**PillBox Documentation**

**Date:** Sep 14, 2023 **Version:** 1.0 **author:** Stoyan Stoyanov

**1. Цел на проекта:**

Проектът има за цел да създаде и поддържа система за управление на информацията за медицински продукти и инвентар на аптека.

1.1. Реализиране на функционалности за добавяне на медицински продукти в базата данни от външен източник (Excel файл).

1.2. Реализиране на функционалности за търсене на медицински продукти според различни критерии.

1.3. Управление на инвентар, доставчици и фактури.

1.4. Създаване и управление на потребители с различни права (администратори и обикновени потребители).

1.5. Реализиране на функционалности за продажби на медицински продукти и актуализиране на инвентара след продажба.

Целта на проекта е да предостави инструмент за ефективно управление на медицински продукти и инвентар в аптека.

**2. Потребителски изисквания:**

**1. Вход**

1.1. Идентификация на потребителя

* Потребителите трябва да могат да се вписват с избрано работно място.
* Работните места трябва да бъдат изброени като опции за потребителите.
* Потребителите трябва да предоставят потребителско име и парола за идентификация.

**2. Основно меню**

2.1. Продажби

* Потребителят трябва да може да търси продукти по име или идентификационен номер.
* Потребителят трябва да може да добавя продукти към текущата продажба.
* Потребителят трябва да може да премахва продукти от текущата продажба.
* Приложението трябва да изчислява общата сума на продажбата, включително сума без ДДС, ДДС и обща сума с ДДС.
* Потребителят трябва да може да завърши продажбата.
* Потребителят трябва да може да затвори текущата продажба, ако тя е празна.
* Приложението трябва да може да записва и обработва продажбите.
* Приложението трябва да обработва грешки, като например опит за добавяне на продукт с недостатъчно количество.

2.2. Номенклатура

* Потребителите трябва да имат достъп до секцията „Номенклатура“.
* Секцията „Номенклатура“ трябва да улеснява търсенето на медицински продукти въз основа на различни критерии.
* Трябва да е възможно да се търсят продукти по име или ID.

2.3. Управление на инвентара

* Потребителите трябва да имат достъп до управлението на инвентара, по-специално до функцията „Приемане на инвентар“.
* Функцията „Приемане на инвентар“ трябва да даде възможност на потребителите да добавят нови медицински продукти към инвентара.
* Потребителите трябва да могат да въвеждат данни като име, количество, цена на доставка, дата на изтичане и партиден номер. \* Системата трябва да валидира въведените данни.

2.4. Навигация

* Потребителите трябва да могат да се движат между различните раздели на приложението.
* Навигацията трябва да бъде интуитивна и потребителите трябва лесно да превключват между „Продажби“, „Номенклатура“ и „Управление на инвентара“.

2.5. Изход

* Потребителите трябва да могат да излизат от приложението.
* Излизането трябва да върне потребителите към началния екран за вход.

**3. Функционалност за търсене**

3.1. Търсене

* В секцията „Номенклатура“ потребителите трябва да могат да търсят медицински продукти.
* Търсенето трябва да поддържа критерии като име или ID.
* Резултатите от търсенето трябва да бъдат изобразени по ясен и организиран начин.

3.2. Страница

* Ако резултатите от търсенето обхващат множество страници, потребителите трябва да могат да се движат между страниците.
* Бутоните „Назад“ и „Напред“ трябва да позволят на потребителите да се движат през резултатите от търсенето.

**4. Управление на инвентара**

4.1. Получаване на инвентар

* Потребителите трябва да могат да получават инвентар и да добавят медицински продукти към базата данни.
* Системата трябва да подкани потребителите да въведат данни за всеки продукт, включително количество, цена на доставка, дата на изтичане и партиден номер.
* Валидацията на входните данни трябва да гарантира точността на данните.

4.2. Изчисляване на общата сума

* Приложението трябва да изчисли и изведе общата сума за получен инвентар.
* Тази сума трябва да се актуализира динамично, когато се добавят или модифицират елементи.

4.3. Съхраняване на инвентара

* Потребителите трябва да могат да потвърдят и съхранят получените данни за инвентар

**4. Разработка**

4.1. Архитектура на приложението:

* MVC – Model View Controller
* Singelton pattern при Датабаза модела

4.2. Файлова структура:

* **main.py**–Този скрипт създава и инициализира главния прозорец на приложението, настройва връзката между моделите, представянето и контролера, и стартира графичния потребителски интерфейс .
* **view.py**–Съдържа основните класове, които визуализират приложението и го стартират.
* **db.py**–Този файл съдържа основните настройки и инструменти за управление на база данни чрез библиотеката SQLAlchemy.
* **models.py**– Съдържа основните модели за приложението
* **controller.py**–Съдържа основния контролер за приложението, който отговаря за бизнес логиката

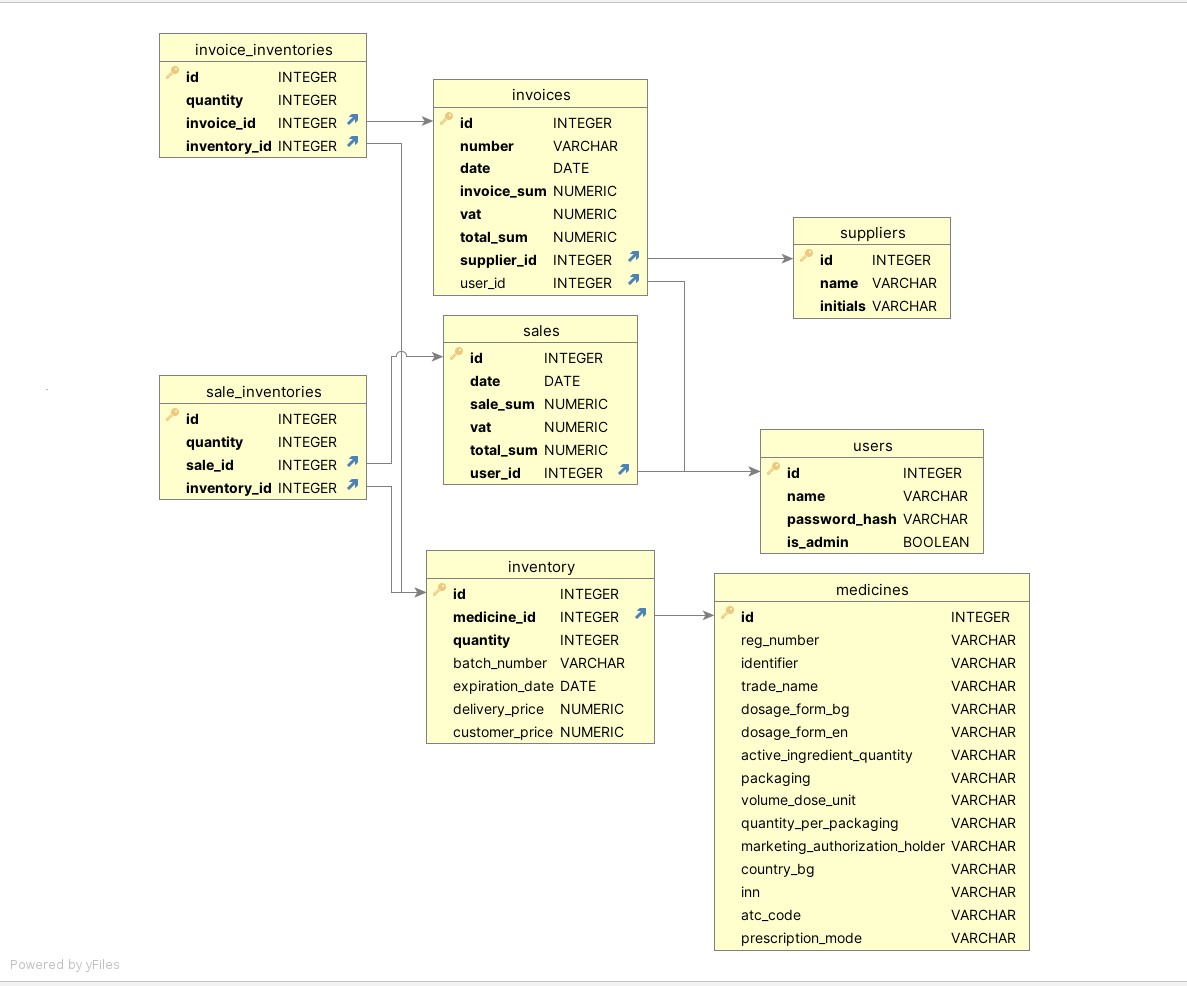
* **IAL\_Register\_07\_2023.xls**–Съдържа лекарствен списък на ИАЛ в xls формат, служи за външен източник за добавяне на медицински продукти
* **utils.py**–Съдържа клас с наименованието WindowUtils, който предоставя удобни методи за центриране на графичните прозорци
* **Папка images**–Съдържаща основните изображения, с които работи приложението
* **requirement.txt**–Съдържа списък от Python пакети и техните версии, които трябва да бъдат инсталирани във виртуалната среда на проекта.
* **database.db**–Датабазата на приложението, първоначално не съществува, създава се след стартирането на main.py.

4.3. Библиотеки и модули:

* **tkinter**–Библиотеката за създаване на графични потребителски интерфейси (GUI) в Python.
* **ttkbootstrap**–Предоставя теми и стилове за изгледа на tkinter
* **decimal**–Предоставящ възможност за работа с десетични числа с фиксирана точност
* **Sqlalchemy**–Предоставя ORM(Object-Relational Mapping) инструменти за управление на релационни база данни
* **random**- Предоставя функции и методи за генериране на случайни числа и данни
* **datetime**–Предоставя функционалност за работа с дата и време.
* **xlrd**–Позволява четене на данни от файлове във формат Excel(.xls)
* **bcrypt**–Използва се за хеширане и сигурно съхранение на пароли

4.3. *Диаграми, връзки и взаимодействия в приложението:*

* *Er diagram*

 Разяснения:

Базата данни се състой от осем таблици.

* **medicines**

Съдържа номенклатура на разрешените за употреба лекарства в България(над 24000 записа).

Полетата **identifier**(номер на разрешително) и **trade\_name** заедно са уникални за всеки запис.

* **users**

Съдържа информация за потребителите, които имат право да работят с приложението.

* **suppliers**

Съдържа информация за доставчиците на лекарствени средства.

* **inventory**

Съдържа информация за наличните количества в склада, сериен номер на партида, срок на годност, доставна цена и продажна цена.

Поле **medicine\_id** референция към **id** на дадено лекарство.

Едно лекарство може да има няколко различни наличности според номер ана партидата.

* **sales**

Съдържа информация за продажбите дата, сума, сума + ДДС.

Поле **user\_id** референция към **id** на даден потребител извършил продажбата.

* **invoices**

Съдържа информация за фактурите .

Поле **user\_id** референция към **id** на даден потребител заприходил стоката.

Поле **supplier\_id** референция към **id** на даден доставчик.

* **invoice\_inventories**

Поле **invoice\_id** референция към **id** на дадена фактура.

Поле **inventory\_id** референция към **id** на дадено количество в склада.

Тази таблица съдържа информация за всички заприходени стоки.

* **sale\_inventories**

Поле **sale\_id** референция към **id** на дадена продажбата.

Поле **inventory\_id** референция към **id** на ддено налично количество в склад.

Тази таблица съдържа информация за всички извършени продажбите

При заприхождаване на стока се извършва свързана транзакция с възможност за отмяна. Едновременно се създава фактура, складова наличност и се запазва референция към фактура, която съхранява в база данни заприходеното количество.

При продажба също се извършва свързана транзакция с възможност за отмяна.

Едновременно се създава продажба, изважда се количеството на продажбата от складовата наличност и се създава референция към продажба , която съхранява в база данни продаденото количество.

4.4. Модули, Класове и методи*:*

* **main.py:**

Mодулът представлява основния стартов файл на приложението. Този модул се използва за настройка и стартиране на графичния потребителски интерфейс(GUI), връзката между изгледа(view) и контролера(controller), както и инициализацията на моделите.

* **db.py**

Mодулът представлява инициализация на SQLAlchemy.

Създава базов клас **Base** – ще се използва за декларативно дефиниране на моделите(таблиците), основна структура, която ще се наследява от другите класове

Създава обект от тип **Engine,** който представлява връзка към SQLite база дaнни

Създава обект от тип **Session**, който се използва за управление на операцииете(заявки и транзакции)

* **utils.py**

Модулът съдържа клас **WindowUtils**, който се използва за центриране на прозорците в средата на екрана.

Основният метод е **center\_window**, който изчислява позицията на която трябва да се позиционира прозореца и настройва геометрията на прозореца.

* **models.py**

Съдържа различни класове, който са дефинирани в базата данни и служат за взаимодействие с нея.

**SessionManager -** Този клас използва Singleton pattern за управление сесиите с базата данни чрез SQLAlchemy.

Има методи:

**get\_session(cls)** – ако няма създадена инстанция създава нова, ако съществува вече инстанция връща тази инстанция.

**close\_session(cls)** – затваря връзката с базата данни.

**create\_table(self)** – създава всички таблици в базата, който не са създадени вече.

**Medicine** – Този клас е модел за лекарствените средства. Създава таблица **medicines** в базата данни.

Има следните методи:инженторът

**extract\_and\_insert\_data(self)** – Този метод взима от външен източник ексел файл **IAL\_Register\_07\_2023.xls** информация за лекарствени продукти и ги записва в базата данни

**search\_medicine(self, search\_term, search\_criteria)** – търси в базата данни по зададен критерии име или id. По име връща всички записи който започват с даден текст или ако използваме %АБ – връща всички имена, които съдържат АБ, резултата връща всички съществуващи лекарства без значение от количеството.

**search\_medicine\_only\_in\_stock(self, search\_term)** – Връща резултат по име или id за всички лекарства който съдържат количества по големи от нула.

**Inventory -** Този клас представлява модел за наличността на лекарствата. Създава таблица **inventory** в базата данни, съдържа конструктор.

**Supplier** - Този клас представлява модел за доставчици, създава таблица **suppliers**, съдържа конструктор и метод **create\_fake\_suppliers(cls, num\_suppliers=5) –** Създава доставчици от списък с произволни имена.

**find\_supplier\_id\_by\_name(self, name) -** Приема име и връща id на доставчик.

**get\_all\_supplier\_names(self) –** връща списък с имената на всички доставчици.

**Invoice** - Този клас представлява модел за фактурите. Създава таблица **invoices** в базата данни, има конструктор и метод:

**create\_invoice(self, invoice\_data, get\_invoice\_fields, user\_id)–** Този метод се използва за създаване на нова фактура в базата , записва количествата в **Inventory** и **InvoiceInventories** като в **Inventory** проверява ако съществува партида със същият номер добавя количество, ако не създава нова наличност с новият номер. Ако се случи грешка в процеса на създаване, транзакцията се отменя.

**InvoiceInventories** - Този клас представлява модел с референция към наличност и фактура, създава таблица **invoice\_inventories**, пази своя отделна наличност на заприходено количество, което не се променя.

**SaleInventory** - Този клас представлява модел с референция към наличност и продажба, създава таблица **sale\_inventories**, пази своя отделна наличност на продаденото количество, което не се променя.

**Sale** - Този клас представлява модел за продажба на лекарства. Създава таблица **sales** в базата данни, има конструктор и метод:

**create\_sale(self, sale\_data, sum\_no\_vat, vat, sum\_vat, user\_id) –** Служи за създаване на запис за продажба в базата данни. В Inventory намалява количеството спрямо избраната партида от продажби и записва в **sale\_inventories** количеството на продажбата**,** което служи за архив на продаденото количество от медикамента. Ако се случи грешка в процеса на създаване, транзакцията се отменя.

**User** - Този клас представлява модел за потребителите на системата. Създава таблица **users** в базата данни, има конструктор и метод:

**create\_user(self, name, password, is\_admin=False) –** Създава два потребителя – администратор и обикновен потребител.

* + - **controller.py**

Модулът съдържа клас **Controller**, който се използва за обработка на заявките от потребителите и координиране на данните между моделите и изгледа. Съдържа следните методи:

**\_\_init\_\_(self, view=None, \*\*models**) – Конструктора приема параметрири за изгледа(view) и няколко модела, които се подават като ключ стойност – с цел да се постигне dependency injection и да не се създават моделите в контролера, а да бъдат подавани отвън.

**\_update\_total\_sales(self, quantity, price)** – вътрешен метод част от логиката за изчисляване на общата сума на продажбите, създаден с цел да се избегне повторение.

**create\_database(self)** - Този метод се извиква при стартирането на приложението и се уверява, че базата данни е създадена и попълнена с начални данни, ако тя е празна, прдотвратява повторна запълване на базата с данни.

**login(self, workplace, username, password)** – Проверява, използвайки **bcrypt** потребителското име и парола на потребителя и ако са валидни, позволява на потребителя да влезе в системата.

**show\_nomenclature(self, search\_term, search\_criteria)** – Извлича от модела информация за медикаментите според определени критерии за търсена и подава тази информация на изгледа, който е визуализира.

**calculate\_total(self, invoice\_data)** – Изчислява общата сума на фактурата с ДДС и е подава на изгледа(**view**), който е визуализира.

**calculate\_customer\_price(self, delivery\_price)** – Изчислява цена на клиента за определен продукт.

**add\_total(self, price, quantity)** – добавя цената на отделните позиции на фактурата към общата сума.

**recalculate\_total(self)** – Ако има редакция на отделните редове на фактурата преизчислява сумата.

**add\_medicine\_to\_invoice(self, medicine\_id, name) -** Викадруги два метода **update\_name\_entry** и **cleanup\_view**, и присвоява стойност на променлива в изгледа.

**update\_name\_entry(self, name)** – Актуализира изгледа с името на продукта.

**cleanup\_view(self)** – премахва текущия изглед(номенклатура).

**update\_selected\_data(self, selected\_data)** – Обновява данните за избрания продукт при фактуриране по време на редакция.

**humanize\_text(self, label) –** Премахва snake case синтаксиса и прави думите да започват с главна буква.

**save\_invoice(self, get\_invoice\_fields, invoice\_data) -** Запазва фактурата и връща id на фактурата.

**save\_sale(self, sale\_data, sum\_no\_vat, vat, sum\_va) –** Запазва продажбата в базата данни.

**validate\_numeric\_input(self, value) –** Проверява дали въведената стойност е числова.

**is\_valid\_date(self, date\_string) –** Проверява дали въведената дата е в определен формат.

**get\_suppliers(self) –** Връща списък с имената на всички доставчици.

**change\_supplier\_name\_to\_id(self, invoice\_data) –** Получава данни за фактурата и превръща името на доставчика в съответстващото му id.

**search\_product\_for\_sale(self, searched\_product\_for\_sale) –** Извършва търсене на продукт за продажба, които получава от модела, като филтрира данните само с нужната информация(лекарство и свързаните с него наличности в инвентара, създава речник с ключ - id на лекарство и стойност – всички записи от инвентара който принадлежат към това лекарство).

**lock\_product(self, product\_id) –** Проверява дали даден продукт е заключен.

**calculate\_sales\_total(self, quantity, price) –** Изчислява общата сума на продажбите след добавяне на нов продукт.

**recalculate\_sale\_sum(self, quantity, price) –** Преизчислява общата сума след изтриване на продукт от кошницата за продажби.

**sell\_sum\_for\_invoice(self) –** Изчислява сума на продажбата без ДДС, ДДС, ума на продажбата с ДДС.

* **view.py**

Модулът съдържа клас View, който отговаря за създаване на интерфейса и стартиране на приложението. Има следните методи:

**def \_\_init\_\_(self, master, controller=None)** – Приема главния прозорец(master) и контролера. Настройва стиловете на приложението и инициализира няколко променливи.

Извиква **load\_login\_window()** и **load\_widgets()**

**load\_login\_window(self) - с**тартира Toplevel прозорец за вписване на потребителя, основният прозорец се стартира в withdraw режим, при затваряне на прозореца от потребителя чрез **X** бутона се стартира **on\_login\_window\_close(),** или се извиква **on\_login()** при вписване

**create\_entry(self, parent, placeholder, \*\*kwargs)** – Създава входа за формата за вписване

**clear\_placeholder(self, event)** – изчиства placeholder от формите .

**restore\_placeholder(self, event)** възстановява placeholder от формите .

**on\_login(self) –** Взима входа на потребителя и ги препраща към контролера за верификация.

**on\_login\_window\_close(self) –** затваря прозореца за вписване и основния прозорец.

**load\_widgets(self) –** Инициализира **Notebook** компонент и извиква . **load\_menu()**

**load\_menu(self) –** Създава меню по динамичен начин. Чрез него се извикват следните методи **set\_superhero\_theme() set\_flatly\_theme()** **log\_out() receive\_inventory() load\_search() sales()**

**add\_submenu(self, parent\_menu, label, submenu\_data) –** Помощен метод за създаване на подменюта.

**add\_menu\_item(self, menu, label, command) –** помощен мето за извикване на командиде от менюто.

**execute\_command(self, command) –** помощен метод, който прехвърля директно командата от менто към команд в контролера със същото име.

**set\_superhero\_theme(self) –** сменя цветовата тема чрез bootstrap.

**set\_flatly\_theme(self) -** сменя цветовата тема чрез bootstrap.

**log\_out(self) –** затваря отворените Notebook елементи зарежда прозореца за вписване, скрива основния прозорец и отписва потребителя.

**load\_search(self)-** Зарежда прозорец за търсене извиква при натискане на бутон **on\_search.**

**on\_search(self) –** Приема входа от потребителя и го подава къмконтролера.

**create\_notebook(self, page\_title, data\_list, page, results\_per\_page)** – Извиква се от контролера и зарежда изгледа за номенклатурата на лекарствата има страници(pagination).

**load\_next\_page(self, current\_page, results\_per\_page, data\_list) –** Извиква се от **create\_notebook** зарежда следваща страница.

**load\_previous\_page(self, current\_page, results\_per\_page, data\_list) -** Извиква се от **create\_notebook** зарежда предишна страница.

**toggle\_label\_visibility(self, label, data) -** Извиква се от **create\_notebook** свива и разпъва видимата част на текста**.**

**close\_notebook(self, frame\_to\_close) –** затваря отвореният Notebook и нулира логически всички свързани с нея променливи, използва се в няколко метода.

**receive\_inventory(self) –** Зарежда полетата от горната част на фактуратас и логиката при тяхното зареждане.

**add\_rows\_to\_inventory(self, entries, canvas, labels) –** Добавя позиции динамично към долната част на фактурата.

**clear\_fields(self, frame\_to\_close)** – презарежда рамката за фактура.

on\_combobox\_select(self, event) – При избор на опция от фактурата търсене по име зарежда **search\_by\_name.**

**search\_by\_name(self) –** Извиква **load\_search** – описана по-горе.

**validate\_entries(self, entries, labels, labels\_to\_validate)** – Проверява дали има празни полета.

**edit\_row(self, row\_index**) и **update\_row() -** Двете са вложени функции и служат за редактиране на редове от фактурата и съответно за оновяване на изгледа след редактиране.

**confirm\_inventory(self, invoice\_labels=None, invoice\_entries=None, calculated=None)** – Задейства метод от контролера, който записва фактурата в базата данни.

**show\_date\_format\_error(self, entry\_value, expected\_format) –** Изкарва съобщение за грешка при невалидно въведена дата.

**sales(self) –** Извиква се за да покаже разделът Продажби в изгледа.

**close\_sales(self) –** извиква се когато потребителя реши да затвори раздела за продажби.

**search\_product(self) –** Извиква се когато потребителя използва функцията за търсене на продукт.

**on\_tree\_select(self, event) -** Извиква се когато потребителя избере продукт от таблицата с продукти.

**open\_result\_window(self, inventory\_medicine) –** Създава нов прозорец който показва всички налични партиди од даден продукт.

**on\_second\_tree\_select(self, event) –** Извиква се когато потребител избере наличност от творения прозорец.

**delete\_from\_second\_tree(self, event) –** Когато потребителя иска да изтрие продукт от долната таблица.

**on\_process(self) –** Извиква се когато потребителя реши да завърши процеса на продажба.

4.5. Употреба на приложението*:*

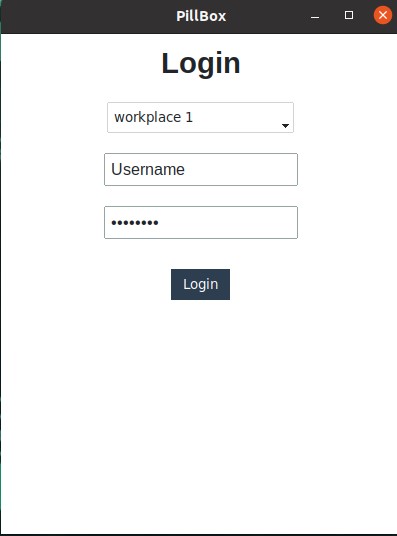
***Инсталиране:***

* Разархивираме файла в избрана от нас папка.
* Навигирам в терминала до създадената папка.
* Създаваме виртуална среда **python -m venv venv**
* Активираме средата **source venv/bin/activate**
* Инсталираме файла с библиотеките и свързаните зависимости **pip install -r requirements.txt**
* Изпълняваме командата **python3 main.py**

**Инструкции за потребителя:**

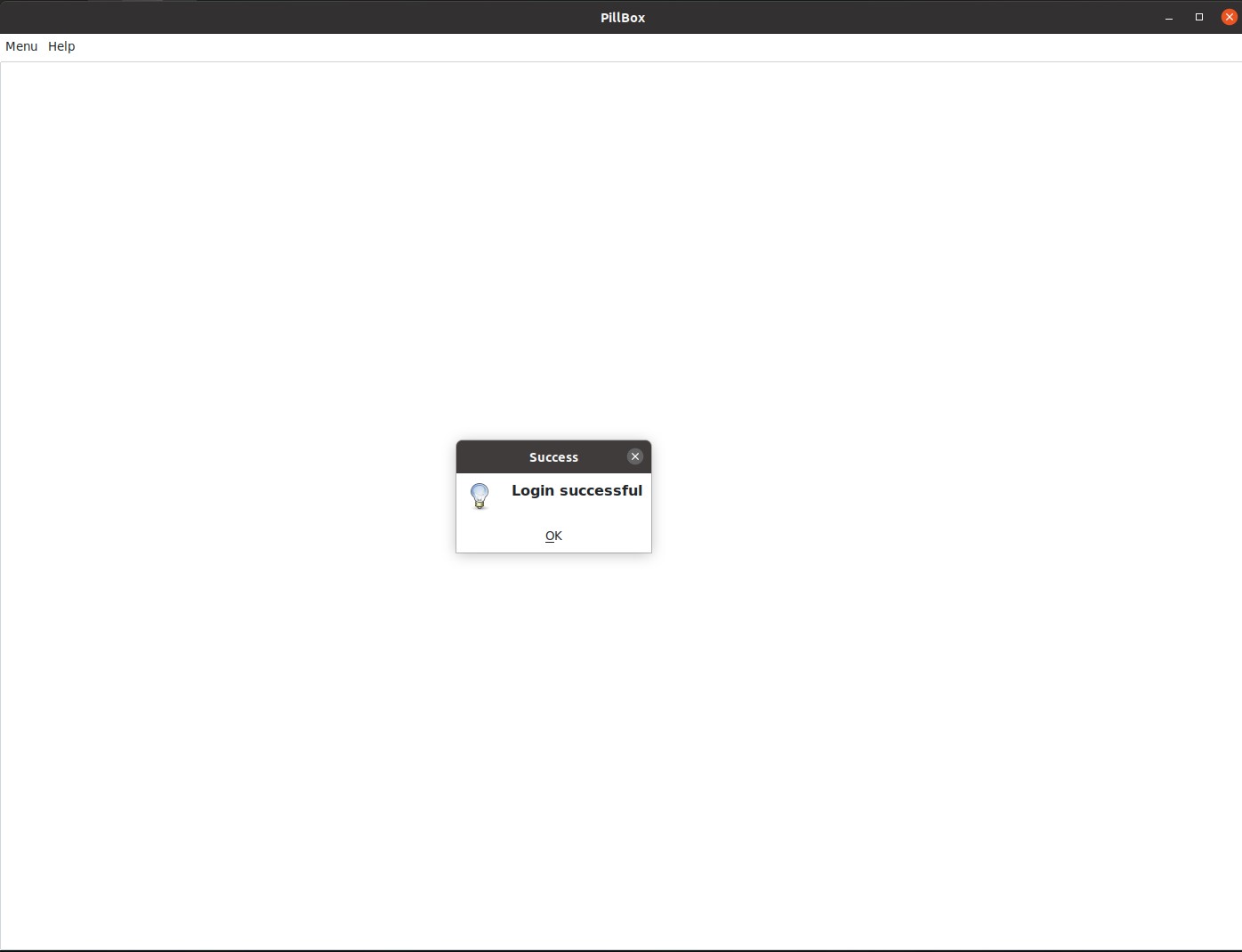
За целите на учебното демо ще бъде показан в инструкциите само щастливия път, без да се засягат случаите, когато се връща грешка към потребителя.

При стартиране, потребителя вижда прозорец за идентификация:



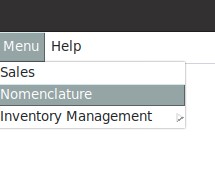
*Избира име, парола и работно място, създадени са потребител с административни права и обикновен потребител (username:admin, pass:admin; username:user, pass:user).*

*След успешна идентификация:*

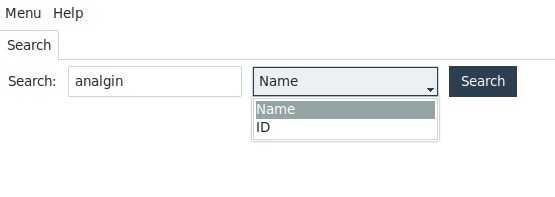
**

***Търсена на лекарство в списъка на ИАЛ.***

*Избираме Menu -> Nomenclature*

**

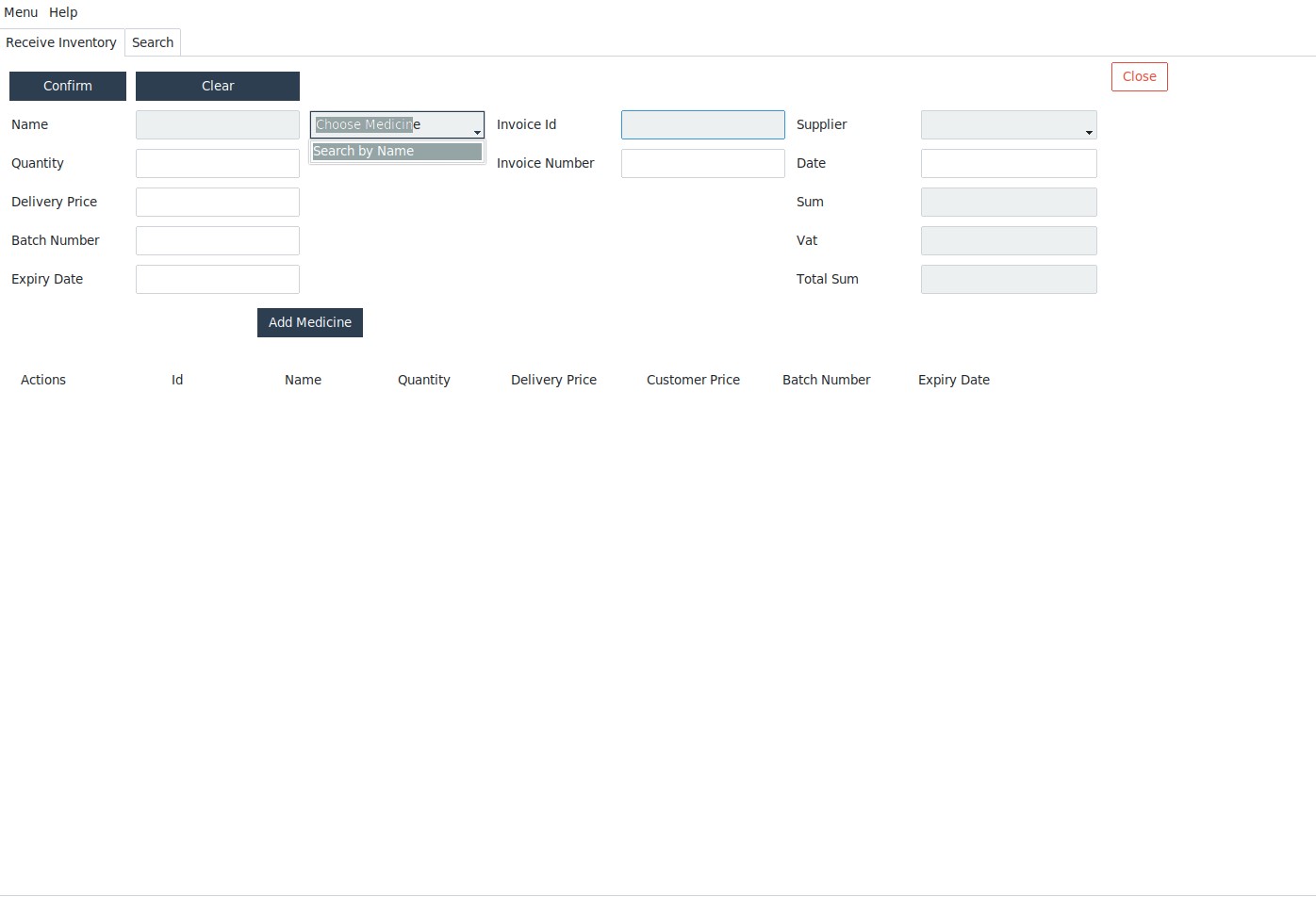
*Отваря се search tab, където може да търсим по имe или id*

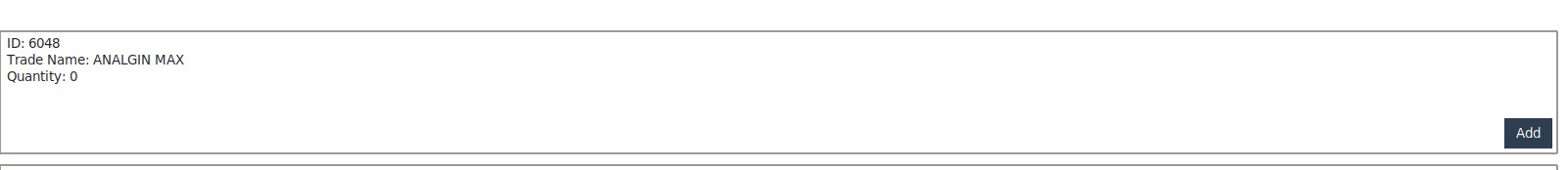
**

*След избор на търсеното от нас лекарство се отваря номенклатурата на лекарствата започващи с въведените букви, ако записите са повече може да навигирам а различни страници за да ги разгледаме може и да търсим по букви независимо къде се намират в името, като ползваме „****%“*** *пред търсените букви. При кликване върху някоя от позициите се показва подробна информация, при повторно кликване се скрива. Бутон close – горе вдясно* *затваря номенклатурата.*

***Заприхождаване на стока:***

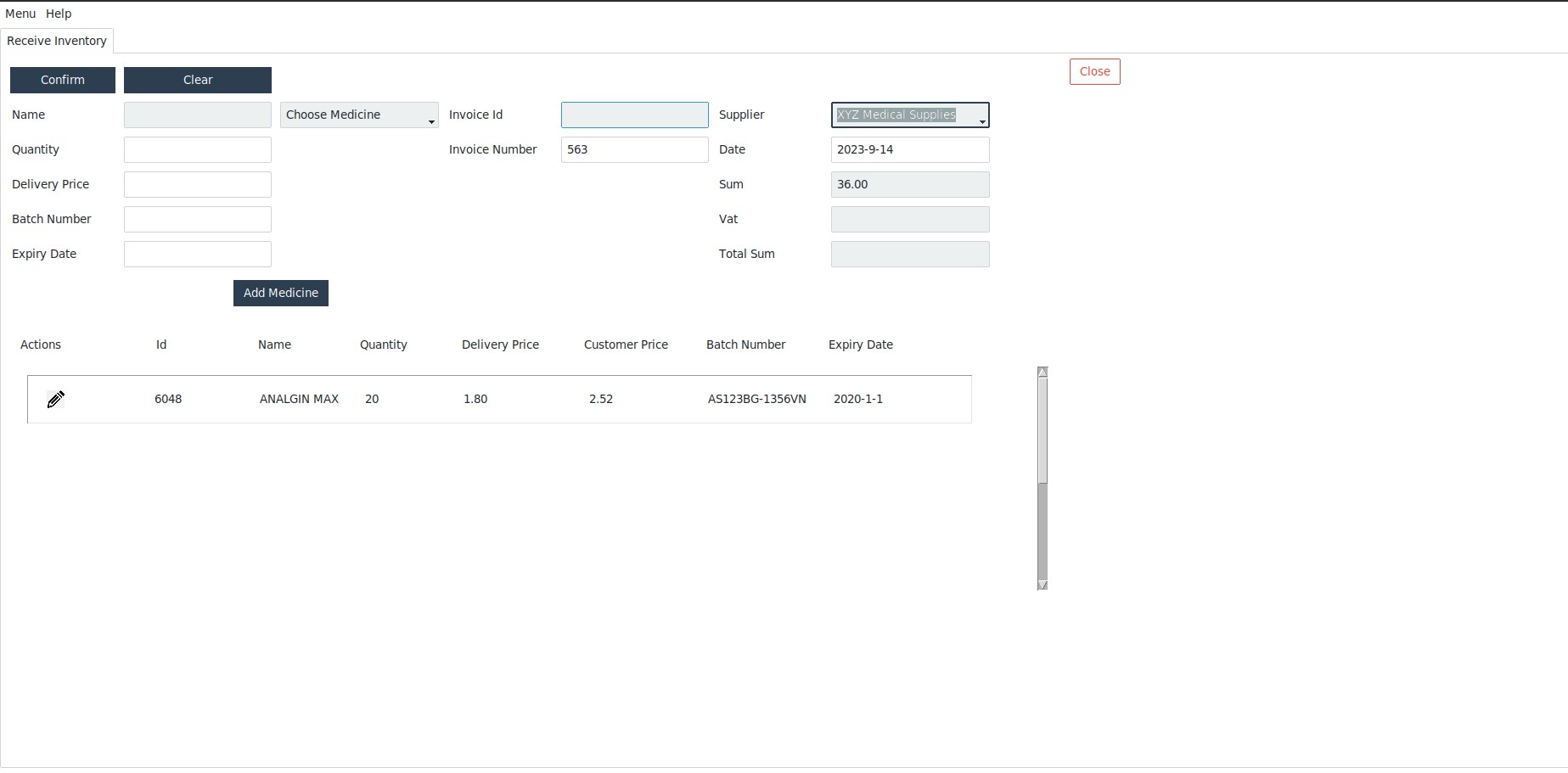
*Избираме Menu -> Inventory Management -> Recieve Inventory*

При избор на **Search by Name** се отваря отново номенклатура но този път записите имат възможност за добавяне чрез бутон **Add.**

****

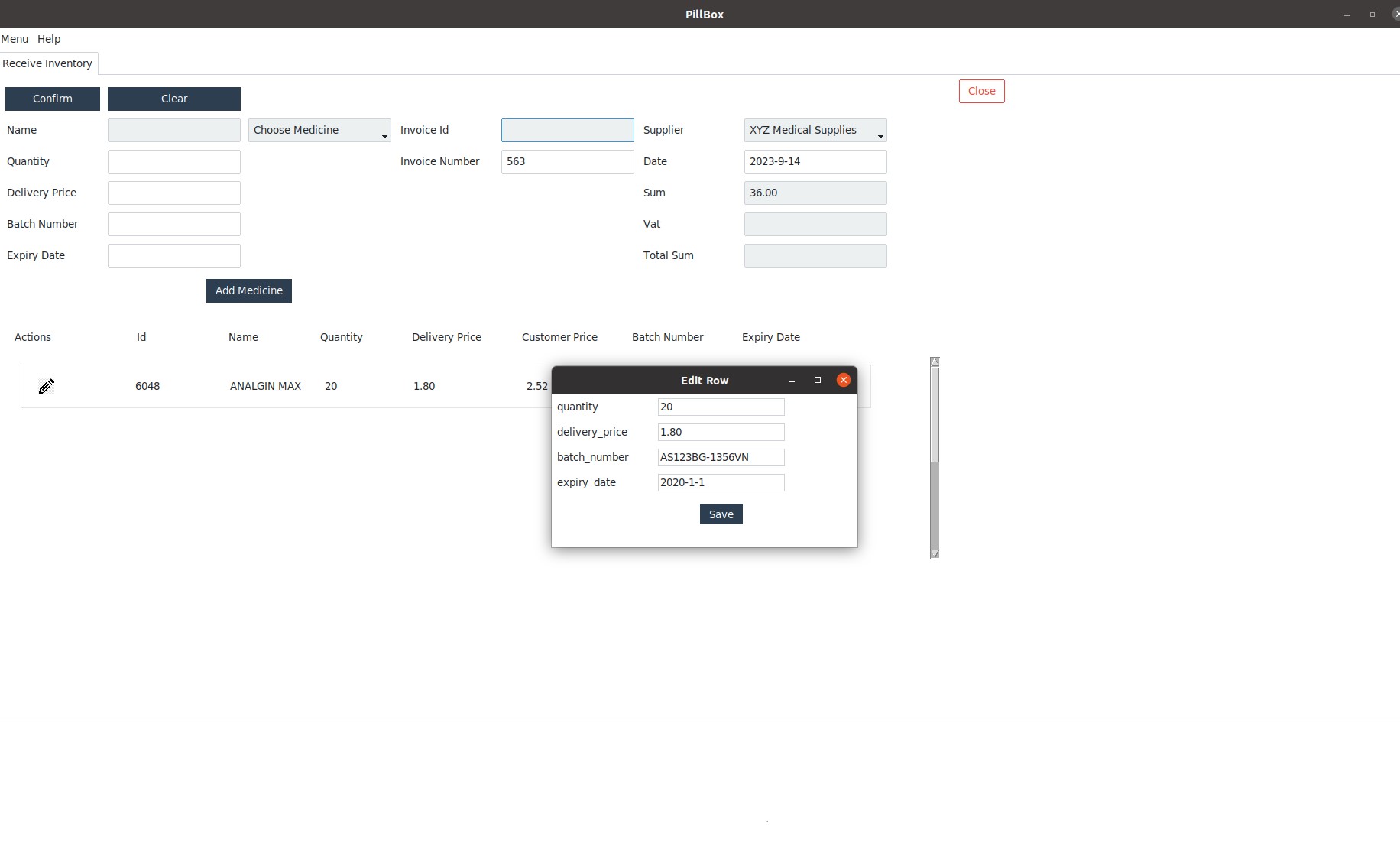
Попълваме нужните полета в лявои натискаме бутон **Add Medicine**

След попълване и на полетата вдясно прозореца има следния изглед:

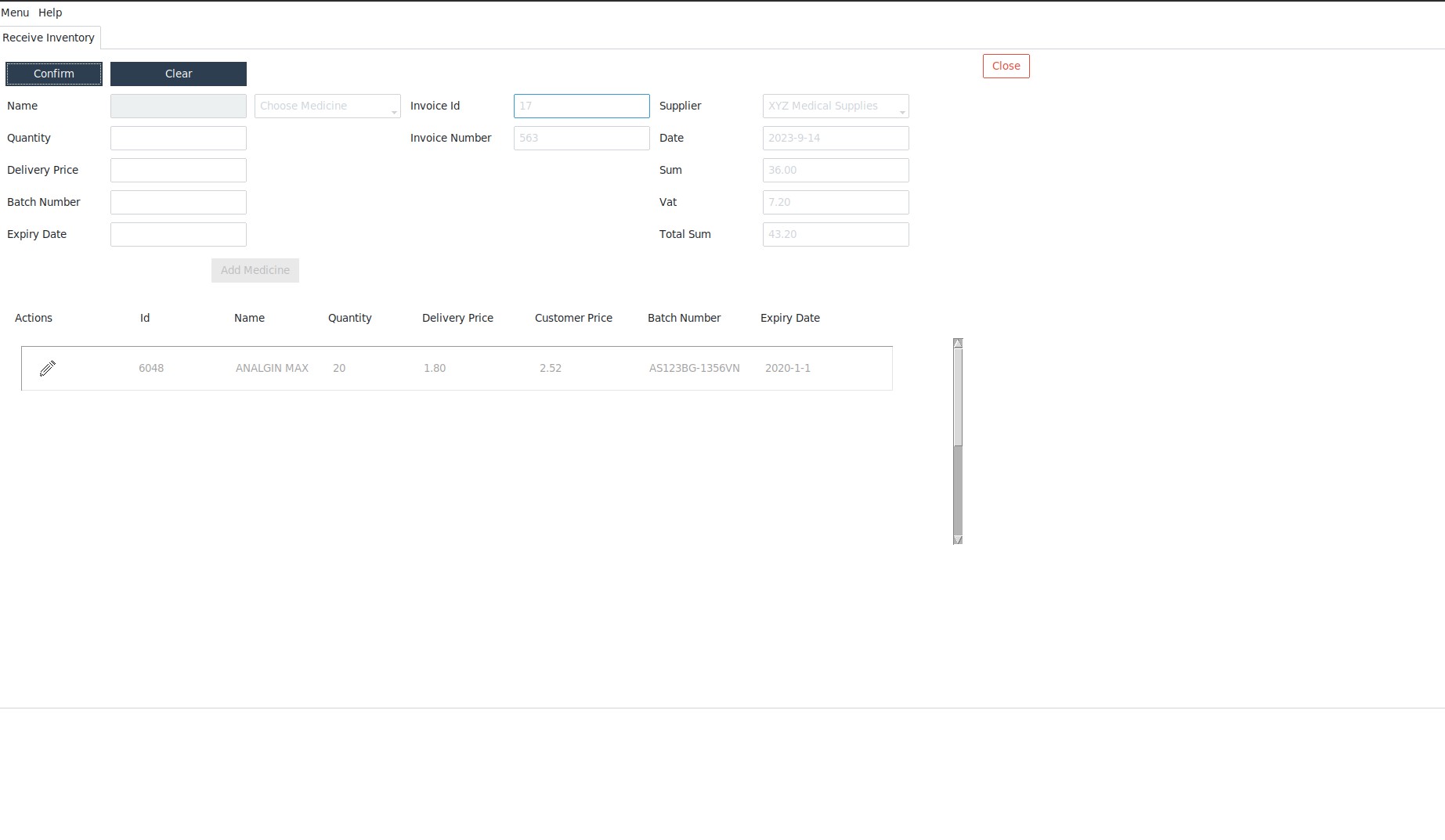
**

Можем да потвърдим или да редактираме позиция от фактурата

**edit – Бутон изглеждащ като писалка:**

**Confirm:**

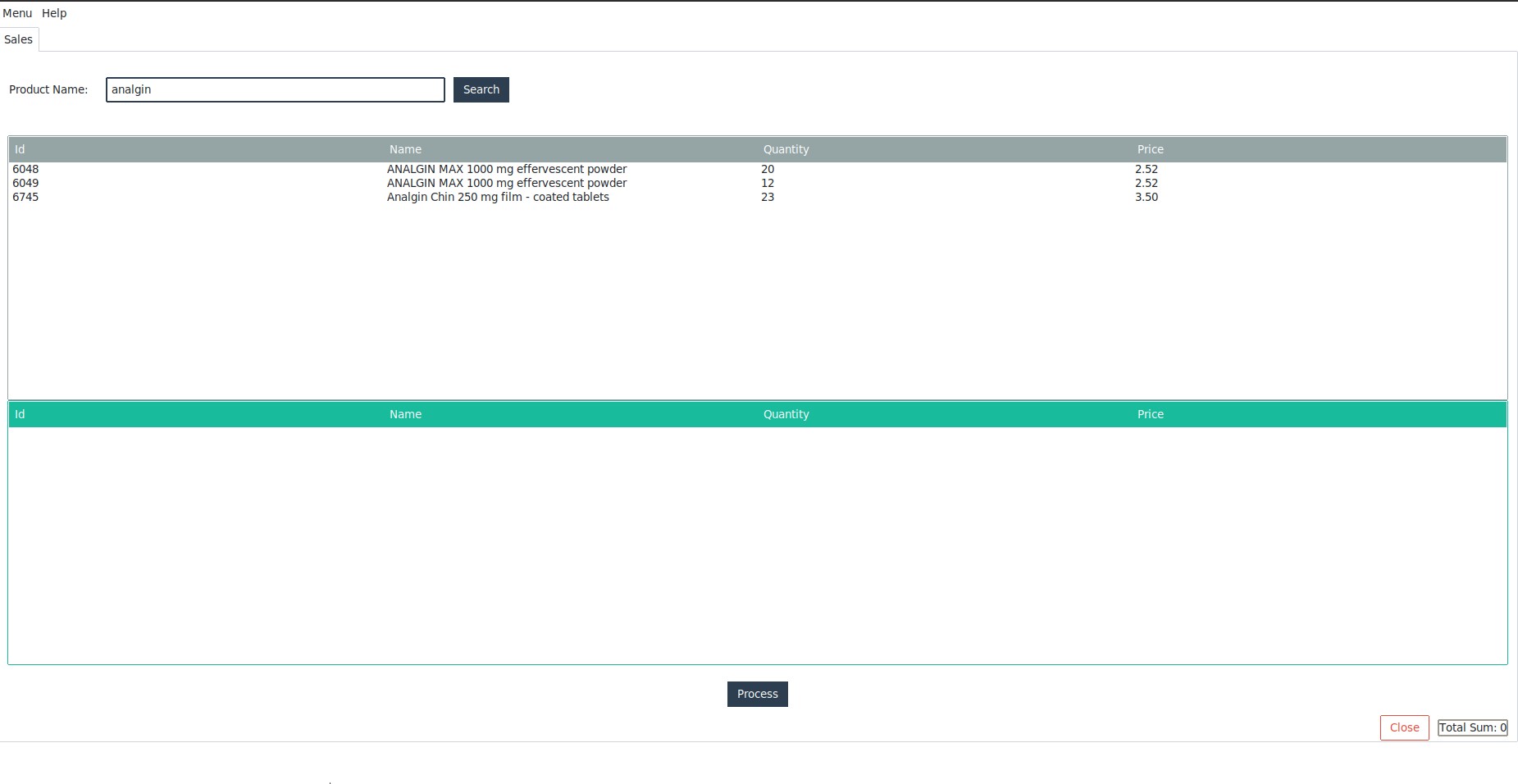
След натискане на **Confirm** лекарствата за заприходени и записани в дата базата.

**

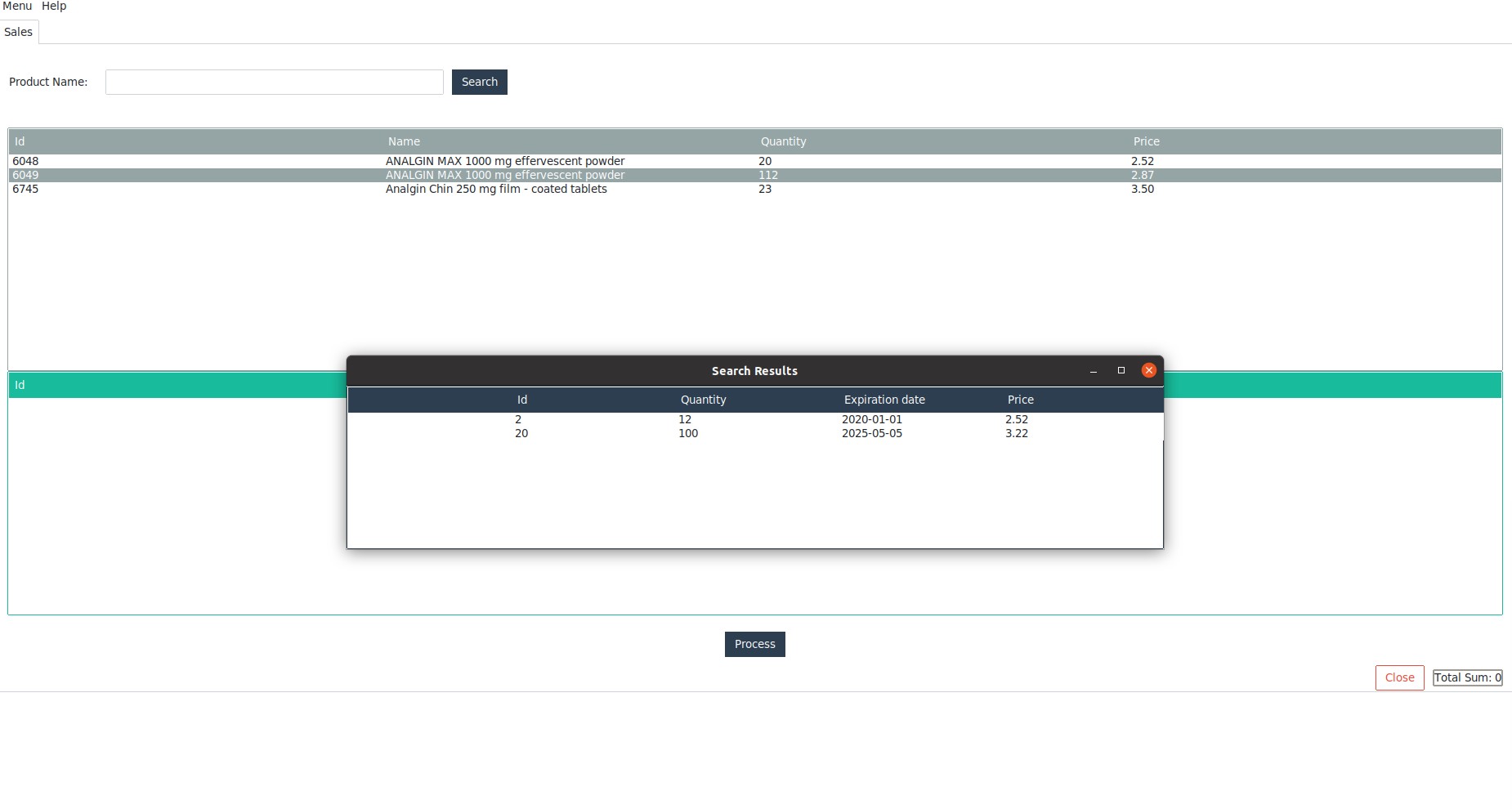
Бутон **Clear** отваря нова празна бланка за фактура а бутон **Close** затваря отворения таб.

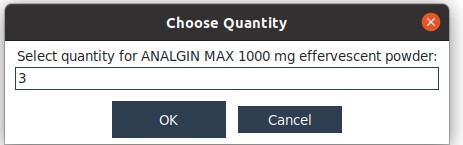
**Продажби:**

*Избираме Menu - > Sales*

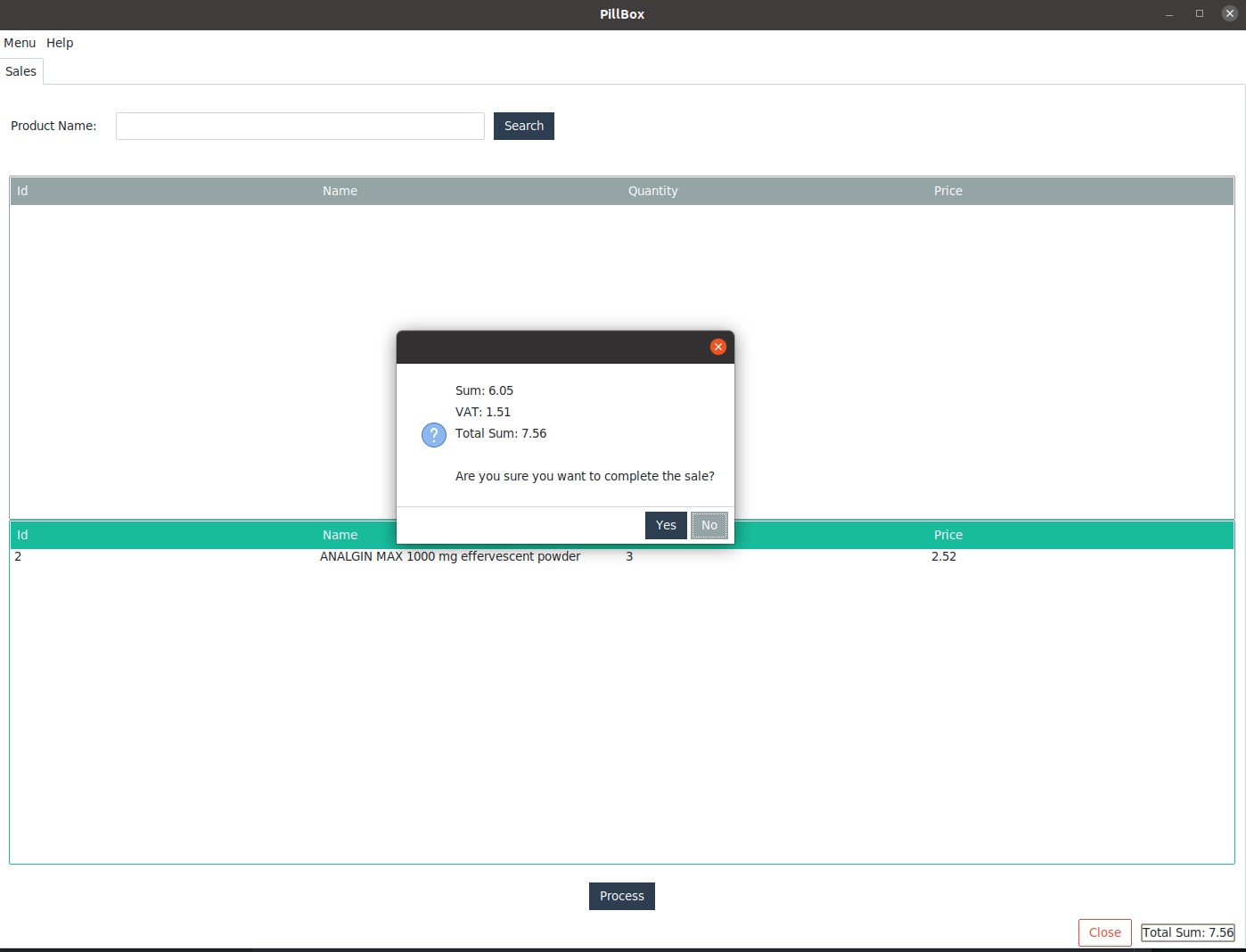
*Отваря се прозорец продажби,където в горната част търсим по име* *или id.*

*При избор на желаната позиция се отваря наличност в склада избираме от коя партида ще извършим продажба.*

*Избираме количество:*

**

*При натискане на Process се извършва продажба като данни за продажбата се запазват в базата данни:*

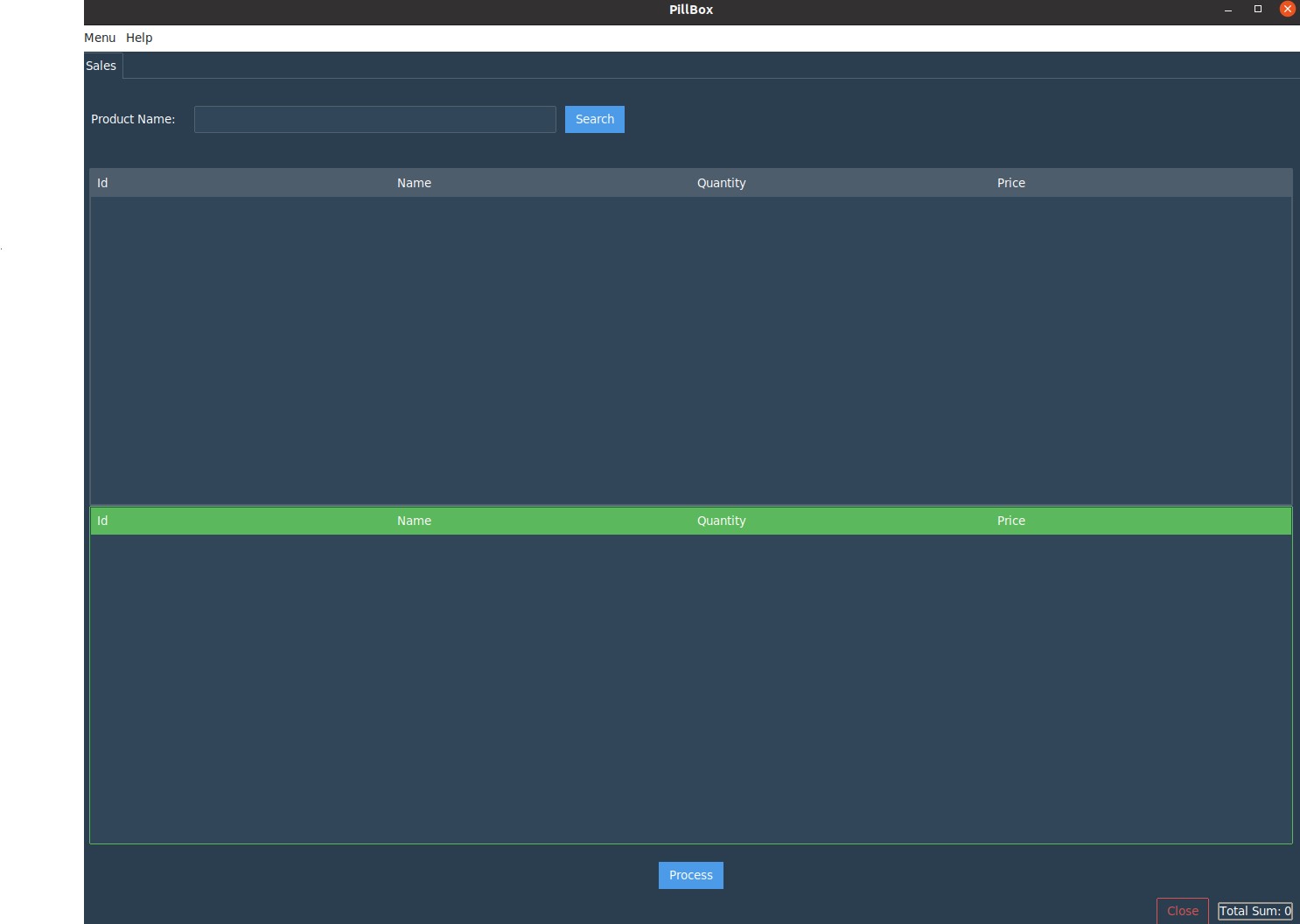
**

*Бутон* ***Close*** *затваря продажбата ако няма избрани позиции в долния прозорец.*

***Total Sum*** *– показва общата сума до момента с ДДС.*

*Help -> Toggle Theme -> Dark Theme*

*Показва тъмна цветова гама:*

**

*Help -> Log Out*

*Отписва от приложението потребителя и връща към екрана за идентификация.*

***Заключение:***

Приложението е с много базова функционалност съобразено с част от изискванията за заприхождаване и продажба на аптечен софтуер.

5. Бъдещо развитие и недостатъци

**Развитие**

Възможностите за развитие са безкрайни ще изброя само част от тях

* Добавяне на функционалност за преглед на фактури.
* Добавяне на функционалност за редактиране на вече заприходени фактури, ако не е извършена продажба от тяхното количество
* Добавяне на функционалност за заприхождаване на стока автоматично (xml файл)
* Добавяне на функционалност за справки по фактури за период.
* Добавяне на функционалност за работа с НЗОК продажби и заприхождаване - водят се отделно от останалите продажби и склад.
* Справка за продажби по потребители.
* Сторниране на продажба.
* Възможност за работа и синхронизация с касов апарат .
* Даване на заявки на база продажби и складова наличност за избран период.

**Недостатъци**

* + Проблем с полето за дата, използва се обикновено поле за потребителски вход в момента.
  + Добавяне на повече и по-ясни съобщения при взаимодействие на потребителя с програмата, когато нямаме щастлив път.
  + Недостатъчно ръчно тестване на функционалността на програмата поради което възможност от поява на непредвидени грешки.