Caiman web

Generated by Doxygen 1.9.1

Chapter 1

Hierarchical Index

1.1 Class Hierarchy

This inheritance list is sorted roughly, but not completely, alphabetically:

Chapter 2

Data Structure Index

2.1 Data Structures

Here are the data structures with brief descriptions:

Chapter 3

File Index

3.1 File List

Here is a list of all documented files with brief descriptions:

Chapter 4

Data Structure Documentation

4.1 Administrator Class Reference

Public Member Functions

- __construct ()
- addGame (string $name, string $description, string $imageName, int $consoleId, $gameFileName)
- uploadGame ($gameFileName, $consoleId)
- updateGame ($idGame, $name, $description, $consoleId)
- uploadGameImage ($imageFileName)
- getConsoleFolderName ($id)
- getListConsole ()

Data Fields

- **$search_username** = null
- **$search_password** = null
- **$arrayInfo** = null
- **$psUploadGame** = null
- **$psUploadFile** = null
- **$psUpdateGame** = null

4.1.1 Detailed Description

Definition at line 9 of file administrator.php.

4.1.2 Constructor & Destructor Documentation

4.1.2.1 __construct()

```
__construct ( )
```

default contructor

Definition at line 32 of file administrator.php.

```
00033     {
00034         if ($this->dbh == null) {
00035             try {
00036                 $this->dbh = new PDO('mysql:host=' . HOST . ';dbname=' . DBNAME, USER, PASSWORD,
      array(
00037                     PDO::MYSQL_ATTR_INIT_COMMAND => "SET NAMES utf8",
00038                     PDO::ATTR_PERSISTENT => true
00039                 ));
00040                 // get list of console
00041                 $sqlGetListConsole = "SELECT * FROM consol";
00042                 $this->psGetListConsole = $this->dbh->prepare($sqlGetListConsole);
00043                 $this->psGetListConsole->setFetchMode(PDO::FETCH_ASSOC);
00044 
00045                 // upload game
00046                 $sqlUploadGame = "INSERT INTO game  (name, description, imageName, idConsole, idFile)
00047                 VALUES (:insert_name, :insert_description, :insert_imageName, :insert_idConsole,
      :insert_idFile)";
00048                 $this->psUploadGame = $this->dbh->prepare($sqlUploadGame);
00049                 $this->psUploadGame->setFetchMode(PDO::FETCH_ASSOC);
00050 
00051                 // upload file
00052                 $sqlUploadFile = "INSERT INTO file  (filename, dateUpdate)
00053                 VALUES (:insert_filename, NOW() )";
00054                 $this->psUploadFile = $this->dbh->prepare($sqlUploadFile);
00055                 $this->psUploadFile->setFetchMode(PDO::FETCH_ASSOC);
00056 
00057                 // update game
00058                 $sqlUpdateGane = "UPDATE game  SET name = :update_name, description =
      :update_description, idConsole = :update_idConsole WHERE id = :update_id";
00059                 $this->psUpdateGame = $this->dbh->prepare($sqlUpdateGane);
00060                 $this->psUpdateGame->setFetchMode(PDO::FETCH_ASSOC);
00061 
00062                 // get folder name of console
00063                 $sqlGetNameConsoleFolder = "SELECT folderName FROM consol WHERE id = :console_id";
00064                 $this->psGetNameConsoleFolder = $this->dbh->prepare($sqlGetNameConsoleFolder);
00065                 $this->psGetNameConsoleFolder->setFetchMode(PDO::FETCH_ASSOC);
00066             } catch (PDOException $e) {
00067                 print "Erreur !: " . $e->getMessage() . "<br>";
00068                 die();
00069             }
00070         }
00071     }
```

4.1.3 Member Function Documentation

4.1.3.1 addGame()

```
addGame (
            string $name,
            string $description,
            string $imageName,
            int $consoleId,
             $gameFileName )
```

add a game to the database

Parameters

string	*$name*	
string	*$description*	
string	*$imageName*	
integer	*$consoleId*	
	[type]	$gameFileName

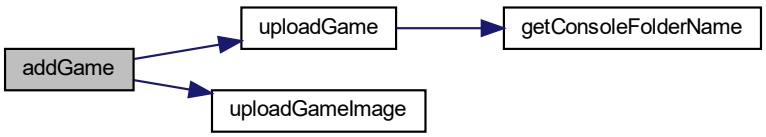
Returns

void

Definition at line 82 of file administrator.php.

```
00083     {
00084
00085         if ($this->uploadGame($gameFileName, $consoleId) && $this->uploadGameImage($imageName)) {
00086             try {
00087                 $this->psUploadFile->execute(array(':insert_filename' => $gameFileName));
00088             } catch (PDOException $e) {
00089                 print "Erreur !: " . $e->getMessage() . "<br>";
00090                 die();
00091             }
00092             $lastInsertId = $this->dbh->lastInsertId();
00093             try {
00094                 $this->psUploadGame->execute(array(':insert_name' => $name, ':insert_description' =>
      $description, ':insert_imageName' => $imageName, ':insert_idConsole' => $consoleId, ':insert_idFile'
      => $lastInsertId));
00095             } catch (PDOException $e) {
00096                 print "Erreur !: " . $e->getMessage() . "<br>";
00097                 die();
00098             }
00099         }
00100     }
```

Here is the call graph for this function:



4.1.3.2 getConsoleFolderName()

```
getConsoleFolderName (
              $id )
```

get the path name of an console

Parameters

[type]	$id

Returns

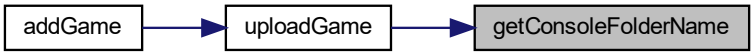
void

Definition at line 197 of file administrator.php.

```
00198     {
00199         $returnArray = null;
00200         try {
```

```
00201             $this->psGetNameConsoleFolder->execute(array(':console_id' => $id));
00202             $returnArray = $this->psGetNameConsoleFolder->fetchAll();
00203         } catch (PDOException $e) {
00204             print "Erreur !: " . $e->getMessage() . "<br>";
00205             die();
00206         }
00207         return $returnArray[0]['folderName'];
00208     }
```

Here is the caller graph for this function:



4.1.3.3 getListConsole()

```
getListConsole ( )
```

returns list of all consoles

Returns

void

Definition at line 214 of file administrator.php.

```
00215     {
00216         $returnArray = null;
00217         try {
00218             $this->psGetListConsole->execute();
00219             $returnArray = $this->psGetListConsole->fetchAll();
00220         } catch (PDOException $e) {
00221             print "Erreur !: " . $e->getMessage() . "<br>";
00222             die();
00223         }
00224         return $returnArray;
00225     }
```

4.1.3.4 updateGame()

```
updateGame (
              $idGame,
              $name,
              $description,
              $consoleId )
```

update da of a game

Parameters

[type]	$idGame
[type]	$name
[type]	$description
[type]	$consoleId

Returns

void

Definition at line 147 of file administrator.php.

```
00148     {
00149         try {
00150             $this->psUpdateGame->execute(array(':update_name' => $name, ':update_description' =>
      $description, ':update_idConsole' => $consoleId, ':update_id' => $idGame));
00151         } catch (PDOException $e) {
00152             print "Erreur !: " . $e->getMessage() . "<br>";
00153             die();
00154         }
00155     }
```

4.1.3.5 uploadGame()

```
uploadGame (
              $gameFileName,
              $consoleId )
```

upload a game

Parameters

[type]	$gameFileName
[type]	$consoleId

Returns

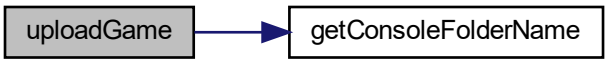
void

Definition at line 108 of file administrator.php.

```
00109     {
00110         $uploadIsValid = false;
00111         $target_dir = "../games/" . $this->getConsoleFolderName($consoleId) . "/";
00112
00113         $target_file =  basename($_FILES["fileGame"]["name"]);
00114         $uploadOk = 1;
00115         $fileType = strtolower(pathinfo($target_file, PATHINFO_EXTENSION));
00116
00117         //rename file
00118         $newfilename = $gameFileName . '.' . $fileType;
00119
00120
00121         // Check if file already exists
00122         if (file_exists($target_file)) {
00123             echo "Sorry, file already exists.";
00124             $uploadOk = 0;
00125         }
00126         if ($uploadOk == 0) {
```

```
00127             echo "Sorry, your file was not uploaded.";
00128             // if everything is ok, try to upload file
00129         } else {
00130             if (move_uploaded_file($_FILES["fileGame"]["tmp_name"], $target_dir . $newfilename)) {
00131                 $uploadIsValid = true;
00132             } else {
00133                 //Sorry, there was an error uploading your file
00134             }
00135         }
00136         return $uploadIsValid;
00137     }
```

Here is the call graph for this function:



Here is the caller graph for this function:

4.1.3.6 uploadGameImage()

```
uploadGameImage (
              $imageFileName )
```

upload an image

Parameters

[type]	$imageFileName

Returns

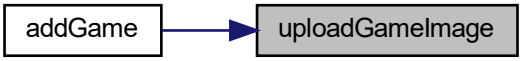
void

Definition at line 162 of file administrator.php.

```
00163     {
```

```
00164         $uploadIsValid = false;
00165         $target_dir = "img/games/";
00166 
00167         $target_file =  basename($_FILES["image"]["name"]);
00168         $uploadOk = 1;
00169         $fileType = strtolower(pathinfo($target_file, PATHINFO_EXTENSION));
00170 
00171         //rename file
00172         $newfilename = $imageFileName;
00173 
00174         // Check if file already exists
00175         if (file_exists($target_file)) {
00176             echo "Sorry, file already exists.";
00177             $uploadOk = 0;
00178         }
00179         if ($uploadOk == 0) {
00180             echo "Sorry, your file was not uploaded.";
00181             // if everything is ok, try to upload file
00182         } else {
00183             if (move_uploaded_file($_FILES["image"]["tmp_name"], $target_dir . $newfilename)) {
00184                 $uploadIsValid = true;
00185             } else {
00186                 //Sorry, there was an error uploading your file
00187             }
00188         }
00189         return $uploadIsValid;
00190     }
```

Here is the caller graph for this function:

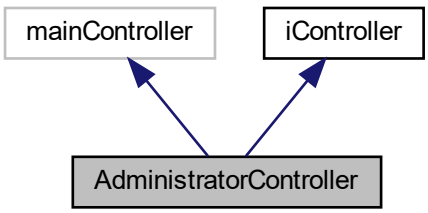


The documentation for this class was generated from the following file:

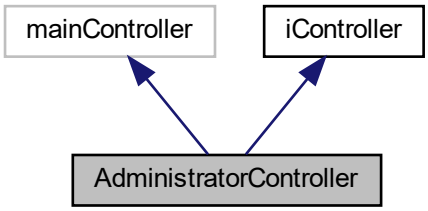
- models/administrator.php

4.2 AdministratorController Class Reference

Inheritance diagram for AdministratorController:



Collaboration diagram for AdministratorController:



Public Member Functions

- formHandler ()
- __construct ()
- printHTML ()

Data Fields

- **$administrator**
- **$game**
- **$categorie**

4.2.1 Detailed Description

Definition at line 10 of file administratorController.php.

4.2.2 Constructor & Destructor Documentation

4.2.2.1 __construct()

```
__construct ( )
```

default constructor

Definition at line 192 of file administratorController.php.

```
00193   {
00194     $this->administrator  = new Administrator();
00195     $this->game = new Games();
00196     $this->categorie = new Categories();
00197   }
```

4.2.3 Member Function Documentation

4.2.3.1 formHandler()

`formHandler ( )`

used to handle if the user has resquest something

Returns

void

Implements iController.

Definition at line 23 of file administratorController.php.

```
00024   {
00025     $result = null;
00026
00027
00028     if (isset($_GET['e'])) {
00029       $this->e = filter_input(INPUT_GET, 'e', FILTER_SANITIZE_STRING);
00030       //redirige l'utilisateur qui n'a pas les bon droits
00031       $this->allowAccessTo(array(1));
00032     }
00033     // update game
00034     if ($this->e == "updateGame") {
00035       if (isset($_GET['id'])) {
00036         $requestGame = filter_input(INPUT_GET, 'id', FILTER_SANITIZE_STRING);
00037         $this->idGameToUpdate = $requestGame;
00038       } else {
00039         header('Location:' . $_SERVER['HTTP_REFERER']);
00040         exit;
00041       }
00042     }
00043
00044     // add game categorie
00045     if ($this->e == "addGameCategorie") {
00046       if (isset($_GET['id'])) {
00047         $requestGame = filter_input(INPUT_GET, 'id', FILTER_SANITIZE_STRING);
00048         $this->idGameToUpdate = $requestGame;
00049       } else {
00050         header('Location:' . $_SERVER['HTTP_REFERER']);
00051         exit;
00052       }
00053     }
00054
00055     // add categorie to game
00056     if ($this->e == "addGameCategorieAdd") {
00057       if (isset($_GET['idGame'])) {
00058         $idGame = filter_input(INPUT_GET, 'idGame', FILTER_SANITIZE_STRING);
00059       } else {
00060         header('Location:' . $_SERVER['HTTP_REFERER']);
00061         exit;
00062       }
00063       if (isset($_GET['idCategorie'])) {
00064         $idCategorie = filter_input(INPUT_GET, 'idCategorie', FILTER_SANITIZE_STRING);
00065       } else {
00066         header('Location:' . $_SERVER['HTTP_REFERER']);
00067         exit;
00068       }
00069
00070       $this->categorie->addCategorieToGame($idGame, $idCategorie);
00071       header('Location:' . $_SERVER['HTTP_REFERER']);
00072       exit;
00073     }
00074
00075     // delete categorie from a game
00076     if ($this->e == "delGameCategorie") {
00077       if (isset($_GET['idGame'])) {
00078         $idGame = filter_input(INPUT_GET, 'idGame', FILTER_SANITIZE_STRING);
00079       } else {
00080         header('Location:' . $_SERVER['HTTP_REFERER']);
00081         exit;
```

```
00082       }
00083       if (isset($_GET['idCategorie'])) {
00084         $idCategorie = filter_input(INPUT_GET, 'idCategorie', FILTER_SANITIZE_STRING);
00085       } else {
00086         header('Location:' . $_SERVER['HTTP_REFERER']);
00087         exit;
00088       }
00089
00090       $this->categorie->delCategorieFromGame($idGame, $idCategorie);
00091       header('Location:' . $_SERVER['HTTP_REFERER']);
00092       exit;
00093     }
00094
00095     //add game
00096     if ($this->e == "addGameUpload") {
00097       if (isset($_POST['name'])) {
00098         $name = filter_input(INPUT_POST, 'name', FILTER_SANITIZE_STRING);
00099       } else {
00100         header('Location:' . $_SERVER['HTTP_REFERER']);
00101         exit;
00102       }
00103
00104       if (isset($_POST['description'])) {
00105         $description = filter_input(INPUT_POST, 'description', FILTER_SANITIZE_STRING);
00106       } else {
00107         header('Location:' . $_SERVER['HTTP_REFERER']);
00108         exit;
00109       }
00110
00111       if (isset($_POST['imageName'])) {
00112         $imageName = filter_input(INPUT_POST, 'imageName', FILTER_SANITIZE_STRING);
00113       } else {
00114         header('Location:' . $_SERVER['HTTP_REFERER']);
00115         exit;
00116       }
00117
00118       if (isset($_POST['console'])) {
00119         $consoleId = filter_input(INPUT_POST, 'console', FILTER_SANITIZE_STRING);
00120       } else {
00121         header('Location:' . $_SERVER['HTTP_REFERER']);
00122         exit;
00123       }
00124
00125       if (isset($_POST['gameFileName'])) {
00126         $gameFileName = filter_input(INPUT_POST, 'gameFileName', FILTER_SANITIZE_STRING);
00127       } else {
00128         header('Location:' . $_SERVER['HTTP_REFERER']);
00129         exit;
00130       }
00131
00132
00133       $this->administrator->addGame($name, $description, $imageName, $consoleId, $gameFileName);
00134
00135       header('Location:' . $_SERVER['HTTP_REFERER']);
00136       exit;
00137     }
00138
00139     //add game
00140     if ($this->e == "updateGameUpdate") {
00141       if (isset($_POST['name'])) {
00142         $name = filter_input(INPUT_POST, 'name', FILTER_SANITIZE_STRING);
00143       } else {
00144         header('Location:' . $_SERVER['HTTP_REFERER']);
00145         exit;
00146       }
00147
00148       if (isset($_POST['description'])) {
00149         $description = filter_input(INPUT_POST, 'description', FILTER_SANITIZE_STRING);
00150       } else {
00151         header('Location:' . $_SERVER['HTTP_REFERER']);
00152         exit;
00153       }
00154
00155       if (isset($_POST['console'])) {
00156         $consoleId = filter_input(INPUT_POST, 'console', FILTER_SANITIZE_STRING);
00157       } else {
00158         header('Location:' . $_SERVER['HTTP_REFERER']);
00159         exit;
00160       }
00161
00162       if (isset($_POST['idGame'])) {
00163         $idGame = filter_input(INPUT_POST, 'idGame', FILTER_SANITIZE_STRING);
00164       } else {
00165         header('Location:' . $_SERVER['HTTP_REFERER']);
00166         exit;
00167       }
00168
```

```
00169
00170       $this->administrator->updateGame($idGame, $name, $description, $consoleId);
00171
00172       header('Location:' . $_SERVER['HTTP_REFERER']);
00173       exit;
00174     }
00175
00176
00177     if ($this->e == "addCategorie") {
00178       if (isset($_POST['name'])) {
00179         $name = filter_input(INPUT_POST, 'name', FILTER_SANITIZE_STRING);
00180       }
00181
00182       if (isset($name)) {
00183         $result = $this->categorie->addCategorie($name);
00184       }
00185     }
00186   }
```

4.2.3.2 printHTML()

```
printHTML ( )
```

print the html for the resquested content

Returns

void

Implements iController.

Definition at line 205 of file administratorController.php.

```
00206   {
00207
00208     $html = '<main style="margin-top:20px">
00209         <div class="container-md">';
00210     $html .= $this->errorHandler();
00211     if ($this->e == null) {
00212       $html .= $this->htmlAdministratorHome();
00213     }
00214
00215     if ($this->e == "addGame") {
00216       $html .= $this->htmlNewGame();
00217     }
00218
00219     if ($this->e == "updateGame") {
00220       $html .= $this->htmlUpdateGame();
00221     }
00222
00223     if ($this->e == "addCategorie") {
00224       $html .= $this->htmlAddCategorie();
00225     }
00226
00227     if ($this->e == "addGameCategorie") {
00228       $html .= $this->htmlAddCategorieToGame();
00229     }
00230
00231
00232
00233     $html .= "</div></main> ";
00234     echo $html;
00235   }
```

The documentation for this class was generated from the following file:

- controllers/administratorController.php

4.3 Categories Class Reference

Public Member Functions

- __construct ()
- getListAllCategories ()
- getCategoriesOfGame (int $idGame)
- addCategorie (string $categorieName)
- addCategorieToGame (int $idGame, int $idCategorie)
- delCategorieFromGame (int $idGame, int $idCategorie)

4.3.1 Detailed Description

Definition at line 10 of file categorie.php.

4.3.2 Constructor & Destructor Documentation

4.3.2.1 __construct()

```
__construct ( )
```

default constructor

Definition at line 31 of file categorie.php.

```
00032     {
00033         if ($this->dbh == null) {
00034             try {
00035                 $this->dbh = new PDO('mysql:host=' . HOST . ';dbname=' . DBNAME, USER, PASSWORD,
      array(
00036                     PDO::MYSQL_ATTR_INIT_COMMAND => "SET NAMES utf8",
00037                     PDO::ATTR_PERSISTENT => true
00038                 ));
00039                 //get all categories
00040                 $sqlGetAllCategories = "SELECT * FROM categorie";
00041                 $this->psGetAllCategories = $this->dbh->prepare($sqlGetAllCategories);
00042                 $this->psGetAllCategories->setFetchMode(PDO::FETCH_ASSOC);
00043 
00044                 //add categorie
00045                 $sqlAddCategorie = "INSERT INTO categorie (name) VALUES (:categorie_name)";
00046                 $this->psAddCategorie = $this->dbh->prepare($sqlAddCategorie);
00047                 $this->psAddCategorie->setFetchMode(PDO::FETCH_ASSOC);
00048 
00049                 //add categorie to game
00050                 $sqlAddCategorieToGame = "INSERT INTO gamehascategorie (idGame,idCategorie) VALUES
      (:insert_idGame, :insert_idCategorie)";
00051                 $this->psAddCategorieToGame = $this->dbh->prepare($sqlAddCategorieToGame);
00052 
00053                 //check if game has a specific categorie
00054                 $sqlCheckIfGameHasCategorie = "SELECT * FROM gamehascategorie WHERE idCategorie =
      :insert_idCategorie AND idGame = :insert_idGame";
00055                 $this->psCheckIfGameHasCategorie = $this->dbh->prepare($sqlCheckIfGameHasCategorie);
00056                 $this->psCheckIfGameHasCategorie->setFetchMode(PDO::FETCH_ASSOC);
00057 
00058                 //del categorie from game
00059                 $sqlDelCategorieFromGame = "DELETE FROM gamehascategorie  WHERE idCategorie =
      :del_idCategorie AND idGame = :del_idGame";
00060                 $this->psDelCategorieFromGame = $this->dbh->prepare($sqlDelCategorieFromGame);
00061 
00062                 //get categories of a game
00063                 $sqlGameCategorie = "SELECT c.name, c.id FROM `gamehascategorie` as ghc
00064                 LEFT JOIN categorie as c
00065                 ON ghc.idCategorie = c.id
```

```
00066                 LEFT JOIN game as g
00067                 ON ghc.idGame = g.id
00068                 WHERE idGame = :search_id";
00069                 $this->psGameCategorie = $this->dbh->prepare($sqlGameCategorie);
00070                 $this->psGameCategorie->setFetchMode(PDO::FETCH_ASSOC);
00071             } catch (PDOException $e) {
00072                 print "Erreur !: " . $e->getMessage() . "<br>";
00073                 die();
00074             }
00075         }
00076     }
```

4.3.3 Member Function Documentation

4.3.3.1 addCategorie()

```
addCategorie (
             string $categorieName )
```

add a new categorie in the database

Parameters

string	*$categorieName*	

Returns

void

Definition at line 117 of file categorie.php.

```
00118     {
00119         try {
00120             $this->psAddCategorie->execute(array(':categorie_name' => $categorieName));
00121             $result = $this->psAddCategorie->fetchAll();
00122         } catch (PDOException $e) {
00123             print "Erreur !: " . $e->getMessage() . "<br>";
00124             die();
00125         }
00126         return $result;
00127     }
```

4.3.3.2 addCategorieToGame()

```
addCategorieToGame (
             int $idGame,
             int $idCategorie )
```

add a categorie to a game

Parameters

integer	*$idGame*	
integer	*$idCategorie*	

Returns

void

Definition at line 135 of file categorie.php.

```
00136     {
00137         $result = null;
00138         try {
00139             $this->psCheckIfGameHasCategorie->execute(array(':insert_idCategorie' => $idCategorie,
      ':insert_idGame' => $idGame));
00140             $result = $this->psCheckIfGameHasCategorie->fetchAll();
00141         } catch (PDOException $e) {
00142             print "Erreur !: " . $e->getMessage() . "<br>";
00143             die();
00144         }
00145         if ($result == null) {
00146             try {
00147                 $this->psAddCategorieToGame->execute(array(':insert_idCategorie' => $idCategorie,
      ':insert_idGame' => $idGame));
00148             } catch (PDOException $e) {
00149                 print "Erreur !: " . $e->getMessage() . "<br>";
00150                 die();
00151             }
00152         }
00153     }
```

4.3.3.3 delCategorieFromGame()

```
delCategorieFromGame (
            int $idGame,
            int $idCategorie )
```

delete a cotegorie of a game

Parameters

integer	*$idGame*	
integer	*$idCategorie*	

Returns

void

Definition at line 161 of file categorie.php.

```
00162     {
00163         $result = null;
00164         try {
00165             $this->psDelCategorieFromGame->execute(array(':del_idCategorie' => $idCategorie,
      ':del_idGame' => $idGame));
00166         } catch (PDOException $e) {
00167             print "Erreur !: " . $e->getMessage() . "<br>";
00168             die();
00169         }
00170     }
```

4.3.3.4 getCategoriesOfGame()

```
getCategoriesOfGame (
            int $idGame )
```

get the categories of a game

Parameters

integer	*$idGame*	

Returns

list of categories

Definition at line 100 of file categorie.php.

```
00101     {
00102         try {
00103             $this->psGameCategorie->execute(array(':search_id' => $idGame));
00104             $result = $this->psGameCategorie->fetchAll();
00105         } catch (PDOException $e) {
00106             print "Erreur !: " . $e->getMessage() . "<br>";
00107             die();
00108         }
00109         return $result;
00110     }
```

4.3.3.5 getListAllCategories()

```
getListAllCategories ( )
```

returns list of all categories

Returns

array with all list

Definition at line 82 of file categorie.php.

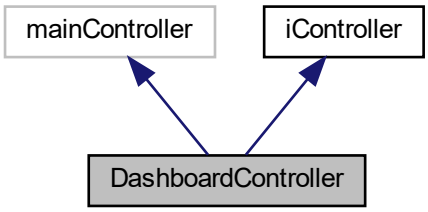
```
00083     {
00084
00085         try {
00086             $this->psGetAllCategories->execute();
00087             $result = $this->psGetAllCategories->fetchAll();
00088         } catch (PDOException $e) {
00089             print "Erreur !: " . $e->getMessage() . "<br>";
00090             die();
00091         }
00092         return $result;
00093     }
```

The documentation for this class was generated from the following file:

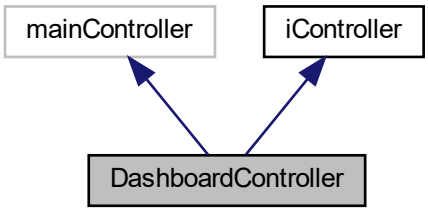
- models/categorie.php

4.4 DashboardController Class Reference

Inheritance diagram for DashboardController:



Collaboration diagram for DashboardController:



Public Member Functions

- **formHandler** ()
- **printHTML** ()
- **htmlFormUpdatePassword** ()

Data Fields

- **$game**

4.4.1 Detailed Description

Definition at line 10 of file dashboardController.php.

The documentation for this class was generated from the following file:

- controllers/dashboardController.php

4.5 Download Class Reference

Public Member Functions

- downloadCaiman ()

Data Fields

- **$aMemberVar** = 'aMemberVar Member Variable'
- **$aFuncName** = 'aMemberFunc'

4.5.1 Detailed Description

Definition at line 10 of file download.php.

4.5.2 Member Function Documentation

4.5.2.1 downloadCaiman()

```
downloadCaiman ( )
```

used to download caiman

Returns

void

Definition at line 20 of file download.php.

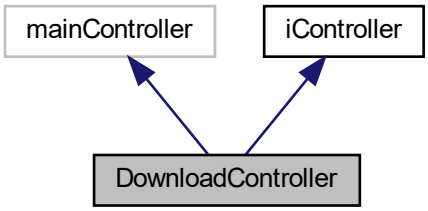
```
00021     {
00022         $filename = '../release/caiman.jpg'; // of course find the exact filename....
00023         header('Pragma: public');
00024         header('Expires: 0');
00025         header('Cache-Control: must-revalidate, post-check=0, pre-check=0');
00026         header('Cache-Control: private', false); // required for certain browsers
00027         header('Content-Type: application/jpg');
00028 
00029         header('Content-Disposition: attachment; filename="' . basename($filename) . '";');
00030         header('Content-Transfer-Encoding: binary');
00031         header('Content-Length: ' . filesize($filename));
00032 
00033         readfile($filename);
00034 
00035         exit;
00036     }
```

The documentation for this class was generated from the following file:

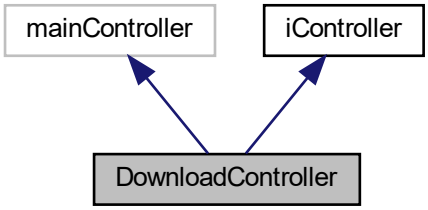
- models/download.php

4.6 DownloadController Class Reference

Inheritance diagram for DownloadController:



Collaboration diagram for DownloadController:



Public Member Functions

- **formHandler** ()
- **printHTML** ()

Data Fields

- **$download**

4.6.1 Detailed Description

Definition at line 9 of file downloadController.php.

The documentation for this class was generated from the following file:

- controllers/downloadController.php

4.7 Games Class Reference

Public Member Functions

- __construct ()
- getAllGames ()
- getRequestGames (string $gameName)
- getTimeInGameUser (int $idUser, int $idGame)
- getListOfGameWithTimeUser (int $idUser)
- getGameDetail (int $idGame)
- getGamesInCategorie (int $idCategorie)
- getFavoriteGamesOfUser (int $idUser)
- addGameToFavoris (int $idUser, int $idGame)
- removeGameFromFavoris (int $idUser, int $idGame)
- checkIfGameIsAlreadyInFavoris (int $idUser, int $idGame)

4.7.1 Detailed Description

Definition at line 9 of file games.php.

4.7.2 Constructor & Destructor Documentation

4.7.2.1 __construct()

```
__construct ( )
```

default contructor

Definition at line 37 of file games.php.

```
00038     {
00039         if ($this->dbh == null) {
00040             try {
00041                 $this->dbh = new PDO('mysql:host=' . HOST . ';dbname=' . DBNAME, USER, PASSWORD,
      array(
00042                     PDO::MYSQL_ATTR_INIT_COMMAND => "SET NAMES utf8",
00043                     PDO::ATTR_PERSISTENT => true
00044                 ));
00045                 //get all games
00046                 $sqlGetAllGames = "SELECT * FROM game";
00047                 $this->psGetAllGames = $this->dbh->prepare($sqlGetAllGames);
00048                 $this->psGetAllGames->setFetchMode(PDO::FETCH_ASSOC);
00049 
00050                 //get request games
00051                 $sqlRequestGames = "SELECT * FROM game WHERE name LIKE :search_game";
00052                 $this->psRequestGames = $this->dbh->prepare($sqlRequestGames);
00053                 $this->psRequestGames->setFetchMode(PDO::FETCH_ASSOC);
00054 
00055                 //get Time in game user
00056                 $sqlTimeInGame = "SELECT * FROM timeingame WHERE idGame = :search_idGame AND idUser =
      :search_idUser ";
00057                 $this->psGetTimeInGame = $this->dbh->prepare($sqlTimeInGame);
00058                 $this->psGetTimeInGame->setFetchMode(PDO::FETCH_ASSOC);
00059 
00060                 //get game with time user
00061                 $sqlGetGameWithTime = "SELECT * FROM timeingame WHERE idUser = :search_idUser ORDER BY
      timeInMinute DESC";
00062                 $this->psGetGameWithTime = $this->dbh->prepare($sqlGetGameWithTime);
00063                 $this->psGetGameWithTime->setFetchMode(PDO::FETCH_ASSOC);
00064 
```

```
00065                 //get detail game
00066                 $sqlGameDetail = "SELECT * FROM game WHERE id = :search_id";
00067                 $this->psGameDetail = $this->dbh->prepare($sqlGameDetail);
00068                 $this->psGameDetail->setFetchMode(PDO::FETCH_ASSOC);
00069 
00070                 //add game to favoris
00071                 $sqlAddGameToFavoris = "INSERT INTO favoritegame  (idGame, idUser)
00072                 VALUES (:search_idGame, :search_idUser)";
00073                 $this->psAddGameToFavori = $this->dbh->prepare($sqlAddGameToFavoris);
00074                 $this->psAddGameToFavori->setFetchMode(PDO::FETCH_ASSOC);
00075 
00076                 //remove game to favoris
00077                 $sqlRemoveGameFormFavoris = "DELETE FROM favoritegame
00078                 WHERE idUser = :search_idUser AND idGame = :search_idGame";
00079                 $this->psRemoveGameFromFavori = $this->dbh->prepare($sqlRemoveGameFormFavoris);
00080                 $this->psRemoveGameFromFavori->setFetchMode(PDO::FETCH_ASSOC);
00081 
00082                 //check if already in favoris
00083                 $sqlCheckIfAlreadyFavoris = "SELECT * FROM favoritegame
00084                 WHERE iduser = :search_idUser AND idGame = :search_idGame";
00085                 $this->psCheckIfFavoris = $this->dbh->prepare($sqlCheckIfAlreadyFavoris);
00086                 $this->psCheckIfFavoris->setFetchMode(PDO::FETCH_ASSOC);
00087 
00088                 //get favorite game of user
00089                 $sqlFavoriteGameOfUser = "SELECT g.name, g.id, g.imageName FROM `favoritegame` as fg
00090                 LEFT JOIN game as g
00091                 ON fg.idGame = g.id
00092                 LEFT JOIN user as u
00093                 ON fg.iduser = u.id
00094                 WHERE iduser = :search_id";
00095                 $this->psFavoriteGameOfUser = $this->dbh->prepare($sqlFavoriteGameOfUser);
00096                 $this->psFavoriteGameOfUser->setFetchMode(PDO::FETCH_ASSOC);
00097 
00098 
00099 
00100                 //get list of games in a categorie
00101                 $sqlGameInCategorie = "SELECT g.name, g.id, g.imageName FROM `gamehascategorie` as ghc
00102                 LEFT JOIN game as g
00103                 ON ghc.idGame = g.id
00104                 LEFT JOIN categorie as c
00105                 ON ghc.idCategorie = c.id
00106                 WHERE idCategorie = :search_id";
00107                 $this->psGameInCategorie = $this->dbh->prepare($sqlGameInCategorie);
00108                 $this->psGameInCategorie->setFetchMode(PDO::FETCH_ASSOC);
00109 
00110             } catch (PDOException $e) {
00111                 print "Erreur !: " . $e->getMessage() . "<br>";
00112                 die();
00113             }
00114         }
00115     }
```

4.7.3 Member Function Documentation

4.7.3.1 addGameToFavoris()

```
addGameToFavoris (
            int $idUser,
            int $idGame )
```

returns add a game to a user's favorites

Returns

void

Definition at line 247 of file games.php.

```
00248     {
00249
00250         try{
00251             $this->psAddGameToFavori->execute(array(':search_idUser' => $idUser,':search_idGame' =>
      $idGame));
00252             $result = $this->psAddGameToFavori->fetchAll();
00253
00254         }catch (PDOException $e) {
00255             print "Erreur !: " . $e->getMessage() . "<br>";
00256             die();
00257         }
00258         return $result;
00259     }
```

4.7.3.2 checkIfGameIsAlreadyInFavoris()

```
checkIfGameIsAlreadyInFavoris (
             int $idUser,
             int $idGame )
```

returns if a game is already in favorite

Returns

void

Definition at line 282 of file games.php.

```
00283     {
00284         $boolResult = true;
00285
00286         try{
00287             $this->psCheckIfFavoris->execute(array(':search_idUser' => $idUser,':search_idGame' =>
      $idGame));
00288             $result = $this->psCheckIfFavoris->fetchAll();
00289             if ($result != null) {
00290                 $boolResult = false;
00291             }
00292
00293         }catch (PDOException $e) {
00294             print "Erreur !: " . $e->getMessage() . "<br>";
00295             die();
00296         }
00297         return $boolResult;
00298     }
```

4.7.3.3 getAllGames()

```
getAllGames ( )
```

returns all games from the database

Returns

array of all games

Definition at line 121 of file games.php.

```
00122     {
00123
00124         try{
00125             $this->psGetAllGames->execute();
00126             $result = $this->psGetAllGames->fetchAll();
00127
00128
00129         }catch (PDOException $e) {
00130             print "Erreur !: " . $e->getMessage() . "<br>";
00131             die();
00132         }
00133         return $result;
00134     }
```

4.7.3.4 getFavoriteGamesOfUser()

```
getFavoriteGamesOfUser (
             int $idUser )
```

returns the list of the favorite game of a user

Returns

array of games

Definition at line 229 of file games.php.

```
00230     {
00231         try{
00232             $this->psFavoriteGameOfUser->execute(array(':search_id' => $idUser));
00233             $result = $this->psFavoriteGameOfUser->fetchAll();
00234
00235
00236         }catch (PDOException $e) {
00237             print "Erreur !: " . $e->getMessage() . "<br>";
00238             die();
00239         }
00240         return $result;
00241     }
```

4.7.3.5 getGameDetail()

```
getGameDetail (
             int $idGame )
```

returns details of a specif game

Returns

array with game detail

Definition at line 193 of file games.php.

```
00194     {
00195
00196         try{
00197             $this->psGameDetail->execute(array(':search_id' => $idGame));
00198             $result = $this->psGameDetail->fetchAll();
00199
00200
00201         }catch (PDOException $e) {
00202             print "Erreur !: " . $e->getMessage() . "<br>";
00203             die();
00204         }
00205         return $result;
00206     }
```

4.7.3.6 getGamesInCategorie()

```
getGamesInCategorie (
            int $idCategorie )
```

returns games of a certain category

Returns

array of games

Definition at line 213 of file games.php.

```
00214     {
00215         try{
00216             $this->psGameInCategorie->execute(array(':search_id' => $idCategorie));
00217             $result = $this->psGameInCategorie->fetchAll();
00218         }catch (PDOException $e) {
00219             print "Erreur !: " . $e->getMessage() . "<br>";
00220             die();
00221         }
00222         return $result;
00223     }
```

4.7.3.7 getListOfGameWithTimeUser()

```
getListOfGameWithTimeUser (
            int $idUser )
```

returns returns the games the player has played

Returns

array of games

Definition at line 175 of file games.php.

```
00176     {
00177         try{
00178             $this->psGetGameWithTime->execute(array(':search_idUser' => $idUser));
00179             $result = $this->psGetGameWithTime->fetchAll();
00180
00181
00182         }catch (PDOException $e) {
00183             print "Erreur !: " . $e->getMessage() . "<br>";
00184             die();
00185         }
00186         return $result;
00187     }
```

4.7.3.8 getRequestGames()

```
getRequestGames (
            string $gameName )
```

returns games whose name matches the search

Returns

array of games

Definition at line 140 of file games.php.

```
00141     {
00142
00143         try{
00144             $this->psRequestGames->execute(array(':search_game' => '%'.$gameName.'%'));
00145             $result = $this->psRequestGames->fetchAll();
00146
00147
00148         }catch (PDOException $e) {
00149             print "Erreur !: " . $e->getMessage() . "<br>";
00150             die();
00151         }
00152         return $result;
00153     }
```

4.7.3.9 getTimeInGameUser()

```
getTimeInGameUser (
            int $idUser,
            int $idGame )
```

returns play time of a user for a specific game

Returns

array time in game

Definition at line 159 of file games.php.

```
00160     {
00161         try{
00162             $this->psGetTimeInGame->execute(array(':search_idGame' => $idGame, ':search_idUser' =>
      $idUser));
00163             $result = $this->psGetTimeInGame->fetchAll();
00164         }catch (PDOException $e) {
00165             print "Erreur !: " . $e->getMessage() . "<br>";
00166             die();
00167         }
00168         return $result;
00169     }
```

4.7.3.10 removeGameFromFavoris()

```
removeGameFromFavoris (
            int $idUser,
            int $idGame )
```

returns remove a game to a user's favorites

Returns

void

Definition at line 265 of file games.php.

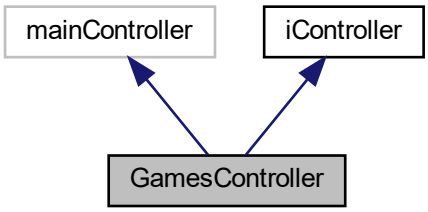
```
00266     {
00267
00268         try{
00269             $this->psRemoveGameFromFavori->execute(array(':search_idUser' => $idUser,':search_idGame'
      => $idGame));
00270
00271
00272         }catch (PDOException $e) {
00273             print "Erreur !: " . $e->getMessage() . "<br>";
00274             die();
00275         }
00276     }
```

The documentation for this class was generated from the following file:

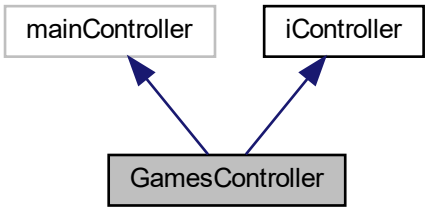
- models/games.php

4.8 GamesController Class Reference

Inheritance diagram for GamesController:



Collaboration diagram for GamesController:



Public Member Functions

- **formHandler** ()
- **printHTML** ()
- **getListAllGames** ()
- **getRequestedGames** ()
- **getGameDetail** ()
- **getGamesFromCategorie** ()
- **recherchFull** ()
- **recherchNotFull** ()

Data Fields

- **$games**
- **$categorie**
- **$requestedgame** = null

4.8.1 Detailed Description

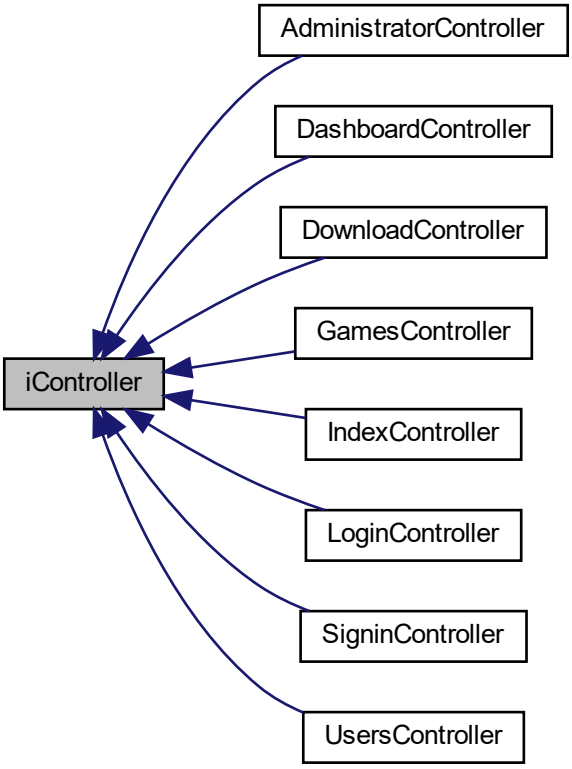
Definition at line 9 of file gamesController.php.

The documentation for this class was generated from the following file:

- controllers/gamesController.php

4.9 iController Interface Reference

Inheritance diagram for iController:



Public Member Functions

- **formHandler** ()
- **printHTML** ()

4.9.1 Detailed Description

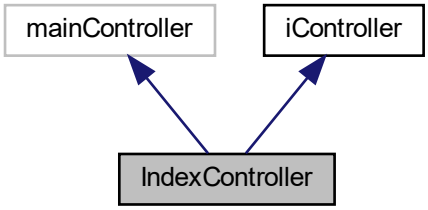
Definition at line 9 of file interfaceController.php.

The documentation for this interface was generated from the following file:

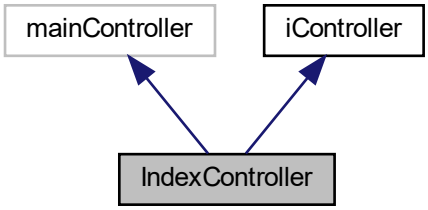
- controllers/interfaceController.php

4.10 IndexController Class Reference

Inheritance diagram for IndexController:



Collaboration diagram for IndexController:



Public Member Functions

- **formHandler** ()
- **printHTML** ()

4.10.1 Detailed Description

Definition at line 9 of file indexController.php.

The documentation for this class was generated from the following file:

- controllers/indexController.php

4.11 Login Class Reference

Public Member Functions

- checkLogin ()

Data Fields

- **$search_username** = null
- **$search_password** = null
- **$arrayInfo** = null

4.11.1 Detailed Description

Definition at line 9 of file login.php.

4.11.2 Member Function Documentation

4.11.2.1 checkLogin()

```
checkLogin ( )
```

check if there is a match

Returns

bool

Definition at line 47 of file login.php.

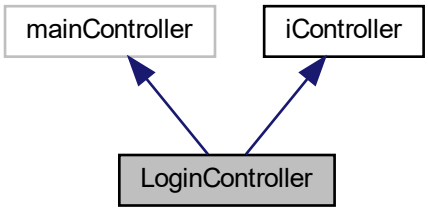
```
00048     {
00049         $returnArray = null;
00050         try{
00051             $this->psLogin->execute(array(':search_username' => $this->search_username));
00052             $result = $this->psLogin->fetchAll();
00053             if ($result != null) {
00054                 if (password_verify( $this->search_password,$result[0]["password"])   ) {
00055                     $returnArray = $result;
00056                     $_SESSION['error'] = "Welcome back: ". $result[0]['username'];
00057                 }else
00058                 {
00059                     $_SESSION['error'] = "Invalid log in";
00060                 }
00061             }
00062 
00063         }catch (PDOException $e) {
00064             print "Erreur !: " . $e->getMessage() . "<br>";
00065             die();
00066         }
00067         return $returnArray;
00068     }
```

The documentation for this class was generated from the following file:

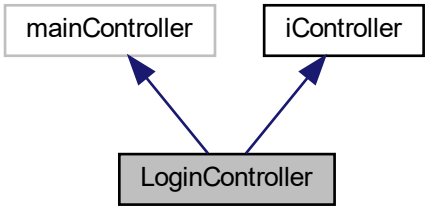
- models/login.php

4.12 LoginController Class Reference

Inheritance diagram for LoginController:



Collaboration diagram for LoginController:



Public Member Functions

- **formHandler** ()
- **printHTML** ()

Data Fields

- **$login**

4.12.1 Detailed Description

Definition at line 9 of file loginController.php.

The documentation for this class was generated from the following file:

- controllers/loginController.php

4.13 MainController Class Reference

Public Member Functions

- **allowAccessTo** ($allowAccessToId)
- **errorHandler** ()

4.13.1 Detailed Description

Definition at line 9 of file mainController.php.

The documentation for this class was generated from the following file:

- controllers/mainController.php

4.14 Signin Class Reference

Public Member Functions

- __construct ()
- newUser ()
- checkIfUsernameAlreadyTaken ()
- checkIfEmailAlreadyTaken ()

Data Fields

- **$insert_username** = null
- **$insert_password** = null
- **$insert_password_repeat** = null
- **$insert_email** = null

4.14.1 Detailed Description

Definition at line 9 of file signin.php.

4.14.2 Constructor & Destructor Documentation

4.14.2.1 __construct()

```
__construct ( )
```

default constructor

Definition at line 30 of file signin.php.

```
00031     {
00032         if ($this->dbh == null) {
00033             try {
00034                 $this->dbh = new PDO('mysql:host=' . HOST . ';dbname=' . DBNAME, USER, PASSWORD,
      array(
00035                     PDO::MYSQL_ATTR_INIT_COMMAND => "SET NAMES utf8",
00036                     PDO::ATTR_PERSISTENT => true
00037                 ));
00038 
00039                 // check if email alredy used
00040                 $sqlrequestEmail = "SELECT * FROM user WHERE email = :search_email ";
00041                 $this->psCheckEmail = $this->dbh->prepare($sqlrequestEmail);
00042                 $this->psCheckEmail->setFetchMode(PDO::FETCH_ASSOC);
00043 
00044                 // check if username alredy used
00045                 $sqlRequestUsername = "SELECT * FROM user WHERE username = :search_username ";
00046                 $this->psCheckUsername = $this->dbh->prepare($sqlRequestUsername);
00047                 $this->psCheckUsername->setFetchMode(PDO::FETCH_ASSOC);
00048 
00049                 $sqlInsert = "INSERT INTO user  (username, password, email)
00050                               VALUES (:insert_username, :insert_password, :insert_email)";
00051                 $this->psInsert = $this->dbh->prepare($sqlInsert);
00052                 $this->psInsert->setFetchMode(PDO::FETCH_ASSOC);
00053             } catch (PDOException $e) {
00054                 print "Erreur !: " . $e->getMessage() . "<br>";
00055                 die();
00056             }
00057         }
00058     }
```

4.14.3 Member Function Documentation

4.14.3.1 checkIfEmailAlreadyTaken()

```
checkIfEmailAlreadyTaken ( )
```

check that the email is not already taken

Returns

void

Definition at line 112 of file signin.php.

```
00113     {
00114         $istaken = true;
00115         try {
00116             $this->psCheckEmail->execute(array(':search_email' => $this->insert_email));
00117             $result = $this->psCheckEmail->fetchAll();
00118             if ($result == null) {
00119                 $istaken = false;
00120             }
00121         } catch (PDOException $e) {
00122             print "Erreur !: " . $e->getMessage() . "<br>";
00123             die();
00124         }
00125         return $istaken;
00126     }
```

Here is the caller graph for this function:

4.14.3.2 checkIfUsernameAlreadyTaken()

```
checkIfUsernameAlreadyTaken ( )
```

check that the username is not already taken

Returns

void

Definition at line 92 of file signin.php.

```
00093     {
00094         $istaken = true;
00095         try {
00096             $this->psCheckUsername->execute(array(':search_username' => $this->insert_username));
00097             $result = $this->psCheckUsername->fetchAll();
00098             if ($result == null) {
00099                 $istaken = false;
00100             }
00101         } catch (PDOException $e) {
00102             print "Erreur !: " . $e->getMessage() . "<br>";
00103             die();
00104         }
00105         return $istaken;
00106     }
```

4.14.3.3 newUser()

```
newUser ( )
```

add a mew user in the database

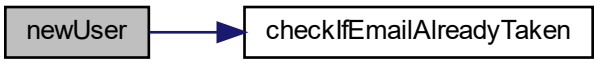
Returns

void

Definition at line 64 of file signin.php.

```
00065     {
00066         $isValid = true;
00067         if ($this->checkIfEmailAlreadyTaken()) {
00068             $_SESSION['error'] = "Email already used";
00069             $isValid = false;
00070         }
00071         if ($this->checkifUsernameAlreadyTaken()) {
00072             $_SESSION['error'] = "Username alredy used";
00073             $isValid = false;
00074         }
00075         if ($isValid) {
00076 
00077 
00078             try {
00079                 $this->psInsert->execute(array(':insert_username' => $this->insert_username,
      ':insert_password' => password_hash($this->insert_password, PASSWORD_DEFAULT), ':insert_email' =>
      $this->insert_email));
00080                 $_SESSION['error'] = "Acount created";
00081             } catch (PDOException $e) {
00082                 print "Erreur !: " . $e->getMessage() . "<br>";
00083                 die();
00084             }
00085         }
00086     }
```

Here is the call graph for this function:

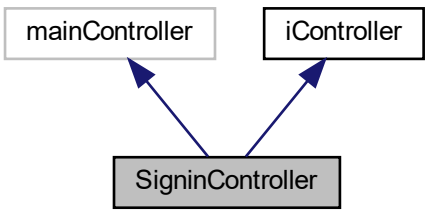


The documentation for this class was generated from the following file:

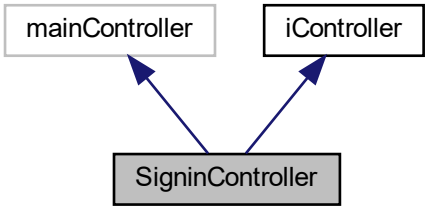
- models/signin.php

4.15 SigninController Class Reference

Inheritance diagram for SigninController:



Collaboration diagram for SigninController:



Public Member Functions

- **formHandler** ()
- **printHTML** ()

Data Fields

- **$signin**

4.15.1 Detailed Description

Definition at line 9 of file signinController.php.

The documentation for this class was generated from the following file:

- controllers/signinController.php

4.16 User Class Reference

Public Member Functions

- __construct (string $usernamep, string $emailp, string $idRolep, int $idUserp)
- updatePassword (string $newPassword, string $newPasswordRepeat, string $oldPassword)
- updatePrivateAccount ()
- getPrivateAccount ()

Data Fields

- **$username**
- **$email**
- **$role**
- **$idUser**

4.16.1 Detailed Description

Definition at line 10 of file user.php.

4.16.2 Constructor & Destructor Documentation

4.16.2.1 __construct()

```
__construct (
             string $usernamep,
             string $emailp,
             string $idRolep,
             int $idUserp )
```

default constructor

Parameters

string	*$usernamep*	
string	*$emailp*	
string	*$idRolep*	
integer	*$idUserp*	

Definition at line 25 of file user.php.

```
00026     {
00027
00028         $this->username = $usernamep;
00029         $this->email = $emailp;
00030         $this->role = $idRolep;
00031         $this->idUser = $idUserp;
00032     }
```

4.16.3 Member Function Documentation

4.16.3.1 getPrivateAccount()

```
getPrivateAccount ( )
```

use to know if the account is privat or not

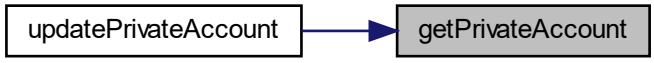
Returns

void

Definition at line 106 of file user.php.

```
00107     {
00108         $dbh = new PDO('mysql:host=' . HOST . ';dbname=' . DBNAME, USER, PASSWORD, array(
00109             PDO::MYSQL_ATTR_INIT_COMMAND => "SET NAMES utf8",
00110             PDO::ATTR_PERSISTENT => true
00111         ));
00112 
00113         try {
00114             $sqlGetPrivateAccount = "SELECT privateAccount FROM user WHERE id = :id_user";
00115             $psGetPrivateAccount = $dbh->prepare($sqlGetPrivateAccount);
00116             $psGetPrivateAccount->setFetchMode(PDO::FETCH_ASSOC);
00117             $psGetPrivateAccount->execute(array(':id_user' => $this->idUser));
00118             $result = $psGetPrivateAccount->fetchAll();
00119         } catch (PDOException $e) {
00120             print "Erreur !: " . $e->getMessage() . "<br>";
00121             die();
00122         }
00123 
00124         return $result[0]['privateAccount'];
00125     }
```

Here is the caller graph for this function:



4.16.3.2 updatePassword()

```
updatePassword (
             string $newPassword,
             string $newPasswordRepeat,
             string $oldPassword )
```

update the user password in the database

Parameters

string	*$newPassword*	
string	*$newPasswordRepeat*	
string	*$oldPassword*	

Returns

void

Definition at line 42 of file user.php.

```
00043     {
00044         $dbh = new PDO('mysql:host=' . HOST . ';dbname=' . DBNAME, USER, PASSWORD, array(
00045             PDO::MYSQL_ATTR_INIT_COMMAND => "SET NAMES utf8",
00046             PDO::ATTR_PERSISTENT => true
00047         ));
00048 
00049         $hasBeenUpdated = 1;
00050         if (password_verify($oldPassword, $this->getUserPassword())) {
00051 
00052             if ($newPasswordRepeat == $newPassword) {
00053                 try {
00054                     $sqlUpdatePassword = "UPDATE user  SET password = :update_password WHERE id =
      :id_user";
00055                     $psUpdatePassword = $dbh->prepare($sqlUpdatePassword);
00056                     $psUpdatePassword->execute(array(':update_password' => password_hash($newPassword,
      PASSWORD_DEFAULT), ':id_user' => $this->idUser));
00057 
00058                     $hasBeenUpdated = 0;
00059                 } catch (PDOException $e) {
00060                     print "Erreur !: " . $e->getMessage() . "<br>";
00061                     die();
00062                 }
00063             } else {
00064                 $hasBeenUpdated = 2;
00065             }
00066         } else {
00067             $hasBeenUpdated = 4;
00068         }
00069 
00070         return $hasBeenUpdated;
00071     }
```

4.16.3.3 updatePrivateAccount()

```
updatePrivateAccount ( )
```

update if the account is private or not

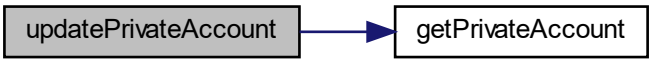
Returns

void

Definition at line 77 of file user.php.

```
00078     {
00079         $dbh = new PDO('mysql:host=' . HOST . ';dbname=' . DBNAME, USER, PASSWORD, array(
00080             PDO::MYSQL_ATTR_INIT_COMMAND => "SET NAMES utf8",
00081             PDO::ATTR_PERSISTENT => true
00082         ));
00083 
00084         $userisPrivate = $this->getPrivateAccount();
00085 
00086         if ($userisPrivate == 0) {
00087             $userSetPrivateTo = 1;
00088         } else {
00089             $userSetPrivateTo = 0;
00090         }
00091 
00092         try {
00093             $sqlUpdatePrivateAccount = "UPDATE user  SET privateAccount = :update_private_account
      WHERE id = :id_user";
00094             $psUpdatePrivateAccount = $dbh->prepare($sqlUpdatePrivateAccount);
00095             $psUpdatePrivateAccount->execute(array(':update_private_account' => $userSetPrivateTo,
      ':id_user' => $this->idUser));
00096         } catch (PDOException $e) {
00097             print "Erreur !: " . $e->getMessage() . "<br>";
00098             die();
00099         }
00100     }
```

Here is the call graph for this function:



The documentation for this class was generated from the following file:

- models/user.php

4.17 UserData Class Reference

Public Member Functions

- __construct ()
- getUsersByUsername ($username)
- getUserData ($iduser)

4.17.1 Detailed Description

Definition at line 10 of file userdata.php.

4.17.2 Constructor & Destructor Documentation

4.17.2.1 __construct()

```
__construct ( )
```

default constructor

Definition at line 21 of file userdata.php.

```
00022     {
00023         if ($this->dbh == null) {
00024             try {
00025                 $this->dbh = new PDO('mysql:host=' . HOST . ';dbname=' . DBNAME, USER, PASSWORD,
      array(
00026                     PDO::MYSQL_ATTR_INIT_COMMAND => "SET NAMES utf8",
00027                     PDO::ATTR_PERSISTENT => true
00028                 ));
00029 
00030                 // get list of user by username
00031                 $sqlRequestUsers = "SELECT * FROM user WHERE username LIKE :search_username AND
      privateAccount = 0";
00032                 $this->psGetUsersByUsername = $this->dbh->prepare($sqlRequestUsers);
00033                 $this->psGetUsersByUsername->setFetchMode(PDO::FETCH_ASSOC);
00034 
00035                 // get list of user by username
00036                 $sqlGetDataUser = "SELECT * FROM user WHERE id =:search_idUser";
00037                 $this->psGetDataUser = $this->dbh->prepare($sqlGetDataUser);
00038                 $this->psGetDataUser->setFetchMode(PDO::FETCH_ASSOC);
00039             } catch (PDOException $e) {
00040                 print "Erreur !: " . $e->getMessage() . "<br>";
00041                 die();
00042             }
00043         }
00044     }
```

4.17.3 Member Function Documentation

4.17.3.1 getUserData()

```
getUserData (
              $iduser )
```

get data of a specific user

Parameters

[type]	$iduser

Returns

void

Definition at line 68 of file userdata.php.

```
00069     {
00070         try {
00071             $this->psGetDataUser->execute(array(':search_idUser' => $iduser));
00072             $result = $this->psGetDataUser->fetchAll();
00073         } catch (PDOException $e) {
00074             print "Erreur !: " . $e->getMessage() . "<br>";
00075             die();
00076         }
00077         return $result;
00078     }
```

4.17.3.2 getUsersByUsername()

```
getUsersByUsername (
              $username )
```

get users by their username

Parameters

[type]	$username

Returns

void

Definition at line 51 of file userdata.php.

```
00052     {
00053         try {
00054             $this->psGetUsersByUsername->execute(array(':search_username' => '%' . $username . '%'));
00055             $result = $this->psGetUsersByUsername->fetchAll();
00056         } catch (PDOException $e) {
```

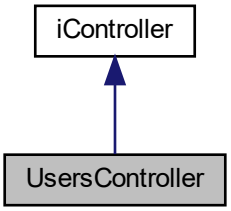
```
00057             print "Erreur !: " . $e->getMessage() . "<br>";
00058             die();
00059         }
00060         return $result;
00061     }
```

The documentation for this class was generated from the following file:

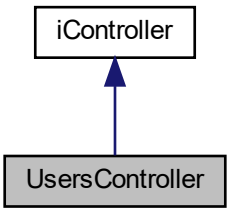
- models/userdata.php

4.18 UsersController Class Reference

Inheritance diagram for UsersController:



Collaboration diagram for UsersController:



Public Member Functions

- **formHandler** ()
- **errorHandler** ()
- **printHTML** ()
- **htmlrecherchUsers** ()
- **htmlrequestUser** ()
- **htmlDetailUser** ()

Data Fields

- **$userData**

4.18.1 Detailed Description

Definition at line 10 of file usersController.php.

The documentation for this class was generated from the following file:

- controllers/usersController.php

Chapter 5

File Documentation

5.1 controllers/administratorController.php File Reference

Class used to handle request for the administrator page.

Data Structures

- class AdministratorController

5.1.1 Detailed Description

Class used to handle request for the administrator page.

5.1.1.1 BDCC

Author

Lorenzo Bauduccio lorenzo.bdcc@eduge.ch

Copyright

Copyright (c) 2021 BDCC

Definition in file administratorController.php.

5.2 administratorController.php

```
00001 <?php
00002
00010 class AdministratorController extends mainController implements iController
00011 {
00012   public $administrator;
00013   public $game;
00014   public $categorie;
00015   private $e = null;
00016   private $idGameToUpdate;
00017
00023   public function formHandler()
00024   {
00025     $result = null;
00026
00027
00028     if (isset($_GET['e'])) {
00029       $this->e = filter_input(INPUT_GET, 'e', FILTER_SANITIZE_STRING);
00030       //redirige l'utilisateur qui n'a pas les bon droits
00031       $this->allowAccessTo(array(1));
00032     }
00033     // update game
00034     if ($this->e == "updateGame") {
00035       if (isset($_GET['id'])) {
00036         $requestGame = filter_input(INPUT_GET, 'id', FILTER_SANITIZE_STRING);
00037         $this->idGameToUpdate = $requestGame;
00038       } else {
00039         header('Location:' . $_SERVER['HTTP_REFERER']);
00040         exit;
00041       }
00042     }
00043
00044     // add game categorie
00045     if ($this->e == "addGameCategorie") {
00046       if (isset($_GET['id'])) {
00047         $requestGame = filter_input(INPUT_GET, 'id', FILTER_SANITIZE_STRING);
00048         $this->idGameToUpdate = $requestGame;
00049       } else {
00050         header('Location:' . $_SERVER['HTTP_REFERER']);
00051         exit;
00052       }
00053     }
00054
00055     // add categorie to game
00056     if ($this->e == "addGameCategorieAdd") {
00057       if (isset($_GET['idGame'])) {
00058         $idGame = filter_input(INPUT_GET, 'idGame', FILTER_SANITIZE_STRING);
00059       } else {
00060         header('Location:' . $_SERVER['HTTP_REFERER']);
00061         exit;
00062       }
00063       if (isset($_GET['idCategorie'])) {
00064         $idCategorie = filter_input(INPUT_GET, 'idCategorie', FILTER_SANITIZE_STRING);
00065       } else {
00066         header('Location:' . $_SERVER['HTTP_REFERER']);
00067         exit;
00068       }
00069
00070       $this->categorie->addCategorieToGame($idGame, $idCategorie);
00071       header('Location:' . $_SERVER['HTTP_REFERER']);
00072       exit;
00073     }
00074
00075     // delete categorie from a game
00076     if ($this->e == "delGameCategorie") {
00077       if (isset($_GET['idGame'])) {
00078         $idGame = filter_input(INPUT_GET, 'idGame', FILTER_SANITIZE_STRING);
00079       } else {
00080         header('Location:' . $_SERVER['HTTP_REFERER']);
00081         exit;
00082       }
00083       if (isset($_GET['idCategorie'])) {
00084         $idCategorie = filter_input(INPUT_GET, 'idCategorie', FILTER_SANITIZE_STRING);
00085       } else {
00086         header('Location:' . $_SERVER['HTTP_REFERER']);
00087         exit;
00088       }
00089
00090       $this->categorie->delCategorieFromGame($idGame, $idCategorie);
00091       header('Location:' . $_SERVER['HTTP_REFERER']);
00092       exit;
00093     }
00094
00095     //add game
00096     if ($this->e == "addGameUpload") {
00097       if (isset($_POST['name'])) {
```

```
00098         $name = filter_input(INPUT_POST, 'name', FILTER_SANITIZE_STRING);
00099       } else {
00100         header('Location:' . $_SERVER['HTTP_REFERER']);
00101         exit;
00102       }
00103
00104       if (isset($_POST['description'])) {
00105         $description = filter_input(INPUT_POST, 'description', FILTER_SANITIZE_STRING);
00106       } else {
00107         header('Location:' . $_SERVER['HTTP_REFERER']);
00108         exit;
00109       }
00110
00111       if (isset($_POST['imageName'])) {
00112         $imageName = filter_input(INPUT_POST, 'imageName', FILTER_SANITIZE_STRING);
00113       } else {
00114         header('Location:' . $_SERVER['HTTP_REFERER']);
00115         exit;
00116       }
00117
00118       if (isset($_POST['console'])) {
00119         $consoleId = filter_input(INPUT_POST, 'console', FILTER_SANITIZE_STRING);
00120       } else {
00121         header('Location:' . $_SERVER['HTTP_REFERER']);
00122         exit;
00123       }
00124
00125       if (isset($_POST['gameFileName'])) {
00126         $gameFileName = filter_input(INPUT_POST, 'gameFileName', FILTER_SANITIZE_STRING);
00127       } else {
00128         header('Location:' . $_SERVER['HTTP_REFERER']);
00129         exit;
00130       }
00131
00132
00133       $this->administrator->addGame($name, $description, $imageName, $consoleId, $gameFileName);
00134
00135       header('Location:' . $_SERVER['HTTP_REFERER']);
00136       exit;
00137     }
00138
00139     //add game
00140     if ($this->e == "updateGameUpdate") {
00141       if (isset($_POST['name'])) {
00142         $name = filter_input(INPUT_POST, 'name', FILTER_SANITIZE_STRING);
00143       } else {
00144         header('Location:' . $_SERVER['HTTP_REFERER']);
00145         exit;
00146       }
00147
00148       if (isset($_POST['description'])) {
00149         $description = filter_input(INPUT_POST, 'description', FILTER_SANITIZE_STRING);
00150       } else {
00151         header('Location:' . $_SERVER['HTTP_REFERER']);
00152         exit;
00153       }
00154
00155       if (isset($_POST['console'])) {
00156         $consoleId = filter_input(INPUT_POST, 'console', FILTER_SANITIZE_STRING);
00157       } else {
00158         header('Location:' . $_SERVER['HTTP_REFERER']);
00159         exit;
00160       }
00161
00162       if (isset($_POST['idGame'])) {
00163         $idGame = filter_input(INPUT_POST, 'idGame', FILTER_SANITIZE_STRING);
00164       } else {
00165         header('Location:' . $_SERVER['HTTP_REFERER']);
00166         exit;
00167       }
00168
00169
00170       $this->administrator->updateGame($idGame, $name, $description, $consoleId);
00171
00172       header('Location:' . $_SERVER['HTTP_REFERER']);
00173       exit;
00174     }
00175
00176
00177     if ($this->e == "addCategorie") {
00178       if (isset($_POST['name'])) {
00179         $name = filter_input(INPUT_POST, 'name', FILTER_SANITIZE_STRING);
00180       }
00181
00182       if (isset($name)) {
00183         $result = $this->categorie->addCategorie($name);
00184       }
```

```
00185     }
00186   }
00187
00188
00192   public function __construct()
00193   {
00194     $this->administrator  = new Administrator();
00195     $this->game = new Games();
00196     $this->categorie = new Categories();
00197   }
00198
00205   public function printHTML()
00206   {
00207
00208     $html = '<main style="margin-top:20px">
00209         <div class="container-md">';
00210     $html .= $this->errorHandler();
00211     if ($this->e == null) {
00212       $html .= $this->htmlAdministratorHome();
00213     }
00214
00215     if ($this->e == "addGame") {
00216       $html .= $this->htmlNewGame();
00217     }
00218
00219     if ($this->e == "updateGame") {
00220       $html .= $this->htmlUpdateGame();
00221     }
00222
00223     if ($this->e == "addCategorie") {
00224       $html .= $this->htmlAddCategorie();
00225     }
00226
00227     if ($this->e == "addGameCategorie") {
00228       $html .= $this->htmlAddCategorieToGame();
00229     }
00230
00231
00232
00233     $html .= "</div></main> ";
00234     echo $html;
00235   }
00236
00242   private function htmlAdministratorHome()
00243   {
00244
00245     $html = "";
00246
00247     $html .= '<div class=" jumbotron DarkJumbotron width100" style="background-color: #161b22;">';
00248
00249     $html .= '<div class="container">
00250             <div class="row"><h2>Administrator pannel</h2></div>
00251             <div class="row">
00252               <div class="col-sm">
00253                <p>To update a game you need to go on his own page and click on the update button</p>
00254               </div>
00255                 <div class="col-sm">
00256                     <div style="width: 70%; margin: auto;">
00257                         <a href="?r=administrator&e=addGame" class="btn btn-warning my-2">Add new
      game</a>
00258                         <a href="?r=administrator&e=addCategorie" class="btn btn-warning my-2">Add new
      catégorie</a>
00259                     </div>
00260                 </div>
00261           </div>';
00262
00263     $html .= '
00264
00265           </div>
00266         </div>';
00267
00268     return $html;
00269   }
00270
00276   private function htmlNewGame()
00277   {
00278     $html = "";
00279
00280     $html .= '<div class="jumbotron jumbotron-fluid DarkJumbotron width100" style="background-color:
      #161b22;"">
00281         <div class="container">';
00282
00283     $html .= '<div class="container">
00284         <div class="row">
00285           <div class="col-sm">
00286           <div style="width: 70%; margin: auto;">
00287         <h2>Add game</h2>
```

```
00288         <form action="?r=administrator&e=addGameUpload" method="post" enctype="multipart/form-data">
00289               <div class="form-group">
00290                 <label for="name">Game\'s name</label>
00291                 <input type="texte" name="name" class="form-control" id="name" aria-describedby="name"
      placeholder="Enter name">
00292               </div>
00293               <div class="form-group">
00294                 <label for="description">Game\'s description</label>
00295                 <textarea class="form-control" id="description" name="description"
      rows="5"></textarea>
00296               </div>
00297               <div class="form-group">
00298                 <label for="ImageName">Game\'s image name</label>
00299                 <p>Example: Super Mario Sunshine -> SUPER_MARIO_SUNSHINE.png </p>
00300                 <input type="texte" name="imageName" class="form-control" id="imageName"
      aria-describedby="imageName" placeholder="Enter image name">
00301               </div>
00302               <div class="form-group">
00303                 <label for="file">Image File</label>
00304                 <input type="file" class="form-control-file" id="image" name="image">
00305               </div>
00306               <div class="form-group">
00307                 <label for="file">Console</label>
00308                 <select class="form-control" name="console">
00309                     ';
00310    $html .= $this->htmlgetListConsole();
00311    $html .= '
00312                 </select>
00313               </div>
00314               <div class="form-group">
00315                 <label for="gameFileName">Game\'s file name</label>
00316                 <p>Example: Super Mario Sunshine -> SUPER_MARIO_SUNSHINE.iso </p>
00317                 <input type="texte" name="gameFileName" class="form-control" id="gameFileName"
      aria-describedby="gameFileName" placeholder="Enter file name">
00318               </div>
00319               <div class="form-group">
00320                 <label for="file">Game file</label>
00321                 <input type="file" class="form-control-file" id="fileGame" name="fileGame">
00322               </div>
00323
00324               <div class="form-group">
00325               <button type="submit" class="btn btn-success">Add Game</button>
00326           </div>
00327         </form>
00328      </div>
00329         </div>
00330
00331       </div>
00332     </div>';
00333
00334    $html .= '
00335
00336       </div>
00337     </div>';
00338
00339    return $html;
00340  }
00341
00347  private function htmlAddCategorieToGame()
00348  {
00349    $gameData = $this->game->getGameDetail($this->idGameToUpdate);
00350    $gameCategorie = $this->categorie->getCategoriesOfGame($this->idGameToUpdate);
00351    $allCategorie = $this->categorie->getListAllCategories();
00352    $html = "";
00353
00354    $html .= '<div class="jumbotron jumbotron-fluid DarkJumbotron width100" style="background-color:
      #161b22;">';
00355
00356    $html .= '<div class="container">
00357        <div class="row">
00358            <div style="width: 70%; margin: auto;">
00359              <h2>Update ' . $gameData[0]['name'] . '</h2>
00360            </div>
00361        </div>
00362
00363               <div class="row">
00364               <div class="col-sm">
00365                 <div class="form-group">
00366                   <h3 class="card-title centerText">Actual game\'s categories</h3>
00367
00368
00369                     <div class="list-group">';
00370
00371    foreach ($gameCategorie as $key => $cat) {
00372      $html .= '<a class="btn btn-outline-danger btnCategorie margintop10"
      href="?r=administrator&e=delGameCategorie&idCategorie=' . $cat['id'] . '&idGame=' .
      $gameData[0]['id'] . '" role="button">DELETE: ' . $cat['name'] . '</a>';
```

```
00373     }
00374
00375     $html .= '      </div>
00376                   </div>
00377                 </div>
00378
00379
00380               <div class="col-sm">
00381                 <div class="form-group">
00382                   <h3 class="card-title centerText">Add categories</h3>
00383                     <div class="list-group">';
00384
00385     foreach ($allCategorie as $key => $cat) {
00386       $html .= '<a class="btn btn-outline-success btnCategorie margintop10 "
      href="?r=administrator&e=addGameCategorieAdd&idCategorie=' . $cat['id'] . '&idGame=' .
      $gameData[0]['id'] . '" role="button">ADD: ' . $cat['name'] . '</a>';
00387     }
00388
00389     $html .= '  </div>
00390               </div>
00391       </div>';
00392
00393     $html .= '
00394
00395         </div>
00396       </div>';
00397
00398     return $html;
00399   }
00400
00406   private function htmlUpdateGame()
00407   {
00408     $gameData = $this->game->getGameDetail($this->idGameToUpdate);
00409     $html = "";
00410
00411     $html .= '<div class="jumbotron jumbotron-fluid DarkJumbotron width100" style="background-color:
      #161b22;"">
00412         <div class="container">';
00413
00414     $html .= '<div class="container">
00415         <div class="row">
00416           <div class="col-sm">
00417           <div style="width: 70%; margin: auto;">
00418         <h2>Update game</h2>
00419         <form action="?r=administrator&e=updateGameUpdate" method="post"
      enctype="multipart/form-data">
00420         <input id="idGame" name="idGame" type="hidden" value="' . $gameData[0]['id'] . '">
00421               <div class="form-group">
00422                 <label for="name">Game\'s name</label>
00423                 <input type="texte" name="name" class="form-control" id="name" aria-describedby="name"
      placeholder="Enter name" value="' . $gameData[0]['name'] . '">
00424               </div>
00425               <div class="form-group">
00426                 <label for="description">Game\'s description</label>
00427                 <textarea class="form-control" id="description" name="description" rows="5" >' .
      $gameData[0]['description'] . '</textarea>
00428               </div>
00429               <div class="form-group">
00430                 <label for="file">Console</label>
00431                 <select class="form-control" name="console">
00432                     ';
00433     $html .= $this->htmlgetListConsole();
00434     $html .= '
00435                 </select>
00436               </div>
00437
00438               <div class="form-group">
00439               <button type="submit" class="btn btn-success">Update Game</button>
00440             </div>
00441           </form>
00442       </div>
00443           </div>
00444
00445         </div>
00446       </div>';
00447
00448     $html .= '
00449
00450         </div>
00451       </div>';
00452
00453     return $html;
00454   }
00455
00461   private function htmlgetListConsole()
00462   {
00463     $html = "";
```

```
00464
00465     foreach ($this->administrator->getListConsole() as $key => $console) {
00466       $html .= '<option value="' . $console['id'] . '">' . $console['name'] . '</option>';
00467     }
00468
00469     return $html;
00470   }
00471
00477   private function htmlAddCategorie()
00478   {
00479
00480     $html = "";
00481
00482     $html .= '<div class="jumbotron jumbotron-fluid DarkJumbotron width100" style="background-color:
      #161b22;"">
00483         <div class="container">';
00484
00485     $html .= '<div class="container">
00486         <div class="row">
00487           <div class="col-sm">
00488           <div style="width: 70%; margin: auto;">
00489         <h2>Add categorie</h2>
00490         <form action="?r=administrator&e=addCategorie" method="post">
00491               <div class="form-group">
00492                 <label for="username">Categorie\'s name</label>
00493                 <input type="texte" name="name" class="form-control" id="name" aria-describedby="name"
      placeholder="Enter name">
00494               </div>
00495
00496               <div class="form-group">
00497               <button type="submit" class="btn btn-success">Add</button>
00498             </div>
00499           </form>
00500       </div>
00501           </div>
00502           <div class="col-sm">
00503
00504           </div>
00505         </div>
00506       </div>';
00507
00508     $html .= '
00509
00510         </div>
00511       </div>';
00512
00513     return $html;
00514   }
00515 }
```

5.3 controllers/controllers.php File Reference

file used to include all the controller of the project

5.3.1 Detailed Description

file used to include all the controller of the project

5.3.1.1 BDCC

Author

Lorenzo Bauduccio lorenzo.bdcc@eduge.ch

Copyright

Copyright (c) 2021 BDCC

Definition in file controllers.php.

5.4 controllers.php

```
00001 <?php
00009 include_once "mainController.php";
00010 include_once "interfaceController.php";
00011 include_once "dashboardController.php";
00012 include_once "downloadController.php";
00013 include_once "gamesController.php";
00014 include_once "indexController.php";
00015 include_once "loginController.php";
00016 include_once "signinController.php";
00017 include_once "administratorController.php";
00018 include_once "usersController.php";
00019
00020 ?>
```

5.5 controllers/dashboardController.php File Reference

Class used to handle request for the dashboard of the user.

Data Structures

- class DashboardController

5.5.1 Detailed Description

Class used to handle request for the dashboard of the user.

5.5.1.1 BDCC

Author

Lorenzo Bauduccio lorenzo.bdcc@eduge.ch

Copyright

Copyright (c) 2021 BDCC

Definition in file dashboardController.php.

5.6 dashboardController.php

```
00001 <?php
00009 include_once "./models/class.php";
00010 class DashboardController extends mainController implements iController
00011 {
00012     public $game;
00013     private $e = null;
00014
00015
00016     public function formHandler()
00017     {
00018         $this->allowAccessTo(array(1,3));
00019
00020         $oldPassword = null;
00021         $newPasswordRepeat = null;
00022         $newPassword = null;
00023
```

```
00024         if (isset($_GET['e'])) {
00025             $this->e = filter_input(INPUT_GET, 'e', FILTER_SANITIZE_SPECIAL_CHARS);
00026         }
00027         // form update
00028         if ($this->e == "updatePassword") {
00029             if (isset($_POST['oldPassword'])) {
00030                 $oldPassword = filter_input(INPUT_POST, 'oldPassword', FILTER_SANITIZE_STRING);
00031             }
00032             if (isset($_POST['newPassword'])) {
00033                 $newPassword = filter_input(INPUT_POST, 'newPassword', FILTER_SANITIZE_STRING);
00034             }
00035             if (isset($_POST['newPasswordRepeat'])) {
00036                 $newPasswordRepeat = filter_input(INPUT_POST, 'newPasswordRepeat',
      FILTER_SANITIZE_STRING);
00037             }
00038 
00039             if (isset($oldPassword) && isset( $newPassword) && isset( $newPasswordRepeat)) {
00040                 $_SESSION['user']->updatePassword($newPassword,$newPasswordRepeat,$oldPassword);
00041             }
00042         }
00043 
00044         // update if account if visible or not
00045         if ($this->e == "updatePrivateAccount") {
00046 
00047             if ($_SESSION['user']->idUser != -1) {
00048                 $_SESSION['user']->updatePrivateAccount();
00049                 header('Location:'.$_SERVER['HTTP_REFERER']);
00050             }
00051         }
00052     }
00053 
00054     public function __construct()
00055     {
00056         $this->game = new Games();
00057     }
00058 
00059 
00060     public function printHTML()
00061     {
00062 
00063         $html = '<main  style="margin-top:20px ">
00064         <div class="container-md">';
00065         echo $_SESSION['error'];
00066         $html .= $this->errorHandler();
00067 
00068         if ($this->e == null) {
00069             $html .= $this->htmlFormHead();
00070             $html .= $this->htmlFavoriteGames();
00071             $html .= $this->htmlGameTime();
00072         }
00073 
00074         if ($this->e == "updatePassword") {
00075             $html .= $this->htmlFormUpdatePassword();
00076         }
00077 
00078         $html .= "</div></main> ";
00079 
00080         echo $html;
00081     }
00082 
00083     private function htmlFormHead()
00084     {
00085         $html = "";
00086 
00087         $html .= '<div class=" jumbotron DarkJumbotron width100" style="background-color: #161b22;">';
00088 
00089         $html .= '<div class="container">
00090         <div class="row"><h2>User\'s Informations</h2></div>
00091         <div class="row">
00092           <div class="col-sm">
00093             <ul class="list-group">';
00094                 $html .= '<li class="list-group-item">Username: '.$_SESSION['user']->username.'</li>';
00095                 $html .= '<li class="list-group-item">Email: '.$_SESSION['user']->email.'</li>';
00096                 if ($_SESSION['user']->getPrivateAccount() == 1) {
00097 
00098                 $html .= '<li class="list-group-item">Your account is not visible for other
      users</li>';
00099                 }else {
00100                     $html .= '<li class="list-group-item">Your account is visible for other
      users</li>';
00101                 }
00102                 $html .='
00103             </ul>
00104           </div>
00105             <div class="col-sm">
00106             <div style="width: 70%; margin: auto;">
00107             <a href="?r=dashboard&e=updatePassword" class="btn btn-warning my-2">Update password</a>
```

```
00108             <a href="?r=dashboard&e=updatePrivateAccount" class="btn btn-warning my-2">Update if
      account is private</a>
00109          </div>
00110            </div>
00111        </div>';
00112
00113      $html.='
00114
00115        </div>
00116      </div>';
00117
00118      return $html;
00119    }
00120
00121    private function htmlFavoriteGames()
00122    {
00123        $html = '<div class="d-inline-flex  jumbotron DarkJumbotron  width100"
      style="background-color: #161b22;" >
00124        <div class="container">
00125        <div class="row"><h2>User\'s favorites games</h2></div>
00126        <div class="row cardGameBox box">
00127        ';
00128
00129        $listGamesBrut =  $this->game->getFavoriteGamesOfUser($_SESSION['user']->idUser);
00130
00131        foreach ($listGamesBrut as $key => $games) {
00132
00133
00134            $html .= $this->createCardHTML($games);
00135        }
00136
00137        $html .= '</div>
00138        </div>
00139        </div>';
00140        return $html;
00141    }
00142
00143    private function htmlGameTime()
00144    {
00145        $html = '<div class="d-inline-flex  jumbotron DarkJumbotron  width100"
      style="background-color: #161b22;" >
00146        <div class="container">
00147        <div class="row"><h2>User\'s time in games</h2></div>
00148        <div class="row cardGameBox box">
00149        ';
00150
00151        $listGamesBrut =  $this->game->getListOfGameWithTimeUser($_SESSION['user']->idUser);
00152        foreach ($listGamesBrut as $key => $games) {
00153
00154
00155            $html .= $this->createCardHTMLTime($games['idGame']);
00156        }
00157
00158        $html .= '</div>
00159        </div>
00160        </div>';
00161        return $html;
00162    }
00163
00164
00165
00166
00167    private function createCardHTML($game)
00168    {
00169        $html = '
00170        <div class="card cardBootstarp" style=" max-width: 15rem;  margin:10px; padding:0;
      background-color: #161b22; border:2px solid #28a745;">
00171        <a href="?r=games&e=detail&idGame=' . $game['id'] . '">
00172        <img src="./img/games/' . $game['imageName'] . '." class="card-img-top imageCard"  >
00173        <div class="card-body ">
00174        <h6 class="card-title whiteTexte">' . $game['name'] . '</h5>
00175        <div class="card-body ">
00176            ';
00177
00178                    $html.= '<a class="btn btn-outline-success cardContent"
      href="?r=games&e=removeFavoris&idGame=' . $game['id'] . '" role="button"><i class="fa fa-heart
      "></i></a>';
00179
00180
00181            $html .= '
00182        </div>
00183        </div>
00184        </a>
00185        </div>';
00186        return $html;
00187    }
00188
```

```
00189     private function createCardHTMLTime($game)
00190     {
00191         $gameDetail = $this->game->getGameDetail($game);
00192         $gameDetail = $gameDetail[0];
00193         $gameTime = $this->game->getTimeInGameUser($_SESSION['user']->idUser, $game);
00194         $html = '
00195         <div class="card cardBootstarp" style=" max-width: 15rem; margin:10px; padding:0;
      background-color: #161b22; border:2px solid #28a745;">
00196         <img src="./img/games/' . $gameDetail['imageName'] . '." class="card-img-top imageCard"  >
00197         <div class="card-body ">
00198         <h6 class="card-title whiteTexte">' . $gameDetail['name'] . '</h5>
00199             <p class="greenTexte">';
00200             $heure = (int)($gameTime[0]['timeInMinute'] / 60 );
00201             $minutes = ( $gameTime[0]['timeInMinute'] % 60);
00202             if ($minutes == 60) {
00203                 $heure ++;
00204                 $minutes = 0;
00205             }
00206             $html .= $heure. "h".$minutes. " minutes";
00207             $html .=' </p>
00208         </div>
00209         </div>';
00210         return $html;
00211     }
00212
00213     public function htmlFormUpdatePassword()
00214     {
00215         $html = '<div class="d-inline-flex p-2 jumbotron  width100 DarkJumbotron "
      style="background-color: #161b22;" >
00216         <div class="container">
00217         <div class="row"><h2>Update your password</h2></div>
00218         <div class="row">
00219
00220
00221         <form action="?r=dashboard&e=updatePassword" method="post">
00222             <div class="form-group">
00223                 <label for="oldPassword">Old password</label>
00224                 <input type="password" class="form-control" id="oldPassword" name="oldPassword"
      placeholder="Old password">
00225             </div>
00226             <div class="form-group">
00227                 <label for="newPassword">Password</label>
00228                 <input type="password" class="form-control" id="newPassword" name="newPassword"
      placeholder="New password">
00229             </div>
00230             <div class="form-group">
00231                 <label for="newPasswordRepeat">Password</label>
00232                 <input type="password" class="form-control" id="newPasswordRepeat"
      name="newPasswordRepeat" placeholder="New password repeat">
00233             </div>
00234
00235             <button type="submit" class="btn btn-primary">Submit</button>
00236         </form>
00237         </div>
00238                 </div>
00239                 </div>';
00240                 return $html;
00241     }
00242 }
```

5.7 models/administrator.php File Reference

Class used to handle request for the administrator.

Data Structures

- class Administrator

5.7.1 Detailed Description

Class used to handle request for the administrator.

5.7.1.1 BDCC

Author

Lorenzo Bauduccio lorenzo.bdcc@eduge.ch

Copyright

Definition in file administrator.php.

5.8 administrator.php

```
00001 <?php
00009 class Administrator
00010 {
00011
00012     private $dbh = null;
00013
00014     private $psLogin = null;
00015
00016     public $search_username = null;
00017
00018     public $search_password = null;
00019
00020     public $arrayInfo = null;
00021
00022     public $psUploadGame = null;
00023
00024     public $psUploadFile = null;
00025
00026     public $psUpdateGame = null;
00027
00032     public function __construct()
00033     {
00034         if ($this->dbh == null) {
00035             try {
00036                 $this->dbh = new PDO('mysql:host=' . HOST . ';dbname=' . DBNAME, USER, PASSWORD,
    array(
00037                     PDO::MYSQL_ATTR_INIT_COMMAND => "SET NAMES utf8",
00038                     PDO::ATTR_PERSISTENT => true
00039                 ));
00040                 // get list of console
00041                 $sqlGetListConsole = "SELECT * FROM consol";
00042                 $this->psGetListConsole = $this->dbh->prepare($sqlGetListConsole);
00043                 $this->psGetListConsole->setFetchMode(PDO::FETCH_ASSOC);
00044
00045                 // upload game
00046                 $sqlUploadGame = "INSERT INTO game  (name, description, imageName, idConsole, idFile)
00047                 VALUES (:insert_name, :insert_description, :insert_imageName, :insert_idConsole,
    :insert_idFile)";
00048                 $this->psUploadGame = $this->dbh->prepare($sqlUploadGame);
00049                 $this->psUploadGame->setFetchMode(PDO::FETCH_ASSOC);
00050
00051                 // upload file
00052                 $sqlUploadFile = "INSERT INTO file  (filename, dateUpdate)
00053                 VALUES (:insert_filename, NOW() )";
00054                 $this->psUploadFile = $this->dbh->prepare($sqlUploadFile);
00055                 $this->psUploadFile->setFetchMode(PDO::FETCH_ASSOC);
00056
00057                 // update game
00058                 $sqlUpdateGane = "UPDATE game  SET name = :update_name, description =
    :update_description, idConsole = :update_idConsole WHERE id = :update_id";
00059                 $this->psUpdateGame = $this->dbh->prepare($sqlUpdateGane);
00060                 $this->psUpdateGame->setFetchMode(PDO::FETCH_ASSOC);
00061
00062                 // get folder name of console
00063                 $sqlGetNameConsoleFolder = "SELECT folderName FROM consol WHERE id = :console_id";
00064                 $this->psGetNameConsoleFolder = $this->dbh->prepare($sqlGetNameConsoleFolder);
00065                 $this->psGetNameConsoleFolder->setFetchMode(PDO::FETCH_ASSOC);
00066             } catch (PDOException $e) {
00067                 print "Erreur !: " . $e->getMessage() . "<br>";
00068                 die();
00069             }
00070         }
```

```
00071     }
00082     public function addGame(string $name, string $description, string $imageName, int $consoleId,
      $gameFileName)
00083     {
00084 
00085         if ($this->uploadGame($gameFileName, $consoleId) && $this->uploadGameImage($imageName)) {
00086             try {
00087                 $this->psUploadFile->execute(array(':insert_filename' => $gameFileName));
00088             } catch (PDOException $e) {
00089                 print "Erreur !: " . $e->getMessage() . "<br>";
00090                 die();
00091             }
00092             $lastInsertId = $this->dbh->lastInsertId();
00093             try {
00094                 $this->psUploadGame->execute(array(':insert_name' => $name, ':insert_description' =>
      $description, ':insert_imageName' => $imageName, ':insert_idConsole' => $consoleId, ':insert_idFile'
      => $lastInsertId));
00095             } catch (PDOException $e) {
00096                 print "Erreur !: " . $e->getMessage() . "<br>";
00097                 die();
00098             }
00099         }
00100     }
00108     public function uploadGame($gameFileName, $consoleId)
00109     {
00110         $uploadIsValid = false;
00111         $target_dir = "../games/" . $this->getConsoleFolderName($consoleId) . "/";
00112 
00113         $target_file =  basename($_FILES["fileGame"]["name"]);
00114         $uploadOk = 1;
00115         $fileType = strtolower(pathinfo($target_file, PATHINFO_EXTENSION));
00116 
00117         //rename file
00118         $newfilename = $gameFileName . '.' . $fileType;
00119 
00120 
00121         // Check if file already exists
00122         if (file_exists($target_file)) {
00123             echo "Sorry, file already exists.";
00124             $uploadOk = 0;
00125         }
00126         if ($uploadOk == 0) {
00127             echo "Sorry, your file was not uploaded.";
00128             // if everything is ok, try to upload file
00129         } else {
00130             if (move_uploaded_file($_FILES["fileGame"]["tmp_name"], $target_dir . $newfilename)) {
00131                 $uploadIsValid = true;
00132             } else {
00133                 //Sorry, there was an error uploading your file
00134             }
00135         }
00136         return $uploadIsValid;
00137     }
00147     public function updateGame($idGame, $name, $description, $consoleId)
00148     {
00149         try {
00150             $this->psUpdateGame->execute(array(':update_name' => $name, ':update_description' =>
      $description, ':update_idConsole' => $consoleId, ':update_id' => $idGame));
00151         } catch (PDOException $e) {
00152             print "Erreur !: " . $e->getMessage() . "<br>";
00153             die();
00154         }
00155     }
00162     public function uploadGameImage($imageFileName)
00163     {
00164         $uploadIsValid = false;
00165         $target_dir = "img/games/";
00166 
00167         $target_file =  basename($_FILES["image"]["name"]);
00168         $uploadOk = 1;
00169         $fileType = strtolower(pathinfo($target_file, PATHINFO_EXTENSION));
00170 
00171         //rename file
00172         $newfilename = $imageFileName;
00173 
00174         // Check if file already exists
00175         if (file_exists($target_file)) {
00176             echo "Sorry, file already exists.";
00177             $uploadOk = 0;
00178         }
00179         if ($uploadOk == 0) {
00180             echo "Sorry, your file was not uploaded.";
00181             // if everything is ok, try to upload file
00182         } else {
00183             if (move_uploaded_file($_FILES["image"]["tmp_name"], $target_dir . $newfilename)) {
00184                 $uploadIsValid = true;
00185             } else {
```

```
00186                 //Sorry, there was an error uploading your file
00187             }
00188         }
00189         return $uploadIsValid;
00190     }
00197     public function getConsoleFolderName($id)
00198     {
00199         $returnArray = null;
00200         try {
00201             $this->psGetNameConsoleFolder->execute(array(':console_id' => $id));
00202             $returnArray = $this->psGetNameConsoleFolder->fetchAll();
00203         } catch (PDOException $e) {
00204             print "Erreur !: " . $e->getMessage() . "<br>";
00205             die();
00206         }
00207         return $returnArray[0]['folderName'];
00208     }
00214     public function getListConsole()
00215     {
00216         $returnArray = null;
00217         try {
00218             $this->psGetListConsole->execute();
00219             $returnArray = $this->psGetListConsole->fetchAll();
00220         } catch (PDOException $e) {
00221             print "Erreur !: " . $e->getMessage() . "<br>";
00222             die();
00223         }
00224         return $returnArray;
00225     }
00226 }
```

5.9 models/categorie.php File Reference

Class used to handle request for the table categorie.

Data Structures

- class Categories

5.9.1 Detailed Description

Class used to handle request for the table categorie.

5.9.1.1 BDCC

Author

Lorenzo Bauduccio lorenzo.bdcc@eduge.ch

Copyright

Copyright (c) 2021 BDCC

Definition in file categorie.php.

5.10 categorie.php

```
00001 <?php
00002
00010 class Categories
00011 {
00012
00013     private $dbh = null;
00014
00015     private $psGetAllCategories = null;
00016
00017     private $psGameCategorie = null;
00018
00019     private $psAddCategorieToGame = null;
00020
00021     private $psCheckIfGameHasCategorie = null;
00022
00023     private $psDelCategorieFromGame = null;
00024
00025
00026
00027
00031     public function __construct()
00032     {
00033         if ($this->dbh == null) {
00034             try {
00035                 $this->dbh = new PDO('mysql:host=' . HOST . ';dbname=' . DBNAME, USER, PASSWORD,
      array(
00036                     PDO::MYSQL_ATTR_INIT_COMMAND => "SET NAMES utf8",
00037                     PDO::ATTR_PERSISTENT => true
00038                 ));
00039                 //get all categories
00040                 $sqlGetAllCategories = "SELECT * FROM categorie";
00041                 $this->psGetAllCategories = $this->dbh->prepare($sqlGetAllCategories);
00042                 $this->psGetAllCategories->setFetchMode(PDO::FETCH_ASSOC);
00043
00044                 //add categorie
00045                 $sqlAddCategorie = "INSERT INTO categorie (name) VALUES (:categorie_name)";
00046                 $this->psAddCategorie = $this->dbh->prepare($sqlAddCategorie);
00047                 $this->psAddCategorie->setFetchMode(PDO::FETCH_ASSOC);
00048
00049                 //add categorie to game
00050                 $sqlAddCategorieToGame = "INSERT INTO gamehascategorie (idGame,idCategorie) VALUES
      (:insert_idGame, :insert_idCategorie)";
00051                 $this->psAddCategorieToGame = $this->dbh->prepare($sqlAddCategorieToGame);
00052
00053                 //check if game has a specific categorie
00054                 $sqlCheckIfGameHasCategorie = "SELECT * FROM gamehascategorie WHERE idCategorie =
      :insert_idCategorie AND idGame = :insert_idGame";
00055                 $this->psCheckIfGameHasCategorie = $this->dbh->prepare($sqlCheckIfGameHasCategorie);
00056                 $this->psCheckIfGameHasCategorie->setFetchMode(PDO::FETCH_ASSOC);
00057
00058                 //del categorie from game
00059                 $sqlDelCategorieFromGame = "DELETE FROM gamehascategorie  WHERE idCategorie =
      :del_idCategorie AND idGame = :del_idGame";
00060                 $this->psDelCategorieFromGame = $this->dbh->prepare($sqlDelCategorieFromGame);
00061
00062                 //get categories of a game
00063                 $sqlGameCategorie = "SELECT c.name, c.id FROM `gamehascategorie` as ghc
00064                 LEFT JOIN categorie as c
00065                 ON ghc.idCategorie = c.id
00066                 LEFT JOIN game as g
00067                 ON ghc.idGame = g.id
00068                 WHERE idGame = :search_id";
00069                 $this->psGameCategorie = $this->dbh->prepare($sqlGameCategorie);
00070                 $this->psGameCategorie->setFetchMode(PDO::FETCH_ASSOC);
00071             } catch (PDOException $e) {
00072                 print "Erreur !: " . $e->getMessage() . "<br>";
00073                 die();
00074             }
00075         }
00076     }
00082     public function getListAllCategories()
00083     {
00084
00085         try {
00086             $this->psGetAllCategories->execute();
00087             $result = $this->psGetAllCategories->fetchAll();
00088         } catch (PDOException $e) {
00089             print "Erreur !: " . $e->getMessage() . "<br>";
00090             die();
00091         }
00092         return $result;
00093     }
00100     public function getCategoriesOfGame(int $idGame)
00101     {
00102         try {
```

```
00103             $this->psGameCategorie->execute(array(':search_id' => $idGame));
00104             $result = $this->psGameCategorie->fetchAll();
00105         } catch (PDOException $e) {
00106             print "Erreur !: " . $e->getMessage() . "<br>";
00107             die();
00108         }
00109         return $result;
00110     }
00117     public function addCategorie(string $categorieName)
00118     {
00119         try {
00120             $this->psAddCategorie->execute(array(':categorie_name' => $categorieName));
00121             $result = $this->psAddCategorie->fetchAll();
00122         } catch (PDOException $e) {
00123             print "Erreur !: " . $e->getMessage() . "<br>";
00124             die();
00125         }
00126         return $result;
00127     }
00135     public function addCategorieToGame(int $idGame, int $idCategorie)
00136     {
00137         $result = null;
00138         try {
00139             $this->psCheckIfGameHasCategorie->execute(array(':insert_idCategorie' => $idCategorie,
      ':insert_idGame' => $idGame));
00140             $result = $this->psCheckIfGameHasCategorie->fetchAll();
00141         } catch (PDOException $e) {
00142             print "Erreur !: " . $e->getMessage() . "<br>";
00143             die();
00144         }
00145         if ($result == null) {
00146             try {
00147                 $this->psAddCategorieToGame->execute(array(':insert_idCategorie' => $idCategorie,
      ':insert_idGame' => $idGame));
00148             } catch (PDOException $e) {
00149                 print "Erreur !: " . $e->getMessage() . "<br>";
00150                 die();
00151             }
00152         }
00153     }
00161     public function delCategorieFromGame(int $idGame, int $idCategorie)
00162     {
00163         $result = null;
00164         try {
00165             $this->psDelCategorieFromGame->execute(array(':del_idCategorie' => $idCategorie,
      ':del_idGame' => $idGame));
00166         } catch (PDOException $e) {
00167             print "Erreur !: " . $e->getMessage() . "<br>";
00168             die();
00169         }
00170     }
00171 }
```

5.11 models/class.php File Reference

Class used to handle include all models.

5.11.1 Detailed Description

Class used to handle include all models.

5.11.1.1 BDCC

Author

Lorenzo Bauduccio lorenzo.bdcc@eduge.ch

Copyright

Copyright (c) 2021 BDCC

Definition in file class.php.

5.12 class.php

```
00001 <?php
00009 include_once "login.php";
00010 include_once "userdata.php";
00011 include_once "user.php";
00012 include_once "signin.php";
00013 include_once "games.php";
00014 include_once "download.php";
00015 include_once "categorie.php";
00016 include_once "administrator.php";
00017 ?>
```

5.13 models/download.php File Reference

Class used to handle the download of Caiman.

Data Structures

- class Download

5.13.1 Detailed Description

Class used to handle the download of Caiman.

5.13.1.1 BDCC

Author

Lorenzo Bauduccio lorenzo.bdcc@eduge.ch

Copyright

Copyright (c) 2021 BDCC

Definition in file download.php.

5.14 download.php

```
00001 <?php
00002
00010 class Download
00011 {
00012     public $aMemberVar = 'aMemberVar Member Variable';
00013     public $aFuncName = 'aMemberFunc';
00014
00020     public function downloadCaiman()
00021     {
00022         $filename = '../release/caiman.jpg'; // of course find the exact filename....
00023         header('Pragma: public');
00024         header('Expires: 0');
00025         header('Cache-Control: must-revalidate, post-check=0, pre-check=0');
00026         header('Cache-Control: private', false); // required for certain browsers
00027         header('Content-Type: application/jpg');
00028
00029         header('Content-Disposition: attachment; filename="' . basename($filename) . '";');
00030         header('Content-Transfer-Encoding: binary');
00031         header('Content-Length: ' . filesize($filename));
00032
00033         readfile($filename);
00034
00035         exit;
00036     }
00037 }
```

5.15 models/games.php File Reference

Class servant a gerer les requetes en lien avec la table game.

Data Structures

- class Games

5.15.1 Detailed Description

Class servant a gerer les requetes en lien avec la table game.

5.15.1.1 BDCC

Author

Lorenzo Bauduccio lorenzo.bdcc@eduge.ch

Copyright

Copyright (c) 2021 BDCC

Definition in file games.php.

5.16 games.php

```
00001 <?php
00009 class Games {
00010
00011     private $dbh = null;
00012
00013     private $psGetAllGames = null;
00014
00015     private $psRequestGames;
00016
00017     private $psGameDetail;
00018
00019     private $psFavoriteGameOfUser;
00020
00021     private $psGameInCategorie;
00022
00023     private $psAddGameToFavori;
00024
00025     private $psRemoveGameFromFavori;
00026
00027     private $psCheckIfFavoris;
00028
00029     private $psGetTimeInGame;
00030
00031     private $psGetGameWithTime;
00032
00033
00037     public function __construct()
00038     {
00039         if ($this->dbh == null) {
00040             try {
00041                 $this->dbh = new PDO('mysql:host=' . HOST . ';dbname=' . DBNAME, USER, PASSWORD,
      array(
00042                     PDO::MYSQL_ATTR_INIT_COMMAND => "SET NAMES utf8",
00043                     PDO::ATTR_PERSISTENT => true
00044                 ));
00045                 //get all games
```

```
00046                 $sqlGetAllGames = "SELECT * FROM game";
00047                 $this->psGetAllGames = $this->dbh->prepare($sqlGetAllGames);
00048                 $this->psGetAllGames->setFetchMode(PDO::FETCH_ASSOC);
00049 
00050                 //get request games
00051                 $sqlRequestGames = "SELECT * FROM game WHERE name LIKE :search_game";
00052                 $this->psRequestGames = $this->dbh->prepare($sqlRequestGames);
00053                 $this->psRequestGames->setFetchMode(PDO::FETCH_ASSOC);
00054 
00055                 //get Time in game user
00056                 $sqlTimeInGame = "SELECT * FROM timeingame WHERE idGame = :search_idGame AND idUser =
      :search_idUser ";
00057                 $this->psGetTimeInGame = $this->dbh->prepare($sqlTimeInGame);
00058                 $this->psGetTimeInGame->setFetchMode(PDO::FETCH_ASSOC);
00059 
00060                 //get game with time user
00061                 $sqlGetGameWithTime = "SELECT * FROM timeingame WHERE idUser = :search_idUser ORDER BY
    timeInMinute DESC";
00062                 $this->psGetGameWithTime = $this->dbh->prepare($sqlGetGameWithTime);
00063                 $this->psGetGameWithTime->setFetchMode(PDO::FETCH_ASSOC);
00064 
00065                 //get detail game
00066                 $sqlGameDetail = "SELECT * FROM game WHERE id = :search_id";
00067                 $this->psGameDetail = $this->dbh->prepare($sqlGameDetail);
00068                 $this->psGameDetail->setFetchMode(PDO::FETCH_ASSOC);
00069 
00070                 //add game to favoris
00071                 $sqlAddGameToFavoris = "INSERT INTO favoritegame  (idGame, idUser)
00072                 VALUES (:search_idGame, :search_idUser)";
00073                 $this->psAddGameToFavori = $this->dbh->prepare($sqlAddGameToFavoris);
00074                 $this->psAddGameToFavori->setFetchMode(PDO::FETCH_ASSOC);
00075 
00076                 //remove game to favoris
00077                 $sqlRemoveGameFormFavoris = "DELETE FROM favoritegame
00078                 WHERE idUser = :search_idUser AND idGame = :search_idGame";
00079                 $this->psRemoveGameFromFavori = $this->dbh->prepare($sqlRemoveGameFormFavoris);
00080                 $this->psRemoveGameFromFavori->setFetchMode(PDO::FETCH_ASSOC);
00081 
00082                 //check if already in favoris
00083                 $sqlCheckIfAlreadyFavoris = "SELECT * FROM favoritegame
00084                 WHERE iduser = :search_idUser AND idGame = :search_idGame";
00085                 $this->psCheckIfFavoris = $this->dbh->prepare($sqlCheckIfAlreadyFavoris);
00086                 $this->psCheckIfFavoris->setFetchMode(PDO::FETCH_ASSOC);
00087 
00088                 //get favorite game of user
00089                 $sqlFavoriteGameOfUser = "SELECT g.name, g.id, g.imageName FROM `favoritegame` as fg
00090                 LEFT JOIN game as g
00091                 ON fg.idGame = g.id
00092                 LEFT JOIN user as u
00093                 ON fg.iduser = u.id
00094                 WHERE iduser = :search_id";
00095                 $this->psFavoriteGameOfUser = $this->dbh->prepare($sqlFavoriteGameOfUser);
00096                 $this->psFavoriteGameOfUser->setFetchMode(PDO::FETCH_ASSOC);
00097 
00098 
00099 
00100                 //get list of games in a categorie
00101                 $sqlGameInCategorie = "SELECT g.name, g.id, g.imageName FROM `gamehascategorie` as ghc
00102                 LEFT JOIN game as g
00103                 ON ghc.idGame = g.id
00104                 LEFT JOIN categorie as c
00105                 ON ghc.idCategorie = c.id
00106                 WHERE idCategorie = :search_id";
00107                 $this->psGameInCategorie = $this->dbh->prepare($sqlGameInCategorie);
00108                 $this->psGameInCategorie->setFetchMode(PDO::FETCH_ASSOC);
00109 
00110             } catch (PDOException $e) {
00111                 print "Erreur !: " . $e->getMessage() . "<br>";
00112                 die();
00113             }
00114         }
00115     }
00121     public function getAllGames()
00122     {
00123 
00124         try{
00125             $this->psGetAllGames->execute();
00126             $result = $this->psGetAllGames->fetchAll();
00127 
00128 
00129         }catch (PDOException $e) {
00130             print "Erreur !: " . $e->getMessage() . "<br>";
00131             die();
00132         }
00133         return $result;
00134     }
00140     public function getRequestGames(string $gameName)
```

```
00141     {
00142
00143         try{
00144             $this->psRequestGames->execute(array(':search_game' => '%'.$gameName.'%'));
00145             $result = $this->psRequestGames->fetchAll();
00146
00147
00148         }catch (PDOException $e) {
00149             print "Erreur !: " . $e->getMessage() . "<br>";
00150             die();
00151         }
00152         return $result;
00153     }
00159     public function getTimeInGameUser(int $idUser,int $idGame)
00160     {
00161         try{
00162             $this->psGetTimeInGame->execute(array(':search_idGame' => $idGame, ':search_idUser' =>
      $idUser));
00163             $result = $this->psGetTimeInGame->fetchAll();
00164         }catch (PDOException $e) {
00165             print "Erreur !: " . $e->getMessage() . "<br>";
00166             die();
00167         }
00168         return $result;
00169     }
00175     public function getListOfGameWithTimeUser(int $idUser)
00176     {
00177         try{
00178             $this->psGetGameWithTime->execute(array(':search_idUser' => $idUser));
00179             $result = $this->psGetGameWithTime->fetchAll();
00180
00181
00182         }catch (PDOException $e) {
00183             print "Erreur !: " . $e->getMessage() . "<br>";
00184             die();
00185         }
00186         return $result;
00187     }
00193     public function getGameDetail(int $idGame)
00194     {
00195
00196         try{
00197             $this->psGameDetail->execute(array(':search_id' => $idGame));
00198             $result = $this->psGameDetail->fetchAll();
00199
00200
00201         }catch (PDOException $e) {
00202             print "Erreur !: " . $e->getMessage() . "<br>";
00203             die();
00204         }
00205         return $result;
00206     }
00207
00213     public function getGamesInCategorie(int $idCategorie)
00214     {
00215         try{
00216             $this->psGameInCategorie->execute(array(':search_id' => $idCategorie));
00217             $result = $this->psGameInCategorie->fetchAll();
00218         }catch (PDOException $e) {
00219             print "Erreur !: " . $e->getMessage() . "<br>";
00220             die();
00221         }
00222         return $result;
00223     }
00229     public function getFavoriteGamesOfUser(int $idUser)
00230     {
00231         try{
00232             $this->psFavoriteGameOfUser->execute(array(':search_id' => $idUser));
00233             $result = $this->psFavoriteGameOfUser->fetchAll();
00234
00235
00236         }catch (PDOException $e) {
00237             print "Erreur !: " . $e->getMessage() . "<br>";
00238             die();
00239         }
00240         return $result;
00241     }
00247     public function addGameToFavoris(int $idUser, int $idGame)
00248     {
00249
00250         try{
00251             $this->psAddGameToFavori->execute(array(':search_idUser' => $idUser,':search_idGame' =>
      $idGame));
00252             $result = $this->psAddGameToFavori->fetchAll();
00253
00254         }catch (PDOException $e) {
00255             print "Erreur !: " . $e->getMessage() . "<br>";
```

```
00256             die();
00257         }
00258         return $result;
00259     }
00265     public function removeGameFromFavoris(int $idUser, int $idGame)
00266     {
00267
00268         try{
00269             $this->psRemoveGameFromFavori->execute(array(':search_idUser' => $idUser,':search_idGame'
      => $idGame));
00270
00271
00272         }catch (PDOException $e) {
00273             print "Erreur !: " . $e->getMessage() . "<br>";
00274             die();
00275         }
00276     }
00282     public function checkIfGameIsAlreadyInFavoris(int $idUser, int $idGame)
00283     {
00284         $boolResult = true;
00285
00286         try{
00287             $this->psCheckIfFavoris->execute(array(':search_idUser' => $idUser,':search_idGame' =>
     $idGame));
00288             $result = $this->psCheckIfFavoris->fetchAll();
00289             if ($result != null) {
00290                 $boolResult = false;
00291             }
00292
00293         }catch (PDOException $e) {
00294             print "Erreur !: " . $e->getMessage() . "<br>";
00295             die();
00296         }
00297         return $boolResult;
00298     }
00299
00300 }
```

5.17 models/login.php File Reference

Class used to connect an user.

Data Structures

- class Login

5.17.1 Detailed Description

Class used to connect an user.

5.17.1.1 BDCC

Author

Lorenzo Bauduccio lorenzo.bdcc@eduge.ch

Copyright

Copyright (c) 2021 BDCC

Definition in file login.php.

5.18 login.php

```
00001 <?php
00009 class Login {
00010
00011     private $dbh = null;
00012
00013     private $psLogin = null;
00014
00015     public $search_username = null;
00016
00017     public $search_password = null;
00018
00019     public $arrayInfo = null;
00020
00021     public function __construct()
00022     {
00023         if ($this->dbh == null) {
00024             try {
00025                 $this->dbh = new PDO('mysql:host=' . HOST . ';dbname=' . DBNAME, USER, PASSWORD,
    array(
00026                     PDO::MYSQL_ATTR_INIT_COMMAND => "SET NAMES utf8",
00027                     PDO::ATTR_PERSISTENT => true
00028                 ));
00029
00030                 $sqllogin = "SELECT * FROM user WHERE username = :search_username ";
00031
00032                 $this->psLogin = $this->dbh->prepare($sqllogin);
00033                 $this->psLogin->setFetchMode(PDO::FETCH_ASSOC);
00034
00035
00036             } catch (PDOException $e) {
00037                 print "Erreur !: " . $e->getMessage() . "<br>";
00038                 die();
00039             }
00040         }
00041     }
00047     public function checkLogin()
00048     {
00049         $returnArray = null;
00050         try{
00051             $this->psLogin->execute(array(':search_username' => $this->search_username));
00052             $result = $this->psLogin->fetchAll();
00053             if ($result != null) {
00054                 if (password_verify( $this->search_password,$result[0]["password"])   ) {
00055                     $returnArray = $result;
00056                     $_SESSION['error'] = "Welcome back: ". $result[0]['username'];
00057                 }else
00058                 {
00059                     $_SESSION['error'] = "Invalid log in";
00060                 }
00061             }
00062
00063         }catch (PDOException $e) {
00064             print "Erreur !: " . $e->getMessage() . "<br>";
00065             die();
00066         }
00067         return $returnArray;
00068     }
00069 }
```

5.19 models/signin.php File Reference

Class used to create a new user.

Data Structures

- class Signin

5.19.1 Detailed Description

Class used to create a new user.

5.19.1.1 BDCC

Author

Lorenzo Bauduccio lorenzo.bdcc@eduge.ch

Copyright

Definition in file signin.php.

5.20 signin.php

```
00001 <?php
00009 class Signin
00010 {
00011
00012     private $dbh = null;
00013
00014     private $psInsert = null;
00015
00016     private $psCheckEmail = null;
00017
00018     private $psCheckUsername = null;
00019
00020     public $insert_username = null;
00021
00022     public $insert_password = null;
00023
00024     public $insert_password_repeat = null;
00025
00026     public $insert_email = null;
00030     public function __construct()
00031     {
00032         if ($this->dbh == null) {
00033             try {
00034                 $this->dbh = new PDO('mysql:host=' . HOST . ';dbname=' . DBNAME, USER, PASSWORD,
      array(
00035                     PDO::MYSQL_ATTR_INIT_COMMAND => "SET NAMES utf8",
00036                     PDO::ATTR_PERSISTENT => true
00037                 ));
00038
00039                 // check if email alredy used
00040                 $sqlrequestEmail = "SELECT * FROM user WHERE email = :search_email ";
00041                 $this->psCheckEmail = $this->dbh->prepare($sqlrequestEmail);
00042                 $this->psCheckEmail->setFetchMode(PDO::FETCH_ASSOC);
00043
00044                 // check if username alredy used
00045                 $sqlRequestUsername = "SELECT * FROM user WHERE username = :search_username ";
00046                 $this->psCheckUsername = $this->dbh->prepare($sqlRequestUsername);
00047                 $this->psCheckUsername->setFetchMode(PDO::FETCH_ASSOC);
00048
00049                 $sqlInsert = "INSERT INTO user  (username, password, email)
00050                               VALUES (:insert_username, :insert_password, :insert_email)";
00051                 $this->psInsert = $this->dbh->prepare($sqlInsert);
00052                 $this->psInsert->setFetchMode(PDO::FETCH_ASSOC);
00053             } catch (PDOException $e) {
00054                 print "Erreur !: " . $e->getMessage() . "<br>";
00055                 die();
00056             }
00057         }
00058     }
00064     public function newUser()
00065     {
00066         $isValid = true;
00067         if ($this->checkIfEmailAlreadyTaken()) {
00068             $_SESSION['error'] = "Email already used";
00069             $isValid = false;
00070         }
00071         if ($this->checkifUsernameAlreadyTaken()) {
00072             $_SESSION['error'] = "Username alredy used";
00073             $isValid = false;
00074         }
00075         if ($isValid) {
00076
```

```
00077
00078             try {
00079                 $this->psInsert->execute(array(':insert_username' => $this->insert_username,
      ':insert_password' => password_hash($this->insert_password, PASSWORD_DEFAULT), ':insert_email' =>
      $this->insert_email));
00080                 $_SESSION['error'] = "Acount created";
00081             } catch (PDOException $e) {
00082                 print "Erreur !: " . $e->getMessage() . "<br>";
00083                 die();
00084             }
00085         }
00086     }
00092     public function checkIfUsernameAlreadyTaken()
00093     {
00094         $istaken = true;
00095         try {
00096             $this->psCheckUsername->execute(array(':search_username' => $this->insert_username));
00097             $result = $this->psCheckUsername->fetchAll();
00098             if ($result == null) {
00099                 $istaken = false;
00100             }
00101         } catch (PDOException $e) {
00102             print "Erreur !: " . $e->getMessage() . "<br>";
00103             die();
00104         }
00105         return $istaken;
00106     }
00112     public function checkIfEmailAlreadyTaken()
00113     {
00114         $istaken = true;
00115         try {
00116             $this->psCheckEmail->execute(array(':search_email' => $this->insert_email));
00117             $result = $this->psCheckEmail->fetchAll();
00118             if ($result == null) {
00119                 $istaken = false;
00120             }
00121         } catch (PDOException $e) {
00122             print "Erreur !: " . $e->getMessage() . "<br>";
00123             die();
00124         }
00125         return $istaken;
00126     }
00127 }
```

5.21 models/user.php File Reference

Class use to manage user.

Data Structures

- class User

5.21.1 Detailed Description

Class use to manage user.

5.21.1.1 BDCC

Author

Lorenzo Bauduccio lorenzo.bdcc@eduge.ch

Copyright

Copyright (c) 2021 BDCC

Definition in file user.php.

5.22 user.php

```
00001 <?php
00002
00010 class User
00011 {
00012
00013     public $username;
00014     public $email;
00015     public $role;
00016     public $idUser;
00025     public function __construct(string $usernamep, string $emailp, string $idRolep, int $idUserp)
00026     {
00027
00028         $this->username = $usernamep;
00029         $this->email = $emailp;
00030         $this->role = $idRolep;
00031         $this->idUser = $idUserp;
00032     }
00033
00042     public function updatePassword(string $newPassword, string $newPasswordRepeat, string
      $oldPassword)
00043     {
00044         $dbh = new PDO('mysql:host=' . HOST . ';dbname=' . DBNAME, USER, PASSWORD, array(
00045             PDO::MYSQL_ATTR_INIT_COMMAND => "SET NAMES utf8",
00046             PDO::ATTR_PERSISTENT => true
00047         ));
00048
00049         $hasBeenUpdated = 1;
00050         if (password_verify($oldPassword, $this->getUserPassword())) {
00051
00052             if ($newPasswordRepeat == $newPassword) {
00053                 try {
00054                     $sqlUpdatePassword = "UPDATE user  SET password = :update_password WHERE id =
      :id_user";
00055                     $psUpdatePassword = $dbh->prepare($sqlUpdatePassword);
00056                     $psUpdatePassword->execute(array(':update_password' => password_hash($newPassword,
      PASSWORD_DEFAULT), ':id_user' => $this->idUser));
00057
00058                     $hasBeenUpdated = 0;
00059                 } catch (PDOException $e) {
00060                     print "Erreur !: " . $e->getMessage() . "<br>";
00061                     die();
00062                 }
00063             } else {
00064                 $hasBeenUpdated = 2;
00065             }
00066         } else {
00067             $hasBeenUpdated = 4;
00068         }
00069
00070         return $hasBeenUpdated;
00071     }
00077     public function updatePrivateAccount()
00078     {
00079         $dbh = new PDO('mysql:host=' . HOST . ';dbname=' . DBNAME, USER, PASSWORD, array(
00080             PDO::MYSQL_ATTR_INIT_COMMAND => "SET NAMES utf8",
00081             PDO::ATTR_PERSISTENT => true
00082         ));
00083
00084         $userisPrivate = $this->getPrivateAccount();
00085
00086         if ($userisPrivate == 0) {
00087             $userSetPrivateTo = 1;
00088         } else {
00089             $userSetPrivateTo = 0;
00090         }
00091
00092         try {
00093             $sqlUpdatePrivateAccount = "UPDATE user  SET privateAccount = :update_private_account
      WHERE id = :id_user";
00094             $psUpdatePrivateAccount = $dbh->prepare($sqlUpdatePrivateAccount);
00095             $psUpdatePrivateAccount->execute(array(':update_private_account' => $userSetPrivateTo,
      ':id_user' => $this->idUser));
00096         } catch (PDOException $e) {
00097             print "Erreur !: " . $e->getMessage() . "<br>";
00098             die();
00099         }
00100     }
00106     public function getPrivateAccount()
00107     {
00108         $dbh = new PDO('mysql:host=' . HOST . ';dbname=' . DBNAME, USER, PASSWORD, array(
00109             PDO::MYSQL_ATTR_INIT_COMMAND => "SET NAMES utf8",
00110             PDO::ATTR_PERSISTENT => true
00111         ));
00112
00113         try {
```

```
00114             $sqlGetPrivateAccount = "SELECT privateAccount FROM user WHERE id = :id_user";
00115             $psGetPrivateAccount = $dbh->prepare($sqlGetPrivateAccount);
00116             $psGetPrivateAccount->setFetchMode(PDO::FETCH_ASSOC);
00117             $psGetPrivateAccount->execute(array(':id_user' => $this->idUser));
00118             $result = $psGetPrivateAccount->fetchAll();
00119         } catch (PDOException $e) {
00120             print "Erreur !: " . $e->getMessage() . "<br>";
00121             die();
00122         }
00123 
00124         return $result[0]['privateAccount'];
00125     }
00131     private function getUserPassword()
00132     {
00133         $dbh = new PDO('mysql:host=' . HOST . ';dbname=' . DBNAME, USER, PASSWORD, array(
00134             PDO::MYSQL_ATTR_INIT_COMMAND => "SET NAMES utf8",
00135             PDO::ATTR_PERSISTENT => true
00136         ));
00137 
00138         try {
00139             $sqlGetUserPassword = "SELECT password FROM user WHERE id = :id_user";
00140             $psGetUserPassword = $dbh->prepare($sqlGetUserPassword);
00141             $psGetUserPassword->setFetchMode(PDO::FETCH_ASSOC);
00142             $psGetUserPassword->execute(array(':id_user' => $this->idUser));
00143             $result = $psGetUserPassword->fetchAll();
00144         } catch (PDOException $e) {
00145             print "Erreur !: " . $e->getMessage() . "<br>";
00146             die();
00147         }
00148 
00149         return $result[0]['password'];
00150     }
00151 }
```

5.23 models/userdata.php File Reference

Class use to manage user data.

Data Structures

- class UserData

5.23.1 Detailed Description

Class use to manage user data.

5.23.1.1 BDCC

Author

Lorenzo Bauduccio lorenzo.bdcc@eduge.ch

Copyright

Copyright (c) 2021 BDCC

Definition in file userdata.php.

5.24 userdata.php

```
00001 <?php
00002
00010 class UserData
00011 {
00012     private $dbh = null;
00013
00014     private $psGetUsersByUsername = null;
00015
00016     private $psGetDataUser = null;
00017
00021     public function __construct()
00022     {
00023         if ($this->dbh == null) {
00024             try {
00025                 $this->dbh = new PDO('mysql:host=' . HOST . ';dbname=' . DBNAME, USER, PASSWORD,
      array(
00026                     PDO::MYSQL_ATTR_INIT_COMMAND => "SET NAMES utf8",
00027                     PDO::ATTR_PERSISTENT => true
00028                 ));
00029
00030                 // get list of user by username
00031                 $sqlRequestUsers = "SELECT * FROM user WHERE username LIKE :search_username AND
      privateAccount = 0";
00032                 $this->psGetUsersByUsername = $this->dbh->prepare($sqlRequestUsers);
00033                 $this->psGetUsersByUsername->setFetchMode(PDO::FETCH_ASSOC);
00034
00035                 // get list of user by username
00036                 $sqlGetDataUser = "SELECT * FROM user WHERE id =:search_idUser";
00037                 $this->psGetDataUser = $this->dbh->prepare($sqlGetDataUser);
00038                 $this->psGetDataUser->setFetchMode(PDO::FETCH_ASSOC);
00039             } catch (PDOException $e) {
00040                 print "Erreur !: " . $e->getMessage() . "<br>";
00041                 die();
00042             }
00043         }
00044     }
00051     public function getUsersByUsername($username)
00052     {
00053         try {
00054             $this->psGetUsersByUsername->execute(array(':search_username' => '%' . $username . '%'));
00055             $result = $this->psGetUsersByUsername->fetchAll();
00056         } catch (PDOException $e) {
00057             print "Erreur !: " . $e->getMessage() . "<br>";
00058             die();
00059         }
00060         return $result;
00061     }
00068     public function getUserData($iduser)
00069     {
00070         try {
00071             $this->psGetDataUser->execute(array(':search_idUser' => $iduser));
00072             $result = $this->psGetDataUser->fetchAll();
00073         } catch (PDOException $e) {
00074             print "Erreur !: " . $e->getMessage() . "<br>";
00075             die();
00076         }
00077         return $result;
00078     }
00079 }
```

Index