Framework e-commerce

Elastyczny szablon sklepu internetowego

Przemysław Magiera

29 listopada 2018

Część I

Zdefiniowane problemy

Architektura sklepów internetowych

Co można powiedzieć o archotekturze i rozszerzalności sklepów internetowych?

- sklepwy nieoparte o frameworki są trudne w utrzymaniu

Architektura sklepów internetowych

Co można powiedzieć o archotekturze i rozszerzalności sklepów internetowych?

- sklepwy nieoparte o frameworki są trudne w utrzymaniu
- dodawanie nowych funkcjonalności jest bardzo kosztowne

Architektura sklepów internetowych

Co można powiedzieć o archotekturze i rozszerzalności sklepów internetowych?

- sklepwy nieoparte o frameworki są trudne w utrzymaniu
- dodawanie nowych funkcjonalności jest bardzo kosztowne
- każda nowa funkcjonalność wymaga implementacji interfejsu do administracji i zarządzania

Wydajność i skalowalność

Jaka jest wydajność sklepów internetowych?

- często klasyczne sklepy wymagają skalowania pionowego, które jest bardzo drogie

Wydajność i skalowalność

Jaka jest wydajność sklepów internetowych?

- często klasyczne sklepy wymagają skalowania pionowego, które jest bardzo drogie
- frameworki nie oferują mechanizmów szybkiego dostępu do najbardziej kluczowych danych - katalog produktowy

Wydajność i skalowalność

Jaka jest wydajność sklepów internetowych?

- często klasyczne sklepy wymagają skalowania pionowego, które jest bardzo drogie
- frameworki nie oferują mechanizmów szybkiego dostępu do najbardziej kluczowych danych - katalog produktowy
- najbardziej obciążone punkty aplikacji nie są odseparowane od reszty

Katalog produktowy

Czy katalogi produktowe są zawsze proste i spełniają swoje zadanie?

- główne zadanie katalogu produktowego to zapewnienie łatwego i szybkiego sposobu na dotarcie do interesującej informacji

Katalog produktowy

Czy katalogi produktowe są zawsze proste i spełniają swoje zadanie?

- główne zadanie katalogu produktowego to zapewnienie łatwego i szybkiego sposobu na dotarcie do interesującej informacji
- brak możliwości konfiguracji i modyfikacji wyszukiwarki

Katalog produktowy

Czy katalogi produktowe są zawsze proste i spełniają swoje zadanie?

- główne zadanie katalogu produktowego to zapewnienie łatwego i szybkiego sposobu na dotarcie do interesującej informacji
- brak możliwości konfiguracji i modyfikacji wyszukiwarki
- trudność w dotarciu do informacji

Stosowane technologie

Czy frameworki są na tyle elastyczne aby nie zostać *w tyle*?

- stosowanie zamkniętych komercyjnych technologii

Stosowane technologie

Czy frameworki są na tyle elastyczne aby nie zostać *w tyle*?

- stosowanie zamkniętych komercyjnych technologii
- programiści nie widzą źródeł i nie mają wpływu na rdzeniowe elementy platformy, na której programują

Stosowane technologie

Czy frameworki są na tyle elastyczne aby nie zostać *w tyle*?

- stosowanie zamkniętych komercyjnych technologii
- programiści nie widzą źródeł i nie mają wpływu na rdzeniowe elementy platformy, na której programują
- w przypadku błędów, nadpisanie komponentow platformy jest bardzo trudne lub nawet niemożliwe

Część II

Zastosowane rozwiązania problemów

Dynamiczny panel administracyjny

Zarządzanie funkcjonalnościami out-of-the-box

- generowana tabelka dla każdej encji danego rodzaju np. kategorii

Dynamiczny panel administracyjny

Zarządzanie funkcjonalnościami out-of-the-box

- generowana tabelka dla każdej encji danego rodzaju np. kategorii
- generowany formularz edycji dowolnej encji

Dynamiczny panel administracyjny

Zarządzanie funkcjonalnościami out-of-the-box

- generowana tabelka dla każdej encji danego rodzaju np. kategorii
- generowany formularz edycji dowolnej encji
- generowany mechanizm zarządzania relacjami dowolnej klasy np. dzieci kategorii

Dynamiczne menu w panelu administracyjnym

Zarządzanie encjami

Przykład

Dodajemy nową klasę do systemu opartego na frameworku, chcemy mieć możliwość zarządzania tą klasą (operacje CRUD). Aby system widział tę klasę (i wszystkie ją rozszerzające) należy dodać rekord do tabeli `admin-menu-item` z nazwą klasy nowej encji. W ten sposób powstały wszystkie funkcjonalności *out-of-the-box*

- Menu w panelu jest oparte na dwóch tabelach `admin-menu-item` oraz `admin-menu-group`

Dynamiczne menu w panelu administracyjnym

Zarządzanie encjami

Przykład

Dodajemy nową klasę do systemu opartego na frameworku, chcemy mieć możliwość zarządzania tą klasą (operacje CRUD). Aby system widział tę klasę (i wszystkie ją rozszerzające) należy dodać rekord do tabeli `admin-menu-item` z nazwą klasy nowej encji. W ten sposób powstały wszystkie funkcjonalności *out-of-the-box*

- Menu w panelu jest oparte na dwóch tabelach `admin-menu-item` oraz `admin-menu-group`
- AdminMenuItem i AdminMenuGroup to też encje, więc konfigurujemy widok panelu admina

Dynamiczne menu w panelu administracyjnym

Zarządzanie encjami

Przykład

Dodajemy nową klasę do systemu opartego na frameworku, chcemy mieć możliwość zarządzania tą klasą (operacje CRUD). Aby system widział tę klasę (i wszystkie ją rozszerzające) należy dodać rekord do tabeli `admin-menu-item` z nazwą klasy nowej encji. W ten sposób powstały wszystkie funkcjonalności *out-of-the-box*

- Menu w panelu jest oparte na dwóch tabelach `admin-menu-item` oraz `admin-menu-group`
- AdminMenuItem i AdminMenuGroup to też encje, więc konfigurujemy widok panelu admina
- efekt jest taki, że każdą encję, którą dodamy do tych tabel będziemy mogli zarządzać z poziomu panelu admina

Zarządzanie nadpisanymi klasami przez panel

Zarządzanie funkcjonalnościami, które mogą zostać dopisane do platformy

Przykład

Programista decyduje się na zaimplementowanie nowej funkcjonalności, która będzie polegała na tym, że każde zamówienie będzie mógło mieć uwagę. W systemie istnieje encja Order, ma już zdefiniowane pola, ma również swój formularz. Nie jest to jednak problem, gdyż w tym przypadku wystarczy zaimplementować klasę `MyOrder extends Order`, która będzie zawierała pole `private String note`. Aplikacja korzystająca z frameworkowego panelu będzie świadoma tego, że encja order została nadpisana i będzie szukać w niej pól z aadnotacją `@AdminVisible`, które zostaną wyświetlone w dynamicznym formularzu i dynamicznej tabelce.

- platforma jest świadoma tego, że klasa którą zarządza panel administracyjny została nadpisana

Zarządzanie nadpisanymi klasami przez panel

Zarządzanie funkcjonalnościami, które mogą zostać dopisane do platformy

Przykład

Programista decyduje się na zaimplementowanie nowej funkcjonalności, która będzie polegała na tym, że każde zamówienie będzie mógło mieć uwagę. W systemie istnieje encja Order, ma już zdefiniowane pola, ma również swój formularz. Nie jest to jednak problem, gdyż w tym przypadku wystarczy zaimplementować klasę `MyOrder extends Order`, która będzie zawierała pole `private String note`. Aplikacja korzystająca z frameworkowego panelu będzie świadoma tego, że encja order została nadpisana i będzie szukać w niej pól z aadnotacją `@AdminVisible`, które zostaną wyświetlone w dynamicznym formularzu i dynamicznej tabelce.

- platforma jest świadoma tego, że klasa którą zarządza panel administracyjny została nadpisana
- generowanie formularza i tabeli odbywa się poprzez wgląd do klas rozszerzających bazową encje

Zarządzanie nadpisanymi klasami przez panel

Zarządzanie funkcjonalnościami, które mogą zostać dopisane do platformy

Przykład

Programista decyduje się na zaimplementowanie nowej funkcjonalności, która będzie polegała na tym, że każde zamówienie będzie mógło mieć uwagę. W systemie istnieje encja Order, ma już zdefiniowane pola, ma również swój formularz. Nie jest to jednak problem, gdyż w tym przypadku wystarczy zaimplementować klasę `MyOrder extends Order`, która będzie zawierała pole `private String note`. Aplikacja korzystająca z frameworkowego panelu będzie świadoma tego, że encja order została nadpisana i będzie szukać w niej pól z aadnotacją `@AdminVisible`, które zostaną wyświetlone w dynamicznym formularzu i dynamicznej tabelce.

- platforma jest świadoma tego, że klasa którą zarządza panel administracyjny została nadpisana
- generowanie formularza i tabeli odbywa się poprzez wgląd do klas rozszerzających bazową encje
- daje to efekt, że właściwie każde pole w encji, którą zarządza platforma i jest zaadnotowane jako `@AdminVisible` jest widziane w tabeli encyjnej i formularzu encyjnym

Zarządzanie relacjami encji

Wyświetlanie i zarządzanie relacjami w dynamicznym formularzu encyjnym

Przykład

Encja `Product` jest w relacji `OneToMany` z encją `Price`. Aby panel administracyjny był w pełni wartościowy, po wejściu do formularza edycyjnego powinniśmy mieć możliwość dodania ceny do produktu. W tym celu należy umieścić adnotację `@AdminVisible(tableVisible = false, className = ćom.example.Price", mappedBy = "product")` nad kolekcją cen. System w panelu administracyjnym wyświetli listę cen w produkcie.

- adnotacja `@AdminVisible(tableVisible = false, className = ćom.example.ClassExample", mappedBy = "przyklad")` odpowiada również za wyświetlanie relacji w dynamicznym formularzu edycji.

Zarządzanie relacjami encji

Wyświetlanie i zarządzanie relacjami w dynamicznym formularzu encyjnym

Przykład

Encja `Product` jest w relacji `OneToMany` z encją `Price`. Aby panel administracyjny był w pełni wartościowy, po wejściu do formularza edycyjnego powinniśmy mieć możliwość dodania ceny do produktu. W tym celu należy umieścić adnotację `@AdminVisible(tableVisible = false, className = ćom.example.Price", mappedBy = "product")` nad kolekcją cen. System w panelu administracyjnym wyświetli listę cen w produkcie.

- adnotacja `@AdminVisible(tableVisible = false, className = ćom.example.ClassExample", mappedBy = "przyklad")` odpowiada również za wyświetlanie relacji w dynamicznym formularzu edycji.
- obsugiwane są relacje OneToMany jak i ManyToMany - a to nie jest oczywiste!

Zarządzanie relacjami encji

Wyświetlanie i zarządzanie relacjami w dynamicznym formularzu encyjnym

Przykład

Encja `Product` jest w relacji `OneToMany` z encją `Price`. Aby panel administracyjny był w pełni wartościowy, po wejściu do formularza edycyjnego powinniśmy mieć możliwość dodania ceny do produktu. W tym celu należy umieścić adnotację `@AdminVisible(tableVisible = false, className = ćom.example.Price", mappedBy = "product")` nad kolekcją cen. System w panelu administracyjnym wyświetli listę cen w produkcie.

- adnotacja `@AdminVisible(tableVisible = false, className = ćom.example.ClassExample", mappedBy = "przyklad")` odpowiada również za wyświetlanie relacji w dynamicznym formularzu edycji.
- obsugiwane są relacje OneToMany jak i ManyToMany - a to nie jest oczywiste!
- wyświetlane relacje można dowolnie modyfikować, usuwać lub dodawać (wszystko jest generowane)

Zarządzanie relacjami encji - jak działa

Jak dokonano implementacji dynamicznego zarządzania relacjami w encji

- Wyświetlenie
- Dodanie
- Usunięcie

Zarządzanie relacjami encji - jak działa

Jak dokonano implementacji dynamicznego zarządzania relacjami w encji

- Wyświetlenie
  - z adnotacji `@AdminVisible` weź nazwę klasy w relacji z encją, dla której generujesz formularz (`className`) oraz nazwę klucza obcego (`mappedBy`)
  - za pomocą dynamicznego dao encyjnego (`DynamicEntityDao`) wyciągnij z bazy danych listę encji powiązanych
- Dodanie
- Usunięcie

Zarządzanie relacjami encji - jak działa

Jak dokonano implementacji dynamicznego zarządzania relacjami w encji

- Wyświetlenie
  - z adnotacji `@AdminVisible` weź nazwę klasy w relacji z encją, dla której generujesz formularz (`className`) oraz nazwę klucza obcego (`mappedBy`)
  - za pomocą dynamicznego dao encyjnego (`DynamicEntityDao`) wyciągnij z bazy danych listę encji powiązanych
- Dodanie
  - znajdź pole w klasie (lub jej pochodnych) odnoszące się do zarządzanej relacji, analogicznie dla drugiej strony
  - Hibernate zwróci PersistentSet'y z encjami relacyjnymi, odpowiednio zmodyfikuj je i zakończ transakcje
  - co tu jest nie tak?
- Usunięcie

Zarządzanie relacjami encji - jak działa

Jak dokonano implementacji dynamicznego zarządzania relacjami w encji

- Wyświetlenie
  - z adnotacji `@AdminVisible` weź nazwę klasy w relacji z encją, dla której generujesz formularz (`className`) oraz nazwę klucza obcego (`mappedBy`)
  - za pomocą dynamicznego dao encyjnego (`DynamicEntityDao`) wyciągnij z bazy danych listę encji powiązanych
- Dodanie
  - znajdź pole w klasie (lub jej pochodnych) odnoszące się do zarządzanej relacji, analogicznie dla drugiej strony
  - Hibernate zwróci PersistentSet'y z encjami relacyjnymi, odpowiednio zmodyfikuj je i zakończ transakcje
  - co tu jest nie tak?
- Usunięcie
  - analogicznie jak dodanie, tylko usuń wartości z setów...

Funkcjonalności biznesowe

What's in the box?

- prawie wszystkie funkcjonalności biznesowe zostały zaimplementowane na podstawie dynamicznego panelu

Funkcjonalności biznesowe

What's in the box?

- prawie wszystkie funkcjonalności biznesowe zostały zaimplementowane na podstawie dynamicznego panelu
- implementacja ich ogarnicza się do stworzenia modelu i napisania funkcji korzystających z tej konkretnej funkcjonalności w kontrolerze/htmlu/komponencie/fasadzie/JS na stronie sklepu/fasadzie

Funkcjonalności biznesowe

What's in the box?

- prawie wszystkie funkcjonalności biznesowe zostały zaimplementowane na podstawie dynamicznego panelu
- implementacja ich ogarnicza się do stworzenia modelu i napisania funkcji korzystających z tej konkretnej funkcjonalności w kontrolerze/htmlu/komponencie/fasadzie/JS na stronie sklepu/fasadzie
- pełna obsługa encji i jej relacji robi się sama

System klasyfikacyjny

Schemat pobierania atrybutów produktowych z systemu kalsyfikacyjnego

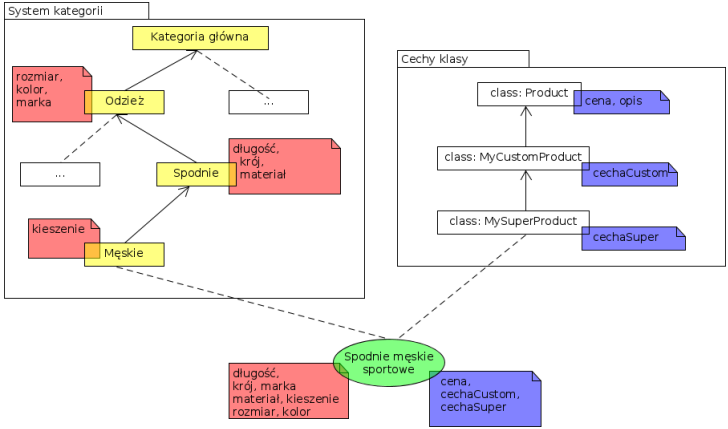
Definicja

Systemu klasyfikacyjny Jest to drzewiasta struktura kategorii połączona z encjami, które mogą przyjmować postać różnych cech produktu, definiować wartości tych cech łączyć je z produktem. Cechy są dzidziczne, zgodnie z ułożeniem drzewa kategorii.

- rysunek 1 bardzo dobrze obrazuje skąd system klasyfikacyjny bierze cechy produktu

System klasyfikacyjny

Schemat pobierania atrybutów produktowych z systemu kalsyfikacyjnego



Rysunek: Diagram przykładowy pochodzenia atrybutów produktu

[TODO] System uprawnień

Jak będzie działać system uprawnień w panelu administracyjnymm

- użytkownik administracyjny będzie miał kolekcję uprawnień

[TODO] System uprawnień

Jak będzie działać system uprawnień w panelu administracyjnymm

- użytkownik administracyjny będzie miał kolekcję uprawnień
- struktura systemu uprawnień będzie podobna do struktury kategorii, uprawnienia będą mogły po sobie dzidziczyć

[TODO] System uprawnień

Jak będzie działać system uprawnień w panelu administracyjnymm

- użytkownik administracyjny będzie miał kolekcję uprawnień
- struktura systemu uprawnień będzie podobna do struktury kategorii, uprawnienia będą mogły po sobie dzidziczyć
- AdminMenuGroup i AdminMenuItem będą miały kolekcje encji 'uprawnienie', podczas *renderu* menu, będą brane pod uwagę, względem tego, kto jest aktualnie zalogowany

Klasyczne funkcjonalności

Inne (niekoniecznie ciekawe) funkcjonalności, które muszą być w sklepie

- koszyk
- logowanie jako uzytkownik sklepu
- możliwość dokonania zakupu
- zarządzanie zamówieniami/użytkownikami/adresami
- i pare innych...

Mechanizm indeksująco-wyszukujący

Integracja płaskiej bazy no-sql z bazą relacyjną

- do zapytań związanych z katalogiem produktowym użyto serwer Apache Solr 7.5 oparty na silniku Lucene

Mechanizm indeksująco-wyszukujący

Integracja płaskiej bazy no-sql z bazą relacyjną

- do zapytań związanych z katalogiem produktowym użyto serwer Apache Solr 7.5 oparty na silniku Lucene
- Solr umożliwia dobrą i wszechstronną analizę danych tekstowych, podświetlanie, podpowiadanie i facetowanie.

Mechanizm indeksująco-wyszukujący

Integracja płaskiej bazy no-sql z bazą relacyjną

- do zapytań związanych z katalogiem produktowym użyto serwer Apache Solr 7.5 oparty na silniku Lucene
- Solr umożliwia dobrą i wszechstronną analizę danych tekstowych, podświetlanie, podpowiadanie i facetowanie.
- (TODO) co jakiś czas uruchamiany jest Job w systemie, który synchronizuje bazę relacyjną z Solrem (na razie za każdym requestem)

Mechanizm indeksująco-wyszukujący

Integracja płaskiej bazy no-sql z bazą relacyjną

- do zapytań związanych z katalogiem produktowym użyto serwer Apache Solr 7.5 oparty na silniku Lucene
- Solr umożliwia dobrą i wszechstronną analizę danych tekstowych, podświetlanie, podpowiadanie i facetowanie.
- (TODO) co jakiś czas uruchamiany jest Job w systemie, który synchronizuje bazę relacyjną z Solrem (na razie za każdym requestem)
- Indeksacja to wydobycie atrybutów wyszukiwalnych i facetowalnych z produktu, zapakowanie w dokumenty i wysłane na serwer Solr - wyciągnięcie atrybutów pokazuje rysunek 1

Indeksacja atrybutów produktu

Jak zachowano elastyczność?

- atrybuty niepochodzące z systemu klasyfikacyjnego są również indeksowane

Indeksacja atrybutów produktu

Jak zachowano elastyczność?

- atrybuty niepochodzące z systemu klasyfikacyjnego są również indeksowane
- pola w klasach rozszerzających `Product` (np. `MyProduct extends Product`) są również możliwe do zaindeksowania, wystarczy że z poziomu panelu administracyjnego dodamy do tabeli SearchField nazwę pola, reszta zrobi się sama

Część III

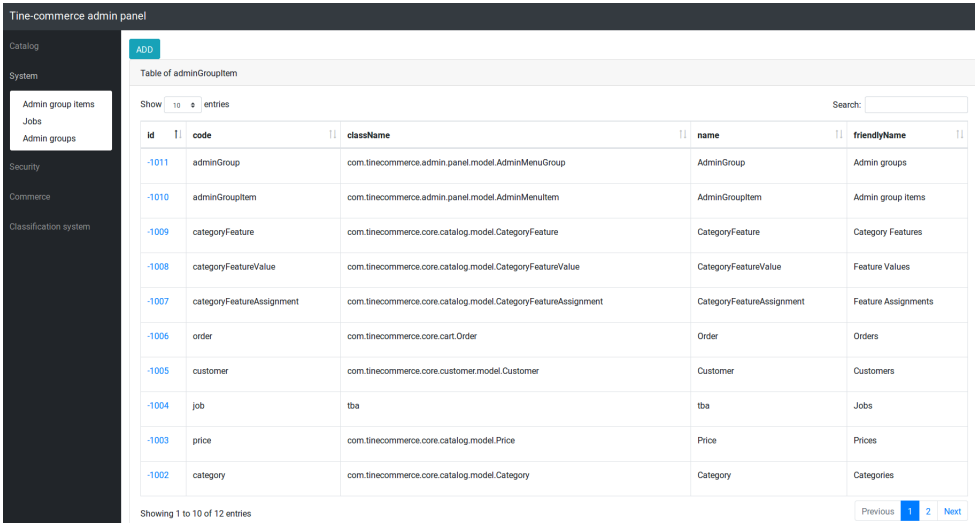
Przykłady działania

Dodanie encji do systemu

```
public class Price extends AbstractEntity {

    @ManyToOne
    @JoinColumn(name = "product_id")
    private Product product;

    @Column(name = "currency")
    private Currency currency;

    @Column(name = "amount")
    private BigDecimal amount;
}
```

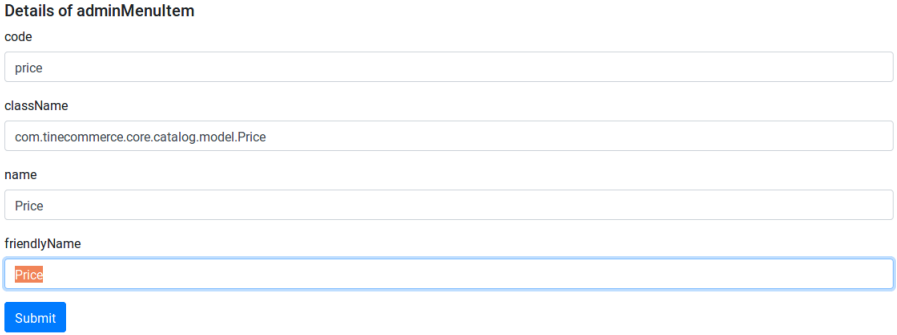
Rysunek: Klasa Price

Dodanie encji do systemu



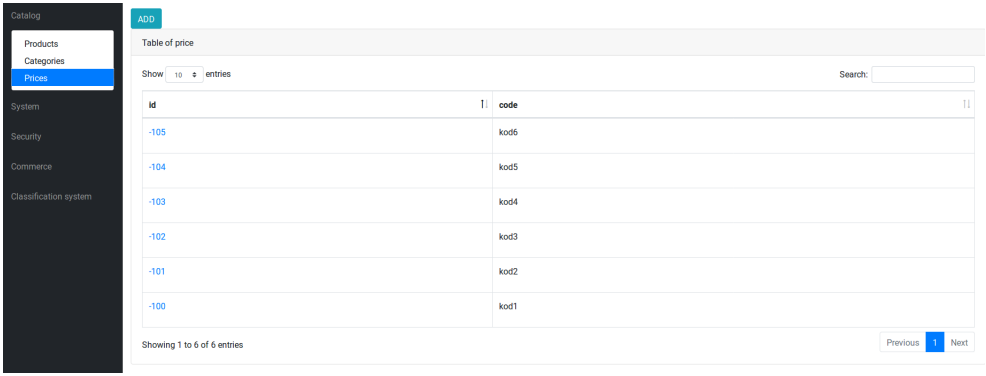
Rysunek: Dodanie klasy Price do obsługi systemu

Dodanie encji do systemu



Rysunek: Podstawowa tabelka encyjna - formularz

Dodanie encji do systemu

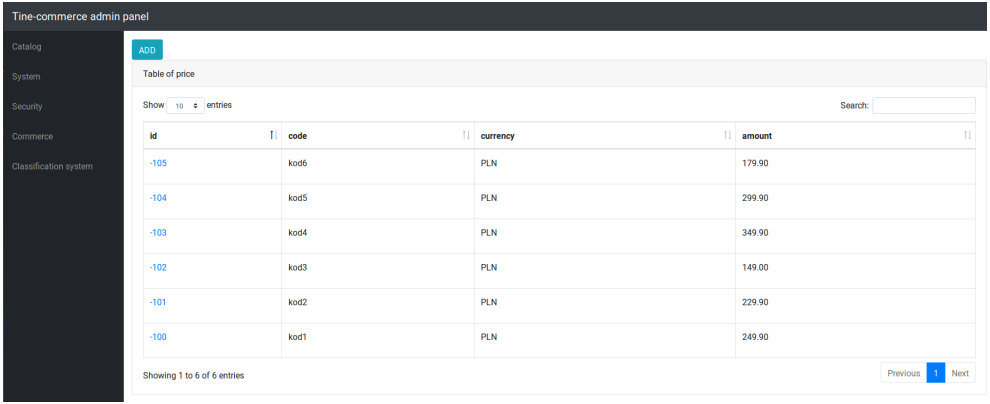


Rysunek: Podstawowa tabelka encyjna

```
public class Price extends AbstractEntity {

    @ManyToOne
    @JoinColumn(name = "product_id")
    private Product product;

    @Column(name = "currency")
    @AdminVisible
    private Currency currency;

    @Column(name = "amount")
    @AdminVisible
    private BigDecimal amount;
}
```

Rysunek: Dopisanie adnotacji @AdminVisible nad polami ceny

Dodanie encji do systemu



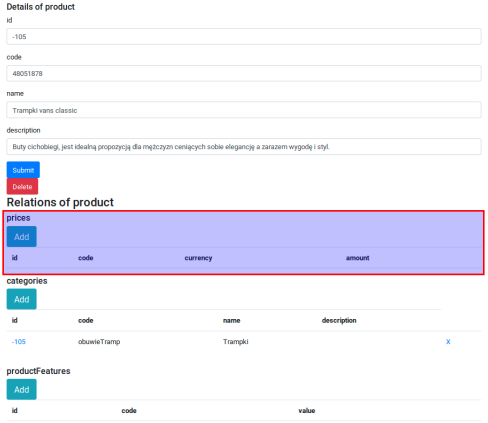
Rysunek: Wyświetlenie zaadnotowanych pól

Dodanie encji do systemu

```
public class Product extends AbstractNameableEntity {

    @Setter(AccessLevel.NONE)
    @OneToMany(cascade = CascadeType.ALL, orphanRemoval = true, mappedBy = "product")
    @AdminVisible(tableVisible = false, className = "com.tinecommerce.core.catalog.model.Price")
    private Set<Price> prices;

    @Setter(AccessLevel.NONE)
    @OneToMany(cascade = CascadeType.ALL, orphanRemoval = true, mappedBy = "product")
    @AdminVisible(tableVisible = false, className = "com.tinecommerce.core.catalog.model.ProductFeature")
    private Set<ProductFeature> productFeatures;

    @Setter(AccessLevel.NONE)
    @ManyToMany(mappedBy = Category.FIELD_PRODUCTS, cascade = CascadeType.ALL, fetch = FetchType.LAZY)
    @AdminVisible(tableVisible = false, className = "com.tinecommerce.core.catalog.model.Category", mappedBy = "products")
    private Set<Category> categories = new HashSet<>();
```

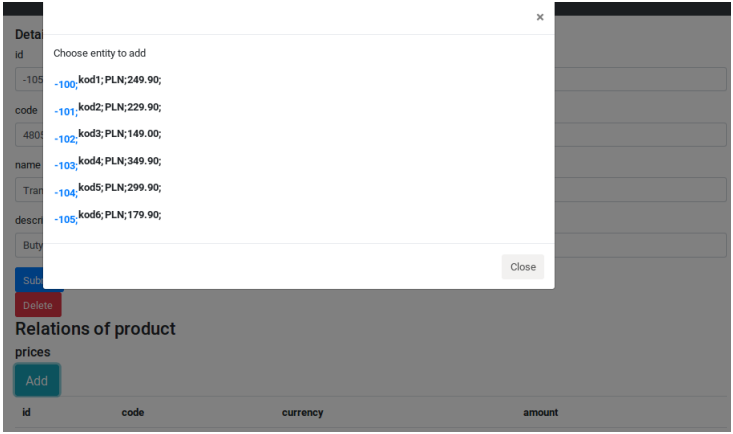
Rysunek: Naniesienie adnotacji admin visible nad kolekcją klasy produkt

Dodanie encji do systemu



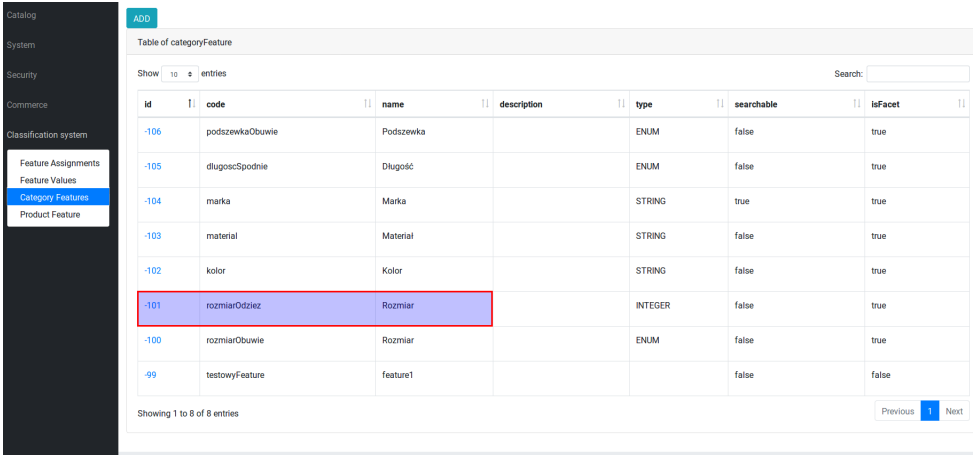
Rysunek: Wygenerowany formularz zmiany relacji

Dodanie encji do systemu



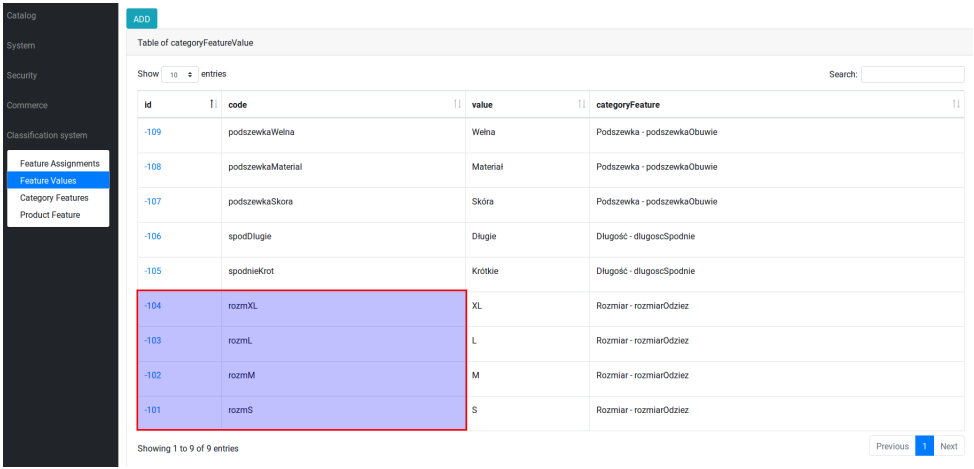
Rysunek: Zarządzanie relacją

Dodanie atrybutu klasyfikacyjnego



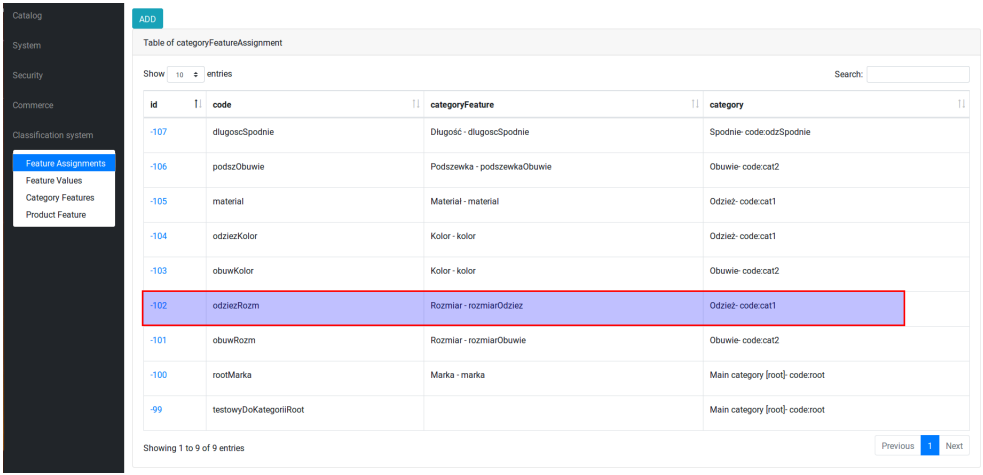
Rysunek: Dodanie nowego atrybutu klasyfikacyjnego

Dodanie atrybutu klasyfikacyjnego



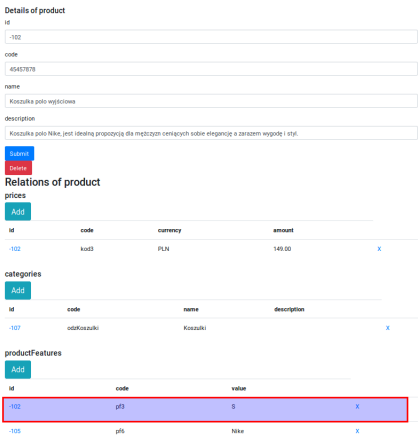
Rysunek: Dodanie jego wariantów

Dodanie atrybutu klasyfikacyjnego



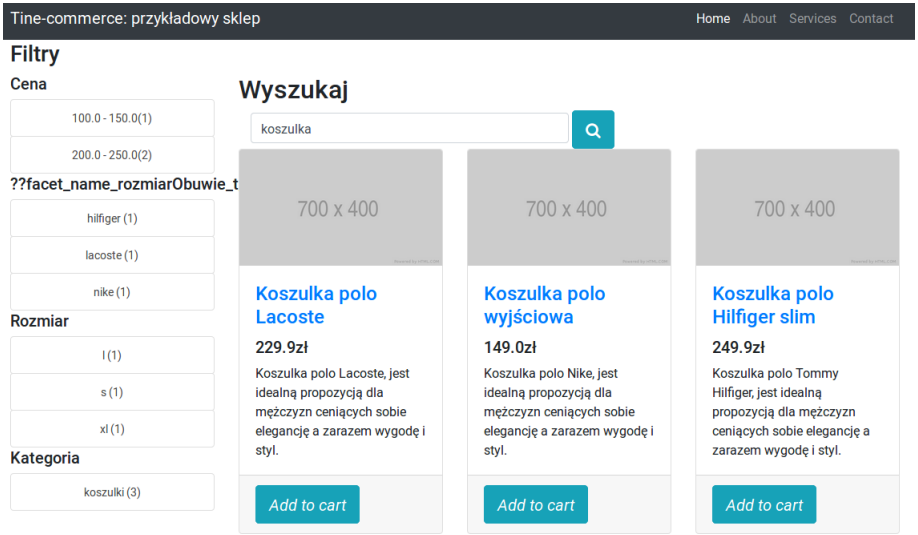
Rysunek: Przypisanie atrybutu do kategorii

Dodanie atrybutu klasyfikacyjnego



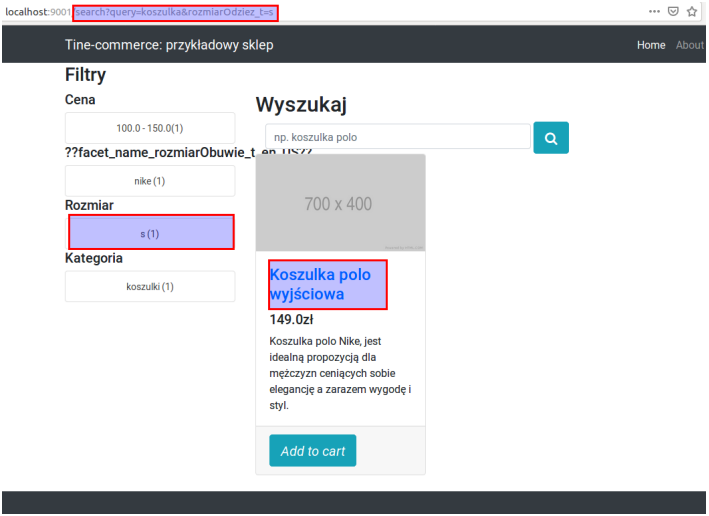
Rysunek: Pojawienie się atrybutu w produkcie, który jest dzieckiem kategorii do której został przypisany atrybut

Dodanie atrybutu klasyfikacyjnego



Rysunek: Wyszukiwanie koszulek i wyświetlanie facetów

Dodanie atrybutu klasyfikacyjnego



Rysunek: Przefiltrowanie koszulki po rozmiarze

Dziękuje za uwagę