Sustav za obradu podataka knjižnice

Gibas, Antonio

VELEUČILIŠTE U BJELOVARU
STRUČNI PRIJEDIPLOMSKI STUDIJ MEHATRONIKA

SUSTAV ZA OBRADU PODATAKA KNJIŽNICE

Završni rad br. 11/MEH/2023

Antonio Gibas

Bjelovar, listopad 2024.

obrazac ZR - 001

Veleučilište u Bjelovaru

Trg E. Kvaternika 4, Bjelovar

1. DEFINIRANJE TEME ZAVRŠNOG RADA I POVJERENSTVA

Student: **Antonio Gibas** JMBAG: **0314019051**

Naslov rada (tema): **Sustav za obradu podataka knjižnice**

Područje: **Tehničke znanosti** Polje: **Računarstvo**

Grana: **Programsko inženjerstvo**

Mentor: **Tomislav Adamović, mag. ing. el.** zvanje: **viši predavač**

Članovi Povjerenstva za ocjenjivanje i obranu završnog rada:

1. **dr. sc. Zoran Vrhovski, predsjednik**
2. **Tomislav Adamović, mag. ing. el., mentor**
3. **Krunoslav Husak, dipl. ing. rač., član**

2. ZADATAK ZAVRŠNOG RADA BROJ: 11/MEH/2023

U sklopu završnog rada potrebno je:

1. Izraditi tablice za obradu podataka knjižnice u bazi podataka Db2
2. Napisati programsku logiku za obardu podataka u programskom jeziku PL/I
3. Izraditi ekrane za obradu podataka knjižnice koristeći CICS (Customer Information Control System)
4. Koristiti JCL (Job Control Language) za upravljanje različitim aspektima PL/I programa za obradu podataka knjižnice.
5. Izraditi izvoz podataka za dnevnu posudbu knjiga iz baze za obradu podataka knjižnice

Datum: 15.12.2023. godine

Mentor: **Tomislav Adamović, mag. ing. el.**

REPUBLIKA HRVATSKA BJELOVAR VELEUČILIŠTE U BJELOVARU

Zahvala

Zahvaljujem svojim roditeljima na strpljivom iščekivanju završetka ovog završnog rada. Ponajviše sam zahvalan mentoru, profesoru Tomislavu Adamoviću, kao i prodekanu za nastavu i studente dr. sc. Zoranu Vrhovskom na njegovom mentorstvu. Također, veoma sam zahvalan svojim kolegama iz CROZ Mainframe tima: Dejanu Cepetiću, Ivi Štampaliji, Filipu Đuričkoviću, Marku Jurmanu, Dejanu Stamatoviću i Filipu Forjanu, na podršci, prijenosu znanja i bodrenju tokom izrade ovog završnog rada.

Sadržaj

1. Uvod

Knjižnice predstavljaju nezamjenjive institucije za pristup znanju, informacijama i literaturi, no često se suočavaju s izazovima upravljanja velikih količina knjiga i ostalih fizičkih informacijskih medija. Tradicionalni sustavi za obradu podataka u knjižnicama mogu biti spori i nepouzdani, što otežava učinkovito upravljanje zbirkama, posudbama te informacijama o korisnicima. Svrha ovog završnog rada je implementacija sustava za obradu podataka knjižnice koristeći PL/I (engl. Programming Language One) programski jezik s CICS-om (engl. Customer Information Control System - CICS) te Db2 bazu podataka za pohranu knjižničnih podataka. Učinkovit sustav za obradu podataka i sama brzina sustava su presudne u kvaliteti poslovanja svake institucije čiji je glavni cilj optimizirati svoje poslovanje i usluge korisnicima.

Struktura rada je podijeljena na generalni tehnološko opisni uvod uz opis poslovne logike, uz opis svake korištene tehnologije i metodologije te osnovni razlog odabira ovakvog načina implementacije u poglavljima „Uvod“, „Sustav za obradu podataka knjižnice“. Popratno opisima, ovaj rad će ići u detaljniji opis implementacije relevantnih navedenih teorijskih koncepta korištene tehnologije na praktičnom primjeru primjene u poglavlju „Proces izrade sistema“.

2. SUSTAV ZA OBRADU PODATAKA KNJIŽNICE

Sustav za obradu podataka knjižnice osmišljen je korištenjem CICS ekrana, prikazanih na terminalu 3270, za reprezentaciju poslovne logike. Svaki ekran kreiran je pomoću CICS Asembler makro jezika, koji se piše planski prema potrebama izgleda sučelja za interakciju s korisnikom. Kompilirani Asembler kod tvori modul, koji kada pozvan unutar CICS se izvodi CICS u Aplikacijskoj regiji sustava (engl. Application Owning Region - AOR) [1]. Taj isti modul se naziva „fizička mapa“. Fizička mapa dolazi u enkodiranom formatu, tipa EBCDIC (engl. Extended Binary Coded Decimal Interchange Code - EBCDIC) [2]. Također, uz fizičku mapu, automatski je kreirana i simbolička mapa. Simbolička mapa je jezično-specifična podatkovna struktura, reprezentirana varijablama i tipovima podataka programskog jezika u kojem kodiramo poslovnu logiku i CICS naredbe [2]. Preko varijabli simboličke mape se prenose podaci korisniku na terminal, prosljeđuju podaci između transakcija i spremaju u bazu podataka.

Korisnička interakcija na svakom setu mapa je uvjetovana povratnom logikom CICS sustava programu koji se izvodi u transakciji. Sve transakcije u ovom radu su uvjetovane naredbama korisnika te iste unutar transakcije ostavljaju povratni kod CICS-u. Podatkovno polje koje skuplja informacije o samom izvođenju transakcije se naziva izvršni blok sučelja (engl. Execute Interface Block - EIB). Aplikacijski program pristupa izvršnom blokom sučelja tijekom izvođenja transakcije [3]. Ovisno o EIB-u, aplikacijska logika se baždari po poželjnom i nepoželjnom korisničkom unosu. Tipke na tipkovnici imaju svoj simbol definiran na kodnoj stranici samog z/OS operacijskog sustava te njihov izlazni povratni kod služi za uvjetovanje izvođenja transakcije. Korisnik prilikom rada na sustavu, ključnom riječi inicira transakciju, koju prikazuje izbornik. Iz tog izbornika korisnik može odabrati poželjnu poslovnu logiku koja mu zatreba u tom trenutku odvijanja posla. Svaki korisnički upit o prikazu inventara knjiga, narudžbi, dobavljača, članova i bivših članova šalje upit preko CICS-a na Db2 bazu podatka. Upit se odvija preko Db2 adresnog prostora za komunikaciju sa servisima [4]. Logika slanja, dohvata i ažuriranja podataka iz Db2 baze podataka, realizirana je u samom PL/I programu putem SQL upita. SQL upite u ovakvom sustavu izvršava CICS Db2 *Attachment Facility* koji je pokrenut kada god aplikacijski program unutar transakcije izvrši SQL upit [5].

Ovakvim pristupom korisniku je omogućen nesmetan pristup svim informacijama o raspolaganju poslovanja knjižnice kao ustanove.

2.1 Poslovna logika

U radu knjižnice istaknuo je nekoliko ključnih operacija potrebnih za njezino učinkovito funkcioniranje.

Među najvažnijima su:

1. Upravljanje članovima
    - Učlanjenje i registracija novih korisnika
    - Ažuriranje podataka korisnika
    - Upravljanje članarinama
    - Ažuriranje podataka korisnika
    - Ukidanje članarine korisnika
2. Upravljanje knjižnim fondom

- Unos novih knjiga
- Ažuriranje podataka knjiga
- Katalogizacija knjiga prema temama, žanru i drugim kriterijima

3. Proces posudbe i vraćanja knjiga
    - Posudba knjiga
    - Vraćanje knjiga
    - Produljenje posudbe
    - Rezervacija posudbe knjiga
4. Nabava knjiga
    - Odabir i narudžba
    - Evidencija nabave
    - Unos podataka nabavljača
5. Izvještaji
    - Izvještaji o posudbama
    - Izvještaji o korisnicima
    - Inventura
    - Analiza potreba

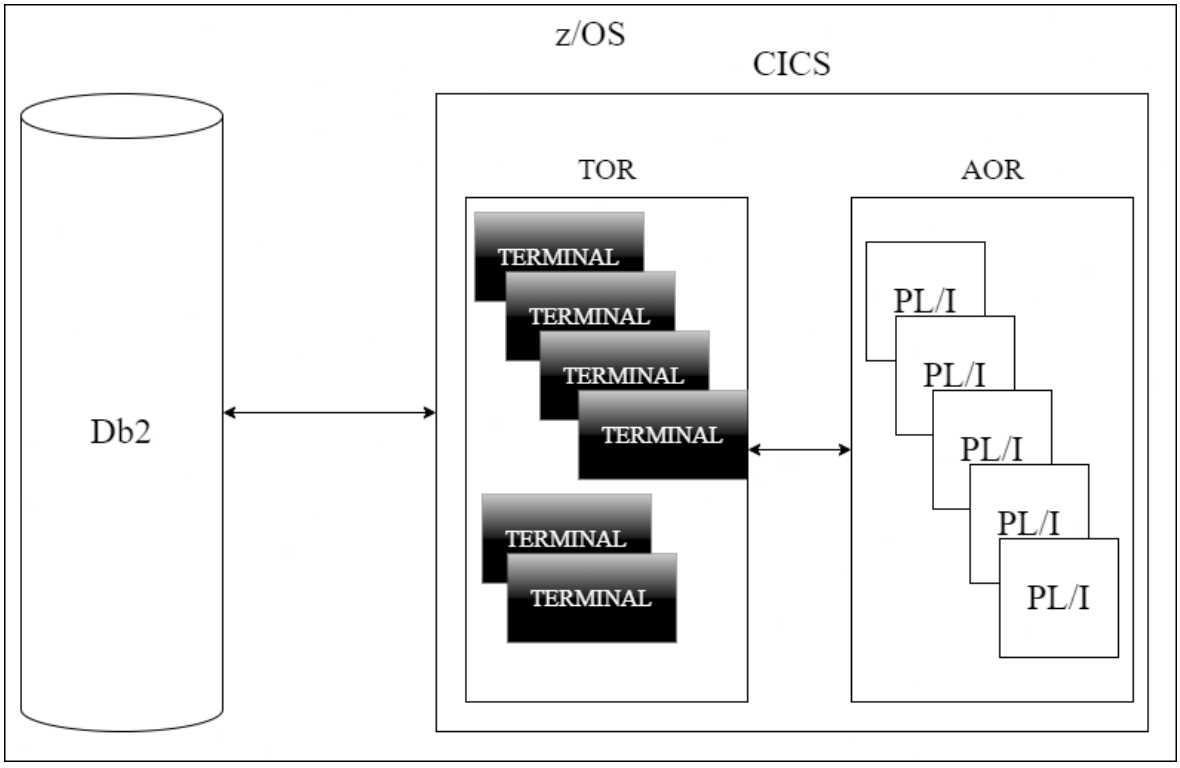
Većina temeljnih poslovnih potreba rada knjižnice može se realizirati implementacijom baze podataka, čije tablice su korespondentne s navedenim poslovnim logikama. Navedene poslovne logike se implementiraju tako da ostane mjesta za dodatne izmjene u budućnosti te je prilagodljiva realističnijim potrebama rada.

3. OPIS TEHNOLOGIJA KORIŠTENIH U IMPLEMENTACIJI

Implementacija sustava za obradu podataka knjižnice koristi niz tehnologija koje omogućuju efikasnu i sigurnu obradu podataka, uz korisničko predznanje kako koristiti dizajnirani sustav pri svom radu. U ovom poglavlju detaljno je opisana korištena tehnologija u dizajnu i implementaciji sustava, uključujući CICS [6], Db2 bazu podataka [7] i programski jezik PL/I [8], na operacijskom sustavu z/OS [9].

Nadvedene tehnologije za sustav knjižnice odabrane su zbog brzine i pouzdanosti, dok u isto vrijeme zahtijevaju detaljno poznavanje funkcija rada na sustavu i ne pružaju dobru korisničku interaktivnost. Prikaz arhitekture komponenata z/OS sustava opisan je na slici 2.1. Zbog tih razloga, CICS terminalni ekrani izrađeni su da pružaju samo-objašnjivu funkcionalnost sa strane korisničke interakcije.

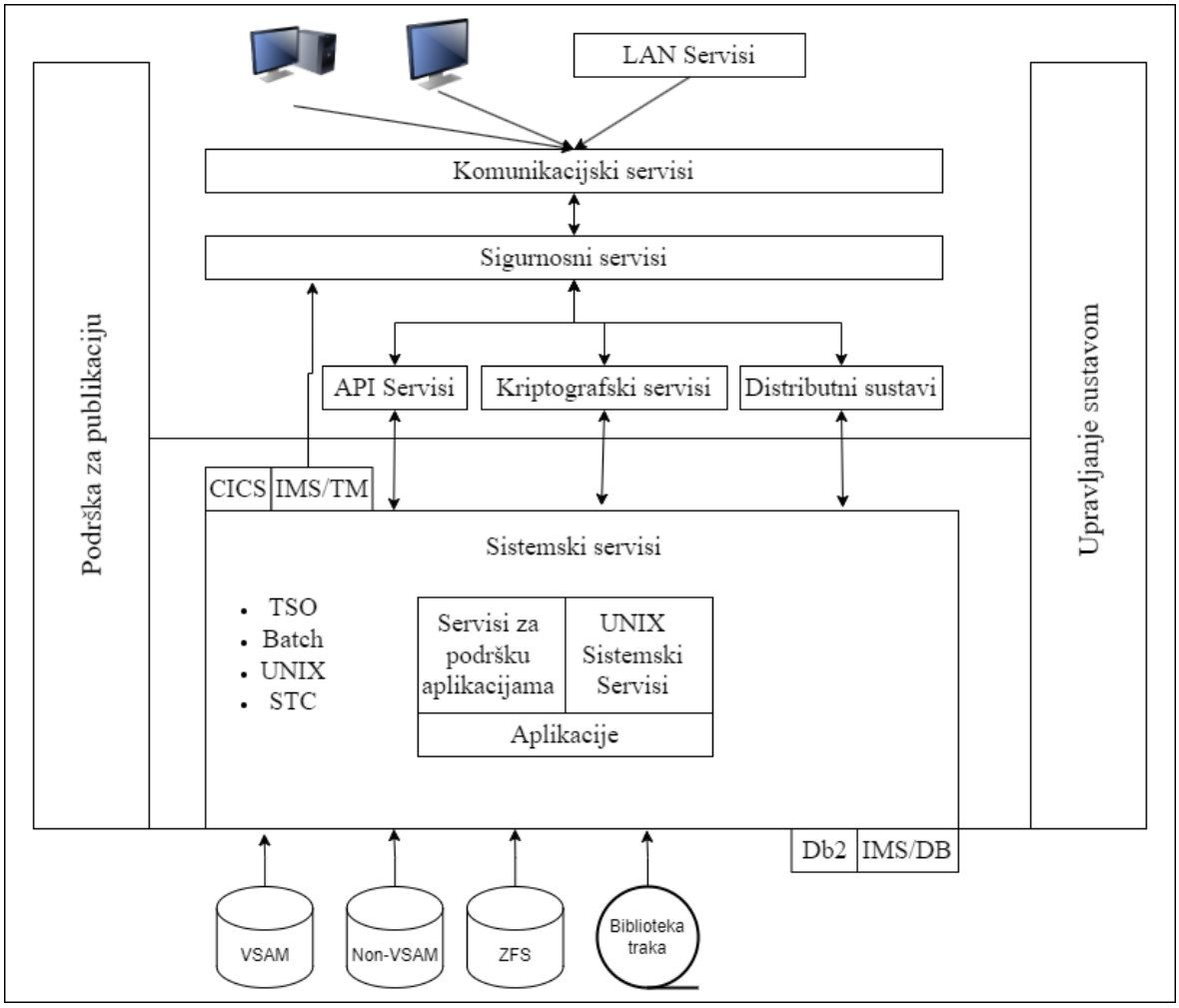
Prikaz arhitekture završnog rada opisan je pomoću veza tehnologija bez poslovne logike. U slici ispod, prikazan je međuodnos komponenti na z/OS operacijskom sustavu koje omogućuju korisničku komunikaciju s bazom podataka preko CICS-a.



Slika 2.1 - Vizualizacija arhitekture komponenata završnog rada.

3.1 Operacijski sustav z/OS

Enterprise z/OS je skalabilno – pouzdan operacijski sustav koji se primarno koristi za rad na IBM *Mainframe* računalima. IBM Z je operacijski sustav dizajniran za kontinuirano odvijanje poslovne logike i transakcija u svijetu [10]. Prvo izdanje softwarea izdala je kompanija IBM u Prosincu 2000. godine [11], za „*Z*“ seriju *Mainframe* računala. Operacijski sustav z/OS podržava rad s velikim brojem istovremenih korisnika, uz obradu velikih količina podataka s visokom učinkovitošću. Takva svojstva čine ga neophodnim u institucijama kao što su banke, bolnice, poštanski servisi, ministarstva, vojske i sl. Arhitektura z/OS operacijskog sustava prikazana je na slici 3.1.



Slika 1.1 Arhitektura z/OS operacijskog sustava

Operacijski sustav se pokreće u logičkoj particiji (engl. Logical partition - LPAR) u *Mainframe-u*. LPAR je određeni podskup *Mainframe* resursa na jednom sistemu. Moguće

je imati do 85 aktivnih logičkih particija, svaka s različitim definicijama i konfiguracijama operacijskih sustava [10]. Hardver *Mainframe* računala sastoji se od procesora i uređaja za zapis podataka (diskovi, trake, DASD), magnetni diskovi i više raznih vrsta korisničkih konzola za rad i upravljanje [12]. Operacijski sustav koristi procesorsku memoriju (engl. Random Access Memory - RAM) tijekom izvođenja kako bi obavljao svoje funkcije i upravljao resursima sustava [12].

Najbitniji dijelovi operacijskog sustava z/OS su: sistemski servisi, komunikacijski servisi, aplikacijski servisi, servisi za podršku aplikacijama, distribuirani sustavi, sigurnosni kriptografski servisi i podrška za mrežu. Sistemski servisi služe kao potpora za cijeli operacijski sustav, u njima se odvija aplikacijski sloj, *UNIX* servisi, skupne obrade i podrške za komunikaciju s bazama. Aplikacijski servisi služe za kontrolu kreiranja, izvođenja i obradu svih sistemskih i novih aplikacija. Komunikacijski servisi omogućuju umreženost svim korisnicima, operatorima, programerima na sustavu. Sam sustav podržava velik broj korisnika koji rade na svojoj instanci sučelja sustava. Po svakoj z/OS *Mainframe* instalaciji, sustav podržava istovremeni rad s nekoliko stotina do nekoliko tisuća korisnika istovremeno, ovisno o hardverskim specifikacijama [9]. Distributni servisi omogućuju podršku za pristupanje distribuiranim aplikacijama pomoću DFS-a. Sigurnosni servisi osiguravaju čitav z/OS sustav, svaki korisnik na operacijskom ustavu ima svoj određeni set prava rada na istom. Servisi sa slike 3.1, omogućuju z/OS operacijskom sustavu neprekidni rad i visoku dostupnost 24 sata u danu. Sve aplikacije na sustavu rade nesmetano, što je esencijalno organizacijama koje ovise o kontinuiranim obradama i dostupnosti podataka [9]. Upravljanje sustavom obuhvaća korištenje alata za pregled spool[1]-a i administraciju.

Kroz alate kao što su SDSF, JES i RACF, sistemski administratori mogu osigurati stabilan i učinkovit rad z/OS operacijskog sustava, brzo reagirati na potencijalne probleme i prilagodbu sustava prema zahtjevima posla ili korisnika. Arhitektura, sigurnosne značajke, integracija s drugim servisima i alati za administraciju čine z/OS potpuno neophodnim za mnoge organizacije koje zahtijevaju stabilno i sigurno odvijanje posla na sistemu.

3.2 Job Control Language

Job Control Language (JCL), je specijalizirani skriptni jezik dizajniran za pokretanje i odvijanje posla (skripti) u z/OS-u. Ovaj jezik omogućuje definiranje serije instrukcija koje

[1] Spool – Prostor SDSF-a za pregled aktivnih i neaktivnih programa

operacijskom sustavu preciziraju koje zadatke treba izvršiti i s kojim resursima [13]. JCL pruža niz instrukcija operacijskom sustavu, specificirajući kako obraditi ulazne podatke, gdje pohraniti izlazne podatke, koje programe pokrenuti i kako upravljati ispisom. [13].

Sintaksa JCL-a slijedi stroga, pozicijski definirana pravila. Primjer JCL kartice posla je prikazan u programskom kodu 3.1: Kartica posla. JCL naredba počinje s dvije kose crte ("//"). Početak JCL-a naziva se *JOB* kartica. Prvi izraz je *JOB* naredba koja definira početak posla. Naziv posla može biti sistemski ili korisnički definiran, nakon čega slijedi niz parametara koji konfiguriraju različite postavke posla. Naslov izraza popraćen je naredbom, u slučaju prvog izraza to je *JOB* naredba, koja definira početak izraza. Nakon *JOB* naredbe dolazi ključna riječ, ona može biti naslov ili informacija o samoj skripti. Parametri koje je poželjno definirati su *MSGCLASS*, klasa izlazne informacije o *JCL* poslu, određuje na što ide ispis samog posla. Startna opcija *MSGLEVEL*, definira razinu izlaznih poruka na spool-u. Parametar *NOTIFY* služi kao upozorenje korisniku kako se odvio njegov JCL.

Programski kod 3.1 - Kartica posla

```
//JCLJOB JOB (KNJIZNICA),MSGCLASS=X,MSGLEVEL=(1,1),
//                       NOTIFY=&SYSUID
```

Vrijednost parametra za klasu poruke „X“, simbolizira ispis u spool. Vrijednost parametra *MSGLEVEL* simbolizira od koje vrste informacija o izvršenom poslu želimo na ispisu u spoolu, u ovom slučaju, parametar 1 simbolizira da će biti ispisani svi JCL izrazi iz skripte, svi kontrolni izrazi, navedene procedure (ako ih ima), dok parametar 2 simbolizira koje sve informacijske poruke treba ispisati na ispis u spoolu, u ovom slučaju to je cijela JCL skripta za referencu, operator, i sve relevantne izlazne poruke o izvršavanju [14]. Vrijednost izraza „1“ govori sistemu da ispiše sve JCL izraze, JES kontrolne izraze, izraze JCL procedura i informacijskih poruka [15]. Parametar *NOTIFY* signalizira sustavu da izda obavijest kada dođe do promjene u statusu posla ,vrijednost *&SYSUID* je varijabla koja dopremi JCL-u ime korisnika koji podnosi (engl. Submit) posao [16].

JCL je vitalna komponenta z/OS sustava, omogućujući precizno definiranje i upravljanje poslom na z/OS sustavu. Precizne i moćne funkcionalnosti ga čine nezamjenjivim za sistemske i z/OS aplikacijske programere. U ovom radu, JCL se primarno koristi za kompiliranje programa i ekrana za CICS upravitelj transakcija.

3.3 Db2 Baza podataka

Db2 je baza podataka i sustav za upravljanje bazom podataka (engl. Database Managment System - DBMS) na z/OS operacijskom sustavu, dizajniran za obradu velikih količina podataka u poslovnim aplikacijama [7]. Bazu podataka kreirao je IBM, kao dio serije proizvoda za upravljanje podacima 1983. godine [17]. Db2 omogućuje pouzdano i sigurno upravljanje podacima, pružajući skalabilnost i visoku dostupnost. Db2 na z/OS-u koristi relacijsku arhitekturu koja omogućuje jednostavno modeliranje podataka i provođenje kompleksnih upita.

Arhitektura Db2 sastoji se od više komponenata koje surađuju kako bi osigurale optimalne performanse i pouzdanost. Upravljač radnog opterećenja z/OS operacijskog sustava određuje servisne klase za Db2 performanse te tako optimizira rad na samoj Db2 bazi podataka [18]. Administrator baze podatka postavlja definicije željenih performansi i prioriteta poslovnih logika unutar servisnih klasa baze podataka [18]. Međuspremnici Db2 obnašaju ulogu kontinuiranog privremenog pohranjivanja prostora tablica ili indeksa [19]. Kada program ili korisnik pristupa tablici ili retku putem upita, Db2 stavlja rezultat tog upita u međuspremnik, što osigurava značajne brzine pri radu s Db2 bazom podataka [19]. Baza također koristi mehanizme za zaključavanje, koji upravljaju kontinuiranim pristupom podataka, osiguravajući integritet. Sadržajem baze i administracijom upravlja se pomoću strukturno upitnog jezika (engl. Structured Query Language - SQL). Upravljanje Db2 bazom podataka zahtijeva planiranje i nadzor posla. Alati za oporavak podataka, pregled performansi i sigurnosne mjere zaštite su neophodne inačice Db2 baze podataka [7].

Db2 na z/OS operacijskom sustavu je temeljna tehnologija za upravljanje masovnom količinom podataka u poslovnim okruženjima. Robusna arhitektura, napredne funkcionalnosti i alati za upravljanje čine Db2 neizostavnom komponentom u arhitekturi većine *Mainframe* sustava. Db2 omogućuje organizacijama i institucijama, efikasno upravljanje podacima i poslovnim procesima. IBM Db2 logotip prikazan je na slici 3.2.

Slika 3.2 - IBM Db2 logotip [20]

3.4 PL/I programski Jezik

PL/I (programski kod 3.2) je višenamjenski, proceduralni programski jezik razvijen 1960-ih godina s ciljem ujedinjavanja karakteristika iz nekoliko popularnih jezika tog vremena, poput Fortrana, COBOL-a i Algola [21]. Namijenjen je prvenstveno za znanstvene, poslovne i tehničke primjene te je dizajniran kako bi pružio fleksibilnost u različitim okruženjima, uključujući poslovne aplikacije, inženjerske simulacije, kao i aplikacije u području znanosti.

PL/I omogućava programerima pisati visoko strukturirani kod koristeći napredne koncepte kao što su višedretveno izvođenje (engl. *multithreading*), rukovanje iznimkama te mogućnosti dinamičke alokacije memorije.

Njegova sposobnost za rad s velikim količinama podataka, kao i integracija s bazama podataka kao što je Db2, čine ga vrlo korisnim u okruženju velikih poslovnih sustava, osobito na platformama poput IBM-ovog z/OS *mainframe* sustava.

Programski kod 3.2 - Hello World u programskom jeziku PL/I

```
HELLO: PROCEDURE OPTIONS(MAIN);
    PUT SKIP EDIT('Hello World')(A);
END HELLO;
```

Jedna od glavnih prednosti PL/I jezika je njegova svestranost, jer omogućava implementaciju proceduralnih paradigmi i direktnog upravljanja memorijom. To ga čini pogodnim za složene transakcijske sustave kao što su oni koji koriste CICS, gdje je ključna brzina obrade velikog broja korisničkih zahtjeva.

U kontekstu ovog završnog rada, PL/I je korišten kao osnovni programski jezik za razvoj sustava upravljanja knjižnicom, uz implementaciju CICS transakcija za komunikaciju s korisnikom putem terminala. Njegova sposobnost rada s Db2 bazom podataka omogućava efikasnu pohranu i dohvaćanje podataka o korisnicima i knjigama unutar knjižničnog sustava. PL/I se pokazao kao idealan alat za razvoj ovakvih poslovnih aplikacija zbog svoje stabilnosti, performansi i integracije s ostalim alatima na z/OS platformi.

3.5 CICS upravitelj transakcija

CICS je upravitelj transakcija koji je razvio IBM za z/OS i z/VSE operacijske sustave [6]. Uz mogućnost upravljanja transakcijama, CICS je i višejezični aplikacijski server koji

omogućuje razvoj i izvršavanje aplikacija koje istovremeno pristupaju bazama podataka i obavljaju online transakcije [6]. CICS pruža visok stupanj pouzdanosti, brzine i sigurnosti pri izvršavanju svog rada na sustavima, što ga čini ključnim elementom poslovnih okruženja u industrijskim sektorima poput financija, bankarstva, maloprodaje i javnih usluga.

Osmišljen za veliki opseg posla i broj korisnika na sustavu, uz upravljanje kritičnim aplikacijama, CICS je nezaobilazan softver za većinu Z[2] sustava. Glavne funkcionalnosti CICS-a su: upravljanje transakcijama, upravljanje resursima, sigurnost i integracija s bazama podataka.

Efikasno upravljanje transakcijama postignuto je uz očuvanje integriteta podataka i simultano izvršavanje tisuća istih bez značajnih gubitaka performansi. Upravljanje resursima poput baza podataka, datoteka, terminala i komunikacijskih kanala uz aplikacijsko pristupanje podacima na siguran i kontroliran način. CICS također podržava standardne sigurnosne mehanizme poput identifikacije i autorizacije korisnika, autorizacija resursa te šifriranje podataka u komunikaciji. CICS posjeduje solidnu integraciju s Db2 bazom podataka. Svaka od ovih operacija tretira se kao transakcija u CICS-u, čime se osigurava da su svi podaci konzistentni i da nema gubitka informacija u slučaju nepredviđenih događaja poput prekida rada sustava.

CICS upravitelj transakcija ključan je element za razvoj robusnih, sigurnih i efikasnih poslovnih aplikacija u z/OS okruženju. Njegova integracija s Db2 bazom podataka omogućuje stvaranje složenih transakcijskih sustava, kao što je sustav za upravljanje knjižnicama. Razumijevanje osnovnih principa rada CICS-a i njegove primjene u različitim poslovnim scenarijima od iznimne je važnosti za uspješnu implementaciju poslovnih aplikacija na *Mainframe* platformi.

[2] Z – IBM Mainframe instalacije

4. PROCES IZRADE SISTEMA

U procesu izrade sustava za obradu podataka knjižnice, ključno je pokrenuti sve funkcionalne cjeline na kojima se izvršavaju aplikacije. Prvi koraci ovog procesa uključuju detaljnu analizu pokretačkih skripti za Db2 bazu podataka, CICS transakcijski sustav te postavljanje modula za kompiliranje aplikacija.

Kako bi se Db2 baza podataka i CICS transakcijski sustav uspješno pokrenuli, potrebno je provjeriti dokumentacijske upute za verziju koja se podiže. Nakon provjerenih pokretačkih skripti, potrebno je unesti komande u sustavu za prikaz i pretraživanje (engl. System Display and Search Facility - SDSF) za određeno pokretanje sustava. Za pokretanje Db2 sustava, koristi se JCL pokretački posao *DBCG,* naredbom: */-DBCG START DB2* [22] koja će pokrenuti procedure s profilom iz svoje sistemske biblioteke DSNC10.PROCLIB (Db2 verzija 12). Nakon puštanja naredbe na SDSF administracijsku konzolu, vrši se provjera postoji li pokrenuta instanca JOB-a na samom sistemu. Pokrenuti poslovi s profilom *DBCG* nalaze se na podsustavu za unos posla (engl. Job Entry Subsystem - JES) spool-u, u stanju *EXECUTION* , što znači da je Db2 pokrenut i da je u stanju izvođenja na z/OS-u. Isti proces se vrši i za CICS transakcijski sustav, samo se naredba za isti pušta pomoću START komande: */S CICSTS61* (Start CICS v6.1) [23] te se također nalazi u „EXECUTION“ stanju na JES spool-u. Nakon podizanja ključnih stavki za sustav knjižnice, potrebno je izraditi ekrane pomoću kojih korisnik ima interakciju s CICS sustavom i bazom podataka te samu Db2 bazu podatka, dizajniranu prema potrebama knjižnice. CICS ekrani za interakciju sa sustavom kreirani su po samoj prezentacijskoj i upravljačkoj logici Db2 baze podataka, tako da svaki ekran simbolizira jednostavniji primjer *SELECT, INSTERT* i *DELETE* SQL upita na sadržaj podataka knjižnice.

Kroz daljnja poglavlja opisan je detaljniji pristup realizacije sustava obrade podataka knjižnice, uključujući i logiku kreiranja sheme baze podataka, normalizacija podataka, logiku odnosa tablica i automatsko ažuriranje. Nakon postavljanja i kreiranja svih ključnih komponenti, na fokus dolazi izrada CICS transakcija u PL/I jeziku, što je vitalni dio ovog sustava. Transakcijska logika opisana je zajedno s logikom izrade ekrana, jer ekrani su vizualna reprezentacija poslovne logike sustava za obradu podataka knjižnice.

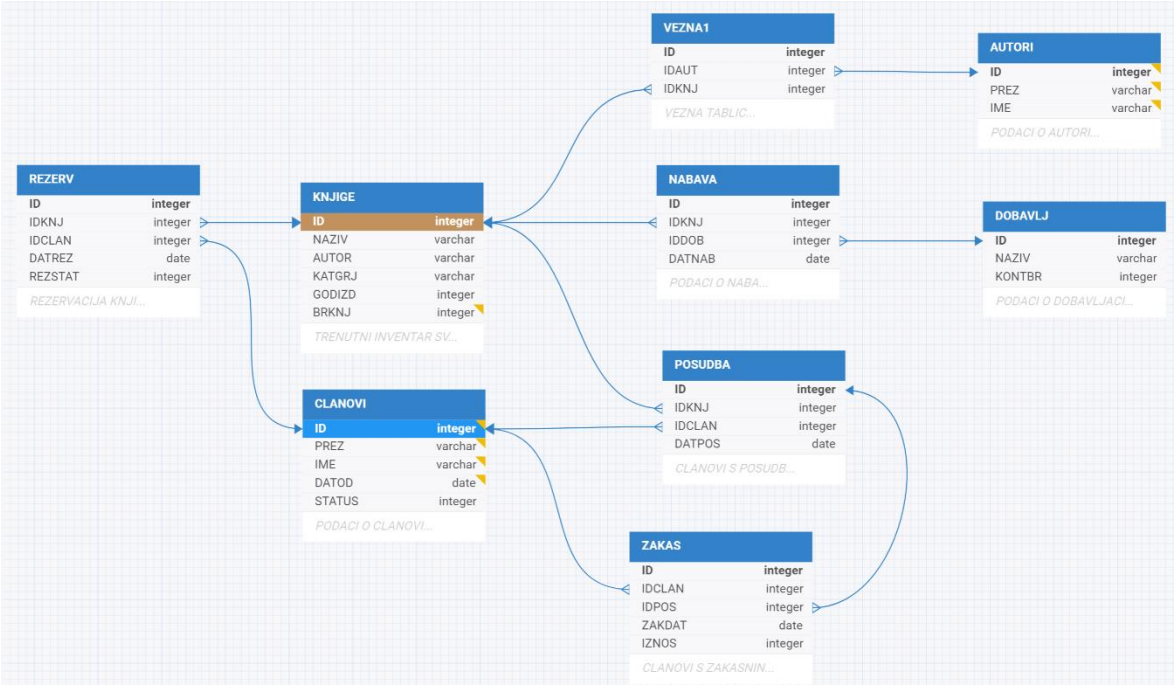
4.1 Izrada Db2 tablica

Proces izrade tablica Db2 baze podataka za potrebe sustava knjižnice opisano je pomoću produkcijskog načina razvoja i standarda. Svaka potreba knjižnice zadovoljena je

implementacijom isplanirane strukture tablica baze podataka. Prikaz sheme Db2 baze podataka sustava knjižnice prikazan je na slici 4.1. Kroz ovo poglavlje je detaljno obrađen pristup kreiranja svake strukture, njihovih stupaca, primarnih i sekundarnih ključeva, indeksa te način na koji su tablice među povezane. Naglasak je na pojašnjenju svrhe i načina izrade svakog elementa unutar baze podataka zbog osiguranja optimalne organizacije i učinkovitosti korištenja sustava.

Kreirane tablice uključuju osnovne podatke o autorima, članovima knjižnice, inventarom knjiga, informacijama o dobavljačima, posudbama, rezervacijama i zakasnini. Svaka tablica ima jasno definiranu strukturu koja uključuje primarni ključ zbog jednoznačne identifikacije svakog retka, kao i potrebne strane ključeve za osiguranje konzistencije i povezivanja podataka između različitih tablica. Osim same izrade tablica, obrađuje se i definiranje jedinstvenih indeksa koji omogućuju brži programski pristup podacima, kao i kreiranje pogleda (engl. view) koji olakšavaju rad s podacima iz više tablica, čineći upite jednostavnijima i preglednijima. Prikazi omogućuju složene operacije pretraživanja i dohvaćanja podataka, uz lakši korisnički pristup i navigaciju podacima. Uz indekse i poglede, dodani su i komentari entiteta baze podataka na razini stupaca, s ciljem boljeg dokumentiranja i lakšeg razumijevanja baze podataka. Komentari pružaju dodatne informacije o svrsi i značenju svakog stupca, čime se olakšava održavanje i mogućnost dodatnog poboljšanja sustava.

Detaljnim opisima postupka kreiranja tablica, indeksa, pogleda i komentara, ovo poglavlje pruža jasan uvid u tehnički aspekt izgradnje baze podataka koja podržava svakodnevno poslovanje knjižnice. Također, naglašava se važnost pravilnog modeliranja podataka kako bi se osigurala efikasnost, integritet i skalabilnost sustava knjižnice.



Slika 4.1- Prikaz sheme Db2 baze podataka sustava knjižnice (Kreirano pomoću DB Designer alata)

4.1.1 *Tablica KNJIGE*

Tablica knjige jedna je od ključnih tablica sustava obrade podataka knjižnice, jer sadrži informacije o svim knjigama dostupnim u knjižnici. Ova tablica služi za pohranu podataka kao što su naslov knjige, naziv autora, kategorija, godina izdanja te broj primjeraka u inventaru. Sama tablica ima prefiks „TAB“ na imenovanju svakog stupca, dok pogled ima samo ime stupca, zbog jednostavnosti. Opis strukture nalazi se u programskom kodu 4.1 – SQL za kreiranje tablice KNJIGE. Primarni ključ tablice je stupac *TABID*, koji osigurava jedinstvenu identifikaciju svake knjige.

Struktura tablice *KNJIGE* sastoji se od stupaca:

- *TABID* - Primarni ključ, automatski generiran za jednoznačnu identifikaciju svake knjige.
- *TABNAZIV* – Naziv knjige, ne može biti prazan, pohranjen je kao tekst od 60 znakova.
- *TABAUTOR* – Ime i prezime ili pseudonim autora knjige, pohranjen kao tekst od 60 znakova
- *TABKATGRJ* – Kategorija knjige, omogućuje bolje sortiranje i pretragu, pohranjen kao tekst od 20 znakova.

- *TABGODIZD* – Godina izdanja, pohranjena kao cjelobrojna vrijednost koja omogućuje praćenje povijesti izdanja.
- *TABBRKNJ* – Broj knjige, cjelobrojna vrijednost koja označava broj primjeraka iste knjige u inventaru.

Za tablicu *KNJIGE* definiran je jedinstveni indeks na stupac *TABID* kako bi se omogućilo brže pretraživanje prema samom identifikacijskom broju knjige. Kreirani pogled, naziva *VIEW_KNJIGE*, omogućuje jednostavan pristup najvažnijim podacima o knjigama bez potrebe za kompleksnim imenovanjem stupaca. Dodani komentari na razini svakog stupca olakšavaju razumijevanje entiteta i održavanje baze podataka.

Programski kod 4.1 - SQL za kreiranje tablice KNJIGE

```
/* Kreiranje tablice KNJIGE s primarnim ključem */
CREATE TABLE KNJIGE (
    TABID INTEGER NOT NULL GENERATED BY DEFAULT AS IDENTITY
        (START WITH 1 INCREMENT BY 1 NO MAXVALUE ORDER), /* TABID - ID
knjige */
    TABNAZIV VARCHAR(60) NOT NULL, /* TABNAZIV - Naziv knjige */
    TABAUTOR VARCHAR(60), /* TABAUTOR - Naziv autora knjige */
    TABKATGRJ VARCHAR(20), /* TABKATGRJ - Kategorija knjige */
    TABGODIZD INTEGER, /* TABGODIZD - Godina izdanja */
    TABBRKNJ INTEGER, /* TABBRKNJ - Broj knjige */

    /* Definiranje primarnog ključa s imenom */
    CONSTRAINT PK_KNJIGE PRIMARY KEY (TABID)
);

/* Kreiranje jedinstvenog indeksa za primarni ključ */
CREATE UNIQUE INDEX IDX_UNIQUE_KNJIGE_ID ON KNJIGE (TABID);

/* Kreiranje view-a za tablicu KNJIGE */
CREATE VIEW VIEW_KNJIGE AS
    SELECT
        TABID AS ID, /* ID knjige */
        TABNAZIV AS NAZIV, /* Naziv knjige */
        TABAUTOR AS AUTOR, /* Naziv autora */
        TABKATGRJ AS KATGRJ, /* Kategorija knjige */
        TABGODIZD AS GODIZD, /* Godina izdanja */
```

```
        TABBRKNJ AS BRKNJ /* Broj knjige */
    FROM KNJIGE;

/* Dodavanje komentara na stupce */
COMMENT ON COLUMN KNJIGE.TABID IS 'TABID - ID knjige';
COMMENT ON COLUMN KNJIGE.TABNAZIV IS 'TABNAZIV - Naziv knjige';
COMMENT ON COLUMN KNJIGE.TABAUTOR IS 'TABAUTOR - Naziv autora knjige';
COMMENT ON COLUMN KNJIGE.TABKATGRJ IS 'TABKATGRJ - Kategorija knjige';
COMMENT ON COLUMN KNJIGE.TABGODIZD IS 'TABGODIZD - Godina izdanja';
COMMENT ON COLUMN KNJIGE.TABBRKNJ IS 'TABBRKNJ - Broj knjige';
```

4.1.2 *Tablica CLANOVI*

Tablica *CLANOVI* opisana je u programskom kodu 4.2 – SQL za kreiranje tablice *CLANOVI*. Ista sadrži informacije o svim članovima knjižnice. Ova tablica pohranjuje osnovne podatke o članovima, kao što su prezime, ime, datum učlanjenja i status članstva. Primarni ključ tablice je stupac *TABID*, koji osigurava jedinstvenu identifikaciju svakog člana.
Struktura tablice *CLANOVI* sastoji se od sljedećih stupaca:

- *TABID* – Primarni ključ automatski generiran za jednoznačnu identifikaciju svakog člana.
- *TABPREZ* - Prezime člana, obavezan podatak, pohranjen kao tekst od 60 znakova.
- *TABIME* - Ime člana, pohranjeno kao tekst od 60 znakova.
- *TABDATOD* – „Datum od…“ Datum učlanjenja, pohranjen kao tip podataka DATE, koji omogućuje praćenje kada se član pridružio knjižnici.
- *TABSTATUS* - Status člana, pohranjen kao cjelobrojna vrijednost, koja označava trenutni status člana (aktivan, čekanje potvrde uplate članarine ili neaktivan).

Za tablicu *CLANOVI* definiran je i jedinstveni indeks koji omogućuje bržu pretragu podataka po *ID-u* člana. Također, kreiran je i pogled pod nazivom *VIEW_CLANOVI*, koji olakšava dohvaćanje ključnih informacija o članovima, kao što su *ID,* prezime, ime, datum učlanjenja i status.

Programski kod 1.2 – SQL za kreiranje tablice CLANOVI

```
/* Kreiranje tablice CLANOVI s primarnim ključem */
CREATE TABLE CLANOVI (
    TABID INTEGER NOT NULL GENERATED BY DEFAULT AS IDENTITY
        (START WITH 1 INCREMENT BY 1 NO MAXVALUE ORDER), /* TABID - ID
člana */
    TABPREZ VARCHAR(60) NOT NULL, /* TABPREZ - Prezime člana */
    TABIME VARCHAR(60), /* TABIME - Ime člana */
    TABDATOD DATE, /* TABDATOD - Datum uclanjenja */
    TABSTATUS INTEGER, /* TABSTATUS - Status člana */

    /* Definiranje primarnog kljuca s imenom */
    CONSTRAINT PK_CLANOVI PRIMARY KEY (TABID)
);
```

4.1.3 *Tablica VEZNA1*

Tablica *VEZNA1* opisana je u programskom kodu 4.3 – SQL za kreiranje tablice *VEZNA1*. Ista se koristi za povezivanje autora i knjiga te osigurava referencijalni integritet između tablice *AUTORI* i tablice *KNJIGE*. Ova tablica omogućuje pohranu informacija o vezi između određenog autora i njegovih knjiga. Primarni ključ tablice je stupac *TABID*, koji osigurava jedinstvenu identifikaciju svake veze.

Struktura tablice *VEZNA1* sastoji se od sljedećih stupaca:

- *TABID* - Primarni ključ, automatski generiran za jednoznačnu identifikaciju svake veze.
- *TABIDAUT* - *ID* autora, strani ključ koji se povezuje s tablicom *AUTORI*, osiguravajući referencijalni integritet.
- *TABIDKNJ* - *ID* knjige, strani ključ koji se povezuje s tablicom *KNJIGE*, osiguravajući referencijalni integritet.

Za tablicu *VEZNA1* definiran je jedinstveni indeks kako bi se omogućila brža pretraga podataka po *ID-u* veze. Također, kreiran je i pogled pod nazivom *VIEW_VEZNA1*, koji olakšava dohvaćanje informacija o vezama između autora i knjiga. Tablica *VEZNA1* koristi strane ključeve kako bi se osigurala konzistentnost podataka između tablica *AUTORI* i *KNJIGE*. Definirani su strani ključevi *FK_VEZNA1_AUT* i *FK_VEZNA1_KNJ*, koji osiguravaju da se svaka veza odnosi na postojećeg autora i knjigu u odgovarajućim tablicama.

Programski kod 4.3 – SQL kod za kreiranje tablice VEZNA1

```
/* Kreiranje tablice VEZNA1 s primarnim i stranim ključevima */
CREATE TABLE VEZNA1 (
    TABID INTEGER NOT NULL GENERATED BY DEFAULT AS IDENTITY
        (START WITH 1 INCREMENT BY 1 NO MAXVALUE ORDER), /* TABID - ID
veze */
    TABIDAUT INTEGER NOT NULL, /* TABIDAUT - ID autora (strani ključ) */
    TABIDKNJ INTEGER NOT NULL, /* TABIDKNJ - ID knjige (strani ključ) */

    /* Definiranje primarnog ključa s imenom */
    CONSTRAINT PK_VEZNA1 PRIMARY KEY (TABID),

    /* Definiranje stranog ključa za autore */
    CONSTRAINT FK_VEZNA1_AUT FOREIGN KEY (TABIDAUT)
        REFERENCES AUTORI(TABID)
        ON DELETE NO ACTION,

    /* Definiranje stranog ključa za knjige */
    CONSTRAINT FK_VEZNA1_KNJ FOREIGN KEY (TABIDKNJ)
        REFERENCES KNJIGE(TABID)
        ON DELETE NO ACTION
);

/* Kreiranje jedinstvenog indeksa za primarni ključ */
CREATE UNIQUE INDEX IDX_UNIQUE_VEZNA1_ID ON VEZNA1 (TABID);

/* Kreiranje view-a za tablicu VEZNA1 */
CREATE VIEW VIEW_VEZNA1 AS
    SELECT
        TABID AS ID, /* ID veze */
        TABIDAUT AS IDAUT, /* ID autora */
        TABIDKNJ AS IDKNJ /* ID knjige */
    FROM VEZNA1;

/* Dodavanje komentara na stupce */
COMMENT ON COLUMN VEZNA1.TABID IS 'TABID - ID veze';
COMMENT ON COLUMN VEZNA1.TABIDAUT IS 'TABIDAUT - ID autora';
COMMENT ON COLUMN VEZNA1.TABIDKNJ IS 'TABIDKNJ - ID knjige';
```

4.1.4 *Tablica AUTORI*

Tablica *AUTORI* opisana je u programskom kodu 4.4 – SQL za kreiranje tablice *AUTORI*. Ista sadrži informacije o svim autorima knjiga u knjižnično-informacijskom sustavu. Ova tablica služi za pohranu podataka kao što su prezime i ime autora. Primarni ključ tablice je stupac *TABID*, koji osigurava jedinstvenu identifikaciju svakog autora.

Struktura tablice *AUTORI* sastoji se od sljedećih stupaca:

- *TABID* - Primarni ključ, automatski generiran za jednoznačnu identifikaciju svakog autora.
- *TABPREZ* - Prezime autora, obavezan podatak, pohranjen kao tekst od 60 znakova.
- *TABIME* – Ime autora, pohranjeno kao tekst od 60 znakova.

Za tablicu *AUTORI* definiran je i jedinstveni indeks kako bi se omogućila brža pretraga podataka po *ID-u* autora. Također, kreiran je i pogled pod nazivom *VIEW_AUTORI*, koji omogućuje jednostavan pristup najvažnijim podacima o autorima, kao što su *ID*, prezime i ime.

Programski kod 4.4 – SQL za kreiranje tablice AUTORI

```
/* Kreiranje tablice AUTORI s primarnim ključem */
CREATE TABLE AUTORI (
    TABID INTEGER NOT NULL GENERATED BY DEFAULT AS IDENTITY
        (START WITH 1 INCREMENT BY 1 NO MAXVALUE ORDER), /* TABID - ID
autora */
    TABPREZ VARCHAR(60) NOT NULL, /* TABPREZ - Prezime autora */
    TABIME VARCHAR(60), /* TABIME - Ime autora */

    /* Definiranje primarnog kljuca s imenom */
    CONSTRAINT PK_AUTORI PRIMARY KEY (TABID)
);
```

4.1.5 *Tablica REZERV*

Tablica *REZERV* opisana je u programskom kodu 4.5 – *SQL za kreiranje tablice REZERV*. Ista se koristi za pohranu podataka o rezervacijama knjiga od strane članova knjižnice. Ova tablica omogućuje praćenje rezervacija, uključujući informacije o knjizi koja je rezervirana,

članu koji je napravio rezervaciju te status i datum rezervacije. Primarni ključ tablice je stupac *TABID*, koji osigurava jedinstvenu identifikaciju svake rezervacije.

Struktura tablice *REZERV* sastoji se od sljedećih stupaca:

- *TABID* - Primarni ključ, automatski generiran za jednoznačnu identifikaciju svake rezervacije.
- *TABIDKNJ* - *ID* knjige, strani ključ koji se povezuje s tablicom *KNJIGE*, osiguravajući referencijalni integritet.
- *TABIDCLAN* - *ID* člana, strani ključ koji se povezuje s tablicom *CLANOVI*, osiguravajući referencijalni integritet.
- *TABDATREZ* - Datum rezervacije, pohranjen kao tip podataka *DATE*, koji označava kada je rezervacija napravljena.
- *TABREZSTAT* - Status rezervacije, pohranjen kao cjelobrojna vrijednost koja označava trenutni status rezervacije (npr. aktivna, otkazana).

Za tablicu *REZERV* definiran je i jedinstveni indeks kako bi se omogućila brža pretraga podataka po *ID-u* rezervacije. Također, kreiran je i pogled pod nazivom *VIEW_REZERV*, koji omogućuje jednostavan pristup ključnim podacima o rezervacijama, kao što su *ID*, *ID* knjige, *ID* člana, datum rezervacije i status rezervacije. Tablica *REZERV* koristi strane ključeve kako bi se osigurala konzistentnost podataka između tablica *KNJIGE* i *CLANOVI*. Definirani su strani ključevi *FK_REZERV_KNJ* i *FK_REZERV_CLAN*, koji osiguravaju da se svaka rezervacija odnosi na postojeću knjigu i člana u odgovarajućim tablicama.

Programski kod 4.5 – SQL za kreiranje tablice REZERV

```
/* Kreiranje tablice REZERV s primarnim i stranim ključevima */
CREATE TABLE REZERV (
    TABID INTEGER NOT NULL GENERATED BY DEFAULT AS IDENTITY
        (START WITH 1 INCREMENT BY 1 NO MAXVALUE ORDER), /* TABID - ID
rezervacije */
    TABIDKNJ INTEGER NOT NULL, /* TABIDKNJ - ID knjige (strani kljuc) */
    TABIDCLAN INTEGER NOT NULL, /* TABIDCLAN - ID clana (strani kljuc)
*/
    TABDATREZ DATE, /* TABDATREZ - Datum rezervacije */
    TABREZSTAT INTEGER, /* TABREZSTAT - Status rezervacije */
```

```
    /* Definiranje primarnog kljuca s imenom */
    CONSTRAINT PK_REZERV PRIMARY KEY (TABID),

    /* Definiranje stranog kljuca za knjige */
    CONSTRAINT FK_REZERV_KNJ FOREIGN KEY (TABIDKNJ)
        REFERENCES KNJIGE(TABID)
        ON DELETE NO ACTION,

    /* Definiranje stranog kljuca za članove */
    CONSTRAINT FK_REZERV_CLAN FOREIGN KEY (TABIDCLAN)
        REFERENCES CLANOVI(TABID)
        ON DELETE NO ACTION
);
```

4.1.6 *Tablica DOBAVLJ*

Tablica *DOBAVLJ* opisana je u programskom kodu 4.6 - SQL za kreiranje tablice *DOBAVLJ*. Ista se koristi za pohranu podataka o dobavljačima koji surađuju s knjižnicom. Ova tablica omogućuje praćenje osnovnih informacija o dobavljačima, uključujući njihov naziv i kontakt broj. Primarni ključ tablice je stupac *TABID*, koji osigurava jedinstvenu identifikaciju svakog dobavljača.

Struktura tablice *DOBAVLJ* sastoji se od sljedećih stupaca:

- *TABID* - Primarni ključ, automatski generiran za jednoznačnu identifikaciju svakog dobavljača.
- *TABNAZIV* - Naziv dobavljača, obavezan podatak, pohranjen kao tekst do 60 znakova.
- *TABKONTBR* - Kontakt broj, pohranjen kao cjelobrojna vrijednost, koji služi za kontaktiranje dobavljača.

Za tablicu *DOBAVLJ* definiran je i jedinstveni indeks kako bi se omogućila brža pretraga podataka po *ID-u* dobavljača. Također, kreiran je i pogled pod nazivom *VIEW_DOBAVLJ,* koji omogućuje jednostavan pristup ključnim podacima o dobavljačima, kao što su ID, naziv i kontakt broj.

Programski kod 4.6 – SQL za kreiranje tablice DOBAVLJ

```
/* Kreiranje tablice DOBAVLJ s primarnim ključem */
CREATE TABLE DOBAVLJ (
    TABID INTEGER NOT NULL GENERATED BY DEFAULT AS IDENTITY
        (START WITH 1 INCREMENT BY 1 NO MAXVALUE ORDER), /* TABID - ID
dobavljača */
    TABNAZIV VARCHAR(60) NOT NULL, /* TABNAZIV - Naziv dobavljača */
    TABKONTBR INTEGER, /* TABKONTBR - Kontakt broj */

    /* Definiranje primarnog kljuca s imenom */
    CONSTRAINT PK_DOBAVLJ PRIMARY KEY (TABID)
);
```

4.1.7 *Tablica NABAVA*

Tablica *NABAVA* opisana je u programskom kodu 4.7 - SQL za kreiranje tablice *NABAVA*. Ista se koristi za pohranu podataka o nabavama knjiga koje knjižnica nabavlja od dobavljača. Ova tablica omogućuje praćenje informacija o tome koja je knjiga nabavljena, tko je dobavljač te kada je nabava obavljena. Primarni ključ tablice je stupac *TABID*, koji osigurava jedinstvenu identifikaciju svake nabave.

Struktura tablice *NABAVA* sastoji se od sljedećih stupaca:

- *TABID* - Primarni ključ, automatski generiran za jednoznačnu identifikaciju svake nabave
- *TABIDKNJ* - ID knjige, strani ključ koji se povezuje s tablicom *KNJIGE* radi osiguranja referencijalnog integriteta
- *TABIDDOB* - ID dobavljača, strani ključ koji se povezuje s tablicom *DOBAVLJ* radi osiguranja referencijalnog integriteta
- *TABDATNAB* - Datum nabave, pohranjen kao tip podataka *DATE*, koji označava datum kada je nabava obavljena

Za tablicu *NABAVA* definiran je i jedinstveni indeks kako bi se omogućila brža pretraga podataka po *ID-u* nabave. Također, kreiran je i pogled pod nazivom *VIEW_NABAVA*, koji omogućuje jednostavan pristup ključnim podacima o nabavama, kao što su *ID* nabave, *ID* knjige, *ID* dobavljača i datum nabave. Tablica *NABAVA* koristi strane ključeve kako bi se osigurala konzistentnost podataka između tablica *KNJIGE* i *DOBAVLJ*. Definirani su strani ključevi *FK_NABAVA_KNJ* i *FK_NABAVA_DOB*, koji osiguravaju da se svaka nabava odnosi na postojeću knjigu i dobavljača u odgovarajućim tablicama.

Programski kod 4.7 - SQL za kreiranje tablice NABAVA

```
/* Kreiranje tablice NABAVA s primarnim i stranim ključevima */
CREATE TABLE NABAVA (
    TABID INTEGER NOT NULL GENERATED BY DEFAULT AS IDENTITY
        (START WITH 1 INCREMENT BY 1 NO MAXVALUE ORDER), /* TABID - ID
nabave */
    TABIDKNJ INTEGER NOT NULL, /* TABIDKNJ - ID knjige (strani kljuc) */
    TABIDDOB INTEGER NOT NULL, /* TABIDDOB - ID dobavljača (strani kljuc)
*/
    TABDATNAB DATE, /* TABDATNAB - Datum nabave */

    /* Definiranje primarnog kljuca s imenom */
    CONSTRAINT PK_NABAVA PRIMARY KEY (TABID),

    /* Definiranje stranog kljuca za knjige */
    CONSTRAINT FK_NABAVA_KNJ FOREIGN KEY (TABIDKNJ)
        REFERENCES KNJIGE(TABID)
        ON DELETE NO ACTION,

    /* Definiranje stranog kljuca za dobavljače */
    CONSTRAINT FK_NABAVA_DOB FOREIGN KEY (TABIDDOB)
        REFERENCES DOBAVLJ(TABID)
        ON DELETE NO ACTION
);
```

4.1.8 *Tablica POSUDBA*

Tablica *POSUDBA* opisana je u programskom kodu 4.8 - SQL za kreiranje tablice *POSUDBA*. Ista se koristi za pohranu podataka o posudbama knjiga koje su članovi knjižnice posudili. Ova tablica omogućuje praćenje informacija o tome koja je knjiga posuđena, tko je član koji je posudio knjigu te kada je posudba obavljena. Primarni ključ tablice je stupac *TABID*, koji osigurava jedinstvenu identifikaciju svake posudbe.

Struktura tablice *POSUDBA* sastoji se od sljedećih stupaca:

- *TABID* - Primarni ključ, automatski generiran za jednoznačnu identifikaciju svake posudbe
- *TABIDKNJ* - ID knjige, strani ključ koji se povezuje s tablicom *KNJIGE* radi osiguranja referencijalnog integriteta
- *TABIDCLAN* - ID člana, strani ključ koji se povezuje s tablicom *CLANOVI* radi osiguranja referencijalnog integriteta
- *TABDATPOS* - Datum posudbe, pohranjen kao tip podataka *DATE*, koji označava datum kada je posudba obavljena

Za tablicu *POSUDBA* definiran je i jedinstveni indeks kako bi se omogućila brža pretraga podataka po *ID-u* posudbe. Također, kreiran je i pogled pod nazivom *VIEW_POSUDBA*, koji omogućuje jednostavan pristup ključnim podacima o posudbama, kao što su *ID* posudbe, *ID* knjige, *ID* člana i datum posudbe.

Tablica *POSUDBA* koristi strane ključeve kako bi se osigurala konzistentnost podataka između tablica *KNJIGE* i *CLANOVI*. Definirani su strani ključevi *FK_POSUDBA_KNJIGA* i *FK_POSUDBA_CLAN*, koji osiguravaju da se svaka posudba odnosi na postojeću knjigu i člana u odgovarajućim tablicama.

Programski kod 4.8 - SQL za kreiranje tablice POSUDBA

```
/* Kreiranje tablice POSUDBA s primarnim i stranim ključevima */
CREATE TABLE POSUDBA (
    TABID INTEGER NOT NULL GENERATED BY DEFAULT AS IDENTITY
        (START WITH 1 INCREMENT BY 1 NO MAXVALUE ORDER), /* TABID - ID
posudbe */
    TABIDKNJ INTEGER NOT NULL, /* TABIDKNJ - ID knjige (strani kljuc) */
    TABIDCLAN INTEGER NOT NULL, /* TABIDCLAN - ID clana (strani kljuc)
*/
    TABDATPOS DATE, /* TABDATPOS - Datum posudbe */

    /* Definiranje primarnog kljuca s imenom */
    CONSTRAINT PK_POSUDBA PRIMARY KEY (TABID),

    /* Definiranje stranog kljuca za knjige */
    CONSTRAINT FK_POSUDBA_KNJIGA FOREIGN KEY (TABIDKNJ)
        REFERENCES KNJIGE(TABID)
        ON DELETE NO ACTION,

    /* Definiranje stranog kljuca za članove */
    CONSTRAINT FK_POSUDBA_CLAN FOREIGN KEY (TABIDCLAN)
        REFERENCES CLANOVI(TABID)
        ON DELETE NO ACTION
);
```

4.1.9 *Tablica ZAKAS*

Tablica *ZAKAS* opisana je u programskom kodu 4.9 - SQL za kreiranje tablice *ZAKAS*. Ista se koristi za pohranu podataka o zakasnini koju članovi knjižnice moraju platiti za kasno vraćanje knjiga. Ova tablica omogućuje praćenje informacija o tome koji je član kasnio s vraćanjem knjige, koja je posudba u pitanju, datum zakasnine te iznos zakasnine. Primarni ključ tablice je stupac *TABID*, koji osigurava jedinstvenu identifikaciju svake zakasnine.

Struktura tablice *ZAKAS* sastoji se od sljedećih stupaca:

- *TABID* - Primarni ključ, automatski generiran za jednoznačnu identifikaciju svake zakasnine
- *TABIDCLAN* - *ID* člana, strani ključ koji se povezuje s tablicom *CLANOVI* radi osiguranja referencijalnog integriteta
- *TABIDPOS* - *ID* posudbe, strani ključ koji se povezuje s tablicom *POSUDBA* radi osiguranja referencijalnog integriteta
- *TABZAKDAT* - Datum zakasnine, pohranjen kao tip podataka *DATE*, koji označava datum kada je zakasnina nastala
- *TABIZNOS* - Iznos zakasnine, pohranjen kao cjelobrojna vrijednost koja označava iznos zakasnine koju član mora platiti

Za tablicu *ZAKAS* definiran je i jedinstveni indeks kako bi se omogućila brža pretraga podataka po *ID-u* zakasnine. Također, kreiran je i pogled pod nazivom *VIEW_ZAKAS*, koji omogućuje jednostavan pristup ključnim podacima o zakasninama, kao što su *ID* zakasnine, *ID* člana, *ID* posudbe, datum zakasnine i iznos zakasnine. Tablica *ZAKAS* koristi strane ključeve kako bi se osigurala konzistentnost podataka između tablica *CLANOVI* i *POSUDBA*. Definirani su strani ključevi *FK_ZAKAS_CLAN* i *FK_ZAKAS_POS*, koji osiguravaju da se svaka zakasnina odnosi na postojećeg člana i posudbu u odgovarajućim tablicama.

Programski kod 4.9 - SQL za kreiranje tablice ZAKAS

```
/* Kreiranje tablice ZAKAS s primarnim i stranim ključevima */
CREATE TABLE ZAKAS (
    TABID INTEGER NOT NULL GENERATED BY DEFAULT AS IDENTITY
        (START WITH 1 INCREMENT BY 1 NO MAXVALUE ORDER), /* TABID - ID
zakasnine */
    TABIDCLAN INTEGER NOT NULL, /* TABIDCLAN - ID clana (strani kljuc)
*/
    TABIDPOS INTEGER NOT NULL, /* TABIDPOS - ID posudbe (strani kljuc)
*/
    TABZAKDAT DATE, /* TABZAKDAT - Datum zakasnine */
    TABIZNOS INTEGER, /* TABIZNOS - Iznos zakasnine */

    /* Definiranje primarnog kljuca s imenom */
    CONSTRAINT PK_ZAKAS PRIMARY KEY (TABID),

    /* Definiranje stranog kljuca za članove */
    CONSTRAINT FK_ZAKAS_CLAN FOREIGN KEY (TABIDCLAN)
        REFERENCES CLANOVI(TABID)
        ON DELETE NO ACTION,

    /* Definiranje stranog kljuca za posudbe */
    CONSTRAINT FK_ZAKAS_POS FOREIGN KEY (TABIDPOS)
        REFERENCES POSUDBA(TABID)
        ON DELETE NO ACTION
);
```

4.2 Postavljanje PL/I, CICS i SQL kompajlera

Prije početka programiranja PL/I koda, potrebno je osposobiti kompajler za isti. PL/I kompajler je modul za kompilaciju koda, koji u sebi sadrži instrukcije za pretvaranje teksta u izvršivi modul. Pokretanjem modula dobiva se rad koji je opisan u tekstnim instrukcijama samoga koda. Za potrebe sustava knjižnice, potreban je kompajler za PL/I programski jezik, SQL kompajler za ugniježđene SQL izraze te CICS translator, koji prevodi sve CICS instrukcije u ostatak koda gdje se iste nalaze [24].

PL/I kompajler, u praktičnom smislu, je niz JCL instrukcija u kojima se nalaze informacije odakle z/OS operacijski sustav pronalazi napisani kod, gdje sprema kompilirani modul PL/I koda, iz kojih setova podataka dodaje podatke u kod i što dodatno u kodu mora obraditi. Ovisno o sadržaju programa koji se piše potrebno je složiti odgovarajuće kompajlere za:

- PL/I kod - Kompajler koji samo obrađuje čisti PL/I kod bez ikakvih ostalih dodataka
- PL/I + SQL – Kompajler koji obrađuje PL/I kod i ugradbeni SQL unutar koda. Pretprocesuiranjem SQL koda, kompajler Db2 bazi kreira zahtijevani modul. Kompajler tvori *bind* paket za PL/I program, koji tvori optimalnu putanju prema referenciranoj tablici ili pogledu [25].
- PL/I + CICS – Kompajler koji obrađuje PL/I kod s *EXEC CICS* izrazima. Kompilerski modul prvo prevede cijeli PL/I kod u komentare, osim *EXEC CICS* izraza, koje zatim prevodi u PL/I jezik. Nakon prevođenja, CICS kompajler povezuje generirani objektni PL/I kod, što omogućava aplikaciji da radi u CICS okruženju [24].
- PL/I + SQL + CICS – Kompajler koji obrađuje PL/I kod s SQL izrazima i *EXEC CICS* izrazima.

Nakon obrade, Kompajler tvori učitani modul (engl. Load Module) u sistemski određenom setu podataka koji je definiran kao CICS radna datoteka. Iz radne datoteke CICS, preko sistemskih instrukcija tvori programsku transakciju. U Programskom kodu 4.10 prikazane su sve korištene kompajlerske opcije za PL/I transakcije.

Programski kod 4.10 – JCL instrukcije PL/I kompajleru

```
//OPTIONS  DD  *
  PP(CICS),
  PP(SQL()),
  DFT(EBCDIC),
  AGGREGATE,
  ATTRIBUTES,
  CMPAT(V2),
  DEFAULT(LINKAGE(SYSTEM)),
  DEFAULT(OVERLAP),
  EXIT,
  EXTRN(FULL),
  FLAG(I),
  INITAUTO,
  LIMITS(EXTNAME(7)),
  LIMITS(FIXEDDEC(31)),
  MACRO,
  MARGINS(2,72,0),
  MAXNEST(BLOCK(50) DO(50) IF(50)),
  OBJECT,
  NOOPTIMIZE,
  OPTIONS,
  NOREDUCE,
  NORENT,
  RULES(LAXCTL),
  SOURCE,
  STMT,
  SYSTEM(CICS),
  NOWRITABLE,
  XREF
/*
```

Same instrukcije služe kao upute za *IBMZPLI* modul za kompiliranje PL/I aplikacija na z/OS operacijskom sustavu. Instrukcije su opisane redoslijedom kao u programskom kodu 4.10:

- *PP(CICS)* - specificira da će program biti obrađen kao CICS aplikacija. Omogućuje potrebne značajke i biblioteke za CICS funkcionalnost.
- *PP(SQL())* - omogućuje podršku za SQL unutar programa, što omogućava korištenje ugrađenih SQL izraza za operacije nad bazom podataka.
- *DFT(EBCDIC)* - specificira da je zadani znakovni kod za program EBCDIC koji se često koristi u *Mainframe* okruženjima.
- *AGGREGATE* - omogućuje korištenje agregatnih tipova podataka, kao što su nizovi ili zapisi, unutar programa.
- *ATTRIBUTES* - omogućuje specificiranje atributa za deklaracije podataka, pružajući dodatne informacije o načinu korištenja podataka.
- *CMPAT(V2)* - postavlja kompatibilnost s verzijom 2 jezika PL/I, osiguravajući da prevoditelj pridržava specifikacije i ponašanje te verzije.

- *DEFAULT(LINKAGE(SYSTEM))* - specificira da su opcije povezivanja postavljene na zadane sustavne postavke. Ovo kontrolira kako se vanjski podaci prosljeđuju između programa.
- *DEFAULT(OVERLAP)* - omogućuje preklapanje pohrane za varijable, što može pomoći u uštedi memorije, ali može zahtijevati pažljivo upravljanje.
- *EXIT* - označava da program može sadržavati izlazne točke, omogućujući programu da završi ili vrati kontrolu operativnom sustavu ili pozivnom programu.
- *EXTRN(FULL)* – vanjske reference se tretiraju na potpuno kvalificiran način, što može pomoći u točnom rješavanju imena tijekom povezivanja.
- *FLAG(I)* - prevoditelja može specificirati korištenje indikatora pogreške, obično za informativne ili dijagnostičke svrhe. "I" može označavati informacijske poruke.
- *INITAUTO* - inicijalizira automatske varijable (lokalne varijable) na njihove zadane vrijednosti, što može pomoći u sprječavanju pogrešaka s neinicijaliziranim varijablama.
- *LIMITS(EXTNAME(7))* - specificira maksimalnu duljinu vanjskih imena (npr. imena varijabli) na 7 znakova.
- *LIMITS(FIXEDDEC(31))* - postavlja maksimalnu preciznost za fiksne decimalne brojeve na 31 znamenku, kontrolirajući raspon i točnost numeričkih vrijednosti.
- *MACRO* - omogućuje korištenje makronaredbi u programu, što omogućuje generiranje i ponovno korištenje koda kroz definicije makronaredbi.
- *MARGINS(2,72,0)* - specificira margine za formatiranje koda, označavajući gdje započinju redci i koliko prostora ostaviti za uvlaku i čitljivost.
- *MAXNEST(BLOCK(50) DO(50) IF(50))* - postavlja limite na maksimalne razine ugniježđena za različite konstrukcije. U ovom slučaju, dozvoljeno je 50 ugniježđenih blokova, 50 ugniježdenih DO izjava i 50 ugniježdenih IF izjava.
- *OBJECT* - označava da će izlaz kompilacije biti objektna datoteka, prikladna za povezivanje s drugim programima ili modulima.
- *NOOPTIMIZE* - onemogućuje optimizaciju tijekom kompilacije, što može olakšati dijagnosticiranje pogrešaka, ali može rezultirati sporijim izvođenjem.
- *OPTIONS* - Ispis kompilerskih opcija u kompilerski ispis na JES spool.
- *NOREDUCE* - sprječava prevoditelj da smanji veličinu koda tijekom kompilacije, što može biti korisno za dijagnosticiranje.

- *NORENT* - označava da program nije reentrantan, što znači da se ne može sigurno izvršavati istovremeno od strane više poslova ili dretvi.
- *RULES(LAXCTL)* – opušta pravila kontrole tijekom kompilacije, što može pomoći kod određenih programskih praksi, ali može dovesti do manje stroge provjere pogrešaka.
- *SOURCE* - označava da bi izvorni kod trebao biti uključen u izlaz, obično za dijagnostičke ili dokumentacijske svrhe.
- *STMT* - omogućuje generiranje brojeva izjava u izlazu, što može biti korisno za dijagnosticiranje i praćenje tijeka izvršenja.
- *SYSTEM(CICS)* - specificira da će program koristiti CICS kao svoje okruženje izvršavanja, omogućujući CICS povezane funkcionalnosti i biblioteke.
- *NOWRITABLE* - označava da program neće moći pisati u datoteku izvorne kodove tijekom izvršenja, što može pomoći u sprječavanju slučajnih izmjena.
- *XREF* - generira popis križnog referenciranja varijabli, pomažući pratiti gdje je svaka varijabla deklarirana i korištena tijekom programa.

4.3 Izrada CICS ekrana za 3270 terminale

Definiranje CICS ekrana za terminale predstavlja ključan aspekt razvoja aplikacija u CICS okruženju. Ovaj proces ne samo da omogućuje interakciju s korisnicima, već i osigurava organizaciju informacija, poboljšanje korisničkog iskustva i validaciju podataka. Osim toga, CICS ekrani igraju vitalnu ulogu u reprezentaciji poslovne logike sustava, slično kao što to čine Db2 tablice. U nastavku su razlozi zbog kojih je definiranje CICS ekrana neophodno.

CICS ekrani služe kao sučelje za interakciju između korisnika i aplikacije. Korištenjem ekrana, korisnici mogu unositi podatke, primati informacije i upravljati funkcionalnostima aplikacije. Bez ovih ekrana, interakcija bi bila nemoguća, a aplikacije ne bi mogle zadovoljiti potrebe korisnika. CICS ekrani omogućuju korisnicima da vizualiziraju poslovnu logiku koja se provodi unutar sustava, čime se povezuje korisnički unos s internim procesima i pravilima. Dizajn ekrana također značajno utječe na estetiku i ukupno korisničko iskustvo. Pravilno osmišljeni ekrani mogu učiniti aplikaciju privlačnijom i intuitivnijom, što korisnicima olakšava rad. Estetski privlačan i funkcionalan ekran može motivirati korisnike i povećati njihovu produktivnost, dok istovremeno reflektira poslovnu logiku sustava. Definiranje ekrana omogućuje implementaciju mehanizama za rukovanje greškama. Ako

korisnik unese neispravne podatke, ekran može pružiti povratne informacije i omogućiti korisniku da brzo ispravi pogreške. Ovo je od vitalnog značaja za održavanje integriteta podataka i ispravnog funkcioniranja aplikacije. Ekrani, kao i tablice, moraju biti dizajnirani tako da učinkovito upravljaju pogreškama, čime se osigurava kontinuitet poslovnih operacija.

U zaključku, definiranje CICS ekrana za terminale neophodno je za uspješan razvoj aplikacija u CICS okruženju. Ovi ekrani ne samo da poboljšavaju interakciju s korisnicima, organizaciju informacija, validaciju podataka, estetiku i ukupno korisničko iskustvo, već također predstavljaju i ključnu komponentu poslovne logike sustava, na sličan način kao i Db2 tablice. S obzirom na sve prednosti koje donose, jasno je zašto je proces definiranja CICS ekrana ključan za funkcioniranje aplikacija i zadovoljavanje potreba korisnika.

4.3.1 *BMS podrška*

BMS (engl. Basic Mapping Support) je CICS funkcionalnost u Asemblerskom jeziku, koja se koristi za definiranje i upravljanje korisničkim sučeljem na terminalima. BMS definicije omogućuju programerima da specificiraju kako će se podaci prikazivati na ekranu i kako će korisnici komunicirati s aplikacijom.

DFHMSD (engl. Map Descriptor), opisan u programskom kodu 4.11. Ovo je osnovna definicija mape koja određuje karakteristike mape. Ovdje se specificiraju informacije kao što su naziv mape, tip terminala, broj redaka i stupaca, vrsta pohrane podataka i opcije bojanja teksta na terminalu.

Programski kod 4.11 – Primjer DFHMSD Map Descriptora

```
NAZIV    DFHMSD TYPE=&SYSPARM,                                  X
                LANG=PLI,                                       X
                MODE=INOUT,                                     X
                TERM=3270-2,                                    X
                CTRL=FREEKB,                                    X
                STORAGE=AUTO,                                   X
                MAPATTS=(COLOR),                                X
                TIOAPFX=YES
```

DFHMDI (engl. Input Descriptor), opisan u programskom kodu 4.12. Ova definicija se koristi za definiranje ulaznih polja na mapi. Uključuje informacije kao što su položaj, duljina i ime polja. *DFHMDI* također može sadržavati opcije za validaciju podataka.

Programski kod 4.12 – Primjer DFHMDI Input Descriptora

```
NAZIV    DFHMDI SIZE=(24,80),                                    X
                COLUMN=1,                                        X
                LINE=1
```

DFHMDF (engl. Field Modifier)), opisan u programskom kodu 4.13. Ova definicija se koristi za modificiranje postojećih polja na mapi. Omogućuje promjenu atributa kao što su boja, stil ili oblikovanje.

Programski kod 4.13 - Primjer DFHMDF Field Modifiera

```
NAZIV  DFHMDF POS=(1,1),                                         X
              LENGTH=5,                                          X
              ATTRB=(PROT,NORM,ASKIP),                           X
              COLOR=BLUE,                                        X
              INITIAL='TEKST'
```

BMS definicije sadrže različite dijelove koji omogućuju programerima precizno definirati kako će se podaci prikazivati na terminalima i kako korisnici upravljaju aplikacijom. Svaka komponenta, od definicije mape do kontrole kursora, igra važnu ulogu u stvaranju korisničkog sučelja koje je funkcionalno, intuitivno i usklađeno s poslovnom logikom. Razumijevanje ovih dijelova ključno je za razvoj učinkovitih aplikacija unutar CICS okruženja.

4.3.2 *Primjer BMS definicije glavnog izbornika sustava knjižnice*

Glavni izbornik sustava je polje za unos opcije rada. Opcija rada predstavlja poslovnu logiku koju zaposlenik knjižnice treba u svojem svakodnevnom radu. Preciznim Asemblerskim instrukcijama definira se ekran glavnog izbornika prikazanog na slici 4.2. Opis instrukcija potrebnih za kreiranje terminalne mape:

DFHMSD, opisan u programskom kodu 4.14. Ova sekcija definira karakteristike mape *MAINM1*. Parametri uključuju:

- *TYPE* - Tip mape definiran kao *&SYSPARM*, što omogućuje dinamičku konfiguraciju.
- *LANG* - Specifikacija jezika, PL/I.
- *MODE* - Postavka za ulaz/izlaz (*INOUT*), što znači da korisnik može unositi podatke i primati ih.
- *TERM* - Tip terminala, ovdje je specificiran kao 3270-2, standardni z/OS terminal.
- *CTRL* - Kontrola tipkovnice, ovdje je postavljena na FREEKB koja omogućuje slobodno korištenje tipkovnice.
- *STORAGE* - Određuje način upravljanja memorijom kao *AUTO* – automatsko određivanje prostora.
- *MAPATTS* - Omogućuje korištenje boja za označavanje elemenata na ekranu.
- *TIOAPFX* - Definira hoće li se koristiti unaprijed definirani predložak u simboličku definiciju mapseta.

Programski kod 4.14 – Opis Map Descriptora za glavni izbornik

```
MAINM1   DFHMSD TYPE=&SYSPARM,                                         X
                LANG=PLI,                                              X
                MODE=INOUT,                                            X
                TERM=3270-2,                                           X
                CTRL=FREEKB,                                           X
                STORAGE=AUTO,                                          X
                MAPATTS=(COLOR),                                       X
                TIOAPFX=YES
```

DFHMDI, opisan u programskom kodu 4.15. Ovdje se definira veličina mape koja će se prikazivati, u ovom slučaju 24 reda i 80 stupaca, što predstavlja omjer radnog terminala. Ova mapa će početi s pozicijom u prvom retku i prvom stupcu.

Programski kod 4.15 – Opis Input Descriptora za glavni izbornik

```
MAIN01   DFHMDI SIZE=(24,80),                                          X
                COLUMN=1,                                              X
                LINE=1
```

DFHMDF, opisan u programskom kodu 4.16. Ova sekcija definira različita polja koja se prikazuju na ekranu. Svako polje je definirano s atributima kao što su pozicija, duljina, boja i inicijalni sadržaj. U ovom programskom kodu, nema naslova za naredbe polja, zbog toga što se ista koriste kao statički tekst naslova ekrana. Atributi polja su postavljeni na *PROT*, što je indikator da se na polje ne može upisivati niša osim inicijalnog teksta i postavljen je indikator *NORM* (skraćeno od *NORMAL*), što znači da je to polje za prikaz teksta.

Programski kod 4.16 – Opis Field Descriptora za naslov glavnog izbornika

```
DFHMDF POS=(1,1),                                              X
       LENGTH=6,                                               X
       ATTRB=(PROT,NORM,ASKIP),                                X
       COLOR=BLUE,                                             X
       INITIAL='MAIN01'
DFHMDF POS=(2,30),                                             X
       LENGTH=20,                                              X
       ATTRB=(PROT,NORM),                                      X
       COLOR=GREEN,                                            X
       INITIAL='********************'
DFHMDF POS=(3,30),                                             X
       ATTRB=(PROT,NORM),                                      X
       LENGTH=20,                                              X
       COLOR=GREEN,                                            X
       INITIAL='* SUSTAV ZA OBRADU *'
DFHMDF POS=(4,30),                                             X
       ATTRB=(PROT,NORM),                                      X
       LENGTH=20,                                              X
       COLOR=GREEN,                                            X
       INITIAL='*PODATAKA KNJIZNICE*'
DFHMDF POS=(5,30),                                             X
       LENGTH=20,                                              X
       ATTRB=(PROT,NORM),                                      X
       COLOR=GREEN,                                            X
       INITIAL='********************'
```

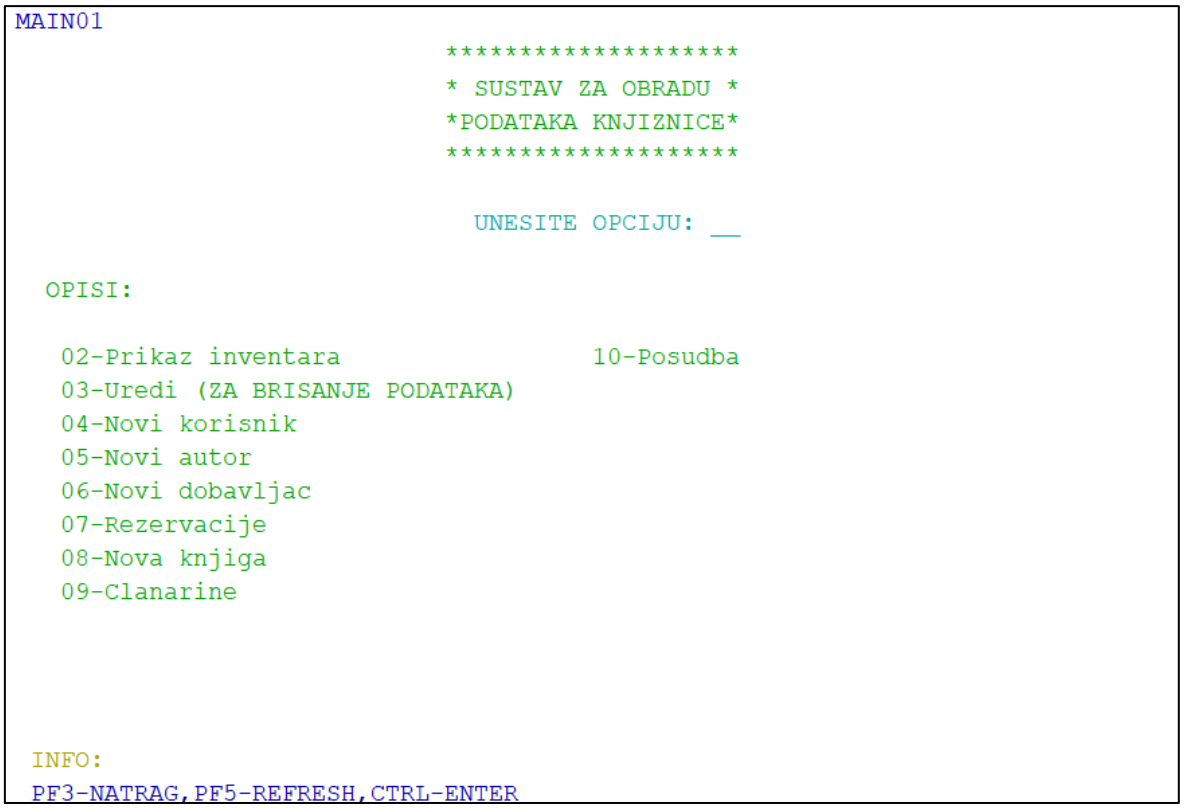
Nakon naslova, slijedi predefinirano područje za naziv polja unosa te samo polje za unos. *DFHMDF* izrazom u programskom kodu 4.17 definirano je poslije za unos, koje također ima atribut za inicijalizaciju kursora na samo polje prilikom prikaza ekrana (engl. Initialize Cursor - IC).

Programski kod 4.17 – Opis naslova polja za unos i polja za unos glavnog izbornika

```
UNASL1   DFHMDF POS=(7,32),                                    X
               LENGTH=15,                                      X
               ATTRB=(PROT),                                   X
               INITIAL='UNESITE OPCIJU:'
UNOS01   DFHMDF POS=(7,48),                                    X
               LENGTH=2,                                       X
               ATTRB=(UNPROT,NUM,IC),                          X
               COLOR=TURQUOISE,                                X
               INITIAL='__'
```

Nakon samog polja za unos stoji šifarnik unosnih komandi, koje otvaraju određene ekrane izbornika, prikazano na slici 4.2. Na svakom ekranu na istoj poziciji stoji *INFO* traka, na kojoj se ispisuju programski određene poruke korisniku. Same poruke služe kao smjernice kako bi korisnik trebao koristiti sučelja ispravno. Ispod same *INFO* trake, stoje statički opisi tipki koje se mogu koristiti u aplikaciji za navigaciju po ekranima:

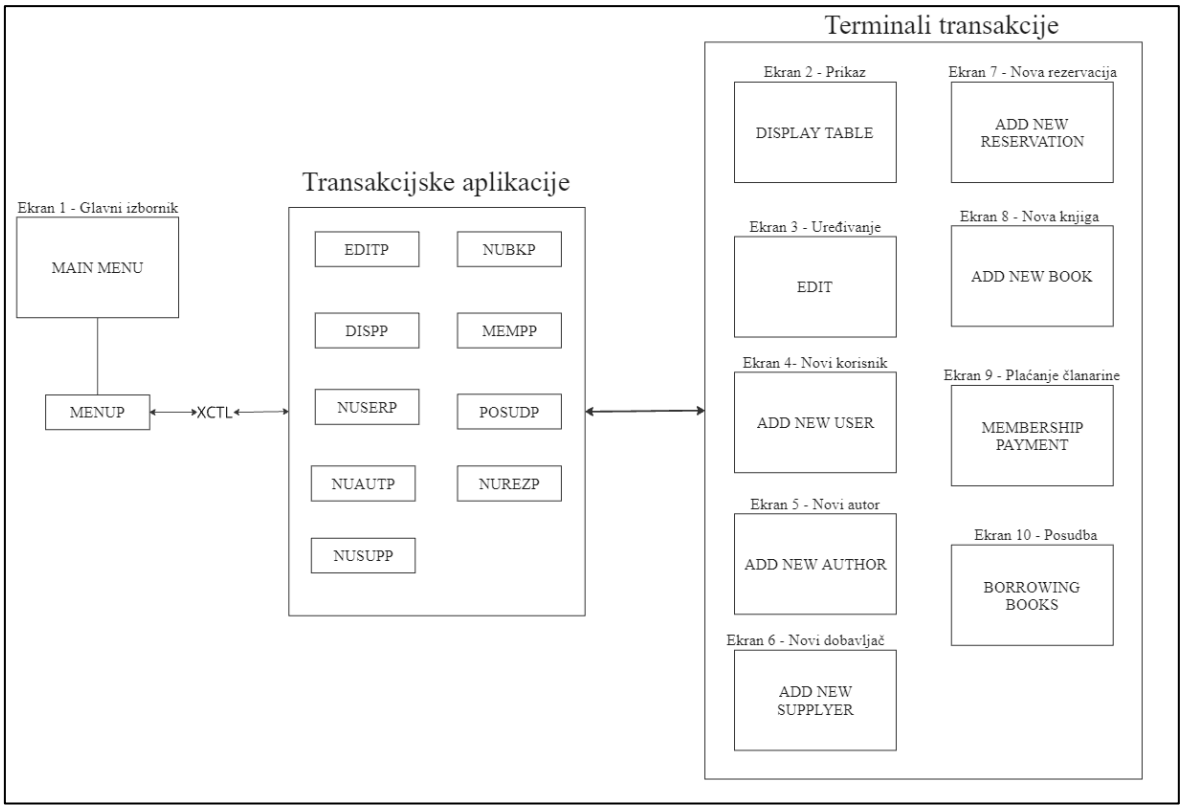
- *F3* – Zatvara trenutni ekran i otvara prethodni. Ukoliko je riječ o glavnom izborniku, *F3* terminira izvođenje transakcije te završava rad aplikacije.
- *F5* – Osvježuje ekran (šalje ponovno istu praznu mapu kako bi se uklonio sav korisnički unos).
- *CTRL* – Potvrdni gumb za izvršavanje unesene radnje na terminalnom ekranu.



Slika 4.2 - Primjer glavnog izbornika sustava za obradu podataka knjižnice

4.3.3 *Ekrani sustava*

Svaki ekran sustava za obradu podataka knjižnice ima svoj korespondentan program za upravljanje. Na slici 4.3 arhitekture ekrana i programa prikazani su svi CICS resursi unutar transakcije za obnašanje poslovne logike rada knjižnice. Transakcija funkcionira na principu da su sve komponente sustava definirane kao resursi.



Slika 4.3 - Programsko upravljanje prikazom ekrana (Kreirano pomoću aplikacije MIRO)

Svi resursi imaju međusobno upravljanje pomoću *XCTL* izvršne naredbe koja prosljeđuje izvođenje programu transakcije. Svaki program transakcije ima svoj korespondentan terminalni ekran preko kojeg korisnik svojim radom određuje što se prikazuje. Glavni ekran, glavni izbornik, korisničkim odabirom prosljeđuje kontrolu programu koji je odgovoran za vrstu poslovne logike koja se prikazuje na njegovom ekranu. Kada korisnik odabere naredbu za izlaz iz ekrana za određenu radnju koja je obavljena, taj program prosljeđuje izvođenje programu glavnog izbornika te prikazuje isti pomoću *XCTL* komade. U tablici programa i ekrana, naglašeno je kojim programima transakcije je dodijeljen ekran poslovne logike koju obnašaju.

Tablica 1 – Transakcijski programi sustava za obradu podataka knjižnice

NAZIV PROGRAMA	**NAZIV EKRANA**	**OPIS**
MENUP	MENU1	Glavni izbornik sustava knjižnice
DISPP	DISP1	Prikaz određenih podataka u sustavu
NUSERP	NUSER1	Dodavanje novog člana u sustav knjižnice
NAUTHP	NAUTH1	Dodavanje novog autora u sustav knjižnice

NSUPP	NSUPP1	Dodavanje novog dobavljača u sustav knjižnice
NUBKP	NBOOK1	Dodavanje nove knjige u sustav knjižnice
MEMPP	MEMPY1	Plaćanje članarine
POSUDP	POSUD1	Posudba
NREZEP	NREZE1	Dodavanje nove rezervacije knjige
EDITP	EDIT1	Direktno uređivanje podataka

4.4 Analiza i razvoj PL/I aplikacija

PL/I program unutar CICS transakcije se kontinuirano odvija unutar svoje glavne procedure. Sama glavna procedura programa postavljena je pod *OPTIONS(MAIN)* opcijom kraj naslova (opisano u programskom kodu 3.2). Unutar svake glavne procedure je niz pravila i varijabli potrebnih za upravljanje transakcijom.

Svaki program transakcije izvodi se u beskonačnoj *DO FOREVER* petlji, koja je odgovorna za izvršavanje transakcije, prikaz korisničkog sučelja i izvođenja sustava prilikom čekanja korisničkog unosa (programski kod 4.18).

Programski kod 4.18 – Prikaz beskonačne petlje koda

```
DO FOREVER;
  CALL DISP_SCR();
  CALL HANDLE_INPUT();

  IF EIBAID = DFHPF3 THEN DO;
    MENU01O.INFO1O = 'IZLAZ IZ APLIKACIJE.';
    EXEC CICS SEND TEXT FROM(MENU01O.INFO1O) ERASE;
    EXEC CICS RETURN;
  END;

  IF EIBAID = DFHPF5 THEN DO;
    CALL INIT_MAP();
    ITERATE;
  END;

END;
```

U petlji, program poziva korisničko sučelje procedurom *DISP_SCR* (Display Screen), prikazano u programskom kodu 4.19. Petljom se također konstantno prati korisnički unos

procedurom *HANDLE_INPUT* (programski kod 4.21). Korisnički unos je uzorkovan iz sistemske CICS varijable *EIBAID*, koja se uspoređuje s definiranim vrijednostima tipki: *F3*, *F5* i *CTRL*. Kada korisnik pritisne *F3*, na glavni ekran se ispisuje izlazni tekst „*IZLAZ IZ APLIKACIJE*“ te se transakcija terminira *EXEC CICS RETURN* naredbom. Pritiskom tipke F5, sustav inicijalizira varijable postavljene u Asemblerskoj definiciji ekrana, te iterira petlju za početni prikaz ekrana transakcije.

4.4.1 *Glavni meni sustava*

Glavni meni sustava je PL/I CICS aplikacija, koja služi za odabir poslovne logike. U samom programu, logika je implementirana na način uvjetovanog slanja zahtijeva *XCTL* komandom za pokretanje programa ovisno o korisničkom unosu. Simboličkom mapom definiranog seta mape za ekran (varijable koje predstavljaju sve što korisnik vidi i ne vidi na ekranu), se šalje zahtjev CICS-u preko *EXEC CICS SEND MAP* naredbe koja inicijalno pošalje ekran korisniku prilikom iniciranja transakcije (programski kod 4.19).

Programski 4.19 – PL/I procedura za slanje glavnog izbornika na korisnički terminal

```
DISP_SCR: PROC;
  EXEC CICS SEND MAP(MENU01) MAPSET(MENU1M)
   FROM(MENU01O)
   ERASE
   RESP(RESPONSE);

  IF RESPONSE <> DFHRESP(NORMAL) THEN DO;
    MENU01O.INFO1O = 'Send Error.';
    EXEC CICS SEND TEXT FROM(MENU01O.INFO1O) ERASE;
    EXEC CICS RETURN;
  END;
END DISP_SCR;
```

Program šalje prazan mapset glavnog izbornika korisniku te čeka korisnički unos. *ERASE* parametar briše sve što je bilo na terminalu prije nego što prikaže mapu glavnog izbornika. Inicijalizacijom kursora na predodređeno mjesto u asemblerskom izrazu definicije samog ekrana te omogući korisniku slobodan unos u polje za odabir opcije, *FREEKB* parametar. Poziv stoji unutar procedure *SEND_MAP*, koja je uvjetovano pozvana na početku transakcije ako je zadnji pritisnuti gumb bio *ENTER* (programski kod 4.20). Varijabla koja je odgovorna

za konstantno uzorkovanje zadnjih akcija na ekranu je *EIBAID*, dok je varijabla koja ima vrijednost tipke *ENTER* dio strukture koja je automatski dodana s CICS translatorom.

Programski kod 4.20 – Procedura za obradu korisničkog unosa

```
HANDLE_INPUT: PROC;
  EXEC CICS RECEIVE MAP(MENU01) MAPSET(MENU1M)
             INTO(MENU01I)
             RESP(RESPONSE);

  IF RESPONSE <> DFHRESP(NORMAL) THEN DO;

    IF EIBAID <> DFHENTER THEN RETURN;

    MENU01O.INFO1O = 'Receive Error.';
    EXEC CICS SEND TEXT FROM(MENU01O.INFO1O) ERASE;
    EXEC CICS RETURN;
  END;

  IF EIBAID = DFHENTER THEN DO;
    /* Obrada korisničkog unosa. */
    SELECT(MENU01I.UNOS01I);
      WHEN('02') CALL OPT_02;
      WHEN('03') CALL OPT_03;
      WHEN('04') CALL OPT_04;
      WHEN('05') CALL OPT_05;
      WHEN('06') CALL OPT_06;
      WHEN('07') CALL OPT_07;
      WHEN('08') CALL OPT_08;
      WHEN('09') CALL OPT_09;
      WHEN('10') CALL OPT_10;
     OTHERWISE CALL INV_OPT;
    END;

  END;

END HANDLE_INPUT;
```

Samim CICS upitom unutar procedure *HANDLE_INPUT* dodana je *RESP* opcija koja šalje odziv izvršene CICS komande u varijablu tipa podatka kao i sami odziv, *FIXED BINARY(31)*. Ukoliko je došlo do greške, izrazom *RETURN*, program prosljeđuje izvođenje glavnoj proceduri unutar koda i izlazi iz procesiranja procedure *HANDLE_INPUT*. Nakon unosa opcije korisnik pritišće *CTRL* tipku, što registrira *IF* uvjet. Uvjet provjerava stanje *EIBAID* pomoćne varijable koja uzorkuje stanje sustava, ukoliko je jednak nizu definiranog za tipku *CTRL* (Enter), izvršava se propadajuća provjera stanja ulazne varijable *UNOS01I*, definirane u Asembleru za ekran. Ukoliko je unos validan, izvođenje se prosljeđuje proceduri za odabranu opciju, unutar koje se izvršava XCTL poziv korespondentnog programa. Pozvani program prikazuje ekran poslovne logike koja se obrađuje.

Ukoliko korisnik unese nešto ne predefinirano programski, *OTHERWISE* uvjet prosljeđuje izvođenje proceduri koja ispisuje grešku korisniku na ekran. Svaki od navedenih *WHEN* izraza u programskom kodu 4.20, gleda specifični niz koji, ako je unesen, poziva proceduru unutar koje stoji *XCTL* upit s kojim se poziva definirani program. Slikom programskog koda 23, prikazana je standardizirana sintaksa *XCTL* poziva programa.

Programski kod 4.21 – Primjer poziva programa za prikaz inventara XCTL pozivom

```
OPT_02: PROC;
  MENU01O.INFO1O = 'Prikaz inventara.';

  EXEC CICS XCTL PROGRAM('DISPP')
       RESP(RESPONSE);

IF RESPONSE <> DFHRESP(NORMAL) THEN DO;
  SELECT(RESPONSE);
    WHEN(DFHRESP(PGMIDERR))
      MENU01O.INFO1O = 'Greska: Program DISPP nije pronaden.';
    WHEN(DFHRESP(NOTAUTH))
      MENU01O.INFO1O = 'Greska: DISPP program.';
    OTHERWISE
      MENU01O.INFO1O = 'Greska poziva Prikazne aplikacije. RESP: ' ||
      CHAR(RESPONSE);
  END;
  CALL DISP_SCR();
END;

END OPT_02;
```

Ukoliko korisnik unese krivu vrijednost u polje za odabir, u programskom kodu 4.21, prikazana je logika s kojom se ponovno šalje mapa bez korisničkog unosa s informacijskom porukom pogreške.

4.4.2 *Komunikacija transakcijskih programa s bazom*

Svaki od pozvanih programa sadrži logiku prikaza, uređivanja ili dodavanja podataka u Db2 bazu podataka SQL upitima. Ovisno o tipu poslovne logike, s terminalnog prikaza uzorkovani korisnički unos prenosi se u statički SQL upit prema korespondentnom pogledu Db2 baze. Pritiskom tipke *F3*, pozvani program se terminira te se izvođenje vraća programu glavnog izbornika pomoću *XCTL* naredbe.

Nakon korisničkog unosa podataka poslovne logike u određeno polje na ekranu, vrijednosti se šalju u pomoćne varijable programa, koje pretvaraju unesenu vrijednost po potrebi iz *CHARACTER* tipa podatka u *FIXED BINARY(15)* cijeli broj. Pozivom procedure koja sadrži statički SQL za određeni upit, vrši se provjera odziva SQL upita pomoću Db2 sistemske varijable *SQLCODE,* koja je dio strukture SQL komunikacijskog prostora (engl. SQL Communications Area - *SQLCA*). Ukoliko je upit prošao bez greške, dohvaćene vrijednosti uvrštavaju se u varijable ekrana kao alfanumerički tip podatka te se korisniku prikazuju na terminalu (programski kod 4.22). Ako je slučaj gdje korisnik radi SQL *INSERT* radnju, gdje unosi nove vrijednosti u sustav, biti će obavješten nakon njenog izvođenja u *INFO* polju za obavijesti preko varijable *INFO1BO*. Standardi imenovanja varijabli su identični za sve simboličke mape ekrana sustava.

Programski kod 4.22 – Primjer SQL upita i slanja dohvaćenih vrijednosti na terminal

```
EXEC SQL
    SELECT IBMUSER.VIEW_KNJIGE.ID,
           IBMUSER.VIEW_KNJIGE.NAZIV,
           IBMUSER.VIEW_KNJIGE.AUTOR,
           IBMUSER.VIEW_KNJIGE.KATGRJ,
           IBMUSER.VIEW_KNJIGE.GODIZD,
           IBMUSER.VIEW_KNJIGE.BRKNJ
    INTO :KNJIGE.ID,
         :KNJIGE.NAZIV,
         :KNJIGE.AUTOR,
         :KNJIGE.KATGRJ,
         :KNJIGE.GODIZD,
         :KNJIGE.BRKNJ
    FROM IBMUSER.VIEW_KNJIGE
    WHERE IBMUSER.VIEW_KNJIGE.ID = :HID;

    SELECT(SQLCA.SQLCODE);
      WHEN(0) CALL DISP_KNJ;
      WHEN(100) DISP1O.INFO1BO = 'Nema podataka za uneseni ID.';
     OTHERWISE DO;
      DISP1O.INFO1BO = 'SQL Error: '||SCRUBOUT(CHAR(SQLCA.SQLCODE),' ');
        CALL DISP_SCR();
        EXEC CICS RETURN;
      END;
    END;


DISP_KNJ: PROC;
   DISP1O.OUTPF1O = KNJIGE.NAZIV;
   DISP1O.OUTPF2O = KNJIGE.AUTOR;
   DISP1O.OUTPF3O = KNJIGE.KATGRJ;
   DISP1O.OUTPF4O = CHAR(KNJIGE.GODIZD);
   DISP1O.OUTPF5O = CHAR(KNJIGE.BRKNJ);
   DISP1O.INFO1BO = 'Podaci o knjizi.';
 END DISP_KNJ;
```

4.5 Kreiranje, instaliranje i testiranje transakcija

Pri završetku razvijanja aplikacija i ekrana sustava, iste je potrebno učiniti vidljivim CICS sustavu za obradu transakcija. Svaki ekran i kod, kompiliran je u izvršivi modul, koji se nalazi u korisnički definiranom mjestu na z/OS sustavu.

Kreiranjem radne grupe unutar CICS sustava, imenom *SUSTAV*, definirano je mjesto gdje će se nalaziti svi programi, transakcije, ekrani i veze. Deklariranjem i instaliranjem biblioteke u CICS sustavu, prosljeđuje se informacija o stvarnoj lokaciji kompiliranih resursa unutar z/OS operacijskog sustava (programski kod 4.23).

Programski kod 4.23 – Kreiranje i instaliranje biblioteke CICS-a

```
CEDA DEF GROUP(SUSTAV)
CEDA INSTALL GROUP(SUSTAV)
CEDA DEF LIBRARY(ANGILIB) GROUP(SUSTAV) DSN1(ANGI.LOAD)
CEDA INSTALL LIBRARY(ANGILIB) GROUP(SUSTAV)
```

CICS ugradbenom transakcijom *CEDA* (engl. Create/Edit/Display/Alter), definiraju se te potom instaliraju resursi u grupu. Programskim kodom 4.23 prikazano je bazično kreiranje grupe *SUSTAV* u kojoj je definirana biblioteka z/OS operacijskog sustava.

Programski kod 4.24 – Kreiranje i instaliranje Db2 konekcije prema CICS sustavu

```
CEDA DEF DB2CONN DB2ID(DBCG) GROUP(SUSTAV)
CEDA INSTALL DB2CONN DB2ID(DBCG) GROUP(SUSTAV)
```

Instalacijom Db2 konekcije, prikazano programskim kodom 4.24, CICS-u definiramo vezu prema Db2 sistemu preko CICS sistemskog korisničkog imena koji ima svoj Db2 plan sistemski definiran inicijalnim postavljanjem Db2 sistema.

Programski kod 4.25 – Kreiranje i instaliranje PL/I programa u CICS grupu

```
CEDA DEF PROG(MENUP) LANGUAGE(PLI) STATUS(ENABLED) GROUP(SUSTAV)
CEDA INSTALL PROG(MENUP) GROUP(SUSTAV)
```

Definiranjem programskog koda (programski kod 4.25), istoimenog kao i u z/OS operacijskom sustavu nakon kompilacije, CICS sustav vidi njegov učitani modul u setu podataka definiranog u biblioteci sustava (programski kod 4.23). Dodavanjem programa i njegova instalacija u grupu, čini ga aktivnim za izvršavanje.

Programski kod 4.26 – Kreiranje i instaliranje ekrana u CICS grupu

```
CEDA DEF MAP(MENU1) MAPSET(MENU1M) GROUP(SUSTAV)
CEDA INSTALL MAPSET(MENU1M) GROUP(SUSTAV)
```

Definicijom ekrana u sustav (programski kod 4.26), istoimene je moguće izvršiti programski te ih prikazati. Nazivlja ekrana ista su kao i kompilirani moduli njihovih BMS definicija u Asembleru. Ekrane je moguće testno prikazati sistemskom transakcijom CECI (engl. CICS Execution Command Interface).

Programski kod 4.27 – Kreiranje i definiranje transakcije u CICS grupu

```
CEDA DEF TRANSACTION(PLM1) PROGRAM(MENUP) STATUS(ENABLED) GROUP(SUSTAV)
CEDA INSTALL TRANSACTION(PLM1) GROUP(SUSTAV)
```

Kreiranjem transakcije (programski kod 4.27), koja sadrži programski kod koji izvršava, sustav je spreman izvršavati aplikaciju unosom naziva transakcije na glavni ekran sustava.

Ponovnim kompiliranjem ekrana ili programa koji su prethodno instalirani unutar grupe, nije napravljena nikakva promjena njihovim izvršnim modulima te je potrebno napraviti novu kopiju istih unutar grupe, kako bi se ažurirale promjene. Kreiranje nove kopije programa prikazano je u programskom kodu 4.28, korištenjem CEMT sistemske transakcije.

Programski kod 4.28 – Definiranje nove kopije programa nakon promjena

```
CEMT SET PROG(MENUP) NEWCOPY
```

5. ZAKLJUČAK

Ovaj rad predstavlja detaljnu analizu i implementaciju sustava za obradu podataka knjižnice koristeći IBM *mainframe* tehnologije. Kroz razvoj ovog sustava, demonstrirane su mogućnosti i fleksibilnost z/OS operacijskog sustava, CICS transakcijskog sustava, Db2 baze podataka i PL/I programskog jezika u rješavanju kompleksnih poslovnih zahtjeva moderne knjižnice.

Razvijeni sustav ne samo da zadovoljava trenutne potrebe knjižnice za upravljanjem resursima, već pruža i solidnu osnovu za buduća proširenja i poboljšanja. Modularni dizajn i korištenje standardnih *mainframe* tehnologija omogućuju lako dodavanje novih funkcionalnosti i prilagodbu promjenjivim zahtjevima knjižnice, kao barkod sustav na knjigama i korisničkim iskaznicama. Ovaj projekt također naglašava kontinuiranu relevantnost *mainframe* tehnologija u suvremenom IT okruženju. Unatoč percepciji da su ove tehnologije zastarjele, demonstrirano je kako one mogu pružiti robusna, skalabilna i sigurna rješenja za složene poslovne sustave.

Zaključno, ovaj rad ne samo da predstavlja uspješnu implementaciju knjižničnog sustava, već služi i kao primjer kako se tradicionalne *mainframe* tehnologije mogu primijeniti u rješavanju suvremenih izazova upravljanja podacima i poslovnim procesima. Iskustva i znanja stečena tijekom ovog projekta pružaju vrijednu osnovu za buduće projekte koji zahtijevaju visoku razinu pouzdanosti, sigurnosti i performansi u obradi podataka i upravljanju transakcijama.

6. LITERATURA

[1] IBM, »CICS Transaction Server for z/OS,« 2023. [Mrežno]. Available: https://www.ibm.com/docs/en/cics-ts/5.6?topic=whats-new. [Pokušaj pristupa 11 lIPANJ 2024].

[2] IBM, »CICS Transaction Server for z/OS - Physical and symbolic map sets,« IBM, 2023. [Mrežno]. Available: https://www.ibm.com/docs/en/cics-ts/5.6?topic=map-physical-symbolic-sets. [Pokušaj pristupa 12 Lipnja 2024].

[3] IBM, »CICS Transaction Server for z/OS - EXEC interface block (EIB),« IBM, 2023. [Mrežno]. Available: https://www.ibm.com/docs/en/cics-ts/5.6?topic=applications-exec-interface-block-eib. [Pokušaj pristupa 12 Lipnja 2024].

[4] IBM, »CICS Transaction Server for z/OS - The Db2 address spaces,« IBM, 2023. [Mrežno]. Available: https://www.ibm.com/docs/en/cics-ts/5.6?topic=db2-address-spaces. [Pokušaj pristupa 12 Lipnja 2024].

[5] IBM, »CICS Transaction Server for z/OS - Enabling CICS Db2 applications to use OTE through threadsafe programming,« IBM, 2023. [Mrežno]. Available: https://www.ibm.com/docs/en/cics-ts/5.6?topic=stopccda-enabling-cics-db2-applications-use-ote-through-threadsafe-programming. [Pokušaj pristupa 12 Lipnja 2024].

[6] Customer Information Control System (CICS) General Information Manual, IBM.

[7] IBM, »Db2 12 for z/OS - What is Db2 for z/OS?,« IBM, 2024. [Mrežno]. Available: https://ibm.com/docs/en/db2-for-zos/12?topic=getting-started-db2-zos. [Pokušaj pristupa 07 Srpanj 2024].

[8] IBM, »Enterprise PL/I for z/OS,« IBM, 2021. [Mrežno]. Available: https://www.ibm.com/docs/en/epfz/5.3. [Pokušaj pristupa 7 Srpanj 2024].

[9] IBM, »z/OS 2.4,« IBM, 2021. [Mrežno]. Available: https://www.ibm.com/docs/en/zos/2.4.0. [Pokušaj pristupa Srpanj 2024].

[10] IBM, »ACBs of z/OS System Programming Volume 1,« IBM RedBooks, 2017. [Mrežno]. Available: https://www.redbooks.ibm.com/redbooks/pdfs/sg246981.pdf. [Pokušaj pristupa Srpnja 2024].

[11] Wikipedia, »Wikipedia,« [Mrežno]. Available: https://en.wikipedia.org/wiki/Z/OS. [Pokušaj pristupa 7 Srpnja 2024.].

[12] IBM, »z/OS Basic Skills - Hardware resources used by z/OS,« IBM, 2010. [Mrežno]. Available: https://www.ibm.com/docs/en/zos-basic-skills?topic=1960s-hardware-resources-used-by-zos. [Pokušaj pristupa Srpanj 2024].

[13] IBM, »z/OS Basic Skills - Reusable JCL Collection,« IBM, 2010. [Mrežno]. Available: https://www.ibm.com/docs/en/zos-basic-skills?topic=collection-basic-jcl-concepts. [Pokušaj pristupa 11 Srpnja 2024].

[14] IBM, »z/OS 2.4.0 Documentation,« IBM, 2021. [Mrežno]. Available: https://www.ibm.com/docs/en/zos/2.4.0?topic=mp-syntax-6. [Pokušaj pristupa 11 Srpnja 2024].

[15] IBM, »z/OS 2.4.0 DOcumentation,« IBM, 2021. [Mrežno]. Available: https://www.ibm.com/docs/en/zos/2.4.0?topic=mp-subparameter-definition-6. [Pokušaj pristupa 11 Srpnja 2024].

[16] IBM, »z/OS 2.4.0 Documentation,« IBM, 2021. [Mrežno]. Available: https://www.ibm.com/docs/en/zos/2.4.0?topic=user-sysuid. [Pokušaj pristupa 11 Srpnja 2024].

[17] Wikipedia, »IBM Db2,« Wikipedia, 12 06 2024. [Mrežno]. Available: https://en.wikipedia.org/wiki/IBM_Db2. [Pokušaj pristupa 12 07 2024].

[18] IBM, »Db2 for z/OS - Determining z/OS Workload Manager velocity goals,« 14 06 2024. [Mrežno]. Available: https://www.ibm.com/docs/en/db2-for-zos/12?topic=db2-determining-zos-workload-manager-velocity-goals. [Pokušaj pristupa 12 07 2024].

[19] IBM, »Db2 for z/OS - Buffer Pools,« IBM, 18 06 2024. [Mrežno]. Available: https://www.ibm.com/docs/en/db2-for-zos/12?topic=objects-buffer-pools. [Pokušaj pristupa 12 07 2024].

[20] »MEDIUM,« [Mrežno]. Available: https://medium.com/codex/walk-through-on-ibm-db2-schema-97aa428f9f32. [Pokušaj pristupa 10 09 2024].

[21] Wikipedia, »PL/I - Wikipedia,« Wikipedia, 18 Rijna 2024. [Mrežno]. Available: https://en.wikipedia.org/wiki/PL/I. [Pokušaj pristupa 23 Rujna 2024].

[22] IBM, »Starting Db2,« IBM, 2024. [Mrežno]. Available: https://www.ibm.com/docs/en/db2-for-zos/12?topic=db2-starting. [Pokušaj pristupa 9 Listopad 2024].

[23] IBM, »What happens when you start CICS,« IBM, 14 Kolovoza 2024. [Mrežno]. Available: https://www.ibm.com/docs/en/cics-ts/6.x?topic=startup-what-happens-when-you-start-cics. [Pokušaj pristupa 9 Listopada 2024].

[24] IBM, »Program Translation,« IBM, 3 Lipnja 2024. [Mrežno]. Available: https://www.ibm.com/docs/en/cics-ts/6.x?topic=applications-program-translation. [Pokušaj pristupa 9 Listopada 2024].

[25] IBM, »Binding application packages and plans,« IBM, 10 Rujna 2024. [Mrežno]. Available: https://www.ibm.com/docs/en/db2-for-zos/12?topic=zos-binding-application-packages-plans. [Pokušaj pristupa 10 Listopada 2024].

[26] IBM, »IBM Documentation,« 2023. [Mrežno]. Available: https://www.ibm.com/docs/en/cics-ts/5.6?topic=routing-basic-mapping-support-bms. [Pokušaj pristupa 11 Lipanj 2024].

[27] IBM, »CICS Transaction Server for z/OS,« 10 Rujna 2023. [Mrežno]. Available: https://www.ibm.com/docs/en/SSGMCP_5.6.0/pdf/api-reference_pdf.pdf. [Pokušaj pristupa 12 Lipnja 2024].

7. OZNAKE I KRATICE

AOR – Aplikacijska regija „Application Owning Region“

API – Programsko sučelje aplikacije „Application Programming Interface“

CICS – Sustav za kontrolu informacija o kupcima „Customer Information Control System“

COBOL – Uobičajeni poslovni jezik „COmmon Business Oriented Language“

DASD – Uređaj za pohranu s izravnim pristupom „Direct Access Storage Device“

Db2 – Baza podataka dva „Database Two“

DBMS – Sistem za upravljanje bazom podataka „Database Managment System“

DBRM – Modul zahtijeva za bazu podataka „Database Request Module“

DFS – Distribuirani sustav datoteka „Distributed File System“

EBCDIC - Prošireni binarno kodirani decimalni kod za razmjenu „Extended Binary Coded Decimal Interchange Code“

HLASM – Asembler visoke razine „High Level Asembler“

IMS – Sustav za upravljanje informacijama „Information Managment System“

ISPF – Interaktivni sustav za produktivnost „Interactive System Productivity Facility“

JCL – Jezik Kontrole Posla „Job Control Language“

JES – Podsustav za unos posla, „Job Entry Subsystem“

PL/I – Programski jezik jedan „Programming Language One“

RACF – Sustav za kontrolu pristupa resursima „Resource Access Control Facility“

SDSF – Sustav za prikaz i pretraživanje sistema „System Display and Search Facility“

SQL – Strukturni Upitni Jezik „Structured Query Language“

STC – Pokrenuti zadaci „Started Tasks“

TOR – Terminalna regija „Terminal Owning Region“

TSO/E – Sučelje za upravljanje opcijama i dodacima „Time Sharing Option/Exstensions“

z/OS – Operacijski sustav Z „Z Operating System“

ZFS – z/OS UNIX File System

8. SAŽETAK

Naslov: Sustav za obradu podataka knjižnice

Ovaj rad predstavlja detaljnu analizu i implementaciju sustava za obradu podataka knjižnice koristeći IBM *mainframe* tehnologije. Cilj je bio razviti robustan i skalabilan sustav koji integrira z/OS operacijski sustav, CICS transakcijski sustav, Db2 bazu podataka i PL/I programski jezik. Rad obuhvaća dizajn i implementaciju Db2 baze podataka prilagođene potrebama knjižnice, razvoj korisničkog sučelja pomoću CICS Basic Mapping Support (BMS) za 3270 terminale te implementaciju poslovne logike kroz PL/I programe. Podaci su prikupljeni kroz analizu zahtjeva knjižničnog sustava i najboljih praksi u *mainframe* razvoju. Rezultat je funkcionalan sustav koji demonstrira mogućnosti *mainframe* tehnologija u rješavanju suvremenih poslovnih izazova, uključujući upravljanje inventarom knjiga, članovima, posudbama, rezervacijama i nabavom. Rad pruža praktičan uvid u proces razvoja, kompiliranja, instaliranja i testiranja CICS transakcija, naglašavajući kontinuiranu relevantnost *mainframe* sustava u modernom IT okruženju.

Ključne riječi: Mainframe, z/OS, CICS, Db2, PL/I, knjižnični sustav.

9. ABSTRACT

Title: Library Data Processing System

This paper presents a detailed analysis and implementation of a library data processing system using IBM mainframe technologies. The goal was to develop a robust and scalable system that integrates the z/OS operating system, CICS transaction system, Db2 database, and PL/I programming language. The work encompasses the design and implementation of a Db2 database tailored to library needs, development of a user interface using CICS Basic Mapping Support (BMS) for 3270 terminals, and implementation of business logic through PL/I programs. Data was collected through analysis of library system requirements and mainframe development best practices. The result is a functional system that demonstrates the capabilities of mainframe technologies in solving contemporary business challenges, including management of book inventory, members, loans, reservations, and acquisitions. The paper provides practical insight into the process of developing, compiling, installing, and testing CICS transactions, emphasizing the continued relevance of mainframe systems in the modren IT environment.

Keywords: Mainframe, z/OS, CICS, Db2, PL/I, library system.

IZJAVA O AUTORSTVU ZAVRŠNOG RADA

Pod punom odgovornošću izjavljujem da sam ovaj rad izradio/la samostalno, poštujući načela akademske čestitosti, pravila struke te pravila i norme standardnog hrvatskog jezika. Rad je moje autorsko djelo i svi su preuzeti citati i parafraze u njemu primjereno označeni.

Mjesto i datum	Ime i prezime studenta/ice	Potpis studenta/ice
U Bjelovaru, Listopada 2024.	Antonio Gibas	Antonio Gibas

U skladu s čl. 58, st. 5 Zakona o visokom obrazovanju i znanstvenoj djelatnosti, Veleučilište u Bjelovaru dužno je u roku od 30 dana od dana obrane završnog rada objaviti elektroničke inačice završnih radova studenata Veleučilišta u Bjelovaru u nacionalnom repozitoriju.

Suglasnost za pravo pristupa elektroničkoj inačici završnog rada u nacionalnom repozitoriju

Antonio Gibac
ime i prezime studenta/ice

Dajem suglasnost da tekst mojeg završnog rada u repozitorij Nacionalne i sveučilišne knjižnice u Zagrebu bude pohranjen s pravom pristupa (zaokružiti jedno od ponuđenog):

a) Rad javno dostupan
b) Rad javno dostupan nakon 31.10.2024. (upisati datum)
c) Rad dostupan svim korisnicima iz sustava znanosti i visokog obrazovanja RH
d) Rad dostupan samo korisnicima matične ustanove (Veleučilište u Bjelovaru)
e) Rad nije dostupan

Svojim potpisom potvrđujem istovjetnost tiskane i elektroničke inačice završnog rada.

U Bjelovaru, Listopada 2024

potpis studenta/ice