Victor Albán Fernández

PROYECTO FINAL - 2ª EVALUACIÓN

DESARROLLO WEB EN ENTORNO CLIENTE

Índice

1. DEFINICIÓN Y OBJETIVO DEL PROYECTO 2
2. RECURSOS, HERRAMIENTAS Y LIBRERÍAS EMPLEADAS 3
3. ESQUEMA, DISEÑO Y PLANIFICACIÓN 4
MyVideogamesList V2.0 – Modelo Relacional 6

1. DEFINICIÓN Y OBJETIVO DEL PROYECTO

MyVideogamesList v.2 es la continuación del proyecto presentado en la primera evaluación adaptado a los nuevos conocimientos adquiridos. La aplicación trata de simular una biblioteca en la cual puedes guardar videojuegos. El principal objetivo de la web es que guardes los videojuegos que has jugado para realizar un seguimiento, además de permitir valorar cada uno de ellos en base a una serie de características: jugabilidad, gráficos, apartado artístico, sonoro, narrativa y multijugador online.

2. RECURSOS, HERRAMIENTAS Y LIBRERÍAS EMPLEADAS

Para la realización del proyecto he utilizado…

- Lenguajes de programación/marcas y Base de Datos:

  - ✓ HTML
  - ✓ CSS
  - ✓ JavaScript
  - ✓ PHP
  - ✓ MySQL

- Herramientas/entornos:

  - ✓ Visual Studio Code
  - ✓ XAMPP
  - ✓ phpMyAdmin

- Librerías:

  - ✓ BootStrap
  - ✓ JQuery
  - ✓ JQuery WayPoints
  - ✓ Animate.css
  - ✓ Google Fonts
  - ✓ Font Awesome

3. ESQUEMA, DISEÑO Y PLANIFICACIÓN

Esquema general:

MyVideogamesListV.2

- ➔ index.html
- ➔ library.html
- ➔ request.html
  - *➔ CSS
    - ➔ styles.css
  - *➔ data
    - ➔ myvideogameslist.sql
  - *➔ JS
    - ➔ index.form.script.js
    - ➔ index.view.script.js
    - ➔ library.script.js
    - ➔ library.view.script.js
    - ➔ request.script.js
  - *➔ PHP
    - ➔ addScore.php
    - ➔ addVideogame.php
    - ➔ getAllVideogames.php
    - ➔ getVideogame.php
    - ➔ removeAllVideogames.php
    - ➔ removeVideogame.php

❖ Estructura de la Base de Datos:

La base de datos consta de 6 tablas ENTIDAD y 4 tablas RELACIÓN

- ENTIDADES

videogame – videojuego

genre – género

developer – desarrollador/a

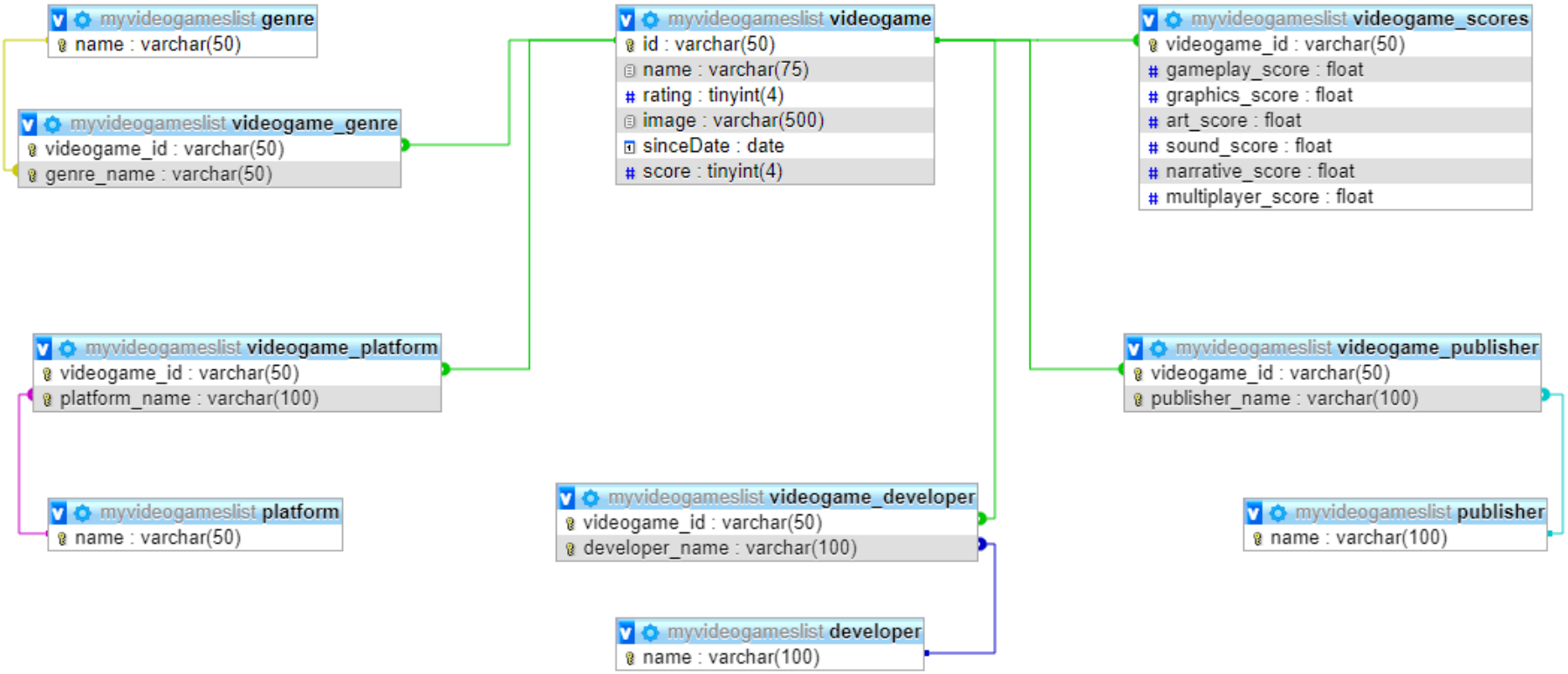
publisher – editora

platform – plataforma

sections_score – puntuación/valoración por apartado

- RELACIONES

videogame_genre	N, M
videogame_developer	N, M
videogame_publisher	N, M
videogame_platform	N, M


myvideogameslist genre
name : varchar(50)
myvideogameslist videogame_genre
videogame_id : varchar(50)
genre_name : varchar(50)
myvideogameslist videogame
id : varchar(50)
name : varchar(75)
rating : tinyint(4)
image : varchar(500)
sinceDate : date
score : tinyint(4)
myvideogameslist videogame_scores
videogame_id : varchar(50)
gameplay_score : float
graphics_score : float
art_score : float
sound_score : float
narrative_score : float
multiplayer_score : float
myvideogameslist videogame_platform
videogame_id : varchar(50)
platform_name : varchar(100)
myvideogameslist videogame_publisher
videogame_id : varchar(50)
publisher_name : varchar(100)
myvideogameslist platform
name : varchar(50)
myvideogameslist videogame_developer
videogame_id : varchar(50)
developer_name : varchar(100)
myvideogameslist publisher
name : varchar(100)
myvideogameslist developer
name : varchar(100)

❖ Descripción de los ficheros

JAVASCRIPT

- index.view.script.js

Script encargado de la manipulación del DOM del fichero index.html

setInterval(función anónima) → esta función se encarga de modificar el background-image de presentación del fichero situado en el header. Se invoca cada 8 segundos.

- index.form.script.js

Script encargado de la gestión de eventos del formulario en el fichero index.html.

start() → asigna la función validate() al evento click del botón de enviar del formulario.

validate(event) → gestiona el evento recibido por parámetro.

validateText(object) → comprueba que los elementos input del DOM que sean de tipo text no estén vacíos.

validateEmail() → comprueba que el formato email sea correcto mediante una expresión regular.

validateTextArea() → comprueba que le área de texto no esté vacío y que no contenga saltos de línea.

sendRequest() → envía los datos obtenidos del formulario. Los datos se guardan en el localStorage y se abre una ventana emergente con un resumen del formulario.

- request.form.script.js

Este script obtiene los datos previamente guardados en el localStorage y los muestra en la ventana emergente.

- library.view.script.js

Script que gestiona parte de los elementos visuales del fichero library.html.

readURL(input) → actualiza la previsualización de la imagen, en el formulario necesario para añadir un videojuego a la biblioteca, de forma dinámica.

- library.script.js

Script principal de la aplicación web. Se encarga de gestionar la biblioteca de videojuegos manipulando el DOM, asignando eventos, etc. Utiliza la librería JQuery en la mayoría de los procedimientos (DOM, AJAX, etc), dejando el JavaScript nativo para las clases y encadenamiento de llamadas AJAX mediante la clase Promise.

CLASES:

VideogamesLibrary

CONSTRUCTOR:

`VideogamesLibrary()`

➔ inicializa un array vacío cuando se instancia.

PROPIEDADES:

`list` :: array en el que se almacenan los videojuegos.

MÉTODOS:

`isEmpty()` → comprueba si la librería está vacía.

return BOOLEAN

`add(Videogame)` → añade un objeto de la clase Videogame a la lista.

remove(videogameID) → elimina un videojuego de la lista.

removeAll() → elimina todos los videojuegos de la lista.

getByID(videogameID) → obtiene un videojuego a partir de su ID.

return Videogame

getGroupByName(string) → obtiene una nueva biblioteca con los videojuegos cuyo nombre tenga alguna coincidencia con el string que recibe por parámetro.

return VideogamesLibrary

sortByName() → ordena la biblioteca por el nombre de los videojuegos en orden alfabético (UNICODE).

Videogame

CONSTRUCTOR:

VideogamesLibrary(id, name, developer, publisher, platform, rating, image, score, sinceDate)

➔ inicializa un videojuego con los valores que recibe por parámetro cuando se instancia.

PROPIEDADES:

id :: identificador del videojuego

name :: nombre/título del videojuego

genre :: género/s a los que pertenece

developer :: empresa desarrolladora

publisher :: empresa editora

platform :: plataforma

rating :: calificación de edad

image :: portada del videojuego

score :: objeto con la valoración de cada apartado

sinceDate :: fecha en la que el videojuego se guarda en la lista.

MÉTODOS:

addScore(gameplay, graphics, genre, developer, publisher, platform, rating, image, score, sinceDate) → añade la valoración de cada apartado del videojuego.

return Object

getAvgScore() → calcula la valoración media sobre 100 del videojuego a partir de la puntuación de los apartados.

return int

$.fn.ready():

Las siguientes funciones gestionan la librería obteniendo los datos de los formularios, llamadas al servidor, etc.

drawVideogamesLibrary(library, animation) → dibuja en el DOM una carta para cada videojuego a través de la siguiente función.

drawVideogameCard(videogame, animation) → establece en el DOM un elemento card con los datos del videojuego y se asigna una animación para su aparición en el DOM.

cloneNodeElement(node) → copia un elemento del DOM con todos sus nodos hijos.

fillData(videogame) → rellena el modal que muestra los datos del videojuego.

fillScoreData(videogame) → muestra un cuadro con la valoración del videojuego en el modal de información del videojuego.

showEmptyLibraryMessage → muestra un mensaje si la biblioteca está vacía.

showNoMatchNamesMessage(name) → muestra un mensaje si no se encuentra ninguna coincidencia al buscar por nombre.

isForbiddenKey(eventKeyCode) → se comprueba si una tecla se considera prohibida a partir de un array con el código de todas las teclas "prohibidas".

return BOOLEAN

getVideogameDataFormValues() → obtiene los datos del formulario necesario para añadir un videojuego.

return Object

getScoreFormValues() → obtiene los datos del formulario necesario para puntuar un videojuego.

return Object

addVideogame(videogame) → encadena una serie de llamadas AJAX por medio de la clase Promise para añadir un videojuego a la base de datos y obtenerlo de nuevo con un ID ya asignado. Una vez terminado este proceso se añade el videojuego a la lista y se muestra en el DOM.

getAllVideogames() → obtiene todos los videojuegos de la Base de Datos mediante una llamada AJAX y los añade a la lista. La aplicación se inicializa con esta función.

addVideogameScore(videogameID, score) → mediante llamadas AJAX encadenadas a través de la clase Promise, se añade una valoración a la base de datos y se obtiene el videojuego valorado para asignarle la puntuación y mostrarla en el modal de información.

removeVideogame(videogameID) → elimina el videojuego, cuyo ID se pasa por parámetro, de la Base de Datos por medio de una llamada AJAX. También lo elimina de la lista y del DOM.

removeAllVideogames() → elimina todos los videojuegos de la Base de Datos y de la biblioteca.

PHP

- addScore.php

Añade la valoración de un videojuego a la Base de Datos. En caso de que ya exista una valoración, ésta se actualiza.

- addVideogame.php

Añade un videojuego a la base de datos, asignándole un ID único. Retorna el id del videojuego añadido.

- getAllVideogames.php

Obtiene todos los videojuegos de la Base de Datos y los retorna en formato JSON.

- getVideogame.php

Obtiene un videojuego de la Base de Datos y lo retorna en formato JSON.

- removeAllVideogames.php

Elimina todos los videojuegos de la Base de Datos.

- removeVideogame.php

Elimina el videojuego indicado de la Base de Datos.