Conceptual Modeling and Database

Practical work

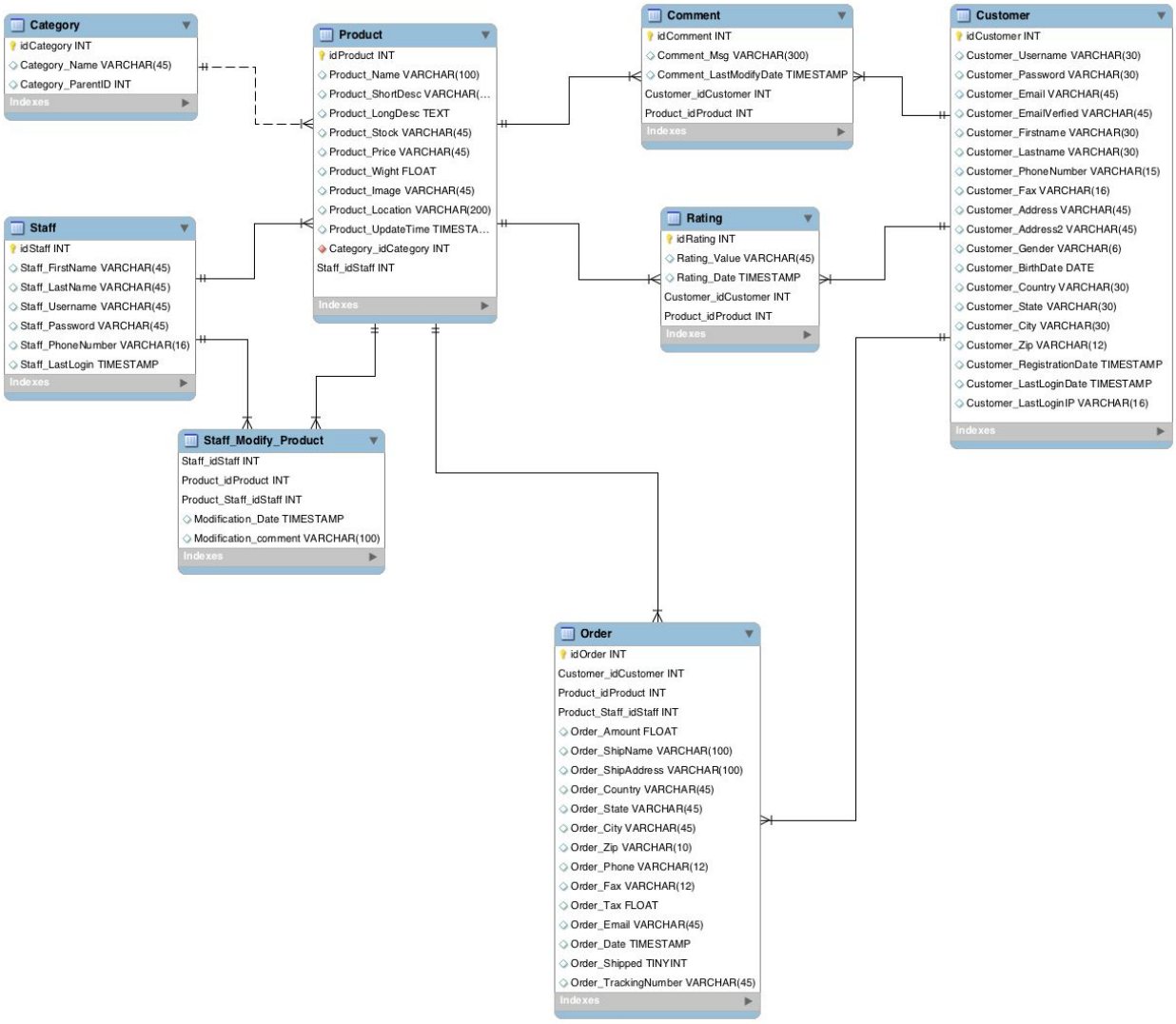
Online Shopping web-site

Introduction

This document contains all data related to practical work of Conceptual Modeling and Database course in TAMK. The subject is about design, implement and analyse an online shopping store. In first step we, ER-modelling by MySQL workbench will be described and tables and attributes will be checked. In second step, SQL script in order to create database will be show and then in next part the SQL queries in order to fetch data from database or modify them will be inspected. In final section details related to a RESTful-interface including PHP-programming and JSON will be examined.

Database design

The online shopping store include the process all steps from visiting web-site by a customer to register and pay for order and shipping to customer's address. It is possible to consider relation between product supplier and store as a part of database, but in this document this part was ignored. The shopping web-site database design would be as below which further in detail will be explained.



As it is shown through EER diagram, in an online store we have these entities:

Customer

Customer table includes these attributes:

- `idCustomer` INT NOT NULL,
- `Customer_Username` VARCHAR(30) NULL COMMENT ' ',
- `Customer_Password` VARCHAR(30) NULL,
- `Customer_Email` VARCHAR(45) NULL,

- `Customer_EmailVerfied` VARCHAR(45) NULL,
- `Customer_Firstname` VARCHAR(30) NULL,
- `Customer_Lastname` VARCHAR(30) NULL,
- `Customer_PhoneNumber` VARCHAR(15) NULL,
- `Customer_Fax` VARCHAR(16) NULL,
- `Customer_Address` VARCHAR(45) NULL,
- `Customer_Address2` VARCHAR(45) NULL,
- `Customer_Gender` VARCHAR(6) NULL,
- `Customer_BirthDate` DATE NULL,
- `Customer_Country` VARCHAR(30) NULL,
- `Customer_State` VARCHAR(30) NULL,
- `Customer_City` VARCHAR(30) NULL,
- `Customer_Zip` VARCHAR(12) NULL,
- `Customer_RegistrationDate` TIMESTAMP NULL,
- `Customer_LastLoginDate` TIMESTAMP NULL,
- `Customer_LastLoginIP` VARCHAR(16) NULL,

Product

Product table includes attributes below:

- `idProduct` INT NOT NULL,
- `Product_Name` VARCHAR(100) NULL,
- `Product_ShortDesc` VARCHAR(200) NULL,
- `Product_LongDesc` TEXT NULL,
- `Product_Stock` VARCHAR(45) NULL,
- `Product_Price` VARCHAR(45) NULL,
- `Product_Wight` FLOAT NULL,
- `Product_Image` VARCHAR(45) NULL,
- `Product_Location` VARCHAR(200) NULL,
- `Product_UpdateTime` TIMESTAMP NULL,

Category

Category table has these attributes:

- `idCategory` INT NOT NULL,
- `Category_Name` VARCHAR(45) NULL,
- `Category_parentID` INT NOT NULL DEFAULT 0,

Staff

Staff table has these attributes:

- `idStaff` INT NOT NULL,
- `Staff_FirstName` VARCHAR(45) NULL,
- `Staff_LastName` VARCHAR(45) NULL,
- `Staff_Username` VARCHAR(45) NULL,
- `Staff_Password` VARCHAR(45) NULL,
- `Staff_PhoneNumber` VARCHAR(16) NULL,
- `Staff_LastLogin` TIMESTAMP NULL,

Order

Order table has these attributes:

- `idOrder` INT NOT NULL,
- `Order_Amount` FLOAT NULL,
- `Order_ShipName` VARCHAR(100) NULL,
- `Order_ShipAddress` VARCHAR(100) NULL,
- `Order_Country` VARCHAR(45) NULL,
- `Order_State` VARCHAR(45) NULL,
- `Order_City` VARCHAR(45) NULL,

- `Order_Zip` VARCHAR(10) NULL,
- `Order_Phone` VARCHAR(12) NULL,
- `Order_Fax` VARCHAR(12) NULL,
- `Order_Tax` FLOAT NULL,
- `Order_Email` VARCHAR(45) NULL,
- `Order_Date` TIMESTAMP NULL,
- `Order_Shipped` TINYINT NULL,
- `Order_TrackingNumber` VARCHAR(45) NULL,

Comment

Comment includes these attributes:

- `idComment` INT NOT NULL,
- `Comment_Msg` VARCHAR(300) NULL,
- `Comment_LastModifyDate` TIMESTAMP NULL,

Rating

Rating table has these attributes:

- `idRating` INT NOT NULL,
- `Rating_Value` VARCHAR(45) NULL,
- `Rating_Date` TIMESTAMP NULL,

1.1. Modification logs

Create tables

In order create tables from designed EER diagram, by forward engineering SQL script could be generated, which the result would be same as below:

```
-- MySQL Script generated by MySQL Workbench

-- Tue Aug 30 21:42:43 2016

-- Model: New Model    Version: 1.0

-- MySQL Workbench Forward Engineering


SET @OLD_UNIQUE_CHECKS=@@UNIQUE_CHECKS, UNIQUE_CHECKS=0;

SET @OLD_FOREIGN_KEY_CHECKS=@@FOREIGN_KEY_CHECKS, FOREIGN_KEY_CHECKS=0;

SET @OLD_SQL_MODE=@@SQL_MODE, SQL_MODE='TRADITIONAL,ALLOW_INVALID_DATES';


-- -----------------------------------------------------

-- Schema mydb

-- -----------------------------------------------------


-- -----------------------------------------------------

-- Schema mydb

-- -----------------------------------------------------

CREATE SCHEMA IF NOT EXISTS `mydb` DEFAULT CHARACTER SET utf8 COLLATE utf8_general_ci ;

USE `mydb` ;


-- -----------------------------------------------------

-- Table `mydb`.`Category`

-- -----------------------------------------------------
```

```
CREATE TABLE IF NOT EXISTS `mydb`.`Category` (
  `idCategory` INT NOT NULL,
  `Category_Name` VARCHAR(45) NULL,
 `Category_parentID` INT NOT NULL DEFAULT 0,
 PRIMARY KEY (`idCategory`))
ENGINE = InnoDB;


-- -----------------------------------------------------
-- Table `mydb`.`Staff`
-- -----------------------------------------------------
CREATE TABLE IF NOT EXISTS `mydb`.`Staff` (
  `idStaff` INT NOT NULL,
  `Staff_FirstName` VARCHAR(45) NULL,
  `Staff_LastName` VARCHAR(45) NULL,
  `Staff_Username` VARCHAR(45) NULL,
  `Staff_Password` VARCHAR(45) NULL,
  `Staff_PhoneNumber` VARCHAR(16) NULL,
  `Staff_LastLogin` TIMESTAMP NULL,
 PRIMARY KEY (`idStaff`))
ENGINE = InnoDB;


-- -----------------------------------------------------
-- Table `mydb`.`Product`
-- -----------------------------------------------------
```

```
CREATE TABLE IF NOT EXISTS `mydb`.`Product` (
  `idProduct` INT NOT NULL,
  `Product_Name` VARCHAR(100) NULL,
  `Product_ShortDesc` VARCHAR(200) NULL,
  `Product_LongDesc` TEXT NULL,
  `Product_Stock` VARCHAR(45) NULL,
  `Product_Price` VARCHAR(45) NULL,
  `Product_Wight` FLOAT NULL,
  `Product_Image` VARCHAR(45) NULL,
  `Product_Location` VARCHAR(200) NULL,
  `Product_UpdateTime` TIMESTAMP NULL,
  `Category_idCategory` INT NOT NULL,
  `Staff_idStaff` INT NOT NULL,
 PRIMARY KEY (`idProduct`, `Staff_idStaff`),
 INDEX `fk_Product_Category1_idx` (`Category_idCategory` ASC),
 INDEX `fk_Product_Staff1_idx` (`Staff_idStaff` ASC),
 CONSTRAINT `fk_Product_Category1`
   FOREIGN KEY (`Category_idCategory`)
   REFERENCES `mydb`.`Category` (`idCategory`)
   ON DELETE NO ACTION
   ON UPDATE NO ACTION,
 CONSTRAINT `fk_Product_Staff1`
   FOREIGN KEY (`Staff_idStaff`)
   REFERENCES `mydb`.`Staff` (`idStaff`)
   ON DELETE NO ACTION
   ON UPDATE NO ACTION)
```

```
ENGINE = InnoDB;



-- -----------------------------------------------------

-- Table `mydb`.`Customer`

-- -----------------------------------------------------

CREATE TABLE IF NOT EXISTS `mydb`.`Customer` (

  `idCustomer` INT NOT NULL,

  `Customer_Username` VARCHAR(30) NULL COMMENT '      ',

  `Customer_Password` VARCHAR(30) NULL,

  `Customer_Email` VARCHAR(45) NULL,

  `Customer_EmailVerfied` VARCHAR(45) NULL,

  `Customer_Firstname` VARCHAR(30) NULL,

  `Customer_Lastname` VARCHAR(30) NULL,

  `Customer_PhoneNumber` VARCHAR(15) NULL,

  `Customer_Fax` VARCHAR(16) NULL,

  `Customer_Address` VARCHAR(45) NULL,

  `Customer_Address2` VARCHAR(45) NULL,

  `Customer_Gender` VARCHAR(6) NULL,

  `Customer_BirthDate` DATE NULL,

  `Customer_Country` VARCHAR(30) NULL,

  `Customer_State` VARCHAR(30) NULL COMMENT '                                                  ',

  `Customer_City` VARCHAR(30) NULL,

  `Customer_Zip` VARCHAR(12) NULL,

  `Customer_RegistrationDate` TIMESTAMP NULL,

  `Customer_LastLoginDate` TIMESTAMP NULL,
```

```
  `Customer_LastLoginIP` VARCHAR(16) NULL,
  PRIMARY KEY (`idCustomer`))
ENGINE = InnoDB;


-- -----------------------------------------------------
-- Table `mydb`.`Comment`
-- -----------------------------------------------------
CREATE TABLE IF NOT EXISTS `mydb`.`Comment` (
  `idComment` INT NOT NULL,
  `Comment_Msg` VARCHAR(300) NULL,
  `Comment_LastModifyDate` TIMESTAMP NULL,
  `Customer_idCustomer` INT NOT NULL,
  `Product_idProduct` INT NOT NULL,
  PRIMARY KEY (`idComment`, `Customer_idCustomer`, `Product_idProduct`),
  INDEX `fk_Comment_Customer_idx` (`Customer_idCustomer` ASC),
  INDEX `fk_Comment_Product1_idx` (`Product_idProduct` ASC),
  CONSTRAINT `fk_Comment_Customer`
    FOREIGN KEY (`Customer_idCustomer`)
    REFERENCES `mydb`.`Customer` (`idCustomer`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION,
  CONSTRAINT `fk_Comment_Product1`
    FOREIGN KEY (`Product_idProduct`)
    REFERENCES `mydb`.`Product` (`idProduct`)
    ON DELETE NO ACTION
```

```
  ON UPDATE NO ACTION)

ENGINE = InnoDB;



-- -----------------------------------------------------

-- Table `mydb`.`Rating`

-- -----------------------------------------------------

CREATE TABLE IF NOT EXISTS `mydb`.`Rating` (

  `idRating` INT NOT NULL,

  `Rating_Value` VARCHAR(45) NULL,

  `Rating_Date` TIMESTAMP NULL,

  `Customer_idCustomer` INT NOT NULL,

  `Product_idProduct` INT NOT NULL,

  PRIMARY KEY (`idRating`, `Customer_idCustomer`, `Product_idProduct`),

  INDEX `fk_Rating_Customer1_idx` (`Customer_idCustomer` ASC),

  INDEX `fk_Rating_Product1_idx` (`Product_idProduct` ASC),

  CONSTRAINT `fk_Rating_Customer1`

    FOREIGN KEY (`Customer_idCustomer`)

    REFERENCES `mydb`.`Customer` (`idCustomer`)

    ON DELETE NO ACTION

    ON UPDATE NO ACTION,

  CONSTRAINT `fk_Rating_Product1`

    FOREIGN KEY (`Product_idProduct`)

    REFERENCES `mydb`.`Product` (`idProduct`)

    ON DELETE NO ACTION

    ON UPDATE NO ACTION)
```

```
ENGINE = InnoDB;



-- -----------------------------------------------------

-- Table `mydb`.`Order`

-- -----------------------------------------------------

CREATE TABLE IF NOT EXISTS `mydb`.`Order` (

  `idOrder` INT NOT NULL,

  `Customer_idCustomer` INT NOT NULL,

  `Product_idProduct` INT NOT NULL,

  `Product_Staff_idStaff` INT NOT NULL,

  `Order_Amount` FLOAT NULL,

  `Order_ShipName` VARCHAR(100) NULL,

  `Order_ShipAddress` VARCHAR(100) NULL,

  `Order_Country` VARCHAR(45) NULL,

  `Order_State` VARCHAR(45) NULL,

  `Order_City` VARCHAR(45) NULL,

  `Order_Zip` VARCHAR(10) NULL,

  `Order_Phone` VARCHAR(12) NULL,

  `Order_Fax` VARCHAR(12) NULL,

  `Order_Tax` FLOAT NULL,

  `Order_Email` VARCHAR(45) NULL,

  `Order_Date` TIMESTAMP NULL,

  `Order_Shipped` TINYINT NULL,

  `Order_TrackingNumber` VARCHAR(45) NULL,

 PRIMARY KEY (`idOrder`, `Customer_idCustomer`, `Product_idProduct`, `Product_Staff_idStaff`),
```

```
  INDEX `fk_Order_Customer1_idx` (`Customer_idCustomer` ASC),
  INDEX `fk_Order_Product1_idx` (`Product_idProduct` ASC, `Product_Staff_idStaff` ASC),
  CONSTRAINT `fk_Order_Customer1`
    FOREIGN KEY (`Customer_idCustomer`)
    REFERENCES `mydb`.`Customer` (`idCustomer`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION,
  CONSTRAINT `fk_Order_Product1`
    FOREIGN KEY (`Product_idProduct`, `Product_Staff_idStaff`)
    REFERENCES `mydb`.`Product` (`idProduct`, `Staff_idStaff`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION)
ENGINE = InnoDB;


-- -----------------------------------------------------
-- Table `mydb`.`Staff_Modify_Product`
-- -----------------------------------------------------
CREATE TABLE IF NOT EXISTS `mydb`.`Staff_Modify_Product` (
  `Staff_idStaff` INT NOT NULL,
  `Product_idProduct` INT NOT NULL,
  `Product_Staff_idStaff` INT NOT NULL,
  `Modification_Date` TIMESTAMP NULL,
  `Modification_comment` VARCHAR(100) NULL,
  PRIMARY KEY (`Staff_idStaff`, `Product_idProduct`, `Product_Staff_idStaff`),
  INDEX `fk_Staff_has_Product_Product1_idx` (`Product_idProduct` ASC, `Product_Staff_idStaff` ASC),
```

```
  INDEX `fk_Staff_has_Product_Staff1_idx` (`Staff_idStaff` ASC),
  CONSTRAINT `fk_Staff_has_Product_Staff1`
    FOREIGN KEY (`Staff_idStaff`)
    REFERENCES `mydb`.`Staff` (`idStaff`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION,
  CONSTRAINT `fk_Staff_has_Product_Product1`
    FOREIGN KEY (`Product_idProduct`, `Product_Staff_idStaff`)
    REFERENCES `mydb`.`Product` (`idProduct`, `Staff_idStaff`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION)
ENGINE = InnoDB;


SET SQL_MODE=@OLD_SQL_MODE;
SET FOREIGN_KEY_CHECKS=@OLD_FOREIGN_KEY_CHECKS;
SET UNIQUE_CHECKS=@OLD_UNIQUE_CHECKS;
```

SQL scripts

In order to have CRUD functionality in database, the SQL queries related to each table and reason is provided as below:

Category Table Scripts

```
insert into Category(Category_Name) Values ("House Holding")
insert into Category(Category_Name) Values ("Gardening")
```

insert into Category(Category_Name, Category_ParentID) Values ("Kitchen",1)

insert into Category(Category_Name, Category_ParentID) Values ("Dining Room",1)

insert into Category(Category_Name, Category_ParentID) Values
("DishWasher",(select idCategory
from Category
where Category_Name="kitchen"));

insert into Category(Category_Name, Category_ParentID)

select "Dish Washer",idCategory
from Category
where Category_Name="kitchen"

*Select * from Category*

Staff Table

INSERT INTO `dbc6mahmad62`.`Staff`

(`Staff_FirstName`, `Staff_LastName`, `Staff_Username`, `Staff_Password`, `Staff_PhoneNumber`)

VALUES

("Mojtaba", "Ahmadi", "Aaaa", "123", "+358466181212");

*select * from Staff*

Product Table

INSERT INTO `dbc6mahmad62`.`Product`

(`Product_Name`, `Product_ShortDesc`,`Product_LongDesc`, `Product_Stock`, `Product_Price`, `Product_Wight`, `Product_Location`, `Category_idCategory`, `Staff_idStaff`)

VALUES

("SPT SD-2224DS Countertop Dishwasher with Delay Start & LED",

"7 wash cycles: Heavy, normal, light, mini party, rinse, speed and soak

Delay start for added convenience : Two, four, six or eight hours

Universal faucet adapter and Quick Connect: For quick and easy connection to most kitchen faucets", "7 wash cycles: Heavy, normal, light, mini party, rinse, speed and soak

Delay start for added convenience : Two, four, six or eight hours

Universal faucet adapter and Quick Connect: For quick and easy connection to most kitchen faucets

Electronic controls with LED display : LED displays remaining time or current running state

Water supply warning indicator and rinse aid warning indicator", "In Stock", "225.99", 15.45, "TAMPERE", 5, 1);

*select * from Product*

Customer Table

INSERT INTO `dbc6mahmad62`.`Customer`

(`Customer_Username`, `Customer_Password`, `Customer_Email`, `Customer_EmailVerfied`, `Customer_Firstname`, `Customer_Lastname`, `Customer_PhoneNumber`, `Customer_Fax`, `Customer_Address`, `Customer_Address2`, `Customer_Gender`, `Customer_BirthDate`, `Customer_Country`, `Customer_State`, `Customer_City`,`Customer_Zip`)

VALUES

("silver", "123", "test@test.com", "test2@test.com", "Mojtaba", "Ahmadi", "+358466181212", "0213123", "Yrttikatu 15B", "", "M", 1986-06-30, "Finland", "Pirkanmaa", "Tampere", "33710");

*select * from Customer*

order Table

INSERT INTO `dbc6mahmad62`.`Orders`

(`Customer_idCustomer`, `Product_idProduct`, `Product_Staff_idStaff`, `Order_Amount`, `Order_ShipName`, `Order_ShipAddress`, `Order_Country`, `Order_State`, `Order_City`, `Order_Zip`, `Order_Phone`, `Order_Fax`, `Order_Tax`, `Order_Email`, `Order_Date`, `Order_TrackingNumber`)

VALUES

(1000, 1, 1, 1, "Tomi", "Satakunnankatu 12", "Finland", "Pirkanmma", "Tampere", "33210", "+354343231213", "0212333434", 12.0333, "book@gmail.com", 2016-08-21, "2123434545G4343");

*select * from Orders*

Comment Table

INSERT INTO `dbc6mahmad62`.`Comment`

(`Comment_Msg`, `Comment_LastModifyDate`, `Customer_idCustomer`, `Product_idProduct`)

VALUES

("The warranty is awful!", 2016-08-20, 1000, 1);

*select * from Comment*

Rating table

INSERT INTO `dbc6mahmad62`.`Rating`

(`Rating_Value`, `Rating_Date`, `Customer_idCustomer`, `Product_idProduct`)

VALUES ("5", 2016-08-21, 1000, 1);

*select * from Rating*

Restful interface

Restful interface includes both PHP codes and JSON outputs. In order to cover CRUD functions in web programming , slim framework which is kind of micro framework could be used. For this purpose we 3 files “db.php” , “functions.php”, “index.php” should be modified which the result would be same as this:

Db.php

```
<?php

function getDB() {

  $dbhost="mydb.tamk.fi";

  $dbuser="c6mahmad"; // Your own username

  $dbpass="Silver123"; // Your own password

  $dbname="dbc6mahmad62"; // Your own database name

  $dbConnection = new PDO("mysql:host=$dbhost;
```

```
dbname=$dbname;charset=utf8",
$dbuser, $dbpass,array(PDO::MYSQL_ATTR_INIT_COMMAND
=> "SET NAMES 'utf8'"));
   return $dbConnection;
}
```

Index.php

```
<?php
header("Access-Control-Allow-Origin: *");
use \Psr\Http\Message\ServerRequestInterface as Request;
use \Psr\Http\Message\ResponseInterface as Response;

require 'vendor/autoload.php';
require 'db.php';
require 'functions.php';

$app = new \Slim\App;



//get all products
$app->get('/Product',function (Request $request, Response $response) {
 $json = getProducts();
 $response->getBody()->write($json);
 return $response;
});

// get Product by id
$app->get('/Product/{id}',function (Request $request, Response $response) {
```

```
    $id = $request->getAttribute('id');
$json = getProductById($id);
    $response->getBody()->write($json);
    return $response;
});
//create Product
$app->post('/Product',function (Request $request, Response $response) {
$body = $request->getBody();
$params = json_decode($body);
var_dump($params);
$json = createProduct($params);
$response->getBody()->write($json);
return $response;
});

//update Product
$app->put('/Product/{id}',function (Request $request, Response $response) {
$id = $request->getAttribute('id');
$body = $request->getBody();
$params = json_decode($body);
var_dump($params);
$json = updateProduct($id,$params);
$response->getBody()->write($json);
return $response;
});

// delete Product by id
$app->delete('/Product/{id}',function (Request $request, Response $response) {
```

```
    $id = $request->getAttribute('id');
$json = deleteProduct($id);
    $response->getBody()->write($json);
    return $response;
});



//get all Customer
$app->get('/Customer',function (Request $request, Response $response) {
 $json = getCustomer();
 $response->getBody()->write($json);
 return $response;
});


// get Customer by id
$app->get('/Customer/{id}',function (Request $request, Response $response) {
    $id = $request->getAttribute('id');
 $json = getCustomerById($id);
    $response->getBody()->write($json);
    return $response;
});
//create Customer
$app->post('/Customer',function (Request $request, Response $response) {
 $body = $request->getBody();
 $params = json_decode($body);
 var_dump($params);
 $json = createCustomer($params);
 $response->getBody()->write($json);
```

```
return $response;

});


// delete Customer by id

$app->delete('/Customer/{id}',function (Request $request, Response $response) {

   $id = $request->getAttribute('id');

$json = deleteCustomer($id);

   $response->getBody()->write($json);

   return $response;

});


//update Customer

$app->put('/Customer/{id}',function (Request $request, Response $response) {

$id = $request->getAttribute('id');

$body = $request->getBody();

$params = json_decode($body);

var_dump($params);

$json = updateCustomer($id,$params);

$response->getBody()->write($json);

return $response;

});



//get all Staff

$app->get('/Staff',function (Request $request, Response $response) {

$json = getStaff();

$response->getBody()->write($json);

return $response;
```

```
});


// get Staff by id
$app->get('/Staff/{id}',function (Request $request, Response $response) {
    $id = $request->getAttribute('id');
$json = getStaffById($id);
    $response->getBody()->write($json);
    return $response;
});
//create Staff
$app->post('/Staff',function (Request $request, Response $response) {
$body = $request->getBody();
$params = json_decode($body);
var_dump($params);
$json = createStaff($params);
$response->getBody()->write($json);
return $response;
});


// delete Staff by id
$app->delete('/Staff/{id}',function (Request $request, Response $response) {
    $id = $request->getAttribute('id');
$json = deleteStaff($id);
    $response->getBody()->write($json);
    return $response;
});


//update Staff
```

```
$app->put('/Staff/{id}',function (Request $request, Response $response) {
$id = $request->getAttribute('id');
$body = $request->getBody();
$params = json_decode($body);
var_dump($params);
$json = updateStaff($id,$params);
$response->getBody()->write($json);
return $response;
});



//get all Category
$app->get('/Category',function (Request $request, Response $response) {
$json = getCategory();
$response->getBody()->write($json);
return $response;
});


// get Category by id
$app->get('/Category/{id}',function (Request $request, Response $response) {
    $id = $request->getAttribute('id');
$json = getCategoryById($id);
    $response->getBody()->write($json);
    return $response;
});
//create Category
$app->post('/Category',function (Request $request, Response $response) {
```

```
$body = $request->getBody();
$params = json_decode($body);
var_dump($params);
$json = createCategory($params);
$response->getBody()->write($json);
return $response;
});

// delete Category by id
$app->delete('/Category/{id}',function (Request $request, Response $response) {
    $id = $request->getAttribute('id');
$json = deleteCategory($id);
    $response->getBody()->write($json);
    return $response;
});

//update Category
$app->put('/Category/{id}',function (Request $request, Response $response) {
$id = $request->getAttribute('id');
$body = $request->getBody();
$params = json_decode($body);
var_dump($params);
$json = updateCategory($id,$params);
$response->getBody()->write($json);
return $response;
});
```

```
//-----Orders


//get all Orders
$app->get('/Orders',function (Request $request, Response $response) {
 $json = getOrders();
 $response->getBody()->write($json);
 return $response;
});


// get Orders by id
$app->get('/Orders/{id}',function (Request $request, Response $response) {
    $id = $request->getAttribute('id');
 $json = getOrdersById($id);
    $response->getBody()->write($json);
    return $response;
});
//create Orders
$app->post('/Orders',function (Request $request, Response $response) {
 $body = $request->getBody();
 $params = json_decode($body);
 var_dump($params);
 $json = createOrders($params);
 $response->getBody()->write($json);
 return $response;
});


// delete Orders by id
$app->delete('/Orders/{id}',function (Request $request, Response $response) {
```

```
    $id = $request->getAttribute('id');
$json = deleteOrders($id);
    $response->getBody()->write($json);
    return $response;
});

//update Orders
$app->put('/Orders/{id}',function (Request $request, Response $response) {
 $id = $request->getAttribute('id');
 $body = $request->getBody();
 $params = json_decode($body);
 var_dump($params);
 $json = updateOrders($id,$params);
 $response->getBody()->write($json);
 return $response;
});


//____Comment

//get all Comment
$app->get('/Comment',function (Request $request, Response $response) {
 $json = getComment();
 $response->getBody()->write($json);
 return $response;
});

// get Comment by id
```

```
$app->get('/Comment/{id}',function (Request $request, Response $response) {
    $id = $request->getAttribute('id');
$json = getCommentById($id);
    $response->getBody()->write($json);
    return $response;
});
//create Comment
$app->post('/Comment',function (Request $request, Response $response) {
$body = $request->getBody();
$params = json_decode($body);
var_dump($params);
$json = createComment($params);
$response->getBody()->write($json);
return $response;
});

// delete Comment by id
$app->delete('/Comment/{id}',function (Request $request, Response $response) {
    $id = $request->getAttribute('id');
$json = deleteComment($id);
    $response->getBody()->write($json);
    return $response;
});

//update Comment
$app->put('/Comment/{id}',function (Request $request, Response $response) {
$id = $request->getAttribute('id');
$body = $request->getBody();
```

```
$params = json_decode($body);

var_dump($params);

$json = updateComment($id,$params);

$response->getBody()->write($json);

return $response;

});


//____Rating



//get all Rating

$app->get('/Rating',function (Request $request, Response $response) {

$json = getRating();

$response->getBody()->write($json);

return $response;

});


// get Rating by id

$app->get('/Rating/{id}',function (Request $request, Response $response) {

   $id = $request->getAttribute('id');

$json = getRatingById($id);

   $response->getBody()->write($json);

   return $response;

});

//create Rating

$app->post('/Rating',function (Request $request, Response $response) {

$body = $request->getBody();

$params = json_decode($body);
```

```
var_dump($params);
$json = createRating($params);
$response->getBody()->write($json);
return $response;
});

// delete Rating by id
$app->delete('/Rating/{id}',function (Request $request, Response $response) {
   $id = $request->getAttribute('id');
$json = deleteRating($id);
   $response->getBody()->write($json);
   return $response;
});

//update Rating
$app->put('/Rating/{id}',function (Request $request, Response $response) {
$id = $request->getAttribute('id');
$body = $request->getBody();
$params = json_decode($body);
var_dump($params);
$json = updateRating($id,$params);
$response->getBody()->write($json);
return $response;
});

$app->run();
```

Function.php

```
<?php


// get all Products

function getProducts() {

    $sql="Select * from Product";

    try {

        $db = getDB();

        $stmt = $db->query($sql);

        $object = $stmt->fetchAll(PDO::FETCH_OBJ);

        $db = null;

        return '{"data": ' . json_encode($object, JSON_UNESCAPED_UNICODE) . '}';

        } catch(PDOException $e) {

            return '{"error":{"text":'. $e->getMessage() .'}}';

    }

}


// get product by id

function getProductById($id) {

    $sql="Select * from Product

    WHERE idProduct=:id";

    try {

        $db = getDB();

        $stmt = $db->prepare($sql);
```

```
        $stmt->bindParam(':id', $id);
        $stmt->execute();
        $object = $stmt->fetchAll(PDO::FETCH_OBJ);
        $db = null;
        return '{"data": ' . json_encode($object, JSON_UNESCAPED_UNICODE) . '}';
        } catch(PDOException $e) {
                        return '{"error":{"text":'. $e->getMessage() .'}}';
    }
}

//create Products
function createProduct ($params) {
$sql="INSERT INTO Product(`Product_Name`, `Product_ShortDesc`,`Product_LongDesc`, `Product_Stock`, `Product_Price`,
`Product_Wight`, `Product_Location`, `Category_idCategory`, `Staff_idStaff`) VALUES (:Product_Name,
:Product_ShortDesc,:Product_LongDesc, :Product_Stock, :Product_Price, :Product_Wight, :Product_Location, :Category_idCategory,
:Staff_idStaff)";
try {
        $db = getDB();
        $stmt = $db->prepare($sql);
        $stmt->bindParam(':Product_Name', $params->Product_Name);
        $stmt->bindParam(':Product_ShortDesc', $params->Product_ShortDesc);
        $stmt->bindParam(':Product_LongDesc', $params->Product_LongDesc);
        $stmt->bindParam(':Product_Stock', $params->Product_Stock);
        $stmt->bindParam(':Product_Price', $params->Product_Price);
        $stmt->bindParam(':Product_Wight', $params->Product_Wight);
        $stmt->bindParam(':Product_Location', $params->Product_Location);
        $stmt->bindParam(':Category_idCategory', $params->Category_idCategory);
```

```
	$stmt->bindParam(':Staff_idStaff', $params->Staff_idStaff);

		$result = $stmt->execute ();

		echo $stmt->debugDumpParams().'\n'.var_export($stmt->errorInfo());

		$db = null; //closes pdo-connection

		if ($result)

			return '("info": "ok")';

		else

			return '("info": "nok")';

} catch(PDOException $e) {


return '{"error":{"text":'. $e->getMessage() .'}}';


}

}




//update Products

function updateProduct($id,$params) {

$sql="Update Product

		SET Product_Name=:Product_Name, Product_ShortDesc=:Product_ShortDesc, Product_LongDesc=:Product_LongDesc,
Product_Stock=:Product_Stock, Product_Price=:Product_Price, Product_Wight=:Product_Wight,
Product_Location=:Product_Location

	Where idProduct=:id";

try {

		$db = getDB();

	$stmt = $db->prepare($sql);
```

```
        $stmt->bindParam(':id', $id);
        $stmt->bindParam(':Product_Name', $params->Product_Name);
                $stmt->bindParam(':Product_ShortDesc', $params->Product_ShortDesc);
                $stmt->bindParam(':Product_LongDesc', $params->Product_LongDesc);
        $stmt->bindParam(':Product_Stock', $params->Product_Stock);
        $stmt->bindParam(':Product_Price', $params->Product_Price);
        $stmt->bindParam(':Product_Wight', $params->Product_Wight);
        $stmt->bindParam(':Product_Location', $params->Product_Location);
                $result = $stmt->execute ();
                echo $stmt->debugDumpParams().'\n'.var_export($stmt->errorInfo());
                $db = null; //closes pdo-connection
                if ($result)
                        return '("info": "ok")';
                else
                        return '("info": "nok")';
} catch(PDOException $e) {

return '{"error":{"text":'. $e->getMessage() .'}}';

}
}

//delete Product
function deleteProduct($id) {
$id = (int)$id;
if (empty($id)) {
```

```
            exitWithError('invalid or missing id');
}
dbDelete('Product', $id,idProduct);
}

//----Customer

// get all Customer
function getCustomer() {
   $sql="SELECT * From Customer";
   try {
      $db = getDB();
      $stmt = $db->query($sql);
      $object = $stmt->fetchAll(PDO::FETCH_OBJ);
      $db = null;
      return '{"data": ' . json_encode($object, JSON_UNESCAPED_UNICODE) . '}';
      } catch(PDOException $e) {
         return '{"error":{"text":'. $e->getMessage() .'}}';
   }
}

// get Customer by id
function getCustomerById($id) {
   $sql="SELECT * From Customer
         Where idCustomer=:id";
   try {
```

```
	$db = getDB();

	$stmt = $db->prepare($sql);

	$stmt->bindParam(':id', $id);

	$stmt->execute();

	$object = $stmt->fetchAll(PDO::FETCH_OBJ);

	$db = null;

	return '{"data": ' . json_encode($object, JSON_UNESCAPED_UNICODE) . '}';

	} catch(PDOException $e) {

			return '{"error":{"text":'. $e->getMessage() .'}}';

  }

}


//create Customer

function createCustomer ($params) {

$sql="INSERT INTO Customer(`Customer_Username`, `Customer_Password`, `Customer_Email`, `Customer_EmailVerfied`,
`Customer_Firstname`, `Customer_Lastname`, `Customer_PhoneNumber`, `Customer_Fax`, `Customer_Address`,
`Customer_Address2`, `Customer_Gender`, `Customer_Country`, `Customer_State`, `Customer_City`,`Customer_Zip`) VALUES
(:Customer_Username, :Customer_Password, :Customer_Email, :Customer_EmailVerfied, :Customer_Firstname,
:Customer_Lastname, :Customer_PhoneNumber, :Customer_Fax, :Customer_Address, :Customer_Address2, :Customer_Gender,
:Customer_Country, :Customer_State, :Customer_City, :Customer_Zip)";

try {

		$db = getDB();

	$stmt = $db->prepare($sql);

	$stmt->bindParam(':Customer_Username', $params->Customer_Username);

		$stmt->bindParam(':Customer_Password', $params->Customer_Password);

		$stmt->bindParam(':Customer_Email', $params->Customer_Email);

		$stmt->bindParam(':Customer_EmailVerfied', $params->Customer_EmailVerfied);

		$stmt->bindParam(':Customer_Firstname', $params->Customer_Firstname);
```

```
        $stmt->bindParam(':Customer_Lastname', $params->Customer_Lastname);
                $stmt->bindParam(':Customer_PhoneNumber', $params->Customer_PhoneNumber);
                $stmt->bindParam(':Customer_Fax', $params->Customer_Fax);
                $stmt->bindParam(':Customer_Address', $params->Customer_Address);
                $stmt->bindParam(':Customer_Address2', $Customer_Address2);
                $stmt->bindParam(':Customer_Gender', $params->Customer_Gender);
        $stmt->bindParam(':Customer_Country', $params->Customer_Country);
        $stmt->bindParam(':Customer_State', $params->Customer_State);
        $stmt->bindParam(':Customer_City', $params->Customer_City);
        $stmt->bindParam(':Customer_Zip', $params->Customer_Zip);
                $result = $stmt->execute ();
                echo $stmt->debugDumpParams().'\n'.var_export($stmt->errorInfo());
                $db = null; //closes pdo-connection
                if ($result)
                        return '("info": "ok")';
                else
                        return '("info": "nok")';
} catch(PDOException $e) {

return '{"error":{"text":'. $e->getMessage() .'}}';

}
}

//delete Customer
function deleteCustomer($id) {
```

```
$id = (int)$id;

if (empty($id)) {

        exitWithError('invalid or missing id');

}

dbDelete('Customer', $id,idCustomer);

}


//update Customer

function updateCustomer($id,$params) {

$sql="Update Customer

        SET Customer_Username=:Customer_Username, Customer_Password=:Customer_Password,
Customer_Email=:Customer_Email, Customer_EmailVerfied=:Customer_EmailVerfied, Customer_Firstname=:Customer_Firstname,
Customer_Lastname=:Customer_Lastname, Customer_PhoneNumber=:Customer_PhoneNumber, Customer_Fax=:Customer_Fax,
Customer_Address=:Customer_Address, Customer_Address2=:Customer_Address2, Customer_Gender=:Customer_Gender,
Customer_Country=:Customer_Country, Customer_State=:Customer_State, Customer_City=:Customer_City,
Customer_Zip=:Customer_Zip

    Where idCustomer=:id";

try {

        $db = getDB();

    $stmt = $db->prepare($sql);

    $stmt->bindParam(':id', $id);

    $stmt->bindParam(':Customer_Username', $params->Customer_Username);

        $stmt->bindParam(':Customer_Password', $params->Customer_Password);

        $stmt->bindParam(':Customer_Email', $params->Customer_Email);

        $stmt->bindParam(':Customer_EmailVerfied', $params->Customer_EmailVerfied);

        $stmt->bindParam(':Customer_Firstname', $params->Customer_Firstname);

    $stmt->bindParam(':Customer_Lastname', $params->Customer_Lastname);

        $stmt->bindParam(':Customer_PhoneNumber', $params->Customer_PhoneNumber);
```

```
        $stmt->bindParam(':Customer_Fax', $params->Customer_Fax);
        $stmt->bindParam(':Customer_Address', $params->Customer_Address);
        $stmt->bindParam(':Customer_Address2', $Customer_Address2);
        $stmt->bindParam(':Customer_Gender', $params->Customer_Gender);
    $stmt->bindParam(':Customer_Country', $params->Customer_Country);
    $stmt->bindParam(':Customer_State', $params->Customer_State);
    $stmt->bindParam(':Customer_City', $params->Customer_City);
    $stmt->bindParam(':Customer_Zip', $params->Customer_Zip);
        $result = $stmt->execute ();
        echo $stmt->debugDumpParams().'\n'.var_export($stmt->errorInfo());
        $db = null; //closes pdo-connection
        if ($result)
                return '("info": "ok")';
        else
                return '("info": "nok")';
} catch(PDOException $e) {

return '{"error":{"text":'. $e->getMessage() .'}}';

}
}



//---Staff

// get all Staff
```

```
function getStaff() {
    $sql="SELECT * from Staff";
    try {
        $db = getDB();
        $stmt = $db->query($sql);
        $object = $stmt->fetchAll(PDO::FETCH_OBJ);
        $db = null;
        return '{"data": ' . json_encode($object, JSON_UNESCAPED_UNICODE) . '}';
        } catch(PDOException $e) {
            return '{"error":{"text":'. $e->getMessage() .'}}';
    }
}

// get Staff by id
function getStaffById($id) {
    $sql="SELECT * from Staff
    where idStaff=:id";
    try {
        $db = getDB();
        $stmt = $db->prepare($sql);
        $stmt->bindParam(':id', $id);
        $stmt->execute();
        $object = $stmt->fetchAll(PDO::FETCH_OBJ);
        $db = null;
        return '{"data": ' . json_encode($object, JSON_UNESCAPED_UNICODE) . '}';
        } catch(PDOException $e) {
```

```
                return '{"error":{"text":'. $e->getMessage() .'}}';
    }
}

//create Staff
function createStaff ($params) {
$sql="INSERT INTO Staff (`Staff_FirstName`, `Staff_LastName`, `Staff_Username`, `Staff_Password`, `Staff_PhoneNumber`)
VALUES (:Staff_FirstName, :Staff_LastName, :Staff_Username, :Staff_Password ,:Staff_PhoneNumber)";
try {
        $db = getDB();
    $stmt = $db->prepare($sql);
    $stmt->bindParam(':Staff_FirstName', $params->Staff_FirstName);
        $stmt->bindParam(':Staff_LastName', $params->Staff_LastName);
        $stmt->bindParam(':Staff_Username', $params->Staff_Username);
    $stmt->bindParam(':Staff_Password', $params->Staff_Password);
    $stmt->bindParam(':Staff_PhoneNumber', $params->Staff_PhoneNumber);
        $result = $stmt->execute ();
        echo $stmt->debugDumpParams().'\n'.var_export($stmt->errorInfo());
        $db = null; //closes pdo-connection
        if ($result)
                return '("info": "ok")';
        else
                return '("info": "nok")';
} catch(PDOException $e) {

return '{"error":{"text":'. $e->getMessage() .'}}';
```

```
}

}


//delete Staff

function deleteStaff($id) {

$id = (int)$id;

if (empty($id)) {

        exitWithError('invalid or missing id');

}

dbDelete('Staff', $id,idStaff);

}


//update Staff

function updateStaff($id,$params) {

        $sql="Update Staff

         SET Staff_FirstName=:Staff_FirstName, Staff_LastName=:Staff_LastName, Staff_Username=:Staff_Username,
Staff_Password=:Staff_Password, Staff_PhoneNumber= :Staff_PhoneNumber

    Where idStaff=:id";

try {

        $db = getDB();

    $stmt = $db->prepare($sql);

    $stmt->bindParam(':id', $id);

    $stmt->bindParam(':Staff_FirstName', $params->Staff_FirstName);

        $stmt->bindParam(':Staff_LastName', $params->Staff_LastName);

        $stmt->bindParam(':Staff_Username', $params->Staff_Username);
```

```
        $stmt->bindParam(':Staff_Password', $params->Staff_Password);
        $stmt->bindParam(':Staff_PhoneNumber', $params->Staff_PhoneNumber);
                $result = $stmt->execute ();
                echo $stmt->debugDumpParams().'\n'.var_export($stmt->errorInfo());
                $db = null; //closes pdo-connection
                if ($result)
                        return '("info": "ok")';
                else
                        return '("info": "nok")';
} catch(PDOException $e) {

return '{"error":{"text":'. $e->getMessage() .'}}';
}
}

//---Category

// get all Category
function getCategory() {
    $sql="SELECT * from Category";
    try {
        $db = getDB();
        $stmt = $db->query($sql);
        $object = $stmt->fetchAll(PDO::FETCH_OBJ);
        $db = null;
        return '{"data": ' . json_encode($object, JSON_UNESCAPED_UNICODE) . '}';
```

```
    } catch(PDOException $e) {
        return '{"error":{"text":'. $e->getMessage() .'}}';
    }
}

// get Category by id
function getCategoryById($id) {
    $sql="SELECT * from Category
    Where idCategory=:id";
    try {
        $db = getDB();
        $stmt = $db->prepare($sql);
        $stmt->bindParam(':id', $id);
        $stmt->execute();
        $object = $stmt->fetchAll(PDO::FETCH_OBJ);
        $db = null;
        return '{"data": ' . json_encode($object, JSON_UNESCAPED_UNICODE) . '}';
        } catch(PDOException $e) {
                    return '{"error":{"text":'. $e->getMessage() .'}}';
    }
}

//create Category
function createCategory ($params) {
$sql="INSERT INTO Category (Category_Name, Category_ParentID)
VALUES (:Category_Name, :Category_ParentID)";
```

```
try {
		$db = getDB();
	$stmt = $db->prepare($sql);
	$stmt->bindParam(':Category_Name', $params->Category_Name);
		$stmt->bindParam(':Category_ParentID', $params->Category_ParentID);
		echo $stmt->debugDumpParams().'\n'.var_export($stmt->errorInfo());
		$result = $stmt->execute ();
		$db = null; //closes pdo-connection
		if ($result)
			return '("info": "ok")';
		else
			return '("info": "nok")';
} catch(PDOException $e) {

return '{"error":{"text":'. $e->getMessage() .'}}';

}
}

//delete Category
function deleteCategory($id) {
$id = (int)$id;
if (empty($id)) {
		exitWithError('invalid or missing id');
}
dbDelete('Category', $id,idCategory);
```

```
}

//update Category
function updateCategory($id,$params) {
$sql="Update Category
        SET Category_Name=:Category_Name, Category_ParentID=:Category_ParentID
    Where idCategory=:id";
try {
        $db = getDB();
    $stmt = $db->prepare($sql);
    $stmt->bindParam(':id', $id);
     $stmt->bindParam(':Category_Name', $params->Category_Name);
        $stmt->bindParam(':Category_ParentID', $params->Category_ParentID);
        $result = $stmt->execute ();
        echo $stmt->debugDumpParams().'\n'.var_export($stmt->errorInfo());
        $db = null; //closes pdo-connection
        if ($result)
                return '("info": "ok")';
        else
                return '("info": "nok")';
} catch(PDOException $e) {

return '{"error":{"text":'. $e->getMessage() .'}}';
}
}
```

```
//----Orders



// get all Orders
function getOrders() {
    $sql="SELECT * from Orders";
    try {
        $db = getDB();
        $stmt = $db->query($sql);
        $object = $stmt->fetchAll(PDO::FETCH_OBJ);
        $db = null;
        return '{"data": ' . json_encode($object, JSON_UNESCAPED_UNICODE) . '}';
        } catch(PDOException $e) {
            return '{"error":{"text":'. $e->getMessage() .'}}';
    }
}



// get Orders by id
function getOrdersById($id) {
    $sql="SELECT * from Orders
    Where idOrder=:id";
    try {
        $db = getDB();
```

```
	$stmt = $db->prepare($sql);

	$stmt->bindParam(':id', $id);

	$stmt->execute();

	$object = $stmt->fetchAll(PDO::FETCH_OBJ);

	$db = null;

	return '{"data": ' . json_encode($object, JSON_UNESCAPED_UNICODE) . '}';

	} catch(PDOException $e) {

			return '{"error":{"text":'. $e->getMessage() .'}}';

   }

}



//create Orders

function createOrders ($params) {

$sql="INSERT INTO Orders (Customer_idCustomer, Product_idProduct, Product_Staff_idStaff , Order_Amount, Order_ShipName,
Order_ShipAddress, Order_Country, Order_State, Order_City, Order_Zip ,Order_Phone, Order_Fax, Order_Tax ,Order_Email ,
Order_TrackingNumber)

VALUES (:Customer_idCustomer, :Product_idProduct, :Product_Staff_idStaff , :Order_Amount, :Order_ShipName,
:Order_ShipAddress, :Order_Country, :Order_State, :Order_City, :Order_Zip, :Order_Phone, :Order_Fax, :Order_Tax , :Order_Email ,
:Order_TrackingNumber)";

try {

		$db = getDB();

	$stmt = $db->prepare($sql);

	$stmt->bindParam(':Customer_idCustomer', $params->Customer_idCustomer);

		$stmt->bindParam(':Product_idProduct', $params->Product_idProduct);

	$stmt->bindParam(':Product_Staff_idStaff', $params->Product_Staff_idStaff);

	$stmt->bindParam(':Order_Amount', $params->Order_Amount);
```

```
        $stmt->bindParam(':Order_ShipName', $params->Order_ShipName);
        $stmt->bindParam(':Order_ShipAddress', $params->Order_ShipAddress);
        $stmt->bindParam(':Order_Country', $params->Order_Country);
        $stmt->bindParam(':Order_State', $params->Order_State);
        $stmt->bindParam(':Order_City', $params->Order_City);
        $stmt->bindParam(':Order_Zip', $params->Order_Zip);
        $stmt->bindParam(':Order_Phone', $params->Order_Phone);
        $stmt->bindParam(':Order_Fax', $params->Order_Fax);
        $stmt->bindParam(':Order_Tax', $params->Order_Tax);
        $stmt->bindParam(':Order_Email', $params->Order_Email);
        $stmt->bindParam(':Order_TrackingNumber', $params->Order_TrackingNumber);
                echo $stmt->debugDumpParams().'\n'.var_export($stmt->errorInfo());
                $result = $stmt->execute ();
                $db = null; //closes pdo-connection
                if ($result)
                        return '("info": "ok")';
                else
                        return '("info": "nok")';
} catch(PDOException $e) {

return '{"error":{"text":'. $e->getMessage() .'}}';

}
}

//delete Orders
```

```
function deleteOrders($id) {

$id = (int)$id;

if (empty($id)) {

        exitWithError('invalid or missing id');

}

dbDelete('Orders', $id,idOrder);

}


//update Orders

function updateOrders($id,$params) {

$sql="Update Orders

        SET Customer_idCustomer=:Customer_idCustomer , Product_idProduct=:Product_idProduct ,
Product_Staff_idStaff=:Product_Staff_idStaff , Order_Amount=:Order_Amount , Order_ShipName=:Order_ShipName ,
Order_ShipAddress=:Order_ShipAddress , Order_Country=:Order_Country , Order_State=:Order_State , Order_City=:Order_City ,
Order_Zip=:Order_Zip ,Order_Phone=:Order_Phone , Order_Fax=:Order_Fax , Order_Tax=:Order_Tax ,Order_Email=:Order_Email ,
Order_TrackingNumber=:Order_TrackingNumber

    Where idOrder=:id";

try {

        $db = getDB();

    $stmt = $db->prepare($sql);

    $stmt->bindParam(':id', $id);

    $stmt->bindParam(':Customer_idCustomer', $params->Customer_idCustomer);

        $stmt->bindParam(':Product_idProduct', $params->Product_idProduct);

    $stmt->bindParam(':Product_Staff_idStaff', $params->Product_Staff_idStaff);

    $stmt->bindParam(':Order_Amount', $params->Order_Amount);

    $stmt->bindParam(':Order_ShipName', $params->Order_ShipName);

    $stmt->bindParam(':Order_ShipAddress', $params->Order_ShipAddress);
```

```
        $stmt->bindParam(':Order_Country', $params->Order_Country);
        $stmt->bindParam(':Order_State', $params->Order_State);
        $stmt->bindParam(':Order_City', $params->Order_City);
        $stmt->bindParam(':Order_Zip', $params->Order_Zip);
        $stmt->bindParam(':Order_Phone', $params->Order_Phone);
        $stmt->bindParam(':Order_Fax', $params->Order_Fax);
        $stmt->bindParam(':Order_Tax', $params->Order_Tax);
        $stmt->bindParam(':Order_Email', $params->Order_Email);
        $stmt->bindParam(':Order_TrackingNumber', $params->Order_TrackingNumber);
                $result = $stmt->execute ();
                echo $stmt->debugDumpParams().'\n'.var_export($stmt->errorInfo());
                $db = null; //closes pdo-connection
                if ($result)
                        return '("info": "ok")';
                else
                        return '("info": "nok")';
} catch(PDOException $e) {

return '{"error":{"text":'. $e->getMessage() .'}}';
}
}

//__Comment

// get all Comment
```

```
function getComment() {
    $sql="SELECT * from Comment";
    try {
        $db = getDB();
        $stmt = $db->query($sql);
        $object = $stmt->fetchAll(PDO::FETCH_OBJ);
        $db = null;
        return '{"data": ' . json_encode($object, JSON_UNESCAPED_UNICODE) . '}';
        } catch(PDOException $e) {
            return '{"error":{"text":'. $e->getMessage() .'}}';
    }
}



// get Comment by id
function getCommentById($id) {
    $sql="SELECT * from Comment
    Where idComment=:id";
    try {
        $db = getDB();
        $stmt = $db->prepare($sql);
        $stmt->bindParam(':id', $id);
        $stmt->execute();
        $object = $stmt->fetchAll(PDO::FETCH_OBJ);
        $db = null;
        return '{"data": ' . json_encode($object, JSON_UNESCAPED_UNICODE) . '}';
```

```
    } catch(PDOException $e) {
                return '{"error":{"text":'. $e->getMessage() .'}}';
  }
}

//create Comment
function createComment ($params) {
$sql="INSERT INTO Comment (Comment_Msg , Customer_idCustomer, Product_idProduct)
  VALUES (:Comment_Msg, :Customer_idCustomer, :Product_idProduct)";
try {
        $db = getDB();
    $stmt = $db->prepare($sql);
    $stmt->bindParam(':Comment_Msg', $params->Comment_Msg);
        $stmt->bindParam(':Customer_idCustomer', $params->Customer_idCustomer);
    $stmt->bindParam(':Product_idProduct', $params->Product_idProduct);
        echo $stmt->debugDumpParams().'\n'.var_export($stmt->errorInfo());
        $result = $stmt->execute ();
        $db = null; //closes pdo-connection
        if ($result)
                return '("info": "ok")';
        else
                return '("info": "nok")';
} catch(PDOException $e) {

return '{"error":{"text":'. $e->getMessage() .'}}';
```

```
}

}


//delete Comment

function deleteComment($id) {

$id = (int)$id;

if (empty($id)) {

        exitWithError('invalid or missing id');

}

dbDelete('Comment', $id,idComment);

}


//update Comment

function updateComment($id,$params) {

$sql="Update Comment

         SET Comment_Msg= :Comment_Msg , Customer_idCustomer= :Customer_idCustomer, Product_idProduct=
:Product_idProduct

    Where idComment=:id";

try {

        $db = getDB();

    $stmt = $db->prepare($sql);

    $stmt->bindParam(':id', $id);

    $stmt->bindParam(':Comment_Msg', $params->Comment_Msg);

        $stmt->bindParam(':Customer_idCustomer', $params->Customer_idCustomer);

    $stmt->bindParam(':Product_idProduct', $params->Product_idProduct);

        $result = $stmt->execute ();
```

```
        echo $stmt->debugDumpParams().'\n'.var_export($stmt->errorInfo());
        $db = null; //closes pdo-connection
        if ($result)
                return '("info": "ok")';
        else
                return '("info": "nok")';
} catch(PDOException $e) {

return '{"error":{"text":'. $e->getMessage() .'}}';
}
}

//---Rating


// get all Rating
function getRating() {
   $sql="SELECT * from Rating";
   try {
      $db = getDB();
      $stmt = $db->query($sql);
      $object = $stmt->fetchAll(PDO::FETCH_OBJ);
      $db = null;
      return '{"data": ' . json_encode($object, JSON_UNESCAPED_UNICODE) . '}';
      } catch(PDOException $e) {
         return '{"error":{"text":'. $e->getMessage() .'}}';
```

```
    }
}



// get Rating by id
function getRatingById($id) {
    $sql="SELECT * from Rating
    Where idRating=:id";
    try {
        $db = getDB();
        $stmt = $db->prepare($sql);
        $stmt->bindParam(':id', $id);
        $stmt->execute();
        $object = $stmt->fetchAll(PDO::FETCH_OBJ);
        $db = null;
        return '{"data": ' . json_encode($object, JSON_UNESCAPED_UNICODE) . '}';
        } catch(PDOException $e) {
                        return '{"error":{"text":'. $e->getMessage() .'}}';
    }
}


//create Rating
function createRating ($params) {
$sql="INSERT INTO Rating (Rating_Value, Customer_idCustomer, Product_idProduct)
VALUES (:Rating_Value, :Customer_idCustomer, :Product_idProduct)";
try {
```

```
        $db = getDB();
    $stmt = $db->prepare($sql);
    $stmt->bindParam(':Rating_Value', $params->Rating_Value);
        $stmt->bindParam(':Customer_idCustomer', $params->Customer_idCustomer);
    $stmt->bindParam(':Product_idProduct', $params->Product_idProduct);
        echo $stmt->debugDumpParams().'\n'.var_export($stmt->errorInfo());
        $result = $stmt->execute ();
        $db = null; //closes pdo-connection
        if ($result)
                return '("info": "ok")';
        else
                return '("info": "nok")';
} catch(PDOException $e) {

return '{"error":{"text":'. $e->getMessage() .'}}';

}
}

//delete Rating
function deleteRating($id) {
$id = (int)$id;
if (empty($id)) {
        exitWithError('invalid or missing id');
}
dbDelete('Rating', $id,idRating);
```

```
}

//update Rating

function updateRating($id,$params) {

$sql="Update Rating

        SET Rating_Value= :Rating_Value, Customer_idCustomer= :Customer_idCustomer,
Product_idProduct=:Product_idProduct

    Where idRating=:id";

try {

        $db = getDB();

    $stmt = $db->prepare($sql);

    $stmt->bindParam(':id', $id);

    $stmt->bindParam(':Rating_Value', $params->Rating_Value);

        $stmt->bindParam(':Customer_idCustomer', $params->Customer_idCustomer);

    $stmt->bindParam(':Product_idProduct', $params->Product_idProduct);

        $result = $stmt->execute ();

        echo $stmt->debugDumpParams().'\n'.var_export(    $stmt->errorInfo());

        $db = null; //closes pdo-connection

        if ($result)

                return '("info": "ok")';

        else

                return '("info": "nok")';

} catch(PDOException $e) {


return '{"error":{"text":'. $e->getMessage() .'}}';

}
```

```
}



//////////////////////////

function getRequestDataAsObject() {

$request = Slim::getInstance()->request();

$json = $request->getBody();

$object = json_decode($json);

return $object;

}

function exitWithError($text) {

$error = array('error' => array('text' => $text));

$json = json_encode($error);

die($json);

}

function validateDepartmentData($data) {

$error = '';

if (empty($data->id)) {

        $error = 'missing required data (you must provide a ID)';

}

return $error;

}
```

```
function validateEmployeeData($data) {

$error = '';

if (empty($data->id)) {

        $error = 'missing required data (you must provide a ID)';

}

return $error;

}


function validateProjectData($data) {

$error = '';

if (empty($data->id)) {

        $error = 'missing required data (you must provide a ID)';

}

return $error;

}


function validateWorkData($data) {

$error = '';

if (empty($data->id)) {

        $error = 'missing required data (you must provide a ID)';

}

return $error;

}
```

```
//////////////////////////

function dbQuery($sql, $params = array()) {

try {

        $db = getDB();

        $stmt = $db->prepare($sql);

        foreach ($params as $key => $val) {

                $stmt->bindValue($key, $val);

        }

        $stmt->execute();

        $records = $stmt->fetchAll(PDO::FETCH_OBJ);

        $db = null;

        return $records;

} catch(PDOException $e) {

        exitWithError($e->getMessage());

}

}

function dbInsertFromObject($table, $object) {

if (empty($table) || empty($object)) {

        return null;

}

$fields = get_object_vars($object);

$field_names = array_keys($fields);

$sql = "INSERT INTO {$table} (" . implode(', ', $field_names) . ") VALUES (:" . implode(', :', $field_names) . ")";

try {

        $db = getDB();
```

```
        $stmt = $db->prepare($sql);
        foreach ($fields as $key => $val) {
                $stmt->bindValue($key, $val);
        }
        $stmt->execute();
        $id = $db->lastInsertId();
        $db = null;
        return $id;
} catch(PDOException $e) {
        exitWithError($e->getMessage());
}
}
function dbUpdateFromObject($table, $object, $id, $id_field_name = 'id') {
if (empty($table) || empty($object) || empty($id)) {
        return;
}
$fields = get_object_vars($object);
$field_names = array_keys($fields);
$field_pairs = array();
foreach ($field_names as $field_name) {
        $field_pairs[] = "{$field_name}=:{$field_name}";
}
$sql = "UPDATE {$table} SET " . implode(', ', $field_pairs) . " WHERE {$id_field_name}=:{$id_field_name}";
try {
        $db = getDB();
        $stmt = $db->prepare($sql);
```

```
        foreach ($fields as $key => $val) {
                $stmt->bindValue($key, $val);
        }
        $stmt->bindValue($id_field_name, $id);
        $stmt->execute();
        $db = null;
        return;
} catch(PDOException $e) {
        exitWithError($e->getMessage());
}
}
function dbDelete($table, $id, $id_field_name = 'id') {
if (empty($table) || empty($id)) {
        return;
}
$sql = "DELETE FROM {$table} WHERE {$id_field_name}=:{$id_field_name}";
try {
        $db = getDB();
        $stmt = $db->prepare($sql);
        $stmt->bindValue($id_field_name, $id);
        $stmt->execute();
        $db = null;
        return;
} catch(PDOException $e) {
        exitWithError($e->getMessage());
}
```

}

Postman Parameters

Product

http://home.tamk.fi/~c6mahmad/CMD/index.php/Product

Post parameters:

{"Product_Name":"Panasonic","Product_ShortDesc":"Short test","Product_LongDesc":"long test","Product_Stock":"In Stock","Product_Price":"250.90","Product_Wight":"17.01","Product_Location":"Helsinki","Category_idCategory":"5","Staff_idStaff":"1"}

Customer

http://home.tamk.fi/~c6mahmad/CMD/index.php/Customer

Post parameters:

{"Customer_Username":"gold","Customer_Password":"123323","Customer_Email":"gold@test.com","Customer_EmailVerfied":"gold22@test.com","Customer_Firstname":"Mikko","Customer_Lastname":"lialahti","Customer_PhoneNumber":"+35846618123","Customer_Fax":"0213125","Customer_Address":"Yrttikatu 17B","Customer_Address2":"dfg","Customer_Gender":"M","Customer_Country":"Sweden","Customer_State":"stockholm","Customer_City":"stockholm","Customer_Zip":"3432"}

Comment

http://home.tamk.fi/~c6mahmad/CMD/index.php/Comment

Post parameters:

{"Comment_Msg":"The warranty is better that last product!","Customer_idCustomer":"1000","Product_idProduct":"3"}

Rating

http://home.tamk.fi/~c6mahmad/CMD/index.php/Rating

Post parameters:

{"Rating_Value":"4","Customer_idCustomer":"1000","Product_idProduct":"3"}

Orders

http://home.tamk.fi/~c6mahmad/CMD/index.php/Orders

Post parameters:

{"Customer_idCustomer":"1000","Product_idProduct":"3","Product_Staff_idStaff":"1","Order_Amount":"2","Order_ShipName":"moji","Order_ShipAddress":"Satakunnankatu 45","Order_Country":"United states","Order_State":"Masachoset","Order_City":"Texas","Order_Zip":"43v4562","Order_Phone":"+154343243421","Order_Fax":"093114442","Order_Tax":"42.0333","Order_Email":"TestUS@gmail.com","Order_TrackingNumber":"2123434545003"}

Category

http://home.tamk.fi/~c6mahmad/CMD/index.php/Category

Post parameters:

{"Category_Name":"Washing machine","Category_ParentID":"3"}

Table of content

- Introduction
- Database design
    - Customer
    - Product
    - Category
    - Staff
    - Order
    - Comment
    - Rating
- Create tables
- SQL scripts
    - Category Table Scripts
    - Staff Table
    - Product Table
    - Customer Table
    - order Table
    - Comment Table
    - Rating table
- Restful interface
    - Db.php
    - Index.php
    - Function.php
- Postman Parameters
    - Product
    - Customer
    - Comment
    - Rating

- Orders
- Category