

Asignatura: DASI

Tema: Trabajo II

Mail: [acamarara.inf@upsa.es](mailto:acamarara.inf@upsa.es) | [abonomeme.inf@upsa.es](mailto:abonomeme.inf@upsa.es)

Universidad: UPSA

Alumnos: Andrés Bonome Mesías | Ángel Cámara Ramos

**MÓDULO VIDEOJUEGOS-JAKARTA:**

Este módulo es el encargado de obtener los datos de los videojuegos de la base de datos y llevarlos a la aplicación web. Contiene los siguientes paquetes:

* Paquete daos: En este paquete se incluyen la interfaz con las funciones necesarias para hacer las peticiones a la base de datos de los videojuegos y su clase que implementa dichas funciones en un subpaquete impl (implementación)
* GatewayDaoImpl: Se conecta a la dirección del localhost cuyo puerto 8084, es donde se guardan los videojuegos. Contiene las funciones para obtener el listado de videojuegos, un videojuego concreto, actualizar un videojuego, insertar un videojuego y eliminar un videojuego.
* Paquete repository: Aquí se meten la interfaz que incluye las funciones con las que ésta se comunica con el gatewaydao y su clase que las implementa en un subpaquete impl:
* repositoryImpl
* Paquete services: Lleva a cabo, con una interfaz y una clase que implementa sus funciones en un subpaquete impl, el mismo procedimiento que repository con gatewaydao, pero ahora con repository:
* serviceImpl
* Paquete controllers: Contiene un subpaquete forms donde se guarda la clase “VideojuegoForm” que contiene los parámetros para introducir en la URLs y las anotaciones para validar los datos en el formulario. Fuera del subpaquete forms, hay 4 clases más:
* Action: Es una enumeración que contiene 4 parámetros, “INSERT, SELECT, UPDATE, DELETE” que, se usan para indicar la acción a realizar en las diferentes clases del paquete: Insertat, seleccionar, actualizar y eliminar respectivamente.
* FormsControllers: Esta clase es la que se encarga de visualizar los parámetros de un videojuego concreto, añadir nuevos parámetros a un nuevo videojuego o actualizar parámetros existentes de un videojuego, según la acción deseada.

Tiene una función “getFormInsertVideojuegoById()” que se encarga de insertar un nuevo videojuego en la base de datos. El path es “/insert/games” y se visualiza en la página “videojuego.jsp”.

Consta de otra función “getFormUpdateVideojuegoById()” que se encarga de actualizar algún parámetro de un videojuego ya existente en la base de datos. El path es “/update/games” y se visualiza en la página “videojuego.jsp”. Para que se pueda identificar el videojuego a actualizar, se le pasa como pathparam un id.

Consta de otra función “getFormDeleteVideojuegoById()” que se encarga de eliminar un videojuego de la base de datos. El path es “/delete/games” y se visualiza en la página “videojuego.jsp” Para que se pueda identificar el videojuego a eliminar, se le pasa como pathparam un id.

* VideojuegosControllers: Esta clase incluye todas las funciones para insertar, actualizar, eliminar o visualizar la información de los videojuegos.

Consta de una función “getVideojuegos()” con anotación @GET que sirve para obtener el propio listado de videojuegos.

Consta de una función “getVideojuegoById()” con anotación @GET que sirve para obtener del listado de videojuegos anterior, un videojuego concreto. Para llevar esto a cabo, se le pasa como pathparam un id.

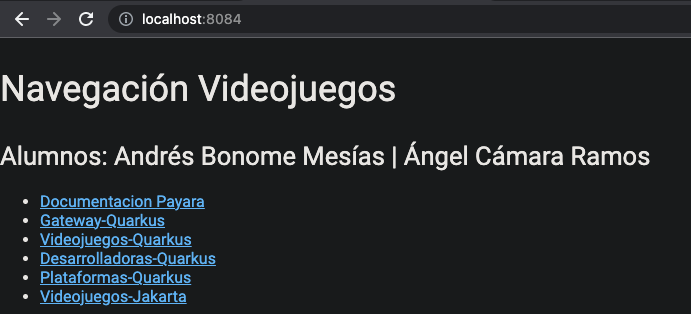
Contiene una función “postVideojuego()” con anotación @POST que sirve para insertar un nuevi videojuego a la base de datos. Para llevar esto a cabo, se le pasa como beanparam un videojuegoForm. El entity consumido es del tipo MediaType.APPLICATION\_FORM\_URLENCODED.

Además, tiene una función “putVideojuegoById()” con anotación @PUT que sirve para actualizar la información de videojuego. Para que esto sea posible, se le pasa como beanparam un videojuegoForm validado y un pathparam id para identificarlo. El entity consumido es del tipo MediaType.APPLICATION\_FORM\_URLENCODED.

Por último de una función “deletejuegoById()” con anotación @DELETE que sirve para actualizar la información de videojuego. Se realiza gracias a que se le pasa un pathparam id para identificarlo. El entity consumido es del tipo MediaType.APPLICATION\_FORM\_URLENCODED.

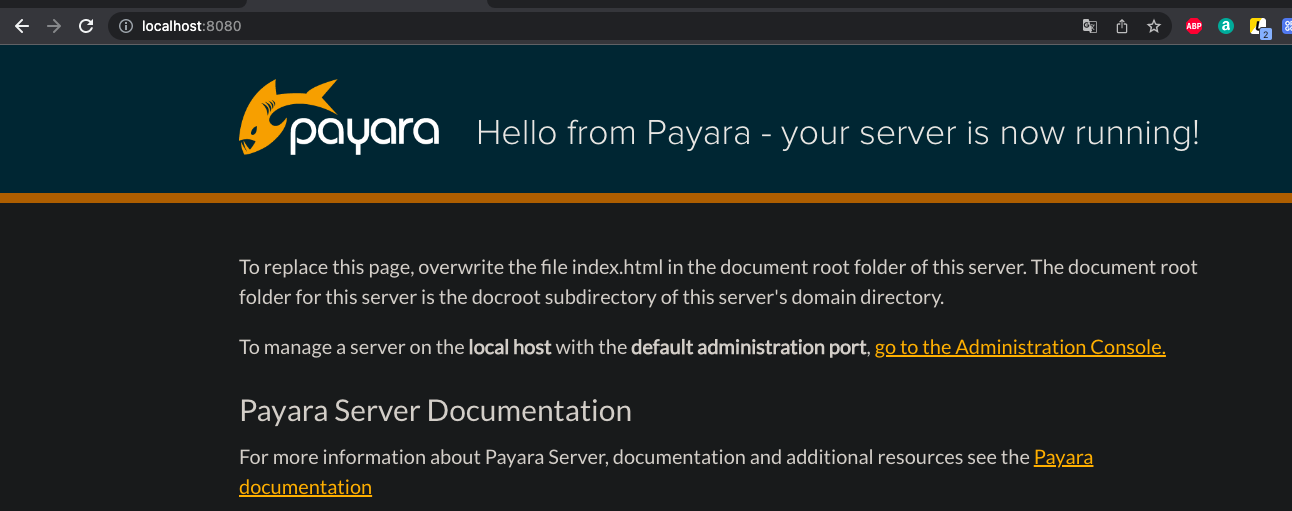
* VideojuegosLocaleResolver: Implementa la inferfaz LocaleResolver, la cual proporciona el método resolveLocale utilizado para la internacionalización de la aplicación.
* Paquete constraintValidation: Contiene una interfaz utilizada para la validación de los datos del tipo fecha “LocalDateMin”. Lo que lleva a cabo es la comprobación de que está por encima de un valor dado. Además, también contiene una clase “LocalDateConstraintMinValidator” que, también, se usa para la validación de datos del tipo fecha.

Hemos implementado dentro de cada localhost una sección de navegación, en ella se puede acceder rápidamente a cada sector de la aplicación, de manera que verás lo siguiente en la interfaz:



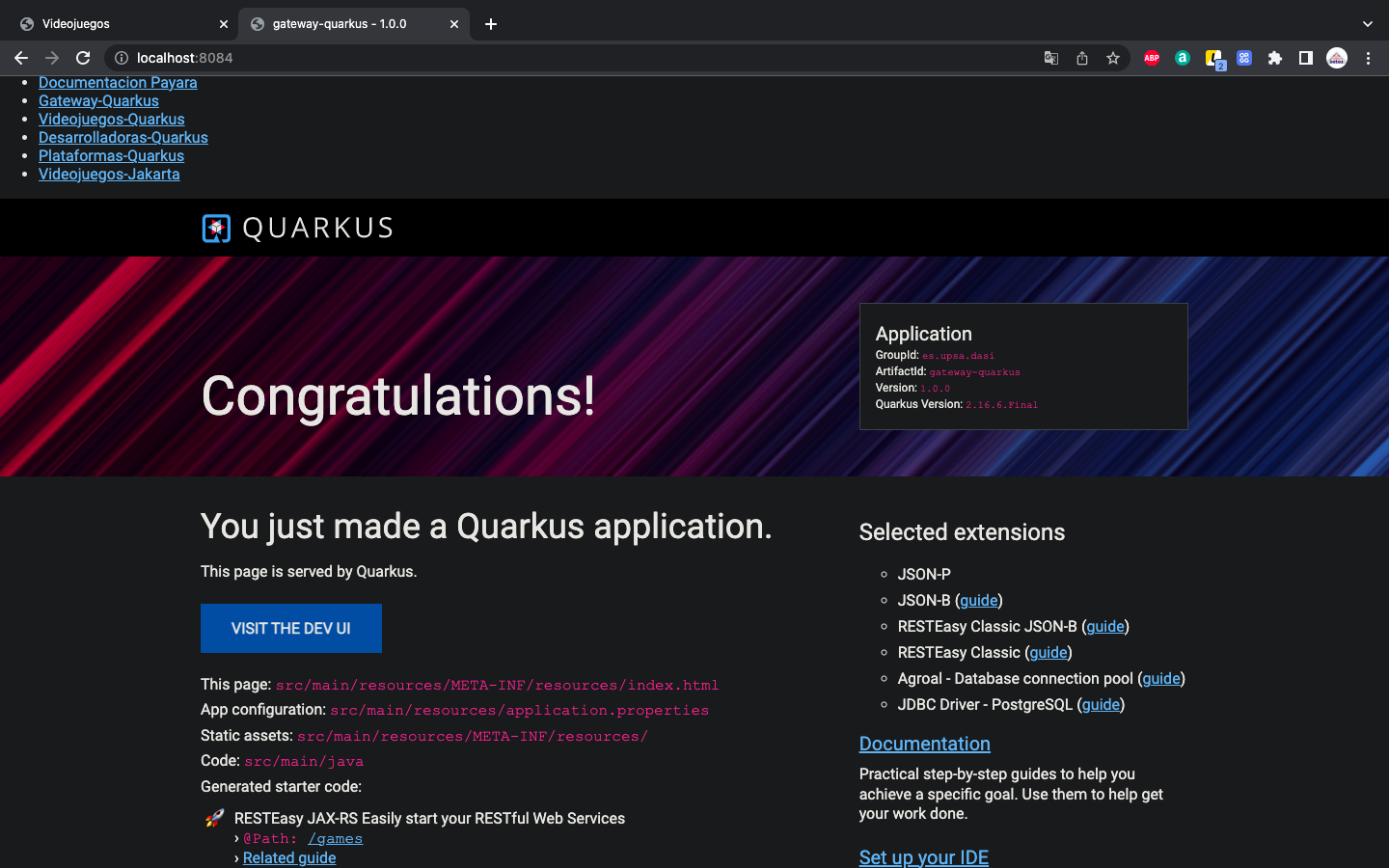
De manera que cada una te lleva a:  
  
**Documentacion Payara:**

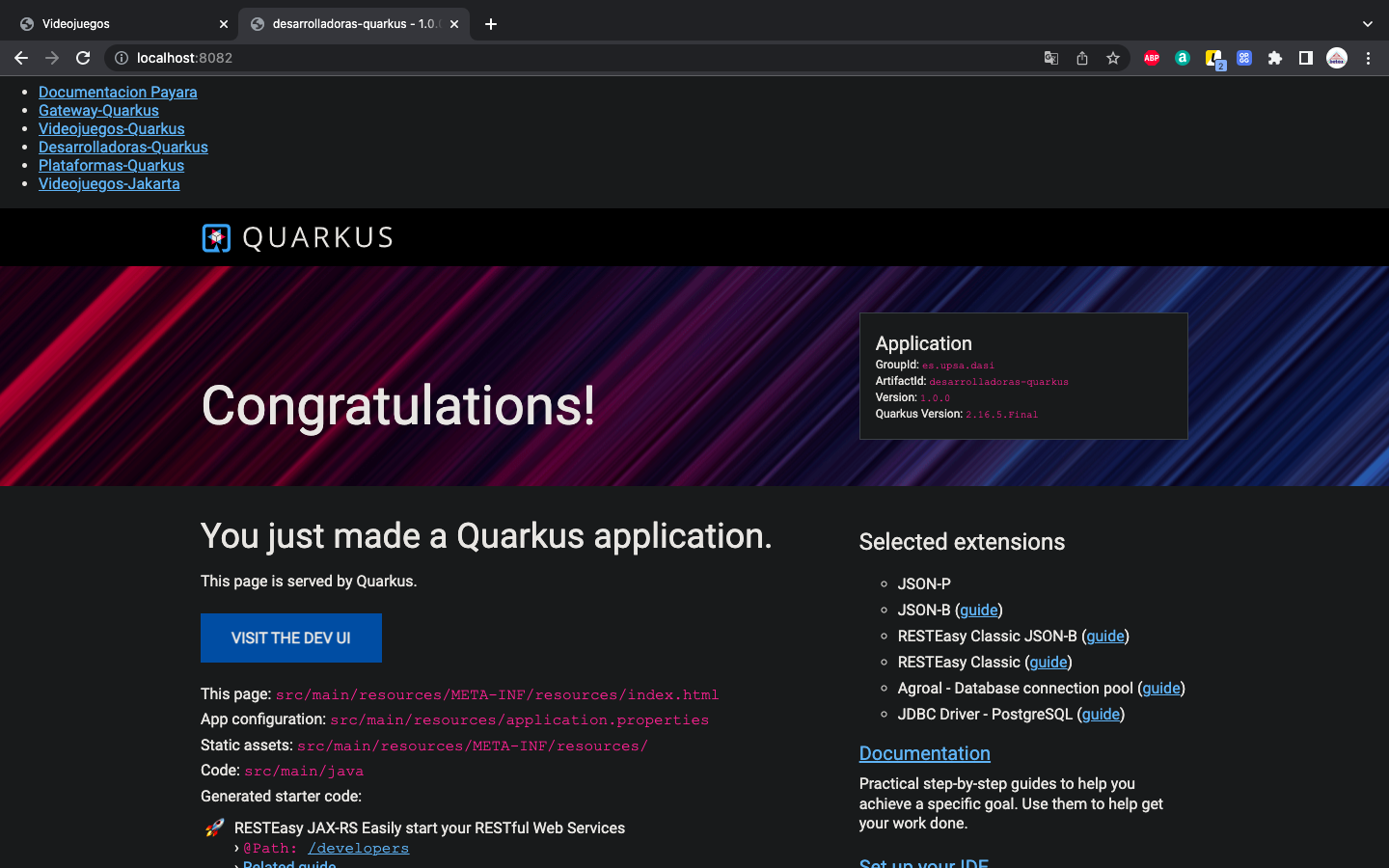
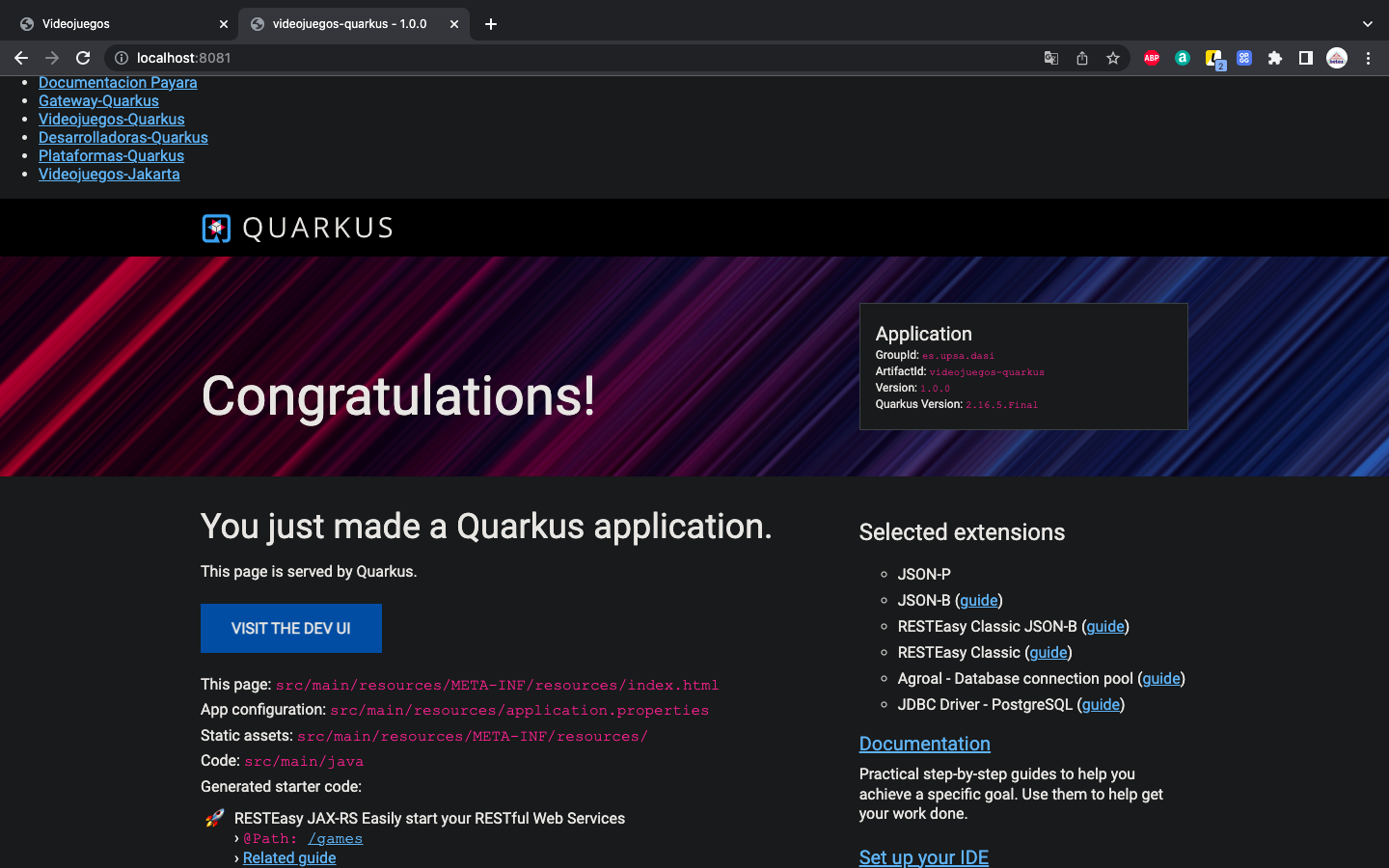
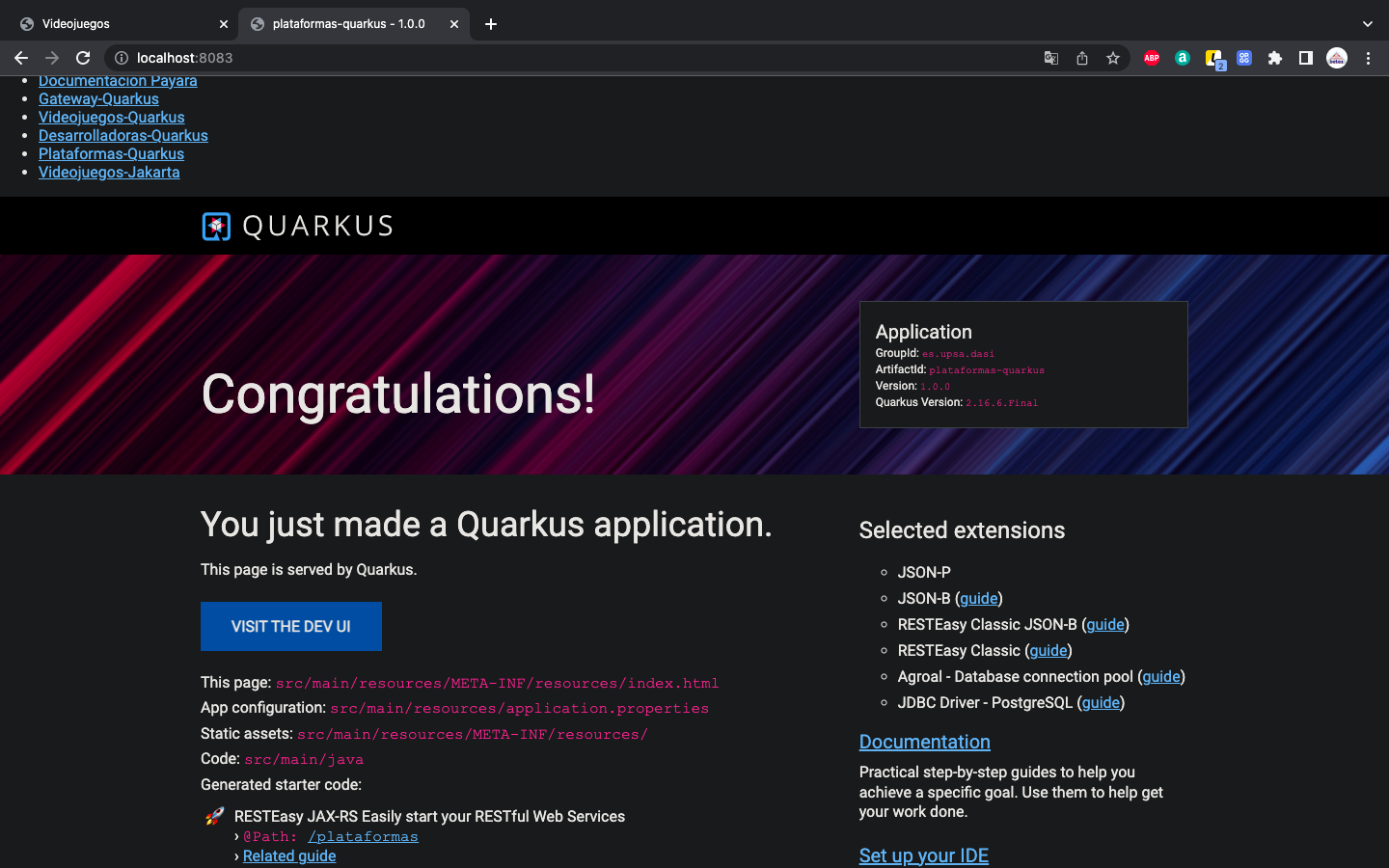
Te lleva a Documentación y Consola de Admin



**Gateway-Quarkus:**

Te lleva a se localhost correspondiente con el enlace /games abajo por si quieres ver los juegos

****

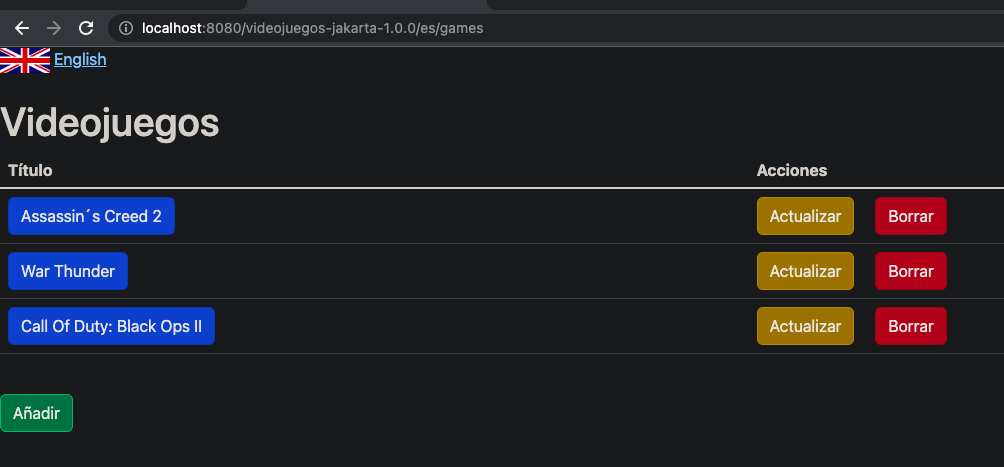
**Videojuegos/Desarrolladora/Plataformas-Quarkus: **

Cada una te lleva a su correspondiente vista

**Videojuegos-payara:**

Esta te lleva a la página de videojuegos donde puedes ver o eliminar las películas.

*Los botones de actualizar y añadir no funcionan*

****

PD: Excepto por la página de Documentación Payara y esta última de Videojuegos-Payara, las demás te permiten ver los enlaces de navegación a las demás tal y como se ven en las capturas.