Extensión IoT aplicada a un juego 2D en Godot Engine: Dino Runner

Javier Antonio Keller
Facultad de Ingeniería - UNaM
Oberá, Argentina
kellerjavier13@gmail.com

***Abstract*—This work presents the development of a 2D video game inspired by Google's T-Rex Runner, created in Godot Engine, with the integration of Internet of Things (IoT) concepts. The project aims to combine entertainment with technological experimentation, connecting the game with a web application developed in Flask that manages players' scores through an API and a MariaDB database. The article describes the main mechanics of the game, the IoT extensions, the use of free assets, and the preliminary results obtained. Additionally, the deployment uses Docker containers to ensure portability and scalability of the system.**

***Keywords*—IoT; Godot Engine; Flask; API; Docker; Video Games**

***Resumen*—Este trabajo presenta el desarrollo de un videojuego 2D inspirado en el T-Rex Runner de Google, creado en el motor Godot Engine, con la integración de conceptos de Internet de las Cosas (IoT). El proyecto busca combinar el entretenimiento con la experimentación tecnológica, conectando el juego con una aplicación web desarrollada en Flask que gestiona los puntajes de los jugadores mediante una API y una base de datos MariaDB. Se describen las principales mecánicas del juego, el uso de assets gratuitos de la comunidad, las extensiones IoT y los resultados preliminares obtenidos. Asimismo, el despliegue del sistema se realiza dentro de contenedores Docker, lo que permite mayor portabilidad y escalabilidad en la infraestructura.**

***Palabras-clave*—IoT; Godot Engine; Flask; API; Docker; Videojuegos**

I. Introducción

El desarrollo de videojuegos se ha consolidado como un espacio de innovación que permite experimentar con nuevas tecnologías en un entorno controlado y atractivo. En este contexto, el proyecto *Dino Runner* surge como una adaptación del conocido *T-Rex Runner*, cuyo desarrollo inicial se basó en metodologías de referencia para la creación de juegos *endless runner* en *Godot Engine* [1], expandiéndose posteriormente con la incorporación de mecánicas adicionales y la integración de elementos de IoT.

El objetivo principal del trabajo es doble: por un lado, desarrollar un videojuego sencillo pero expandido con nuevas características como biomas, *power-ups* y animaciones; por otro lado, aplicar conceptos de IoT para crear un ecosistema que conecte el juego con servicios externos, tales como bases de datos y aplicaciones web.

Para el desarrollo del videojuego se emplea *Godot Engine* [9], un motor de videojuegos de código abierto que destaca por su flexibilidad, su comunidad activa y su compromiso con el software libre, fomentando la transparencia y la accesibilidad en el desarrollo de proyectos interactivos.

De esta manera, el proyecto no se limita al entretenimiento, sino que se convierte en una oportunidad para poner en práctica conocimientos adquiridos en la asignatura *Internet de las Cosas, Sensores y Redes*, de la carrera de Ingeniería en Computación.

II. Desarrollo del Juego

A. Mecánicas principales

El juego se construyó en Godot Engine, utilizando assets gratuitos de la plataforma itch.io y otras fuentes comunitarias. La jugabilidad básica se mantiene similar al *T-Rex Runner* original: el personaje debe saltar o esquivar obstáculos para sobrevivir el mayor tiempo posible, acumulando puntaje a medida que avanza. Se puede capturas del juego en las figuras 1, 2 y 3

Las nuevas mecánicas incluyen:

- **Cambio de biomas** con obstáculos temáticos.
- **Power-ups**, tales como:
  - Reducción de la velocidad del juego.
  - Vida extra.
  - Multiplicador de puntos por tiempo limitado.
  - Disminución temporal de la dificultad.
- **Opciones de control**, como botones para pausar el juego o desactivar música y sonidos.
- **Ranking de jugadores**, tambíen dentro del juego se podrá ver el top 15 de jugadores y la posición del jugador en la tabla.

B. Uso de assets gratuitos y licencias

Una de las características centrales del proyecto es el aprovechamiento de recursos gráficos y sonoros gratuitos,

Fig. 1. Captura del juego en bioma de jungla.

Fig. 2. Captura del juego en bioma de montañas.

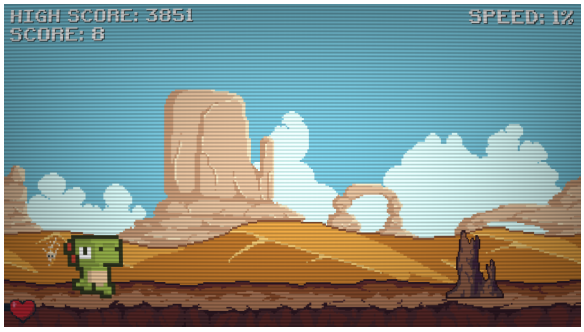

Fig. 3. Captura del juego en bioma de desierto.

provenientes de repositorios abiertos y comunidades como itch.io o Godot Shaders. Estos elementos permiten reducir el tiempo de desarrollo y enfocarse en la programación y diseño de mecánicas.

Para los recursos visuales se utilizaron diversos *assets* de libre uso, entre los cuales se incluyen el conjunto de fondos con efecto de parallax [3], los personajes principales del tipo dinosaurio [5] y los enemigos voladores [6]. Asimismo, algunos sprites fueron diseñados manualmente utilizando *LibreSprite* [13], un software de código abierto orientado a la edición de gráficos en pixel art, lo que refuerza el compromiso del proyecto con el uso de herramientas libres y accesibles.

En cuanto a los elementos tipográficos y sonoros, se emplearon la fuente *Retro Gaming* [7] y efectos generados con la herramienta *SFXR* [2], ambas bajo licencias que permiten su uso no comercial.

Se respetaron siempre las licencias de uso de cada *asset*, lo cual es una práctica fundamental para cualquier proyecto académico o profesional. La utilización de contenido libre no solo agiliza el desarrollo, sino que también fomenta el espíritu colaborativo y el crecimiento de la comunidad de desarrolladores independientes.

III. Extensión IoT

A. Arquitectura propuesta

El componente IoT del proyecto se centra en la integración de una aplicación web desarrollada con *Flask* [10], una base de datos *MariaDB* [11] y una API REST conectada al motor del juego implementado en *Godot Engine* [8].

- **Aplicación en Flask**: actúa como la interfaz pública, mostrando información del juego, el perfil del desarrollador y el top 15 de jugadores con mayores puntajes.
- **API REST**: funciona como intermediaria entre el juego y la base de datos. *Godot Engine* se conecta a la API para enviar y recibir información sobre los puntajes.
- **Capa de Datos**: compuesta por una base de datos *MariaDB*, cuyo esquema fue diseñado para soportar consultas volumétricas y operaciones concurrentes.
- **Base de datos MariaDB**: almacena los nombres de los jugadores y sus máximos puntajes, garantizando integridad y persistencia de la información.

B. Implementación en contenedores Docker

Para garantizar la portabilidad del sistema, todos los servicios (aplicación web, base de datos y API) se despliegan en contenedores Docker dentro de una VPS (Servidor Privado Virtual) con sistema operativo Debian. Esto permite que el sistema sea fácilmente replicable, manteniendo aisladas las dependencias de cada componente y facilitando futuras actualizaciones.

El uso de contenedores también ofrece la ventaja de escalar horizontalmente el sistema, por ejemplo, aumentando la capacidad de la base de datos o replicando la API para soportar un mayor número de jugadores concurrentes.

IV. Relevancia en el contexto IoT

Aunque no se conecta directamente con sensores físicos, este proyecto reproduce un esquema muy cercano a soluciones IoT reales: un dispositivo (el juego) envía datos a una API que los procesa y los almacena en la nube, siendo luego visualizados en una interfaz accesible.

V. Resultados preliminares

Lo implemetado hasta el momento es:

- Desarrollo de un prototipo jugable en Godot con biomas y animaciones básicas.
- Implementación inicial de la API en Flask con endpoints para GET y POST de puntajes.
- Conexión entre el juego y la API mediante el nodo HTTPRequest de Godot.
- Almacenamiento exitoso de puntajes en la base de datos MariaDB.
- Pruebas de despliegue en una VPS con Debian utilizando contenedores Docker.
- Diseño preliminar de la arquitectura del sistema, representada en una figura conceptual que muestra cómo interactúan los distintos servicios Figura 4.

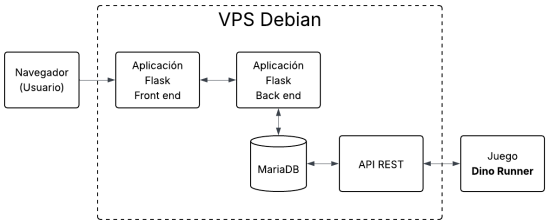

Fig. 4. Diagrama del sistema mencionado (en desarrollo).

Para validar la integración entre el juego y la API, se implementó una metodología de pruebas de comunicación enfocada en la robustez funcional y el manejo de errores. Utilizando el nodo `HTTPRequest` de Godot, se ejecutaron una serie de casos de prueba automatizados para cada endpoint. Para el envío de puntajes (*POST /score*), se verificó la correcta inserción de nuevos jugadores, la actualización condicional de highscores (solo si el nuevo puntaje era superior), y la respuesta de la API ante datos malformados (ej. un highscore no numérico), validando que el juego manejara los códigos de error *HTTP 422 Unprocessable Entity*. Para la obtención del ranking (*GET /top10*), se confirmó que los datos recibidos estuvieran correctamente serializados en formato JSON, ordenados descendentemente, y que el juego interpretara correctamente una respuesta vacía si la base de datos no contenía registros. La verificación de cada prueba se realizó contrastando la respuesta en la consola de Godot (códigos de estado y datos JSON) con la inspección directa de la base de datos MariaDB, asegurando así la integridad de los datos y un comportamiento predecible del sistema ante diversos escenarios.

VI. Planificación de Tareas

Con el fin de organizar el desarrollo del proyecto en un período de dos meses, se estableció una planificación que incluye tanto la finalización de las características pendientes en el juego como la integración de los componentes IoT.

En la Tabla I se detalla la planificación con las actividades específicas de cada fase y los recursos tecnológicos empleados. Cabe aclarar que todas las tareas fueron desarrolladas de manera individual por el autor del proyecto.

TABELA I
Planificación de tareas del proyecto Dino Runner

Actividad	Recursos tecnológicos
Fase 1: Finalización de mecánicas principales	
Implementar el cambio de fondo y suelo al cambiar de bioma	Godot Engine, GDScript
Diseñar y programar obstáculos temáticos por bioma	Godot Engine, LibreSprite, assets gratuitos de itch.io
Agregar menú de pausa e interfaz de usuario mejorada	Godot Engine (UI nodes), fuentes retro
Implementar power-ups (vida extra, multiplicador, ralentización)	Godot Engine, GDScript
Animaciones de daño, corazones y feedback visual	Godot Engine (AnimationPlayer), LibreSprite
Fase 2: Integración y mejoras de jugabilidad	
Optimización del sistema de aparición de objetos y enemigos	Godot Engine (profiler interno)
Implementar efectos visuales (Efecto pantalla vieja CRT, screen shake, etc)	Godot Engine, shaders
Ajuste de dificultad progresiva y tiempos de gracia	Godot Engine (lógica del juego)
Incorporar sonidos retro para acciones y eventos del juego	Sfxr
Primer publicación del juego en *itch.io* para feedback de jugadores	itch.io, navegador web, herramientas de exportación de Godot
Fase 3: Integración IoT y despliegue	
Diseño e implementación de la base de datos de jugadores	MariaDB, VPS con Debian
Desarrollo de la API para comunicación con el juego	Flask (Python), Godot HTTPRequest
Creación de la aplicación web de presentación	Flask, HTML, CSS, Bootstrap, JavaScript
Publicación de la actualización del juego en *itch.io*	itch.io, navegador web, herramientas de exportación de Godot

Finalmente, en el último mes se desarrollará la aplicación web, se integrará la base de datos y la API en contenedores Docker, y se publicará el juego en *itch.io*, dejando el sistema completo en funcionamiento.

VII. Conclusiones

El proyecto Dino Runner con extensión IoT demuestra cómo un videojuego sencillo puede convertirse en una plataforma de integración tecnológica. Se logró combinar el desarrollo en Godot con herramientas de backend modernas, generando un ecosistema en el que la jugabilidad y la conectividad coexisten. Más allá de los logros técnicos alcanzados, el proyecto también

demuestra un gran valor educativo. Sirve como un laboratorio práctico en el que se aplican de manera integrada conceptos de programación de videojuegos, diseño de arquitecturas web, uso de contenedores y gestión de bases de datos. Esta combinación convierte al proyecto en una herramienta formativa que no solo ofrece un producto lúdico, sino también un espacio de experimentación y aprendizaje aplicable a otros entornos tecnológicos y académicos.

Referências

[1] GDQuest, "Make a 2D Endless Runner in Godot," YouTube, 2023. [Online]. Available: https://www.youtube.com/watch?v=nKBhz6oJYsc&t=98s
[2] SFXR, "Sound effect generator," [Online]. Available: https://sfxr.me/
[3] The Pixel Nook, "Parallax Backgrounds Demo," itch.io, 2022. [Online]. Available: https://the-pixel-nook.itch.io/parallax-backgrounds-demo
[4] Free Game Assets, "Free Swamp 2D Tileset Pixel Art," itch.io, 2022. [Online]. Available: https://free-game-assets.itch.io/free-swamp-2d-tileset-pixel-art
[5] Arks, "Dino Characters," itch.io, 2022. [Online]. Available: https://arks.itch.io/dino-characters?download
[6] Nastanliev, "Flying Eyes," itch.io, 2022. [Online]. Available: https://nastanliev.itch.io/flying-eyes
[7] DaFont, "Retro Gaming Font," 2022. [Online]. Available: https://www.dafont.com/retro-gaming.font
[8] Godot Engine Documentation, "Making HTTP requests," [Online]. Available: https://docs.godotengine.org/en/stable/tutorials/networking/http_request_class.html
[9] Godot Engine Documentation, "Index," [Online]. Available: https://docs.godotengine.org/en/stable/index.html
[10] Flask, "Flask Documentation (stable)," Pallets Projects, 2023. [Online]. Available: https://flask.palletsprojects.com/en/stable
[11] MariaDB Foundation, "MariaDB Documentation," [Online]. Available: https://mariadb.org/documentation
[12] Bootstrap Core Team, "Bootstrap Documentation," Bootstrap, 2025. [Online]. Available: https://getbootstrap.com/docs/
[13] LibreSprite Team, "LibreSprite," [Online]. Available: https://libresprite.github.io/