ВИСОКА ШКОЛА ЕЛЕКТРОТЕХНИКЕ И РАЧУНАРСТВАСТРУКОВНИХ СТУДИЈА

**Цветковић Урош**

**Java клијент-сервер апликација за акваристичку продавницу**

**- завршни рад -**



Београд, септембар 2019.

Кандидат: **Цветковић Урош**

Број индекса: ЕЛИТЕ **80/16**

Студијски програм: **Електроника и телекомуникације**

Тема: Java клијент-сервер апликација за акваристичку продавницу

Основни задаци:

**1. Опис Java клијент-сервер архитектуре примењене у раду.**

**2. Имплементација Java клијент-сервер апликације за акваристичку продавницу.**

**3. Опис корисничког интерфејса и уграђене статистике.**

Ментор:

Београд, септембар 2019 годинe.

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

др Перица Штрбац, проф. ВИШЕР

Резиме:

Упрвом поглављу биће описано каква је архитектура клијентске и серверске стране. Како функционише трговина апликацијом, као и остале њене могућности. У другом делу следе детаљна објашњења како је апликација имплементирана и како раде њене функције и обрада статистике. У последњем делу приказаће се како изгледа и како се користи кориснички интерфејс апликације.

**Кључне речи:** архитектура, функције, статистика, интерфејс

ABSTRACT:

The first chapter will describe the architecture of the client and server side. How the app store works, as well as its other features. The second section provides detailed explanations of how the application is implemented and how its functions and statistics work. The last section will show how the application UI looks and how it is used.

**Key words**: architecture, functions, statistics, interface

САДРЖАЈ:

[1. УВОД 1](#_Toc19121806)

[2. Опис јава клијент-сервер архитектуре примењене у раду 2](#_Toc19121807)

[2.1. Архитектура клијентске стране програма 2](#_Toc19121808)

[2.1.1. Избор и куповина робе 2](#_Toc19121809)

[2.1.2. Остале могућности 3](#_Toc19121810)

[2.2. Архитектура серверске стране програма 4](#_Toc19121811)

[2.2.1. Додавање рибица и слика у галерију 4](#_Toc19121812)

[2.2.2. Листа купаца 5](#_Toc19121813)

[2.2.3. Статистика 5](#_Toc19121814)

[3. Имплементација функционалности јава клијент-сервер апликције 6](#_Toc19121815)

[3.1. имплементација Функција клијентске и серверске стране програма 6](#_Toc19121816)

[3.1.1. Кретање кроз клијентски програм 6](#_Toc19121817)

[3.1.2. Имплементација пуњења и брисања листе рибица у корпи 7](#_Toc19121818)

[3.1.3. Имплементација додавања нових риба 8](#_Toc19121819)

[3.1.4. Имплементација додавања нових слика у галерију 9](#_Toc19121820)

[3.2. Мрежно програмирање и Статистика у програму 10](#_Toc19121821)

[3.2.1. Класа SlikeIO 10](#_Toc19121822)

[3.2.2. Слање листе риба и слика за галерију 11](#_Toc19121823)

[3.2.3. Обрада наруџбина 12](#_Toc19121824)

[3.2.4. Дефинисање статистичких података и класа Статистика 13](#_Toc19121825)

[4. Опис корисничког интерфејса 15](#_Toc19121826)

[4.1. Кориснички интерфејс на клијентској страни програма 15](#_Toc19121827)

[4.1.1. Панел рибице 16](#_Toc19121828)

[4.1.2. Панел корпа 17](#_Toc19121829)

[4.1.3. Панел галерија, статистика и О нама 18](#_Toc19121830)

[4.2. Кориснички интерфејс на серверској страни програма 20](#_Toc19121831)

[4.2.1. Панел за додавање нове рибице и панел за убацивање слике 20](#_Toc19121832)

[4.2.2. Панел купци 21](#_Toc19121833)

[4.2.3. Панел статистика 21](#_Toc19121834)

[5. ЗАКЉУЧАК 22](#_Toc19121835)

[6. ИНДЕКС ПОЈМОВА 23](#_Toc19121836)

[7. ЛИТЕРАТУРА 24](#_Toc19121837)

[8. Прилози 25](#_Toc19121838)

[9. изјава о академскoј честитости 26](#_Toc19121839)

# УВОД

Објектно оријентисано програмирање је један од водећих приступа при изради софтвера. Програмски језик Јава је објектно-оријентисани језик и спада међу најполупарнијим. На предмету Објектно програмирање 2 се уче принципи оваквог приступа програмирања коришћењем управо програмског језика Јава и то су једни од разлога одабира теме из овог предмета.

Изабрана тема обухвата опис апликације за продају акваријумских рибица и њен развој. Идеја је таква да потенцијални купци користећи апликацију могу лако да сазнају све основне податке о жељеним рибицама и уколико се за неку одлуче да је лако могу наручити на кућну адресу, док продавац, поред наруџбина, може видети статистичке податке о продаји како би организованије водио посао. Начин и принцип израде се не мора ограничити само на тренутни предмет продаје, него се може користити и при конструисању било које апликације за трговину.

Рад обухвата опис архитектуре и могућности софтвера, као и детаљно објашњење његове имплементације. Преглед и опис графичког интерфејса такође спада у садржај рада.

# Опис јава клијент-сервер архитектуре примењене у раду

У овом делу биће описана архитектура апликације. Биће посебно обрађена архитектура клијентске и серверске стране програма, као и њихове опције и могућности.

## Архитектура клијентске стране програма

Клијентска страна програма је намењена за купца. У овом делу програма врши се преглед листе рибица и њихова куповина. Како би купац стекао увид у предмет трговине постоји могућност приступа галерији слика које поставља продавац, као и статистичке податке које описују робу и њен квалитет.

### Избор и куповина робе

Када купац покрене програм, приказује му се мени са опцијама и листа свих акваријумских рибица које су тренутно доступне (Слика 2.1). Свака ставка са листе је описана сликом и именом. Када купац селектује жељену ставку појављују му се специфични подаци за изабрану рибицу. Подаци описују какво је научно име рибе, какав јој је темперамент, чиме се храни, која је концентрација јона водоника у води и њена температура, колика је величина рибице и колики би акваријум био потребан за њу, као и колика је цена. Ако се купцу допадне врста рибице, може изабрати колико их жели и сместити жељени број у корпу. Након тога појављује му се потврда да је успешно сместио артикал у корпу и може наставити са куповином.



Слика 2.1 – Почетни прозор клијента продавнице.

Након изабраних врста рибица, корисник бира опцију која представља корпу. Тада му се појављује претходно изабране рибице и потенцијални материјал за куповину, као и укупна цена (Слика 2.2). У сваком тренутку купац може избацити ставку из корпе. Такође се могу видети подаци које купац попуњава да би извршио наруџбину.   
У податке спадају име купца, адреса, број телефона и оцена продавнице. Купац потвђује захтев за куповину и уколико су сви подаци попуњени појавњује се потврда о успешној наруџбини.



Слика 2.2 – Корпа продавнице.

### Остале могућности

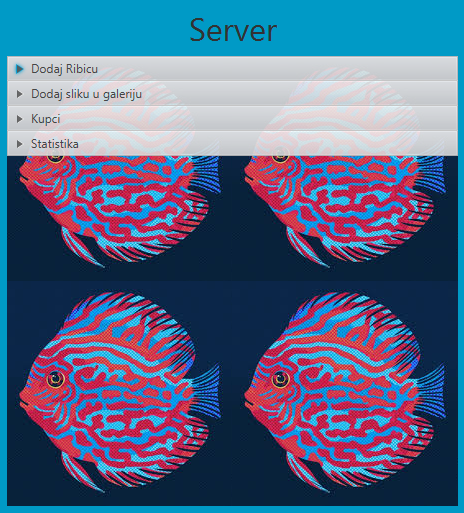
Један од избора главног менија је преглед статистичких података продавнице. Избором те опције појављује се оцена продавнице. Она показује колико су купци задовољни услугом. Такође се могу видети које рибице су најјефтиније, а које су најпродаваније.

Постоји и могућност уласка у галерију слика продавнице преко главног менија. Слике поставља продавац и оне требају да дочарају предмет трговине продавнице.

Такође, постоји и опција која омогућава приступ радном времену продавнице, нјеној адреси, као и контактни број телефона и електронску пошту.

## Архитектура серверске стране програма

Серверска страна програма је намењена за продавца. Продавац преко ње може поставити нове рибице, као и слике у галерију. Може се видети листа купаца са њиховим подацима, као и детаљнију статистику продавнице (Слика 2.3).



Слика 2.3 – Почетни мени сервера продавнице.

### Додавање рибица и слика у галерију

Када продавац изабере опцију за додавање нових акваријумских рибица појављују му се подаци које је потребно попунити. Подаци за попуњавање се тичу описа рибе (име, температура, темперамент, исхрана, величина рибе и акваријума), убацивање слике која приказује њен изглед, као и постављање жељене цене. Продавац мора попунити све податке како би успешно поставио нову рибу. Након потврђивања поставке излази обавештење да је радња успешно одрађена.

Убацивање слике у галерију се врши простим превлачењем слике у одређено поље.

### Листа купаца

Како стижу поруџбине, тако се на серверу прави листа купаца. Продавац одатле може видети ко су му купци. Селекцијом из листе појављују се подаци који описују шта је селектовани купац пазарио, на које име и адресу је потребно послати робу и контакт телефон. Након што продавац пошаље робу, могуће је обрисати купца са листе.

### Статистика

Статистички подаци продавнице дају увид у њен рад. Продавац преко серверске стране програма може детаљно да прати колико је стање зараде и колико су купци задовољни његовом услугом преко оцене. Могуће је пратити који продајни материјали су најјефтинији, а који су најпопуланији, сазнати тачан број најпродаванијих рибица. Продавац може сазнати број купаца који су посетили продавницу, као и колико њих тренутно користи апликацију.

# Имплементација функционалности јава клијент-сервер апликције

У овом поглављу биће описана како је имплементирана функционалност апликације. Детаљно ће бити објашњен сваки део кода који се односи на одређену опцију у апликацији. Опис имплентације ће започети функицијама које су на клијентском и серверском делу. Након тога ће бити описано како су мрежно и међусобно повезани.

## имплементација Функција клијентске и серверске стране програма

Следе објашњења делова кода који су одговорни за функције у клијентском и серверском делу апликације. Описаће се кретање кроз програм, како ради пуњење корпе робом и њено наручивање. Са серверске стране ће бити обрађене опције за додавање нових риба и слика у галерију. Биће описане класе које карактеришу и омогућавају функционалност програма. Алгоритам за приказивање рибица, као и слика из галерије биће описани у делу где се описује мрежна повезаност, као и приказ листе купаца и статистике.

### Кретање кроз клијентски програм

Кретање кроз опције програма се обавља једноставним притискањем дугмића. Постоје пет главних приказа, а то су листа рибица, галерија слика, статистика, информације о продавници и приказ корпе. Циркулација међу њима је извршена тако што за сваки приказ постоји посебно направљен панел, али само је један у датом тренутку видљив. Наведени принцип описује следећи код:

public void buttonActionRibice(){  
 panelONama.setVisible(false);  
 panelStatistika.setVisible(false);  
 panelGalerija.setVisible(false);  
 panelKorpa.setVisible(false);  
 panelRibice.setVisible(!panelRibice.isVisible());  
}

Ово је метода која служи при приказу панела који садржи листу рибица. Када се кликне на дугме које је задужено за овај панел, онда се гаси видљивост свих осталих панела, а пали жељени. Уколико је већ упаљен, онда ће се и он угасити. Приказ осталих панела функционише на исти начин.

Док је упаљен панел са листом робе, кликом на одрећену рибу, могуће је упалити панел који приказује опис жељене рибице. Прелазак са једног на други панел имплементиран је као и прелазак између главних панела. Разлика је у томе што при прибављању података, апликација мора да пошаље захтев серверској страни, након чега прима и исписује дате податке.

### Имплементација пуњења и брисања листе рибица у корпи

Да би се поставио одређени артикал у корпу потребно је да из панела у којем се информишемо о рибици притиснемо дугме на којем пише “Стави у корпу“. Након тога се активира метода која пребацује у корпу име рибе, њену цену и број комада. Такође у збир укупне цене додаје и цену стављене рибице. Након тога избацује обавештење којим се потврђује успешност радње.

public void staviUKorpu(){  
 String s = labelImeRibe.getText() + "( " + cena.getText() +" ) \* " + spinner.getValue();  
 ukupnaCena += (Integer.*parseInt*(cena.getText().replace(" din.","")) \* (int)spinner.getValue());  
 labelUkupnaCena.setText("" + ukupnaCena);  
 listaKorpa.getItems().add(s);  
 Alert obavestenje = new Alert(Alert.AlertType.*INFORMATION*);  
 obavestenje.setHeaderText(labelImeRibe.getText() + " (" + spinner.getValue() + " kom.)");  
 obavestenje.setTitle("Obavestenje");  
 obavestenje.setContentText("Ubaceno u korpu");  
 obavestenje.showAndWait();  
}

У првом реду кода се прави стринг који садржи име рибе, цену и број комада. Након тога се укупној вредности додаје вредност рибице помножена са бројем јединки и записује у лабелу која представља укупну цену. Као потвда приказује се објекат *Alert* који обавештава корисника.

Да би се нешто обрисало из корпе потребно га је селектовати и притиснути одређено дугме. Након тога се активира метода која брише селектовано.

public void izbrisiSelektovano(){  
 int indeks = listaKorpa.getSelectionModel().getSelectedIndex();  
 if(indeks != -1){  
 List<String> brojevi = Arrays.*asList*(listaKorpa.getSelectionModel().getSelectedItem().  
 replaceAll("[^0-9]+", " ").trim().split(" "));  
 int oduzmiOdCene = (Integer.*parseInt*(brojevi.get(0)) \* Integer.*parseInt*(brojevi.get(1)));  
 ukupnaCena -= oduzmiOdCene;  
 labelUkupnaCena.setText(ukupnaCena + "");  
 listaKorpa.getItems().remove(indeks);  
 }  
}

У првом реду се сазнаје индекс ствари која је селектована. Након тога, уколико индекс није -1, што је у случају ако се ништа не селектује, прави листа која садржи вредности цене и броја јединки селектоване рибе. Ти бројеви се помноже и одузму од укупне цене. На крају се брише жељена ствар.

### Имплементација додавања нових риба

На серверској страни је направљена класа која се зове Риба. Преко ње се може направити објекат који садржи атрибуте који описују врсту рибице. Класа садржи и статистички атрибут типа “ArrayList<Riba>“ који складишти објекте типа Риба. Она се аутоматски пуни сваки пут када се направи нов објекат. Ова листа објеката, рибица, се шаље на клијентску апликацију.

Да би продавац успео да убаци нову рибицу, мора се само направити нов објекат типа Риба. Преко сервера се то ради тако сто се само попуне празна поља која представљају атрибуте објекта и тако што се кликне дугме за потврду.

public void praviObjekatRiba(){  
if(!(ime.getText().equals("")||naucnoIme.getText().equals("")||temperament.getText().equals("")||ishrana.getText().equals("")||velicinaAkvarijuma.getText().equals("")||ph.getText().equals("")||temperatura.getText().equals("")||velicinaRibe.getText().equals("")||cena.getText().equals(""))){  
 String apsolutnaPutanjaSlike = putanjaSlike.getText();  
 String imeJpgDatoteke = new File(apsolutnaPutanjaSlike).getName();  
 try {  
 BufferedImage image = ImageIO.*read*(new File(apsolutnaPutanjaSlike));  
 ImageIO.*write*(image,"jpg",new File("src\\server\\slike\\" + imeJpgDatoteke));  
 } catch (IOException e) {  
 e.printStackTrace();  
 }  
 new Riba(ime.getText(),naucnoIme.getText(),temperament.getText(),ishrana.getText(),Integer.*parseInt*(velicinaAkvarijuma.getText()), ph.getText(),temperatura.getText(),velicinaRibe.getText(),Integer.*parseInt*(cena.getText()),imeJpgDatoteke) ;  
 }  
 else {  
 Alert obavestenje = new Alert(Alert.AlertType.*WARNING*);  
 obavestenje.setHeaderText("Nisu popunjena sva polja!");  
 obavestenje.setTitle("Prazna polja");  
 obavestenje.setContentText("Popuniti prazna polja");  
 obavestenje.showAndWait();  
 }  
}

Код приказује како се узимају вредности из поља и како се искоришћавају да би се направио нов објекат Риба. Акција је обезбеђена условом да не сме ниједно поље остати празно.

### Имплементација додавања нових слика у галерију

На серверској страни постоји директоријум који садржи слике које се постављају у галерију. Додавање нове слике је само додавање .jpg фајла у овај директоријум. Продавац то може учинити ако једноставно превуче жељену слику на обележено место у апликацији. Кад то уради активира код за убацивање слика.

public void dodajSliku(DragEvent dragEvent) {  
 Dragboard dragboard = dragEvent.getDragboard();  
 File file= dragboard.getFiles().get(0);  
 dragboard.getFiles().clear();  
 try {  
 BufferedImage image = ImageIO.*read*(file);  
 ImageIO.*write*(image,"jpg",new File("src\\server\\slikeGalerija\\" + file.getName()));  
 } catch (IOException e) {  
 e.printStackTrace();  
 }  
 Alert obavestenje = new Alert(Alert.AlertType.*CONFIRMATION*);  
 obavestenje.setHeaderText(file.getName());  
 obavestenje.setTitle("Obavestenje");  
 obavestenje.setContentText("Slika je ubacena u galeriju");  
 obavestenje.showAndWait();  
}

У прва три реда се узима фајл који представља слику. Након тога се фајл смешта у објекат типа BufferedImage. Он чува податке о слици и омогућава пребацивање слике. Пребацивање се врши у следећем реду преко методе ImageIO.write(). Последњи редови служе за обавештавање о успешности радње.

public void setOnDragOver(DragEvent dragEvent) {  
 if (dragEvent.getDragboard().hasFiles()) {  
 dragEvent.acceptTransferModes(TransferMode.*LINK*);  
 }  
}

Метода setOnDragOver() омогућава рад методе за додавање слике. Када се детектује присутност фајла у пољу, позива се dragEvent.acceptTransferModes() који прослеђује трансфер мод. Мод у овом примеру указује да је порджано повезивање података.

## Мрежно програмирање и Статистика у програму

Следи обрада мрежног програмирања клијентског и серверског дела, њихово повезовање, као и како сервер обрађује захтеве клијента. Прво ће бити описанa класа SlikeIO, па како сервер шаље клијенту листу риба и слике за галерију и како их клијент прихвата. Након тога ће се обрадити како се врше наруџбине, на који начин су послате и примљене. На крају је приказано како су дефинисани статистички подаци.

### Класа SlikeIO

Ова класа садржи методе које се користе при мрежномтранспорту слика од сервера ка клијенту. Њене најбитније методе су за слање и примање слика.

public static void slanjeSlike(String putanja,String imeDatoteke,ServerSocket ss){  
 ServerSocket serverSocket = ss;  
 Socket socket= null;  
 try{  
 socket = serverSocket.accept();  
 OutputStream outputStream = socket.getOutputStream();  
 outputStream.write((imeDatoteke + "\n").getBytes());  
 BufferedImage image= ImageIO.*read*(new File(putanja));  
 ByteArrayOutputStream baos= new baos();  
 ImageIO.*write*(image, "jpg", baos);  
 byte[] size = ByteBuffer.*allocate*(4).putInt(baos.size()).array();  
 outputStream.write(size);  
 outputStream.write(baos.toByteArray());  
 outputStream.flush();  
 socket.close();  
 } catch (IOException e) {  
 e.printStackTrace();  
 }  
}

На прва два реда се праве неопходни објекти за мрежно програмирање. Након тога се чека захтев клијента. Када се клијент појави, шаље му се име датотеке која ће бити послата. Клијент би тада требало да то име резервише и када буде користио послату слику позиваће је тим именом. Слика се затим смешта у објекат типа BufferedImage како би се омогућило слање слике. У следећим редовима се врши конкретно слање слике клијенту тако што се прво резервише величина слике, па се записује и шаље.

public static Object[] primanjeSlike(){  
 Socket socket = null;  
 Object[] o = new Object[2];  
 try {  
 socket = new Socket("127.0.0.1", 7777);  
 InputStream inputStream = socket.getInputStream();  
 BufferedReader bf = new BufferedReader(new InputStreamReader(socket.getInputStream()));  
 String imeDatoteke = bf.readLine();  
 byte[] sizeAr = new byte[4];  
 inputStream.read(sizeAr);  
 int size = ByteBuffer.*wrap*(sizeAr).asIntBuffer().get();  
 byte[] imageAr = new byte[size];  
 inputStream.read(imageAr);  
 BufferedImage image = ImageIO.*read*(new ByteArrayInputStream(imageAr));  
 Image slika = SwingFXUtils.*toFXImage*(image,null);  
 socket.close();  
 bf.close();  
 inputStream.close();  
 o[0] = imeDatoteke;  
 o[1] = slika;  
 }catch (  
 UnknownHostException ex) {  
 ex.printStackTrace();  
 } catch (IOException ex) {  
 ex.printStackTrace();  
 }  
 return o;  
}

Примање слике је функција која се врши са стране клијента. У прва два реда се резервише неопходни елемент за мрежну комуникацију, као и низ од два објеката. Након прављења објеката за примање послатог садржаја, клијент прима име датотеке која је послата. Резервацијом величине слике обезбеђује се њено целокупно слање. Слика се прима као објекат BufferedImage, али се конвертује у објекат који може бити доступан апликацији методом SwingFXUtils.toFXImage(). Слика се складишти у низ који садржи њено име, као и саму слику.

### Слање листе риба и слика за галерију

Листа риба које се продају и слика из галерије клијент добија преко мреже од сервера. У зависности од скупа слика који је направио продавац на клијентској страни се приказују слике. Исто важи и за листу рибу.

При покретању клијентског програма врши се иницијализација. Тада клијент серверу шаље захтеве за приступ слика и листе риба. Сервер обрађује захтев, шаље потребне податке и одваја посебну нит која треба да служи клијента у даљим захтевима. Слике се шаљу методама класе SlikeIO. Следи приказ кода на серверској, па на клијентској страни.

serverSocket = new ServerSocket(7777);  
while(true){  
 SlikeIO.*slanjeBrojaTransfera*(serverSocket,"slike");  
 for(String imeDatoteke:new File("src\\server\\slike").list()) {  
 SlikeIO.*slanjeSlike*("src\\server\\slike\\" + imeDatoteke,imeDatoteke,serverSocket);  
 }  
 SlikeIO.*slanjeBrojaTransfera*(serverSocket,"slikeGalerija");  
 for(String imeDatoteke:new File("src\\server\\slikeGalerija").list()) {  
 SlikeIO.*slanjeSlike*("src\\server\\slikeGalerija\\" + imeDatoteke,imeDatoteke,serverSocket);  
 }  
 socket = serverSocket.accept();  
 new KlijentServerThread(socket, in, out).start();

Одређује се колики је број слика у директоријуму и тај број се шаље клијенту како би се знало колико пута треба користити методу за примање. У *for* петљи се шаљу све слике које се налазе у директоријуму за слике. То се исто понавља и при слању слика за галерију. Сервер на крају прави нит која обрађује захтеве клијента.

br = SlikeIO.*primanjeBrojaTransfera*();  
System.*out*.println("Slanje slika galerija...Broj slanja " + br);  
for (int i = 0; i < br; i++) {  
 Image slikaGalerija = SlikeIO.*primanjeSlikeGalerija*();  
 ImageView view = new ImageView(slikaGalerija);  
 view.prefHeight(250); view.prefWidth(250);  
 view.maxHeight(250); view.maxWidth(250); view.setPreserveRatio(true); view.setSmooth(true);  
 flowPanelGalerija.getChildren().add(view);  
}

На клијету се складишти вредност броја трансфера. У наставку, у петљи се понавља примање слика у зависности од броја трансфера. Када се слика прими, прави се графички елемент за приказивање те слике и смешта се у одређени панел. Исти принцип је искоришћен и код прављења и приказивања листе рибица.

### Обрада наруџбина

Када клијент жели да поручи нарџбину онда он серверу шаље своје име, адресу, број телефона, оцену коју је дао продавници, укупну цену и листу рибица коју жели да купи. Следећи код приказује имплементацију наведеног:

String porudzbina = "";  
for (String s:listaKorpa.getItems()) {  
 porudzbina += s + "/";  
}

out.write("porudzbina\n".getBytes());  
out.write((textFImePorucilac.getText() + "\n").getBytes());  
out.write((textFAdresa.getText() + "\n").getBytes());  
out.write((textFBrTel.getText() + "\n").getBytes());  
out.write((porudzbina + "\n").getBytes());  
out.write((ocena + "\n").getBytes());  
out.write((ukupnaCena + "\n").getBytes());

У стрингу поруџбина се чувају елементи листе. Сваки елемент је одвојен косом цртом. У осталим редовима се шаље поруџбина. Прво слање серверу је реч поруџбина како би сервер знао који захтев треба да обради. У осталим редовима се шаљу купчеви подаци, поруџбина, оцена и укупна цена.

Са серверске стране се примају послати подаци. Објекат класе Купац је задужен за памћење тих информација. Довољно је само направити објекат таквог типа и аутоматски се прави листа купаца.

public Kupac(String ime, String adresa, String brojTel, String kupljeneStvari) {  
 this.id = *brojKupaca* + "";  
 this.ime = ime;  
 this.adresa = adresa;  
 this.brojTel = brojTel;  
 this.kupljeneStvari = kupljeneStvari;  
 *brojKupaca*++;  
 *listaKupaca*.add(this);  
}

Наведени код представља конструктор објекта класе Купац. Следи код који представља коришћење овог објекта.

if(zahtev.equals("porudzbina")){  
 String kupljeneStvari = new Kupac(in.readLine(),in.readLine(),in.readLine(),  
 in.readLine().replaceAll("/","\n")).getKupljeneStvari();  
 Riba.*racunajBrProdatih*(kupljeneStvari);  
 Statistika.*plusBrojOcena*();  
 Statistika.*plusZbirOcena*(Integer.*parseInt*(in.readLine()));  
 Statistika.*plusUkupnaZarada*(Integer.*parseInt*(in.readLine()));

Да би сервер обрадио поруџбину мора да буде испуњен услов у првом реду кода. У другом реду се прави објекат класе Купац. Њени параметри се добијају од података које шаље клијент. Ради скраћивања кода, у једном реду је направљен објекат и направљен стринг који садржи обрађену информацију о купљеним стварима. Класа Риба има функцију која рачуна број продатих врста рибица и она управо користи претходно обрађен стринг. Функције које се тичу статистике биће детаљно обрађене у следећем делу, али у овом коду рачунају колика је просечна оцена продавнице и колика је укупна зарада.

### Дефинисање статистичких података и класа Статистика

Класа Статистика садржи само статичке атрибуте и методе. Њени атрибути су збир, број и просечна оцена, посећеност, тренутна посећеност, укупна зарада и садржи две листе које описују пет најјефтинијих и најпопуларнијих рибица. Све методе су предвиђене за обраду атрибута класе. Следе примери метода:

static void plusZbirOcena(int ocena) {  
 Statistika.*zbirOcena* += ocena;  
 Statistika.*prosecnaOcena* = (double) Statistika.*zbirOcena*/(double)Statistika.*brojOcena*;  
}

Ова метода служи за рачунање просечне оцене продавнице. Укупан збир оцене се дели са укупним бројем оцена и ту вредност смешта у атрибут класе за просечну оцену.

synchronized static void plusUkupnaZarada(int i){  
 *ukupnaZarada* += i;  
}

Метода plusUkupnaZarada() функционише веома једноставно. Параметар који означава колика је цена неког артикла коју је купац наручио, сабира се са вредношћу атрибута класе који представља укупну зараду.

static ArrayList<Riba> getTop5Najjeftiniji() {  
 ArrayList<Riba> r = new ArrayList<>();  
 for (int i = 0; i < 5; i++) {  
 r.add(Riba.*porediPoCeni*().get(i));  
 }  
 return r;  
}

Наведена метода служи за прављење листе објеката Риба која је сортирана по најнижој цени. У петљи се попуњава листа која је направљена у првом реду и то тако што се узимају елемети листе коју враћа метода која пореди по рибице по цени. На исти начин је конструисана и метода која пореди рибице по продаји.

Све остале методе служе за једноставно инкрементирање одређеног садржаја или за његово узимање.

Статистички подаци на серверу се ажурирају на сваких десет секунди, као и листа купаца. Због тога њихове функције су одвојене у посебну нит која је активна током рада сервера.

new Thread(() -> { /// azuriranje statistike i liste kupaca na 10 sec  
 while (true){  
 azurirajListuKupaca();  
 azurirajStatistiku();  
 try {  
 Thread.*sleep*(10000);  
 } catch (InterruptedException e) {  
 e.printStackTrace();  
 }  
 }  
}).start();

Ажурурање листе купаца је једноставно графичко представљање листе која се налази у класи Купац преко графичког објекта класе *ListView*.

public void azurirajListuKupaca(){  
 listaKupaca.getItems().setAll(Kupac.*getListaKupaca*());  
}

Ажурирање статистике врши обраду статистичких података помоћу већ описаних метода и њихово приказивање.

public void azurirajStatistiku(){  
 Platform.*runLater*(() -> {  
 labelPosecenost.setText(Statistika.*getPosecenost*() + "");  
 labelPrOcena.setText(Statistika.*getProsecnaOcena*() + "");  
 labelTrPosecenost.setText(Statistika.*getTrenutnaPosecenost*() + "");  
 labelaZarada.setText(Statistika.*getUkupnaZarada*() + "");  
 top5Jef.getItems().setAll(Statistika.*getTop5Najjeftiniji*());  
 top5Pop.getItems().setAll(Statistika.*getTop5Najprodavaniji*());  
 });  
}

# Опис корисничког интерфејса

У овом поглављу бавићемо се графичким интерфејсом програма, како клијентске тако и серверске стране. Казаће се који графички елементи су употребљени, како су уређени и која је њихова улога. Сва графика програма је рађена преко програма *SceneBuilder*.

## Кориснички интерфејс на клијентској страни програма

Када се покрене клијентски програм, примарне опције које корисник може изабрати су у главном менију који се налази на левој страни (Слика 4.1) и корпа која се налази у горњем десном углу прозора апликације (Слика 4.2).



Слика 4.1 – Мени клијенске стране програма.

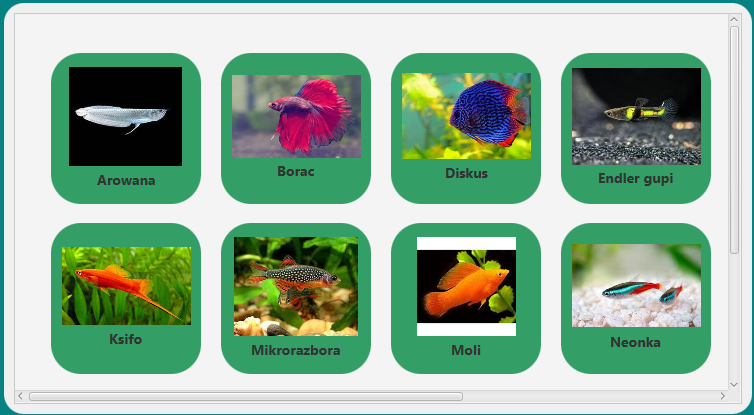


Слика 4.2 – Назив продавнице и дугме Корпа.

Избор у менију представљају графички елементи класе *Button*. Преко њих корисник може одабрати који панел жели да користи. У наставку ће бити приказан сваки панел.

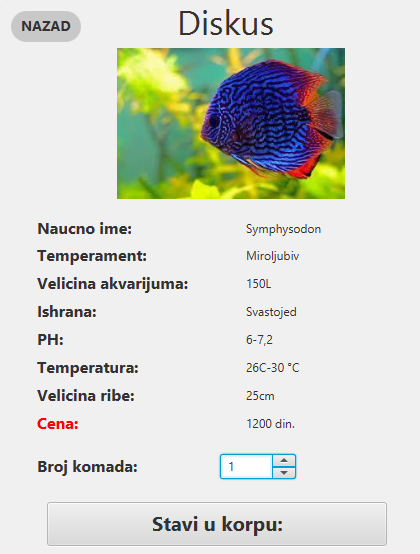
### Панел рибице

У овом панелу (Слика 4.3) је постављена листа рибица које се продају. Панел је објекат класе *ScrollPane*, како би избор рибица могао бити већи. Сваку ставку у панелу заправо представљају дугмићи. Свако дугме садржи слику и име које означавају која је врста рибице у питању.



Слика 4.3 – Приказ панела рибице.

Притиском на дугме из овог панела појављује се садржај који описује врсту рибица и могућност смештања жељеног броја рибица у корпу (Слика 4.4).



Слика 4.4 – Приказ информација о рибици.

### Панел корпа

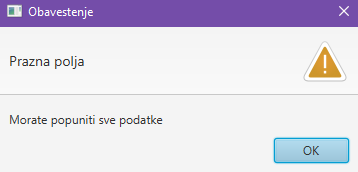
У овом панелу (Слика 4.5) купац може видети шта је све поставио у корпу, колика је цена сваког артикла и колика је укупна цена за плаћање. Преко овог панела постоји могућност наручивања, уколико се попуне сва потребна поља.



Слика 4.5 – Панел корпа.

Подаци које попуњава купац се врше преко графичког објекта *TextField*. За листу артикла је коришћен *ListView*. Остали графички елементи су дугмићи и лабеле.

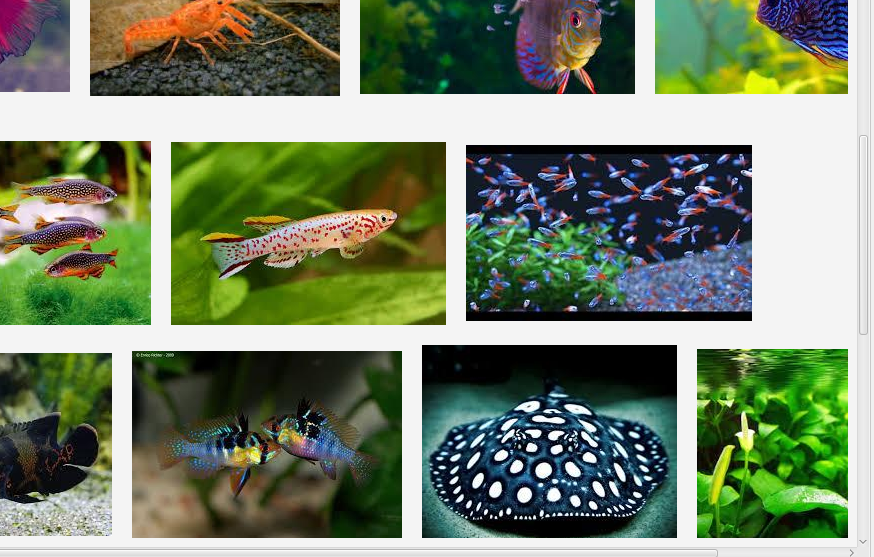
Уколико се не попуне сва поља појављује се обавештење које указује да се морају попунити сва поља (Слика 4.6).



Слика 4.6 – Обавештење коју указује да нису сви подаци попуњени.

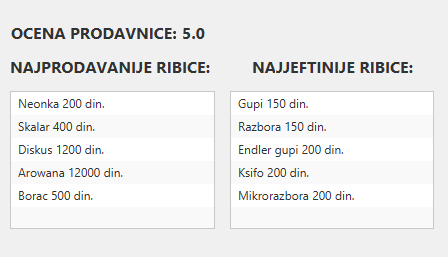
### Панел галерија, статистика и О нама

Ови панели нису интерактивни, они служе за информисање. Корисник може само да посматра слике и чита садржај.



Слика 4.7 – Панел галерија.

У галерији (Слика 4.7) постоје само слике које корисник може посматрати како би стекао бољу представу о акваријумским рибицама. Графички елементи који се користе су објекти класе *ImageView*.



Слика 4.8 – Панел статистика.

У панелу статистика је доступан део информација о статистици продавнице (Слика 4.8).

У панелу који носи име О нама се налазе информације које се тичу радног времена продавнице, контактног броја и електронске поште, као и физичка адреса продавнице (Слика 4.9).

Елементи који су коришћени у овом панелу су лабеле.



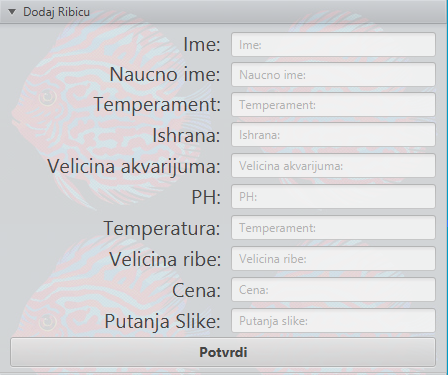
Слика 4.9 – Панел О нама.

## Кориснички интерфејс на серверској страни програма

Следи опис графичког интерфејса на серверској страни. На серверу постоје четири панела, која су смештена у објекат класе *Accordion*. Панел на којем се додају нове рибице и панел који служи за убацивање слика биће обрађени први. Панел купци и статистика ће бити описани посебно.

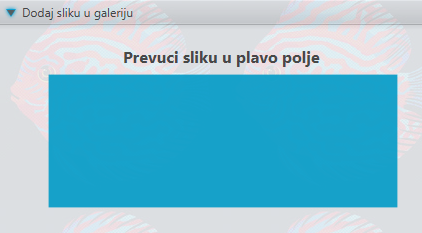
### Панел за додавање нове рибице и панел за убацивање слике

Преко панела за додавање можемо убацити нову рибицу у продају (Слика 4.10). Корисник само мора попунити празна поља која описују рибу. Поље које служи за убацивање путање слике може да се попуни ручно или само да се у њу превуче слика.



Слика 4.10 – Панел за додавање нових рибица.

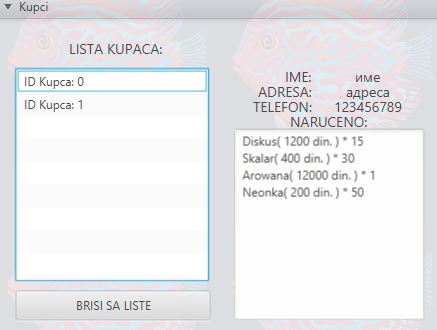
Панел за убацивање слика у галерију садржи само једну лабелу и плаво поље (Слика 4.11). Да би се убацила слика потребно је превући нову слику у плаво поље.



Слика 4.11 – Панел за додавање нових слика.

### Панел купци

У панелу купци може се посматрати листа купаца и прегледати шта су наручили и који су њихови подаци (Слика 4.12).



Слика 4.12 – Панел купци.

Са леве стране је исписана листа купаца. Купци су означени својим идентификационим бројем. Уколико селектујемо неког купца, са десне стране можемо видети његове податке и наруџбину. Такође можемо и обрисати селектованог купца са листе уколико желимо.

### Панел статистика

У панелу статистика се налазе статистички подаци који пружају бољи увид у рад продавнице (Слика 4.13). Може се видети колико је купаца посетило продавницу, тренутни број људи, просечна оцена, укупна зарада и две листе које описују најјефтинију и напродаванију робу.



Слика 4.13 – Панел статистика.

# ЗАКЉУЧАК

У раду је описана идеја, функционалност и изглед једне апликације за акваристичку продавницу. Оваква апликација олакшава трговину. Купци могу наручити робу без одласка у продавницу.

Објашњена је архитектура и клијентске и серверске стране програма. Купци преко клијентске апликације могу лако наручити рибице које желе. Наруџбина стиже у продавцу преко сервера, као и сви потребни подаци за њено слање. Продавац има увид у свој рад преко статистичких података који се редовно ажурирају.

Цела функционалност програма је описана. Битне класе за статистику, за рибице, купце, као и за трансфер слика су такође поменуте. Мрежно програмирање нам је омогућило да спојимо клијентску и серверску страну како би успешно комницирале. Приликом покретања клијентске стране, сервер шаље све потребне материјале како би се клијентска апликација покренула правилно. Такође клијент при наручивању комуницира са сервером.

Цела апликација је пропраћена графичким интерфејсом, како би купац стекао најбољи утисак о продавници и лако је користио. Исто важи и са серверске стране.

Овакав тип апликација и њена имплементација се не мора ограничити само на продају акваријумских рибица. Може се употребити за било какаву трговину.

.

# ИНДЕКС ПОЈМОВА

*Alert* објекат, 7

Апликација, 5, 6, 9, 11, 15

BufferedImage, 10

Графички интерфејс, 1, 15, 20, 22

Класа Купац, 12

Класа Риба, 8

Класа СликеИО, 10

Клијентска страна апликације, 2

Код, 6, 7, 8, 11, 12, 13

Метода, 6, 7, 9, 13, 14

Мрежа, 6, 10, 11, 22

ООП, 1

Серверска страна апликације, 4

Статистика, 5, 10, 13, 18, 21

SceneBuilder, 15

Функција, 6

# ЛИТЕРАТУРА

[1] Herberd Schildt, *Java JDK 7: kompletan priručnik,prevod 8. izdanja*, Mikro knjiga, Београд, 2012.

[2] Yakov Fain, *Java 8 programiranje,* Kompjuter biblioteka, Beograd, 2015.

[3] Easy Programming, " Sending image object through socket in java classic example", <http://easywayprogramming.blogspot.com/2012/06/sending-image-object-through-socket-in.html>, преузето: септембар 2019.

# Прилози

# изјава о академскoј честитости

**ИЗЈАВА О АКАДЕМСКОЈ ЧЕСТИТОСТИ**

|  |  |
| --- | --- |
| **Студент (име, име једног родитеља и презиме):** |  |
| **Број индекса:** |  |

Под пуном моралном, материјалном, дисциплинском и кривичном одговорношћу изјављујем да је завршни рад, под насловом:

1. резултат сопственог истраживачког рада;
2. да овaj рад, ни у целини, нити у деловима, нисам пријављиво/ла на другим високошколским установама;
3. да нисам повредио/ла ауторска права, нити злоупотребио/ла интелектуалну својину других лица;
4. да сам рад и мишљења других аутора које сам користио/ла у овом раду назначио/ла или цитирао/ла у складу са Упутством;
5. да су сви радови и мишљења других аутора наведени у списку литературе/референци који је саставни део овог рада, пописани у складу са Упутством;
6. да сам свестан/свесна да је плагијат коришћење туђих радова у било ком облику (као цитата, прафраза, слика, табела, дијаграма, дизајна, планова, фотографија, филма, музике, формула, вебсајтова, компјутерских програма и сл.) без навођења аутора или представљање туђих ауторских дела као мојих, кажњиво по закону (Закон о ауторском и сродним правима), као и других закона и одговарајућих аката Високе школе електротехнике и рачунарства струковних студија у Београду;
7. да је електронска верзија овог рада идентична штампаном примерку овог рада и да пристајем на његово објављивање под условима прописаним актима Високе школе електротехнике и рачунарства струковних студија у Београду;
8. да сам свестан/свесна последица уколико се докаже да је овај рад плагијат.

У Београду, \_\_. \_\_. 201\_. године

Својеручни потпис студента

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_