Politechnika Świętokrzyska

Wydział Elektrotechniki, Automatyki i Informatyki

Technologie obiektowe

Projekt

Temat: Narzędzia odwzorowania obiektowo - relacyjnego np. Hibernate, Oracle TopLink, Microsoft ADO.NET Entity Framework, ActiveRecord lub innego.

Wykonał: Parzniewski Daniel

Kierunek: Informatyka (niestacjonarnie)

Grupa: 1IZ21A

1. Wstęp

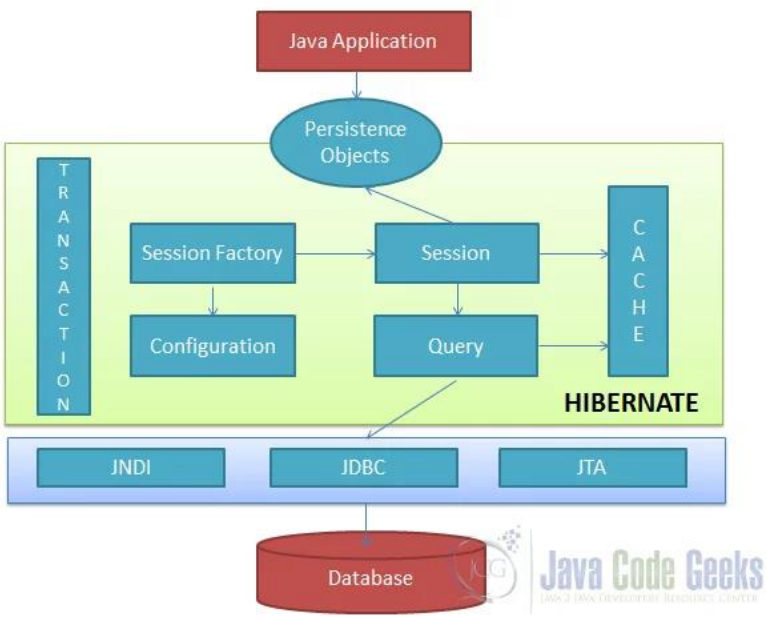
Odwzorowanie obiektowo – relacyjne, nazywane inaczej mapowaniem relacyjno – obiektowym (z ang. Object – relational mapping ORM, O/RM, O/R mapping tool), jest to technika programowania służąca do konwertowania danych przy użyciu języków programowania obiektowego. Tworzona jest wirtualna baza danych obiektów, której można używać z poziomu języka programowania. W projekcie wykorzystano do tego celu narzędzie Hibernate[1] oraz relacyjną bazę danych MySQL[2].

2. Hibernate

Implementacją odwzorowania obiektowo – relacyjnego dla języka Java uznawaną powszechnie za najpopularniejszą jest Hibernate. Cechuje się ona wysoką wydajnością, skalowalnością. Obsługuje asocjacje, kompozycje, dziedziczenie, polimorfizm, mechanizm kolekcji. Jest jedną z implementacji standardu Java Persistence.

2.1 Architektura Hibernate

Na rys. 1 przedstawiono architekturę Hibernate.



Rys. 1. Architektura Hibernate.[3]

[1] https://hibernate.org/

[2] https://www.mysql.com/

[3] https://examples.javacodegeeks.com/wp-content/uploads/2020/04/Hibernate-Architecture-final.png.webp

Configuration – jest to pierwszy obiekt, który zostanie utworzony w aplikacji Hibernate i jest tworzony tylko raz. Zapewnia połączenie bazy danych i struktury mapowania klas;

Session Factory – obiekt bezpieczny wątkowo i tworzący instancje sesji Hibernate. Wymaga obiektu „Configuration". Tworzony podczas uruchamiania aplikacji;

Session – obiekt uzyskiwany każdorazowo w celu wykonania pracy z bazą danych. Nie jest bezpieczny wątkowo;

Querry – reprezentuje zapytania SQL, służące do pobierania lub modyfikowania danych. Obiekt zapytania służy do powiązania parametrów;

Transaction – transakcja reprezentuje pojedynczą jednostkę pracy bazy danych. To jest obiekt opcjonalny;

Cache – Hibernate obsługuje pamięci podręczne na poziomie zapytań i sesji, co podnosi wydajność;

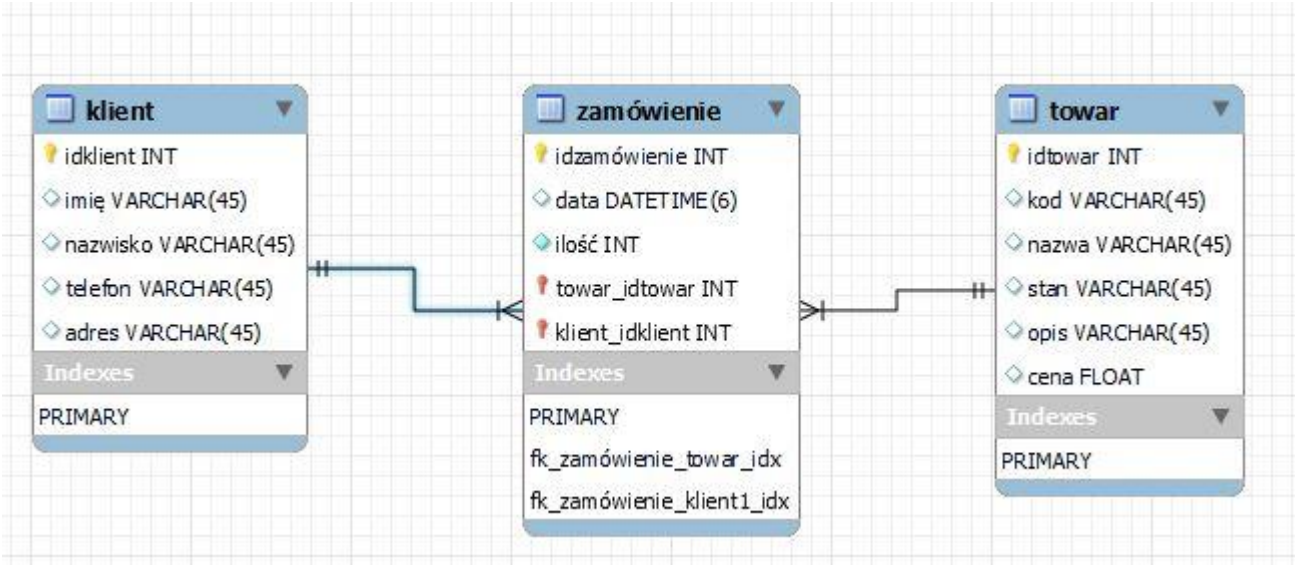
Persistence Objects – obiekty Java, które są odwzorowywane na tabelę relacyjnej bazy danych.

3. MySQL

MySQL to system zarządzania relacyjnymi bazami danych (relational database management system - RDBMS), opracowany i wydany w 1995 roku. Został opracowany i sponsorowany przez firmę MySQL AB, która została następnie przejęta przez firmę Sun Microsystems znaną dziś, jako Oracle Corporation. MySQL to połączenie „My" (córka współzałożyciela) i „SQL".

MySQL był jednym z pierwszych RDBMS o otwartym kodzie źródłowym, jakie kiedykolwiek opracowano i uruchomiono. Obecnie istnieje wiele wariantów MySQL. Jednak podstawowa składnia wszystkich wariacji pozostaje taka sama. Zaprojektowany i napisany w językach programowania C i C++, MySQL jest kompatybilny ze wszystkimi podstawowymi systemami operacyjnymi. Jest to podstawowy składnik szeroko popularnego stosu oprogramowania aplikacji internetowych o otwartym kodzie źródłowym o nazwie LAMP, co oznacza Linux, Apache, MySQL, PHP/Perl/Python.

Poniżej schemat bazy danych użyty w projekcie (rys. 2).



Rys.2. Schemat bazy danych użytej w projekcie.

W projekcie wykorzystano bazę MySQL, nazwaną „shop", zawierającą tabele „*client"* (rysunek 3) oraz „*product*" (rysunek 4).

```
mysql> SELECT * FROM client;
+----+---------------------+-----------------------+-------------------+----------+---------------------+
| id | LastName            | address               | mail              | name     | password            |
+----+---------------------+-----------------------+-------------------+----------+---------------------+
|  1 | Kowalski            | Agrestowa 1,Kielce    | kowalski@mail.pl  | Jan      | kowalski            |
| 10 | Zagajny             | Jesionowa 10,Warszawa | zagajny@mail.pl   | Bogdan   | zagajny             |
|  2 | Nowak               | Boczna 2,Warszawa     | nowak@mail.pl     | Piotr    | nowak               |
|  3 | Brzeczyszczykiewicz | Cicha 3, Gdynia       | grzesiu@mail.pl   | Grzegorz | brzeczyszczykiewicz |
|  4 | Kowalczyk           | Mickiewicza 4,Opole   | kowalczyk@mail.pl | Adam     | kowalczyk           |
|  5 | Krawczyk            | Rynek 5, Szczecin     | krawczyk@mail.pl  | Alina    | krawczyk            |
|  6 | Boczek              | Cwiartki 3/6,Wroclaw  | boczek@mail.pl    | Arnold   | boczek              |
|  7 | Ochodzki            | Teczowa 7,Warszawa    | rysiek@mail.pl    | Ryszard  | ochodzki            |
|  8 | Jarzabek            | Warszawska 8,Warszawa | lubudubu@mail.pl  | Waclaw   | jarzabek            |
|  9 | Dyrman              | Spacerowa 9, Warszawa | dyrman@mail.pl    | Zdzislaw | dyrman              |
+----+---------------------+-----------------------+-------------------+----------+---------------------+
10 rows in set (0.00 sec)
```

Rys. 3. Tabela MySQL „client".

```
mysql> SELECT * FROM product;
+----+--------+-----------------------------------------------------------------+-------------------+-------+-------+
| id | amount | description                                                     | name              | price | state |
+----+--------+-----------------------------------------------------------------+-------------------+-------+-------+
|  1 |    100 | Mikrokontroler STM32F103RBT6 ARM Cortex-M3 LQFP64               | STM32F103RBT6     |    40 | NULL  |
| 10 |     16 | Czujnik temperatury DS18B20 onewire                             | DS18B20           |    20 | NULL  |
| 11 |     98 | Zegar czasu rzeczywistego RTC DS1307 I2C TWI                    | DS1307            |     6 | NULL  |
| 12 |     74 | Potencjometr cyfrowy DS1804-10  10k Ohm I2C                     | DS1804            |    20 | NULL  |
| 13 |     77 | ICL7667 DIP8 Intersil 2x fast driver mosfet 1,5A                | ICL7667           |     2 | NULL  |
| 14 |    200 | ZESTAW REZYSTORÓW SMD 1206 - 1700 sztuk                         | SMD1206           |    50 | NULL  |
| 15 |     99 | ZESTAW REZYSTORÓW SMD 0805 - 1700 sztuk                         | SMD0805           |    60 | NULL  |
| 16 |     65 | ZESTAW REZYSTORÓW SMD 0603 - 1700 sztuk                         | SMD0603           |    60 | NULL  |
| 17 |     52 | Zestaw diod LED SMD 0805 - 30szt                                | LED0805           |     7 | NULL  |
| 18 |     70 | Kondensatory SMD 1206 - 120 szt                                 | C1206             |    18 | NULL  |
| 19 |      5 | Topnik Flux RMA 223 10ccm                                       | RMA223            |    15 | NULL  |
|  2 |     15 | Mikrokontroler AVR ATmega328P-AU SMD                            | ATmega328P        |    17 | NULL  |
| 20 |     23 | Alkohol izopropylowy IPA 100ml                                  | IPA               |     7 | NULL  |
| 21 |     88 | CYNA DO LUTOWANIA 0,80MM 50G SN45%, PB55%                       | Cyna              |    13 | NULL  |
| 22 |     16 | Plecionka do cyny ZD-180 1 mm x 1,5 m                           | ZD-180            |     6 | NULL  |
| 23 |     41 | Tranzystor IRFZ44N N-MOSFET 55V 49A TO-220                      | IRFZ44N           |     2 | NULL  |
| 24 |     90 | Tranzystor P-MOSFET IRF9540 TO-220                              | IRF9540           |    21 | NULL  |
| 25 |    999 | Tranzystor BSS138 SMD N-MOSFET SOT23 10 SZT                     | BSS138            |     2 | NULL  |
| 26 |     14 | Dioda LED LE2835A 6000K 5 sztuk                                 | LE2835A           |     2 | NULL  |
| 27 |   1000 | Tranzystor BC547 TO-92 NPN 45V 0.1A 5 sztuk                     | BC547             |     4 | NULL  |
| 28 |    900 | Tranzystor PNP 65V/100mA BC556B 10 sztuk                        | BC556             |     2 | NULL  |
| 29 |    199 | Dioda 1N4148 - 150mA/100V  50sztuk                              | 1N4148            |     1 | NULL  |
|  3 |      0 | Wyswietlacz LCD 2×16 znakow zielony HD44780                     | HD44780           |    10 |       |
| 30 |    600 | Wzmacniacz operacyjny TL072 SO-08 - 2 sztuki                    | TL072             |     2 | NULL  |
|  4 |     54 | Wyswietlacz dotykowy TFT LCD 2,4? 240x320px z czytnikiem SD SPI | TFT LCD 2,4       |    55 | NULL  |
|  5 |      5 | Zestaw uruchomieniowy Arduino Atmega2560 Rev.3                  | Arduino MEGA 2560 |    50 | NULL  |
|  6 |      2 | Zestaw uruchomieniowy STM32 NUCLEO-F411RE mbed ARM Cortex-M4    | NUCLEO-F411RE     |   100 | NULL  |
|  7 |      6 | Programator AVR ATB USBASP 4.2                                  | ATB-USBASP        |    30 | NULL  |
|  8 |      1 | Konwerter ATB-USB-RS232                                         | ATB-USB-RS232     |    80 | NULL  |
|  9 |      4 | Modul WiFi ESP8266-01S                                          | ESP8266-01S       |    11 | NULL  |
+----+--------+-----------------------------------------------------------------+-------------------+-------+-------+
30 rows in set (0.00 sec)
```

Rys. 4. Tabela MySQL „product”.

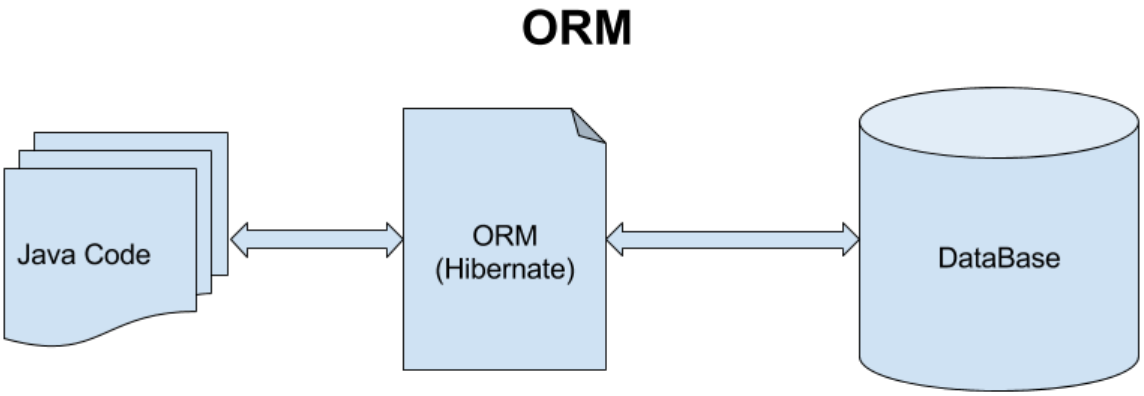
4. Mapowanie obiektowo-relacyjne z wykorzystaniem Hibernate

Mapowanie obiektowo-relacyjne (ang. Object-relational mapping ORM) jest to konwertowanie danych z tabel w relacyjnej bazie danych na obiekty aplikacji klienckiej i na odwrót. Większość dzisiejszych aplikacji jest zorientowanych obiektowo, ale dane przechowuje się wciąż w relacyjnych bazach danych. ORM pozwala na używanie obiektów java, jako reprezentacji relacyjnej bazy danych. To mapuje dwie koncepcje (zorientowane obiektowo i relacyjne). Mapowanie pozwala na:

- zredukowanie ilości pracy związanej z oprogramowaniem dostępu do danych;
- skorzystanie z zalet relacyjnych baz danych jednocześnie nie rezygnując z obiektowości programowania;
- automatyczną obsługę transakcji i połączenia z bazą;

Hibernate jest faworkiem ORM – służy do opisu, w jaki sposób obiekty są reprezentowane w bazie danych, a Hibernate zajmuje się konwersją. Hibernate zwiększa wydajność operacji na bazie danych dzięki buforowaniu i minimalizacji liczby przesyłanych zapytań. Jest to projekt rozwijany jako open source.

JDBC jest API dostępu do bazy danych i działa "w sposób relacyjny" – poprzez odpytywanie tabel i odczyt zawartości wierszy i kolumn. Hibernate używa JDBC pod maską, aby pobrać dane, a następnie przekonwertować je do obiektów. Blokowy schemat mapowania przedstawia rysunek 5.



Rys. 5. Mapowanie ORM.[4]

4.1 Plik konfiguracyjny Hibernate

Konfiguracja zawarta jest w pliku *„hibernate.cfg.xml”*. W nim zawarte są dane konfiguracyjne dotyczące bazy danych na której działamy. W projekcie przedstawia się on następująco:

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE hibernate-configuration PUBLIC "-//Hibernate/Hibernate Configuration DTD
3.0//EN" "http://hibernate.org/dtd/hibernate-configuration-3.0.dtd">
<hibernate-configuration>
    <session-factory>
        <property name="hibernate.connection.driver_class">
com.mysql.jdbc.Driver</property>
        <property name="hibernate.connection.url">
```

[4] https://www.techcret.com/images/resource/orm.png

```
jdbc:mysql://127.0.0.1:3306/shop</property>
        <property name="hibernate.connection.username">shop</property>
        <property name="hibernate.connection.password">shoppass</property>
        <property name="hibernate.show_sql">true</property>
        <property name="hibernate.dialect">
org.hibernate.dialect.MySQL8Dialect</property>
       <!-- <property name="hibernate.hbm2ddl.auto">update</property> -->
    </session-factory>
</hibernate-configuration>
```

Poniżej opis najważniejszych elementów:

connection.driver_class, connection.url, connection.username, connection.password – informacje o połączeniu z bazą danych: sterownik, adres i port, nazwę, nazwę użytkownika i hasło.

dialect – informacja o rodzaju użytej bazy danych.

<mapping resource="nazwa_pliku.hbm.xml"/> – deklaracja wszystkich plików mapujących.

4.2 Sterowanie procesem odwzorowania.

Aktualnie Hibernate wspiera odwzorowanie za pomocą zarówno adnotacji, jak i XML (pliki .hbm.xml). Jednakże ten drugi sposób jest nie jest już zalecany.[5]

- Pliki mapujące XML

  Te pliki zawierają właściwy opis mapowania, czyli tłumaczenia relacyjnego opisu danych na obiektowy i na odwrót. Zazwyczaj jeden plik *.hbm.xml odpowiada jednej klasie bądź tabeli. Dzieje się tak zwłaszcza wtedy, gdy mamy prostą strukturę, gdzie tabele odpowiadają bezpośrednio klasom. Podstawowe elementy pliku mapującego:

```
<?xml version="1.0"?>
<!DOCTYPE hibernate-mapping PUBLIC
      "-//Hibernate/Hibernate Mapping DTD 3.0//EN"
      "http://hibernate.sourceforge.net/hibernate-mapping 3.0.dtd">
<hibernate-mapping>
      <class name=" " table=" ">
            <id name="id" type="int" column="ID">
                  <generator class="increment"/>
            </id>
            <property name=" "/>
            <property name=" " column=" "/>
            <property name=" " column=" "
      </class>
</hibernate-mapping>
```

[5] https://docs.jboss.org/hibernate/orm/current/userguide/html_single/Hibernate_User_Guide.html

<class

name=" "

table=" "

>

Określa klasę i tabelę której dotyczy mapowanie.

<property

name=" "

type=" "

column=" "

length=" "

/>

Opisuje pojedyncze pole i odpowiadającą mu kolumnę w tabeli.

<many-to-one

name=" "

class=" "

not-null="true"

>

<column name=" " />

</many-to-one>

Opisu relacji, w tym przypadku wiele-do-jednego.

- Odwzorowanie poprzez adnotacje.

Encja (entity) jest to obiekt służący do reprezentacji trwałych danych. Reprezentuje tabelę z relacyjnej bazy danych. Definiowany jest w formie klasy encji i klas pomocniczych.

Dla klas encji wymagane jest:
- Plain Old Java Object (POJO) z adnotacją @Entity;
- bezargumentowy konstruktor (public lub protected);
- implementacja Serializable, jeśli obiekty będą odłączane;

W projekcie wykorzystano ten sposób mapowania. Poniżej przykład dla tabeli „client”.

```
@Entity
@Table(name = "client")
public class Client implements Serializable {

    @Id
    @GeneratedValue(strategy = GenerationType.AUTO)
    private String id;

    private String name;
    private String password;
    private String mail;
    private String LastName;
    private String address;

    public String getId() {
        return id;
    }

    public void setId(String id) {
        this.id = id;
    }

//gettery i settery
}
```

4.3 Związki między encjami.

- Liczność

  - 1:1 (@OneToOne);

  - 1:N (@OneToMany);

  - N:1 (@ManyToOne);

  - N:M (@ManyToMany);

- Kierunkowość

  - dwukierunkowe;

  - jednokierunkowe;

- Kaskada operacji

  - PERSIST;

  - MERGE;

  - REMOVE;

  - REFRESH;

Poniżej przedstawiono przykłady asocjacji @ManyToOne i @ManyToMany:

```
@Entity
@Table(name = "shopping_cart")
public class ShoppingCart {

    @Id @GeneratedValue(generator="system-uuid")
    @GenericGenerator(name="system-uuid", strategy = "org.hibernate.id.UUIDGenerator")
    private String id;
    private String state;

    @ManyToOne(fetch = FetchType.EAGER)
    @JoinColumn(name = "client_id")
    private Client client;

    @ManyToMany(cascade = {CascadeType.ALL})
    @JoinTable(
            name = "product_shopping",
            joinColumns = {@JoinColumn(name = "s_id")},
            inverseJoinColumns = {@JoinColumn(name = "p_id")})
    List<Product> products = new ArrayList<>();
```

4.4 Dziedziczenie encji.

Encje są zwykłymi klasami Javy. Tym samym jak najbardziej mogą rozszerzać inne klasy, a także być rozszerzane. Model obiektowy pozwala bardzo efektywnie przedstawić złożoność rozwiązywanych problemów biznesowych poprzez sensowne zastosowanie dziedziczenia. Z drugiej zaś strony mamy relacyjną bazę danych, która opiera się o tabele i relacje. Nie za wiele wie ona o dziedziczeniu. Istnieje kilka sposobów na rozwiązanie tego problemu.

- Single – Table Strategy

Zgodnie z nazwą dziedziczenie jest realizowane w oparciu o jedną tabelę w bazie danych. Automatycznie implikuje to istnienie w niej kolumn, które będą w stanie przechować stan wszystkich encji, które realizują dziedziczenie w oparciu o tę metodę.

- Joined Strategy

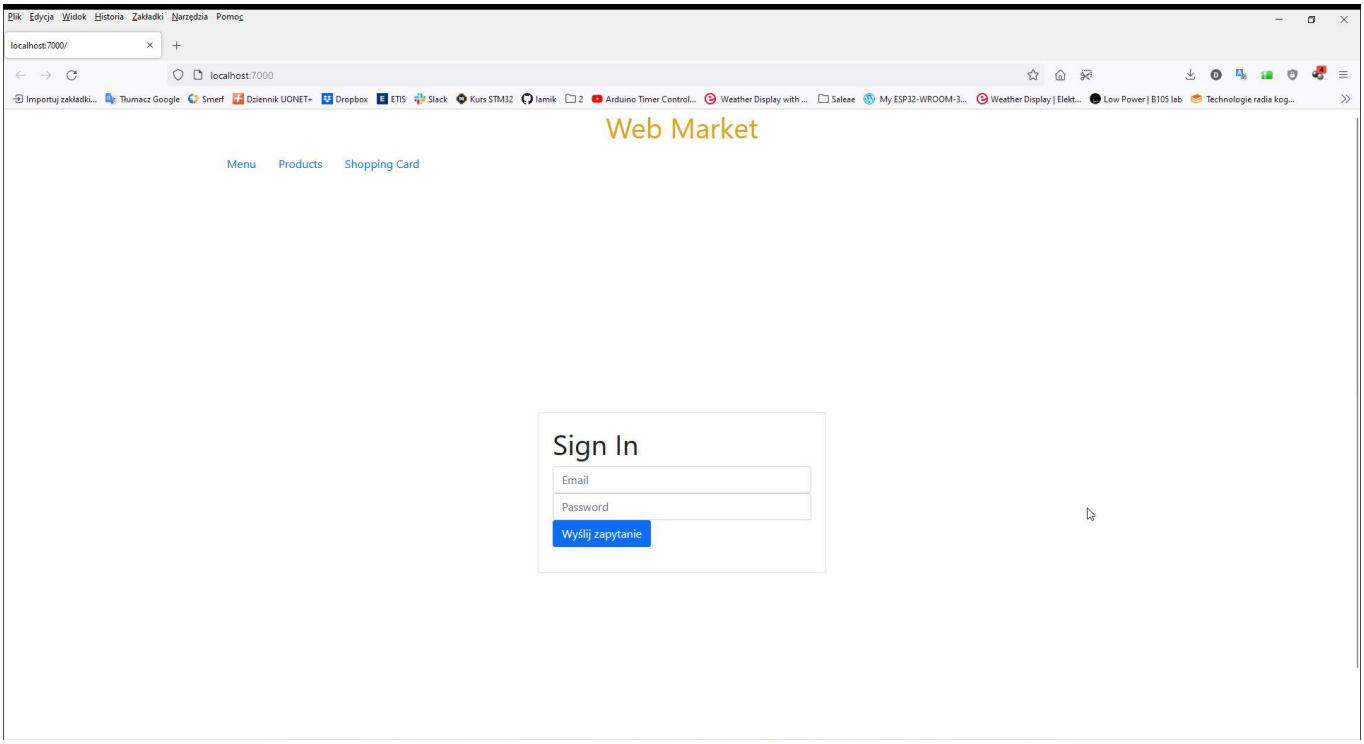
Zastosowanie strategii jednej tabeli może być na dłuższą metę dość zasobożerne. Nagle okazuje się, że w przypadku rozbudowanych hierarchii nasza tabela zawiera dużą ilość kolumn, które w znacznej mierze zawierają wartości NULL. Jeżeli nasza hierarchia dziedziczenia będzie dość rozbudowana można zastosować inną strategię – joined strategy. Wykorzysta ona siłę współczesnych baz danych – relacje.

- Table – per – Class Strategy

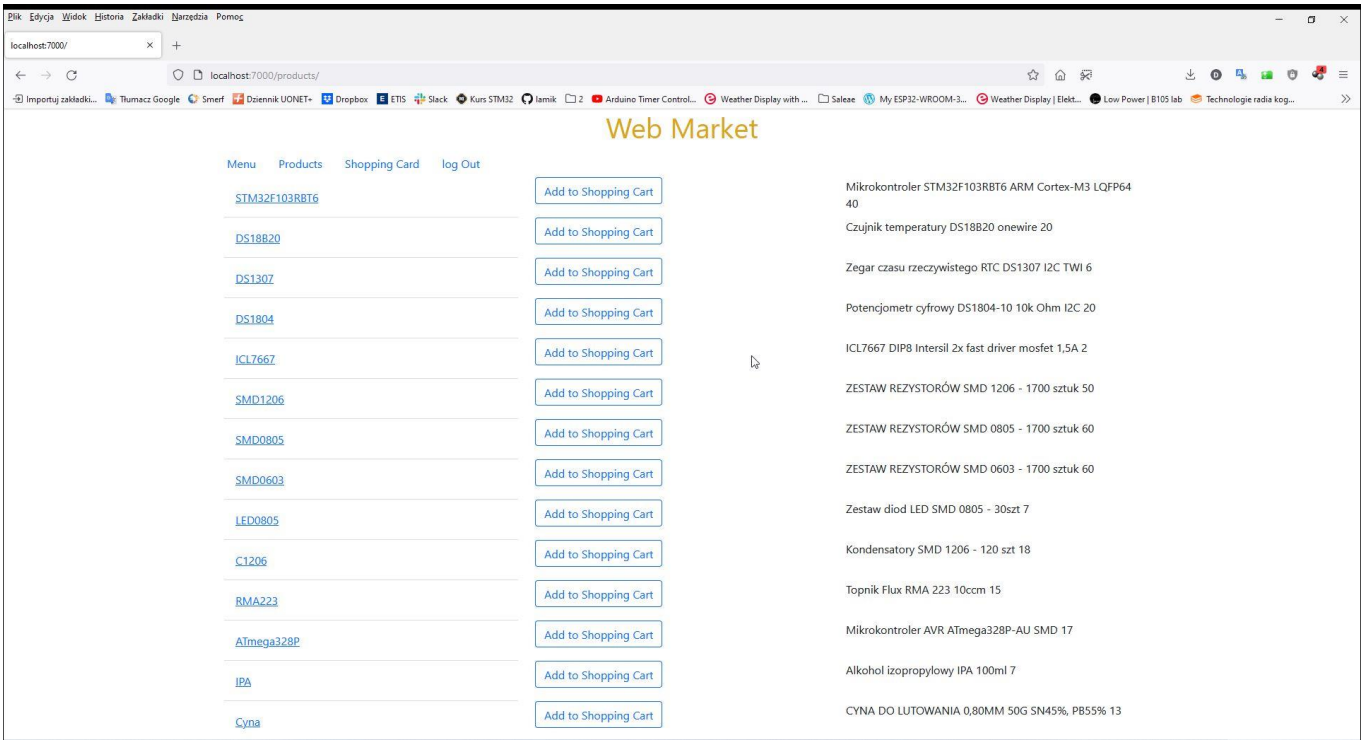
Jest to strategia, w której każda klasa w hierarchii dziedziczenia otrzymuje swoją własną tabelę. Adnotacja: @Inheritance na InheritanceType.TABLE_PER_CLASS. W bazie będą występowały informacje w sposób zdublowany. Jednakże rozwiązanie takie jest o wiele szybsze niż strategia relacyjna – zapytanie wybierające rekordy jest kierowane od razu do konkretnej tabeli, z pominięciem złączeń. Zatem o ile miejsce zajmowane przez dane nie jest istotne, albo też dane w bazie są prawie niezmienne, zaś często odpytywane warto zwrócić uwagę na tę strategię.

5. Aplikacja

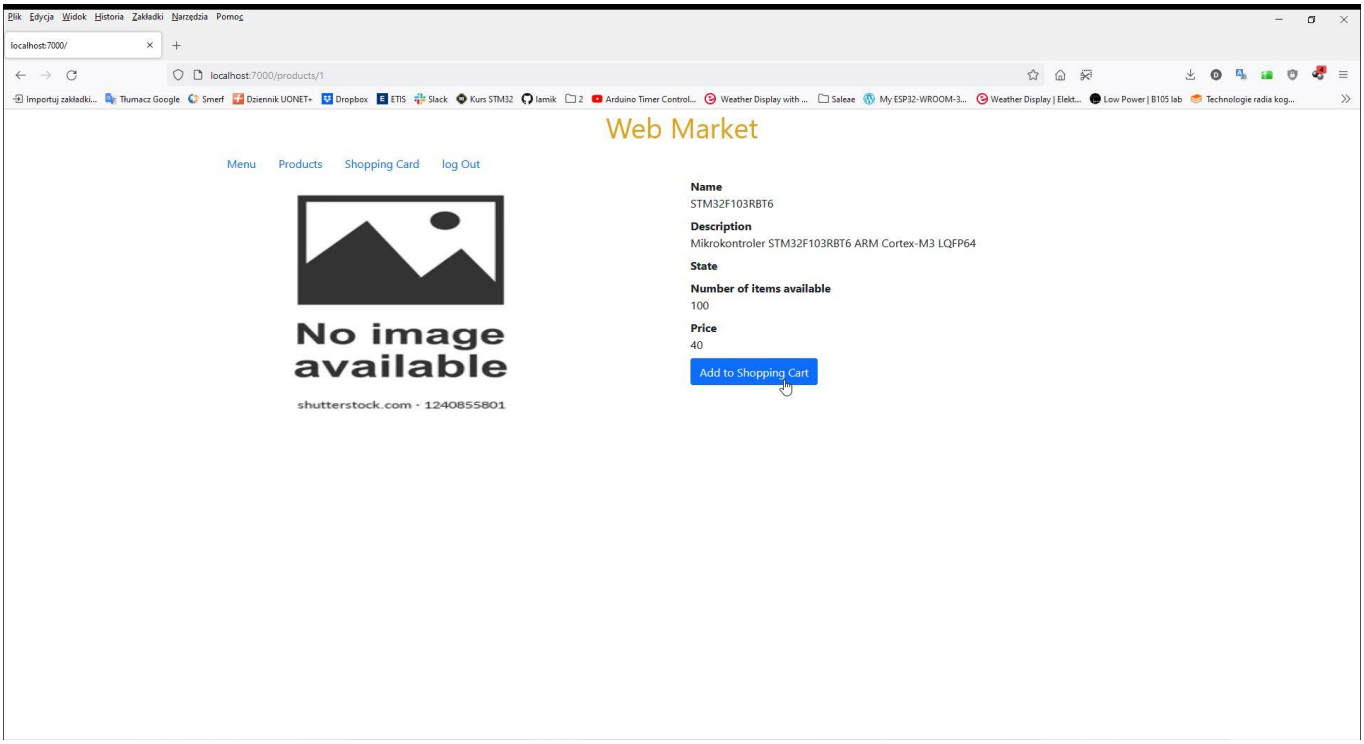
Rysunki 6 - 9 przedstawiają okno aplikacji.



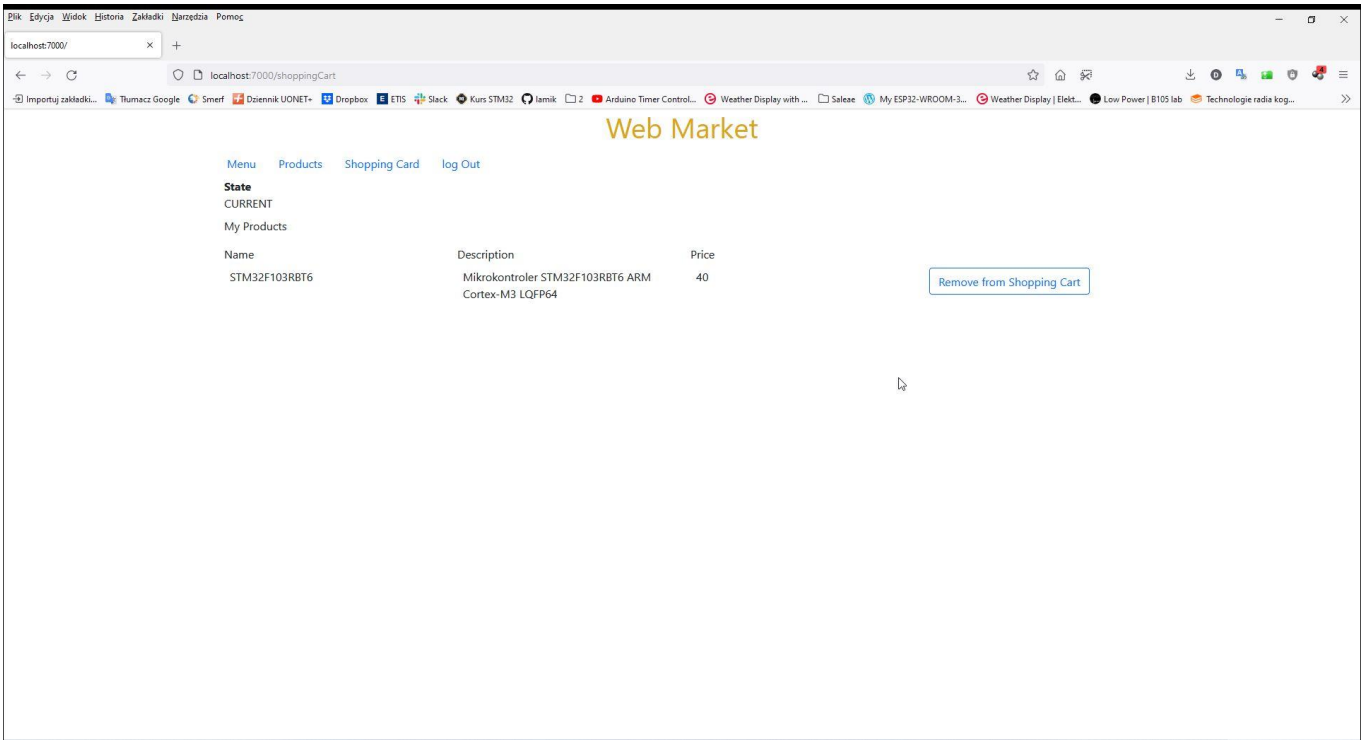
Rys. 6. Okno aplikacji przed zalogowaniem.



Rys. 7. Okno z listą produktów.



Rys. 8. Okno z opisem produktu.



Rys. 9. Okno z zawartością koszyka.

6. Wnioski

Jako zalety Hibernate należy wskazać wygodne mapowanie z wykorzystaniem adnotacji. Mniejsza liczba kodu przekłada się na mniej możliwych błędów. Encje może reprezentować dowolna klasa.

Z drugiej strony dla złożonych danych mapowanie mapowanie z obiektu do tabeli zmniejsza wydajnosć i zwiększa czas konwersji. Hibernate nei pozwala używać niektórych zapytań, które są dostepne w JDBC, np. nie pozwala wstawiać wiele obiekttów do tej samej tabei używając jednego zapytania.