SVEUČILIŠTE U MOSTARU

FAKULTET STROJARSTVA RAČUNARSTVA I ELEKTROTEHNIKE

# 

# Programirnje u Javi

# *Projektni zadatak:* Construction company

Mijo Kozina i Matko Dugandžić

Akademska godina 2021/2022.

**PROJEKTNI ZADATAK:**

**Opis problema:**

Tema ovog projekta je optimizacija i pojednostavljivanje poslovanja građevinske tvrtke. Ideja svega bila je napraviti aplikaciju koja će omogućiti lakšu interakciju između poslodavaca i radnika, te naravno i bolji uvid poslodavca u odrađene poslove i potrošeni materijal na tim poslovima.

Naime, bez ovakve aplikacije, poslodavac bi za željeni uvid o radu bio primoran pretraživati arhivu spisa i računa te i dalje ne bi imao čist i jasan uvid. Korištenjem ove aplikacije, poslodavac bi imao jasan i detaljan uvid o izvršenom radu, radnicima i klijentu (kupcu). Određeni radni nalog (račun) može potražiti po datumu, imenu kupca, imenima radnika koji su radili na tom radnom nalogu ili pak po jedinstvenom broju računa. Također, ukoliko poslodavca zanima pretraga svih računa, na kojima je radio određeni radnik ili svi računi koji su izdati određenom kupcu, takva pretraga je naravno moguća. U slučaju da poslodavac zaprimi bilo kakvu žalbu od klijenta, za međusobno razumjevanje i riješavanje problema klijent jedino treba znati broj računa koji je otisnut u zaglavlju istog. Budući da se aplikacija brine o jedinstvenom numeriranju računa prilikom žalbe ne može doći do nesporazuma ukoliko više klijenata ima isto ime. Nadalje, poslodavac ima mogućnost uvida u trenutno stanje materijala na stanju, te prilikom izdavanja novog radnog naloga (ili korekcijom postojećeg) stanje materijala se automatski mijenja. Naravno samo unošenje materijala u sustav (prilikom kupovine novih materijala ili slično) je briga administratora.

Ova aplikacija omogućuje poslodavcu da na jednostavan način dodaje nove radnike, klijente, materijale i njihovo stanje na skladištu, te da izda novi radni nalog i da isti dodijeli nekom od svojih radnika (uz Administratorske ovlasti). Taj isti radnik (preko iste aplikacije) bi imao mogućnost pregleda radnih naloga, ali samo onih koji su dodjeljeni izravno njemu. Radnik se može prijaviti u sustav sa svojim korisničkim podatcima koji su mu dodjeljeni od strane poslodavca. Radnik može pregledati i aktivne i predhodne radne naloge koji su mu dodjeljeni.

Poslodavac bi također mogao kao administrator dodati, pregledati, urediti i izbrisati(CRUD) bilo koji podatak iz baze podataka, nad bilo kojom tablicom.

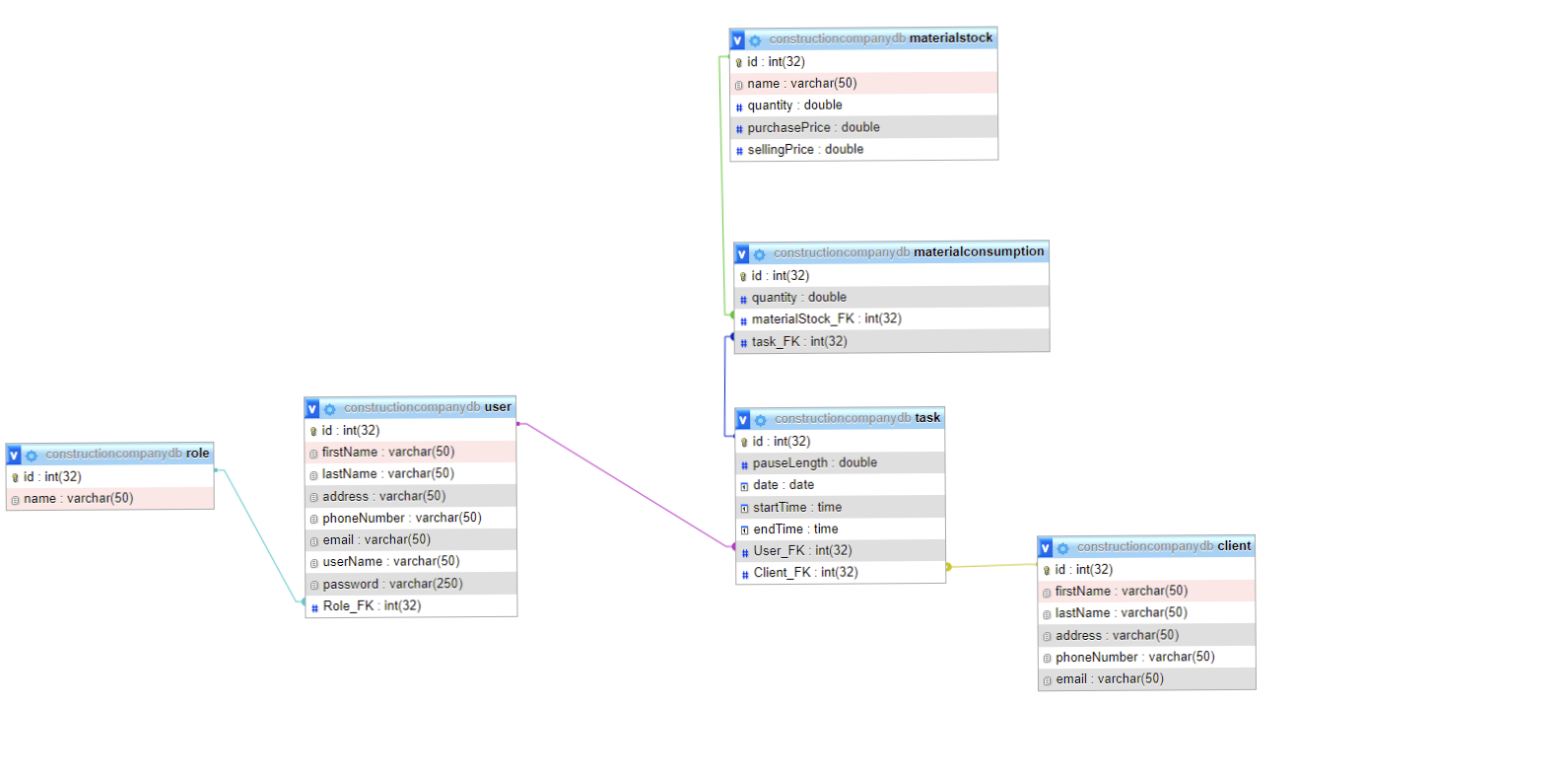
Radnik bi osim pregledanja svojih prethodnih i budućih zadatak mogao još pregledati osnovne podatke o svom profilu i po želji urediti neke od njih.

***1. Modeliranje podataka***

Prvi dio zadatka se radi u .

* Poglavlje 1.1. (Relacijski model podataka) - potrebno je staviti sliku relacijskog modela
* Poglavlje 1.2. (Opis tablica u relacijskom modelu) - potrebno je navesti i opisati sve tablice (i atribute unutar tablica) iz dobivenog modela te na kraju priložiti sliku
* Poglavlje 1.3. (Aplikacija u Javi) - sadrži screenshotove kreiranih formi Javi.

***1.1. Relacijski model podataka***



***1.2. Opis tablica u relacijskom modelu***

User

|  |  |  |
| --- | --- | --- |
| **Naziv atributa** | **Integritet**  **(PK/FK, NULL / NOT NULL)** | **Kratki opis atributa** |
| **id** | PK, NOT NULL | Primarni ključ tablice, auto\_increment |
| **firstName** | NOT NULL | Ime korisnika |
| **lastName** | NOT NULL | Prezime korisnika |
| **address** | NOT NULL | Adresa korisnika |
| **phoneNumber** | NOT NULL | Broj telefona korisnika |
| **email** | NOT NULL | Email korisnika |
| **password** | NOT NULL | Password korisnika |
| **Role\_FK** | FK,NOT NULL | ID role koja mu je dodijeljena |

**Role**

|  |  |  |
| --- | --- | --- |
| **Naziv atributa** | **Integritet**  **(PK/FK, NULL / NOT NULL)** | **Kratki opis atributa** |
| **id** | PK, NOT NULL | Primarni ključ tablice, auto\_increment |
| **name** | NOT NULL | Naziv role |

**Task**

|  |  |  |
| --- | --- | --- |
| **Naziv atributa** | **Integritet**  **(PK/FK, NULL / NOT NULL)** | **Kratki opis atributa** |
| **id** | PK, NOT NULL | Primarni ključ tablice, auto\_increment |
| **name** | NOT NULL | Naziv role |
| **pauseLength** | NOT NULL | Dužina pauze u minutama |
| **Date** | NULL | Datum izvršenja radnog naloga |
| **startTime** | NOT NULL | Vrijeme početka rada |
| **endTime** | NOT NULL | Vrijme završetka rada |
| **User\_FK** | FK, NOT NULL | ID radnika koji je dodijeljen zadatku |
| **Client\_FK** | FK, NOT NULL | ID klijenta kod kojeg se zadatak obavlja |

**Client**

|  |  |  |
| --- | --- | --- |
| **Naziv atributa** | **Integritet**  **(PK/FK, NULL / NOT NULL)** | **Kratki opis atributa** |
| **id** | PK, NOT NULL | Primarni ključ tablice, auto\_increment |
| **firstName** | NOT NULL | Ime klijenta |
| **lastName** | NOT NULL | Prezime klijenta |
| **address** | NOT NULL | Adresa klijenta |
| **phoneNumber** | NOT NULL | Broj telefona klijenta |
| **email** | NOT NULL | Email klijenta |

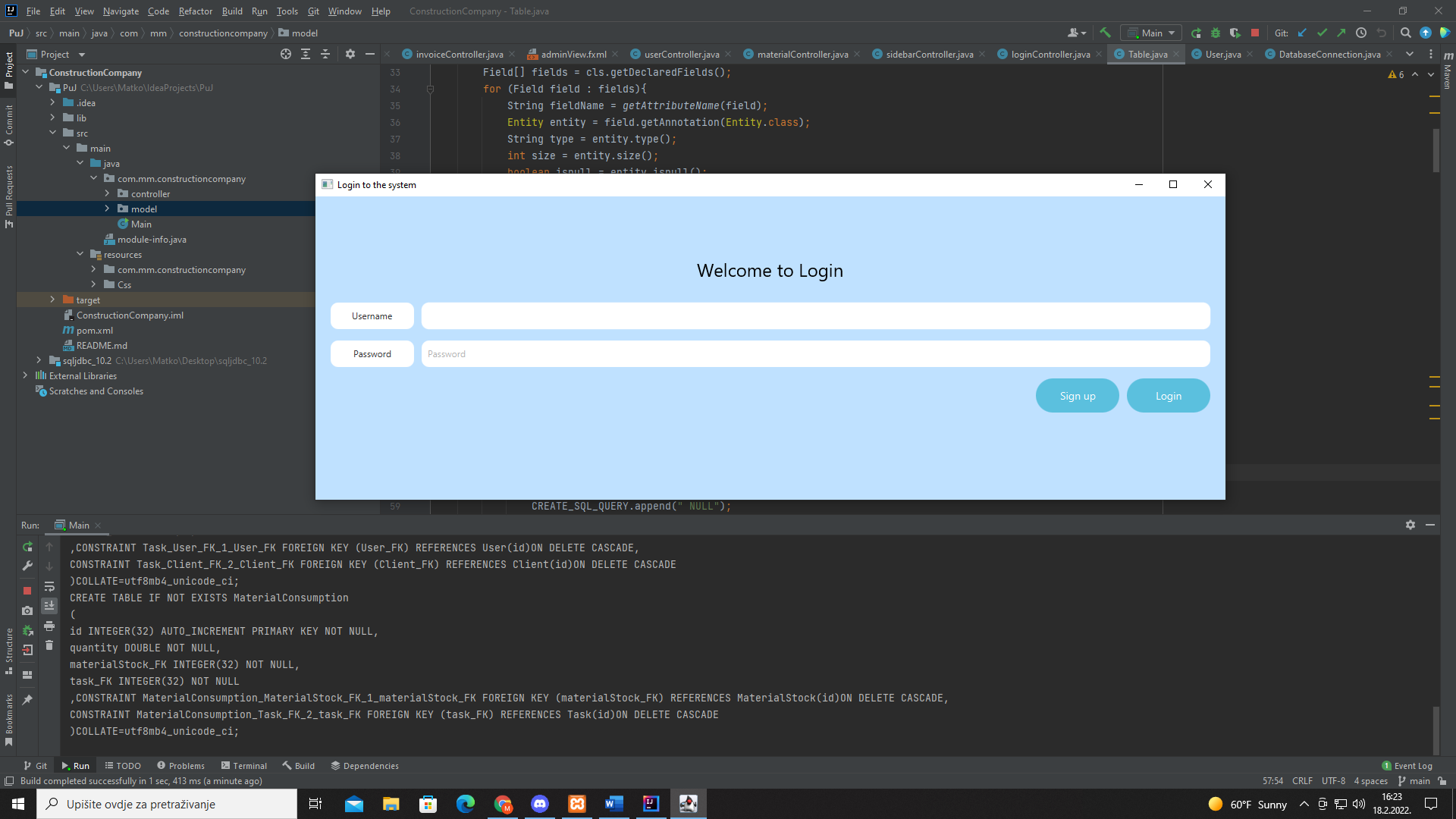
**MaterialStock**

|  |  |  |
| --- | --- | --- |
| **Naziv atributa** | **Integritet**  **(PK/FK, NULL / NOT NULL)** | **Kratki opis atributa** |
| **id** | PK, NOT NULL | Primarni ključ tablice, auto\_increment |
| **name** | NOT NULL | Naziv artikla |
| **quantity** | NOT NULL | Količina na zalihi |
| **purchasePrice** | NOT NULL | Kupovna cijena artikla |
| **sellingPrice** | NOT NULL | Prodajna cijena artikla |

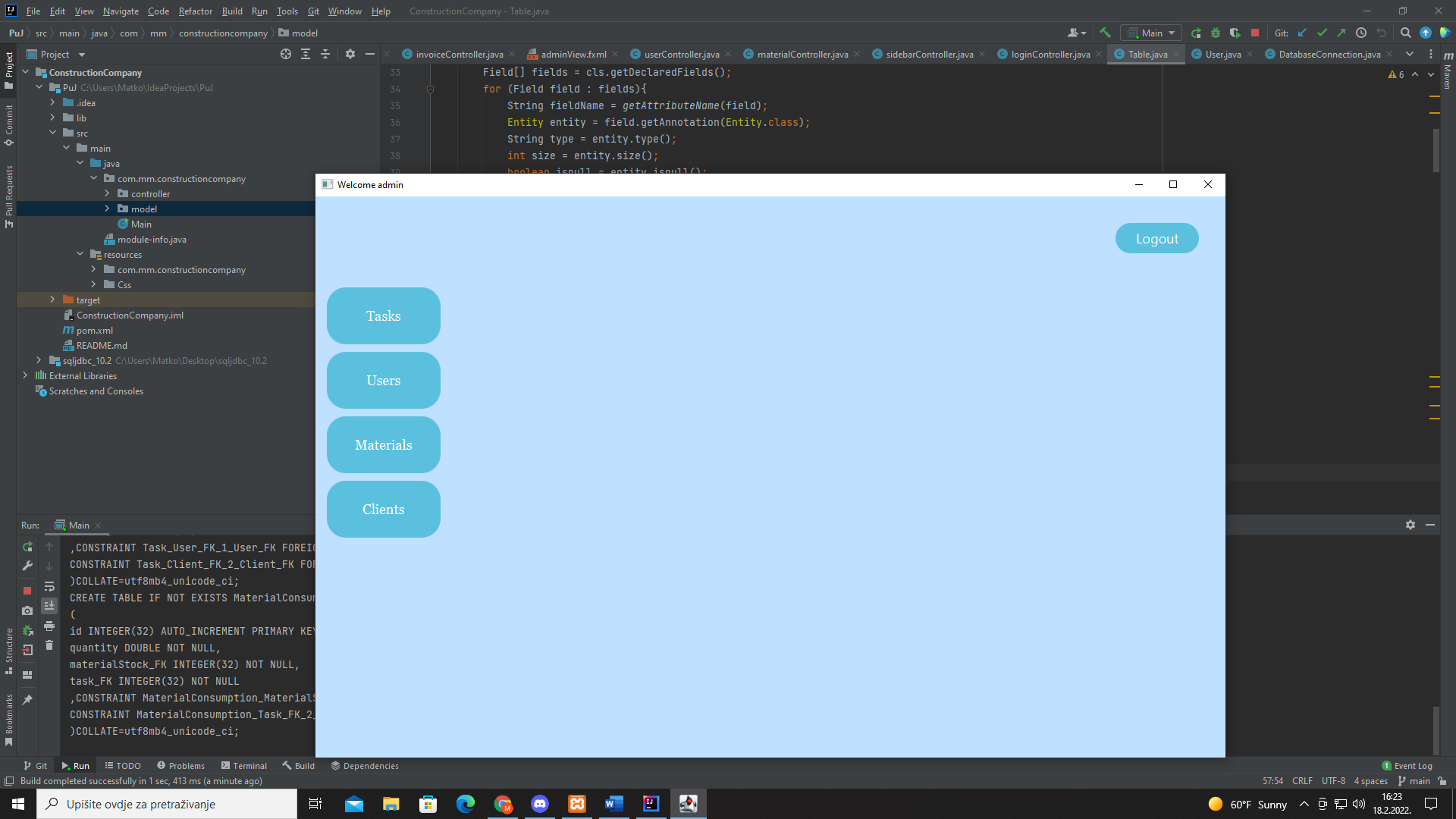
**MaterialConsumption**

|  |  |  |
| --- | --- | --- |
| **Naziv atributa** | **Integritet**  **(PK/FK, NULL / NOT NULL)** | **Kratki opis atributa** |
| **id** | PK, NOT NULL | Primarni ključ tablice, auto\_increment |
| **quantity** | NOT NULL | Količina utrošenog artikla |
| **materialStock\_FK** | FK, NOT NULL | ID utrošenog materijala |
| **task\_FK** | FK, NOT NULL | ID zadatka na kojem se materijal utroši |

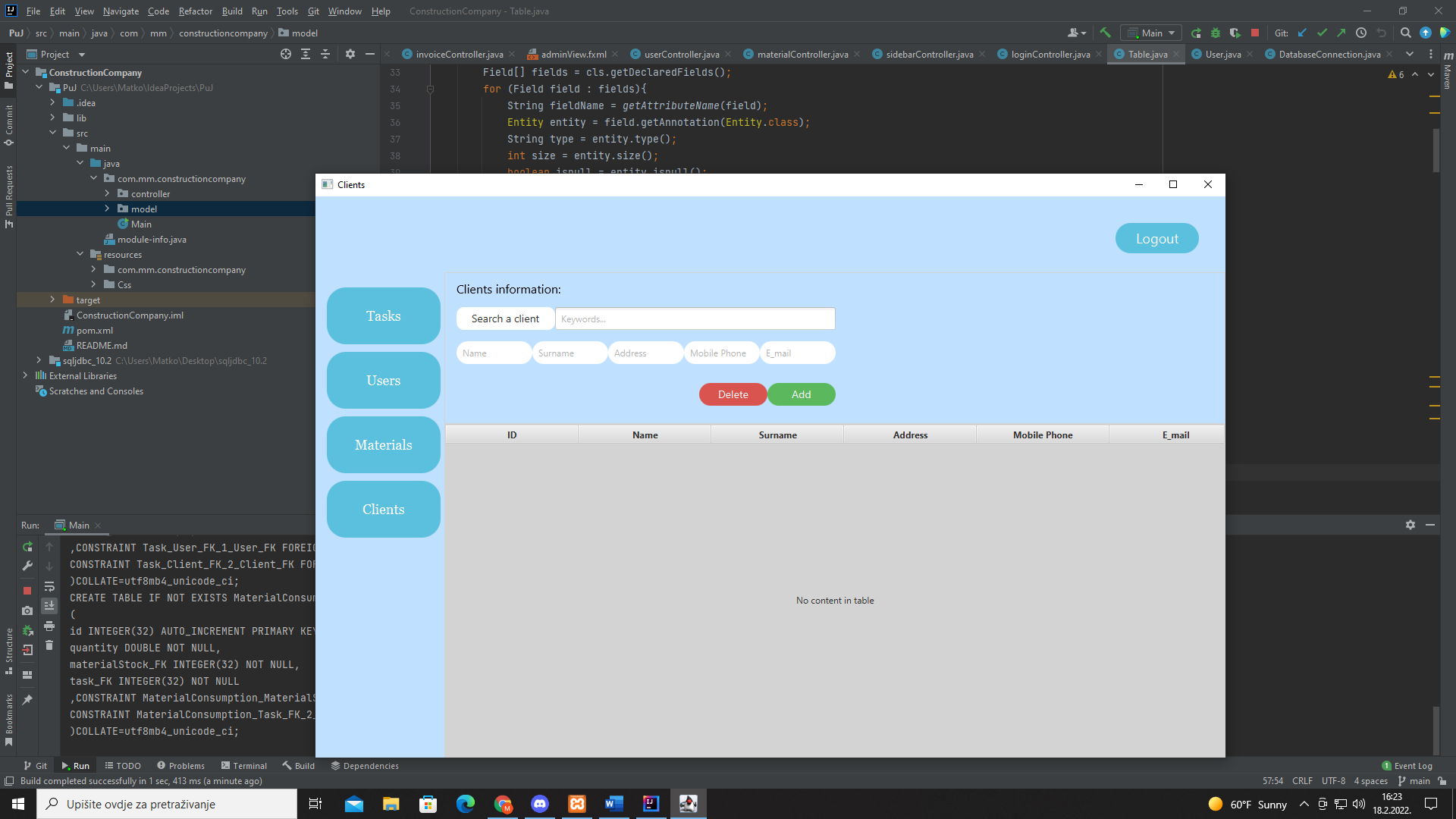
***1.3. Aplikacija u Javi***



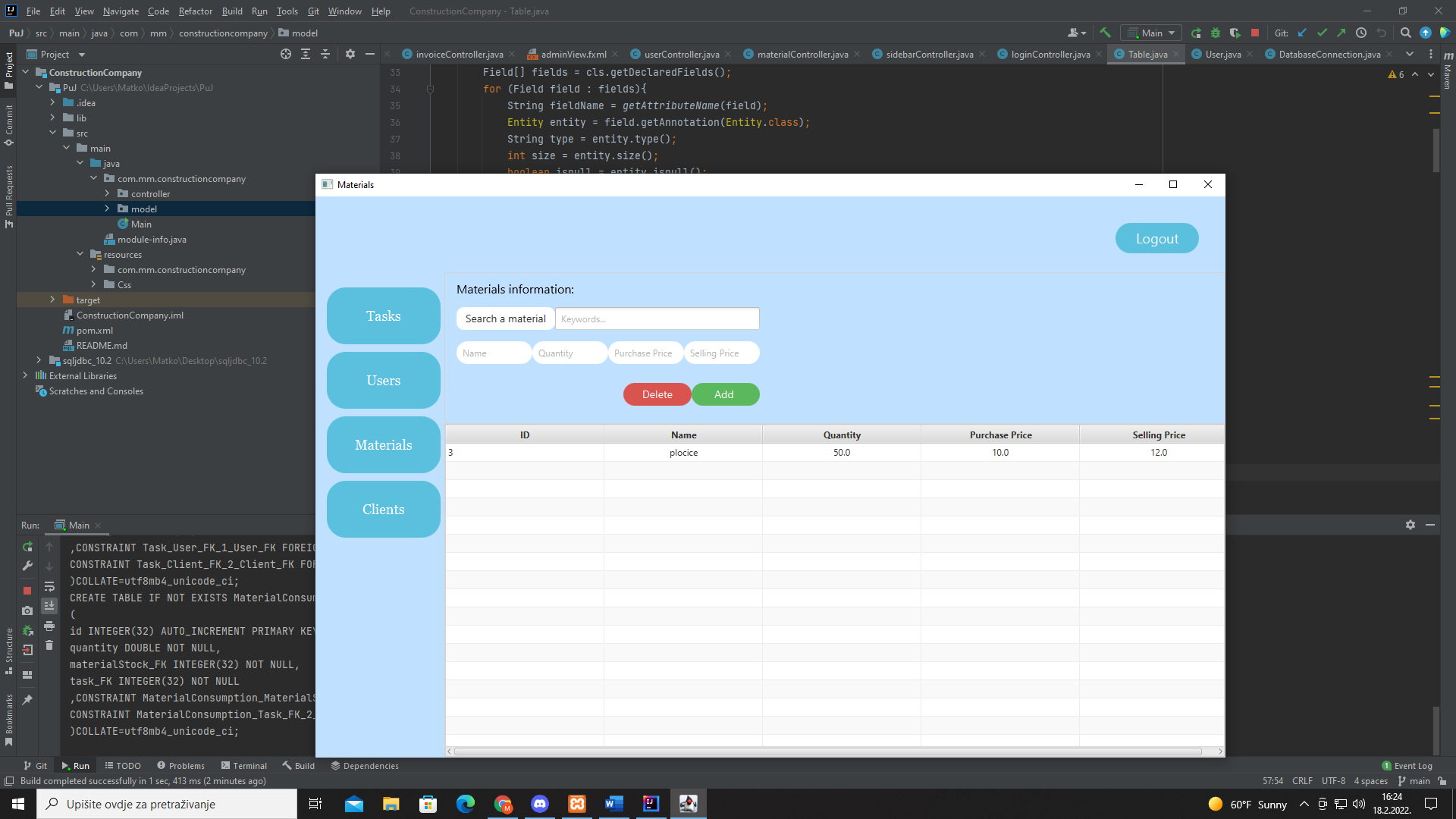
Slika 1 Prijava korisnika na sustav



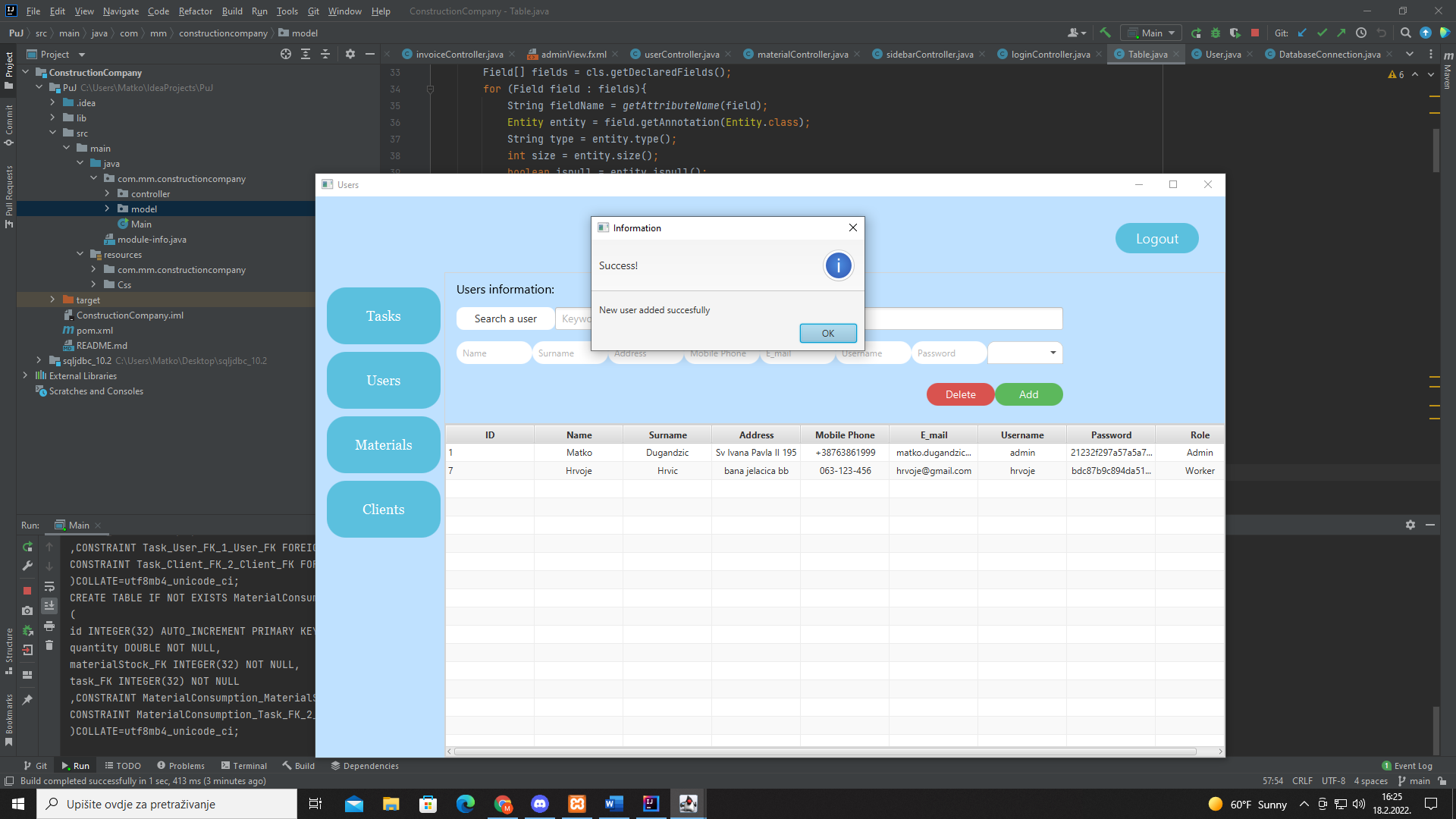
Slika 2 Početni zaslon nakon prijave



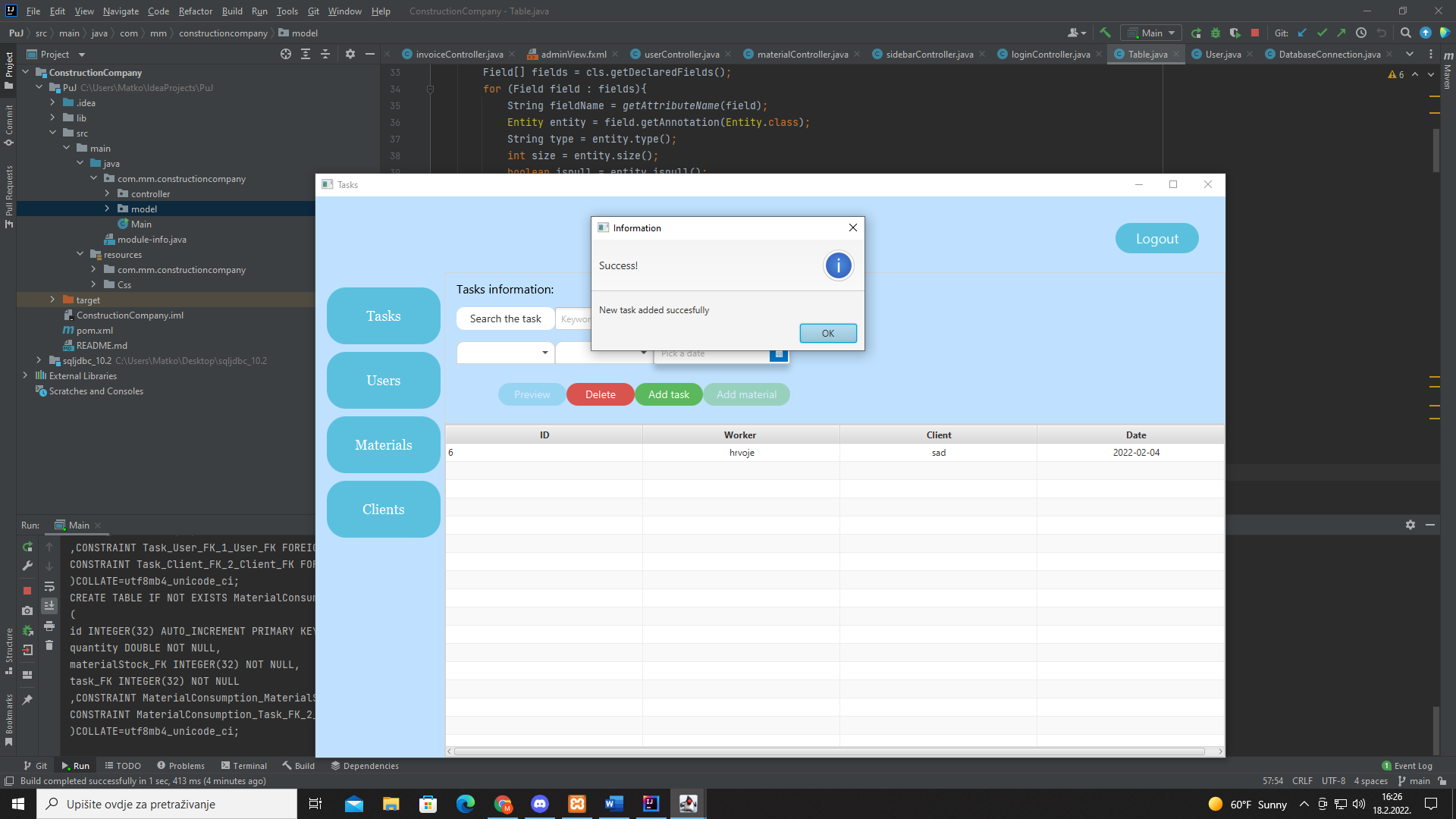
Slika 3 Informacije o klijentima



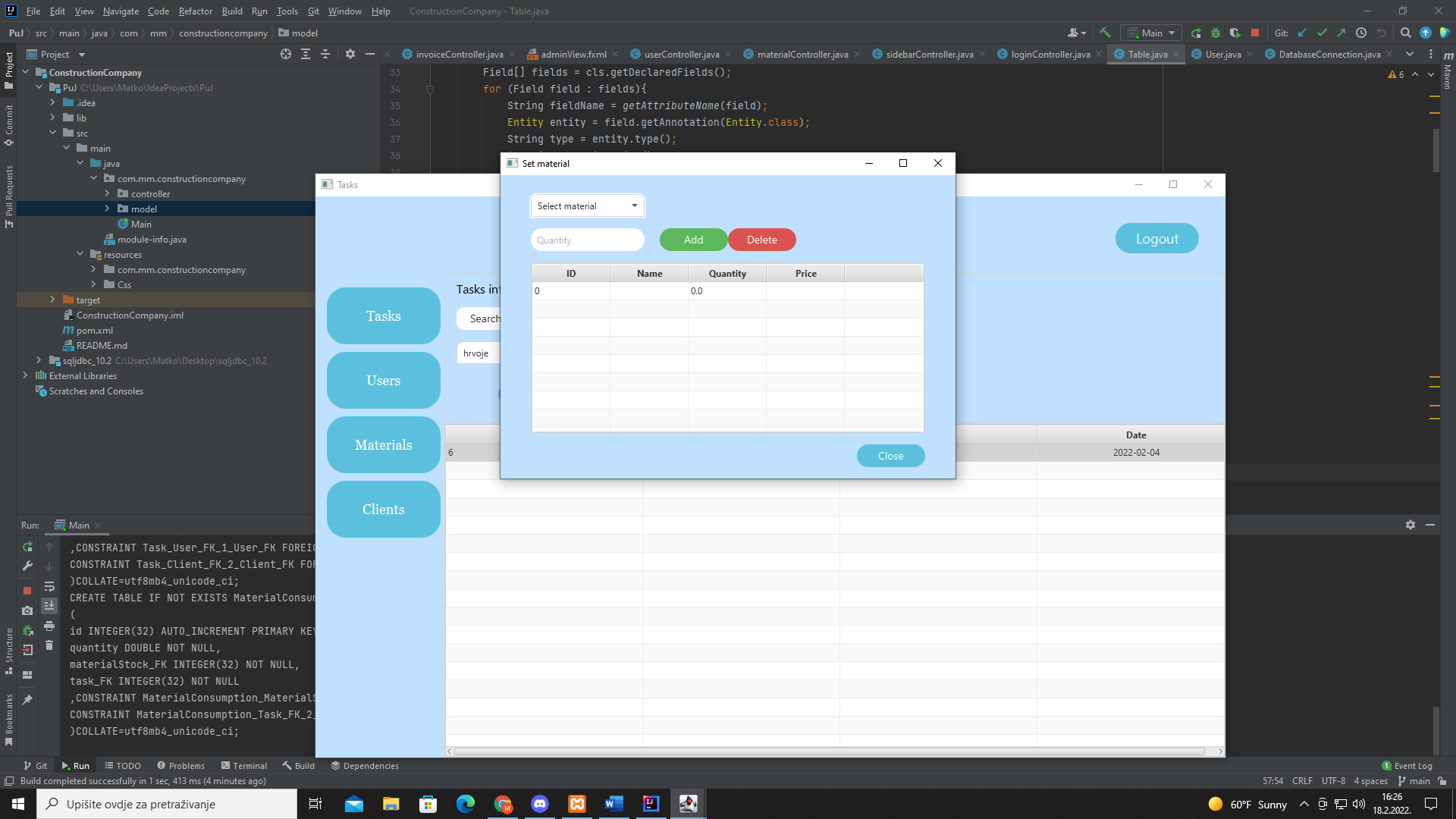
Slika 4 Informacije o materijalu



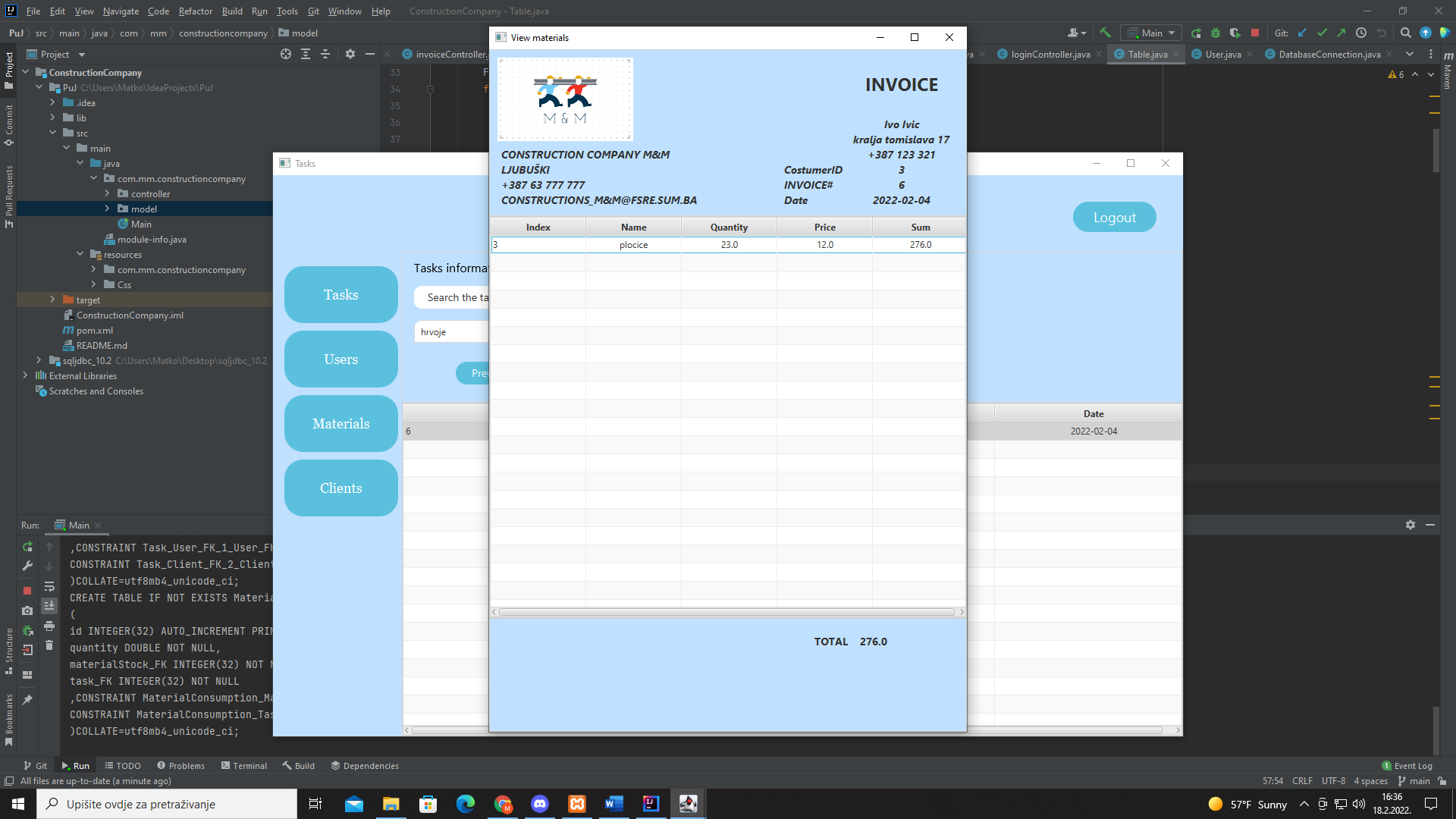
Slika 5 Dodavanje novog korisnika



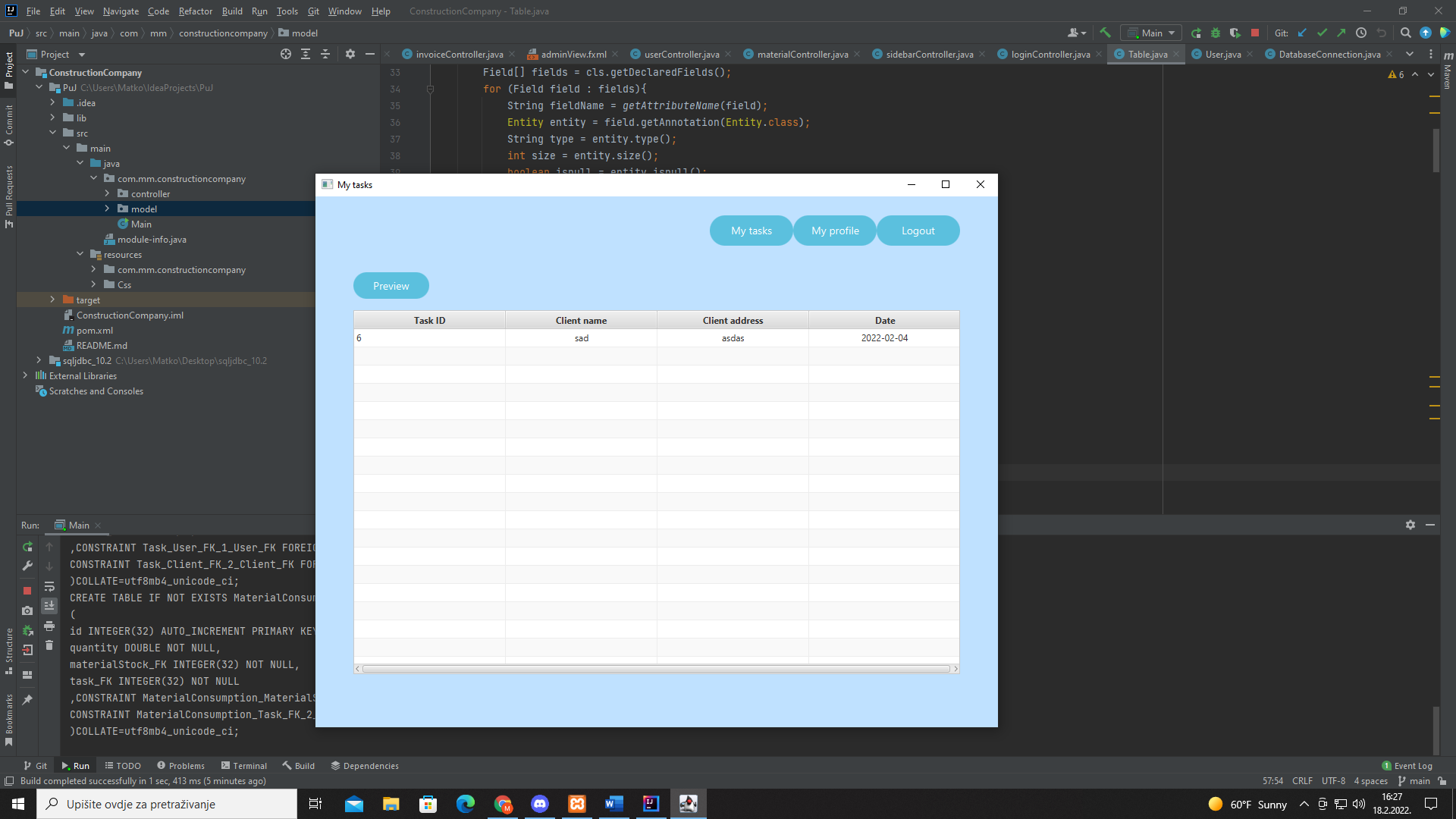
Slika 6 Dodavanje novog zadatka



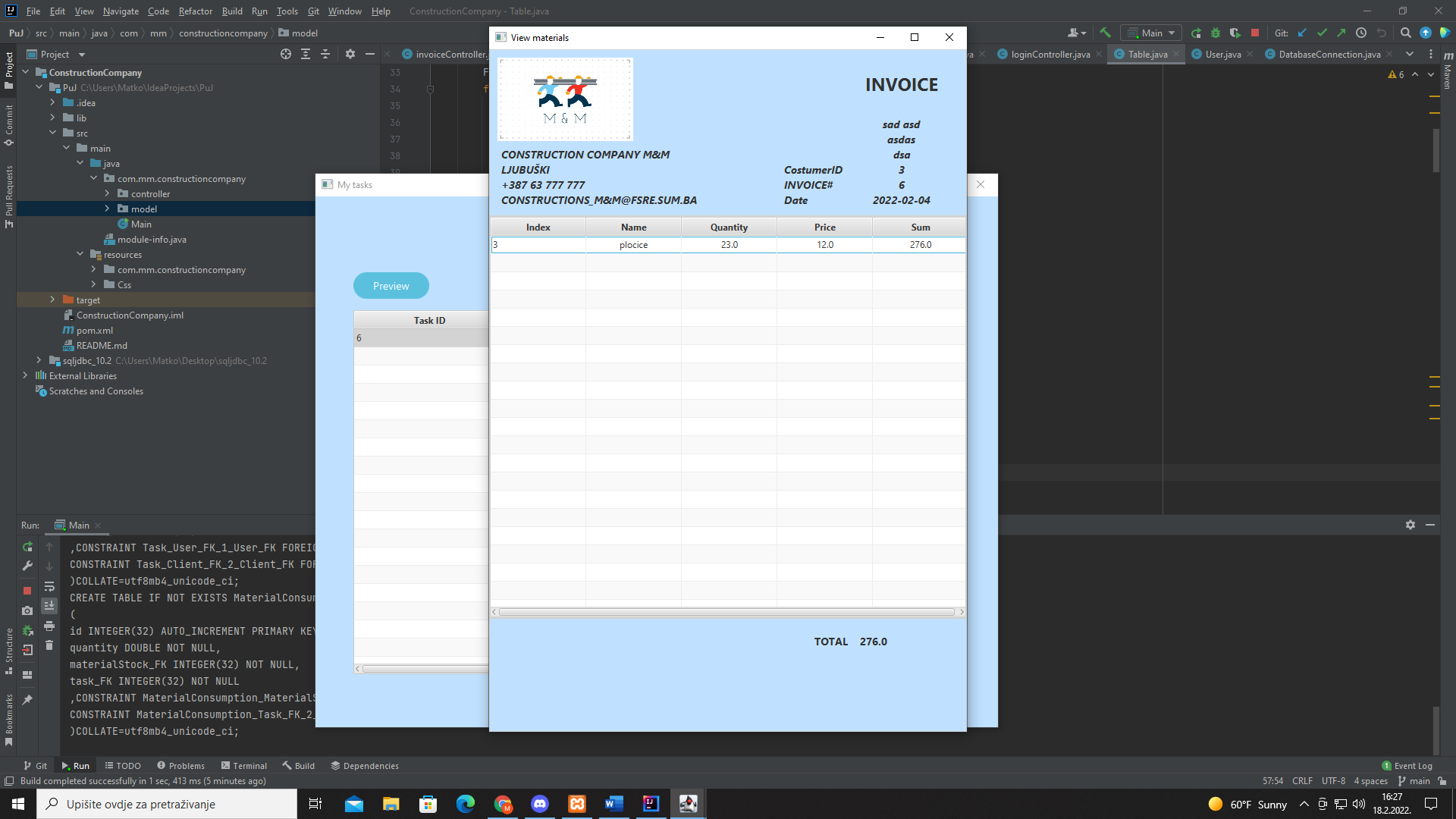
Slika 7 Dodavanje materijala na odabrani zadatak



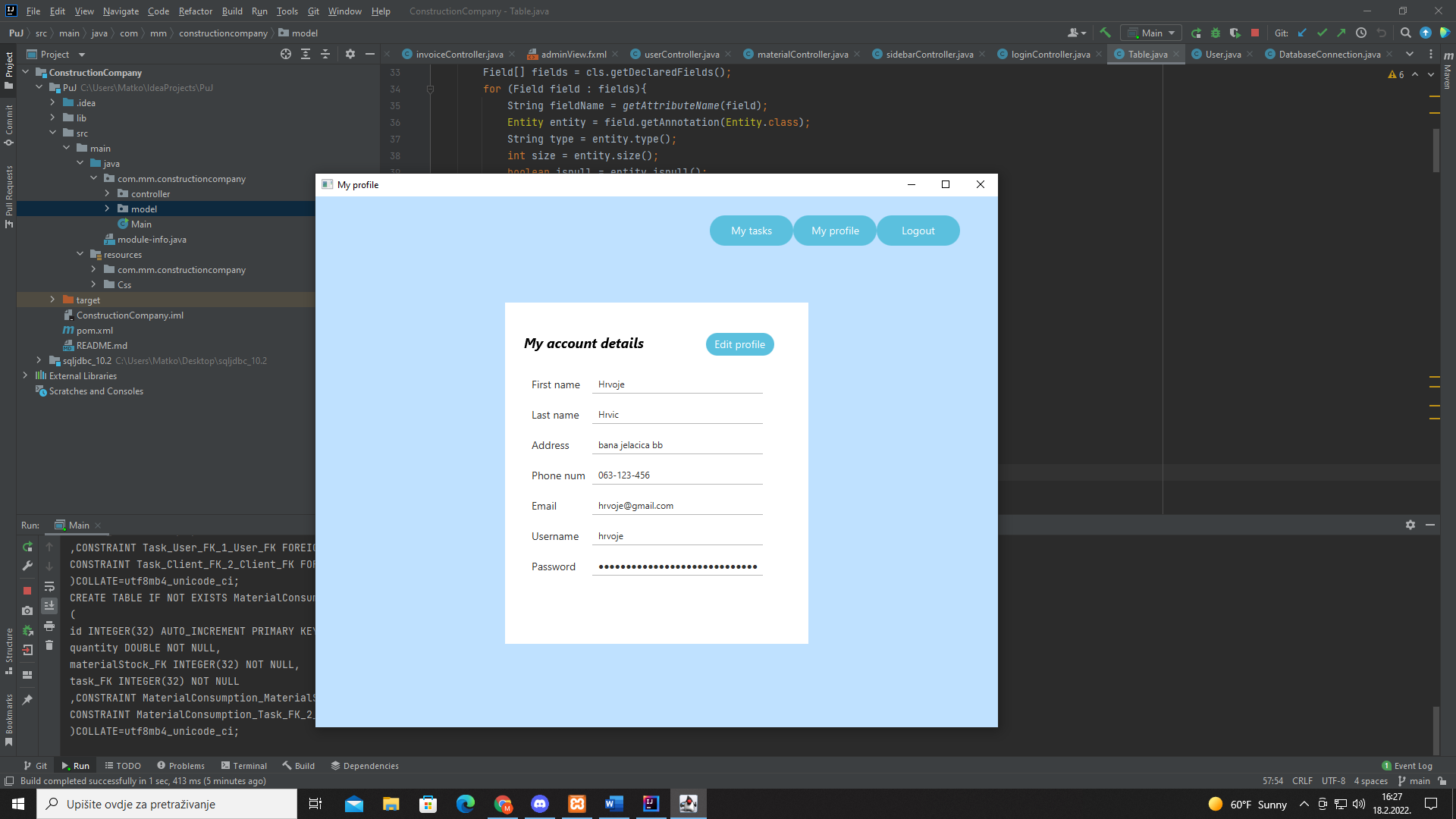
Slika 8 Pregled računa odabranog zadatka



Slika 9 Prikaz zadatka prijavljenom radniku



Slika 10 Prikaz računa prijavljenom radniku



Slika 11 Prikaz podataka profila

***2. Model programske podrške***

Drugi dio zadatka se odnosi na modeliranje programske podrške, detaljno ćete raspisati sve klase i na kraju priložiti dijagram klase koji opisuje izgled programske podrške:

* Napraviti class dijagram
* Opisati metode koje se nalaze unutar svake klase

U navedenim klasama potrebno je opisati :

1. Paketi – kratko opisati paket vaše programske podrške
2. Klase – kratko opisati klase koje koristite (ukratko za što će se klasa koristiti
   1. Atribute – kratko pisati atribute koje se nalaze unutar klase;
   2. Metode – kratko opisati metode koje se koriste za što služe i opišite njihov povratni tip;

***2.1. Paketi***

Projekt “Construction company” sastoji se od dva paketa (com.mm.constructioncompany i resources).

com.mm.constructioncompany paket sastoji se od dva podpaketa (controller i model) i klase Main koja pokreće glavni program.

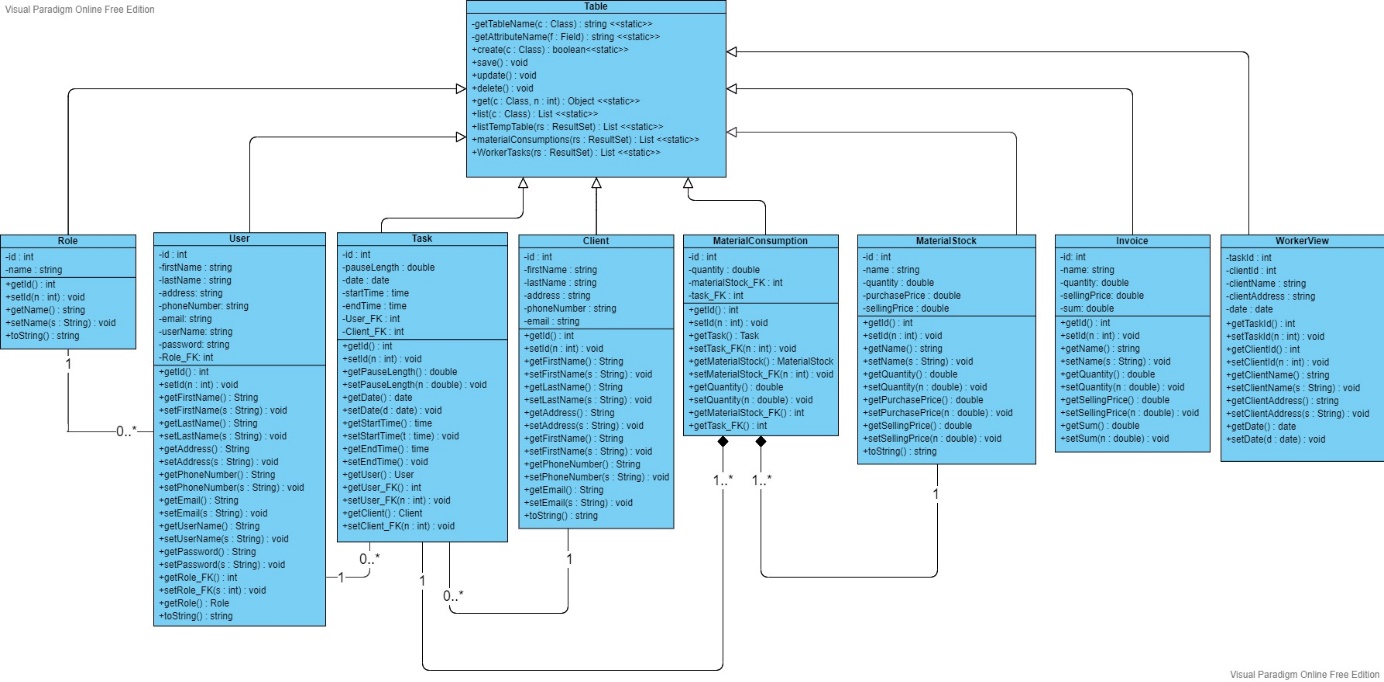
Naš controller paket sastoji se od 11 klasa tj. kontrolera. Oni upravljaju protokom podataka u objektu modela i ažuriraju pogled kad god se podaci promijene te osiguravaju samu funkcionalnost aplikacije.

Naš model sadrži strukturu baze podataka.

resources sadrži 12 FXML datoteka koji su zaslužni za frontend tj. za izgled korisnickog sučelja, te css folder s korištenim stilovima u korisničkom sučelju.

***2.2. Klase***

Slika 12 Dijagram klasa



***2.2. Klase model***

Popis klasa u paketu model:

-Client

-User

-Role

-Task

-MaterialStock

-MaterialConsumption

-Table

-WorkerView

-CryptMD5

-DatabaseConnection

***2.2.1 Klasa Client***

Klasa Client koristi se za pohranu podataka vezanih za klijente i njihove atribute.

***2.2.1.1. Atributi klase Client***

@Entity(type = "INTEGER",size = 32,primary = true)  
int id;  
  
@Entity(type = "VARCHAR",size = 50)  
String firstName;  
  
@Entity(type = "VARCHAR",size = 50)  
String lastName;  
  
@Entity(type = "VARCHAR",size = 50)  
String address;  
  
@Entity(type = "VARCHAR",size = 50)  
String phoneNumber;  
  
@Entity(type = "VARCHAR",size = 50)  
String email;

***2.2.1.2. Metode klase Client***

public int getId() - metoda koja vraća ID klijenta  
  
public void setId(int id) - metoda koja postavlja ID klijenta   
  
public String getFirstName() - metoda koja vraća ime klijenta

public void setFirstName(String firstName) - metoda koja postavlja ime klijenta  
  
public String getLastName() - metoda koja vraća prezime klijenta  
  
public void setLastName(String lastName) - metoda koja postavlja prezime klijenta  
  
public String getAdress() - metoda koja vraća adresu klijenta  
  
public void setAdress(String adress) - metoda koja postavlja adresu klijenta  
  
public String getPhoneNumber() - metoda koja vraća broj telefona klijenta  
  
public void setPhoneNumber(String phoneNumber) - metoda koja postavlja broj telefona klijenta  
  
public String getEmail() - metoda koja vraća email klijenta  
  
public void setEmail(String email) - metoda koja postavlja email klijenta  
  
public String toString() - metoda koja override-a toString metodu i vraca ime klijenta

***2.2.2 Klasa User***

Klasa User Koristi se za pohranu podataka vezanih za korisnike aplikacije i njihove atribute.

***2.2.2.1. Atributi klase User***

@Entity(type="INTEGER", size=32, primary = true)  
int id;  
  
@Entity(type="VARCHAR", size=50)  
String firstName;  
  
@Entity(type="VARCHAR", size=50)  
String lastName;  
  
@Entity(type="VARCHAR", size=50)  
String address;  
  
@Entity(type="VARCHAR", size=50)  
String phoneNumber;  
  
@Entity(type="VARCHAR", size=50)  
String email;  
  
@Entity(type="VARCHAR", size=50)  
String userName;  
  
@Entity(type = "VARCHAR",size=250)  
String password;  
  
@Entity(type = "INTEGER",size = 32)  
@ForeignKey(table = "Role",attribute = "id")  
int Role\_FK;

***2.2.2.2. Metode klase User***

public int getId() - metoda koja vraća ID korisnika  
  
public void setId(int id) - metoda koja postavlja ID korisnika  
  
public String getFirstName() - metoda koja vraća ime korisnika

public void setFirstName(String firstName) - metoda koja postavlja ime korisnika

public String getLastName() - metoda koja vraća prezime korisnika

public void setLastName(String lastName) - metoda koja postavlja prezime korisnika

public String getAddress() - metoda koja vraća adresu korisnika

public void setAddress(String address) - metoda koja postavlja adresu korisnika

public String getPhoneNumber() - metoda koja vraća broj telefona korisnika

public void setPhoneNumber(String phoneNumber) - metoda koja postavlja broj telefona korisnika

public String getEmail() - metoda koja vraća email korisnika

public void setEmail(String email) - metoda koja postavlja email korisnika

public String getUserName() - metoda koja vraća korisničko ime korisnika

public void setUserName(String userName) - metoda koja postavlja korisničko ime korisnika

public String getPassword() - metoda koja vraća lozinku korisnika

public void setPassword(String password) - metoda koja postavlja lozinku korisnika

public int getRole\_FK() - metoda koja vraća ID role koja pripada korisniku

public void setRole\_FK(int Role\_FK) - metoda koja postavlja ID role

public Role getRole() - metoda koja vraća objekt tipa Role

public String toString() - metoda koja override-a toString metodu i vraca ime korisnika

***2.2.3 Klasa Role***

Klasa Role koristi se za pohranu rola aplikacije.

***2.2.3.1. Atributi klase Role***

@Entity(type = "INTEGER", size = 32, primary = true)  
int id;  
  
@Entity(type = "VARCHAR",size = 50)  
String name;

***2.2.3.2. Metode klase Role***

public int getId() - metoda koja vraća ID role

public void setId(int id) - metoda koja postavlja ID role

public String getName() - metoda koja vraća ime role

public void setName(String name) - metoda koja postavlja ime role

public String toString() - metoda koja override-a toString metodu i vraca ime role

***2.2.4 Klasa Task***

Klasa Task koristi se za pohranu svih podataka vezanih za određeni zadatak

***2.2.4.1. Atributi klase Task***

@Entity(type="INTEGER",size = 32, primary = true)  
int id;  
  
@Entity(type="DOUBLE",size = 25)  
Double pauseLength;  
  
@Entity(type = "DATE",isnull = false)  
Date date;  
  
@Entity(type = "TIME")  
Time startTime;  
  
@Entity(type = "TIME")  
Time endTime;  
  
@Entity(type="INTEGER",size=32)  
@ForeignKey(table ="User", attribute = "id")  
int User\_FK;  
  
@Entity(type = "INTEGER",size = 32)  
@ForeignKey(table = "Client",attribute = "id")  
int Client\_FK;

***2.2.4.2. Metode klase Task***

public int getId() - metoda koja vraća ID zadatka

public void setId(int id) - metoda koja postavlja ID zadatka

public Double getPauseLength() - metoda koja vraća duzinu pauze na zadatku

public void setPauseLength(Double pauseLength) - metoda koja postavlja duzinu pauze na zadatku

public Date getDate() - metoda koja vraća datum zadatka

public void setDate(Date date) - metoda koja postavlja datum zadatka

public Time getStartTime() - metoda koja vraća početno vrijeme zadatka

public void setStartTime(Time startTime) - metoda koja postavlja

početno vrijeme zadatka

public Time getEndTime() - metoda koja vraća vrijeme završetka zadatka

public void setEndTime(Time endTime) - metoda koja postavlja vrijeme završetka zadatka

public User getUser() - metoda koja vraca objekt korisnika koji radi na zadatku

public int getUser\_FK() - metoda koja vraca ID korisnika koji radi na zadatku

public void setUser\_FK(int user\_FK) - metoda koja postavlja ID korisnika koji radi na zadatku

public Client getClient() - metoda koja vraca objekt klijenta kod kojeg se zadatak radi  
  
public void setClient\_FK(int client\_FK) - metoda koja postavlja ID klijenta kod kojeg se zadatak radi

***2.2.5 Klasa MaterialStock***

Klasa MaterialStock služi nam za pohranjivanje podataka vezanih za materijal na zalihi.

***2.2.5.1. Atributi klase MaterialStock***

@Entity(type="INTEGER", size=32, primary = true)  
int id;  
  
@Entity(type="VARCHAR", size=50)  
String name;  
  
@Entity(type="DOUBLE",isnull = false)  
Double quantity;  
  
@Entity(type="DOUBLE", size=25)  
Double purchasePrice;  
  
@Entity(type="DOUBLE", size=25)  
Double sellingPrice;

***2.2.5.2. Metode klase MaterialStock***

public int getId() - metoda koja vraca ID materijala

public void setId(int id) - metoda koja postavlja ID materijala

public String getName() - metoda koja vraca ime materijala

public void setName(String name) - metoda koja postavlja ime materijala

public Double getQuantity() - metoda koja vraca kolicinu materijala na zalihi

public void setQuantity(Double quantity) - metoda koja postavlja kolicinu materijala na zalihi

public Double getPurchasePrice() - metoda koja vraca kupovnu cijenu materijala

public void setPurchasePrice(Double purchasePrice) - metoda koja postavlja kupovnu cijenu materijala

public Double getSellingPrice() - metoda koja vraca prodajnu cijenu materijala

public void setSellingPrice(Double sellingPrice) - metoda koja postavlja prodajnu cijenu materijala

public String toString() - metoda koja override-a toString metodu i vraca naziv materijala

***2.2.6 Klasa MaterialConsumption***

Klasa MaterialConsumption služi nam za pohranu podataka o potrošnji materijala.

***2.2.6.1. Atributi klase MaterialConsumption***

@Entity(type = "INTEGER",size=32,primary = true)  
int id;  
  
@Entity(type = "DOUBLE",size = 25)  
Double quantity;  
  
@Entity(type = "INTEGER",size=32)  
@ForeignKey(table = "MaterialStock",attribute = "id")  
int materialStock\_FK;  
  
@Entity(type = "INTEGER",size = 32)  
@ForeignKey(table = "Task",attribute = "id")  
int task\_FK;

***2.2.6.2. Metode klase MaterialConsumption***

public int getId() - metoda koja vraca ID potrošenog materijala

public void setId(int id) - metoda koja postavlja ID potrosenog materijala

public Task getTask() - metoda koja vraca Task objekt, tj. zadatak na kojem se materijal utrošio

public void setTask\_FK(int Task\_FK) - metoda koja postavlja ID zadatka na kojem se materijal utrošio

public MaterialStock getMaterialStock() - metoda koja vraca objekt MaterialStock, tj. materijal koji se utrošio

public void setMaterialStock\_FK(int materialStock\_FK) - metoda koja postavlja ID materijala koji se utrošio

public Double getQuantity() - metoda koja vraca količinu materijala koji se utrošio

public void setQuantity(Double quantity) - metoda koja postavlja količinu materijala koji se utrošio

public int getMaterialStock\_FK() - metoda koja vraca ID materijala koji se utrošio

public int getTask\_FK() - metoda koja vraca ID zadatka na kojem se materijal utrošio

***2.2.7 Klasa Table***

Klasa Table služi nam za crud operacije nad bazom i ostale operacije s bazom podataka.

***2.2.7.1. Atributi klase Table***

-

***2.2.7.2. Metode klase Table***

private static String getTableName(Class cls) - metoda koja vraća ime tablice

private static String getAttributeName(Field field) - metoda koja vraća ime atributa

public static boolean create(Class cls) - metoda koja kreira tablicu u bazi podataka

public void save() - metoda koja sprema redak u tablicu podataka

public void update() - metoda koja ažurira redak u tablici podataka

public void delete() - metoda koja briše redak iz tablice podataka

public static Object get(Class cls, int id) - metoda koja vraća određeni objekt iz neke tablice baze podataka

public static List<?> list(Class cls) - metoda koja vraća listu svih redaka u nekoj tablici

public static List<Invoice> Invoice(ResultSet rs) - metoda koja vraća listu svih redaka u tablici WorkerView

public static List<MaterialConsumption> materialConsumptions(ResultSet rs) - metoda koja vraća listu svih redaka u tablici MaterialConsumption

public static List<WorkerView> WorkerTasks(ResultSet rs) - metoda koja vraća listu svih redaka u tablici WorkerView.

***2.2.8 Klasa WorkerView***

Klasa WorkerView služi za pohranu podataka vezanih za zadakte koje pojedini radnik dobije.

***2.2.8.1. Atributi klase WorkerView***

private int taskId;

private int clientId;

private String clientName;

private String clientAddress;

private String date;

***2.2.8.2. Metode klase WorkerView***

public int getTaskId() - metoda koja vraća ID

public void setTaskId(int taskId) - metoda koja postavlja ID

public int getClientId() - metoda koja vraća ID klijenta

public void setClientId(int clientId) - metoda koja postavlja ID klijenta

public String getClientName() - metoda koja vraća ime klijenta

public void setClientName(String clientName) - metoda koja postavlja ime klijenta

public String getClientAddress() - metoda koja vraća adresu klijenta

public void setClientAddress(String clientAddress) - metoda koja postavlja adresu klijenta

public String getDate() - metoda koja vraća datum

public void setDate(String date) - metoda koja postavlja datum

***2.2.9 Klasa CryptMD5***

Klasa CryptMD5 koristi se za enkripciju stringova.

***2.2.9.1. Atributi klase CryptMD5***

private static MessageDigest md

***2.2.9.2. Metode klase CryptMD5***

public static String cryptWithMD5(String pass)-metoda koja enktriptira prosljeđeni string i vraća enkriptirani string.

***2.2.10 Klasa DatabaseConnection***

Klasa koja se koristi za povezivanje na bazu.

***2.2.10.1. Atributi klase DatabaseConnection***

private static String hostname = "localhost";

private static String user = "root";

private static String password = "";

private static String database = "constructioncompanydb";  
  
public static Connection CONNECTION = null;

***2.2.10.2. Metode klase DatabaseConnection***

***-***

***2.3. Klase controller***

Popis klasa u paketu controller:

-ClientController

-UserController

-TaskController

-MaterialStockController

-AddMaterialController

-InvoiceController

-LoginController

-SidebarController

-WorkerViewController

-WorkerTaskController

-WorkerProfileController

***2.3.1 Klasa ClientController***

Klasa ClientController koristi se za crud operacije nad modelom Client, tj. za prikaz svih klijenata, ažuriranje odabranog klijenta, brisanje odabranog klijenta i kreiranje novog klijenta.

***2.3.1.1. Atributi klase ClientController***

private TableView clientsTbl;  
  
private TableColumn IDTblCol;  
  
private TableColumn nameTblCol;  
  
private TableColumn surnameTblCol;  
  
private TableColumn addressTblCol;  
  
private TableColumn mobilePhoneTblCol;  
  
private TableColumn e\_mailTblCol;  
  
private TextField nameTxt;

private TextField surnameTxt;  
  
private TextField addressTxt;  
  
private TextField mobilePhoneTxt;  
  
private TextField e\_mailTxt;  
  
private Button btnAdd;  
  
private TextField searchTxt;  
  
Client selectedClient=null;

***2.3.1.2. Metode klase Client***

public void initialize(URL url, ResourceBundle resourceBundle) - metoda koja inicijalizira kontroler

private void fillClients() - metoda koja popunjava tablicu klijenata

public void onDelete(ActionEvent event) - metoda koja na klik briše označeni redak iz tablice

public void onSave(ActionEvent event) - metoda koja sprema upisanog klijenta u bazu podataka

public void selectClient(MouseEvent evt) - metoda koja klikom odabranog pohranjuje u gore navedenu varijablu selectedClient

protected void removeSelection() - metoda koja briše unesena polja i poništava odabir

private boolean searchFindsClient(Client client, String searchText) - metoda koja pronalazi određenu riječ u tablici klijenata

private ObservableList<Client> filterList(List<Client> list, String searchText) - metoda koja vraća filtriranu listu klijenata

public void remove(MouseEvent event) - metoda koja na klik van tabele poništava odabir

***2.3.2 Klasa UserController***

Klasa UserController koristi se za crud operacije nad modelom User, tj. za prikaz svih korisnika, ažuriranje odabranog korisnika, brisanje odabranog korisnika i kreiranje novog korisnika.

***2.3.2.1. Atributi klase UserController***

private TableView usersTbl;

private TableColumn IDTblCol;  
  
private TableColumn nameTblCol;

private TableColumn surnameTblCol;

private TableColumn addressTblCol;

private TableColumn mobilePhoneTblCol;

private TableColumn e\_mailTblCol;

private TableColumn userNameTblCol;

private TableColumn passwordTblCol;

private TableColumn roleTblCol;  
  
private TextField nameTxt;

private TextField surnameTxt;

private TextField addressTxt;

private TextField mobilePhoneTxt;

private TextField e\_mailTxt;

private ComboBox roleTxt;

private TextField userNameTxt;

private TextField passwordTxt;

private Button btnSave;

private TextField searchTxt;  
  
User selectedUser=null;

***2.3.2.2. Metode klase UserController***

private void fillUsers() - metoda koja popunjava tablicu korisnika

public void onSave (ActionEvent evt) - metoda koja sprema upisanog korisnika u bazu podataka

public void selectUser(MouseEvent evt) - metoda koja klikom odabranog korisnika pohranjuje u gore navedenu varijablu selectedKorisnik

protected void removeSelection() - metoda koja briše unesena polja i poništava odabir

public void onDelete() - metoda koja na klik briše označeni redak iz tablice

private boolean searchFindsUser(User user, String searchText) - metoda koja pronalazi određenu riječ u tablici korisnika

private ObservableList<User> filterList(List<User> list, String searchText) - metoda koja vraća filtriranu listu korisnika

public void remove(MouseEvent event) - metoda koja na klik van tabele poništava odabir

***2.3.3 Klasa TaskController***

Klasa TaskController koristi se za crud operacije nad modelom Task, tj. za prikaz svih radnih naloga, ažuriranje odabranog radnog naloga, brisanje odabranog radnog naloga i kreiranje novog radnog naloga.

***2.3.3.1. Atributi klase TaskController***

private TextField searchTxt;

private TableView tasksTbl;  
  
private TableColumn IDTblCol;

private TableColumn workerTblCol;

private TableColumn dateTblCol;

private TableColumn clientTblCol;

private DatePicker datePick;

private Button btnSave;

private Button btnPreview;

private Button btnAddMaterial;

private ComboBox selectWorker;

private ComboBox selectClient;  
  
public Task selectedTask = null;

***2.3.3.2. Metode klase TaskController***

public void initialize(URL url, ResourceBundle resourceBundle) - metoda koja inicijalizira kontroler

private void fillTasks() - metoda koja popunjava tablicu radnih naloga

public void onDelete(ActionEvent event) - metoda koja na klik briše označeni redak iz tablice

public void onSave(ActionEvent event) - metoda koja sprema upisani radni nalog u bazu podataka

public void selectTask(MouseEvent evt) - metoda koja klikom odabranog pohranjuje u gore navedenu varijablu selectedTask

protected void removeSelection() - metoda koja briše unesena polja i poništava odabir

private boolean searchFindsTask(Task task, String searchText) - metoda koja pronalazi određenu riječ u tablici radnih naloga

private ObservableList<Tasl> filterList(List<Task> list, String searchText) - metoda koja vraća filtriranu listu radnih naloga

public void remove(MouseEvent event) - metoda koja na klik van tabele poništava odabir

public void onPreview() - metoda koja otvara novi prozor koji prikazuje račun za odabrani radni nalog

public void onAddMaterial() - metoda koja otvara novi prozor za dodavanje materijala na odabrani radni nalog

***2.3.4 Klasa MaterialController***

Klasa MaterialController koristi se za crud operacije nad modelom MaterialStock, tj. za prikaz svih materijala, ažuriranje odabranog materijala, brisanje odabranog materijala i kreiranje novog materijala.

***2.3.4.1. Atributi klase MaterialController***

private TableView materialsTbl;  
  
private TableColumn IDTblCol;  
  
private TableColumn nameTblCol;  
  
private TableColumn quantityTblCol;  
  
private TableColumn purchasePriceTblCol;  
  
private TableColumn sellingPriceTblCol;  
  
private TextField searchTxt;  
  
private TextField nameTxt;  
  
private TextField quantityTxt;  
  
private TextField purchasePriceTxt;  
  
private TextField sellingPriceTxt;  
  
private Button btnAdd;  
  
MaterialStock selectedMaterial=null;

***2.3.4.2. Metode klase MaterialController***

public void initialize(URL url, ResourceBundle resourceBundle) - metoda koja inicijalizira kontroler

private void fillMaterials() - metoda koja popunjava tablicu materijala

public void onDelete(ActionEvent event) - metoda koja na klik briše označeni redak iz tablice

public void onSave(ActionEvent event) - metoda koja sprema upisani materijal u bazu podataka

public void selectMaterial(MouseEvent evt) - metoda koja klikom odabranog pohranjuje u gore navedenu varijablu selectedMaterial

protected void removeSelection() - metoda koja briše unesena polja i poništava odabir

private boolean searchFindsMaterial(MaterialStock material, String searchText) - metoda koja pronalazi određenu riječ u tablici materijala

private ObservableList<MaterialStock> filterList(List<MaterialStock> list, String searchText) - metoda koja vraća filtriranu listu materijala

public void remove(MouseEvent event) - metoda koja na klik van tabele poništava odabir

public static boolean isNumeric(String strNum) - metoda koja provjerava da li je proslijeđeni string broj

***2.3.5 Klasa AddMaterialController***

Klasa AddMaterialController koristi se za crud operacije nad modelom MaterialConsumption, tj. za prikaz svih utrošenih materijala od strane nekog radnog naloga, ažuriranje odabranog materijala, brisanje odabranog materijala i kreiranje novog utroška materijala.

***2.3.5.1. Atributi klase AddMaterialController***

private TextField quantityTxt;

private ComboBox selectMaterial;

private TableView materialsTbl;  
  
private TableColumn nameTblCol;  
  
private TableColumn priceTblCol;  
  
private Button btnAdd;  
  
private Button btnClose;  
  
private TableColumn quantityTblCol;  
  
private TableColumn idTblCol;  
  
private MaterialConsumption selectedMaterial=null;  
  
private Task selectedTask=null;

***2.3.5.2. Metode klase AddMaterialController***

Public void closeWindow(ActionEvent event) - metoda koja na klik zartvara prozor

public void onDelete(ActionEvent event) - metoda koja na klik briše označeni redak iz tablice

public void onSave(ActionEvent event) - metoda koja sprema upisani materijal za određeni radni nalog u bazu podataka

public void initialize(URL url, ResourceBundle resourceBundle) - metoda koja inicijalizira kontroler

private void fillMaterialConsumption() - metoda koja popunjava tablicu utrošenih materijala u radnom nalogu

public void selectMaterial(MouseEvent evt) - metoda koja klikom odabranog materijala pohranjuje u gore navedenu varijablu selectedMaterial

public void setSelectedTask(Task t) - metoda koja postavlja selectedTask varijablu

protected void removeSelection() - metoda koja briše unesena polja i poništava odabir

public ResultSet makeSelectQuery() - metoda koja iz baze dohvaća podatke o utrošenim materijalima za pojedini radni nalog

void remove(MouseEvent event) - metoda koja na klik van tabele poništava odabir

***2.3.6 Klasa SidebarController***

Klasa SidebarController koristi se za navigaciju kroz ponuđeni sidebar menu.

***2.3.6.1. Atributi klase SidebarController***

-

***2.3.6.2. Metode klase SidebarController***

public void btnClientsClicked(ActionEvent event) - metoda koja na klik prebacuje scenu na prozor s klijentima

public void btnMaterialsClicked(ActionEvent event) - metoda koja na klik prebacuje scenu na prozor s mateijalima

public void btnTasksClicked(ActionEvent event) - metoda koja na klik prebacuje scenu na prozor s radnim nalozima

public void btnUsersClicked(ActionEvent event) - metoda koja na klik prebacuje scenu na prozor s korisnicima

public void btnLogoutClicked(ActionEvent event) - metoda koja na klik odjavljuje korisnika iz sustava i prebacuje scenu na prijavu korisnika u sustav

public void initialize(URL url, ResourceBundle resourceBundle) - metoda koja inicijalizira kontroler

***2.3.7 Klasa InvoiceController***

Klasa InvoiceController koristi se za prikaz računa odabranog radnog naloga i popisa svih materijala tog radnog naloga.

***2.3.7.1. Atributi klase ClientController***

private TableView materialsTbl;  
  
private TableColumn idTblCol;  
  
private TableColumn nameTblCol;  
  
private TableColumn quantityTblCol;  
  
private TableColumn priceTblCol;  
  
private TableColumn sumTblCol;  
  
private Label clientAdressLbl;  
  
private Label clientIdLbl;  
  
private Label clientNameLbl;  
  
private Label clientPhoneLbl;  
  
private Label dateLbl;  
  
private Label invoiceLbl;  
  
private Label totalSumLbl;  
  
private static Task selectedTask=null;

***2.3.7.2. Metode klase InvoiceController***

public void initialize(URL url, ResourceBundle resourceBundle) - metoda koja inicijalizira kontroler

protected void fillInvoice() - metoda koja služi za popunjavanje računa podacima

public void setSelectedTask(Task t) - metoda koja postavlja selectedTask varijablu

public ResultSet makeQuery() - metoda koja pravi upit nad bazom za podatke koje ćemo postaviti na račun

public void initController() - metoda koja ponovno popunjava tablicu račun, te ostale podatke kao što su ime klijenta, adresa klijenta itd.

public String getSum() - metoda koja vraća zbroj svih artikala s računa

***2.3.8 Klasa LoginController***

Klasa LoginController koristi se za prijavu korisnika u aplikaciju.

***2.3.8.1. Atributi klase LoginController***

private PasswordField tfpassword;  
  
private TextField tfusername;  
  
private TextField signupAddress;  
  
private TextField signupEmail;  
  
private TextField signupName;  
  
private PasswordField signupPassword;  
  
private TextField signupPhoneNumber;  
  
private TextField signupSurname;  
  
private TextField signupUsername;  
  
protected static User loggedUser;

***2.3.8.2. Metode klase LoginController***

public void btnloginclicked(ActionEvent event) - metoda koja na klik pokušava prijaviti korisnika u sustav

public void backToLogin() - metoda koja vraća korisnika na početno login sučelje

public void btnSignupClicked() - metoda koja poziva Signup formu za registraciju novih korisnika

public void signUp() - metoda koja pokušava registrirati novog korisnika u sustav

public void initialize(URL url, ResourceBundle resourceBundle) - metoda koja inicijalizara kontroler

***2.3.9 Klasa WorkerViewController***

Klasa WorkerViewController koristi se za navigaciju kroz ponuđeni sidebar menu kod logiranog radnika

***2.3.9.1. Atributi klase WorkerViewController***

-

***2.3.9.2. Metode klase WorkerViewController***

public void logout(ActionEvent event) - metoda koja odjavljuje korisnika i na klik prebacuje scenu na prozor za prijavu u sustav

public void myProfile(ActionEvent event) - metoda koja na klik prebacuje scenu na profil radnika

public void myTasks(ActionEvent event) - metoda koja na klik prebacuje scenu na tablicu radnih naloga radnika

public void initialize(URL url, ResourceBundle resourceBundle) - metoda koja inicijalizira kontroler

***2.3.10 Klasa WorkerTaskController***

Klasa WorkerTaskController koristi se za navigaciju kroz ponuđeni sidebar menu kod logiranog radnika

***2.3.10.1. Atributi klase WorkerTaskController***

private TableColumn clientAddressTblCol;

private TableColumn clientNameTblCol;

private TableColumn idTblCol;

private TableColumn taskDateTblCol;

private TableView tasksTbl;

private WorkerView selectedTask=null;

***2.3.10.2. Metode klase WorkerTaskController***

public void initialize(URL url, ResourceBundle resourceBundle) - metoda koja inicijalizira kontroler

private void fillWorkerTasks() - metoda koja popunjava tablicu radnih naloga koja je vidljiva radniku

public ResultSet makeQueryforTasks() - metoda koja dohvaća podatke o radnim nalozima iz baze podataka koji su dostupni za pregled radniku koji je na njima radio

public void selectTask(MouseEvent evt) - metoda koja klikom odabrani radni nalog pohranjuje u gore navedenu varijablu selectedTask

void taskDetails(ActionEvent event) - metoda koja na klik otvara novi prozor u kojem se nalaze detalji o odabranom radnom nalogu

void remove(MouseEvent event) - metoda koja na klik van tabele poništava odabir

***2.3.11 Klasa WorkerProfileController***

Klasa WorkerProfileController koristi se za ažuriranje korisnikovog profila

***2.3.11.1. Atributi klase WorkerProfileController***

private TextField addressTxt;  
  
private TextField emailTxt;

private TextField firstNameTxt;  
  
private TextField lastNameTxt;

private PasswordField passwordTxt;  
  
private TextField phoneNumTxt;  
  
private TextField userNameTxt;

***2.3.11.2. Metode klase WorkerProfileController***

public void initialize(URL url, ResourceBundle resourceBundle) - metoda inicijlizira kontroler

void editProfile(MouseEvent event) - metoda koja na klik ažurira korisnikov profil