1. Introducción

1.1. Propósito del documento

El propósito de este documento es proporcionar una guía clara para el diseño y desarrollo del videojuego "Rush Market". Servirá como referencia durante el proyecto, estableciendo las mecánicas del juego y los objetivos de desarrollo.

Además, apoyará el enfoque de Desarrollo Guiado por Pruebas (TDD), asegurando que todos los componentes sean probados y validados adecuadamente.

1.2. Alcance del proyecto

El proyecto abarca la creación de un videojuego multijugador en el que dos jugadores compiten por recoger la mayor cantidad de productos en un supermercado dentro de un tiempo límite, quedando con la victoria el jugador con mayor puntaje.

El juego incluirá un gran y variado mapa, una mecánica de retrasar al rival mediante zancadillas y objetos que se puedan lanzar y un sistema de puntuación, y se desarrollará únicamente para PC.

1.3. Audiencia objetivo

Este documento está dirigido a:

- **Desarrolladores de software**: Los encargados de implementar las funcionalidades descritas, como la mecánica de juego, lógica multijugador, y la gestión de la base de datos.

- **Testers**: Responsables de verificar que todas las características y mecánicas del juego funcionan correctamente de acuerdo a lo descrito en este TDD.

- **Diseñadores de juego:** Que puedan utilizar este documento para entender y afinar las mecánicas clave y las interacciones entre jugadores.

1.4. Definiciones y acrónimos

TDD: Test-driven development (desarrollo guiado por pruebas de software)
CPU: Central processing Unit (unidad central de procesamiento)
GB: gigabytes
RAM: Random access memory (memoria de acceso aleatorio)
PC: personal computer (ordenador personal)
PS5: Playstation 5

1.5. Resumen de juego

El juego es una competencia multijugador ambientada en un supermercado, donde dos jugadores corren contra el tiempo para recolectar el mayor número de productos de una lista de la compra.
A lo largo de la partida, los jugadores pueden interferir en el progreso del rival echándole zancadillas o haciéndolo tropezar, lo que añade una capa de estrategia y diversión al juego. El jugador que logre recoger más productos al final del tiempo será el ganador.

El enfoque está en la rapidez y la interacción entre jugadores, con partidas dinámicas y llenas de acción. El juego está diseñado para ofrecer una experiencia casual y accesible, apta tanto para jugadores novatos como para aquellos que disfrutan de competencias ligeras pero emocionantes.

2. Visión general de Juego

2.1. Descripción general

El juego es un multijugador competitivo en el que dos jugadores compiten dentro de un supermercado para recolectar la mayor cantidad de productos de una lista de la compra en un tiempo limitado. Los jugadores deben moverse rápidamente por los pasillos, recolectar ítems y llenar su carrito, mientras intentan obstaculizar a su rival con mecánicas divertidas, como echarle zancadillas para hacerlo caer y retrasar su progreso.

La clave del juego radica en equilibrar la rapidez para completar la lista de productos con el uso estratégico de las zancadillas y otras interacciones para ganar ventaja sobre el oponente. El entorno del supermercado está diseñado con varios

obstáculos, lo que añade un componente de habilidad para navegar eficientemente por los pasillos.

2.2. Género del juego

El juego pertenece al género de acción competitiva casual, con elementos de carrera y sabotaje multijugador.

Se enfoca en la interacción rápida y directa entre los jugadores, con un tono ligero y accesible. Los jugadores deben competir por recolectar ítems mientras utilizan tácticas simples para interferir en el progreso del oponente, lo que añade un componente estratégico y de comedia.

2.3. Jugadores objetivos

El juego está dirigido a jugadores casuales que disfruten de partidas rápidas, además también está dirigido a jugadores que disfruten de los juegos competitivos ligeros con estrategias sencillas como la recolección de objetos y tácticas para obstaculizar al rival.

A la par, los jugadores que gocen de juegos sociales o familiares son objetivos de “Rush Market” gracias a sus partidas rápidas y divertidas.

2.4. Requisitos mínimos y recomendados del sistema

Requisitos mínimos:

- Sistema Operativo: Windows 10 (64-bit)
- CPU: Intel Core i3 o equivalente
- Memoria: 4 GB RAM
- Gráficos: Intel HD 4000 o superior
- Almacenamiento: 2 GB disponibles

Requisitos recomendados:

- Sistema Operativo: Windows 10 (64-bit)
- CPU: Intel Core i5 o superior
- Memoria: 8 GB RAM
- Gráficos: NVIDIA GTX 970 o superior

- Almacenamiento: 2 GB disponibles

2.5. Plataformas soportadas

El juego será compatible con la siguiente plataforma:

- **PC**: Soportado en sistemas operativos Windows (Windows 10 y superiores) y macOS (versión 10.15 y superiores). El juego se podrá jugar tanto con teclado y ratón como con mando.

3. Mecánicas de Juego

3.1. Estructura del juego

El juego sigue una estructura basada en rondas con una duración limitada, diseñada para ofrecer partidas rápidas y competitivas. A continuación, se detalla la estructura básica:

- Inicio de la partida:
    - Los jugadores se unen a la partida y se les presenta una lista de productos aleatorios que deben recolectar en el supermercado.
    - La partida comienza con los jugadores posicionados en la entrada del supermercado, con un contador de tiempo visible.

- Desarrollo de la partida:
    - Los jugadores se desplazan por los pasillos del supermercado buscando los productos de su lista.
    - Pueden recoger productos acercándose a ellos y agarrandolos para luego, llevarlos al carrito de la compra.
    - Para interactuar con su oponente, los jugadores tienen la opción de utilizar la mecánica de ataque para atacar al rival y ralentizar su progreso.
    - El entorno del supermercado presenta diferentes obstáculos y oportunidades estratégicas, como pasillos estrechos y zonas con tráfico de NPCs (clientes del supermercado), lo que añade un nivel adicional de dificultad.

- Fin de la partida:
    - La partida termina cuando el jugador termina la lista de la compra antes que el resto de sus rivales .
    - Los puntos de cada jugador se calculan según la cantidad.

    - En caso de que el tiempo de la partida termine y ninguno haya llegado a completar la lista de la compra. Se declara ganador al jugador con más puntos. En caso de empate, se utiliza el tiempo en que cada jugador completó su lista como criterio de desempate.

- Pantalla de resultados:
    - Al finalizar, los jugadores son llevados a una pantalla de resultados donde se muestran las estadísticas de la partida: productos recolectados, ataques realizados, y el tiempo total.

Este ciclo se repite en cada partida, con nuevas listas de productos generadas aleatoriamente, lo que asegura variabilidad y rejugabilidad.

3.2. Reglas de juego

Las reglas del juego están definidas para garantizar una competencia estructurada entre los jugadores.

El objetivo del juego es completar la lista de la compra antes de que se agote el tiempo establecido.

La partida comienza con un contador que delimita la duración del juego. Todos los jugadores inician desde un punto de partida en el supermercado y deben desplazarse para localizar y recoger los objetos necesarios, cada uno de los cuales otorga una cantidad de puntos.

Durante el juego, los participantes tienen la opción de realizar acciones que dificulten el progreso del oponente, lo que puede ralentizar su avance y reducir sus posibilidades de acumular más puntos.

Al finalizar el tiempo, el jugador con la mayor cantidad de puntos es declarado ganador.

3.3. Mecánicas de movimiento y control de personaje

Las mecánicas de movimiento y control de los personajes están diseñadas para ser intuitivas y responsivas, permitiendo a los jugadores navegar de manera efectiva por el entorno del supermercado. A continuación se describen las mecánicas clave:

1. Controles básicos:

- Movimiento: Los jugadores utilizan las teclas de dirección (W, A, S, D) en PC o el joystick izquierdo en mandos de consola para moverse en cuatro direcciones (adelante, atrás, izquierda, derecha).

- Interacción: Los jugadores pueden recoger productos al acercarse a ellos y presionar el botón de interacción (por ejemplo, la tecla "E" en PC o un botón en el mando).

2. Ataque:

- Los jugadores pueden ejecutar un ataque para interferir con el movimiento de su oponente presionando un botón específico (por ejemplo, la tecla "Q" en PC, o “X” del botón de PS5).

- El ataque tiene un rango limitado y se activa al acercarse al oponente. La animación del ataque es fluida y momentánea, permitiendo al jugador continuar moviéndose inmediatamente después.

3.4. Interacciones con el entorno

Las interacciones con el entorno del supermercado son fundamentales para la jugabilidad y añaden una capa adicional de estrategia y diversión. A continuación se detallan las principales interacciones que los jugadores pueden realizar:

1. Recolección de productos:
    - Los jugadores pueden interactuar con estantes de productos para recoger ítems de su lista de compra. Al acercarse a un producto y presionar el botón de interacción, el ítem se añade al carrito del jugador.

    - Algunos productos pueden requerir más tiempo para recoger, añadiendo una dinámica de riesgo y recompensa.

2. Obstáculos físicos:
    - El entorno incluye estantes, carritos de compras y otros objetos que los jugadores deben navegar. Colisionar con estos obstáculos puede

ralentizar el movimiento del jugador y dificultar la recolección de productos.

- Los jugadores pueden usar el entorno a su favor, bloqueando el camino del rival con un carrito o empujando objetos para crear barreras temporales.

3.5. Sistema de puntuación o progresión

El sistema de puntuación y progresión en el juego está diseñado para recompensar la habilidad, la rapidez y la estrategia de los jugadores. A continuación se describen los elementos clave del sistema:

1. Puntos por recolección de productos:

- Cada producto recolectado suma un número determinado de puntos al total del jugador.
    - Azúcar: 2 puntos
    - Manzana: 12 puntos
    - Plátanos: 5 puntos

4. Diseño multijugador

4.1 Uso de Netcode for GameObjects

4.1.1 Estructura de componentes y prefabs

En Rush Market utilizaremos de la siguiente manera los siguientes componentes

- **NetworkObjects**: Estos componentes los utilizaran como por ejemplo, los jugadores y items ya que son datos que manejaremos a través del servidor.

- **NetworkBehaviors**: Estos componentes están definidos en los propios managers del Rush Market ya que se necesitará comunicar y sincronizar el estado de los objetos y las interacciones entre los jugadores a través de la red.

- **Sincronización de estados**: Se utilizan **NetworkVariables** y **NetworkTransforms** en los propios jugadores y sus componentes para poder tratarlos a través del servidor.

4.2 Conexión y sincronización

En este sistema multijugador, la conexión es gestionada por un único host, quien actúa tanto como servidor como jugador. Los demás jugadores se conectan a la partida del host, creando una sala compartida donde todos los participantes pueden interactuar y sincronizar sus acciones en tiempo real.

- **Host**: Es el jugador que inicia la partida y actúa como servidor para los demás. El host maneja toda la lógica central del juego, incluyendo la sincronización de datos y el control del estado de la partida.

- **Clientes**: Son los jugadores que se unen a la sala creada por el host. Los clientes dependen del host para recibir datos sobre el estado del juego y enviar sus propias acciones, como movimientos y actualizaciones de estado, al host para que se sincronicen en la sala.

Este modelo de conexión simplifica la arquitectura del juego, ya que centraliza la lógica y los datos en el host. La sincronización entre el host y los clientes se realiza mediante Netcode for GameObjects, que garantiza que el estado del juego (como posiciones, inventarios y otros datos compartidos) sea consistente y actualizado en tiempo real para todos los participantes.

4.3 Estructura de servidores

4.4 Gestión de jugadores (matchmaking, salas, reconexión)

La gestión de partidas en este videojuego se organiza mediante un sistema de salas o "lobbies", donde un jugador actúa como anfitrión y los demás se conectan a su sala antes de que la partida inicie.

Anfitrión de la sala: Un jugador crea la sala y asume el rol de anfitrión, encargado de iniciar y coordinar la partida. El anfitrión puede controlar la configuración de la sala, como el número de jugadores y las opciones de juego, y tiene la capacidad de iniciar la partida directamente desde el menú de la sala una vez que todos los jugadores estén listos.

Unión de jugadores: Los demás jugadores se unen a la sala del anfitrión mediante un sistema de matchmaking de salas, que muestra las salas disponibles en la pantalla de búsqueda de partidas.

Este sistema utiliza servidores de retransmisión **(Relay servers)** que actúan como puntos de conexión públicos accesibles por todos los jugadores, resolviendo problemas comunes como cambios de red, direcciones IP, traducción de direcciones de red **(NAT)** y firewalls entre jugadores. Cada jugador se conecta a la misma dirección IP y puerto del servidor de retransmisión seleccionado, lo que garantiza que la información de conexión permanezca constante durante toda la sesión de juego.

4.5 Comunicación en el juego (chat de voz, chat de texto)

4.6 Mecanismos de sincronización y latencia

En Rush Market, la sincronización de datos y la gestión de la latencia se abordan mediante el uso de servidores de retransmisión (Relay servers). Este servidor actúa como intermediario en la comunicación entre los jugadores, facilitando el intercambio de mensajes con baja latencia sin que los jugadores se conecten directamente entre sí.

Sincronización de datos: El servidor de retransmisión se ocupará de entregar mensajes entre los jugadores utilizando un intercambio de datagramas de baja latencia.

Gestión de la latencia: Al utilizar este servidor de retransmisión, se abordan problemas comunes como cambios de red, direcciones IP, traducción de direcciones de red (NAT) y firewalls entre jugadores.

Es importante destacar que la comunicación a través de los servidores de retransmisión se realiza en una única región seleccionada por el anfitrión.

5. Diseño técnico

5.1. Arquitectura del software en Unity

5.1.1 Estructura de componentes y prefabs

- **Componentes**

  En Rush Market, utilizamos componentes típicos como el Rigidbody 2D, Collider 2D, entre otros. Además, empleamos componentes específicos orientados al sistema de

red (**Networking**) de las salas, como el **NetworkManager**, que se encarga de gestionar el Netcode para ofrecer un servicio fluido y confiable a los jugadores.

- **GameManagers**

  Los **GameManagers** son esenciales para la gestión de las partidas. En Rush Market, contamos con varios tipos de Managers que desempeñan funciones específicas:

  - **Gestión de ítems:** Un Manager se encarga de administrar los objetos que los jugadores adquieren durante la partida.
  - **Datos del jugador:** Otro Manager está dedicado exclusivamente a gestionar la información de los jugadores, permitiendo el correcto manejo de las salas y sus participantes.
  - **Datos finales:** Este Manager organiza la información final, como los puntos obtenidos por los jugadores, para ser reflejados en la escena del podio.

  Gracias a estos Managers, aseguramos un control eficiente de todos los aspectos de la partida.

- **Prefabs**

  En Rush Market contamos con **prefabs** únicos para los ítems y el jugador, los cuales se integran con el sistema de **Netcode** para ser reutilizados dentro de la partida.

  Además, utilizamos **Prefab Variants**, lo que permite crear variaciones de un mismo prefab base sin necesidad de repetir el proceso desde cero. Por ejemplo:

- El prefab base de los ítems incluye todas las variables necesarias.
  - A partir de este, generamos variantes específicas como Manzana, Azúcar, etc.

  Este enfoque optimiza la gestión de objetos y minimiza el riesgo de errores futuros, asegurando un flujo de trabajo más eficiente y organizado.

5.2 Lógica del servidor y cliente en Netcode for GameObjects

5.2.1 Funciones de NetworkManager

En Rush Market, utilizamos NetworkManager para gestionar las conexiones y la sincronización del juego multijugador:

1. **Inicialización del NetworkManager**

- ApplicationController: Inicializa el sistema cliente-servidor
- Crea instancias de HostSingleton y ClientSingleton
- Gestiona la autenticación inicial

2. **Gestión de Conexiones**

- Host: HostGameManager maneja:

    - Creación de sala con StartHostAsync()
    - Límite de 4 jugadores máximo
    - Sistema de relay para conexiones P2P
    - Creación de lobbies
- Cliente: ClientGameManager maneja:
    - Conexión a partidas existentes
    - Autenticación del cliente
    - Unión a lobbies mediante códigos

3. **Eventos de Red**
    - OnNetworkSpawn(): Llamado cuando un objeto se instancia en la red
    - OnDestroy(): Limpieza de recursos y desuscripción de eventos

<u>5.2.2 Sincronización de NetworkVariables y NetworkTransforms</u>

1. **NetworkVariables**
   Utilizamos NetworkVariables para sincronizar datos importantes:
- **Datos del Jugador** (PlayerDataList):
    - playerID: ID único del jugador
    - playerNetworkName: Nombre del jugador
    - playerPointsnetwork: Puntuación del jugador
- **Estado del Juego** (GameTimer):
    - networkTimeLeft: Tiempo restante de partida
    - networkIsTimerRunning: Estado del temporizador

2. **NetworkTransforms**
- ClientNetworkTransform:
    - Sincroniza posiciones de jugadores
    - Maneja la autoridad del cliente
    - Actualiza transformaciones en tiempo real
3. **Sistemas de Sincronización**

   a) **Objetos del Juego**:

   - Items y productos sincronizados mediante NetworkObject
   - Sistema de autoridad basado en el propietario
   - Replicación automática de estados

   b) **Lista de Compras**:

   - Sincronización de productos recogidos
   - Actualización de puntuaciones en tiempo real
   - Gestión de inventario compartido

   c) **Sistema de Puntuación**:

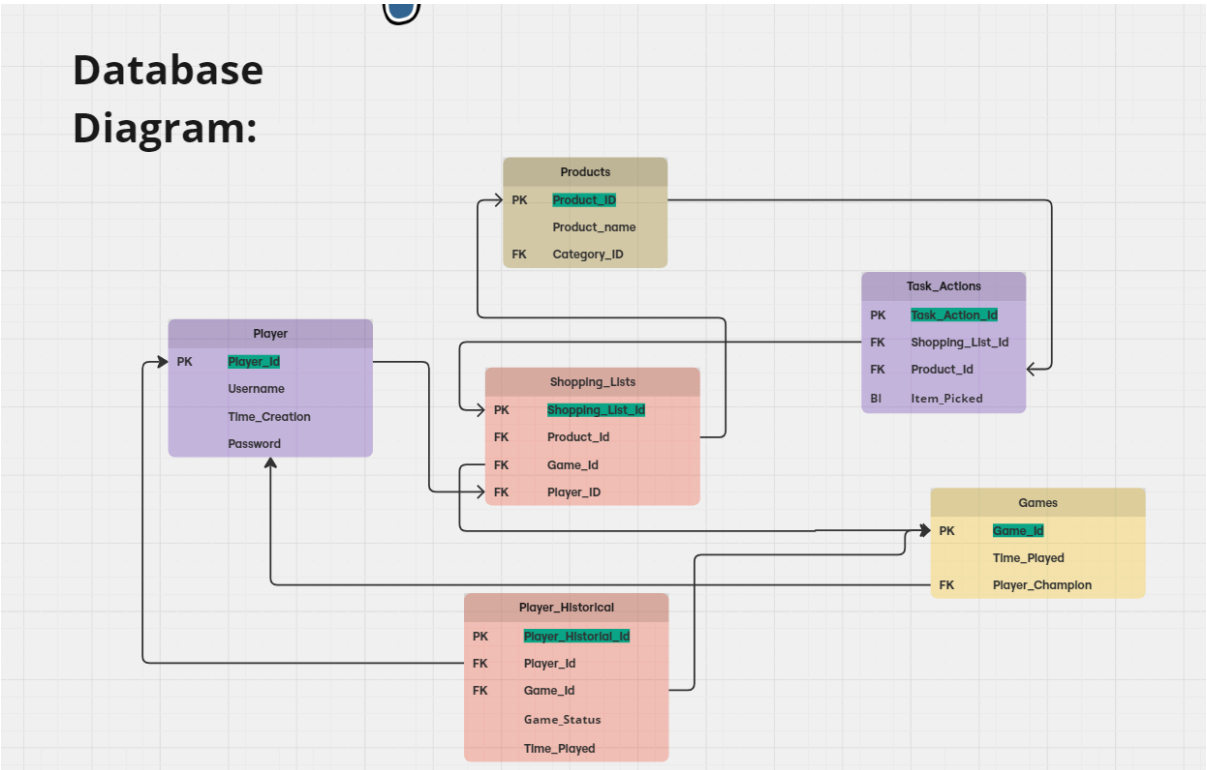
   - Actualización servidor-autoridad
   - Propagación a todos los clientes
   - Verificación de cambios

4. **Permisos y Autoridad**
    - NetworkVariableReadPermission.Everyone: Todos pueden leer
    - NetworkVariableWritePermission.Server: Solo el servidor puede escribir
    - NetworkVariableWritePermission.Owner: Solo el propietario puede escribir

5. **RPCs (Remote Procedure Calls)**
    - ServerRpc: Llamadas del cliente al servidor
        - Actualización de puntuaciones
        - Recogida de items
    - ClientRpc: Llamadas del servidor a los clientes
        - Actualización de UI
        - Sincronización de estado

Esta implementación permite una experiencia multijugador fluida donde:

- Los jugadores pueden competir en tiempo real
- Los datos se mantienen sincronizados
- La autoridad del servidor previene trampas
- El sistema es escalable y mantenible

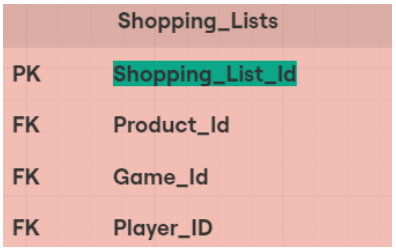


5.3 Base de datos

- **Jugador (Player):** Se conecta con sus listas de compras (**Shopping_Lists**) mediante el campo Player_Id y con el historial de juegos (**Player_Historical**) para registrar su desempeño en cada juego.

- **Juegos (Games):** Está relacionado con los jugadores históricos (**Player_Historical**) a través de Game_Id, y también con las listas de compras (**Shopping_Lists**) para asociar las compras realizadas en un juego específico.

- **Productos (Products):** Están vinculados a través de las listas de compras (**Shopping_Lists**) y las acciones de tarea (**Task_Actions**), lo que permite gestionar qué productos fueron seleccionados o utilizados en los juegos.



- **Listas de Compras (Shopping_Lists):** Actúa como un punto central conectando a jugadores (**Player**), productos (**Products**) y juegos (**Games**) para registrar las compras realizadas por cada jugador en cada juego.

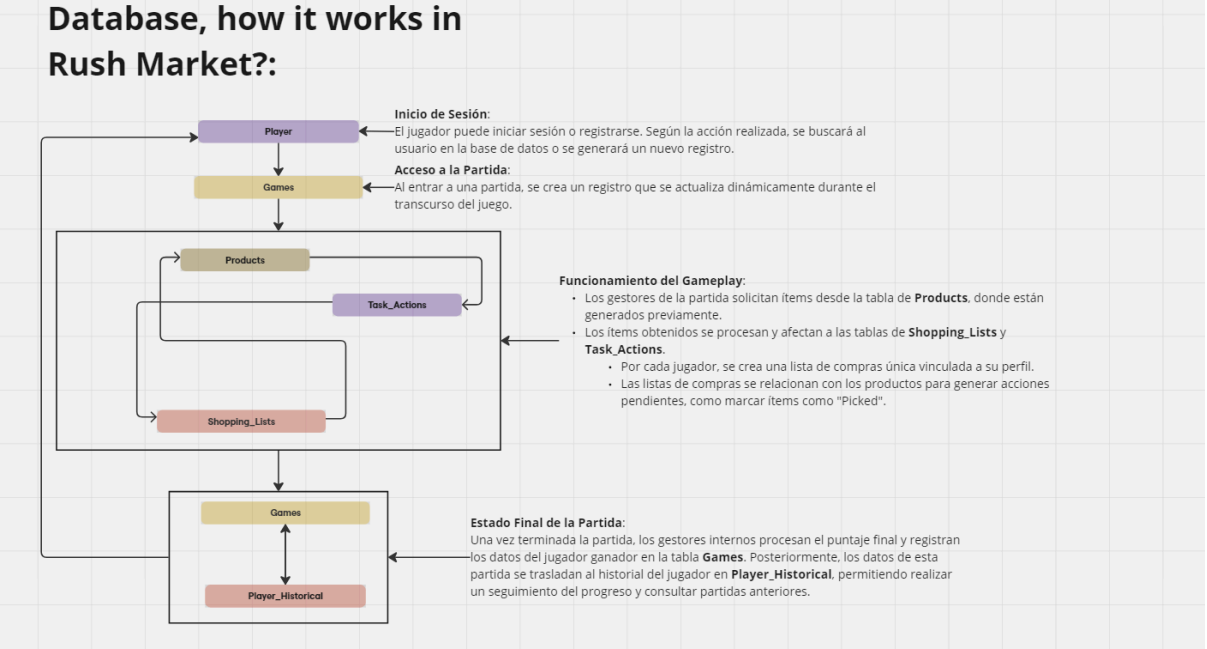
Task_Actions	
PK	Task_Action_Id
FK	Shopping_List_Id
FK	Product_Id
BI	Item_Picked

- **Acciones de Tarea (Task_Actions):** Se relaciona con las listas de compras (**Shopping_Lists**) y los productos (**Products**) para registrar acciones específicas (como si un producto fue seleccionado) realizadas en las listas de compras.

Player_Historical	
PK	Player_Historial_Id
FK	Player_Id
FK	Game_Id
	Game_Status
	Time_Played

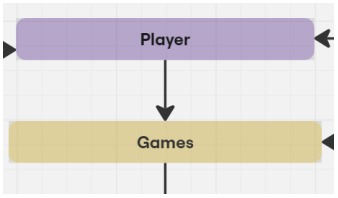
- **Historico del jugador (Player_Historial):** Se relaciona con los datos del jugador (Player) y la partida (Games) para poder generar un historial de partida del jugador.

5.3.1 Cómo funciona internamente la base de datos



- **Paso a paso:**

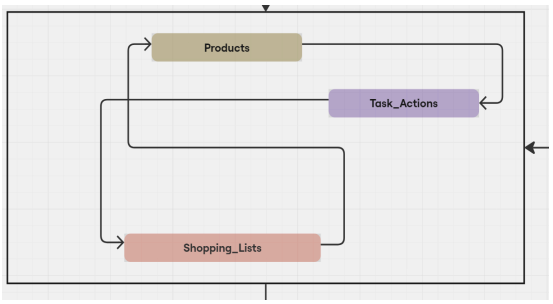
Inicio de Sesión y Acceso a la Partida:



El jugador puede iniciar sesión o registrarse. Según la acción realizada, se buscará al usuario en la base de datos o se generará un nuevo registro.

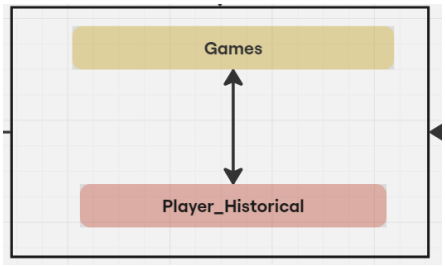
Al entrar a una partida, se crea un registro que se actualiza dinámicamente durante el transcurso del juego.

Funcionamiento del Gameplay:



- Los gestores de la partida solicitan ítems desde la tabla de **Products**, donde están generados previamente.
- Los ítems obtenidos se procesan y afectan a las tablas de **Shopping_Lists** y **Task_Actions**.
    - Por cada jugador, se crea una lista de compras única vinculada a su perfil.
    - Las listas de compras se relacionan con los productos para generar acciones pendientes, como marcar ítems como "Picked".

Estado Final de la Partida:



Una vez terminada la partida, los gestores internos procesan el puntaje final y registran los datos del jugador ganador en la tabla **Games**. Posteriormente, los datos de esta partida se trasladan al historial del jugador en **Player_Historical**, permitiendo realizar un seguimiento del progreso y consultar partidas anteriores.

5.4.2 Gestión de perfiles y progreso de jugadores

- **Red social Rush Market**

    En la pantalla del podio, los jugadores tendrán acceso a un botón de Twitter que les permitirá compartir sus estadísticas de partida de una manera rápida y visual. Este botón ofrece una forma sencilla y

atractiva de interactuar con redes sociales directamente desde el juego. A continuación, se explica cómo funciona:

Funcionamiento del Botón de Twitter

1. **Generación Automática de la Imagen de Stats:**
   Al final de la partida, el juego creará automáticamente una imagen personalizada con las estadísticas del jugador. Esto incluirá datos como:

   - Puntos obtenidos.
   - Posición final en la partida.
   - Ítems recogidos.
   - Tiempo total jugado.

2. **Interfaz de Usuario Intuitiva:**
   En la pantalla del podio, aparecerá un botón con el ícono de Twitter y un texto descriptivo, como "Compartir en Twitter". Este botón estará situado de forma prominente para facilitar su uso.

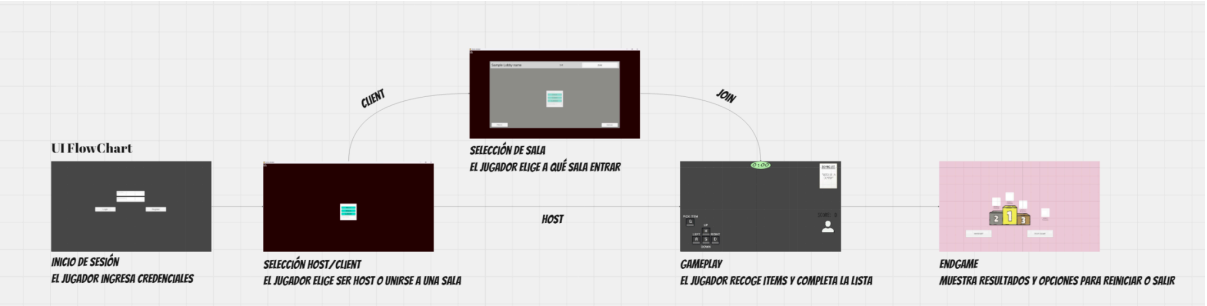
3. **Publicación en Twitter:**
   Al presionar el botón, se activará la funcionalidad para:

   - Redirigir al jugador a Twitter (o abrir una ventana emergente en el navegador).
   - Incluir automáticamente la imagen generada y un mensaje predeterminado, como: "¡He terminado en la posición 1° en Rush Market con 1500 puntos! ¿Puedes superarme? Descúbrelo jugando ahora. #RushMarket"

4. **Personalización del Mensaje:**
   Si lo deseas, el jugador podría editar el mensaje antes de publicarlo, añadiendo su toque personal.

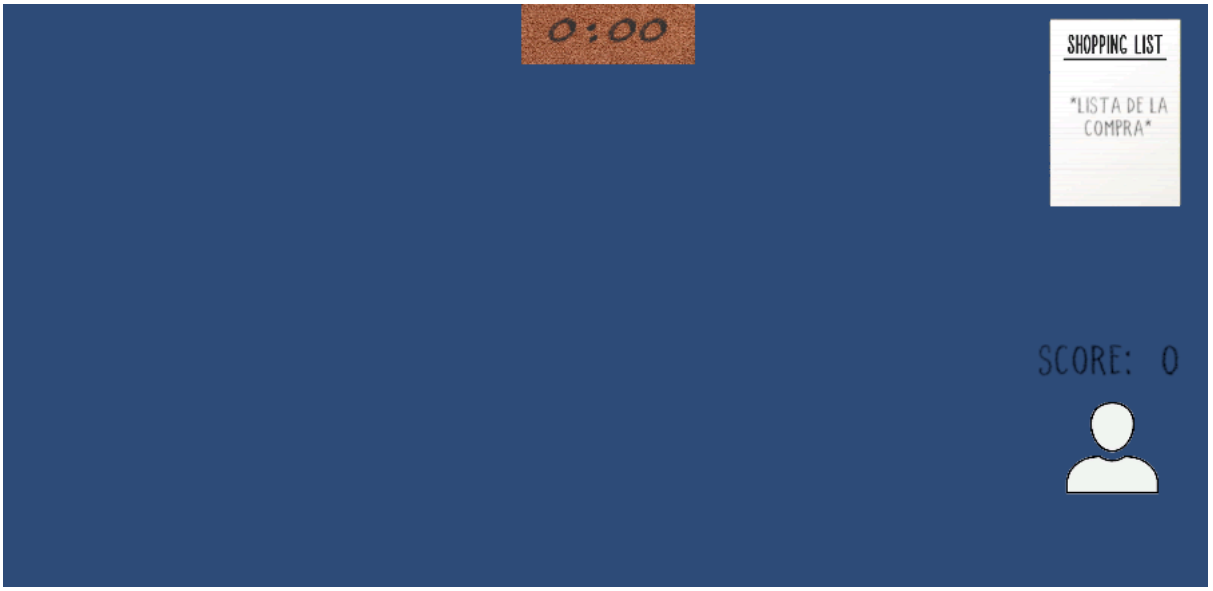
6. Interfaces de usuario

6.3. HUD (Heads-up Display) y elementos en pantalla durante el juego

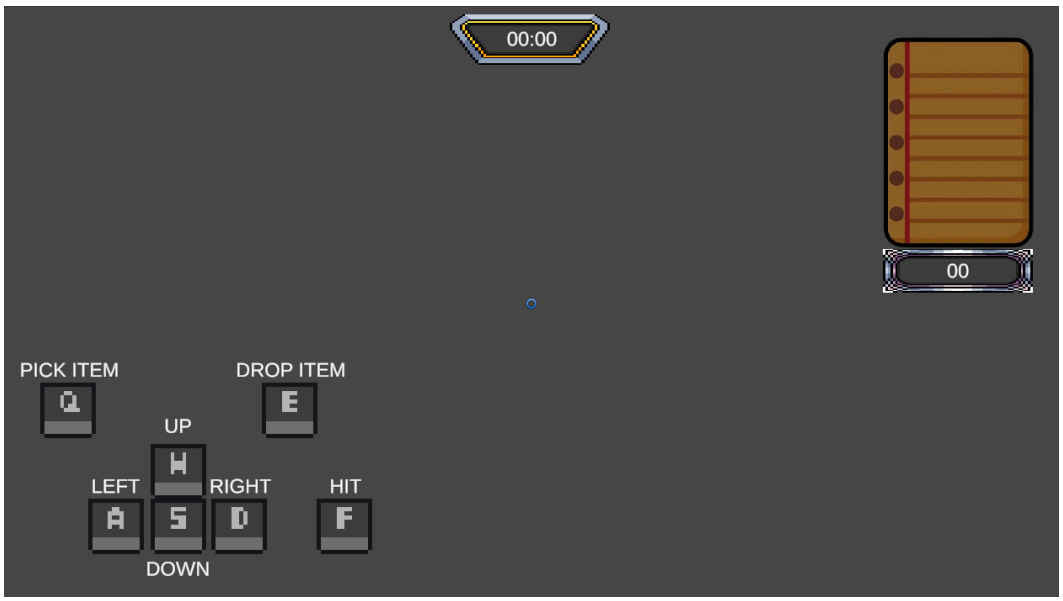


Flow Chart interfaces

Prototipo/Esquema UI In-game



Segundo prototipo UI In-Game



Prototipo Final UI In-Game

7. Gráficos y Sonido

8. Networking

1. **Estructura Principal**
   - Utilizamos Unity Netcode for GameObjects como framework base
   - Arquitectura Cliente-Servidor donde un jugador actúa como host
   - Sistema de relay de Unity para conexiones P2P seguras

2. **Componentes Principales**
   - ApplicationController: Inicializador principal del juego
   - HostSingleton: Gestiona la parte del servidor/host
   - ClientSingleton: Maneja la parte del cliente
   - ClientNetworkTransform: Sincronización de movimientos

3. **Flujo de Conexión**
   - Autenticación inicial del jugador
   - Creación de sala (host) o unión a sala (cliente)
   - Establecimiento de conexión mediante código de unión
   - Sincronización de estado de juego

4. **Características Importantes**
   - Límite de 4 jugadores por partida
   - Códigos de unión para conexión simplificada
   - Sistema de heartbeat para mantener conexiones activas
   - Registro de partidas en base de datos externa

5. **Variables Clave**
   - MaxConnections = 4
   - joinCode (código para unirse)
   - lobbyId (identificador único de sala)

6. **Sistemas de Seguridad**
   - Autenticación anónima mediante Unity Services
   - Verificación de conexiones activas
   - Validación de datos del servidor
   - Protección contra desconexiones

7. **Gestión de Estado**
   - Sincronización automática de posiciones
   - Sistema de autoridad basado en el cliente
   - Actualización en tiempo real del estado de juego
   - Manejo de desconexiones y reconexiones

8. **Integración con Backend**
   - Registro de partidas en PHP/Postgresql
   - Sistema de heartbeat cada 15 segundos
   - Almacenamiento de datos de sesión
   - Tracking de estadísticas

9. Pruebas y Debugging

9.1 Plan de pruebas

El plan de pruebas de "Rush Market" está orientado a garantizar la estabilidad, funcionalidad y rendimiento del juego en sus distintos aspectos: jugabilidad, mecánicas de interacción, interfaz de usuario y rendimiento general. El enfoque principal será verificar que todos los componentes individuales y combinados

(pruebas unitarias y de integración) funcionen según lo esperado, y que el juego mantenga su desempeño en distintas configuraciones de hardware.

El plan de pruebas incluye:

- **Pruebas funcionales**
- **Pruebas de interfaz**
- **Pruebas de rendimiento**
- **Pruebas de momentos o situaciones críticas**

10. Cronograma y Gestión del Proyecto

11. Apéndices

Hecho por Alejandro Martínez, Oscar Peris y Marcos Fargueta

11.1. Bibliotecas, paquetes y frameworks de Unity utilizados

Base de datos utilizada: PostgreSQL ver.:14.13-1

Relay Servers Unity: Relay Servers

How to create social buttons in unity / How to make hyperlinks in unity - LoadC…

11.2. Herramientas de desarrollo y testing

Enlace video playesting en instancias separadas: Playtesting video

Enlace video playesting en la misma instancia: Playtesting Video Misma Instancia

Enlace Guia de playtesting: Playtesting Guia

Enlace Readme guia de como jugar (QR formulario): Guia inicio partida

Enlace formulario Playtesting: Formulario

Repositorio Proyecto: Github

Repositorio Proyecto PHP: Github

Diagrama base de datos / Referencias Proyecto: Miro

Flujo de trabajo o tabla de tareas: Trello