MODELO RELACIONAL

Map = { mapID }
mapID: número de identificação de um mapa que pode ser utilizado por um jogador.

HaveMapRegion = { mapID, coordinates }
coordinates: código de identificação das coordenadas do jogador;

Region = { coordinates*¹, biome, monument, dangerLevel, monument*¹º }
dangerLevel: nível de perigo que a região apresenta;

Biomes = { biomesID*², coordinates*¹, resourceAbundance, resourceAvailability, type, climate*³ }
resourceAbundance: quantidade de recursos que um bioma pode apresentar — em questão de estar abundante ou não — a um ou mais jogadores;
resourceAvailability: apresenta se há ou não recursos no bioma;

Flora = { biomes*², flora }

Fauna = { biomes*², fauna }

Climate = { climateID*³, temperature, event, statusEffect, visibility }

Characters = { charactersID*⁴, name, position, characterModel, climate*³, type, item }

GatherYield = { character*⁴, gatherYield }

EnterCombatCharacters = { enterCombatCharactersID*⁵, firstCharacter*⁴, secondCharacter*⁴ }
firstCharacter: código do primeiro jogador no combate;
secondCharacter: código do segundo jogador no combate;

CombatLog = { enterCombatCharacters*⁵, indexLog, log }

PlayerCharacters = { charactersID*⁴, hydration, poisoned, hunger, type, equipedItems1, equipedItems2, equipedItems3, equipedItems4, equipedItems5, backpack*¹³ }

RecruitableCharacters = { charactersID*⁴, specialization, recruited }

MainCharacter = { charactersID*⁴, owner*⁶ }

RespawmLocation = { ownerID*⁶, description, timer, X, Y }

NPCs = { charactersID*⁴, isAgressive, aggroRange, enemyGrade, type }
isAgressive: código que apresenta se um NPC é agressivo ou não.

Animals = { charactersID*⁴, sound, modelType }

Scientists = { charactersID*⁴, hasDialogue }
hasDialogue: código que apresenta se existe a possibilidade de diálogo com o 'scientist'.

DialogueText = { character*⁴, dialogue }

Items = { itemsID*⁷, stackSize, lootGrade*⁸, name, type, quantity, durability, craftable, type, backpack*¹³, character*⁴ }
craftable: código que apresenta se um item pode ser criado a partir da junção de outros itens por parte dos jogadores.

Ingredients = { items*⁷, lootGrade*⁸, ingredient }

Weapons = { itemsID*⁷, type }

Melee = { itemsID*⁷, canBeThrown, fleshGatherRate, oreGatherRate, treeGatherRate }

Ranged = { itemsID*⁷. recoil, attackRange, amnoCapacity, modSlots, fireMode, fireRate, accuracyModifier }

Consumables = { itemsID*⁷, instantHeal, healOverTime, hidratationYield, bleedingYield, radiationYield, poisonYield, hungerYield, vomitChance, type }
hidratationYield: código que apresenta a quantidade de hidratação que um consumível oferece ao personagem.
bleedingYield: código que apresenta a quantidade de cura de sangramento que um consumível oferece ao personagem.
radiationYield: código que apresenta a quantidade de redução ou aumento de radiação que um consumível oferece ao personagem.
poisonYield: código que apresenta a quantidade de dano ou cura de envenenamento que um consumível oferece ao personagem.
hungerYield: código que apresenta a quantidade de redução de fome que um consumível oferece ao personagem.
vomitChance: código que apresenta a chance que um consumível oferece ao personagem de sofrer a êmese.
Teas = { itemsID*⁷, statusUpgradeType, upgradePercentage }

Clothing = { itemsID*⁷, coldResistance, radResistance, explosionResistance, meleeResistance, rangedResistance, biteResistance, equipmentSlot, wetResistance}
coldResistance: valor que representa a resistência ao frio concedida pelo equipamento
radResistance: valor que representa a resistência à radiação concedida pelo equipamento.
explosionResistance: valor que representa a resistência à dano de explosão concedida pelo equipamento.
meleeResistance: valor que representa a resistência à dano corpo-a-corpo concedida pelo equipamento.
rangedResistance: valor que representa a resistência à dano a distância concedida pelo equipamento.
biteResistance: valor que representa a resistência à dano de mordidas concedida pelo equipamento.
wetResistance: o item concede resistência ao status de molhado. IE: Roupa de radiação, roupa de mergulho.

Components = { itemsID*⁷ }

Resources = { itemsID*⁷, isPrimary }

DropCharactersItems = { dropCharactersItemsID, item*⁷, lootGrade*⁸, character*⁴ }

PlayerCharactersGeneratesItem = { character*⁴, items*⁷ }

WeaponsAreComposedOfComponentsResources = { weapons*⁷, components*⁷, resouces*⁷ }

ConsumablesAreComposedOfComponentsResources = { consumables*⁷, components*⁷, resouces*⁷ }

ClothingAreComposedOfComponentsResources = { clothing*⁷, components*⁷, resouces*⁷ }

ResourceNodes = { resourceNodesID*⁹, nodeType, maxYield, durabilityDamage, biomes*² }

ResourceNodesGenerateItems = { resourceNodes*⁹, item*⁷ }

Monuments = { name*¹º, monumentSize, lootGrade, enemyGrade }

regionsMonuments = { name*¹º, regionMonument }

Structures = { structureID*[11] , monument*[10] }

LootCrates = { lootCratesID*[12], grade }

StructuresContainsLootCrates = { structure*[11], lootCrates*[12] }

Party = { partyID, character*[4], capacity }
partyID: código do grupo de jogadores que estão jogando junto.

Backpack = { ownerID*[13], slot(totalSlots - (totalSlots -1)) … slot(totalSlots-1), slot(totalSlots), totalSlots }
ownerID: código do dono da 'backpack'.
slots: A mochila terá uma quantidade de slots determinada pelo valor de totalSlots, cada slot será um atributo diferente, possuindo um valor específico para o ID item que o ocupa ou 0 caso não haja item.