Implementacija alata umjetne inteligencije za automatsku analizu i pohranu podataka iz nestrukturiranih izvora

Barešić, Dino

Sveučilište u Zadru

Odjel za informacijske znanosti

Stručni prijediplomski studij

Informacijske tehnologije

Dino Barešić

IMPLEMENTACIJA ALATA UMJETNE INTELIGENCIJE ZA AUTOMATSKU ANALIZU I POHRANU PODATAKA IZ NESTRUKTURIRANIH IZVORA

Završni rad

Zadar, 2024.

Sveučilište u Zadru
Odjel za informacijske znanosti
Stručni prijediplomski studij
Informacijske tehnologije

IMPLEMENTACIJA ALATA UMJETNE INTELIGENCIJE ZA AUTOMATSKU ANALIZU I POHRANU PODATAKA IZ NESTRUKTURIRANIH IZVORA

Završni rad

Student/ica: Dino Barešić

Mentor/ica: Doc. dr sc. Tomislav Jakopec

Zadar, 2024.

Izjava o akademskoj čestitosti

Ja, **Dino Barešić,** ovime izjavljujem da je moj **završni** rad pod naslovom **Implementacija alata umjetne inteligencije za automatsku analizu i pohranu podataka iz nestrukturiranih izvora** rezultat mojega vlastitog rada, da se temelji na mojim istraživanjima te da se oslanja na izvore i radove navedene u bilješkama i popisu literature. Ni jedan dio mojega rada nije napisan na nedopušten način, odnosno nije prepisan iz necitiranih radova i ne krši bilo čija autorska prava.

Izjavljujem da ni jedan dio ovoga rada nije iskorišten u kojem drugom radu pri bilo kojoj drugoj visokoškolskoj, znanstvenoj, obrazovnoj ili inoj ustanovi.

Sadržaj mojega rada u potpunosti odgovara sadržaju obranjenoga i nakon obrane uređenoga rada.

Zadar, 31. svibnja 2024.

SAŽETAK

Ovaj rad istražuje tehnologije i pristupe implementaciji alata umjetne inteligencije za analizu i pohranu nestrukturiranih podataka. Tehnologije koje se koriste u praktičnom dijelu i bit će obrađene u ovom radu su: Java programski jezik, razvojni okvir Eclipse Scout, PostgreSQL baza podataka i PG Admin upravitelj baze podataka. Alat umjetne inteligencije koji se implementira u programsko rješenje je *Receipt & Invoice OCR* od tvrtke *Taggun*. Korištene vanjske biblioteke u praktičnom dijelu rada su: JDBC, Gson i okHttp. Vanjske biblioteke koje su navedene su bile potrebne za spajanje na bazu podataka i za implementaciju samog AI alata. U prvom dijelu opisuje se svrha i cilj ovog rada te nakon toga funkcionalni zahtjevi za razvoj programskog rješenja i same implementacije ovog alata. Drugi dio istražuje korištene tehnologije i pojmove iz teorije koje su bliske ovom alatu umjetne inteligencije. Zadnji dio rada fokusira na praktični dio implementacije alata umjetne inteligencije za prepoznavanje i strukturiranje podataka iz nestrukturitranih izvora, te teorijski dio korištenih vanjskih biblioteka. Kompletan kod aplikacije dostupan je s javnog repozitorija na GitHubu i nalazi se na poveznici : https://github.com/dinobaresic/Zavrsni_rad_IT_2024.

Ključne riječi: Umjetna inteligencija, Eclipse Scout, PostgreSQL, Java programski jezik, AI, OCR

POPIS KORIŠTENIH KRATICA

AI Artificial Intelligence = umjetna inteligencija

OCR Optical Character Recognition = optičko prepoznavanje znakova

BSI Business System Integration = poslovna sistemska itegracija

IBM International Business Machines = multinacionalna IT tvrtka sa sjedištem u SAD-u

JS JavaScript = programski jezik

SQL Structured Query Language = skriptni jezik za upravljanje relacijskim bazama podataka

JDBC Java Database Connectivity = vanjska biblioteka koja omogućuje programima spajanje na bazu podataka

JPG Joint Photographic Group = format ili ekstenzija za fotografije

PNG Portable Network Graphics = format ili ekstenzija za fotografije

PDF Portable Document Format = format ili ekstenzija za dokumente

API Application Programming Interface = skup definiranih funkcija i protokola koje programi koriste za komunikaciju međusobno ili s vanjskim sustavima.

SADRŽAJ

UVOD

Ovaj rad se fokusira na rješavanje jednog od ključnih problema u poslovnim sustavima, obradu podataka iz nestrukturiranih izvora. U praktičnom dijelu rada se unapređuje proces unosa podataka o troškovima u aplikaciji za praćenje troškova korisnika. Kako bi smanjili vrijeme čitanja podataka s računa i unosa u aplikaciju ispunjavanjem formi, u praktičnom dijelu rada implementirati će se alat umjetne inteligencije koji će sve to obavljati umjesto samog korisnika. Alat umjetne inteligencije koristi napredne algoritme kojim čita tražene podatke iz nestrukturiranog izvora kao što je slika i popunjava formu. Implementacija ovog alata predstavnja značajan korak ka automatizaciji i poboljšanju efikasnosti poslovanja. Rad će detaljno razmotriti korištene tehnologije i njihovu primjenu u praktičnom dijelu rada. Na početku rada opisati će se ciljevi i svrha te funkcionalni zahtjevi razvoja aplikacije. Drugi dio rada opisuje korištena programska rješenja te teorijski dio koji je povezan s alatom koji je implementiran u aplikaciju. Zadnji dio rada istražuje praktični dio implementacije alata umjetne inteligencije za analizu i pohranu podataka iz nestrukturiranih izvora, fokusirajući se na kod same implementacije.

1. Cilj i svrha implementacije OCR alata umjetne inteligencije

Jedan od glavnih ciljeva implementacije alata umjetne inteligencije za automatsko prepoznavanje znakova je unaprijeđenje poslovnog procesa sustava za praćenje financija jednog korisnika. U konačnici ovaj alat će značajno unaprijediti funkciju same aplikacije tako što će kao što je navedeno u uvodu značajno ubrzati obradu podataka iz nestrukturiranih izvora, unos računa iz izvora npr. slika računa. Sa ovim OCR AI alatom dobiti će se prijedlog direktno za strukturirane podatke iz takvih izvora i tako će se uštedjeti velika količina vremena samih korisnika aplikacije. OCR i analiza dokumenata još uvijek igraju važnu ulogu u poslovnim procesima, kako u razvijenim tako i u zemljama u razvoju. Prepoznavanje teksta blisko je povezan problem.[1] Kroz implementaciju ovog alata, cilj je osigurati korisnicima jednostavan i učinkovit način za praćenje njihovih financija, pružajući im pouzdano rješenje za upravljanje njihovim troškovima i planiranje budućih financijskih strategija. Korištenje naprednih tehnologija poput umjetne inteligencije za interpretaciju prepoznatih znakova i OCR za prepoznavanje znakova omogućuje aplikaciji da se prilagodi i optimizira za različite vrste dokumenata i različite jezike, što doprinosi njenoj univerzalnoj primjenjivosti i skalabilnosti.

Svrha implementacije ovog alata je olakšati proces analize i pohrane podataka iz nestrukuritanih izvora, pružajući automatizirano rješenje za prepozavanje i strukturiranje potrebnih podataka koje poslovni sustav traži. Odabir ovakvog projekta proizlazi iz potrebe za implementacijom naprednih i aktualnih tehnologija kako bi se osigurala visoka razina točnosti i pouzdanosti u prepoznavanju relevantnih podataka. Korisnicima će se pružiti pouzdan alat za upravljanje svojim financijama na učinkovit i intuitivan način. Također svrha implementacije ovog alata je poboljšati učinkovitost poslovnih procesa povezanih s praćenjem troškova tako što će omogućiti bržu i precizniju analizu podataka, smanjujući vrijeme i resurse potrebne za ručni unos i provjeru podataka. Implementacija ovog OCR AI alata omogućiti će korisnicima bržu analizu financijskih podataka, što će rezultirati boljim donošenjem odluka i optimizacijom njihovih financijskih strategija, dok će istovremeno tvrtki pružiti konkurentsku prednost kroz smanjenje operativnih troškova i povećanje produktivnosti.

Implementacija AI alata u poslovne sustave može transformirati različite sektore u poslovanju kao što su zdravstvena industrija, financijski sektor, logistika i obrazovanje. Na

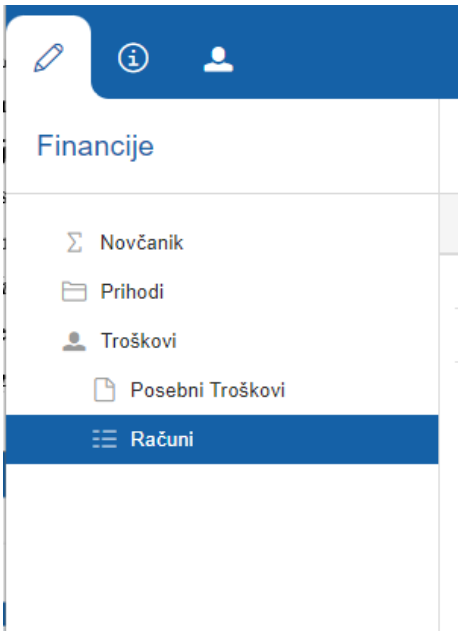
[1] Thomas Breuel portfolio mrežna stranica, http://www.tmbdev.net/projects/ocr/ [pristupljeno 26. ožujka 2024.]

primjer, u zdravstvenoj industriji, AI može automatski interpretirati medicinske dokumente, što omogućuje bržu dijagnozu i obradu pacijenata. U financijskom sektoru, AI može automatizirati unos podataka iz faktura i financijskih izvještaja, čime se smanjuje potreba za ručnim unosom podataka i povećava točnost i brzina financijskih transakcija.[2] AI također može značajno unaprijediti obrazovanje, učenje povijesti može biti unaprijeđeno korištenjem AI aplikacija za vizualizaciju povijesnih događaja, simulacije i interaktivne aktivnosti koje omogućuju dublje razumijevanje i angažman učenika. Korištenje AI u obrazovanju može omogućiti personalizirano učenje, gdje AI asistenti pomažu u prepoznavanju individualnih potreba učenika i prilagođavanju nastavnih metoda. To može dovesti do boljih rezultata i povećane motivacije učenika. No, ključno je osigurati da tehnologija bude korištena etički i odgovorno, uz poštivanje privatnosti i sigurnosti podataka učenika.[3]

1.1. Funkcionalni zahtjevi razvoja programskog rješenja i implementacije OCR alata

Implementacija AI OCR alata je glavni zadatak i zahtjev praktičnog dijela rada, također jedan od zahtjeva je razvoj aplikacije na kojoj je primjenjena implementacija ovog AI alata. Kroz praktični dio ovog rada prvo je razvijena aplikaciju na koju se kasnije implementirao sami alat. Aplikacija korisniku omogućuje praćenje troškova, kao što su ulazni računi korisnika.

[2] Neha Soni, Enakshi Khular Sharma, Narotam Singh, Amita Kapoor, *Artificial Intelligence in Business: From Research and Innovation to Market Deployment,*
https://www.sciencedirect.com/science/article/pii/S1877050920307389, [pristupljeno 31.svibnja 2024.]
[3] Miljenko Hajdarović, *Umjetna inteligencija, ChatGPT i poučavanje Povijesti*,
https://hrcak.srce.hr/clanak/437453, [pristupljeno 31.svibnja.2024.]



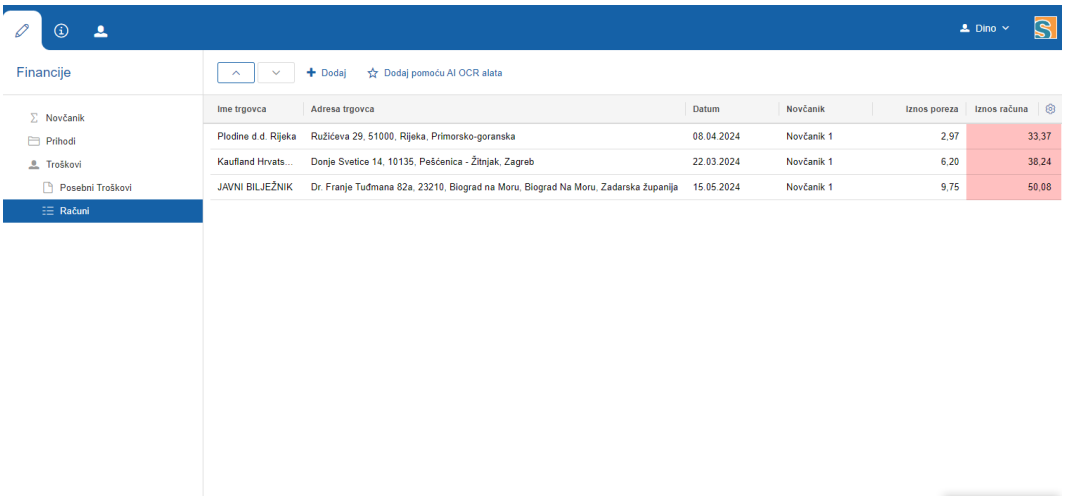
Slika 1: modul Financije [4]

Prema navedenim zahtjevima aplikacija je podijeljena u tri modula. Prvi modul pod nazivom *Financije,* ovaj modul sadrži tri podmodula, a to su *Novčanik, Prihodi i Troškovi.* Prvi podmodul *Novčanik* nam prikazuje sve novčanike koje imamo i trenutno stanje na svakom, također imamo mogućnost brisanja i dodavanja novčanika. Drugi podmodul *Prihodi* nam daje tablicu s prikazanim svim prihodima koje smo dodavali, također imamo mogućnost brisanja i dodavanja prihoda i na ovom modulu. *Troškovi* je stranica čvora koji se sastoji od dvije pojedinačne stranice to su *Posebni troškovi i Računi,* na stranici *Računi* dodaju se podaci o ulaznim računima iz nestrukturiranih izvora, a na stranici *Posebni Troškovi* imamo mogućnost dodavanja troškova koji nisu u strukturi računa. U podmodulu *Računi* sam implementirao OCR AI alat za prepoznavanje relevantnih podataka sa slike (JPG ili PNG) i drugih nestrukturiranih izvora kao što je PDF, te spremanja istih u bazu podataka i prikaz u tablici računa.

U podmodulu *Računi* imamo prikaz svih podataka o računima tablično i mogućnost dodavanja novih računa te brisanja postojećih. U dodavanju novih ulaznih računa je trebalo impelementirati OCR AI alat, klikom na gumb menija *Dodaj pomoću AI OCR alata* otvara se prazna forma za unos novih računa. Kod dodavanja računa bez korištenja OCR AI alata, ispunjavamo tražene podatke kao što su: *ime trgovca, adresa trgovca, iznos računa, iznos poreza, datum i naziv novčanika.* Ukoliko želimo korisititi OCR AI alat, potrebno je kliknuti na *BillField* koji je u praktičnom dijelu klasa koja proširuje klasu *AbstractFileChooserField.*

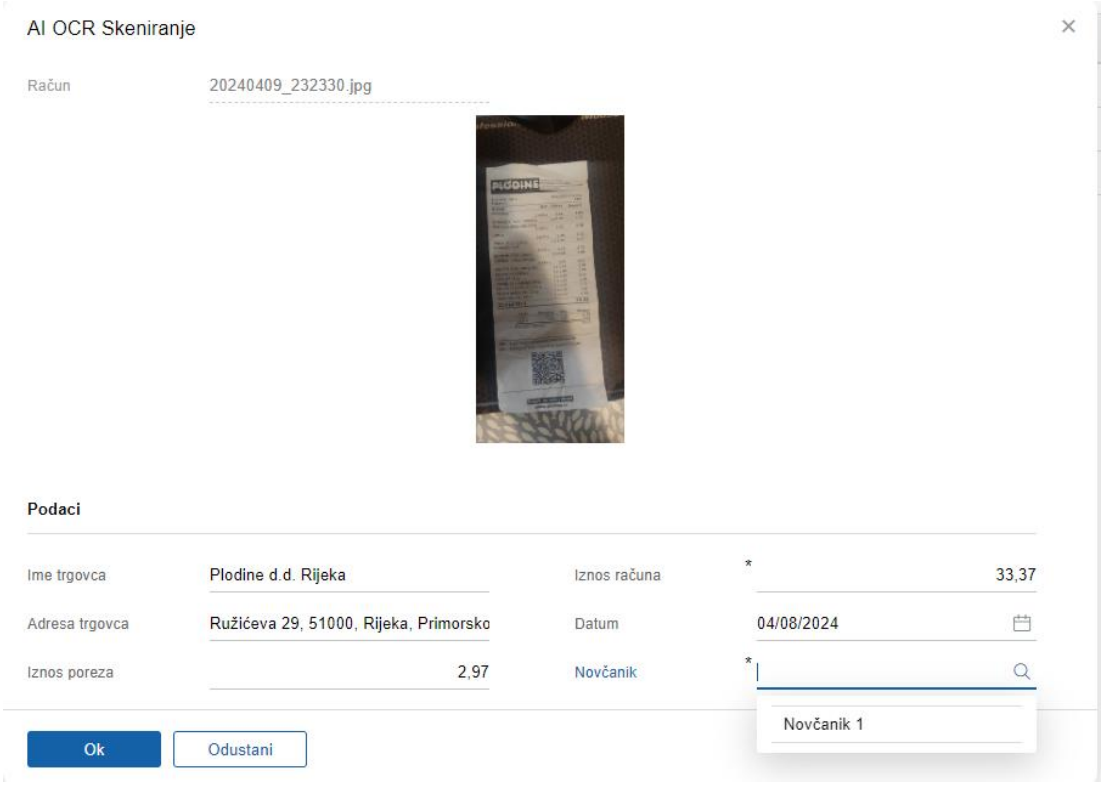
[4] Slika zaslona iz praktičnog djela rada

Klikom se to polje otvara nam se sučelje za odabir datoteke nestrukturiranog izvora. Odabir datoteke je trebalo ograničiti na *jpg*, *png* i *pdf*, a to je postignuto metodom *getConfiguredFileExtensions()*, koja vraća listu s dozvoljenim ekstenzijama datoteke koja će se učitati na polje. Ako se pokuša učitati neka druga datoteka koja nije dozvoljenog formata, dobit će se poruka *Rejected file*. Kad se učita slika ako je sve prošlo uredu, učitana slika se prikazuje na *ImageFieldu* te se očitani podaci pomoću alata prikazuju svaki na svom polju. Prijedlog OCR AI alata može se prihvatiti te spremiti kao takav ili promjeniti ako ne želimo tu očitanu vrijednost spremiti na određeno polje. Ako se učita pdf datoteka, neće se dobiti prikaz tog pdf na formi nego će se samo ispuniti polja očitanim vrijednostima, te također imamo mogućnost promjene. Prije spremanja podataka u bazu i prikaza u tablici potrebno je još odabrati novčanik na polju *naziv novčanika* koji je u praktičnom dijelu klasa *WalletField* koji proširuje klasu *AbstractSmartField<Long>*. To znači da se na to polje prikazuju podaci koji su već unaprijed definirani, u ovom slučaju će prikazivati podatke iz baze podataka i to se naziva *SqlLookup*. Prikazivati ćemo podatke iz baze podataka iz tablice *wallet*, po njihovom *id* a prikazivati će se naziv novčanika. Nakon što su sva polja ispunjena u formi, klikom na *Ok* podaci se spremaju u bazu podataka. Novi uneseni podaci bit će vidljivi u tablici s računima u podmodulu *Računi*.



Slika 2: pregled podataka o računima [5]

[5] Slika zaslona iz praktičnog djela rada



Slika 3: forma za unos novog računa [6]

[6] Slika zaslona iz praktičnog djela rada

2. Programsko rješenje

Programsko rješenje predstavljaju sve tehnologije koje su korištene u razvoju ove funkcionalnosti aplikacije, te u implementaciji alata umjetne inteligencije koji ima svrhu automatske analize nestrukturiranih izvora. U procesu razvoja ovog programskog rješenja korišten je niz tehnologija koje su bile nužne kako bi mogli razviti ovu funkcionalnost aplikacije, te na kraju implementirati OCR alat i unaprijediti poslovanje ovakovog sustava. Također u ovom dijelu bit će obrađeni pojmovi vezani za ovaj alat, umjetna inteligancija i OCR.

2.1. Umjetna inteligencija

Umjetna inteligencija je u današnje vrijeme često glavna tema, pogotovo od kada se počela primjenjivati u svim porama poslovnog ali i društvenog svijeta. Alat koji je implementiran u praktičnom dijelu rada da bi unaprijedili sustav prepoznaje relevantne podatke jednim dijelom pomoću umjetne inteligencije. Umjetna inteligencija je područje računalne znanosti koje se bavi razvojem sustava sposobnih za obavljanje zadataka koji obično zahtijevaju ljudsku inteligenciju, poput prepoznavanja govora, donošenja odluka i učenja. AI obuhvaća širok spektar tehnika, modela i pristupa, kao što su strojno učenje, duboko učenje i velike jezične modele.[7] Ovaj alat umjetne inteligencije koristi velike jezične modele (LLM), trenirane nad podacima pomoću strojnog učenja (ML). Veliki jezični modeli su specifičan tip modela strojnog učenja treniranih na ogromnim količinama tekstualnih podataka kako bi razumjeli i generirali ljudski jezik. Veliki jezični modeli se temelje na transformator modelu, arhitekturi koja koristi *pozornost na sebe* mehanizam za pretvaranje jedne sekvence tokena u drugu, što je idealno za obradu prirodnog jezika. Ovi modeli koriste napredne tehnike dubokog učenja za obradu i generiranje prirodnog jezika s visokom razinom preciznosti i koherencije.[8] Primjer je model poput generativnog unaprijed obučenog transformatora (GPT). Strojno učenje je podskup AI-a koji se fokusira na razvoju algoritama koji omogućuju računalima da uče iz podataka. Umjesto da se programira svaki korak rješenja problema, algoritmi strojnog učenja koriste velike količine podataka za učenje i poboljšanje svojih performansi tijekom nekog vremena.[9] Umjetna inteligencija crpi inspiraciju iz mozga, kognitivni znanstvenici i neuroznanstvenici čiji je cilj razumjeti funkcioniranje mozga i prema tome grade modele

[7] Russell, S. J., & Norvig, P. , *Artificial Intelligence: A Modern Approach (4. izdanje)*, 2020., str. 1-5.
[8] Ashish Vaswani, Noam Shazeer, *Attention is All you Need,* 2017., str. 1-10.
[9] ETHEM ALPAYDIN, *Machine Learning: The New AI,* 2016., str. 19.

neuronskih mreža.

2.2. OCR

OCR je tehnologija za automatsko izdvajanje i pohranjivanje raznih podataka iz skeniranih dokumenata ili slika. OCR prepoznaje i obrađuje tekst sa slike pretvarajući pojedinačne znakove u riječi, a riječi u rečenice te ih digitalizira u podatke poznate računalu. Kombinacija fizičke (optički skeneri) i programske opreme omogućuje pretvorbu tiskanih fizičkih dokumenata u računalu prepoznatljiv format. AI koristi OCR za napredne metode prepoznavanja znakova, uključujući prepoznavanje jezika. OCR također se može koristiti za automatizaciju unosa podataka, pomaže i osobama s oštećenim vidom i optimizira obradu velikih podataka. Korištenjem OCR poslovne tvrtke mogu smanjiti troškove i ubrzati radne procese u poslovanju.[10]

2.3. Java programski jezik

Java programski jezik je korišten za razvoj ovog programskog rješenja, te također za implementaciju samog AI alata. Java je objektno orijentirani programski jezik kojeg je dizajnirao James Gosling 1995. godine dok je radio u tvrtci Sun Microsystems. Tvrtku je 2010. godine kupila tvrtka Oracle. Java je programski jezik opće namjene koji se koristi u svim industrijama i za gotovo sve vrste aplikacija.[11] Java je višeplatformski, objektno orijentiran programski jezik. Brz, siguran i pouzdan za kodiranje svega od poslovnih programa do aplikacija za velike podatke i tehnologija na strani poslužitelja. Java je prisutna u različitim područjima, uključujući razvoj igara, analizu velikih podataka, ta naravnoj i samoj umjetnoj inteligenciji. Java se ističe svojom jednostavnošću korištenja i sposobnosti da podrži razvoj različitih vrsta aplikacija.[12]

2.4. Razvojni okvir Eclipse Scout

Eclipse Scout je besplatni razvojni okvir koji omogućuje razvoj profesionalnog softvera u Java ili JavaScript programskom jeziku. Razvijen od strane tvrtke BSI Software, Scout pruža jedinstveno okruženje s jasnim konceptima, snažnim aplikacijskim modelom i svestranim korisničkim sučeljem. Osnovana 1996. godine, BSI je privatno vlasništvo s djelatnicima koji

[10] IBM službena stranica, https://www.ibm.com/blog/optical-character-recognition/, [pristupljeno, 22. svibnja 2024.]
[11] Yakov Fain, *Java programming*, 2015., str. 1.
[12] Amazon mrežna stranica, *https://aws.amazon.com/what-is/java/*, [pristupljeno, 19. travnja 2024.]

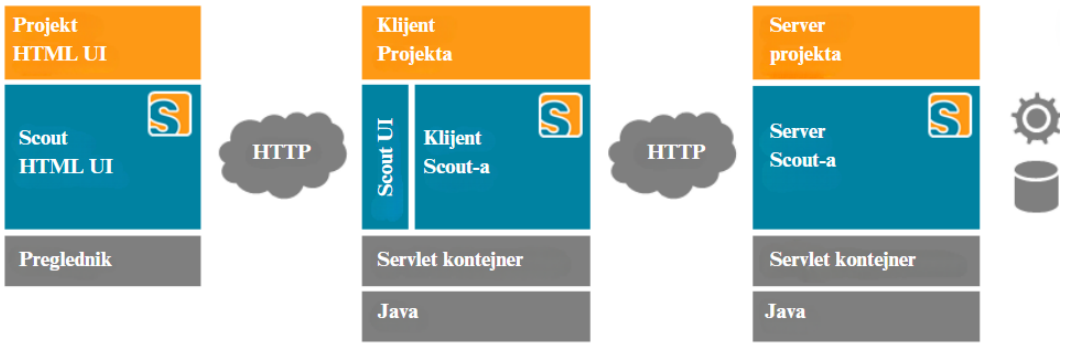
su dijelom vlasnici, bez hijerarhija. Njihov uspjeh temelji se na snažnim vrijednostima, proizvodima najviše klase i vrhunskim zaposlenicima. Njihova programska rješenja podržavaju digitalizaciju dijaloga s korisnicima i misiju zadovoljstva korisnika, kombinirajući najbolje tehnologije, poslovno znanje i upravljanje projektima za savršeno korisničko iskustvo na svim točkama kontakta. Scout se koristi za razvoj poslovnih, mrežnih aplikacija i aplikacija za radne površine. [13] Kod za Scout aplikaciju može biti baziran na Java programskom jeziku ili JavaScript-u, ima dvije opcije korištenja osnovnog programskog jezika u razvijanju programskog rješenja, za praktični dio ovog rada odabrana je Java. Zadnja verzija Scouta je 24.1, izašla je u ožujku 2024. godine.[14] Također, jedna od novosti u vezi Scout aplikacija je da sada, osim Eclipse IDE-a, za razvoj Scout aplikacija možete koristiti i IntelliJ IDE od JetBrains-a. Izražavajući interes za stjecanjem znanja o tome kako izraditi aplikaciju koristeći IntelliJ kao IDE, upravo je on korišten za razvoj ove aplikacije i implementaciju samog alata umjetne inteligencije.

2.4.1. Radna Arhitektura Eclipse Scout-a

Scout je razvojni okvir za kreiranje suvremenih poslovnih aplikacija koji pruža mogućnost razdvajanja aplikacija na više slojeva radi bolje organizacije i efikasnosti. Uobičajena Scout aplikacija obuhvaća serverski, klijentski i korisnički sloj, svaki s jasno definiranim zadacima i odgovornostima. Serverski sloj obično se bavi pohranom podataka u bazu podataka i pružanjem te korištenjem web usluga, uz pomoć korisnih alata koje pruža Scout za pojednostavljenje ovih zadataka. Klijentski sloj, s druge strane, odgovoran je za upravljanje korisničkim sučeljem, koristeći model koji se sastoji od običnih Java klasa te usluga i alata za implementaciju ponašanja povezanog s klijentskim kodom. Scout također pruža alate koji olakšavaju razvoj klijentskog sloja, omogućujući programerima da se fokusiraju na implementaciju poslovne logike i funkcionalnosti. Konačno, korisnički sloj ima zadaću prikazivanja klijentskog modela u pregledniku, koristeći JS, HTML i CSS kod koji je već pružen unutar Scout okvira. Ovaj sloj često zahtijeva minimalan projektni specifični kod, budući da većina funkcionalnosti već postoji unutar Scout-a. Ukupno, arhitektura Scout-a omogućuje jasno razdvajanje odgovornosti i funkcionalnosti svakog sloja, što olakšava razvoj, održavanje i proširenje Scout aplikacija. Ovaj pristup pruža stabilnost, pouzdanost i

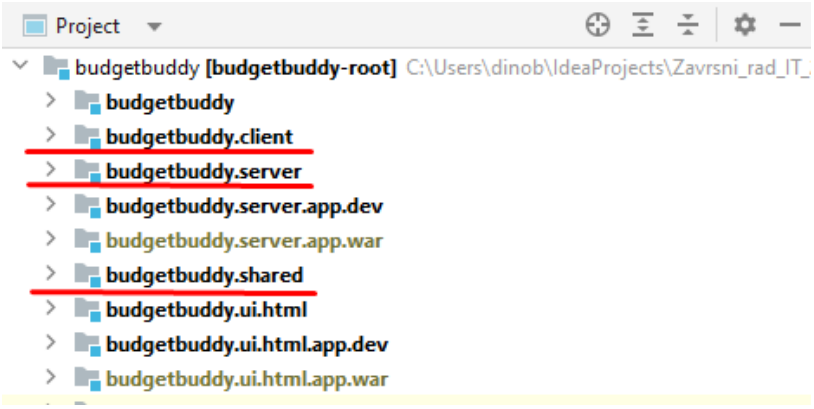
[13] Službena mrežna stranica BSI kompanije, https://www.bsi-lifesciences.com/about-bsi, [pristupljeno: 20.travnja 2024.]
[14] Službena dokumentacija Eclipse Scout-a, https://eclipsescout.github.io/scout-docs/24.1/getstarted/helloscout.html, [pristupljeno: 20.travnja 2024.]

skalabilnost, što je ključno za uspjeh modernih poslovnih aplikacija u dinamičnom okruženju današnjeg poslovanja.[15]



Slika 4: arhitektura Scout aplikacije [16]

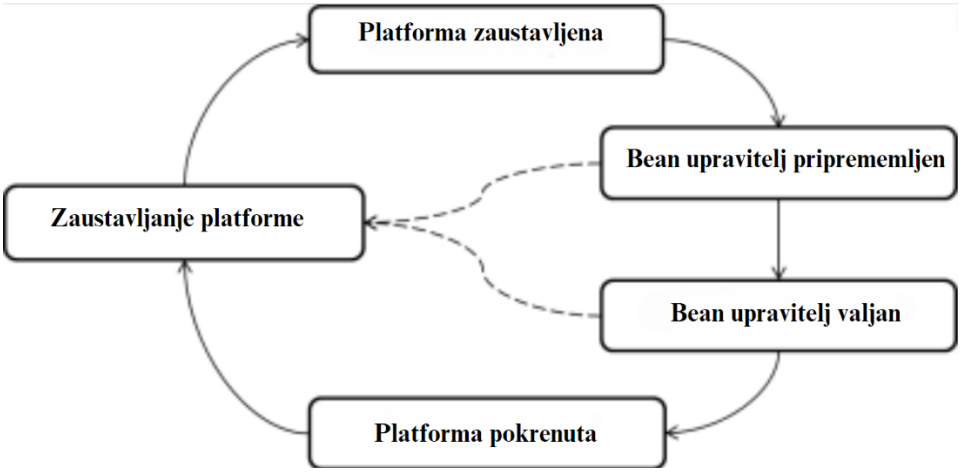
Unutar Scout aplikacija, interakcija između klijentskog i serverskog dijela odvija se putem dijeljenih direktorija klasa i sučelja, omogućujući stvaranje zajedničkih objekata za prijenos podataka između dva dijela aplikacije. Na klijentskoj strani, stvaraju se zastupnički objekti, poznati kao proxy objekti, koji preko dijeljenih resursa ili sučelja prenose upite za određene metode do serverskog dijela aplikacije. Serverski dio aplikacije, na temelju primljenih upita i podataka putem određenog sučelja, obrađuje podatke i izvršava odgovarajuće akcije. Klijentski dio aplikacije smješten je u direktoriju s ekstenzijom .client i komunicira s serverskom stranom, koja se nalazi u direktoriju s ekstenzijom .server, preko sučelja koja se nalaze u direktoriju s ekstenzijom .shared.



Slika 5:arhitektura paketa Scout aplikacije [17]

[15] Eclipse mrežna stranica dokumentacije, https://eclipsescout.github.io/scout-docs/24.1/technical-guide/technical-guide, [pristupljeno: 30. travnja 2024.]
[16] Eclipse mrežna stranica dokumentacije, https://eclipsescout.github.io/scout-docs/24.1/technical-guide/technical-guide, [pristupljeno: 30. travnja 2024.]
[17] Slika zaslona iz praktičnog djela rada

Scout također sadrži platformu koja pruža osnovne funkcionalnosti potrebne za mnoge programske aplikacije. Platforma je odgovorna za različite zadatke kao što su upravljanje ciklusom aplikacija, upravljanje instancama objekata, upravljanje konfiguracijom te inventar aplikacija. Životni ciklus Scout aplikacije kontrolira se implementacijama *org.eclipse.scout.rt.platform.IPlatform* sučelja. Ovo sučelje sadrži metode za pokretanje i zaustavljanje aplikacije te za dobivanje upravitelja *Bean-ova* povezanog s tom aplikacijom. Klasa *org.eclipse.scout.rt.platform.Platform* omogućuje pristup trenutnoj instanci platforme. Prilikom pristupa, platforma se automatski stvara i pokreće. Tijekom svog pokretanja, platforma prolazi kroz nekoliko stanja. Ovisno o stanju platforme, neke se komponente mogu već inicijalizirati i biti spremni za upotrebu, dok drugi možda još nisu dostupni. Enum *org.eclipse.scout.rt.platform.IPlatform.State* je odgovoran za opis svakog stanja i što je dostupno u određenom stanju. Dodavanje ove platforme Scout-u osigurava standardiziranu infrastrukturu i podršku za osnovne funkcionalnosti, što omogućuje konzistentan i pouzdan rad Scout aplikacija.[18]



Slika 6: stanja Socut aplikacije [19]

2.5. Baza podataka

U praktičnom dijelu rada za bazu podataka korištena je relacijska baza PostgreSQL, potiče iz paketa POSTGRES razvijenog na Sveučilištu u Berekeleyu. Kroz razvoj postao je jedna od najnaprednijih baza podataka dostupna bilo gdje.[20] U početku, projekt POSTGRES predvodio je profesor Michael Stonebraker i bio je sponzoriran od strane nekoliko istraživačkih

[18] Scout dokumentacija, https://eclipsescout.github.io/scout-docs/24.1/technical-guide/common-concepts/platform.html#sec-bean.manager, [pristupljeno 30. travnja 2024.]
[19] Scout dokumentacija, https://eclipsescout.github.io/scout-docs/24.1/technical-guide/common-concepts/platform.html#sec-bean.manager, [pristupljeno 30. travnja 2024.]
[20] 6Sense web stranica, https://6sense.com/tech/relational-databases, [pristupljeno 28.svibnja 2024.]

agencija. Sama implementacija započela je 1986. godine, a prve operativne verzije bile su dostupne 1987. godine. Koristio se u različite svrhe, uključujući financijsku analizu podataka, praćenje performansi mlaznih motora, praćenje asteroida te medicinske i geografske informacijske sustave. Nakon toga 1994. godine Andrew Yu i Jolly Chen dodali su interpreter, što je rezultiralo novim imenom *Postgres95.* Ova verzija je bila ANSI C kompatibilna i poboljšala je performanse i održivost u odnosu na prethodne verzije. Postgres95 je predstavljao otvoreni izvor originalnog POSTGRES Berkeley koda te je donio brojna poboljšanja, uključujući zamjenu PostQUEL upita s SQL-om, poboljšanu podršku za GROUP BY upite i nove alate za interaktivne SQL upite.[21] Navedeno je da je PostgreSQL relacijska baza podataka što znači da su podaci smješteni unutar tablica, a veze odnosno relacije između njih temelje se na posjedovanju jedinstvenih primarnih ključeva koji se povezuju s vanjskim ključevima. PostresSQL baza podataka je besplatna za preuzimanje i korištenje. Nakon instalacije da bi napravili novu bazu i spojili je na praktični dio te korisitili tablice potrebno je napraviti instancu i korisnika s lozinkom putem koje će se pristupiti samoj bazi podataka.

2.5.1. Spajanje baze podataka s Scout projektom

Da bi spojili PostgreSQL bazu podataka na Scout projekt bilo je potrebno za početak dodati vanjske biblioteke koje će biti potrebne za spajanje na bazu podataka. Prva koju trebamo je JDBC, koja određuje na koji način se kod napisan u Java programskom jeziku može spojiti na bazu podataka i koja omogućuje rad upita nad podacima. Druga je PostgreSQL JDBC koja specificira da se radi o PostgreSQL bazi podataka i koja omogućuje konekciju. Da bi dodali potrebne vanjske biblioteke zadužan je *Maven,* alat koji se koristi za izgradnju i upravljanje bilo kojim Java baziranim projektom.[22] Zavisnost se dodaje u pom.xml datoteku na server strani aplikacije, primjer dodavanja zavisnosti se vidi na slici 7.

[21] PostgreSQL mrežna stranica, https://www.postgresql.org/docs/current/history , [pristupljeno 14.svibnja 2024.]
[22] Maven mrežna stranica, https://maven.apache.org/what-is-maven , [pristupljeno 14.svibnja 2024.]

```
36        <dependency>
37          <groupId>org.eclipse.scout.rt</groupId>
38          <artifactId>org.eclipse.scout.rt.server.jdbc</artifactId>
39        </dependency>
40        <dependency>
41          <groupId>org.postgresql</groupId>
42          <artifactId>postgresql</artifactId>
43          <version>42.5.1</version>
44        </dependency>
```

Slika 7: kod za spajanje na bazu podataka[23]

Sljedeći korak je upis podataka za spajanje na tu bazu podataka čiji je konektor sad dio projekta. Podaci za pristup unose se u konfiguracijsku datoteku pod nazivom *config.properties* u kojoj se nalaze podaci za pristup određenim uslugama, u ovom slučaju bazi podataka.

```
11    ### DB Connection
12    db.url=jdbc:postgresql://localhost:5432/budget_app_dev
13    db.username=postgres
14    db.password=nHeaDeRm
```

Slika 8: podaci za pristup bazi podataka[24]

Da bi spojili aplikaciju na bazu ostao je samo još jedan korak, a to je kreiranje dvije klase iz kojih će se dohvaćati podaci za prijavu na bazu podataka. Prva će se nazvati *DatabaseProperties* kreirana klasa će imati tri unutarnje klase koje nasljeđuju metode iz abstraktne klase *AbstractStringConfigProperty*. Metode se zovu *URLProperty*, *UsernameProperty* i *PasswordProperty* koje vraćaju kao rezultat podatke za prijavu (*Slika 9: Kod klase DatabaseProperties*). Drugu klasu će se nazvati *PostgreSqlService*, ona nasljeđuje motode abstraktne klase *AbstractPostgreSqlService* koja pokupi vrijednosti iz prve klase i prosljeđuju ih ostalim klasama vanjske PostgreSQL JDBC biblioteke koje zatim prilikom slanja upita stvaraju i prekidaju vezu s bazom podataka. Sad je baza spojena i projekt je spreman za upite prema bazi podataka (*Slika 10: Kod klase PostgreSqlService*).

[23] Slika zaslona iz praktičnog djela rada
[24] Slika zaslona iz praktičnog djela rada

```
3   import org.eclipse.scout.rt.platform.config.AbstractStringConfigProperty;
4
5   public class DatabaseProperties {
6     public static class URLProperty extends AbstractStringConfigProperty {
7
8       @Override
9       public String getKey() {
10        return "db.url";
11      }
12
13      @Override
14      public String description() {
15        return "DB Connection";
16      }
17    }
18
19    public static class UsernameProperty extends AbstractStringConfigProperty {
20
21      @Override
22      public String getKey() {
23        return "db.username";
24      }
25
26      @Override
27      public String description() {
28        return "DB username";
29      }
30    }
31
32    public static class PasswordProperty extends AbstractStringConfigProperty
33
34    {
35
36      @Override
37      public String getKey() {
38
39        return "db.password";
40      }
41
42      @Override
43      public String description() {
44
45        return "DB password";
46      }
47    }
48  }
```

Slika 9: kod klase *DatabaseProperties*[25]

[25] Slika zaslona iz praktičnog djela rada

```
import org.eclipse.scout.apps.budgetbuddy.server.database.DatabaseProperties.PasswordProperty;
import org.eclipse.scout.apps.budgetbuddy.server.database.DatabaseProperties.URLProperty;
import org.eclipse.scout.apps.budgetbuddy.server.database.DatabaseProperties.UsernameProperty;
import org.eclipse.scout.rt.platform.config.CONFIG;
import org.eclipse.scout.rt.server.jdbc.postgresql.AbstractPostgreSqlService;
import org.eclipse.scout.rt.server.jdbc.postgresql.PostgreSqlStyle;
import org.eclipse.scout.rt.server.jdbc.style.ISqlStyle;

public class PostgreSqlService extends AbstractPostgreSqlService {

  @Override
  protected Class<? extends ISqlStyle> getConfiguredSqlStyle() {
    return PostgreSqlStyle.class;
  }

  @Override
  protected String getConfiguredJdbcMappingName() {
    return CONFIG.getPropertyValue(DatabaseProperties.URLProperty.class);
  }

  @Override
  protected String getConfiguredPassword() {
    return CONFIG.getPropertyValue(PasswordProperty.class);
  }

  @Override
  protected String getConfiguredUsername() {
    return CONFIG.getPropertyValue(UsernameProperty.class);
  }

}
```

Slika 10: kod klase *PostgreSqlService*[26]

2.5.2. PG Admin

PGAdmin je besplatni alat za upravljanje i razvoj baza podataka PostgreSQL. Korišten je u ovom projektu kako bi olakšao administraciju baze podataka, omogućio brži razvoj i učinkovitije upravljanje podacima. Ovaj alat pruža korisne funkcionalnosti poput grafičkog sučelja za pregled i uređivanje podataka, upravljanje korisnicima i privilegijama, izvršavanje SQL upita, dizajniranje baza podataka putem grafičkog sučelja, uvoz i izvoz podataka, praćenje performansi baze podataka te upravljanje shemama i objektima baze podataka. [27]

[26] Slika zaslona iz praktičnog djela rada
[27] Mrežna službena stranica PgAdmin, https://www.pgadmin.org/ , [pristupljeno 14. svibnja 2024.]

3. Implementacija alata umjetne inteligencije

Za Implementaciju *Receipt & Invoice OCR AI* alata u samo programsko rješenje bilo je potrebno nekoliko koraka. Prvi korak bio je dobivanje API ključa od tvrtke *TAGGUN*, ključ inače nije besplatan, ali u kontaktu s tvrtkom omogućen je besplatni pristup za potrebe izrade završnog rada. API ključ je alfanumerički niz koji razvijatelji programskog rješenja koriste kako bi kontrolirali pristup svojem programskom rješenju, omogućavajući ograničavanje i praćenje pristupa njihovim funkcijama.[28] Nakon što je riješen prvi korak drugi je bio dodavanje svih potrebnih vanjskih biblioteka, pri čemu je korišten *Maven*, koji je već bio spomenut.

3.1. Vanjske biblioteke

Vanjske biblioteke koje su korištene pri implementaciji alata umjetne inteligencije u aplikacijsko rješenje su : Gson i OkHttp. Navedene vanjske biblioteke su bile potrebne kako bi na pravi način implementirao alat umjetne inteligencije u praktični dio rada.

```
31        <dependency>
32          <groupId>com.squareup.okhttp3</groupId>
33          <artifactId>okhttp</artifactId>
34          <version>4.9.3</version>
35        </dependency>
36
37        <dependency>
38          <groupId>com.google.code.gson</groupId>
39          <artifactId>gson</artifactId>
40          <version>2.8.8</version> <!-- Replace with the l
41        </dependency>
```

Slika 11: vanjske biblioteke[29]

[28] Amazon službena stranica, https://aws.amazon.com/what-is/api-key/, [pristupljeno 15.svibnja 2024.]
[29] Slika zaslona iz praktičnog djela rada

3.1.1. Gson biblioteka

Gson je vanjska biblioteka u vlasništvu *Google-a*, koja služi za pretvaranje Java objekta u JSON format, te pretvaranje JSON datoteke u Java objekt. U praktičnom dijelu rada ova vanjska biblioteka bila je potrebna kod pretvorbe pročitanih podataka iz nestrukturiranih izvora (račun) u željenu strukturu. Podaci bi došli iz OCR AI alata kao odgovor u JSON formatu, te bi se pretvorili u Java objekt kako bi se mogli obraditi do tražene strukture. [30]

3.1.2. OkHttp biblioteka

OkHttp je vanjska biblioteka tvrtke *Square* koja služi za HTTP komunikaciju u programskom jeziku Java. Omogućuje slanje zahtjeva na web poslužitelje i primanje odgovora i upravljanjem svim aspektima HTTP komunikacije. Koristila se za interakciju s API-jem od *Tagguna* preko ključa koji je izradila sama tvrtka.[31]

3.2. Praktični dio implementacije

Nakon što su zadovoljeni svi preduvjeti za implementaciju, napravljene su dvije klase, prva je nazvana *ReceiptAiOcr*, a drugu *ProcessReceipt.* Te dvije klase su bile ključne kod implementacije samog alata umjetne inteligencije jer su one bile zadužene za uspostavu komunikacije preko API ključa i spremanje u mapu treženih podataka sa skeniranog računa. Prva klasa *ReceiptAiOcr* ima metodu *processFile* koja prima kao argument objekt tipa *File.* U njoj je i uspostava veze s OCR AI alatom preko API ključa. Nakon što se uspješnom uspostavi veza s alatom, pozivala bi se druga metoda od klase *ProcessReceipt* imena *getReceiptData* i primala je kao argument odgovor od zahtjeva prema alatu koji je bio *JSON String*. Dalje, metoda *getReceiptData* pretvara JSON String pomoću Gson vanjske biblioteke u enkapsulirani objekt koji prima podatke o računu koji su očitani. Metoda nakon toga vraća u klijentski dio Mapu koja kao ključ ima naziv relevantnog podatka npr. *totalAmount* i kao vrijednost ima vrijednost koja je očitana pomoću alata umjetne inteligencije (Slike 12 i 13).

[30] Github gson biblioteka, https://github.com/google/gson, [pristupljeno 15.svibnja 2024.]
[31] Službena stranica Square tvrtke, https://square.github.io/okhttp/, [pristupljeno 15.svibnja 2024.]

```
12  public class ReceiptAiOcr {
13    public static Map<String, String> processFile(File file){
14      OkHttpClient client = new OkHttpClient();
15
16      Map<String, String> receiptMap = new HashMap<String, String>();
17
18      RequestBody body = new MultipartBody.Builder()
19        .setType(MultipartBody.FORM)
20        .addFormDataPart("file", "receipt.jpg", RequestBody.create(MediaType.parse("image/jpeg"), file))
21        .addFormDataPart("refresh", "false")
22        .addFormDataPart("incognito", "false")
23        .addFormDataPart("extractTime", "false")
24        .build();
25
26      Request request = new Request.Builder()
27        .url("https://api.taggun.io/api/receipt/v1/verbose/file")
28        .post(body)
29        .addHeader("accept", "application/json")
30        .addHeader("apikey", "aec05140d4c111eeb72409b0b60cbbed")
31        .build();
32
33      try {
34        Response response = client.newCall(request).execute();
35        if (response.isSuccessful()) {
36          String responseData = response.body().string();
37          receiptMap = ProcessReceipt.getReceiptData(responseData);
38
39        } else {
40          System.out.println("Error: " + response.code() + " " + response.message());
41        }
42      } catch (IOException e) {
43        throw new RuntimeException(e);
44      }
45
46      if (receiptMap == null) {
47        throw new RuntimeException("Failed to process the receipt");
48      }
49
50      return receiptMap;
51
52    }
53  }
```

Slika 12: metoda *processFile* klase *ReceiptAiOcr* [32]

[32] Slika zaslona iz praktičnog djela rada

```
8  public class ProcessReceipt {
9    public static Map<String, String> getReceiptData(String jsonString){
10
11     Gson gson = new Gson();
12     ReceiptData receiptData = gson.fromJson(jsonString, ReceiptData.class);
13
14     Map<String, String> receiptMap = new HashMap<>();
15     if (receiptData.merchantName != null) {
16       receiptMap.put("merchantName", receiptData.merchantName.data);
17     }
18     if (receiptData.merchantAddress != null) {
19       receiptMap.put("merchantAddress", receiptData.merchantAddress.data);
20     }
21     if (receiptData.totalAmount != null && receiptData.totalAmount.data != null) {
22       receiptMap.put("totalAmount", receiptData.totalAmount.data);
23     }
24     if (receiptData.taxAmount != null && receiptData.taxAmount.data != null) {
25       receiptMap.put("taxAmount", receiptData.taxAmount.data);
26     }
27     if (receiptData.date != null && receiptData.date.data != null) {
28       receiptMap.put("date", receiptData.date.data);
29     }
30
31     return receiptMap;
32
33   }
34
```

Slika 13: metoda *getReceiptData* klase *ProcessReceipt*[33]

Nakon definiranja ove dvije klase i njihove funkcionalnosti u samoj implementaciji, na klijentskom dijelu dodan je menu pod nazivom *Dodaj pomoću AI OCR* na podmodulu *Računi*. Završni dio implementacije u kodu se upravo nalazi u formi koja se otvara iz tog menija i to u *BillField* klasi koja proširuje *AbstractFileChooserField*. Klasa sadrži metodu za ograničavanje učitavanja datoteka na: pdf, jpg i png, kao što je i ranije navedeno u funkcionalnim zahtjevima. Također *BillField* klasa sadrži metodu *execChangedValue()* koja se izvršava nakon što se datoteka učita na polje (Slika 14 i 15).

[33] Slika zaslona iz praktičnog djela rada

```
105             @Order(1000)
106             public class BillField extends AbstractFileChooserField {
107                 @Override
108                 protected String getConfiguredLabel() {
109                     return TEXTS.get("Bill0");
110                 }
111 
112                 @Override
113                 protected java.util.List<String> getConfiguredFileExtensions() {
114                     return CollectionUtility.arrayList("pdf","jpg", "png");
115                 }
116                 protected void execChangedValue() {
117 
118                     if(getBillField().isEmpty()) {
119                         return;
120                     }
121                     getNameField().resetValue();
122                     getDateField().resetValue();
123                     getAmountField().resetValue();
124                     getAddressField().resetValue();
125                     getTaxAmountField().resetValue();
126 
127                     BinaryResource file = getBillField().getValue();
128 
129                     try {
130 
131                         File file2 = createFileFromBinaryResource(file, "bill");
132                         Map<String,String> receiptMap  = ReceiptAiOcr.processFile(file2);
133 
```

Slika 14: unutarnja klasa *BillField* [34]

Učitana datoteka s polja za učitavanje nestrukturiranog izvora se proslijedi kao *BinaryResource* metodi *createFileFromBinaryResource* koja vraća objekt tipa *File* iz učitanog objekta. Nakon što se dobije objekt tipa *File* poziva se metoda *processFile* klase *ReceiptAiOcr* koja prima taj objekt. Dalje, se odvijaju isti procesi kao što su navedeni u opisu napravljenih klasa. Metoda na kraju vraća mapu *stringova* s imenom polja kao ključem i vrijednosti. Kad se dobiju traženi podatci s računa nakon obrade alata, postavljaju se na odgovarajuća polja s odgovarajućim vrijednostima. Jednostavnim uvjetnim naredbama ispitujemo da li u mapi koju nam je vratila metoda *processFile* s očitanim podacima postoje traženi podaci za odgovarajuća polja, ako postoje, postavljaju se na odgovarajuće polje. Ako ne postoji ni jedan podatak dobijamo poruku da nije pronađen niti jedan relevantan podataka, te da probamo skenirati drugu datoteku. Ako su podaci dobro očitani i ne želimo ništa mjenjati, nakon klika *Ok* na formi podaci se spremaju u bazu i prikazuju u tablici.

[34] Slika zaslona iz praktičnog djela rada

```
134                     try {
135
136                         File file2 = createFileFromBinaryResource(file, "bill");
137                         Map<String,String> receiptMap  = ReceiptAiOcr.processFile(file2);
138
139                         if(receiptMap.get("merchantName") != null) {
140                             String merchantName = receiptMap.get("merchantName");
141                             getNameField().setValue(merchantName);
142                         }
143
144                         if(receiptMap.get("merchantAddress") != null) {
145                             String merchantAddress = receiptMap.get("merchantAddress");
146                             getAddressField().setValue(merchantAddress);
147                         }
148
149                         if(receiptMap.get("taxAmount") != null) {
150                             Double taxAmountDo = Double.parseDouble(receiptMap.get("taxAmount"));
151                             BigDecimal taxAmount = BigDecimal.valueOf(taxAmountDo);
152                             getTaxAmountField().setValue(taxAmount);
153                         }
154
155
156                         if(receiptMap.get("date") != null) {
157                             String dateString = receiptMap.get("date");
158                             OffsetDateTime dateTime = OffsetDateTime.parse(dateString);
159                             Date date = new Date();
160                             date.setTime(dateTime.toInstant().toEpochMilli());
161                             getDateField().setValue(date);
162                         }
163
164                         if(receiptMap.get("totalAmount")!= null) {
165                             Double totalAmountDouble = Double.parseDouble(receiptMap.get("totalAmount"));
166                             BigDecimal totalAmount = BigDecimal.valueOf(totalAmountDouble);
167                             getAmountField().setValue(totalAmount);
168                         }
---
```

Slika 15: drugi dio klase *BillField* [35]

Spremanje podataka o novom računu u bazu podataka se odvija u unutarnjoj kasi *NewHandler* iste forme. Klasa *NewHandler* služi za dohvaćanje metode za spremanje novih podataka u bazu sa *.server* strane aplikacije, ima metodu *execStore()* koja enkapsulirani objekt podataka s forme obično naziva *formData* šalje u *.server* dio aplikacije. Poziva *create()* metodu sučelja *IAddBillAiOcrService* koji je impelementiran u klasi *AddBillAiOcrService* i prosljeđuje mu taj enkapsulirani objekt iz forme u kojem se nalaze svi podaci. *AddBillAiOcrService* klasa ima implementiranu metodu *create()* u kojoj ima napisan SQL *insert* koji se izvršava i tako sprema novi račun s mapiranim podacima s forme u bazu podataka (Slike 16 i 17).

[35] Slika zaslona iz praktičnog djela rada

```
360
361     public class NewHandler extends AbstractFormHandler {
362         @Override
363         protected void execLoad() {
364             AddBillAiOcrFormData formData = new AddBillAiOcrFormData();
365             exportFormData(formData);
366             formData = BEANS.get(IAddBillAiOcrService.class).prepareCreate(formData);
367             importFormData(formData);
368
369         }
370
371         @Override
372         protected void execStore() {
373             AddBillAiOcrFormData formData = new AddBillAiOcrFormData();
374             exportFormData(formData);
375             formData = BEANS.get(IAddBillAiOcrService.class).create(formData);
376             importFormData(formData);
377         }
378     }
379
380     public class ModifyHandler extends AbstractFormHandler {
```

Slika 16: unutarnja klasa *newHandler*[36]

```
12  public class AddBillAiOcrService implements IAddBillAiOcrService {
13      @Override
14      public AddBillAiOcrFormData prepareCreate(AddBillAiOcrFormData formData) {
15
16          return formData;
17      }
18
19      @Override
20      public AddBillAiOcrFormData create(AddBillAiOcrFormData formData) {
21
22          String stmt = "INSERT INTO bills (name, address, taxamount, price, date, wallet_id) VALUES (:Name, :Address, :TaxAmount, :Amount, :Date, :Wallet)";
23          SQL.insert(stmt, formData);
24          String stmt1 = "Update wallet set balance = balance - :Amount, last_used = now() where id = :Wallet";
25          SQL.update(stmt1, formData);
26          return formData;
27      }
28
29
```

Slika 17: metoda *create()* klase *AddBillAIOcrService*[37]

[36] Slika zaslona iz praktičnog djela rada
[37] Slika zaslona iz praktičnog djela rada

3.3. Taggun OCR AI alat

Receipt & Invoice OCR je napredni OCR alat tvrtke *Taggun* koji koristi umjetnu inteligenciju, dizajniran je za automatsko skeniranje i izdvajanje podataka s računa. OCR tehnologija skenira slike računa i digitalizira podatke u strukturirane i smislene podatke koje aplikacija može obraditi i koje su joj potrebne. Najčešće izdvojeni podaci uključuju ukupni iznos računa, iznos poreza računa, datum i naziv trgovca, te mnoge druge. Također alat koristi algoritme umjetne inteligencije za visoku točnost koje ovom alatu prelazi 90%[38], rezultate pruža u stvarnom vremenu i podržava više jezika te je kompaktibilan za Google Vision i Microsoft Services API-ima. Sve ove karakteristike ovog alata čine ga idealnim za implementaciju za aplikacije kao što je upravljanje troškovima.[39]

Slika 18: logo Taggun AI OCR alata

3.4. Tehnički detalji implementacije

Nakon implementacije alata umjetne inteligencije od *Tagguna* rješenje je testirano na više različitih primjera računa. Obrada računa traje od 2 do 5 sekundi ovisi o kompleksnosti računa, a točnost skeniranih podataka je preko 90% kako navodu iz *Tagguna*, što se može i potvrditi samim testiranjem koje je provedeno nakon završetka implementacije. Ni u jednom slučaju rješenje nije očitalo krive podatke stoga njihov alat možemo sa sigurnošću reći da ima postotak točnosti koji je naveden. Možda kad bi se skeniralo više tisuća slika računa bi u nekim slučajevima dao pogrešan rezultat ali okvir postotka točnosti je ispravan. Također nisam imao mogućnost provesti takav test zbog ograničenja kojeg su mi dali, 50 skeniranja mjesečno. U samoj implemantaciji su još obrađene neke provjere točnosti skeniranih podataka, formata datuma, pa je ta točnost podataka još i veća jer u protivnom aplikacija će nas upozoriti da datum nije valjan. Kompletan kod aplikacije dostupan je s javnog repozitorija na GitHubu i nalazi se

[38] Službena stranica Taggun OCR AI alata, https://www.taggun.io/#:~:text=Multilingual%20OCR%20Capability,for%20a%20chat., [pristupljeno 28.svibnja 2024.]
[39] Službena stranica Taggun OCR AI alata, https://www.taggun.io/, [pristupljeno 21.svibnja 2024.]

na poveznici : https://github.com/dinobaresic/Zavrsni_rad_IT_2024. U datoteci README.md na repozitoriju nalazi se kratki opis aplikacije s korištenim alatima.

3.4.1. Unos podataka računa i pohrana u bazu podataka

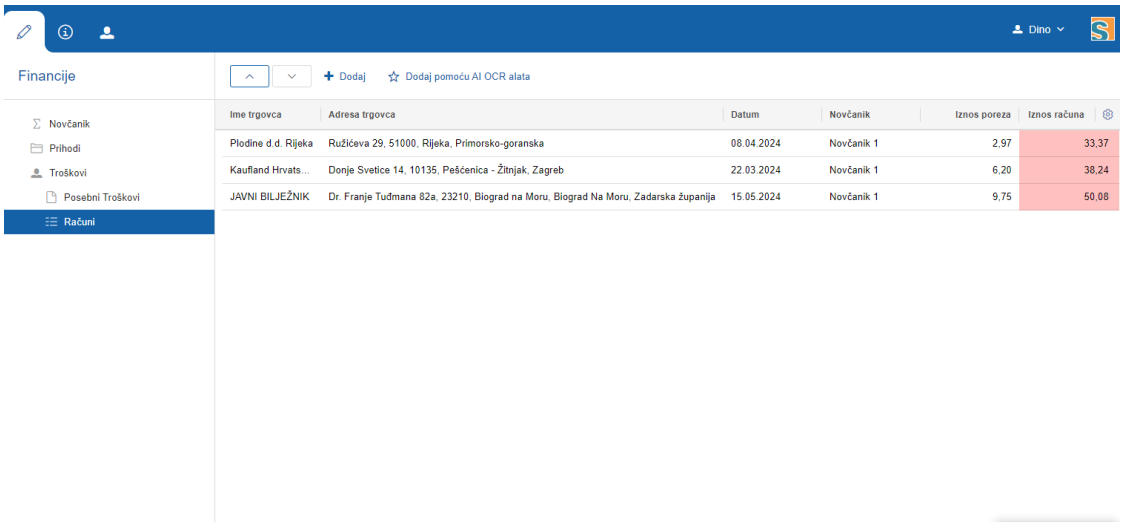
Forma za unos novog računa se otvara iz podmodula *Računi*, iz menija koji se zove *Dodaj pomoću AI OCR*. Kada se odabere slika pomoću alata umjetne inteligencije prepoznaju se traženi podaci kao što su: ime trgovca, adrese trgovca, iznos računa, iznos poreza i datum. Podaci se spremaju na polja na formi, moguće ih je još izmjeniti ili prihvatiti prijedlog koji je dao OCR AI alat, potrebno je još odabrati novčanik na koji želimo dodati trošak. Nakon klika *Ok* strukturirani podaci se spremaju u bazu podataka. Prikaz podataka u bazi vidljiv je na slici 19. Svaki trošak ima vanjski ključ wallet_id u bazi podataka kako bi se raspoznalo koji je trošak iz kojeg novčanika.

	id [PK] bigint	name character varying	price numeric (9,2)	date date	taxamount numeric (9,2)	address character varying	is_deleted boolean	deleted_at timestamp without time zone	wallet_id bigint
1	56	JAVNI BILJEŽNIK	50.08	2024-05-15	9.75	Dr. Franje Tuđmana 82a, 23210, Biograd na Moru, Biograd Na Moru, Zadarska župan...	false	[null]	3
2	53	Plodine d.d. Rijeka	33.37	2024-04-08	2.97	Ružićeva 29, 51000, Rijeka, Primorsko-goranska	false	[null]	3
3	52	Kaufland Hrvatska k.d.	38.24	2024-03-22	6.20	Donje Svetice 14, 10135, Peščenica - Žitnjak, Zagreb	false	[null]	3

Slika 19: prikaz unesenih podataka o računima u bazi podataka[40]

Nakon što se podaci spremljeni u bazu podataka oni su vidljivi u tablici u modulu *Računi*, slika 20 prikazuje tablični prikaz podataka o učitanim računima.

[40] Slika zaslona iz praktičnog djela rada



Slika 20: tablični prikaz podataka o računima[41]

[41] Slika zaslona iz praktičnog djela rada

4. Zaključak

Implementacija alata umjetne inteligencije, poput *Receipt & Invoice OCR* od Tagguna, za automatsku analizu i pohranu podataka iz nesutrukturiranih izvora, primjer je implementacije doba na svitanju korištenja AI alata. Sve češća je potreba za korištenjem umjetne inteligencije u poslovanju i automatizaciji procesa poslovanja kao što je u primjeru ovog rada. Ovaj alat značajno poboljšava funkcionalnost aplikacije ubrzavanjem obrade podataka iz nestrukturiranih izvora, što je često problem jednog poslovnog sustava. Alat smanjuje potrebu za ručnim unosom i provjerom podataka, te pruža pouzdano rješenje za upravljanje troškovima i planiranjem financijskih strategija. Implementacija ovog OCR AI alata imala je izazova kao što su korištenje valjskih biblioteka u samom uspostavljanju veze sa API alata, u pretvorbi očitanih podataka u podatke koje su prepoznatljive samom programskom jeziku. Također jedan od izazova je bio i pretvorba i usklađivanje tipova objekta s kojima rade polja za čitanje datoteka Scout aplikacija i onog tipa objekta koji se slao na obradu ovom OCR AI alatu. Cilj ovog rada je bio prikazati mogućnosti primjena tehnologija koje koriste umjetnu inteligenciju u poslovnoj aplikaciji. Ovo programsko rješenje omogućuje korisnicima učinkovito upravljanje financijama i povećava produktivnost. Implementacija ovog alata također omogućava brži unos podataka koji služe za analizu financijskih podataka, što rezultira donošenjem boljih odluka za samog korisnika. Tema implementacije umjetne inteligencije za automatsku analizu i pohranu podataka iz nestrukturitranih izvora uspješno je obrađena u teorijskom djelu rada i primjenjena u praktičnom djelu rada, postignut je željeni razultat, implementiran je OCR AI alat koji ubrzava unos podataka iz nestrukturiranih izvora. Budući radovi mogu se fokusirati na istraživanje drugih AI alata, i implementaciju sličnih u različite poslovne sustave, što bi dodatno unaprijedilo poslovne procese ovakvih aplikacija.

implementation of artificial intelligence tool for automatic analysis and data storage from unstructured sources

Summary

This paper aims to research the technologies and the approach to implementation of AI tool for the analysis and storage of unstructured data. The technologies which will be used in the practical part, as well as researched in this paper are as follows: Java programming language, Eclipse Scout framework, PostgreSQL database and PG Admin database administrator. The tool being implemented in the application is Receipt & Invoice OCR, developed by the company *Taggun*. The external libraries used in the practical part of the paper are: JDBC, Gson and okHttp. The external libraries mentioned were necessary for connection to the database and the implementation of the AI OCR tool. In the first part, the purpose and the goal of the paper is explained, after that, the functional requirements for the development of the application and the implementation of the tool. The second part aims to analyze the technologies used and the theoretical terms which are related to the AI OCR tool. The last part of the paper focuses on the practical part of the implementation of the AI tools for recognizing and structuring data from unstructured sources and the theoretical part of the external libraries used. The complete application code is available from the public repository on GitHub and can be found at the link: https://github.com/dinobaresic/Zavrsni_rad_IT_2024.

Key words: artificial intelligence, Eclipse Scout, PostgreSQL, Java programming language, AI, OCR

Popis literature

[1] J. Fain, *Java programming*, 2015.

[2] BSI, *BSI Software*, 2024., Dohvaćeno iz: https://www.bsi-lifesciences.com/about-bsi.

[3] Eclipse, *Eclipse Scout Documentation*, 2024., Dohvaćeno iz: https://eclipsescout.github.io/scout-docs/24.1/getstarted/helloscout.html.

[4] E. Foundation, *Eclipse Scout*, 2024., Dohvaćeno iz: https://eclipse.dev/scout/features.html.

[5] PostgreSQL, *PostgreSQL*, 2024., Dohvaćeno iz: https://www.postgresql.org/.

[6] Maven, *Maven repository*, 2024., Dohvaćeno iz: https://maven.apache.org/what-is-maven.

[7] PGAdmin, *PG Admin*, 2024., Dohvaćeno iz: https://www.pgadmin.org/.

[8] Amazon, *Amazon web pages*, 2024., Dohvaćeno iz: https://aws.amazon.com/what-is/java/, https://aws.amazon.com/what-is/api-key/.

[9] Square, *okHttp*, 2024., Dohvaćeno iz: https://square.github.io/okhttp/.

[10] Google, *Gson*, 2024., Dohvaćeno iz: https://github.com/google/gson.

[11] TAGGUN, *Taggun*, 2024., Dohvaćeno iz: https://www.taggun.io/.

[12] T. Breuel, *OCR and Scene Text*, 2024., Dohvaćeno iz: http://www.tmbdev.net/projects/ocr/.

[13] 6Sense, *Relational Databases*, 2024., Dohvaćeno iz: https://6sense.com/tech/relational-databases.

[14] P. N. Stuart Russel, *Artificial Intelligence : A modern approach (4.ed)*, 2020., pp. 1-5.

[15] A. Vaswani, *Attention is all you need*, 2017., pp. 1-10.

[16] E. Alpaydın, *Machine Learning : The new AI*, 2016.

[17] IBM, *OCR*, 2024., Dohvaćeno iz: https://www.ibm.com/blog/optical-character-recognition/.

[18] M. Hajdarović, *Umjetna inteligencija, ChatGPT i poučavanje Povijesti*, 2023., Dohvaćeno iz: https://hrcak.srce.hr/clanak/437453.

[19] N. Soni i S. Enakshi, *Artificial Intelligence in Business : From Research and Inovation to Market Deployment*, 2020., Dohvaćeno iz: https://www.sciencedirect.com/science/article/pii/S1877050920307389.