E-ispiti: dizajn i izrada korisničkog sučelja koristeći Oracle ADF

Gjurović, Robert

Sveučilište Jurja Dobrile u Puli
Fakultet informatike
Preddiplomski sveučilišni studij

ROBERT GJUROVIĆ

E-ISPITI: DIZAJN I IZRADA KORISNIČKOG SUČELJA KORISTEĆI ORACLE ADF

Završni rad

Pula, 2020.

Sveučilište Jurja Dobrile u Puli
Fakultet informatike u Puli
Preddiplomski sveučilišni studij

ROBERT GJUROVIĆ

E-ISPITI: DIZAJN I IZRADA KORISNIČKOG SUČELJA KORISTEĆI ORACLE ADF

Završni rad

JMBAG: 0303038977, izvanredni student
Studijski smjer: Informatika

Kolegij: Informatički praktikum

Znanstveno područje: Društvene znanosti
Znanstveno polje: Informacijske i komunikacijske znanosti
Znanstvena grana: Informacijski sustavi i informatologija

Mentorica: prof. dr. sc. Vanja Bevanda

Pula, rujan 2020.

IZJAVA O AKADEMSKOJ ČESTITOSTI

Ja, dolje potpisani _____, kandidat za prvostupnika informatike ovime izjavljujem da je ovaj Završni rad rezultat isključivo mojega vlastitog rada, da se temelji na mojim istraživanjima te da se oslanja na objavljenu literaturu kao što to pokazuju korištene bilješke i bibliografija. Izjavljujem da niti jedan dio Završnog rada nije napisan na nedozvoljen način, odnosno da je prepisan iz kojega necitiranog rada, te da ikoji dio rada krši bilo čija autorska prava. Izjavljujem, također, da nijedan dio rada nije iskorišten za koji drugi rad pri bilo kojoj drugoj visokoškolskoj, znanstvenoj ili radnoj ustanovi.

Student

_____

U Puli, _____, _____ godine

IZJAVA

o korištenju autorskog djela

Ja, _____ dajem odobrenje Sveučilištu Jurja Dobrile

u Puli, kao nositelju prava iskorištavanja, da moj završni rad pod nazivom _____ _____ _____ koristi na način da gore navedeno autorsko djelo, kao cjeloviti tekst trajno objavi u javnoj internetskoj bazi Sveučilišne knjižnice Sveučilišta Jurja Dobrile u Puli te kopira u javnu internetsku bazu završnih radova Nacionalne i sveučilišne knjižnice (stavljanje na raspolaganje javnosti), sve u skladu s Zakonom o autorskom pravu i drugim srodnim pravima i dobrom akademskom praksom, a radi promicanja otvorenoga, slobodnoga pristupa znanstvenim informacijama.

Za korištenje autorskog djela na gore navedeni način ne potražujem naknadu.

U Puli, _____ (datum)

Potpis

_____

Sadržaj

1. **Uvod**

U ovom radu biti će prikazan dizajn i izrada korisničkog sučelja aplikacije „e-ispiti“ za potrebe Termoelektrane Plomin (TE Plomin) za polaganje ispita iz zaštite na radu.

Ovaj rad se nadovezuje na završni rad „E-ispiti: Model baze podataka i implementacija poslovnih pravila koristeći Oracle ADF“ u kojem je opisana analiza korisničkog zahtjeva te je izrađena baza podataka na kojoj se temelji sama aplikacija odnosno online ispit.

Cilj ovog rada je bio napraviti aplikaciju u Jdeveloperu bez pisanja programskog koda u Javi budući da ADF omogućuje deklarativni pristup razvoju aplikacije.

Ukratko, projekt E-ispiti omogućuje polaganje zaštite na radu svim zaposlenicima vanjskih tvrtki kako bi kod remontnih radova mogli ući u postrojenje termoelektrane Plomin i upoznali se s radom u pogonu, bili potpuno sigurni te kako bi se svaki rizik sveo na minimum.

Dosadašnja je praksa bila angažiranje vanjskih tvrtki kako bi provodile zaštitu na radu u HEP postrojenju no međutim stvaranje ovakve aplikacije bi eliminiralo takav angažman te potrebu za dodatnim vanjskim suradnicima.

Nakon uvoda ćemo se prisjetiti analize korisničkog zahtjeva, a nakon toga i same relacijske baze podataka i tablica koje će nam biti potrebne u samoj izradi aplikacije, ali i poslovnih pravila.

Nakon kratkog osvrta analize korisničkog zahtjeva te baze podataka opisana je proširena Oracle ADF arhitektura koja se sastoji od Business Services sloja te MVC (Model-View-Controller) standardne softverske arhitekture. Zatim je prikazano kreiranje Business Components sloja gdje su kreirani novi view objekti, view linkovi i view criterie koje su potrebne u izradi korisničkog sučelja.

Na kraju rada su prikazani kreirani task flowovi u aplikaciji te programska logika.

2. **Analiza korisničkog zahtjeva**

Prilikom izvođenja remontnih radova na generatorima u TE Plomin, HEP ima potrebu zapošljavati odnosno angažirati vanjske tvrtke, njihove zaposlenike i inženjere kako bi se sami radovi odvili na najkvalitetniji mogući način.

Tu nastaje problematika za HEP budući da svaki zaposlenik ili vanjski suradnik mora imati položenu zaštitu na radu kako bi mogao ući ili se uopće mogao kretati samim postrojenjem te je dosad bila praksa angažiranja specijaliziranih tvrtki za polaganje zaštite na radu. Kako to predstavlja dodatne komplikacije te trošak za samu tvrtku došlo se do ideje da bi sam HEP mogao provoditi ispite zaštite na radu u obliku online ispita za što je potrebna on-line aplikacija.

Ispit je osmišljen na način da se polagač prijavi točno određenom lozinkom za takav tip ispita (na primjer električar dobije šifru samo i isključivo za ispite električne struje) te pristupa samom ispitu koji je osmišljen na način da se polagaču najprije predočuje tekst koji bi isti trebao pročitati, kako bi se prisjetio naučenog te dodatno naučio nešto novo. Nakon toga polagač pristupa samom ispitu koji se sastoji od tri pitanja (uz mogućnost povećanja ili smanjenja broja pitanja i odgovora), ako točno odgovori na ponuđena pitanja dolazi do kraja te uspješno polaže zadani ispit, u protivnom u slučaju pogrešnog odgovora isti biva vraćen na sam tekst u svezi ponuđenog odgovora kako bi se polagač mogao ponovno prisjetiti te točno odgovoriti na zadano pitanje. U slučaju da točno odgovori prelazi na sljedeće ptanje koje je zadao instruktor ili pak prilazi samom kraju ispita.

3. **Pregled baze podataka**

Baza podataka se sastoji od deset međusobno povezanih relacija, odnosno tablica. Neke tablice se odnose na definiranje pojedinog ispita, dok se neke tablice odnose na polaganje ispita. Na sljedećoj slici mogu se vidjeti sve tablice, njihove veze te primarni, strani i jedinstveni ključevi. Budući da se ovaj rad zasniva na završnom radu „E-ispiti: Model baze podataka i implementacija poslovnih pravila koristeći Oracle ADF“ tablice na kojima će se bazirati aplikacija i koje su kreirane u bazi podataka su redom:

ORGANIZACIJA predstavlja tablicu u bazi podataka gdje su upisane sve termoelektrane koje pripadaju HEP-u.

KORISNICI je tablica u kojoj su upisani oni korisnici koji mogu upravljati aplikacijom, odnosno useri, superuseri i administratori.

TIP_ISPITA je tablica koja sadrži podatke o tipu ispita koji može biti bodovan ili ne bodovan.

Tablica DEF_ISPITI predstavlja tablicu u kojoj su definirani podaci o ispitima, poput šifre ispita, broja pitanja u ispitu i slično.

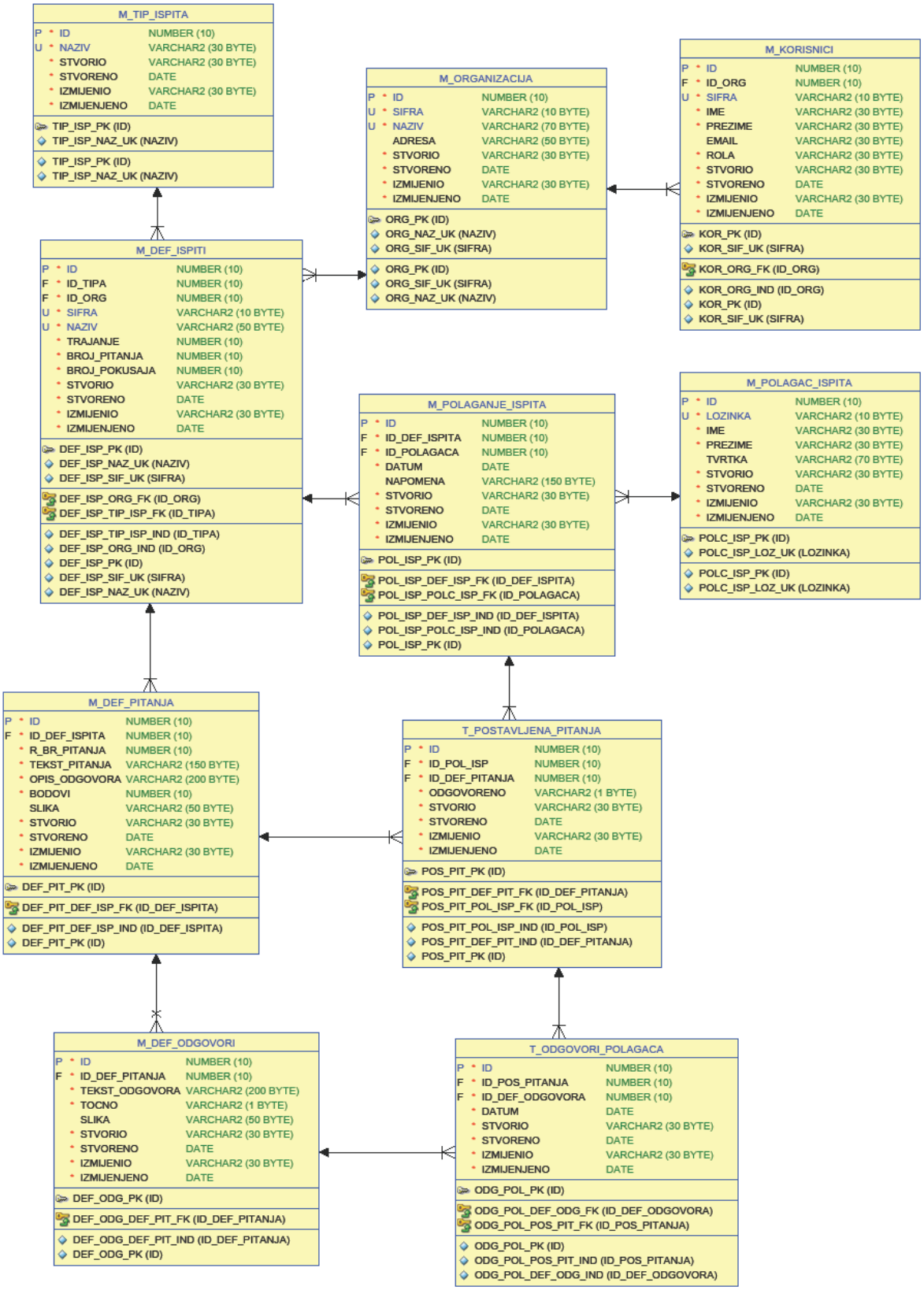
U tablicama DEF_PITANJA i DEF_ODGOVORI su prikazana sva moguća pitanja te ponuđeni odgovori.

POLAGAC_ISPITA je tablica koja sadrži popis osoba koje polažu ispit.

POLAGANJE_ISPITA je izvedena tablica koja sadrži podatke o ispitu i o polagaču ispita na točno određeni datum i u točno određeno vrijeme.

POSTAVLJENA_PITANJA je također izvedena tablica koja sadrži definirana pitanja koja su metodom slučajnog odabira postavljena polagaču ispita.

U tablici ODGOVORI_POLAGACA se nalaze svi odgovori na koje je polagač ispita odgovorio bili oni točni ili netočni.



Slika 1. Relacijski dijagram baze podataka e-ispiti

4. **Zašto Oracle ADF?**

Tehnologija se vrlo brzo mijenja, a rješenja koja su danas na vrhu tehnološkog dostignuća će za maksimalno dvije godine postati neprivlačna i nezanimljiva. Svijet se brzo kreće, tehnologija napreduje stoga je za neko poduzeće vrlo važno da poslovne aplikacije budu usklađene s rastućom bazom korisnika. Drugim riječima, poslovna aplikacija trebala bi biti dovoljno pametna da se prilagodi promjenama u poslovnom svijetu i razvija se paralelno s rastom poduzeća.

Zato je vrlo važno odabrati pravi alat i platformu za razvoj bilo koje poslovne aplikacije. Alat treba biti cjelovit i dovoljno fleksibilan kako bi udovoljio zahtjevima različitih faza životnog ciklusa aplikacije. Odabir alata koji pruža vizualno i deklarativno iskustvo također utječe i na produktivnost programera što je vrlo bitno za isporuku kvalitetnog softverskog proizvoda koji udovoljava zahtjevima kupaca.

Kako bi proizvod uspio na tržištu, osim generičkih značajki, trebao bi biti i prilagodljiv tako da udovolji jedinstvenim potrebama različitih grupa korisnika budući da potrebe kupaca svakim danom postaju sve složenije. Radi toga bi gotov softverski proizvod uvijek bi trebao predvidjeti promjene kako bi preživio na tržištu.

Tvrtke se razvijaju. Samim time raste i broj poslovnih korisnika, a poslovna aplikacija tvrtke se treba nositi sa sve većim brojem zahtjeva. Učinkovitost, skalabilnost i pouzdanost zaista su važni za bilo koju primjenu u poduzeću. Na primjer, kada posao raste za neko poduzeće, možda će trebati uzeti u obzir veću korisničku bazu. To na kraju može rezultirati povećanjem broja aktivnih korisnika poslovnih aplikacija koje se koriste u poduzeću. Poslovna aplikacija i temeljna tehnologija trebaju biti dovoljno skalabilni da udovolje sutrašnjim potrebama.

Na tržištu postoji mnogo alata i tehnologija za izradu poslovnih aplikacija, ali ako nam je potreban alat koji je uistinu sposoban odgovoriti na današnje izazove, popis se smanjuje i nemamo puno izbora. Oracle ADF smatra se jednim od rijetkih najboljih *Framework*-a za izgradnju poslovnih aplikacija.

Evo što Oracle ADF čini jednim od najboljih alata za izgradnju poslovnih aplikacija (Purushothaman, 2012):

Cjelovito rješenje: Oracle ADF pruža cjelovito rješenje za izgradnju korporativnih aplikacija od početka do faze izlaza u produkciju, adresirajući zahtjeve iz svakog sloja aplikacija.

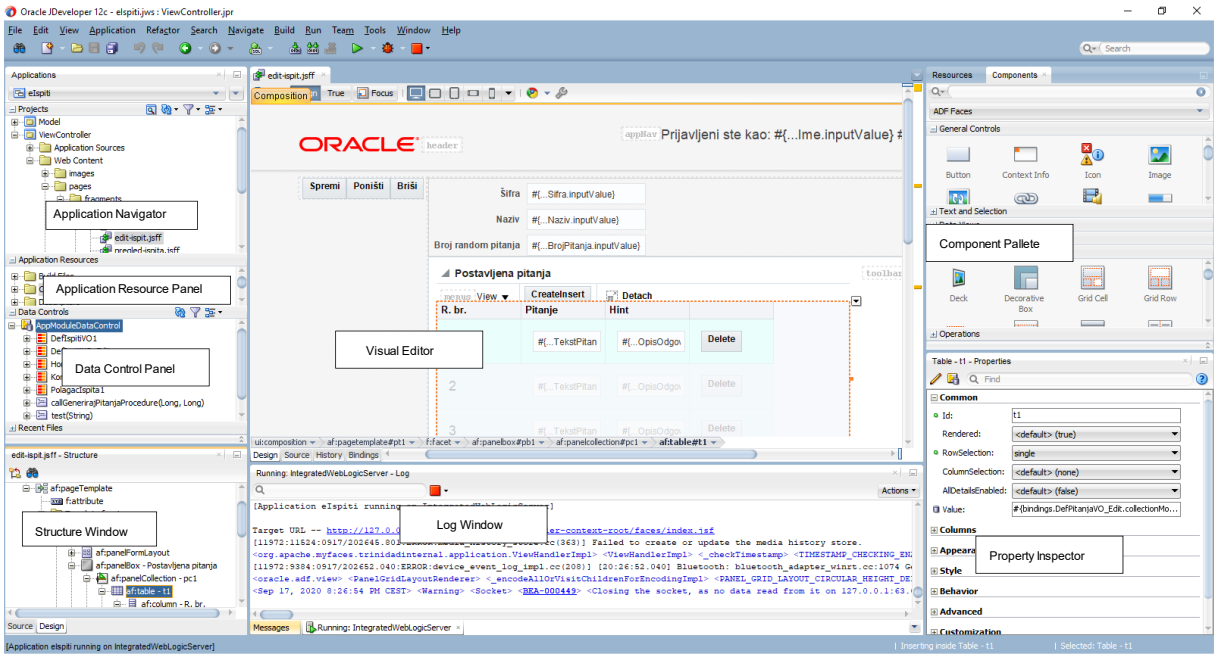
Poboljšana produktivnost programera: Deklarativna priroda ADF-a poboljšava produktivnost programera, omogućavajući korisnicima da se usredotoče na poslovnu logiku aplikacije, umjesto da se usredotoče na složenost tehnologije.

Rich Internet Application (RIA) i Web 2.0: ADF Rich Client ima preko 150 komponenti korisničkog sučelja, uključujući različite grafove i dijagrame što omogućuje asinhroni JavaScript i XML (AJAX).

Izbor tehnologije: Oracle ADF omogućava programeru da odabere više tehnologija za svaki od slojeva aplikacije i ne nameće određenu tehnologiju ili određeni stil razvoja.

Referentna, skalabilna i modularna arhitektura: Poslovne aplikacije izgrađene pomoću Oracle ADF-a imaju mogućnost podešavanja i nadograđivanja u slučaju povećanog opterećenja, odnosno većeg broja poslovnih korisnika. Također, Oracle ADF podržava modularnu arhitekturu za poslovne aplikacije što znači da se nekoliko modula može povezati zajedno kako bi se stvorio cjeloviti složeni ADF program. Ovi se moduli također mogu ponovo upotrijebiti u višestrukim ADF aplikacijama.

Cilj JDevelopera je pojednostaviti posao programera u razvoju aplikacija pružajući pritom vizualna i deklarativna rješenja. Najčešće korišteni prozori i alati za uređivanje, a koje možemo vidjeti na sljedećoj slici su:



Slika 2. JDeveloper IDE

Application navigator pomaže nam u upravljanju sadržajem i povezanim resursima aplikacije. Ovdje se stvaraju novi projekti i izvorne datoteke koristeći opcije dostupne u ovom prozoru.

Application resource panel prikazuje resurse i konfiguracijske datoteke na razini aplikacije. To uključuje podatke o vezi s bazom podataka, datoteke metapodataka koje se koriste za konfiguriranje ADF Business Components-a i tako dalje.

Data control panel, odnosno upravljačka ploča podataka, prikazuje skupove podataka, atribute, ugrađene operacije poput *commit*-a i *rollback*-a i poslovne metode iz *Business Components* sloja kao registar podataka. Izložene stavke s upravljačke ploče podataka mogu se povući i pustiti (tzv. „drag'n'drop“) na korisničko sučelje, što će stvoriti XML datoteku metapodataka za povezivanje poslovnih podataka s korisničkim sučeljem.

Structure window, kao što sam naziv govori, prikazuje strukturni prikaz podataka u onom dokumentu koji je trenutno odabran u aktivnom prozoru, a koristi se za pregled ili uređivanje sadržaja.

Visual editor, odnosno prozor vizualnog uređivača pomaže programerima u vizualnoj izgradnji korisničkog sučelja za ADF aplikacije. Pruža vizualni uređivač za HTML,

JSP, JSF, Facelete, izvorno mobilno sučelje i Java Swing. JDeveloper sinkronizira odabir strukturnog prikaza podataka s *Visual editor*-om i obrnuto.

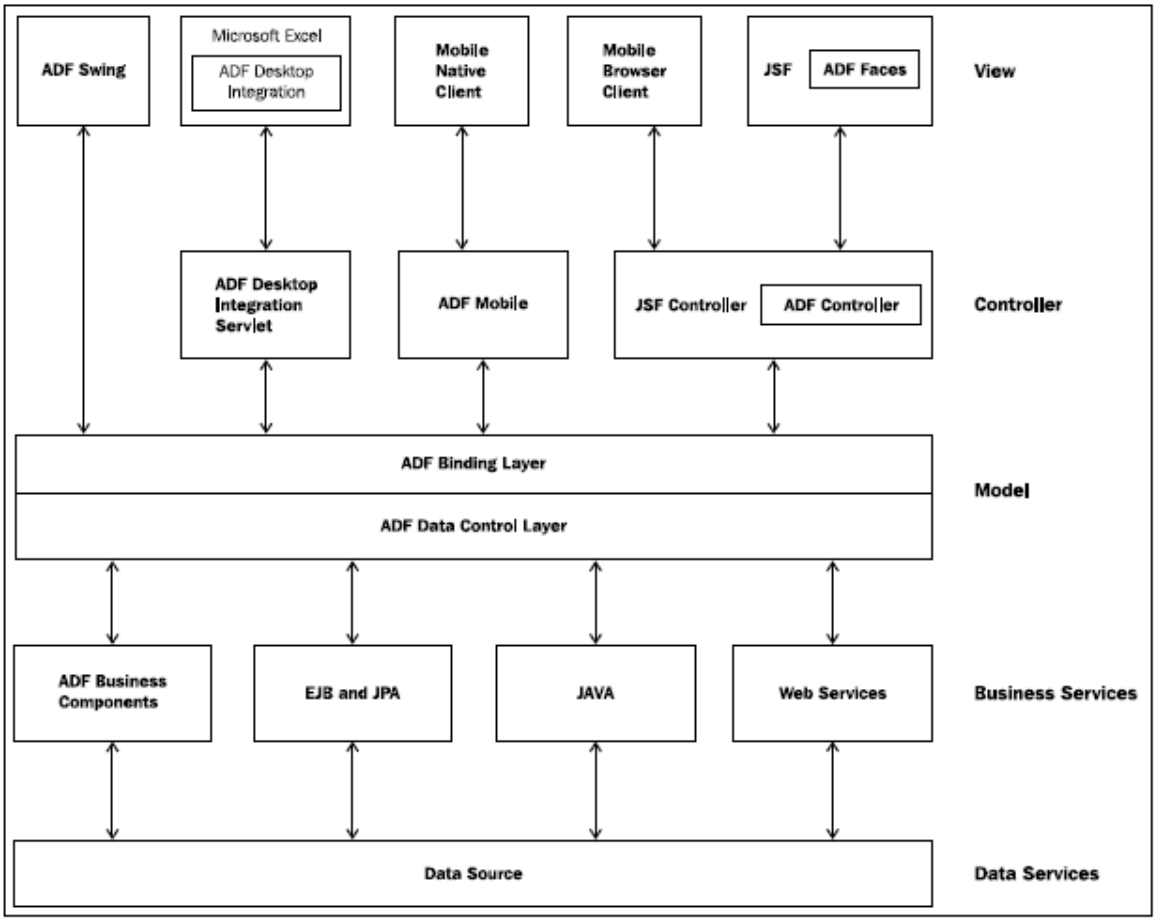
Component palette sadrži palete dostupnih komponenti povezanih s odabranom tehnologijom koja se koristi za dizajniranje stranica ili za definiranje navigacije.

Property inspector je svojstvo trenutno odabranog atributa klase ili komponente koji može utjecati na njegov izgled ili njegove performanse.

Na kraju, tu je i *Log window* koji sadrži zapise različitih komponenti poput kompajlera, audit pravila i debuggera.

5. **Oracle ADF arhitektura**

Oracle ADF ima proširivu i slojevitu arhitekturu, što poboljšava fleksibilnost, održivost i skalabilnost aplikacije. Na sljedećoj slici se može vidjeti ta raznovrsnost i slojevitost ADF arhitekture.



Slika 3. ADF arhitektura (izvor: Purushothaman, 2012)

Kao projektant aplikacija, možete odabrati najbolju tehnologiju uklapanja iz širokog spektra popisa tijekom sastavljanja svakog od slojeva. Na primjer, ADF podržava razne načine izrade *Business Service* sloja, uključujući EJB ili (JPA), web usluge, jednostavne Java objekte i ADF *Business Components* (ADF BC). Na sloju klijenta, aplikacije mogu birati između Java Swinga, Java Server Faces-a (JSF), ADF Faces-a ili ADF Mobile UI-a. Oracle ADF zajedno s JDeveloper IDE nudi dosljedno iskustvo u razvoju kroz različite tehnologije.

5.1. View sloj

View sloj sadrži korisničko sučelje ADF aplikacija i podržava različite prezentacijske kanale kao što su web preglednici, pametni telefoni, tableti i tako dalje. View sloj se temelji na Java Server Faces-u (JSF) i uključuje komponente ADF Faces Rich Client-a, Apache MyFaces Trinidad. Koristeći podršku za razvoj mobilnih uređaja u JavaServer Facesu, ADF aplikacije mogu se prikazivati i na uređajima poput iPhona i dlanovnika (PDA).

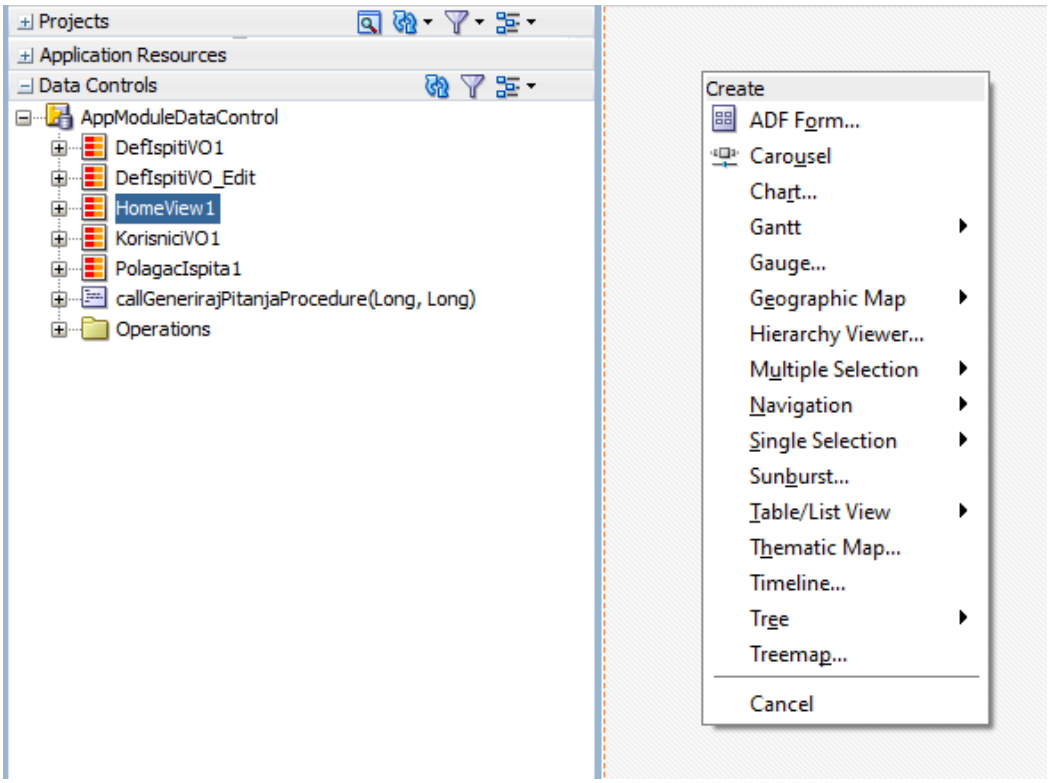
5.2. Controller sloj

Kao što samo naziv sloja govori, *Controller* sloj kontrolira tok aplikacije. ADF Controller koristi se za ADF Faces aplikacije, što pruža olakšano modeliranje navigacije i upravljanje nekom poslovnom aplikacijom. Najveća prednost ADF *Controller*-a nad navigacijskim modelom je ta što poboljšava modularnost sustava dijeljenjem jednog monolitnog navigacijskog modela na više slučajeva navigacije za višekratnu upotrebu poznate kao Task Flow-ovi, u prijevodu tijekovi zadataka, o kojima će biti riječi kasnije u ovome radu. Task Flow-ovi su deklarativna rješenja stoga programeri obično ne trebaju pisati kôd za definiranje navigacije u aplikacijama.

5.3. Model sloj

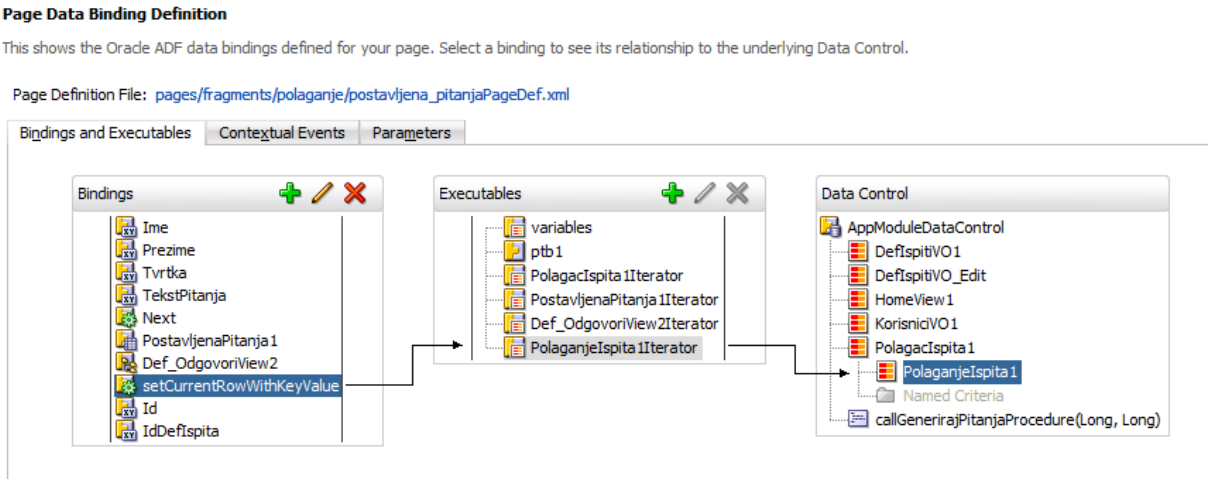
Model sloj povezuje korisničko sučelje s *Business Service* slojem, apstrahirajući detalje implementacije. Model sloj funkcionalno je podijeljen u dvije komponente – *Data Control*, odnosno kontrolu podataka i *Data Binding*, odnosno povezivanje podataka.

Data Control je sinkroniziran s aplikacijskim modulom, odnosno sadrži sve ono što sadrži jedan aplikacijski modul, a to uključuje, *View* objekte, atribute, metode i ugrađene CRUD operacije. Ovaj rad se zapravo i usredotočuje na kontrolu podataka ADF Business Components-a.



Slika 4. Data Control

Sloj Model koristi koncepte koji omogućuju odvajanje korisničkog sučelja od primjene poslovnih pravila. ADF Bindings je sastavni dio sloja Model. Apstrakcija modela podataka putem sloja Model pruža deklarativni pristup u razvoju web aplikacija.



Slika 5. Data Bindings

ADF-ov Model sloj ima vrlo važnu ulogu u čitavoj tehnologiji. Upravo taj sloj zajedno s JDeveloper IDE pruža krajnjem korisniku, odnosno razvojnom programeru vizualno i deklarativno iskustvo u razvoju korisničkog sučelja, bez obzira na tehnologiju koja se koristi za izgradnju *Business Service* sloja.

5.4. Business Service sloj

ADF Business Components pojednostavljuje implementaciju Business Service sloja oslobađajući programera od pisanja infrastrukturnog koda za izradu poslovnih aplikacija. ADF Business Components uglavnom se sastoji od Entity objekata, View objekata i aplikacijskog modula.

Entity objekti su bazirani na tablicama iz baze podataka i dostupni su aplikaciji kao Java objekti.

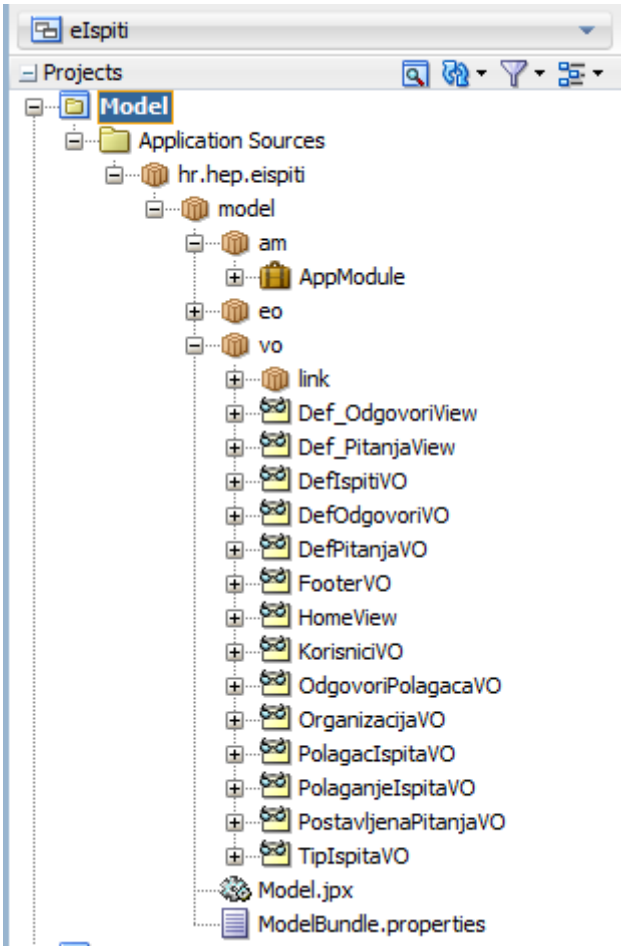
View objekti su temeljeni na entity objektima te mogu biti sastavljeni od jednog ili više entity objekta. Oni sadrže informacije koje će se prikazivati krajnjem korisniku.

Aplikacijski modul sadrži instance view objekata te se osim za testiranje aplikacije koristi i za povezivanje sloja Model sa slojem View preko Data Controlsa, o čemu će biti riječi kasnije u radu.

6. **Kreiranje Business Componentsa**

Zapravo je teško podijeliti kreiranje aplikacije u ADF-u po nekim koracima, budući da se uvijek možemo vratiti na Model, odnosno View ili Controller sloj ovisno o našim potrebama, te ili dodati neki novi view objekt ili iznova napraviti Data Control objekt koji u sebi sadrži View Criteriu koja se odnosi samo na instancu view objekta, a ne na cijeli objekt, ili pak želimo nešto istestirati u aplikacijskom modulu.

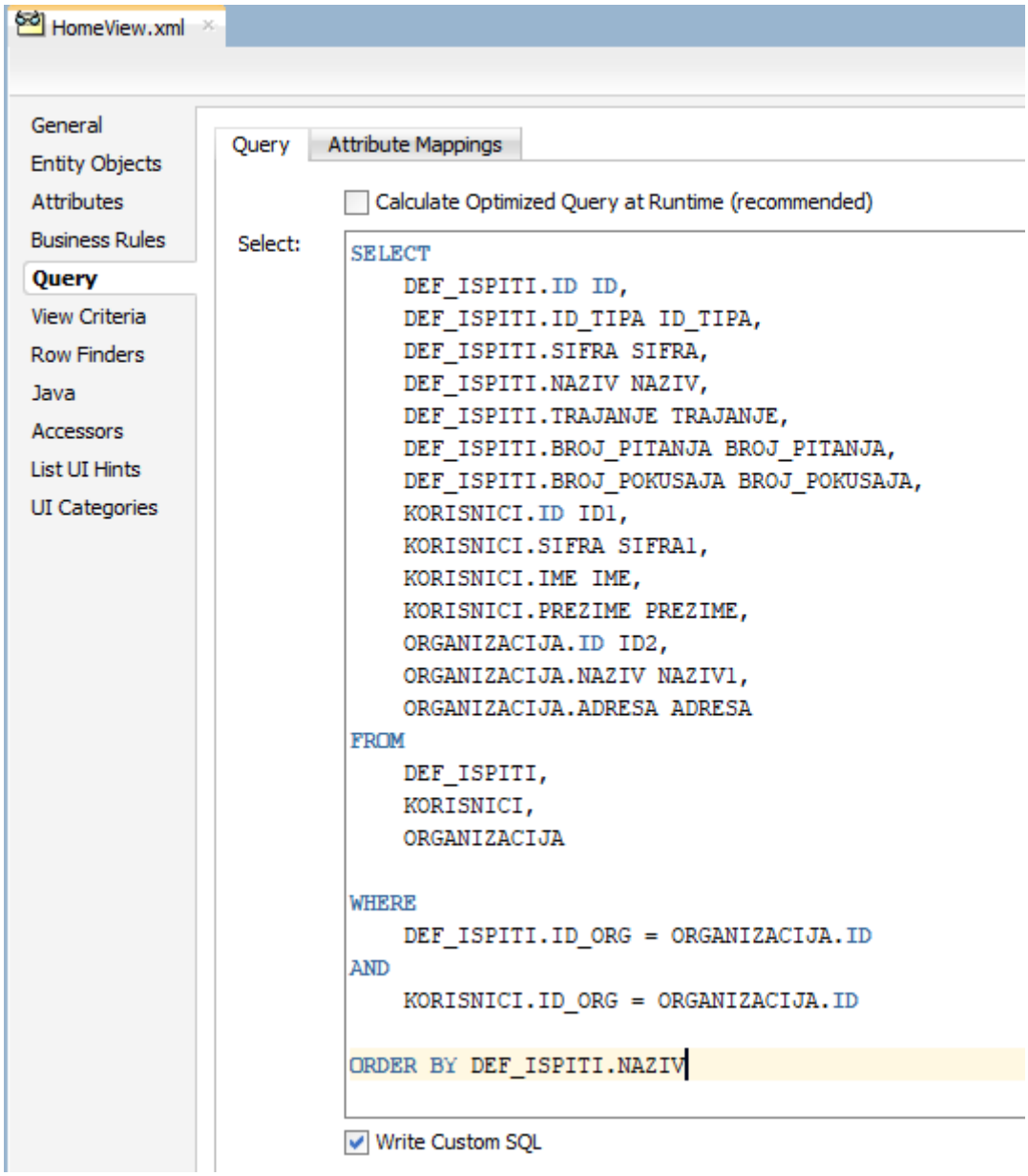
Kao što je ranije navedeno Model sloj povezuje korisničko sučelje s *Business Service* slojem. Na sljedećoj slici možemo vidjeti view objekte koji su dio Model sloja te koji su bazirani na entity objektima, odnosno na pojedinim tablicama iz baze podataka. Također, bitno je napomenuti kako su view objekti međusobno povezani view linkovima. Kada se aplikacija temelji na bazi podataka, atributi view objekta odgovaraju stupcima upita, koji često odgovaraju stupcima tablice. Međutim, u ADF-u postoji i mogućnost da te view objekte sami prilagodimo našim potrebama.



Slika 6. View objekti

Ovdje je to slučaj sa na primjer view objektom HomeView koji nam je potreban kako bi prikazivao relevantne podatke kada se admin prijavi u aplikaciju.

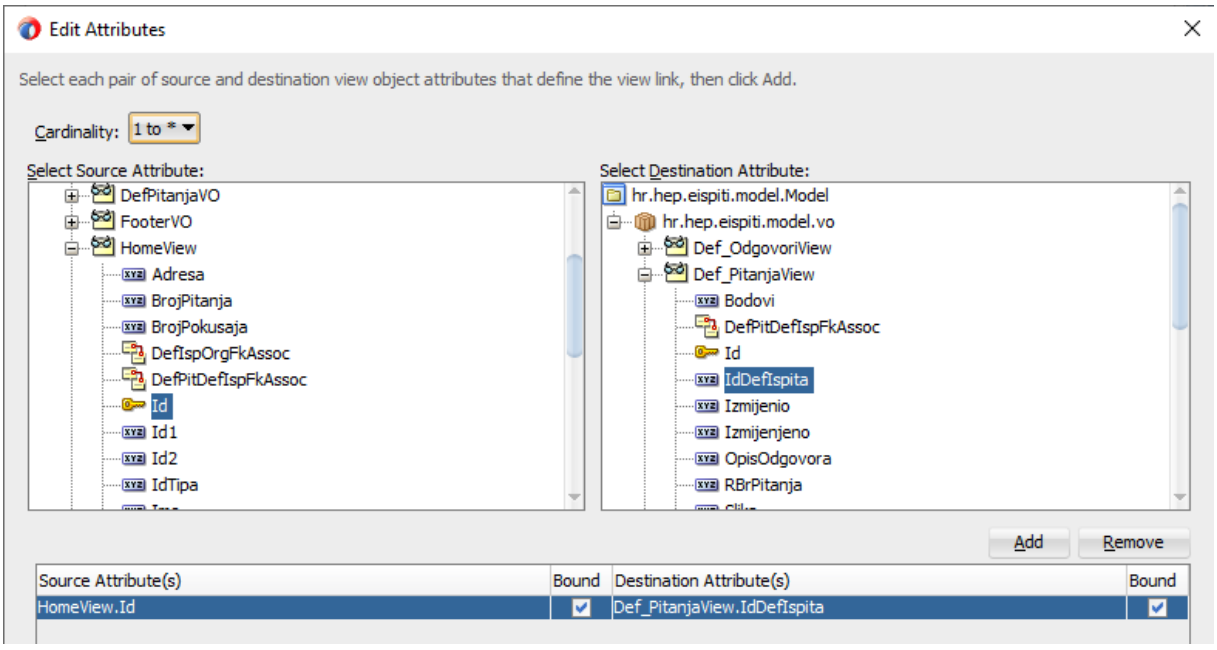
Oracle ADF može automatski povlačiti upite iz baze podataka te dohvaćati rezultate, međutim postoji i mogućnost da sami kreiramo SQL upit prema našim potrebama kao što je to i učinjeno za HomeView objekt.



Slika 7. View objekt baziran na više entity objekata

HomeView objekt je u ovom slučaju baziran na tri entity objekta, a to su Def_ispitiEO, KorisniciEO te OrganizacijaEO, budući da se željelo postići prikazivanje informacija iz sva tri objekta na istoj stranici aplikacije E-ispiti. Osim HomeView objekta dodani su još i Def_OdgovoriView te Def_PitanjaView.

Kao što automatski kreira view objekte, ADF automatski kreira i njihove view linkove koji zapravo predstavljaju relacije među tablicama u bazi podataka. Međutim, ako se view objekt kreira kasnije ili ako je kao u ovom slučaju baziran na SQL upitu potrebno je napraviti view link koji povezuje kreirani view objekt s nekim drugim view objektom. Budući da se sa stranice pregled-ispita.jsff o kojoj će biti riječi kasnije, može ići na editiranje ili kreiranje nekog ispita potrebno je bilo napraviti view link koji povezuje view objekt HomeView sa view objektom Def_PitanjaView što je prikazano na sljedećoj slici.

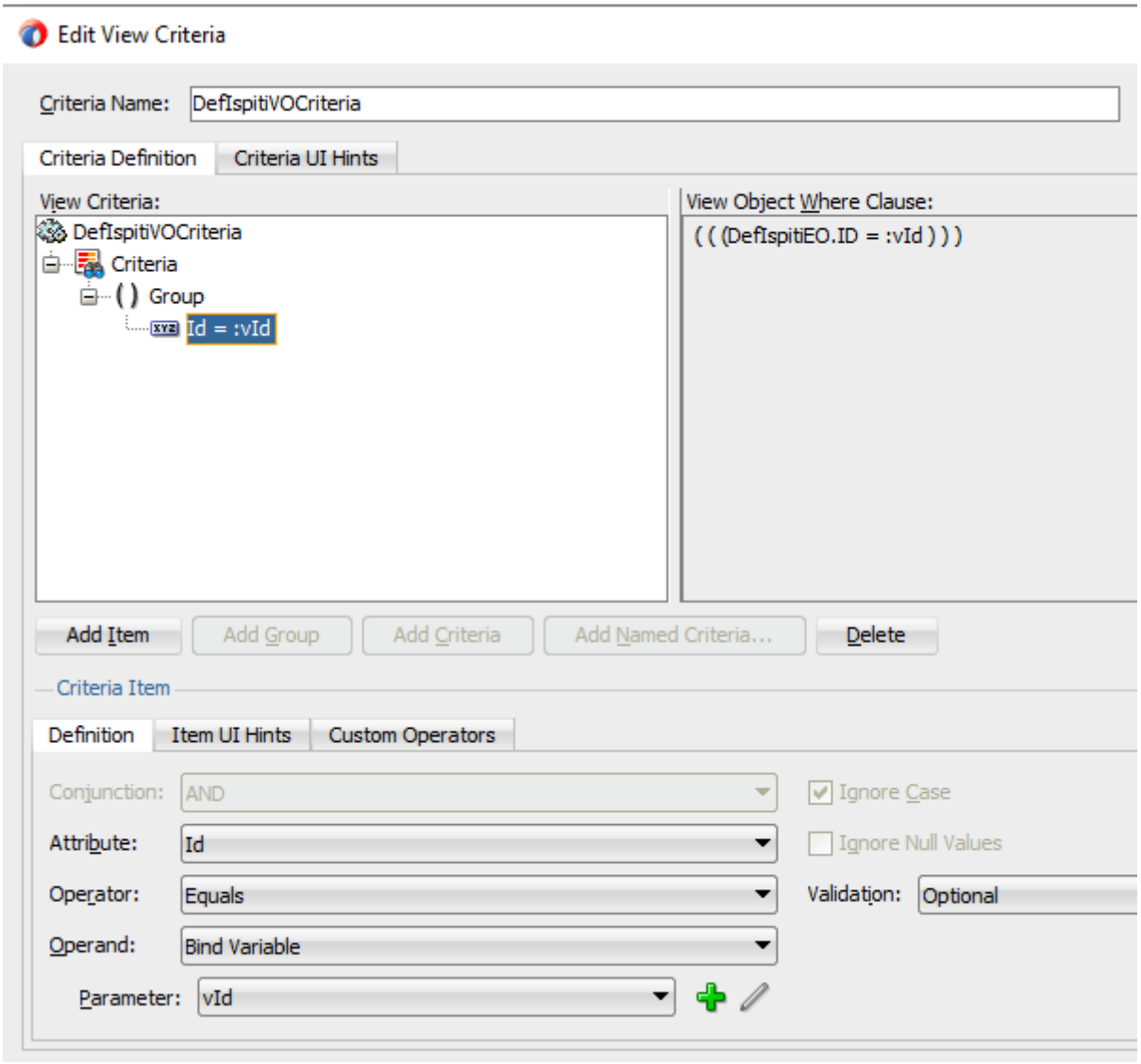


Slika 8. View link

Osim view linkova kreirane su još i view criterie. View criteria je zapravo WHERE uvjet nekog objekta, ali se ne odnosi na cijeli objekt nego samo na instancu view objekta. Na sljedećoj slici je prikazana view criteria na view objektu DefIspitiVO gdje

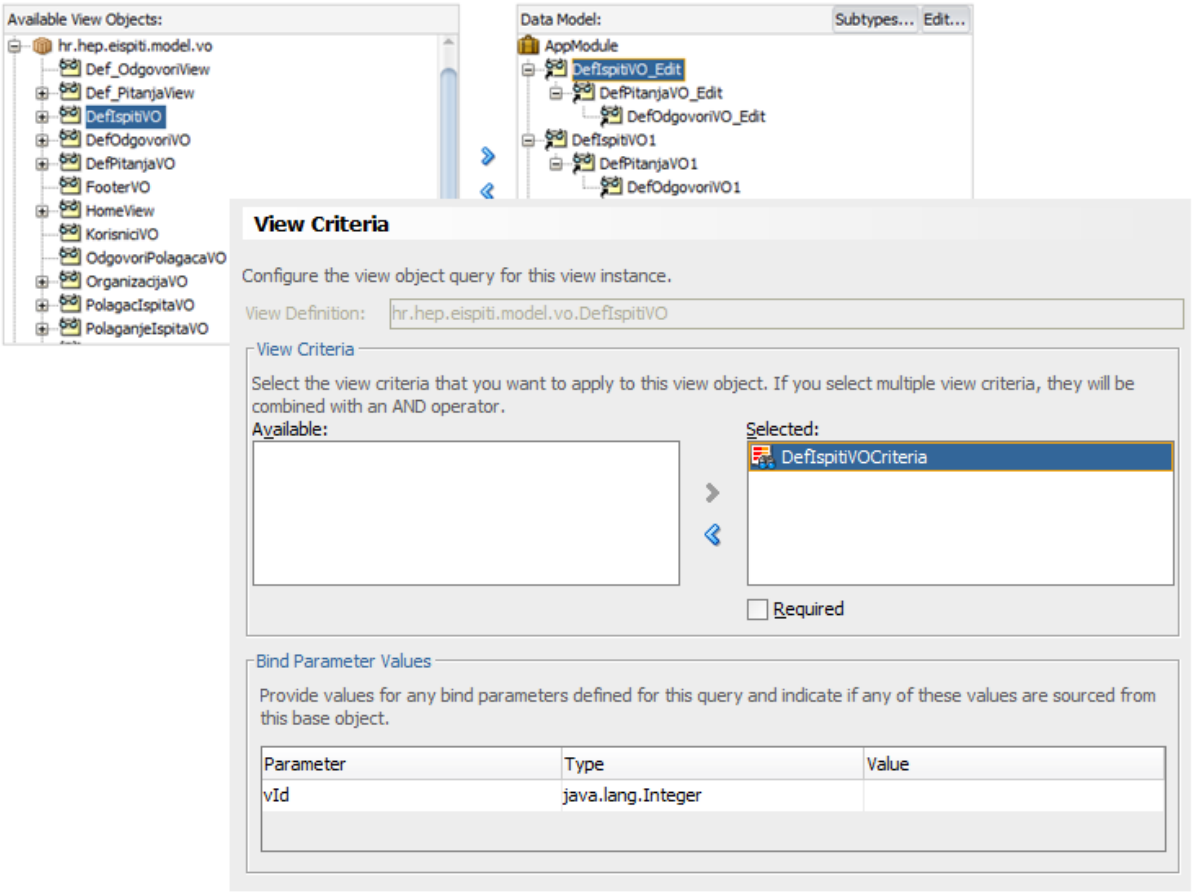
se Id uspoređuje s kreiranom bind varijablom vId, odnosno vrijednosti koju korisnik unese ili odabere.

Bind varijabla se može ubaciti i izravno u WHERE dio upita view objekta kako bi se uključile vrijednosti koje se mogu mijenjati od izvršenja do izvršenja. U tom slučaju, bind varijable služe kao rezervirana mjesta u SQL nizu čija se vrijednost lako može promijeniti tijekom izvođenja bez da se mijenja SQL upit. Budući da se upit ne mijenja, baza podataka može ponovno učinkovito upotrijebiti isti raščlanjeni prikaz upita u više izvršenja, što dovodi do većih performansi izvršavanja same aplikacije.



Slika 9. Kreiranje view criterie

Primjer korištenja view criterie se može vidjeti na sljedećoj slici. U aplikacijskom modulu je umjesto kreiranja novog view objekta sa WHERE uvjetom dovoljno u Data Model „dovući“ njegovu instancu, u ovom slučaju instancu view objekta DefIspitiVO, te mu dodati kreiranu view criteriu DefIspitiVOCriteria i promijeniti naziv instance u DefIspitiVO_Edit. Kao što i samo ime te instance govori, ona nam je potrebna kod editiranja ispita kako bi vrijednosti varijabli ostale zapamćene kada se odabere određeni ispit kojeg se treba izmijeniti.



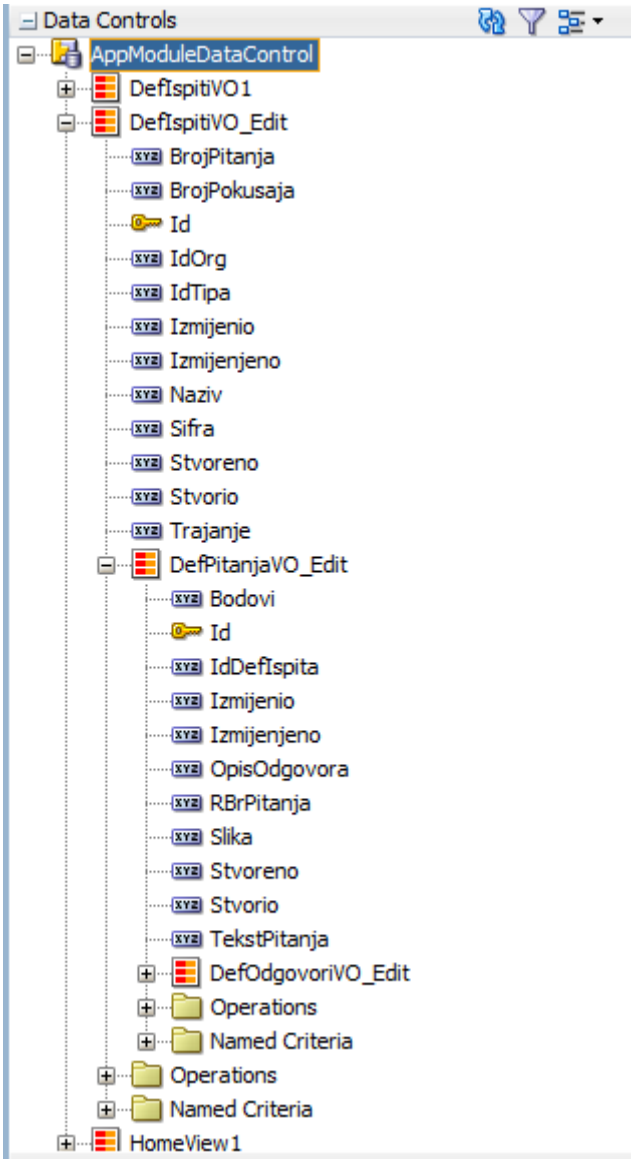
Slika 10. Dodavanje view criterie u aplikacijskom modulu

Naravno i view linkove i view criterie je moguće najprije istestirati u aplikacijskom modulu, bez kreiranja i dizajniranja same aplikacije, što je velika prednost Jdevelopera.

7. Povezivanje ADF Business Componentsa s korisničkim sučeljem

Cilj ovog rada je bio objasniti Business Components sloj ADF Fusion aplikacije koji zapravo povezuje bazu podataka s JDeveloperom. Međutim, budući da je arhitektura ADF-a slojevita, ovdje je fokus na samoj izgradnji poslovne aplikacije i njenog korisničkog sučelja.

Data Controls povezuje Business Components, odnosno aplikacijski modul sa onime što želimo dovući na stranicu i sa čime želimo kreirati korisničko sučelje na način da poziva Business Components sloj i vraća rezultat umotan u generičku strukturu podataka kojega tada koristi Binding sloj. Sljedeća slika prikazuje Data Controls koji sadrži model podataka koji se koristi u aplikaciji E-ispiti.



Slika 11.Master-detail-detail relacije u Data Controls sloju

Ovdje su konkretno prikazane takozvane master-detail-detail instance view objekta DefIspitiVO-DefPitanjaVO-DefOdgovoriVO.

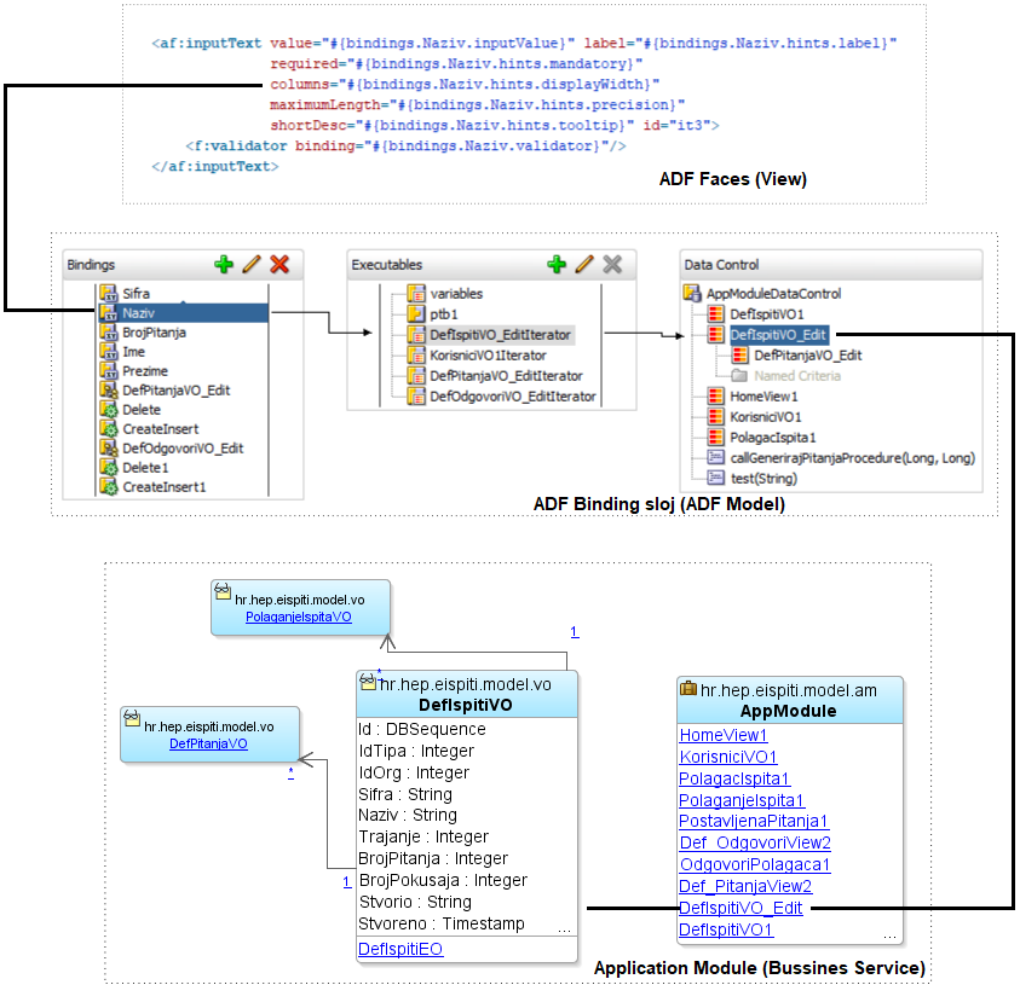
Postoje mnogi slučajevi u kojima se podaci trebaju prikazivati na hijerarhijski način. U odnosu master-detail, kada korisnik promijeni odabranu stavku u master objektu, skup podataka prikazan u detail objektu također se mijenja. Na primjer, odabirom određenog ispita u master objektu trebaju se prikazati samo ona pitanja koja pripadaju tom ispitu u detail objektu.

Binding sloj povezuje korisničko sučelje (UI) s Data Controls slojem, odnosno upravljačkim slojem podataka. Bitno je napomenuti kako UI sloj može biti JSF stranica, Excel tablica, Swing UI i ADF Mobile stranica.

Data Controls je apstraktni sloj koji pristupa sloju poslovnih servisa, tako da programer korisničkog sučelja ne mora znati s kojom je specifičnom tehnologijom implementiran Business service sloj. U ADF-u postoje različiti tipovi poslovnih servisa kao što su Web servisi, URL servisi koji pristupaju podacima preko URL-a, zatim EJB servisi koji pristupaju podacima preko Java EE EJBs-a (*Enterprise Java Beans*), te u slučaju aplikacije E-ispiti aplikacijskog modula koji je baziran na relacijskoj bazi podataka. Ti različiti pristupi podacima su enkapsulirani u Data Controls sloju koji pruža zajedničko sučelje koristeći atribute, kolekcije i operacije koje se koriste u binding sloju. U binding sloju nije važno pristupa li se upitima i ažuriranju podataka preko RDBMS-a, to jest relacijske baze podataka, Web servisa ili nekog drugog servisa.

Prilikom implementacije jednostavne aplikacije s RDBMS pozadinom zapravo se niti ne može vidjeti prava korist tog dodatnog sloja budući da su instance view objekata odmah dostupne u Data Controlsu nakon definiranja podatkovnog modela u aplikacijskom modulu. No to vrijedi samo kada se koriste poslovni servisi temeljeni na aplikacijskom modulu. U tom se slučaju svi meta podaci za Data Controls dohvaćaju iz podatkovnog modela aplikacijskog modula. Prava se prednost može vidjeti tek kada se koristi neka druga vrsta poslovnih servisa, poput Web servisa, za koje su potrebne i dodatne .xml datoteke stvorene za konfiguriranje Data Controls sloja.

Sljedeća slika ilustrira kako je UI komponenta vezana na definiciju atributa Naziv koji je prisutan u view objektu DefIspitiVO.

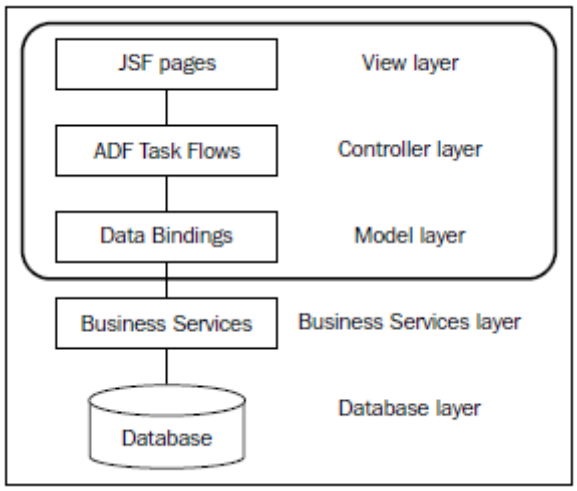


Slika 12.Veza između Business Service, Model i View sloja

Velika prednost kod korištenja ADF frameworka je deklarativno kreiranje korisničkog sučelja kojeg omogućuje takozvani Data Binding sloj. Neke od njegovih funkcionalnosti su navigacija kroz retke u pojedinoj tablici ili formi, ugrađene CRUD operacije i slično.

8. **Kreiranje ADF Controller sloja – Task Flows**

Navigacijom u Oracle Fusion web aplikacijama upravlja takozvani kontroler, koji nudi modularni pristup dizajniranja toka neke aplikacije. Za razliku od standardnog navigacijskog modela JSF stranica, koji za vrijeme izvođenja koristi jedan navigacijski graf definiran u jednoj ili više kopija konfiguracijske datoteke faces-config.xml, ADF Controller sloj omogućava programerima razdijeliti tok neke aplikacije u procese za višestruku upotrebu. Na sljedećoj slici može se vidjeti ADF Controller sloj kojem pripadaju takozvani *task flow*-ovi, odnosno tokovi zadataka koji su jedan od ključnih elemenata bilo koje Oracle ADF aplikacije.



Slika 13. Model View Controler slojevi (izvor: Vesterli, 2014)

Jedan od najvećih izazova s kojim se programeri suočavaju jest kako se nositi sa složenim navigacijskim potrebama ogromnih poslovnih aplikacija. Iako značajke upravljačkog toka koje nudi JSF udovoljavaju osnovnim zahtjevima većine poslovnih aplikacija, programeri će možda trebati uložiti dodatne napore kada su u pitanju stvarno složeni slučajevi upotrebe (*use case*).

Oracle ADF ima podršku za sve takve navigacijske zahtjeve te koristi task flowove koji osim navigacije između stranica omogućuju i uključivanje pozadinske logike koja se ne vidi u korisničkom sučelju, kao što su pozivi metoda, uvjetovanih smjerova i/ili poziva podtokova.

Task flowovi su integralni dio Oracle ADF-a i aplikacija koje su napravljene u Oracle Jdeveloperu. Task flow je modularni te ponovno iskoristivi dio jedinice poslovnog dijela same arhitekture između pogleda (View) te nevizualnih dijelova aktivnosti kao što su routeri i metode.

Kroz sam dizajn task flowovi omogućuju mogućnost ponovnog iskorištavanja dijelova programa, mogućnost mapiranja poslovnih procesa unutar aplikacije te efektivnog slaganja same arhitekture cijele aplikacije. U kontekstu task flowova isti također omogućuju kompletnu slobodu i kontrolu transakcijskog dijela arhitekture u smislu da je programer u direktnoj interakciji s bazom podataka.

Zahvaljujući Oracle ADF task flowovima smo u mogućnosti odvojiti se od standardne JSF arhitekture gdje su pojedine stranice povezane samo i isključivo oslabljenom URL vezom gdje je u ovome slučaju sama suština aplikacije i transakcije puno dublja.

Sama funkcionalnost transakcijskog dijela arhitekture znači da smo, u današnje vrijeme kada korisnik ima sve više zahtjeva u jednoj aplikaciji, u mogućnosti kao programeri dostaviti aplikaciju koja omogućuje veliki dio transakcija čak između „udaljenih" dijelova same aplikacije.

Kao primjer se može navesti HEP-ov admin čiji je zadatak kreiranje i ažuriranje ispita. Primjerice, kod izmjene određenog dijela ispita ili točnije brisanja odgovora, on zasigurno ne bi želio izgubiti sve podatke koje je do danog trenutka unio poput definiranog pitanja ili čak cijelog ispita. Dakle, to je klasični primjer odvojenosti samih transakcija unutar same arhitekture kako bi se u konačnici olakšalo samome korisniku.

Postoje dvije vrste task flowova u ADF-u, a pitanje koje se najčešće postavlja jest da li koristiti *bounded* ili *unbounded* task flow.

Unbounded task flow predstavlja neograničeni tok zadataka i dizajniran je za upravljanje navigacijom između stranica koje ne moraju nužno slijediti bilo koju određenu ulaznu ili izlaznu točku. Na primjer, stranice povezane putem izbornika alatne trake u aplikaciji. Krajnji korisnik može odabrati bilo koju opciju izbornika koja mu se sviđa ili osoba čak može upisati URL izravno u preglednik za pregled stranice i može izaći sa stranice ili prebaciti se na novu stranicu u bilo kojem trenutku.

Navigacijski model za takve stranice (neograničeni prikaz) definira se korištenjem neograničenog toka zadataka. Unbounded task flow definira navigaciju za neograničene prikaze u Fusion web aplikaciji.

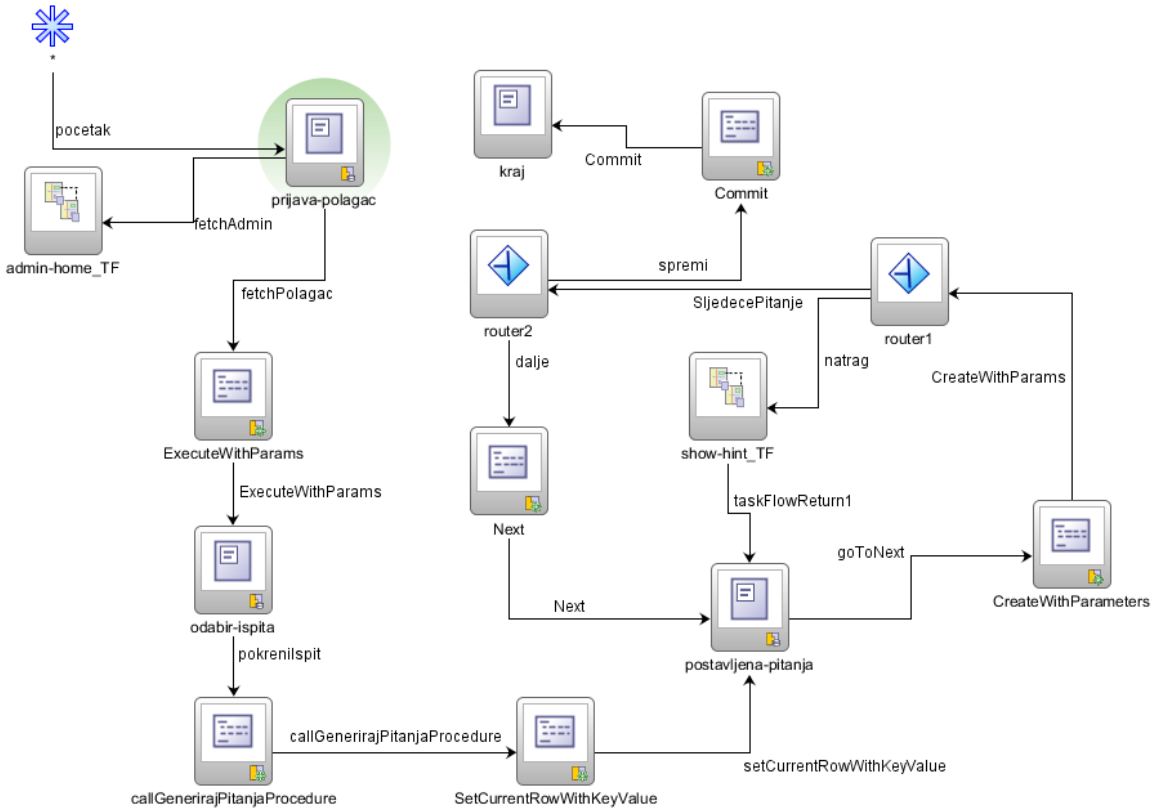
Suprotno od unbounded task flowa, bounded task flow predstavlja ograničeni tok zadataka i prvenstveno je dizajniran za implementaciju navigacijskih slučajeva za višekratnu upotrebu s određenim ulaznim i izlaznim točkama. Bounded task flow sadrži vlastiti skup pravila, aktivnosti i *managed beans*. On može imati samo jednu ulaznu točku međutim može imati više izlaznih točaka, a može se izgraditi pomoću fragmenata stranice (.jsff) ili kompletnih JSF stranica. Također, može preuzeti parametre od pozivatelja prije početka izvršavanja, kao i poslati povratnu vrijednost pozivaocu nakon izvršenja.

8.1. Kreiranje ADF Task Flowova

U aplikaciji E-ispiti korišteni su bounded task flowovi. Na sljedećoj slici prikazan je polaganje-ispita_TF.xml task flow koji je „mozak“ aplikacije budući da se u njemu nalazi sva programska logika koja je potrebna kako bi se odvio jedan e-ispit.

Sadrži tri fragmenta prijava-polagac.jsff, odabir-ispita.jsff i postavljena-pitanja.jsff te dohvaća dva bounded task flowa admin-home_TF i show-hint_TF.

Baziran je na master-detail-detail relaciji view objekata PolagacIspitaVO → PolaganjeIspitaVO (koji se sastoji od entity objekata PolaganjeIspitaEO i DefIspitiEO) → PostavljenaPitanjaVO (koji se sastoji od entity objekata PostavljenaPitanjaEO i DefPitanjaEO) → OdgovoriPolagačaVO i DefOdgovoriView.

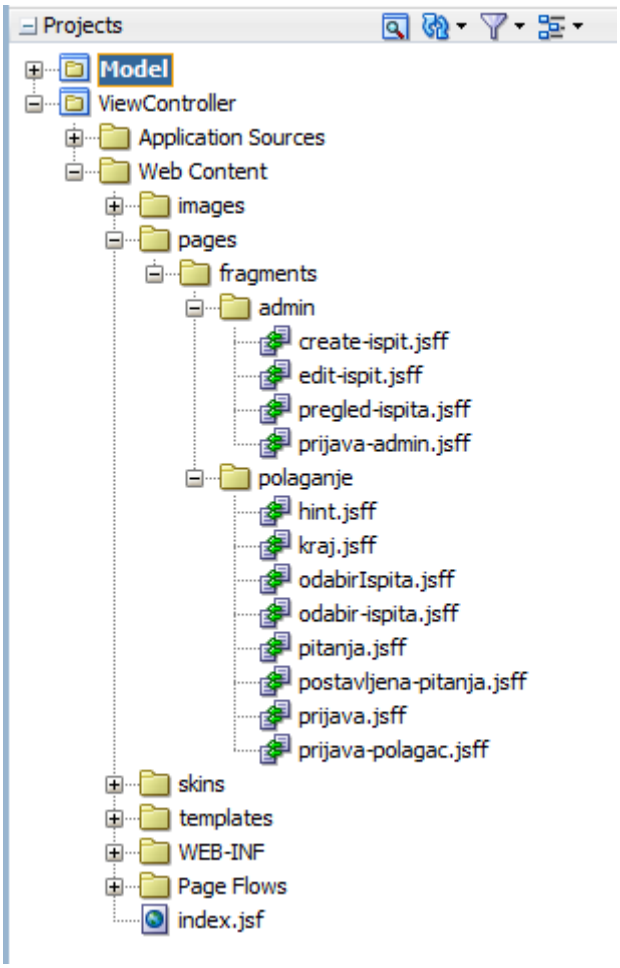


Slika 14.Bounded task flow polaganja ispita

U fragmentu prijava-polagac.jsff kao što sam naziv govori polagač ispita se prijavljuje i sa svojom lozinkom dohvaća novi fragment odabir-ispita.jsff, nakon toga se pokreće ispit gdje se poziva procedura za generiranje pitanja koja će polagač dobiti u ispitu.

.jsff (JSF fragments) stranica je fragment JSF (Java Server Faces) stranice. Ponekad stranice postaju previše složene te ih je u tom slučaju potrebno podijeliti u fragmente kako bi se izbjegle poteškoće u uređivanju i održavanju. Budući da se ne mogu samostalno izvoditi, potrebna je baza .jsf (JSF stranica) ili .jspx (JSP / XML) stranice.

U slučaju e-ispita je kao baza korištena stranica index.jsf dok se ostale .jsff stranice koriste za dizajniranje korisničkog sučelja. Sve fragmente možemo vidjeti na sljedećoj slici.

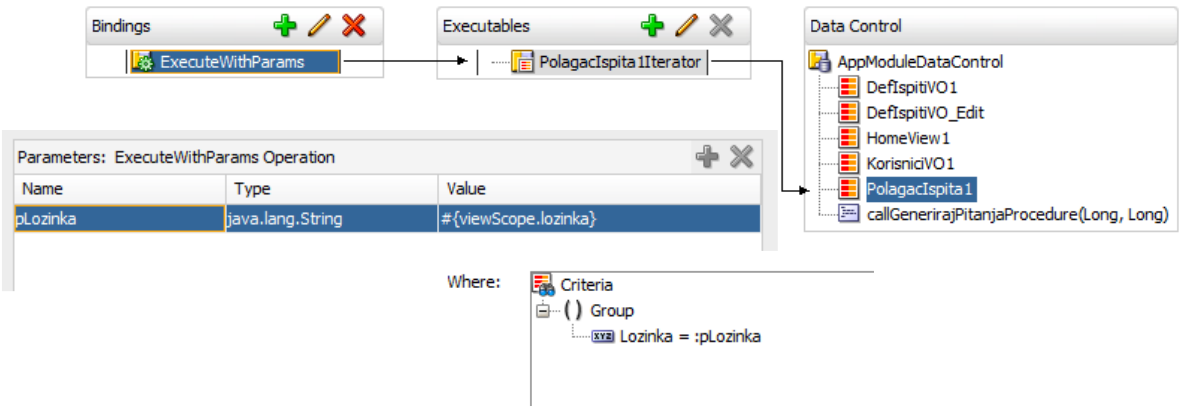


Slika 15. Fragmenti aplikacije e-ispiti

Također, na slici broj 15, osim fragmenata možemo vidjeti i operacije kao što su to ExecuteWithParams, CreateWithParams te Routere.

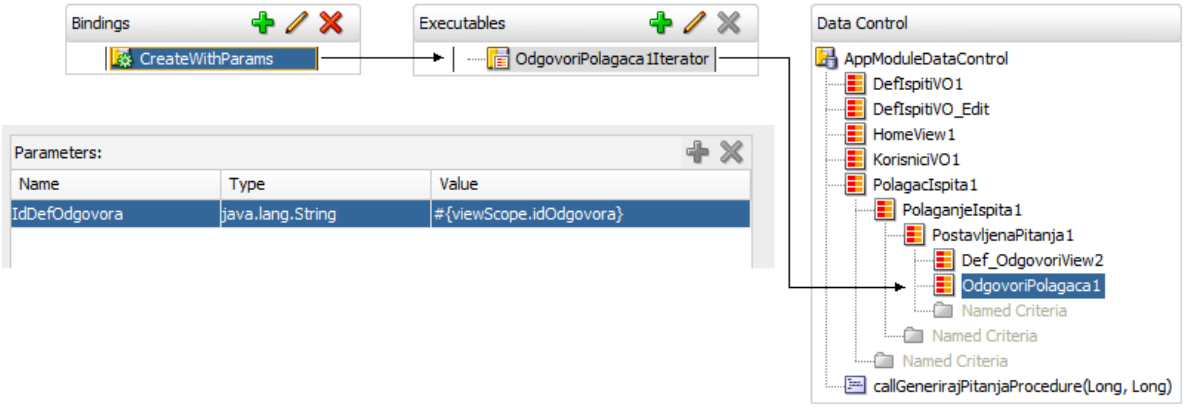
Operacija ExecuteWithParams ima jednu od glavnih uloga u Oracle ADF BC aplikaciji. Na sljedećoj slici možemo vidjeti primjer ExecuteWithParams operacije koja se temelji na tablici PolagacIspita

Kao što je ranije navedeno, u view objektu PolagacIspitaVO dodana je view criteria u where uvjet gdje se provjerava lozinka koju korisnik unese s bind varijablom.



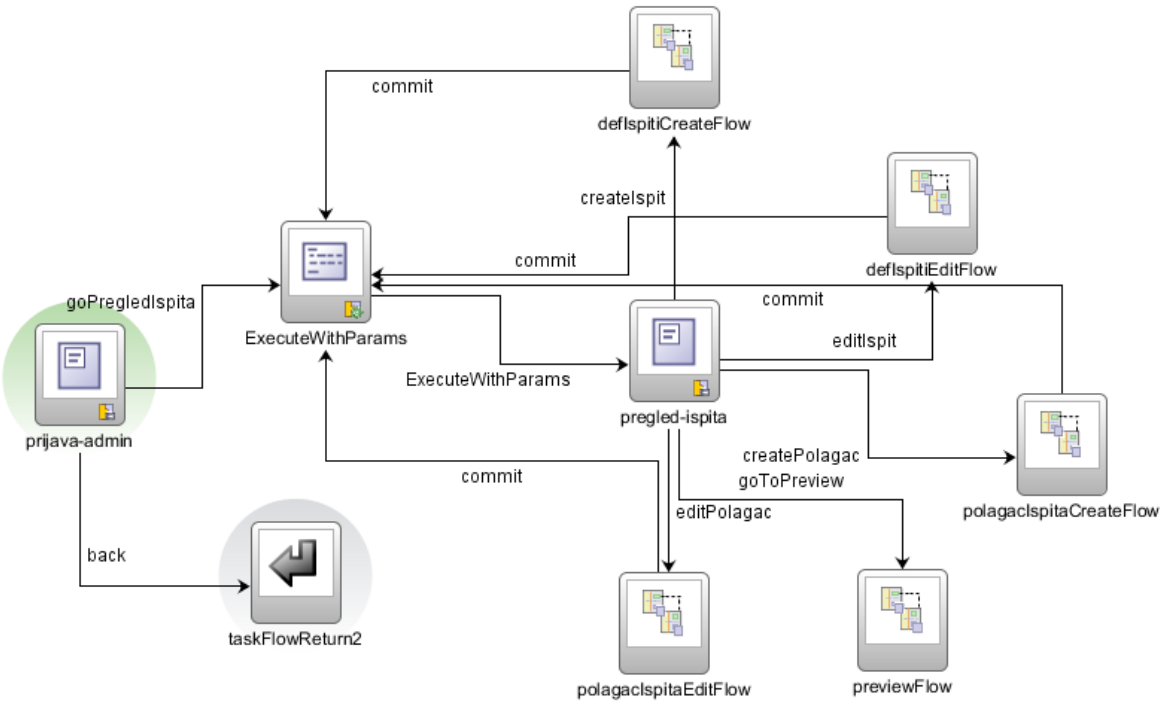
Slika 16. Execute With Parameters

Operacija CreateWithParams je standardna operacija ADF BC Data Control sloja koja nam omogućuje definiranje parametra kako bismo ga dodijelili novom retku za neki određeni atribut. Na primjer, na sljedećoj slici možemo vidjeti parametar IdDefOdgovora iz view objekta OdgovoriPolagaca čija se vrijednost puni preko view scopea #{viewScope.idOdgovora} koji dohvaća Id iz view objekta Def_OdgovoriView te na taj način provjerava da li je polagač točno odgovorio na pitanje ili nije. Za razliku od operacije ExecuteWithParams koja samo uspoređuje vrijednosti atributa.



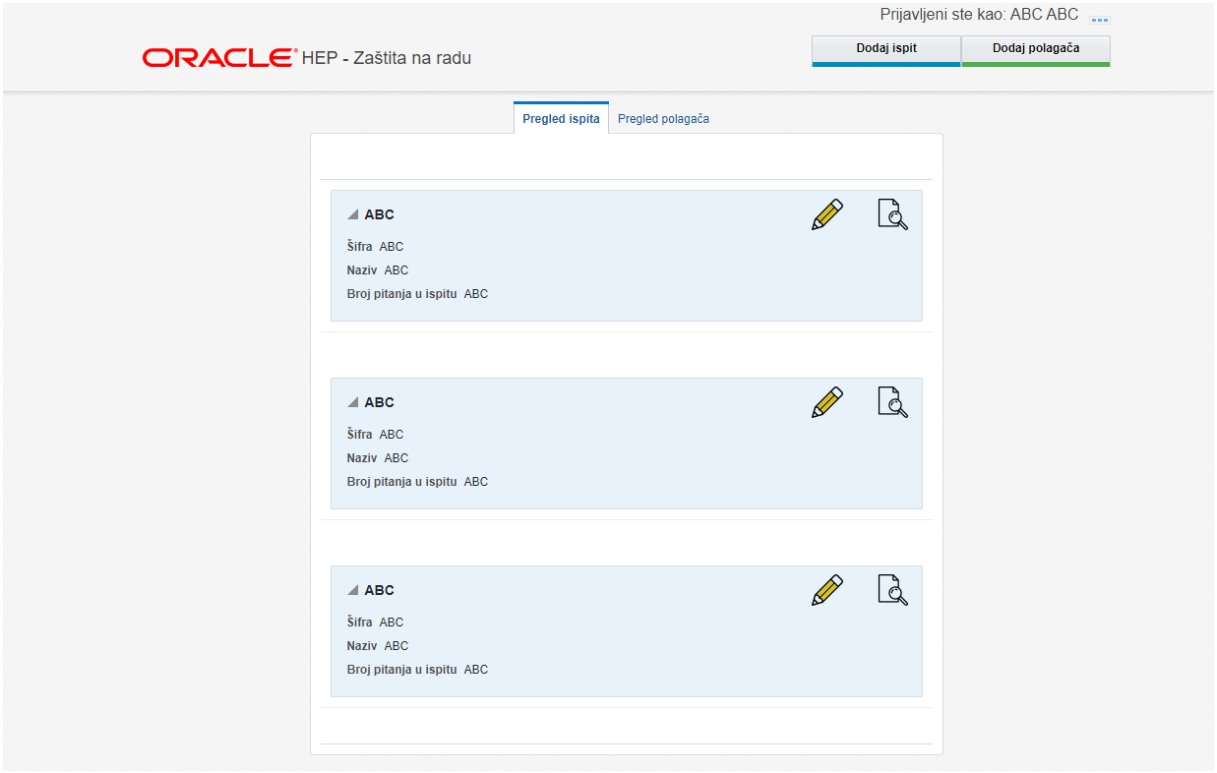
Slika 17. Create With Parameters

Iz task flowa se prelazi u bounded task flow admin-home_TF.xml koji zapravo sadrži osnovne CRUD operacije, poput dodavanja ili editiranja ispita ili polagača i koji se nalazi na sljedećoj slici.



Slika 18. Bounded task flow adminove početne stranice

Bitno je još napomenuti kako ti kreirani fragmenti zapravo predstavljaju stranice korisničkog sučelja. Na sljedećim slikama možemo vidjeti fragmente pregled-ispita i prijava. To je zapravo dizajn prijave u ispit te adminovog korisničkog sučelja u kojem on može pregledavati i ažurirati ispite, odnosno pitanja i odgovore, zatim pregledavati ili mijenjati podatke o polagačima, ali i napraviti prewiev ispita kao što će ga vidjeti i polagač ispita, u tom slučaju sa svim pitanjima, a ne samo sa određenim brojem pitanja.



Slika 19. Dizajn adminovog korisničkog sučelja

Slika 20. Dizajn fragmenta prijave u ispit

Dakle Oracle ADF daje ogroman potencijal samome programeru budući da se obične web stranice povezuju URL-ovima te <a href> tagovima te sama pretraga istih otežava sam rad jedne online aplikacije budući da je potrebno čitanje svakog pojedinog <a href> taga. Java Server Faces je nadišao ovu problematiku implementacijom faces-config.xml datoteke koja pripisuje samu navigaciju između stranica odnosno točno određenih pravila kojima se korisnik koristi. Takva arhitektura omogućava puno lakši pristup navigaciji korisnika unutar same aplikacije.

8.2. Pozivanje procedure iz baze podataka

Programska logika se može jednim dijelom rješavati u bazi podataka. Na taj način se smanjuje opterećenost aplikacije stoga je kod kreiranja aplikacije u ADF-u puno puta potrebno pozvati PL/SQL proceduru za neki određeni zahtjev. U bazi podataka E-ispita postoji jedna takva procedura *generiraj_pitanja* koja metodom nasumičnog odabira polagaču ispita zadaje određeni broj pitanja za određeni ispit.

Na sljedećoj slici su prikazani parametri procedure, odnosno ulazni podaci koji kreiraju objekt, u ovom slučaju tablicu *Polaganje_ispita*, a zatim se dobiveni podaci upisuju u tablicu *Postavljena_pitanja*.

```
procedure generiraj_pitanja (
  p_id_def_ispita IN def_ispiti.id%type,
  p_id_polagaca IN polaganje_ispita.id_polagaca%TYPE,
  p_id_datum IN polaganje_ispita.datum%TYPE,
  p_id_polaganja OUT polaganje_ispita.id%TYPE
) AS
```

Slika 21. Parametri procedure generiraj_pitanja

Ponekad kad pozivamo proceduru potrebno je dodati OUT parametre koji se koriste kako bi se vratila vrijednost procedure. Može postojati *n* OUT parametara, međutim u ovoj je proceduri naknadno dodan OUT parametar koji puni tablicu *Polaganje_ispita* kako bi izlaz iz procedure napunio bazu.

Na sljedećoj slici možemo vidjeti kako izgleda metoda za poziv procedure u Javi kreirana u AppModuleImpl klasi.

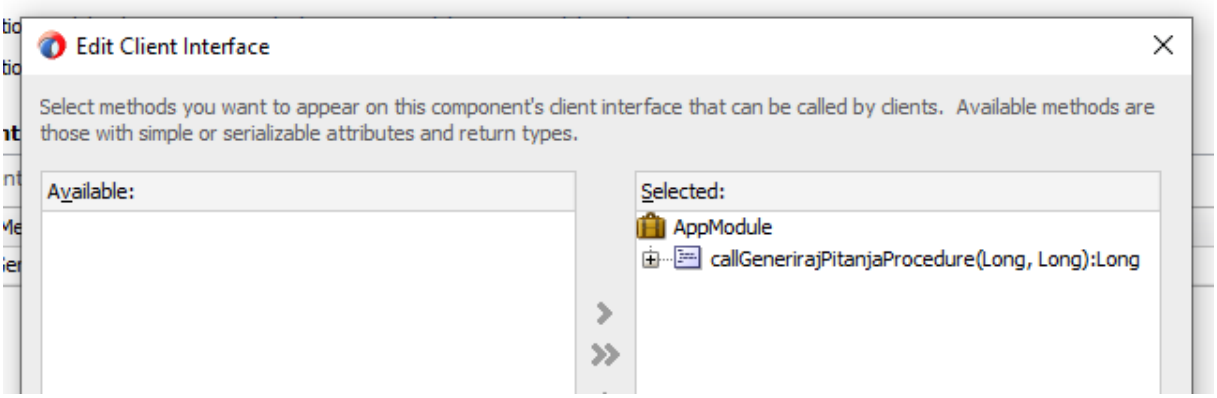
```
public Long callGenerirajPitanjaProcedure(final Long idDefIspita, final Long idPolagaca) {
    System.out.println("--------------------");
    final String procedure = "{call RANDOM.generiraj_pitanja(?,?,?,?)}";

    final DBTransaction dBTransaction = getDBTransaction();
    System.out.println("------------" + idDefIspita + " " + idPolagaca);

    try (final CallableStatement stmt = dBTransaction.createCallableStatement(procedure, 0)) {
        stmt.setObject(1, idDefIspita);
        stmt.setObject(2, idPolagaca);
        stmt.setObject(3, new Date(new java.util.Date().getTime()));
        stmt.registerOutParameter(4, Types.NUMERIC);
        stmt.executeQuery();
        final Long idPolaganjaIspita = stmt.getLong(4);
        System.out.println("--------------------");
        System.out.println("izvršeno");
        System.out.println("--------------------");
        System.out.println(idPolaganjaIspita);
        System.out.println("--------------------");
        return idPolaganjaIspita;
    } catch (SQLException e) {
        System.out.println("--------------------");
        System.out.println("error ");
        e.printStackTrace();
        System.out.println("--------------------");
    }
    return -1L;
}
```

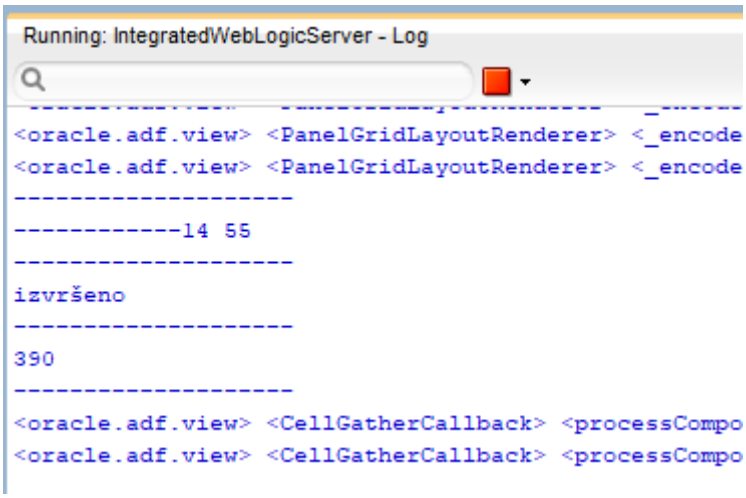
Slika 22.Poziv procedure random

Potrebno je metodu dodati u Client Interface kako bi se ista mogla koristiti u kreiranju korisničkog sučelja.



Slika 23. Dodavanje metode u Client Interface

Nakon što korisnik odabere ispit koji želi polagati to se može istestirati u Message Logu. Na sljedećoj slici možemo vidjeti kako se izvršio poziv procedure u trenutku kada je polagač ispita krenuo rješavati ispit te su mu se izgenerirala pitanja metodom slučajnog odabira.



Slika 24. Testiranje aplikacije u Log-u

9. **Oracle ALTA**

Oracle koristi Oracle Alta UI za razvoj najnovijih verzija Oracle Fusion aplikacija te niza inovativnih mobilnih aplikacija. S manje elemenata i minimalnim dizajnom korisničkog sučelja, Oracle ALTA UI omogućuje brže učitavanje kritičnih sadržaja i pomaže korisnicima da se usredotoče na prikaz podataka. U nastavku će biti razmotreni ključni koncepti kod stvaranja modernog korisničkog sučelja *responsive* aplikacije e-ispiti.

Iz perspektive dizajnera i krajnjeg korisnika, Oracle ALTA UI se temelji se na nekoliko ključnih principa (Bors). Ovo započinje s „Mobile first“ dizajnom radi čega su gumbi, izbornici, tekst i ostale komponente korisničkog sučelja veće budući da veće komponente omogućuju lakše rukovanje na mobilnim telefonima i tabletima. Nadalje, ALTA UI omogućuje dizajniranje jednostavnog i čistog dizajna korisničkog sučelja što omogućuje korisnicima da se usredotoče na podatke, umjesto da moraju misliti o tome gdje uopće pronaći relevantne podatke na stranici. Zapravo Oracle ALTA UI želi privući korisnike vizualnim sadržajem kojima su u današnje vrijeme potrebne instant informacije i koji nemaju vremena u potpunosti čitati web stranice. U usporedbi s drugim dizajnima korisničkih sučelja, Oracle ALTA UI je više grafičke naravi, odmičući se tako od tradicionalnog korisničkog sučelja koje se temelji na tekstu.

S tehničke i razvojne perspektive, Oracle ALTA UI temelji se na korištenju pojednostavljenih struktura komponenata te jednostavnijim prikazom zadržava fokus na podacima. Pored toga, manja iskorištenost grafičkog korisničkog sučelja znači i brže učitavanje sadržaja.

Nadalje, Oracle ALTA UI podržava i web i mobilnu tehnologiju, te pruža fleksibilnost s jednostavnijim i modularnijim komponentama, a samim time i aplikacije modernog izgleda. Za razliku od zadanog Oracle korisničkog sučelja i aplikacija koje koriste Skyros skins, ALTA UI označava lagano korisničko sučelje s jasnim i konciznim bojama.

Tipičan, *defaultni* izgled ADF aplikacije je implementiran u takozvanom *skin*-u skyros koji dolazi s Oracle ADF-om, a koji se nalazi u trinidad-config.xml. Kako bi se u

JDeveloperu (u verziji 12.1.3) mogla koristiti Oracle ALTA UI jednostavno je potrebno promijeniti skyros u alta.

```
<?xml version="1.0" encoding="windows-1250"?>
<skins xmlns="http://myfaces.apache.org/trinidad/skin">
  <skin>
    <id>WorkBetter.desktop</id>
    <family>WorkBetter</family>
    <extends>alta-v1.desktop</extends>
    <render-kit-id>org.apache.myfaces.trinidad.desktop</render-kit-id>
    <style-sheet-name>skins/skin1/WorkBetter.css</style-sheet-name>
    <bundle-name>hr.hep.eispiti.view.skinBundle</bundle-name>
  </skin>
</skins>
```

U aplikaciji E-ispiti se zapravo koristi takozvani prošireni skin, odnosno skin koji sadrži sve ono što sadrži jedan alta skin sa svime onime što se definira u novoj css datoteci, u ovom slučaju nazvanoj skin1.css. Ovo je dobra praksa, budući da se tako novodefinirane klase mogu ponovno upotrijebiti.

10. **Zaključak**

Cilj ovog rada je bio napraviti aplikaciju u JDeveloperu bez pisanja programskog koda u Javi, međutim već za pozivanje procedure iz baze podataka je bilo potrebno napisati metodu u Javi koja poziva tu proceduru, a i vjerojatno postoje i elegantnija rješenja od ovih koja nam nudi deklarativan pristup programiranja.

Budući da u HEP-u imaju riješen sustav prijave, ovaj rad nije bio fokusiran na zaštitu podataka i razinu sigurnosti što je svakako jako bitno u realnom poslovnom svijetu o čemu bi se mogao napisati još jedan rad.

Kao što to u praksi najčešće biva, ovaj bi rad mogao biti jedan početak i prva iteracija od mnogo njih kako bi se stvorila prava online aplikacija koja bi mogla koristiti instruktorima kao i polagačima.

Volio bih na kraju napomenuti kako jedna web stranica ili aplikacija se može naizgled činiti jednostavnom no kada se uđe malo dublje u problematiku, programsku logiku i slično, najjednostavnija aplikacija se može pretvoriti u nešto vrlo zamršeno.

Literatura

Knjige:

PURUSHOTHAMAN, J. (2012) *Oracle ADF Real World Developer's Guide*. Packt Publishing.

RONALD, G. (2011) *Quick Start Guide to Oracle Fusion Development: Oracle JDeveloper and Oracle ADF*. 1st edition. McGraw-Hill Education.

VESTERLI, S. E. (2014) *Oracle ADF Enterprise Application Development – Made Simple*. 2nd edition. Packt Publishing.

Internet izvori:

https://www.doag.org/formes/pubfiles/7371111/2015-K-DEV-Luc_Bors-Oracle_ALTA_UI___Smashing_UI_for_Web_and_Mobile-Manuskript.pdf (preuzeto 12. 8. 2019)

https://www.oracle.com/technetwork/developer-tools/adf/learnmore/adf-task-flow-trans-fund-v1-1-1864319.pdf (preuzeto 15. 5. 2020)

https://www.oracle.com/technetwork/developer-tools/jdev/adf-task-flow-design-132904.pdf (preuzeto 15. 5. 2019)

Popis slika

Sažetak

U radu je prikazana izrada i dizajn korisničkog sučelja za potrebe održavanja online ispita iz zaštite na radu za potrebe TE Plomin. Prije nego što se krene u dizajniranje aplikacije potrebno je analizirati korisničke zahtjeve i bazu podataka na kojoj se temelji aplikacija. Programeri se svakodnevno susreću sa složenim navigacijskim potrebama ogromnih poslovnih aplikacija, međutim Oracle ADF ima podršku za sve takve navigacijske zahtjeve te koristi task flowove koji osim navigacije između stranica omogućuju i uključivanje pozadinske logike koja se ne vidi u korisničkom sučelju, kao što su pozivi metoda, uvjetovanih smjerova i/ili poziva podtokova.

Ključne riječi: E-ispiti, aplikacija, Oracle ADF Business Components, MVC arhitektura, ADF Task Flow

Summary

The Thesis deals with the development and design of a user interface for the needs of holding an online exam on safety at work for necessity of TPP Plomin. Before starting of application design, it is necessary to analyze the user requirements and the database on which the application is based. Developers face the complex navigation needs of huge business applications on a daily basis, but Oracle ADF has support for all such navigation requirements and uses task flows that, in addition to navigating between pages, enable background logic not seen in the user interface, such as method calls. conditioned directions and / or call streams.

Keywords: E-exams, application, Oracle ADF Business Components, MVC architecture, ADF Task Flow