UDC: 37.018.43:004.738.5 Info M: str. 19-27

INTEGRACIJA MOODLE I COALA SISTEMA
AN INTEGRATION OF THE MOODLE AND THE COALA SYSTEM

Ljiljana Gavović, Žarko Stanisavljević
Elektrotehnički fakultet, Univerzitet u Beogradu

REZIME: Sistemi za pomoć u učenju imaju značajnu ulogu u savremenom obrazovnom procesu. Njihov razvoj doživljava ekspanziju početkom ovog veka. Mogu se razlikovati dve grupe ovih sistema: sistemi za upravljanje obrazovnim procesom i laboratorijski sistemi za pomoć u učenju. Tema ovog rada jeste integracija po jednog predstavnika svake od grupa. U radu su najpre opisani Moodle i COALA sistemi, a nakon toga je dat predlog njihove integracije. Rezultat ovog rada je Moodle plugin koji na jednoj web stranici omogućava prikaz pitanja u okviru Moodle testa i COALA simulatora upakovanog u Java applet.

KLJUČNE REČI: elektronsko učenje, Moodle, COALA, LMS, Java

ABSTRACT: E-learning tools have a significant role in the modern educational process. The development of these tools has been expanding from the beginning of this century. There are two groups of these systems: learning management systems and laboratory educational systems. This paper presents an integration of one representative of each group. The paper first describes Moodle and COALA systems and after that proposes one way of their integration. The result of this work is a Moodle plugin that enables the presentation of both Moodle questions and COALA simulation within the same web page.

KEY WORDS: e-learning, Moodle, COALA, LMS, Java

1. UVOD

Obrazovanje se danas smatra uslovom opstanka i razvoja modernog društva. S razvojem modernog društva menjaju se i uslovi učenja i obrazovanja, pa se tako, u cilju da se obrazovanje učini dostupnim svima, razvija sistem učenja na daljinu. U početku učenje na daljinu izvodilo se korišćenjem pošte, a zatim, sa razvojem tehnologije, počinju da se koriste i radio i televizija, da bi se početkom XXI-og veka, uporedo sa razvojem Interneta, razvilo i elektronsko učenje, kao jedan vid učenja na daljinu koji se obavlja putem Interneta.

U poslednjim godinama povećava se broj univerziteta koji koriste elektronsko učenje, pa tako studenti preko Interneta mogu pohađati kurseve sa mnogih svetskih univerziteta, poput *Stanford*-a [1], *Cambridge*-a [2], *Harvard*-a [3], *MIT*-a [4] i drugih. Pored toga postoje i posebni sistemi koji na jednom mestu nude kurseve sa različitih univerziteta, primer su *Courser*-a [5], *eDX* [6], *Iversity* [7]. Za neke od ovih kurseva koji se mogu pohađati putem Interneta mogu se dobiti i sertifikati.

Na Elektrotehničkom fakultetu u Beogradu elektronsko učenje se pre svega koristi za razmenu materijala i evaluaciju rada studenata na laboratorijskim vežbama. Za sada još uvek ne postoji nijedan predmet koji se može položiti na dajinu, najbliži tome je kurs Računarske mreže na kome su svi materijali, uključujući i video predavanja, dostupni elektronski, dok se ispit, iako se radi elektronski, ne može raditi van zgrade fakulteta.

Jedan od pionira, a danas i jedan od vodećih sistema u oblasti sistema za upravljanje obrazovnim procesom, je *Moodle* sistem [8].*Moodle* predstavlja *web* sistem koji se može koristiti iz *web* pretraživača bez potrebe za ikakvim dodatnim softverom. Korišćenje sistema je intuitivno, a s obzirom da je *Moodle* softver otvorenog koda njegovo unapređivanje i konfigurisanje je dostupno svima. Značajno je i to što veliki broj ljudi učestvuje u unapređivanju, tako da se svi problemi ili nedoumice mogu rešiti u interakciji sa drugim članovima *Moodle* zajednice. Zbog svega prethodno navedenog *Moodle* sistem se koristi kao pomoćno sredstvo u učenju na mnogim univerzitetima. Na Elektrotehničkom fakultetu u Beogradu ovaj sistem se prvenstveno koristi za evaluaciju rada studenata [9] i to na kursevima: Zaštita podataka, Algoritmi i strukture podataka, Računarske osnove interneta i Računarske mreže.

Uporedo sa razvojem sistema za upravljanje obrazovnim procesom razvijaju se i laboratorijski sistemi za pomoć u učenju. Osnovna namena ovih sistema jeste da olakšaju studentima razumevanje gradiva, sa posebnim akcentom na praktičan rad. Na Elektrotehničkom fakultetu u Beogradu takvi sistemi su razvijeni za potrebe predstavljanja rada procesora i drugih elemenata računara na predmetima: Osnovi računarske tehnike, Arhitektura računara, Arhitektura i organizacija racunara, za rad sa modelom entiteta i veza u sklopu kursa Baze podataka, kao i za vizuelnu reprezentaciju različitih tipova algoritama na predmetima: Algoritmi i strukutre podataka, Multiporcesorski sistemi i Zaštita podataka.

Na kursu Zaštita podataka koristi se COALA sistem [10], [11], [12]. Ovaj sistem se koristi za vizuelnu reprezentaciju kriptografskih algoritama i omogućava praćenje izvršavanja pet vrsta algoritama, pri čemu prikazuje detalje izvršavanja za svaki od podržanih algoritama. Svi parametri izvršavanja su konfigurabilni kako bi različiti primeri izvršavanja algoritama mogli da se prikažu lako i brzo. Na ovaj način, praćenjem različitih scenarija izvršavanja algoritama studenti mnogo lakše i bolje razumeju gradivo iz ove oblasti.

Kurs Zaštita podataka drži se na Elektrotehničkom fakultetu u Beogradu na Odseku za softversko inženjerstvo i Odseku za računarsku tehniku i informatiku. Cilj ovog kursa je upoznavanje studenata sa mehanizmima koji se primenjuju u oblasti zaštite podataka, aplikacijama koje se koriste kako bi se obezbedili sigurnosni servisi i načinima primene tih aplikacija. Na ovom predmetu uvedene su laboratorijske vežbe čiji

je cilj da se studentima olakša savladavanje gradiva koje se odnosi na kriptografske algoritme.

U toku ovih laboratorijskih vežbi student dobija skup pitanja na koja treba da odgovori. Jedan deo ovih pitanja koncipiran je tako da usmeri studente da isprate ceo proces šifrovanja i dešifrovanja. Kako bi lakše odgovorili na data pitanja i savladali algoritme studentima je omogućeno korišćenje COALA sistema. Prvih godina korišćenja COALA sistema studenti su odgovore davali na papiru, ti odgovori su kasnije pregledani i ocenjivani. Od školske 2013/2014 pitanja su prebačena u *Moodle* sistem.

Kako se u toku izrade laboratorijske vežbe koriste i *Moodle* i COALA sistem došlo se na ideju njihove integracije.Sama integracija mogla bi se razviti u više pravaca. Prvi mogući način je povezivanje prethodno navedena dva sistema tako da se pitanja i COALA prikazuju u istom prozoru. Na ovaj način studentima se olakšava odgovaranje na pitanja. Više nije potrebno menjanje fokusa između ova dva prozora, čime se smanjuje mogućnost slučajne greške pri odgovaranju. Pored ovoga olakšava se i sam proces administracije. Pri izmenama i unapređenju COALA sistema više nije neophodno ponovo instalirati sistem na svim računarima u laboratoriji, dovoljno je samo promeniti verziju na serveru.S druge strane, sistemi bi se mogli povezati tako da u proveri tačnosti odgovora učestvuju oba sistema. Kada je potrebno proveriti da li je student tačno odgovorio *Moodle* sistem bi iz COALA sistema dobijao tačan odgovor, a zatim vršio potrebne provere. Na ovaj način olakšava se posao nastavnika, jer više nije potrebno definisanje tačnog odgovora pri samom kreiranju pitanja, tako da nastavnik ne mora da prolazi kroz ceo algoritam, za konkretne vrednosti, kako bi kreirao pitanje.

Ostatak rada organizovan je na sledeći način. U drugom poglavlju prikazan je pregled *Moodle* sistema. U trećem poglavlju predstavljeni su podržani algoritmi, struktura i način korišćenja COALA sistema. U četvrtom poglavlju dat je opis implementacije i objašnjene su izmene koje je bilo potrebno načiniti u oba sistema prilikom integracije.Peto poglavlje obuhvata kratak rezime rada uz prikaz mogućih unapređenja.

2. MOODLE

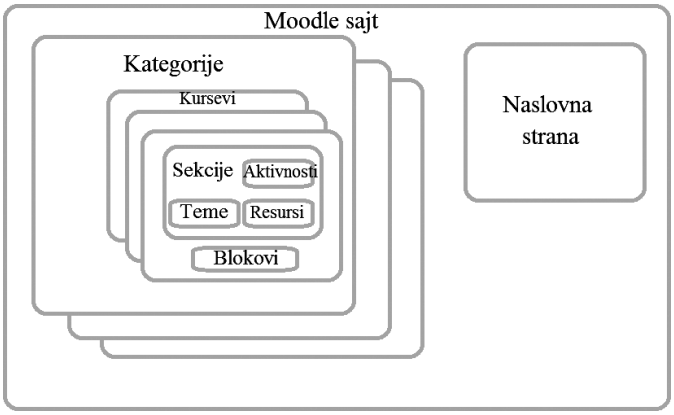
U ovom poglavlju dat je opis *Moodle* sistema. Najpre su opisane glavne odlike arhitekture sistema,zatim je prikazana struktura *Moodle* sajta i struktura direktorijuma, kao i uloge ovog sistema ina kraju je prikazan pregled *Moodle plugin*-ova.

A. ARHITEKTURA

Moodle je primer LAMP aplikacije [13]. LAMP aplikacije su aplikacije pisane u *web* skripting jeziku koje koriste SQL bazu za čuvanje podataka i mogu se izvršavati na bilo kom operativnom sistemu. Pisan je u PHP-u i to je jedina komponenta LAMP arhitekture koja ne može biti zamenjena. Trenutna verzija *Moodle* 2.9 zahteva minimum PHP 5.4.4.*Moodle* sloj baze podataka pisan je korišćenjem *PHPADOdb* biblioteke koja omogućava transparentan pristup sistemu baze podataka. Zahvaljujući tome *Moodle* pruža podršku za korišćenje različitih baza podataka uključujući: *MySQL*, *PostgreSQL*, *MicrosoftSQL*, *MariaDB* i *Oracle*. Iako pruža podršku za različite baze podataka većina sistema koristi *MySQL* bazu. Iako *Moodle* podržava sve operativne sisteme koji imaju odgovarajuću verziju PHP-a i bazu podataka, najbolje performanse se postižu na *Linux* operativnom sistemu. Što se *web* servera tiče *Moodle*se može pokrenuti na bilo kom koji podržava zahtevanu *php* verziju. Najčešće korišćeni *web* server je *Apache*, a nakon toga IIS.

B. STRUKTURA

Prvo što korisnik vidi pri pristupu *Moodle* sajtu jeste naslovna strana. Sama stranica se može dosta menjati i prilagođavati i na njoj se uglavnom nalaze informacije o osnivaču. Osnovna strukturalna celina *Moodle*-a jesu kursevi koji su zapravo stranice na kojima nastavnici mogu podeliti nastavne materijale i aktivnosti studentima. Mogu imati različite *layout*-e, ali uglavnom uključuju centralne sekcije i blokove. Unutar sekcija prikazane su aktivnosti i resursi koji mogu biti organizovani u teme (ili na neki drugi način, npr. nedeljni format). Blokovi su smešteni sa strane i omogućavaju dodatne funkcionalnosti i informacije [14].Kursevi su organizovani u kategorije. Struktura *Moodle* sajta prikazana je na slici 2.1.



Slika 2.1. *Struktura Moodle sajta*

C. STRUKTURA DIREKTORIJUMA

Svaki *Moodle* sistem može biti podeljen u tri celine:

- *Moodle* kod - predstavlja jedan direktorijum, sa poddirektorijumima u kojima su smešteni moduli,
- *Moodle* baza podataka - mesto gde se čuvaju informacije o korisnicima, kursevima itd,
- *Moodle* direktorijum za podatke - mesto gde se čuvaju fajlovi koje su korisnici otpremili na *Moodle*.

M u *Moodle* potiče od reči modularan, a ta modularnost može se uočiti i u strukturi direktorijuma koda. Svaki poddirektorijum u korenom direktorijumu predstavlja jednu veću komponentu *Moodle*-a. Veliki broj tih komponenti podržava

plugin-ove. Svaki *plugin* ima svoj direktorijum unutar direktorijuma glavne komponente. Moduli se instaliraju kopiranjem koda *plugin*-a u odgovarajući direktorijum. Pri sledećem logovanju administratora *Moodle* detektuje i instalira dodati kod.U bazi se čuva većina informacija na sajtu. U bazi se takođe čuvaju i objekti kreirani pomoću *Moodle*-a, npr. *Moodle* omogućava kreiranje stranica za kurseve. Kod ovih stranica je zapravo smešten u bazi, a pored toga u bazi se čuvaju i: veze, podešavanja, sadržaj foruma, kvizovi.

D. ULOGE

Uloga je skup dozvola definisanih na nivou sistema koja može biti dodeljena određenom korisniku u okviru konteksta. Kombinacija uloga i konteksta definiše mogućnost korisnika da radi nešto na bilo kojoj strani.

Neke od uloga u okviru *Moodle*-a [15]:

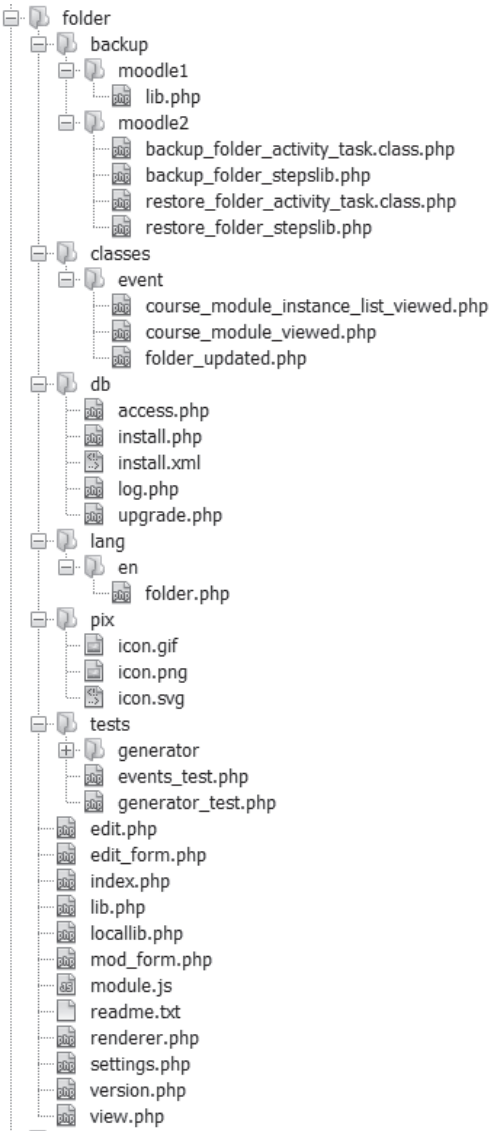
- administrator sajta - ima dozvolu da radi bilo šta na sistemu, privilegije ove uloge se ne mogu menjati; može biti dodeljena korisniku od strane drugog administratora, i samo od strane drugog administratora može biti editovana (ili brisana); prvobitni administrator (kreiran pri kreiranju sajta) ne može biti obrisan,
- menadžer - standardno ova uloga može pristupati kursevima i menjati ih, a može obavljati određene poslove na administrativnom nivou u vezi sa kursevima, korisnicima, podešavanjima itd; može biti dodeljena na nivou sajta ili na nivou kategorije; osnovna razlika u odnosu na administratorsku ulogu je u tome što se privilegije dodeljene ovoj ulozi mogu menjati,
- kreator kursa - može kreirati kurseve; ukoliko je podešavanje "Uloga kreatora kursa u novom kursu" postavljeno na nastavnik onda je kreator kursa prijavljen kao nastavnik na kursu koji je kreirao; ova uloga se najčešće dodeljuje glavnom nastavniku, rukovodiocu odseka ili koordinatoru programa,
- nastavnik - može raditi bilo šta na kursu (dodavati i menjati aktivnosti, ocenjivati studente itd.); može dodeljivati uloge nastavnika, nastavnika bez prava izmene i studenta; dodeljuje se na nivou kursa,
- student - može pohađati kurs, aktivnosti, pregledati materijale, videti svoje ocene; način na koji student pristupa kursu i koje su njegove privilegije određuju nastavnici i administratori.

Uloge nastavnika, studenta, itd.ne biraju se pri registraciji, već su dodeljene od strane administratora.

E. PROŠIRIVOST

Osnovna tačka proširenja *Moodle*-a su *plugin*-ovi [16]. Najlakši način za dodavanje nove funkcionalnosti u *Moodle* jeste pisanje jednog od standardnih tipova *plugin*-ova. Ukoliko nijedan od standardnih tipova ne odgovara može se napisati lokalni *plugin*. Neki od standardnih tipova *plugin*-ova su:moduli aktivnosti, blokovi, formati kurseva, filteri, načini ocenjivanja, ponašanje pitanja, tipovi pitanja, formati pitanja, pravila o pristupu kvizovima, izveštaji u vezi kvizova.

Moduli aktivnosti predstavljaju najznačajniji tip *plugin*-a, obezbeđuju aktivnosti u kursu i smešteni su u /mod direktorijumu. Svaki moduo smešten je u odvojenom direktorijumu i sastoji se od obaveznih fajlova i svih ostalih fajlova potrebnih programeru. Na slici 2.2 prikazana je struktura folder *plugin*-a.



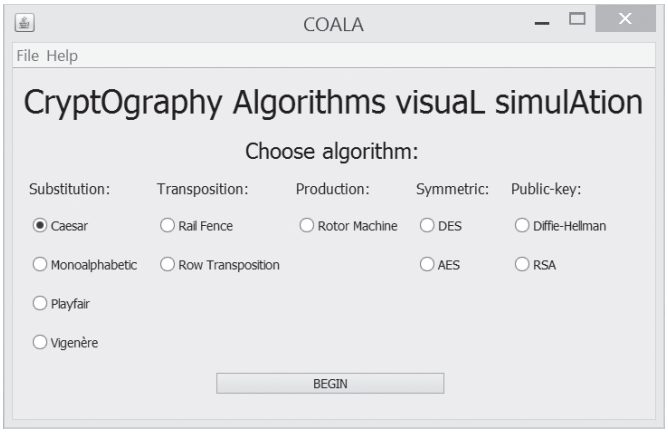
Slika 2.2. *Struktura folder plugin-a*

Blokovi omogućavaju prikazivanje blokova informacija ili alatki. Mogu biti smešteni bilo gde na stranici, ali se najčešće nalaze sa leve ili desne strane, takođe postoji mogućnost pisanja nove teme koja će upravljati njihovim položajem. Ukoliko je to potrebno mogu biti prikazivani na svim stranicama unutar određenog konteksta (npr. u okviru jednog kursa, ili aktivnosti). Njihov kod smešten je u /blocks direktorijumu.Formati kurseva određuju izgled kursa (prikaz glavne stranice, generisanje stabla za navigaciju, itd.). Njihov kod smešten je u / course/format direktorijumu.Filteri se koriste za automatsku transformaciju sadržaja pre njegovog prikaza, obrađuju sav

tekstualni sadržaj na stranici (resurse, korisničke profile, itd.), mogu promeniti format teksta, reči, ugraditi medije itd. Njihov kod smešten je u /filter direktorijumu.Načini ocenjivanja obezbeđuju napredne načine ocenjivanja aktivnosti. Standardno je uključen *Rubrics plugin*, koji obezbeđuje ocenjivanje na osnovu kriterijuma, ali ukoliko je to potrebno mogu se definisati i novi *plugin*-ovi. Njihov kod treba smestiti u /grade/grading/form direktorijum.Ponašanje pitanja definiše načine na koji studenti interaguju sa pitanjima u toku kursa. Njihov kod smešten je u /question/behaviour direktorijumu.Tipovi pitanja daju mogućnost za definisanje različitih tipova pitanja koja se mogu koristiti unutar testova ili lekcija. Kod ovih *plugin*-ova smešten je u /question/type direktorijumu. Formati pitanja zaduženi su za uvoz (i izvoz pitanja) u kurs iz drugih *Moodle* kurseva ili iz drugih LMS sistema. Omogućavaju uvoz (i izvoz) pitanja u različitim formatima (npr. XML, QTI...). Kod za uvoz (i izvoz) uglavnom se piše u format.php fajlu, a kod samog *plugin*-a je smešten u /question/format direktorijumu.Pravila o pristupu kvizovima definišu različite uslove koji moraju biti zadovoljeni da bi student mogao da započne rešavanje testa (npr. test se može raditi samo sa određene IP adrese, student mora uneti lozinku za test, broj pokušaja rešavanja testa je ograničen). Njihov kod smešten je u /mod/quiz/accessrule direktorijumu.Izveštaji u vezi kvizova obezbeđuju različite prikaze i analizu rezultata kvizova. Kod im je smešten u /mod/quiz/reports direktorijumu, i moraju sadržati klasu koja proširuje quiz_default_report.

3.COALA

COALA predstavlja sistem za vizuelnu reprezentaciju kriptografskih algoritama. Namenjen je prvenstveno kao sredstvo za pomoć studentima koji su novi u oblasti zaštite podataka kako bi lakše, brže i bolje savladali gradivo. Sistem se trenutno koristi na Elektrotehničkom fakultetu u Beogradu na kursu pod nazivom Zaštita podataka na kome se najpre daje teorijska osnova za razne algoritme koji se koriste za enkripciju i dekripciju podataka, a onda se praktičnim radom u laboratoriji korišćenjem COALA-e studenti upoznaju interaktivno sa radom algoritama. Sistem ima mogućnost vizuelne reprezentacije pet vrsta kriptografskih algoritama (Slika 3.1).



Slika 3.1. *COALA-odabir algoritma*

Kod klasičnih krpitografskih algoritama koji su podržani u COALA sistemu osnovna ideja vizuelizacije je da se postigne što bolja preglednost u okviru jednog ekrana na kojem je prikazan čitav algoritam. Zato su uvedena određena ograničenja (npr. za dužine poruka koje se mogu prikazati) koja omogućavaju bolju organizaciju vizuelizacije, ali ne umanjuju mogućnost demonstracije rada samih algoritama.Za sve klasične algoritme postoji mogućnost unosa originalne poruke u vidu teksta koji se sastoji od ASCII karaktera, kao i mogućnost odabira ključa (za one za koje to ima smisla). Samim tim moguće je imati veliki broj različitih primera, što omogućava studentu da vizuelizuje rad algoritama sa različitm ulaznim vrednostima.Kod simetričnih algoritama osnovna ideja kojom se vodi COALA je da se omogući praćenje celog toka izvršavanja ovih algoritama (s obzirom na veliki broj koraka u svakom od njih). Ulazni parametri su takođe promenljivi kao kod klasičnih algoritama i korisnik može unositi poruku i ključ u vidu heksadecimalnih vrednosti (kako bi se olakšalo korisnicima).Kod algoritama sa javnim ključem osnovna ideja je da se omogući praćenje algoritama sa parametrima koji bi bili pogodni za vizuelizaciju (nije moguće pratiti realne slučajeve kao u ostalim algoritmima zbog parametara koji bi tada bili preveliki). Kod svih algoritama velika pažnja se posvećuje prikazivanju detalja korišćenjem raznih upadljivih boja ili oznaka kako bi za korisnika bilo lako da prati promene stanja prilikom rada samog algoritma.

Implementacija COALA-e je urađena u *Java* jeziku što sistemu daje visok stepen portabilnosti. Za vizuelizaciju je korišćen *Swing* API koji ima veliki broj komponenti koje su veoma korisne za predstavljanje agoritama.COALA je podeljena na određene celine koje su razdvojene po paketima.Glavni paket je COALA i tu se nalaze svi ostali paketi. Dva glavna paketa su paket control i paket view koji su sadržani u paketu COALA. U paketu control se nalaze klase koje su neophodne za samo izvršavanje algoritma dok se u paketu view nalaze klase zadužene za vizuelizaciju.Sistem je vrlo modularan i dobro strukturiran tako da promene u načinu prikazivanja algoritama ne zahtevaju nikakve promene u samom kodu koji je zadužen za simuliranje ponašanja algoritama. Zato je sistem COALA moguće transformisati u neku *web* verziju prikazivanja sadržaja, a da se veliki deo koda može iskoristiti.

4. INTEGRACIJA

U prethodnim poglavljima detaljnije su objašnjeni *Moodle* i COALA. U ovom poglavlju biće objašnjen i sam postupak njihove integracije.

A. IZMENE UNUTAR COALA-E

U ovom poglavlju biće navedene izmene koje su bile potrebne unutar COALA-e kako bi integracija bila moguća. Kako je COALA *Java* aplikacija, a *Moodle web* aplikacija, bilo je potrebno prilagoditi COALA-u tako da može da se izvršava u internet pretraživačima. Prvi korak ka postizanju tog cilja jeste kreiranje apleta koji će zapravo biti omotač oko COALA-e. U

nastavku je prikazan kod tog apleta (Slika 4.1) i njegovo detaljno objašnjenje. Korišćen je *JPanel* iz *Swing* paketa, zato što je isti paket korišćen i pri implemetaciji COALA-e.

```
package cavs.applet;
import cavs.view.CAVSExpiredView;
import javax.swing.*;
import javax.swing.JApplet;
import cavs.view.CAVSIntroView;
import java.util.*;
import java.awt.Container;
import java.awt.GridBagLayout;

public class AppletWrapper extends JApplet{

    private List<JPanel> previousList = new ArrayList<JPanel> ();
    private Container content;

    public void addPanel(JPanel c){
        JPanel previous;
        try
        {
            previous = (JPanel)content.getComponent(0);
            previousList.add(previous);
        }
        catch(java.lang.IndexOutOfBoundsException e){

        }
        content.removeAll();
        content.add(c);
        this.repaint();

    }

    public void back(){
        try{
            content.removeAll();
            content.add(previousList.remove(previousList.size()-1));
            this.repaint();
        }catch(Exception e){
            this.repaint();
        }
    }

    public void init() {
        content = getContentPane();
        content.setLayout(new GridBagLayout());

        cavs.CAVSApp app = new cavs.CAVSApp();
        app.applet = this;
        Calendar cal = Calendar.getInstance();
        if (cal.get(Calendar.YEAR) <= 2015 && cal.get(Calendar.MONTH) < 9){
          new CAVSIntroView(app);
        }
        else
            new CAVSExpiredView(app);
    }
}
```

Slika 4.1. *Kod apleta koji predstavlja omotač oko COALA-e*

Ulazna tačka apleta je njegova *init* metoda. U njoj se vrši inicijalizacija apleta. Prve dve linije *init* metode dohvataju glavni okvir apleta i postavljaju njegov izgled tako da sadržaj bude centriran. Kasnije objekat *content* biće korišćen za izmenu sadržaja apleta.U naredne tri linije instancira se COALA aplikacija i inicijalizuje se referenca na aplet u *cavs.CAVSApp* klasi kako bi se izgled apleta menjao kada i izgled aplikacije. Lista *previousList* služi za čuvanje panela koji su prethodno bili prikazani, kako bi se korisnik mogao kretati kroz aplikaciju.Metoda *addPanel* služi za prikazivanje naredne strane COALA aplikacije. U *previousList* čuva se trenutno prikazani panel, i nakon toga prikazuje se novi panel.Metoda *back* služi za vraćanje na prethodnu stranu aplikacije. Uklanja se trenutni sadržaj i prikazuje se panel koji je pre ovoga bio vidljiv. Ova metoda poziva se iz *back* metoda klasa unutar COALA aplikacije. Nakon svake izmene sadržaja poziva se metoda *repaint* kako bi se prikazao ažurirani sadržaj.Kako bi se u apletu prikazivao sadržaj COALA aplikacije bilo je potrebno izvršiti i određene izmene unutar same aplikacije.

Za početak bilo je potrebno čuvati referencu na objekat apleta u klasi *CAVSApp* (Slika 4.2).

```
public class CAVSApp extends SingleFrameApplication {
  public AppletWrapper applet;
```

Slika 4.2. *Prikaz čuvanja reference naAppletWrapper klasu*

U konstruktoru svake klase koja služi za grafički prikaz nakon inicijalizacije komponenti dodat je poziv metode *addPanel* iz apleta, kako bi se njegov sadržaj promenio. Metodi *addPanel* kao argument prosleđivan je objekat *mainPanel* koji se ranije prikazivao u objektu tipa *Frame*. Izbačene su reference na objekat *Frame* (koje su služile za navigaciju kroz aplikaciju) jer je umesto *Frame* na aplet dodavan panel (*Frame* nije bilo moguće dodati, a da se pri tome zadrži odgovarajući izgled), tako da je sada navigacija implementirana na nivou panela. Na primeru klase *CAVSCesarMainView* mogu se videti navedene izmene, izmene su analogne i u ostalim klasama (Slika 4.3).

```
public CAVSCaesarMainView (CAVSApp app, FrameView fv) {
    super(app);

    this.getFrame().
         setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
    this.getFrame().setResizable(false);
    this.getFrame().pack();
    this.app = app;
    this.previous = fv;

    initComponents();
    createViewArrays();
    setArraysOpaque(true);
    clearPlaintextView();
    clearEncryptionTableView();
    clearCiphertextView();
    clearDecryptionTableView();
    clearPlaintextFView();
    createViewListeners();
    jButton3.setEnabled(false);
    jButton4.setEnabled(false);
}
```

```
public CAVSCaesarMainView(CAVSApp app, FrameView fv) {
    super(app);
    this.app = app;

    initComponents();
    app.applet.addPanel(mainPanel);

    createViewArrays();
    setArraysOpaque(true);
    clearPlaintextView();
    clearEncryptionTableView();
    clearCiphertextView();
    clearDecryptionTableView();
    clearPlaintextFView();
    createViewListeners();
    jButton3.setEnabled(false);
    jButton4.setEnabled(false);
}
```

Slika 4.3. *Konstruktor klase CAVSCesarMainView pre izmena (levo) i nakon izmena (desno)*

Bilo je potrebno izmeniti i *back* metodu svake klase koja je korišćena za grafički prikaz, njen kod pre i nakon izmena prikazan je na narednoj slici (Slika 4.4).

```
public void back() {
    this.getFrame().setVisible(false);
    previous.getFrame().setVisible(true);
}
```

```
public void back() {
    app.applet.back();
}
```

Slika 4.4. *Kod back metode pre izmena (levo) i nakon izmena (desno)*

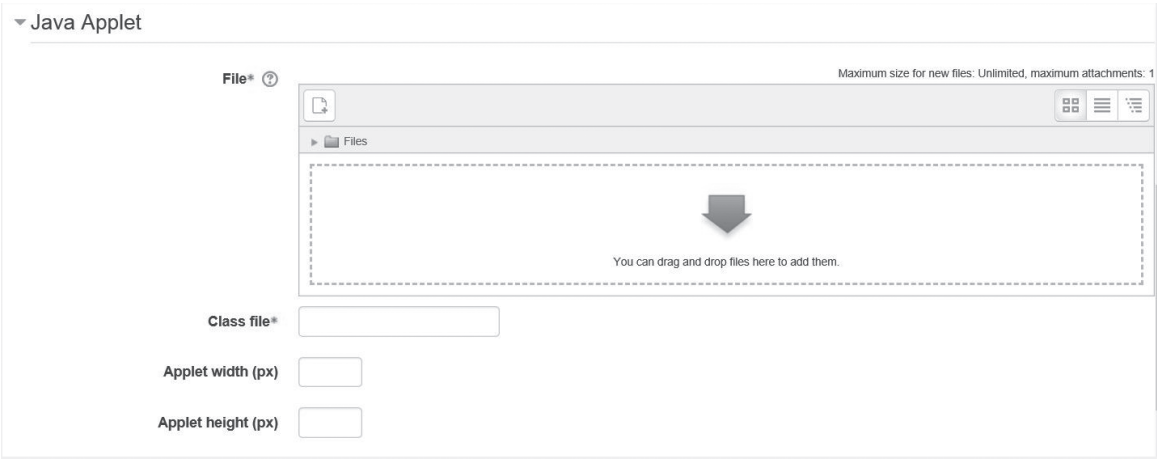
F. IZMENE UNUTAR MOODLE-A

Za proveru znanja studenata na laboratorijskim vežbama iz predmeta Zaštita podataka koristi se *Moodle* sistem. Studenti dobijaju set pitanja na koja treba da odgovore. Ova pitanja prikazuju se kroz *Moodle* sistem. Set pitanja prikazuje se u okviru *Moodle* testa, a nakon završenog pokušaja rešavanja testa odgovori se automatski ocenjuju i studentu se prikazuje postignuti rezultat. U okviru ovog testa koriste se sledeći tipovi pitanja:pitanja sa odgovorom tačno ili netačno, pitanja sa uparivanjem odgovora, pitanja sa više ponuđenih odgovora, pitanja gde student treba da upiše kratak odgovor. Kako bi se studentima olakšalo odgovaranje na ova pitanja došlo se na ideju da se pored samih pitanja na *Moodle* stranici prikazuje i COALA simulator. Najbolji način da se to obezbedi jeste pisanje *Moodle plugin*-a.U razmatranje su uzeta sledeća dva tipa *plugin*-a:tip pitanja i moduo aktivnosti.

Problem sa kreiranjem novog tipa pitanja jeste to što bi zapravo trebalo napraviti 4 nova tipa pitanja (prikazati COALA-u u svakom od korišćenih tipova pitanja). Takvo rešenje nije skalabilno. Ukoliko bi se odlučilo da se pored navedenih tipova pitanja koristi još neki tip bilo bi potrebno pisati i dodatni *plugin*. Pored toga problem se javlja i pri navigaciji između pitanja, u svim pitanjima trebalo bi prikazati istu instancu COALA apleta, ili bar različite instance sa istim stanjem. Kako bi se taj problem rešio trebalo bi obezbediti neki vid komunikacije između pitanja. Na taj način gubi se nezavisnost i modularnost (što je osnovna odlika *Moodle* sistema), a opet s druge strane ni samo obezbeđivanje komunikacije ne bi bio trivijalan posao.Razmatranjem druge mogućnosti došlo se do zaključka da problem skalabilnosti ne bi bio prisutan. Što se problema pri navigaciji između pitanja tiče ispostavilo se da je prisutan, ali u značajno manjoj meri.Zbog svega prethodno navedenog odabrano je drugo rešenje.Moduli aktivnosti obezbeđuju različite vrste aktivnosti. Za prikaz pitanja na predmetu Zaštita podataka već se koristi *plugin* kviz. Ovaj *plugin* omogućuje nastavnicima da kreiraju testove koji se sastoje od različitih tipova pitanja. Pored toga obezbeđuje i korišćenje specifičnih izveštaja, kao i određena prava pristupa. Kako je sve ovo potrebno i u novom *plugin*-u koji će imati podršku i za COALA-u odlučeno je da se kod kviz *plugin*-a uzme kao osnova za novi *plugin*.Kao ime za novi *plugin* odabrano je jquiz. Ovo ime simbolizuje spoj *Jave* i kviza (j-*Java*, *quiz*-kviz) što bi ovaj *plugin* i trebalo da omogući.

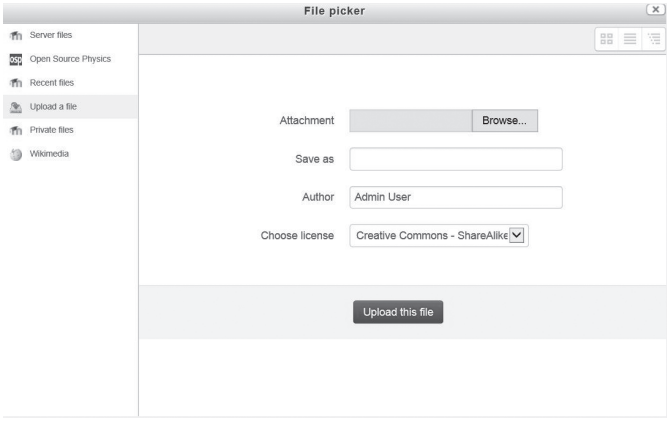
Prvi korak u implementaciji bio je izmena forme za kreiranje kviza, tako da se omogući postavljanje arhive sa apletom kako bi aplet kasnije mogao biti korišćen. Ovo je učinjeno izmenom klase *mod_jquiz_mod_form* iz fajla mod_form.php. Promenjena je metoda *definition* tako što su na formu dodati potrebni elementi. Na slici 4.5 prikazan je izgled izmenjenog dela forme.

Polja *File* i *Class file* su obavezna (što je i naznačeno crvenom zvezdicom), dok su *Applet width(px)* i *Applet height(px)* opcioni.Polje *File* zapravo predstavlja *Moodle* komponentu sakupljač fajlova (eng. *file picker*). Komponenta sakupljač fajlova obezbeđuje selekciju i prikaz fajlova unutar *Moodle*. Najčešće ovi fajlovi se kopiraju u *Moodle* bez obzira na to gde se nalazili (na korisničkom računaru, u repozitorijuma fajlova, itd.). Sama selekcija fajla može se izvesti na dva načina: prevlačenjem fajla u selektovanu oblast ili klikom na odgovarajuću ikonicu. Klikom na ikonicu otvara se prozor prikazan na slici4.6. Na levoj strani prozora nalaze se veze ka repozitorijumima sa kojih se može preuzeti fajl. Jedna od tih veza je i *Upload a file* koja služi za preuzimanje fajla sa korisničkog računara. Klikom na dugme *Browse* otvara se prozor u kome korisnik može odabrati fajl sa svog računara. Klikom na dugme *Upload this file* selektovani fajl se kopira u *Moodle*. Pri kreiranju jquiz-a u polje *File* treba dodati .zip arhivu koja sadrži sve .jar fajlove koji su potrebni za izvršavanje apleta (.jar fajl apleta, ali i sve dodatne .jar fajlove koji su korišćeni kao spoljašnje biblioteke).U polje *Class file* potrebno je navesti putanju do



Slika 4.5. *Odabir arhive sa apletom*

aplet klase (putanju kroz pakete). Recimo ukoliko je naša aplet klasa *AppletWrapper* i smeštena je u cavs.applet paketu, odgovarajuća putanja bi bila *cavs.applet.AppletWrapper*. Ovo polje ne bi bilo potrebno ukoliko bi se .jre fajl koji sadrži aplet i .jre fajlovi od kojih aplet zavisi odvojeno postavljali na server. U tom slučaju putanja bi se mogla odrediti programskim putem.



Slika 4.6. *File picker*

Polja *Applet width (px*) i *Applet height (px)* služe za odabir širine i visine prozora u kome će se aplet prikazivati. Ukoliko se ne unese neka vrednost, koristiće se podrazumevana širina i visina (500px). Kako bi podaci vezani za aplet mogli biti sačuvani bilo je potrebno i dodati potrebna polja u bazi podataka. To je učinjeno definicijom polja u fajlu install.xml. Primer definicije jednog polja u tebeli baze podataka prikazan je na slici 4.7.

Nakon klika na neko od dugmadi za predaju forme poziva se i metoda *jquiz_add_instance* iz lib.php fajla. Ova metoda dodaje u bazu novi objekat jquiz-a koji je inicijalizovan podacima iz forme. Na kraju ove metode poziva se metoda *update_jquiz_and_files_tables* iz fajla loacallib.php. Ova metoda inicijalizuje polja koja su dodata u bazu za potrebe apleta. Prvo se inicijalizuju polja u *files* tabeli i kreira .zip fajl u *dataroot* direktorijumu. Nakon toga u direktorijumu *jarfiles* kreiraju se potrebni direktorijumi. Struktura direktorijuma je takva da se prvo kreira direktorijum koji odgovara kursu (naziv je id kursa), a unutar njega direktorijum koji odgovara jquiz-u (naziv je id jquiz-a). Navedeni direktorijumi se kreiraju ukoliko već ne postoje. Zatim se direktorijumu koji odgovara jquiz-u kopira predata arhiva i u navedenom direktorijumu se ekstrahuje arhiva. Na krajuse u *files* tabeli dodaje ulaz za .zip fajl iz *jarfiles* direktorijuma. Ovim je završeno kreiranje potrebne direktorijumske strukture i ažuriranje *files* tabele.Na kraju ažurira se i tabela jquiz, postavljanjem odgovarajućih vrednosti za polja *applet_name, custom_width, custom_height, codebase*.

Slika 4.8. *Struktura jarfiles direktorijuma*

Sada kada je arhiva sa apletom postavljena na *Moodle* preostaje još prikazivanje apleta u toku pokušaja rešavanja testa. Da bi se to postiglo potrebno je promeniti fajlove attempt.php i renderer.php.U fajlu attempt.php izvršavaju se sve potrebne validacije i inicijalizuje se *JavaScript* funkcija. Na kraju fajla dohvata se sadržaj stranice pozivom metode *attempt_page* klase *mod_jquiz_renderer* iz fajla renderer.php i taj sadržaj prikazuje se korisniku.Kako je nama potreban drugačiji sadržaj stranice potrebno je promeniti metodu *attempt_page*. Ali pre toga potrebno je proveriti da li se u *jarfiles* folderu nalazi arhiva sa apletom i da li je sadržaj te arhive ažuran. Ovo se radi u metodi *prepare_file* koja je dodata u locallib.php fajl. Kao argument ovoj metodi prosleđuje se objekat jquiz-a. Iz tabele *files* čita se odgovarajući fajl (pomoću id-a jquiz-a). Nakon toga proverava se da li u direktorijumu *jarfiles* postoji potrebni fajl, putanja do traženog fajla određuje se korišćenjem vrednosti iz objekta jquiz. Ukoliko fajl postoji proverava se da li je njegov sadržaj validan, ukoliko nije, kopira se validni sadržaj. Ukoliko fajl ne postoji kreira se.Metodi *attempt_page* dodat je jedan argument koji zapravo predstavlja kod koji treba ugraditi. Jedina izmena u samom kodu te metode jeste dodavanje prosleđenog koda u string koji metoda vraća. Ovaj kod dodat je neposredno pre formiranja prikaza za pitanja kako bi aplet bio prikazan odmah iznad pitanja.Pre poziva metode *attempt_page* potrebno je formirati ugrađeni kod. Ovo se radi u metodi *generate_embedding_code*. Kod kojim se formira ugrađeni kod prikazan je na sledećoj slici (Slika 4.9).

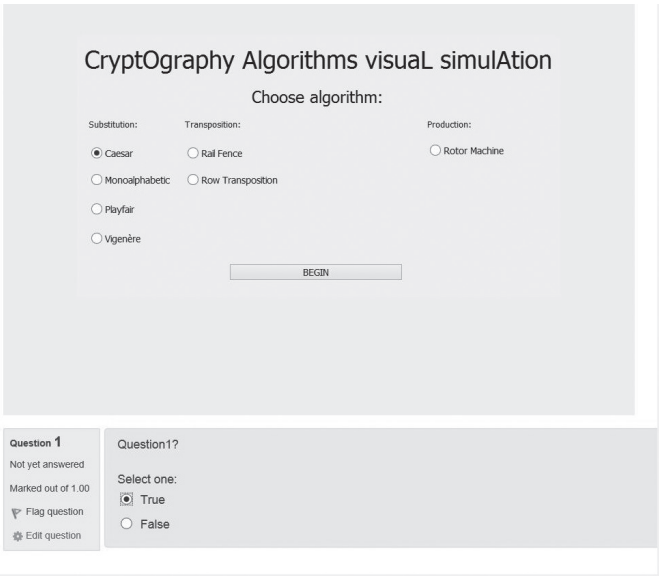
U objektu $archive upisani su nazivi svih .jar fajlova iz odgovarajućeg direktorijuma.Nakon načinjenih izmena stranica za pokušaj rešavanja izgleda kao na slici 4.10:

```
<FIELD NAME="class_file" TYPE="text" LENGTH="medium" NOTNULL="false" SEQUENCE="false"
COMMENT="applet main class file"/>
```

Slika 4.7. *Primer definicije polja u tabeli baze podataka*

```
$code = "<applet code='".$jquiz->class_file."' codebase='".$CFG->wwwroot.$jquiz->codebase."' archive='".$archive."'
width='".$jquiz->custom_width."' height='".$jquiz->custom_height."'></applet>";
```

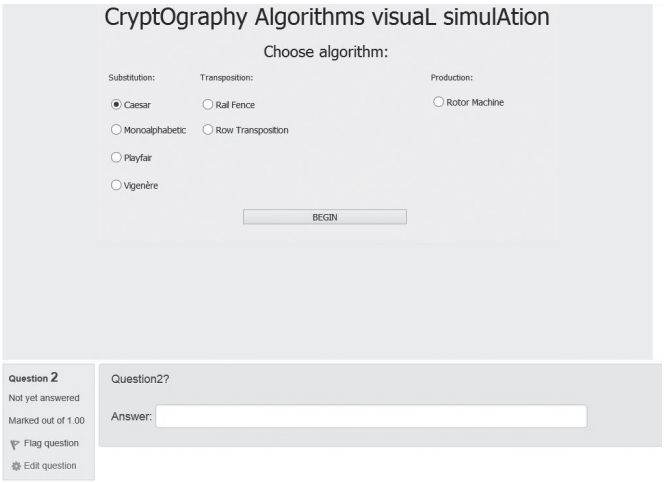
Slika 4.9. *Kod kojim se formira ugrađeni kod*



Slika 4.10. *Prikaz Moodle stranice u toku odgovaranja na prvo pitanje*

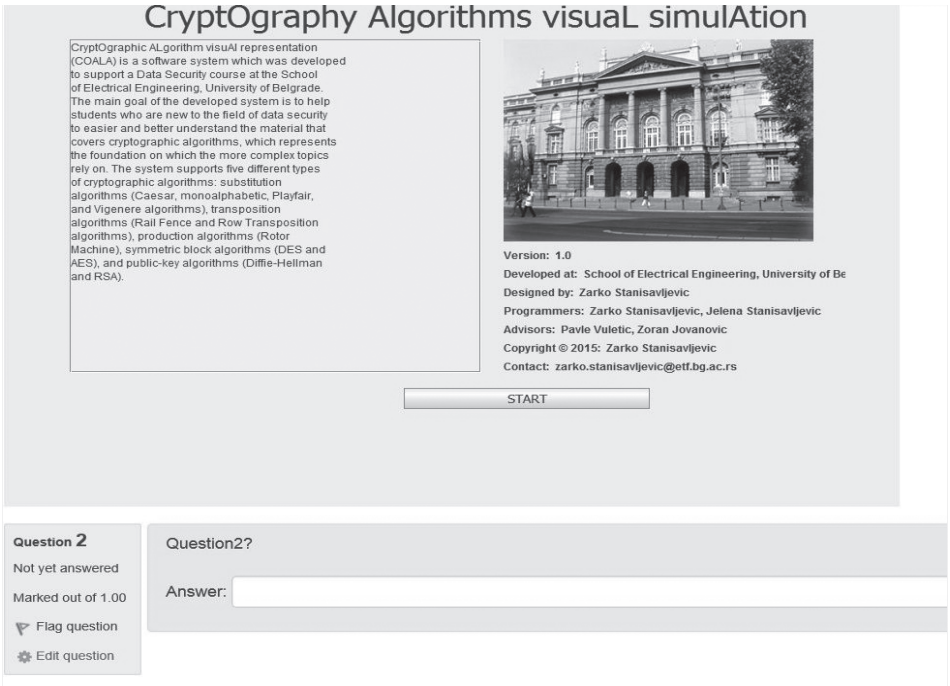
Nakon toga kada korisnik pređe da odgovara na sledeće pitanje pojavljuje se nova stranica sa novim pitanjem i sa apletom koji je ponovo učitan (Slika4.11). Stanje apleta nije sačuvano pri promeni stranice. Ovaj problem već je pomenut pri razmatranju mogućih tipova *plugin*-a koji bi bili adekvatno rešenje. Jedno moguće rešenje jeste čuvanje stanja apleta, pri kliku na dugme *Next*, i njegovo restauriranje, pri učitavanju naredne stranice. Ovo se može postići čuvanjem trenutnog stanja apleta u njegovoj *destroy* metodi i restauriranjem tog stanja u *init* metodi. Stanje apleta može se čuvati na dva načina: eksternalizacijom ili serijalizacijom. U slučaju eksternalizacije aplet čuva samo informacije koje su neophodne da bi se vratio u prethodno stanje. U slučaju serijalizacije aplet čuva sve informacije, svaku promenljivu i njenu trenutnu vrednost, svaku klasu itd. Preporučeno je koristiti eksternalizaciju. Ovo rešenje zahtevalo bi veće promene unutar apleta i *Java* aplikacije, dok bi kod na serverskoj strani ostao neizmenjen ili bi pretrpeo manje izmene.

Drugo moguće rešenje jeste korišćenje *Ajax*-a. Pri prelasku na narednu stranu pozvala bi se *Ajax* funkcija koja bi ažurirala oblast sa pitanjima i navigacioni blok za test, ostatak stranice ostao bi nepromenjen (Slika 4.12). Ovo rešenje ne bi zahtevalo nikakve izmene unutar koda apleta i *Java* aplikacije, jedine izmene bile bi u kodu novog *plugin*-a.Odabrano je drugo rešenje i u nastavku će biti izložena njegova implementacija.



Slika 4.12. *Prikaz Moodle stranice koja koristi Ajax u toku odgovaranja na prvo pitanje*

Bez *Ajax*-a, klikom na dugme next predaje se forma i izvršava se fajl process_attempt.php, unutar njega obrađuju se odgovori sa trenutne strane i veza se preusmerava na attempt.php, gde se prikazuje naredna strana. Predaju ove forme potrebno je preusmeriti tako da sve ono što se radi u fajlu process_attepmt.php bude odrađeno, ali da se veza ne preusmeri, već da se ostane na istoj stranici, a da se pritom promeni sadržaj forme za pitanja.U funkciji



Slika 4.11. *Prikaz Moodle stranice nakon prelaska na drugu stranicu za odgovaranje*

M.mod_jquiz.init_attempt_form u fajlu module.js inicijalizuje se forma za pitanja tako da je potrebno promeniti kod ove funkcije. Potrebno je postaviti novu funkciju koja će se pozivati pri predaji forme. Unutar ove funkcije prvo se, korišćenjem Jquery-ja, dohvataju podaci iz forme. Zatim se ti podaci prosleđuju process_attempt.php fajlu korišćenjem *Ajax* funkcije, ukoliko je poziv funkcije bio uspešan sadržaj forme za pitanja menja se sadržajem koji je vratila *Ajax* funkcija. U fajlu process_attempt.php potrebno je dodati logovanje strane, kao i ažuriranje rednog broja strane u bazi podataka (ovo se ranije radilo u fajlu attepmt.php). Umesto preusmeravanja stranice na attepmt.php *Ajax* funkciji se vraća forma za pitanja (echo $attempt_form).Pri uspešnom povratku iz *Ajax* funkcije potrebno je i reinicijalizovati sve potrebne *JavaScript* funkcije. Da bi se adekvatno inicijalizovalo ažuriranje oznaka na pitanjima bilo je potrebno napisati dodatnu funkciju za inicijalizaciju. Jedina razlika u odnosu na već postojeću funkciju je u tome što se u ovoj funkciji ne dodaje osluškivač za klik miša. Osluškivač je potrebno dodati samo jednom kako se na jedan klik ne bi više puta menjalo stanje obeleživača. S druge strane sva polja se moraju ponovo inicijalizovati, jer je sadržaj stranice promenjen tako da stara polja više ne postoje. Kako je *JavaScript* kod koji je zadužen za rad sa obeleživačima pitanja smešten u flags.js fajlu koji se nalazi u question modulu, kako bi se zadržala modularnost bilo je potrebno prebaciti taj kod u module.js (/jquiz/module.js).Takođe pri uspešnom povratku iz *Ajax* funkcije bilo je potrebno ažurirati i navigacioni blok za pitanja, ovo je odrađeno pozivom još jedne *Ajax* funkcije. Pri uspešnom povratku iz te funkcije sadržaj koji je ona vratila upisuje se u stranicu na mesto navigacionog bloka.U sledećoj tabeli prikazano je koji su fajlovi promenjeni kao i najznačajnije izmene u njima.

Fajl	Izmena
attempt.php	promena izgleda stranice
processattempt.php	generisanje novog sadržaja forme za pitanja
module.js	izmena *JavaScript* funkcija, dodavanje ajaxa
renderer.php	izmena koda za generisanje izgleda stranice
mod_form.php	dodavanje forme za preuzimanje apleta
db/install.xml	dodatna polja u bazi podataka
lang/en/jquiz.php	dodatni stringovi za prikaz
locallib.php	uslužne funkcije
version.php	informacije o verziji
generate_embedding_code	generisanje koda za ugrađivanje apleta
getnavblock	ažuriranje navigacionog bloka
attempt.php	promena izgleda stranice
processattempt.php	generisanje novog sadržaja forme za pitanja
module.js	izmena *JavaScript* funkcija, dodavanje ajaxa
attempt.php	promena izgleda stranice

Tabela 4.1. *Prikaz najznačajnijih izmena u MOODLE fajlovima*

5. ZAKLJUČAK

U radu su prvo predstavljeni sistemi COALA i *Moodle* koji predstavljaju dve različite platforme namenjene prvenstveno za edukaciju studenata. Ovaj rad kao glavni cilj ima da ujedini te dve platforme kako bi studentima omogućio da lakše savladaju gradivo iz predmeta Zaštita podataka na Elektrotehničkom fakultetu u Beogradu. Integracijom COALA i *Moodle* sistema student u jednom prozoru pretraživača ima na raspolaganju sve što mu je potrebno za izradu laboratorijske vežbe. Samim tim znatno se olakšava praćenje rada samih algoritama sa simultanim odgovaranjem na pitanja koja proveravaju nivo znanja studenta.

Ideja korišćenja *Ajax*-a u *Moodle*-u može se iskoristiti i u nekim budućim projektima u kojima osvežavanje stranice prilikom promene pitanja nije prihvatljivo rešenje. Aplet koji predstavlja rad COALA sistema može se zameniti nekim drugim i sa dodavanjem novih pitanja može se dobiti potpuno funkcionalan sistem koji bi služio za demonstraciju i proveru znanja na raznim drugim kursevima. Sa infrastrukturom koja je razvijena u sklopu ovog projekta integracija drugih aplikacija koje se koriste na Elektrotehničkom fakultetu u Beogradu na drugim kursevima bi zahtevala relativno mali napor.

Pri izradi ovog rada izvršena je integracija samo na jednom nivou, u budućnosti ovaj rad bi se mogao unaprediti tako da se integracija izvrši i na drugom nivou, tako da tačne odgovore na pitanja *Moodle* sistem dobija direktno iz COALA sistema, a ne da ih nastavnici obezbeđuju pri kreiranju pitanja.

6. LITERATURA

[1] Stanford Online Courses, dostupno na: http://online.stanford.edu/courses, poslednji pristup: septembar 2015.
[2] Cambridge Online Courses, dostupno na: http://www.ice.cam.ac.uk/courses/online-courses, poslednji pristup: septembar 2015.
[3] Harvard Online Courses, dostupno na: http://online-learning.harvard.edu/, poslednji pristup: septembar 2015.
[4] MIT Online Courses, dostupno na: http://ocw.mit.edu/index.htm, poslednji pristup: septembar 2015.
[5] Coursera Online Courses, dostupno na: https://www.coursera.org/, poslednji pristup: septembar 2015.
[6] EDX Online Courses, dostupno na: https://www.edx.org/, poslednji pristup: septembar 2015.
[7] Iversity Online Courses, dostupno na: https://iversity.org/, poslednji pristup: septembar 2015.
[8] Moodle sistem za upravljanje obrazovnim procesom, dostupno na:https://moodle.org/, poslednji pristup: septembar 2015.
[9] Stanisavljević Ž., Drašković D., Mišić M., "Primena Moodle platforme u nastavi računarske tehnike i informatike," TELFOR 2014, Beograd, pp. 1039-1042, Novembar 2014.
[10] Stanisavljevic Z., Stanisavljevic J., Vuletic P., Jovanovic Z., "COALA - System for Visual Representation of Cryptography Algorithms," IEEE Transactions on Learning Technologies, Vol. 7, No. 2, pp. 178-190, Jun 2014
[11] Stanisavljević Ž., Stanisavljević J., "Softverski sistem za vizuelnu reprezentaciju klasičnih kriptografskih algoritama," Info M, Vol. 48, pp. 21-28, Dec 2013.
[12] Stanisavljevic Z., Stanisavljevic, J., "Softverski sistem za vizuelnu reprezentaciju Advanced Encryption Standard algoritma," TELFOR 2011, Beograd, pp. 1364-1367, Novembar 2011.
[13] Gerner, J., Owens, M., Naramore, E., Warden, M., Professional LAMP: Linux, Apache, MySQL and Development, Indianapolis, 2006.
[14] Moodle struktura, dostupno na:https://docs.moodle.org/28/en/Moodle_site_-_basic_structure, poslednji pristup: septembar 2015.
[15] Moodle uloge, dostupno na:https://docs.moodle.org/29/en/Standard_roles, poslednji pristup: septembar 2015.
[16] Moodle tipovi plugin-ova, dostupno na:https://docs.moodle.org/dev/Plugin_types, poslednji pristup: septembar 2015.

dipl. inž. Ljiljana Gavović, student master studija, Elektrotehnički fakultet Univerziteta u Beogradu.
Kontakt: ljgavovic@gmail.com
Oblasti interesovanja: baze podataka, zaštita podataka, razvoj mobilnih aplikacija, arhitektura i organizacija računara

dr Žarko Stanisavljević, docent, Elektrotehnički fakultet Univerziteta u Beogradu.
Kontakt: zarko.stanisavljevic@etf.bg.ac.rs
Oblasti interesovanja: zaštita podataka i računarskih sistema, razvoj softverskih alata za elektronsko učenje, programiranje internet aplikacija, arhitektura i organizacija računara