Music Store

Simple music store website template for software engineering exam based on **WordPress** framework, developed on **Underscores** starter theme (_s), with **SASS** interpreter, using **PhpMyAdmin** as database to store information of website, customers and purchases through **PHP** and **MySQL**, designed with **ArgoUML**, tested with **PHPUnit** and licensed under **GNU General Public License v2.0**.

- **Version**: 1.0
- **Creator**: Andrea Bazerla
- **License**: GNU General Public License v2.0 or later

Project

The project consists to create UML diagrams as documentation of a software prototype following requirements specifications.

Files

- **/src/uml/**: **.uml** file exported with **ArgoUML**.
- **/doc/**
  - **BRIEFING.md**: briefing of project
  - **EXAM.md**: exam's instructions TO-DO list
  - **/png/**: all **UML diagrams** (use cases, sequence, activity, class, etc.).
- **/website/**
  - **/phpMyAdmin-4.6.3-all-languages/**: **database**.
  - **/wp-content/themes/music-store/**: **theme**.
    - **/test/: tests**.

UML

Use Cases

Search Case

Use Case Name: SearchCase

ID: SC

Actors: Customer

Pre-coditions:

1. Customer is in website.

Sequence:

1. The use case starts when custumer is in website.
2. If costumer chooses to show store catalog of website…
   - 2.1. System shows all item of store catalog to customer.
   - 2.2. If customer is on store catalog…
     - 2.2.1. Customer can click on instruments to show more details.
3. If customer search an instrument in store catalog...
   - 3.1. System allows customer to search an instrument by category, model or price and it shows relatives instruments as a catalog.

Alternative sequence:

1. At any time customer can exit from website.

Post-conditions:

1. Costumer has found instrument(s) that he was looking for.

Buy Case

Use Case Name: BuyCase

ID: BC

Actors: Customer

Pre-coditions:

1. Customer is in website.
2. Customer is logged in website.

Sequence:

1. The use case starts when custumer is logged in website.
2. If the costumer chooses to create a new order…
   - 2.1. Customer chooses an instrument to buy from store catalog.
   - 2.2. Customer add to cart the instrument.
   - 2.3. If customer confirms his order...
     - 2.3.1. System shows to customer his cart updated.
     - 2.3.2. Customer chooses shipping method.
     - 2.3.3. Customer makes payment.
       - 2.3.3.1. If customer chooses transfer as payment method…
         - 2.3.3.1.1. System send to customer a receipt.
       - 2.3.3.2. If customer chooses paypal as payment method…
         - 2.3.3.2.1. Customer inserts his credentials for payment.

2.3.3.3. 3. If customer chooses credit card as payment method…

2.3.3.3.1. Customer inserts his credentials for payment.

3. If customer edit his cart…
    3.1. Customer can delete or increment instruments order.
    3.2. System shows to customer his cart updated.

Alternative sequence:
1. At any time customer can exit from website.
2. Until the customer has not confirmed the order, he can cancel purchase.

Post-conditions:
1. Customer has bought at least one instrument.

Discount Case

Use Case Name: DiscountCase

ID: DC

Actors: (Professional musician / Music school owner) -> Professional customer -> Customer

Pre-coditions:
1. Customer is in website.

Sequence:
1. The use case starts when custumer is in website.
2. If costumer chooses to sign up in website to get more informations about his price…
    2.1. Customer inserts his credential in the system.
    2.2. System stores in database his credential to register him.
3. Customer chooses to log in the system.
4. Customer chooses to show store catalog of instruments in website.
5. If customer is registered and logged in system and if he has already bought at least 3 items more than €3000.00 each in 1 year…
    5.1. System shows to customer additional discounts and free delivery of shipping costs with a special order.
6. If customer chooses to do a special order…
    6.1. System check customer's requirements.
    6.2. If customer can get a special order…
        6.2.1. System allows customer to get a special order.

Alternative sequence:
1. At any time customer can exit from website.
2. At sign up procedure customer can cancel his registration

Post-conditions:
1. Customer has made a special order.

Sale Case

Use Case Name: SaleCase

ID: SC

Actors: Professional musician -> Customer

Pre-coditions:

1. Customer is in website.
2. Customer is logged in website as a professional musician.

Sequence:

1. The use case starts when custumer is in the website and logged in as a professional musician.
2. If customer chooses to insert a sale of an instrument on website…
    - 2.1. System asks to customer to insert informations about his instrument and, optionally, a photo of it.
3. If his instrument is sold…
    - 3.1. System send an e-mail to instrument's customer with details about sale, payment and shipment.

Alternative sequence:

1. At any time customer can exit from website.
2. At inclusion sale customer can cancel the sale.

Post-conditions:

1. Costumer has inserted an instrument or he has sold an instrument.

Sequence Diagrams

Search Sequence

Buy Sequence

Discount Sequence

Sale Sequence

Activity Diagram

Class Diagram

Sequence Diagram

Website

I have created a prototype of website that allows customer to signup, loggin and to see instrument catalog. It allows administrator of website to create new pages as Home, Profile, Login etc. easily and quickly. Customer can add instrument to cart clicking on theme in item page after they have see their details by store page.

Website is based on **WordPress**, the most popular free and open-source framework content management system (CMS) based on **PHP** and **MySQL**; it is created on **Underscores** (_s) starter theme with **SASS** to be easy to install, edit and publish.

Database

To store information about customers, instruments and sales we have created a database using **PhpMyAdmin** at localhost address **127.0.0.1/phpMyAdmin-4.6.3-all-languages** with 3 tables:

- **users**: to store information of customers for signup and login in website.

id	firstname	lastname	fc	city	phone	mobile	type	username	password

- **store**: to store information of instruments in website’s catalog.

id	timestamp	name	description	price	weight	photo	type	discount	level

- **cart**: to store IDs of users, instruments and their purchases.

id	username	instrument

Design Patterns

The core of prototype is created in **PHP** with some design patterns as

- **Factory Pattern:** it is implemented to create all pages of website as **Home**, **Profile**, **Login** etc. through the print of their titles.

Example:

php.php

```
interface Page {
	public function create($title);
}

class Home implements Page {
	public function create($title) {
		echo $title;
	}
}

class PageFactory {
	public function getPage($page) {
		$page = ucfirst(strtolower($page));
		(new $page)->create($page);
	}
}

class WebsiteFactory {
	public function createWebsite($page) {
		return (new PageFactory)->getPage($page);
	}
}

$website = new WebsiteFactory;
```

home.php

```
<?php $website->createWebsite("HOME"); ?>
```

- **Iterator Pattern:** it is implemented to iterate on all instruments of website’s store catalog in **Store** page, stored in table **store** of database and to display theme.

php.php

```php
class Store implements Page, IteratorAggregate {

      private $instruments = array(array());

      public function create($title) {
            echo $title;
      }

      public function getIterator() {
            return new Instrument($this);
      }

      public function createStore() {

            global $conn, $store;

            $sql = "SELECT * FROM store";
            $result = $conn->query($sql);
            $num = mysqli_num_rows($result);

            for ($id = 0; $id < $num; $id++) {
                  $store->addInstrument($id);
            }

            return $store;

      }

      public function addInstrument($id) {

            global $conn;

            $sql = "SELECT * FROM store WHERE id = $id";
            $result = mysqli_query($conn, $sql);
            if (mysqli_num_rows($result) == 1) {
                  $row = mysqli_fetch_array($result, MYSQLI_ASSOC);
                  $id = $row["id"];
                  $timestamp = $row["timestamp"];
                  $name = $row["name"];
                  $description = $row["description"];
                  $price = $row["price"];
                  $weight = $row["weight"];
                  $photo = $row["photo"];
                  $type = $row["type"];
```

```
                $discount = $row["discount"];
                $level = $row["level"];
            }

            $this->instruments[$id][0] = $id;
            $this->instruments[$id][1] = $timestamp;
            $this->instruments[$id][2] = $name;
            $this->instruments[$id][3] = $description;
            $this->instruments[$id][4] = $price;
            $this->instruments[$id][5] = $weight;
            $this->instruments[$id][6] = $photo;
            $this->instruments[$id][7] = $type;
            $this->instruments[$id][8] = $discount;
            $this->instruments[$id][9] = $level;

        }

        public function getInstrument($id) {
            if (isset($this->instruments[$id])) {
                $id = $this->instruments[$id][0];
                $timestamp = $this->instruments[$id][1];
                $name = $this->instruments[$id][2];
                $description = $this->instruments[$id][3];
                $price = $this->instruments[$id][4];
                $weight = $this->instruments[$id][5];
                $photo = $this->instruments[$id][6];
                $type = $this->instruments[$id][7];
                $discount = $this->instruments[$id][8];
                $level = $this->instruments[$id][9];
                return array($id, $timestamp, $name, $description,
$price, $weight, $photo, $type, $discount, $level);
            }
            return null;
        }

        public function isEmpty() {
            return empty($instruments);
        }

}

class Instrument implements Iterator {

        private $id = 0;
        private $store;
```

```
	public function __construct(Store $store) {
		$this->store = $store;
	}

	public function current() {
		return $this->store->getInstrument($this->id);
	}

	public function key() {
		return $this->id;
	}

	public function next() {
		$this->id++;
	}

	public function rewind() {
		$this->id = 0;
	}

	public function valid() {
		return !is_null($this->store->getInstrument($this->id));
	}

}
```

store.php

```
<?php

	$store = new Store;
	$store = $store->createStore();

	foreach ($store as $instrument) {
		echo "<form class='item'
action='http://127.0.0.1/index.php/item/' method='POST'>";
			echo "<input type='hidden' name='id'
value='$instrument[0]' />";
			echo "<p>Name: " . $instrument[2] . "</p>";
			echo "<p>Price: " . $instrument[4] . "</p>";
			echo "<p>Photo: " . $instrument[6] . "</p>";
			echo "<input class='submit' type='submit'
value='Details' />";
		echo "</form>";
	}

?>
```

- **Template Pattern:** it is implemented to create on **DefaultCustomer** class, a concrete class that implement **Customer** abstract class, different type of customer extending their classes.

php.php

```
<?php

      abstract class Customer {
            abstract function getProfile($username);
      }

      class DefaultCustomer extends Customer {

            public function getProfile($username) {

                  global $conn;

                  $sql = "
                        SELECT firstname, lastname, fc, city, phone,
mobile, type, username
                        FROM users
                        WHERE username = '$username';
                  ";

                  $result = mysqli_query($conn, $sql);

                  if (!$conn->query($sql)) {
                        echo "Error: " . $conn->error;
                  } else {
                        if (mysqli_num_rows($result) != 1) {
                              die("Error");
                        } else {
                              $row = mysqli_fetch_array($result,
MYSQLI_ASSOC);
                              $firstname = $row["firstname"];
                              $lastname = $row["lastname"];
                              $fc = $row["fc"];
                              $city = $row["city"];
                              $phone = $row["phone"];
                              $mobile = $row["mobile"];
                              $type = $row["type"];
                              return array($firstname, $lastname, $fc,
$city, $phone, $mobile, $type);
                        }
                  }
            }

      }

      class OccasionalCustomer extends DefaultCustomer {}
      class SchoolOwner extends DefaultCustomer {}
      class ProfessionalMusician extends DefaultCustomer {}

?>
```

Test

1. Customer in website can’t buy instruments if he isn’t logged in; he can only view theme in store page.
2. All instrument’s **ID** are hidden to prevent **DDoS** attacks: we have implemented **$_POST** method instead **$_GET** so customer could buy instruments only clicking on theme in item page.
3. If a customer went in **Profile**, **Cart** or **Sell** page and he isn't logged in website, it launches an error.
4. I’ve implemented in project **PHPUnit** (https://phpunit.de/)**,** a testing framework for PHP (PHP 5.6 or later), and I’ve created a new directory in **/wp-content/themes/music-store/test where** there are all test for website. For example, the first snippet test is for page creation checking their title and the second test is for store creation checking instruments’s IDs.
   To launch PHPUnit by terminal, type this: **phpunit --bootstrap inc/php.php test/<name of test>.php** from **/website/wp-content/themes/music-store/.**

pageTest.php

```
      use PHPUnit\Framework\TestCase;

      class pageTest extends TestCase {

            public function testPagesCreation() {

                  $pagesArray = array();
                  $pagesArray = ["HOME", "PROFILE", "SEARCH", "STORE",
      "SALE", "CART", "SIGNUP", "LOGIN"];

                  foreach ($pagesArray as $pageName) {

                        $newWebsite = new WebsiteFactory;
                        $pageTitle = print
      $newWebsite->createWebsite($pageName);
                        $this->assertEquals(print
      ucfirst(strtolower($pageName)), $pageTitle);
```

```
                }

            }

        }
```

storeTest.php

```
        use PHPUnit\Framework\TestCase;

        class storeTest extends TestCase {

            public function testStoreCreation() {

                global $conn, $store;

                $id = 0;

                $store = new Store;
                $store = $store->createStore();

                foreach ($store as $instrument) {
                    $id++;
                    $this->assertEquals($id, $instrument[0]);
                }

            }

        }
```

Changelog

- **1.0**
    - Created the project
    - Created **README.md**
    - Created **LICENSE.md**

Credits

Created by **Andrea Bazerla**.

License