UNIVERSIDADE TECNOLÓGICA FEDERAL DO PARANÁ

ALESSANDRO BUENO DE SOUZA

ACQUA - Architecture for Contractual Quality and Unified Auditing

CURITIBA

2025

ACQUA - Architecture for Contractual Quality and Unified Auditing

Requisito técnico apresentado como critério parcial para obtenção do título de Mestre em Computação Aplicada do Programa de Pós-Graduação em Computação Aplicada da Universidade Tecnológica Federal do Paraná (UTFPR).
Orientadora: Prof. Laudelino Bastos

CURITIBA
2025

Introdução

O sistema Acqua foi projetado para disponibilizar contratos inteligentes na rede Blockchain, destinados a contratação e aquisição de serviços e produtos de revitalização da água. Para proporcionar mais transparencia, segurança e integridade aos participantes do processo. Permitindo que os contratos sejam criados, armazenados e geridos diretamente em uma rede blockchain.

A documentação descreve elementos funcionais do sistema, abrangendo:

- Modelo conceitual
- Regras funcionais
- Diagrams UML
- Dicionário de Dados
- Casos de teste

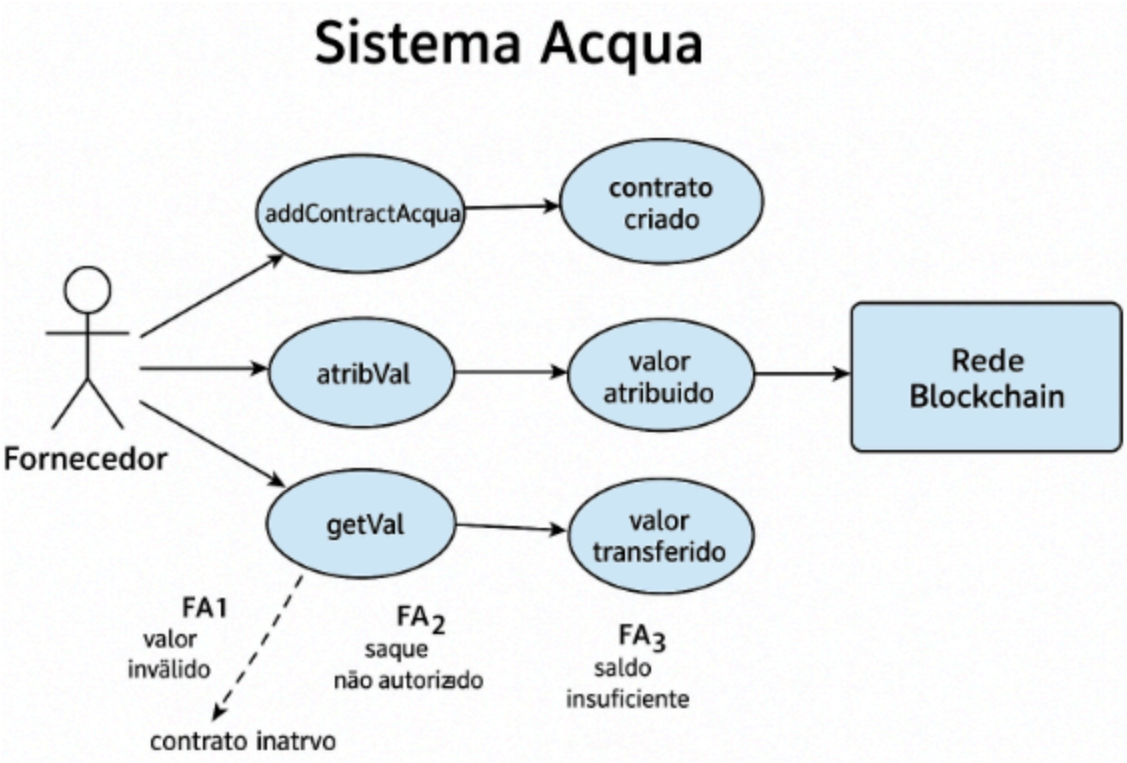
B1. Objetivo Geral

O sistema Acqua gerencia o backend de contratos inteligentes (smart contracts) na blockchain Ethereum, permitindo criar contratos, atribuir valores e efetuar saques, respeitando regras de segurança e controle de acesso. O principal objeto é o ContratoAcqua, representado como uma estrutura (struct)

B1.1 Caso de Uso: Gerenciar Contratos Acqua

Abaixo temos a figura X Use Case, que demonstra a interação entre o ator e as funcionalidades do sistema, antes das informações chegarem a rede blockchain.

Figura x Use Case



Fonte: Autoria própria(2025)

Atores

Abaixo temos a descrição dos atores que atuam no sistema com suas responsabilidades.

- **Fornecedor** – Cria contratos, atribui valores, saca valores.
- **Sistema Blockchain** – Valida, registra e executa as transações.

B1.2 Fluxo Principal

1. **Fornecedor** solicita **criação de contrato** (*addContractAcqua*).
2. **Sistema Blockchain** valida os campos obrigatórios e cria contrato com status *ativo*.
3. **Fornecedor** atribui valor (*atribVal*), enviando Ether.

4. **Sistema Blockchain** valida se contrato está ativo e adiciona valor ao saldo.
5. **Fornecedor** solicita saque (*getVal*).
6. **Sistema Blockchain** valida:
    - Fornecedor é o criador.
    - Contrato está ativo.
    - Saldo > taxa.
7. **Sistema Blockchain** transfere o saldo líquido ao fornecedor e desativa o contrato.

B1.3 Fluxos Alternativos

- **FA1 – Falha na criação do contrato**
    - Ocorre se algum campo obrigatório estiver vazio ou inválido.
    - O sistema rejeita a criação.
- **FA2 – Atribuição de valor inválida**
    - Ocorre se msg.value <= 0 ou contrato estiver inativo.
    - O sistema rejeita a transação.
- **FA3 – Saque não autorizado**
    - Ocorre se o solicitante não for o fornecedor original.
    - O sistema rejeita a operação.
- **FA4 – Saldo insuficiente para saque**
    - Ocorre se saldo ≤ taxa.

- O sistema rejeita a operação.

- **FA5 – Saque de contrato inativo**
  - Ocorre se status == false.
  - O sistema rejeita a operação.

B2. Estrutura do Objeto

A seguir temos a descrição e atributos do objeto ContratoAcqua

Nome do Objeto: ContratoAcqua

Nome	Tipo	Obrigatório	Descrição
fornecedor	address	Sim	Endereço do fornecedor do contrato
titulo	string	Não	Nome ou descrição do contrato
tipo	string	Não	Tipo do contrato
localidade	string	Sim	Local onde o contrato será executado
processo	string	Sim	Identificação do processo licitatório ou administrativo
dataAssinatura	uint256	Sim	Data da assinatura do contrato (timestamp Unix)
valorContrato	uint256	Sim	Valor total do contrato em wei

descricao	string	Sim	Descrição detalhada do contrato
item	string	Sim	Item contratado
qtde	uint256	Sim	Quantidade do item contratado
valorItem	uint256	Sim	Valor unitário do item contratado em wei
status	bool	Não	Estado do contrato (true = ativo, false = inativo)

B3. Métodos

3.1 addContractAcqua
Objetivo: Criar um contrato com status ativo na blockchain.

Pré requisito: Ator deve possuir saldo na carteira digital

Assinatura: function addContractAcqua(string memory _titulo, string memory _tipo, string memory _localidade, string memory _processo, uint256 _dataAssinatura, uint256 _valorContrato, string memory _descricao, string memory _item, uint256 _qtde, uint256 _valorItem) public returns (uint256)

Descrição: Cria um novo contrato com o endereço do remetente (msg.sender) como fornecedor, os dados obrigatórios e opcionais informados, e status inicial true.

Retorno: Número identificador do contrato (ID).

```
// --- Criar contrato ---
function addContractAcqua(    undefined gas
    DadosBasicos calldata basicos,
    DadosFinanceiros calldata financeiros
) public {
    nextId++;
    ContratoAcqua storage novo = contratos[nextId];
    novo.fornecedor = msg.sender;
    novo.basicos = basicos;
    novo.financeiros = financeiros;
    novo.saldo = 0;
    novo.ativo = true;
}
```

3.2 getVal
Objetivo: Obter todos os valores atribuídos a um contrato ativo e desativá-lo.

Pré requisito: 1) Ator deve possuir saldo na carteira digital para taxa da rede blockchain. 2) contrato deve possuir valor. 3) contrato deve estar ativo

Assinatura: function getVal(uint256 _numeroContrato) public returns (bool)

Descrição: Recebe o número do contrato, verifica se está ativo e se o solicitante é o fornecedor, transfere os valores e desativa o contrato.

Retorno: true para sucesso ou false para falha.

```
// --- Resgatar valor (similar ao withdraw) ---
function getVal(uint256 id) public {    infinite gas
    ContratoAcqua storage contrato = contratos[id];
    require(contrato.fornecedor == msg.sender, "Sem permissao");
    require(contrato.ativo == true, "Contrato encerrado");
    require(contrato.saldo > taxa, "Saldo insuficiente");

    address payable destinatario = payable(contrato.fornecedor);
    destinatario.transfer(contrato.saldo - taxa);

    contrato.ativo = false;
}
```

3.3 atribVal
Objetivo: Atribuir um valor a um contrato ativo.

Pré requisito: 1) Ator deve possuir saldo na carteira digital. 2) contrato deve estar ativo.

Assinatura: function atribVal(uint256 _numeroContrato) public payable returns (bool)

Descrição: Recebe o número do contrato e o valor enviado na transação (msg.value). Apenas contratos ativos podem receber valores.

Retorno: true para sucesso ou false para falha.

```
// --- Atribuir valor (similar ao donate) ---
function atribVal(uint256 id) public payable {    infinite gas
    require(msg.value > 0, "Valor deve ser > 0");
    require(contratos[id].ativo == true, "Contrato inativo");

    contratos[id].saldo += msg.value;
}
```

B4. Requisitos Funcionais

- RF01 - Sistema deve ser capaz de criar contrato na rede blockchain
- RF02 - Somente o fornecedor criador do contrato pode sacar valores.
- RF03 - Apenas contratos ativos podem receber valores.
- RF04 - Somente contratos ativos permitem saque de valores.
- RF05 - Após o saque de valores, o contrato deve ser automaticamente desativado.

B4.1 Requisitos de Negócio

- **RN01 – Valor do contrato obrigatório**
  O contrato deve possuir um valor total maior que zero, definido no momento da criação.
- **RN02 – Data de assinatura válida**
  A data de assinatura do contrato não pode ser futura em relação ao momento de criação na rede Blockchain.
- **RN03 – Localidade obrigatória**
  Todo contrato deve conter uma localidade definida, vinculando o contrato ao contexto geográfico.
- **RN04 – Processo obrigatório**
  Cada contrato deve estar obrigatoriamente associado a um processo, garantindo rastreabilidade.
- **RN05 – Pelo menos um item obrigatório**
  O contrato deve possuir no mínimo **um item cadastrado** com quantidade maior que zero.
- **RN06 – Quantidade dos itens**
  A quantidade de cada item deve ser um número positivo.
- **RN07 – Valor unitário dos itens**
  O valor unitário de cada item deve ser maior que zero.
- **RN08 – Cálculo automático do valor total**
  O valor total do contrato deve ser consistente com a soma dos valores de todos os itens cadastrados.
- **RN09 – Apenas o fornecedor criador pode resgatar valores**
  Somente o fornecedor que criou o contrato tem permissão para executar o saque (`getVal`).
- **RN10 – Contratos inativos não aceitam atribuições de valores**
  Se o contrato for finalizado ou desativado, não poderá mais receber atribuições (`atribVal`).
- **RN11 – Taxa de operação**
  Toda transferência de valor para resgate deve considerar a taxa de operação definida no contrato.

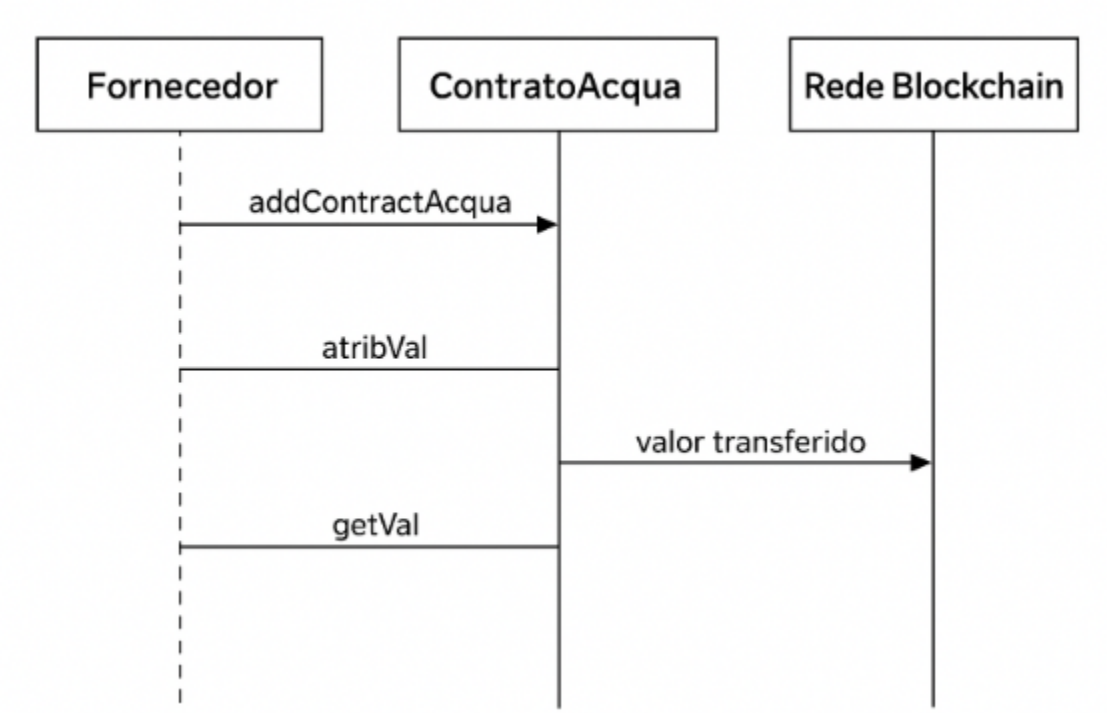
B5. Fluxo Básico de Operações

1. Criação do contrato: chamar addContractAcqua(...) com todos os campos obrigatórios e receber o ID.
2. Atribuição de valores: chamar atribVal(_numeroContrato) enviando Ether (msg.value).
3. Saque de valores: chamar getVal(_numeroContrato), transferir fundos e desativar.

B5.1 Diagrama de sequência.

A seguir na Figura X Sequência, temos os eventos que ocorrem durante a execução do sistema.

Figura X Sequência



Fonte autoria própria 2025.

B6. Código Fonte

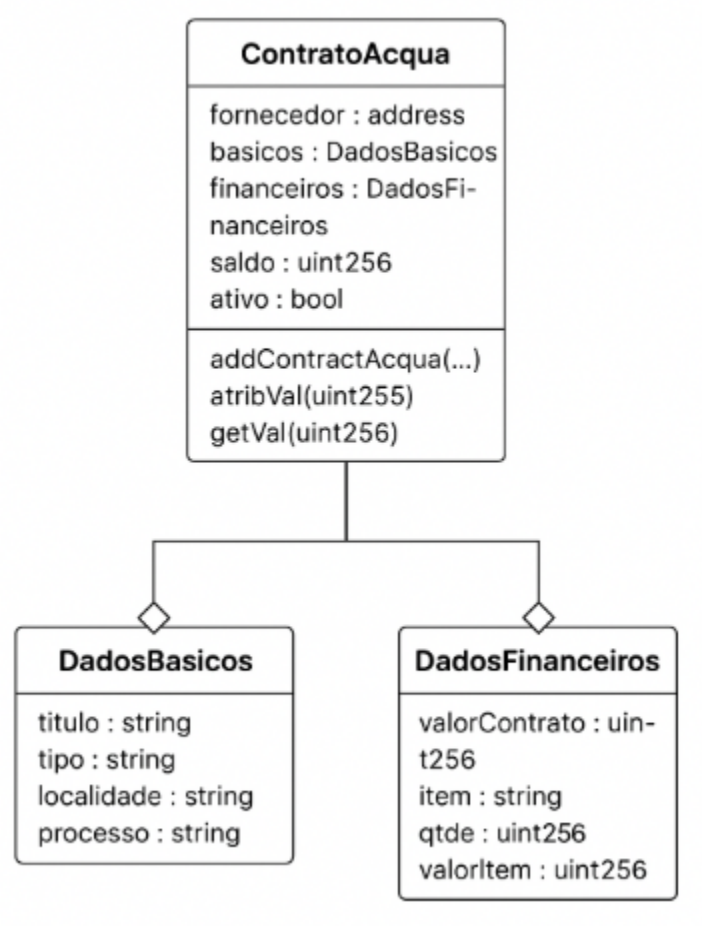
Abaixo é apresentado na figura 01 - Tela remix, o código backend, da aplicação desenvolvida na plataforma remix da ethereum.

Disponível non github em: https://github.com/alebueno/acqua

B7. Classe ContratoAcqua

A seguir temos a descrição da classe ContratoAcqua, com seus atributos e metodos conforme figura x Class ContratoAcqua, conforme abaixo:

Figura x Class ContratoAcqua



Fonte Própria (2025)

B7.1.Dicionário de Dados

Quadro X - Didionário de Dados

Campo	Tipo de Dado	Tamanho	Obrigatório	Descrição
idContrato	uint256	32 bits	Sim	Identificador único do contrato na blockchain (chave primária).
fornecedor	address	42	Sim	Endereço blockchain do fornecedor criador do contrato.
tipo	string	255	Não	Classificação ou categoria do contrato.
localidade	string	255	Sim	Local de execução do contrato.
processo	string	255	Sim	Número ou código do processo associado.
dataAssinatura	string	20	Sim	Data em que o contrato foi assinado (formato ISO: YYYY-MM-DD).

valorContrato	**uint256**	**32 bits**	**Sim**	**Valor total previsto no contrato.**
descricao	**string**	**500**	**Sim**	**Descrição textual do objeto contratual.**
titulo	**string**	**255**	**Não**	**Nome curto ou título de identificação.**
itens[]	**Array**	**-**	**Sim**	**Lista de itens vinculados ao contrato (ItemContrato).**
saldo	**uint256**	**32 bits**	**Sim**	**Valor acumulado de atribuições (atribVal).**
ativo	**bool**	**1 bit**	**Sim**	**Indica se o contrato está ativo (true) ou encerrado (false).**

Fonte Própria (2025)c

B8. Casos de Teste – Sistema Acqua

ID CT	Descrição do Caso de Teste	RF Atendido	RN	Entrada / Ação	Resultado Esperado
CT01	Criar contrato com valor válido	RF01 – Criar Contrato	RN01	Valor total > 0	Contrato é registrado com sucesso na rede Blockchain
CT02	Criar contrato com valor = 0	RF01	RN01	Valor total = 0	Rejeição da criação, erro exibido
CT03	Criar contrato com data futura	RF01	RN02	Data assinatura = data futura	Rejeição da criação, erro exibido
CT04	Criar contrato sem localidade	RF01	RN03	Localidade = vazio	Rejeição da criação

CT05	Criar contrato sem processo associado	RF01	RN04	Processo = vazio	Rejeição da criação
CT06	Criar contrato sem itens	RF01	RN05	Lista de itens = vazia	Rejeição da criação
CT07	Criar contrato com item de quantidade negativa	RF01	RN06	Item quantidade = -5	Rejeição da criação
CT08	Criar contrato com item de valor unitário zero	RF01	RN07	Item valor = 0	Rejeição da criação
CT09	Validar soma do valor do contrato	RF01	RN08	Itens com valores inconsistentes	Sistema corrige ou rejeita inconsistência
CT10	Atribuir valor a contrato ativo	RF02 – Atribuir valor	RN10	Contrato ativo, valor > 0	Valor atribuído corretamente
CT11	Atribuir valor a contrato inativo	RF02	RN10	Contrato desativado	Rejeição da atribuição
CT12	Resgatar valor como autor	RF03 – Resgatar valor	RN09, RN11	Autor solicita resgate	Valor transferido – taxa (fee), contrato desativado
CT13	Resgatar valor como não autor	RF03	RN09	Outro usuário solicita resgate	Rejeição da operação
CT14	Resgatar contrato sem saldo	RF03	RN11	Contrato saldo < fee	Rejeição do saque
CT15	Resgatar contrato com saldo válido	RF03	RN11	Contrato saldo > fee	Transferência realizada, contrato fechado

B8.1 CT01 – Criar Contrato na Blockchain

- **Objetivo:** Validar a criação de um contrato na rede blockchain.
- **Pré-condição:** Fornecedor autenticado.
- **Entrada:**
    1. Fornecedor (obrigatório)
    2. Tipo
    3. Localidade (obrigatório)
    4. Processo (obrigatório)
    5. Data Assinatura (obrigatório)
    6. Valor Contrato (obrigatório)
    7. Descrição (obrigatório)
    8. Item (obrigatório)
    9. Qtde (obrigatório)
    10. Valor do Item (obrigatório)
    11. Título
- **Passos:**
    1. Chamar addContractAcqua com os parâmetros obrigatórios.
    2. Confirmar inclusão na blockchain.
- **Resultado Esperado:** Contrato ativo com número único gerado.
- **Requisitos Cobertos:** RF01, RF02.

B8.2 CT02 – Atribuir Valor a Contrato Ativo

- **Objetivo:** Validar atribuição de valor em um contrato ativo.
- **Pré-condição:** Contrato existente e ativo.
- **Entrada:** Número do contrato, valor.
- **Passos:**
    1. Chamar atribVal com ID do contrato e valor.
    2. Validar se o contrato está ativo.
- **Resultado Esperado:** Valor adicionado ao saldo do contrato.
- **Requisitos Cobertos:** RF02.

B8.3 CT03 – Impedir Atribuição de Valor a Contrato Inativo

- **Objetivo:** Garantir que contratos inativos não recebam valor.
- **Pré-condição:** Contrato inativo.
- **Entrada:** Número do contrato, valor.
- **Passos:**
    1. Chamar atribVal com ID do contrato inativo.
- **Resultado Esperado:** Mensagem de erro “Contrato inativo”.
- **Requisitos Cobertos:** RF02.

B8.4 CT04 – Resgatar Valor de Contrato Ativo (Fornecedor Criador)

- **Objetivo:** Validar saque do valor do contrato pelo fornecedor criador.
- **Pré-condição:** Contrato ativo com saldo > taxa.
- **Entrada:** Número do contrato.
- **Passos:**
    1. Chamar getVal com ID do contrato.
    2. Validar que msg.sender é o fornecedor criador.
    3. Transferir saldo menos taxa.
    4. Alterar status para inativo.
- **Resultado Esperado:** Valor transferido ao fornecedor, contrato desativado.
- **Requisitos Cobertos:** RF01, RF03, RF04.

B8.5 CT05 – Impedir Resgate por Usuário Não Autorizado

- **Objetivo:** Garantir que somente o fornecedor criador possa sacar.
- **Pré-condição:** Contrato ativo com saldo > taxa.
- **Entrada:** Número do contrato.
- **Passos:**
    1. Chamar getVal com usuário diferente do criador.
- **Resultado Esperado:** Mensagem de erro “Você não tem permissão”.
- **Requisitos Cobertos:** RF01.

B8.6 CT06 – Impedir Resgate de Contrato Inativo

- **Objetivo:** Garantir que não seja possível sacar de contratos inativos.
- **Pré-condição:** Contrato inativo.
- **Entrada:** Número do contrato.
- **Passos:**
    1. Chamar getVal.
- **Resultado Esperado:** Mensagem de erro “Contrato inativo”.
- **Requisitos Cobertos:** RF03.

B9. Conclusão

O sistema Acqua busca alinhar a tecnologia Blockchain com a gestão de contratos para garantir a transparência, segurança, integridade e rastreabilidade das informações de forma independente e automatica.