



# Kanastra Documentation

This documentation exists to explain a integration between Rcell (Sankhya ERP) and Kanastra (infrastructure for private credit funds and securitization).

Here will have 4 important topics:


1. Kanastra API/Webhooks
2. Process
3. Sankhya ERP
4. Queue

- [1. Kanastra API /Official Doc](#)
- [2. Queues](#)
- [3. Views](#)
- [4. Process](#)
- [5. Webhooks](#)
- [Especificações](#)
- [Routes](#)

## 1. Kanastra API /Official Doc

Build, Collaborate & Integrate APIs | SwaggerHub

Join thousands of developers who use SwaggerHub to build and design great APIs. Signup or login today.

 [https://app.swaggerhub.com/apis-docs/Kanastra/tech-hub/1.1.0#/Renegotiations/post\\_api\\_credit\\_originators\\_\\_creditOriginatorId\\_\\_renegotiations](https://app.swaggerhub.com/apis-docs/Kanastra/tech-hub/1.1.0#/Renegotiations/post_api_credit_originators__creditOriginatorId__renegotiations)

## 2. Queues

Todas as operações a seguir, tem como primeiro passo, buscar no Sankhya, visto que ele é totalmente passivo.

▼ **Offers** *(Cria uma nova oferta, no qual é o documento que inicia o processo de aquisição do título. Isso aciona as validações da oferta para verificar a elegibilidade da oferta.)*

### ▼ 1. Criação da offer no database e do evento (Publisher)

Após buscar offers na view AD\_VWOFFERSKANASTRA, é necessário parsear o array de "items", pois todas os dados do Sankhya vem como string, então não é possível passa-lo diretamente para a validação do zod que espera receber um array.

```
const { offers, errors } = await createOffersUseCase.execute({
  fields: '*',
  viewName: 'AD_VWOFFERSKANASTRA',
  transform: {
    coobrigation: (value => { return value === 'true' }),
  },
});
```

```

    items: (value => {
      const dataParsed = JSON.parse(value)
      return dataParsed
    }),
  },
})

```

Iterando os registros para verificar se o registro já existe, começando na validação de schema para garantir que não existe dados nulos.

```

z.object({
  externalId: z.string(),
  sponsorName: z.string(),
  sponsorPersonType: z.string(),
  sponsorGovernmentId: z.string(),
  sponsorExternalCode: z.string(),
  sponsorAddress: z.string(),
  sponsorAddressNumber: z.string(),
  sponsorAddressComplement: z.string(),
  sponsorNeighborhood: z.string(),
  sponsorCity: z.string(),
  sponsorState: z.string(),
  sponsorCountry: z.string(),
  sponsorZipCode: z.string(),
  sellerGovernmentId: z.string(),
  coobrigation: z.boolean(),
  customFields: z.string(),
  items: z.array(z.object({
    assetType: z.string(),
    invoiceNumber: z.string(),
    invoiceDate: z.string(),
    invoiceKey: z.string(),
    totalInstallments: z.number(),
    paymentValue: z.number(),
    paymentDate: z.string(),
    customFields: z.object({
      preCalculatedAcquisitionPrice: z.number(),
      rateType: z.string(),
    }),
  })),
  installments: z.array(
    z.object({
      externalId: z.string(),
      amount: z.number(),
      customFields: z.object({}),
      dueDate: z.coerce.string(),
    })
  )
}))
})

```

Após validação, é criado o registro no database como status "progress" e criação do evento na queue "send-offers" passando 2 parâmetros, "id" e "addToNextStepQueue" para envio. segue exemplo de output:

```
{
  "externalId": "****",
  "sponsorName": "****",
  "sponsorPersonType": "****",
  "sponsorGovernmentId": "****",
  "sponsorExternalCode": "****",
  "sponsorAddress": "****",
  "sponsorAddressNumber": "****",
  "sponsorAddressComplement": "****",
  "sponsorNeighborhood": "****",
  "sponsorCity": "****",
  "sponsorState": "****",
  "sponsorCountry": "****",
  "sponsorZipCode": "****",
  "sellerGovernmentId": "****",
  "coobrigation": false,
  "customFields": "",
  "items": [
    {
      "assetType": "****",
      "invoiceNumber": "****",
      "invoiceDate": "****",
      "invoiceKey": "****",
      "totalInstallments": "****",
      "paymentValue": "****",
      "paymentDate": "****",
      "customFields": {
        "preCalculatedAcquisitionPrice": "****",
        "rateType": "****"
      },
      "installments": [
        {
          "externalId": "****",
          "amount": "****",
          "customFields": {},
          "dueDate": "****"
        }
      ]
    }
  ]
}
```

## ▼ 2. Enviar offer (Subscriber)

Como está etapa é um listener, para cada registro adicionado, existem 2 parâmetros conforme citado na 1 etapa.

Caso registro seja encontrado no database, é necessário buscar o canhoto no Sankhya para complementar o body da primeira etapa:

```
SELECT
    ATA.CONTEUDO AS content,
    NULL nfe_xml,
    'comprovante_assinatura' AS category,
    CONCAT(CAB.NUNOTA, '.jpg') AS name
FROM
    SANKHYA.TGFCAB CAB (NOLOCK)
    LEFT JOIN TSIATA ATA ON ATA.CODATA = CAB.NUNOTA
WHERE
    ATA.CODATA = CAB.NUNOTA
    AND CAB.NUNOTA = $ { externalId }
    AND ATA.DESCRICAO LIKE '%CANHOTO%'
UNION ALL
SELECT
    NULL content,
    CAST((NFE.XMLNVLCLI) AS VARCHAR(MAX)) AS nfe_xml,
    'nfe_xml' AS category,
    CONCAT(CAB.NUNOTA, '.xml') AS name
FROM
    SANKHYA.TGFCAB CAB (NOLOCK)
    LEFT JOIN SANKHYA.TGFNFE NFE (NOLOCK) ON NFE.NUNOTA = CAB.NUNOTA
WHERE
    CAB.NUNOTA = NFE.NUNOTA
    AND CAB.NUNOTA = $ { externalId }
```

A query foi desenvolvida com union para trazer os 2 dados evitando request desnecessários, sendo assim a tag “content” sempre deve estar presente com o conteúdo, porém não é possível trazer ambos os dados no mesmo campo, pois um vem como varchar e o outro como hexadecimal.

Seguindo essa linha foi criado a logica que quando o content vier preenchido, se da a entender que é do tipo hexadecimal e os demais em uma coluna adicional, exemplo:

	content	nfe_xml	category	name
1	0xFFD8FFE000104A4649460001010100600000FFDB00...	NULL	comprovante_assinatura	2370458.jpg
2	NULL	<nfeProc xmlns="http://www.portalfiscal.inf.br/nfe..."	nfe_xml	2370458.xml

Então todos os demais campos são ocultos, os demais campos oriundos de varchar, devem conter o nome da coluna com o mesmo nome do “category”, por exemplo, na imagem acima, a categoria do anexo é um “nfe\_xml” então deve existir uma coluna com o mesmo nome e o seu valor, assim este conteúdo passara a ser o “content” e a coluna excluída, não sendo necessário mexer na logica do campo para cada anexo novo. exemplo de saída:

**Ao final o conteúdo de ambos devem ser passados para base64:**

```
[
  {
    "category": "comprovante_assinatura",
```

```

    "name": "****.jpg",
    "content": "*****" // Buffer.from(file.content, 'hex').toString('base64')
  },
  {
    "category": "nfe_xml",
    "name": "****.xml",
    "content": "*****" // Buffer.from(fileContent).toString('base64')
  }
]

```

É buscado as preferencias para executar a operação, como:

- *Refresh Token*
- *Credit Id (Query parameter)*

Caso seja enviado com sucesso, é atualizado no database para status "saving"(caso o parâmetro "addToNextStepQueue" seja true, é atribuído a fila de "save-offers"), senão, é atualizado para "error".

### ▼ 3. Salvar offer (Subscriber)

Como está etapa é um listener, para cada registro adicionado, existe 1 parâmetro conforme citado na 2 etapa.

O id recebido como parâmetro é usado para busca-lo no database, usando os campos "api\_res" para pegar a resposta de criação da offer.

exemplo do body no "api\_res":

```

{
  "id": "42e93c98-92f1-4731-a7c7-27395bd8cf39",
  "status": "saved",
  "buyer_id": "725854b3-f762-4083-801f-1a8918f264f2",
  "sponsor_id": "1feaace9-372c-4155-b8ba-c2d4be20bf9c",
  "seller_id": "7118d497-6a60-45b5-9131-8709e7ff6861",
  "created_at": "2022-10-13T17:06:52.000000Z",
  "updated_at": "2022-10-13T17:06:52.000000Z",
  "credit_originator_id": 3,
  "external_id": "12345abcdef",
  "daily_discount_rate": 0.00069,
  "daily_discount_rate_observations": "relevant information about this type of",
  "daily_discount_rate_calculator": "XPT0DiscountRateCalculator",
  "daily_discount_rate_calculated_at": "2022-10-13T17:06:52.000000Z",
  "daily_discount_rate_apply_on": "present_value",
  "payment_value": 4000,
  "installments": [
    {
      "external_id": "external_id_12345",
      "present_amount": 4501.43,
      "future_amount": 5041.6
    }
  ],
}

```

```

"validations": [
  {
    "id": 117,
    "offer_id": "1f14c9b9-5a2a-4e63-8ed1-0ce5d83b8dab",
    "validator_class": "ValidateInvoiceWithXPT0",
    "name": "Invoice validation with XPT0",
    "result": 1,
    "executed_in": 0.02,
    "error_messages": "Warning: XPT0 service was disabled.\n\n",
    "created_at": "2022-10-05T16:13:33.000000Z",
    "updated_at": "2022-10-05T16:13:33.000000Z"
  }
]
}

```

segue abaixo a query:

```

<?xml version="1.0" encoding="ISO-8859-1"?>
<serviceRequest serviceName="crud.save">
  <requestBody>
    <entity name="AD_OFFERKANASTRA">
      <campo nome="NUNOTA">${external_id}</campo>
      <campo nome="OFFERSENT">S</campo>
      <campo nome="IDOFFER">${id}</campo>
      <campo nome="STATUS">${status}</campo>
    </entity>
  </requestBody>
</serviceRequest>

```

Caso inserido com sucesso, é gravado o retorno no database e alterado o status para "done", senão, e passado o erro no campo "errors" com o status de "erro".

## ▼ Acquisitions (Envie uma remessa de aquisição para processamento)

### ▼ 1. Criação da acquisition no database (Publisher)

Após buscar offers na view AD\_VWREPURCHASESKANASTRA, é necessário parsear o array de "items", pois todas os dados do Sankhya vem como string, então não é possível passa-lo diretamente para a validação do zod que espera receber um array.

```

const { acquisition, errors } = await createAcquisitionUseCase.execute({
  fields: '*',
  viewName: 'AD_VWACQUISITIONKANASTRA',
  transform: {
    id: value => value
  }
})

```

```
}
})
```

Iterando os registros para verificar se o registro já existe, começando na validação de schema para garantir que não existe dados nulos.

```
const acquisitionSchema = z.object({
  id: z.string(),
})
```

Após validação, é criado o registro no database como status "progress" que até o momento é entendido como "pending", visto que a Kanastra só recebe títulos e acquisitions até as 16:00, então como o processo ainda não está 100%, o envio da acquisition está sendo feito manualmente.

## ▼ 2. Envio da offer (Futuro Subscriber)

Até o momento, não foi definido como será o envio da acquisition automaticamente, então está sendo enviado manualmente através de um front personalizado até que seja definido.

<input type="checkbox"/> acquisition.id ↑↓	id ↑↓	acquisition.status ↑↓	acquisition.created_at ↑↓	
<input type="checkbox"/> 30a4bea1-ea62-4385-b909-8928739b18cf	6662109040758a5f394f0b68	Success	06/06/2024 16:40:00	...
<input type="checkbox"/> 39e27d58-65ef-43f3-94ac-82e77deb1d5f	66689424bd71d6950e6bbadd	Success	11/06/2024 15:15:00	Send →
<input type="checkbox"/> 05928714-14d5-4b1b-ef01-5fe48cf9e4db	66689978bd71d6950e6bbadf	Success	11/06/2024 15:37:44	Delete 🗑
<input type="checkbox"/> b5703a37-c291-4b7c-b5c8-68e77616dbca	666c5af6c7f737941cc64015	Success	14/06/2024 12:00:06	...
<input type="checkbox"/> 4d978a12-e966-4c98-8122-f4d8e59548b7	666c60ccc7f737941cc641a3	Success	14/06/2024 12:25:00	...

É buscado as preferencias para executar a operação, como:

- *Refresh Token*
- *Scope (Permissões que são atribuídas na chamada)*
- *Credit Id (Query parameter)*

Caso inserido com sucesso, é gravado o retorno no database e alterado o status para "done", senão, e passado o erro no campo "api\_res" com o status de "erro".

## ▼ Liquidations (Cria uma liquidação individual de um ativo)

### ▼ 1. Criação da liquidation no database e do evento (Publisher)

Após buscar offers na view AD\_VWLIQUIDATIONSKANASTRA, é necessário parsear os elementos dentro da key transform, pois todas os dados do Sankhya vem como string, então não é possível passa-lo diretamente para a validação do zod.

```
const { liquidations, errors } = await createLiquidationsUseCase.execute({
  fields: '*',
  viewName: 'AD_VWLIQUIDATIONSKANASTRA',
  transform: {
    nunota: (value => { return Number(value) }),
    externalId: (value => { return value.trim() }),
    payment: (value => {
      const dataParsed = JSON.parse(value)
      return dataParsed
    }),
    payer: (value => {
      const dataParsed = JSON.parse(value)
      return dataParsed
    }),
    customFields: (() => { return {} })
  },
})
```

Iterando os registros para verificar se o registro já existe, começando na validação de schema para garantir que não existe dados nulos.

```
z.object({
  externalId: z.string(),
  nunota: z.number(),
  payment: z.object({
    amount: z.number(),
    method: z.string(),
    reference: z.string(),
    date: z.coerce.string(),
  }),
  customFields: z.object({}),
  payer: z.object({
    governmentId: z.string(),
    country: z.string(),
  }),
  settlementType: z.string(),
})
```

Após validação, é criado o registro no database como status "progress" e criação o evento na queue "send-liquidations" passando 2 parâmetros, "id" e "addToNextStepQueue" para envio. segue exemplo de output:

```
{
  "externalId": "123456789",
  "nunota": 987654321,
  "payment": {
    "amount": ****,
    "method": "TED",
```



```

    "reference": 123456,
    "date": "2024-04-30 00:00:00"
  },
  "customFields": {},
  "payer": {
    "governmentId": "123456.789012.34",
    "country": "Brazil"
  },
  "settlementType": "TOTAL"
}

```

## ▼ 2. Enviar liquidations (Subscriber)

Como está etapa é um listener, para cada registro adicionado, existem 2 parâmetros conforme citado na 1 etapa, então para cada registro.

É buscado as preferencias para executar a operação, como:

- *Refresh Token*
- *Credit Id (Query parameter)*

**Antes de realizar o envio, é necessário remover o "nunota" do json, visto que essa informação não faz parte do requestBody para a Kanastra, porém é de extrema importância para a próxima etapa.**

Caso seja enviado com sucesso, é atualizado no database para status "saving" (caso o parâmetro "addToNextStepQueue" seja true, é atribuído a fila de "save-liquidations"), senão, é atualizado para "error".

## ▼ 3. Salvar liquidation (Subscriber)

Como está etapa é um listener, para cada registro adicionado, existe 1 parâmetro conforme citado na 2 etapa, então para cada registro.

O id recebido como parâmetro é usado para busca-lo no database, usando os campos "api\_res" para pegar a resposta de criação da offer.

exemplo do body:

```

{
  "id": "7b62522d-a4b6-471e-a5d6-d67be32fb6d5",
  "status": "SAVED",
  "assets": [
    {
      "uuid": "39a407df-5f90-4039-a6d2-606c4d723326",
      "acquisition_asset": 2471879,
      "status": "SAVED",
      "amount": 71200,
      "date_of_credit": "2024-04-30"
    }
  ],
  "created_at": "2024-06-03T19:21:33.000000Z",
}

```

```
"updated_at": "2024-06-03T19:21:33.000000Z"
}
```

Query para inserção do retorno da liquidation:  
(external\_id e nunota vem do campo "data" ta etapa anterior)

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<serviceRequest serviceName="crud.save">
  <requestBody>
    <entity name="AD_LIQUIDATIONSKANASTRA">
      <campo nome="NUFIN">${external_id}</campo>
      <campo nome="NUNOTA">${nunota}</campo>
      <campo nome="IDLIQUIDATION">${id}</campo>
      <campo nome="ASSETS">${assets}</campo>
      <campo nome="STATUS">${status}</campo>
      <campo nome="CREATEDAT">${created_at}</campo>
      <campo nome="UPDATEDAT">${updated_at}</campo>
    </entity>
  </requestBody>
</serviceRequest>
```

Caso inserido com sucesso, é gravado o retorno no database e alterado o status para "done", senão, e passado o erro no campo "errors" com o status de "erro".

## Webhooks

### ▼ Repurchases (Recompra de uma oferta que ainda não foi liquidada)

#### ▼ 1. Criação da repurchase no database e do evento (Publisher)

Após buscar offers na view AD\_VWREPURCHASESKANASTRA, é necessário parsear os elementos dentro da key transform, pois todas os dados do Sankhya vem como string, então não é possível passa-lo diretamente para a validação do zod.

```
const { repurchases, errors } = await createRepurchasesUseCase.execute({
  fields: '*',
  viewName: 'AD_VWREPURCHASESKANASTRA',
  transform: {
    withReactivation: (value => {
      return value === 'true' ? true : false
    }),
    newExternalId: (value => {
      if(!value) {
        return null
      }
    })
  }
})
```

```

        return JSON.parse(value)
    }},
  },
})

```

Iterando os registros para verificar se o registro já existe, começando na validação de schema para garantir que não existe dados nulos.

```

z.object({
  externalId: z.string(),
  nunota: z.string().transform(Number),
  expectedAmount: z.string().transform(Number),
  reason: z.enum(['COMMERCIAL_CHANGES', 'DISQUALIFICATION']),
  description: z.string().optional().default('sem descricao'),
  newExternalId: z.array(z.string()).default([]),
  type: z.enum(['CASH', 'ACQUISITION']),
  withReactivation: z.boolean().optional().default(false),
  files: z.any().optional().default([]),
})

```

Após validação, é criado o registro no database como status "progress" e criação o evento na queue "send-repurchases" passando 2 parâmetros, "id" e "addToNextStepQueue" para envio. segue exemplo de output:

```

{
  "externalId": "123456789",
  "nunota": 1234567,
  "expectedAmount": 1234.0,
  "reason": "COMMERCIAL_CHANGES",
  "description": "sem descricao",
  "newExternalId": [],
  "type": "CASH",
  "withReactivation": false,
  "files": []
}

```

## ▼ 2. Enviar repurchases (Subscriber)

Como está etapa é um listener, para cada registro adicionado, existem 2 parâmetros conforme citado na 1 etapa, então para cada registro.

É buscado as preferencias para executar a operação, como:

- *Refresh Token*
- *Credit Id (Query parameter)*

**Antes de realizar o envio, é necessário remover o "nunota" do json, visto que essa informação não faz parte do requestBody para a Kanastra, porém é de extrema importância para a próxima**

### **etapa.**

Caso seja enviado com sucesso, é atualizado no database para status "saving" (caso o parâmetro "addToNextStepQueue" seja true, é atribuído a fila de "save-repurchases"), senão, é atualizado para "error".

### ▼ **3. Salvar repurchases (Subscriber)**

Como está etapa é um listener, para cada registro adicionado, existe 1 parâmetro conforme citado na 2 etapa, então para cada registro.

O id recebido como parâmetro é usado para busca-lo no database, usando os campos "api\_res" para pegar a resposta de criação da offer.

exemplo do body:

```
{
  "id": "7b62522d-a4b6-471e-a5d6-d67be32fb6d5",
  "status": "WAITING_COUNTERPART"
}
```

Query para inserção do retorno da liquidation:

(external\_id e nunota vem do campo "data" ta etapa anterior)

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<serviceRequest serviceName="crud.save">
  <requestBody>
    <entity name="AD_REPURCHASESKANASTRA">
      <campo nome="NUFIN">${externalId}</campo>
      <campo nome="NUNOTA">${nunota}</campo>
      <campo nome="IDREPURCHASE">${id}</campo>
      <campo nome="STATUS">${status}</campo>
    </entity>
  </requestBody>
</serviceRequest>
```

Caso inserido com sucesso, é gravado o retorno no database e alterado o status para "done", senão, e passado o erro no campo "errors" com o status de "erro".

### ▼ **Renegotiations** *(Renegociar uma oferta que ainda não foi liquidada)*

**Not implemented.**

### ▼ **Webhooks** *(Busca eventos e inserir no Sankhya)*

## ▼ 1. Buscar Events (Publisher)

Tendo como responsabilidade de buscar os eventos com status "open" no database, até o momento sendo MongoDB, collection webhook\_kanastra.

exemplo dos eventos:

```
[
  {
    "id": "667f06d065cb732c16d93704",
    "type": "offers",
    "status": "processed",
    "attempts": 0,
    "content": {
      "offer_id": "f5203103-655f-4714-be85-6929600532ed",
      "external_id": "2411463",
      "offer_status": "Teste Webhook",
      "acquisition_status": "Teste Webhook",
      "acquisition_id": "0a38a02e-efbb-4462-ad29-9a409f163a12",
      "acquisition_price": "70904.58",
      "future_amount": "24115.38",
      "created_at": "20/02/2023",
      "updated_at": "27/05/2023",
      "validations_with_error": [
        "Teste Webhook"
      ],
      "installments": [
        "Teste Webhook"
      ],
    },
    "request_type": "Incoming Request",
    "from": {
      "sign": {
        "sub": "666af3f8ea3abc2767acb86f"
      },
      "iat": 1719591379
    },
    "created_at": "2024-06-10T10:55:00.000",
    "updated_at": "2024-06-10T10:55:00.000",
    "message": null
  },
  {
    "id": "667f07a165cb732c16d93706",
    "type": "liquidations",
    "status": "processed",
    "attempts": 0,
    "content": {
      "id": "42597938-a406-48e2-ad5c-d8479040446f",
      "status": "SETTLED",
      "event": "liquidation_bond",
      "offer_id": "725854b3-f762-4083-801f-1a8918f264f2",
      "offer_external_id": "12345abcdef",
    }
  }
]
```

```

    "installment_external_id": "12345abcdef"
  },
  "request_type": "Incoming Request",
  "from": {
    "sign": {
      "sub": "666af3f8ea3abc2767acb86f"
    },
    "iat": 1719591379
  },
  "created_at": "2024-06-10T10:55:00.000",
  "updated_at": "2024-06-10T10:55:00.000",
  "message": null
}
]

```

Visto que os webhooks recebidos da Kanastra, são: "offers", "liquidations", "repurchases" e "renegotiations". ("acquisitions" não possui uso na nossa operação)

Apenas as offers possuem um conteúdo diferente, sendo assim, foi adotado a ideia de ter apenas uma queue para offers (alta demanda durante a criação) e 1 para todas as demais, já que tem um padrão genérico.

Para que não execute uma sessão/inserção no Sankhya para cada webhook recebido, essa queue irá rodar de 5 em 5 minutos (\* \*/5 \* \*) porque não é algo que necessita de uma alta demanda, ela irá consumir todos eventos e quebrar por tipos, assim inserindo nos eventos offers → "webhook-offer-handler" e os demais generic → "webhook-generic-handler", seguindo a capacidade máxima de inserção de registros que é 50.

```

/*
  - Primeiro parametro deve ser o array que será particionado

  - Segundo parametro deve ser o campo que será usado como comparador para
  o particionamento.

  - Recebe tambem um generic para inferir o tipo do dado dentro do array
*/

const webhooksByType = this.partitionWebhooksByType<Webhooks>(webhooks, 'type'

```

Exemplo de saída:

```

{
  "offers": [
    {
      "id": "667f06d065cb732c16d93704",
      "type": "offers",
      "status": "processed",
      "attempts": 0,
      "content": {

```

```

        "offer_id": "f5203103-655f-4714-be85-6929600532ed",
        "external_id": "2411463",
        "offer_status": "Teste Webhook",
        "acquisition_status": "Teste Webhook",
        "acquisition_id": "0a38a02e-efbb-4462-ad29-9a409f163a12",
        "acquisition_price": "70904.58",
        "future_amount": "24115.38",
        "created_at": "20/02/2023",
        "updated_at": "27/05/2023",
        "validations_with_error": [
            "Teste Webhook"
        ]
    },
    "request_type": "Incoming Request",
    "from": {
        "sign": {
            "sub": "666af3f8ea3abc2767acb86f"
        },
        "iat": 1719591379
    },
    "created_at": "2024-06-10T10:55:00.000",
    "updated_at": "2024-06-10T10:55:00.000",
    "message": null
}
],
"liquidations": [
    {
        "id": "667f07a165cb732c16d93706",
        "type": "liquidations",
        "status": "processed",
        "attempts": 0,
        "content": {
            "id": "42597938-a406-48e2-ad5c-d8479040446f",
            "status": "SETTLED",
            "event": "liquidation_bond",
            "offer_id": "725854b3-f762-4083-801f-1a8918f264f2",
            "offer_external_id": "12345abcdef",
            "installment_external_id": "12345abcdef"
        },
        "request_type": "Incoming Request",
        "from": {
            "sign": {
                "sub": "666af3f8ea3abc2767acb86f"
            },
            "iat": 1719591379
        },
        "created_at": "2024-06-10T10:55:00.000",
        "updated_at": "2024-06-10T10:55:00.000",
        "message": null
    }
]
}

```

Após inserido nas filas, é efetuado o update dos mesmo para o status "processing", para que seja removido das próximas consultas.

### ▼ **2.1 webhook-offer-handler (Subscriber)**

Recebendo um array dos eventos, é iterado para manter apenas o "content", assim escrevendo o body de inserção no Sankhya, como essa operação é um pouco diferente das demais, segue abaixo um exemplo de entrada e saída dos dados:

```
const requestBody = saveWebhookOfferQuery({
  entityName: "AD_WEBHOOKOFFERKANASTRA",
  fields: [
    "OFFERID",
    "EXTERNALID",
    "OFFERSTATUS",
    "ACQUISITIONSTATUS",
    "ACQUISITIONID",
    "ACQUISITIONPRICE",
    "FUTUREAMOUNT",
    "CREATEDAT",
    "UPDATEDAT",
    "VALIDATIONSWITHERROR",
    "INSTALLMENTS"
  ],
  values, // array de offers
})
```

Input:

```
[
  {
    "offer_id": "f5203103-655f-4714-be85-6929600532ed",
    "external_id": "2411463",
    "offer_status": "Teste Webhook",
    "acquisition_status": "Teste Webhook",
    "acquisition_id": "0a38a02e-efbb-4462-ad29-9a409f163a12",
    "acquisition_price": "70904.58",
    "future_amount": "24115.38",
    "created_at": "20/02/2023",
    "updated_at": "27/05/2023",
    "validations_with_error": [
      "Teste Webhook"
    ],
    "installments": [
      "Teste Webhook"
    ]
  }
]
```



```
  },  
]
```

Output:

```
{  
  "entityName": "AD_WEBHOOKOFFERKANASTRA",  
  "standAlone": false,  
  "fields": [  
    "OFFERID",  
    "EXTERNALID",  
    "OFFERSTATUS",  
    "ACQUISITIONSTATUS",  
    "ACQUISITIONID",  
    "ACQUISITIONPRICE",  
    "FUTUREAMOUNT",  
    "CREATEDAT",  
    "UPDATEDAT",  
    "VALIDATIONSWITHERROR",  
    "INSTALLMENTS"  
  ],  
  "records": [  
    {  
      "values": {  
        "0": "f5203103-655f-4714-be85-6929600532ed",  
        "1": "2411463",  
        "2": "Teste Webhook",  
        "3": "Teste Webhook",  
        "4": "0a38a02e-efbb-4462-ad29-9a409f163a12",  
        "5": "70904.58",  
        "6": "24115.38",  
        "7": "20/02/2023",  
        "8": "27/05/2023",  
        "9": "[\"Teste Webhook\"]",  
        "10": "[\"Teste Webhook\"]"  
      }  
    }  
  ]  
}
```

Ao final da inserção, é alterado seu status para "processed" no database, senão, alterado para "error" com a mensagem de erro oriunda do Sankhya.

## ▼ **2.2. webhook-generic-handler (Subscriber)**

Recebendo um array dos eventos, é iterado para manter apenas o "content", assim escrevendo o body de inserção no Sankhya, como essa operação é um pouco diferente das demais, segue abaixo

um exemplo de entrada e saída dos dados:

```
const requestBody = saveWebhookGenericQuery({
  entityName: "AD_WEBHOOKKANASTRA",
  fields: [
    "ID_WEBHOOK",
    "STATUS",
    "EVENT",
    "OFFER_ID",
    "OFFER_EXTERNAL_ID",
    "INSTALLMENT_EXTERNAL_ID"
  ],
  values, // array de generics (liquidations, repurchases, renegotiations)
})
```

Input:

```
[
  {
    "id": "42597938-a406-48e2-ad5c-d8479040446f",
    "status": "SETTLED",
    "event": "liquidation_bond",
    "offer_id": "725854b3-f762-4083-801f-1a8918f264f2",
    "offer_external_id": "12345abcdef",
    "installment_external_id": "12345abcdef"
  }
]
```

Output:

```
{
  "entityName": "AD_WEBHOOKKANASTRA",
  "standAlone": false,
  "fields": [
    "ID_WEBHOOK",
    "STATUS",
    "EVENT",
    "OFFER_ID",
    "OFFER_EXTERNAL_ID",
    "INSTALLMENT_EXTERNAL_ID"
  ],
  "records": [
    {
      "values": {
        "0": "42597938-a406-48e2-ad5c-d8479040446f",
        "1": "SETTLED",
        "2": "liquidation_bond",

```

```

    "3": "725854b3-f762-4083-801f-1a8918f264f2",
    "4": "12345abcdef",
    "5": "12345abcdef"
  }
}
]
}

```

Ao final da inserção, é alterado seu status para “processed” no database, senão, alterado para “error” com a mensagem de erro oriunda do Sankhya.

## 3. Views

### Informações de todas as views que são utilizadas na integração

Name	Production	Dev
Create Offers	AD_VWOFFERSKANASTRA	AD_VWOFFERSKANASTRATESTE
Create Acquisitions	AD_VWACQUISITIONKANASTRA	AD_VWACQUISITIONKANASTRATESTE
Create Liquidations	AD_VWLIQUIDATIONSKANASTRA	AD_VWLIQUIDATIONSKANASTRATESTE
Create Repurchases	AD_VWREPURCHASESKANASTRA	AD_VWREPURCHASESKANASTRATESTE
Create Renegotiations	AD_VWRENEGOTIATIONSKANASTRA	AD_VWRENEGOTIATIONSKANASTRATESTE
Create Offer Repurchase	AD_VWOFFERSREPURCHASEKANASTRA	AD_VWOFFERSREPURCHASEKANASTRATESTE
Create Liquidation Repurchase	AD_VWLIQUIDATIONSREPURCHASEKANASTRA	AD_VWLIQUIDATIONSREPURCHASEKANASTRATESTE
Retorno Webhook Generic (Repurchases, Liquidations, Renegotiations)	AD_WEBHOOKKANASTRA	AD_WEBHOOKKANASTRATESTE
Retorno Webhook Offers	AD_WEBHOOKOFFERKANASTRA	AD_WEBHOOKOFFERKANASTRATESTE

#### ▼ AD\_VWOFFERKANASTRA

A query abaixo é responsável pela criação das Ofertas na Kanastra, os registros só aparecem nessa view, quando o time do financeiro, alteram os títulos para os seguintes tipos de negociação: **206 e 207**.

O **sponsorPersonType** está como **LEGAL\_PERSON**, pois se trata de pessoa jurídica.

O Join na tabela **TSIATA** refere-se ao anexo do canhoto, que é adicionado pelo time da logística, como nem sempre as ofertas estão com esse anexo durante o envio, retiramos o filtro de anexo.

O join na tabela **AD\_OFFERKANASTRA** é para trazermos somente dados que ainda não foram enviados para a Kanastra. Uma vez que enviados e retornado sucesso, essa tabela é alimentada, fazendo com que o dado não retorne mais na View.

```
SELECT
    DISTINCT C.NUNOTA AS externalId,
    C.VLRNOTA,
    CASE
        WHEN P.RAZAOSOCIAL LIKE '%&%' THEN SUBSTRING(REPLACE(P.RAZAOSOCIAL, '&',
            ELSE SUBSTRING(REPLACE(P.RAZAOSOCIAL, '&', 'E'), 1, 60)
    END AS sponsorName,
    'LEGAL_PERSON' AS sponsorPersonType,
    P.CGC_CPF AS sponsorGovernmentId,
    P.CODPARC AS sponsorExternalCode,
    EN.NOMEEND AS sponsorAddress,
    dbo.FN_FORMATAR_TEXTO(P.NUMEND) AS sponsorAddressNumber,
    CASE
        WHEN P.COMPLEMENTO = '' THEN NULL
        ELSE dbo.FN_FORMATAR_TEXTO(P.COMPLEMENTO)
    END AS sponsorAddressComplement,
    B.NOMEBAI AS sponsorNeighborhood,
    dbo.FN_FORMATAR_TEXTO(CI.NOMECID) AS sponsorCity,
    UF.UF AS sponsorState,
    'Brasil' AS sponsorCountry,
    P.CEP AS sponsorZipCode,
    '04904042000108' AS sellerGovernmentId,
    'false' AS coobrigation,
    --Co-obrigação por parte do cedente?
    NULL AS customFields,
    CONCAT (
        '',
        '[{ "assetType": "NOTA_FISCAL",
        "invoiceNumber": "',
        F.NUMNOTA,
        '"',
        "invoiceDate": "',
        + ISNULL(
            CONVERT(VARCHAR, FORMAT(C.DTFATUR, 'yyyyMMdd'), 102),
            0
        ),
        '"',
        "invoiceKey": "',
        C.CHAVENFE,
        '"',
        "totalInstallments":',
        CASE
            WHEN F.DESDOBRAMENTO IS NULL THEN 0
            ELSE(
```

```

        SELECT
            COUNT(FIN2.NUFIN) TOTALINSTALLMENTS
        FROM
            TGFFIN (NOLOCK) FIN2
        WHERE
            FIN2.NUNOTA = C.NUNOTA
            AND FIN2.CODTIPTIT IN (206, 207)
    )
END,
',
"paymentValue": ',
(
    SELECT
        SUM(FIN2.VLRDESDOB) PAYMENTVALUE
    FROM
        TGFFIN (NOLOCK) FIN2
    WHERE
        FIN2.NUNOTA = C.NUNOTA
        AND FIN2.CODTIPTIT IN (206, 207)
),
',
"paymentDate": '',
(
    SELECT
        MAX(
            ISNULL(
                CONVERT(VARCHAR, FORMAT(FIN2.DTVENC, 'yyyyMMdd'), 102),
                0
            )
        )
    FROM
        TGFFIN FIN2
    WHERE
        FIN2.NUNOTA = C.NUNOTA
),
'',
"customFields": {
    "preCalculatedAcquisitionPrice":',
(
    SELECT
        SUM(FIN2.VLRDESDOB) PAYMENTVALUE
    FROM
        TGFFIN (NOLOCK) FIN2
    WHERE
        FIN2.NUNOTA = C.NUNOTA
        AND FIN2.CODTIPTIT IN (206, 207)
),
',
    "rateType": "PRE"
},

```

```

        "installments":',
        CASE

            WHEN F.DESDOBRAMENTO IS NULL THEN '[]}]'
            ELSE (

                SELECT
                    REPLACE(
                        '[' + (

                            SELECT
                                '{"externalId":"' + RTRIM(CONVERT(VARCHAR(100), F
                                    + '"amount":"' + RTRIM(CONVERT(DECIMAL(10,
                                    + '"customFields":"' + '{}' + ', ' + '"dueD
                                    CONVERT(VARCHAR, FORMAT(F.DTVENC, 'yyyyMMdd')
                                    0
                                ) + '"}',
                                ', '
                            FROM
                                SANKHYA.TGFFIN F (NOLOCK)
                            WHERE
                                F.NUNOTA = C.NUNOTA
                                AND F.CODTIPTIT IN (206, 207) FOR XML PATH ('')
                        ) + ']',
                        ', ]',
                        '[]}]'
                    )
                )
            ) AS items,
        DATACORRENT
FROM
    SANKHYA.TGFCAB C (NOLOCK)
    LEFT JOIN SANKHYA.TGFPAR P (NOLOCK) ON P.CODPARC = C.CODPARC
    LEFT JOIN SANKHYA.TGFFIN F (NOLOCK) ON F.NUNOTA = C.NUNOTA
    LEFT JOIN SANKHYA.TSICID CI (NOLOCK) ON CI.CODCID = P.CODCID
    LEFT JOIN SANKHYA.TSIBAI B (NOLOCK) ON B.CODBAI = P.CODBAI
    LEFT JOIN SANKHYA.TSIEND EN (NOLOCK) ON EN.CODEND = P.CODEND
    LEFT JOIN SANKHYA.TSIUFS UF (NOLOCK) ON UF.CODUF = CI.UF
    LEFT JOIN SANKHYA.TSIATA ATA (NOLOCK) ON ATA.CODATA = C.NUNOTA
    LEFT JOIN SANKHYA.AD_OFFERKANASTRA KA (NOLOCK) ON KA.NUNOTA = C.NUNOTA
WHERE
    F.CODTIPTIT IN (206, 207)
    AND F.DTALTER BETWEEN DATEADD(DD, -1, GETDATE())
    AND GETDATE()
    AND KA.NUNOTA IS NULL

```

#### ▼ AD\_VWACQUISITIONKANASTRA

Essa query, retorna acquisitions que deverão ser submetidas, cruzando os dados que foram enviados na tabela de [AD\\_OFFERKANASTRA](#) e [AD\\_WEBHOOKOFFERKANASTRA](#).

Tudo que estiver na tabela [AD\\_WEBHOOKOFFERKANASTRA](#) com **acquisition-status** diferente de **invalid**, **submitted**, igual a **open** e, **offer-status** igual a **pre-validated**, serão retornados nessa view e

enviados para a Kanastra.

```
SELECT
    DISTINCT H.ACQUISITIONID AS 'id'
FROM
    AD_OFFERKANASTRA K (NOLOCK)
    INNER JOIN AD_WEBHOOKOFFERKANASTRA H (NOLOCK) ON K.NUNOTA = H.EXTERNALID
WHERE
    OFFERSTATUS = 'pre-validated'
    AND ACQUISITIONSTATUS = 'open'
    AND NOT EXISTS (

        SELECT
            1
        FROM
            AD_WEBHOOKOFFERKANASTRA K2 (NOLOCK)
        WHERE
            K2.EXTERNALID = H.EXTERNALID
            AND K2.ACQUISITIONID = H.ACQUISITIONID
            AND K2.ACQUISITIONSTATUS IN ('invalid', 'submitted')
    )
    AND DATEIN BETWEEN DATEADD(DD, -1, GETDATE())
    AND GETDATE()
```

#### ▼ AD\_VWLIQUIDATIONSKANASTRA

Essa query retorna liquidações que são importadas na tela **AD\_KANASTAIMPOR - Liquidação Kanastra**, dentro do Sankhya pelo time do financeiro. Além disso, para serem liquidadas precisamos receber as seguintes **acquisition-status** no Webhook (***acquired, paid\_to\_seller***).

O join na tabela de AD\_OFFERKANASTRA é obrigatório, pois eu só posso transacionar nessa view, registros que tiveram suas offers em algum momento criadas na Kanastra.

```
SELECT
    DISTINCT C.nunota,
    F.NUFIN AS externalId,
    NULL AS customFields,
    (
        SELECT
            REPLACE(
                '' + (
                    SELECT
                        '{\"amount\":' + RTRIM(CONVERT(DECIMAL(10, 2), LIQUI.VLRDESDOB)) + ', '
                        + '\"method\":\"' + CASE
                            WHEN F.CODBCO != 999 THEN 'BOLETO'
                            WHEN F.CODBCO = 999 THEN 'TED'
                            ELSE 'BOLETO'
                        END + '\", ' + '\"reference\":\"'
                        + ISNULL(RTRIM(CONVERT(VARCHAR(100), F.NOSSONUM)), 0)
                        + '\", ' +
                        '\"date\":\"' + ISNULL(
                            CONVERT(
                                VARCHAR,
                                FORMAT(LIQUI.DHBAIXA, 'yyyy-MM-dd' ' 'HH:mm:ss'),

```

```

        102
        ),
        0
    ) + '"}',
    ', ' FOR XML PATH ( '' )
) + ']',
',]',
'',
''
)
) AS payment,
(
    SELECT
        REPLACE(
            '' + (
                SELECT
                    '{"governmentId":"' + CGC_CPF + '",' + '"country":"' + 'Brazil' + '"'
                    ', ' FOR XML PATH ( '' )
                ) + ']',
            ',]',
            ''
        )
    ) AS payer,
    'TOTAL' AS settlementType,
    F.DTVENC

FROM

SANKHYA.TGFCAB C          (NOLOCK)
LEFT JOIN SANKHYA.TGFPAR P (NOLOCK) ON P.CODPARC = C.CODPARC
LEFT JOIN SANKHYA.TGFVEN V (NOLOCK) ON V.CODVEND = C.CODVEND
LEFT JOIN SANKHYA.TGFFIN F (NOLOCK) ON F.NUNOTA = C.NUNOTA
AND CODTIPTIT IN (206, 207)
LEFT JOIN SANKHYA.TGFTPV T (NOLOCK) ON T.CODTIPVENDA = C.CODTIPVENDA
AND T.DHALTER = C.DHTIPVENDA
LEFT JOIN SANKHYA.AD_OFFERKANASTRA KA (NOLOCK) ON KA.NUNOTA = C.NUNOTA
LEFT JOIN AD_WEBHOOKOFFERKANASTRA W (NOLOCK) ON W.EXTERNALID = f.NUNOTA
LEFT JOIN AD_LIQUIDATIONSKANASTRA K (NOLOCK) ON K.NUFIN = F.NUFIN
INNER JOIN AD_KANASTAIMPOR LIQUI (NOLOCK) ON LIQUI.NUFIN = F.NUFIN
WHERE
    C.TIPMOV = 'V'
    AND F.DHBAIXA IS NOT NULL
    AND W.ACQUISITIONSTATUS IN ('ACQUIRED', 'paid_to_seller')
    AND W.OFFERSTATUS = 'VALIDATED'
    AND K.STATUS IS NULL

```

#### ▼ **AD\_VWRENEGOTIATIONSKANASTRA**

A premissa para a renegociação é o título estar em aberto junto a Kanastra, o mesmo serve para alterar valores e datas de vencimentos.

A query abaixo, retorna dados que foram importados na tela

**AD\_RENEGOTIATIONSKANASTRA** pelo time do Financeiro. O join com a tabela **AD\_RENEGNUFIN** refere-se a novas parcelas que serão renegociadas.



```

SELECT DISTINCT
C.nunota,
R.NUFIN AS externalId,
R.paidamount,
COALESCE(
(
SELECT
REPLACE(
'[' + (
SELECT
'{"externalId":"' + ISNULL(RTRIM(CONVERT(VARCHAR(100), N.NEWNUFIN)), '')
+ '",' +
'"amount":"' + ISNULL(RTRIM(CONVERT(VARCHAR(100), N.amount)), '') + ',' +
'"acquisitionprice":"' + ISNULL(RTRIM(CONVERT(VARCHAR(100), N.ACQUISIT
'') + ',' +
CASE
WHEN N.feeAmount IS NOT NULL THEN
'"feeAmount":"' + RTRIM(CONVERT(VARCHAR(100), N.feeAmount)) + ',
ELSE
''
END +
CASE
WHEN N.settlementType IS NOT NULL THEN
'"settlementType":"' + ISNULL(RTRIM(CONVERT(VARCHAR(100),
N.settlementType)), '') + ','
ELSE
''
END +
'"dueDate":"' +
ISNULL(CONVERT(VARCHAR, FORMAT(N.DUEDATE, 'yyyy-MM-dd'), 102), '') + '
','
)
WHERE
R.NUFIN = N.NUFIN FOR XML PATH ('')
) + ']',
'],]',
']'
)
),
'[]'
) AS conditions

FROM SANKHYA.TGFCAB C (NOLOCK)
LEFT JOIN TGFFIN F (NOLOCK) ON F.NUNOTA = C.NUNOTA
INNER JOIN AD_RENEGOTIATIONSKANASTRA R (NOLOCK) ON R.NUFIN = F.NUFIN
INNER JOIN AD_RENEGNUFIN N (NOLOCK) ON N.NUFIN = R.NUFIN
LEFT JOIN AD_RESRENEGOTIATIONKANASTRA RET (NOLOCK) ON RET.NUFIN = R.NUFIN

WHERE RET.NUFIN IS NULL

```

#### ▼ AD\_VWREPURCHASESKANASTRA

A query abaixo, retorna dados que foram importados na tela **AD\_RECOMPRAKANASTRAFIN** pelo time do Financeiro. O join com a tabela **AD\_NEWEXTERNALIDREPURCHASE** refere-se a recompras que serão realizadas com outros títulos do financeiro caso exista.

A recompra ocorre quando você quer retirar o título do estoque.

Existem dois tipos de recompra:

Type **CASH** : Esse tipo de recompra, envolve depósito do dinheiro em conta.

Se for enviado com esse type, é necessário enviar a rota de liquidação na sequência para o mesmo título que foi recomprado, desse cenário surgiu a necessidade da criação da seguinte view:

**AD\_VWLIQUIDATIONSREPURCHASEKANASTRA.**

Type **ACQUISITION** : Esse tipo de recompra é realizado através de um ou mais títulos do mesmo valor do que está sendo recomprado, Exemplo: Tenho o título X com o valor de 5 milhões e quero recompra-lo. Posso operacionalizar com 5 novos títulos de 1 milhão cada, ou um único título de 5 milhões, mas com o vencimento maior do que o atual.

Se for enviado com esse type, é necessário enviar uma nova sessão logo na sequência, ou seja, uma nova offer esse título, desse cenário surgiu a necessidade da criação dessa view:

**AD\_VWOFFERSREPURCHASEKANASTRA.**

Por isso a importância da tabela **AD\_NEWEXTERNALIDREPURCHASE**, pois os novos títulos de type **ACQUISITION** estarão nessa tabela.

```
SELECT
  DISTINCT C.nunota,
  R.NUFIN AS externalId,
  R.EXPECTEDAMOUNT expectedAmount,
  CASE
    WHEN R.REASON IN ('C', 'COMMERCIAL_CHANGES') THEN 'COMMERCIAL_CHANGES'
    ELSE 'DISQUALIFICATION'
  END reason,
  CASE
    WHEN R.TYPE IN ('ACQ', 'ACQUISITION') THEN 'ACQUISITION'
    ELSE 'CASH'
  END type,
  COALESCE(
    (
      SELECT
        REPLACE(
          '[' + (
            SELECT
              + RTRIM(CONVERT(VARCHAR(100), N.NEWNUFIN)) + ', '
            FROM
              SANKHYA.AD_NEWEXTERNALIDREPURCHASE N (NOLOCK)
            WHERE
              F.NUFIN = N.NUFIN FOR XML PATH ('')
          ) + ']',
          ', ]',
          ''
        )
      ),
    '['
  ) AS newExternalId
FROM
```

```

SANKHYA.TGFCAB C (NOLOCK)
LEFT JOIN TGFFIN F (NOLOCK) ON F.NUNOTA = C.NUNOTA
INNER JOIN AD_RECOMPRAKANASTRAFIN R (NOLOCK) ON R.NUFIN = F.NUFIN
LEFT JOIN AD_NEWEXTERNALIDREPURCHASE N (NOLOCK) ON N.NUFIN = R.NUFIN
LEFT JOIN AD_REPURCHASESKANASTRA RET (NOLOCK) ON RET.NUFIN = R.NUFIN
WHERE
    RET.NUFIN IS NULL

```

#### ▼ AD\_VWLIQUIDATIONSREPURCHASEKANASTRA

O gatilho dessa view é acionado uma vez que a recompra foi realizada pelo Type **CASH**, sendo necessária a liquidação imediata do título que está sendo recomprado.

A query abaixo, retorna dados que foram realizados na operação de recompra e retornaram sucesso, onde o type é igual a

**CASH.**

```

SELECT DISTINCT

C.nunota,
F.NUFIN AS externalId,
null AS customFields,

(SELECT REPLACE(''+
    ( SELECT

        '{"amount":'+RTRIM(CONVERT(DECIMAL(10,2),RE.EXPECTEDAMOUNT))

        '"method":'+CASE WHEN F.CODBCO != 999 THEN 'BOLETO'
        WHEN F.CODBCO = 999 THEN 'TED' ELSE 'BOLETO' END  +',''+

        '"reference":'+ISNULL(RTRIM(CONVERT(VARCHAR(100),F.NOSSONUM

        '"date":'+ISNULL(CONVERT(VARCHAR,FORMAT(GETDATE(),
        'yyyy-MM-dd' ' 'HH:mm:ss'), 102),0)+'"'',',''

        FOR XML PATH (''))+']','',[','']) AS payment,

    (SELECT REPLACE(''+
        ( SELECT

            '{"governmentId":'+CGC_CPF+'','+

            '"country":'+Brazil+'"'',',''

            FOR XML PATH (''))+']','',[','']) AS payer,

    'TOTAL' AS settlementType,
    F.DTVENC,
    FORMAT(C.VLRNOTA, 'C', 'pt-BR') VLRNOTA

```

```

FROM SANKHYA.TGFCAB C                                (NOLOCK)
LEFT JOIN SANKHYA.TGFPAR P                            (NOLOCK) ON P.CODPARC      = C.CODPARC
LEFT JOIN SANKHYA.TGFVEN V                            (NOLOCK) ON V.CODVEND      = C.CODVEND
LEFT JOIN SANKHYA.TGFFIN F                            (NOLOCK) ON F.NUNOTA       = C.NUNOTA
LEFT JOIN SANKHYA.TGFTPV T                            (NOLOCK)
ON T.CODTIPVENDA = C.CODTIPVENDA AND T.DHALTER = C.DHTIPVENDA
INNER JOIN SANKHYA.AD_OFFERKANASTRA KA (NOLOCK) ON KA.NUNOTA      = C.NUNOTA
LEFT JOIN AD_LIQUIDATIONSKANASTRA K   (NOLOCK) ON K.NUFIN        = F.NUFIN
INNER JOIN AD_REPURCHASESKANASTRA R   (NOLOCK) ON R.NUFIN        = F.NUFIN
LEFT JOIN AD_RECOMPRAKANASTRAFIN RE   (NOLOCK) ON RE.NUFIN = R.NUFIN

WHERE K.STATUS IS NULL
AND RE.TYPE = 'CASH'

```

#### ▼ AD\_VWOFFERSREPURCHASEKANASTRA

O gatilho dessa view é acionado uma vez que a recompra foi realizada pelo Type **ACQUISITION**, sendo necessário o envio de uma nova sessão do título que está sendo recomprado.

A query abaixo, retorna dados que foram realizados na operação de recompra e retornaram sucesso, onde o type é igual a

**ACQUISITION.**

```

SELECT DISTINCT

C.NUNOTA AS externalId,
--C.DTFATUR,
--f.nufin,
C.VLRNOTA,
REPLACE(P.RAZAOSOCIAL, '&', 'E') AS sponsorName,
'LEGAL_PERSON' AS sponsorPersonType,
P.CGC_CPF AS sponsorGovernmentId,
P.CODPARC AS sponsorExternalCode,
EN.NOMEEND AS sponsorAddress,
dbo.FN_FORMATAR_TEXTO(P.NUMEND) AS sponsorAddressNumber,
CASE WHEN P.COMPLEMENTO = ''
THEN NULL ELSE dbo.FN_FORMATAR_TEXTO(P.COMPLEMENTO) END AS sponsorAddressCompleme
B.NOMEBAI AS sponsorNeighborhood,
dbo.FN_FORMATAR_TEXTO(CI.NOMECID) AS sponsorCity,
UF.UF AS sponsorState,
'Brasil' AS sponsorCountry,
P.CEP AS sponsorZipCode,
'04904042000108' AS sellerGovernmentId,

'false' AS coobrigation,
null AS customFields,
CONCAT ('', [{ "assetType": "NOTA_FISCAL",
"invoiceNumber": '', Fin.NUMNOTA, '',
"invoiceDate": '', +ISNULL(CONVERT(VARCHAR, FORMAT(C.DTFATUR, 'yyyyMMdd'), 102), 0),
"invoiceKey": '', C.CHAVENFE, '',

```

```

"totalInstallments":',CASE WHEN Fin.DESDOBRAMENTO IS NULL THEN 0 ELSE
(SELECT COUNT(FIN2.NUFIN) TOTALINSTALLMENTS
FROM TGFFIN (NOLOCK) FIN2 WHERE FIN2.NUFIN = NEWNUFIN.NUFIN ) END,',
"paymentValue": ',(SELECT SUM(FIN2.VLRDESDOB) PAYMENTVALUE FROM TGFFIN (NOLOCK)
WHERE FIN2.NUFIN = NEWNUFIN.NUFIN),',
"paymentDate": '',(SELECT MAX(ISNULL(CONVERT(VARCHAR,FORMAT(FIN2.DTVENC,'yyyyMMdd'
102),0)) FROM TGFFIN FIN2 WHERE FIN2.NUNOTA = C.NUNOTA) ,'',
"customFields": {
    "preCalculatedAcquisitionPrice":',(SELECT SUM(FIN2.VLRDESDOB) PAYMENTVALU
FROM TGFFIN (NOLOCK) FIN2 WHERE FIN2.NUFIN = NEWNUFIN.NUFIN),',
    "rateType": "PRE"
},
"installments":',CASE WHEN Fin.DESDOBRAMENTO IS NULL THEN '[]]' ELSE
(SELECT REPLACE('[ ' +

( SELECT

'{"externalId":"' +RTRIM(CONVERT(VARCHAR(100),FIN.NUFIN))+''',' +

'"amount":'+RTRIM(CONVERT(DECIMAL(10,2),Fin.VLRDESDOB))+''',' +

'"customFields":'+'{ }'+''',' +

'"dueDate":"' +ISNULL(CONVERT(VARCHAR,FORMAT(F.DTVENC,'yyyyMMdd'), 102),0)+''''}

FROM SANKHYA.TGFFIN F (NOLOCK)
WHERE F.nufin = NEWNUFIN.nufin

FOR XML PATH ('') ) +']',' ','[]]]'))END ) as items

```

```

FROM SANKHYA.AD_NEWEXTERNALIDREPURCHASE NEWNUFIN (NOLOCK)
LEFT JOIN SANKHYA.TGFFIN FIN (NOLOCK) ON FIN.NUFIN = NEWNUFIN.NUFIN
LEFT JOIN SANKHYA.AD_RECOMPRAKANASTRAFIN REFIN (NOLOCK)
ON REFIN.NUFIN = NEWNUFIN.NUFIN
INNER JOIN SANKHYA.AD_REPURCHASESKANASTRA RETREP (NOLOCK)
ON RETREP.NUFIN = NEWNUFIN.NUFIN
LEFT JOIN SANKHYA.AD_OFFERKANASTRA OFFER (NOLOCK) ON OFFER.NUNOTA = FIN.NUNOTA
LEFT JOIN SANKHYA.TGFCAB C (NOLOCK) ON C.NUNOTA = FIN.NUNOTA
LEFT JOIN SANKHYA.TGFPAR P (NOLOCK) ON P.CODPARC = C.CODPARC
LEFT JOIN SANKHYA.TGFVEN V (NOLOCK) ON V.CODVEND = C.CODVEND
LEFT JOIN SANKHYA.TGFTPV T (NOLOCK) ON T.CODTIPVENDA = C.CODTIPVENDA
AND T.DHALTER = C.DHTIPVENDA
LEFT JOIN SANKHYA.TGFPAR M (NOLOCK) ON P.CODPARCMATRIZ = M.CODPARC
LEFT JOIN SANKHYA.TSICID CI (NOLOCK) ON CI.CODCID = P.CODCID
LEFT JOIN SANKHYA.TSIBAI B (NOLOCK) ON B.CODBAI = P.CODBAI
LEFT JOIN SANKHYA.TSIEND EN (NOLOCK) ON EN.CODEND = P.CODEND
LEFT JOIN SANKHYA.TSIUFS UF (NOLOCK) ON UF.CODUF = CI.UF

```

```

LEFT JOIN SANKHYA.TSIEMP EMP (NOLOCK) ON EMP.CODEMP = C.CODEMP
LEFT JOIN SANKHYA.TSICID CID (NOLOCK) ON EMP.CODCID = CID.CODCID
LEFT JOIN SANKHYA.TSIEND TEND (NOLOCK) ON TEND.CODEND = EMP.CODEND
LEFT JOIN SANKHYA.TSIBAI BAI (NOLOCK) ON BAI.CODBAI = EMP.CODBAI
LEFT JOIN SANKHYA.TSIUFS UF1 (NOLOCK) ON UF1.CODUF = CID.UF
LEFT JOIN SANKHYA.TSIATA ATA (NOLOCK) ON ATA.CODATA = C.NUNOTA
LEFT JOIN SANKHYA.AD_OFFERKANASTRA KA (NOLOCK) ON KA.NUNOTA = C.NUNOTA

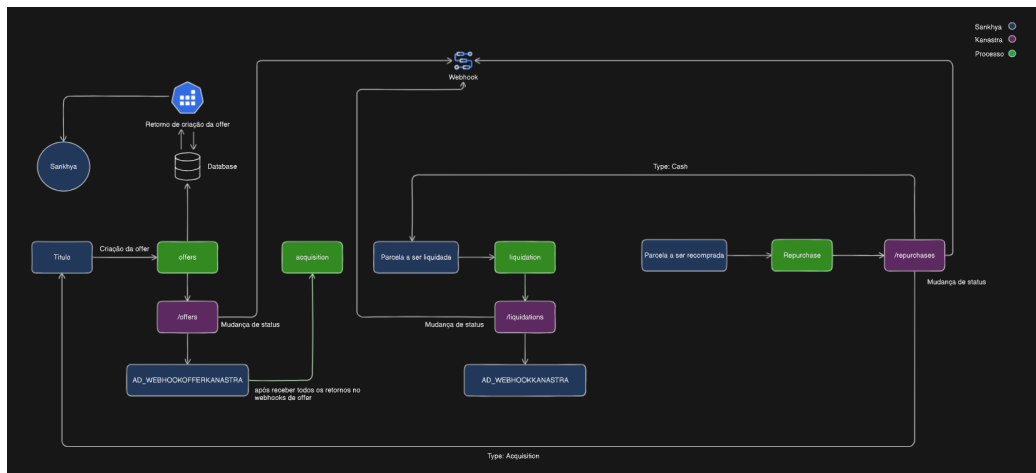
```

```

WHERE TYPE = 'ACQ'
AND OFFER.STATUS IS NULL

```

## 4. Process



## 5. Webhooks

### Especificações

Domain: [rtech-app.com](https://rtech-app.com) (Hostinger)

DNS: Cloudfare

Subdomain: kanastra

Server: t2.micro AWS EC2

IP: 18.229.201.145

DNS IPv4: [ec2-18-229-201-145.sa-east-1.compute.amazonaws.com](https://ec2-18-229-201-145.sa-east-1.compute.amazonaws.com)

URL: <https://kanastra.rtech-app.com/>

SSH: kanastra.pem

Rules:

- Security Group
- IP's All CIDR 32

1. 34.138.109.139 - Kanastra 1
2. 34.139.14.100 - Kanastra 2
3. 34.138.103.47 - Kanastra 3
4. 177.92.85.202 - Rcell 1

SSH

1. 177.92.85.202 - Rcell 1

- Elastic IP

## Routes

Todos os webhooks são criadas com status "open" para seguir para processamento, citado na etapa de queues.

### ▼ /access\_token

Rota para gerar bearer token. Quando um novo for gerado, o anterior será invalidado.

```
{
  "email": "teste@localhost.com.br",
  "password": "123456789"
}
```

Response:

```
{
  "token": "eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJzawduIjp7InN1YiI6IjY2N2YxNmI
}
```

### ▼ /offers

Webhook recebido após a validação de elegibilidade do título criado na rota de /offers

```
{
  "offer_id": "725854b3-f762-4083-801f-1a8918f264f2",
  "external_id": "12345abcdef",
  "offer_status": "received",
  "acquisition_status": "open",
  "acquisition_id": "42e93c98-92f1-4731-a7c7-27395bd8cf39",
  "acquisition_price": 1293901.12,
  "future_amount": 1294000.12,
  "created_at": "2022-10-19",
  "updated_at": "2022-10-19",
  "validations_with_error": [
    "string"
  ]
}
```

```
]
}
```

#### ▼ /liquidations

Webhook recebido após a validação do título liquidado.

```
{
  "id": "42597938-a406-48e2-ad5c-d8479040446f",
  "status": "SETTLED",
  "event": "liquidation_bond",
  "offer_id": "725854b3-f762-4083-801f-1a8918f264f2",
  "offer_external_id": "12345abcdef",
  "installment_external_id": "12345abcdef"
}
```

#### ▼ /repurchases

Webhook recebido após a validação da parcela recomprada.

```
{
  "id": "42597938-a406-48e2-ad5c-d8479040446f",
  "status": "SETTLED",
  "event": "repurchase",
  "offer_id": "725854b3-f762-4083-801f-1a8918f264f2",
  "offer_external_id": "12345abcdef",
  "installment_external_id": "12345abcdef"
}
```

#### ▼ /renegotiations

Webhook recebido após a validação da parcela renegociada.

```
{
  "id": "42597938-a406-48e2-ad5c-d8479040446f",
  "status": "SETTLED",
  "event": "renegotiation",
  "offer_id": "725854b3-f762-4083-801f-1a8918f264f2",
  "offer_external_id": "12345abcdef",
  "installment_external_id": "12345abcdef"
}
```

---

**Last updated:** 09 de outubro de 2024 (09/10/2024 09:46:00)