

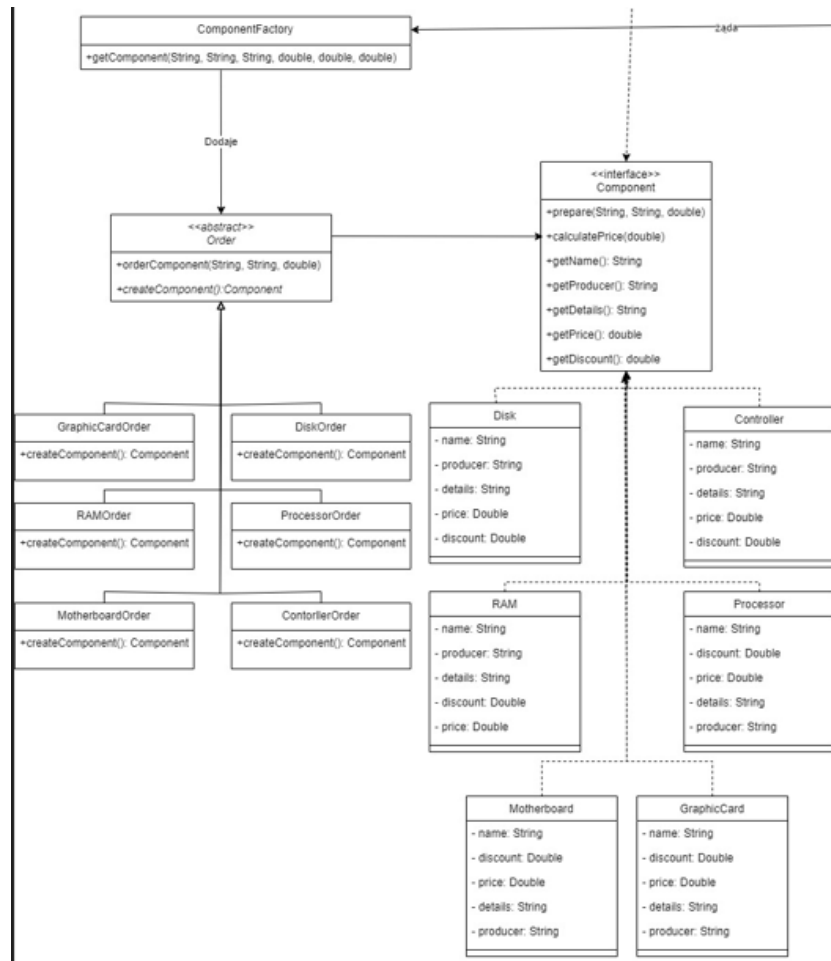
Tematem naszego projektu jest aplikacja, która będzie wspierała zarządzanie biznesem (obszar dokumentacji zamówień) w dziedzinie składania komputerów stacjonarnych. Aplikacja będzie umożliwiała m.in. tworzenie dokumentów sprzedaży, określanie dokładnych danych na temat zamówień, przeglądanie historii zamówień(dokumentów). Aplikacja jest przeznaczona dla małych przedsiębiorstw.

1. Dostęp do bazy przechowującej dane na temat zamówień i ich historii
2. Utworzenie dokumentu potwierdzającego dokonania zakupu(faktura/paragon)
3. Sortowanie dokumentów
4. Podgląd szczegółów w dokumencie

```

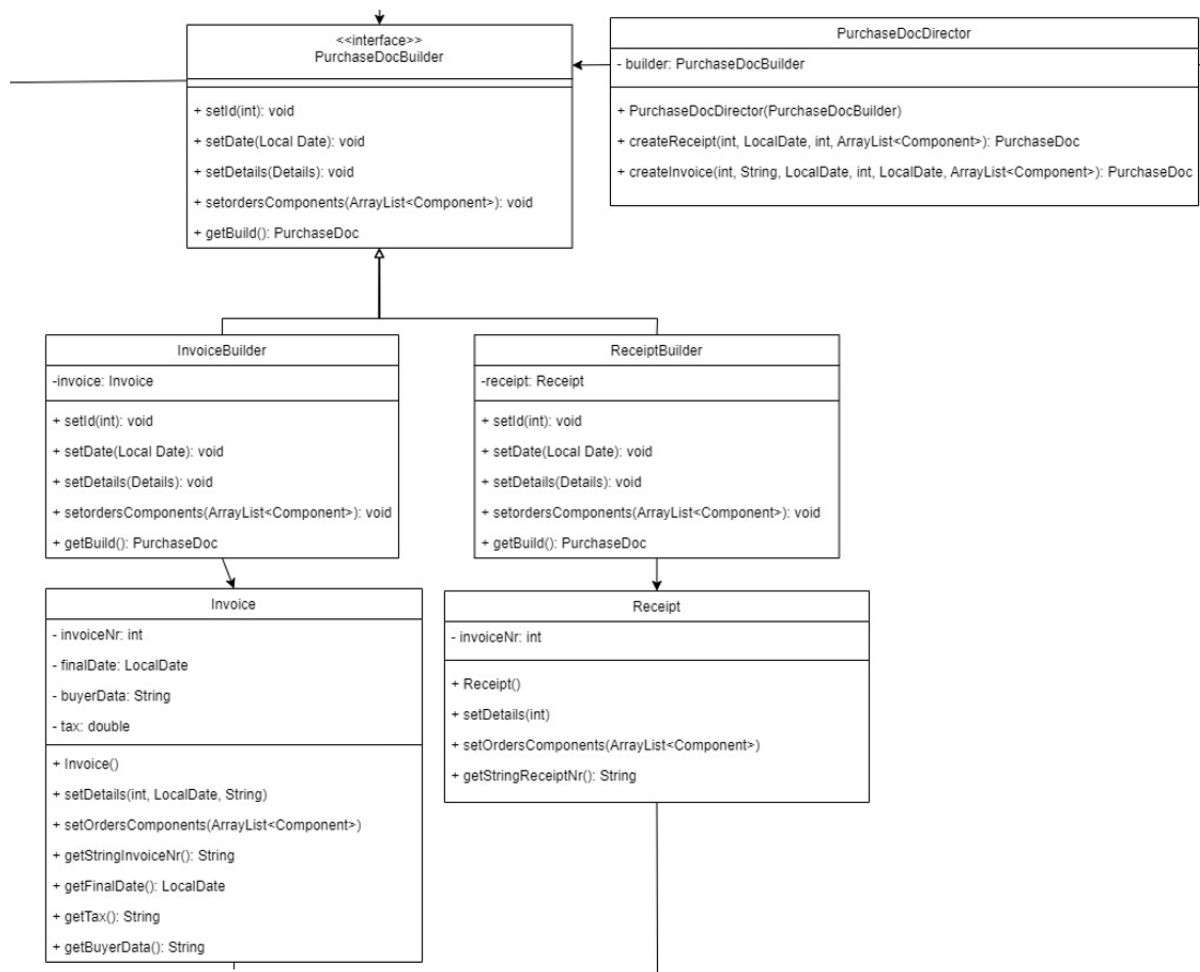
classDiagram
    class SaleDecorator {
        +SaleDecorator(Component)
        +calculatePrice() double
    }
    class CodeDecorator {
        +CodeDecorator(Component)
        +calculatePrice() double
    }
    class PurchaseDocBuilderController {
        +builder PurchaseDocBuilder
        +director PurchaseDocDirector
    }
    class PurchaseDocBuilder {
        +builder PurchaseDocBuilder
    }
    class PurchaseDocDirector {
        +PurchaseDocDirector(PurchaseDocBuilder)
        +createReceipt(LocalDate, int, ArrayList<Component>) PurchaseDoc
        +createInvoice(int, String, LocalDate, int, LocalDate, ArrayList<Component>) PurchaseDoc
    }
    class ComponentFactory {
        +getComponent(String, String, String, double, double, double)
    }
    class Order {
        +getComponent(String, String, String, double)
        +createComponent() Component
    }
    class Disk {
        +name String
        +producer String
        +details String
        +price Double
        +discount Double
    }
    class Controller {
        +name String
        +producer String
        +details String
        +price Double
        +discount Double
    }
    class RAM {
        +name String
        +producer String
        +details String
        +discount Double
        +price Double
    }
    class Processor {
        +name String
        +producer String
        +details String
        +price Double
        +discount Double
    }
    class Motherboard {
        +name String
        +discount Double
        +price Double
        +details String
        +producer String
    }
    class GraphicCard {
        +name String
        +discount Double
        +price Double
        +details String
        +producer String
    }
    class InvoiceBuilder {
        +invoice Invoice
        +setDetails() void
        +setDate(LocalDate) void
        +setDetails() void
        +setDetails() void
        +setDetails() void
        +getBuild() PurchaseDoc
    }
    class ReceiptBuilder {
        +receipt Receipt
        +setDetails() void
        +setDate(LocalDate) void
        +setDetails() void
        +setDetails() void
        +setDetails() void
        +getBuild() PurchaseDoc
    }
    class Invoice {
        +invoice int
        +startDate LocalDate
        +endDate LocalDate
        +tax double
        +items()
        +setDetails(int, LocalDate, String)
        +getItems() ArrayList<Component>
        +getInvoice() String
        +getInvoice() LocalDate
        +getTax() String
        +getInvoice() String
    }
    class Receipt {
        +receipt int
        +receipt()
        +setDetails() void
        +setDetails() void
        +setDetails() void
        +setDetails() void
        +getBuild() PurchaseDoc
    }
    class Defragment {
        +defragment Defragment
        +defragment() void
        +defragment() void
        +defragment() void
        +defragment() void
        +defragment() void
    }
    class Defragment {
        +defragment Defragment
        +defragment() void
        +defragment() void
        +defragment() void
        +defragment() void
        +defragment() void
    }
    class History {
        +doc ArrayList<PurchaseDoc>
        +strategy SortStrategy
        +defragment() void
        +defragment() void
        +defragment() void
        +defragment() void
        +defragment() void
    }
    class SortByDate {
        +sort() void
    }
    class SortByDate {
        +sort() void
    }
    class HistoryController {
        +history History
        +start int
        +end int
        +getHistory() void
        +getHistory() void
        +getHistory() void
    }
    class PurchaseDocController {
        +controller PurchaseDocController
    }
    class PurchaseDocController {
        +doc Receipt
        +doc Invoice
        +orderDate String
        +PurchaseDocController()
        +setPurchaseDoc() void
        +getPurchaseDoc() void
    }
    SaleDecorator --|> CodeDecorator
    PurchaseDocBuilderController --|> PurchaseDocBuilder
    PurchaseDocBuilderController --|> PurchaseDocDirector
    ComponentFactory --|> Order
    Order --|> Disk
    Order --|> Controller
    Order --|> RAM
    Order --|> Processor
    Order --|> Motherboard
    Order --|> GraphicCard
    InvoiceBuilder --|> Invoice
    ReceiptBuilder --|> Receipt
    Defragment --|> Defragment
    History --|> SortByDate
    History --|> SortByDate
    HistoryController --|> History
    PurchaseDocController --|> PurchaseDocController
    PurchaseDocController --|> PurchaseDocController
    
```

- Metoda fabrykująca



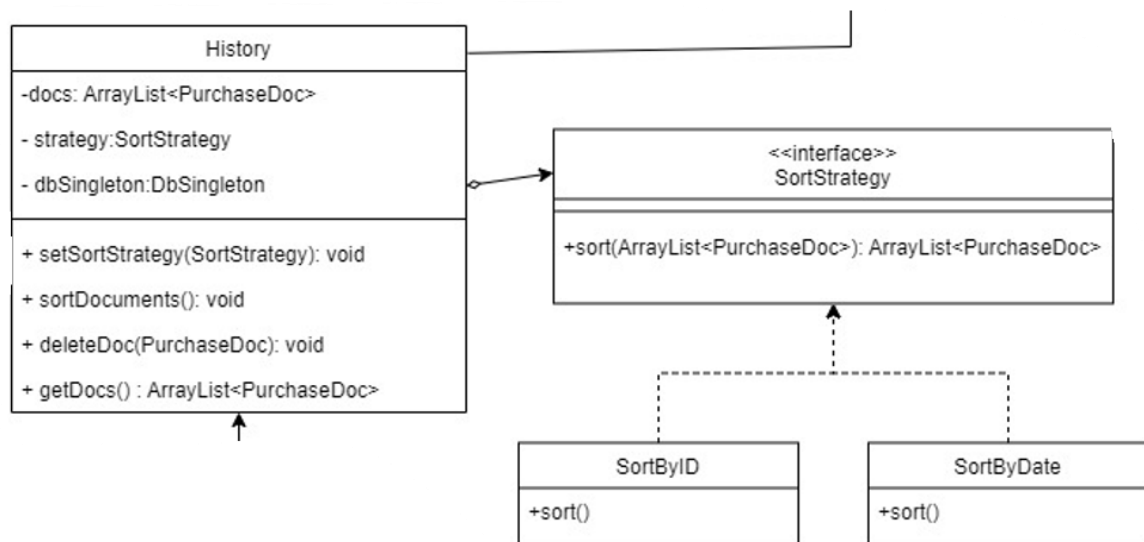
Wzorzec ten został wybrany przez nas do dodawania poszczególnych komponentów do dokumentu zakupu. Metodą fabrykującą w naszym przypadku jest bezparametrowa metoda abstrakcyjna `createComponent()`. Lokalizacja: pakiet `factoryMethod`, wektor zmian: możliwość dodania kolejnych komponentów Orderów do rodziny `Order` oraz implementacje do interfejsu `Component`

- Builder



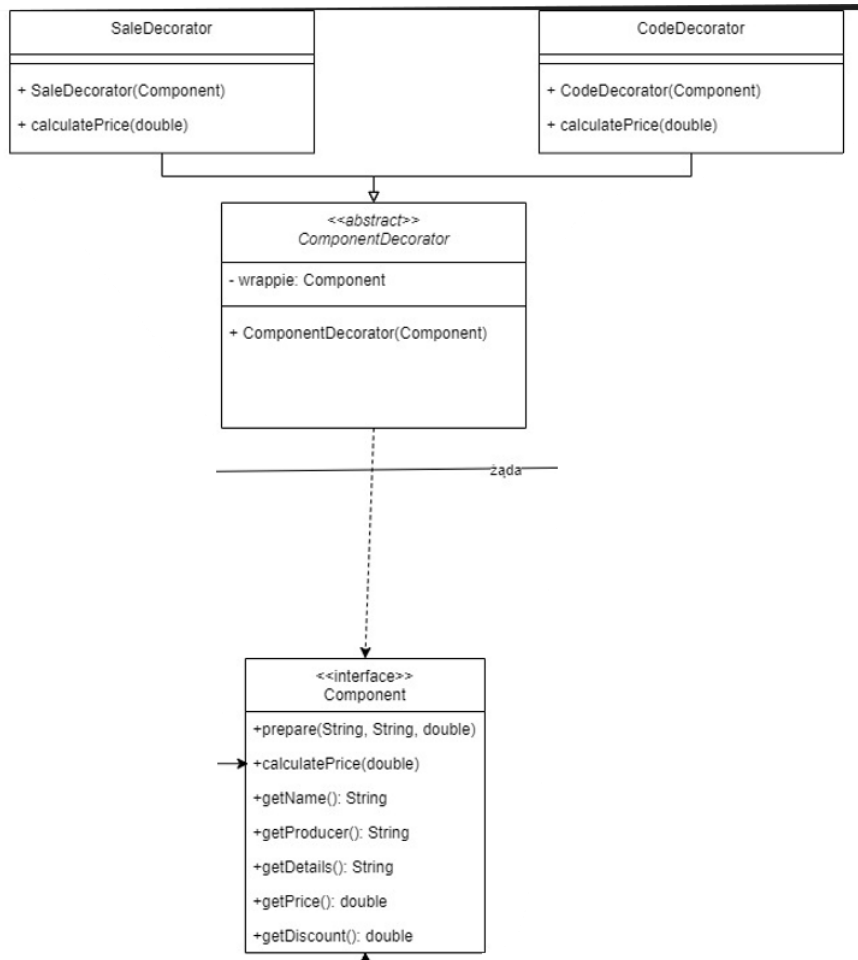
Builder wykorzystamy do tworzenia dokumentu potwierdzającego zakupu danego zamówienia. W zależności od tego czy klient zażyczy sobie fakturę lub paragon taki dokument zostanie utworzony. Klasa, która pełni funkcję director to PurchaseDocDirector. Lokalizacja: pakiet builder, wektor zmian: możliwość dodania nowego builder obok InvoiceBuilder oraz ReceiptBuilder.

- Strategia



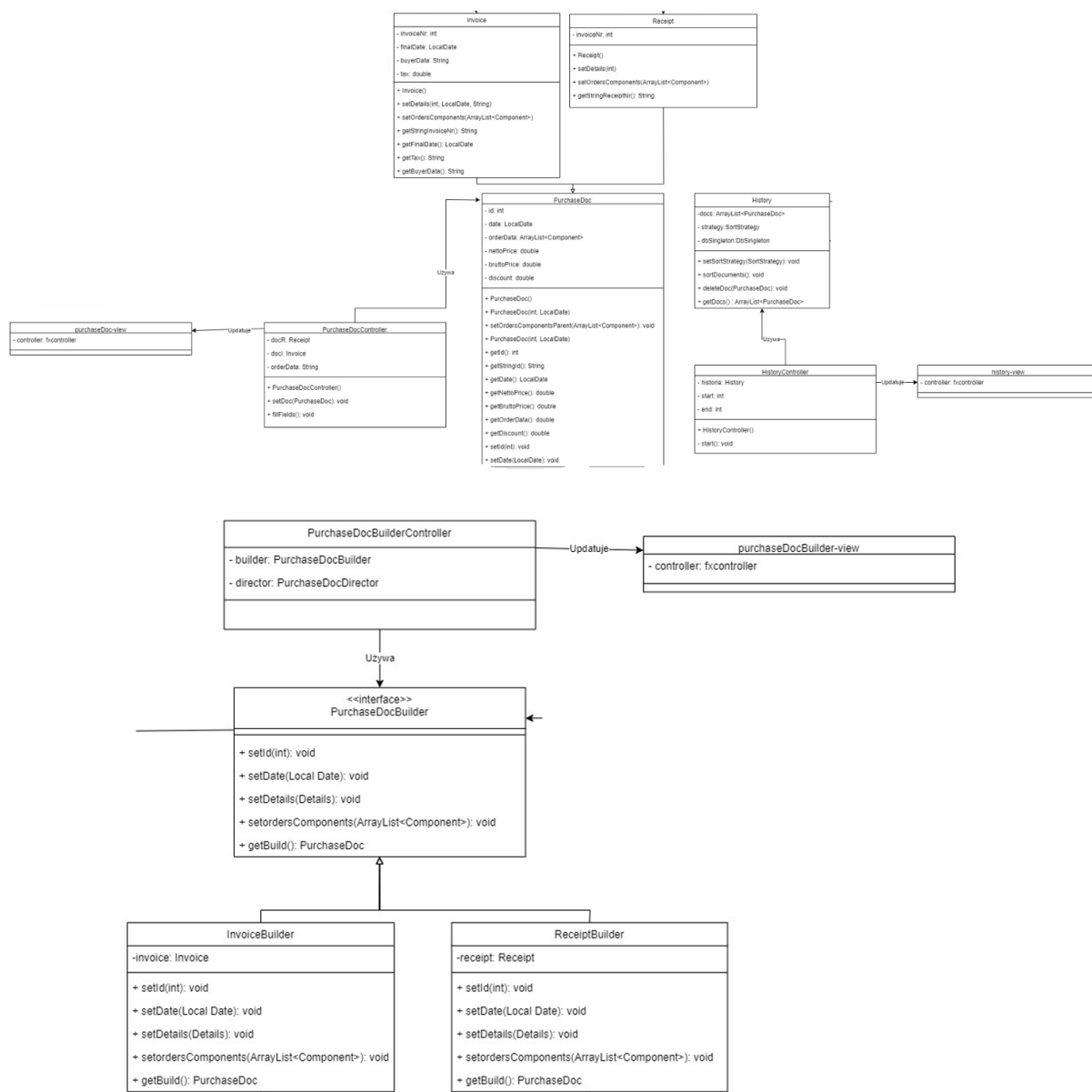
Wzorzec strategii będzie służył nam do odpowiedniego posortowania danych względem danego parametru jak: id dokumentu czy data utworzenia dokumentu. Kontekstem w tym przypadku jest klasa History. Lokalizacja: pakiet strategy, wektor zmian: możliwość dodania nowego sortowania według innego parametru np. po ilości produktów w zamówieniu, dodanie klasy implementacji SortStrategy

- Dekorator



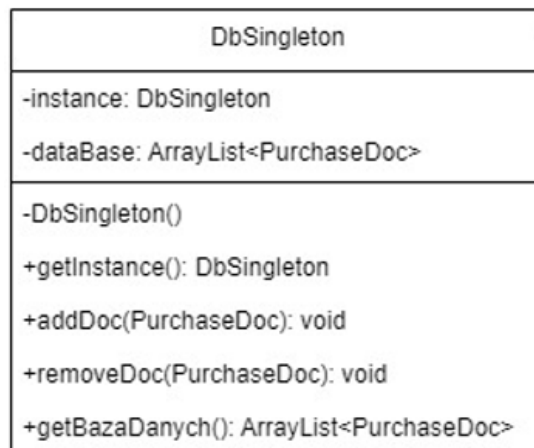
Wzorzec dekoratora będzie służył nam do przydzielania komponentom nowej ceny promocyjnej np. cena wyprzedażowa lub cena uwzględniająca kod rabatowy klienta. Lokalizacja: pakiet decorator, wektor zmian: możliwość dodania innego dekoratora zmieniającego cenę danego komponentu np. Zmiana ceny podając procentową zniżkę.

- MVC



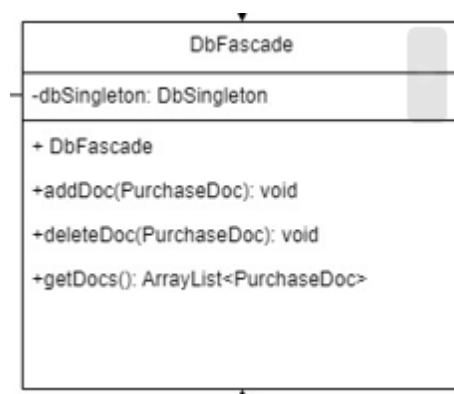
Wzorec MVC wydziela trzy podstawowe warstwy: Model, View, Controller. Naszymi modelami są np. klasy przedstawiające dokumenty zakupu, czyli Invoice i Receipt, które dziedziczą ze wspólnego rodzica PurchaseDoc oraz klasa History reprezentująca historię dokumentów. Lokalizacja: pakiet models, wszystkie controllery przedstawione na screenie powyżej oraz, pliki fxml w katalogu resources, wektor zmian: możliwość rozbudowy o kolejne widoki np. Dodanie widoku/okienka wyświetlającego się przy usuwaniu dokumentu z bazy z zapytaniem czy na pewno chcemy usunąć wybrany dokument.

- Singleton



Singleton będzie zapewniała wystąpienie w aplikacji tylko jednej instancji dostępu do danych dotyczących dokumentów. Lokalizacja: klasa DbSingleton

- Fasada



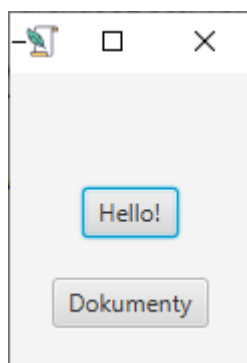
Rolą fasady jest udostępnienie interfejsu do znacznie ułatwionej obsługi bazy danych Singleton. Zamiast wywoływać w całym programie instrukcje operujące bezpośrednio na bazie danych, przygotowana została fasada, która oferuje niezbędne metody do poprawnej komunikacji między aplikacją a bazą danych. Lokalizacja: klasa DbFascade, wektor zmian: możliwość rozszerzenia klasy przy rozszerzaniu możliwości operowania na bazie danych np. Edycja danych.

## Instrukcja instalacji

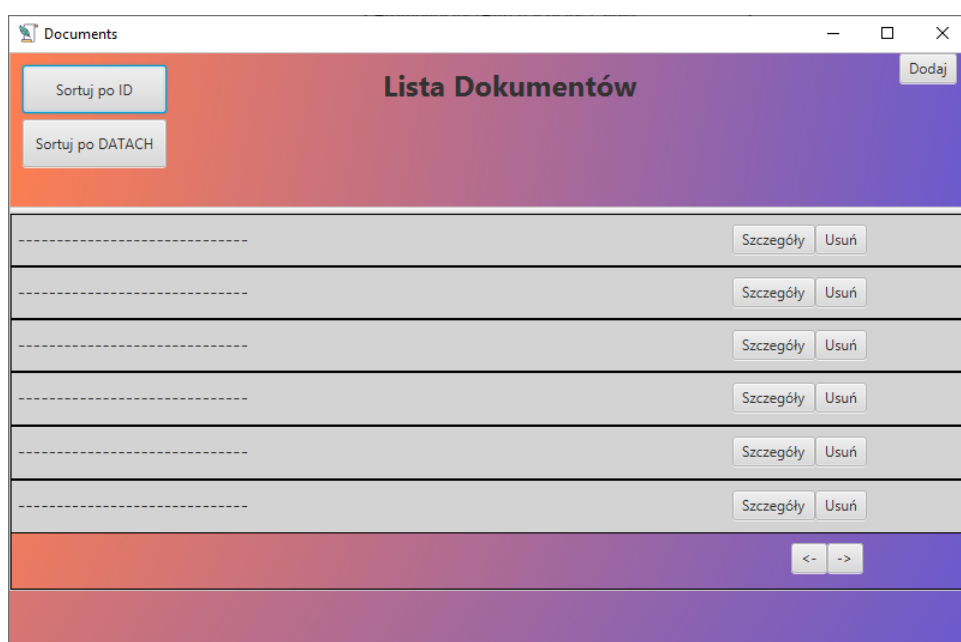
Aby aplikacja poprawnie działała wymagane jest oprogramowanie Java ([www.java.com/pl/](http://www.java.com/pl/)). Należy pobrać repozytorium, rozpakować je oraz za pomocą wybranego narzędzia (np. IntelliJ IDEA) otworzyć projekt.

## Instrukcja użytkownika

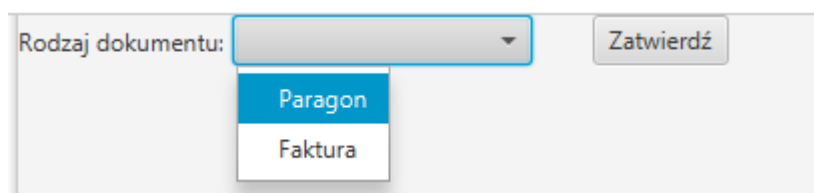
Po uruchomieniu aplikacji na ekranie pojawi się małe okienko na którym powinniśmy wygrać przycisk "Dokumenty"



Następnie przekierowuje nas do głównego widoku na którym możemy przeglądać listę dokumentów. Klikając na jeden z przycisków w lewym górnym rogu możemy sortować listę po wybranym parametrze. Jeżeli chce rozpocząć tworzenie dokumentu należy wybrać przycisk “Dodaj” znajdujący się w prawym górnym rogu.



Gdy pojawi się widok tworzenia dokumentu możemy rozpocząć uzupełniać wymagane pola danymi, rozpoczynając od wyboru typu dokumentu.



Documents

Rodzaj dokumentu: Paragon Zatwierdź

Id dokumentu: 1 Zatwierdź Id paragonu: 1 Zatwierdź

Data zakupu: 29.01.2023 Zatwierdź

Dodaj nowy produkt

Procesor: Zatwierdź AMD Zatwierdź

8 rdzeni Zatwierdź 950 Zatwierdź

5 Dodaj zniżkę rabatową

Dodaj produkt do listy

Zakończ tworzenie dokumentu

Wybierając opcję “Zakończ tworzenie dokumentu”, wychodzimy z obecnego okna i wracamy do listy dokumentów z już no utworzonym dokumentem.

Documents

Sortuj po ID Dodaj

Sortuj po DATA

### Lista Dokumentów

Paragon-> ID: 1 Data: 2023-01-29	Szczegóły	Usuń
-----	Szczegóły	Usuń
-----	Szczegóły	Usuń
-----	Szczegóły	Usuń
-----	Szczegóły	Usuń
-----	Szczegóły	Usuń
<- ->		

Mając dokumenty w liście mamy możliwość podejrzenia szczegółów dokumentu wybierając przycisk “Szczegóły”.



Szczegóły dokumentu

Paragon

Data wystawienia: 2023-01-29  
Data świadczenia usług: -----

Numer dokumentu: 1

SPRZEDAWCA  
ComputerStuff.PL  
ul. Osiedlowa 1  
15-001  
NIP: 987654321  
e-mail: firmaComputerStuff@gmail.com

Netto	VAT	Brutto	Zniżka
2105.95	23%	2735.0	15.0

Processor 945.0  
MotherBoard 895.0  
MotherBoard 895.0

Szczegóły dokumentu

Faktura

Data wystawienia: 2023-01-27  
Data świadczenia usług: 2023-02-01

Numer dokumentu: 1

SPRZEDAWCA  
ComputerStuff.PL  
ul. Osiedlowa 1  
15-001  
NIP: 987654321  
e-mail: firmaComputerStuff@gmail.com

Nabywca  
Firma RoITek Poland

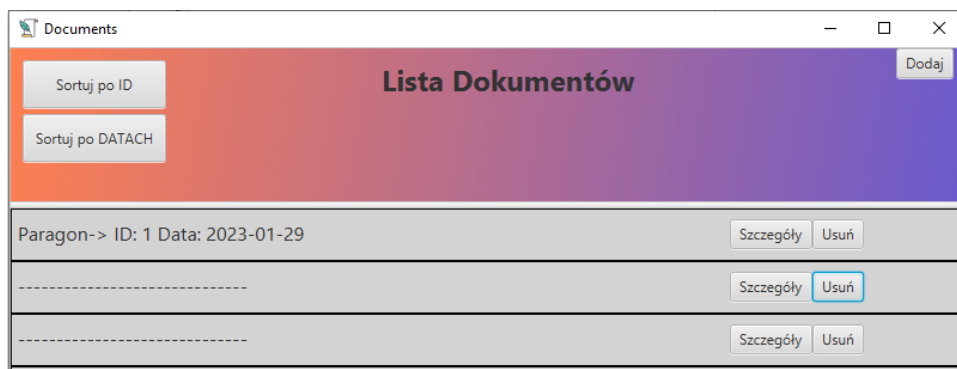
Netto	VAT	Brutto	Zniżka
3773.0	1127.0	4900.0	100.0

Graphic Card 1180.0  
Graphic Card 1180.0  
Graphic Card 1180.0  
Disk 680.0  
Disk 680.0

-----  
Osoba upoważniona do odbioru

-----  
Osoba upoważniona do wystawienia

Istnieje również możliwość usuwania dokumentów z listy wybierając opcję “Usuń”



Przyciski ze strzałkami w prawym dolnym rogu służą do nawigacji pomiędzy stronami listy z dokumentami.



## Opis specyficznego rozwiązania w użytej technologii

Zastosowanie technologii JavaFx umożliwia tworzenie graficznego interfejsu za pomocą wbudowanego scene buildera lub edytora tekstowego.

Link: <https://openjfx.io/>