in interakcija z uporabnikom 17

4 IMPLEMENTACIJA SISTEMA 21

4.1 ExtendScript in omejitve okolja 21
4.2 Shranjevanje stanja med sejami 23
4.3 Skaliranje elementov 25
4.4 Podpora različnim formatom 28
4.5 Validacija in preprečevanje napak 30

5 TESTIRANJE 32

5.1 Funkcionalno testiranje 32
5.2 Testiranje ročnega in avtomatskega dela 35
5.3 Uporabniško testiranje 38
5.4 Preverjanje validacije in sporočil o napakah 41

6 TEHNIČNA DOKUMENTACIJA 44

6.1 Opis modulov 44
6.2 UML-diagrami 47

7 UPORABNIŠKA NAVODILA 49

7.1 Uvod 49
7.2 Namestitev 49

7.2.1 Nastavitev bližnjic 50
7.3 ZAJEM ARTIKLA 51
7.4 IZBIRA FORMATA 53
7.4.1 Mali poličnik 53
7.5 DODAJANJE VEČ ARTIKLOV 54
7.6 PREGLED IN GENERIRANJE 54
7.7 REZULTAT 55
7.8 POGOSTI PROBLEMI 56
8 SKLEP 57
9 VIRI, LITERATURA 60

KAZALO SLIK

SLIKA 1: PRIMER LETAKA 12
SLIKA 2: PRAZNA IN IZPOLNJENA PREDLOGA OZNAČEVALCA 13
SLIKA 3: MAPNA STRUKTURA SISTEMA OZNACEVALEC 15
SLIKA 4: DIALOG ZA IZBIRO FORMATA PREDLOGE 18
SLIKA 5: DIALOG ZA DODATNE PARAMETRE PRI FORMATU MALI POLIČNIK 19
SLIKA 6: DIALOG ZA RAZPOREDITEV DVEH SLIK 19
SLIKA 7: DIALOG S PREGLEDOM ZAJETIH ARTIKLOV 20
SLIKA 8: NAPAKA OB POSKUSU UPORABE VGRAJENEGA OBJEKTA JSON V EXTENDSCRIPT 22
SLIKA 9: PRIMER VSEBINE DATOTEKE STATE.JSON PO ZAJEMU ENEGA ARTIKLA 23
SLIKA 10: NAPAČNO SKALIRANI ELEMENTI CENE PRI FORMATIH UGODNO IN VEDNO POCENI 26
SLIKA 11: PRAVILNO SKALIRANI ELEMENTI CENE PO POPRAVKU 26
SLIKA 12: PRIMER IKON TIPA GROUP 27
SLIKA 13: PRIMER IKON TIPA RECTANGLE OZIROMA GRAPHIC FRAME 27
SLIKA 14: PRIMERJAVA PREDLOG PO VELIKOSTI PROSTORA IN ORIENTACIJI 29
SLIKA 15: PRAZNA IN IZPOLNJENA PREDLOGA FORMATA MALI POLIČNIK 29
SLIKA 16: PRIMERA SPOROČIL O NAPAKI OB NEUSPEŠNI VALIDACIJI SELEKCIJE 31
SLIKA 17: DIAGRAM RAZREDOV ARTICLEMODEL IN APPSTATE 47
SLIKA 18: DIAGRAM STANJ SISTEMA OZNACEVALEC 48
SLIKA 19: DIAGRAM ZAPOREDJA ZAJEMA IN GENERIRANJA ARTIKLA 48
SLIKA 20: SLIKA VRSTICE #INCLUDEPATH 49

Slika 21: Dodelitev bližnjice Ctrl+Shift+1 skripti startCapture.jsx 50
Slika 22: Dodelitev bližnjice Ctrl+Shift+2 skripti confirmCapture.jsx 51
Slika 23: Primer označenih elementov artikla v letaku 52
Slika 24: Primeri različnih cenovk glede na format 53
Slika 25: Poziv za dodajanje novega artikla 54
Slika 26: Generirane datoteke v mapi izvoz/ 55

KAZALO TABEL

Tabela 1: Rezultati funkcionalnega testiranja 32
Tabela 2: Časi izvedbe ročnega in avtomatiziranega postopka 36
Tabela 3: Statistični povzetek meritev 36
Tabela 4: Opažanja med uporabniškim testiranjem 39
Tabela 5: Rezultati ankete – Blok A (navodila in samostojnost) 40
Tabela 6: Sporočila o napakah ob neuspešni validaciji selekcije 41
Tabela 7: Rezultati ankete – blok B (sporočila o napakah) 43
Tabela 8: Pogosti problemi in njihove rešitve 56

KAZALO GRAFIKONOV

Graf 1: Primerjava časa priprave med ročnim in avtomatiziranim postopkom 37

1 UVOD

Prvo poglavje predstavlja izhodišče diplomskega dela. Najprej opišem obravnavano področje in problem. Nato predstavim namen, cilje in hipoteze dela, opišem predpostavke in omejitve, s katerimi sem se srečal pri razvoju sistema Oznacevalec. Na koncu pa sledijo še raziskovalne metode, ki sem jih uporabil za odgovor na zastavljena raziskovalna vprašanja.

1.1 Opis področja in opredelitev problema

Diplomsko delo obravnava avtomatizacijo priprave promocijskih označevalcev v podjetju Jagros, d. o. o. (Trgovine Jager). Promocijski označevalec je tiskana cenovka, ki se namesti na prodajno polico ali izdelek in vsebuje sliko izdelka, naziv, ceno in po potrebi dodatne ikone ali oznako akcije oziroma popusta. Avtomatizacija procesov v specializiranih grafičnih orodjih, kot je Adobe InDesign, je v strokovni literaturi redko obravnavana, skriptiranje ExtendScript pa je slabo dokumentirano in omejeno predvsem na uradno dokumentacijo Adobe (Adobe Systems, 2013; Adobe Systems, 2026)

Za to temo sem se odločil, ker mi jo je med prakso pri Jagros, d. o. o., predstavil mentor Aleš Pongračič, ki je opozoril na to, kako zamuden je ročni postopek priprave označevalcev, opisan v poglavju 2. Grafik je moral za vsak artikel posebej odpreti predlogo, ročno skopirati elemente iz letaka in jih prilagoditi velikosti predloge, kar je pri večjem številu artiklov predstavljalo precejšnjo časovno obremenitev. Ideja se mi je zdela zanimiva in koristna, zato sem se odločil, da jo razvijem v okviru diplomskega dela.

Raziskovalni problem je, ali je mogoče s skriptiranjem v okolju Adobe InDesign ExtendScript obravnavani proces zadostno avtomatizirati ob upoštevanju tehničnih omejitev okolja, kot so podpora zgolj standardu ES3, odsotnost vgrajenega mehanizma za serializacijo podatkov in pomanjkanje trajnega spomina med zagoni skripte. Rešitev tega problema je pomembna, saj podjetju neposredno prihrani čas pri pripravi promocijskih materialov.

1.2 Namen, cilji in osnovne trditve

Namen diplomskega dela je razviti sistem, ki avtomatizira pripravo promocijskih označevalcev, in preveriti, ali taka avtomatizacija prinaša časovno prednost pred ročnim postopkom, ne da bi pri tem žrtvovala zanesljivost ali enostavnost uporabe.

Konkretni cilji diplomskega dela so naslednji. Prvi je razviti delujoč sistem Oznacevalec, ki podpira zajem elementov iz letaka in generiranje končnih datotek za dvanajst formatov predlog. Drugi cilj je rešiti tehnične omejitve okolja ExtendScript, ki jih podrobneje opisujem v poglavju 4.1. Tretji cilj je zagotoviti pravilno skaliranje in pozicioniranje grafičnih elementov različnih tipov. Četrti cilj je izmeriti in primerjati čas priprave med ročnim in avtomatiziranim postopkom, peti pa je pripraviti uporabniška navodila in pri uporabnikih, ki nimajo predhodnega tehničnega znanja, preveriti njihovo razumljivost.

Na podlagi opredeljenega problema sem postavil pet hipotez:

H1: Avtomatizacija priprave promocijskih označevalcev z Adobe InDesign ExtendScript skrajša čas priprave serije označevalcev v primerjavi z ročnim postopkom.

H2: Tehnične omejitve okolja ExtendScript (ES3, odsotnost vgrajenega JSON in trajnega spomina med sejami) je mogoče premostiti z lastnimi implementacijami pomožnih mehanizmov.

H3: Sistem je mogoče razviti tako, da podpira različne tipe grafičnih elementov in različne formate predlog znotraj enotne arhitekture brez podvajanja kode.

H4: Sistem Oznacevalec je mogoče dokumentirati z navodili uporabe, tako da je razumljiv uporabniku brez predhodnega tehničnega znanja.

H5: Sistem Oznacevalec z vgrajeno validacijo selekcije in jasnimi sporočili o napakah zmanjšuje možnost uporabniških napak, skladno z načeli človeku usmerjene informatike.

1.3 Predpostavke in omejitve

Pri pripravi diplomskega dela sem izhajal iz predpostavke, da ima podjetje Jagros, d. o. o., za vsak format označevalca že pripravljeno predlogo v Adobe InDesignu, saj sistem Oznacevalec predlog ne ustvarja, temveč jih le zapolni s podatki iz letaka. Prav tako sem predpostavil, da so elementi artikla v letaku pripravljeni ločeno kot posamezno izberljivi InDesign objekti (slika, tekstovno polje in skupina za ceno), ne pa združeni v eno samo sliko iz celotnega letaka.

Ena od omejitev raziskave je majhen vzorec testiranja uporabniške izkušnje, opisanem v poglavju 5.3, saj je sodelovalo le pet nestrokovnjakov in trije grafiki. Vzorec je bil omejen s časovnimi okviri diplomskega dela in razpoložljivostjo udeležencev. Zato rezultati odražajo zgolj vtise konkretnega vzorca, vendar kljub temu ponujajo koristen vpogled v razumljivost sistema pri uporabnikih brez predhodnega tehničnega znanja.

Podobna omejitev velja za meritve časa priprave, opisane v poglavju 5.2, ki temeljijo na treh ponovitvah vsakega postopka. Majhno število ponovitev daje le okvirno oceno razlike med postopkoma in ne zadostuje za statistično zanesljivo povprečje.

Dodatna omejitev je pomanjkanje strokovne literature s področja avtomatizacije procesov v specializiranih grafičnih orodjih, kot je Adobe InDesign, saj večina dostopnih virov obravnava splošna programska okolja. Tudi ExtendScript skriptiranje je dokumentirano predvsem prek uradnih virov Adobe, saj ga širša strokovna literatura komaj obravnava.

1.4 Uporabljene raziskovalne metode

Raziskava temelji na razvojno-projektnem pristopu. Sistem Oznacevalec je bil razvit iterativno v sodelovanju z mentorjem v podjetju Alešem Pongračičem, ki je podajal funkcionalne zahteve in povratne informacije, celotno tehnično zasnovo in implementacijo pa sem izvedel sam, kot je podrobneje opisano v uvodu poglavja 4.

Za odgovor na vprašanje o časovni primerjavi ročnega in avtomatiziranega procesa (H1) sem izvedel neposredno meritev časa priprave pri obeh postopkih, kar je opisano v poglavju 5.2. Glede odgovora na vprašanje o tehničnih omejitvah okolja (H2) sem pregledal uradno dokumentacijo Adobe (Adobe Systems, 2013; Adobe Systems, 2026), dokumentacijo standarda JSON (MDN Web Docs, b.d.-b) in referenčni priročnik JavaScript (Flanagan, 2020), rešitve pa sem dokumentiral v poglavjih 4.1 in 4.2. Na vprašanje o skaliranju in arhitekturi formatov (H3) sem odgovoril z analiziranjem lastne implementacije modula layoutRules.jsx, kjer sem upošteval vire s področja arhitekture sistemov (Sommerville, 2016). Odgovor na vprašanji o razumljivosti navodil (H4) in vplivu validacije na uporabniško izkušnjo (H5) sem pridobil empirično, in to s testiranjem uporabnikov brez predhodnega znanja InDesigna in z anketo, opisano v poglavju 5.3, pri čemer sem za H5 dodatno upošteval literaturo s področja preprečevanja napak (Flaherty, 2018).

2 ANALIZA OBSTOJEČEGA PROCESA

Izhodišče priprave označevalca je letak PDF oziroma InDesign dokument z aktualno akcijsko ponudbo, ki vsebuje slike, nazive in cene vseh artiklov. Za vsak format označevalca obstaja vnaprej pripravljena prazna predloga InDesigna z rezerviranim prostorom za sliko, naziv, ceno in morebitne ikone, ki jo je treba zapolniti s podatki posameznega artikla.



Slika 1: Primer letaka

Vir: (Lastni vir)

Pred razvojem sistema Oznacevalec je bil proces priprave promocijskih označevalcev v celoti ročen. Grafik je v Adobe InDesignu odprl letak in poiskal artikel, za katerega je bilo treba pripraviti označevalec. Vse elemente artikla, to je sliko, naziv in cenovko, je naenkrat kopiral iz letaka in jih prilepil v ustrezno predlogo. Elementi so imeli v letaku svoje dimenzije in pozicije, grafik pa je moral pri vsakem posebej ročno prilagoditi velikost in pozicijo, s čimer je ustrezno zapolnil predviden prostor v predlogi. Večina predlog za označevalce je bila zbrana v enem dokumentu InDesigna, nekatere, na primer formata Postrežno meso in Mali poličnik, pa so imele svoj ločeni InDesign dokument. Grafik je postopek ponovil za vsak artikel in na koncu shranil ustrezen dokument.



Slika 2: Prazna in izpolnjena predloga označevalca

Vir: (Lastni vir)

Podjetje Jagros, d. o. o., redno pripravlja promocijske označevalce. V povprečju je bilo treba pripraviti trideset označevalcev naenkrat, za kar je grafik pri ročnem postopku porabil približno dvajset minut za trideset artiklov. Natančnejša meritev in analiza ročnega in avtomatiziranega procesa je opisana v poglavju 5.2. Ker je bil vsak označevalec pripravljen ročno, je bil čas priprave neposredno odvisen od števila artiklov in formatov, ki jih je bilo treba pokriti.

Ročni proces je bil zamuden predvsem pri večjem številu označevalcev, kjer se je čas priprave seštevал za vsak artikel posebej. Poleg tega je vsaka sprememba, na primer drugi format ali drugačna postavitev elementov, zahtevala ročno popravljanje posameznega označevalca. Glavna motivacija za razvoj sistema Oznacevalec je bila avtomatizacija ponavljajočega se procesa in prihranek časa, ki ga je grafik prej porabil za ročno prilagajanje elementov.

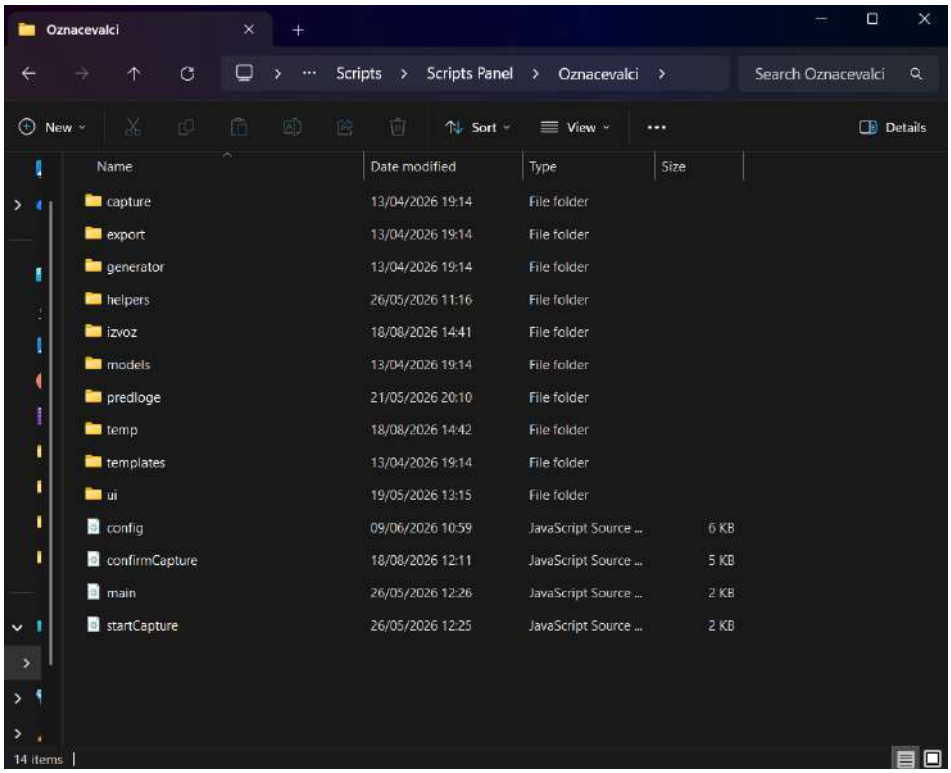
Razvoj sistema Oznacevalec je bil tako neposreden odgovor na opisane težave. Cilj je bil grafiku prihraniti čas in mu omogočiti, da se posveti zahtevnejšim nalogam. Podobne presnove procesov, pri katerih se ročno delo s tiskanimi gradivi nadomesti z digitalnim, torej z avtomatiziranim postopkom, opisujejo diplomska dela s področja reinženiringa poslovnih procesov (Brumen, 2025).

3 NAČRTOVANJE SISTEMA

Pred začetkom razvoja sistema Oznacevalec sem moral sprejeti več načrtovalskih odločitev, ki so vplivale na celotno arhitekturo. Ker ExtendScript ne podpira modernih programskih vzorcev in ima številne tehnične omejitve, opisane v poglavju 4.1, sem moral prilagoditi pristop razvoja. Odločil sem se za objektno orientiran pristop, kjer je vsak del sistema ločen objekt s svojimi funkcijami in lastnostmi (MDN Web Docs, b.d.-c). Tako je na primer AppState objekt, ki upravlja globalno stanje aplikacije, ScalingEngine skrbi za skaliranje elementov, FrameUtils pa je objekt s pomožnimi funkcijami za delo z okvirji. Tak pristop je med moduli omogočil jasno razdelitev odgovornosti. Vsak objekt skrbi samo za svoje področje, brez poseganja v delo drugih objektov. Poleg objektno orientiranega pristopa sem moral načrtovati tudi način komunikacije med moduli, strukturo podatkovnega modela za posamezen artikel in mehanizem shranjevanja stanja med sejami, saj ExtendScript med zagoni skripte nima spomina. Vsaka od teh odločitev je vplivala na končno arhitekturo sistema (Adobe Systems, 2026).

3.1 Arhitektura sistema

Sistem Oznacevalec sem zasnoval kot modularen sistem, razdeljen na več datotek, od katerih ima vsaka jasno nalogo. Tak pristop sem izbral zato, ker bi en sam dolg skript hitro postal nepregleden in težko vzdržen, posamezne module pa je lažje testirati in nadgrajevati neodvisno drug od drugega (Sommerville, 2016). Vse datoteke so organizirane v mape glede na funkcionalnost: models/ vsebuje podatkovne modele, capture/ skrbi za zajem elementov, ui/ vsebuje dialoge za interakcijo z uporabnikom, generator/ skrbi za generiranje označevalcev, helpers/ vsebuje pomožne funkcije in export/ skrbi za shranjevanje končnih datotek.



Slika 3: Mapna struktura sistema Oznacevalec

Vir: (Lastni vir)

Vstopni točki sistema sta dve skripti, startCapture.jsx in confirmCapture.jsx, ki ju uporabnik sproži prek tipkovničnih bližnjic. Obe skripti vključujeta modul main.jsx, ki ob zagonu v pravilnem vrstnem redu naloži vse ostale module. Vrstni red vključevanja je pomemben, saj mora biti vsak modul naložen, preden ga drug modul pokliče. Tako mora biti jsonHelper.jsx naložen pred appState.jsx, ker ga ta uporablja za branje in pisanje datoteke state.json. Ker obe vstopni skripti vključujeta main.jsx, so vsi moduli vedno naloženi ne glede na to, katero bližnjico uporabnik pritisne.

Centralna točka celotnega sistema je datoteka config.jsx, ki vsebuje konstante in nastavitve sistema na enem mestu. Tu so definirani formati predlog v objektu FORMATS, poti do predlog v objektu TEMPLATES, omejitve velikosti pisave v objektu FONT_SIZES ter mere in koordinate za format Mali poličnik. Konfiguracijo sem centraliziral na enem mestu. Brez tega bi bile vrednosti razpršene po celotni kodi in vsaka sprememba, na primer dodajanje novega formata, bi zahtevala popravke na več mestih hkrati. Da se sprememba odraža v celotnem sistemu, je dovolj, da se s config.jsx spremeni vrednosti samo na enem mestu (Sommerville, 2016).

Podatki tečejo skozi sistem v jasno določenem vrstnem redu. Uporabnik najprej pritisne prvo bližnjico, ki zažene startCapture.jsx. Ta sistem označi kot aktiven in shrani informacijo v datoteko state.json, da jo druga bližnjica ob zagonu prebere in ve, da je bil sistem pravilno aktiviran. Nato uporabnik v letaku označi elemente artikla v določenem vrstnem redu: najprej slike, nato naziv in ceno ter nazadnje ikone.

Ko pritisne drugo bližnjico, se zažene confirmCapture.jsx, ki najprej obnovi stanje iz state.json in preveri, ali je sistem aktiven. Če je selekcija veljavna, captureArticle.jsx prebere označene elemente in zapolni articleModel objekt, ki vsebuje podatke o artiklu. Temu sledi dialog za izbiro formata predloge, po potrditvi pa sistem vpraša, ali želi uporabnik dodati še en artikel. Ko so vsi artikli zajeti, se uporabniku prikaže okno, kjer ima pregled nad vsemi artikli. Tam se odloči, ali bo katerega od artiklov odstranil, sprožil generiranje končnih datotek ali pa proces prekinil in začel znova.

Če uporabnik izbere generiranje, potem MarkerGenerator.jsx za vsak artikel po vrsti odpre ustrezno predlogo prek templateLoader.jsx, ki pot do predloge prebere iz config.jsx. Elemente kopira v predlogo, layoutRules.jsx pa jih pozicionira glede na koordinate in dimenzije področij. ScalingEngine.jsx jih skalira na pravo velikost, medtem ko textAdjustments.jsx prilagodi velikost fonta naziva glede na omejitve iz FONT_SIZES v config.jsx. Na koncu exportHandler.jsx shrani izpolnjeno predlogo kot novo datoteko v mapo izvoz/.

Modularna arhitektura sistema Oznacevalec je omogočila pregleden razvoj in enostavno odpravljanje napak, saj je bilo ob nastanku težav vedno jasno, kateri modul je odgovoren za določeno funkcionalnost. Vsaka sprememba zahtev s strani mentorja v podjetju je zahtevala zgolj popravek v ustreznem modulu. Tak pristop se je izkazal kot ključna prednost pri iterativnem razvoju sistema, kjer so se zahteve sproti dopolnjevale in spreminjale (Sommerville, 2016).

3.2 Podatkovni model

Vsak artikel, ki ga uporabnik zajame iz letaka, je v sistemu predstavljen kot JavaScript objekt, ustvarjen s funkcijo createArticle(). Ta objekt sem poimenoval articleModel in vsebuje vsa polja, ki so potrebna za generiranje označevalca: sliko, naziv, ceno, ikono, format in dodatne parametre za format Mali poličnik, kot so popust, opis popusta, znamka in datum veljavnosti. Poleg tega vsebuje metapodatke, kot so unikatni ID, časovni žig in ime izvornega dokumenta.

Podatkovni model v obliki ločenega objekta je zagotovil, da so vsi podatki o artiklu shranjeni na enem mestu in dostopni vsem tistim modulom sistema, ki jih potrebujejo.

Polja slike, ikone in origSlikeImena so definirana kot prazni seznami, medtem ko so vsa ostala polja privzeto nastavljena na null. Polja slike predstavljajo seznam, saj sistem podpira več slik za en artikel, prav tako pa so seznam tudi ikone, saj je lahko tudi teh več. V Adobe InDesignu ima vsak element v dokumentu svoje ime, kar je vidno v Layers panelu, in ga grafiki pogosto ročno nastavijo za lažjo orientacijo v kompleksnih dokumentih. Ker sistem ob zajemu vsakemu elementu dodeli unikatno ime za iskanje prek state.json, bi ta trajno prepisal originalno ime elementa v letaku. V izogib navedenemu sem dodal posebno skupino polj: origSlikeImena, origNazivIme, origCenaIme in origIkoneImena, ki shranijo originalna imena pred preimenovanjem. Funkcija reset() jih po zaključku generiranja obnovi, tako da elementi v letaku ohranijo svoja prvotna imena.

Ko je articleModel objekt zapolnjen z vsemi podatki, ga captureArticle.jsx preda AppState.articles, kjer se shrani skupaj z vsemi ostalimi zajetimi artikli. Skozi celoten proces generiranja moduli dostopajo do objekta za branje podatkov. MarkerGenerator.jsx iz njega prebere slike, naziv, ceno in ikone za kopiranje v predlogo, layoutRules.jsx prebere format za določitev koordinat pozicioniranja, exportHandler.jsx pa format za sestavo imena izhodne datoteke. Na ta način articleModel služi kot skupna podatkovna struktura, ki jo za svoje potrebe berejo in uporabljajo vsi moduli sistema.

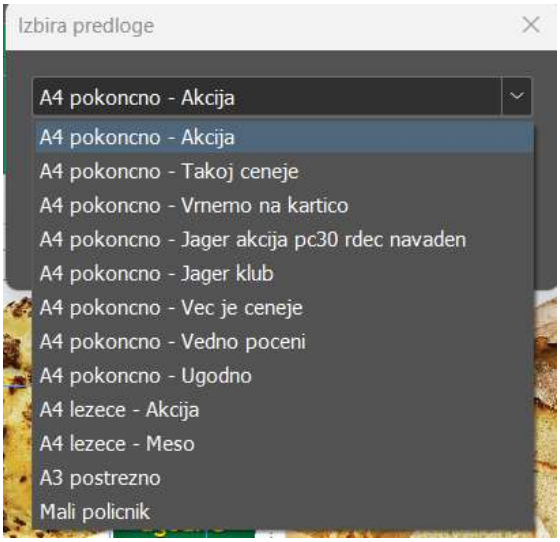
Podatkovni model articleModel se je med razvojem izkazal kot dobra osnova celotnega sistema. Ker je vsak artikel samostojni objekt z vsemi potrebnimi podatki na enem mestu, so moduli med seboj neodvisni, vsak pa dostopa do tistih podatkov, ki jih potrebuje. Ko je mentor v podjetju med razvojem pri enem artiklu zahteval podporo več slik, je bilo treba polje slike zamenjati z njenim seznamom. Enako je veljalo za dodajanje posebnih parametrov za format Mali poličnik, kjer sem dodal nova polja za popust, popustOpis, znamko in datum. Tak pristop je omogočil, da se je sistem brez večjih posegov v obstoječo kodo sproti prilagajal novim zahtevam.

3.3 Dialogi in interakcija z uporabnikom

Interakcija uporabnika s sistemom Oznacevalec poteka prek štirih dialogov, ki se med procesom zajema prikazujejo v določenem vrstnem redu. Pri načrtovanju dialogov sem se osredotočil na to, da so čim enostavnejši in razumljivejši, saj sem želel, da lahko sistem

uporablja tudi uporabnik brez tehničnega znanja (Flaherty, 2018). Vsak dialog od uporabnika zahteva zgolj en vnos, in to brez tehničnih informacij, ki bi ga lahko zavedle ali zmotile.

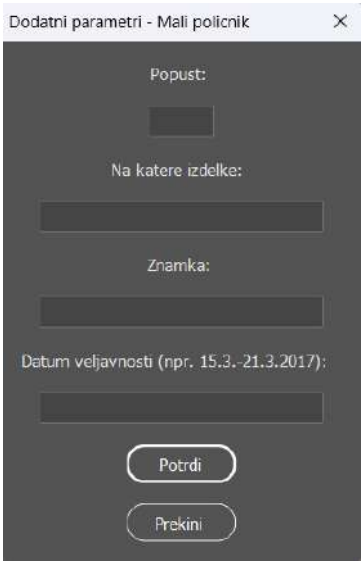
Prvi dialog, ki se prikaže po zajemu elementov, je formatDialog, kjer uporabnik iz spustnega seznama izbere format predloge. Seznam vsebuje vseh dvanajst formatov, privzeto pa je izbrana prva možnost. Namesto gumbov sem se odločil za spustni seznam, saj je število formatov za gumbe preveliko, hkrati pa je spustni seznam standardni element vmesnika, ki ga večina uporabnikov že pozna.



Slika 4: Dialog za izbiro formata predloge

Vir: (Lastni vir)

Ko uporabnik izbere format Mali poličnik, se samodejno odpre extraParamsDialog z dodatnimi polji, specifičnimi za ta format. Uporabnik tu vnese odstotek popusta in opis, na katere izdelke se popust nanaša, poleg tega pa še ime znamke in datum veljavnosti. Za razliko od ostalih formatov uporabnik podatke vnese ročno, saj so na formatu Mali poličnik prikazani drugače kot na standardnem označevalcu. Dialog ima polja za vnos besedila ter gumba potrditve in preklica.



Slika 5: Dialog za dodatne parametre pri formatu Mali poličnik

Vir: (Lastni vir)

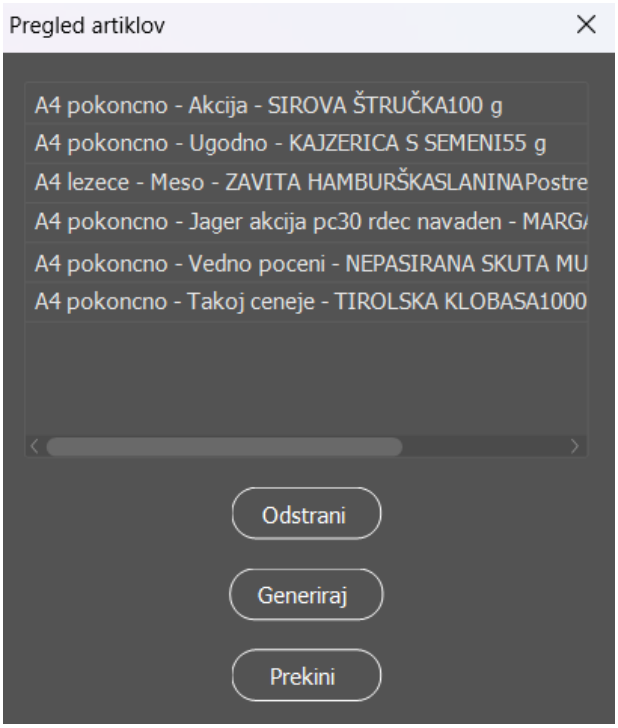
Kadar uporabnik zajame dve sliki, se po izbiri formata prikaže slikeRazporeditevDialog, ki vpraša, kako naj sistem razporedi sliki v predlogi. Uporabnik izbere med dvema možnostma postavitve slik: ena zraven druge ali ena nad drugo. Tako na primer dve pokončni sliki vizualno delujeta bolje ena zraven druge, medtem ko je postavitev ene nad drugo primernejša za bolj podolgovate slike. Ta dialog se prikaže samo pri izboru dveh slik, saj je pri eni in treh slikah razporeditev že vnaprej določena. Za ločen dialog sem se odločil, ker grafik najbolje presodi, katera postavitev je za posamezni artikel najprimernejša.



Slika 6: Dialog za razporeditev dveh slik

Vir: (Lastni vir)

Ko so zajeti vsi artikli, se prikaže listingDialog vseh zajetih artiklov. Vsak je prikazan z imenom formata in nazivom artikla, kar uporabniku omogoča hiter pregled zajetega. V primeru napake lahko uporabnik v tem dialogu posamezen artikel tudi odstrani. Na dnu dialoga sta gumba »Generiraj«, ki sproži generiranje vseh označevalcev, in »Prekini«, ki prekine proces brez generiranja. Možnost pregleda pred generiranjem je bila zahteva mentorja v podjetju, saj je generiranje večjega števila artiklov časovno zamudno, kar bi se zlasti odrazilo v primeru, če bi moral uporabnik zaradi napačno zajetega artikla postopek začeti znova.



Slika 7: Dialog s pregledom zajetih artiklov

Vir: (Lastni vir)

Dialogi sistema Oznacevalec so zasnovani tako, da uporabnika skozi proces vodijo korak za korakom, pri tem pa mu ni treba razumeti tehničnega ozadja sistema. Vsak dialog se prikaže ob pravem trenutku in zahteva samo tisto informacijo, ki je v danem koraku potrebna. Pri načrtovanju dialoga sem moral pretehtati, kako postaviti vprašanje, katere možnosti ponuditi in kako poimenovati gumbe, da bi bil vmesnik čim bolj intuitiven. Cilj je bil pripraviti uporabniška navodila, ki bi tudi uporabniku brez predhodnega poznavanja sistema omogočila jasno in celovito razumevanje vseh korakov postopka brez potrebe po dodatnih pojasnilih. Navedeno bo preverjeno s testom uporabniške izkušnje v poglavju 5.3 (Flaherty, 2018).

4 IMPLEMENTACIJA SISTEMA

V tem poglavju je opisana implementacija sistema Oznacevalec. Mentor v podjetju mi je na začetku razvoja predstavil zamisel, kako naj sistem deluje in kako naj bo videti končni rezultat. Razložil je, katere funkcionalnosti potrebuje in kateri koraki naj bodo avtomatizirani. Samostojno sem izdelal celotno tehnično zasnovo in implementacijo, torej arhitekturo, prav tako pa tudi podatkovni model in vso kodo. Med razvojem sem mentorja sproti seznanjal z razvojem sistema, on pa je ob tem podal predloge in zahteve, ki sem jih upošteval. Vsaka tehnična rešitev, opisana v nadaljevanju tega poglavja, je lastna odločitev, sprejeta kot odgovor na konkretno omejitev okolja ExtendScript ali na zahtevo, ki se je pojavila med razvojem. Poglavje je razdeljeno na pet podpoglavij, ki pokrivajo ključne tehnične izzive: omejitve okolja ExtendScript, shranjevanje stanja med sejami, skaliranje grafičnih elementov, podporo različnim formatom in validacijo selekcije.

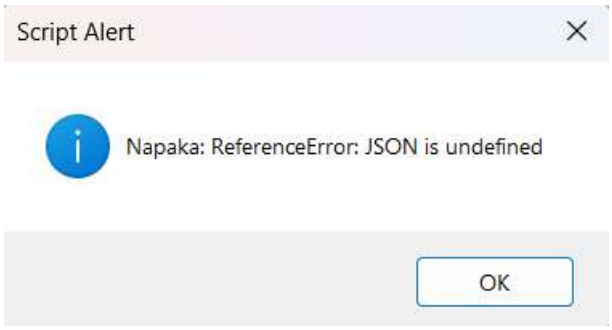
4.1 ExtendScript in omejitve okolja

Okolje ExtendScript je Adobeova implementacija skriptnega jezika JavaScript (MDN Web Docs, b.d.), ki jo Adobe uporablja za avtomatizacijo svojih aplikacij, kot so InDesign, Illustrator in Photoshop (Adobe Systems, 2013; Hopkins, 2009; Sagar, 2025). Čeprav temelji na standardu ECMAScript, ExtendScript podpira samo starejši standard ES3 iz leta 1999, kar pomeni, da številne moderne funkcionalnosti JavaScripta niso na voljo (Flanagan, 2020). To je pri razvoju sistema Oznacevalec predstavljalo pomembno tehnično omejitev.

Ker ExtendScript okolje podpira samo standard ES3, sem bil pri razvoju omejen na uporabo starejših konstruktov jezika. Na voljo so bile samo spremenljivke deklarirane z var, brez podpore za let in const, brez arrow funkcij, brez razredov in brez destrukturiranja objektov (prav tam). To pomeni, da sem moral pisati obsežnejšo kodo in se zavedati posebnosti deklaracij var, kot je function scope namesto block scope. Pri razvoju sistema Oznacevalec je to povzročilo konkretne težave, zlasti pri zankah, kjer sta bili spremenljivki i in j večkrat deklarirani znotraj istih funkcij, kar je povzročalo nepredvideno obnašanje. Kjer je prihajalo do takih težav, sem spremenljivke deklariral na vrhu funkcije, kar predstavlja standardno prakso pri pisanju kode v skladu z ES3.

Ena izmed večjih tehničnih omejitev okolja ExtendScript je odsotnost vgrajenega JSON objekta, ki je v modernih JavaScript okoljih standardno orodje serializacije in deserializacije

podatkov (MDN Web Docs, b.d.-b). Ker sistem Oznacevalec potrebuje shranjevanje stanja med sejami, je bilo treba za pretvorbo JavaScript objektov v JSON format in nazaj implementirati lastni mehanizem. V ta namen je bil razvit modul JSONHelper, ki vsebuje dve funkciji: stringify() za pretvorbo objekta v niz JSON in parse() za pretvorbo niza JSON nazaj v objekt. Funkcija parse() za pretvorbo uporablja new Function(), saj eval() v ExtendScriptu povzroča napake. Ta ustvari anonimno funkcijo in jo takoj pokliče, s čimer niz JSON varno pretvori nazaj v JavaScript objekt (Adobe Systems, 2013).



Slika 8: Napaka ob poskusu uporabe vgrajenega objekta JSON v ExtendScript

Vir: (Lastni vir)

Dodatna omejitev okolja ExtendScript je odsotnost trajnega spomina med posameznimi zagoni skripte. V modernih programskih okoljih je mogoče podatke shraniti v globalne spremenljivke ali pomnilnik aplikacije, ki ostanejo med sejami aktivni. V ExtendScriptu bi bil za to primeren objekt $.global, ki naj bi omogočal shranjevanje podatkov med zagoni. Vendar pa se v praksi ob vsakem zagonu skripte ponastavi na začetne vrednosti (Adobe Systems, 2026). To je pri razvoju sistema Oznacevalec predstavljalo ključen izziv, saj sistem deluje v več korakih, in sicer uporabnik najprej s prvim ukazom sistem aktivira, nato označi elemente in z drugim ukazom potrdi zajem. Med tema dvema zagonoma je bilo treba ohraniti stanje sistema. Rešitev se je pokazala v shranjevanju stanja v zunanjo datoteko state.json, ki se ob zagonu prebere in ob spremembi zapiše na disk.

Opisane tehnične omejitve okolja ExtendScript so bistveno vplivale na arhitekturne odločitve pri razvoju sistema Oznacevalec. Vsaka omejitev je zahtevala svojo rešitev, ki je bila razvita iterativno med samim razvojem sistema. Lastna implementacija JSON serializacije je podrobneje opisana v poglavju 4.2, kjer je predstavljeno celotno shranjevanje stanja med sejami.

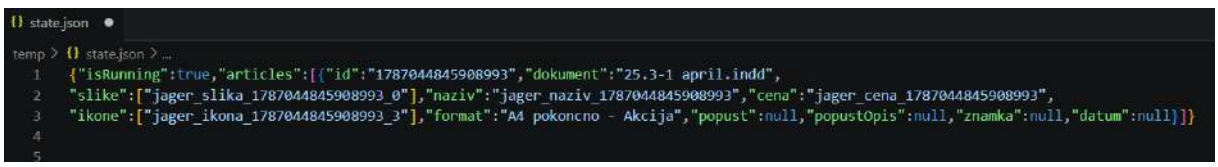
4.2 Shranjevanje stanja med sejami

Sistem Oznacevalec deluje v dveh ločenih korakih, ki ju uporabnik sproži z dvema različnima ukazoma. V prvem koraku uporabnik aktivira sistem in označi elemente artikla v letaku, v drugem koraku pa potrdi zajem in izbere format predloge. Ker sta ta dva koraka izvedena kot ločena zagona skripte, je bilo treba med njima zagotoviti mehanizem za ohranitev stanja. Brez shranjevanja stanja bi sistem ob vsakem zagonu izgubil vse prej zajete podatke, kar bi onemogočilo zajem več artiklov v eni seji.

Preden sem se odločil za pristop z zunanjo datoteko, sem preizkusil dve drugi rešitvi, ki pa nista bili uspešni. Prva je bila uporaba objekta $.global, ki je v ExtendScriptu namenjen shranjevanju podatkov med zagoni skripte. Pristop je sprva deloval obetavno, saj so se dialogi prikazovali pravilno. Vendar je bilo pri testiranju zajema več artiklov ugotovljeno, da se seznam ne poveča, kajti listingDialog je ne glede na to, koliko artiklov je bilo zajetih, vedno prikazal samo enega. Razlog je bil v tem, da se objekt $.global v okolju InDesign 2025 ob vsakem zagonu skripte prek bližnjice ponastavi na začetno stanje, zaradi česar se vrednosti med posameznimi zagoni skripte ne ohranijo (Adobe Systems, 2026).

Druga možnost je bila, da bi skripta ostala aktivna v while zanki znotraj confirmCapture.jsx. Skripta bi torej ostala v zanki in čakala, da uporabnik označi naslednji artikel, ob tem pa se ne bi zaključila in ponovno zagnala. Ta pristop je bil zavrnjen, kajti dialog confirm(), ki upravlja zanko, blokira celoten InDesign. Med odprtim pogovornim oknom uporabnik ne more dostopati do elementov letaka in jih neposredno označevati.

Za shranjevanje stanja sem po preizkusu različnih pristopov izbral zapisovanje v zunanjo datoteko state.json, ki se nahaja znotraj projektne strukture v mapi temp/. Pristop z zunanjo datoteko je bil izbran, ker objekt $.global med sejami ne ohrani vrednosti, ExtendScript pa zanesljivo podpira branje in pisanje datotek. Ob vsakem zajemu artikla se stanje shrani v datoteko, ob naslednjem zagonu skripte pa se prebere in obnovi. Datoteka state.json vsebuje seznam vseh zajetih artiklov in vrednost isRunning, ki izraža trenutno aktivnost sistema.



Slika 9: Primer vsebine datoteke state.json po zajemu enega artikla

Vir: (Lastni vir)

Za format datoteke sem namesto zapisa CSV izbral JSON, saj mora datoteka hraniti gnezdeno strukturo podatkov. Vsak artikel v seznamu lahko ima več slik in ikon, kar pomeni sezname spremenljive dolžine znotraj posameznega zapisa, poleg tega pa articleModel vsebuje tudi polja, ki so lahko null. CSV je zasnovan za ravne tabelarične podatke, kjer ima vsaka vrstica enako število stolpcev, zato bi shranjevanje seznamov znotraj posamezne vrstice zahtevalo dodatno ločevanje z ločili znotraj celic in ročno razčlenjevanje ob branju, kar bi posledično lahko pripeljalo do več napak. JSON tovrstno gnezdeno strukturo podpira neposredno, zato je predstavljal naravnejši izbor shranjevanja stanja sistema Oznacevalec.

Elementi InDesigna, kot so slike, tekstovni okvirji in elementi tipa Group, so kompleksni C++ objekti, ki jih ni mogoče neposredno pretvoriti v JSON format, saj ta podpira samo nize znakov, številke, logične vrednosti, sezname in navadne objekte (MDN Web Docs, b.d.-b). Zato sem se odločil za pristop, ko sem ob zajemu artikla vsakemu elementu dodelil unikatno ime, sestavljeno iz predpone glede na tip elementa in unikatnega ID-artikla, originalna imena pa so se hranila v articleModel, kar je opisano v poglavju 3.2. V datoteko state.json se nato shrani ime kot navaden niz znakov, ob obnovi stanja pa sistem med vsemi elementi odprtega dokumenta poišče element po tem imenu.

Shranjevanje stanja je znotraj modula appState.jsx implementirano v funkciji saveState(). Funkcija najprej zbere vse zajete artikle in za vsakega pripravi objekt, v katerem so elementi zamenjani z njihovimi imeni (na primer tekstovni okvir se z nazivom artikla shrani kot niz znakov "jager_naziv_001"). Temu doda še vrednost isRunning in celoten objekt s pomočjo lastne implementacije JSONHelper.stringify() pretvori v niz JSON. Rezultat se zapiše v datoteko state.json.

Obnova stanja je implementirana v funkciji loadState(), ki ob zagonu prebere datoteko state.json, jo razčleni s pomočjo JSONHelper.parse() in za vsako shranjeno ime elementa poišče dejanski objekt med elementi odprtega dokumenta. Da bi se izognil počasnemu iskanju artiklov po vseh elementih dokumenta, sem implementiral razpršeno tabelo. Za vsak odprt dokument sistem enkrat prebere vse elemente in jih shrani v objekt, kjer je ključ elementa njegovo ime, vrednost pa dejanski element. Ko sistem išče element po imenu, ga najde direktno po ključu brez pregledovanja celotnega seznama. Pri večjem številu zajetih artiklov to bistveno pospeši obnovo stanja.

Ko uporabnik zaključi generiranje ali proces prekine, se v modulu appState.jsx pokliče funkcija reset(). Najprej se obnovijo originalna imena vseh elementov, ki so se med zajemom

preimenovali v unikatna imena, kot je na primer "jager_naziv_001". To je pomembno, saj elementi ostanejo odprti v izvornem dokumentu InDesigna in bi trajno preimenovanje motilo nadaljnje delo. Po obnovi imen se izprazni seznam artiklov, vrednost isRunning se nastavi na false in datoteka state.json se izbriše z diska. Vsaka naslednja seja se tako začne v začetnem stanju, brez podatkov iz predhodne seje.

Mehanizem shranjevanja stanja je omogočil, da sistem Oznacevalec deluje zanesljivo tudi pri zajemu večjega števila artiklov v eni seji. Edina omejitev je, da mora biti InDesign dokument, iz katerega so bili artikli zajeti, do zaključka generiranja ves čas odprt. Če uporabnik dokument zapre, sistem ob obnovi stanja ne more najti elementov po imenu, zato take artikle preskoči.

4.3 Skaliranje elementov

Ker uporabnik elemente artikla zajema iz letaka, ki ima svoje dimenzije in postavitev, jih je treba po kopiranju v predlogo ustrezno prilagoditi prostoru, ki je v predlogi rezerviran za posamezen element. Slika iz letaka je lahko poljubno velika, cena in ikone pa so kompleksni objekti tipa Group, ki lahko vsebujejo gnezdene skupine, tekstovne okvirje in grafične elemente, kar je pri skaliranju zahtevalo posebno obravnavo. V ta namen je bil razvit modul scalingEngine.jsx, ki skrbi za proporcionalno skaliranje vseh elementov na ciljni prostor v predlogi. Skaliranje je proporcionalno, kar pomeni, da se razmerje stranic elementa vedno ohrani, tako da ni element nikoli videti popačen.

Najlažje izvedljivo je bilo skaliranje slik. Slikovnemu okvirju sem glede na rezerviran prostor nastavil nove dimenzije, nato pa poklical funkcijo fit() z možnostma PROPORTIONALLY in CENTER_CONTENT. InDesign sliko ob ohranjanju razmerja stranic samodejno skrči ali poveča in jo centrira znotraj okvirja.

Skaliranje cene je bilo zahtevnejše, ker cena ni preprost slikovni element, temveč predstavlja Group, ki vsebuje gnezdene skupine, tekstovne okvirje in grafične elemente, ki tvorijo ozadje. Najprej sem izračunal dva faktorja skaliranja: enega za širino in enega za višino, tako da sem širino in višino prostora v predlogi delil s širino in višino Groupa. Nato sem izbral manjšega od teh dveh faktorjev, da Group na obeh straneh ne bi presegel mej predloge. Group sem nato skaliral z metodo resize(). Težava je bila v tem, da resize() ne skalira besedila znotraj tekstovnih okvirjev v ceni, zato so se grafični elementi pravilno skalirali, številke pa so ostale enako velike. Rešitev sem našel v ročnem skaliranju lastnosti pointSize za vsak odstavek znotraj tekstovnega okvirja v Groupu. Ker je bilo besedilo pozicionirano z lastnostjo baselineShift, ki določa

vertikalni odmik besedila od osnovnice, sem moral to vrednost skupaj s pointSize proporcionalno skalirati, sicer bi bilo po skaliranju besedilo na napačnem mestu.

Pri dveh formatih pa resize() ni bil dovolj. Po skaliranju slikovni element »Vedno poceni« na dnu cene in modri pas pri različici »Slovenska košarica« nista sledila preostalim elementom, temveč sta ostala v zgornjem levem kotu strani namesto na predvidenem mestu v cenovnem elementu. Podobno se je zgodilo pri formatu Ugodno, kjer zeleni kvadrat pri napisu Ugodno ni sledil skaliranju.



Slika 10: Napačno skalirani elementi cene pri formatih Ugodno in Vedno poceni

Vir: (Lastni vir)

Težavo sem odkril tako, da sem v generiranem označevalcu pregledal posamezne plasti v panelu Layers in za vsako posebej preveril, kje je pozicionirana. Ko sem ugotovil, kateri elementi so ostali na napačnem mestu, sem vsakega ročno postavil na pravilno pozicijo in odčital koordinate, ki sem jih nato kot fiksne vrednosti zapisal neposredno v kodo. Pri Vedno poceni sem problematične elemente poiskal po tipu znotraj strukture Groupa, pri Ugodnem pa sem iskal po indeksu allPageItems, ker je struktura zelenega kvadrata vsebovala več kot 300 gnezdenih elementov, zato iskanje po tipu ni bilo smiselno. Na tak način je bil rešen problem napačnega sledenja določenim delom Groupa.



Slika 11: Pravilno skalirani elementi cene po popravku

Vir: (Lastni vir)

Skaliranje ikon je zahtevalo poseben pristop, saj strukture ikon med seboj niso enake. Med testiranjem sem ugotovil, da se pri določenih ikonah nekateri elementi po skaliranju niso premaknili na pravo mesto, kar sem enako kot pri cenah odkril s pregledom plasti v panelu Layers. Ker so bile ikone različnih tipov, sem implementiral logiko, ki najprej preveri tip elementa in glede na to izbere ustrezen pristop.



Slika 12: Primer ikon tipa Group

Vir: (Lastni vir)

Pri ikoni Group sistem najprej preveri, ali med svojimi elementi vsebuje gnezdeno skupino. Če jo vsebuje, sistem uporabi metodo resize(), ki skalira celotno strukturo skupaj, nato pa jo Group premakne na izračunano pozicijo. Če gnezdene skupine ni, sistem z geometricBounds nastavi nove koordinate celotnega Groupa, zadnjemu elementu v allPageItems pa nastavi enake koordinate, ker se sicer ta ne premakne skupaj z Groupom.



Slika 13: Primer ikon tipa Rectangle oziroma Graphic frame

Vir: (Lastni vir)

Pri ikoni Rectangle ali slikovnemu okviru, to so ikone navadnih slik, sistem nastavi nove dimenzije. Nato za proporcionalno skaliranje vsebine pokliče fit() in element premakne na pravo mesto.

Ko sem razrešil vse opisane težave, je skaliranje elementov pri vseh formatih delovalo zanesljivo. Cene in ikone se pri vsakem generiranju ustrezno skalirajo ne glede na njihovo velikost v izvornem letaku. Edina znana omejitev ostaja pri slikah z belim robom, kjer sistem ne more ločiti vsebine od praznega prostora in jih je treba po generiranju ročno popraviti v InDesignu. Kljub tej omejitvi modul scalingEngine.jsx zagotavlja, da so vsi elementi po generiranju proporcionalno pravilno skalirani in v predlogi pozicionirani znotraj rezervnega prostora.

4.4 Podpora različnim formatom

Sistem Oznacevalec podpira dvanajst formatov predlog, ki se med seboj razlikujejo po velikosti, orientaciji in razporeditvi prostora za elemente artikla. Vsi formati so deklarirani v modulu config.jsx, kjer so določeni njihovi nazivi za prikaz v spustnem seznamu dialoga ter poti do ustreznih predlog. Na podlagi formata, ki ga uporabnik izbere v formatDialogu, templateLoader odpre ustrezno predlogo, layoutRules pa v njej za vsak element določi koordinate prostora. Ker predloge za elemente nimajo poimenovanih področij, sem koordinate za vsak format izmeril ročno in neposredno v predlogah, ki mi jih je posredoval mentor.

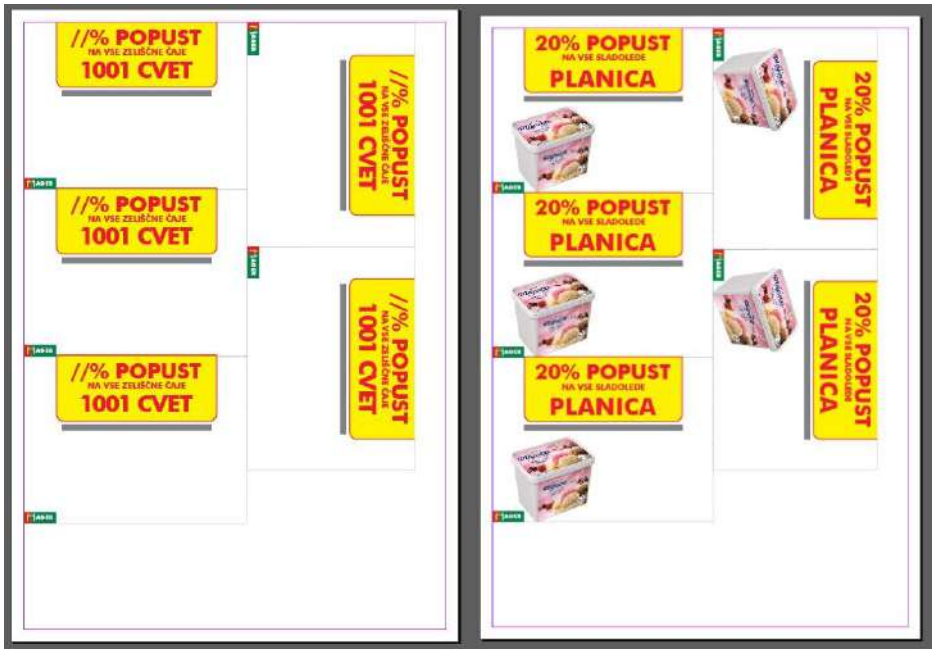
Formati se med seboj razlikujejo v treh pogledih. Prva razlika so dimenzije prostora za elemente, saj ima format Akcija, ki nima noge, večji prostor za sliko in ceno. Formata Takoj ceneje in Vrnemo na kartico pa imata poleg glave tudi nogo, kar prostor za elemente zmanjša. Druga razlika je orientacija strani: formati, kot sta A4 ležeče in A3 postrežno, uporabljajo ležeče predloge, zato so v datoteki layoutRules.jsx koordinate prostorov za posamezne elemente prilagojene, tako da je njihova širina večja od višine, medtem ko veljajo pri pokončnih formatih obratna razmerja. Tretja razlika je način pridobivanja podatkov o artiklu: večina formatov vse elemente (sliko, naziv, ceno in ikone) samodejno zajame iz letaka, format Mali poličnik pa naziv in ceno pridobi drugače. Uporabnik ju vnese ročno, sistem pa vrednosti vstavi v vnaprej poimenovane tekstovne okvirje v predlogi.



Slika 14: Primerjava predlog po velikosti prostora in orientaciji

Vir: (Lastni vir)

Format Mali poličnik se od vseh ostalih formatov najbolj razlikuje, zato zahteva drugačen pristop. Na eni strani formata A4 je pet poličnikov, kjer vsak prikazuje isti artikel. Ker Mali poličnik ne prikazuje naziva in cene artikla iz letaka, se ti dve polji po zajemu nastavita na null, čeprav ju validacija selekcije še vedno zahteva. Ko uporabnik izbere ta format, se odpre dodatni dialog extraParamsDialog, kjer vpiše za ta format specifične podatke: odstotek popusta in opis, na katere izdelke se popust nanaša, znamko ter datum veljavnosti. Slika artikla se nato kopira na vseh pet mest v predlogi, ostali podatki pa se vpišejo v vnaprej poimenovane tekstovne okvirje.



Slika 15: Prazna in izpolnjena predloga formata Mali poličnik

Vir: (Lastni vir)

Ta rešitev pa ima pomanjkljivost. Ker sistem ne glede na format selekcijo vedno validira po enakih pravilih, mora uporabnik pri Malem poličniku kljub temu označiti tudi naziv in ceno, čeprav ju format ne uporablja in se obe polji v sistemu nastavita na null. V praksi to ne pomeni veliko dodatnega dela, saj gre le za dve dodatni označitvi elementov, vseeno pa gre za nepotrebno zahtevo, ki izhaja iz tega, da se format izbere šele po zajemu elementov. Mogoča rešitev bi bila, da uporabnik format izbere pred zajemom elementov in se torej validacija izvede na izbran format, tako da ima Mali poličnik svoj ločeni postopek zajema, ali pa da validacija za ta format ne zahteva nepotrebnih elementov. Take spremembe bi pomenile poseg v strukturo celotnega procesa zajema, zato se v tej različici nisem odločil za tovrstno rešitev, predstavlja pa možno izboljšavo v prihodnosti.

Zaradi rokov in velikih razlik med obstoječimi predlogami sem se, namesto da bi iskal skupno logiko vseh formatov, odločil za rešitev z ločenimi pravili za posamezne formate. Prednost take rešitve je enostavno prilagajanje posameznega formata, saj je mogoče spremeniti le funkcijo, ki se nanaša na ta format, brez vpliva na druge. Slabost pa je večje podvajanje kode med formati in posledično težje vzdrževanje, če bi bilo treba naenkrat spremeniti lastnost, ki je skupna vsem formatom. Med razvojem sem med formati poskusil najti skupne vzorce, ki bi omogočili bolj poenoteno implementacijo, vendar so bile razlike med predlogami prevelike.

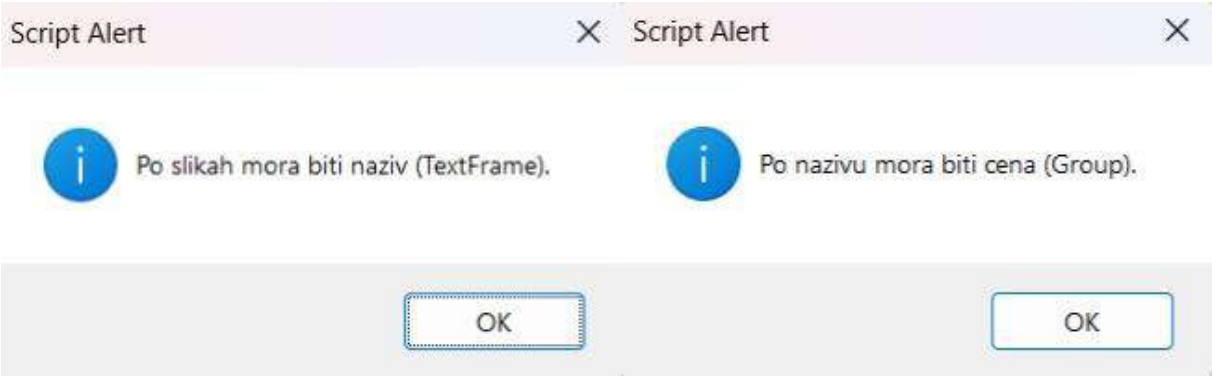
4.5 Validacija in preprečevanje napak

Ker sistem Oznacevalec zahteva, da uporabnik označi elemente artikla v točno določenem vrstnem redu, sem v modulu selectionValidation.jsx. implementiral mehanizem validacije selekcije. Brez validacije bi napačno označeni elementi povzročili napako med generiranjem ali pa bi se v sistem Oznazevalec kopirali napačni elementi, kar bi uporabnik opazil šele na koncu. Validacija se izvede takoj, ko uporabnik pritisne drugo bližnjico in še preden sistem zajame kateri koli element.

Validacija preveri selekcijo v točno določenem vrstnem redu, ki odraža pričakovano zaporedje elementov. Najprej preveri, ali selekcija ni prazna in ali so označeni vsaj trije obvezni elementi, to so slika, naziv in cena. Preveri tudi, ali je prvi element ustrezen za sliko, kar pomeni, da je Rectangle, Oval, GraphicFrame, Polygon ali Image. Ker je slik lahko več, sistem z zanko prešteje vse elemente istega tipa, dokler ne zazna elementa drugačnega tipa. Slikam mora slediti tekstovni okvir, ki predstavlja naziv artikla, za njim pa je Group, ki predstavlja ceno. Ikone po ceni so opcijske, zato zanje ni predvidena posebna validacija tipa; sistem preprosto vključi vse

preostale elemente v selekciji, ki sledijo ceni. To ne pomeni, da jih uporabnik ne more označiti napačno, saj bi lahko izbral elemente v pravilnem strukturnem vrstnem redu, ki pa na letaku pripadajo povsem drugemu artiklu. Validacija preverja samo strukturo selekcije, torej vrstni red in tip elementov, ne preverja pa, ali izbrani elementi dejansko pripadajo istemu artiklu.

Ko katera od validacij ne uspe, sistem prikaže sporočilo o napaki, ki uporabniku pove, kaj je narobe in kaj mora popraviti. Če za slikami ni označen tekstovni okvir, sistem prikaže sporočilo "Po slikah mora biti naziv (TextFrame)", če pa za nazivom ni označen Group, prikaže "Po nazivu mora biti cena (Group)". Taka sporočila uporabniku omogočajo, da napako hitro razume in jo odpravi brez nepotrebnega ugibanja. To je skladno z načelom preprečevanja napak, ki pravi, da mora sistem uporabnika obvestiti o napaki na način, ki mu omogoča takojšnje ukrepanje (Flaherty, 2018).



Slika 16: Primera sporočil o napaki ob neuspešni validaciji selekcije

Vir: (Lastni vir)

Validacija ima dvojno vlogo, in sicer uporabnik dobi jasno povratno informacijo, ko naredi napako, prav tako pa sistem nikoli ne poskuša zajeti elementov iz neveljavne selekcije, s čimer se preprečijo nepredvidene napake med generiranjem. Sporočila o napakah so oblikovana brez tehničnega žargona, tako da uporabniku točno povedo, kaj je narobe in kako nadaljevati. Ali so ta sporočila uporabniku brez tehničnega predznanja dovolj jasna, bo preverjeno s testom uporabniške izkušnje, kar je prikazano v poglavju 5.4 (prav tam).

5 TESTIRANJE

V tem poglavju so opisani postopki testiranja sistema Oznacevalec. Testiranje je potekalo med razvojem, kjer sem sproti po vsaki novi funkcionalnosti preveril, ali je delovanje pričakovano. Poglavje pokriva funkcionalno testiranje posameznih modulov, primerjavo ročnega in avtomatiziranega procesa ter test uporabniške izkušnje, s katerim sta preverjeni hipotezi H4 in H5.

5.1 Funkcionalno testiranje

Funkcionalno testiranje je zajemalo preizkus vseh ključnih scenarijev delovanja sistema Oznacevalec, od pravilnega generiranja vsakega formata predlog do robnih primerov. Vsak testni primer je bil izveden ročno v Adobe InDesignu, rezultat pa primerjan s pričakovanim obnašanjem sistema glede na implementacijo, opisano v poglavju 4.

Tabela 1: Rezultati funkcionalnega testiranja

Testni primer	Pričakovan rezultat	Uspeh	Opomba
format A4 pokončno – Akcija	označevalec generiran s pravilno postavljeno sliko, nazivom, ceno in ikonami	da	Označevalec je bil ustrezno generiran.
format A4 pokončno – Takoj ceneje	enako, manjši prostor za elemente zaradi glave in noge predloge	da	Označevalec je bil ustrezno generiran.
format A4 pokončno – Vrnemo na kartico	enako, manjši prostor za elemente zaradi glave in noge predloge	da	Označevalec je bil ustrezno generiran.
format A4 pokončno – Jager	označevalec generiran skladno s to predlogo	da	Označevalec je bil ustrezno generiran.

format A4 pokončno – Jager klub	označevalec generiran skladno s to predlogo	da	Označevalec je bil ustrezno generiran.
Format A4 pokončno -– Več je ceneje	označevalec generiran skladno s to predlogo	da	Označevalec je bil ustrezno generiran.
Format A4 pokončno – Vedno poceni	slikovni element na dnu cene pravilno sledi skaliranju	da	Označevalec je bil ustrezno generiran.
format A4 pokončno – Ugodno	zeleni kvadrat pri napisu pravilno sledi skaliranju	da	Označevalec je bil ustrezno generiran.
format A4 ležeče – Akcija	predloga Landscape, elementi se pravilno pozicionirajo	da	Označevalec je bil ustrezno generiran.
format A4 ležeče – Meso	predloga Landscape, elementi se pravilno pozicionirajo	da	Označevalec je bil ustrezno generiran.
format A3 postrežno	označevalec generiran skladno s to predlogo	da	Označevalec je bil ustrezno generiran.
Mali poličnik	Odpre se ExtraParamsDialog, slika kopirana na vseh 5 mest, besedilni okvirji se ustrezno zapolnijo	da	Označevalec je bil ustrezno generiran.
ena slika	slika proporcionalno skalirana in centrirana v slikaArea	da	Slika je lepo zapolnila dani prostor.

dve sliki	odpre se slikeRazporeditevDialog, izbrana razporeditev upoštevana	delno	Dialog se je prikazal. Slike so se ustrezno prilepile, vendar je pri nekaterih bilo potrebno tudi ročno popravljanje, da se je prostor lepše zapolnil.
tri slike	tri slike generirane brez dodatnega dialoga (vnaprej določena postavitev)	delno	Slike so se ustrezno prilepile, vendar je bila pri nekaterih označevalcih potrebna še ročna poprava pozicij, da so na strani lepše razvrstile.
slike z belo obrobo	slika lepo zapolni prostor tudi z belim robom	ne	Program ne prepozna, kdaj je prazni rob in kdaj se začne slika. Sliko je treba ročno povečati, da zapolni prostor.
brez ikon	označevalec generiran brez javljanja napake validacije.	da	Ne javi nobene napake, ikona ni obvezen element generiranja.
z več ikonami	vse ikone kopirane in pravilno skalirane/pozicionirane	da	
kratek naziv	naziv ostane pri privzeti velikosti pisave	da	Naziv je ustrezne velikosti, ne sega čez rob.
zelo dolg naziv	pisava naziva ustrezno pomanjšana znotraj mej FONT_SIZES (6-40pt)	da	Naziv se ustrezno poveča glede na dani prostor.
različne cene	pravilno skalirane cene	da	
prazna selekcija	opozorilo "Ni izbranih elementov", zajem prekinjen	da	Sistem javi, da je treba izbrati vsaj sliko, naziv in ceno.

napačen vrstni red selekcije	ustrezno sporočilo o napaki (npr. "Po slikah mora biti naziv (TextFrame)."	da	Sistem ustrezno javi napako.
manjkajoča predloga	TemplateLoader javi napako, brez generiranja tega artikla	da	Ta problem je težko doseči. Do manjkajoče predloge pride takrat, ko je odprt spustni seznam in uporabnik klikne izven seznama, tako da ni označena nobena predloga.

Vir: (Lastni vir)

Od 25 primerov jih je 22 testiranje opravilo popolnoma uspešno. Pri zajemu dveh ali treh slik se nekatere slike ustrezno skalirajo same, pri nekaterih pa je po generiranju potrebna manjša ročna korekcija pozicije. Pri primeru slike z izrazito belo obrobo sistem ni deloval pričakovano, saj ne loči vsebine slike od praznega prostora okoli nje. Rezultati potrjujejo, da sistem v veliki večini deluje zanesljivo, hkrati pa razkrivajo konkretno omejitev pri obravnavi slik z belim robom, ki jo podrobneje obravnavam v poglavju 4.3.

5.2 Testiranje ročnega in avtomatskega dela

Za preverjanje hipoteze H1 sem izmeril čas priprave enake serije označevalcev z ročnim postopkom in s sistemom Oznacevalec. Meritev sem izvedel trikrat za vseh trideset označevalcev, pri čemer sem pri obeh postopkih uporabil iste artikle in formate, s čimer sem zagotovil primerljivost rezultatov. Čas sem meril s štoparico. Pri avtomatiziranem postopku je izmerjeni čas zajemal celoten proces, od aktivacije sistema s Ctrl+Shift+1 do končnega pregleda vseh generiranih označevalcev, vključno z morebitnimi ročnimi popravki, na primer prilagoditvijo slik z belimi robovi, kjer sistem ni mogel samodejno ločiti vsebine od praznega prostora, kot je opisano v poglavju 4.3.

Prva izvedba je vključevala formate A4 pokončno – Takoj ceneje (7 kosov), A4 pokončno – Akcija (6), A4 pokončno – Jager (4), A4 pokončno – Vrnemo na kartico (4), A4 pokončno – Ugodno (4) in A4 pokončno – Vedno poceni (5). Druga izvedba je vključevala format Mali poličnik (7), A3 postrežno (7), A4 ležeče – Meso (7), A4 ležeče – Akcija (6) in A4 pokončno

– Več je ceneje (3). Tretja izvedba je vključevala vseh dvanajst formatov, večina po dva kosa, le pri formatu Mali poličnik in A3 postrežno je bilo pripravljenih pet kosov.

Tabela 2: Časi izvedbe ročnega in avtomatiziranega postopka

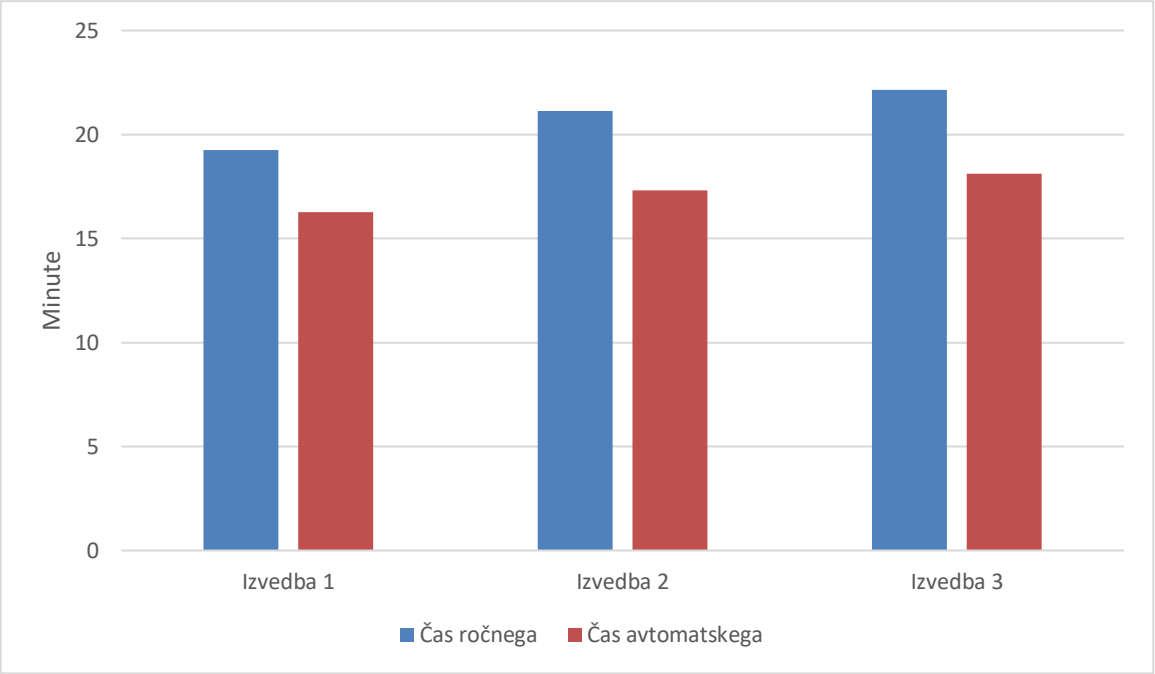
Izvedba	Formati	Št. označevalcev	Ročni postopek [min:s]	Avtomatizirani postopek [min:s]	Razlika [min:s]	Prihranek
1	Takoj ceneje, Akcija, Jager, Vrnemo na kartico, Ugodno, Vedno poceni	30	19:24	16:26	2:58	15,3 %
2	Mali poličnik, A3 postrežno, Meso, Akcija (ležeče), Več je ceneje	30	21:12	17:50	3:22	15,9 %
3	Vseh 12 formatov (večina 2 kosa, Mali poličnik in A3 postrežno pa 5)	30	22:14	18:11	4:03	18,2 %

Vir: (Lastni vir)

Tabela 3: Statistični povzetek meritev

	Ročni postopek	Avtomatiziran postopek
Skupen čas (3 izvedbe)	62 minut 50 sekund	52 minut 27 sekund
Povprečen čas na izvedbo	20 minut 57 sekund	17 minut 29 sekund
Povprečna razlika na izvedbo	3 minute 27 sekund	
Skupen prihranek časa	16,5 %	

Vir: (Lastni vir)



Grafikon 1: Primerjava časa priprave med ročnim in avtomatiziranim postopkom

Vir: (Lastni vir)

Sistem Oznacevalec je v vseh treh izvedbah čas priprave skrajšal, v povprečju za približno tri minute in pol na trideset označevalcev, kar predstavlja 16,5 odstotka prihranka časa. Prihranek je zmeren, kar je glede na to, da avtomatiziran postopek še vedno v letaku vključuje ročno označevanje elementov in občasne popravke slik, smiselno.

Pri drugi izvedbi, ki je vključevala format Mali poličnik, sem opazil, da je postopek zaradi ročnega vnosa besedila na vseh petih mestih na strani še posebej zamuden. Prav to je ena od nalog, ki jih sistem pri tem formatu v celoti avtomatizira.

Glavni razlog prihranka časa je v tem, da pri ročnem postopku uporabnik nenehno preklaplja med letakom in predlogo, saj mora vsakemu artiklu posebej odpreti predlogo, prekopirati vanjo elemente, ročno prilagoditi velikosti, nato pa se vrniti po naslednji artikel. Pri sistemu Oznacevalec uporabnik ves čas ostane v izvornem dokumentu z letakom in zgolj označuje elemente, medtem ko preklapljanje med predlogami in prilagajanje velikosti prevzame sistem.

Izmerjeni rezultati kažejo, da sistem Oznacevalec v vseh treh izvedbah v primerjavi z ročnim postopkom skrajša čas priprave serije označevalcev. Prihranek je zmeren in ne odpravi ročnega dela v celoti, saj mora uporabnik elemente še vedno ročno označiti, občasno pa je potrebna tudi ročna korekcija. Pomen tega dejstva za H1 je povzet v sklepu.

5.3 Uporabniško testiranje

Test uporabniške izkušnje preverja hipotezi H4 in H5. H4 trdi, da je sistem Oznacevalec mogoče dokumentirati z uporabniškimi navodili, tako da je razumljiv tudi uporabniku brez predhodnega tehničnega znanja, H5 pa predpostavlja, da vgrajena validacija selekcije in jasna sporočila o napakah zmanjšujejo možnost uporabniških napak. Testiranje sem izvedel z dvema skupinama udeležencev: nestrokovnjaki, ki Adobe InDesigna ne poznajo ali imajo o njem samo osnovno znanje, in grafiki, ki Adobe InDesign pri svojem delu redno uporabljajo. Skladno z metodologijo, opredeljeno v dispoziciji diplomskega dela, je bila skupina nestrokovnjakov sestavljena iz petih oseb, kot primerjalno skupino pa sem dodal tri grafike, s čimer sem lahko preveril, ali se rezultati nestrokovnjakov razlikujejo od rezultatov izkušenih uporabnikov Adobe InDesigna.

Testiranje z nestrokovnjaki je bilo izbrano zato, ker hipoteza H4 preverja, ali lahko uporabnik sistem uporablja brez predhodnega znanja programa Adobe InDesign in zgolj na podlagi pisnih navodil. Izkušen grafik sicer prav tako sledi navodilom in korakom sistema, saj gre za posebne bližnjice in specifičen vrstni red označevanja, ki jih vnaprej ne pozna, vendar razume osnovne pojme in delovanje Adobe InDesigna, na primer razliko med posameznimi tipi elementov. V kolikor bi testirali zgolj poznavalce Adobe InDesigna, bi lahko dobili rezultate, iz katerih bi bilo težko razbrati, ali so navodila razumljiva, zlasti osebam, ki ne poznajo osnovnih pojmov. Prav tako se lahko zgodi, da bi označevalce pri podjetju Jagros, d. o. o., pripravljali novi zaposleni ali študenti na praksi, ki še ne obvladajo dela z Adobe InDesignom. Zato je priročno, da je sistem uporaben tudi brez predznanja in da lahko uporabnik ob delu z njim spozna osnove delovanja skript.

Testiranje je potekalo individualno in v živo na mojem računalniku. Vsak udeleženec je najprej prejel v branje uporabniška navodila sistema Oznacevalec, nato pa nalogo, da s pomočjo teh navodil in sistema generira označevalec za določen testni artikel. Med izvedbo sem beležil, ali je udeleženec nalogo uspešno zaključil, koliko časa je zanjo porabil, koliko napak je naredil, pri katerih korakih je potreboval pomoč in ali je razumel prikazana sporočila o napakah. Po zaključku naloge je vsak udeleženec izpolnil anketni vprašalnik o razumljivosti navodil, jasnosti sporočil o napakah in splošni oceni sistema.

Podatki iz testiranja so bili obravnavani anonimno. Vsak udeleženec je pred začetkom naloge izžrebal svojo oznako, ki jo je uporabil pri izpolnjevanju ankete. Udeležence sem med izvedbo naloge opazoval v živo, zato njihova identiteta zame ni bila neznana. Kljub temu je oznaka

zagotovila, da zapisanih opažanj in anketnih odgovorov v končni analizi ni bilo mogoče povezati s konkretno osebo.

Tabela 4: Opažanja med uporabniškim testiranjem

Udeleženec	Skupina	Čas izvedbe	Št. napak	Opombe
uporabnik 1	nestrokovnjak	pribl. 5 min	2	ne pritisne takoj bližnjice, nehote označi napačen prvi element
uporabnik 2	nestrokovnjak	pribl. 4 min	1	poskusi označevati več artiklov naenkrat, po ponovnem branju navodil popravi
uporabnik 3	nestrokovnjak	pribl. 6 min	2	sprva ne pritisne gumba Shift, po nesreči označi nepotrebno sliko, po gumbu Prekliči pozabi ponovno aktivirati sistem
uporabnik 4	nestrokovnjak	pribl. 3 min	0	brez težav
uporabnik 5	nestrokovnjak	pribl. 6 min	2	negotovost pri selekciji elementov, pozabi ponovno aktivirati sistem po gumbu Prekliči
uporabnik 1	grafik	pribl. 2–3 min	0	brez težav
uporabnik 2	grafik	pribl. 3 min	1	nehote odznači naziv po sporočilu, takoj popravi
uporabnik 3	grafik	pribl. 2 min	0	brez težav

Vir: (Lastni vir)

Vseh pet nestrokovnjakov in vsi trije grafiki so nalogo uspešno zaključili. Grafiki so bili v povprečju občutno hitrejši (2 do 4 minute) in so naredili manj napak (0 do 1) kot nestrokovnjaki (4 do 7 minut, 0 do 3 napake). Pri slednjih so se ponovile tri vrste zatikanj: negotovost, kdaj je element v InDesignu dejansko označen, poskus označevanja elementov za več artiklov hkrati namesto enega in pozabljena ponovna aktivacija sistema po kliku gumba Prekini. Prva težava je verjetno povezana z osnovnim poznavanjem dela v InDesignu, ki ga imajo grafiki, nestrokovnjaki pa ne. Drugi dve težavi pa sta bolj povezani s tem, da so grafiki zaradi vsakodnevnega dela po strukturiranih postopkih dosledneje sledili točno določenemu vrstnemu redu korakov sistema.

Tabela 5: Rezultati ankete – Blok A (navodila in samostojnost)

Oznaka	Poznavanje InDesigna	Navodila jasna (1–5)	Navodila strukt. (1–5)	Znal začeti (1–5)	Št. odprt navodil	Samost. uporaba (1–5)	Primerno brez znanja (1–5)
L1	brez predhodnih izkušenj uporabe	5	5	5	3–5x	5	4
L2	brez predhodnih izkušenj uporabe	4	4	4	3–5x	4	4
L3	osnovno poznavanje	5	4	5	3–5x	5	5
L4	znano, a brez predhodnih izkušenj uporabe	5	5	3	3–5x	5	5
L5	brez predhodnih izkušenj uporabe	5	5	4	3–5x	5	4
G1	redna uporaba	5	5	5	2–3x	5	5
G2	redna uporaba	4	5	5	3–5x	5	5
G3	redna uporaba	4	5	5	2–3x	5	5

Vir: (Lastni vir)

Pri odprtih vprašanjih so se v navodilih pri nestrokovnjakih ponovili trije vzorci: negotovost, kdaj je element v InDesignu dejansko označen, potreba po jasnejšem navodilu, da se naenkrat zajema en artikel, ter predlog navodila za dodajanje več slik, ki so ga podali dva od petih nestrokovnjakov in eden od treh grafikov.

Vseh pet nestrokovnjakov je brez predhodnega znanja InDesigna nalogo uspešno zaključilo izključno na podlagi pisnih navodil, ocene razumljivosti le-teh in njihove samostojne uporabe pa so bile v povprečju visoke (4,2 do 4,8 od možnih 5). Grafiki so bili hitrejši in so naredili manj napak, kar je glede na njihove predhodne izkušnje z InDesignom pričakovano. Kljub temu razlika med obema skupinama ne kaže, da je naloga brez predhodnih izkušenj z InDesignom neizvedljiva, kar je bil tudi osrednji namen testiranja obeh skupin. Pomen teh rezultatov za hipotezo H4 je povzet v sklepu. Analiza sporočil o napakah, ki jo zajema H5, sledi v poglavju 5.4.

5.4 Preverjanje validacije in sporočil o napakah

To poglavje preverja hipotezo H5, ki trdi, da vgrajena validacija selekcije in jasna sporočila o napakah zmanjšujejo možnost uporabniških napak skladno z načeli preprečevanja napak (Flaherty, 2018). Preverjanje temelji na dveh virih: v analizi mehanizma validacije v modulu selectionValidation.jsx, opisanem v poglavju 4.5, ter v odgovorih udeležencev testiranja na vprašanja o razumljivosti sporočil o napakah iz poglavja 5.3 (blok ankete B).

Flaherty (2018) navaja, da mora sporočilo o napaki uporabniku brez tehničnega žargona povedati, kaj je narobe in kako naj nadaljuje. Sistem Oznacevalec ob neuspešni validaciji selekcije prikaže eno izmed naslednjih sporočil:

Tabela 6: Sporočila o napakah ob neuspešni validaciji selekcije

Sporočilo	Čas prikaza	Sledenje načelom preprečevanja napak
"Ni izbranih elementov"	Selekcija je prazna.	da
"Izberi vsaj 3 elemente"	Selekcija ima manj kot 3 elemente.	da
"Prvi element mora biti slika"	Prvi element ne ustreza slikovnemu tipu.	da

"Po slikah mora biti naziv (TextFrame)"	Za slikami ni označen tekstovni okvir.	da
"Po nazivu mora biti cena (Group)"	Za nazivom ni označen Group.	da
"Najprej aktiviraj sistem Oznacevalec s Ctrl+Shift+1"	Sistem pred zajemom ni bil aktiviran.	da
"Format ni prepoznan"	Izbran format ni najden v config.jsx.	ne
"Predloga ne obstaja"	Datoteka predloge ni najdena na disku.	ne

Vir: (Lastni vir)

Iz tabele je razvidno, da večina sporočil sledi načelom preprečevanja napak, saj poimenujejo konkreten manjkajoč ali napačen element in navajajo pričakovani tip (npr. "naziv (TextFrame)"). Dve sporočili, "Format ni prepoznan" in "Predloga ne obstaja", pa uporabnika o težavi le obvestita, in sicer brez navodila, kako naj jo odpravi, pri čemer gre bolj za tehnično napako (manjkajoča datoteka na disku). Zato je uporabnik v tem primeru odvisen od pomoči administratorja sistema.

Poleg analize sporočil v kodi je H5 preverjena tudi prek ankete – blok B, opisanega v poglavju 5.3. Ker so lahko na ti dve vprašanji smiselno odgovorili le udeleženci, ki so med izvedbo dejansko naleteli na sporočilo o napaki, sem v analizo vključil samo njih. To so bili štirje od petih nestrokovnjakov (izjema je bil udeleženec, ki napake ni imel) in eden od treh grafikov (druga dva napake nista imela).

Tabela 7: Rezultati ankete – blok B (sporočila o napakah)

Oznaka	Sporočilo o pojasnitvi napake (1–5)	Uspešnost nadaljevanja (1–5)	Najkoristnejše sporočilo	Nejasno sporočilo
L1	5	5	ponovna aktivacija sistema	nobeno
L2	5	4	napačna označitev elementa	nobeno
L3	5	5	prvo sporočilo (naziv pred ceno)	nobeno
L5	5	5	ponovna aktivacija sistema	nobeno
G2	5	5	drugi element je naziv	nobeno

Vir: (Lastni vir)

Ker je na to vprašanje odgovarjal zgolj en grafik, to predstavlja premajhen vzorec za posplošitev, kar je treba upoštevati kot omejitev tega dela testiranja.

Analiza sporočil v kodi je pokazala, da večina sporočil sledi načelom preprečevanja napak (Flaherty, 2018), z izjemo dveh tehničnih sporočil (manjkajoča predloga in neznan format), ki ne povesta, kako napako odpraviti. To se ujema z opazovanjem in anketo: vsi udeleženci, ki so naleteli na sporočilo o napaki, so iz njega razbrali, kaj morajo popraviti, tako da so nalogo nadaljevali brez dodatne pomoči. Ocene jasnosti sporočil so bile visoke (5 pri nestrokovnjakih), pri grafikih pa je bil vzorec premajhen za posplošitev. Kaj ti rezultati pomenijo za hipotezo H5, je povzeto v sklepu.

6 TEHNIČNA DOKUMENTACIJA

Šesto poglavje vsebuje tehnično dokumentacijo sistema Oznacevalec, ki je namenjena lažjemu razumevanju delovanja sistema in morebitnemu nadaljnjemu vzdrževanju ali nadgradnji. V poglavju 6.1 je opisana vloga vsakega posameznega modula znotraj arhitekture, predstavljene v poglavju 3.1. V poglavju 6.2 so s pomočjo UML-diagramov prikazani še podatkovni model, potek podatkov med moduli in stanje sistema med delovanjem.

6.1 Opis modulov

Sistem Oznacevalec je sestavljen iz dvajsetih datotek, razporejenih v mape glede na funkcionalnost, kar opisuje poglavje 3.1. V nadaljevanju je opisana vloga vsakega modula.

Koren projekta

Modul config.jsx je centralna datoteka z vsemi nastavitvami sistema. Za pozicioniranje slik na petih poličnikih ene strani vsebuje objekt FORMATS z nazivi vseh dvanajstih formatov za spustni seznam v formatDialogu, objekt TEMPLATES s potmi do predlog .indd, objekt FONT_SIZES z mejama velikosti pisave naziva, objekt POLICNIK in tri sezname koordinat (POLICNIK_ENA_SLIKA, POLICNIK_DVE_SLIKI, POLICNIK_TRI_SLIKE).

Modul main.jsx je bootstrap datoteka, ki z ukazom #include v pravilnem vrstnem redu naloži vse ostale module. Vključujeta ga obe vstopni skripti, startCapture.jsx in confirmCapture.jsx.

Modul startCapture.jsx je prva vstopna točka sistema, sprožena s Ctrl+Shift+1. Obnovi morebitno prejšnje stanje iz state.json, sistem označi kot aktiven (isRunning = true), stanje shrani nazaj v datoteko in uporabnika z opozorilnim oknom opozori na pravilen vrstni red označevanja elementov.

Modul confirmCapture.jsx je druga vstopna točka, sprožena s Ctrl+Shift+2. Obnovi stanje in preveri, ali je sistem aktiven, nato pokliče SelectionValidation.validate(). Če je selekcija veljavna, pokliče CaptureArticle.capture(), odpre formatDialog in po potrebi SlikeRazporeditevDialog, pri formatu Mali poličnik nastavi naziv in ceno na null, artikel doda v AppState in vpraša, ali uporabnik želi zajeti še en artikel. Ob zaključku odpre listingDialog in glede na izbiro uporabnika pokliče MarkerGenerator.generate() ali samo počisti stanje.

Models/

Modul articleModel.jsx vsebuje funkcijo createArticle(), ki ustvari prazen artikel objekta z vsemi polji, potrebnimi za en artikel: slikami, nazivom, ceno, ikonami, formatom, dodatnimi parametri za Mali poličnik in metapodatki (id, timestamp in dokument).

Modul appState.jsx vsebuje objekt AppState, ki hrani seznam vseh zajetih artiklov in zastavico isRunning. Poleg osnovnih funkcij dodajanja, odstranjevanja in posodabljanja artiklov vsebuje tudi saveState() in loadState(), ki prek state.json skrbita za shranjevanje in obnovo stanj, kot je opisano v poglavju 4.2, ter reset(), ki ob zaključku generiranja obnovi originalna imena elementov in počisti stanje.

Modul jsonHelper.jsx vsebuje lastno implementacijo JSON serializacije, ker je v ExtendScriptu vgrajeni JSON objekt odsoten. Funkcija stringify() rekurzivno pretvori vrednosti katere koli vrste (null, string, number, boolean, array ali object) v niz JSON. Funkcija parse() niz pretvori nazaj v objekt z uporabo new Function() namesto eval(), ki v ExtendScriptu povzroča napake.

Modul frameUtils.jsx vsebuje pomožne funkcije za delo z okvirji InDesign: getBounds() in setBounds() za branje in nastavljanje geometricBounds, moveFrame() za premik okvirja po ohranitvi velikosti ter getWidth() in getHeight() za izračun dimenzij.

Capture/

Modul captureArticle.jsx prebere app.selection in zapolni articleModel objekt. Slike pobira, dokler ne naleti na prvi element, ki ni slikovnega tipa. Nato sledita naziv in cena, preostanek selekcije pa se shrani kot ikone. Vsakemu okvirju ob tem dodeli unikatno ime, originalno ime pa shrani za poznejšo obnovitev.

Modul selectionValidation.jsx preveri selekcijo, preden se zajamejo elementi. Preveri, da selekcija ni prazna in da vsebuje vsaj tri elemente, da je prvi element slikovnega tipa, da slikam sledi besedilni okvir za naziv, temu pa Group za ceno. Ob neuspešni izbiri elementov se prikaže sporočilo, ki pove, kaj manjka.

Ui/

Modul formatDialog.jsx prikaže spustni seznam vseh dvanajstih formatov, iz katerega uporabnik izbere predlogo za trenutni artikel. Če izbere format Mali poličnik, samodejno odpre ExtraParamsDialog.

Modul extraParamsDialog.jsx prikaže dodatna polja formata Mali poličnik: popust, opis popusta, znamko in datum veljavnosti. Uporabnik jih vnese ročno, saj ta format teh podatkov ne pridobi iz letaka.

Modul slikeRazporeditevDialog.jsx se prikaže, če sta zajeti točno dve sliki in format ni Mali poličnik. Uporabnik izbere, ali naj bosta sliki postavljeni ena zraven druge ali ena nad drugo.

Modul listingDialog.jsx prikaže seznam vseh zajetih artiklov v listboxu, z možnostjo odstranitve posameznega artikla. Uporabnik lahko izbere artikel v seznamu in s klikom na gumb Odstrani izbrani artikel tudi odstrani. Gumb Generiraj sproži generiranje, gumb Prekini pa proces prekine brez generiranja.

Templates/

Modul templateLoader.jsx glede na izbrani format pokliče ustrezno pot v TEMPLATES. Iz config.jsx preveri, ali datoteka obstaja in jo odpre v InDesignu. Če predloga ne obstaja ali format ni prepoznan, uporabnika opozori in prekine generiranje artikla.

Generator/

Modul scalingEngine.jsx skalira kopirane elemente na prostor, rezerviran v predlogi. Slike z ohranjanjem razmerja stranic prilagodi s funkcijo fit(), ceno (Group) s funkcijo resize() ter z ročnim skaliranjem pointSize in baselineShift besedila znotraj skupine. Pri formatih Vedno poceni in Ugodno dodatno prilagodi položaj elementov, ki jih funkcija resize() ne premakne samodejno. Ikone skalira glede na njihov tip (Group z ali brez gnezdenega Group-a, Rectangle, GraphicFrame ali drugi tip).

Modul textAdjustments.jsx prilagodi velikost fonta naziva, tako da ga postopoma povečuje, dokler se besedilo ne preliva ali dokler ne doseže FONT_SIZES.MAX_SIZE oziroma ga zmanjšuje, če je besedilo predolgo znotraj meje FONT_SIZES.MIN_SIZE.

Modul layoutRules.jsx vsebuje funkcijo position(), ki glede na format pokliče ustrezno podfunkcijo z njegovimi fiksno kodiranimi koordinatami za slikaArea, cenaArea in ikonaArea. Za format Mali poličnik position() pokliče positionMaliPolicnik(), ki v config.jsx uporablja koordinate iz POLICNIK_ENA_SLIKA, POLICNIK_DVE_SLIKI ali POLICNIK_TRI_SLIKE.

Modul markerGenerator.jsx je osrednji orkestrator generiranja. Funkcija generate() artikle najprej združi po formatu, nato za vsako skupino enkrat odpre ustrezno predlogo, doda toliko

strani, kolikor je artiklov te skupine, elemente iz letaka kopira na vsako stran (copyElements), pokliče LayoutRules.position(), na koncu pa dokument preda ExportHandlerju.

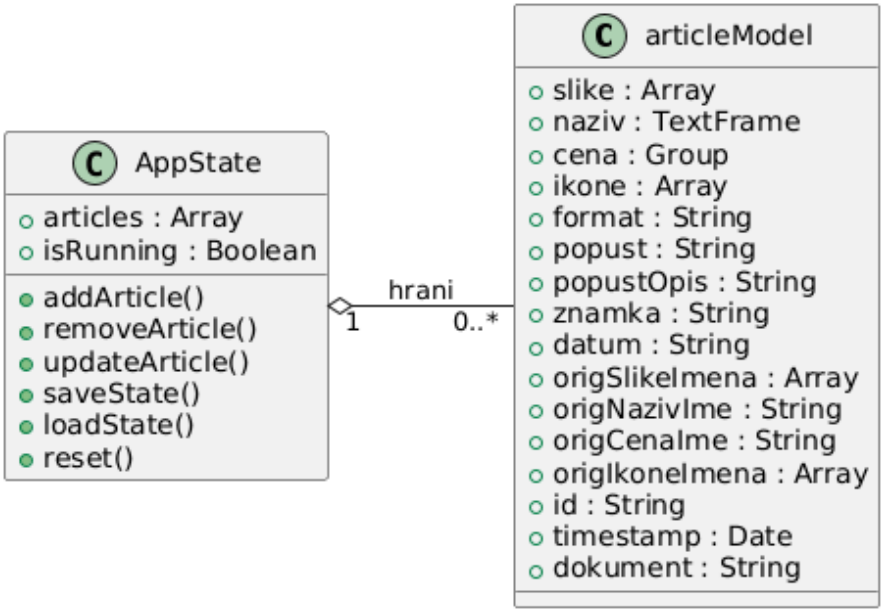
Export/

Modul exportHandler.jsx shrani izpolnjeno predlogo kot datoteko .indd v mapo izvoz/. Če ime datoteke na disku že obstaja ali je dokument z istim imenom že odprt v InDesignu, v izogib prepisovanju v oklepaju doda zaporedno številko.

6.2 UML-diagrami

Za lažje razumevanje strukture in delovanja sistema Oznacevalec so v tem poglavju predstavljeni trije UML-diagrami: diagram razredov, ki prikazuje podatkovni model sistema, diagram zaporedja, ki med moduli prikazuje potek podatkov med enim ciklom zajema in generiranja, ter diagram stanj, ki prikazuje spreminjanje stanja sistema med posamezno sejo.

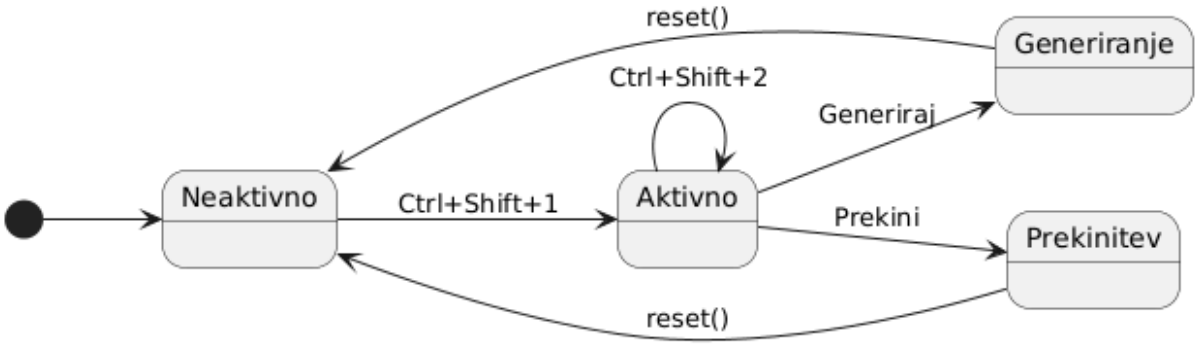
Diagram razredov prikazuje strukturo objekta articleModel in AppState, kar je opisano v poglavjih 3.2 in 6.1. Prav tako so prikazana polja objekta articleModel (slike, naziv, cena, ikone, format, popust, popustOpis, znamka, datum, origSlikeImena, origNazivIme, origCenaIme, origIkoneImena, id, timestamp in dokument) ter metode objekta AppState (addArticle, removeArticle, updateArticle, saveState, loadState in reset) skupaj z njegovima lastnostma articles in isRunning.



Slika 17: Diagram razredov articleModel in AppState

Vir: (Lastni vir)

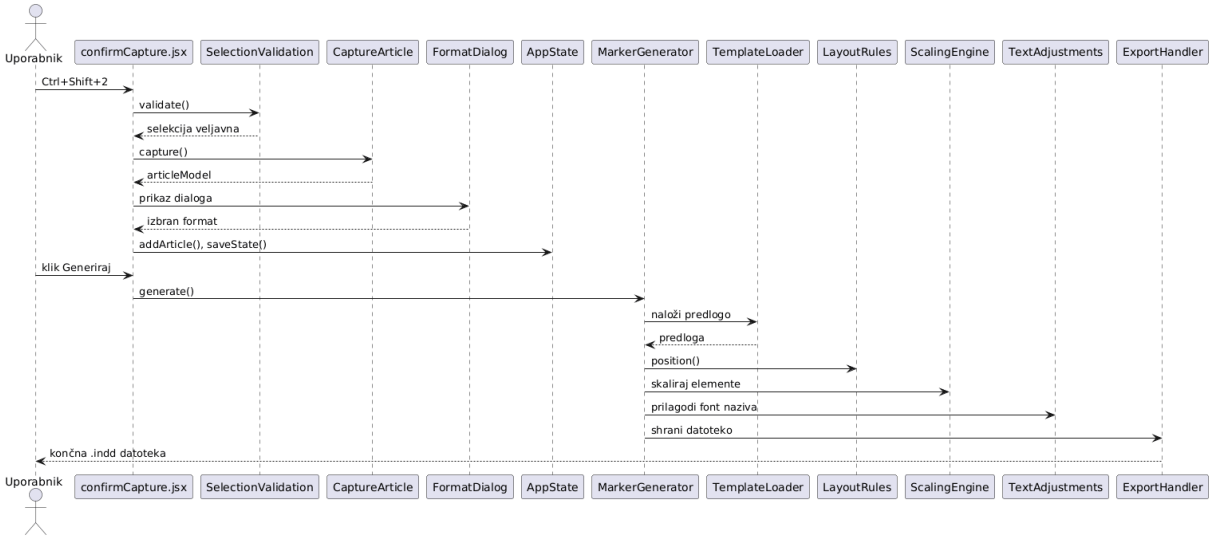
Diagram zaporedja prikazuje potek enega cikla zajema in generiranja artikla skozi module sistema, in to od trenutka, ko uporabnik pritisne Ctrl+Shift+2 in gre potek skozi selectionValidation, CaptureArticle, FormatDialog in AppState, do trenutka, ko MarkerGenerator prek TemplateLoader, LayoutRules, ScalingEngine, TextAdjustments in ExportHandler ustvari končno datoteko.



Slika 18: Diagram stanj sistema Oznacevalec

Vir: (Lastni vir)

Diagram stanj prikazuje spreminjanje vrednosti isRunning in vsebine AppState.articles skozi eno sejo, in sicer od neaktivnega stanja pred pritiskom Ctrl+Shift+1 prek aktivnega stanja med zajemom več artiklov do ponastavitve stanja po zaključenem generiranju ali prekinitvi.



Slika 19: Diagram zaporedja zajema in generiranja artikla

Vir: (Lastni vir)

7 UPORABNIŠKA NAVODILA

V tem poglavju so predstavljena uporabniška navodila za sistem Oznacevalec, ki so bila za preverjanje hipoteze H4 uporabljena tudi pri testu uporabniške izkušnje, kar je opisano v poglavju 5.3,.

7.1 Uvod

Oznacevalec je skripta Adobe InDesigna, ki iz elementov letaka samodejno generira promocijske označevalce cen. Skripta samodejno kopira elemente v predlogo. Uporabnik jih označi v letaku, izbere format predloge, skripta Oznacevalec pa pripravi končno datoteko.

Artikli istega formata se združijo v eno datoteko z več stranmi, kar olajša nadaljnje urejanje in tiskanje.

7.2 Namestitev

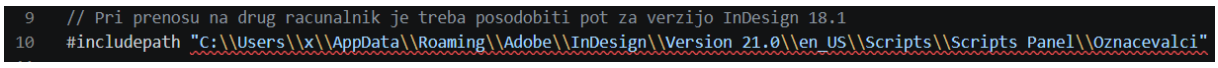
Celotno mapo Oznacevalci je treba skopirati v mapo Scripts Panel programa Adobe InDesign:

C:\Users[uporabnik]\AppData\Roaming\Adobe\InDesign\Version [X]\en_US\Scripts\ScriptsPanel\

Mapa mora ostati nedotaknjena. Vse podmape in datoteke morajo biti na svojem mestu.

Ob prvi namestitvi je treba v datotekah startCapture.jsx, confirmCapture.jsx in main.jsx ročno posodobiti pot do mape Oznacevalci. Na vrhu vsake datoteke je vrstica #includepath, ki se spremeni tako, da bo kazala na pot do mape Oznacevalci.

Pot do mape se nahaja v InDesignu pod panelom Scripts, kjer se z desnim klikom na mapo Oznacevalci izbere Reveal in Explorer. V oknu, ki se odpre, je treba klikniti na naslovno vrstico in kopirati pot. Nato se ta pot v vseh treh datotekah zamenja v vrstici #includepath. Ob tem je treba paziti, da so v poti vse poševnice obrnjene in dvojne (\\ in ne //).



Slika 20: Slika vrstice #includepath

Vir: (Lastni vir)

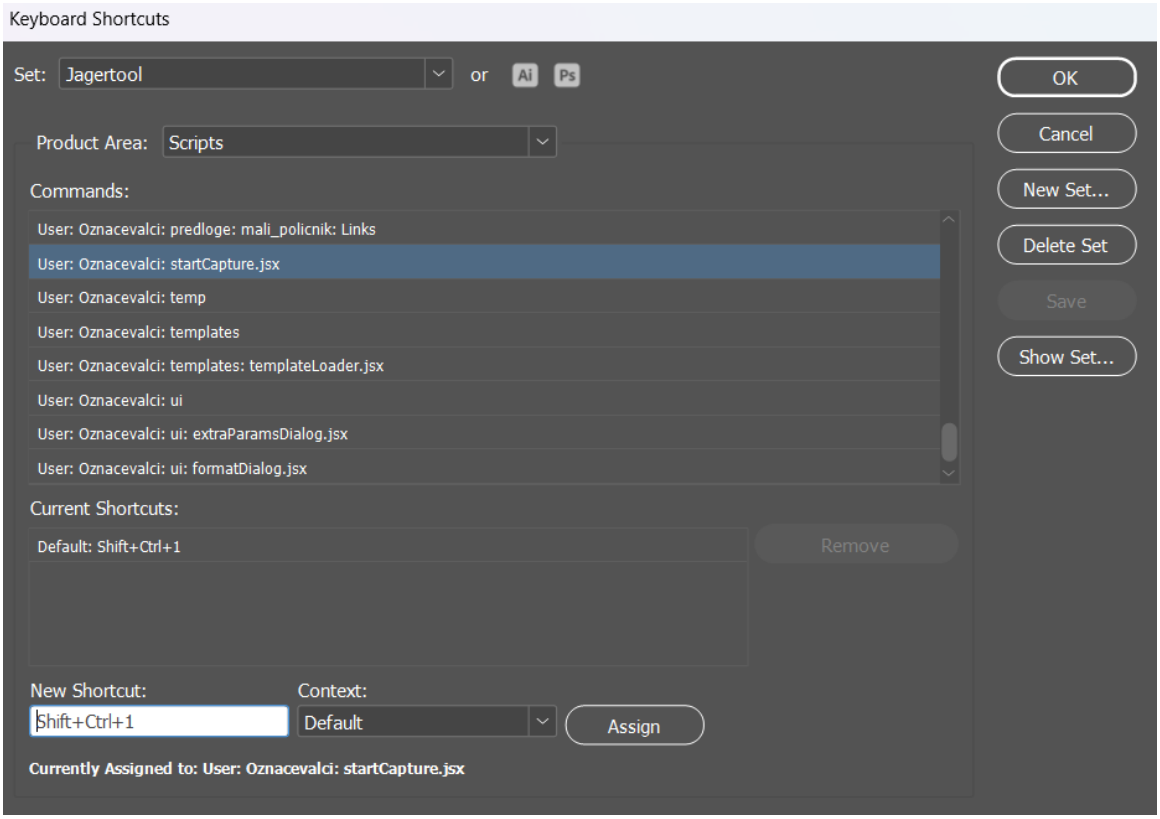
7.2.1 Nastavitev bližnjic

Bližnjici se nastavita v InDesignu pod Edit > Keyboard Shortcuts. Priporočene bližnjice:

- startCapture.jsx – Ctrl+Shift+1
- confirmCapture.jsx – Ctrl+Shift+2

Za nastavitev bližnjic je treba slediti naslednjim korakom:

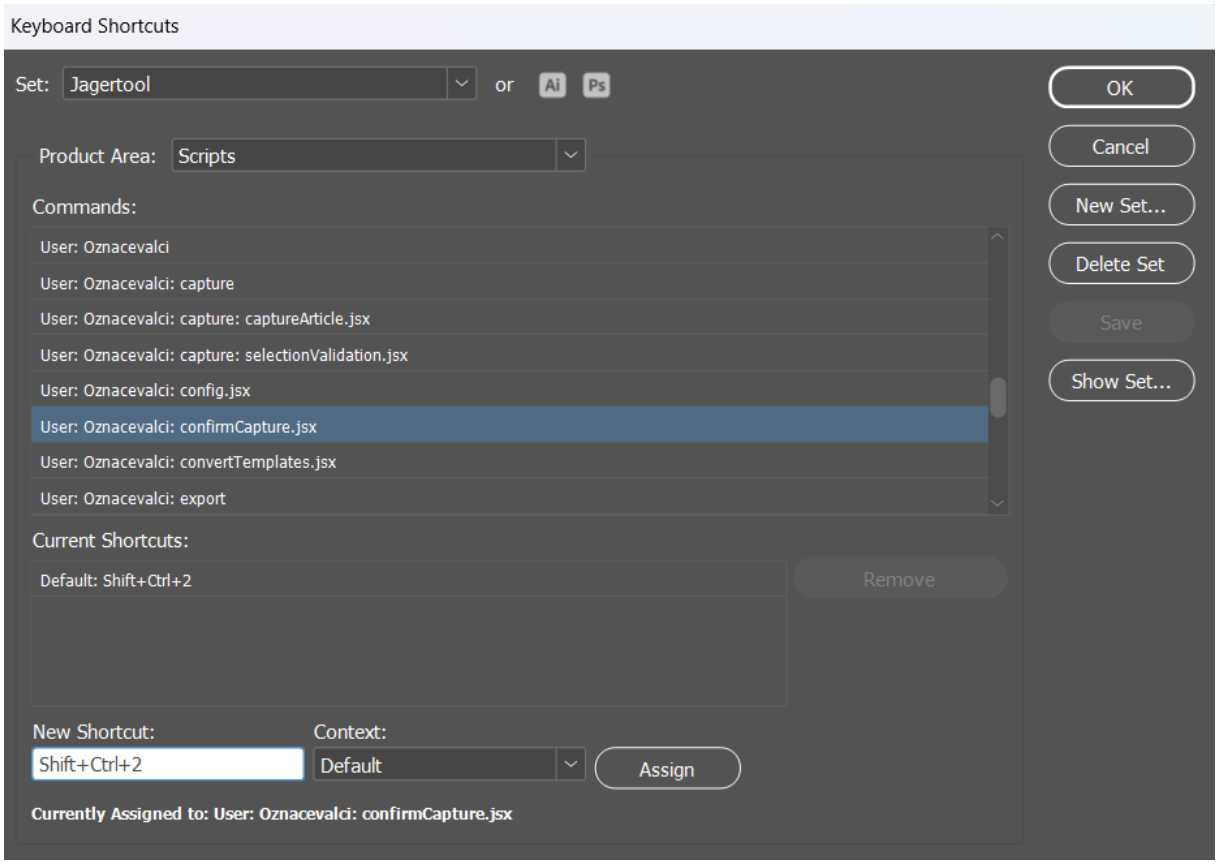
1. Odpremo Edit > Keyboard Shortcuts.
2. V polju Product area izberemo Scripts.
3. Poiščemo User:Oznacevalci:startCapture.jsx in ji dodelimo Ctrl+Shift+1.



Slika 21: Dodelitev bližnjice Ctrl+Shift+1 skripti startCapture.jsx

Vir: (Lastni vir)

4. Poiščemo User:Oznacevalci:confirmCapture.jsx in ji dodelimo Ctrl+Shift+2.



Slika 22: Dodelitev bližnjice Ctrl+Shift+2 skripti confirmCapture.jsx

Vir: (Lastni vir)

5. Potrdimo s klikom na gumb OK.

Opomba: Bližnjici se lahko glede na želje spremenita. Če se spremenijo, je treba v datoteki startCapture.jsx v pogovornem oknu, kjer piše "Ko zakljucis, pritisni Ctrl+Shift+2", posodobiti besedilo. Spremeniti je treba zgolj besedilo, ne same kode.

7.3 Zajem artikla

Korak 1: Aktivacija

Preden se označujejo elementi, je treba pritisniti Ctrl+Shift+1. Pojavilo se bo obvestilo z navodili za vrstni red označevanja.

Korak 2: Označevanje elementov

V letaku se označijo elementi v naslednjem vrstnem redu:

1. slika (lahko jih je več, največ 3),

2. naziv (besedilno polje z imenom artikla),

3. cena (skupina elementov s ceno) in

4. ikone (opcijsko, če so prisotne).

Opomba: Elementi se označijo z držanjem tipke Shift in klikanjem po vrstnem redu. Upoštevanje vrstnega reda je obvezno, saj skripta prebere elemente točno tako, kot so bili označeni.

Pomembno: Elementi se označujejo tako, da se naenkrat označi en artikel. Ko se zaključi en artikel in pritisne Ctrl+Shift+2, sistem vpraša, ali naj doda naslednji artikel in šele takrat se lahko začne označevanje elementov naslednjega artikla.

Element je označen, ko se okoli njega prikaže tanka modra obroba, ki ostane vidna, dokler se ga ne odznači ali se klikne drugam.



Slika 23: Primer označenih elementov artikla v letaku

Vir: (Lastni vir)

Korak 3: Potrditev

Ko so elementi označeni, s pritiskom na Ctrl+Shift+2 skripta elemente preveri in odpre dialog za izbiro formata.

7.4 Izbira formata

Po pritisku Ctrl+Shift+2 se odpre dialog z izborom predloge. Na voljo so naslednji formati:

- A4 pokončno: formati Akcija, Takoj ceneje, Vrnemo na kartico, Jager, Jager klub, Več je ceneje, Vedno poceni in Ugodno;
- A4 ležeče: formati Akcija in Meso;
- A3 postrežno;
- format Mali poličnik.

Format prepoznamo po cenovki artikla, ki se glede na vrsto akcije razlikuje, zato lahko na podlagi njenega videza določimo, kateri format je treba izbrati. Če cena vsebuje napis "Takoj ceneje", izberemo format Takoj ceneje in podobno za vse ostale formate.



Slika 24: Primeri različnih cenovk glede na format

Vir: (Lastni vir)

Izberemo format in kliknemo gumb Potrdi.

7.4.1 Mali poličnik

Ob izbiri formata Mali poličnik se odpre dodatno pogovorno okno (slika v poglavju 3.3), kjer je treba vnesti:

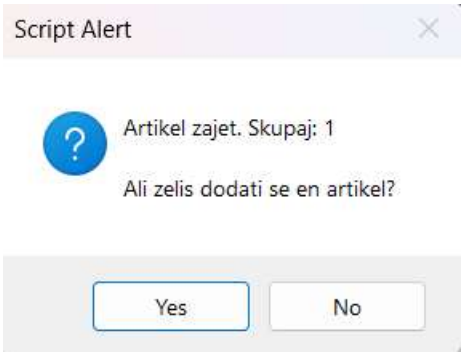
- odstotek popusta (samo številka, brez znaka %),
- opis popusta (na katere izdelke se nanaša),
- znamko in

• datum veljavnosti (npr. 15. 3.–21. 3. 2026).

Opomba: Pri formatu Mali poličnik je treba označiti sliko, naziv in ceno, saj sistem te tri elemente zahteva za validacijo selekcije. Naziv in cena se pri generiranju ne uporabita, sta pa obvezna za potrditev zajema.

7.5 Dodajanje več artiklov

Po zajemu prvega artikla skripta vpraša, ali želimo dodati še en artikel. Če kliknemo pritrdilni gumb, se sistem samodejno pripravi za naslednji zajem.



Slika 25: Poziv za dodajanje novega artikla

Vir: (Lastni vir)

Sledi označitev elementa naslednjega artikla in pritisk Ctrl+Shift+2. Postopek je treba ponoviti za vsak artikel. Skupno število zajetih artiklov je prikazano v pogovornem oknu.

Opomba: Ko se začne označevanje naslednjih artiklov, je treba paziti, da elementi prejšnjega artikla niso več označeni. Zato je pred označevanjem naslednjega artikla treba klikniti izven strani, s čimer se odznačijo elementi prejšnjega artikla.

7.6 Pregled in generiranje

Ko so vsi artikli zajeti, je treba pri vprašanju o dodajanju novega artikla klikniti NE. Odpre se pregled vseh zajetih artiklov (slika v poglavju 3.3).

V pregledu lahko:

• vidimo vse zajete artikle z njihovimi formati;

• odstranimo posamezen artikel, tako da ga izberemo v seznamu in kliknemo gumb Odstrani;

• začnemo generiranje z gumbom Generiraj;

• prekinemo brez generiranja z gumbom Prekini.

Ko kliknemo gumb Generiraj, skripta samodejno pripravi vse označevalce. Artikli istega formata se združijo v eno datoteko z več stranmi.

7.7 Rezultat

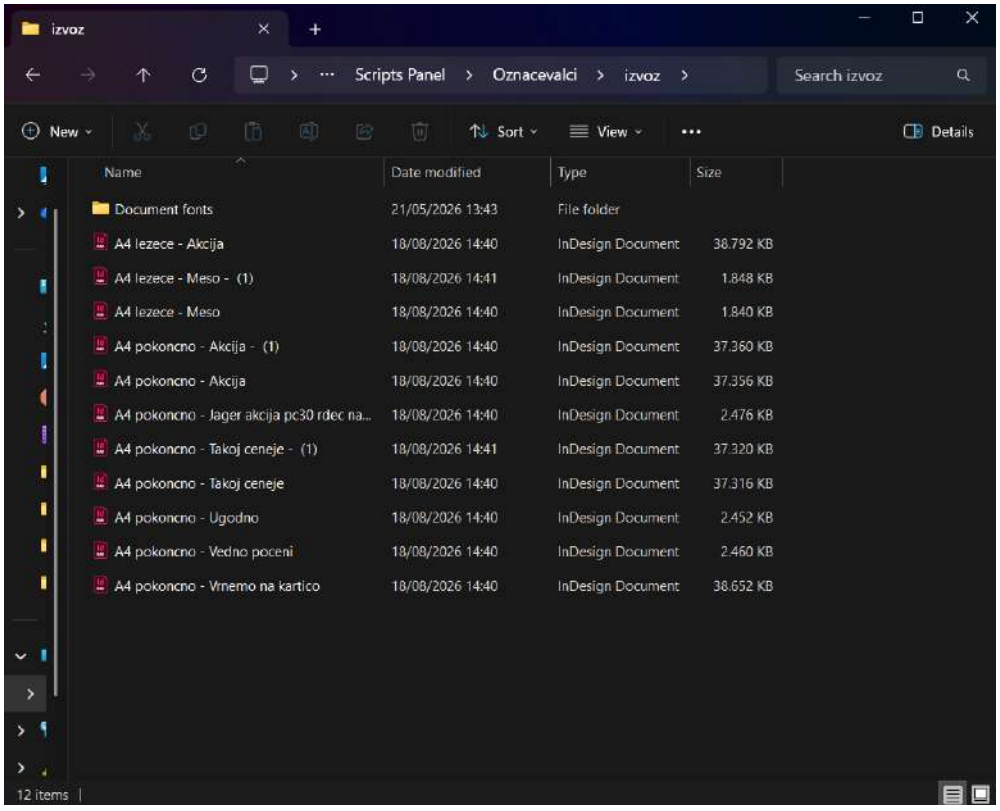
Generirane datoteke se shranijo v mapo izvoz/, ki se nahaja znotraj mape Oznacevalci v mapi Scripts Panel:

C:\Users[uporabnik]\AppData\Roaming\Adobe\InDesign\Version [X]\en_US\Scripts\ScriptsPanel\Oznacevalci\izvoz

Ime datoteke je sestavljeno iz imena formata, na primer:

A4 pokončno – Takoj ceneje.indd

Če datoteka z istim imenom že obstaja, se samodejno doda številka v oklepaju, npr. A4 pokončno – Takoj ceneje (1).indd



Slika 26: Generirane datoteke v mapi izvoz/

Vir: (Lastni vir)

Po zaključku generiranja se datoteke samodejno odprejo v InDesignu in so pripravljene za tiskanje ali nadaljnje urejanje.

7.8 Pogosti problemi

Tabela 8: Pogosti problemi in njihove rešitve

Problem	Rešitev
sporočilo "Format ni prepoznan"	Preverimo, ali je izbrani format v config.jsx pravilno zapisan in ustreza seznamu formatov.
sporočilo "Predloga ne obstaja"	Preverimo, ali je pot do predloge v config.jsx pravilna in ali datoteka na disku dejansko obstaja.
sistemsko nezaznavanje novega elementa po kliku Prekini	Preden se začne novo označevanje elementov, je treba po vsaki prekinitvi sistem ponovno aktivirati s Ctrl+Shift+1.
viden bel rob slike po generiranju	Sistem ne loči vsebine slike od praznega roba okoli nje. Sliko ročno prilagodimo prostoru.
nujnost označitve naziva in cene pri formatu Mali poličnik (čeprav ju format ne uporablja)	To je trenutna omejitev sistema, saj sta naziv in cena za potrditev zajema obvezna, in to ne glede na izbrani format.
sporočanje sistema o napačnem vrstnem redu elementov	Elemente vedno označujemo v točno določenem vrstnem redu: slika, naziv, cena in po potrebi ikone.
označenost elementov prejšnjega artikla	Pred zajemom naslednjega artikla enkrat kliknemo izven strani letaka, da se prejšnja selekcija odznači.

Vir: (Lastni vir)

8 SKLEP

V diplomskem delu sem razvil sistem Oznacevalec, to je ExtendScript aplikacija za Adobe InDesign, ki avtomatizira pripravo promocijskih označevalcev za podjetje Jagros, d. o. o. Namen dela je bil ugotoviti, ali je v tem okolju mogoče s skriptiranjem avtomatizirati proces tako, da se čas priprave skrajša, in to ob upoštevanju tehničnih omejitev okolja ExtendScript in zahtevnosti pozicioniranja različnih grafičnih elementov znotraj fiksnih predlog. Cilj dela je bil dosežen, saj sem sistem uspešno razvil, njegovo delovanje je zanesljivo, prav tako pa je bil sistem preizkušen prek funkcionalnega testiranja, meritve časa in testiranja z dejanskimi uporabniki.

V nadaljevanju so navedene ugotovitve glede na pet postavljenih hipotez.

H1, ki trdi, da avtomatizacija skrajša čas priprave serije označevalcev v primerjavi z ročnim postopkom, se na podlagi meritev iz poglavja 5.2 potrjuje. Sistem je v vseh treh izvedenih meritvah skrajšal čas priprave, v povprečju za približno 16,5 odstotka. Prihranek je zmeren, kar je smiselno glede na to, da avtomatiziran postopek še vedno vključuje ročno označevanje elementov v letaku in občasne popravke slik.

H2 predpostavlja, da je tehnične omejitve okolja ExtendScript mogoče nadomestiti z lastnimi implementacijami pomožnih mehanizmov, kar je prav tako potrjeno. Odsotnost vgrajenega objekta JSON sem rešil z lastno implementacijo v modulu JSONHelper, odsotnost trajnega spomina med sejami pa s shranjevanjem stanja v datoteko state.json, kar je opisano v poglavjih 4.1 in 4.2. Sistem je bil v poglavju 5.1 funkcionalno testiran na 25 testnih primerih. V 22 primerih je deloval popolnoma pričakovano, pri zajemu več slik (dveh ali treh) pa je bila pri nekaterih slikah po generiranju potrebna manjša ročna korekcija pozicije. Pri enem primeru (slika z izrazito belo obrobo) sistem ni deloval pričakovano, kar se sklada z omejitvijo, opisano v nadaljevanju tega poglavja. Rezultati kljub temu potrjujejo, da rešitve v produkcijskem okolju v veliki večini primerov delujejo zanesljivo.

H3, ki trdi, da je sistem mogoče razviti tako, da podpira različne tipe grafičnih elementov in različne formate predlog znotraj enotne arhitekture brez podvajanja kode, se na podlagi poglavja 4.4 ne potrjuje. Razlike med dvanajstimi formati predlog so bile prevelike, da bi jih bilo mogoče brez dodatne kompleksnosti pokriti s skupno logiko, zato sem zaradi rokov in navedenih razlik izbral rešitev z ločenimi pravili za posamezen format v modulu layoutRules.jsx. Prednost te rešitve je preprosta prilagoditev posameznega formata, slabost pa

večje podvajanje kode in posledično težje vzdrževanje. To je bila zavestna odločitev, ki ni bila posledica pomanjkanja prizadevanj za iskanje skupnih vzorcev.

H4 trdi, da je sistem mogoče dokumentirati z uporabniškimi navodili, tako da je razumljiv uporabniku brez predhodnega tehničnega znanja, in se na podlagi testiranja iz poglavja 5.3 potrjuje. Vseh pet udeležencev brez izkušenj z Adobe InDesignom je nalogo uspešno zaključilo izključno na podlagi pisnih navodil. Ocene razumljivosti navodil so bile v povprečju visoke. Primerjava s skupino izkušenih uporabnikov InDesigna je pokazala, da je predhodno znanje nalogo olajšalo in pospešilo, ni pa bilo pogoj uspešne izvedbe.

H5, ki predpostavlja, da vgrajena validacija selekcije in jasna sporočila o napakah zmanjšujejo možnost uporabniških napak, se na podlagi poglavja 5.4 delno potrjuje. Za šest sporočil, vezanih na validacijo selekcije ("Ni izbranih elementov", "Izberi vsaj 3 elemente", "Prvi element mora biti slika", "Po slikah mora biti naziv", "Po nazivu mora biti cena" in "Najprej aktiviraj Oznacevalec s Ctrl+Shift+1"), hipotezo potrjujem, saj vsa sporočila uporabniku takoj povedo, kaj je narobe in kako naj nadaljuje. To so med testiranjem udeleženci, ki so na sporočila naleteli, z visokimi ocenami v anketi in brez potrebe po dodatni pomoči tudi potrdili. Za dve tehnični sporočili, to sta "Format ni prepoznan" in "Predloga ne obstaja", pa hipoteze ne morem potrditi, saj uporabniku povesta le, da je nekaj narobe, ne pa tudi tega, kako naj napako odpravi. Teh dveh sporočil med testiranjem ni sprožil noben udeleženec, zato ocena temelji izključno na analizi kode.

Skozi razvoj sistema Oznacevalec so se pokazale tudi konkretne omejitve. Sistem ne loči vsebine slike od morebitnega belega roba okoli nje, kar pri določenih slikah po generiranju zahteva ročni popravek. Format Mali poličnik zaradi arhitekture sistema od uporabnika zahteva označitev naziva in cene, čeprav ju ta format ne uporablja, ker se format izbere šele po zajemu elementov. Obe omejitvi bi bilo mogoče z drugačno zasnovo procesa odpraviti, na primer z izbiro formata pred zajemom elementov namesto po njem, vendar bi to zahtevalo poseg v celotno strukturo procesa, ki jo je na začetku razvoja opredelil mentor v podjetju. Prav tako ostaja podvajanje kode med formati, opisano pri H3, tehnični dolg sistema, ki bi ga bilo ob morebitnem nadaljnjem razvoju vredno ponovno proučiti, morda z uvedbo skupnih vzorcev za formate s podobnejšo strukturo.

Na podlagi ugotovitev predlagam naslednje izboljšave nadaljnjega razvoja sistema. Prva izboljšava je sprememba vrstnega reda korakov, tako da se format izbere pred zajemom elementov. To bi omogočilo validacijo, ki bi bila prilagojena vsakemu formatu posebej, hkrati

pa bi odpravilo nepotrebno označevanje naziva in cene pri formatu Mali poličnik. Druga izboljšava bi bila dopolnitev tehničnih sporočil o napakah, kjer bi manjkajoče predloge ali neprepoznan format opremili s konkretnimi navodili za odpravo težave. Tretja izboljšava bi bil razmislek o delnem poenotenju arhitekture modula layoutRules.jsx za formate s podobnejšo strukturo, kolikor seveda razlike med predlogami to dopuščajo.

Sistem Oznacevalec je bil razvit iterativno v sodelovanju z mentorjem v podjetju, ki je podajal funkcionalne zahteve in predloge, celotno tehnično zasnovo, arhitekturo in implementacijo pa sem izvedel sam. Diplomsko delo je pokazalo, da je mogoče s skriptiranjem v okolju Adobe InDesign ExtendScript, kljub njegovim tehničnim omejitvam, zgraditi zanesljiv sistem, ki skrajša čas priprave promocijskih označevalcev, poleg tega pa je razumljiv tudi uporabnikom brez predhodnega tehničnega znanja. Diplomsko delo je prav tako razkrilo konkretne kompromise in omejitve, s katerimi sem se pri razvoju sistema moral soočiti v realnem in časovno omejenem delovnem okolju.

9 VIRI, LITERATURA

Adobe Systems. (2013). *JavaScript Tools Guide.* Pridobljeno iz https://blog.cutterman.cn/assets/files/JavaScript-Tools-Guide-CC.pdf

Adobe Systems. (2026). *Scripting in Adobe InDesign.* Pridobljeno iz https://helpx.adobe.com/indesign/desktop/automation-and-scripting/document-automation/automate-workflows-with-scripts.html

Brumen, N. (2025). *Reinženiring poslovnega procesa pri prehodu tiskanega medija na digitalno obliko.* Pridobljeno iz Digitalna knjižnica Univerze v Mariboru (DKUM): https://dk.um.si/IzpisGradiva.php?id=93391

Flaherty, K. (16. januar 2018). *What the erroneous Hawaiian missile alert can teach us about error prevention*. Pridobljeno iz Nielsen Norman Group: https://www.nngroup.com/articles/error-prevention/

Flanagan, D. (2020). *JavaScript: The Definitive Guide.* O'Reilly Media.

Hopkins, S. W. (2009). *Automating Adobe InDesign CS4 with ExtendScript.* Adobe Press.

MDN Web Docs. (b.d.). *JavaScript.* Pridobljeno iz https://developer.mozilla.org/en-US/docs/Web/JavaScript

MDN Web Docs. (b.d.-b). *JSON.* Pridobljeno iz https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global_Objects/JSON

MDN Web Docs. (b.d.-c). *Object-oriented programming.* Pridobljeno iz https://developer.mozilla.org/en-US/docs/Learn_web_development/Extensions/Advanced_JavaScript_objects/Object-oriented_programming

Sagar, G. (3. februar 2025). *How to use Adobe ExtendScript to automate InDesign, Illustrator and Acrobat*. Pridobljeno iz MetaDesign Solutions: https://metadesignsolutions.com/blog/how-to-use-adobe-extendscript-to-automate-indesign-illustrator-and-acrobat

Sommerville, I. (2016). *Software Engineering.* Pearson.