

System sprzedaży biletów na imprezy masowe

Spis treści

Założenia projektowe	3
Wymagania funkcjonalne	3
Wymagania niefunkcjonalne	3
Harmonogram prac	4
Baza danych	4
Diagramy UML	6
Diagram przypadków użycia	6
Architektura serwera	6

Założenia projektowe

System pozwala na prowadzenie internetowej sprzedaży biletów na imprezy masowe, np. koncerty. Internauci mogą przeglądać planowane wydarzenia i dokonać zakupu biletu on-line przy pomocy płatności elektronicznych. Klienci otrzymują zakupiony bilet w postaci elektronicznej (pdf). Weryfikacja biletu jest przeprowadzana za pomocą elektronicznych czytników kodów zainstalowanych w bramkach wejściowych lub w wersji przenośnej obsługiwanej przez uprawniony personel.

System został oparty na darmowej aplikacji *OpenCart*, udostępnianej na licencji otwartego oprogramowania. Jej głównymi zaletami jest szybkość działania i intuicyjny panel administracyjny posiadający znaczne możliwości konfiguracji i przystosowania do sprzedaży każdego rodzaju produktów. Panel ten jest bardzo prosty w obsłudze nawet dla początkującego użytkownika. Aplikacja posiada budowę opartą na modelu MVC i została zaimplementowana w języku php.

Aplikacja *OpenCart* została wybrana, ponieważ jest to rozwiązanie dobrze przetestowane, z obszerną dokumentacją, dużą liczbą dodatków oraz z bardzo dobrym wsparciem społeczności.

Wymagania funkcjonalne

Tabela 1. Wymagania funkcjonalne

Nr	Wymaganie	Aktor
1	Użytkownik może przeszukać zbiór wszystkich wydarzeń lub wybrać tylko ich podzbiór.	Użytkownik
2	System udostępnia odpowiednie narzędzia do pobrania biletu.	Użytkownik
3	System pozwala dokonać płatności za zamówienie.	Użytkownik
4	Użytkownik może założyć konto w systemie.	Użytkownik, Administrator
5	Każde zamówienie oznaczona jest unikatowym identyfikatorem, przypisanym do konta użytkownika.	
6	System udostępnia możliwość pobrania faktury przez użytkownika.	Użytkownik
7	System pozwala na dodanie jednego lub wielu biletów do zamówienia.	Użytkownik
8	System udostępnia odpowiednie narzędzia do składania zamówienia.	Użytkownik
9	System udostępnia panel administracyjny do zarządzania sklepem.	Administrator
10	System pozwala na dodawanie nowych wydarzeń	Administrator, Pracownik
11	System pozwala na wyświetlenie statystyk sprzedaży	Administrator, Pracownik
12	System pozwala na modyfikacje uprawnień grup w systemie	Administrator

Wymagania niefunkcjonalne

Tabela 2. Wymagania niefunkcjonalne

Nr	Wymaganie
1	System oparty o oprogramowanie OpenCart.
2	System wykorzystuje protokół SSL.
3	System działa w najnowszych wersjach przeglądarek Mozilla Firefox, Google

	Chrome, Opera, Microsoft Internet Explorer.
4	Sklep jest w języku polskim
5	Sklep obsługuje walutę polską - złotówki

Harmonogram prac

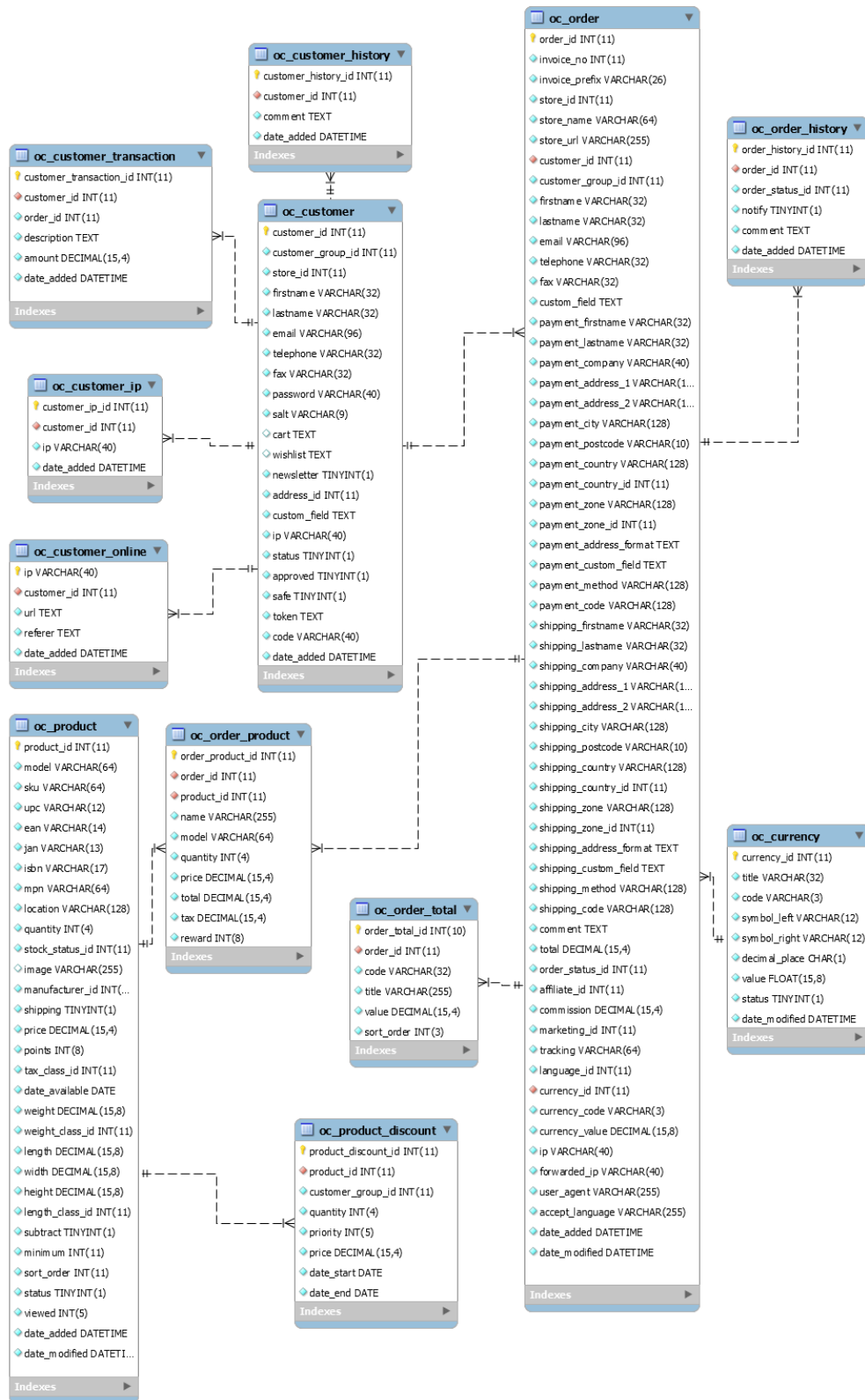
Tabela 3. Harmonogram prac

Data	Opis
15.03.16	Przygotowanie założeń projektowych oraz listy wymagań funkcjonalnych i нефункциональных.
29.03.16	Wdrożenie podstawowej wersji aplikacji OpenCart.
12.04.16	Instalacja i konfiguracja wybranych modułów.
26.04.16	Wdrożenie wybranych zabezpieczeń.
10.05.16	Testy sklepu internetowego
24.05.16	Naprawa wykrytych luk w bezpieczeństwie programu.
07.06.16	Oddanie projektu.

Baza danych

Poniższy schemat bazy danych został wygenerowany programem MySQL Workbench na podstawie bazy danych aplikacji OpenCart. Baza danych posiada 126 tabel. Ze względu na złożoność bazy danych, poniżej przedstawiony został wyłącznie jej najważniejszy fragment.

Na rys. 1 przedstawiono fragment bazy danych, który zawiera tabele ważne z punktu widzenia sklepu internetowego oraz część ich relacji. Tabele te przechowują informacje na temat zamówień, produktów oraz klientów, a także innych danych powiązanych z nimi - historia zamówień, dane o rabatach czy dane o adresach IP klienta.

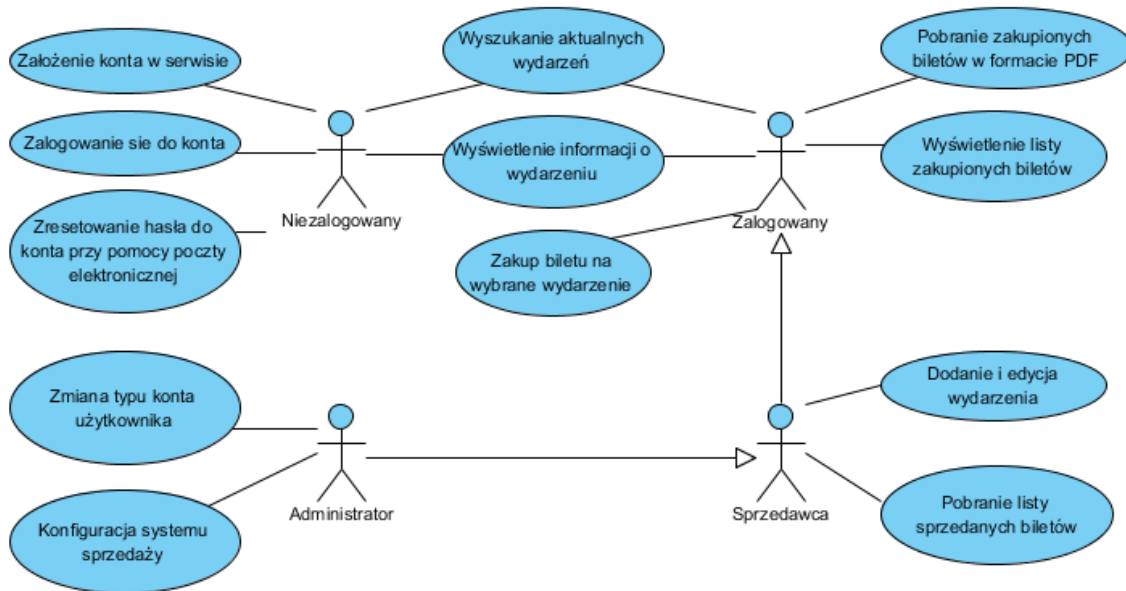


Rysunek 1. Schemat przedstawiający niektóre tabele bazy danych oraz ich relacje

Diagramy UML

Diagram przypadków użycia

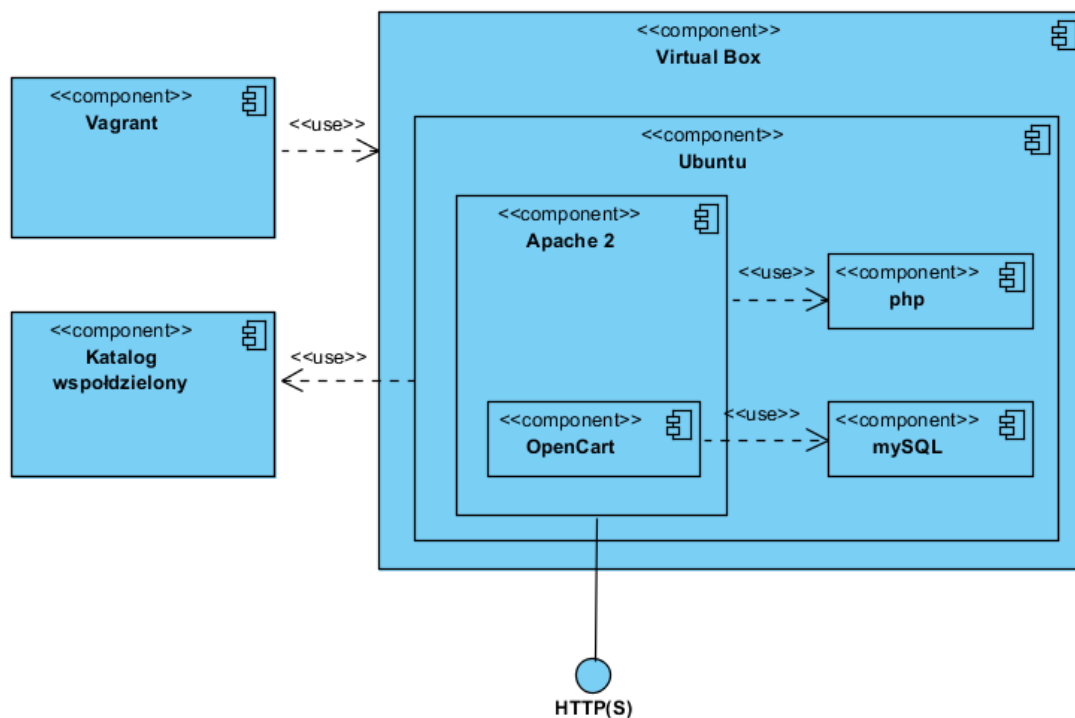
Rysunek 2 obrazuje, przy pomocy notacji UML, dane zawarte w wymaganiach funkcjonalnych opisanych w tabeli 1.



Rysunek 2. Diagram przypadków użycia

Architektura serwera

Rysunek 3 przedstawia diagram UML przedstawiający architekturę serwera aplikacji.



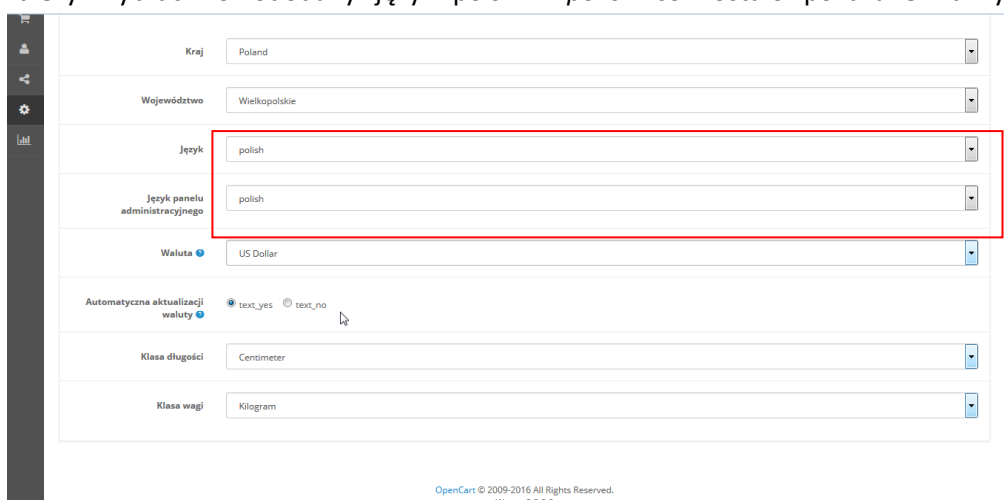
Rysunek 3. Architektura serwera

Konfiguracja sklepu

Język polski

W sklepie został zainstalowany język polski, który nie jest dostępny w wersji podstawowej aplikacji.

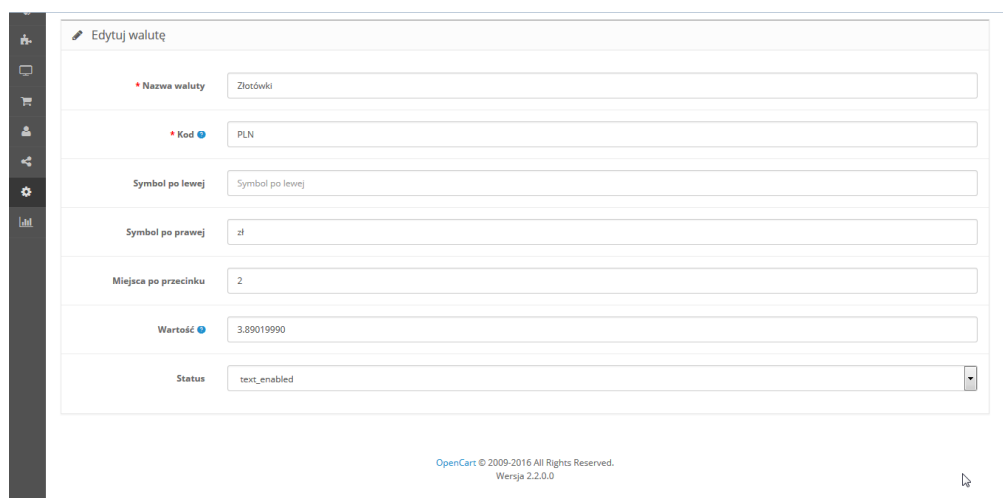
1. Do katalogów aplikacji *admin/language/* należy skopiować pliki z tłumaczeniem do katalogu *admin/language/pl-PL*.
2. Do katalogów aplikacji *catalog/language/* należy skopiować pliki z tłumaczeniem do katalogu *catalog/language/pl-PL*.
3. W ustawieniach sklepu *Settings->Store->Local*, w polach *Language*, *Administrator language* należy wybrać nowododany język polski - *polish* co zostało pokazane na rysunku 4.



Rysunek 4. Wybór języka

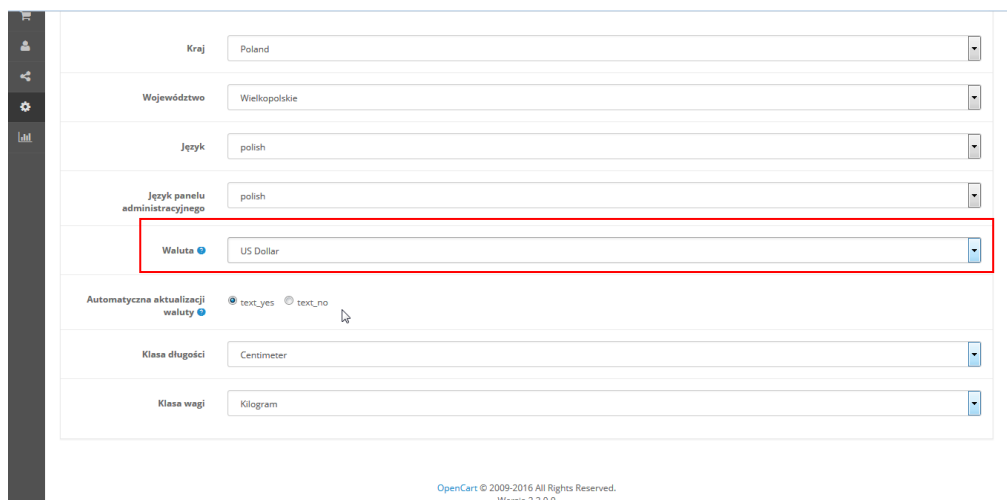
Waluta

W podstawowej konfiguracji sklepu dostępne są trzy waluty: euro, dollars oraz funty. Aby zmienić walutę na inną niż standardowa należy najpierw skonfigurować nową walutę w zakładce *Ustawienia->Lokalizacja->Waluta*, co widoczne jest na rysunku 5.



Rysunek 5. Konfiguracja nowej waluty

Następnie w celu zmiany waluty używanej w sklepie należy przejść do zakładki *Ustawienia->Sklep->Lokalne* i w polu waluta wybrać odpowiednią opcję - złotówki, co zostało przedstawione na rysunku 6.

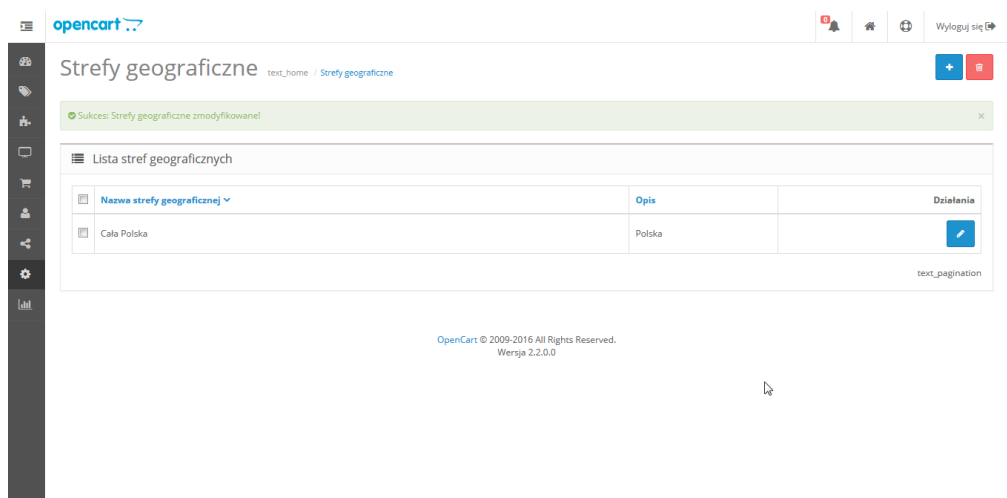


OpenCart © 2009-2016 All Rights Reserved.
Wersja 2.2.0.0

Rysunek 6. Wybór waluty

Podatki

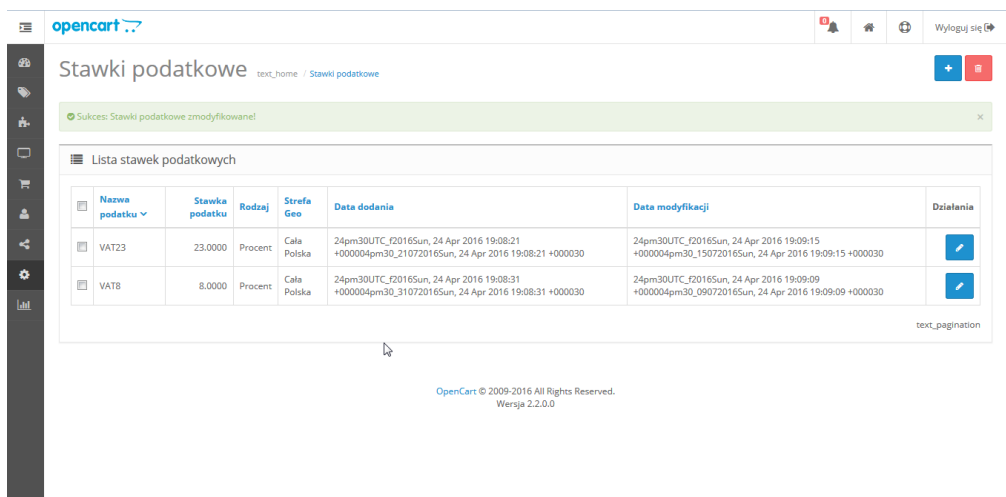
W celu skonfigurowanie odpowiednich podatków najpierw należy ustawić strefy geograficzne, których dotyczyć będą wprowadzane podatki. W celu dodania nowej strefy należy przejść do zakładki *Ustawienia->Lokalizacja->Strefy geograficzne*. Lista przykładowych stref widoczna jest na rysunku 7.



OpenCart © 2009-2016 All Rights Reserved.
Wersja 2.2.0.0

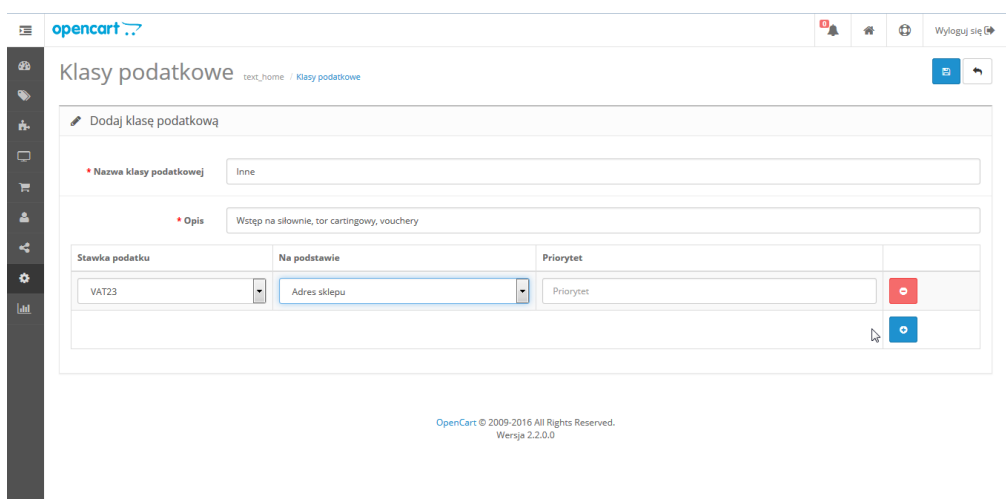
Rysunek 7. Lista skonfigurowanych stref geograficznych

Po wprowadzeniu odpowiednich stref geograficznych należy dodać stawki podatków. W przypadku sprzedaży biletów należy wprowadzić dwie stawki podatku VAT - 8% oraz 23%. Można to zrobić wybierając z menu *Ustawienia->Lokalizacja->Podatki->Stawki podatkowe*. Lista stawek po skonfigurowaniu widoczna jest na rysunku 8.



Rysunek 8. Lista stawek podatkowych po skonfigurowaniu

Ostatnim krokiem przy konfiguracji podatków jest wprowadzenie klas podatkowych. W przypadku sprzedaży biletów na koncerty, występy kabaretowe oraz widowiska artystyczne podatek VAT wynosi 8%. W przypadku sprzedaży voucherów, biletów do siłowni czy na tor kartingowy podatek ten wynosi 23%. Konfiguracja klasy podatkowej została przedstawiona na rysunku 9.



Rysunek 9. Konfigurowanie klasy podatkowej

Ceny brutto

Domyślnie wprowadzając produkt do sklepu internetowego zbudowanego na aplikacji *OpenCart* należy podać cenę netto. Jest to dość problematyczne przy cenach biletów, gdyż organizatorzy podają ceny biletów jako ceny brutto. Aby móc wprowadzać towary w cenach brutto należy:

1. edytować plik `admin\language\polish\catalog\product.php` ;
2. odnaleźć w nim linijkę `$_['entry_price'] = 'Cena:'` ;
3. dodać za nią następującą linię `$_['entry_price_gross'] = 'Cena brutto:'` ;
4. edytować plik `admin\view\template\catalog\product_form.tpl`

5. za kodem

```
<div class="warning"><?php echo $error_warning; ?></div>
<?php } ?>
```

6. należy dodać

```
<script type="text/javascript">
var tax_rates = new Array();
<?php foreach ($tax_classes as $tax_class) { ?>
tax_rates["<?php echo $tax_class['tax_class_id']; ?>"] = <?php echo
$tax_class['rate']; ?>;
<?php } ?>

function doRound(x, places) {
    return Math.round(x * Math.pow(10, places)) / Math.pow(10, places);
}

function getTaxRate() {
    var selected_value =
document.forms["form"].tax_class_id.selectedIndex;
    var parameterVal =
document.forms["form"].tax_class_id[selected_value].value;

    if ((parameterVal > 0) && (tax_rates[parameterVal] > 0)) {
        return tax_rates[parameterVal];
    } else {
        return 0;
    }
}

function updateGross() {
    var taxRate = getTaxRate();
    var grossValue = document.forms["form"].price.value;

    if (taxRate > 0) {
        grossValue = grossValue * ((taxRate / 100) + 1);
    }

    document.forms["form"].price_gross.value = doRound(grossValue, 4);
}

function updateNet() {
    var taxRate = getTaxRate();
    var netValue = document.forms["form"].price_gross.value;

    if (taxRate > 0) {
        netValue = netValue / ((taxRate / 100) + 1);
    }

    document.forms["form"].price.value = doRound(netValue, 4);
}
</script>
```

7. następnie należy odnaleźć kod:

```
<tr>
    <td><?php echo $entry_price; ?></td>
```

```

        <td><input type="text" name="price" value="<?php echo $price;
?>" /></td>
</tr>

```

i zamienić go na

```

<tr>
    <td><?php echo $entry_price; ?></td>
    <td><input type="text" name="price" value="<?php echo $price;
?>" onKeyUp="updateGross()" /></td>
</tr>

```

oraz dodać za nim

```

<tr bgcolor="#f7f7f7">
    <td><?php echo $entry_price_gross; ?></td>
    <td><input type="text" name="price_gross" value="<?php echo
$price; ?>" OnKeyUp="updateNet()" /></td>
</tr>

```

8. w tym samym pliku należy odnaleźć kod `<script type="text/javascript"><!--`
`function addRelated() {`

i zamienić go na

```

<script type="text/javascript"><!--
updateGross();

function addRelated() {

```

9. w kolejnym kroku należy edytować plik `admin\controller\catalog\product.php`

10. za linią `$this->data['entry_price'] = $this->language->get('entry_price');`

należy dodać kod

```

$this->data['entry_price_gross'] = $this->language->get('entry_price_gross');

```

11. w tym samym pliku należy odnaleźć kod

```

$this->data['tax_classes'] = $this->model_localisation_tax_class->getTaxClasses();

```

i dodać za nim

```

for ($i=0, $n=sizeof($this->data['tax_classes']); $i<$n; $i++) {
    if ($this->data['tax_classes'][$i]['tax_class_id'] > 0) {
        $r = $this->model_localisation_tax_class->getTaxRates($this->data['tax_classes'][$i]['tax_class_id']);
        $this->data['tax_classes'][$i]['rate'] = $r[0]['rate'];
    }
}

```

Po wykonaniu tych czynności w panelu administracyjnym przy dodawaniu oraz edycji produktu powinno się pokazywać pole *Cena brutto*. Po wprowadzeniu ceny do tego pola cena netto zostaje obliczona automatycznie.

Certyfikat SSL

Dla sklepu został wygenerowany odpowiedni certyfikat SSL. W celu skorzystania z tego certyfikatu został on dodany do konfiguracji serwera *Apache*. Po skonfigurowaniu serwera WWW, w sklepie włączona została opcja korzystania z połączenia bezpiecznego wykorzystującego wygenerowany certyfikat SSL.

Bilety

Do sklepu internetowego został napisany dodatek pozwalający na generowanie biletów w formacie PDF. Przykładowy bilet wygenerowany za pomocą tego narzędzia został przedstawiony na rysunku 10.



Rysunek 10. Przykładowy bilet wygenerowany ze sklepu internetowego

Na bilecie zostały umieszczone takie informacje jak:

- numer rezerwacji,
- numer biletu,
- kod QR,
- nazwa, adres i data wydarzenia,
- numer zamówienia,
- wykupione miejsca,

- regulamin korzystania z biletu,
- informacje kontaktowe do sklepu.