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: The percentage that the determined item reduces in incoming ***cold*** *exposure.*
radResistance: The percentage that the determined item reduces in incoming ***radiation*** *exposure.*
explosionResistance: The percentage that the determined item reduces in incoming ***explosion*** *damage.*
meleeResistance: The percentage that the determined item reduces in incoming ***melee*** *damage.*
rangedResistance: The percentage that the determined item reduces in incoming ***ranged*** *damage.*
biteResistance: The percentage that the determined item reduces in incoming ***bite*** *damage.*
wetResistance: Does the item provide immunity to the wet modifier (ex: wetsuit)
equipmentSlot: Which equipment slot does the item occupy, IE: cabeça, perna, peito, pés, peito, sobre-peito, sobre-perna.

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*1o }

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.