Politechnika Białostocka

Zaawansowane techniki programistyczne

Dokumentacja projektowa

Przepływ dokumentów w firmie zajmującej się sprzedażą

Autorzy projektu:

Piotr Go

Łukasz Chojecki

Konrad Wasilewski

Studia dzienne

Kierunek: **Informatyka**

Rok: **III**

Semestr: **V**

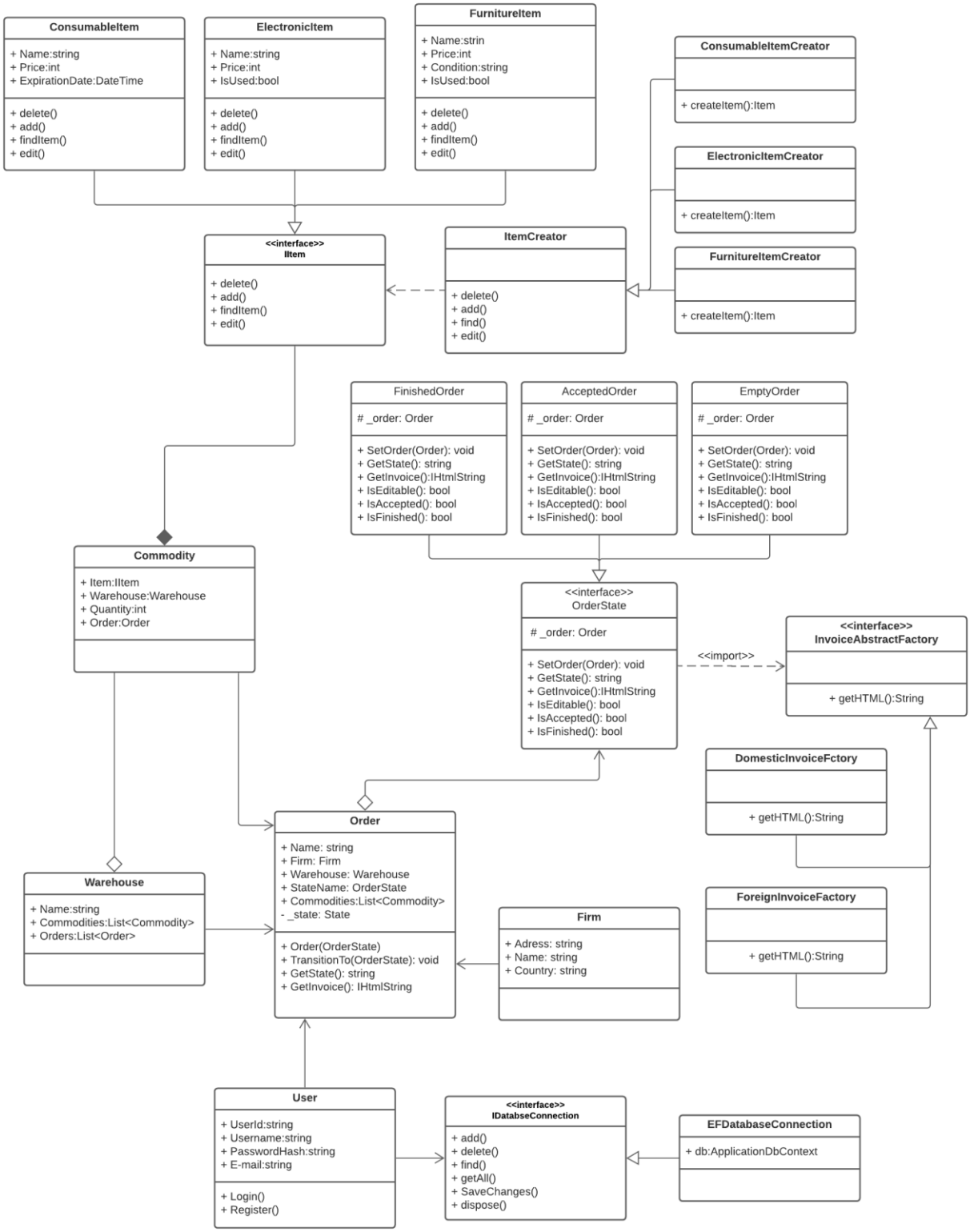
Grupa zajęciowa: **PS 6**

Prowadzący zajęcia: **dr inż. Jerzy Krawczuk**

1. Opis projektu

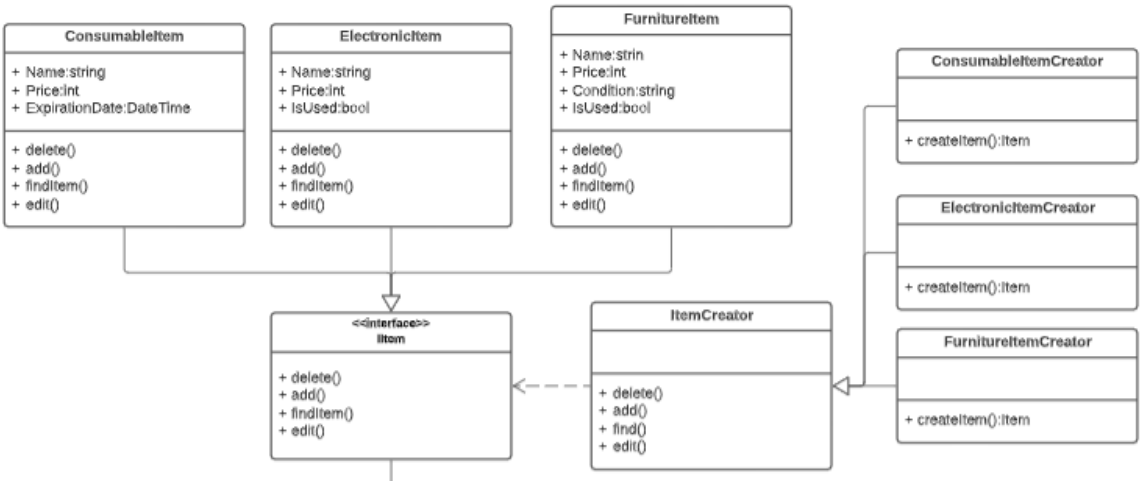
Tematem projektu jest system wspomagający przepływ dokumentów w firmie zajmującej się sprzedażą, aplikacja internetowa pomagać będzie pracownikom firmy w zarządzaniu przedmiotami znajdującymi się w magazynach, składaniu i realizacji zamówień, wystawianiu faktur. Projekt połączony będzie z bazą danych oraz będzie wspierać tworzenie kont i logowanie.

2. Diagram klas



3. Indywidualny opis każdego wzorca

- Metoda wytwórcza



Cel użycia:

Wzorzec ten został wykorzystany w celu ułatwienia tworzenia nowych przedmiotów różnych klas, przedmioty podzielone będą na przedmioty zużywalne (np. jedzenie), elektroniczne oraz na meble. Podklasy kreatora będą umożliwiać zmianę typu tworzonych obiektów.

Przyporządkowanie klas do ról wzorca:

Klasa IItem deklaruje interfejs, który jest wspólny dla obiektów zwracanych przez klasę ItemCreator oraz jej podklasy. Klasy dziedziczące z interfejsu IItem są różnymi implementacjami przedmiotu. Klasa ItemCreator deklaruje metodę wytwórczą, która zwraca nowe obiekty produktów, zawiera także metody działające dla tworzonych obiektów. Klasy dziedziczące z ItemCreator nadpisują bazową metodę wytwórczą, zwraca przez to inne typy obiektów.

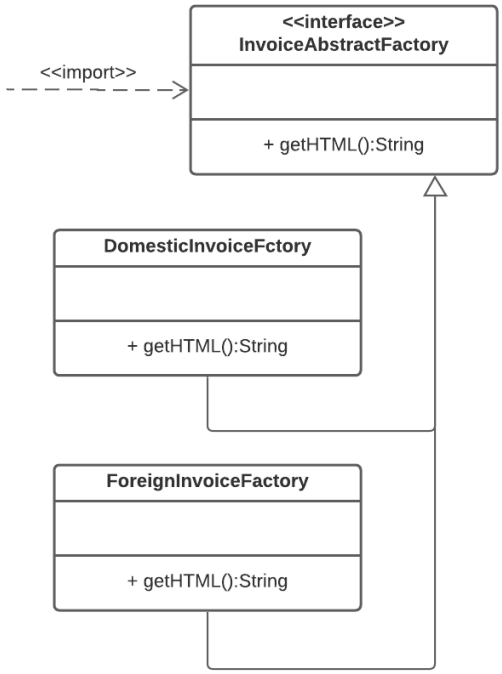
Lokalizacja wzorca w kodzie:

Klasy deklarujące metodę wytwórczą znajdują się w */Logic/FactoryMethod/*. Klasy deklarujące przedmioty znajdują się w */Models/IItem.cs* oraz */Models/ItemViewModels.cs*. Użycie wzorca znajduje się w plikach kontrolerów przedmiotów np. */Controllers/ElectronicItemController.cs*, gdzie wykorzystana jest klasa łącząca która deklaruje statyczne metody działające dla wszystkich obiektów różnych klas przedmiotów (np. linia 33).

Wektor zmian:

Dzięki oddzieleniu kod tworzącego produkty od kodu, który z tych produktów korzysta. Można w łatwiejszy sposób dodawać nowe rodzaje produktów bez integracji w resztę kodu.

- Fabryka abstrakcyjna



Cel użycia:

Fabryka abstrakcyjna wykorzystana jest w projekcie w celu tworzenia 2 typów faktury, lokalnej oraz zagranicznej. Różnicą między nimi będzie przystosowanie czytelności, lokalnie zostaną użyte złotówki i polskie pojęcia a w fakturze użyte zostaną euro i angielskie pojęcia.

Przyporządkowanie klas do ról wzorca:

InvoiceAbstractFactory odpowiada za generowanie kodu HTML przedstawiającego fakturę. DomesticInvoiceFactory jest implementacją odpowiadającą za generowanie faktur przystosowanych dla klientów wewnątrz kraju, ForeignInvoiceFactory natomiast odpowiada za generowanie faktur przystosowanych dla klientów zagranicznych.

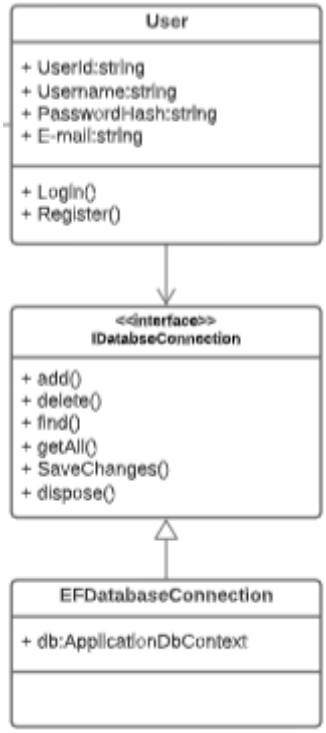
Lokalizacja wzorca w kodzie:

Kod znajduje się w folderze Logic/AbstractFactory w plikach projektu, podobnie wygląda przestrzeń nazw wzorca która wygląda następująco “PrzeplywDokumentowWFirmie.Logic.AbstractFactory”.

Wektor zmian:

Wektorem zmian w tym wzorcu jest klasa Order, wraz ze zmianami w klasie przedstawiającej zamówienie należy zaktualizować generowane dokumenty w celu przedstawienia nowych informacji.

- Fasada



Cel użycia:

Fasada stworzona została, żeby ułatwić obsługę bazy danych. Fasada zawierać będzie metody operujące na bazie danych, dzięki temu nie trzeba będzie w całym programie wykorzystywać instrukcji operujących bezpośrednio na bazie danych.

Przyporządkowanie klas do ról wzorca:

Klasa EFDatabaseConnection implementuje interfejs IDatabaseConnection i jest fasadą.

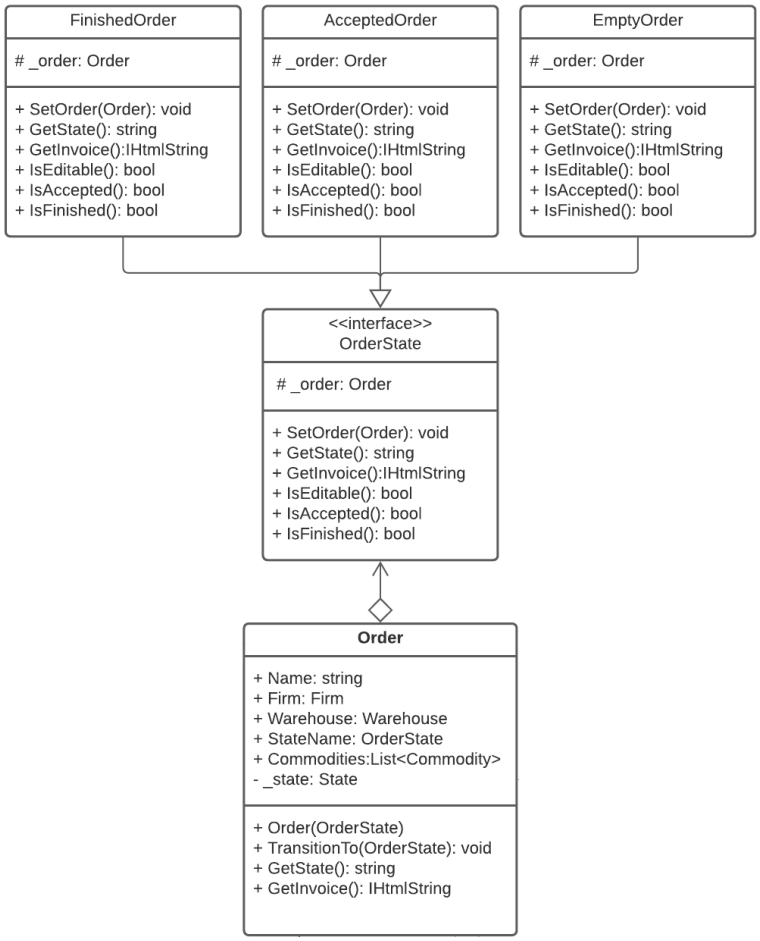
Lokalizacja wzorca w kodzie:

Klasy deklarujące fasadę znajdują się w */Logic/Facade/*. Użycie wzorca znajduje się w całym projekcie np. w plikach kontrolerów.

Wektor zmian:

W przypadku zmiany metody łączenia się z bazą danych nie trzeba refaktoryzować całego programu, wystarczy zmiana fasady.

- Stan (state)



Cel użycia:

Stan wykorzystywany w celu dostosowania metod dostępowych obiektu (kontekstu) Order.

Przyporządkowanie klas do ról wzorca:

1. EmptyOrder: stan oznaczający stworzone zamówienie, w którym nie zostały jeszcze zatwierdzone wszystkie jego elementy.
    - GetState() - zwraca string aktualnego stanu (“Incomplete Order”), używany we front-endowej części programu
    - GetInvoice() - zwraca pusty string – oznacza to brak możliwości wydrukowania faktury
    - IsEditable, IsAccepted, IsFinished - przydzielają dostęp do odpowiednich funkcji programu w zależności od stanu. W EmptyOrder IsEditable zwraca true.
2. AcceptedOrder: stan oznaczający zatwierdzone zamówienie, oczekujące realizacji polegającej na zebraniu wystarczających przedmiotów w magazynie i “wysłaniu” ich do klienta.
    - GetState() - zwraca string aktualnego stanu (“Completed Order”)
    - GetInvoice() zwraca pusty string – oznacza to brak możliwości wydrukowania faktury

    - W tym stanie IsAccepted zwraca true – co umożliwia dostęp do odpowiednich funkcji
3. FinishedOrder: stan oznaczający zrealizowane zamówienie.
    - GetState() - zwraca string aktualnego stanu (“Finished Order”)
    - GetInvoice() - zwraca kod HTML zawierający gotową fakturę na podstawie zamówienia. Możliwe jest wyświetlenie faktury na stronie bądź pobranie jej w formacie .pdf.
    - W tym stanie IsFinished zwraca true – co umożliwia dostęp do odpowiednich funkcji.

Lokalizacja wzorca w kodzie:

Klasy odpowiadające za implementacje interfejsu stanu oraz poszczególnych stanów znajdują się w /Logic/State. Klasa kontekstu Order znajduje się w /Models. Użycie wzorca znajduje się we wszystkich widokach oraz kontrolerem modelu Order.

Wektor zmian:

Wykorzystując stan projekt został pozbawiony wielu złożonych instrukcji warunkowych (if/switch). Umożliwiło to także rozbicie jednej dużej skomplikowanej klasy na kilka mniejszych klas.

- MVC

Projekt został oparty na wzorcu Model-View-Controller, który pozwala na rozdzielenie logiki programu, implementacji UI oraz modeli danych. Ułatwia to w znaczącym stopniu rozszerzanie programu o nowe klasy oraz tworzenie do nich widoków.

4. Opis co najmniej jednego rozwiązania specyficznego dla użytej technologii

PdfSharp – Jest to biblioteka zbudowana dla rozwiązań w technologii .NET. Pozwala ona na operacje specyficzne dla plików PDF a dokładniej w projekcie jest używane do generowania pliku PDF na podstawie kodu HTML.

ASP.NET - Jest frameworkiem specyficznym dla technologii .NET. Służy on do tworzenia interaktywnych aplikacji sieciowych. Jest on szeroko używany w większości plików projektu.

5. Opis podziału pracy w grupie

Piotr Go:

- Metoda fabrykująca
- Fasada
- Modele przedmiotów
- Kontrolery
- Widoki

Łukasz Chojecki:

- Stan
- Modele
- Kontrolery
- Widoki

Konrad Wasilewski:

- Fabryka abstrakcyjna
- Tworzenie plików PDF
- Stylizacja CSS

Konkretny podział pracy można znaleźć na repozytorium github, ale trzeba także wziąć pod uwagę wspólnej pomocy przy wystąpieniu problemów.

6. Instrukcja użytkownika oraz funkcjonalności

Po uruchomieniu strony widoczna jest widok witający użytkownika. Wszystkie potrzebne funkcjonalności można znaleźć w menu głównym znajdującym się w górnej porcji ekranu. Na początek należy założyć konto lub przy ponownym korzystaniu skorzystać z opcji logowania. Po zalogowaniu odblokowywany jest dostęp do funkcjonalności.

Warto rozpocząć od dodania do bazy produktów, można to zrobić poprzez zakładki "*Electronic Items*", "*Consumable Items*" oraz "*Furniture Items*". Każda z zakładek odpowiada za inny typ towaru, po wejściu do zakładek można w prosty sposób tworzyć, usuwać oraz edytować wcześniej wspomniane produkty.

Następnym krokiem powinno być stworzenie magazynów oraz firm w zakładkach "*Warehouses*" i "*Firms*". Po ukończeniu tego kroku należy wypełnić magazyny, można to zrobić w zakładce "*Commodities*". Dodając towar wybierany jest magazyn, do którego następuje dostawa, typ dostarczonego produktu, jego ilość oraz cena.

Kiedy wszystko jest gotowe można rozpocząć składanie zamówień w zakładce "*Orders*". Tworząc nowe zamówienie nadaje mu się nazwę, wybiera się zamawiającą firmę oraz magazyn, do którego składane jest zamówienie. Po stworzeniu zamówienia można znaleźć je na liście, aby posunąć zamówienie do kolejnego kroku należy kliknąć na "*Complete order*", w nowo wyświetlonym oknie można rozpocząć uzupełnianie zawartości zamówienia poprzez dodawanie towarów przyciskiem "*Add new commodity*". Po ukończeniu wybierania produktów należy zaakceptować zamówienie, następuje wtedy przeniesienie do listy wszystkich zamówień. Ostatecznym krokiem zamówienia jest jego realizacja przyciskiem "*Realize Order*", jeżeli w magazynie znajduje się wystarczająca ilość produktów zamówienie zostanie zrealizowane oraz usuwana jest odpowiednia liczba przedmiotów z magazynu.

W momencie, kiedy na liście jest zrealizowane zamówienie można otrzymać z niego fakturę przez przycisk "*Get Invoice*", w ten sposób generowany i pokazywany jest dokument HTML przedstawiający fakturę. Jeśli istnieje taka potrzeba to można z widoku zawierającego fakturę pobrać ją w postaci pliku PDF poprzez kliknięcie na przycisk "*Download Invoice*".

7. Instrukcja instalacji

Stworzone oprogramowanie jest aplikacją sieciową co oznacza, że do jej działania potrzebny jest serwer. W celu prezentacji funkcjonowania korzystamy z lokalnego serwera IIS Express, należy uruchomić projekt w Visual Studio, zaktualizować bazę danych przy pomocy polecenia “Update-Database” wpisanego do konsoli menedżera pakietów, a następnie uruchomić aplikację za pomocą wcześniej wspomnianego serwera. Serwer ten jest wbudowany w Visual Studio co sprawia, że uruchomienie jest proste.