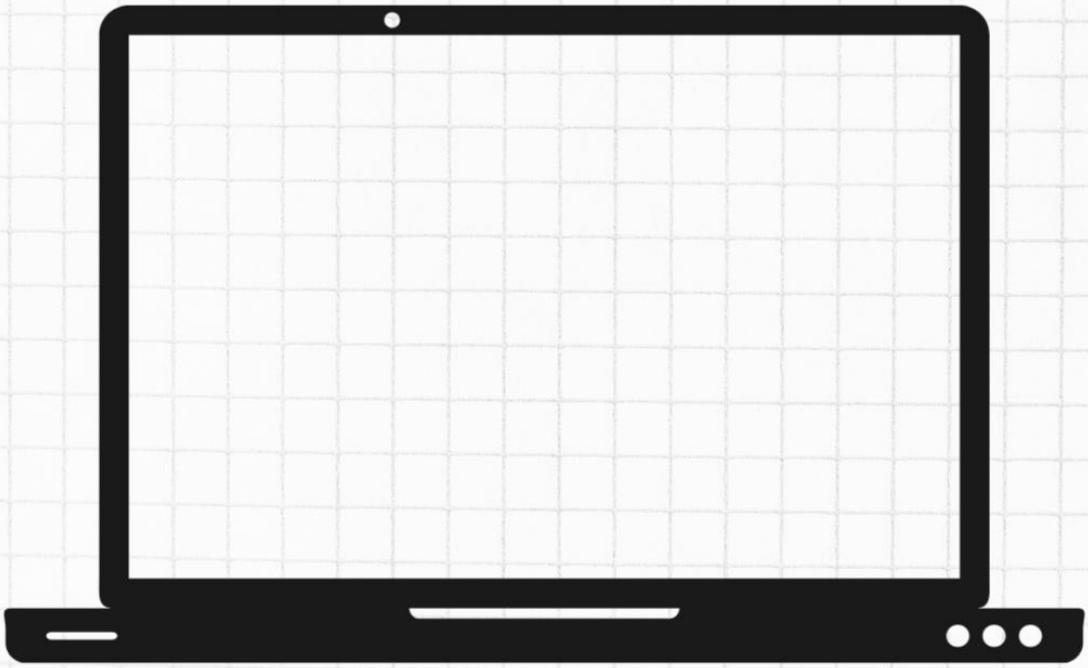
GESTION DE DESARROLLO DE SOFTWARE

EDGAR MISAEL GALAN OROZCO

SELECCIÓN DE DISEÑO ARQUITECTONICO Y DISEÑO DETALLADO "TRABAJO EN EQUIPO"

PROF. SANDRA LILIA TORRES TREJO



1¨Selecciona 2 tecnicas para la recolección de requerimientos de tu proyecto justifica, elabora y evidencia

1. Lluvia de Ideas

Justificación:

Esto nos permitirá como los creadores a qué los usuarios nos den los puntos que pudieran observar y mejorar sobre la marcha, e igual las necesidades que pueda tener el mercado actualmente.

Elaboración:

Reunir a usuarios que consuman el tipo de videojuegos y nos puedan aportar algunas historias que atraigan a los consumidores de este tipo de videojuego, asi como a las edades pertinentes.

Evidencia:

Una lista de conceptos o un mapa mental con ideas como: "uvas explosivas", "abono de sangre zombi" o "barriles de vino como barricadas".

2. Talleres de Diseño

- **Justificación:**

  Acelera la toma de decisiones técnicas y creativas al reunir a programadores, artistas y diseñadores en un solo lugar para resolver conflictos de requisitos.

- **Elaboración:**

  Sesiones para definir la arquitectura del videojuego asi de los conflictos qué pueda tener en el momento de pruebas y sea interactivo con el usuario

- **Evidencia:**

  Un Documento de Diseño de Juego preliminar con acuerdos firmados sobre el motor gráfico y la estructura de niveles, para la compatibilidad con usuarios.

- 2**. Selección del diseño arquitectónico. especifica la arquitectura y justifica la razon para la cual se usará según las necesidades de su proyecto.**

Arquitectura Basada en Componentes

Está arquitectura basada en componentes es un enfoque de diseño de software donde las aplicaciones se construyen a partir de componentes modulares y reutilizables que contienen una funcionalidad específica.

Entidad: El objeto base (Ej: El Jugador).

Componentes: Guiones (scripts) específicos
(Ej: ControladorMovimiento, InventarioCultivo, SaludJugador).

Justificación

Primeramente, revisando algunos foros y propuestas que han hecho anteriormente para este tipo de proyectos se necesita de una arquitectura basada en componentes que permite crear un objeto "Planta" y añadirle un componente de CrecimientoAgricola.si por ejemplo luego se decide que esa planta puede atacar, solo le añadimos un componente DetecciónDeProximidad y otro de Ataque y por lo cuál es más optimizado y no necesitamos reescribir todo el código.

- **3. Diseño detallado. Segun ls que especifica en esté enlace https://sourcemaking.com/design_patterns**

Como nos menciona en el foro anterior en el proyecto se necesitara de los siguientes diseños

1. Fábrica Abstracta

Crea una instancia de varias familias de clases sin especificar sus clases concretas.

Esto se usará para generar el Ecosistema de Temporada el juego necesita crear familias de objetos (semillas, enemigos, clima) que dependan de la estación actual (Verano vs. Invierno), para mejorar aún más los tipos de ambientes y el usuario pueda interactuar con ello.

2. Proxy

Un objeto que representa otro objeto para controlar el acceso o reducir costos.

Aplicación en el proyecto: Se usará para la Gestión de Hordas en el Mundo Abierto (Virtual Proxy). Procesar la IA detallada de 500 zombis que están lejos del viñedo agotaría los recursos.

Patrones de diseño de comportamiento

3. Mediador (Mediator)

Define un objeto que encapsula cómo interactúan un conjunto de objetos, fomentando el acoplamiento débil.

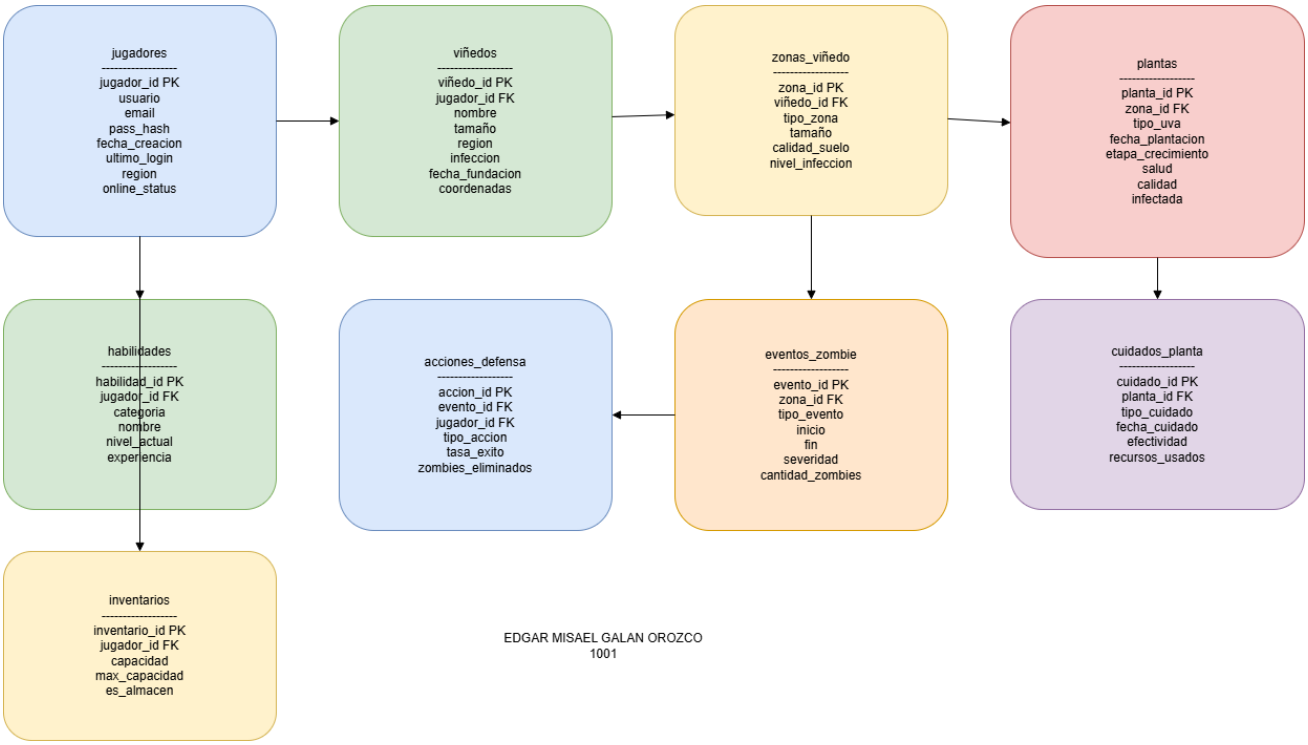
Esto se usará para el Sistema de Alerta del Viñedo en un mundo abierto, muchos sistemas (luces, trampas, perros guardianes, música de tensión) deben reaccionar cuando un zombi entra en el perímetro en lugar de que cada zombi avise a cada objeto, el zombi avisa al MediadorAlerta.

4. Memento

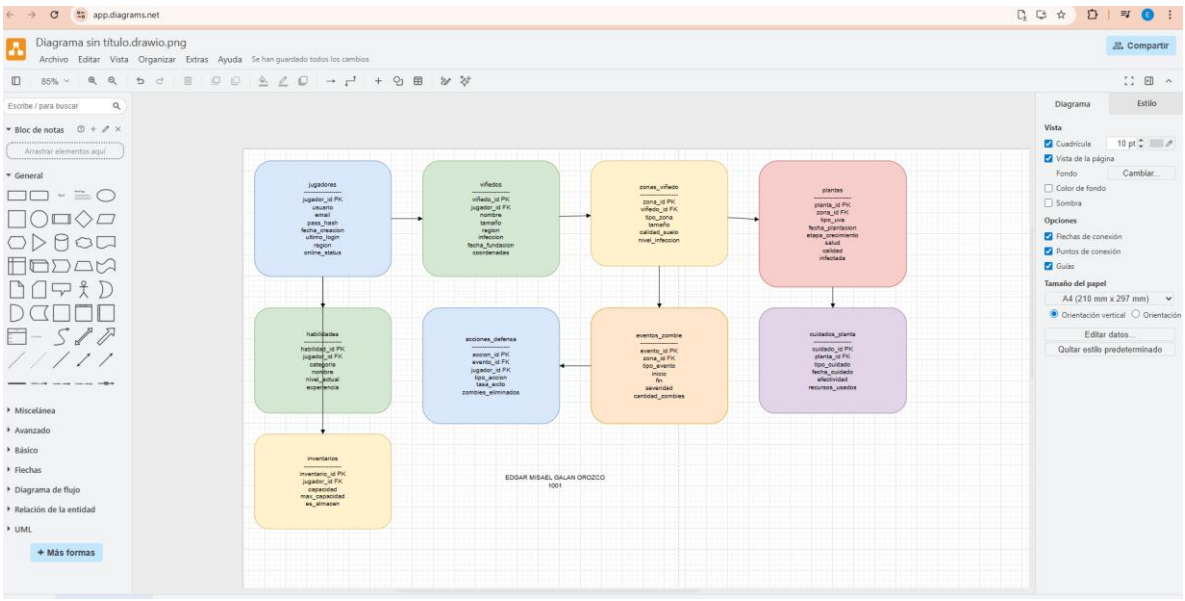
Captura y externaliza el estado interno de un objeto para que pueda restaurarse más tarde sin violar la encapsulación.

En este caso el usuario usa un artefacto para "retroceder el tiempo" en una zona del viñedo que fue destruida.

4. Diseño de la Base de Datos . Justifica modelo de datos, diseño de diagrama entidad relación, diccionario de datos


jugadores
jugador_id PK
usuario
email
pass_hash
fecha_creacion
ultimo_login
region
online_status
viñedos
viñedo_id PK
jugador_id FK
nombre
tamaño
region
infeccion
fecha_fundacion
coordenadas
zonas_viñedo
zona_id PK
viñedo_id FK
tipo_zona
tamaño
calidad_suelo
nivel_infeccion
plantas
planta_id PK
zona_id FK
tipo_uva
fecha_plantacion
etapa_crecimiento
salud
calidad
infectada
habilidades
habilidad_id PK
jugador_id FK
categoria
nombre
nivel_actual
experiencia
acciones_defensa
accion_id PK
evento_id FK
jugador_id FK
tipo_accion
tasa_exito
zombies_eliminados
eventos_zombie
evento_id PK
zona_id FK
tipo_evento
inicio
fin
severidad
cantidad_zombies
cuidados_planta
cuidado_id PK
planta_id FK
tipo_cuidado
fecha_cuidado
efectividad
recursos_usados
inventarios
inventario_id PK
jugador_id FK
capacidad
max_capacidad
es_almacen
EDGAR MISAEL GALAN OROZCO
1001

Creación


app.diagrams.net
Diagrama sin título.drawio.png
Archivo Editar Vista Organizar Extras Ayuda
Compartir
Diagrama
Estilo
EDGAR MISAEL GALAN OROZCO
1001

Justificación:

Decisión clave: MySQL 8.0 relacional en lugar de enfoque NoSQL

Argumentos a fvor:

Naturaleza transaccional del juego: Las operaciones (plantar, cosechar, defender) requieren ACID (atomicidad, consistencia, aislamiento y durabilidad)=

Estructura jerárquica clara: Viñedo → Zonas → Plantas es inherentemente relacional

Consultas complejas predecibles: JOINs para progresión, estadísticas, rankings

Mantenibilidad: Esquema explícito = menos bugs en producción

Ecosistema maduro: Herramientas de backup, replicación, monitorización

Bibliografía

DiCesare, M. (s.f.). Obtenido de https://www.mendix.com/blog/what-is-component-based-architecture/

sourcemaking. (s.f.). Obtenido de https://sourcemaking.com/design_patterns