„Куриерска фирма“

1. Условие на Проекта

Text

Description automatically generated

2.Анализ на Проекта

2.1)Функционални изисквания

2.2)Структура на Проекта

2.3)Дефиниция на модулите на системата

3.Проектиране на системата

3.1)Проектиране на отделните модули

Софтуерът се състои от следните модули , База данни за която сме използвали MSSQL , която се състои от 6 таблици , които покриват всички нужни функционалности. и е добавен MSSQL JDBC Driver за връзка на базата данни със самият софтуер и бизнес логиката.

За GUI сме използвали JavaFX заедно с SceneBuilder за построяването на интерфеиса и манипулирането на fxml файловете.

Класовете са разделени на : „entities“ - класове модели за различни таблици от базата данни за по-четлив код. Класовете контролери които оперират бизнес логиката и различната функционалност за дадената сцена, главен клас “HelloAplication” за стартиране на приложението и класове “SQLClass” „Utils” които съдържаt всички методи които се използват за общуване с базата данни.

3.2)UML – Sequence, Use Case , Er Diagram, Class Diagram

Diagram

Description automatically generatedDiagram

Description automatically generated

Diagram

Description automatically generated

Diagram

Description automatically generated

4.Реализация на системата

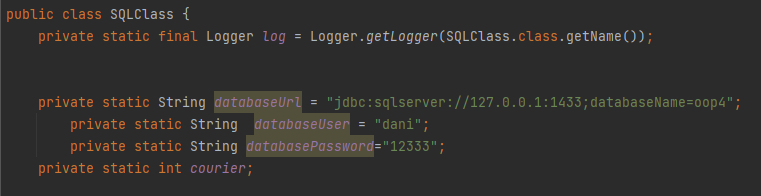
4.1)Реализацията на базата данни

Базата данни е реализирана с MSSQL, състои се от 6 таблици “poruchki” - съдържа информацията за всички поръчки , “admin” – админ който управлява системата , ”courier” – куриер който създава клиенти, поръчки и следи техния статус , ”client”- съдържа нужната информация за работата на програмата, ”status” – съдържа нужната информация за всяка проъчка , ”vidPoruchki” – съдържа нужната информация за всеки вид поръчка.

4.2)Реализация на слоя за работа с базата данни

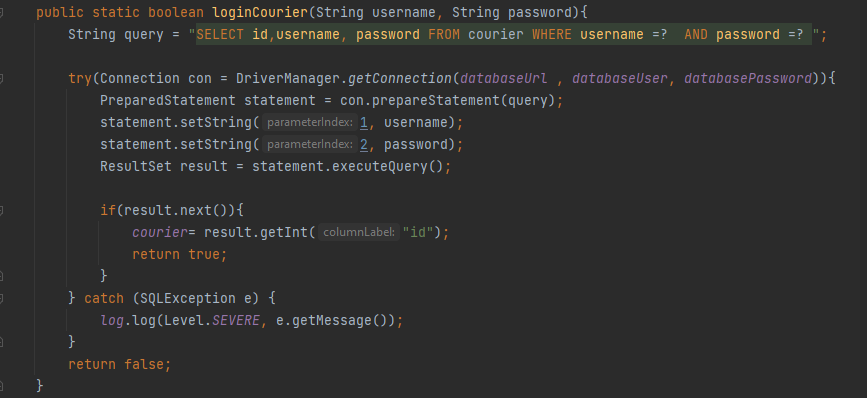
За реализация на слоя за работа с базата данни се използва файла “SQLClass” и „Utils”.

За начало са декларирани глобални констати за връзка с базата данни.

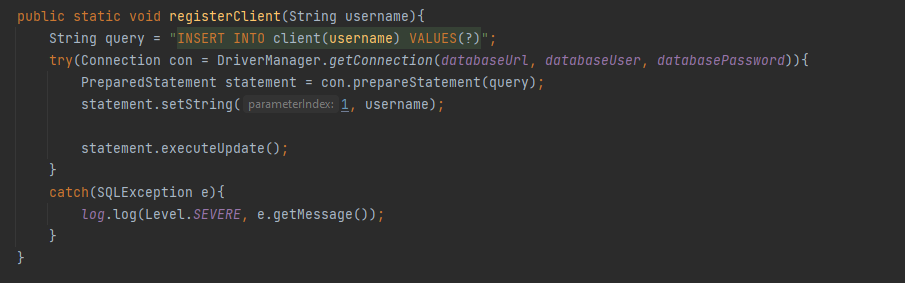


След което са подредени главните функции за работа с базата, като първо е функцията за логване в базата данни.

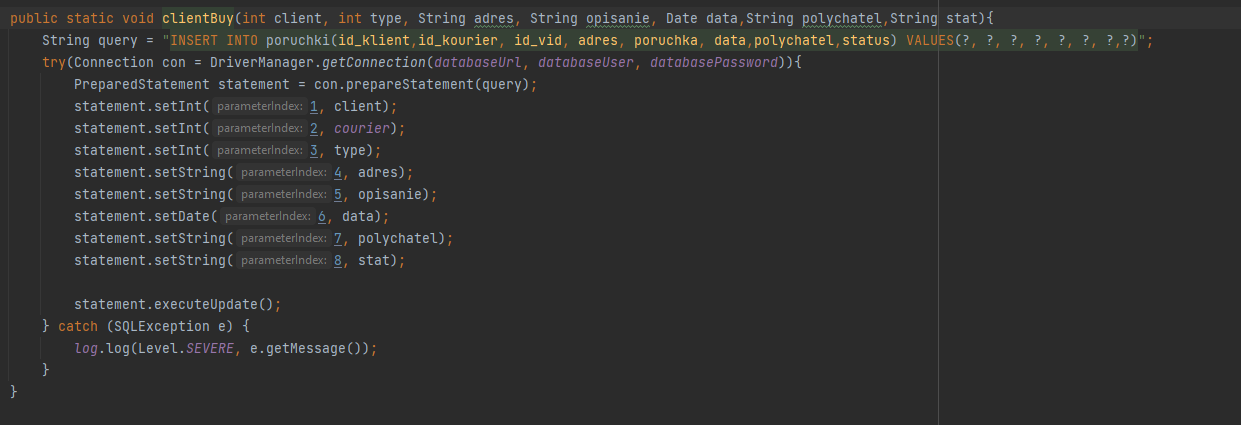
Тя се извиква в “AdminloginController” и ‘‘CourierloginController“ в случаят (CourierloginController). Ако информацията не съвпада с Database таблицата дава грешка.



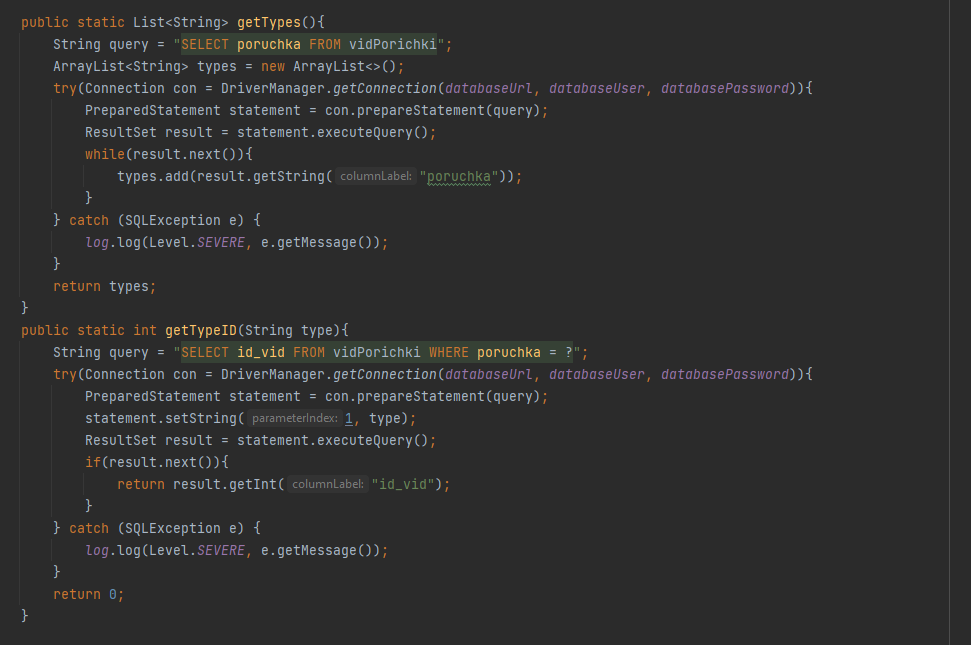
Куриерът регистрира клиент с INSERT заявка.



Създава се обек от типа Porichka, който ще видите в следващата секция, при което чрез get методите се попълва “prepared statement” и се изпълнява INSERT заявка.



Mетод чрез който избираме вид на пратката. От “getTypeID” вземаме ID-то, а от “getTypes”” вземаме името.



Метод за избиране на клиент чрез combobox.

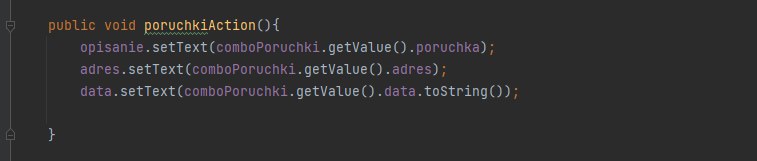


След като сме направили поръчката трябва да й променим статуса в Check orders.

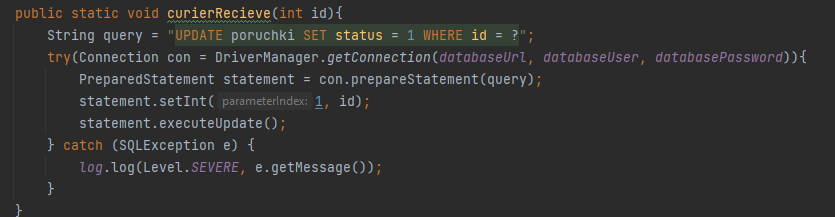
От “getPoruchki“ взимаме информацията.



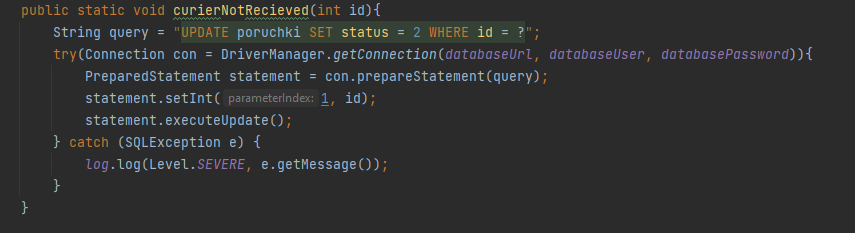
От “poruchkiAction“ взимаме данни за поръчката.



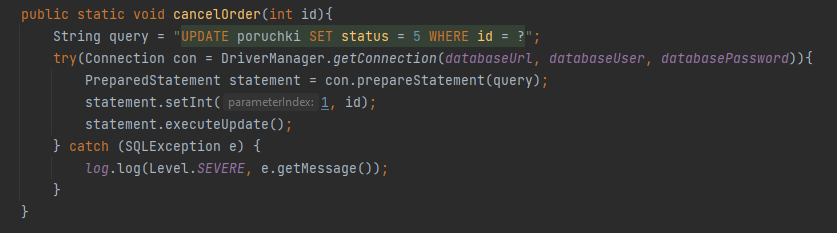
Чрез “curierReceive“ променяме статуса на получена.



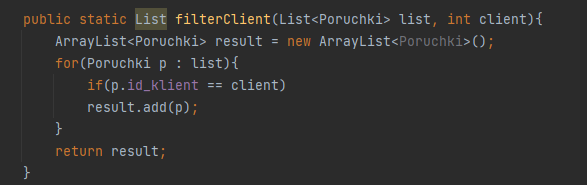
Чрез “curierNotReceived“ променяме статуса на неполучена.



Чрез “cancelOrder“ променяме статуса на отказана.

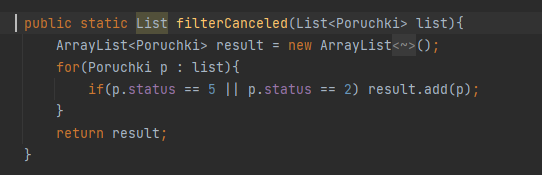


Чрез “filterClient“ проверяваме статуса на клиента.



Чрез “ListView listCanceled“ показваме известия за неприетите и отказаните поръчки.

Информацията се взима от “filterCanceled“.



От ”ListView precenka” се определя дали куриерът ще направи поръчка.

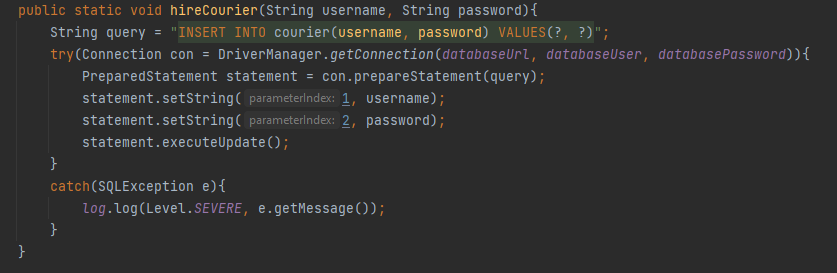


**Следващите** методи отговарят за функционалността на админина.

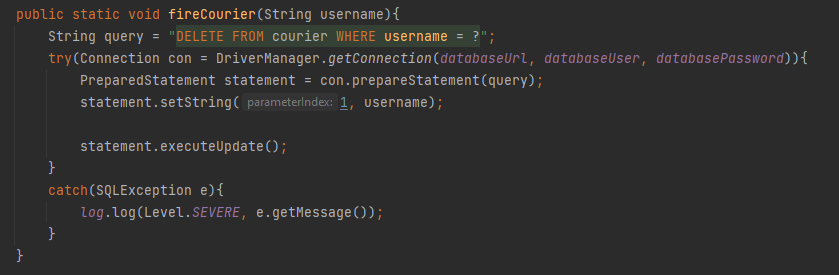
Логваме се в базата данни с админ с “AdminloginController” . Ако информацията не съвпада с Database таблицата дава грешка.



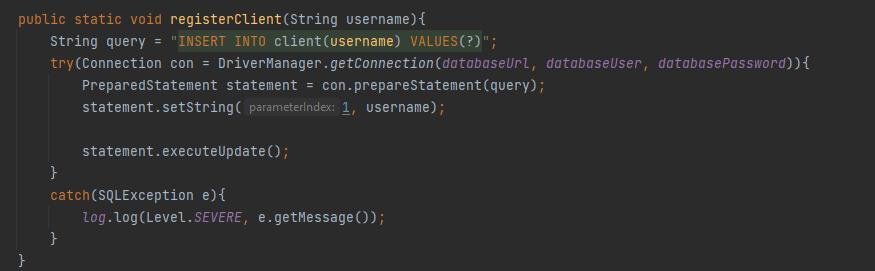
Администраторът назначава куриер с INSERT заявка.



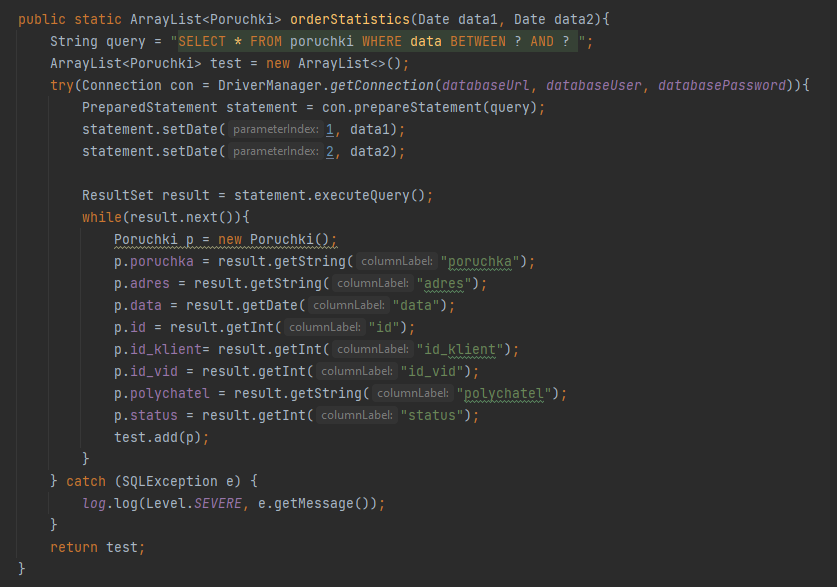
Администраторът може да уволни куриер с DELETE заявка.

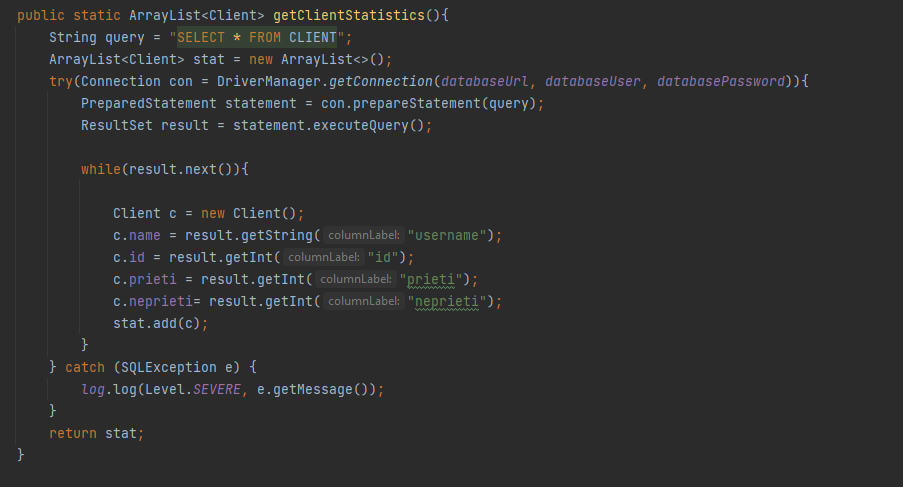


Администраторът може да регистрира клиент с INSERT заявка.

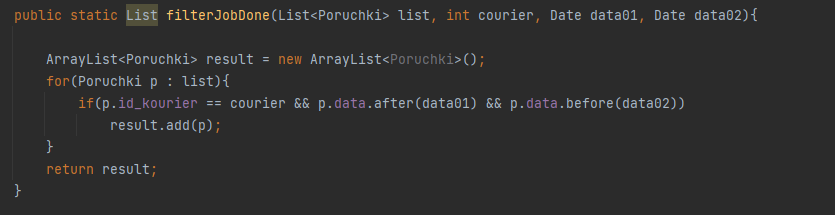


Администраторът може да провери статистиката на поръчките направени от куриерът както да провери статистиката на клиента.





Чрез combobox „izbor” селектираме името на куриерът който искаме да проверим след това селектираме периода, в който курирът е извършил поръчка, а чрез “filterJobDone” извеждаме информацията за извършената работа и броят направени от него поръчки.



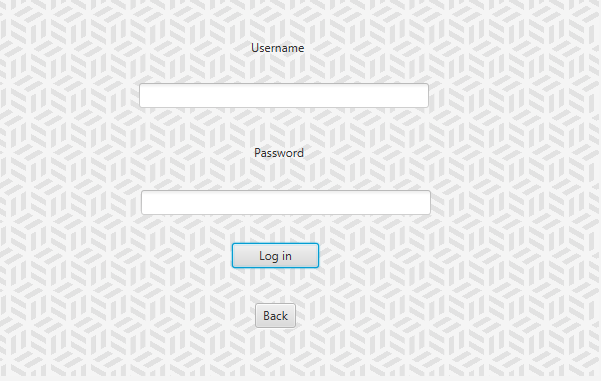
4.3)Реализация на бизнеслогика и графичен интерфейс

ш

4.3.1)Реализация на графичен интерфейс

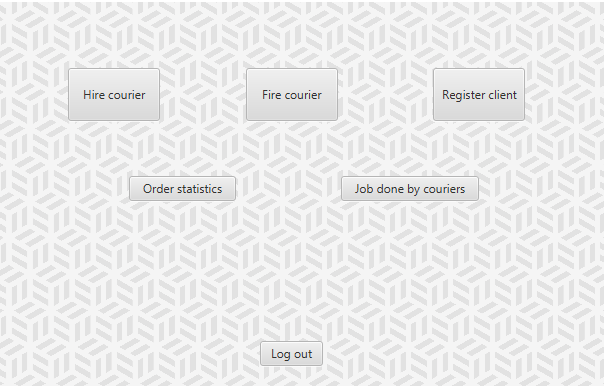
При отваряне на приложението , консуматора на програмата е посрещнат от интерфейс за избор на логване, съдържа два бутона Administrator и Courier.



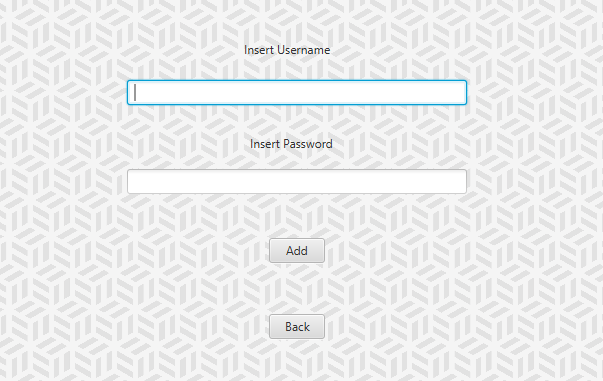


В зависимост от човека логващ се в системата ще получи различен екран и правомощия, при логване като администратор ще се отвори администраторския панел контролиран от “AdminViewController” ,ако се логне куриер ще се отвори куриерският панел контролиран от “CourierViewController”.

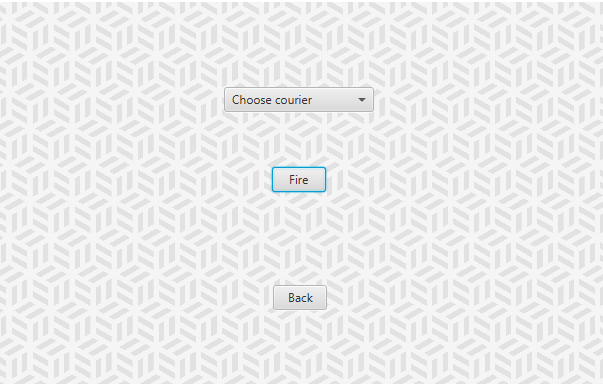
Първо логване като администратор , администраторът ще бъде посрещнат с интерфеис съдържащ шест бутона които са: Hire courier с който назначаваме куриер, Fire courier с който уволняваме куриер, Register client с който добавяме клиент, Order Statistic – статистика на поръчките, Job done by couriers – статистика на куриерите и бутон за logout-ване.



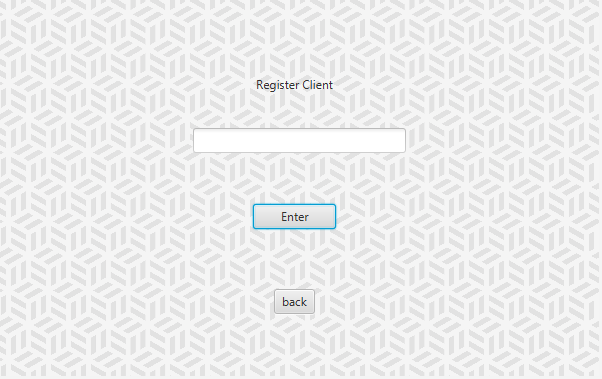
Hire courier – служи за назначаване на куриери с име и парола.



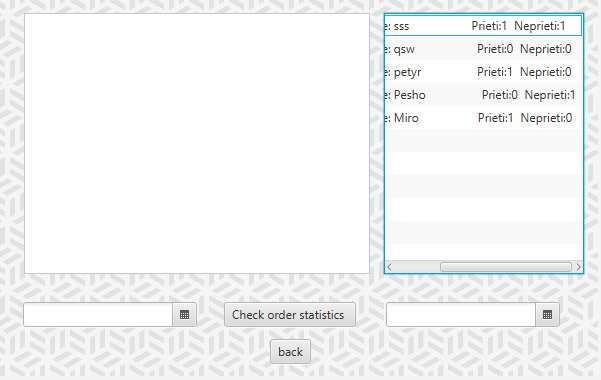
Fire courier – служи за уволняване на куриери.



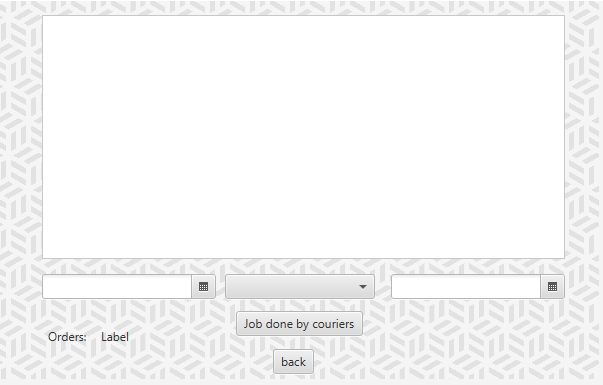
Register client – служи за добавяне на клиенти.



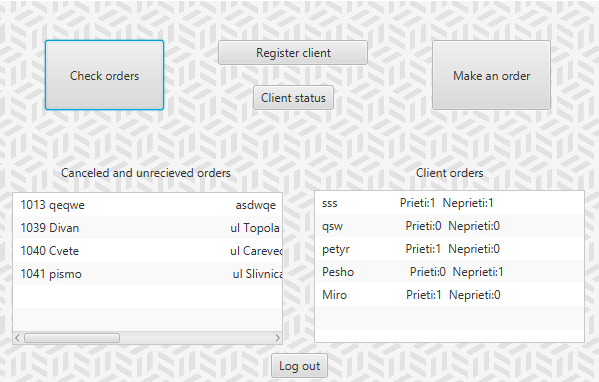
Order statistics – служи за проверка за статистиката на поръчката и клиентите.



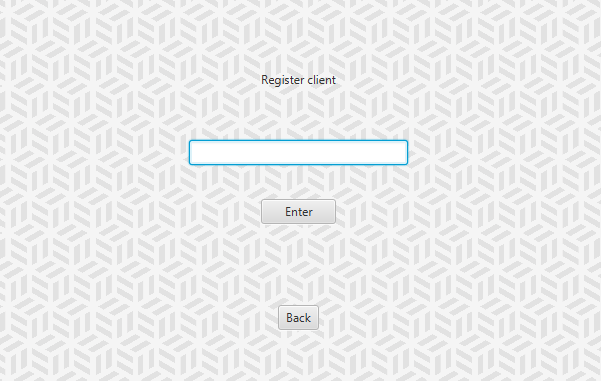
Job done by couriers – служи за проверка за статистиката на направените поръчки от куриерите и показва броя на направенитепоръчки.



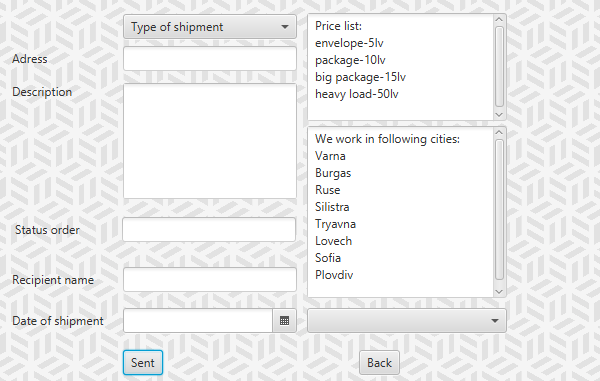
При логване на куриер, потребителят ще бъде посрещнат от пет бутона и два listbox-а. Бутоните са “Register client“ – добавяне на клиент, “Make an order“ – създаване на поръчка, “Check orders“– проверка на направените поръчки, “Client status“ – статусът на клиента, “Listbox – Canceled and unrecieved orders “ (ляво) – отказани и неполучени пратки и “Listbox Client orders“ (дясно) – статистика на приети и неприети поръчки от всеки клиент.



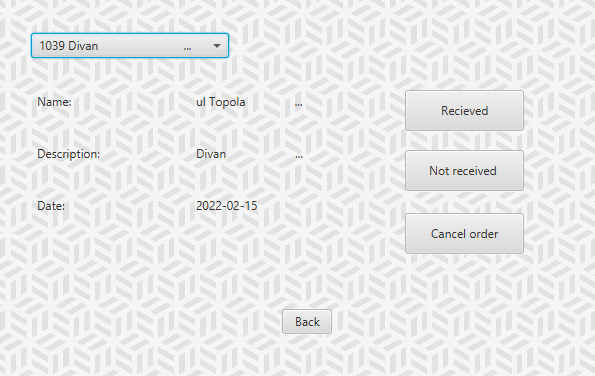
Register client – служи за добавяне на клиенти.



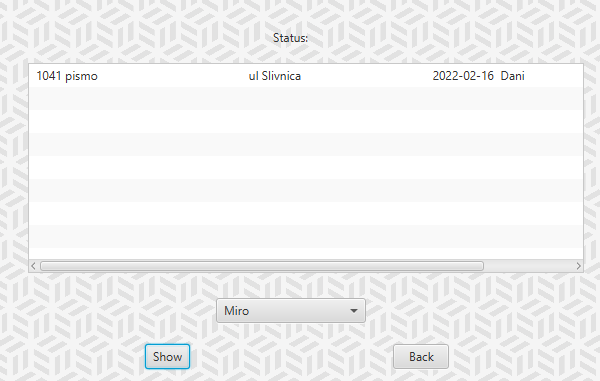
Make an order – служи за създаване на поръчки.



Check orders – служи за проверка на поръчките и променянето на тяхното състояние.

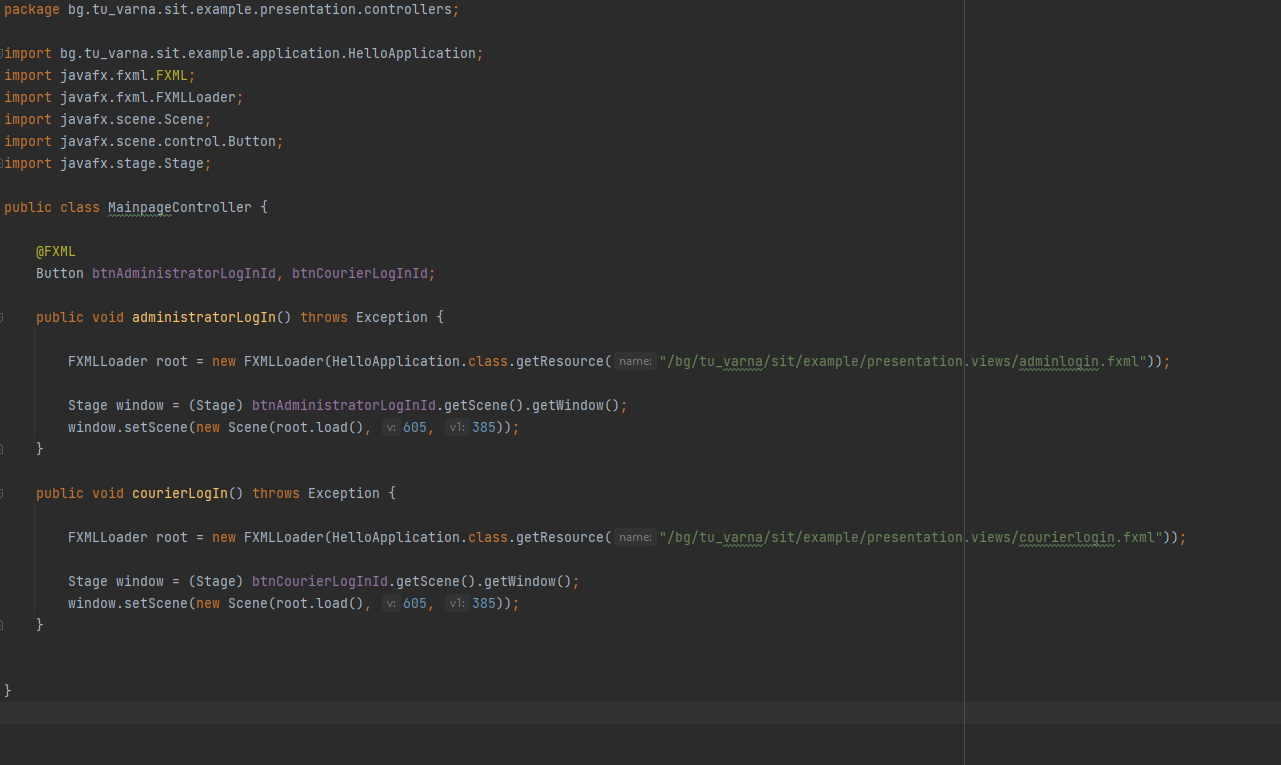


Client status – служи за проверка на статуса на поръчките на клиентите.



4.3.2)Реализация на бизнеслогика

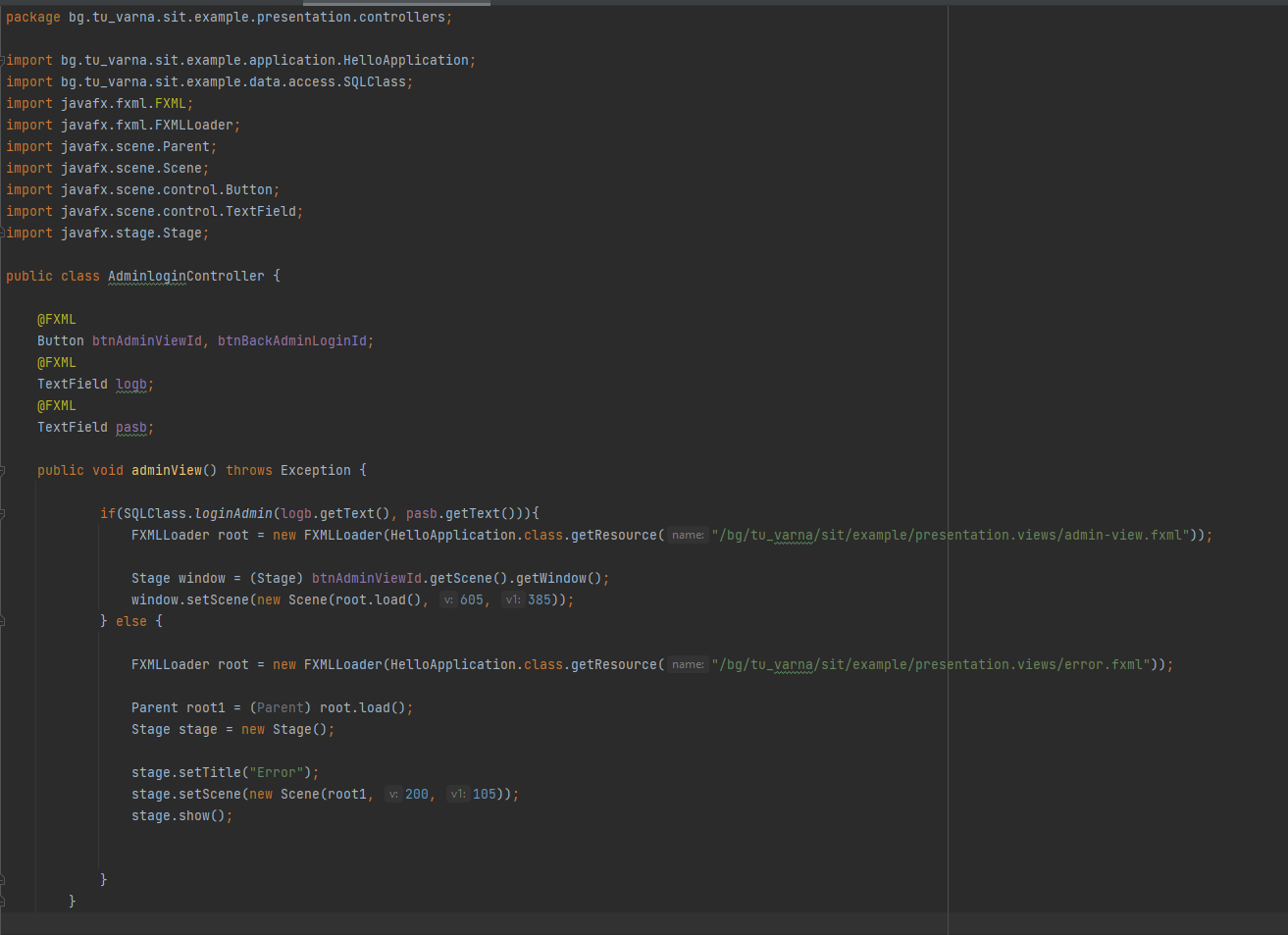
Първоначално трябва да изберем като какъв искаме да се логнем (администратор или куриер). Това става с два бутона “btnAdministratorLogInId“ (за администратор) и “btnCourierLogInId“ (за куриер).

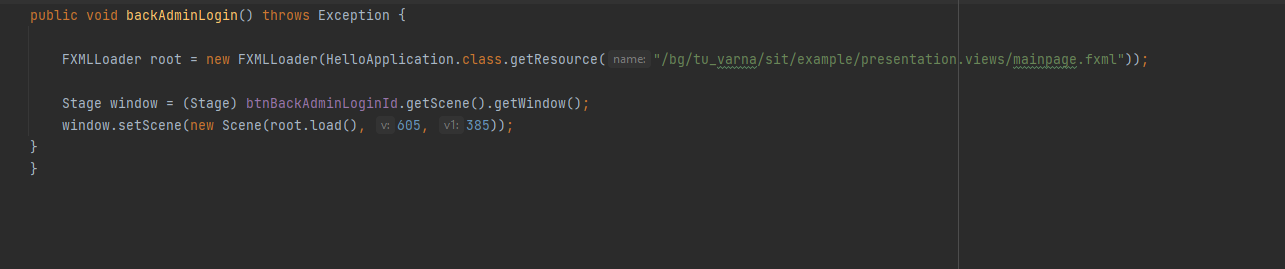


За контролирането на логин панелите се използват конролерите “AdminloginController”, „CourierloginController”, „ErrorController“ в случаят (AdminloginController).

**-”btnAminViewId”:** Логваме се с username и password. Ако информацията не съвпада с Database таблицата дава грешка.

**-”btnBackAdminLoginId” :** затваря логин панела и се връщаме на предишната страница.

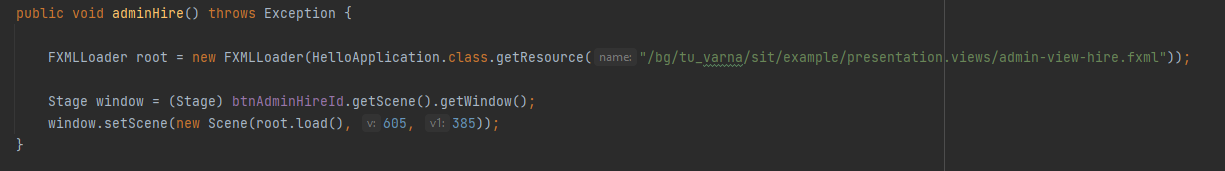




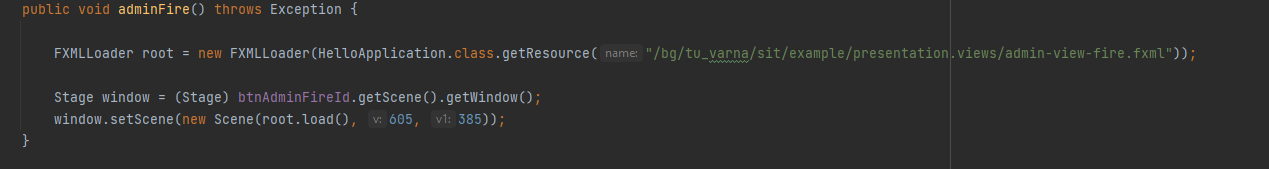
Администраторският панел се състои от един контролер - “adminView”.

“AdminViewController” съдържа шест метода :

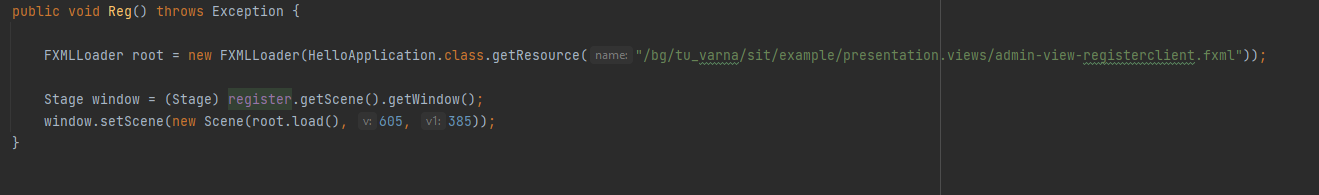
**-” adminHire”:** С натискането на бутона “btnAdminHireId“ се извиква метод, чрез който се сменя сцената за назначаване на куриер.



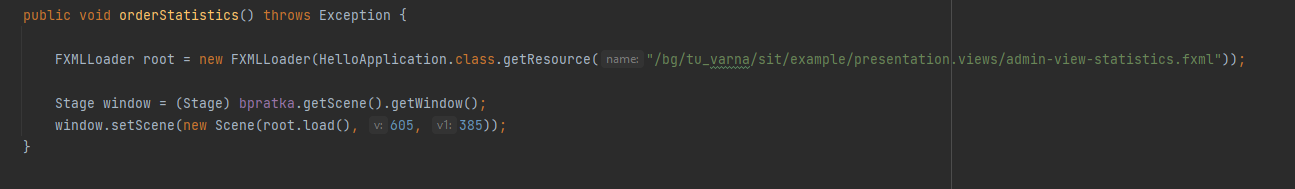
**-” adminFire”:** С натискането на бутона “btnAdminFireId“ се извиква метод, чрез който се сменя сцената за уволняване на куриер.



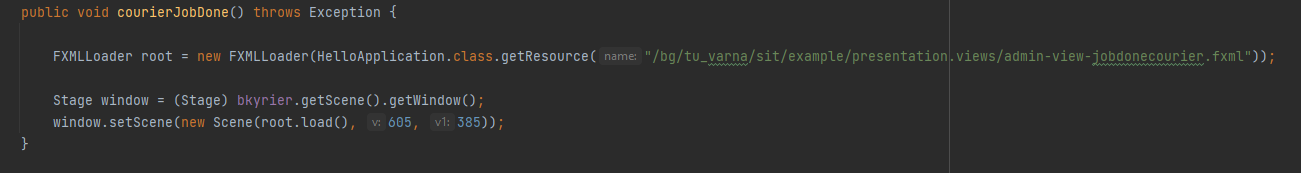
**-” Reg”:** С натискането на бутона “register“ се извиква метод, чрез който се сменя сцената за добавяне на клиент.



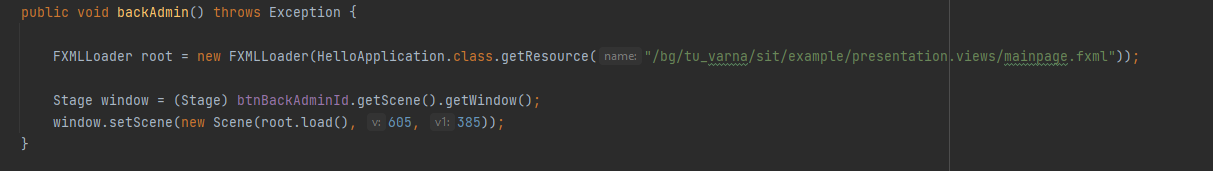
**-”orderStatistics”:** С натискането на бутона “bpratka“ се извиква метод чрез който се сменя сцената за статистика на пратките и клиентите.



**-”courierJobDone”:** С натискането на бутона “bkyrier“ се извиква метод чрез който се сменя сцената за работата на куриерите.



**-”backAdmin”:** С натискането на бутона “btnBackAdminId“ се извиква метод, с който се връщаме към първоначалната страница.



“AdminViewHireController” съдържа два метода:

**-”addCurier”:** С натискането на бутона “btnBackHireId“ се извиква метод, с който назначаваме куриер.

**-”backHire”:** С натискането на бутона “btnBackHireId“ се връщаме към предходният контролер.



“AdminViewFireController” съдържа три метода:

**-”initialize”:** Метод чрез който селектираме назначен куриер.

**-”fireCourier”:** С натискането на бутона “fireB“ уволняваме селектираният куриер.

**-”backFire”:** С натискането на бутона “btnBackFireId“ се връщаме към предходният контролер.



“AdminViewRegisterclientController” съдържа два метода:

**-”Register”:** С натискането на бутона “imekl“ добавяме клиент.

**-”backAdminView”:** С натискането на бутона “btnBackAdminViewId“ се връщаме към предходният контролер.

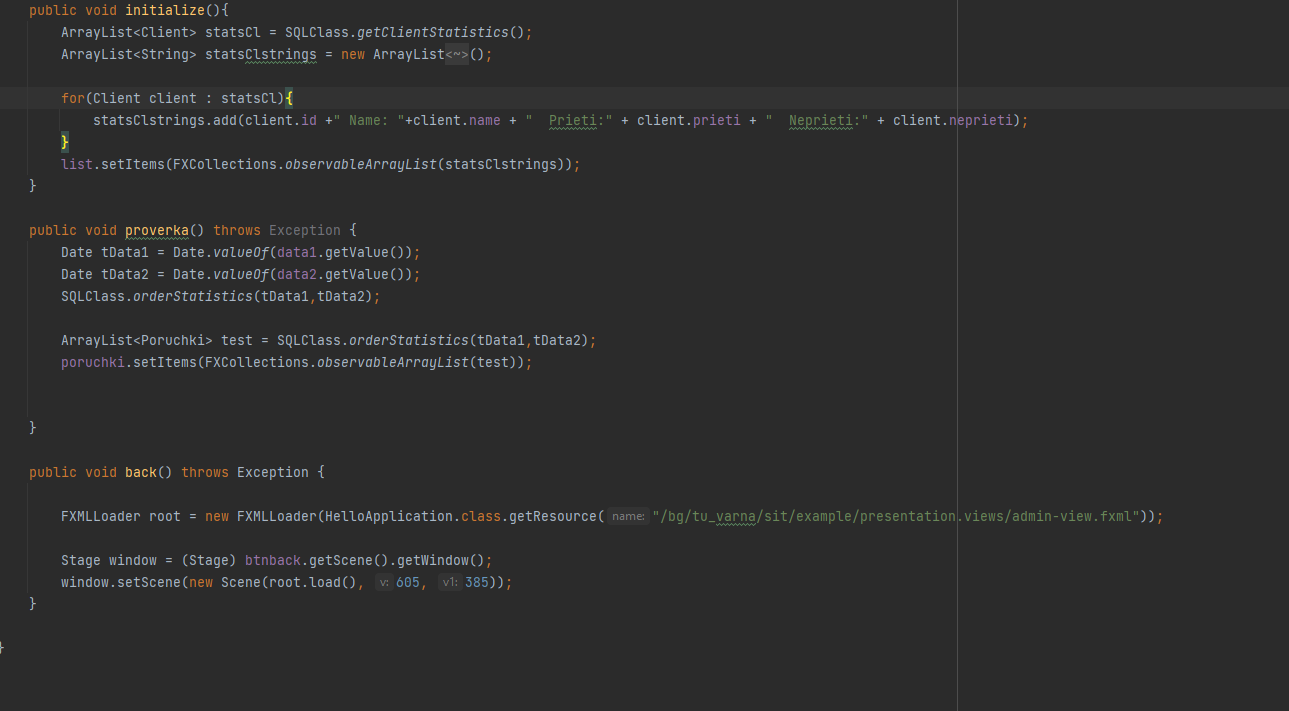


“AdminViewStatisticsController” съдържа три метода:

**-”initialize”:** Метод чрез който показва приетите и неприетите пратки на клиента.

**-”proverka”:** С натискането на бутона “poruchki“ извеждаме направените поръчки от клиентите за даден период от време.

**-”back”:** С натискането на бутона “btnback“ се връщаме към предходният контролер.

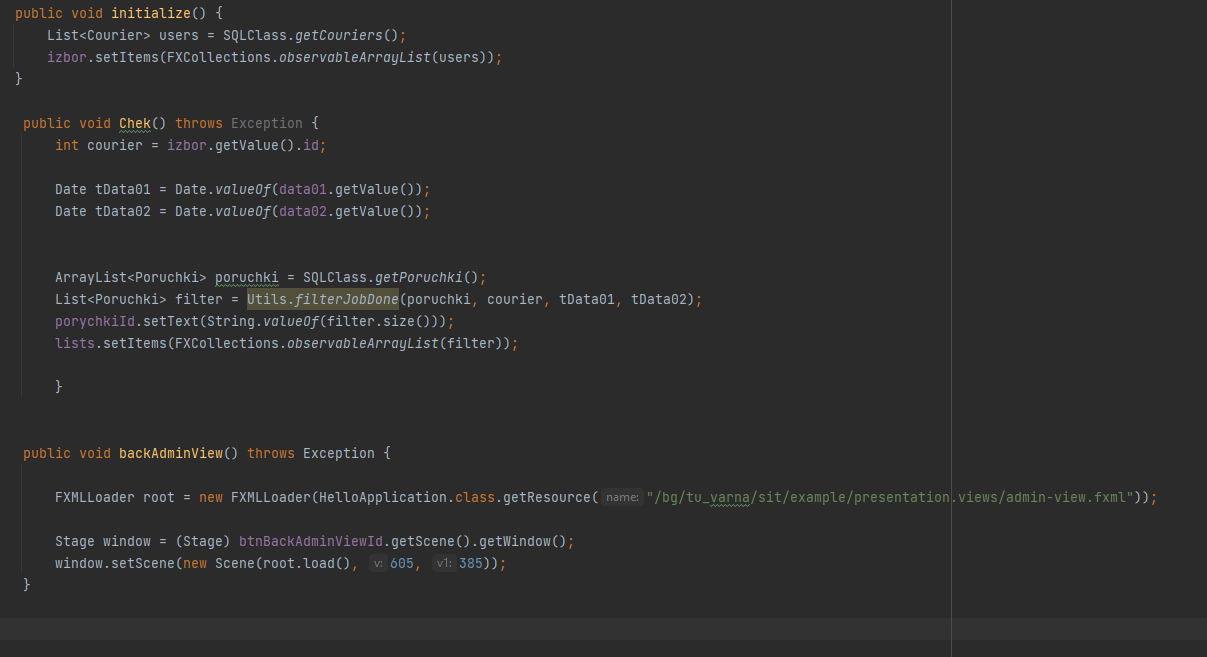


“AdminViewJobdonecourierController” съдържа три метода:

**-”initialize”:** Метод чрез който селектираме името на куриерът от combobox.

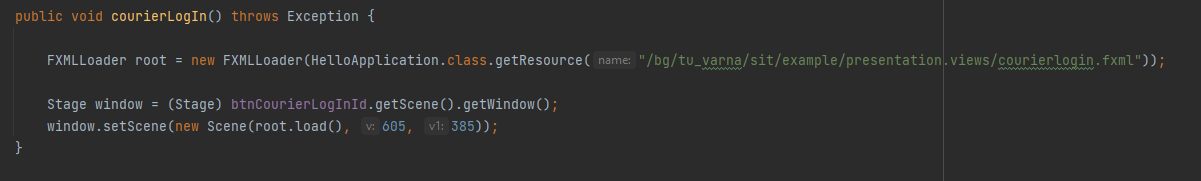
**-”Check”:** С натискането на бутона “chek“ извеждаме направените поръчки от куриерите за даден период от време.

**-”backAdminView”:** С натискането на бутона “btnBackAdminViewId“ се връщаме към предходният контролер.



Courier Login

След като се logout-нем от админ излизаме отново на първоначалната страница и се логваме с бутонът “btnCourierLogInId“, който е в метод courierLogIn.



“CourierloginController” съдържа два метода:

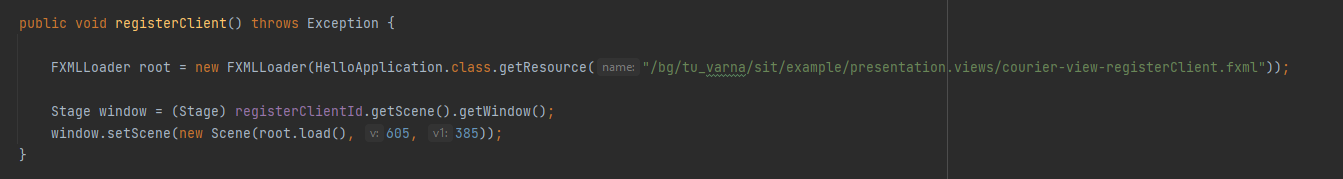
**-”courierView”:** Логваме се с username и password. Ако информацията не съвпада с Database таблицата дава грешка

**-”backCourierLogin”:** С натискането на бутона “btnBackCourierLoginId“ се връщаме към предходният контролер.

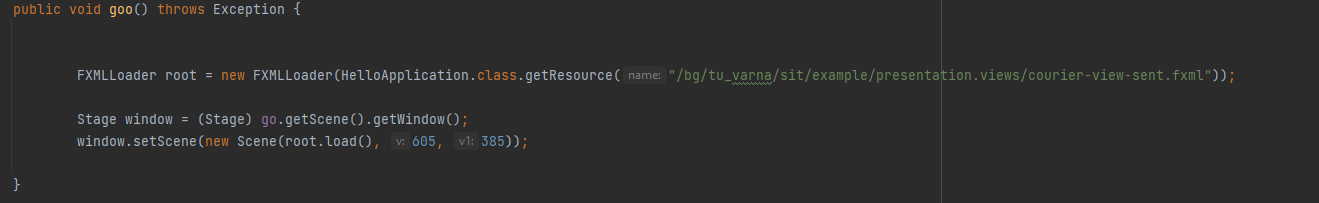


“CourierViewController” съдържа шест метода:

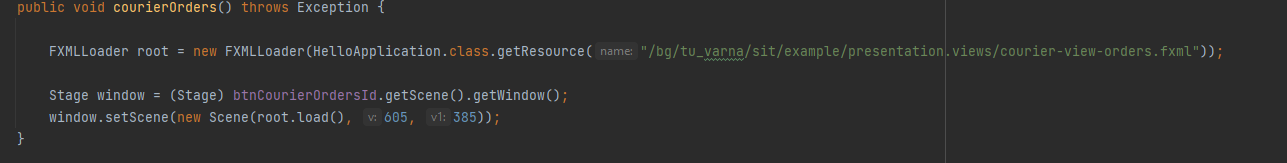
**-”registerClient”:** С натискането на бутона “registerClientId“ се извиква метод, чрез който се сменя сцената за добавяне на клиент.



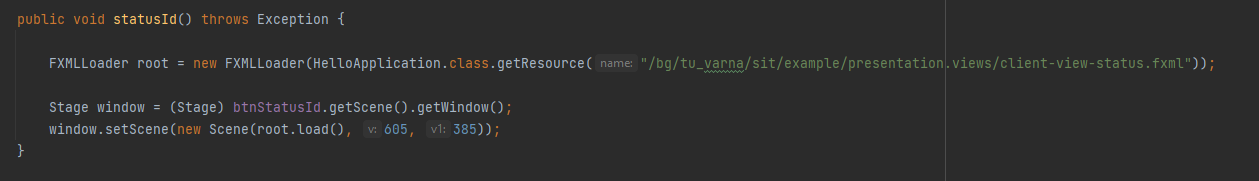
**-”goo”:** С натискането на бутона “go“ се извиква метод, чрез който се сменя сцената за правене на поръчка.



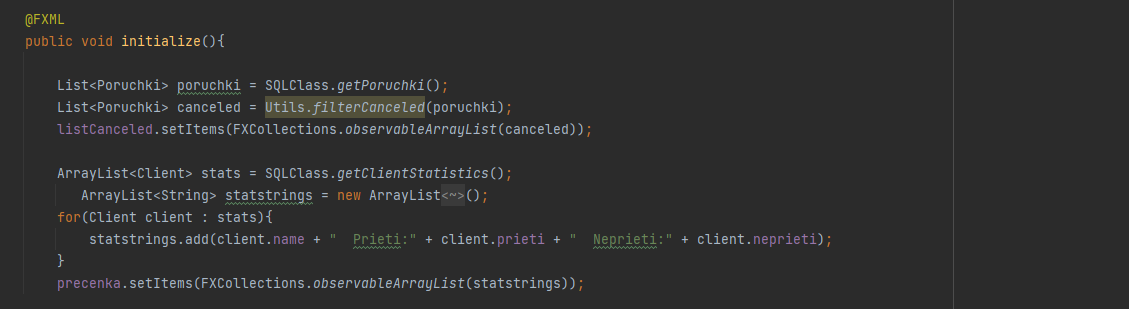
**-”courierOrders”:** С натискането на бутона “btnCourierOrdersId“ се извиква метод, чрез който се сменя сцената за проверка на поръчките.



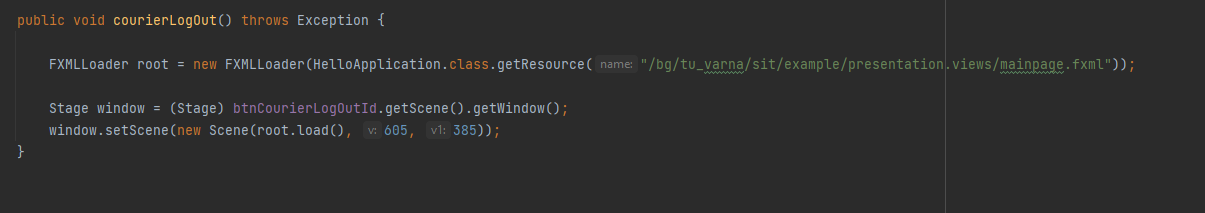
**-”statusId”:** С натискането на бутона “btnStatusId“ се извиква метод, чрез който се сменя сцената за проверката на статуса на клиентите.



**-”initialize”:** Извежда в List<Poruchki> canceled отказаните и неприетите поръчки на получателите, а статистиката на клиентие се извежда в List<Poruchki> poruchki.



**-”courierLogOut”:** С натискането на бутона “btnCourierLogOutId“ се връщаме към първоначалният контролер.



„CourierViewRegisterClient” съдържа два метода:

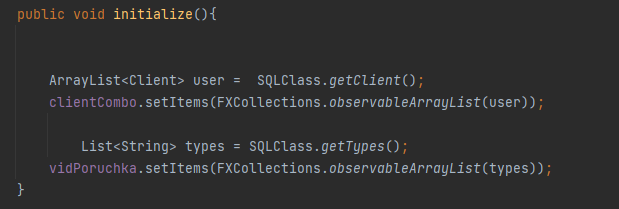
**-”add”:** С натискането на бутона “addClient“ се извиква метод чрез който се добавя клиент.

**-”backCourierView”:** С натискането на бутона “btnBackCourierViewId“ се връщаме към предходният контролер.



“CourierViewController” съдържа четири метода:

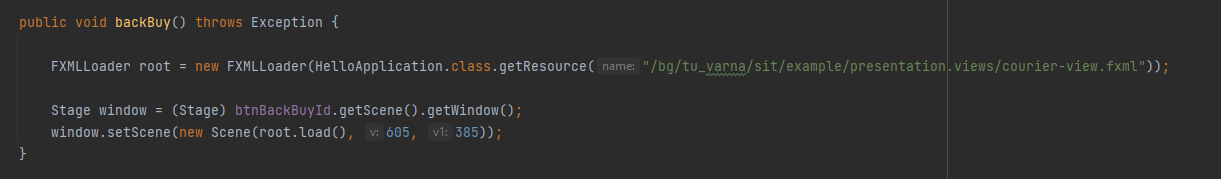
**-”initialize”:** С натискането на combobox-a clientCombo се извиква метод“ чрез който се избира клиент, а чрез бутонът vidPoruchka избираме видът на поръчката.



**-”clientBuy”:** Метод чрез който натискаме бутонът “btnClientBuyId“ изпращаме поръчката към database-а с информация за видът на поръчката, адресът на получателят, описанието, датата на пращане, статуса на пращане и името на получателят, когато въведем всияки данни правилно се изкарва нов прозорец с текст „Order successful“.



**-”backBuy”:** С натискането на бутона “btnBackBuyId“ се връщаме към предходният контролер.



“CourierViewOrdersController” съдържа шест метода:

**-”initialize”:** С натискането на combobox-a “comboPoruchki“ се извиква метод, чрез който се избира клиент на който може да проверим поръчката.

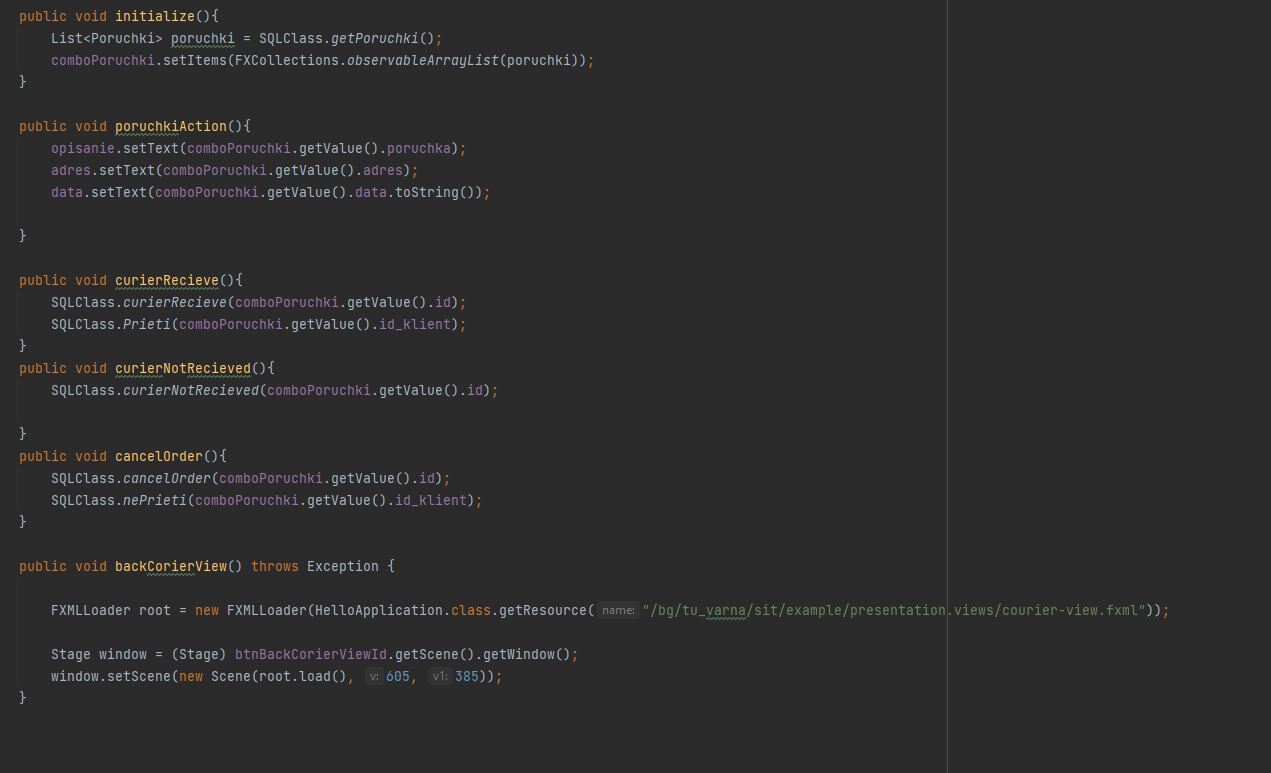
**-”poruchkiAction”:** Извиква се метод който променя описанието на label-ите opisanie,adres и data съответно в описание, адрес и дата.

**-”curierRecieve”:** С натискането на бутона “curierRecieveId“ се променя статуса на поръчката като получена.

**-”curierNotRecieved”:** С натискането на бутона “curierNotRecievedId“ се променя статуса на поръчката като неполучена.

**-”cancelOrder”:** С натискането на бутона “cancelOrderId“ се променя статуса на поръчката като отказана.

**-”backCorierView”:** С натискането на бутона “btnBackCorierViewId“ се връщаме към предходният контролер.



“ ClientStatusController” съдържа три метода:

**-”initialize”:** С натискането на combobox-a “clientCombo“ се извиква метод, чрез който се избира клиент, който е направил поръчка.

**-”Go”:** С натискането на бутона “go“ се показва статуса на поръчката в List<Poruchki> poruchki, на който проверяваме статуса поръчката.

**-”BackStatus”:** С натискането на бутона “btnBackStatusId“ се връщаме към предходният контролер.



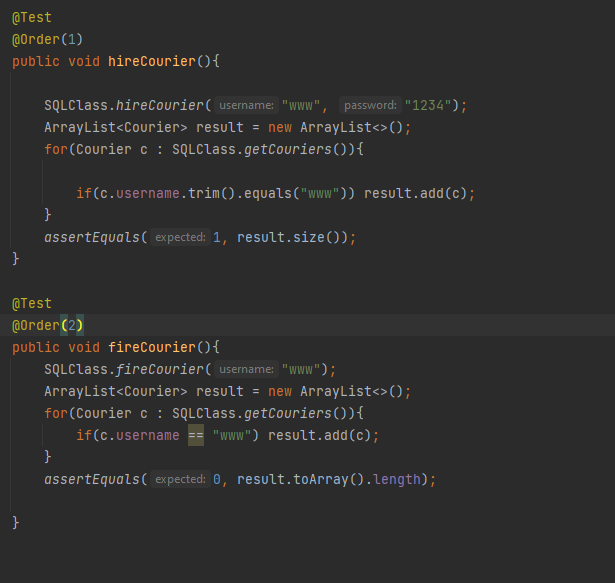
4.4)Реализация на модул за регистриране на събития в системата

За логване на събития в системата сме използвали log4J.

5)Тестови резултати

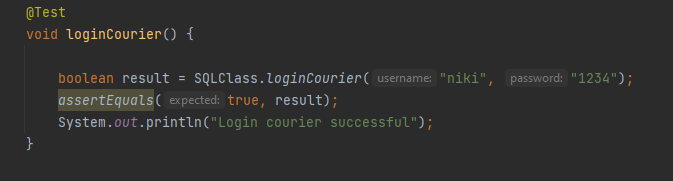
Тестовите резултати са направени в папка Tests под името SQLClassTest. При тестовете няма опасност database-ът да се препълни или натовари от информация.

Тест за назначаване на куриер и уволняването му с“ test oreder“.



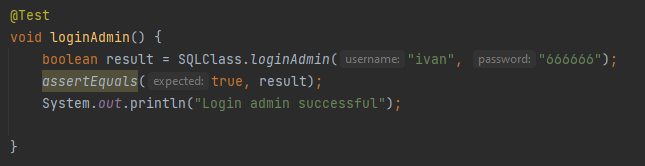


Тест за логване на куриер.



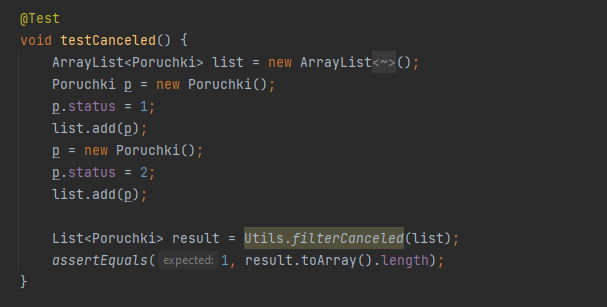


Тест за логване на администратор.



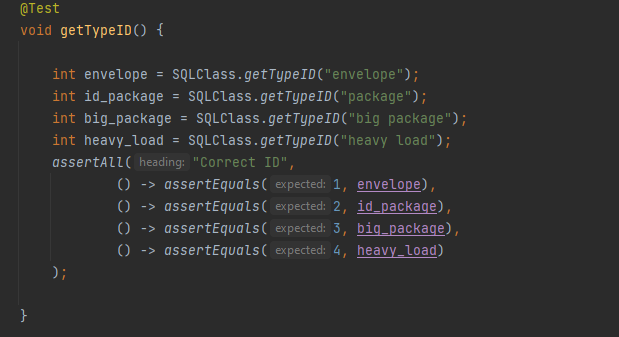


Тест отказана поръчка.



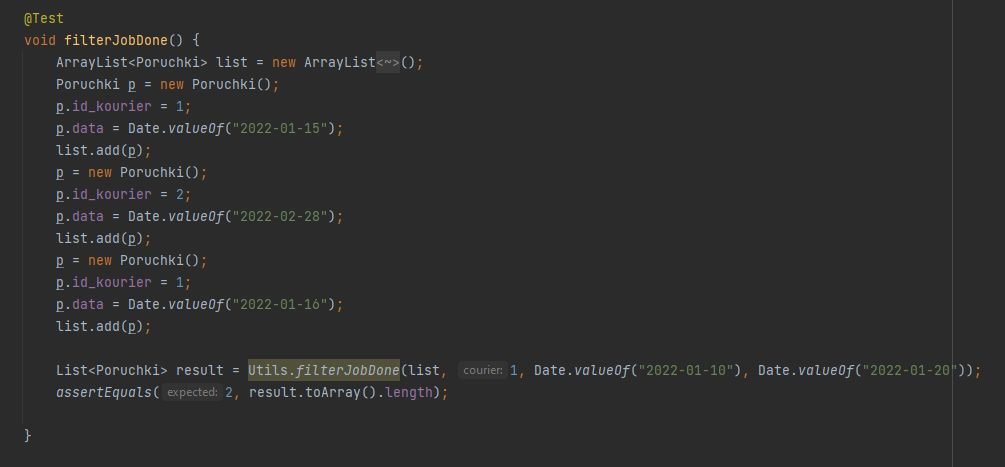


Тест за съдържанието на “vid\_Porichki“.





Тест за поръчки на куриер в даден период..





Тест за проверка на поръчки от клент.

Text

Description automatically generated



Тест за проверка на поръчка от несъществуващ куриер.

