Institut Puig Castellar
Projecte / Crèdit de síntesi
Document funcional

DOCUMENTO FUNCIONAL DEL PROYECTO

ALUMNO/GRUPO: Alessandro Nadal

---

1. Introducción y contexto

Descripción general: PuigCraft será un servidor (o framework de servidor) de Minecraft escrito en Python, construído sobre la librería Twisted — que implementa el protocolo de red de Minecraft. El objetivo es facilitar el desarrollo de servidores ligeros, proxies o servidores de propósito especial (minijuegos, entornos controlados…), aprovechando la flexibilidad de Python.

Problema/Necesidad: Muchos proyectos de servidor Minecraft tienen código complejo, son pesados o difíciles de modificar. Con una base en Python + Twisted, se reduce la barrera de entrada, se facilita la experimentación, la configuración, scripting de minijuegos, proxies, etc. Es útil para desarrolladores que quieren prototipar o montar servidores personalizados sin necesidad de C/C++.

Usuario/Cliente final:

- Desarrolladores que quieren crear servidores Minecraft personalizados (minijuegos, proxies, herramientas de red).
- Educadores o estudiantes interesados en protocolos de red / desarrollo de servidores en un lenguaje de alto nivel.
- Administradores de servidores que buscan flexibilidad y facilidad de scripting.

Solución propuesta: Implementar PuigCraft como proyecto en Python, usando Twisted como base del protocolo. Exponer una API/fábrica de servidor que permita definir lógicas propias: gestión de jugadores, mundo simplificado o personalizado, minijuegos, comandos, logs, configuración flexible.

---

2. Análisis de requisitos

2.1. Requisitos funcionales (RF)

Código	Descripción del requisito funcional
RF1	El servidor escuchará conexiones TCP en el puerto por defecto de Minecraft (por ejemplo, 25565) y aceptará clientes compatibles con las versiones soportadas.
RF2	Permitir mantener la sesión del jugador (estado básico: conexión, nombre, posiblemente posición, aunque la lógica del “mundo” deberá programarse).
RF3	Permitir definir “mundo / entorno / lógica de juego” mediante código Python: minijuegos, lógica de bloques o entorno simplificado, chat/comandos, teletransportes, spawn, etc.
RF4	Proveer comandos administrativos / de configuración, idealmente mediante consola o comandos via chat.
RF5	Ser fácilmente extensible: permitir que el desarrollador de server agregue hooks o handlers para eventos (conexión, movimiento, chat, acciones, minijuegos).

2.2. Requisitos no funcionales (RNF)

Código	Descripción del requisito no funcional
RNF1	Seguridad razonable: manejo correcto de paquetes, validación, evitar errores/crashes cuando un cliente envía datos inesperados (heredados de Quarry).
RNF2	Portabilidad: funciona en cualquier sistema con Python 3.9+ e instalaciones estándar de dependencias (por ejemplo, twisted, cryptography).

2.3. Restricciones

- Lenguaje: Python (versión compatible con Quarry: Python 3.9+).
- Dependencias: twisted, cryptography, noise.
- Dependencias o limitaciones técnicas: Falta de conocimientos del protocolo a seguir

---

3. Análisis de usuarios y roles

Rol	Descripción	Permisos principales
Administrador	Persona que despliega y configura el servidor PuigCraft	Iniciar/detener el servidor Configurar lógica o minijuegos Gestionar comandos de admin Ver logs
Desarrollador	Escribe la lógica del servidor: mundos, minijuegos, comandos personalizados	Crear/modificar scripts Python que definen el comportamiento del servidor; desplegar lógica
Jugador	Usuario cliente que se conecta con un cliente Minecraft compatible	Conectarse, jugar (según la lógica definida: chat, minijuegos, etc.), interactuar conforme a reglas del servidor

---

4. Casos de uso / Escenarios de uso

Código	Nombre del caso de uso	Actor principal	Descripción	Resultado esperado
CU1	Encender el servidor	Administrador	Ejecutar el servidor PuigCraft (script Python) en el host. El servidor arranca, carga configuración (scripts, plugins, lógica) y comienza a escuchar en puerto 25565.	Mensaje de arranque, servidor listo para aceptar conexiones.

CU2	Se conecta al servidor	Jugador	Un jugador abre su cliente Minecraft, apunta a la IP/puerto, se hace handshake/login con el servidor. El servidor acepta la conexión y crea un objeto de sesión para el jugador.	Jugador conectado, visible para el servidor; puede recibir mensajes, eventos, participar en lógica definida.
CU3	Mantenimiento del servidor	Administrador	A través de consola del servidor o comandos dentro del juego (chat/comandos), cambiar configuración, reiniciar minijuego, expulsar jugadores, recargar scripts/plugins.	Cambios aplicados, servidor sigue funcionando, notificaciones a jugadores si es necesario.

---

5. Modelo de datos o estructura de la información

Jugador (Player): id de sesión, nombre, conexión (socket/handler), estado (conectado, en lobby, en minijuego, etc.), metadata (puede incluir posición, inventario o datos personalizados si la lógica lo define).

Servidor (Server): instancia principal del servidor PuigCraft — contiene configuración global, lista de jugadores conectados, lógica activa (módulos, minijuegos), logs.

Mundo / Entorno (World / GameState): dependa de la lógica: podría ser un mapa simplificado, datos de spawn, zonas, reglas de juego, estado de minijuego.

Módulo / Plugin / Script: unidad de lógica personalizada — define handlers de eventos (on_connect, on_chat, on_move, on_packet, etc.), reglas del juego, comandos, lógica de mundo.

Conexión / Protocolo (Connection / ProtocolHandler): objeto gestionado por Quarry/Twisted que representa la conexión cliente-servidor, maneja paquetes de red, compresión/encriptación, login, intercambio de datos.

---

6. Diseño de la interfaz

Como servidor de red — no hay GUI integrada (o en su defecto consola). Interfaz en consola para iniciar servidor, ver logs, administrar comandos.

---

7. Planificación técnica

- Lenguajes y frameworks: Python, Twisted y opcionalmente Flask/Quart para la página web.
- Base de datos: MariaDB
- Herramientas de diseño o edición: Ninguna
- Cronograma (puede incluir un diagrama de Gantt):

---

8. Análisis de riesgos

8.1. Identificación de riesgos

- Velocidad / rendimiento menor que un servidor en C o Java (Python + Twisted) — podría no ser apto para muchos jugadores simultáneos.
- Incompletitud del protocolo “play”: la librería Quarry no implementa por defecto todos los paquetes en modo “play”: muchas funcionalidades deben ser implementadas manualmente.
- Dependencia de versiones de Minecraft compatibles con Quarry — nuevas versiones podrían no ser soportadas inmediatamente.
- Complejidad de implementar un “mundo completo” — probablemente habrá limitaciones si se quiere replicar un servidor full-featured.
- Seguridad / robustez: dado que mucha lógica estará en manos de plugins Python, un error podría comprometer la estabilidad; manejo de paquetes malformados, validaciones, protección frente a exploits.

8.2. Valoración y respuesta

Clasifica cada riesgo según su probabilidad e impacto, e indica cómo se mitigará.

Riesgo	Probabilidad	Impacto	Plan de prevención o contingencia
Rendimiento limitado	Medio	Medio	Enfocar el proyecto a servidores pequeños (minijuegos, pocos jugadores), realizar pruebas de estrés, optimización, uso moderado de lógica intensiva.
Protocol “incompleto”	Alta	Alta	Documentar bien qué paquetes need implementar; desarrollar sólo las funcionalidades necesarias; priorizar lo esencial (login, chat, comandos, spawn).
Incompatibilidad de versión	Medio	Medio	Hacer copias de seguridad semanales.

Errores de plugins / seguridad	Medio	Alto	Definir buenas prácticas, validación de datos, excepciones manejadas; pruebas unitarias; limitar privilegios.

---

9. Validación y criterios de éxito

- El servidor PuigCraft arranca sin errores, y permite que un cliente Minecraft compatible se conecte (login, handshakes, estado).
- Se pueden definir y cargar módulos/plugins de lógica de servidor de forma dinámica.
- Jugadores pueden interactuar mediante chat / comandos / lógica personalizada.
- Un minijuego simple o lógica personalizada (por ejemplo, spawn + mensajes + comandos) funciona correctamente para al menos 2 jugadores en LAN.
- Logs y administración básica operativos (consola, comandos, estado).
- Documentación mínima de uso (README, cómo crear plugins, cómo lanzar servidor).

Pruebas previstas: pruebas con 2–5 jugadores; pruebas de login/logout; pruebas de comandos; prueba de plugin; pruebas de desconexión/errores; prueba de estabilidad tras varias conexiones.

---

10. Conclusión

PuigCraft aspira a ser una alternativa ligera y flexible para montar servidores Minecraft personalizados usando Python. Basándose en la librería Quarry para manejar el protocolo de red, PuigCraft se centrará en ofrecer una API extensible mediante código Python, permitiendo definir mundos, minijuegos o comportamientos a medida, sin necesidad de compilar en C o Java, y facilitando la experimentación, scripting y prototipado rápido.

El análisis funcional proporciona una visión clara y realista del proyecto. A partir de aquí, comienza la fase de desarrollo.