

# AGH

**AKADEMIA GÓRNICZO-HUTNICZA IM. STANISŁAWA STASZICA W KRAKOWIE**

**WYDZIAŁ INFORMATYKI, ELEKTRONIKI I TELEKOMUNIKACJI**

**KATEDRA INFORMATYKI**

## **BAZY DANYCH**

Technologia: **Python + Django**

Serwer BD: **MS Sql**

Repository: <https://github.com/robertcialowicz/bazydanych2.git>

Autorzy:

*Chwała Paweł <pchwala@student.agh.edu.pl>*

*Ciałowicz Robert <robcial@student.agh.edu.pl>*

*Kozaczekiewicz Łukasz <kozaczki@student.agh.edu.pl>*

*Szpila Magdalena <mszpila@student.agh.edu.pl>*

Kraków, 2020



## Spis treści

Opis projektu	<b>3</b>
Struktura projektu	<b>4</b>
Serwer Microsoft SQL	<b>6</b>
Uruchomienie	6
Struktura projektu	6
Skrypt naprawczy	7
Projekt django	<b>7</b>
Jak zacząć	8
Generowanie modelu	8
Django admin	9
Dostosowanie formularzy	12
Galeria	<b>15</b>
Wnioski	<b>18</b>

## Opis projektu

Aplikacja napisana w języku Python z wykorzystaniem Django Rest Framework.

W projekcie wykorzystano serwer bazy danych MSSql.

Przykładową bazą, na której wykonywane będą operacje jest baza Northwind.

Celem projektu jest zaimplementowanie operacji CRUD na dowolnej tabeli, operacji składania zamówienia oraz możliwość tworzenia raportów (do zdefiniowania).

## Struktura projektu

W lokalizacji `./src` znajduje się plik `docker-compose.yml`. Aby wystartować aplikację należy z konsoli wywołać `docker-compose up --build`.

Uruchomią się dwa kontenery dockerowe zdefiniowane w pliku `yml`: jeden z serwerem MSSql z bazą danych Northwind oraz drugi z aplikacją pythonową.

UWAGA!

W obecnej wersji aplikacja nie działa na systemach platformy Windows.

**Struktura projektu:**



**src/sqlserver** - konfiguracja dockera udostępniającego serwer sql

**src/web** - konfiguracja dockera udostępniającego aplikację webową

**src/myapp**, `myapi` - aplikacje oparte na frameworku Django wchodzące w skład projektu

**src/manage.py** - Narzędzie linii komend, które pozwala oddziaływać z projektem Django. Więcej szczegółów na temat `manage.py` zostało opisanych w punkcie Projekt Django

**src/inspect.db** - plik zawierający automatycznie wygenerowane modele na podstawie bazy danych Northwind. Został wygenerowany z wykorzystaniem `manage.py` i komendy `inspectdb`. Pełni funkcje jedynie poglądowe i został zachowany aby zaprezentować narzędzia wspomagające integrowanie aplikacji w Django z “odziedziczonymi” bazami danych.

**src/docker-compose.yml** - konfiguracja dla uruchomienia wszystkich składowych projektu

## Serwer Microsoft SQL

### Uruchomienie

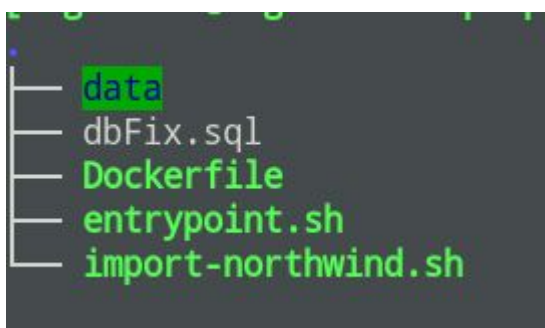
Skorzystaliśmy z gotowego obrazu dockerowego Microsoft SQL Server 2019. Najprostszym sposobem na uruchomienie serwera bazy danych jest wykonanie komendy z poziomu katalogu /src

```
docker-compose -up --build sqlserver
```

Po starcie serwera najpierw jest importowana baza Northwind, a następnie nakładana jest na to nasza poprawka.

### Struktura projektu

Zawartość katalogu **src/sqlserver**



**import-northwind.sh** - skrypt uruchamiający serię komend SQL-owych pozwalający utworzyć i spopulować bazę Northwind - jeżeli nie istnieje na serwerze MS SQL

**entrypoint.sh** - skrypt uruchamiający serwer MS SQL a następnie wywołujący skrypt import-northwind.sh wewnątrz wystartowanego dockera

**dbFix.sh** - skrypt wywołujący serię komend SQL-owych modyfikujących tabele w bazie zgodnie z wymaganiami frameworka Django

**data** - wolumen z danymi bazy, obecnie "nie podpinany"

**Dockerfile** - plik z konfiguracją dockera

## Skrypt naprawczy

Na potrzeby projektu musieliśmy dokonać pewnych zmian w bazie danych. Było to spowodowane tym, że Django wymaga, aby każda tabela miała dokładnie jeden klucz główny. W bazie Northwind tabela *OrderDetails* nie ma takiego klucza. Poniżej zamieszczamy skrypt, który dodaje brakujący klucz.

```
USE [Northwind]
GO
ALTER TABLE [Order Details] DROP CONSTRAINT IF EXISTS pk_order_details
ALTER TABLE [Order Details] DROP CONSTRAINT IF EXISTS uk_order_details
ALTER TABLE [Order Details] DROP COLUMN IF EXISTS orderdetailid
ALTER TABLE [Order Details] ADD orderdetailid INTEGER IDENTITY(1,1) NOT NULL
GO
ALTER TABLE [Order Details] ADD CONSTRAINT pk_order_details PRIMARY KEY (orderdetailid)
ALTER TABLE [Order Details] ADD CONSTRAINT uk_order_details UNIQUE (orderid, productid)
```

# Projekt django

## Jak zacząć

Webowy projekt dla poprawnego działania wymaga możliwości połączenia do bazy danych co zapewnia konfiguracja w docker-compose.yml. Najlepszym sposobem na uruchomienie aplikacji jest wykonanie komendy z poziomu katalogu /src

```
docker compose -up --build web
```

Przy starcie dockera instalowane są potrzebne biblioteki do pracy z frameworkiem Django i bazą danych, kopiowany jest kod projektu do wnętrza kontenera a następnie uruchamiany jest skrypt wait-for-it.sh, którego zadaniem jest wstrzymanie uruchomienia aplikacji Django do momentu aż dostępny będzie host i port serwera MS SQL. Projekt Django uruchamiany jest komendą

```
python3 manage.py runserver 0.0.0.0:8000
```

**manage.py** - Narzędzie linii komend, które pozwala oddziaływać z tym projektem Django na wiele sposobów. Automatycznie generowany w każdym projekcie Django. Z najważniejszych udostępnianych komend warto zwrócić uwagę na:

**makemigrations** - tworzy nowe migracje na podstawie zmian wykrytych w modelu

**migrate** - synchronizuje aktualny stan bazy danych ze stanem modelu i migracji

**createsuperuser** - usługa dostępna przy zainstalowanym pakiecie **django.contrib.auth**, tworzy konto użytkownika z pełnymi prawami dostępu. Przydatne przy rozpoczynaniu pracy z modułem **admin**

**runserver** - uruchamia lekki serwer webowy na lokalnej maszynie



## Generowanie modelu

Aby wygenerować model bazy Northwind użyto funkcji inspectdb frameworku Django. Poszczególne kroki przedstawiają się następująco:

1. Uruchomienie aplikacji - `python src/manage.py startapp myapp`
2. Wygenerowanie i zapisanie modelu - `python src/manage.py inspectdb > src/myapi/models.py`
3. Naniesienie poprawek na model:
  - w przypadku relacji many-to-many (Employees Territories, Order Details, CustomerCustomerDemo) usunięto `ManyToManyField` z tabel tworzących te relacje
  - zmieniono kolejność klas w modelu zgodnie z odwołaniami. Końcowa kolejność to: Categories, Shippers, CustomerCustomerDemo, CustomerDemographics, Customer, Region, Territories, Employees, Suppliers, Products, Orders, OrderDetails, EmployeeTerritories
  - w tabelach, które nie miały klucza głównego ustawiono odpowiedni klucz główny, poprzez `primary_key = True`

Pomimo faktu, że framework django jest polecany przede wszystkim do green-field development czyli budowania projektu od zera, w przypadku bazy Northwind wygenerowanie modelu i poprawki było relatywnie proste. Możliwość autogeneracji modelu przez framework jest ogromną zaletą gdyż pozwala oszczędzić czas oraz uniknąć pomyłek. Należy jednak pamiętać, że takie podejście może doprowadzić do niepożądanych zachowań. W trakcie pracy nad tym projektem takich sytuacji nie zaobserwowano

## Django admin

Panel Django admin jest to interface użytkownika dostarczany przez framework, umożliwiający przygotowanie w szybki sposób uproszczonego front endu na potrzeby developmentu i testowania aplikacji. Po uruchomieniu serwera django poprzez `python manage.py runserver` panel ten jest dostępny domyślnie pod `localhost:8000/admin`

W przypadku istniejących obiektów, np klasa `Product`:

w pliku `myapi/admin.py` należy

1. zaimportować tę klasę: `from .models import Products`
2. zarejestrować ją w panelu: `admin.site.register(Products)`

Dostarczanie przez framework panelu admin jest bardzo pomocna, gdyż przy minimalnym nakładzie pracy umożliwia w prosty sposób testowanie funkcjonalności aplikacji.

Domyślne zachowanie zaimplementowane w ten sposób, nie zwraca jednak zbyt wiele informacji o obiekcie. Zamiast nazw wyświetlane są jedynie id obiektów referujących przez klucze obce. Aby zmienić to zachowanie należy w pliku models.py nadpisać funkcję `__str__(self)`. W ten sposób możemy reprezentować obiekt przez dowolnie zdefiniowany ciąg znaków. W przypadku tego projektu dla klasy Products jest to:

```
def __str__(self):  
    return self.productname
```

Dla pozostałych klas w modelu zaimplementowano analogiczne rozwiązanie.

Warto zauważyć, że to rozwiązanie nie zmienia nic w wydajności programu. Do serwera SQL wysyłane są dokładnie te same zapytania. Oznacza to, że django automatycznie pobiera z bazy całe obiekty, nawet jeśli tylko referują one przez klucz obcy.

```
QUERY = 'SELECT [Products].[ProductID], [Products].[ProductName], [Products].[SupplierID], [Products].[CategoryID],  
[Products].[QuantityPerUnit], [Products].[UnitPrice], [Products].[UnitsInStock], [Products].[UnitsOnOrder],  
[Products].[ReorderLevel], [Products].[Discontinued] FROM [Products] WHERE [Products].[ProductID] = %s' - PARAMS =  
(76,); args=(76,)  
  
QUERY = 'SELECT [Suppliers].[SupplierID], [Suppliers].[CompanyName], [Suppliers].[ContactName],  
[Suppliers].[ContactTitle], [Suppliers].[Address], [Suppliers].[City], [Suppliers].[Region],  
[Suppliers].[PostalCode], [Suppliers].[Country], [Suppliers].[Phone], [Suppliers].[Fax], [Suppliers].[HomePage] FROM  
[Suppliers]' - PARAMS = (); args=()  
  
QUERY = 'SELECT [Categories].[CategoryID], [Categories].[CategoryName], [Categories].[Description],  
[Categories].[Picture] FROM [Categories]' - PARAMS = (); args=()
```

Ponadto panel Django admin umożliwia dodanie formularza z obiektami wbudowanymi (inline), wtedy w jednym widoku widoczne są obiekty danego typu, ale także obiekty referujące do nich przez klucz obcy. Poniżej przykład z projektu, gdzie obiektem nadrzędnym jest obiekt Product, a obiekty wbudowane to obiekty typu OrderDetails.

```
class ProductsAdmin(admin.ModelAdmin):  
    inlines = (OrderDetailsInline, )  
  
admin.site.register(Products, ProductsAdmin)
```

Warto zwrócić uwagę na to, że zmienił się sposób rejestrowania obiektu.

Wykorzystując powyższą instrukcję jesteśmy w stanie wykonywać podstawowe operacje na obiektach zależnych, czyli w naszym przypadku produktach. Możemy je dodawać, usuwać i edytować w bardzo prosty sposób z poziomu podglądu zamówienia. Dostajemy tą funkcjonalność “za darmo” w postaci odpowiednich przycisków obok każdego z produktów.

PRODUCTID	UNITPRICE	QUANTITY	DISCOUNT	DELETE?
Product Chang within Order no. 11077				
Chang	19,0000	24	0,20000000298023224	<input type="checkbox"/>

Wprowadzenie obiektów typu inline wprowadza również zmiany w zapytaniach SQL wysyłanych do serwera. Dla obiektu Products bez wbudowanych obiektów zapytanie wygląda jak te przedstawione powyżej. Dla obiektu Products z wbudowanymi obiektami typu OrderDetails zapytań znacznie przybywa.

```
QUERY = 'SELECT [Products].[ProductID], [Products].[ProductName], [Products].[SupplierID], [Products].[CategoryID],
[Products].[QuantityPerUnit], [Products].[UnitPrice], [Products].[UnitsInStock], [Products].[UnitsOnOrder],
[Products].[ReorderLevel], [Products].[Discontinued] FROM [Products] WHERE [Products].[ProductID] = %s' - PARAMS =
(76,); args=(76,)
```

```
QUERY = 'SELECT [Order Details].[OrderID], [Order Details].[ProductID], [Order Details].[UnitPrice], [Order
Details].[Quantity], [Order Details].[Discount], [Order Details].[orderdetailid] FROM [Order Details] WHERE [Order
Details].[ProductID] = %s ORDER BY [Order Details].[orderdetailid] ASC' - PARAMS = (76,); args=(76,)
```

```
QUERY = 'SELECT [Suppliers].[SupplierID], [Suppliers].[CompanyName], [Suppliers].[ContactName],
[Suppliers].[ContactTitle], [Suppliers].[Address], [Suppliers].[City], [Suppliers].[Region],
[Suppliers].[PostalCode], [Suppliers].[Country], [Suppliers].[Phone], [Suppliers].[Fax], [Suppliers].[HomePage] FROM
[Suppliers]' - PARAMS = (); args=()
```

```
QUERY = 'SELECT [Categories].[CategoryID], [Categories].[CategoryName], [Categories].[Description],
[Categories].[Picture] FROM [Categories]' - PARAMS = (); args=()
```

```
QUERY = 'SELECT [Products].[ProductID], [Products].[ProductName], [Products].[SupplierID], [Products].[CategoryID],
[Products].[QuantityPerUnit], [Products].[UnitPrice], [Products].[UnitsInStock], [Products].[UnitsOnOrder],
[Products].[ReorderLevel], [Products].[Discontinued] FROM [Products] WHERE [Products].[ProductID] = %s' - PARAMS =
(76,); args=(76,)
```

```
QUERY = 'SELECT [Orders].[OrderID], [Orders].[CustomerID], [Orders].[EmployeeID], [Orders].[OrderDate],
[Orders].[RequiredDate], [Orders].[ShippedDate], [Orders].[ShipVia], [Orders].[Freight], [Orders].[ShipName],
[Orders].[ShipAddress], [Orders].[ShipCity], [Orders].[ShipRegion], [Orders].[ShipPostalCode], [Orders].[ShipCountry]
FROM [Orders] WHERE [Orders].[OrderID] = %s' - PARAMS = (10343,); args=(10343,)
```

```
QUERY = 'SELECT [Orders].[OrderID], [Orders].[CustomerID], [Orders].[EmployeeID], [Orders].[OrderDate],
[Orders].[RequiredDate], [Orders].[ShippedDate], [Orders].[ShipVia], [Orders].[Freight], [Orders].[ShipName],
[Orders].[ShipAddress], [Orders].[ShipCity], [Orders].[ShipRegion], [Orders].[ShipPostalCode], [Orders].[ShipCountry]
FROM [Orders]' - PARAMS = (); args=()
```

```
QUERY = 'SELECT [Products].[ProductID], [Products].[ProductName], [Products].[SupplierID], [Products].[CategoryID],
[Products].[QuantityPerUnit], [Products].[UnitPrice], [Products].[UnitsInStock], [Products].[UnitsOnOrder],
[Products].[ReorderLevel], [Products].[Discontinued] FROM [Products] WHERE [Products].[ProductID] = %s' - PARAMS =
(76,); args=(76,)
```

```
QUERY = 'SELECT [Orders].[OrderID], [Orders].[CustomerID], [Orders].[EmployeeID], [Orders].[OrderDate],
[Orders].[RequiredDate], [Orders].[ShippedDate], [Orders].[ShipVia], [Orders].[Freight], [Orders].[ShipName],
[Orders].[ShipAddress], [Orders].[ShipCity], [Orders].[ShipRegion], [Orders].[ShipPostalCode], [Orders].[ShipCountry]
FROM [Orders] WHERE [Orders].[OrderID] = %s' - PARAMS = (10267,); args=(10267,)
```

```
QUERY = 'SELECT [Orders].[OrderID], [Orders].[CustomerID], [Orders].[EmployeeID], [Orders].[OrderDate],
[Orders].[RequiredDate], [Orders].[ShippedDate], [Orders].[ShipVia], [Orders].[Freight], [Orders].[ShipName],
[Orders].[ShipAddress], [Orders].[ShipCity], [Orders].[ShipRegion], [Orders].[ShipPostalCode], [Orders].[ShipCountry]
FROM [Orders]' - PARAMS = (); args=()
```

```
...
```

Widać, że dla każdego obiektu zależnego wysyłane są 3 dodatkowe zapytania SQL. W przypadku dużej ilości obiektów powiązanych, znacząco wpływa to na czas ładowania widoku pojedynczego produktu. Przykładowo dla Produktu Lakkalikööri czas wzrasta z około 800 ms do 17 sekund.

Dla czytelności panelu administracyjnego usunięto domyślne formularze dla użytkowników i grup za pomocą metody *unregister*:

```
admin.site.unregister(User)
admin.site.unregister(Group)
```

## Dostosowanie formularzy

Walidacja pól formularza znajduje się w metodach *clean* klas *OrdersForm* oraz *OrderDetailsInlineFormSet* w zależności od tego czy pracujemy z obiektami typu wbudowanego czy nie. W metodzie *clean* możemy natomiast dowolnie zdefiniować swoje kryteria oraz treść zwracanego wyjątku.

Dla zamówienia sprawdzamy czy data zamówienia (*OrderDate*) jest co najmniej dzień przed datą zapotrzebowania (*RequiredDate*).

```
def clean(self):
    orderDate = self.cleaned_data.get('orderdate')
    requiredDate = self.cleaned_data.get('requireddate')
    if orderDate >= (requiredDate - datetime.timedelta(days=1)):
        raise forms.ValidationError("Orderdate must be at least 24 hours before Required date!")
    return self.cleaned_data
```

Natomiast dla produktu sprawdzamy czy nie został dodany dwa razy do tego samego zamówienia, czy jego cena za jednostkę jest większa od 0, czy zniżka jest w zakresie [0, 1] oraz czy w magazynie znajduje się wystarczająca ilość tego produktu. Kod walidacji produktu znajduje się poniżej.

```
def clean(self):
    setOfProducts = set()
    if(self.is_valid()):
        for productForm in self.cleaned_data:
            reservedQuantity = productForm.get('quantity')
            product = productForm.get('productid')
            unitsInStock = product.unitsinstock
            price = productForm.get('unitprice')
            discount = productForm.get('discount')
            if product in setOfProducts:
                raise forms.ValidationError("Product " + str(product) + " was added more than once!")
            else:
                setOfProducts.add(product)
                if price <= 0:
                    raise forms.ValidationError("Unitprice for product " + str(product) + " has to be greater than 0!")
                if discount < 0 or discount > 1:
                    raise forms.ValidationError("Discount for product " + str(product) + " has to be value between 0 and 1!")
                if reservedQuantity > unitsInStock:
                    raise forms.ValidationError("Maximum quantity for product " + str(product) + " is " + str(unitsInStock) + "!")
```

Dodanie do zamówienia listy przypisanych do niego produktów powoduje wykonanie dodatkowych zapytań do bazy danych, co z kolei ma wpływ na szybkość ładowania się strony. Przed dodaniem do formularza listy produktów po wejściu w edycję istniejącego zamówienia były wykonywane poniższe zapytania. Zapytania przed dodaniem listy produktów do widoku zamówienia:

```
SELECT [Orders].[OrderID], [Orders].[CustomerID], [Orders].[EmployeeID], [Orders].[OrderDate],
[Orders].[RequiredDate], [Orders].[ShippedDate], [Orders].[ShipVia], [Orders].[Freight], [Orders].[ShipName],
[Orders].[ShipAddress], [Orders].[ShipCity], [Orders].[ShipRegion], [Orders].[ShipPostalCode],
[Orders].[ShipCountry] FROM [Orders] WHERE [Orders].[OrderID] = 11079;

SELECT [Customers].[CustomerID], [Customers].[CompanyName], [Customers].[ContactName], [Customers].[ContactTitle],
[Customers].[Address], [Customers].[City], [Customers].[Region], [Customers].[PostalCode], [Customers].[Country],
[Customers].[Phone], [Customers].[Fax] FROM [Customers];

SELECT [Employees].[EmployeeID], [Employees].[LastName], [Employees].[FirstName], [Employees].[Title],
[Employees].[TitleOfCourtesy], [Employees].[BirthDate], [Employees].[HireDate], [Employees].[Address],
[Employees].[City], [Employees].[Region], [Employees].[PostalCode], [Employees].[Country], [Employees].[HomePhone],
[Employees].[Extension], [Employees].[Photo], [Employees].[Notes], [Employees].[ReportsTo], [Employees].[PhotoPath]
FROM [Employees];

SELECT [Shippers].[ShipperID], [Shippers].[CompanyName], [Shippers].[Phone] FROM [Shippers];
```

Natomiast po dodaniu listy produktów ilość wykonywanych zapytań wzrosła dwukrotnie. Na szczęście czas ładowania strony wzrósł tylko nieznacznie. Całość zajęła średnio około 600ms, a na odpowiedź serwera czekaliśmy 200ms. Zapytania po dodaniu listy produktów do widoku zamówienia:

```
SELECT [Orders].[OrderID], [Orders].[CustomerID], [Orders].[EmployeeID], [Orders].[OrderDate],
[Orders].[RequiredDate], [Orders].[ShippedDate], [Orders].[ShipVia], [Orders].[Freight], [Orders].[ShipName],
[Orders].[ShipAddress], [Orders].[ShipCity], [Orders].[ShipRegion], [Orders].[ShipPostalCode],
[Orders].[ShipCountry] FROM [Orders] WHERE [Orders].[OrderID] = 11079;

SELECT [Order Details].[OrderID], [Order Details].[ProductID], [Order Details].[UnitPrice], [Order
Details].[Quantity], [Order Details].[Discount], [Order Details].[orderdetailid] FROM [Order Details] WHERE [Order
Details].[OrderID] = 11079 ORDER BY [Order Details].[orderdetailid] ASC;

SELECT [Customers].[CustomerID], [Customers].[CompanyName], [Customers].[ContactName], [Customers].[ContactTitle],
[Customers].[Address], [Customers].[City], [Customers].[Region], [Customers].[PostalCode], [Customers].[Country],
[Customers].[Phone], [Customers].[Fax] FROM [Customers];

SELECT [Employees].[EmployeeID], [Employees].[LastName], [Employees].[FirstName], [Employees].[Title],
[Employees].[TitleOfCourtesy], [Employees].[BirthDate], [Employees].[HireDate], [Employees].[Address],
[Employees].[City], [Employees].[Region], [Employees].[PostalCode], [Employees].[Country], [Employees].[HomePhone],
[Employees].[Extension], [Employees].[Photo], [Employees].[Notes], [Employees].[ReportsTo], [Employees].[PhotoPath]
FROM [Employees];

SELECT [Shippers].[ShipperID], [Shippers].[CompanyName], [Shippers].[Phone] FROM [Shippers];

SELECT [Order Details].[OrderID], [Order Details].[ProductID], [Order Details].[UnitPrice], [Order
Details].[Quantity], [Order Details].[Discount], [Order Details].[orderdetailid] FROM [Order Details] WHERE [Order
Details].[OrderID] = 11079;

SELECT [Products].[ProductID], [Products].[ProductName], [Products].[SupplierID], [Products].[CategoryID],
[Products].[QuantityPerUnit], [Products].[UnitPrice], [Products].[UnitsInStock], [Products].[UnitsOnOrder],
[Products].[ReorderLevel], [Products].[Discontinued] FROM [Products] WHERE [Products].[ProductID] = 21;

SELECT [Orders].[OrderID], [Orders].[CustomerID], [Orders].[EmployeeID], [Orders].[OrderDate],
[Orders].[RequiredDate], [Orders].[ShippedDate], [Orders].[ShipVia], [Orders].[Freight], [Orders].[ShipName],
[Orders].[ShipAddress], [Orders].[ShipCity], [Orders].[ShipRegion], [Orders].[ShipPostalCode],
[Orders].[ShipCountry] FROM [Orders] WHERE [Orders].[OrderID] = 110791;

SELECT [Products].[ProductID], [Products].[ProductName], [Products].[SupplierID], [Products].[CategoryID],
[Products].[QuantityPerUnit], [Products].[UnitPrice], [Products].[UnitsInStock], [Products].[UnitsOnOrder],
[Products].[ReorderLevel], [Products].[Discontinued] FROM [Products];

SELECT [Products].[ProductID], [Products].[ProductName], [Products].[SupplierID], [Products].[CategoryID],
```

```
[Products].[QuantityPerUnit], [Products].[UnitPrice], [Products].[UnitsInStock], [Products].[UnitsOnOrder],  
[Products].[ReorderLevel], [Products].[Discontinued] FROM [Products];
```

Pod każdym zamówieniem umieściliśmy dodatkowo krótkie podsumowanie. Zostało ono dodane poprzez dodanie do klasy modelu *Order* property *summary*, a następnie rozdzielenia formularza na dwa formsety. Pierwszy z nich zawiera wszystkie pola z modelu *Order*, a drugi tylko *summary*. Dzięki temu są one wyświetlane osobno. *Summary* wylicza całkowitą wartość zamówienia sumując dla każdego produktu w zamówieniu jego cenę używając poniższego wzoru:

$$X = P * Q * (1 - D)$$

gdzie:

X - ostateczna cena produktu

P - cena jednej sztuki produktu

Q - ilość produktów w zamówieniu

D - zniżka z przedziału [0, 1]

Zdefiniowanie dodatkowej property w pliku *models.py*:

```
class Orders(models.Model)  
  
    def summary(self):  
  
        sum = 0  
  
        for item in self.orderdetailsFK.through.objects.filter(orderid=self.orderid):  
  
            sum += float(item.unitprice) * float(item.quantity) * (1 -  
float(item.discount))  
  
        return "${0}".format(round(sum, 2))
```

Kod odpowiedzialny za zdefiniowanie formsetów z pliku *admin.py* znajduje się poniżej.

```
class OrdersAdmin(admin.ModelAdmin):  
  
    form = OrdersForm  
  
    inlines = (OrderDetailsInline, )  
  
    fieldsets = [  
  
        (None, {'fields': ['customerid', 'employeeid', 'orderdate', 'requireddate',  
'shippeddate', 'shipvia', 'freight', 'shipname', 'shipaddress', 'shipcity', 'shipregion',  
'shippostalcode', 'shipcountry', ]}),  
  
        ('Order Summary', {'fields': ['summary', ]}),  
  
    ]  
  
    readonly_fields = ('summary', )
```





## Galeria

### Change orders

Customerid:	Alfreds Futterkiste	▼
Employeeid:	Andrew Fuller	▼
Orderdate:	Date: 2021-01-04	Today   📅
	Time: 18:20:42	Now   🕒
	Note: You are 1 hour ahead of server time.	
Requireddate:	Date: 2021-01-07	Today   📅
	Time: 18:20:44	Now   🕒
	Note: You are 1 hour ahead of server time.	
Shippeddate:	Date: 2021-01-07	Today   📅
	Time: 18:20:45	Now   🕒
	Note: You are 1 hour ahead of server time.	
Shipvia:	United Package	▼
Freight:	0,0001	
Shipname:	Titanic	
Shipaddress:	Iceberg	
Shipcity:		
Shipregion:		
Shippostalcode:		
Shipcountry:		

Dodanie/edycja zamówienia.

Order Summary

Summary: \$1255.72

ORDER DETAILS

PRODUCTID	UNITPRICE	QUANTITY	DISCOUNT	DELETE?
Product Chang within Order no. 11077				
<div>Chang</div>	<div>19,0000</div>	<div>24</div>	<div>0,20000000298023224</div>	<div></div>
Product Aniseed Syrup within Order no. 11077				
<div>Aniseed Syrup</div>	<div>10,0000</div>	<div>4</div>	<div>0,0</div>	<div></div>
Product Chef Anton's Cajun Seasoning within Order no. 11077				
<div>Chef Anton's Cajun Seasoning</div>	<div>22,0000</div>	<div>1</div>	<div>0,0</div>	<div></div>
Product Grandma's Boysenberry Spread within Order no. 11077				
<div>Grandma's Boysenberry Spread</div>	<div>25,0000</div>	<div>1</div>	<div>0,01999999955296516</div>	<div></div>
Product Uncle Bob's Organic Dried Pears within Order no. 11077				
<div>Uncle Bob's Organic Dried Pears</div>	<div>30,0000</div>	<div>1</div>	<div>0,05000000074505806</div>	<div></div>
Product Northwoods Cranberry Sauce within Order no. 11077				
<div>Northwoods Cranberry Sauce</div>	<div>40,0000</div>	<div>2</div>	<div>0,10000000149011612</div>	<div></div>
Product Ikura within Order no. 11077				

Podsumowanie i widok produktów należących do zamówienia.

Order Summary
Summary:
\$1255.72

ORDER DETAILS

PRODUCTID	UNITPRICE	QUANTITY	DISCOUNT
Product Chang within Order no. 11077			
Chang	19,0000	24	0,20000000298023224
Product Aniseed Syrup within Order no. 11077			
Aniseed Syrup	10,0000	4	0,0
Product Chef Anton's Cajun Seasoning within Order no. 11077			
Chef Anton's Cajun Seasoning	22,0000	1	0,0
Product Grandma's Boysenberry Spread within Order no. 11077			
Grandma's Boysenberry Spread	25,0000	1	0,01999999955296516
Product Uncle Bob's Organic Dried Pears within Order no. 11077			
Uncle Bob's Organic Dried Pears	30,0000	1	0,05000000074505806
Product Northwoods Cranberry Sauce within Order no. 11077			
Northwoods Cranberry Sauce	40,0000	2	0,10000000149011612
Product Ikura within Order no. 11077			
Ikura	31,0000	1	0,0

Change products | Django site admin - Google Chrome

Niezabezpieczona | dev-pchwała:8000/admin/myapi/products/2/change/?\_to\_field=productid&\_popup=1

Change products
Productname:
Chang
Supplierid:
Exotic Liquids
Categoryid:
Beverages
Quantityperunit:
24 - 12 oz bottles

Czekam na dev-pchwała...

Edycja produktu z poziomu zamówienia.

# Django administration

Home › Myapi › Orders

## Select orders to change

Action:   0 of 100 selected

☐ ORDERS

☐ Order no. 11079

☐ Order no. 11077

☐ Order no. 11076

☐ Order no. 11075

☐ Order no. 11074

☐ Order no. 11073

☐ Order no. 11072

☐ Order no. 11071

☐ Order no. 11070

☐ Order no. 11069

☐ Order no. 11068

Lista zamówień.

```
ode], [Orders].[ShipCountry] FROM [Orders] WHERE [Orders].[OrderID] = %s' - PARAMS = (11076,); args=(11076,)
web 1 | (0.001) QUERY = 'SELECT [Products].[ProductID], [Products].[ProductName], [Products].[SupplierID], [Products].[CategoryID], [Products].[QuantityPerU
nit], [Products].[UnitPrice], [Products].[UnitsInStock], [Products].[UnitsOnOrder], [Products].[ReorderLevel], [Products].[Discontinued] FROM [Products]' - PARAMS
= (); args=()
web 1 | (0.001) QUERY = 'SELECT [Products].[ProductID], [Products].[ProductName], [Products].[SupplierID], [Products].[CategoryID], [Products].[QuantityPerU
nit], [Products].[UnitPrice], [Products].[UnitsInStock], [Products].[UnitsOnOrder], [Products].[ReorderLevel], [Products].[Discontinued] FROM [Products] WHERE [Pro
ducts].[ProductID] = %s' - PARAMS = (14,); args=(14,)
web 1 | (0.000) QUERY = 'SELECT [Orders].[OrderID], [Orders].[CustomerID], [Orders].[EmployeeID], [Orders].[OrderDate], [Orders].[RequiredDate], [Orders].[S
hippedDate], [Orders].[ShipVia], [Orders].[Freight], [Orders].[ShipName], [Orders].[ShipAddress], [Orders].[ShipCity], [Orders].[ShipRegion], [Orders].[ShipPostalC
ode], [Orders].[ShipCountry] FROM [Orders] WHERE [Orders].[OrderID] = %s' - PARAMS = (11076,); args=(11076,)
web 1 | (0.001) QUERY = 'SELECT [Products].[ProductID], [Products].[ProductName], [Products].[SupplierID], [Products].[CategoryID], [Products].[QuantityPerU
nit], [Products].[UnitPrice], [Products].[UnitsInStock], [Products].[UnitsOnOrder], [Products].[ReorderLevel], [Products].[Discontinued] FROM [Products]' - PARAMS
= (); args=()
web 1 | (0.001) QUERY = 'SELECT [Products].[ProductID], [Products].[ProductName], [Products].[SupplierID], [Products].[CategoryID], [Products].[QuantityPerU
nit], [Products].[UnitPrice], [Products].[UnitsInStock], [Products].[UnitsOnOrder], [Products].[ReorderLevel], [Products].[Discontinued] FROM [Products] WHERE [Pro
ducts].[ProductID] = %s' - PARAMS = (19,); args=(19,)
web 1 | (0.000) QUERY = 'SELECT [Orders].[OrderID], [Orders].[CustomerID], [Orders].[EmployeeID], [Orders].[OrderDate], [Orders].[RequiredDate], [Orders].[S
hippedDate], [Orders].[ShipVia], [Orders].[Freight], [Orders].[ShipName], [Orders].[ShipAddress], [Orders].[ShipCity], [Orders].[ShipRegion], [Orders].[ShipPostalC
ode], [Orders].[ShipCountry] FROM [Orders] WHERE [Orders].[OrderID] = %s' - PARAMS = (11076,); args=(11076,)
web 1 | (0.000) QUERY = 'SELECT [Products].[ProductID], [Products].[ProductName], [Products].[SupplierID], [Products].[CategoryID], [Products].[QuantityPerU
nit], [Products].[UnitPrice], [Products].[UnitsInStock], [Products].[UnitsOnOrder], [Products].[ReorderLevel], [Products].[Discontinued] FROM [Products]' - PARAMS
= (); args=()
web 1 | (0.001) QUERY = 'SELECT [Products].[ProductID], [Products].[ProductName], [Products].[SupplierID], [Products].[CategoryID], [Products].[QuantityPerU
nit], [Products].[UnitPrice], [Products].[UnitsInStock], [Products].[UnitsOnOrder], [Products].[ReorderLevel], [Products].[Discontinued] FROM [Products]' - PARAMS
= (); args=()
web 1 | (0.001) QUERY = 'SELECT SYSDATETIME()' - PARAMS = (); args=None
web 1 | (0.001) QUERY = 'SELECT [django_session].[session_key], [django_session].[session_data], [django_session].[expire_date] FROM [django_session] WHERE
([django_session].[expire_date] > %s AND [django_session].[session_key] = %s)' - PARAMS = (datetime.datetime(2021, 1, 8, 6, 15, 1, 773373), 'mgh2qy8ns4106dkabkhw18
4qxvwachwa'); args=(datetime.datetime(2021, 1, 8, 6, 15, 1, 773373), 'mgh2qy8ns4106dkabkhw184qxvwachwa')
web 1 | (0.001) QUERY = 'SELECT [auth_user].[id], [auth_user].[password], [auth_user].[last_login], [auth_user].[is_superuser], [auth_user].[username], [aut
h_user].[first_name], [auth_user].[last_name], [auth_user].[email], [auth_user].[is_staff], [auth_user].[is_active], [auth_user].[date_joined] FROM [auth_user] WHE
RE [auth_user].[id] = %s' - PARAMS = (1,); args=(1,)
web 1 | (0.001) QUERY = 'SELECT COUNT BIG(*) AS [count] FROM [Orders]' - PARAMS = (); args=()
web 1 | (0.001) QUERY = 'SELECT COUNT BIG(*) AS [count] FROM [Orders]' - PARAMS = (); args=()
web 1 | (0.001) QUERY = 'SELECT TOP 100 [Orders].[OrderID], [Orders].[CustomerID], [Orders].[EmployeeID], [Orders].[OrderDate], [Orders].[RequiredDate], [Or
ders].[ShippedDate], [Orders].[ShipVia], [Orders].[Freight], [Orders].[ShipName], [Orders].[ShipAddress], [Orders].[ShipCity], [Orders].[ShipRegion], [Orders].[Shi
pPostalCode], [Orders].[ShipCountry] FROM [Orders] ORDER BY [Orders].[OrderID] DESC' - PARAMS = (); args=()
web 1 | (0.001) QUERY = 'SELECT SYSDATETIME()' - PARAMS = (); args=None
web 1 | (0.001) QUERY = 'SELECT [django_session].[session_key], [django_session].[session_data], [django_session].[expire_date] FROM [django_session] WHERE
([django_session].[expire_date] > %s AND [django_session].[session_key] = %s)' - PARAMS = (datetime.datetime(2021, 1, 8, 6, 15, 2, 47924), 'mgh2qy8ns4106dkabkhw184
qxvwachwa'); args=(datetime.datetime(2021, 1, 8, 6, 15, 2, 47924), 'mgh2qy8ns4106dkabkhw184qxvwachwa')
web 1 | (0.000) QUERY = 'SELECT [auth_user].[id], [auth_user].[password], [auth_user].[last_login], [auth_user].[is_superuser], [auth_user].[username], [aut
h_user].[first_name], [auth_user].[last_name], [auth_user].[email], [auth_user].[is_staff], [auth_user].[is_active], [auth_user].[date_joined] FROM [auth_user] WHE
RE [auth_user].[id] = %s' - PARAMS = (1,); args=(1,)
```

Podgląd na żywo wykonywanych zapytań do bazy danych.

## Wnioski

# wnioski