Institut Puig Castellar
Santa Coloma de Gramenet

GameServer

Projecte de desenvolupament

CFGS Administració de Sistemes Informàtics i Xarxes

Rubén Castillejos y Izan Hernández
ASIX2B
2025-2026

A) Creative Commons:

Resum del projecte (màxim 250 paraules):

Ha de quedar el següent:

- *Temàtica del projecte.*
- *Objectiu del projecte.*
- *Metodologia seguida per a aconseguir l'objectiu.*
- *Resum de les conclusions.*

Este proyecto llamado GameServer, consiste en el desarrollo de una página web que permitirá a los usuarios crear y gestionar servidores de Minecraft de forma sencilla. Para ello, se utilizarán tres máquinas virtuales: una destinada al servidor de Minecraft ,la segunda máquina será para alojar la página web donde si quieres realmente crear un servidor tendrás que registrarte para poder interaccionar. También tendrá nuestra base de datos donde se guardará la información de los usuarios que se registren y los servidores que se creen, y la última se utilizará para las copias de seguridad. Nuestro objetivo principal es comprobar el funcionamiento del sistema y asegurar que el servidor pueda soportar los mods seleccionados. En un inicio, los propios desarrolladores serán los usuarios finales para verificar que todo funciona correctamente. Sin embargo, en un futuro se plantea la posibilidad de abrir la plataforma a más usuarios para que puedan crear sus propios servidores de Minecraft. Nuestra metodología para conseguir el objetivo es ir probando poco a poco cosas nuevas que vamos viendo al investigar para poder hacerlo correctamente, si vemos que hay algunos errores pues nos paramos a investigar el porqué ha fallado y hacemos lo posible para poder solucionar todos los problemas que nos van surgiendo a lo largo del proyecto. Nuestras conclusiones están claras, nosotros no nos esperábamos que fuera un trabajo fácil, pero tampoco nos esperábamos que nos dieran más errores que aciertos y nos sacó mucho de quicio, pero poco a poco acabamos solucionando los problemas.

Paraules clau (entre 4 i 8):

Han de servir per a trobar el projecte si fem servir un cercador

Página web

Crear

Servidores

Interaccionar

Usuarios

Errores

Abstract (in English, 250 words or less):

This project, called GameServer, consists of the development of a website that will allow users to create and manage Minecraft servers easily. To achieve this, three virtual machines will be used: one dedicated to the Minecraft server, the second machine will host the website where users must register in order to interact and

actually create a server. It will also contain our database, where the information of registered users and the created servers will be stored, and the last machine will be used for backups. Our main objective is to test the system's functionality and ensure that the server can support the selected mods. At first, the developers themselves will be the end users in order to verify that everything works correctly. However, in the future, there are plans to open the platform to more users so they can create their own Minecraft servers. Our methodology for achieving this objective is to gradually test new things that we discover while researching how to implement them correctly. If we encounter errors, we stop to investigate why they happened and do our best to solve all the problems that arise throughout the project. Our conclusions are clear: we did not expect this project to be easy, but we also did not expect to encounter more errors than successes, which was sometimes very frustrating. However, little by little, we managed to solve the problems.

Keywords (entre 4 i 8):

Website

Create

Servers

Interact

Users

Errors

Índex

Llista de figures

1 Introducció

El presente proyecto consiste en el desarrollo de una página web que permitirá a los usuarios crear y gestionar servidores de Minecraft de forma sencilla. Para ello, se utilizarán tres máquinas virtuales: una destinada al servidor de Minecraft donde nosotros podremos ver gracias a Docker si se ha guardado correctamente y está encendido. La siguiente máquina será para alojar la página web, el cual, para poder crear los servidores que los usuarios quieran tendrán que registrarse, ya que sino solo podrán entrar como visitante y solo verán nuestra introducción donde nosotros explicamos quienes somos y como “posible ayuda” a usuarios que vayan a jugar por primera vez al Minecraft, pondremos varios “pro tips” para que ya tengan una idea. Esta segunda máquina también tendrá nuestra base de datos la cual se guardará todos los datos de los usuarios y los servidores que ellos creen, si han usado mods y en el caso que hayan usado cuales y cuantos han usado, también guardaremos en que versión y si es Vanilla o Forge. Y una tercera dedicada a las copias de seguridad, con el fin de evitar la pérdida de datos durante el desarrollo o uso del sistema.

La página web incluirá un diseño con un menú informativo, un logo personalizado y diferentes apartados que permitirán a los usuarios seleccionar la versión del juego, añadir mods compatibles y gestionar su servidor (encenderlo o apagarlo). Además, se incorporarán medidas como las copias de seguridad para garantizar la seguridad del proyecto.

El objetivo principal es comprobar el funcionamiento del sistema y asegurar que el servidor pueda soportar los mods seleccionados. En un inicio, los propios desarrolladores serán los usuarios finales para verificar que todo funciona correctamente. Sin embargo, en un futuro se plantea la posibilidad de abrir la plataforma a más usuarios para que puedan crear sus propios servidores de Minecraft.

La solución propuesta se basa en el desarrollo progresivo del proyecto, evaluando su funcionamiento en cada fase para garantizar que tanto el servidor, la página web, la base de datos y el sistema de copias de seguridad funcionen correctamente.

1.1 Context

Actualmente, la creación y gestión de servidores de Minecraft puede resultar compleja para usuarios sin conocimientos técnicos, especialmente cuando se trata de instalar versiones, mods o gestionar el estado del servidor. Este proyecto surge en un entorno académico con recursos limitados, donde se dispone de un equipo con 32 GB de RAM y tres máquinas virtuales para desarrollar una solución funcional. Además, se plantea la necesidad de comprobar el rendimiento del servidor ante el uso de mods y múltiples jugadores, así como garantizar la seguridad mediante copias de seguridad.

1.2 Justificació

Este proyecto es relevante porque permite aprender y aplicar conocimientos sobre desarrollo web, gestión de servidores y uso de máquinas virtuales. Además, responde a la necesidad de simplificar la creación de servidores de Minecraft mediante una interfaz accesible. También es importante porque permite comprobar la viabilidad

técnica del sistema, especialmente en cuanto al rendimiento del servidor y la compatibilidad de mods.

1.3 Objectius

1.3.1 Objectiu general

Desarrollar una página web para todos los públicos que permita crear contenedores separados para cada usuario y que puedan gestionar sus servidores de Minecraft de forma sencilla, asegurando su correcto funcionamiento incluyendo copias de seguridad de los servidores.

1.3.2 Objectius específics

Permitir el registro e inicio de sesión de usuarios.
Crear una interfaz web para gestionar servidores.
Permitir seleccionar versiones de Minecraft y el tipo de versión que quieres utilizar
Permitir añadir y gestionar mods compatibles.
Que estos mods se puedan descargar desde la página.
Encender, apagar y reiniciar servidores desde la web.
El servidor arranca con lo que los usuarios han escogido en la web.
Implementar copias de seguridad para evitar pérdida de datos.
Comprobar el rendimiento del servidor con diferentes configuraciones.

1.4 Estratègia i planificació del projecte

La estrategia consiste en desarrollar un producto nuevo desde cero utilizando tecnologías web y herramientas de gestión de servidores. Se ha optado por esta estrategia porque permite adaptar completamente el sistema a las necesidades del proyecto. El desarrollo se realizará de forma progresiva, probando cada parte para comprobar su funcionamiento antes de continuar. Ya que si nos encontramos varios errores poder dedicarle el tiempo suficiente para solucionar todos los problemas que nos vayan surgiendo y nuevas funcionalidades que podamos añadir al proyecto y que pueda beneficiarlo, pero, los errores es muy importante poder dedicarle un buen tiempo y con calma ya que de un error te pueden salir más de 5 si no los miras con claridad y sin informarte primero de como solucionarlo.

1.5 Metodologia de treball

Nuestra metodología de trabajo es clara y concisa, es ir progresivamente cada día trabajando en el proyecto y evitando los errores, pero, si nos encontramos los errores nos paramos a dedicarnos a solucionar los errores y por ellos nos vamos informando para ello. Creemos que esta metodología es la más apropiada para nuestro proyecto ya que al tener servidores, una página web, una base de datos y

copias de seguridad que están en distintas máquinas el riesgo de perderlo todo puede ser muy alto. Y por ello después de comentarlo los dos decidimos ir progresivamente y cada día proponernos avanzar poco a poco y guardando las cosas en un documento llamado “Cosas del Servidor” por si acaso se nos perdía todo.

Nosotros trabajaremos para ir viendo cómo vamos progresando y mejorando dia a día con nuestro diagrama de Gantt, que va sobre un poco desde que decidimos que íbamos de hacer de proyecto hasta acabando con la memoria y la presentación final.

Trabajaremos también con Trello que será más para cosas puntuales y cosas más técnicas. Nos servirá para ver que pusimos el día de antes y cosas puntuales como preguntar dudas a nuestros profesores de proyecto y que nos aconsejen. Tener como objetivo un día acabar una parte del proyecto y arrancar con otra parte

Diagrama de Gantt.xlsx

https://trello.com/invite/b/6a0347c85c610b814bfbe61f/ATTI6ae0c4dc480ef208ce2c03f5ab3305702B6D5810/my-trello-board

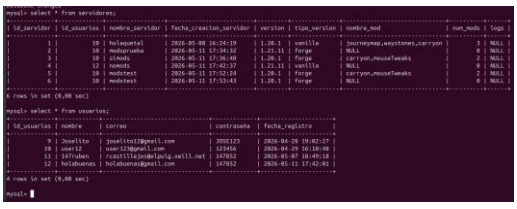
1.6 Estudi econòmic i pressupostari

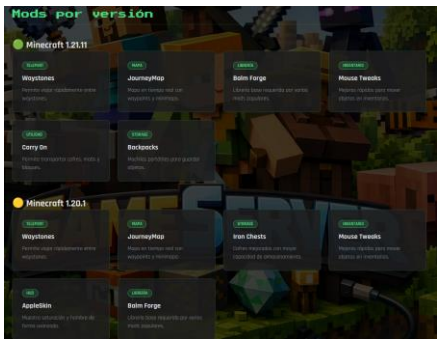
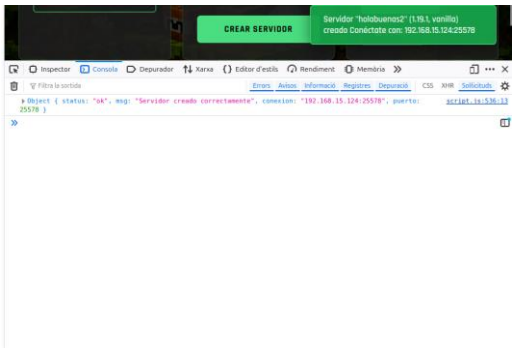
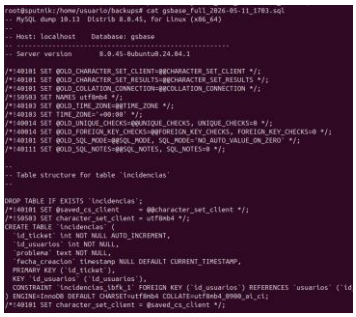
El proyecto se desarrolla con recursos ya disponibles, como un ordenador del instituto con 32 GB de RAM y tres máquinas virtuales. No se requiere inversión en hardware adicional. Los principales costes serían el tiempo de desarrollo y mantenimiento del sistema. El mantenimiento incluye la gestión de servidores, copias de seguridad y posibles mejoras. En cuanto a beneficios, en el futuro se podría abrir la plataforma a otros usuarios para que utilicen el servicio. Pero de momento todo lo que vamos a utilizar es totalmente gratuito.

2 Descripció del projecte

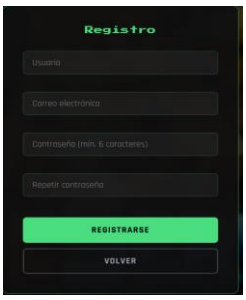
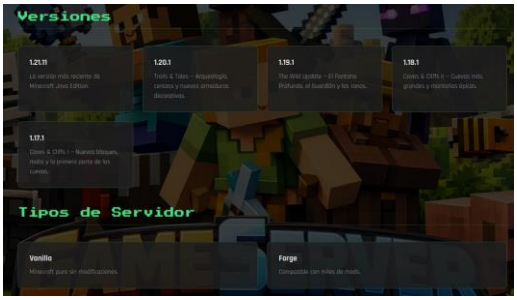
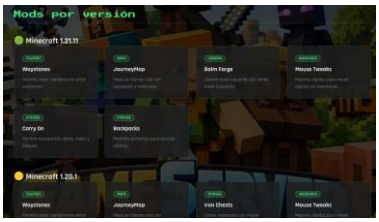
2.1 Anàlisi de requisits

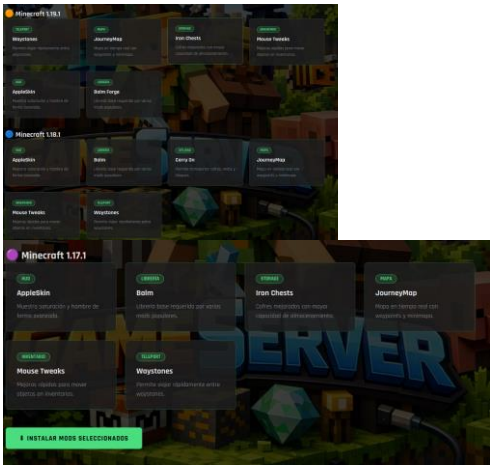
Esta es nuestra lista para dar como finalizado el proyecto:

Conectar las tres máquinas entre sí para poder hacer más fácil el trabajo	
Que nuestra base de datos esté bien conectada con nuestra página para poder guardar la información de los usuarios que se registran y los usuarios que crean.	

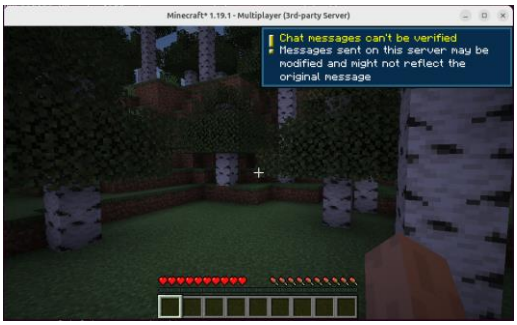
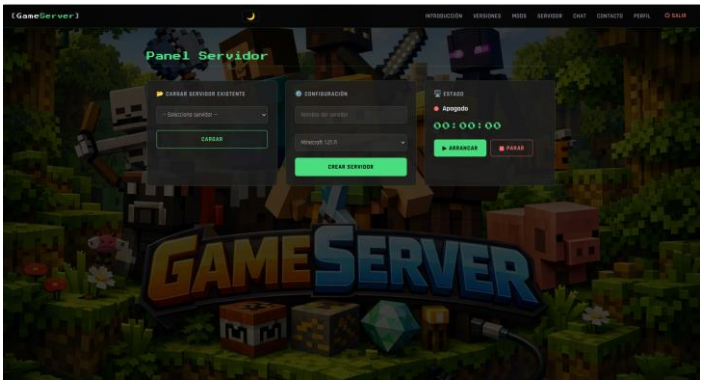
Todas las funciones que hemos creado como por ejemplo: cargarServidoresExistentes, crearServidor, arrancarServidor, apagarServidor, que las opciones de escoger version, tipo de versión y los mods funcionen correctamente	
Una vez que se crea el servidor salga un mensaje con la ip que tienes que poner y el puerto para poder jugar correctamente a tu servidor. Básicamente que se pueda controlar desde la página todo el tema del servidor.	
Una vez nos ha salido el mensaje entrar al Minecraft en la versión correspondiente y entrar al servidor. Asi podrás disfrutar del mundo que has creado.	
Que las copias de seguridad funcionen correctamente y poder hacer una copia de nuestra base de datos	

2.1.1 Requisits funcionals

Registro de usuarios.	
Inicio de sesión y gestión de perfil.	
Mostrar información del servidor del usuario.	
Encender y apagar servidores desde la web.	
Seleccionar versiones de Minecraft.	
Añadir y gestionar mods.	

	
Crear copias de seguridad.	
Separar servidores mediantes contenedores.	

2.1.2 Requisits no funcionals

Acceso desde dispositivos móviles.	
Buen rendimiento del Servidor. Los servidores todos en general funcionan correctamente, buenos graficos, no da esos típicos tirones que fastidian	
Facilidad de uso de la interfaz. Creemos que hemos conseguido una interfaz bastante fácil de entender y de usar prácticamente	

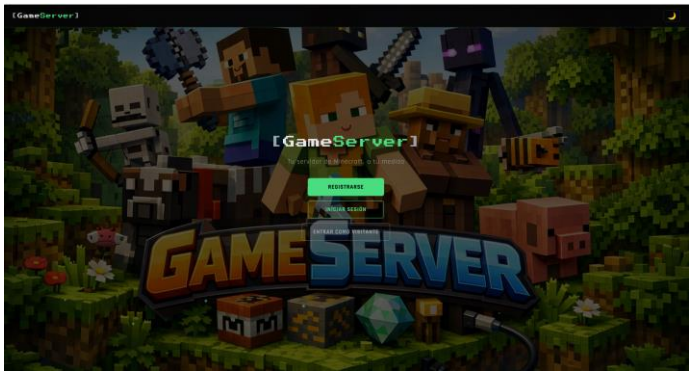
2.1 Previsió de tasques d'investigació [projecte d'investigació]

Hay varias tareas que nos han hecho ir por otro camino o dejar de pensar en otras opciones que añadir como por ejemplo los errores que nos daban mientras hacíamos lo principal que queríamos hacer. Entonces tomamos la decisión de tirar

para adelante con lo que teníamos desde un principio y ya veríamos si nos da tiempo.

Después de esto, esta es la lista detallada de las tareas:

- Comprender más a fondo cómo funciona el tema de la creación de una página web. El tema del html y procurar no tener ningún fallo a la hora de entrar a la dicha página, el css para modificar como queramos y añadirle color a la web y por último el tema del archivo js, ya que este era el que menos habíamos tocado y más tuvimos que investigar sobre ello . Sobre como poner las funciones, como conectar esas funciones con la máquina del hosting y cómo hacer que funcione correctamente en la web.



- Investigar sobre cómo hacer que se conecte nuestra base de datos con nuestra página. Esto había que profundizar un pelín más ya que era reciente cuando lo comenzamos a hacer en clase y teníamos pocos conocimientos sobre ello. Decidimos hacer a partir de php conectar nuestra base de datos con nuestra web, a partir de un archivo donde pusimos el host de donde estaba la base de datos, el usuario, la contraseña del usuario y el nombre de nuestra base de datos.

```php
<?php
$host = "192.168.15.123";
$usuario = "gsuser";
$contraseña = "1234Gs@";
$base_datos = "gsbase";

$conn = new mysqli($host, $usuario, $contraseña, $base_datos);

if ($conn->connect_error) {
    die(json_encode(["status" => "error", "msg" => "Error de conexión: " . $conn->connect_error]));
}
?>
```

-Quisimos instalar phpMyAdmin para gestionar mejor la base de datos MySQL, pero al final no lo hicimos debido a problemas durante la instalación. phpMyAdmin requería instalar Apache, mientras que nosotros utilizábamos Nginx como servidor web. Intentamos adaptarlo para que funcionara con Nginx, pero no encontramos una forma adecuada de configurarlo.

Finalmente, decidimos no instalarlo, ya que nuestra base de datos era bastante sencilla y, además, no queríamos cambiar a Apache porque nuestro servidor web con Nginx ya estaba muy avanzado.

Aquí dejamos una captura de pantalla del resultado final de nuestra base de datos.

```
mysql> SHOW TABLES;
+------------------+
| Tables_in_gsbase |
+------------------+
| incidencias      |
| respuestas       |
| servidores       |
| usuarios         |
+------------------+
4 rows in set (0,00 sec)
```

- Al principio comenzamos a debatir cómo podíamos crear los contenedores para los servidores, decidimos que queríamos experimentar con Docker ya que vimos que era para nosotros más cómodo y más completo que lxc. Queríamos experimentar si la conexión entre la máquina virtual, que en este caso era la de hosting, y en el local donde nos descargamos el Launcher para poder comprobar si funcionaba correctamente. Después cuando ya teníamos casi todo sobre la web y la base de datos comenzamos a informarnos sobre como tener la información de los servidores en nuestra base de datos nuestra web. Esto nos dio bastantes errores pero lo conseguimos hacer correctamente.

Y en esta foto se puede ver como vemos en nuestra base de datos la información de los servidores.

```
mysql> select * from servidores;
+-------------+-------------+-----------------+-------------------------+---------+-
------------+-------------------------------+----------+------+
| id_servidor | id_usuarios | nombre_servidor | fecha_creacion_servidor | version |
tipo_version | nombre_mod                    | num_mods | logs |
+-------------+-------------+-----------------+-------------------------+---------+-
------------+-------------------------------+----------+------+
|           1 |          10 | holaquetal      | 2026-05-08 16:24:19     | 1.20.1  |
vanilla      | journeymap,waystones,carryon |        3 | NULL |
```

2.2 Tecnologies

2.2.1 Comparativa de les tecnologies valorades

Se han considerado diferentes herramientas para desarrollo web, la gestión de servidores y para la automatización de ellos. Para gestionar los servidores y crear cada servidores en un contenedor distinto utilizamos Docker pero estuvimos comparándolo con lxc. Para empezar vimos que tenían diferentes objetivos totalmente distinto que son:

- Docker: Está más pensado para ejecutar aplicaciones y nos parecía mucho mejor en tema gestionar servidores con su versión, con sus mods. Es decir, estuvimos comparándolos y vimos que el más interesante para nuestro proyecto era Docker.

- Lxc: Está pensado para poder ejecutar sistemas Linux completos y nosotros no queríamos un sistema operativo totalmente completo, nosotros queríamos poder gestionar nuestros servidores en contenedores diferentes y no para tener un Debian completo.

A la hora del desarrollo web lo tuvimos bastante claro cuál utilizar, que en este caso hemos usado: HTML, CSS, JavaScript y php para poder conectar nuestra base de datos con la web.
Donde todo esto está dentro de NGINX. Nos lo comentó David Garcia que era mejor hacerlo de esta manera y tuvo razón, es un servidor web bastante rápido ya que los entrega directamente al navegador.

- HTML: HTML es un lenguaje de marcado que se utiliza para crear estructurar el contenido de una página web, es decir, es lo primero que ves al buscar una web.

- CSS: Este es un lenguaje de estilos ya que gracias al css se puede diseñar a tu gusto la web, puedes modificar el color de las letras, poner una imagen de fondo, usar unas animaciones para ciertas cosas. Si quieres hacer una web necesitas si o si html y css.

- JavaScript: JavaScript es un lenguaje de programación que añade interactividad y lógica, es decir, la creación de las funciones de una página web como crear un servidor, que una vez acabes de registrarse te lleve a la introducción de la web. Sirve más que nada para las funciones, los botones dinámicos como registrar, iniciar sesión, visitante y volver.

- PHP: Php es un lenguaje de programación backend ya que este procesa los datos en el servidor, gracias a él hemos conseguido conectar nuestra web con la base de datos, también hemos conseguido que el login se haga correctamente y sobre todo para la creación de los servidores.

- Nginx: Es un software que puede actuar de diversas maneras, pero nosotros lo utilizamos para servidor web. Ya que nos informamos antes de todo y vimos que funcionaba más rápido y hacía que nuestra web pudiera soportar muchas visitas al mismo tiempo.

Para la base de datos no hubo debate, decidimos utilizar MySQL, ya que nos parecía más sencillo de utilizar porque durante el curso pasado y este lo hemos ido utilizando.

- MySQL: Este sistema de datos suele usarse para el desarrollo web, especialmente junto con php. Este sirve para guardar, organizar y consultar los datos de nuestros usuarios, las contraseñas, los servidores que han creado los diferentes usuarios que se han registrado en nuestra web, en que versión y que mods han creado su servidor y sobretodo el nombre que le han puesto a su servidor. Es fácil de aprender, es bastante rápido a la hora de crear tablas y hacer consultas y es compatible con php.

Para la automatización de los servidores utilizamos Ansible, ya que lo teniamos más reciente y es exactamente para lo que nosotros lo queríamos utilizar

- Ansible: Ansible es una herramienta de automatización que se usa para administrar servidores, que era lo que nosotros queríamos hacer. Gracias a ansible hicimos un playbook donde cada vez que se iniciaba un servidor se creará una carpeta del dicho servidor, con su carpeta con los datos del servidor, que saliera el nombre, la versión, en que tipo de versión están jugando (es decir, o Vanilla o Forge).

```yaml
- hosts: localhost
  become: yes
  vars:
    base_path: "/mnt/minecraft/minecraft"

  tasks:

    - name: Crear carpeta del servidor
      file:
        path: "{{ base_path }}/{{ nombre }}"
        state: directory

    - name: Crear carpeta de datos
      file:
        path: "{{ base_path }}/{{ nombre }}/data"
        state: directory

    - name: Crear docker-compose.yml
      copy:
        dest: "{{ base_path }}/{{ nombre }}/docker-compose.yml"
        content: |
          version: "3"
          services:
            {{ nombre }}:
              image: itzg/minecraft-server
              container_name: {{ nombre }}
              ports:
                - "{{ puerto }}:25565"
              environment:
                EULA: "TRUE"
                VERSION: "{{ version }}"
                TYPE: "{{ tipo }}"
                ONLINE_MODE: "FALSE"
              volumes:
                - "{{ base_path }}/{{ nombre }}/data:/data"

    - name: Iniciar servidor
      command: docker compose up -d
      args:
        chdir: "{{ base_path }}/{{ nombre }}"
```

2.2.2 Tecnologies escollides

- Ansible: Automatizar la configuración y administración de servidores para poder desplegar nuevas instancias rápidamente, mantener todo actualizado, gestionar plugins y configuraciones de manera uniforme y controlar entornos con varios servidores usando playbooks.

- MySQL: Almacenar y sincronizar eficientemente grandes cantidades de datos de jugadores y plugins (inventarios, experiencia, registros, etc.) entre diferentes servidores, mejorando el rendimiento y permitiendo funcionalidades avanzadas en redes de servidores (networks), como compartir inventarios o economías, lo que reemplaza el almacenamiento local más lento por una base de datos centralizada y organizada.

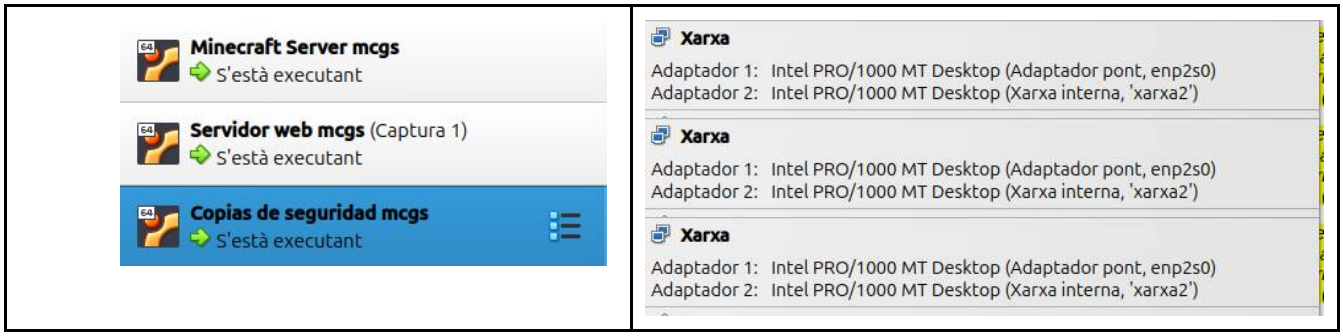
- Php: PHP es un lenguaje de programación usado principalmente para el desarrollo web del lado del servidor. Permite crear páginas dinámicas, procesar formularios, gestionar usuarios y conectarse a bases de datos como MySQL. El código PHP se ejecuta en el servidor y luego envía el resultado al navegador en forma de HTML.

- Html y Css: Aunque el servidor sea solo para activar mods, uso HTML y CSS para crear un Panel de Control personalizado y una Guía Visual de Instalación; con HTML organizó el listado de mods y sus requisitos para que los jugadores sepan qué descargar, y con CSS diseño una interfaz sencilla e intuitiva que les permite ver el estado del servidor (si está activo o en mantenimiento) sin tener que entrar a la consola técnica, haciendo que la gestión del servidor sea mucho más clara para todos.

- JavaScript: JavaScript es un lenguaje de programación que permite añadir interactividad y dinamismo a las páginas web. Se ejecuta principalmente en el navegador y se utiliza para crear menús dinámicos, validar formularios, animaciones y aplicaciones web. Es una de las tecnologías fundamentales del desarrollo web moderno junto con HTML y CSS.

- Nginx: Es un servidor web muy rápido y ligero que utilizamos para alojar nuestra página y gestionar todas las peticiones que hacen los usuarios. Se encarga de mostrar la web, enviar los archivos al navegador y trabajar junto con PHP- FPM para que el backend funcione correctamente. Es ideal para nuestros proyectos porque consume pocos recursos y responde muy rápido, permitiendo que la página cargue sin problemas incluso cuando se realizan acciones como crear servidores o consultar la base de datos.

2.3 Estructura del projecte



El proyecto se divide en tres máquinas virtuales que están conectadas entre sí y están estructuradas de la siguiente manera:

- La primera máquina que se llama Minecraft Server mcgs contiene nuestros servidores de Minecraft en diferentes contenedores. También contiene el backend que sirve para que funcione las diferentes funcionalidades que tenemos en el js. Tenemos un ansible el cual cada vez que se crea un servidor, se le crea inmediatamente su contenedor.

- La segunda máquina que se llama Servidor web mcgs contiene nuestra web, que tenemos subida a nginx para que vaya más rápido y a la hora de soportar a mucha gente enviando solicitudes para registrarse o para crear servidores. Hemos creado el usuario visitante que este solo podrá acceder a la introducción, si quiere interaccionar con los otros apartados tendrá que registrarse con nosotros y así poder crear los servidores que quiera.

También contiene nuestra base de datos, la cual contiene la información de todos los usuarios que se han registrado con su usuario, su contraseña, su correo electrónico, los servidores que han creado con su nombre, el tipo de versión que están jugando, es decir, si están jugando con mods o sin mods, si contiene mods en la base de datos saldrá que mods y cuantos ha añadido.

- Y la última máquina como bien indica su nombre será dirigida para las copias de seguridad. Será la encargada de hacer copias de seguridad cada tres días a las 2 de la mañana y las copias de seguridad más antiguas cada 30 días se borran automáticamente.

Nosotros lo hemos querido organizar así porque lo veíamos más cómodo y mucha mejor práctica que hacerlo todo en una misma máquina y así provocar que la máquina reviente y perder nuestro proyecto.

2.4 Descripció dels components

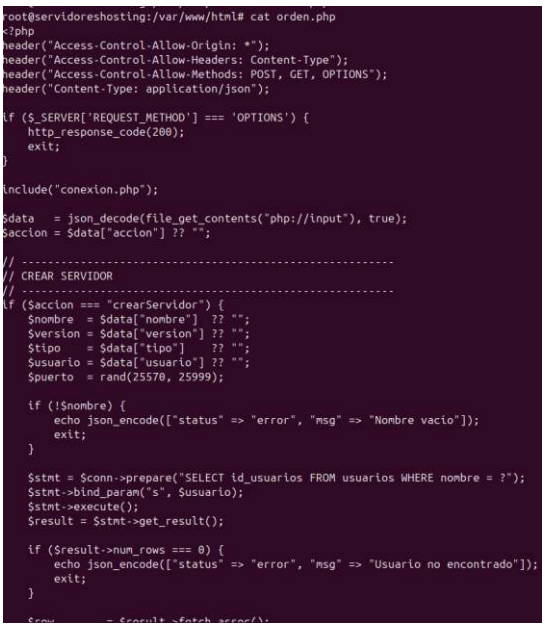
2.4.1 Servidor de Minecraft

Nuestro servidor de hosting consiste en poder tener los servidores en distintos contenedores y que los usuarios puedan crear su servidor con el nombre que quieran y con la versión y los mods que quieran.

Como vemos en la primera imagen es nuestro ansible que hemos creado para que cuando cualquier usuario cree un servidor se le cree automáticamente una carpeta con todos los datos de su servidor donde podrán poner el nombre que ellos quieran al servidor. Una vez está creado el servidor con su carpeta, el puerto del servidor y con su tipo de versión que será con mods o sin mods en la última parte de nuestro ansible hemos puesto que se inicie automáticamente el servidor para que nosotros lo veamos utilizando el comando “docker ps”. Y evidentemente lo veremos en nuestra base de datos que eso lo explicaremos en el siguiente apartado.

En la segunda imagen que aparece se ve un cacho de nuestro backend llamado “orden.php”. Este backend lo que hace es hacer posible que funcionen las diferentes funcionalidades que tiene nuestra web como son: crearServidor, arrancarServidor, apagarServidor, cargarServidoresExistentes, cargarModsInstlados. Y esto se hace posible gracias al backend que hace la función de hacerlo todo “detrás de todo, lo que nadie ve”, es decir, actúa desde cosas que no vemos nosotros cómo hacer funcionar las funciones

```yaml
- hosts: localhost
  become: yes
  vars:
    base_path: "/mnt/minecraft/minecraft"

  tasks:

    - name: Crear carpeta del servidor
      file:
        path: "{{ base_path }}/{{ nombre }}"
        state: directory

    - name: Crear carpeta de datos
      file:
        path: "{{ base_path }}/{{ nombre }}/data"
        state: directory

    - name: Crear docker-compose.yml
      copy:
        dest: "{{ base_path }}/{{ nombre }}/docker-compose.yml"
        content: |
          version: "3"
          services:
            {{ nombre }}:
              image: itzg/minecraft-server
              container_name: {{ nombre }}
              ports:
                - "{{ puerto }}:25565"
              environment:
                EULA: "TRUE"
                VERSION: "{{ version }}"
                TYPE: "{{ tipo }}"
                ONLINE_MODE: "FALSE"
              volumes:
                - "{{ base_path }}/{{ nombre }}/data:/data"

    - name: Iniciar servidor
      command: docker compose up -d
      args:
        chdir: "{{ base_path }}/{{ nombre }}"
```



```
root@servidoreshosting:/var/www/html# cat orden.php
<?php
header("Access-Control-Allow-Origin: *");
header("Access-Control-Allow-Headers: Content-Type");
header("Access-Control-Allow-Methods: POST, GET, OPTIONS");
header("Content-Type: application/json");

if ($_SERVER['REQUEST_METHOD'] === 'OPTIONS') {
    http_response_code(200);
    exit;
}

include("conexion.php");

$data   = json_decode(file_get_contents("php://input"), true);
$accion = $data["accion"] ?? "";

// ------------------------------------------------------
// CREAR SERVIDOR
// ------------------------------------------------------
if ($accion === "crearServidor") {
    $nombre  = $data["nombre"]  ?? "";
    $version = $data["version"] ?? "";
    $tipo    = $data["tipo"]    ?? "";
    $usuario = $data["usuario"] ?? "";
    $puerto  = rand(25570, 25999);

    if (!$nombre) {
        echo json_encode(["status" => "error", "msg" => "Nombre vacio"]);
        exit;
    }

    $stmt = $conn->prepare("SELECT id_usuarios FROM usuarios WHERE nombre = ?");
    $stmt->bind_param("s", $usuario);
    $stmt->execute();
    $result = $stmt->get_result();

    if ($result->num_rows === 0) {
        echo json_encode(["status" => "error", "msg" => "Usuario no encontrado"]);
        exit;
    }

    $row    = $result->fetch_assoc();
```

2.4.2 Servidor web y base de datos

Nuestro servidor web estará subido en nginx para facilitar el trabajo y que sea más rápido y que pueda soportar bastantes solicitudes a la vez. En esta máquina tendremos los archivos suficientes para poder conectar nuestra base de datos con la web. Los archivos son los siguientes: en conexión.php será el archivo principal, es el archivo donde guardará la información de nuestra base de datos creada, el nombre de la base datos, el usuario con el que debería entrar, donde se sitúa la base de datos y su contraseña. En el archivo login.php y registrar.php evidentemente tendrá que tener toda la información sobre nuestra base de datos para que se pueda guardar correctamente la información de cuando se registren y se loguea incluyendo el archivo conexion.php, ya que si este no se pone no será posible.

El archivo index.nginx-debian.html tendrá nuestra web, ya que es el archivo que viene por defecto cuando se instala NGINX. Su función es actuar como página principal y, al acceder desde el navegador escribiendo **http://la-ip-de-la-máquina**, podremos entrar directamente a la web alojada en el servidor.

La carpeta css contiene el archivo style.css, donde se encuentran todas las modificaciones visuales de la página web. Aquí se configuran aspectos como la imagen de fondo, los colores, el tamaño de la letra y, en general, todo el diseño y estilo de la interfaz.

Por último, la carpeta script contiene las funciones y scripts que hemos ido añadiendo para darle más funcionalidades a la web. Gracias a ello y al backend, es posible realizar acciones como arrancar servidores, crear nuevos servidores o cargar la información de un usuario si ya tiene uno creado. También se han añadido algunos efectos visuales, como partículas animadas que aparecen durante la navegación por la página web.

```
-rw-r--r-- 1 root    root      303 may 15 16:48 conexion.php
-rw-r-xr-x 1 root    root      893 may 15 16:54 copia.sh*
drwxr-xr-x 2 root    root     4096 may 13 19:41 css/
drwxr-xr-x 2 root    root     4096 abr 22 16:33 foto/
-rw-r--r-- 1 root    root    24116 may 13 19:17 index.nginx-debian.html
-rw-r--r-- 1 root    root     2496 abr 28 17:26 index.php
-rw-r--r-- 1 root    root       17 abr 27 16:48 info.php
-rw-r--r-- 1 root    root      843 abr 29 17:00 login.php
drwx------ 7 usuario usuario  4096 may  8 17:22 mods_gs/
-rw-r--r-- 1 root    root     1037 may  7 18:38 registrar.php
drwxr-xr-x 2 root    root     4096 may 15 17:55 script/
root@servidorweb:/var/www/html#
```

La base de datos es la encargada de almacenar toda la información necesaria para el funcionamiento de la página web y de los servidores creados por los usuarios. En ella se guardan datos como los usuarios registrados, las contraseñas cifradas, la información de cada servidor creado, las configuraciones seleccionadas y otros datos importantes para el sistema.

Gracias a la base de datos, la web puede recordar la información de cada usuario cuando inicia sesión y cargar automáticamente sus servidores y configuraciones guardadas. Además, permite que el backend pueda gestionar correctamente acciones como crear servidores, modificarlos o comprobar si un usuario ya dispone de uno creado.

```
mysql> SHOW TABLES;
+------------------+
| Tables_in_gsbase |
+------------------+
| incidencias      |
| respuestas       |
| servidores       |
| usuarios         |
+------------------+
4 rows in set (0,00 sec)
```

2.4. Sistema de copias de seguridad

La máquina de copias de seguridad es un servidor independiente dentro de la red del proyecto cuya función principal es almacenar de forma segura los backups de la base de datos del sistema web. La comunicación entre esta máquina y el servidor web se realiza mediante SSH, lo que permite transferir archivos de forma remota y segura sin necesidad de introducir la contraseña en cada ejecución.

En el servidor web se encuentra el script copia.sh, desarrollado en Bash, que es el encargado de generar las copias de seguridad. Este script utiliza mysqldump para exportar la base de datos MySQL a un archivo temporal con la fecha y hora actual, lo que permite identificar cada backup de forma única. Una vez generado el archivo, se envía automáticamente al servidor de copias mediante rsync a través de la conexión SSH.

Para evitar la introducción manual de contraseñas en cada transferencia, se ha configurado autenticación mediante claves SSH. Para ello se genera un par de claves

pública y privada, y la clave pública se copia en la máquina de copias de seguridad, permitiendo así la conexión automática y segura entre servidores.

Una vez transferido el archivo, el script elimina la copia temporal del servidor web y ejecuta un comando remoto para borrar los backups con más de 30 días de antigüedad, manteniendo así el almacenamiento organizado y evitando acumulación de archivos innecesarios.

La automatización del proceso se realiza mediante crontab, que ejecuta el script cada 3 días a las 2:00 de la madrugada. Gracias a esto, el sistema realiza copias de seguridad de forma totalmente automática sin intervención del usuario.

En conjunto, esta máquina garantiza la seguridad y persistencia de los datos del sistema web, asegurando que siempre exista una copia externa actualizada de la base de datos.

```
drwxrwxr-x 2 usuario usuario 4096 may 13 17:15 ./
drwxr-x--- 5 usuario usuario 4096 may 13 16:20 ../
-rw-r--r-- 1 usuario usuario 5938 may 13 17:15 gsbase_full_2026-05-13_1715.sql
root@sputnik:/home/usuario/backups#
```

2.5 Definició de les tasques

2.5.1 Tarea servidor hosting

En nuestro servidor de hosting ha sido necesario desarrollar algunos componentes propios para poder automatizar mejor el funcionamiento de toda la plataforma. Uno de los más importantes ha sido el backend en PHP, que es el encargado de gestionar las diferentes funciones internas de la web y la comunicación con los contenedores y la base de datos.

Gracias a este backend hemos podido realizar acciones como la creación de servidores, poder arrancarlos y apagarlos desde la web, cargar el servidor o los usuarios que haya creado el usuario y poder gestionar los mods que están instalados.

Hemos desarrollado también un sistema de automatización como es Ansible. Se encarga de preparar automáticamente cada servidor cuando un usuario crea uno nuevo. Lo que hace este Ansible es crear la carpeta del servidor, asignarle un puerto disponible, configurar la versión que haya seleccionado el usuario e inicia automáticamente el servidor/contenedor.

2.5.2 Tarea servidor web y base de datos

El servidor web está basado en NGINX, donde va a tener la función de mostrar la página a los usuarios y gestionar las peticiones que llegan desde el navegador. Es el encargado de servir los archivos de la web que son el HTML, el Css y el script, y de permitir el acceso a la plataforma a través de la dirección IP del servidor. Además, actúa como intermediario entre el usuario y el backend, haciendo posible que la web responda de forma rápida y estable incluso con varias peticiones a la vez.

Por otro lado tenemos nuestra base de datos llamada “gsbase”, que será la encargada de almacenar toda la información importante del sistema. Aquí se guardan los usuarios registrados, las contraseñas, los servidores creados, sus configuraciones y el estado de cada uno. Gracias a ella, la plataforma puede

recordar los datos de cada usuario y recuperar su información cada vez que inicia sesión.

Y si hacemos un resumen de cada tarea que hace nuestro servidor web y nuestra base de datos sería de la siguiente manera:

- Servidor web: El servidor web se encarga de la parte visible y del acceso a la web.

- Base de datos: La base de datos mientras de va a encargar de guardar y mantener a salvo la información de los usuarios y los servidores.

2.5. 3. Copias de seguridad

Una de las tareas principales de las copias de seguridad es asegurar la protección y la disponibilidad de los datos del sistema web mediante la gestión de copias externas. Se encargan de recibir y almacenar de la forma más segura las copias de seguridad en la base de datos que se generan en el servidor principal, y estas se envían de manera automática y segura mediante SSH.

Tiene la tarea de cada tres días a las dos de la mañana hacer las suficientes copias de seguridad para poder tener la información guardada, y cada mes las copias de seguridad más antiguas serán borradas automáticamente, para que evitar la acumulación de copias de seguridad con los datos y así ganar espacio disponible para nuevas copias de seguridad.

Esta máquina funciona de manera totalmente automática, realizando la recepción y gestión de las copias de seguridad de forma periódica, sin necesidad de intervención manual, lo que garantiza que el sistema se mantenga siempre protegido y operativo.

2.5 Definició de les funcionalitats

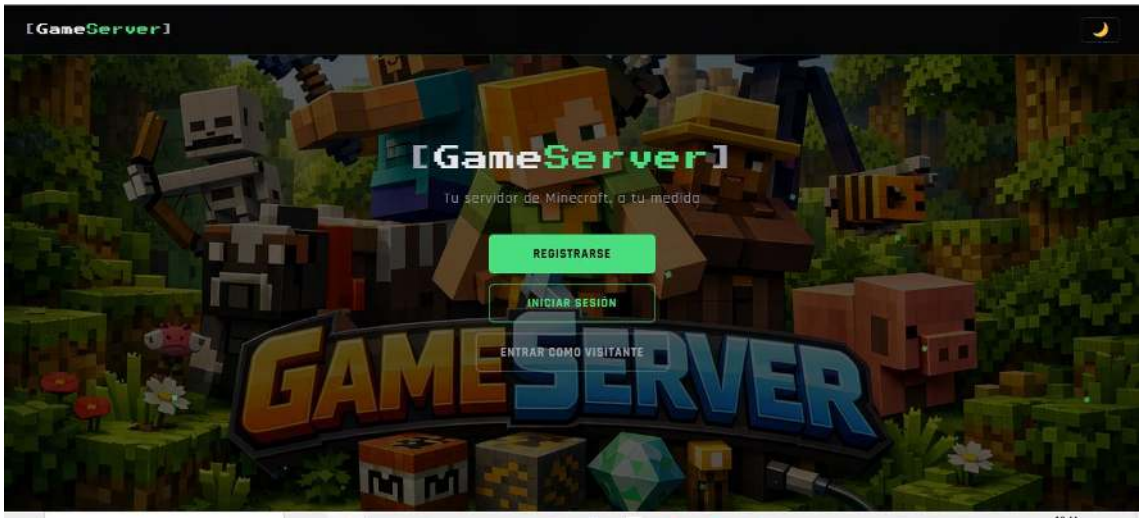
2.5.1 Registro de usuario

La página web dispone de un sistema de registro e inicio de sesión desarrollado con JavaScript, PHP y MySQL. Esta funcionalidad permite a los usuarios crear una cuenta y acceder posteriormente a la web de forma sencilla.

El archivo JavaScript contiene las funciones encargadas de validar los datos introducidos en los formularios. Antes de enviar la información al servidor, el sistema comprueba que el nombre de usuario tenga la longitud mínima, que el correo electrónico sea válido y que las contraseñas coinciden. Si algún dato es incorrecto, se muestran mensajes de error en pantalla.

Cuando las validaciones son correctas, JavaScript envía los datos mediante fetch() a los archivos PHP del servidor. El archivo registrar.php recibe los datos del nuevo usuario y los guarda en la base de datos MySQL utilizando consultas preparadas. Por otro lado, login.php comprueba si el usuario existe y si la contraseña introducida coincide con la almacenada en la base de datos.

Toda la información se almacena en la tabla de usuarios, donde se guardan datos como el nombre, correo electrónico, contraseña y fecha de registro. Esta funcionalidad ha sido implementada completamente y permite a los usuarios registrarse e iniciar sesión correctamente en la página web.

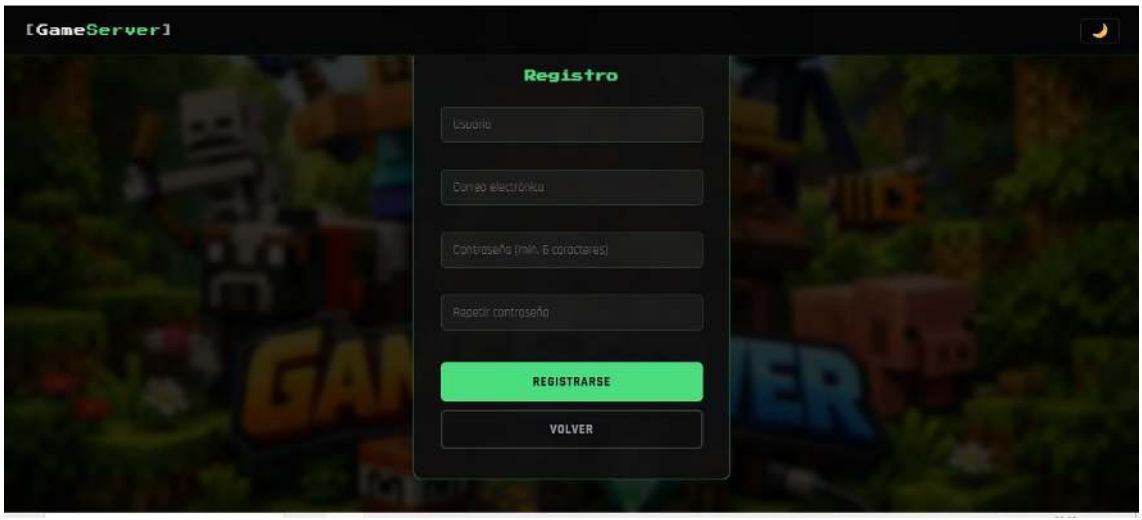


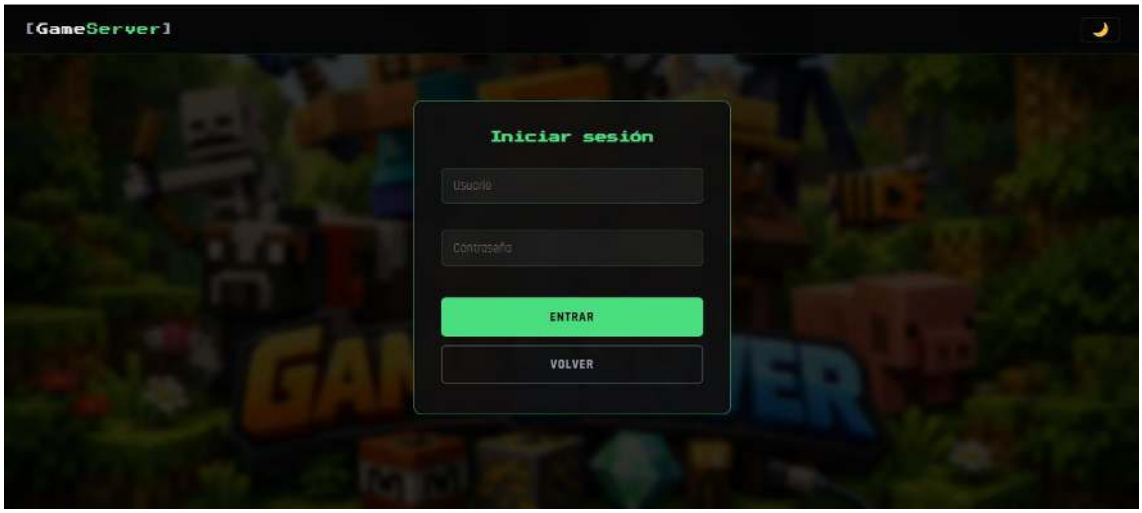

```
mysql> DESC usuarios;
+----------------+--------------+------+-----+-------------------+-------------------+
| Field          | Type         | Null | Key | Default           | Extra             |
+----------------+--------------+------+-----+-------------------+-------------------+
| id_usuarios    | int          | NO   | PRI | NULL              | auto_increment    |
| nombre         | varchar(100) | NO   |     | NULL              |                   |
| correo         | varchar(150) | NO   | UNI | NULL              |                   |
| contraseña     | varchar(255) | NO   |     | NULL              |                   |
| fecha_registro | timestamp    | YES  |     | CURRENT_TIMESTAMP | DEFAULT_GENERATED |
+----------------+--------------+------+-----+-------------------+-------------------+
5 rows in set (0,00 sec)

mysql>
```

```
mysql> SELECT * FROM usuarios;
+-------------+-----------+-----------------------------+-------------+---------------------+
| id_usuarios | nombre    | correo                      | contraseña  | fecha_registro      |
+-------------+-----------+-----------------------------+-------------+---------------------+
|           9 | Joselito  | joselito12@gmail.com        | JOSE123     | 2026-04-28 19:02:27 |
|          10 | user12    | user123@gmail.com           | 123456      | 2026-04-29 16:10:48 |
|          11 | 147ruben  | rcastillejos@elpuig.xeill.net | 147852    | 2026-05-07 18:49:18 |
|          12 | holabuenas | holabuenas@gmail.com       | 147852      | 2026-05-11 17:42:01 |
|          13 | Rubén     | rubenauer2006@gmail.com     | 123098      | 2026-05-12 18:54:05 |
|          14 | bienvenido | bienvenido@gmail.com       | 147852      | 2026-05-13 18:39:55 |
|          15 | peperosa  | pepe.rosa@gmail.com         | p2p2#666    | 2026-05-13 19:00:23 |
+-------------+-----------+-----------------------------+-------------+---------------------+
7 rows in set (0,00 sec)
```

2.5.2 Para abrir el registro, para abrir el login y para la opción volver

Estas funcionalidades sirven para como bien indica sus nombres, para poder entrar a las funcionalidades para poder registrarse con el objetivo de permitir que el usuario tenga su cuenta personal para guardar su información, como los servidores que los usuarios vayan a crear. iniciar sesión sirve para verificar la identidad del usuario y permitir acceder a la información personal y poder configurar sus servidores.Y la opción volver que permite regresar a una pantalla anterior y así facilita la navegación por la web y evitar que se quede estancado en un apartado de la web.





2.5.3 copias de seguridad

La página web dispone de un sistema de copias de seguridad automáticas desarrollado con un script en Bash y tareas programadas con cron. Esta funcionalidad permite crear backups de la base de datos MySQL y almacenarlos en otra máquina de la red para mayor seguridad.

El script copia.sh genera una copia de la base de datos utilizando mysqldump y posteriormente la envía al servidor remoto mediante rsync y conexión SSH. Después de la transferencia, el sistema elimina las copias antiguas de más de 30 días para mantener el almacenamiento organizado.

La automatización se realiza mediante crontab, que ejecuta el script cada 3 días a las 2:00 de la madrugada. Además, el proceso guarda un registro en un archivo log para comprobar que las copias se han realizado correctamente.

2.5.4 Creación de servidor

La máquina web es la interfaz principal del sistema, desde la cual los usuarios pueden gestionar servidores Minecraft de forma remota. Esta parte del proyecto está

desarrollada principalmente con JavaScript en el frontend y PHP en el backend, conectándose con la API del servidor de gestión mediante peticiones HTTP (fetch).

La funcionalidad de creación de servidores se implementa en la función crearServidor(). Esta función recoge los datos introducidos por el usuario en la interfaz web (nombre, versión y tipo de servidor) y realiza validaciones básicas para asegurar que los campos no estén vacíos. Una vez validados, los datos se envían mediante fetch() al backend (orden.php) con la acción crearServidor. A partir de ahí, el servidor se encarga de crear la instancia real del servidor Minecraft en la máquina de hosting. Si el proceso es correcto, el sistema devuelve la IP y el puerto de conexión, que se muestran al usuario mediante notificaciones en pantalla.

Además, la web permite controlar el estado de los servidores. La función pararServidor() envía una petición al backend para detener un servidor activo, mientras que arrancarServidor() lo vuelve a iniciar. Estas acciones actualizan también la interfaz, modificando indicadores visuales como el estado del servidor, el contador de tiempo de actividad y el icono de conexión.

La web también incluye un panel donde el usuario puede cargar servidores existentes, seleccionarlos desde un desplegable y gestionarlos sin necesidad de crearlos de nuevo. Toda esta información se obtiene dinámicamente desde la base de datos a través del backend.

En conjunto, esta parte del sistema actúa como punto central de control, ya que permite al usuario interactuar con los servidores sin acceder directamente a la máquina de hosting. La funcionalidad está completamente implementada y operativa dentro de la red interna del proyecto.

A partir de esta imagen vemos las diferentes funcionalidades que funcionan gracias al backend de máquina hosting.

2.5.5 Cargar servidor creado anteriormente con la información

La función cargarServidoresExistentes(). Se ejecuta cada vez que el usuario entra a la sección "Servidor". Hace una petición al backend pidiendo la lista de servidores del usuario, y rellena el desplegable con los que encuentre. Cada opción guarda también la versión y el tipo del servidor como datos ocultos para usarlos luego. Evidentemente

si el usuario tiene un servidor ya creado desde antes sino esta función no le interesa y creará un servidor desde cero.

La siguiente función cargarModsInstalados(nombre). Recibe el nombre del servidor y pregunta al backend qué mods tiene instalados. Primero desmarca todos los mods, y luego marca como seleccionados solo los que el backend confirme que ya están en ese servidor. Así cuando el usuario carga un servidor ve exactamente qué mods tiene activos sin tener que recordarlo.

La última función es el botón que se ve en la imagen de arriba que se llama cargarServidor(). Esta función se ejecuta cuando el usuario pulsa el botón "Cargar" después de elegir un servidor del desplegable. Coge el servidor seleccionado, rellena el campo de nombre y la versión en el formulario de configuración, y llama a cargarModsInstalados() para que aparezcan marcados los mods que ya tiene ese servidor. Al final muestra un toast diciéndole al usuario que pulse Arrancar.

2.5.6 Gestión del servidor versiones y mods

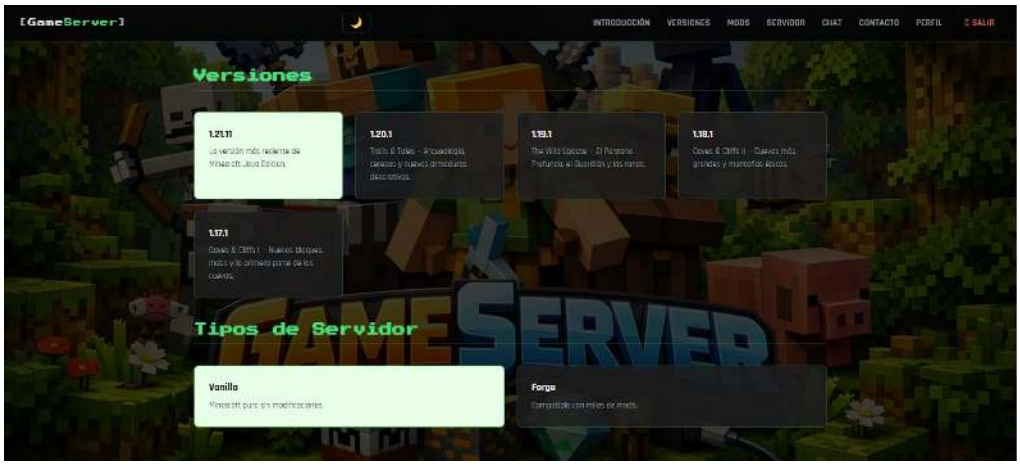
La página web incluye un panel de gestión de servidores Minecraft que permite a los usuarios crear, iniciar, detener y administrar servidores de forma remota. Esta funcionalidad está desarrollada con JavaScript en el frontend y PHP en el backend, mediante el archivo orden.php, que actúa como intermediario con la máquina de hosting.

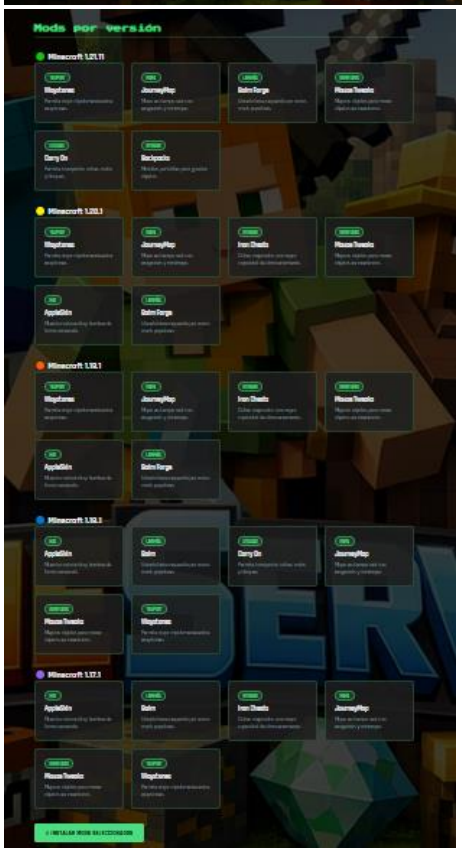
La creación de servidores se realiza a través de la función crearServidor(), que recoge el nombre, la versión y el tipo de servidor (Vanilla o Forge). La selección de la versión y del tipo es obligatoria, por lo que el sistema no permite crear un servidor si estos campos no están completos. Los datos se envían al backend mediante una petición fetch().

En el servidor, orden.php valida la información, la guarda en la base de datos MySQL y ejecuta un playbook de Ansible que despliega un contenedor Docker en la máquina de hosting. Una vez creado, el sistema devuelve un mensaje al usuario con la IP y el puerto de conexión.

Además, la web permite cargar servidores existentes, así como arrancarlos y detenerlos, actualizando su estado en tiempo real. También se incluye la gestión de mods, que se almacenan en la base de datos y pueden consultarse posteriormente.

En conjunto, esta funcionalidad conecta la web con la máquina de hosting mediante PHP, Ansible y Docker, permitiendo una gestión completa de servidores Minecraft desde la interfaz web.



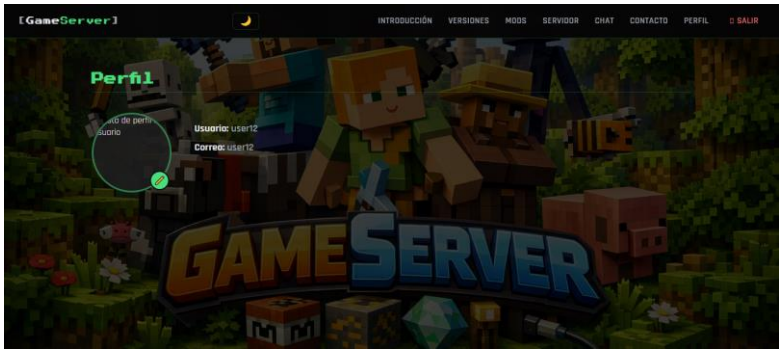


2.5.7 Subir imagen de perfil

La función subirImagen(event) permite que un usuario cargue una imagen de su perfil en la aplicación. Primero, usa FileReader para acceder a la foto elegida desde el dispositivo del usuario, transformándola en un formato que pueda ser visualizado y almacenado por el navegador.

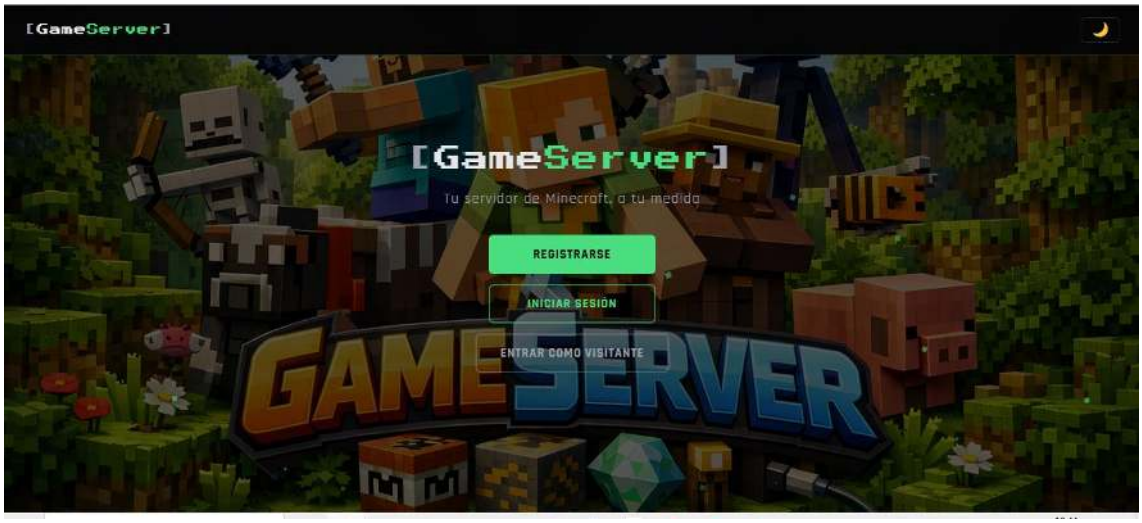
Una vez que la imagen se ha cargado por completo, se asigna automáticamente al área de la foto de perfil para mostrarla en la pantalla. Luego, la función verifica si hay un usuario conectado y, en caso afirmativo, almacena la nueva imagen en los datos de dicho usuario, actualiza la sesión y modifica la información guardada en localStorage para que la foto se mantenga incluso si se cierra o vuelve a abrir la aplicación. Por

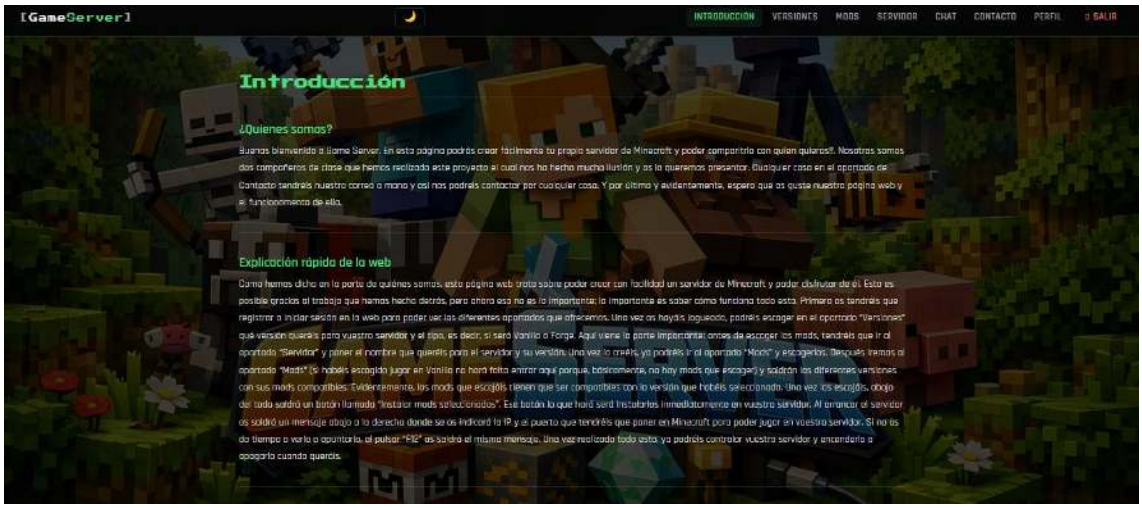
último, se accede al archivo que el usuario ha seleccionado a través del input de archivos para finalizar el proceso de carga de la imagen.



2.5.8 Visitante

La función entrarVisitante() permite que un usuario se introduzca en la aplicación en modo invitado sin la necesidad de crear una cuenta o empezar sesión. Al ejecutarla, la variable visitante se establece en true, lo que indica que el usuario ha ingresado como un invitado. Luego, se oculta la pantalla principal y solo se visualiza la parte de introducción a través de la función showSection('intro'). Este tipo de acceso es restringido, ya que el visitante solo puede ver la introducción de la aplicación, mientras que las otras secciones y funciones, como la creación de servidores o el acceso a áreas completas, quedan inhabilitadas. Para habilitar todas las características y funcionalidades de la aplicación, el usuario debe registrarse y acceder con su propia cuenta.





2.5.9 MostrarMenu y mostrar secciones

La función mostrarMenu() se utiliza para que se muestre el menú principal de la aplicación. En primer lugar, se obtiene el elemento del menú usando su ID y luego se modifica su propiedad display a flex, lo que permite que el menú aparezca en la pantalla. También se añade la clase menu-visible, que se puede usar para aplicar estilos o animaciones visuales a través de CSS. Con esta función, el usuario puede acceder a las diversas opciones y secciones de la aplicación de forma clara y fácil.

La función showSection(id) tiene el propósito de presentar las distintas partes de la aplicación, dependiendo de la elección del usuario. Primero, verifica si el usuario ha ingresado como visitante y si intenta acceder a una parte que no sea la introducción; en tal caso, se muestra un mensaje que indica que necesita registrarse para acceder a ese contenido y la función se detiene. Luego, oculta todas las secciones que están visibles y muestra solo la sección que corresponde al identificador proporcionado. También desplaza la pantalla de manera suave hacia la sección seleccionada para mejorar la experiencia de navegación. Además, según la sección que esté abierta, puede llevar a cabo acciones adicionales como iniciar un chat con bots o cargar servidores ya existentes. Por último, cierra el menú lateral quitando la clase open para que la interfaz se vea más ordenada y limpia.

3 Altres capítols

4 Conclusions

4.1 Conclusions generals del projecte

Nuestras conclusiones están claras, nosotros no nos esperábamos que fuera un trabajo fácil evidentemente.Nos esperábamos un proyecto difícil que tiene su complicación y puede llegar a ser estresante, pero tampoco nos esperábamos la realidad pura y dura que ha sido, que en algunos momentos nos dieran más errores técnicos que salían por todos los lados y no sabíamos cómo solucionarlo. Esto nos hizo perder bastante tiempo en investigar cómo solucionar los errores lo antes posible para poder continuar con el proyecto. Nos hizo que en algunos momentos nos sacará bastante de quicio, porque parecía que habíamos solucionado un error y volvían a salir tres más.

A pesar de todas las dificultades que hemos tenido durante el proyecto, poco a poco fuimos solucionando cada problema, investigando para que no volviera a suceder el error y aprendiendo de todos los errores que hemos tenido.

Ha sido complicado pero el resultado final y todo lo que hemos aprendido durante todos estos meses de trabajo han hecho que valga la pena luchar hasta el final y conseguir solucionar todos los problemas y que funcione todo correctamente.

4.2 Consecució dels objectius

Los objetivos principales del proyecto se han cumplido satisfactoriamente, ya que se ha conseguido desarrollar una página web funcional con conexión a una base de datos, un backend operativo y un servidor hosting configurado correctamente. Además, también se ha podido comprobar el rendimiento y la estabilidad del servidor durante las pruebas realizadas, verificando que la plataforma es capaz de funcionar de manera correcta y mantener la información almacenada de forma segura mediante las copias de seguridad implementadas.

4.3 Valoració de la metodologia i planificació

La metodología que se centra en una planificación por etapas ha hecho posible estructurar el trabajo de manera más efectiva y tener un mejor monitoreo del avance del proyecto. Separar el proyecto en distintas fases contribuyó a identificar fallos más velozmente y a resolver inconvenientes antes de seguir adelante. Adicionalmente, esta organización ayudó a repartir las tareas, a observar el progreso y a mejorar continuamente cada aspecto de la página web, desde la parte del usuario hasta el ajuste del servidor, la base de datos y el alojamiento.

4.4 Visió de futur

En un futuro se está considerando la idea de aumentar la plataforma en el futuro, permitiendo que un mayor número de usuarios la utilicen y aprovechen todas sus características. También se busca facilitar a los usuarios la posibilidad de crear y administrar sus propios servidores de Minecraft de manera más accesible y adaptada a sus necesidades. Además, se podrían introducir mejoras adicionales que aborden el rendimiento, la seguridad, las copias de seguridad y la optimización del servicio de alojamiento, con el fin de

brindar una experiencia más sólida y satisfactoria a todos los usuarios de la plataforma.

5. Glossari

Ansible: Herramienta de automatización de servidores que permite ejecutar tareas de configuración mediante playbooks. En el proyecto se usa para crear automáticamente los contenedores de cada servidor cuando un usuario crea uno nuevo.

Backend: Parte del sistema que se ejecuta en el servidor, invisible para el usuario. En el proyecto, el archivo orden.php actúa como backend gestionando la comunicación entre la web, la base de datos y la máquina de hosting.

Frontend: La parte del sistema que el usuario ve y con la cual interactúa como los botones de registrarse, el botón de iniciar sesión, el botón volver y todas las funciones que tenemos en la web.

Forge y Vanilla: Estos son los dos tipos de servidor los cuales, los usuarios los podrán utilizar. Vanilla es el tipo de servidor que se utiliza para cuando quieres jugar sin mods, en cambio, el tipo de servidor Forge se utiliza para poder añadir mods y poder jugar como por ejemplo el mod del mapa.

Crontab: Es una utilidad de Linux que sirve para programar tareas periódicas. Es decir, nosotros la utilizamos para ejecutar el script de las copias de seguridad cada tres días a las 2 de la mañana.

Mysqldump: Herramienta de MySQL para exportar la base de datos a un archivo. Se utiliza en el script copia.sh para generar las copias de seguridad.

Fetch(): Esta es una función de JavaScript que consiste en permitir hacer las peticiones HTTP al backend para poder enviar o recibir datos de manera no simultánea.

Rsync: Es una herramienta que sirve para transferir archivos entre diferentes sistemas a partir de SSH. Se usa para enviar las copias de seguridad de la máquina web a la máquina de las copias de seguridad.

6. Bibliografia

Docker:
Esta es una página web que visitamos al comenzar el proyecto para informarnos sobre Docker https://docs.docker.com/get-started/get-docker Es como una web sobre Docker y tiene apartado de un bot que le puedes preguntar cualquier cosa sobre Docker.

Ansible:
En esta página entramos relativamente en febrero de 2026 para poder investigar sobre Ansible. Y esta página nos explica todo sobre Ansible https://docs.ansible.com/projects/ansible/latest/getting_started/index.html. Aunque también le hicimos unas cuestiones a nuestro profesor Oscar Torrente de redes, que también nos ayudó con este tema.

Php:
Sobre Php entramos en esta página alrededor de principios de febrero o mitas de febrero. Nos ayudó a saber como poder conectar nuestra base de datos con nuestra web y también a organizar nuestro backend y frontend.
https://www.php.net/manual/es/book.mysqli.php

JavaScript:

JavaScript lo comenzamos a investigar sobre ello a finales de febrero, cuando comenzamos a subir la web a nginx. Para poder tener más conocimiento sobre ello y facilitar a la hora de como conectar el backend para que se hagan las diferentes funciones. https://developer.mozilla.org/es/docs/Web/JavaScript

Y para las otras cosas utilizamos apuntes que teníamos de clase o preguntamos dudas a nuestros profesores o compañeros que nos pudieran ayudar.

7 Annexos

Manual para los usuarios:

A.1 Acceso a la plataforma

Para acceder a la plataforma GameServer, el usuario debe abrir el navegador web e introducir la IP del servidor web en la barra de direcciones. Una vez cargada la página principal, aparecerán tres opciones disponibles: *Registrarse*, *Iniciar Sesión* y *Entrar como Visitante*.

Si es la primera vez que se accede a la plataforma, será necesario registrarse. Para ello, se debe pulsar la opción *Registrarse* y completar el formulario con el nombre de usuario, el correo electrónico y una contraseña de mínimo 6 caracteres. Después de completar el registro, el usuario podrá iniciar sesión utilizando las credenciales creadas.

A.2 Crear un servidor

Una vez iniciada la sesión, el usuario podrá acceder al apartado *Servidor* desde el menú principal. En esta sección será posible crear un nuevo servidor de Minecraft introduciendo primero un nombre identificativo.

Posteriormente, se deberá seleccionar la versión de Minecraft deseada, como por ejemplo la 1.21.11, 1.20.1 o 1.19.1, entre otras disponibles. También será necesario escoger el tipo de servidor: *Vanilla*, para jugar sin modificaciones, o *Forge*, para permitir el uso de mods.

En caso de escoger *Forge*, aparecerá el apartado *Mods*, desde donde se podrán seleccionar los mods compatibles con la versión elegida. Finalmente, al hacer clic en el botón *Crear Servidor*, el sistema generará el servidor y mostrará la IP y el puerto necesarios para conectarse.

A.3 Conectarse al servidor desde Minecraft

Para conectarse al servidor creado, es necesario abrir el *Minecraft Launcher* y seleccionar la misma versión utilizada durante la creación del servidor.

Después, dentro del juego, se debe acceder a la opción *Multijugador* y seleccionar *Añadir servidor*. A continuación, el usuario deberá introducir la IP y el puerto proporcionados por la plataforma en formato *IP:puerto*. Finalmente, solo habrá que pulsar *Unirse al servidor* para entrar al mundo creado.

A.4 Gestionar el servidor

La plataforma también permite gestionar los servidores ya creados desde el *Panel Servidor*. Para iniciar un servidor, el usuario deberá pulsar el botón *Arrancar*. Si se desea detener el servidor, será necesario utilizar el botón "Parar".

Además, si existen varios servidores guardados, será posible cargar uno seleccionándolo desde el menú desplegable y haciendo clic en *Cargar*.

A.5 Perfil de usuario

El usuario puede consultar su información personal accediendo al apartado *Perfil*. En esta sección se mostrarán el nombre de usuario y el correo electrónico asociados a la cuenta.

También existe la posibilidad de personalizar el perfil subiendo una imagen. Para ello, solo es necesario pulsar sobre la fotografía de perfil y seleccionar un archivo de imagen desde el dispositivo.

A.6 Modo visitante

La plataforma dispone de un modo visitante destinado a los usuarios que todavía no se han registrado. Este modo únicamente permite acceder al apartado *Introducción*, donde se muestra información básica sobre el funcionamiento de GameServer.

Para poder utilizar todas las funcionalidades disponibles, como crear o gestionar servidores, es necesario registrarse e iniciar sesión con una cuenta de usuario.