

MEMORIA DEL PROYECTO FINAL 2022 DE TDW

REALIZADO POR VIOLETA MACIAS DE MIGUEL Y DAVID RAMAJO FERNANDEZ

Introducción

El objetivo de este proyecto es crear una aplicación web utilizando otras prácticas realizadas en la asignatura de TDW que permita a un usuario realizar la gestión de aportaciones a la ciencia con productos, entidades y personas.

La aplicación sigue un patrón arquitectónico cliente servidor y en la práctica se hace uso del ORM(mapeador relacional de objetos) doctrine, que hace uso del patrón de diseño Data Mapper.

Este trabajo será realizado con html y css (del ejercicio del tema HTML), javascript (de la Práctica de Front-End), en la parte de frontend se distinguen dos subsistemas diferenciados según el tipo de usuario (reader o writer) que se identifique mediante un nombre de usuario y una contraseña.

El subsistema tipo writer permite operar a un usuario writer con las mismas opciones que había en los casos de uso de la primera práctica a través de un menú. Se ha añadido una opción para la gestión completa (consultas, modificaciones y bajas) de todos los datos del sistema (usuarios, productos, entidades y personas guardadas por los usuarios). Pueden permitir el acceso al sistema a los usuarios registrados (pudiéndose asignar permisos de reader o de writer). Al dar de baja a un usuario, se eliminan todos sus datos correspondientes(delete cascade). Hemos añadido una funcionalidad que permite que un writer pueda suspender a un usuario cambiando su estado actividad a "inactivo" hasta que un writer se lo cambie a "activo". Este método permite conservar los datos de un usuario suspendido para que éste pueda recuperar el acceso a ellos más tarde

El subsistema tipo reader permite a un usuario no registrado acceder a la pantalla de presentación y "registrarse" con un nombre de usuario y una contraseña nuevos para nuestro sistema, a continuación el usuario accederá a una página en la que podrá introducir el resto de sus datos personales (correo electrónico, fecha de nacimiento, urls, etc.).Será necesario que un usuario writer lo autorice para que el usuario nuevo pueda acceder a las diferentes funcionalidades del sistema, que consisten en las opciones contempladas en los casos de uso indicados en la primera práctica.

El usuario reader dispone de una opción para gestionar su cuenta personal ("perfil") pudiendo ver y/o modificar sus datos personales (nombre, dirección de correo electrónico, fecha de nacimiento y contraseña). El identificador de usuario que se ha introducido al registrarse es único en el sistema y se permite su visualización pero no su modificación, además éste está presente en todo momento desde que el usuario ha accedido.

El backend se ha realizado con PHP ampliando e integrando la API REST realizada en el trabajo de php, que incorpora los usuarios del sistema.

También se ha dejado de utilizar el localStorage del navegador que se utilizaba en la práctica de javascript y se ha reemplazado por la base de datos de la práctica de php que nos permite tener persistencia en los datos.

Instalación:

Para comenzar el proyecto abrimos un terminal del ordenador el directorio raíz del proyecto y ejecutamos el comando:

```
> composer install
```

Composer es un estándar para administrar, descargar e instalar dependencias y librerías. Este proyecto tiene múltiples fuentes de instalación, el composer descarga cada dependencia automáticamente, así no hay que hacerlo manualmente.

Doctrine es un Object-Relational Mapper que proporciona persistencia transparente para objetos PHP a través del patrón Data Mapper con el objetivo de obtener un desacoplamiento completo entre la lógica de negocio y la persistencia de los datos en los sistemas de gestión de bases de datos.

```
> bin\doctrine orm:clear-cache:metadata
```

Este comando sirve para limpiar la memoria caché de los metadatos (se explica en el problema 1 de la sección problemas encontrados en este documento.).

Repetimos la instalación del composer:

```
> composer install
```

Para la ejecución del proyecto se utiliza xampp, que nos proporciona un servidor web con apache, que es el encargado de gestionar bases de datos con un MySQL, una vez instalado el proyecto se accederá a la aplicación en la ruta <http://127.0.0.1:8000/> (direccion ip y puerto 8000).

Una vez esté instalado se creará un usuario tipo writer y a continuación se debe poner como inactivo al usuario adminUser.

Cambios en PHP:

Para comenzar esta práctica hemos añadido en la clase "User" los atributos "birthDate", "Url", "name" y "Activo":

```
/**
 * @ORM\Column(
 *     name="birthDate",
 *     type="string",
 *     length = 60,
 *     nullable=true
 * )
 */
protected string | null $birthDate = null;
```

```
/**
 * @ORM\Column(
 *     name = "Url",
 *     type = "string",
 *     length = 60,
 *     nullable = true
 * )
 */
protected string | null $Url = null;
```

```
/**
 * @ORM\Column(
 *     name="name",
 *     type="string",
 *     length = 60,
 *     nullable=true
 * )
 */
protected string | null $name = null;
```

Los atributos Url y birthDay los queremos añadir al registrar un usuario pero también se podrán modificar una vez se haya creado, ambos son de tipo string.

```

/**
 * @ORM\Column(
 *     name      = "Activo",
 *     type      = "integer",
 *     length    = 60,
 *     nullable  = true
 * )
 */
protected int | null $Activo = null;

```

El atributo Activo se utilizará para indicar si el usuario está operativo o si ha sido suspendido, éste es de tipo int y tomará los valores 0, si está “Inactivo” o 1 si está “Activo”, siempre que se cree un usuario se creará inactivo, a la espera de que un usuario tipo writer lo active.

Estos atributos se han añadido al constructor:

```

/**
 * User constructor.
 *
 * @param string $username username
 * @param string $email email
 * @param string $password password
 * @param string $role Role::ROLE_READER | Role::ROLE_WRITER
 * @param string|null $name name
 * @param string|null $birthDate birthDate
 * @param string|null $url url
 * @param int|null $Activo Activo
 *
 * @throws UnexpectedValueException
 */

```

```

public function __construct(
    string $username = '',
    string $email = '',
    string $password = '',
    string $role = Role::ROLE_READER,
    ?string $name = null,
    ?string $birthDate = null,
    ?string $url = null,
    ?int $Activo = 0
) {
    $this->id = 0;
    $this->username = $username;
    $this->email = $email;
    $this->setPassword($password);
    $this->name = $name;
    $this->birthDate = $birthDate;
    $this->url = $url;
    $this->Activo = $Activo;
    try {
        $this->setRole($role);
    } catch (UnexpectedValueException) {
        throw new UnexpectedValueException( message: 'Unexpected Role');
    }
}

```

También se han creado los correspondientes métodos setters y getters:

```

/**
 * @return string|null
 */
public function getbirthDate(): ?string
{
    return $this->birthDate;
}

/**
 * @param string|null $birthDate birthDate
 * @return void
 */
public function setbirthDate(?string $birthDate): void
{
    $this->birthDate = $birthDate;
}

```

```

/**
 * @return string|null
 */
public function getUrl(): ?string
{
    return $this->Url;
}

/**
 * @param string|null $Url Url
 * @return void
 */
public function setUrl(?string $Url): void
{
    $this->Url = $Url;
}

```

```

public function getName(): ?string
{
    return $this->name;
}

/**
 * @param string|null $name name
 * @return void
 */
public function setName(?string $name): void
{
    $this->name = $name;
}

```

```

public function getActivo(): ?int
{
    return $this->Activo;
}

/**
 * @param int|null $Activo Activo
 * @return void
 */
public function setActivo(?int $Activo): void
{
    $this->Activo = $Activo;
}

```

Y se han añadido a métodos propios de esta clase que luego serán útiles para crear el código json correspondiente:

```

public function __toString(): string
{
    return
        sprintf(
            format: '[%s: (id=%04d, username="%s", email="%s", role="%s", name="%s", birthDate="%s", Ur
            basename( path: self::class),
            $this->getId(),
            $this->getUsername(),
            $this->getEmail(),
            $this->role,
            $this->getname(),
            $this->getbirthDate(),
            $this->getUrl(),
            $this->getActivo()
        );
}

```

```

public function jsonSerialize(): array
{
    return [
        'user' => [
            'id' => $this->getId(),
            'username' => $this->getUsername(),
            'email' => $this->getEmail(),
            'role' => $this->role->__toString(),
            'name' => $this->getname() ?? null,
            'birthDate' => $this->getbirthDate() ?? null,
            'UrL' => $this->getUrl() ?? null,
            'Activo' => $this->getActivo() ?? null,
        ]
    ];
}

```

En la clase UserController también hemos añadido un método llamado updateUser, que nos permitirá hacer la actualización de la información del usuario en caso de que la cambie,

obviamente el Identificador no se modificará ya que tiene que ser único y es algo interno del sistema además de que permitirlo podría generar inconsistencias con la base de datos.

```

/ ^ ^
* Update $User with $data attributes
*
* @param Element $element
* @param array $data
*/
protected function updateUser(User $user, array $data): void
{
    foreach ($data as $attr => $datum) {
        switch ($attr) {
            case 'name':
                $user->setname($datum);
                break;
            case 'birthDate':
                $user->setbirthDate($datum);
                break;
            case 'Url':
                $user->setUrl($datum);
                break;
            case 'Activo':
                $user->setActivo($datum);
                break;
        }
    }
}

```

Además hemos añadido al método put estos nuevos parámetros:

```

if (isset($req_data['birthDate'])) {
    $user->setbirthDate($req_data['birthDate']);
}

if (isset($req_data['Url'])) {
    $user->setUrl($req_data['Url']);
}

if (isset($req_data['Activo'])) {
    $user->setActivo($req_data['Activo']);
}

```

Tanto en ElementBaseController.php como en UserController.php hemos añadido la función checkReaderScope:


```
protected function checkReaderScope(Request $request): bool
{
    $scopes = $request->getAttribute( name: 'token')->claims()->get('scopes', null);
    return in_array( needle: Role::ROLE_READER, $scopes, strict: true) || in_array( needle: Role::ROLE_WRITER,
        $scopes, strict: true);
}
```

Esta es llamada desde los métodos get:

```
public function cget(Request $request, Response $response): Response
{
    if (!$this->checkReaderScope($request)) { // 403
        return Error::error($response, statusCode: StatusCode::STATUS_NOT_FOUND);
    }
}
```

Por último a los archivos routersPersons.php, routersEntities.php y routersProducts.php les hemos añadido lo siguiente:

```
->add(JwtMiddleware::class);
```

Modificación en la base de datos:

En la tabla user se han añadido 4 columnas:

id	username	email	name	birthDate	Url	Activo	password	role_value
1	adminUser	adminUser@example.com				1	\$2y\$10\$Y5DqJTucd4vuSdCBFFDQcOKQsmkyrbWnQvx08L78QoV...	1
3	username0	User_email0@example.com	user	2000-12-25	http://u_image07.example.com/	1	\$2y\$10\$VYpJoTMvPx9V11GIGh6Qukzt0iKOx4LJ74U Xim4fp4t...	0

Para poder dar de alta usuarios se han añadido las columnas “name”, “birthDate” y “url”, ya que nuestro sistema tendrá que guardar esos datos al registrar un nuevo usuario y asociarlos a su perfil.

name	birthDate	Url
------	-----------	-----


user	2000-12-25	http://u_image07.example.com/
------	------------	-------------------------------


Adicionalmente se ha creado una columna llamada “Activo” que es la que posteriormente nos proporcionará la funcionalidad de suspender a un usuario temporalmente sin tener que borrar sus datos. Dicha variable sólo será accedida por usuarios tipo writer.

Activo
1
1


Cambios realizados en la API:


Estos son los esquemas de personas y entidades antes de la modificación:

 Persons Person management			^
GET	/persons	Retrieves the collection of Person resources.	▼
POST	/persons	Creates a Person resource.	▼ 🔒
OPTIONS	/persons	Provides the list of HTTP supported methods.	▼
GET	/persons/{personId}	Retrieves a Person resource based on a single ID.	▼
PUT	/persons/{personId}	Updates the Person resource.	▼ 🔒
DELETE	/persons/{personId}	Removes the Person resource.	▼ 🔒
OPTIONS	/persons/{personId}	Provides the list of HTTP supported methods.	▼
GET	/persons/personname/{personname}	Determines if personname exists	▼
GET	/persons/{personId}/entities	List of entities related to the person	▼
GET	/persons/{personId}/persons	List of persons related to the person	▼
PUT	/persons/{personId}/entities/{operation}/{entityId}	Sets or remove the relationship person-entity	▼ 🔒
PUT	/persons/{personId}/products/{operation}/{productId}	Sets or remove the relationship person-product	▼ 🔒

 Entities Entity management			^
GET	/entities	Retrieves the collection of Entity resources.	▼
POST	/entities	Creates a Entity resource.	▼ 🔒
OPTIONS	/entities	Provides the list of HTTP supported methods.	▼
GET	/entities/{entityId}	Retrieves a Entity resource based on a single ID.	▼
PUT	/entities/{entityId}	Updates the Entity resource.	▼ 🔒
DELETE	/entities/{entityId}	Removes the Entity resource.	▼ 🔒
OPTIONS	/entities/{entityId}	Provides the list of HTTP supported methods.	▼
GET	/entities/entityname/{entityname}	Determines if entityname exists	▼
GET	/entities/{entityId}/products	List of products related to the entity	▼
GET	/entities/{entityId}/persons	List of persons related to the entity	▼
PUT	/entities/{entityId}/products/{operation}/{productId}	Sets or remove the relationship entity-product	▼ 🔒
PUT	/entities/{entityId}/persons/{operation}/{personId}	Sets or remove the relationship entity-person	▼ 🔒

Estos son los esquemas después de la modificación:

 Login User login			^
POST	/access_token	Returns api token	▼


 Users User management			^
GET	/users	Retrieves the collection of User resources.	▼ 🔒
POST	/users	Creates a User resource.	▼ 🔒
OPTIONS	/users	Provides the list of HTTP supported methods.	▼
GET	/users/{userId}	Retrieves a User resource based on a single ID.	▼ 🔒
PUT	/users/{userId}	Updates the User resource.	▼ 🔒
DELETE	/users/{userId}	Removes the User resource.	▼ 🔒
OPTIONS	/users/{userId}	Provides the list of HTTP supported methods.	▼
GET	/users/username/{username}	Determines if username exists	▼

💡 Products Product management

GET	/products	Retrieves the collection of Product resources.	✓ 🔒
POST	/products	Creates a Product resource.	✓ 🔒
OPTIONS	/products	Provides the list of HTTP supported methods.	✓
GET	/products/{productId}	Retrieves a Product resource based on a single ID.	✓ 🔒
PUT	/products/{productId}	Updates the Product resource.	✓ 🔒
DELETE	/products/{productId}	Removes the Product resource.	✓ 🔒
OPTIONS	/products/{productId}	Provides the list of HTTP supported methods.	✓
GET	/products/productname/{productname}	Determines if productname exists	✓
GET	/products/{productId}/entities	List of entities related to the product	✓ 🔒
GET	/products/{productId}/persons	List of persons related to the product	✓ 🔒
PUT	/products/{productId}/entities/{operation}/{entityId}	Sets or remove the relationship product-entity	✓ 🔒
OPTIONS	/products/{productId}/entities/{operation}/{entityId}	Provides the list of HTTP supported methods.	✓
PUT	/products/{productId}/persons/{operation}/{personId}	Sets or remove the relationship product-person	✓ 🔒
OPTIONS	/products/{productId}/persons/{operation}/{personId}	Provides the list of HTTP supported methods.	✓

😬 Persons Person management

GET	/persons	Retrieves the collection of Person resources.	✓ 🔒
POST	/persons	Creates a Person resource.	✓ 🔒
OPTIONS	/persons	Provides the list of HTTP supported methods.	✓
GET	/persons/{personId}	Retrieves a Person resource based on a single ID.	✓ 🔒
PUT	/persons/{personId}	Updates the Person resource.	✓ 🔒
DELETE	/persons/{personId}	Removes the Person resource.	✓ 🔒
OPTIONS	/persons/{personId}	Provides the list of HTTP supported methods.	✓
GET	/persons/personname/{personname}	Determines if personname exists	✓
GET	/persons/{personId}/entities	List of entities related to the person	✓ 🔒
GET	/persons/{personId}/persons	List of persons related to the person	✓ 🔒
PUT	/persons/{personId}/entities/{operation}/{entityId}	Sets or remove the relationship person-entity	✓ 🔒
OPTIONS	/persons/{personId}/entities/{operation}/{entityId}	Provides the list of HTTP supported methods.	✓
PUT	/persons/{personId}/products/{operation}/{productId}	Sets or remove the relationship person-product	✓ 🔒
OPTIONS	/persons/{personId}/products/{operation}/{productId}	Provides the list of HTTP supported methods.	✓



Entities

Entity management

GET

/entities

Retrieves the collection of Entity resources.

POST

/entities

Creates a Entity resource.

OPTIONS

/entities

Provides the list of HTTP supported methods.

GET

/entities/{entityId}

Retrieves a Entity resource based on a single ID.

PUT

/entities/{entityId}

Updates the Entity resource.

DELETE

/entities/{entityId}

Removes the Entity resource.

OPTIONS

/entities/{entityId}

Provides the list of HTTP supported methods.

GET

/entities/entityname/{entityname}

Determines if entityname exists

GET

/entities/{entityId}/products

List of products related to the entity

GET

/entities/{entityId}/persons

List of persons related to the entity

PUT

/entities/{entityId}/products/{operation}/{productId}

Sets or remove the relationship entity-product

OPTIONS

/entities/{entityId}/products/{operation}/{productId}

Provides the list of HTTP supported methods.

PUT

/entities/{entityId}/persons/{operation}/{personId}

Sets or remove the relationship entity-person

OPTIONS

/entities/{entityId}/persons/{operation}/{personId}

Provides the list of HTTP supported methods.

Como se puede ver después de la modificación se ha añadido seguridad a casi todos los gets; todos los métodos tienen candados lo que quiere decir que son métodos privados a los que solo tienen acceso los usuarios con permisos especiales. Se ha dejado como públicas las operaciones options y el get por nombre de todos los esquemas para poder determinar si existe el nombre de esa entidad, persona...etc. También hemos añadido 2 métodos options en productos, entidades y personas cuya finalidad es solucionar el Problema 3 de la sección problemas encontrados de este documento.

Aplicación:

La pantalla de inicio es común a todos los usuarios:



Me registro con un usuario x:

Anales de la Ciencia

[INICIO](#)

Guarda todo

Formulario de Usuarios

Contraseña (obligatorio):

x

Usuario (obligatorio):

x

Nombre y Apellidos:

Fecha de nacimiento (aaaa-mm-dd)

email:

ROL :

reader

Url:

Archivo | C:/Users/viole/Downloads/practicaFinalTDWdevdd/index.html

Gmail

YouTube

Maps

Noticias

Esta página dice

Está usted Inactivado en esta BD. Póngase en contacto con un Writer

Aceptar

Usuario: x

Contraseña: •

Login

Registrarse

Se pueden editar los usuarios con permisos de writer, voy a ponerlo en activo:

Archivo C:/Users/viole/Downloads/practicaFinalTDWdevdd/Regis_Edita_Usuario.html

Gmail YouTube Maps Noticias

Esta página dice
Modificaciones realizadas
Aceptar

INICIO Guarda todo Usuario: adminUser

Formulario de Usuarios

Activo (inactivo=0, activo=1): 1

Usuario: x

Nombre y Apellidos: sin datos

Fecha de nacimiento (aaaa-mm-dd): sin datos

email: sin datos

ROL : reader

Url: sin datos

Cambiar contraseña:

Acciones para Writer

Usuario ID: 2 Editar Borrar

ID	Usuario	Activo
1	adminUser	1
2	x	0

Al loguearte con un usuario tipo writer te aparece esta pantalla:

Anales de la Ciencia

Logout Usuario: adminUser Edita Usuarios

 [Youtube](#) Borrar  [fuego](#) Borrar  [banco](#) Borrar

Crear Producto Crear Persona Crear Entidad

Al loguearte con un usuario tipo reader aparecerá esta otra:

Anales de la Ciencia

Logout Usuario: x Edita Usuarios

 [Youtube](#)  [fuego](#)  [banco](#)

Se puede observar que los usuarios writer pueden añadir, modificar y borrar productos entidades y personas.

Un usuario writer podrá editar y borrar un usuario por su id:

Anales de la Ciencia

[INICIO](#)

Guarda todo

Usuario: adminUser

Formulario de Usuarios

Activo (inactivo=0, activo=1):

Usuario:

Nombre y Apellidos:

Fecha de nacimiento (aaaa-mm-dd)

email:

ROL :

Url:

Cambiar contraseña:

Acciones para Writer

Usuario ID:

Editar

Borrar

ID	Usuario	Activo
1	adminUser	1

En cambio un usuario tipo reader solo podrá modificarse a sí mismo:

Anales de la Ciencia

[INICIO](#)

Guarda todo

Usuario: x

Formulario de Usuarios

Activo (inactivo=0, activo=1):

1

Usuario:

x

Nombre y Apellidos:

sin datos

Fecha de nacimiento (aaaa-mm-dd)

sin datos

email:

sin datos

ROL :

reader

Url:

sin datos

Cambiar contraseña:

Ejemplo creación de un producto:

→ C Archivo | C:/Users/viole/Downloads/practicaFinalTDWdevdd/Crea_Edit.html

Aplicaciones Gmail YouTube Maps Noticias

Esta página dice
elemento creado correctamente

Aceptar

INICIO Guarda todo Usuario: adminUser

Formulario para CREAR producto

Nombre (obligatorio): Youtube

Fecha de nacimiento, creación, ...: 00-00-0000

Fecha de defunción, utilidad, ...: 00-00-0000

Imagen, retrato, logo, ..., url a la imagen (Obligatorio): https://www.pixelstalk.net/wp-content/uploads/2012/03/

Wiki, url al elemento: https://www.youtube.com/

Personas que han participado en su desarrollo:

Entidades que han participado en su desarrollo:

AYUDA PARA IDENTIDADES

Persona ---> ID	<input type="text"/>	Buscar
Entidad ---> ID	<input type="text"/>	Buscar
ID ---> Persona	<input type="text"/>	Buscar
ID ---> Entidad	<input type="text"/>	Buscar

Como no se ha guardado la fecha por tener un formato incorrecto lo edito:

→ C Archivo | C:/Users/viole/Downloads/practicaFinalTDWdevdd/Crea_Edit.html

s Gmail YouTube Maps Noticias

Esta página dice
elemento actualizado correctamente

Aceptar

INICIO Guarda todo Usuario: adminUser

Formulario para EDITAR producto

Nombre (obligatorio): Youtube

Fecha de nacimiento, creación, ...:(aaaa-mm-dd) 1999-89-08

Fecha de defunción, utilidad, ...:(aaaa-mm-dd) 9877-98-05

Imagen, retrato, logo, ..., url a la imagen (Obligatorio): https://www.pixelstalk.net/wp-content/uploads/2012/03/

Wiki, url al elemento: https://www.youtube.com/

Personas que han participado en su desarrollo:
(id1,id2,id3,...) sin datos

Entidades que han participado en su desarrollo:
(id1,id2,id3,...) sin datos

AYUDA PARA IDENTIDADES

Persona ---> ID	<input type="text"/>	Buscar
Entidad ---> ID	<input type="text"/>	Buscar
ID ---> Persona	<input type="text"/>	Buscar
ID ---> Entidad	<input type="text"/>	Buscar

Se puede buscar entidades y personas:

Archivo | C:/Users/viole/Downloads/practicaFinalTDWdevdd/Crea_Edita.html

nes Gmail YouTube Maps Noticias

Esta página dice
la persona con ID 1 es: fuego

Aceptar

[INICIO](#) [Guarda todo](#) Usuario: adminUser

Formulario para EDITAR producto

Nombre (obligatorio): Youtube

Fecha de nacimiento, creación, ...:(aaaa-mm-dd) 2006-05-08

Fecha de defunción, utilidad, ...:(aaaa-mm-dd) 9885-02-05

Imagen, retrato, logo, ..., url a la imagen (Obligatorio): https://www.pixelstalk.net/wp-content/uploads/2015/03/youtube.png

Wiki, url al elemento: https://es.wikipedia.org/wiki/YouTube

Personas que han participado en su desarrollo:
(id1,id2,id3,...) sin datos

Entidades que han participado en su desarrollo:
(id1,id2,id3,...) sin datos

AYUDA PARA IDENTIDADES

Persona ---> ID	<input type="text"/>	Buscar
Entidad ---> ID	<input type="text"/>	Buscar
ID ---> Persona	1	Buscar
ID ---> Entidad	<input type="text"/>	Buscar

Personas participantes y entidades también se pueden añadir:

Archivo | C:/Users/viole/Downloads/practicaFinalTDWdevdd/Crea_Edita.html

aciones Gmail YouTube Maps Noticias

Esta página dice
elemento actualizado correctamente

Aceptar

[INICIO](#) [Guarda todo](#) Usuario: adminUser

Formulario para EDITAR producto

Nombre (obligatorio): Youtube

Fecha de nacimiento, creación, ...:(aaaa-mm-dd) 2006-05-08

Fecha de defunción, utilidad, ...:(aaaa-mm-dd) 9885-02-05

Imagen, retrato, logo, ..., url a la imagen (Obligatorio): https://www.pixelstalk.net/wp-content/uploads/2015/03/youtube.png

Wiki, url al elemento: https://es.wikipedia.org/wiki/YouTube

Personas que han participado en su desarrollo:
(id1,id2,id3,...) 1

Entidades que han participado en su desarrollo:
(id1,id2,id3,...) 1

AYUDA PARA IDENTIDADES

Persona ---> ID	<input type="text"/>	Buscar
Entidad ---> ID	<input type="text"/>	Buscar
ID ---> Persona	1	Buscar
ID ---> Entidad	1	Buscar

Resultado:

Anales de la Ciencia

INICIO

Usuario: adminUser

EDITAR elemento

Youtube

2006-05-08

--

9885-02-05





WIKIPEDIA
La enciclopedia libre

- Portada
- Portal de la comunidad
- Actualidad
- Cambios recientes
- Páginas nuevas
- Página aleatoria
- Ayuda
- Donaciones
- Notificar un error
- Herramientas
- Lo que enlaza aquí
- Cambios en enlazadas
- Subir archivo
- Páginas especiales
- Enlace permanente

No has accedido [Discusión](#) [Contribuciones](#) [Crear una cuenta](#) [Acceder](#)

Artículo [Discusión](#) Leer [Editar](#) [Ver historial](#)

Coordenadas:  [37°37'41"N 122°25'36"O](#) (mapa)

YouTube

Para el canal, véase [YouTube \(canal\)](#).

YouTube (pronunciado [juˈtub]) es un [sitio web](#) de origen estadounidense dedicado a compartir videos. Presenta una variedad de clips de películas, programas de televisión y videos musicales, así como contenidos amateur como [videoblogs](#) y YouTube Gaming. Las personas que crean contenido para esta plataforma generalmente son conocidas como *you tubers*.

Fue creado por tres antiguos empleados de [PayPal](#) en febrero de 2005³ y, en octubre de 2006 fue adquirido por [Google Inc.](#) a cambio de 1650 millones de dólares y ahora opera como una de sus filiales. Es el sitio web de su tipo más utilizado en internet.

YouTube usa un reproductor en línea basado en [HTML5](#), que incorporó poco después de que la [W3C](#) lo presentara y que es soportado por los [navegadores web](#) más difundidos. Antiguamente su reproductor funcionaba con [Adobe Flash](#), pero esta herramienta fue

YouTube

YouTube, LLC

Broadcast Yourself
(*Transmite tú mismo*)



Tipo Servicio de alojamiento de videos

Industria Internet

Forma legal Empresa privada de responsabilidad limitada

Fundación 14 de febrero de 2005 (17 años)

Fundador [Steve Chen](#)
[Jawed Karim](#)

*Personas participantes en su desarrollo: **fuego**;*

*Entidades participantes en su desarrollo: **banco**;*

Un usuario reader lo visualizará de esta forma:

Anales de la Ciencia

INICIO

Usuario: x

Youtube

2006-05-08

--

9885-02-05





WIKIPEDIA
La enciclopedia libre

- Portada
- Portal de la comunidad
- Actualidad
- Cambios recientes
- Páginas nuevas
- Página aleatoria
- Ayuda
- Donaciones
- Notificar un error
- Herramientas
- Lo que enlaza aquí
- Cambios en enlazadas
- Subir archivo
- Páginas especiales
- Enlace permanente

No has accedido [Discusión](#) [Contribuciones](#) [Crear una cuenta](#) [Acceder](#)

Artículo [Discusión](#) Leer [Editar](#) [Ver historial](#)

Coordenadas:  [37°37'41"N 122°25'36"O](#) (mapa)

YouTube

Para el canal, véase [YouTube \(canal\)](#).

YouTube (pronunciado [juˈtub]) es un [sitio web](#) de origen estadounidense dedicado a compartir videos. Presenta una variedad de clips de películas, programas de televisión y videos musicales, así como contenidos amateur como [videoblogs](#) y YouTube Gaming. Las personas que crean contenido para esta plataforma generalmente son conocidas como *you tubers*.

Fue creado por tres antiguos empleados de [PayPal](#) en febrero de 2005³ y, en octubre de 2006 fue adquirido por [Google Inc.](#) a cambio de 1650 millones de dólares y ahora opera como una de sus filiales. Es el sitio web de su tipo más utilizado en internet.

YouTube usa un reproductor en línea basado en [HTML5](#), que incorporó poco después de que la [W3C](#) lo presentara y que es soportado por los [navegadores web](#) más difundidos. Antiguamente su reproductor funcionaba con [Adobe Flash](#), pero esta herramienta fue

YouTube

YouTube, LLC

Broadcast Yourself
(*Transmite tú mismo*)



Tipo Servicio de alojamiento de videos

Industria Internet

Forma legal Empresa privada de responsabilidad limitada

Fundación 14 de febrero de 2005 (17 años)

Fundador [Steve Chen](#)
[Jawed Karim](#)

*Personas participantes en su desarrollo: **fuego**;*

*Entidades participantes en su desarrollo: **banco**;*

Se puede hacer de forma análoga para personas y entidades.

Problemas encontrados:

Problema 1: Al modificar el mapeado la tabla user de la base de datos no era modificada, ni tampoco los métodos put, post y delete de la api ya que seguía usando el antiguo mapeado. Lo hemos solucionado borrando la memoria caché de los metadatos utilizando el siguiente comando:

```
> bin\doctrine orm:clear-cache:metadata
```

Problema 2: La cabecera de respuesta de la api no permitía ver todas las cabeceras, para solucionarlo hemos añadido una cabecera en la respuesta editando la función process del fichero CorsMiddleware.php:

```
$response = $response->withHeader( name: 'Access-Control-Expose-Headers', value: $requestHeaders ?: '*' );
```

Problema 3: El servidor no permitía la conexión por la falta de un preflight al realizar una operación put entre los elementos productos, entidades y personas. Para arreglar esto hemos implementado dos métodos option en estos 3 elementos (productos, entidades y personas), ya que el método options es el encargado del preflight, para lograrlo hemos modificado los siguientes archivos: openapi.yaml, para añadir las 2 funcionalidades

Dentro de la carpeta config:

Routesentities.php: Se han añadido las rutas correspondientes a productos y personas en entidades para que estos 2 métodos options funcionen.

```
// OPTIONS: Provides the list of HTTP supported methods
$app->options(
    $_ENV['RUTA_API'] . EntityController::PATH_ENTITIES . $REGEX_ENTITY_ID . '/products/rem' . $REGEX_STUFF_ID,
    EntityController::class . ':options'
)->setName('en1optionsEntity');

// OPTIONS: Provides the list of HTTP supported methods
$app->options(
    $_ENV['RUTA_API'] . EntityController::PATH_ENTITIES . $REGEX_ENTITY_ID . '/products/add' . $REGEX_STUFF_ID,
    EntityController::class . ':options'
)->setName('en1optionsEntity');

// OPTIONS: Provides the list of HTTP supported methods
$app->options(
    $_ENV['RUTA_API'] . EntityController::PATH_ENTITIES . $REGEX_ENTITY_ID . '/persons/rem' . $REGEX_STUFF_ID,
    EntityController::class . ':options'
)->setName('en2optionsEntity');

// OPTIONS: Provides the list of HTTP supported methods
$app->options(
    $_ENV['RUTA_API'] . EntityController::PATH_ENTITIES . $REGEX_ENTITY_ID . '/persons/add' . $REGEX_STUFF_ID,
    EntityController::class . ':options'
)->setName('en2optionsEntity');
```

Routesproduct.php: Se han añadido las rutas correspondientes a entidades y personas en productos para que estos 2 métodos options funcionen.

```
// OPTIONS: Provides the list of HTTP supported methods
$app->options(
    $_ENV['RUTA_API'] . ProductController::PATH_PRODUCTS . $REGEX_PRODUCT_ID . '/persons/rem' . $REGEX_STUFF_ID,
    ProductController::class . ':options'
)->setName('pr2optionsProduct');

// OPTIONS: Provides the list of HTTP supported methods
$app->options(
    $_ENV['RUTA_API'] . ProductController::PATH_PRODUCTS . $REGEX_PRODUCT_ID . '/persons/add' . $REGEX_STUFF_ID,
    ProductController::class . ':options'
)->setName('pr2optionsProduct');

// OPTIONS: Provides the list of HTTP supported methods
$app->options(
    $_ENV['RUTA_API'] . ProductController::PATH_PRODUCTS . $REGEX_PRODUCT_ID . '/entities/rem' . $REGEX_STUFF_ID,
    ProductController::class . ':options'
)->setName('pr1optionsProduct');

// OPTIONS: Provides the list of HTTP supported methods
$app->options(
    $_ENV['RUTA_API'] . ProductController::PATH_PRODUCTS . $REGEX_PRODUCT_ID . '/entities/add' . $REGEX_STUFF_ID,
    ProductController::class . ':options'
)->setName('pr1optionsProduct');
```

Routespersons.php: Se han añadido las rutas correspondientes a productos y entidades en personas para que estos 2 métodos options funcionen.

```
// OPTIONS: Provides the list of HTTP supported methods
$app->options(
    $_ENV['RUTA_API'] . PersonController::PATH_PERSONS . $REGEX_PERSON_ID . '/products/rem' . $REGEX_STUFF_ID,
    PersonController::class . ':options'
)->setName('pe2optionsPerson');

// OPTIONS: Provides the list of HTTP supported methods
$app->options(
    $_ENV['RUTA_API'] . PersonController::PATH_PERSONS . $REGEX_PERSON_ID . '/products/add' . $REGEX_STUFF_ID,
    PersonController::class . ':options'
)->setName('pe2optionsPerson');

// OPTIONS: Provides the list of HTTP supported methods
$app->options(
    $_ENV['RUTA_API'] . PersonController::PATH_PERSONS . $REGEX_PERSON_ID . '/entities/rem' . $REGEX_STUFF_ID,
    PersonController::class . ':options'
)->setName('pe2optionsPerson');

// OPTIONS: Provides the list of HTTP supported methods
$app->options(
    $_ENV['RUTA_API'] . PersonController::PATH_PERSONS . $REGEX_PERSON_ID . '/entities/add' . $REGEX_STUFF_ID,
    PersonController::class . ':options'
)->setName('pe1optionsPerson');
// PUT /persons/{personId}/entities/add/{stuffId}
```

Src\controller\elementrelationsbasecontroller.php: Hemos añadido la función options:

```
public function options(Request $request, Response $response): Response
{
    $routeContext = RouteContext::fromRequest($request);
    $routingResults = $routeContext->getRoutingResults();
    $methods = $routingResults->getAllowedMethods();

    return $response
        ->withStatus( code: 204)
        ->withAddedHeader( name: 'Cache-Control', value: 'private')
        ->withAddedHeader(
            name: 'Allow',
            implode( separator: ', ', $methods)
        );
}
```