Void Eye Game

Author: Jose Antonio Duarte Perez

Description

Final project for 2ºDAW, its a website is intended for support in the search for the best offers and alternatives when buying video games. In addition, together with the detailed description of the video games that will be made through videos and images, you will find the different links to purchase websites and a series of comments written by users. They will also be able to report bugs to developers and view the location of our company on a map. In short, a page that helps customers to see on which page it is sold at the best price and the different prices at the time to buy video games. **Remember that games cannot be purchased on this page, since its purpose is consultation, and not the purchase of video games.**

Installation

Download this repository and mount it on a web server, to give you an idea of how the repository should be located, here would be the repository on a XAMPP server: `C:/xampp/htdocs/VoidEyeGames/`, leaving this file located (`INSTALLATION.md`) in the `VoidEyeGames` folder along with the rest of the files.

Database

In your database, (in local) we proceed to the execution of the following files in the order listed:

1. sql/SQLSentences.sql
2. sql/SQLMyUser.sql
3. sql/SQLInserts.sql to check that everything went correctly, run `sql/SQLQueries.sql`.

Dependencies

IMPORTANT!! Have NodeJs and Composer installed We execute the following commands from the terminal being located in `C:/xampp/htdocs/VoidEyeGames/` execute the command `npm run install` or `npm run deploy`.

Execution

API

Run the **web** service and **database**, if when accessing `http://localhost/VoidEyeGames/void-eye-games-api/` you get this:

```
{"status": "UP"}
```

Then the API was installed correctly.

Client

Move from the terminal to the `C:/xampp/htdocs/VoidEyeGames/` directory and run `npm run start` command, when finished, then you can access from the browser to `localhost:3000/VoidEyeGames/`. **It may take a while, be patient**.

Utils

See `utils/` folder for some utils tips and short features.

Solucion de errores de la instalacion

If you find any error, comment it to me so we can fix it.

Goals

1. That the **users** can *compare the price of video games on each platform (including discounts)* and *select the one that sells it at the best price to buy*.
2. That the **users** can *consult the details in more depth about a game*, i mean, a *description*, a *series of images/videos/trailers* and *the opinion of the users* about the game.
3. That the **users** that is previously **logged in/registered**, can comment on their experience in the game.
4. If a **user** find **a wrong price/bug/error** on our page, whether *you are registered or not*, be able to *inform us* and *help us* **improve** the page, through an *error reporting form*.
5. That the **users** of our page can *find* their game as quickly as possible.

Specifications

1. The application has an **API**, and can be accessed through a **Web Client**.
2. The **API** is made with **PHP** (with **Slim Framework**).
3. The **Web Client** is made with **NodeJS** (with **React**).
4. The layout of the page is made with **Bootstrap** (and therefore **Sass**) and with **Fontawesome**.
5. Games are updated and added through the **Administrator user**, while that **logged in users** can *make comments* about their experience in the game.
6. Data **consultation and updating** is done **manually**, through pages only accessible by the **Administrator user**.

Void Eye Game - Client

The client side application is the one that will be used by them, and the one that will query the data in the **server database through the API**, *repetitive requests are not a problem*, see API for more information.

In this *Users manual* you will see some showcases of Web Client **usage** and a **description** *explaining features and tips* for each page.

Home page

The *home page* is the **main page or entrance to our web client**, here you can find a **slider** of those games that have a *discount applied to certain platforms*, and a **news** section where you will find *the latest games added to the platform*.

One *feature* of this page is that if you **cannot check the status/availability of the API**, a **modal dialog** will be displayed indicating that the API is currently unavailable, and it will not be possible to show you the games

at that time.

Games page

In the *games page* you will find a **aside of filters** and a **section where the filtered games will be listed**, **IMPORTANT!! games will not be filtered until you click the 'Filter' button**, however, the **search box** *in the menu bar at the top of the page does filter in* **real time**, constantly sending requests to the server due to which requires fewer resources.

The list of games is **paginated** and *12 games will be shown per page*, when you click **"show more"**, the following **12 games will be loaded**, the same thing happens in the case of *filters*, only these go from 5 to 5.

Games details page

This is the most important page, in this we can find the details of a game, in these details enter:

- A gallery of medias (images and videos)
- The best three platforms in which the game is sold, *if you click on any you will be redirected to the platform page*.
- The description of the game.
- A comment section for users to share their experience with the game, **IMPORTANT! You can only comment if you are logged in**, and you can do it directly from the game details page.

Support page

On the support page you can find out about who we are, how we work and where we are (thanks to a geolocation map), you can also access the error reporting form.

Report form page

From the error report form, *whether you are registered or not*, you can *inform us* and *help us* **improve** the page, reporting **inaccurate prices**, or **bugs**.

Login form page

On this page, you can **login**, or if **you don't have an account**, *you can access the page to register it*, you can also **recover your account password** on this page.

Signin form page

On this page you can **register your user account**, as long as there is not already *one with the same name/email*.

Game form page *Admin side*

IMPORTANT! to this page only can be accessed by admin users accounts.

On this page, you can **add/edit** games, **add categories to them**, **define a cover/main image for the game**, and **add media to the gallery**. (you will **not be able to add the game to the platforms** from this page).

Tips

On this page, you can find lists, if you **left or right click** on them, a **context menu will be displayed** that will *allow you to perform operations on these items*, in the case of **categories**, you can *delete* the category of this game, and in the case from the **gallery medias**, you can also *delete* them, **but in the case of images there is a special case, when you put the mouse over the item, the resource you have the mouse over will be shown (either image or video)**

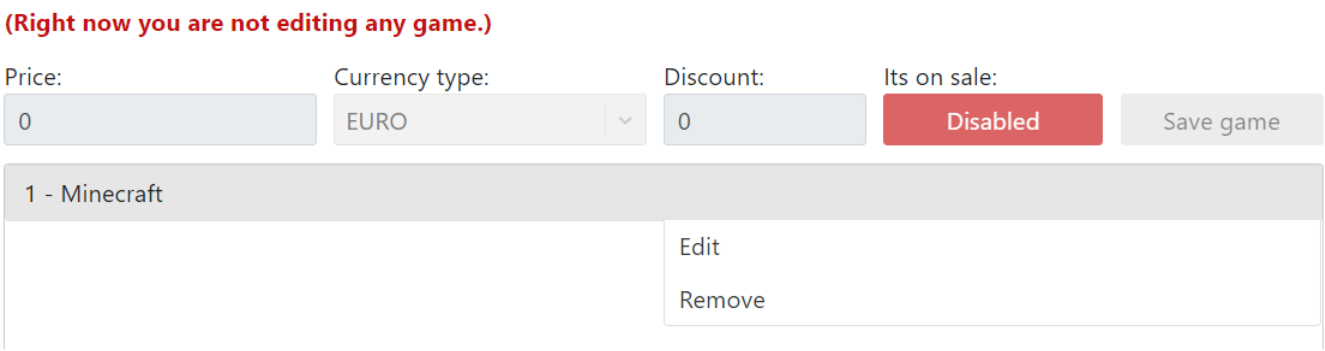
Plataform form page *Admin side*

IMPORTANT! to this page only can be accessed by admin users accounts.

On this page, you can create/edit platforms, and the games they sell on it, the platform will have a **(unique) name**, a **URL**, and an **image**, then it has a *list of games*, when you add a game, you can indicate its **price**, the **type of currency**, the **discount** in percentage (minimum 0, and max 1, 1 equals 100%) and finally, if the game is **currently for sale on that platform** (true or false).

Tips

In the *context menu of the items in the game list*, you will have **2 possible actions**, *delete* the game from the platform, or *edit* its features, in this way, we can edit the **price / type of currency / discount / if it is enabled in said platform**.



Categories form page *Admin side*

IMPORTANT! to this page only can be accessed by admin users accounts.

On this page, you can *create/edit* categories, the categories only require a **unique name**, and they have a list where you can *add games* to this category, the **context menu** is simple, there is only **1 action** which is to **remove the game from this category**.

Void Eye Game - Api

The API is the one that *provides data to the **web client*** and *manage the session system*, see Web Client for more information.

In this *Users manual* you will see the API **usage** and a **description** *explaining features*.

Health system

Sending a **GET** request to */* or */health* you will retrieve this as a response:

```
{
    "status":"UP"
}
```

This **UP** means that the API is reachable on the net, and you can start requesting data.

URL Params

The *health* feature has some *optionals* URL Arguments, these are:

- **showComponents**: apart from *status* it also returns a components object, which indicates the current status of the services. (**Database service**, and **Atlas service**)
- **showLibraries**: apart from *status* it also returns a components object, which indicates the current status of the libraries. (**Logger library**, and **Mailer library**)
- **showDetails**: apart from *status* it also returns a details object, which indicates the current status of the details.

Here you have a example with all params in true:

```
{
   "status":"UP",
   "components":{
      "service.database":{
         "status":"UP",
         "details":{
            "type":"mysql"
         }
      },
      "service.atlas":{ "status":"UP" },
      "library.logger":{ "status":"UP" },
      "library.mailer":{ "status":"UP" }
   },
   "details":{
      "table.users.count":5,
      "table.categories.count":14,
      "table.plataforms.count":12,
      "table.games.count":18,
      "table.games.limit-per-page":12
   }
}
```

Session system

Used for **login/signin** action and for credetials check.
To prevent unidentified users from accessing certain services, a credentials object is required to identify you for said action, the middleware will check if said credentials are valid.

Here you have a example of a request that requires permissions:

```
{
    "credentials": {
        "token": "77c0032ad3bccc14d129c75f2a8837e0",
        "user": "JoseDuarte",
        "expiration": 1653134799000,
        "accountType": 1
    },
    "data": {

    }
}
```

If the **credentials** object is equals to the one saved in the API then the action will be done, otherwise it will return a **403 Not allowed** response.

Login and Singin

You can login sending a simple *username* and *password* to the `/login` url path.

```
{
    "username": "username",
    "password": "password"
}
```

But for signin request, you have to send the followed JSON to the `/signin` url path:

```
{
    "name": "<your name>",
    "password": "<your password encripted with MD5>",
    "confirmationPassword": "<your password encripted with MD5>",
    "email": "<your email>",
    "terms": true
}
```

Data System

For each type of object **[games, categories, platforms and comments]** *similar requests* will be sent to their respective *directories*, in case of being about only 1 object, the directory will be in **singular**, in case of being about several, the directory will be in **plural**, for example:

For only 1 category, we will send the request to `/category`. For more than one category, we will send the request to `/categories`.

Consults

In this way, to **consult** the data of a specific game (we will need the game id) the following request will be sent to `/game?id=<id>`, but if you want more than 1 game, you will send the request to `/games`, too if you need minimized data, you can request to `/games/listed`, in this way, the received data will be considerably lower. The same for the other **Object types**.

Inserts

To **add/insert** more objects of some type, send a **POST** request, with the object type data to the `/game` path.

Updates

If you want to **update** one object type, you will add the **singular object type path** plus `/update` path, so, for category update, you have to send the request to `/category/update`.

Deletes

In no case can *any of the objects created in the database be **deleted***, since we like to have a record of what is happening in the market.

Special object type cases

As a special case, the following types have differences with respect to the previously mentioned.

Games

For the game object type, there is a difference in its way of querying more than 1 game *(the `/games` path)*, since this type has **URL Params** that the rest do not.

These **URL Params** are:

- **pageNum** is the page num that you are loading. (Number)
- **name** is used to search for specific game names. (String)
- **sort** can be one of the follows: name, price, plataform.
- **categories** are the categories ids of the searched games. (Array of Numbers)
- **plataforms** are the plataforms ids of the searched games. (Array of Numbers)

As you can see, the `/games` has a **pageNum** param, this is to allow paging of games, and **limit the size of the data** you can receive from a request, thus allowing for **faster speeds**.

Comments

For the comments object type, this type has only the insert method allowed, so you **cant consult for one, more or minimized comments** and you **cant update this ones**, only you can do is **add** comments.

Object types

This section is to show a **example of object type data** as **json object**

Games


```
{
   "id":"1",
   "name":"[name]",
   "descripcion":"[description]",
   "medias":[
      {
        "id":"1",
        "name":"image/png",
        "gamesId":"1",
        "mediaType":"image/png"
      }
   ],
   "plataforms_games":[
      {
         "plataformsId":"6",
         "gamesId":"1",
         "price":"12.40",
         "priceUnit":"USD",
         "discount":"0.00",
         "isEnabled":"1",
         "plataforms":{
            "id":"6",
            "name":"[Plataform name]",
            "url":"[Plataform url]"
         }
      }
   ],
   "categories":[
      {
         "id":"1",
         "name":"Accion"
      }
   ],
   "comments":[
      {
         "id":"1",
         "usersId":"2",
         "gamesId":"1",
         "description":"Un comentario de ejemplo",
         "users":{
            "id":"2",
            "name":"Juan Alverto"
         }
```

```
        }
    ]
}
```

Categories

```
{
    "id":"1",
    "name":"Accion",
    "games":[
        {
            "id":"1",
            "name":"[Game name]",
            "descripcion":"[Description]"
        }
    ]
}
```

Plataforms

```
{
    "id":"1",
    "name":"[Plataform name]",
    "url":"https:/[Plataform domain]/",
    "plataforms_games":[
        {
            "plataformsId":"1",
            "gamesId":"1",
            "price":"25.99",
            "priceUnit":"EUR",
            "discount":"0.00",
            "isEnabled":"1"
        }
    ],
    "games":[
        {
            "id":"1",
            "name":"[Game name]",
            "descripcion":"[Description]"
        }
    ]
}
```

Others

In this *Others* section, you can find the supports requests, like **password recovery** and **bug reports**.

Password recovery

To recovery a password, you will have to send a **POST** request to `/recovery`, The request must have the following format:

```
{
    "data": {
        "email": "<Email associated with the account>"
    }
}
```

After you send the request, you will **receive an email with your new password**.

Bug report

To report a bug, you will have to send a **POST** request to `/report`, The request must have the following format:

```
{
    "data": {
        "reason": "<The report reason, normally short, example: 'Account problems'>",
        "issue": "<A descriptibe title of your problem>",
        "description": "<A description of the problem>",
        "email": "<Your contact email>",
        "terms": true
    }
}
```

Paths map

A map of all the paths in which you can make requests.

Path	Methods	Type	Params
`/`	GET	Health	none
`/health`	GET	Health	none
`/login`	POST	Session	none
`/signin`	POST	Session	none
`/report`	POST	Others	none
`/recovery`	POST	Others	none
`/game`	GET, POST	Data Type	**Required**: {id}

Path	Methods	Type	Params
`/game/update`	POST	Data Type	none
`/games`	GET	Data Type	{pageNum}, {name}, {sort}, {categories}, {plataforms}
`/games/listed`	GET	Data Type	none
`/category`	GET, POST	Data Type	**Required**: {id}
`/category/update`	POST	Data Type	none
`/categories`	GET	Data Type	none
`/categories/listed`	GET	Data Type	none
`/plataform`	GET, POST	Data Type	**Required**: {id}
`/plataform/update`	POST	Data Type	none
`/plataforms`	GET	Data Type	none
`/plataforms/listed`	GET	Data Type	none
`/comment`	POST	Data Type	none