**Battleships Documentation**

**Content**

1. Game Rules and options

2. User Interfaces and index files

3. Autoloaders

4. Models

5. Controllers

6. Views

7. Possible upgrades. Testing

8. Git

1. Game Rules and options

[Game Rules. Battleships Programming Test v4.docx](Game%20Rules.%20Battleships%20Programming%20Test%20v4.docx)

Game options. As the most games - this game has option for restarting, exiting, hidden cheat option.

The game works with case insensitive coordinates and case sensitive commands ("show" for example).

2. User Interfaces and index files

There are 3 user interfaces. All of them has a new game option. Show all ships positions option.

2.1.) Web user interface. [index.php](http://localhost/battleships/index.php)

Possible option inputs.

Hit a position. "F5" (for example).

See all ships positions. "show".

Start a new game. There is an "a" tag option. "Start a New Game"

2.2.) Shell user interface. [shell\_index.php](http://localhost/battleships/shell_index.php)

Possible option inputs.

Hit a position. "F5" (for example).

See all ships positions. "show".

Start a new game. "new".

2.3.) Shell Commands user interface. [shell\_commands\_index.php](http://localhost/battleships/shell_commands_index.php)

I've made one more type of user interface. I can call it "Hacker Interface". All the commands are with curl commands in the shell. The nice thing is that the game has session time and will be restarted in 20 minutes.

Possible option commands.

Hit a position. curl --data "coord=h4" http://localhost/battleships/shell\_commands\_index.php /POST Request/

See all ships positions. curl --data "coord=show" http://localhost/battleships/shell\_commands\_index.php /POST Request/

Start a new game. curl http://localhost/battleships/shell\_commands\_index.php?new\_game=1 /GET Request/

3. Autoloaders

I've used autoloaders from internet.

There is used 1 autoloader for php classes loading, and another for views loading.

PHP classes loader.

https://github.com/php-fig/fig-standards/blob/master/accepted/PSR-4-autoloader-examples.md

Views loader. Class \vendor\ViewRender.

https://www.smashingmagazine.com/2011/10/getting-started-with-php-templating/

Namespaces naming conventions are just like the directories names.

4. Models

The models are the data holders, validators and main logic providers.

BattleField class.

It holds and loads 2 empty matrix /10x10/.

private $shipMatrix = array();

private $hitMatrix = array();

The game uses <A-J> indexes for the rows and <1-10> indexes for the columns.

There are some validations in this class like the size of the matrixes $this->isValidMatrix(Array $matrix). Position validation isValidHitPosition($axisX, $axisY).

The program is set to throws several exceptions, as some validations.

Ships classes.

I've made an abstract Ship class with the main idea to obligate the successor classes to set the ship sizes on their construction. There is no option for changing their size. On theory if we change a ship size - we make a new ship.

I've named the successors with their length (Ship4 for example).

GameStatus class. It has 2 public static properties, which holds the game success (is the game finished) and the number of hits before that.

5. Controllers

There are 2 controllers.

ShipPositionsController.

It is used only once - for setting the BattleField matrixes.

HitPositionsController.

It controls the whole game after the establishment. Hits positions. Prints Error messages.

6. Views

There are 3 view files, which are php templates. For this game purposes I've used one template per user interface. There is nothing special about that templates. The main idea is that I can set additional options game development, upgrades, etc.

7. Possible upgrades. Testing

There could be set some options for game statistics, like minimum hits, without the cheat option for example.

Testing. There could be included some automated testing (with PHPUnit for example).

Main thigs to be tested will be the generated empty matrixes for their sizes and ship positions. The ship cannot be deployed outside the BattleField partially or entirely.

The next thing is the game false inputs. Not existing coordinates for example.

Unfortunately - my time for the game development is finished. So, I'll make it - some other time.

8. Git

I’ve used Git version control system. You can see my developments by my comits.