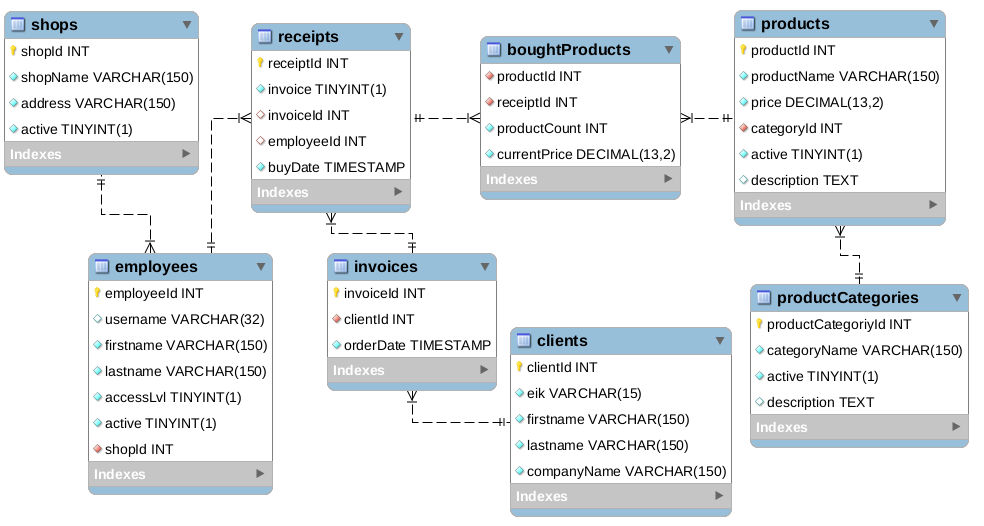
Проект: Приложение за плащане в магазин

1. Описание на проекта

Проектът представлява десктоп приложение за плащане в магазин. Изпълнява четири основни функционалности:

* добавяне/променяне на магазини и служители, които работят в тях
* добавяне/променяне на продукти и техните категории
* изготвяне на касови бележки и фактури
* създаване на отчети

Потребителският интерфейс (създаден с графичния интерфейс за форми на NetBeans) и логиката са написани на Java и библиотеката Swing. Данните се съхраняват в MySQL база данни, чиято структура е описана във Фигура 1. Тя се достъпва от приложението чрез библиотеката JDBC. Ключовете в базата данни се генерират автоматично. Потребителите на програмата имат акаунти, които съвпадат с акаунтите им за базата и всички те се съхраняват като потребителски акаунти в MySQL.



*Фигура 1.* Модел на обектните взаимовръзки на базата данни

1. Описание на кода

Кодът на приложението е разделен на четири основни пакета (виж Апендикс):

* 1. Пакет „db”

Този пакет съдържа два класа за достъп до базата данни. „ConnectionFactory.java“ се използва за създаване на връзки/сесии с базата. Класът „DBwriteThread.java“ наследява нишка и се използва за писане в базата. Има два начина на ползване: когато върви като отделна нишка получава входни SQL заявки от други класове чрез java.util.concurrent.LinkedBlockingQueue и при натрупване на определен брой заявки ги изпълнява заедно. Има и статичен метод за изпълняване на единствена заявка, за която да се върне създаденият от базата уникален ключ, ако той е нужен веднага. Също съдържа и няколко статични метода за форматиране на низове на SQL заявки за контрол на потребителските акаунти.

* 1. Пакет „models“

Тук се съдържат класове, които отговарят на обектите от базата данни. Те имат същите член данни като обектите в базата и всички наследяват абстрактният клас „Model”, който съдържа флагове, за това дали трябва да се обновят в базата (ползват се по време на работа). Тук се съдържат и няколко класа за сравнение, на някои от моделите, в пакета „models. compare“.

* 1. Пакет „infrastructure“

Основният клас в този пакет е „MySqlRepository.java“. Той е generic клас, който може да бъде от тип всеки един от моделите. Инстанция на този клас чете от базата данни за съответния модел (тъй като при четене резултатът ни трябва веднага няма смисъл това да се прави на отделна нишка) – например SELECT -и форматира низове за SQL за писане в базата, които изпраща на DbwriteThread чрез блокиращата опашка – например INSERT. Това се извършва чрез интроспекция на модела-тип и в пакета „infrastructure.interfaces” има анотации, които се поставят в моделите и указват информация за представянето им в базата данни (например кое поле е ключ и т.н.).

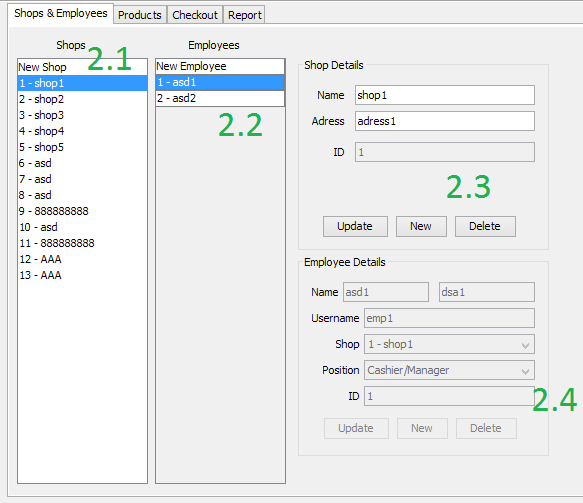
2.3 Пакет „ui”

В този пакет се реализира потребителския интерфейс и функционалността на приложението. Класът „ShopForm.java“ е началната точка на програмата. В него се инициализират гореописаните класове. Съдържа генерираният код от NetBeans за създаване на потребителския интерфейс и код за обработка на събитията. Други по-важни класове от този пакет е „Dialogues.java“, в които има статични функции за създаване на различни прозорци с диалози и „LogDocStream.java“, който пренасочва стандартният изход към компонент JTextArea.

1. Описание на функционалността

Потребителският интерфейс може да се раздели на две основни части: панел четири таба (JTabbedPane), които предоставя функционалността описана по-горе (1.), използвайки гореописаните пакети, и текстово поле, което винаги е видимо и предоставя обратна връзка на потребителя. Преди да може да се достъпи приложението, потребителят трябва да се свърже към базата данни като въведе акаунта и паролата си. В момента на работа текущата функционалност на приложението зависи от показания таб:

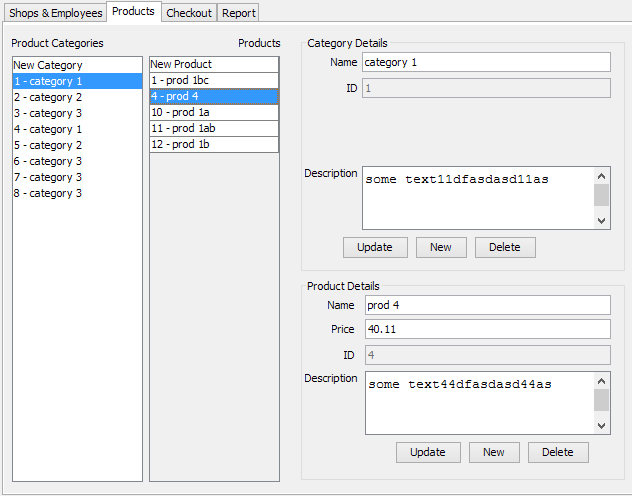
* 1. Панел „Shops” (Фигура 2)

Това е началният панел на приложението, след успешна връзка с базата данни. В него са показани всички магазини (в JList, *Фиг. 2.1*) и служителите в тях (в JTable, *Фиг. 2.2*). Според текущо избрания магазин, таблицата със служители се филтрира, показвайки само хората работещи в него. Детайлите име, адрес и номер на избраният магазин се зареждат и първите две могат да се променят от потребителя, или да се създаде нов (*Фиг. 2.3*). Информацията избрания касиер също се зарежда в друг панел, където може да се променя неговата позиция и магазин, в който работи (*Фиг. 2.4*). Тези две операции могат да се извършват само от потребители с ниво на достъп, което им позволява да контролират акаунти в базата данни. При влизане в този панел от базата данни се зарежда информацията за магазини и работници. При създаване на нов магазин, SQL заявката за това се изпълнява веднага, за да се получи неговия ключ (в случай че потребителят иска веднага да добавя работници в него). Останалите промени по базата данни се изпращат за обработка след като се напусне панела.

*Фигура 2*. Панел „Shops”

* 1. Панел “Products” (Фигура 3)

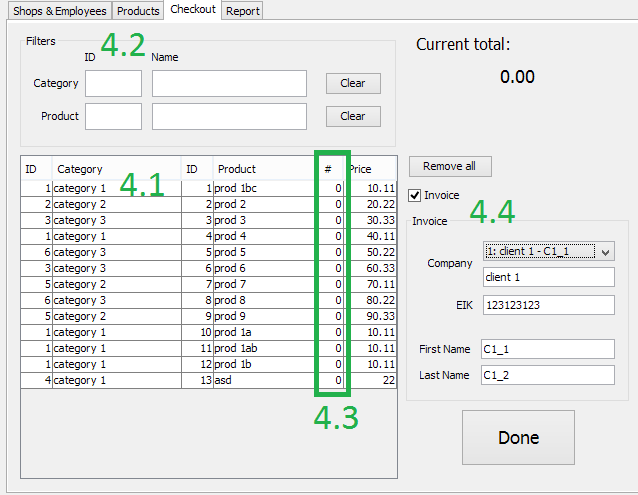
Този панел предоставя функционалността за преглед и промяна на продукти и категории. Действието му е аналогично на панела „Shops”.



*Фигура 3*. Панел “Products”

* 1. Панел „Checkout”

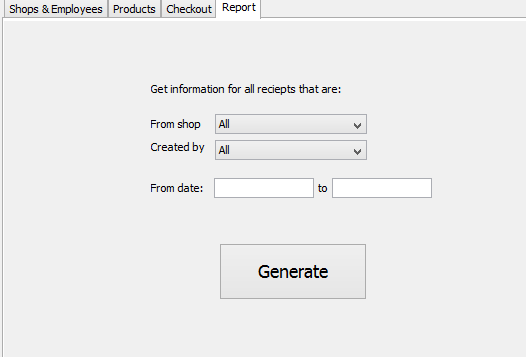
Този панел служи за създаването на касови бележки. Основният компонент в него е един JTable, който съдържа всички продукти, техните категории и цени (*Фиг. 4.1*). Продуктите могат да се филтрират комбинирано по номер и име на продукта и номер и име на категорията (*Фиг 4.2*). Броя на закупения продукт се въвежда от потребителя в съответната колона на таблицата (*Фиг 4.3*) и общата цена се пресмята. Може да се избере издаване на фактура като или да се въвежда нова компания или се избира една от вече използваните (*Фиг. 4.4*). При натискане на бутона за приключване се създава касова бележка и фактура (ако е избрана), които могат да се запазят локално. При влизане в панела се (пре)зареждат продуктите и клиентите от базата данни. При създаването на касова бележка и фактура, SQL заявките за това се изпълняват веднага в нишката на awt (awt-eventqueue), за да се достъпи уникалния им номер, а останалата информация (детайли за закупени продукти) се изпраща на DBwriteThread.java.

 *Фигура 4.* Панел „Checkout”

* 1. Панел „Report“ (Фигура. 5)

В този панел потребителят може да създава отчети за продажбите. В зависимост от избраните филтри се генерира различен отчет:

* Ако е избран някой служител, се показват всички касови бележки, издадени от него, за избрания период и техните суми
* Ако е избран някой магазин и всички служетли, се показва отчет за оборота изработен от всеки служител, в избрания магазин.
* Ако са избрани всички служители и всички магазини се показва отчет за оборотите на всички магазини.



*Фигура 5*. Панел „Report“

Апендикс

Структурата на пакетите на Java кодът.

Пълният код, както и SQL скриптове за дефиниция на базата данни и добавяне на тестови данни, могат да бъдата намерни тук: <https://github.com/kgskgs/java-shop-model/>

.

└── shop

├── audit

│   └── Report.java

├── db

│   ├── ConnectionFactory.java

│   └── DBwriteThread.java

├── infrastructure

│   ├── interfaces

│   │   ├── Foreign.java\*

│   │   ├── IRepository.java

│   │   ├── Key.java

│   │   ├── Name.java\*

│   │   ├── Table.java

│   │   └── Timestamp.java\*

│   └── MySqlRepository.java

├── models

│   ├── BoughtProduct.java

│   ├── Category.java

│   ├── Client.java

│   ├── compare

│   │   ├── EmpSortByShop.java\*

│   │   ├── ProdSortByCategory.java\*

│   │   └── ShopSortById.java

│   ├── Employee.java

│   ├── Invoice.java

│   ├── Model.java

│   ├── Product.java

│   ├── Receipt.java

│   └── Shop.java

└── ui

├── Dialogues.java

├── DocChangeListener.java

├── LogDocStream.java

├── mainUI.java

├── ShopForm.form

├── ShopForm.java

└── TableCellListener.java\*\*

\* не се използва в приложението

\*\* взет е от <https://tips4java.wordpress.com/2009/06/07/table-cell-listener/> с позволение на автора