**LETHAL RUN**

Institut Pedralbes

Izan De La Cruz
David Salvador
Marc Rojano
Brian Orozco

# PRESENTACIÓ TÈCNICA

# 1. Evolució de les Funcionalitats al llarg dels Sprints

## Sprint 1 (20/1/25 - 31/1/25)

### Funcionalitats principals:

- Definició de la idea i estructura del projecte.
- Creació del repositori Git.
- Implementació inicial del mapa interactiu i moviment del personatge.
- Configuració de bases de dades (SQL i MongoDB).
- Sistema de login bàsic amb Google i registre per email.

## Sprint 2 (3/2/25 - 14/2/25)

### Funcionalitats principals:

- Menú d'inici i pausa.
- Mode multijugador bàsic amb Steam:
- Integració de Steamworks API per a autenticació i gestió de sessions.
- Creació de lobbies i sincronització de jugadors mitjançant Steam.
- Checkpoints i interacció amb trampes.
- Pujada d'imatges amb connexió front-back (sockets al backend).

Izan De La Cruz
David Salvador
Marc Rojano
Brian Orozco

## Sprint 3 (31/3/25 - 11/4/25)

### Funcionalitats principals:

- Reorganització del codi i adopció d'arquitectura de microserveis.
- Investigació d'animacions (emotes) i gràfics (graffitis).
- Optimització de scripts i preparació per a escalabilitat.
- Connexió web-backend:
- Ús de WebSockets per a actualitzacions en temps real (p. ex., canvis d'imatges, notificacions).

## Sprint 4 (21/4/25 - 2/5/25)

### Funcionalitats principals:

- Redisseny de la UI (menús, login, selecció de mapes).
- Implementació de sales públiques mitjançant Steam (llistat de partides actives).
- Separació definitiva de microserveis i automatització de processos.
- Comunicació front-back:
- Sincronització de dades no crítiques (p. ex., perfils d'usuari) via API REST i WebSockets.

## Sprint 5 (5/5/25 - 21/5/25)

### Funcionalitats principals:

- Multijugador final amb Steam:
- Sincronització de skins i graffitis entre jugadors.
- Connexió web-backend:
- Implementació de notificacions en temps real per a desbloqueig de cosmètics (WebSockets).
- UI definitiva i revisió del flux d'usuari.
- Implementació de HTTPS i seguretat amb tokens.

# 2. Problemes i Solucions

**LETHAL RUN**

Izan De La Cruz
David Salvador
Marc Rojano
Brian Orozco

# Problemes tècnics:

## Integració de Steamworks (Sprint 2 i 3):

- Problema: Dificultats en la sincronització de dades entre jugadors (ex: posició del personatge).
- Solució: Ús de Steam Netcode per a transmissió eficient de dades crítiques (moviment, accions).

## Sincronització d'animacions en multijugador (Sprint 2):

- Problema: Les animacions no es reproduïen de manera coherent en tots els clients.
- Solució: Implementació de Client RPC (Remote Procedure Call) per a activar animacions de manera sincronitzada a través de la xarxa.

## Sincronització de trampes (Sprint 2):

- Problema: Les trampes no afectaven a tots els jugadors simultàniament.
- Solució: Assignació de Network Identity i Network Transform als objectes de les trampes, garantint que el seu estat i moviment es repliquin a tots els clients.

## Teleportar jugadors sense errors de coordinació (Sprint 4):

- Problema: El teletransport provocava conflictes amb el CharacterController.
- Solució: Ús de SyncVar per a sincronitzar la posició i desactivar temporalment el CharacterController durant el teletransport per evitar forces residuals.

## Filtrat incorrecte de sales de Steam (Sprint 3):

- Problema: Les sales mostraven partides d'altres jocs o estats invàlids.
- Solució: Filtre de sales mitjançant flags personalitzades (identificador del joc), escena activa i nombre màxim de jugadors.

Izan De La Cruz
David Salvador
Marc Rojano
Brian Orozco

### Sales fantasma (Sprint 4):

- Problema: Les sales eliminades seguien apareixent al llistat.
- Solució: Implementació d'un sistema de checks periòdics (cada 30 segons) per eliminar sales inactives o sense jugadors.

### Accés a partides ja iniciades (Sprint 4):

- Problema: Els jugadors podien unir-se a partides en curs.
- Solució: Assignació d'un Game State (estat de la partida) que bloqueja les sales un cop la partida ha començat.

### Limitació de creació de hosts (Sprint 3):

- Problema: Els jugadors podien crear múltiples hosts en menys de 60 segons.
- Solució: Integració d'un cooldown al Network Manager que bloqueja la creació de nous hosts fins que passin 60 segons.

### Errors en canvis d'escena (Sprint 5):

- Problema: La UI i els serveis en línia es desactivaven en tornar a escenes anteriors.
- Solució: Ús de DontDestroyOnLoad per a objectes crítics (ex: Network Manager).

### Reinicialització de la UI Toolkit al mètode Start() de cada escena.

- Destrucció controlada del Network Manager en sortir del joc per evitar crides residuals a Steamworks.
-

### Comunicació web-backend (Sprint 3):

- Problema: Latència en les actualitzacions de la UI web (p. ex., pujada d'imatges).
- Solució: Implementació de WebSockets al backend per a notificacions immediates.

Izan De La Cruz
David Salvador
Marc Rojano
Brian Orozco

## Seguretat del login (Sprint 1 i 5):

- Problema: Vulnerabilitats en l'emmagatzematge de contrasenyes.
- Solució: Hash de contrasenyes amb bcrypt i autenticació amb tokens JWT.

## Traspàs del backend a microserveis (Sprint 3):

- Problema: Traspasar tot el back a microserveis.
- Solució: Reestructuració del backend en diversos serveis independents (auth, lobby, matchmaking, etc.), cadascun amb el seu propi punt d'entrada i responsabilitat. La comunicació entre serveis es realitza mitjançant peticions HTTP a través d'API REST, cosa que ha facilitat l'escalabilitat, el manteniment modular i la detecció d'errors específics per servei.

## Automatitzar registre (Sprint 4)

- Problema: Necessària la implementació de diversos microserveis per crear un usuari.
- Solució: Automatització del procés complet de creació d'usuari: inserció a la base de dades SQL, generació de la referència d'una imatge per defecte, i creació d'un objecte a MongoDB per gestionar les seves skins associades al correu electrònic. Aquesta automatització garanteix la coherència entre serveis i redueix la possibilitat d'errors humans o desincronitzacions.

## Implementació de sockets mitjançant NGINX en producció (Sprint 5):

- Problema: Implementar sockets mitjançant NGINX a producció.
- Solució: Reconfiguració del backend per separar les rutes de peticions API i les connexions en temps real. S'ha adaptat el servidor perquè pugui gestionar les peticions REST a través d'una ruta específica (ex: /auth-service) i, paral·lelament, acceptar connexions per Socket.IO des d'una altra ruta, assegurant la compatibilitat amb la configuració de proxy invers d'NGINX.

Izan De La Cruz
David Salvador
Marc Rojano
Brian Orozco

### Configuració del login amb Google i verificació de correu (Sprint 2):

- Problema: Calia configurar el login amb Google i la verificació de correu per completar el procés de registre.
- Solució: Es va configurar correctament el projecte a la consola de Google Cloud, generant les claus OAuth 2.0 necessàries i activant les API corresponents. A més, es va implementar la integració amb els serveis de Google per enviar correus electrònics de verificació automàtics als usuaris durant el registre, assegurant així la validesa del correu electrònic abans de completar el procés d'autenticació.

## Problemes d'UX:

### UI poc intuïtiva (Sprint 4):

- Problema: Confusió en la navegació entre sales de Steam i la web.
- Solució: Disseny d'un flux unificat amb indicadors visuals clars (icons d'estat, colors per a sales actives/inactives).

### Afegir grafitis (Sprint 3):

- Problema: Va haver-hi problemes per trobar o implementar un croper d'imatges que permetés retallar amb relació d'aspecte 1:1
- Solució: Es va adoptar vue-cropperjs, que ofereix eines per configurar diferents formats de retallada amb flexibilitat i precisió.

### Gashapon (Sprint 5):

- Problema: El gashapon es percebia com un simple botó, mancant d'atractiu visual i sensació de premi.
- Solució: Disseny i implementació d'una animació i estètica personalitzada per simular una experiència més propera a una màquina real de gashapon.

Izan De La Cruz
David Salvador
Marc Rojano
Brian Orozco

# 3. Aspectes Tècnics i Requisits

## Tecnologies Utilitzades:

### Unity (Multijugador):

- Motor principal del joc desenvolupat amb **Unity**.
- **Mirror Networking** per a la gestió del multijugador, integrat amb **Steamworks API** per a:
  - Autenticació mitjançant Steam ID.
  - Gestió de lobbies i connexions entre jugadors.
  - Xat de veu i sincronització d'estat del joc.
- Implementació de la lògica del joc en **C#**, incloent moviment, interaccions, animacions i sincronització de trampes i accions crítiques.

### Frontend (Web):

- Desenvolupat amb Nuxt 3 (framework de Vue 3 amb SSR). Comunicació amb el backend mitjançant fetch per a crides REST i renderització reactiva.
- Disseny modular i responsive amb integració de funcionalitats com: login, gestió de perfil, sistema de pujada d'imatges i visualització de *skins*.
- Actualitzacions dinàmiques de l'UI a través de WebSockets quan cal (p. ex. canvis d'estat o notificacions).

### Backend:

- Desenvolupat amb Node.js i Express.js per a les rutes API.
- Comunicació en temps real amb Socket.IO per gestionar esdeveniments com actualitzacions de perfil, xats, etc.

- Arquitectura orientada a microserveis, amb serveis independents per autenticació, gestió de lobbies, partida i imatges d'usuari.
- Gestió i emmagatzematge d'imatges estàtiques d'usuari (com avatars o grafitis) a nivell de servidor per accés ràpid i referenciació des del frontend.

### Bases de Dades:

#### MySQL:

- Utilitzat per a dades estructurades com usuaris, credencials, permisos, i referències a les seves imatges (avatars, grafitis).
- Aquesta estructura facilita accés eficient a les dades essencials des de qualsevol microservei.

#### MongoDB:

- Utilitzat exclusivament per gestionar el conjunt de *skins* dels usuaris.
- Cada document està associat amb el correu electrònic de l'usuari, permetent una flexibilitat dinàmica i fàcil accés per a la personalització visual.

## Requisits Funcionals:

### Multijugador (Unity + Mirror + Steamworks):

- Creació i unió de partides multijugador mitjançant Steam, amb sincronització d'estats a través de Mirror Networking.
- Sincronització en temps real d'accions crítiques com moviment, interaccions, activació de trampes i animacions de personatge.

- Gestió d'estats de partida i bloqueig d'accés per evitar que jugadors s'uneixin a partides ja iniciades.
- Sincronització d'escenes i transicions entre fases del joc per a tots els clients connectats.

### Web (Nuxt 3 + fetch + WebSockets):

- Visualització i edició de perfil d'usuari, incloent el canvi de nom, correu i imatge d'avatar.
- Sistema de login clàssic i amb Google, amb verificació per correu mitjançant serveis de Google.
- Pujada d'imatges i personalització mitjançant un cropper amb relació d'aspecte 1:1 (vue-cropperjs), amb assignació automàtica de referència a la base de dades.
- Visualització i selecció de skins disponibles per a cada usuari (carregades des de MongoDB en base al seu email).
- Actualitzacions en temps real de la UI (notificacions, estat de partides, canvis visuals) mitjançant WebSockets.
- Gashapon interactiu amb animació i estètica personalitzada, no només un botó genèric.
- Sincronització entre dades del joc i el web, mantenint coherència entre el perfil Steam i la web.

## Requisits No Funcionals:

### Rendiment:

- Latència inferior a 100ms per a accions de joc multijugador mitjançant Mirror i Steam Netcode, assegurant una experiència fluida entre jugadors.

- Temps de resposta del backend <300ms per a operacions habituals (login, consulta de perfil, llistats de skins, etc.), gràcies a la separació de responsabilitats entre microserveis.
- Carregues dinàmiques al frontend amb Nuxt 3 optimitzades mitjançant lazy loading de components i ús de cache local per reduir sol·licituds innecessàries.
- Actualitzacions en temps real eficients gràcies a la integració de WebSockets i gestió selectiva d'esdeveniments (només es transmeten els canvis essencials a cada client).

## Escalabilitat i mantenibilitat:

- Arquitectura basada en microserveis, que permet desenvolupar, desplegar i escalar funcionalitats de manera independent (autenticació, imatges, partides, etc.).
- Separació de canals HTTP i WebSocket al backend, amb configuració específica per gestionar rutes REST i connexions persistents via Socket.IO.
- Referències d'imatges guardades a MySQL, mentre que les skins es gestionen en MongoDB, permetent una separació clara entre dades estructurades i contingut dinàmic.

## Seguretat:

- Comunicació encriptada amb HTTPS, utilitzant certificats de Let's Encrypt en entorn de producció.
- Autenticació segura amb tokens JWT, que inclouen informació bàsica de sessió i expiren automàticament per evitar accessos persistents.
- Hash de contrasenyes amb bcrypt per garantir l'emmagatzematge segur de credencials.
- Verificació per correu electrònic durant el registre d'usuaris, mitjançant les APIs de Google Cloud.
- Restricció d'accés a recursos i rutes protegides, tant al frontend com al backend, segons el rol de l'usuari.

Izan De La Cruz
David Salvador
Marc Rojano
Brian Orozco

Disponibilitat i fiabilitat:

- Mecanismes de validació i control d'errors a cada microservei per garantir la coherència entre serveis (per exemple: verificació que l'usuari existeix abans de crear el seu objecte a MongoDB).
- Evita duplicació de comptes o accions crítiques mitjançant cooldowns (temps d'espera entre accions com crear partides).
- Sistema de neteja automàtica de sales fantasma i verificació periòdica per mantenir la integritat de les llistes de partides disponibles.