

OpenPortal

The gate to the cloud experience



Ernesto Galindo Seijas
I.E.S. Iliberis - 2ºASIR

Índice

Índice.....	1
Estudio del problema.....	2
Introducción.....	3
Objetivos.....	4
Planteamiento y evaluación de diversas soluciones.....	4
Justificación de la solución elegida.....	5
Modelado de la solución.....	6
Recursos humanos.....	6
Recursos hardware.....	6
Recursos software.....	7
Planificación temporal. Etapas.....	8
Documentación técnica.....	9
Fase de pruebas.....	11
Documentación del sistema.....	20
Introducción a la aplicación.....	20
Manual de Instalación.....	20
Manual de usuario.....	21
Manual de administración.....	21
Conclusiones finales.....	22
Grado de cumplimiento de los objetivos fijados.....	22
Modificaciones o ampliaciones futuras del sistema implementado.....	23
Bibliografía empleada.....	24

Estudio del problema

Actualmente hay muchas páginas que nos permiten jugar directamente desde el navegador, pero estas se basan en proporcionar juegos en HTML, debido a esto, grandes juegos desarrollados en lenguajes como C# o C++ no son accesibles a todos los usuarios, muchos de ellos por la necesidad de utilizar una máquina de gama alta o consolas especializadas.

Algunas empresas empezaron a desarrollar servicios de juego en la nube, lo que permitía poder jugar a juegos exigentes con los recursos de máquinas especializadas, únicamente teniendo una buena conexión a internet; Sin embargo estas empresas se centraron en juegos de grandes compañías específicas o juegos con comunidades ya establecidas para asegurarse de atraer a su público objetivo.

Por otra parte, con la comunidad de los videojuegos creciendo mucha gente aprovecha para incluir malware en sus archivos con la intención de que la gente lo descargue creyendo que es un juego, lo que ha provocado una desconfianza en la gente hacia descargar juegos de origen desconocido.

Introducción

PortalGame es una plataforma de streaming de videojuegos en la nube pensado para desarrolladores de juegos indies o de bajo presupuesto.

La plataforma busca ofrecer un servicio donde no se tenga que descargar ningún archivo para poder jugar, ofreciendo el uso de máquinas preparadas ante juegos poco optimizados. También busca dar visibilidad a los desarrolladores más pequeños, sin mucho público, permitiendo hacer más accesibles los juegos para las personas.

De esta forma puede permitir a los desarrolladores mandar demos de sus juegos para que sus tester puedan probarlos de manera más centralizada en vez de tener que mandar los archivos de prueba o los enlaces de descargas cada vez que se crea una nueva versión.

Funciones deseadas

El funcionamiento del proyecto se divide en una parte para usuarios y otra para administradores:

- Usuarios:
 - Registro o inicio de sesión.
 - Selección de juego deseado.
 - Inicio correcto del juego deseado.
 - Libre de errores durante el transcurso de la sesión.
 - Uso de protocolos seguros en la página del proyecto.
- Administradores
 - Registro o modificación de nuevos clientes.
 - Registro o modificación de nuevos juegos en la base de datos.
 - Visualización de los diferentes objetos de la base de datos.

Objetivos

El objetivo principal del proyecto es desarrollar una plataforma de juegos en la nube robusta, escalable y segura. Una vez finalizado, el sistema permitirá a los usuarios disfrutar de una experiencia de juego en la nube sin necesidad de hardware potente o software adicional, mientras que los administradores podrán gestionar eficientemente los juegos y las cuentas de usuario.

Planteamiento y evaluación de diversas soluciones

Para lograr el proyecto se pueden utilizar varios métodos:

- Protocolo WebSocket: Este protocolo crea una conexión continua bidireccional entre el servidor y el cliente, logrando una latencia muy baja en el envío de paquetes de datos. Es el protocolo más recomendable para crear sistemas de mensajería instantánea o videollamadas, pero requiere un muy alto conocimiento sobre el protocolo y programación web.
- Nice DCV: Nice Desktop Cloud Visualization es un software de escritorios remotos proporcionado por Amazon Web Services. Este software promete muchas características útiles para el desarrollo del proyecto y ofrece una mayor integración en AWS, aunque carece de una buena documentación sobre cómo utilizar sus herramientas.
- Moonlight: Es un proyecto de código abierto desarrollado por NVIDIA. Este software permite crear un servidor de juego en la nube profesional en el que puedas almacenar tus juegos para después jugar desde cualquier dispositivo, pero requiere instalar una extensión del navegador (ya descontinuada) para poder utilizarlo desde google chrome, además de enfocarse en un servicio más privado que público.

- Myrtille: Es un software de código abierto que convierte el protocolo RDP en HTML5, aunque también es compatible con HTML para navegadores más antiguos. Este software incluye características muy útiles para desarrollar el proyecto, al igual que NICE DCV, pero no hay mucha documentación ante distintos errores y requiere de configuración extra de windows server al no estar pensado específicamente para un servicio de juegos en la nube.

Justificación de la solución elegida.

Todos los software mencionados anteriormente utilizan en gran parte el protocolo WebSocket, por lo que es mucho más lógico utilizar alguno de estos a crear mi propio software con dicho protocolo.

Para seguir con la idea original de no tener que descargar nada para jugar debemos descartar Moonlight, ya que este requiere la instalación de una aplicación específica para su desarrollo y otra para su uso.

De entre las otras 2 opciones realmente se puede pensar que NICE DCV es la más indicada para el proyecto, por tener más documentación y por tener un equipo más grande para el desarrollo del software, pero al necesitar también una configuración extra al no funcionar ciertas características necesarias he decidido utilizar Myrtille por contar con la posibilidad de iniciar sesión con un usuario concreto mediante la url, una opción que solo viene documentado en el github de Myrtille.

Modelado de la solución

Recursos humanos

- Gerente de Proyecto
- Arquitecto de Software
- Diseñador UI
- Desarrolladores Backend
- Desarrolladores Frontend
- Administrador de Base de Datos
- Especialista en Seguridad
- Especialista en Soporte Técnico
- Tester

Recursos hardware

Servidor de Base de Datos:

- Instancia EC2.
- Familia t2.small
- AMI Debian 12 64 bits

Servidor de FTP:

- Instancia EC2.
- Familia t2.small
- AMI Debian 12 64 bits

Servidor de Web:

- Instancia EC2.
- Familia t2.medium
- AMI Debian 12 64 bits

Servidor de Juegos:

- Instancia EC2.
- Familia t2.large
- AMI Microsoft Windows Server 2022 64 bits

Recursos software

Servidor de Base de Datos:

- MongoDB Community Edition
- Mongosh

Servidor de FTP:

- Vsftpd

Servidor de Web:

- Nginx
- PHP-FPM
- Composer
- Docker
- Minikube
- Kubectl

Servidor de Juegos:

- Rol de Windows - Servicio de Dominios de Active Directory
- Rol de Windows - Servicio de Escritorio Remoto
- Myrtille
- Certbot
- OpenSSL

Planificación temporal. Etapas.

1. Investigación

La primera etapa, está dirigida a la investigación de la tecnología a usar en el proyecto, para permitir el poder jugar desde la web. Este proceso fue el mayor desafío del proyecto, ya que debí cambiar varias veces la forma de implementar la solución para desarrollar el proyecto. Esta etapa duró 6 semanas.

2. Topología.

Durante la segunda etapa, se implementó definitivamente la infraestructura que se utilizó para el proyecto, donde se incluía el servidor de juegos, el servidor web y el servidor de base de datos, además se configuró los grupos de seguridad, ip elásticas y claves de inicio de sesión necesarias desde AWS. Esta etapa tomó 2 semanas.

3. Configuración del servidor

En esta etapa, se configuró el servidor para que funcionara de manera correcta, y se diseñaron las funcionalidades que se quería conseguir con el proyecto, así como las limitaciones que se impondrán a los usuarios. Esta etapa duró 1 semana.

4. Configuración del cliente

Durante esta parte del proyecto, se diseñó la interfaz de usuario final de la página web, con el objetivo de ofrecer una experiencia atractiva y fácil de usar para los usuarios. Esta etapa duró 1 semana.

5. Implementación de mejoras.

En esta etapa del proyecto, se implementaron mejoras en el proyecto, como la creación de una imagen de docker para el uso de kubernetes y un servidor ftp para gestionar las copias de seguridad. Se dedicó el tiempo restante hasta la fecha de entrega del proyecto.

Documentación técnica

Servidor de Base de Datos:

- Solo podremos conectarnos mediante ssh, habiéndonos conectado previamente al servidor web.
- Para entrar a la terminal de la base de datos deberemos conectarnos con el comando:
“mongosh mongodb://portalgameUser@localhost:27017/portalgame”
e ingresar la contraseña del usuario.
- Una vez dentro debemos utilizar la sintaxis de mongosh para poder hacer sentencias en la base de datos.
- El usuario tiene asignado un rol con el que únicamente podrá hacer sentencias Find (Select), Insert, Delete y Update.

Servidor de FTP:

- Para conectarnos al servidor vsftpd lo haremos con el comando:
“ftp <ip_servidor>” He iniciaremos sesión con el usuario *“portal”*
- Se han programado tareas en los servidores del proyecto para mandar periódicamente copias de seguridad al servidor ftp.

Servidor de Web:

- Los ficheros php utilizados en el proyecto están subidos a GitHub en mi repositorio público <https://github.com/ErnesGS/PortalGame/tree/master>
- Se ha necesitado hacer uso del software Composer para instalar el driver de MongoDB para php-fpm.
- Se ha instalado docker y se ha creado un Dockerfile para la imagen del servidor web, el Dockerfile también está en el GitHub.
- Se han creado archivos .yaml para la futura implementación de kubernetes en el proyecto.
- Las modificaciones de la página web se harán en el servidor sobre el directorio "PortalGame" antes de convertirlo en una imagen docker, una vez la prueba sea satisfactoria se subirá la imagen a docker Hub, de esa manera tendremos acceso a un control de versiones.

Servidor de Juegos:

- La conexión al servidor se hará mediante un archivo .rdp proporcionado por AWS.
- Los juegos que se utilizarán en la presentación están almacenados en el directorio "C:\juegos\"
- Los usuarios de AD creados en el servidor están ubicados en diferentes unidades organizativas.
- Cada UO tiene vinculada una GPO para que se ejecute un juego específico cada vez que alguno de sus usuarios inicia sesión.
- En la UO "Controlador de Dominio", donde se encuentran todas las UO, hay una GPO que establece un tiempo máximo para cuentas desconectadas y para cuentas sin actividad, además asigna los script de logon y logoff para todos los usuarios.
- Para la gestión de los certificados y la página del servidor de juego usaremos la herramienta "Internet Information Service (IIS) Manager"

Fase de pruebas

Servidor de Base de Datos:

La base de datos está ubicada en una subred privada a la que solo tiene acceso las máquinas de la subnet pública.

Además solo permite el tráfico por el puerto 27017 (MongoDB) por la ip privada del servidor.

```
GNU nano 7.2 /etc/mongod.conf
# mongod.conf

# for documentation of all options, see:
# http://docs.mongodb.org/manual/reference/configuration-options/

# Where and how to store data.
storage:
  dbPath: /var/lib/mongodb
# engine:
# wiredTiger:

# where to write logging data.
systemLog:
  destination: file
  logAppend: true
  path: /var/log/mongodb/mongod.log

# network interfaces
net:
  port: 27017
  bindIp: 127.0.0.1,192.168.1.16
```

También hago uso de usuarios y roles para la conexión con la base de datos

```
portalgame> show users
[
  {
    _id: 'portalgame.portalgameUser',
    userId: UUID('ddfc12f4-c3de-497e-890a-f58f5a21f0c0'),
    user: 'portalgameUser',
    db: 'portalgame',
    roles: [ { role: 'Manager', db: 'portalgame' } ],
    mechanisms: [ 'SCRAM-SHA-1', 'SCRAM-SHA-256' ]
  }
]
portalgame> db.getRole("Manager", {showPrivileges: true})
{
  _id: 'portalgame.Manager',
  role: 'Manager',
  db: 'portalgame',
  privileges: [
    {
      resource: { db: 'portalgame', collection: '' },
      actions: [ 'find', 'insert', 'remove', 'update' ]
    }
  ],
  roles: [],
  inheritedRoles: [],
  inheritedPrivileges: [
    {
      resource: { db: 'portalgame', collection: '' },
      actions: [ 'find', 'insert', 'remove', 'update' ]
    }
  ]
}
```

Servidor de FTP:

El servidor ftp está configurado con vsftpd de forma que permite al usuario “portal” del servidor ftp conectarse desde los servidores de base de datos y web.

```
admin@ip-192-168-0-248:~$ ftp 192.168.1.12
Connected to 192.168.1.12.
220 Welcome to VSFTP service.
Name (192.168.1.12:admin): portal
331 Please specify the password.
Password:
230 Login successful.
Remote system type is UNIX.
Using binary mode to transfer files.
ftp> |
```

[Conexión al servidor FTP desde el servidor web]

Este servidor también está ubicado en subred privada, permitiendo acceder a él únicamente a los servidores que mandarán las copias de seguridad.

Las copias de seguridad se almacenan en el servidor por medio de tareas de crontab en las otras máquinas, las cuales ejecutan un script con comandos “curl” todos los días a las 02:00 para mandar archivos por ftp.

```
GNU nano 7.2 export.sh
curl -T ~/Dockerfile ftp://portal: @192.168.1.12/~web/Dockerfile
curl -T ~/PortalGame/ ftp://portal: @192.168.1.12/~web/PortalGame
curl -T ~/mongo.yaml ftp://portal: @192.168.1.12/~web/mongo.yaml
curl -T ~/portalgame.yaml ftp://portal: @192.168.1.12/~web/portalgame.yaml
curl -T ~/service.yaml ftp://portal: @192.168.1.12/~web/service.yaml
curl -T ~/nginx.conf ftp://portal: @192.168.1.12/~web/nginx.conf
```

```
GNU nano 7.2 export.sh
mongoexport --db portalgame --collection clientes --out /home/admin/datos/clientes.json
curl -T /home/admin/datos/clientes.json ftp://portal: @192.168.1.12/~db/clientes.json

mongoexport --db portalgame --collection usuarios --out /home/admin/datos/usuarios.json
curl -T /home/admin/datos/usuarios.json ftp://portal: @192.168.1.12/~db/usuario.json

mongoexport --db portalgame --collection juegos --out /home/admin/datos/juegos.json
curl -T /home/admin/datos/juegos.json ftp://portal: @192.168.1.12/~db/juegos.json
```

[Scripts que ejecutan las tareas de crontab]

```

admin@ip-192-168-1-12:~$ sudo find /home/portal/web
/home/portal/web
/home/portal/web/nginx.conf
/home/portal/web/mongo.yaml
/home/portal/web/portalgame.yaml
/home/portal/web/Dockerfile
/home/portal/web/PortalGame
/home/portal/web/service.yaml
admin@ip-192-168-1-12:~$ sudo find /home/portal/db
/home/portal/db
/home/portal/db/usuario.json
/home/portal/db/juegos.json
/home/portal/db/clientes.json
admin@ip-192-168-1-12:~$

```

[Imagen del directorio donde se guardan las copias de seguridad]

Servidor de Web:

El servidor web funciona mediante una imagen Debian de Docker con todos los servicios necesarios y configurada para funcionar con certificados en el puerto 443.

```

admin@ip-192-168-0-248:~$ docker ps
CONTAINER ID   IMAGE                                COMMAND                  CREATED        STATUS
PORTS         NAMES
ef3cb3c1e201   ernestogs/portalgame:web2          "/start.sh"             5 hours ago   Up 5 hours
80/tcp, 0.0.0.0:443->443/tcp, :::443->443/tcp   portalgame
admin@ip-192-168-0-248:~$

```

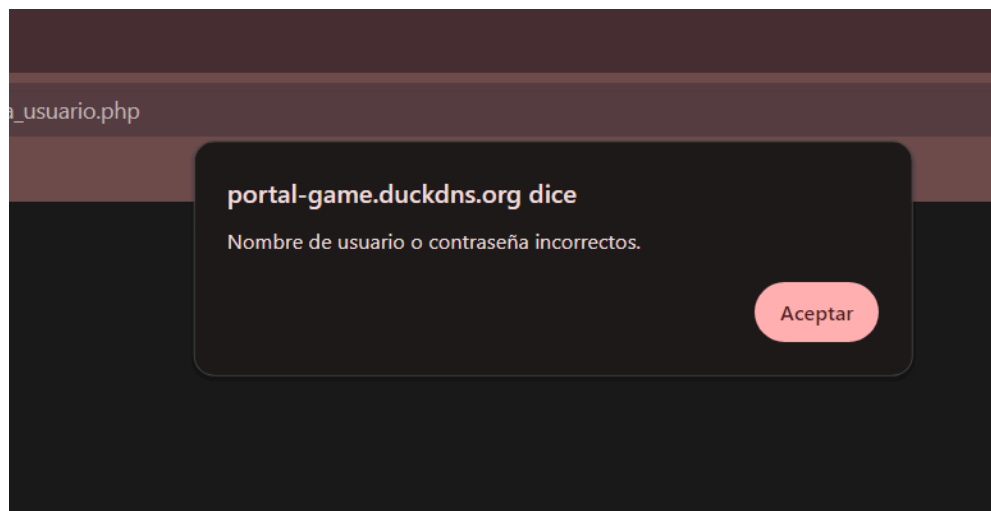
[Contenedor de la página web iniciado]

The screenshot shows a web browser window with the address bar displaying 'portal-game.duckdns.org'. The page has a dark theme. In the center, there is a light gray box titled 'Inicio de Sesión'. Inside this box, there are two input fields: 'Usuario:' with a placeholder 'Usuario o Correo' and 'Contraseña:'. Below these fields are two orange buttons: 'Iniciar Sesión' and 'Registrarse'.

[Formulario de inicio de sesión de la página web]

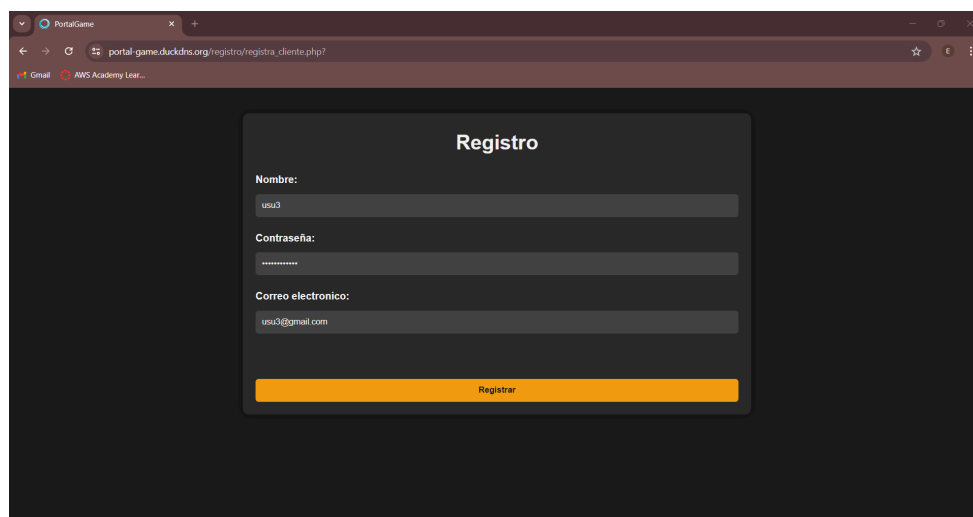
La página de inicio es un formulario que te pide iniciar sesión con tu correo electrónico o nombre de usuario y contraseña.

Si se ingresa un nombre o contraseña incorrectos, la página mostrará un mensaje que nos indicará que nos hemos equivocado.



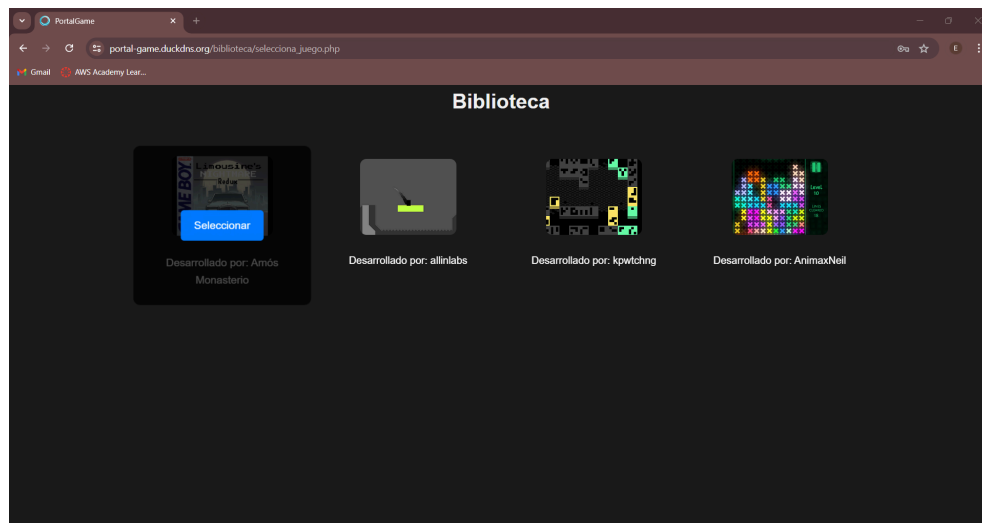
[Mensaje de error ante un usuario no registrado]

Si no tenemos un usuario podemos registrarnos desde el botón registrarse.



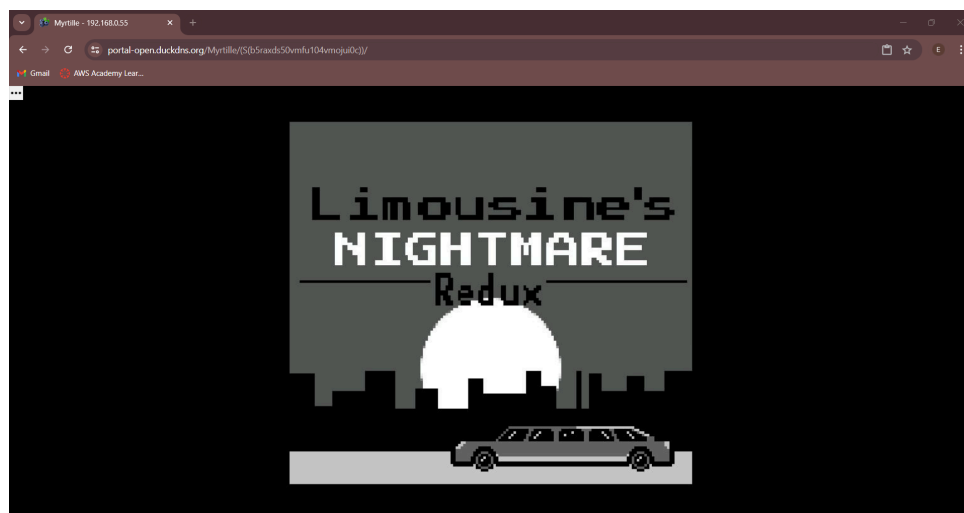
[Formulario de registro de la página web]

Cuando ingresamos correctamente a la página, esta nos redirecciona a la biblioteca, donde podremos seleccionar el juego que nos interese.



[Biblioteca de juegos de la página web]

Después de elegir un juego la página nos manda al servidor de juegos, al que entramos mediante la url directamente con el 1º usuario libre del servidor.



[Imagen del funcionamiento del juego en el navegador]

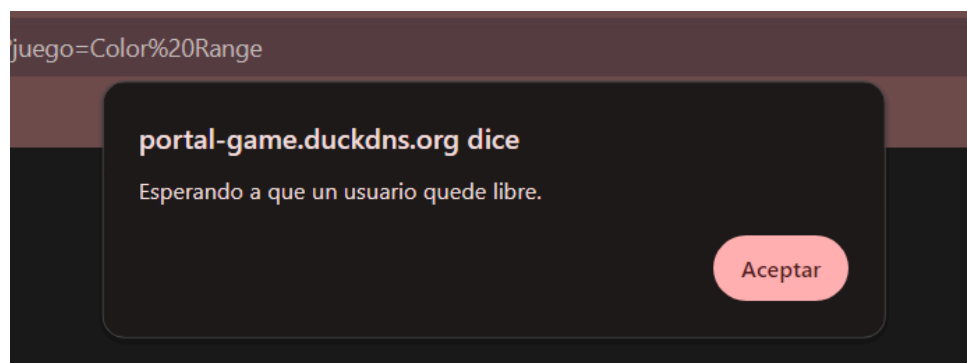
Cuando elegimos un juego, el servidor web hace una búsqueda en la base de datos, de todos los usuarios que estén desconectados y que puedan acceder a ese juego, el 1º usuario que nos manda la base de datos es con el que accedemos al servidor de juegos.

Automáticamente cuando nos conectamos con un usuario, el servidor de juego ejecuta un script que detecta el servidor web para que actualice el estado de ese usuario de desconectado a conectado. Y cuando el usuario se desconecta manda otro script para volver a actualizar el estado.

```
portalgame> db.usuarios.find({game:"Limousine Nightmare"})
[
  {
    _id: ObjectId('666f4519ea1977ab170d3423'),
    name: 'limousine2',
    password: ' ',
    game: 'Limousine Nightmare',
    state: 'connect'
  },
  {
    _id: ObjectId('666f4536ea1977ab170d3424'),
    name: 'limousine1',
    password: ' ',
    game: 'Limousine Nightmare',
    state: 'disconnect'
  }
]
portalgame>
```

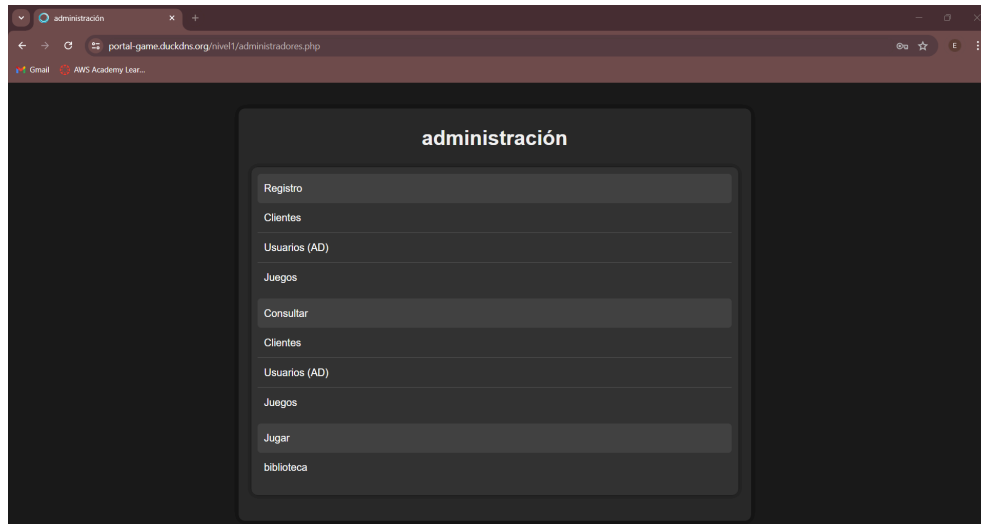
[Imagen de la base de datos con el estado de los usuarios de AD]

Cuando al elegir un juego no hay usuarios disponibles la página nos manda un mensaje avisando de esto.

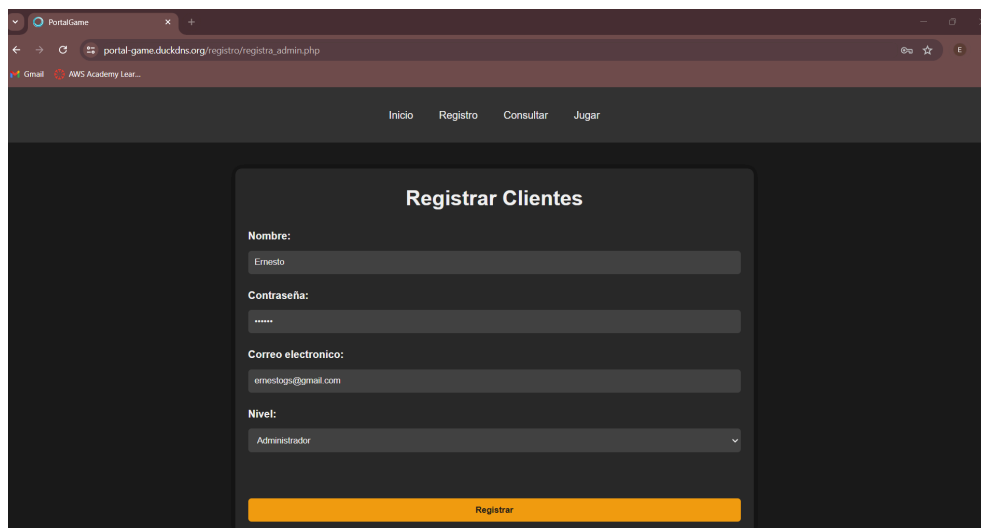


[Mensaje de error ante muchos usuarios conectados]

Los administradores podrán acceder con usuarios específicos a otra página donde podrán crear nuevas cuentas de clientes, nuevas entradas para los juegos y usuarios de AD que funcionen correctamente en el servidor de juego.

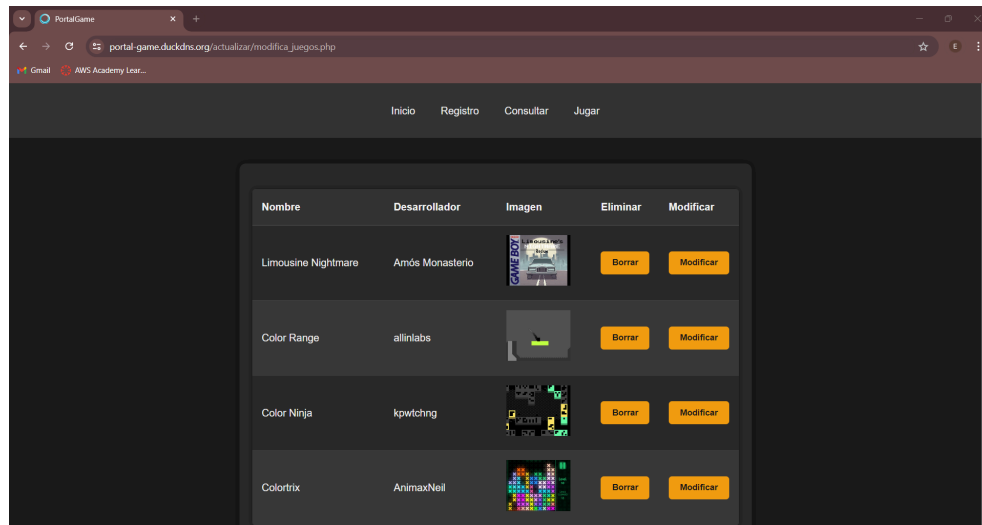


[Imagen de la página de administradores]







[Imagen del formulario de registro de clientes desde administradores]

Además de visualizar las entradas ya existentes.



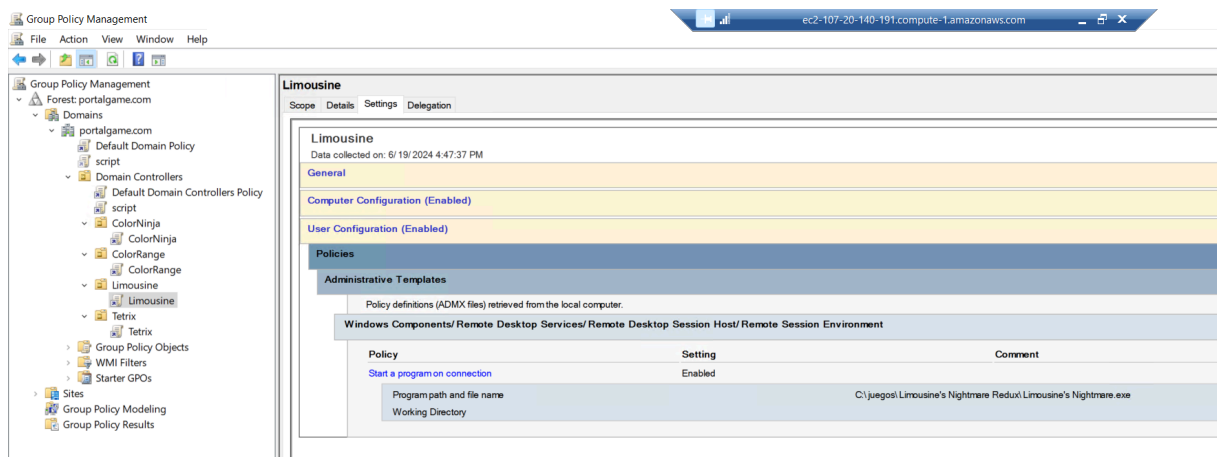
The screenshot shows a web browser window with the URL `portal-game.duckdns.org/actualizar/modifica_juegos.php`. The page displays a table of registered games with the following columns: Nombre, Desarrollador, Imagen, Eliminar, and Modificar. The table contains four entries:

Nombre	Desarrollador	Imagen	Eliminar	Modificar
Limousine Nightmare	Amós Monasterio		Borrar	Modificar
Color Range	allinilabs		Borrar	Modificar
Color Ninja	kpwichng		Borrar	Modificar
Colortrix	AnimaxNeil		Borrar	Modificar

[Tabla de los juegos registrados en la base de datos]

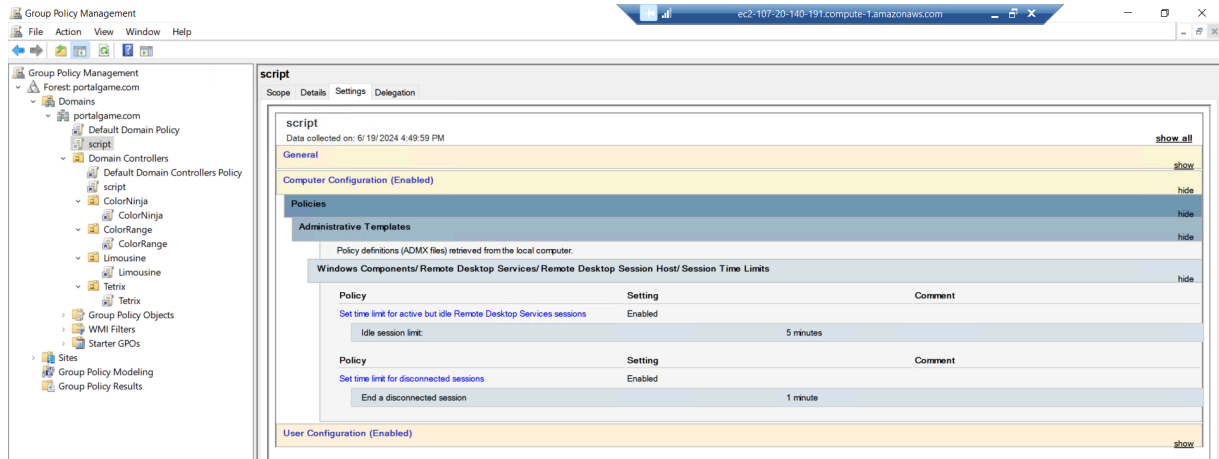
Servidor de Juegos:

El proyecto funciona en mayor parte gracias a la configuración de las GPOs sobre las unidades organizativas, que permite iniciar un juego específico dependiendo del usuario que inicie sesión.



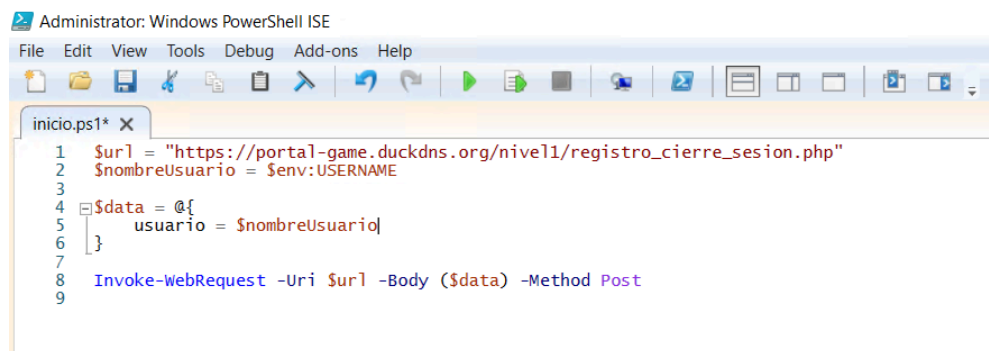
[GPO de inicio de programa para el juego “*Limousine’s Nightmare*”]

A partir de windows server 2019 las conexiones de escritorio remoto que se crean permanecen activas incluso después de desconectarnos, por eso es muy importante establecer límites de tiempo para sesiones desconectadas e inactivas.



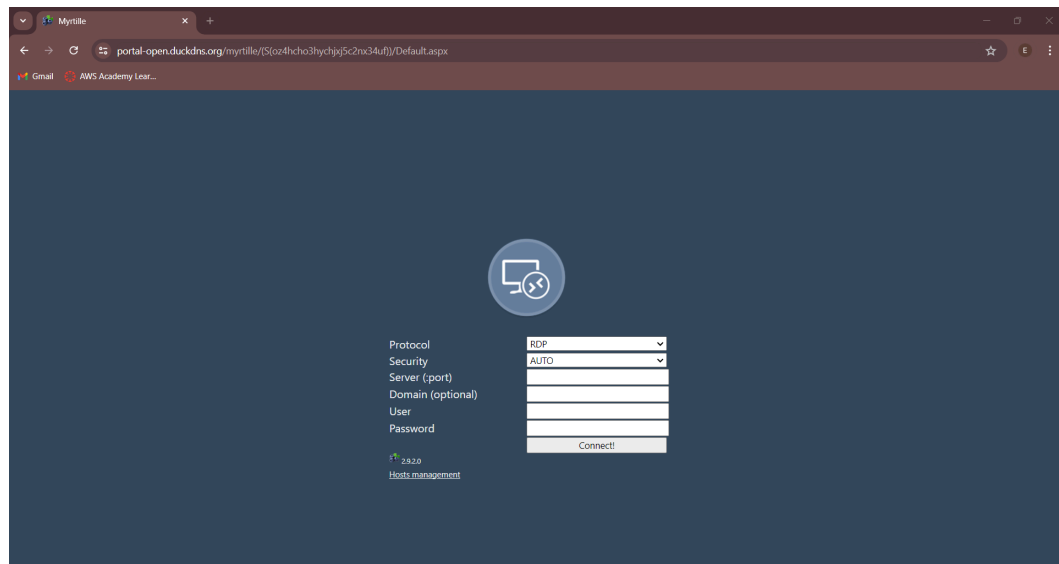
[GPO de límite de tiempo para cuentas desconectadas e inactivas]

Además, para evitar entrar con un usuario ya en uso se han asignado script de inicio y cierre de sesión que actualiza la base de datos para evitar que 2 personas entren con el mismo usuario.



[script de inicio de sesión para actualizar la base de datos]

Por otra parte el software myrtille es fundamental, gracias a su opción de poder iniciar sesión con un usuario desde la url, para evitar pasar por la página de inicio de myrtille.default



[Página de inicio de myrtille por defecto]

Documentación del sistema

Introducción a la aplicación

La aplicación cuenta únicamente con una versión web, esta muestra un catálogo de juegos que abren la ventana del juego elegido.

Manual de Instalación

No requiere ningún tipo de instalación más allá de tener acceso a algún navegador como chrome o firefox.

El acceso a la página debe hacerse buscando directamente en la barra de navegación portal-game.duckdns.org

Manual de usuario.

Debe iniciar sesión con su usuario o correo electrónico y contraseña o debe registrarse en la página con un usuario, contraseña y correo electrónico.

Una vez que se haya iniciado sesión se verá una lista con todos los juegos disponibles, a elección del usuario. Después de elegir, el servidor web buscará algún usuario del servidor de juegos disponible para ese juego, cuando se encuentre se conectará directamente. En el caso de no haber ningún usuario disponible se redirigirá a la página de selección de juegos con un mensaje de advertencia que indica que no hay usuarios disponibles en ese momento.

Manual de administración

Debe iniciar sesión con una cuenta de administrador, al hacerlo se redirigirá a una página de administración donde podrá gestionar los elementos de la base de datos.

El apartado de Registro permite acceder a registrar nuevos clientes o cuentas de administrador, registrar usuarios de active directory o registrar nuevos juegos en la base de datos.

El apartado de Consultar permite observar los datos de las tablas de la base de datos, borrar entradas o modificarlas.

También permite dirigirte a cualquier página de administración desde un menú de selección en la cabecera.

Por último desde la ventana de administración también se puede acceder a la selección de juegos para su visualización.

Conclusiones finales.

He desarrollado un sistema que integra múltiples tecnologías para proporcionar una plataforma de juegos en la nube. Este proyecto busca demostrar un enfoque integral hacia la creación de una infraestructura escalable y segura.

El proyecto es una plataforma de juegos en la nube eficiente y segura con para las necesidades del usuario. Con la posibilidad de poder continuar mejorando y expandiendo las funcionalidades para adaptarte a nuevas demandas y desafíos tecnológicos.

Grado de cumplimiento de los objetivos fijados.

SERVICIOS EN RED

- He usado AWS como nube pública para el proyecto.
- He creado una imagen de docker y archivos para el despliegue de kubernetes.
- Gestiono copias de seguridad usando FTP
- He creado y configurado un servidor web para las necesidades del público que va a acceder a mi proyecto.

APLICACIONES WEB

- He utilizado el lenguaje php para la creación de una página web con HTML.
- He creado un CSS para aplicar un estilo al diseño web.
- Me he conectado a la base de datos que utilizo mediante php.
- Realizo consultas CRUD desde las páginas php.
- Permito el uso de imágenes en la página web.

SISTEMAS GESTORES DE BASES DE DATOS

- Conozco la arquitectura y el funcionamiento del SGBD elegido para el proyecto.
- Implanto métodos de control de acceso de usuarios haciendo uso de perfiles.

SEGURIDAD INFORMÁTICA

- He identificado y securizado los protocolos de los servicios utilizados en el proyecto.
- He configurado redes privadas virtuales en AWS para las conexiones entre las máquinas durante el proyecto.
- He configurado un grupo de seguridad en AWS a modo de cortafuegos para las máquinas del proyecto.
- He analizado ámbitos donde se hace necesario implementar soluciones de alta disponibilidad.

SISTEMAS OPERATIVOS

- He usado los recursos del Sistema Operativo e instalado los servicios necesarios para la correcta funcionalidad del proyecto.

Modificaciones o ampliaciones futuras del sistema implementado.

- Autenticación en 2 pasos y verificación de correo electrónico.
- Página de consulta del perfil personal.
- Buscador de juegos por nombre o desarrollador.
- Página de consulta de juegos con posibilidad de comentar opiniones.
- Página para desarrolladores:
 - Opción a permitir que su juego sea visible solo a ciertos usuarios.
- Herramientas para administradores:
 - Buscador para los diferentes elementos de la base de datos.
 - Permitir la opción de deshabilitar cuentas de los clientes.

- Permitir la opción de borrar comentarios en las opiniones de los juegos.
- Agilizar el proceso de implementación de los juegos en el servidor.
- Integración correcta de kubernetes en el puerto 443

Bibliografía empleada.

Remote Visualization Demo for HPC using NICE DCV on AWS. Amazon Web Services, Inc.

<https://aws.amazon.com/es/hpc/dcv/>

NICE DCV. AWS Documentation.

https://docs.aws.amazon.com/es_es/dcv/

Moonlight Game Streaming Project. GitHub.

<https://github.com/moonlight-stream>

cedrozor/myrtille: A native HTML4 / HTML5 Remote Desktop Protocol and SSH client. GitHub.

<https://github.com/cedrozor/myrtille>

ErnesGS/PortalGame. GitHub.

<https://github.com/ErnesGS/PortalGame/tree/master>

PHP: MongoDB - Manual.

<https://www.php.net/manual/es/set.mongodb.php>

CRUD Operations - PHP Library Manual V1.19.

<https://www.mongodb.com/docs/php-library/current/tutorial/crud/>

Install MongoDB Community Edition on Debian - MongoDB Manual V7.0.

<https://www.mongodb.com/docs/manual/tutorial/install-mongodb-on-debian/#std-label-install-mdb-community-debian>

Introduction - Composer.

<https://getcomposer.org/doc/00-intro.md#installation-linux-unix-macos>

CertBot Instructions. Certbot.

<https://certbot.eff.org/instructions>