Universidad ORT Uruguay

Facultad de Ingeniería

Tecnologías Blockchain para contratos inteligentes Obligatorio

Santiago Aguirre 189217
Damián Mazas 153806

Entregado como requisito de la materia Tecnologías Blockchain para contratos inteligentes

8 de julio de 2021

Índice general

1. Trabajo realizado

1.1. Diagramas

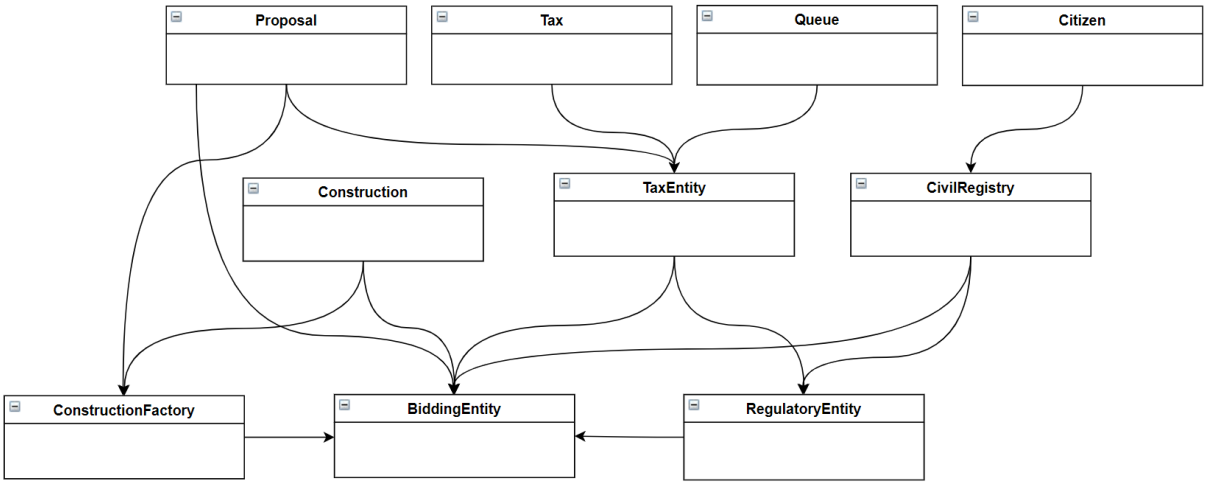

Figura 1.1: Diagrama de dependencias entre contratos.

1.1.1. Contratos

RegulatoryEntity

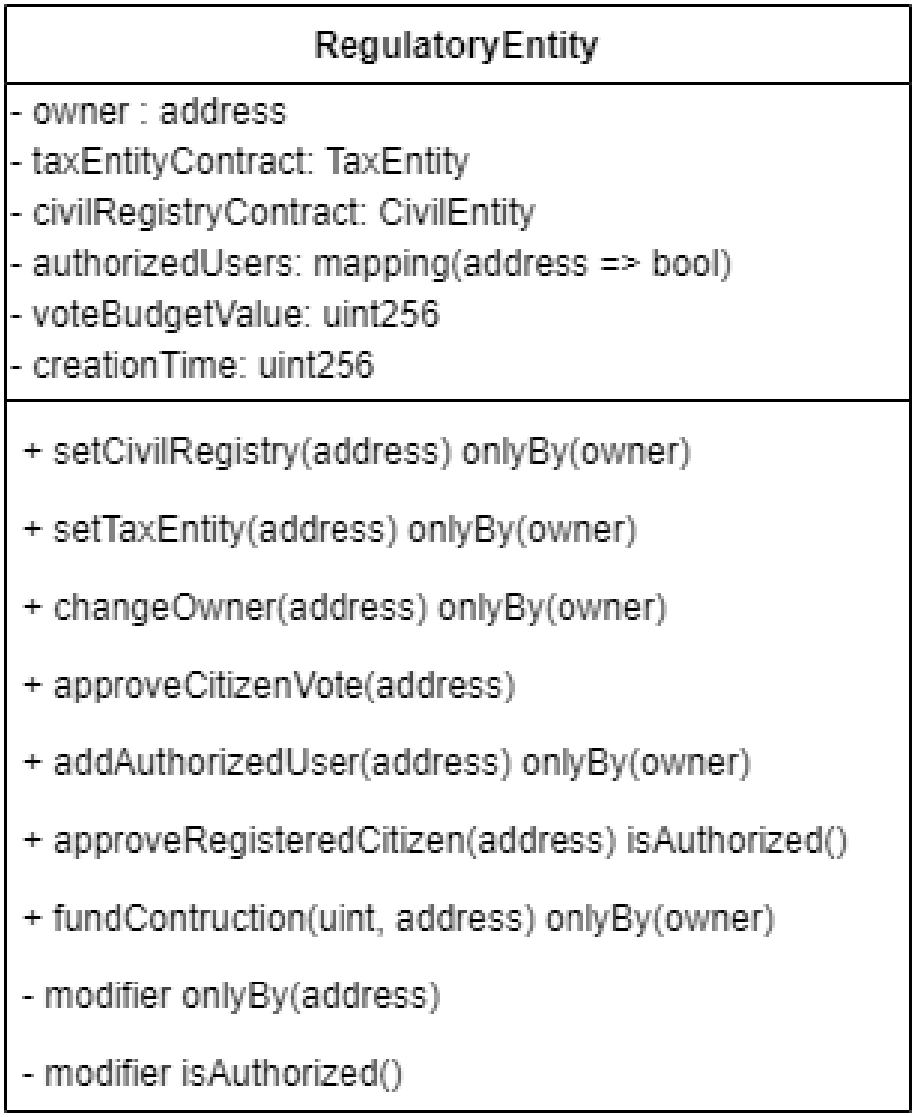

Figura 1.2: Entidad reguladora.

CivilRegistry

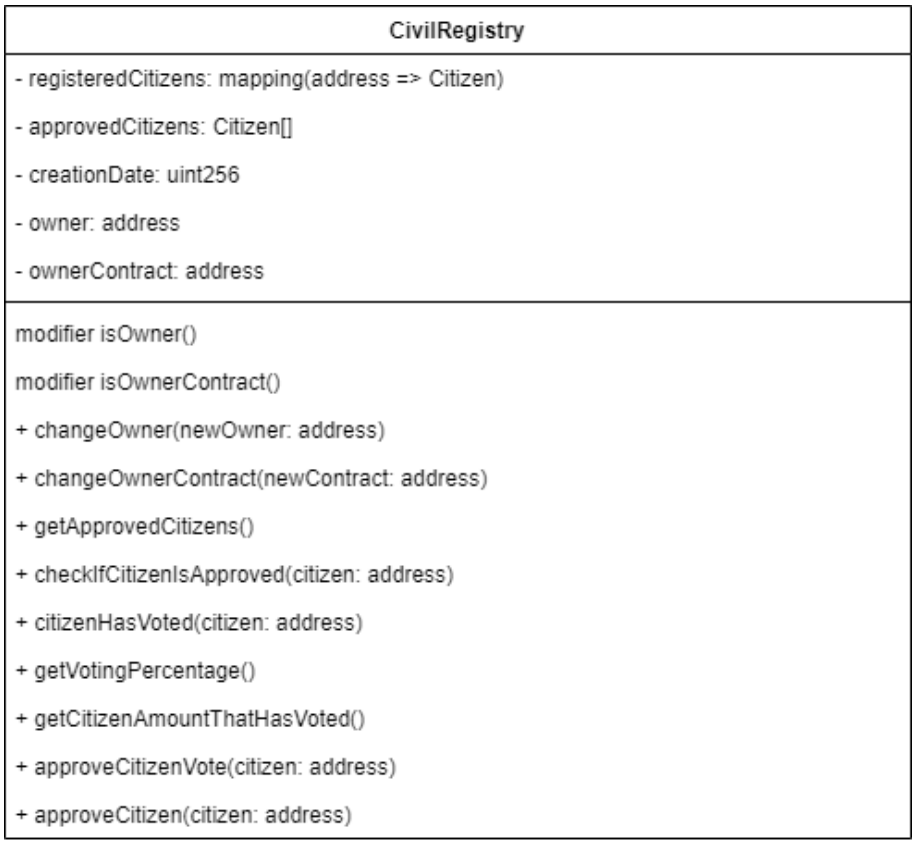

Figura 1.3: Entidad Civil.

Citizen

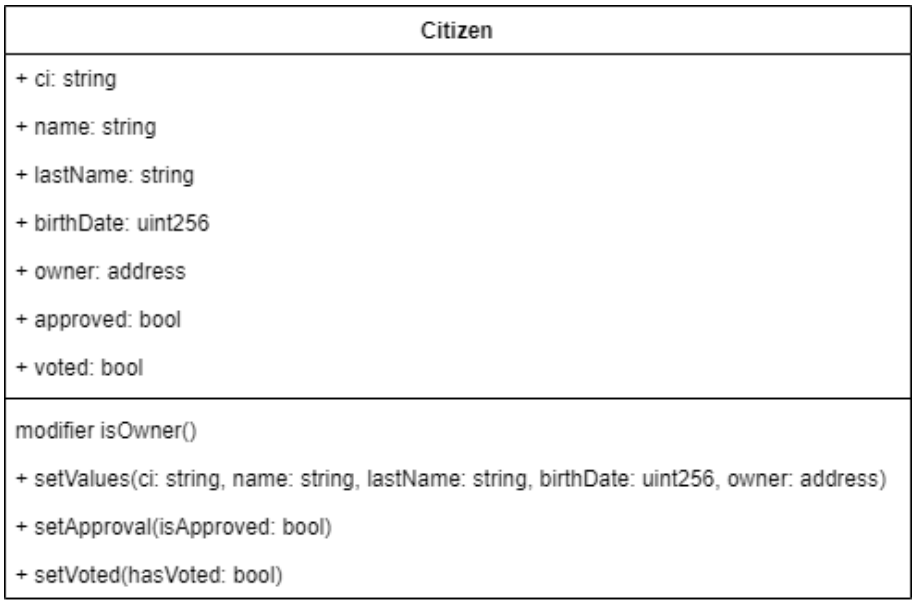

Figura 1.4: Cuidadanos.

TaxEntity

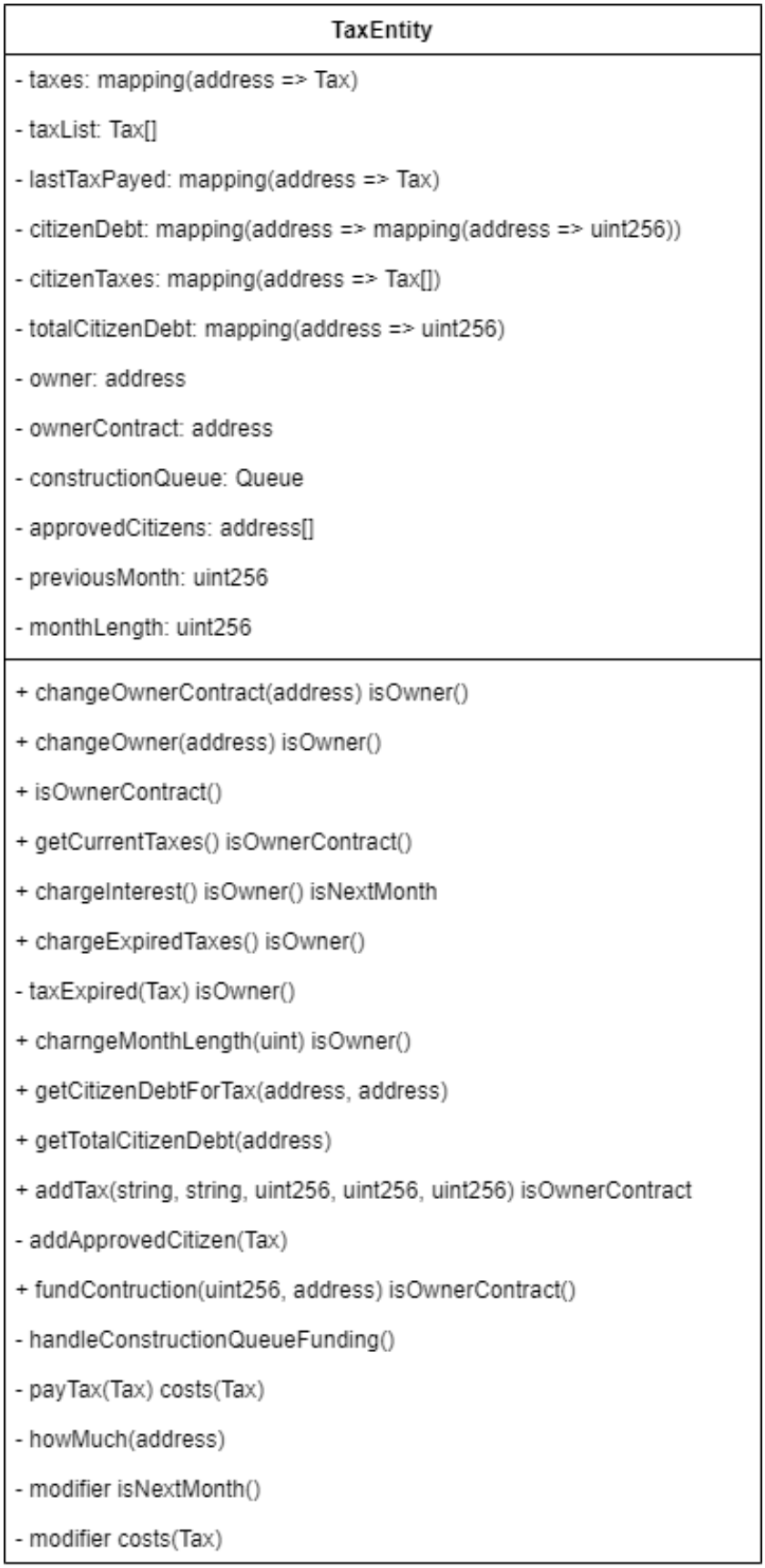

Figura 1.5: Entidad de impuestos.

Tax

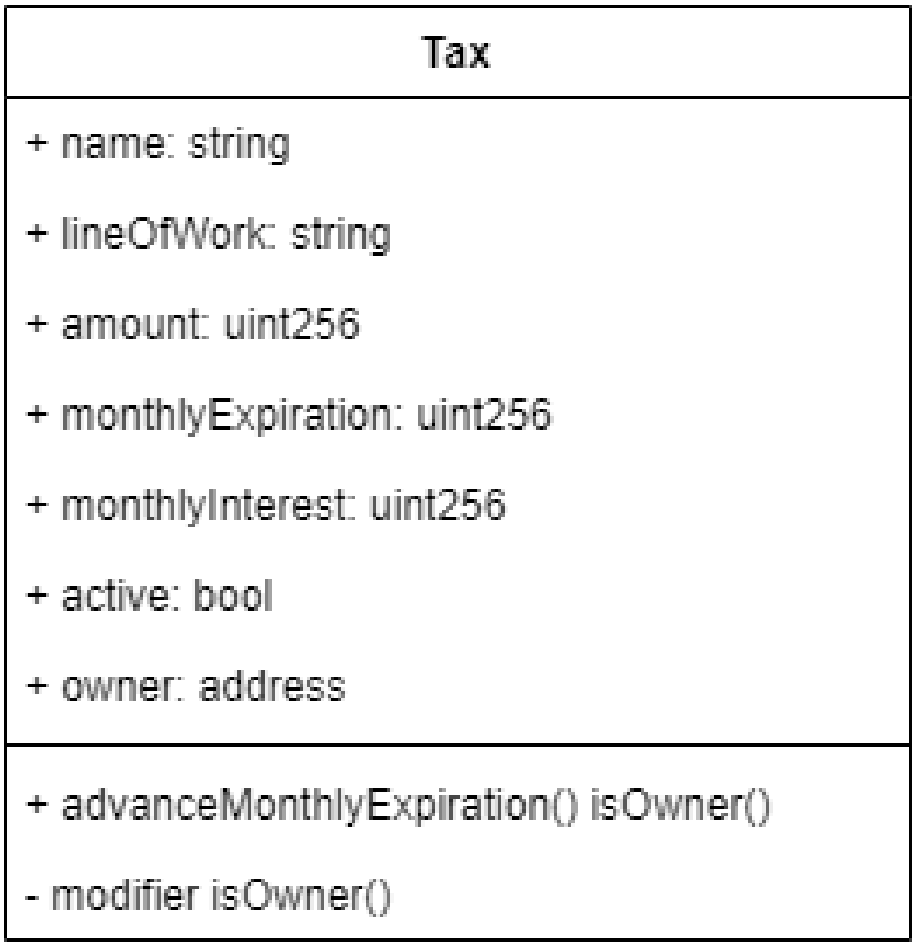

Figura 1.6: Impuestos.

BiddingEntity

BiddingEntity
+ period: Period - proposalCount: uint256 - budgetedProposalsCount: uint256 - votePercentage: uint256 + votingPeriodStart: uint256 + votingPeriodLength: uint256 - taxEntity: TaxEntity - regulatoryEntity: RegultoryEntity - civilRegistry: CivilRegistry - constructionFactory: ConstructionFactory - owner: address + voters: mapping(address => Citizen) + proposalList: Proposal[] + proposals: mapping(address => Proposal) - approvedProposals: Proposal[]
modifier onlyAt(period: Period) modifier userApproved() modifier hasNoDebt() modifer hasNotVoted() modifier isOwner() + changeVotingPeriodLength(votingPeriodLengt: uint256) + nextPeriod() + restartCycle() - closeOutVotingPeriod() - assignBudgetToProposal(proposalAddress: address, budget: uint256) + approveProviderPayment(constructionAddress: address, provider: address) + getConstructionProviders(constructionAddress: address) + receiveProposal(name: string, lineOfWork: string, description) + vote(proposal: address)

Figura 1.7: Entidad de votación.

ConstructionFactory

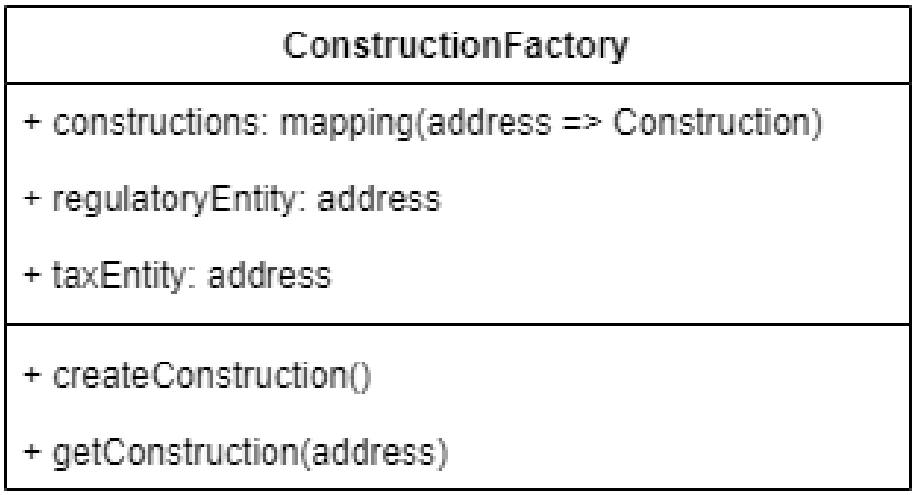

Figura 1.8: Construcción de obra.

Construction

Construction
+ name: string + lineOfWork: string + description: string + initialBudget: uint256 + owner: address + factoryAddress: address + started: bool + designatedProviders: address[] + regulatoryEntity: RegulatoryEntity + taxEntity: TaxEntity + ownerApprovedClose: bool + regulatoryEntityApprovedClose: bool + paymentToProviderMade: bool
+ close() + fundConstruction() + selectProviders(address, address, address) isOwner() hasStarted() + getDesignatedProviders() hasStarted() + approveProviderPayment(address) isRegulatoryEntity() - providerExists(address) + approveCloseOwner() isOwner() + approveCloseRegulatoryEntity() isRegulatoryEntitity() # selfdestructConstruction() - modifier costs() - modifier onlyIfNotFounded() - modifier isOwner() - modifier canCloseConstruction() - modifier hasStarted() - modifier isRegulatoryEntity()

Figura 1.9: Obra.

Proposal

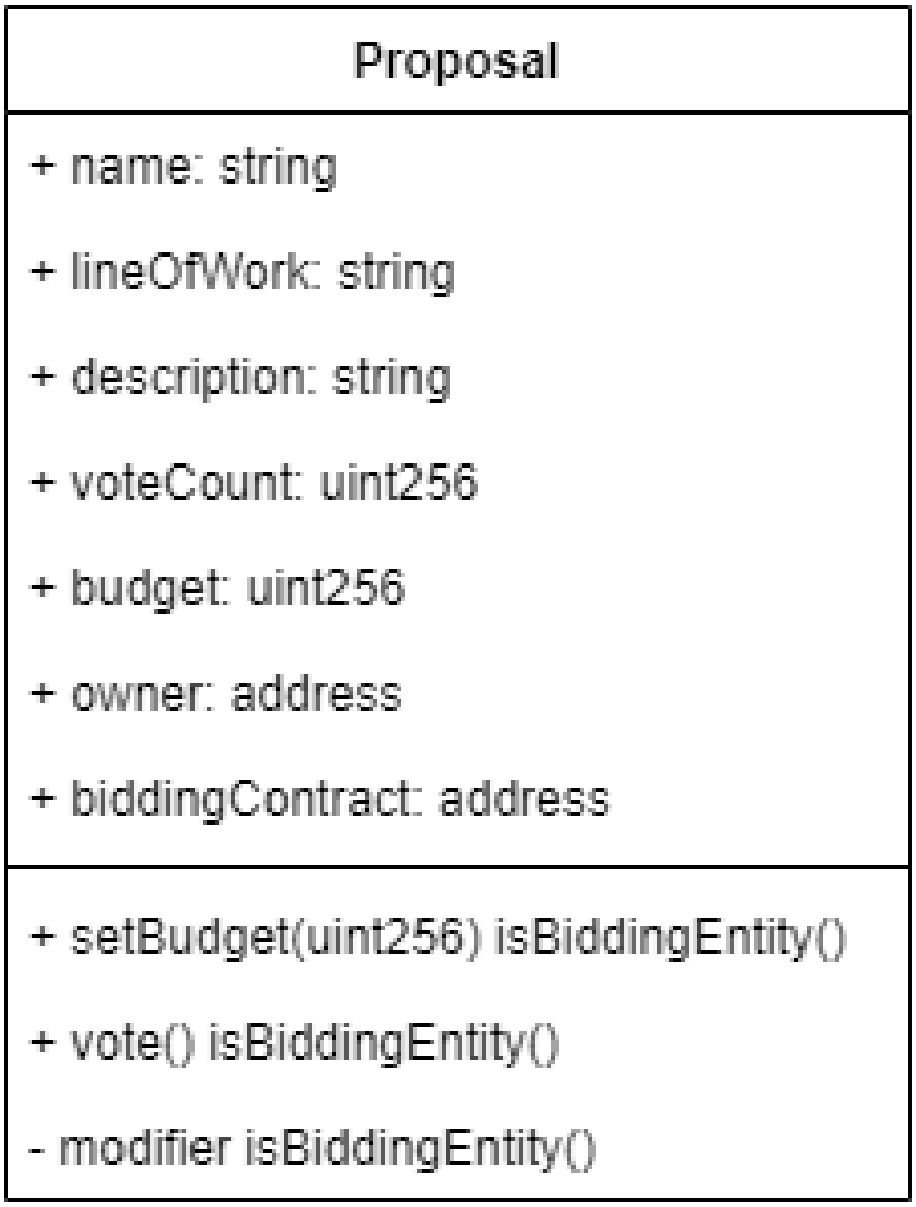

Figura 1.10: Propuestas.

Queue

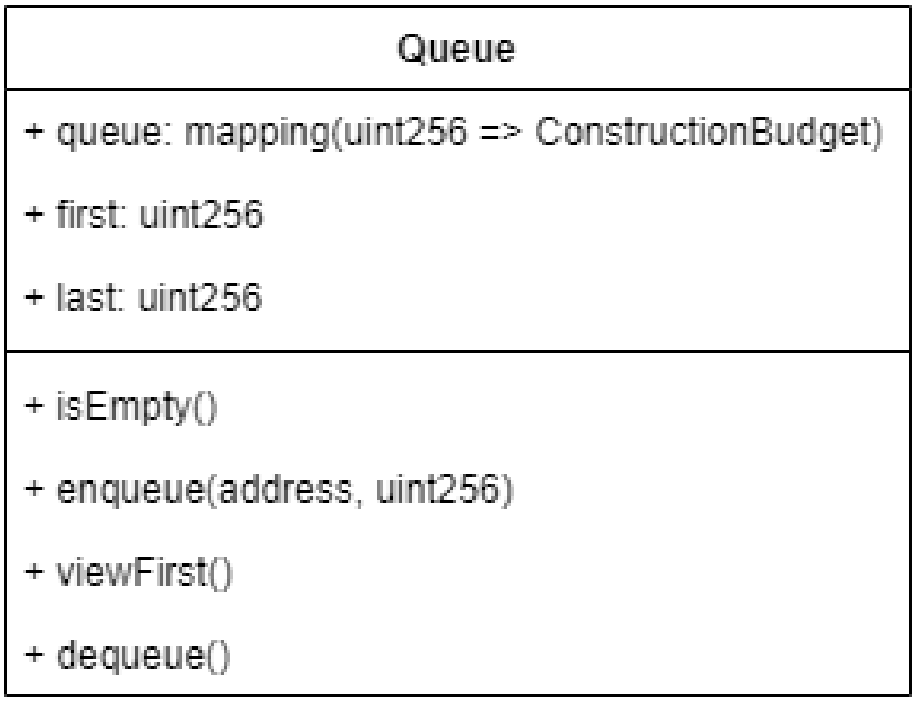

Figura 1.11: Cola de dinero.

1.2. Justificaciones de diseño

Las responsabilidades de cada contrato se dividieron de la siguiente manera:

RegulatoryEntity

El contrato RegulatoryEntity hace las veces de gobierno. Como indica su nombre, es la entidad reguladora encargada de controlar las acciones gubernamentales. La misma trabaja en conjunto con el contrato CivilRegistry y el contrato TaxEntity.

CivilRegistry

Se encarga de registrar los antedichos ciudadanos. A su vez hace las veces de registro civil, permitiendo el registro de nuevos ciudadanos, la aprobación de los mismos para votar y proveer a otros contratos de información pertinente a los ciudadanos.

Citizen

Este contrato contiene la cédula del ciudadano, el nombre, apellido y la fecha de nacimiento. A su vez tiene booleanos para saber si fue aprobado por el registro y si ya votó durante este período.

TaxEntity

Se encarga de permitir la creación de nuevos impuestos por parte del Estado y de la gestión de los pagos de los mismos por parte de los ciudadanos.

Tax

Es el encargado de registrar la información de un impuesto, teniendo como atributos el nombre, el presupuesto, su fecha de vencimiento y el interés a ser cobrado en caso que un ciudadano se atrase con su pago.

BiddingEntity

El contrato BiddingEntity es el encargado de gestionar las propuestas de obras y sus períodos de votación. El contrato contiene todas las propuestas realizadas por los ciudadanos y almacena en una estructura distinta las propuestas aprobadas a ser votadas posteriormente. Verifica que se encuentre el contrato en el estado correcto antes de realizar cualquier acción, emulando una máquina de estados. Además, es el encargado de indicarle al contrato ConstructionFactory cuándo debe crear un contrato Obra (llamado Construction).

ConstructionFactory

ConstructionFactory se encarga de construir una obra en base a una propuesta. Su responsabilidad es la creación y almacenamiento de los contratos Construction.

Construction

El contrato Construction tiene un ciclo de vida desde que se aprueba una propuesta hasta que se finaliza o se cancela la obra, aunque no comienza su ejecución hasta recibir los fondos establecidos en su creación. El mismo es el encargado de gestionar una obra aprobada y su presupuesto.

Proposal

Al ser instanciado, almacena la información de una propuesta hecha por un miembro de la comunidad, es decir, un ciudadano aprobado y sin deudas.

Queue

Este contrato es el encargado de representar una cola de propuestas u obras que aún no han sido finaciadas en el TaxEntity. En cada pago de impuestos, se checkea si el TaxEntity posee suficiente dinero como para pagar el costo de la obra en el primer lugar de la cola. Si se puede pagar, entonces se elimina de la cola y se transfieren los fondos a la obra, la cual comienza su ejecución.

1.2.1. Modificadores

Se utilizaron modificadores constantemente en el desarrollo de la solución, de los cuales resaltamos los siguientes:

- costs: El modificador costs se encarga de verificar que se esté pagando al menos el total del impuesto que se desea pagar, pero si se está pagando más de lo necesario, se reembolsa al usuario que realizó el pago la cantidad sobrante.
- hasNoDebt: El modificador hasNoDebt se encarga de controlar frente a la TaxEntity si es que un ciudadano en particular tiene o no deuda actualmente.
- isOwner: Se utilizó en casi todos los contratos con el fin de controlar que ciertas operaciones solo pudieran ser utilizadas por el dueño del contrato.
- isOwnerContract: A su vez, se utilizó el modificador isOwnerContract para indicar que algunas operaciones deberían ser poder ser utilizadas únicamente por un contrato designado como el contrato dueño. Este modificador fue diseñado para poder garantizar que las funciones del gobierno que dependían de otros contratos solo pudieran ser llamadas por el contrato gobierno o RegulatoryEntity.
- onlyAt: Se utilizó para verificar si una acción puede ser realizada en el contrato BiddingEntity, dependiendo del período en el que se encuentra el contrato (Aceptando propuestas, permitiendo votar o cerrado).

Bibliografía

[1] Universidad ORT Uruguay. (2013) Documento 302 - Facultad de Ingeniería. [Online]. Available: http://www.ort.edu.uy/fi/pdf/documento302facultaddeingenieria.pdf