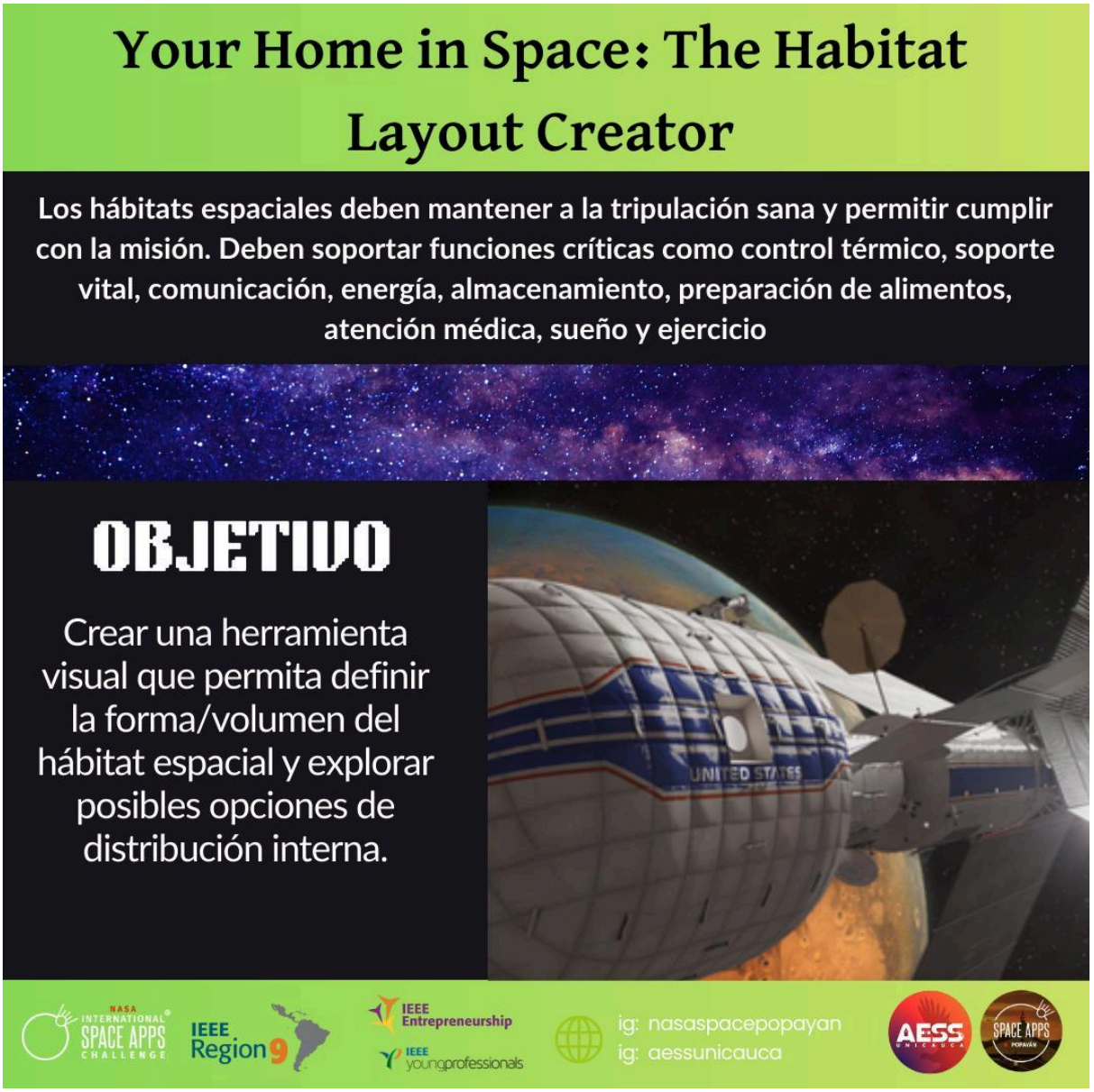

Índice

Introducción: Desafío de Vivir en el Espacio: Los astronautas se enfrentan a toda clase de retos en el espacio, entre ellos la planeación, personalización y optimización del entorno que habitaran en el tiempo de la misión. Esto puede abarcar tanto una estación espacial, una base en algún planeta, una nave para viajes largos, etc. El problema es moldeable a cada necesidad, por lo que se requiere crear una herramienta versátil y generalista.

Planteamiento de la Solución: Se propone la creación de un software que unifica el diseño visual con la simulación de recursos y restricciones que se pueden presentar. El objetivo es crear una herramienta que permita visualizar de forma virtual objetos en un espacio preseleccionado, donde se defina la forma y el volumen del hábitat, asegurando que se cumplan las necesidades básicas sin interferencias o decisiones poco razonables.

Metodología de la Solución: Para la creación de este software se plantea tres pilares que serán fundamentales en el análisis y diseño como serían:

- Diseño modular
- Simulación
- Usabilidad

Análisis de Requisitos: En esta fase se definen los requerimientos funcionales y no funcionales del sistema, desglosando cada pilar:

- Pilar 1: Diseño Modular
    - Requisito 1.1 (Plantillas de Misión): El sistema debe ofrecer al usuario un conjunto de plantillas iniciales que definen el contexto de la misión (Estación Orbital, Base Planetaria, Nave de Viaje). Cada plantilla establecerá restricciones iniciales, como un volumen exterior fijo para la nave o un terreno expandible para la base.
    - Requisito 1.2 (Biblioteca de Componentes): El sistema debe proveer una biblioteca de componentes 3D categorizados (Soporte Vital, Energía, etc.). Cada objeto en esta biblioteca debe tener asociado su modelo 3D, sus puntos de conexión, y sus atributos para la simulación.
    - Requisito 1.3 (Interfaz de Construcción): El sistema debe permitir al usuario seleccionar componentes de la biblioteca y posicionarlos en el entorno 3D de forma intuitiva (arrastrar y soltar). Debe existir un mecanismo para conectar componentes entre sí de manera lógica y visual.
- Pilar 2: Simulación
    - Requisito 2.1 (Modelo de Datos): Cada componente relevante para la simulación debe poseer atributos cuantificables: Masa (kg), Volumen (m^3), Consumo de Recursos (ej: Energía en kW, Agua en L/día) y/o Generación de Recursos.
    - Requisito 2.2 (Motor de Dependencias): El sistema debe tener un motor lógico que verifique constantemente las conexiones. Debe poder determinar si un componente que requiere un recurso (ej: energía) está correctamente conectado a una red que provee dicho recurso.

    - Requisito 2.3 (Cálculo y Retroalimentación): El sistema debe calcular en tiempo real los balances totales de la misión (Masa Total, Balance Energético, etc.) y presentarlos al usuario de forma clara a través de un panel de control o dashboard. El sistema debe generar alertas visuales cuando se incumpla una restricción.
- Pilar 3: Usabilidad
    - Requisito 3.1 (Configuración de Misión): Al inicio, el sistema debe permitir al usuario definir parámetros clave de la misión que afectarán la simulación, principalmente el tamaño de la tripulación y la duración de la misión.
    - Requisito 3.2 (Interfaz Intuitiva): Dado que el público es amplio, la Interfaz Gráfica de Usuario (GUI) debe ser clara, visual y evitar la sobrecarga de información. Se deben usar íconos, colores y descripciones simples.
    - Requisito 3.3 (Navegación y Visualización): El usuario debe poder manipular la vista del hábitat fácilmente (rotar, hacer Zoom) y tener un modo de vista en primera persona para explorar el diseño desde la perspectiva de un astronauta, evaluando la ergonomía y el espacio.

Diseño de la solución: Para la construcción de la arquitectura de software se plantea la implementación de más alto nivel, el uso de espacios de nombre para separar dos lógicas (Back-end y Front-end) estas llevarán la responsabilidad como sería el caso:

Namespaces Lógica de Negocio.

NameSpace: HabitatEngine	NameSpace: HabitatView	NameSpace: HabitatController
Clase Component: Clase LogicComponent: Clase Simulation:		

Namespace HabitatEngine : Clase Component

Creamos el molde para cada componente que tendra atributos genericos como:

- Nombre: String
- ID: int
- Mass: float
- Volume: float
- Importance: int

También creamos una lista de constantes para recursos que se consumirán en la simulación o se tendrán en cuenta. El objetivo es tener recursos generales y posteriormente integrar sin romper la arquitectura.

- Power
- Water
- Oxygen
- Data

Cada recurso usado tendra diferentes propósitos, por lo se crea una mini clase que cumpla este propósito, haciendo que si añádenos más recursos no se modifique la arquitectura si no sea flexible.

Clase ResourceUsage

Teniendo un objeto para definir el tipo y otro para consumo por hora.

- ResourceType Type
- Rate

Creamos un diccionario para la clase Component donde cada componente tendra sus características con el cual acceder.

- DictionaryComponentes <>

Creamos una lista que permita integrar de manera flexible cualquier tipo de recurso

- ListResources <>

Los métodos de la clase Component será su instancia:

- Component() -> DictionaryComponentes <> & ListResources <>

Implementamos métodos para verificar que el componente cargue los datos por default pero los recursos serán manuales, por lo que estos se cargaran cuando se inicialize.

- ComponentState CurrentState() obtenemos el estado
- Initialize() Activa el componente por default

Namespace HabitatEngine : Clase LogicComponent

En listamos los componentes que vamos a usar con los módulos/componente de biblioteca:

1. Control Térmico:
    a. Regulador
    b. Panel Radiador Térmico
2. Soporte Vital:
    a. ECLSS (Generador de Oxígeno)
    b. Purificador
3. Comunicación:
    a. Antena de comunicación
    b. Router Interno
4. Energía:
    a. Generador
    b. Breaker
5. Almacenamiento:
    a. Refrigerador de Alimentos
    b. Tanque de herramientas y equipos
    c. Tanque de agua
6. Preparación de Alimentos:
    a. Hidropónico
    b. Estación Cocina
7. Atención médica:
    a. Estación médica
    b. Kit de primeros auxilios
8. Sueño y Ejercicio:
    a. Litera de astronautas / tripulante
    b. Caminadora espacial

Lógica del grafo por cada componente

1. Energía [Nodo Nivel 0 - Superior]

- a. Componente: Generador
    - Requisitos: Ninguno. Componente raíz del grafo de energía.
    - Lógica: Al activarse, registra la disponibilidad del recurso ResourceType.Power en el HabitatEngine. Sin un Generador activo, ningún componente dependiente de Power puede alcanzar el estado Active.
    - Estado si Falla: No aplica.
- b. Componente: Breaker
    - Requisitos: El recurso ResourceType.Power debe estar disponible (debe existir un Generador activo).

    - Lógica: Actúa como un nodo de protección. Componentes designados como "sensibles" requerirán la presencia de un Breaker activo para evitar un estado de advertencia.
    - Estado si Falla/Advertencia: El componente dependiente entra en estado Warning con el código NoBreakerProtection. La UI debe mostrar el mensaje: "Se recomienda un breaker para la protección del componente {nombre}".

2. Almacenamiento [Nodo Nivel 0 - Superior]

- a. Componente: Tanque de agua
    - Requisitos: Ninguno. Componente raíz del grafo de agua.
    - Lógica: Al activarse, registra la disponibilidad del recurso ResourceType.Water en el HabitatEngine.
    - Estado si Falla: No aplica.
- b. Componente: Refrigerador de Alimentos
    - Requisitos:
        1. Recurso ResourceType.Power disponible.
        2. Recurso ResourceType.Water disponible.
    - Lógica: Este componente requiere una comprobación de múltiples dependencias. El HabitatEngine debe verificar la disponibilidad de Power y Water antes de permitir la activación.
    - Estado si Falla: ComponentState.Disabled. Mensaje de Error: "Requiere acceso a la red de Energía y Agua".

3. Soporte Vital [Nodo Nivel 1 - Importante]

- a. Componente: ECLSS (Generador de Oxígeno)
    - Requisitos:
        1. Recurso ResourceType.Power disponible.
        2. Recurso ResourceType.Water disponible.
    - Estado si Falla: ComponentState.Disabled. Mensaje de Error: "Requiere acceso a la red de Energía y Agua".
- b. Componente: Purificador
    - Requisitos:
        1. Recurso ResourceType.Power disponible.
        2. Recurso ResourceType.Water disponible.
    - Estado si Falla: ComponentState.Disabled. Mensaje de Error: "Requiere acceso a la red de Energía y Agua".

4. Control Térmico [Nodo Nivel 1 - Importante]

- a. Componente: Regulador Térmico
    - Requisitos: Recurso ResourceType.Power disponible.
    - Lógica: Al activarse, este componente habilita la disponibilidad del recurso ResourceType.Data en el hábitat.
    - Estado si Falla: ComponentState.Disabled. Mensaje de Error: "Requiere acceso a la red de Energía".
- b. Componente: Panel Radiador Externo

    - Requisitos: Debe existir al menos un Regulador Térmico con estado Active.
    - Estado si Falla: ComponentState.Disabled. Mensaje de Error: "Requiere un Regulador Térmico para funcionar".

5. Comunicación [Nodo Nivel 2 - Secundario]

- a. Componente: Antena de comunicación
    - Requisitos: Recurso ResourceType.Power disponible.
    - Estado si Falla: ComponentState.Disabled. Mensaje de Error: "Requiere acceso a la red de Energía".
- b. Componente: Router Interno
    - Requisitos: Recurso ResourceType.Power disponible.
    - Estado si Falla: ComponentState.Disabled. Mensaje de Error: "Requiere acceso a la red de Energía".

6. Preparación de Alimentos [Nodo Nivel 2 - Secundario]

- a. Componente: Hidropónico
    - Requisitos:
        1. Recurso ResourceType.Power disponible.
        2. Recurso ResourceType.Water disponible.
    - Estado si Falla: ComponentState.Disabled. Mensaje de Error: "Requiere acceso a la red de Energía y Agua".
- b. Componente: Estación Cocina
    - Requisitos: Recurso ResourceType.Power disponible.
    - Estado si Falla: ComponentState.Disabled. Mensaje de Error: "Requiere acceso a la red de Energía".

7. Atención Médica [Nodo Nivel 3 - Terciario]

- a. Componente: Estación médica
    - Requisitos:
        1. Recurso ResourceType.Power disponible.
        2. Recurso ResourceType.Data disponible.
    - Estado si Falla: ComponentState.Disabled. Mensaje de Error: "Requiere acceso a la red de Energía y Datos".
- b. Componente: Kit de primeros auxilios
    - Requisitos: Ninguno. Componente pasivo.
    - Estado si Falla: No aplica.

8. Sueño y Ejercicio [Nodo Nivel 3 - Terciario]

- a. Componente: Litera de astronautas
    - Requisitos: Recurso ResourceType.Power disponible.
    - Estado si Falla: ComponentState.Disabled. Mensaje de Error: "Requiere acceso a la red de Energía".
- b. Componente: Caminadora espacial
    - Requisitos: Recurso ResourceType.Power disponible.

- Estado si Falla: ComponentState.Disabled. Mensaje de Error: "Requiere acceso a la red de Energía".

Métodos:

Se crea esta clase de tipo interfaz para la lógica de conexión entre nodos con base en como se construyo la lógica anteriormente.

Interface IMethodLogicNodesComponentDefault

Pruebas

Control Térmico:

- **Name:** "Regulador Térmico Habitación 1"
- **Importance:** 8
- **Resources:**
  - { Type: ResourceType.Power, Rate: -2.5f } (consume 2.5 kW/h) (Dependiendo de cuantos equipos están se consumirá una cantidad)
  - { Type: ResourceType.Data, Rate: -0.1f } (consume 0.1 GB/h para logs y telemetría)
- **Properties:**
  - "idealTemperature": 22.0f
  - "affectedZones": ["Habitación 1", “Habitación 2”]