

# Construindo a NuConta

Gustavo Bicalho  
Maurício Verardo

# Agenda

- NuConta
- Microserviços no Nubank
- Transferindo dinheiro entre NuContas
  - Event-sourcing: Modularidade e Escalabilidade
  - Consistência em sistemas distribuídos
- O feed de movimentações
  - Backend for Frontends com GraphQL

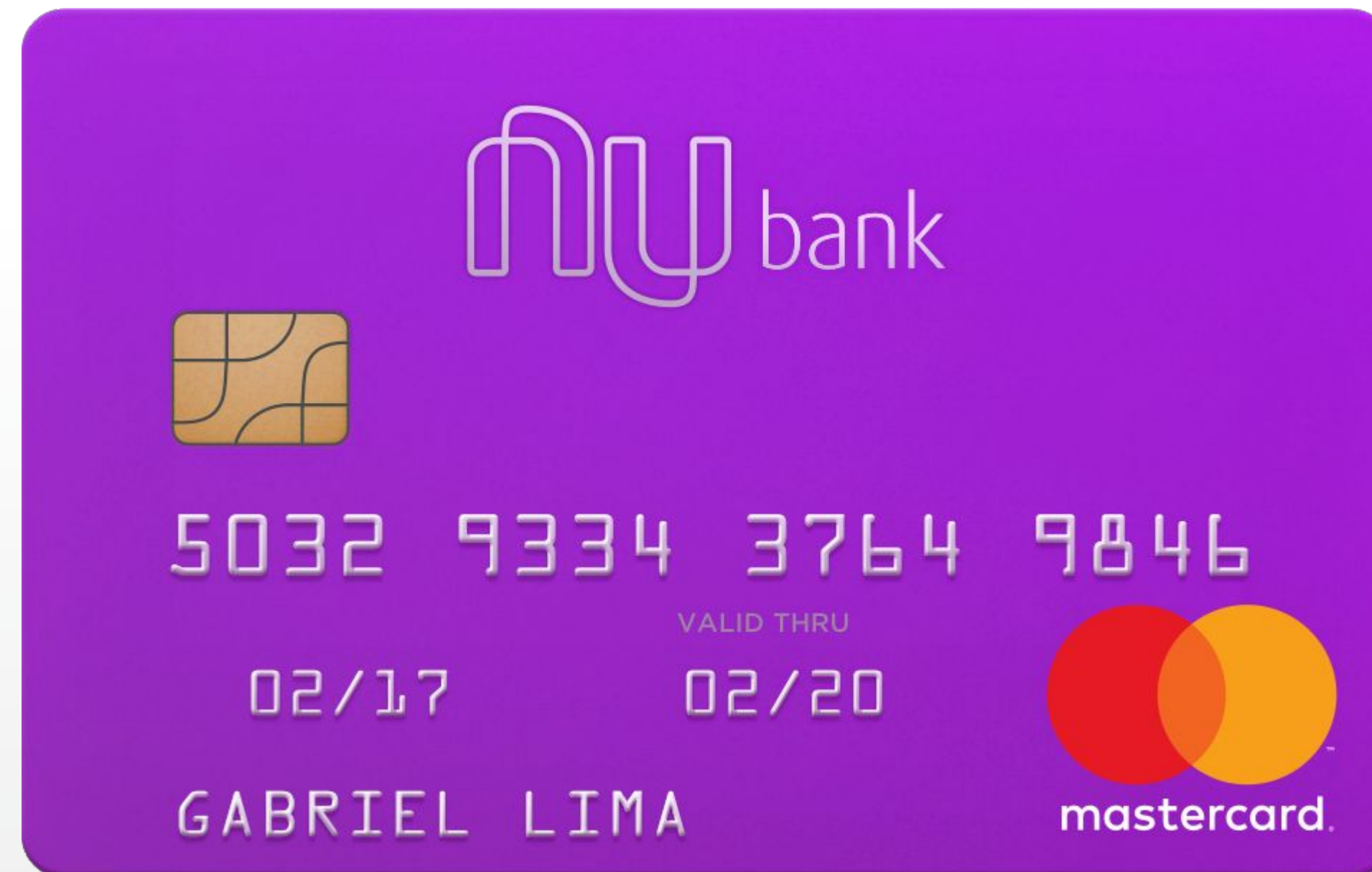
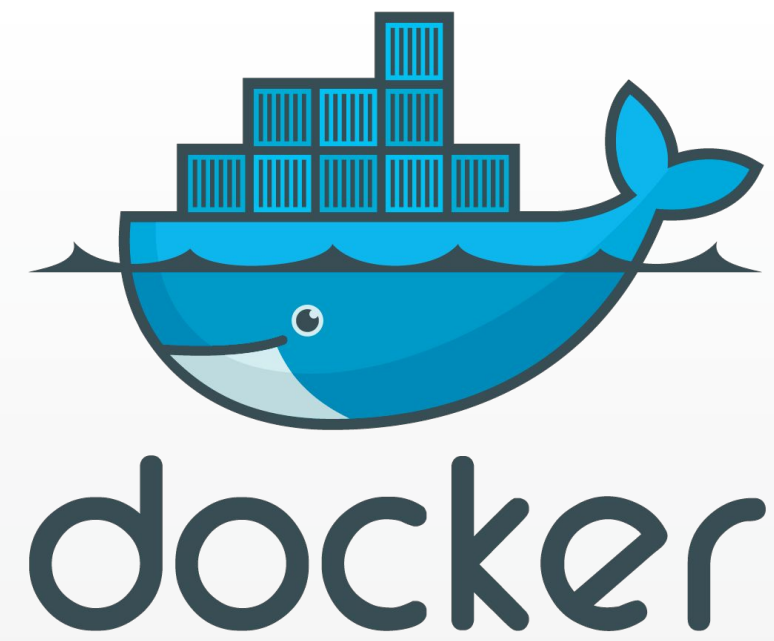


An aerial night photograph of a city, likely San Francisco, showing a winding river (San Francisco Bay) and a coastline. The city lights are visible, and the water is dark. The text "NuConta" is overlaid on the left side of the image.

**NuConta**



# NuConta





# NuConta





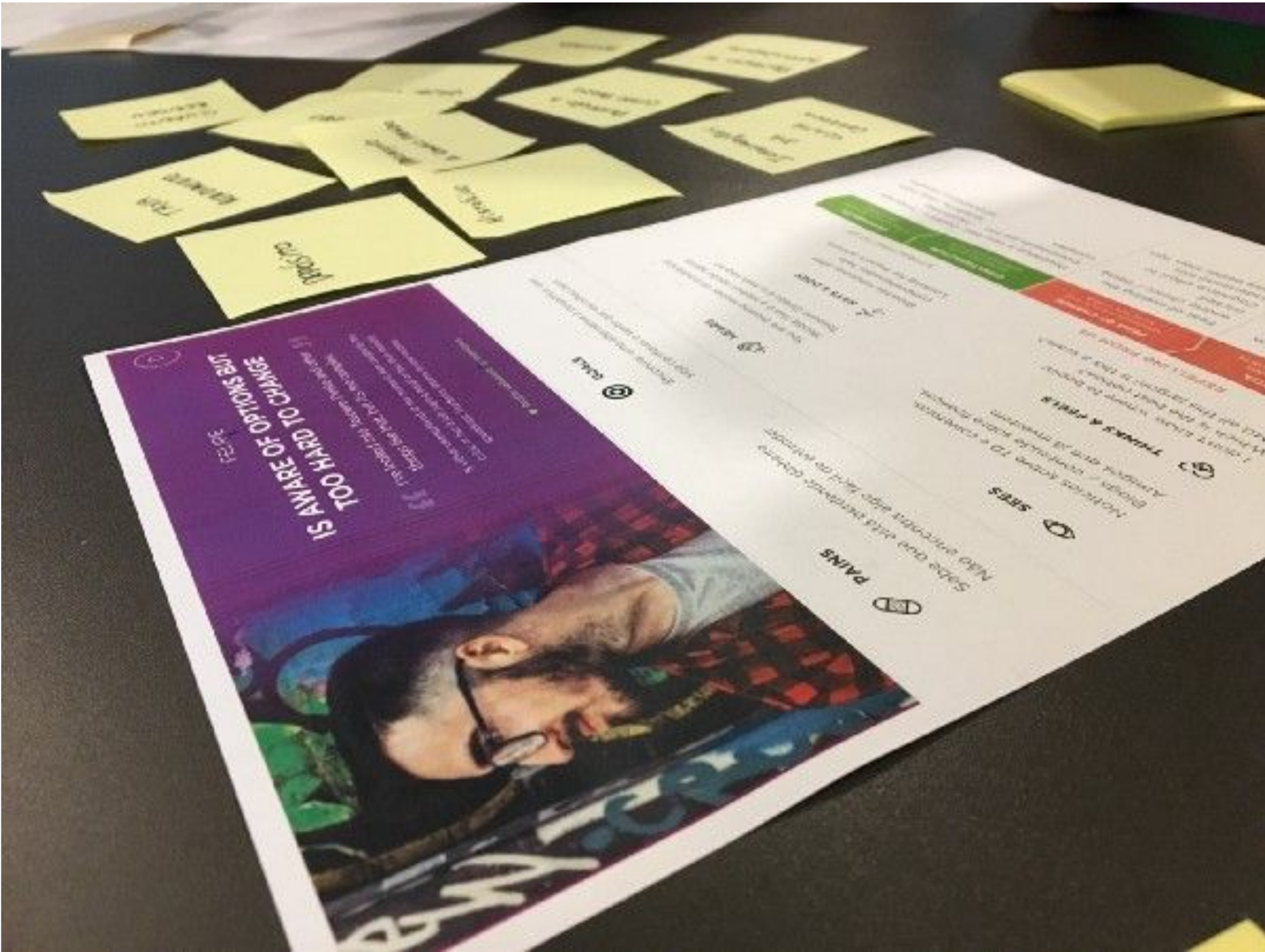
# NuConta



 PESQUISA  
Fale com a gente  
por R\$35,00  
20-40 min em seu tempo

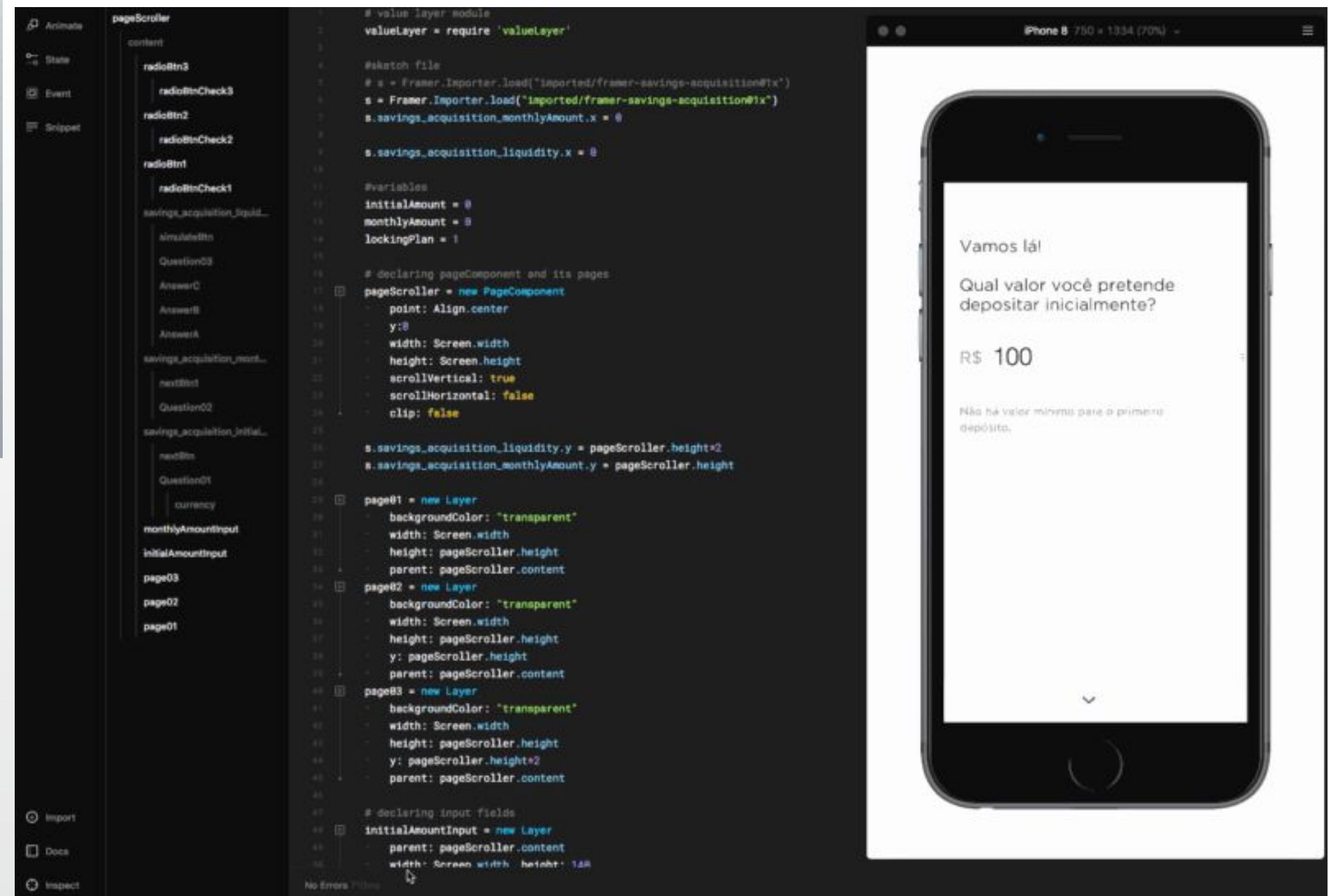


# NuConta



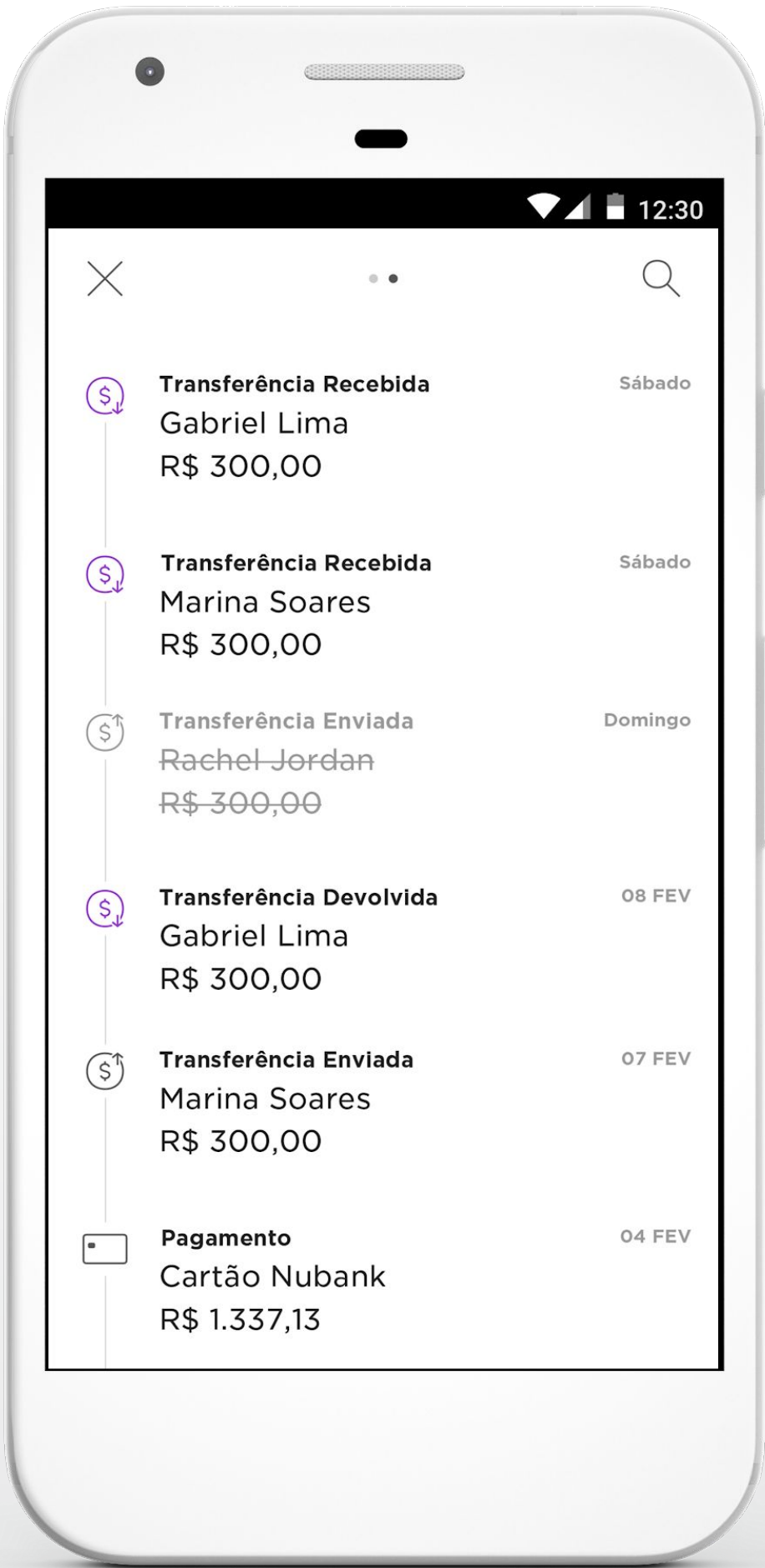
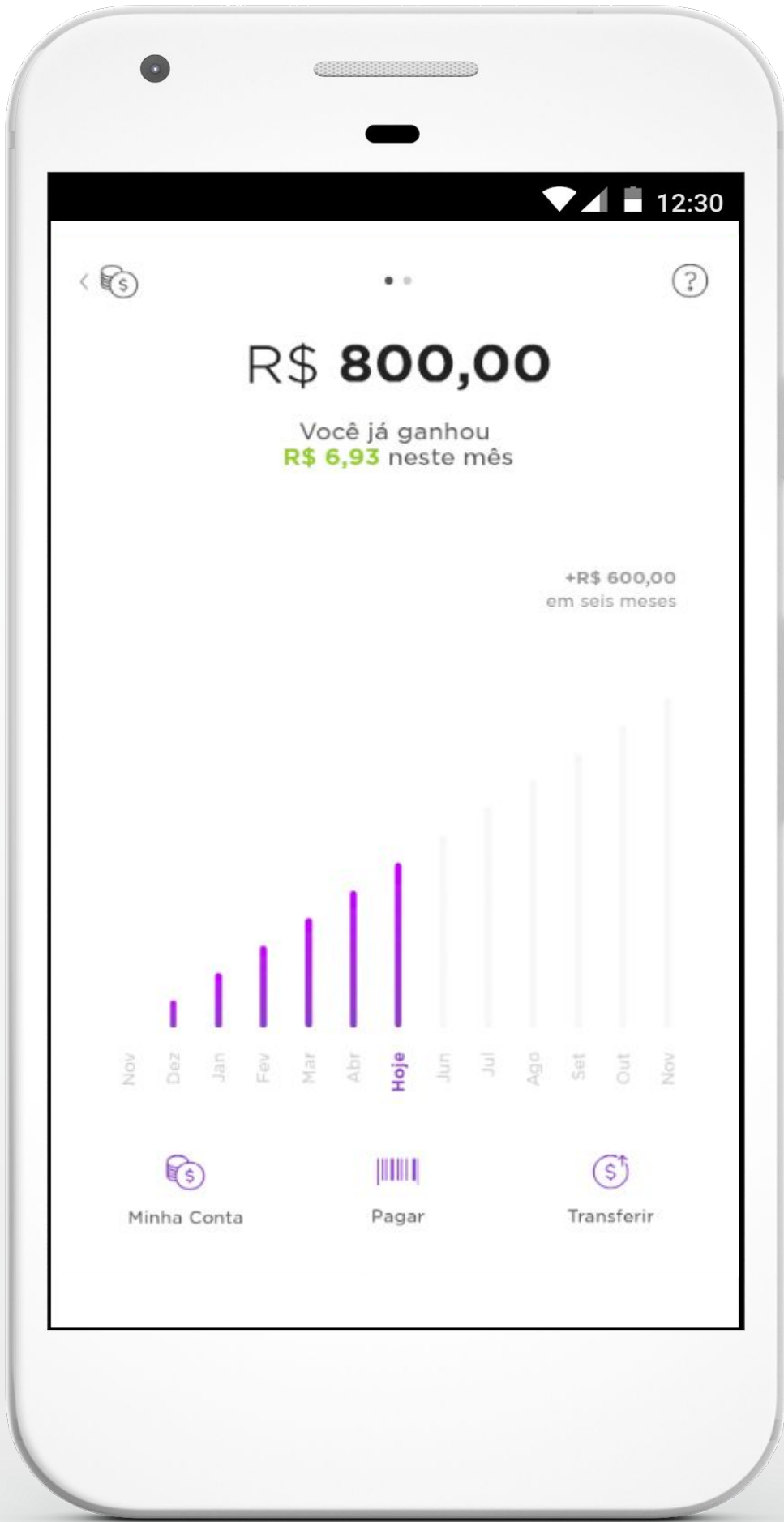


# NuConta





# NuConta





Designing Nubank

Design culture, technology, process, people, and learnings. By the design staff of Nubank.

Follow

Trending



How we designed our Bank Account — NuConta Part II

In Part I, we walked through the process of understanding and defining the problem we were aiming to solve. Here, we are going to dive into...

Lucas Neumann  
Mar 15



How we designed our bank account: NuConta — Part I

In October, Nubank announced our boldest product release since our revolutionary credit card: NuConta, a single account that allows people...

Erick Mazer Yamashita  
Jan 8

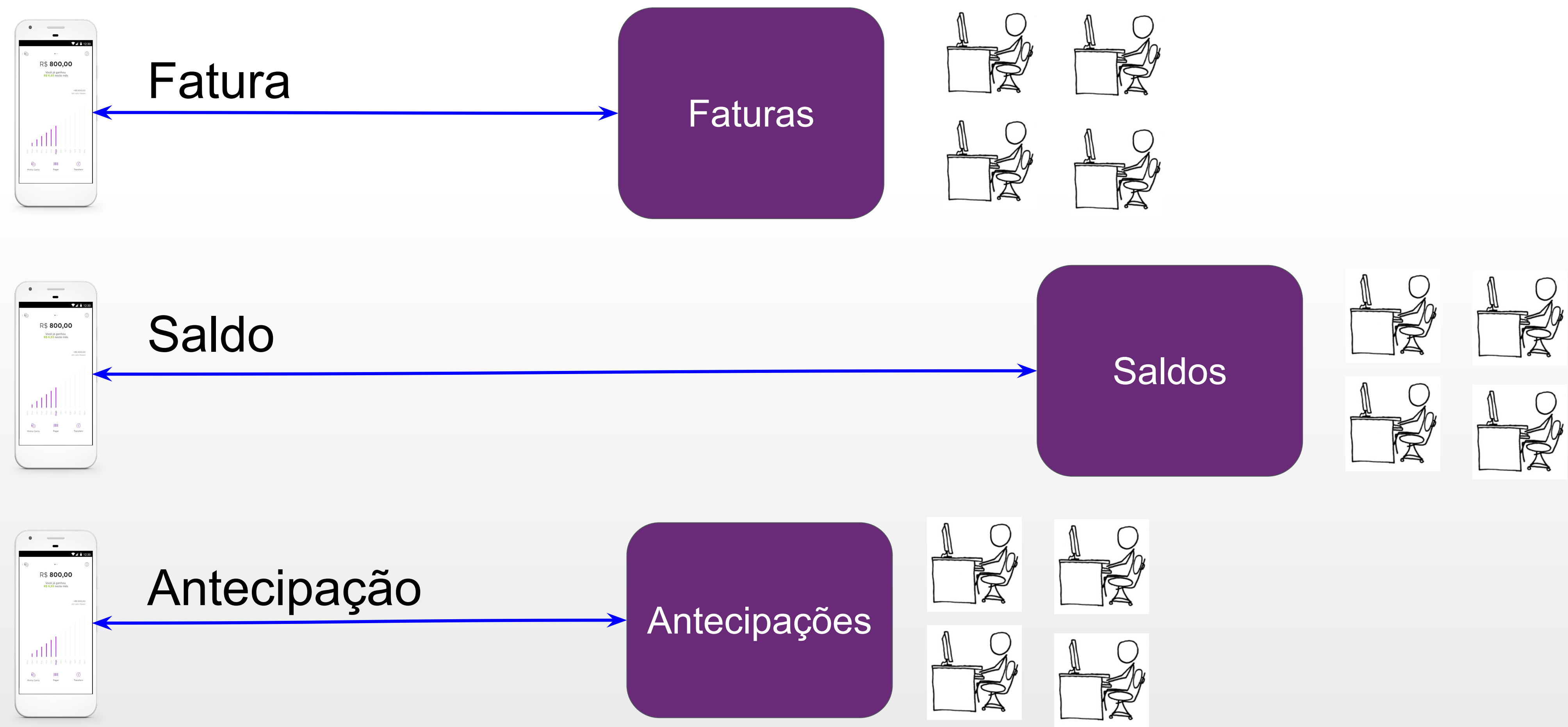


# Microserviços

An aerial night photograph of a city, likely Rio de Janeiro, showing a dense urban area with numerous lights. A prominent river, the Rio de Janeiro, flows through the lower left portion of the image. The city's layout is visible, with various districts and landmarks illuminated. The overall scene is dark, with the city lights providing the primary source of illumination.

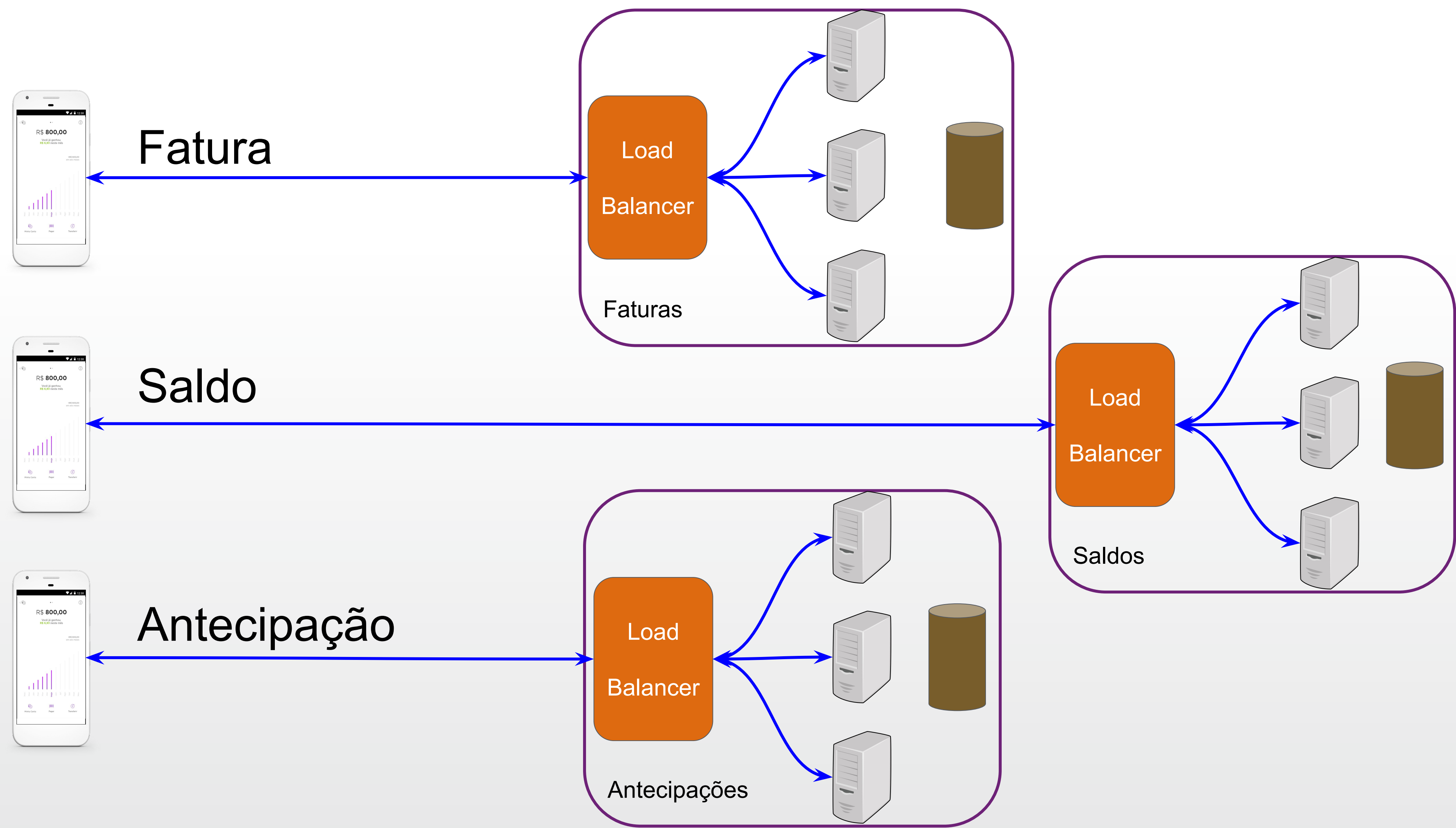


# Microserviços



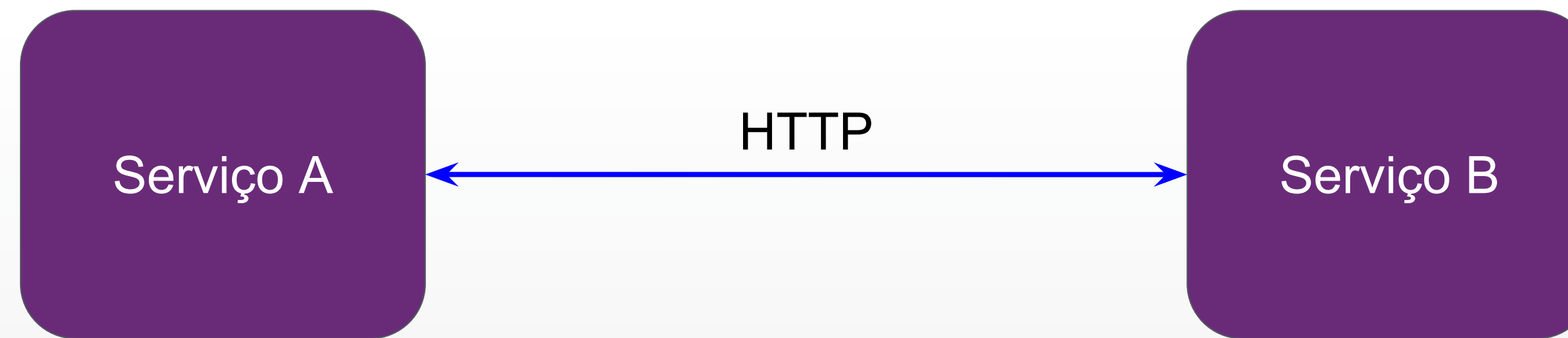


# Microserviços



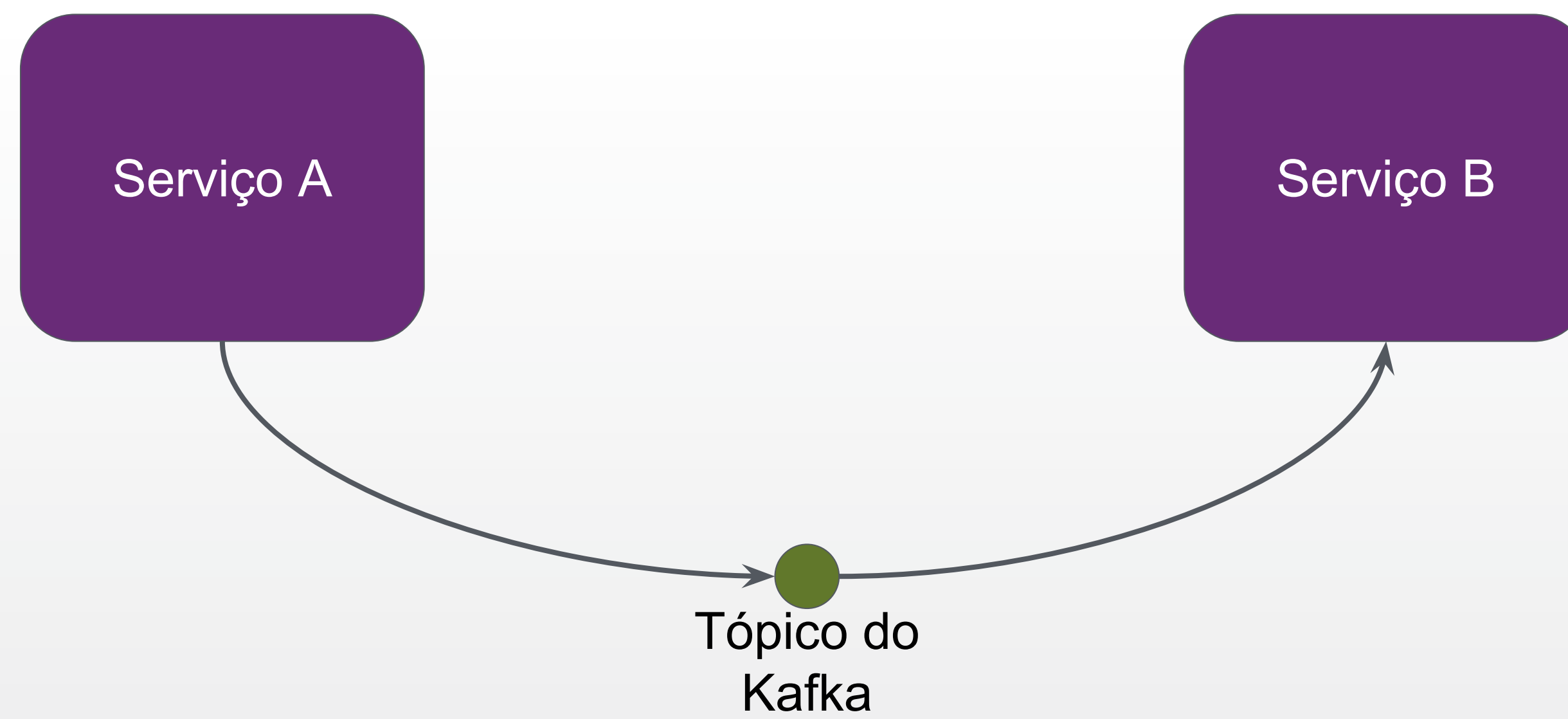


# Microserviços





# Microserviços





A satellite image of the Southeast Brazil region at night, showing a dense network of yellow and white lights representing urban areas and infrastructure. The terrain is visible in shades of brown and green, with a prominent river system winding through the landscape. The coastline is visible on the right side, with the ocean appearing dark blue.

# Transferindo dinheiro entre NuContas

SOUTHEAST BRAZIL REGION FROM SPACE



# Transferindo dinheiro entre NuContas



Pedido de envio  
Envio



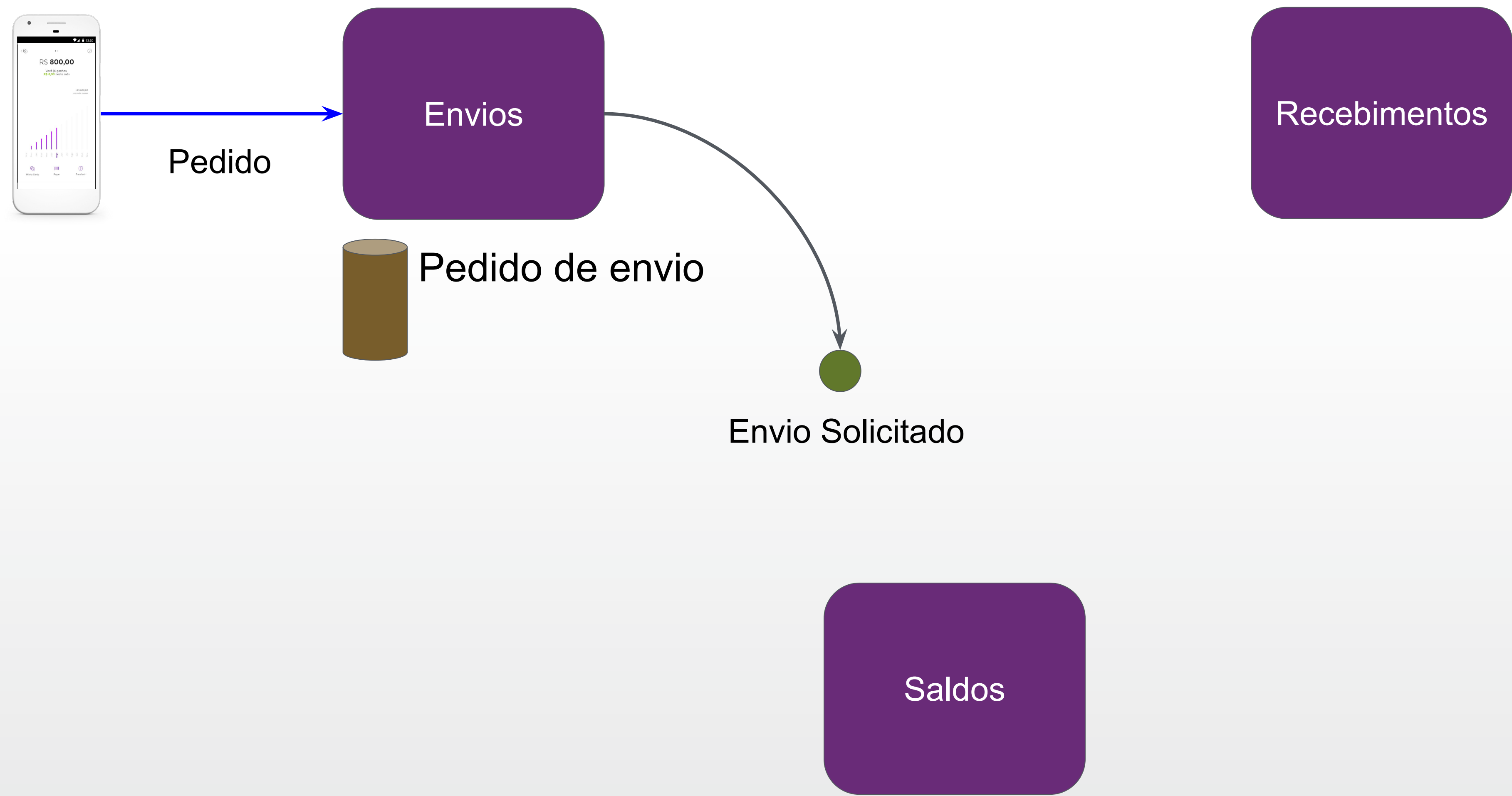
Recebimento



Depósito  
Liquidação

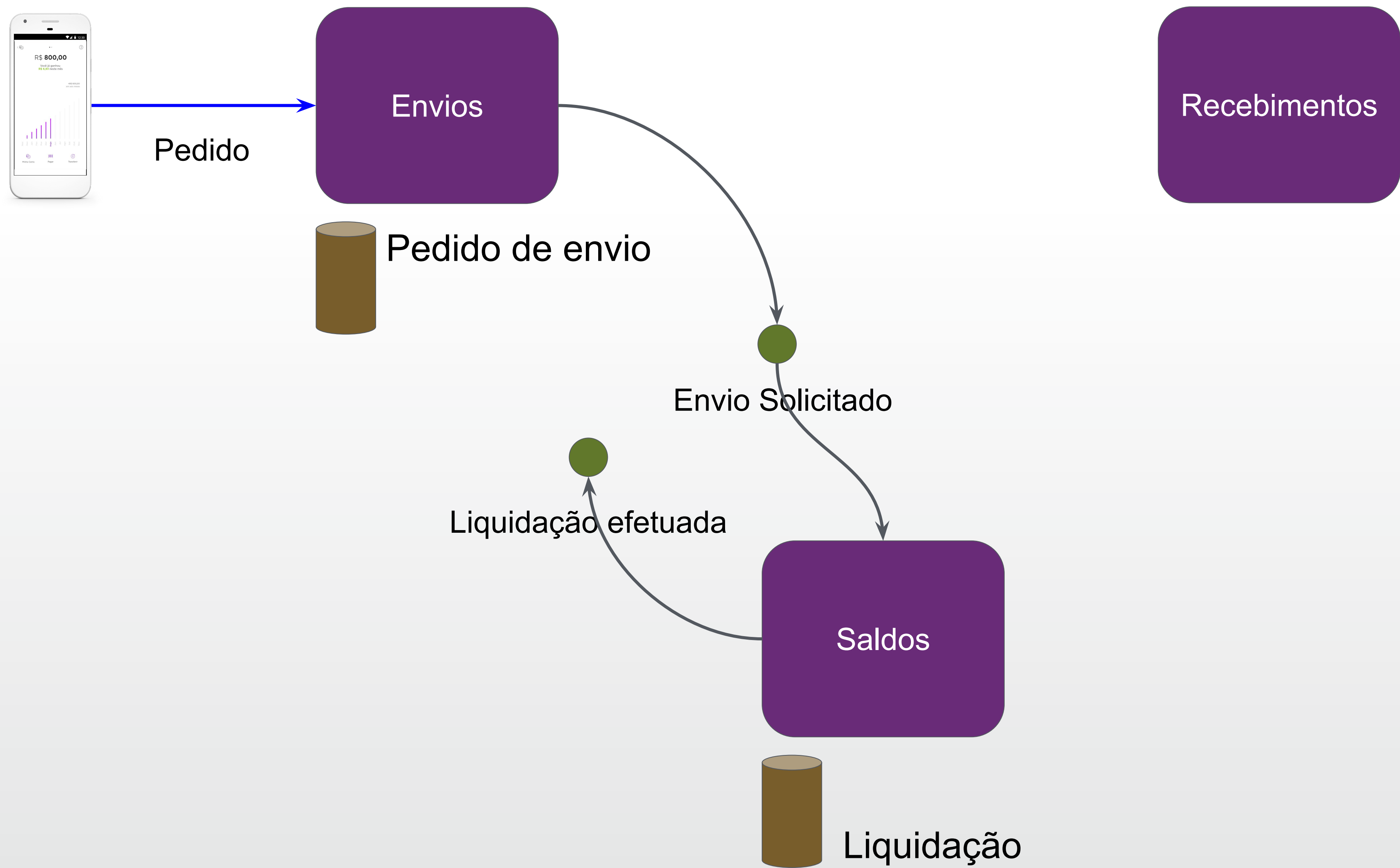


# Transferindo dinheiro entre NuContas



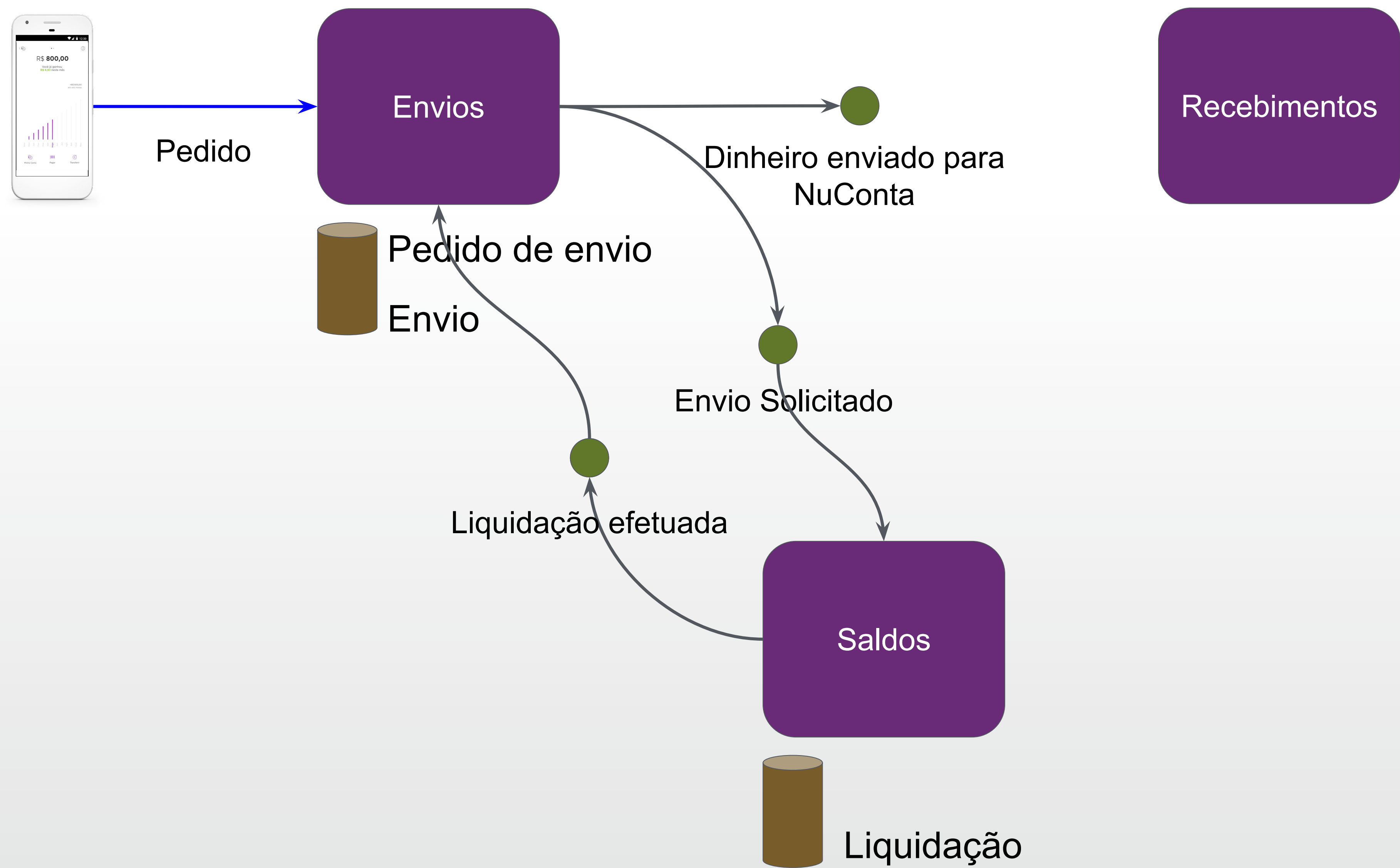


# Transferindo dinheiro entre NuContas



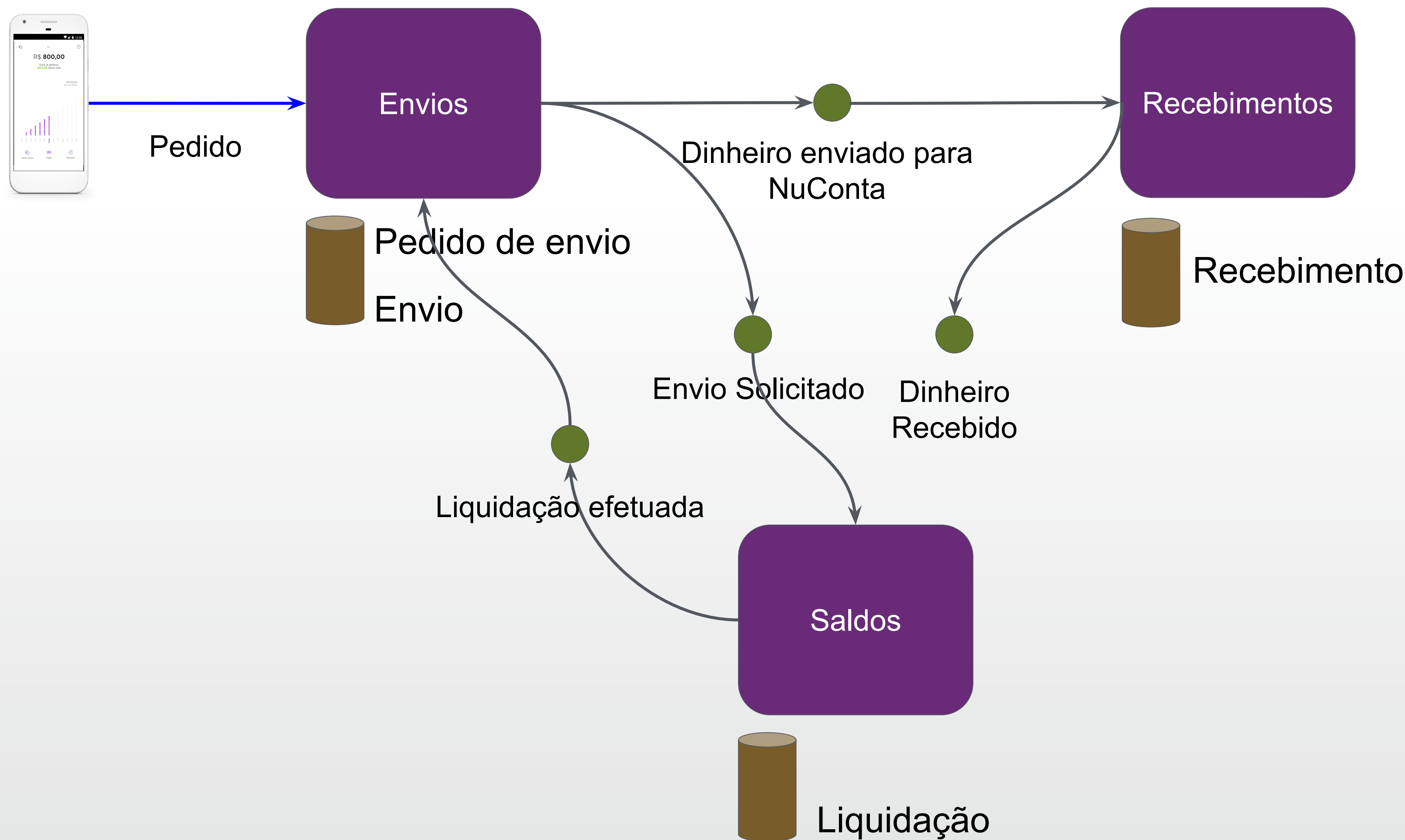


# Transferindo dinheiro entre NuContas



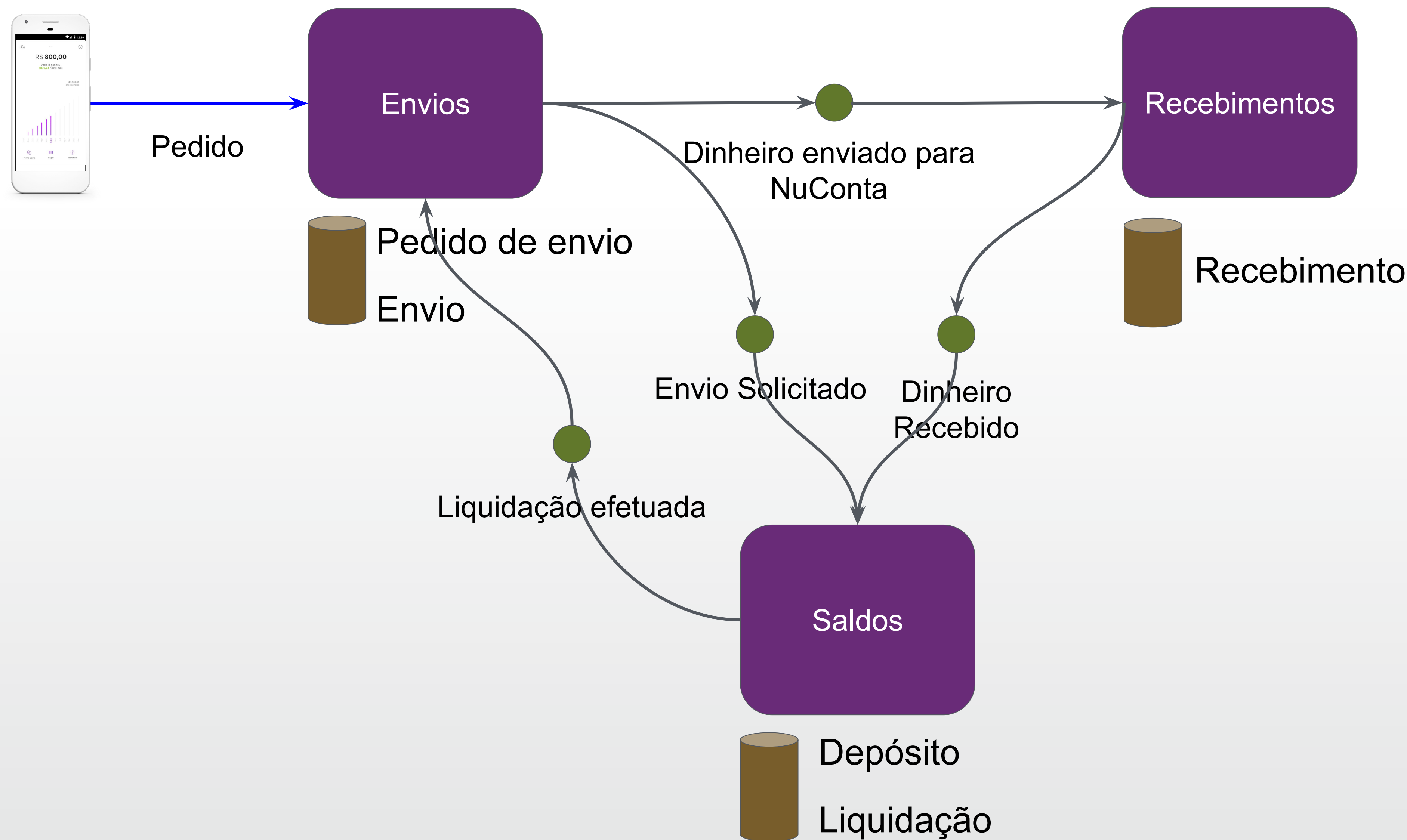


# Transferindo dinheiro entre NuContas



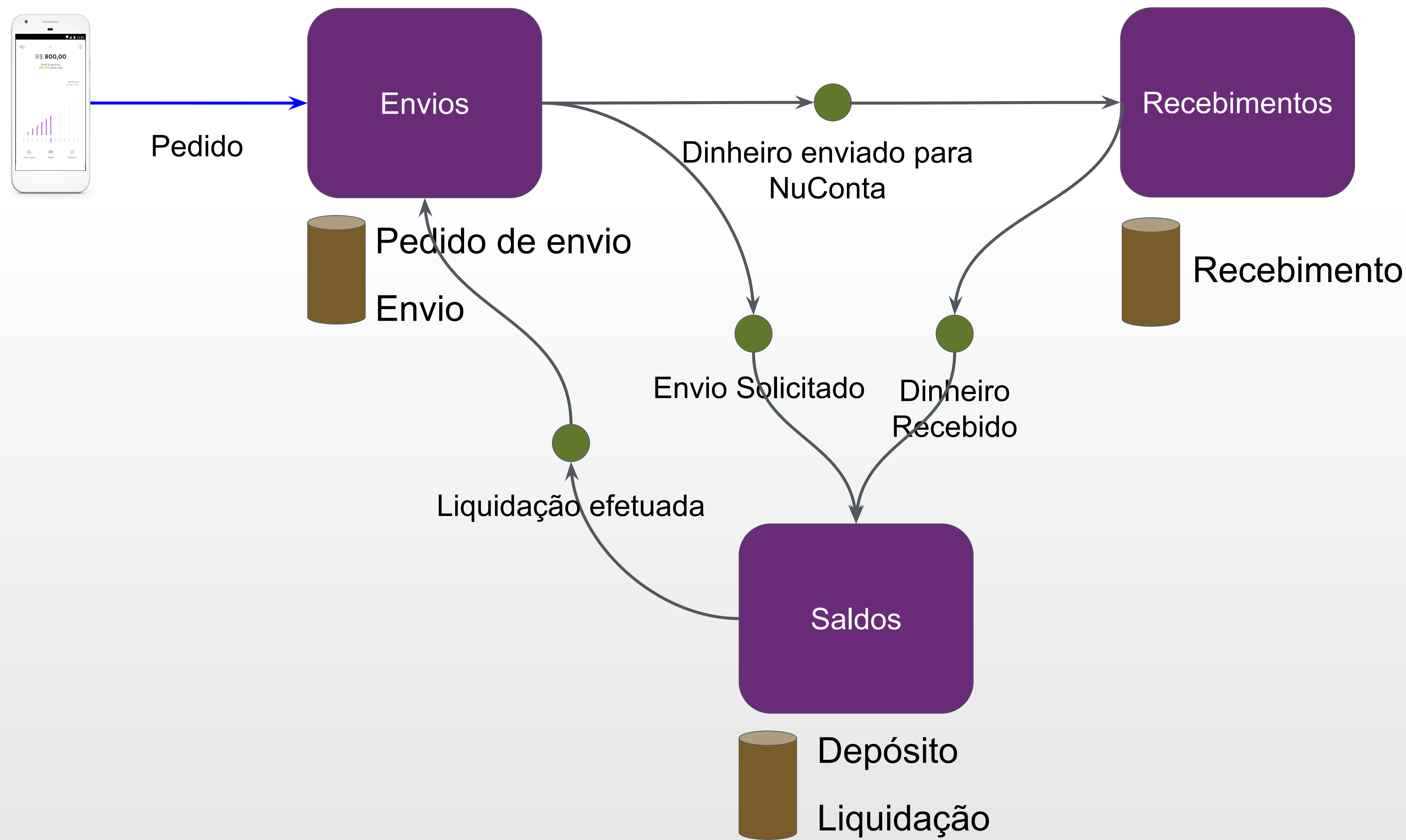


# Transferindo dinheiro entre NuContas





# Características interessantes desse fluxo



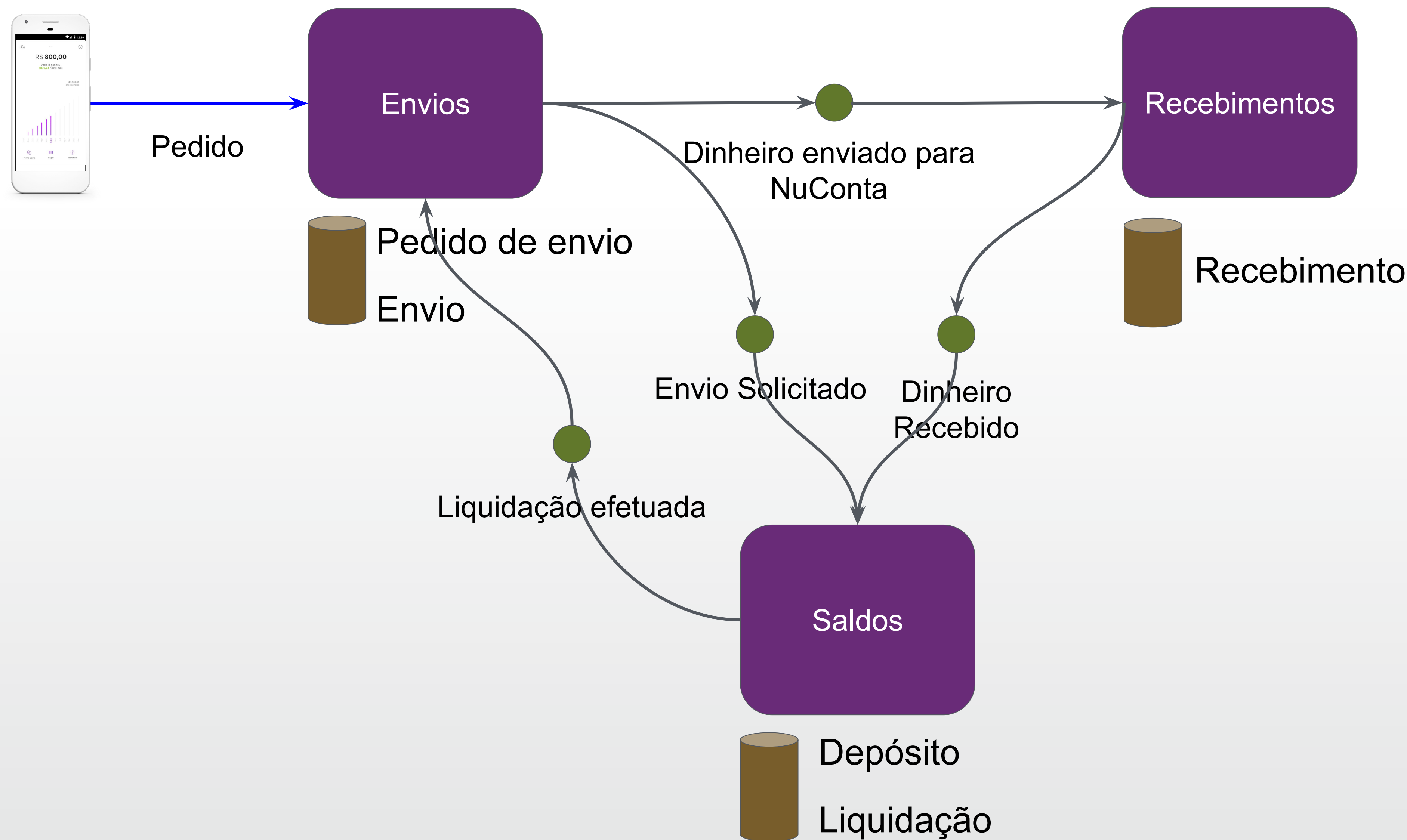


# Event Sourcing



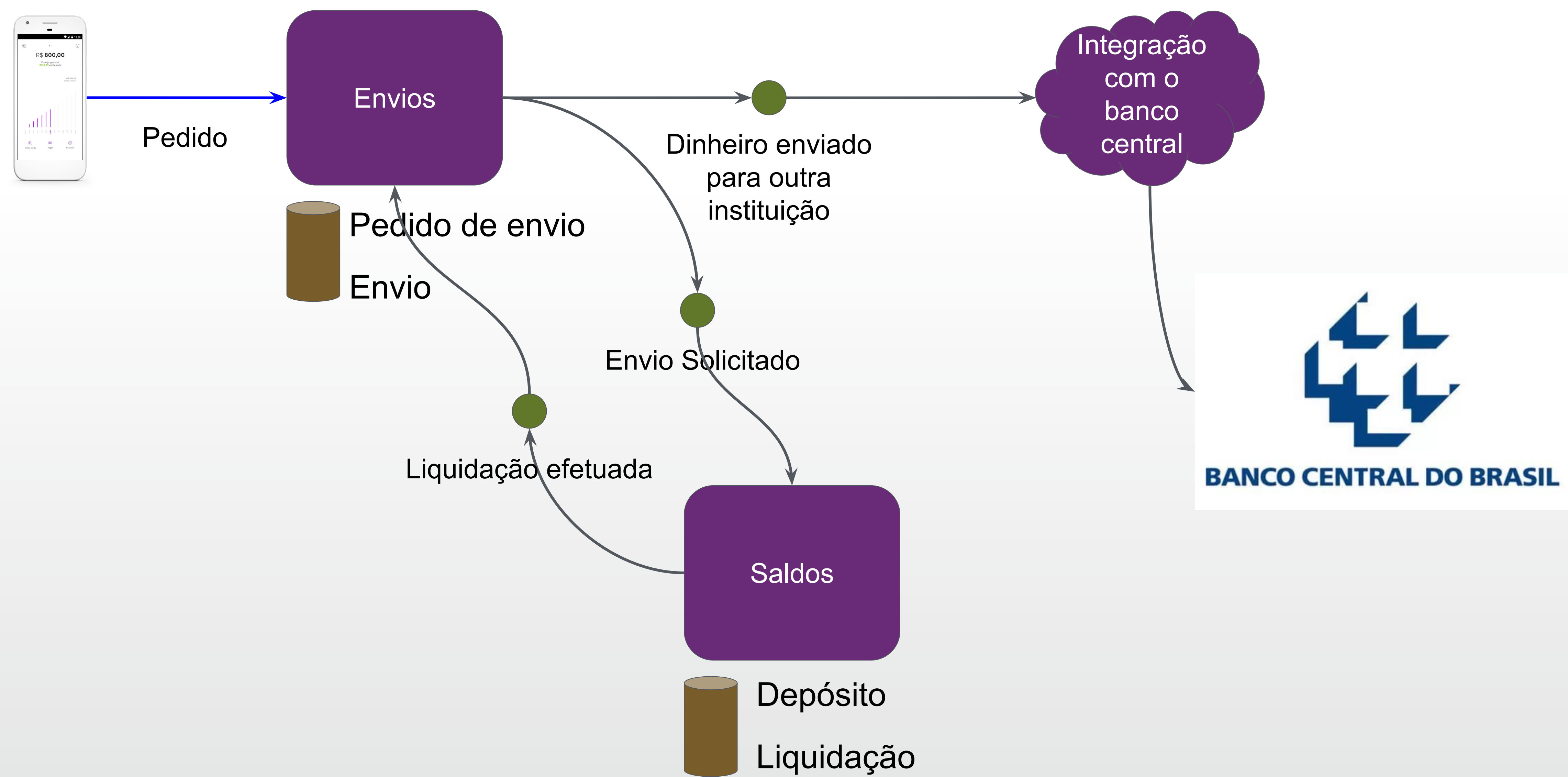


# E se o cliente quiser enviar transferências para outra instituição financeira?

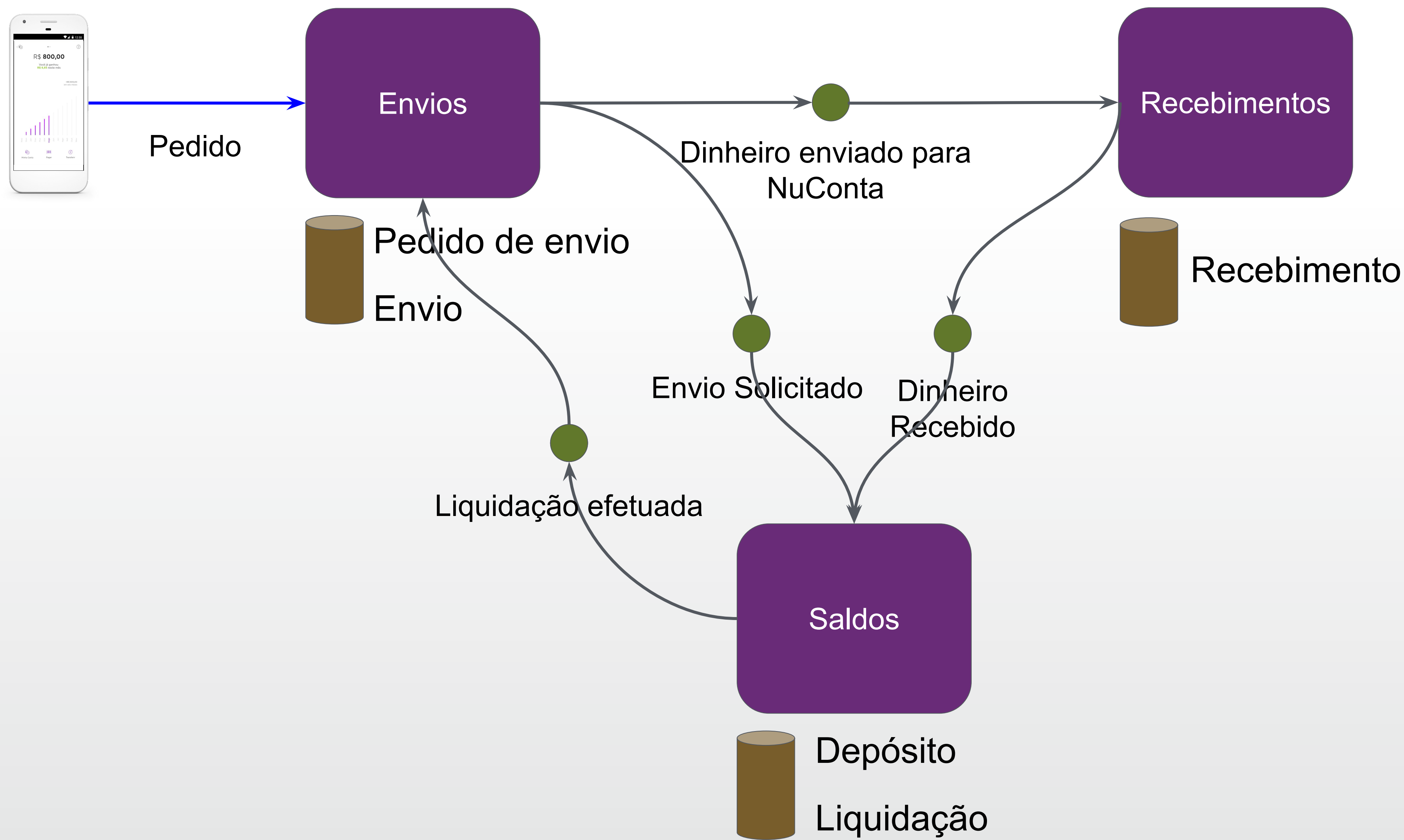




# E se o cliente quiser enviar transferências para outra instituição financeira?

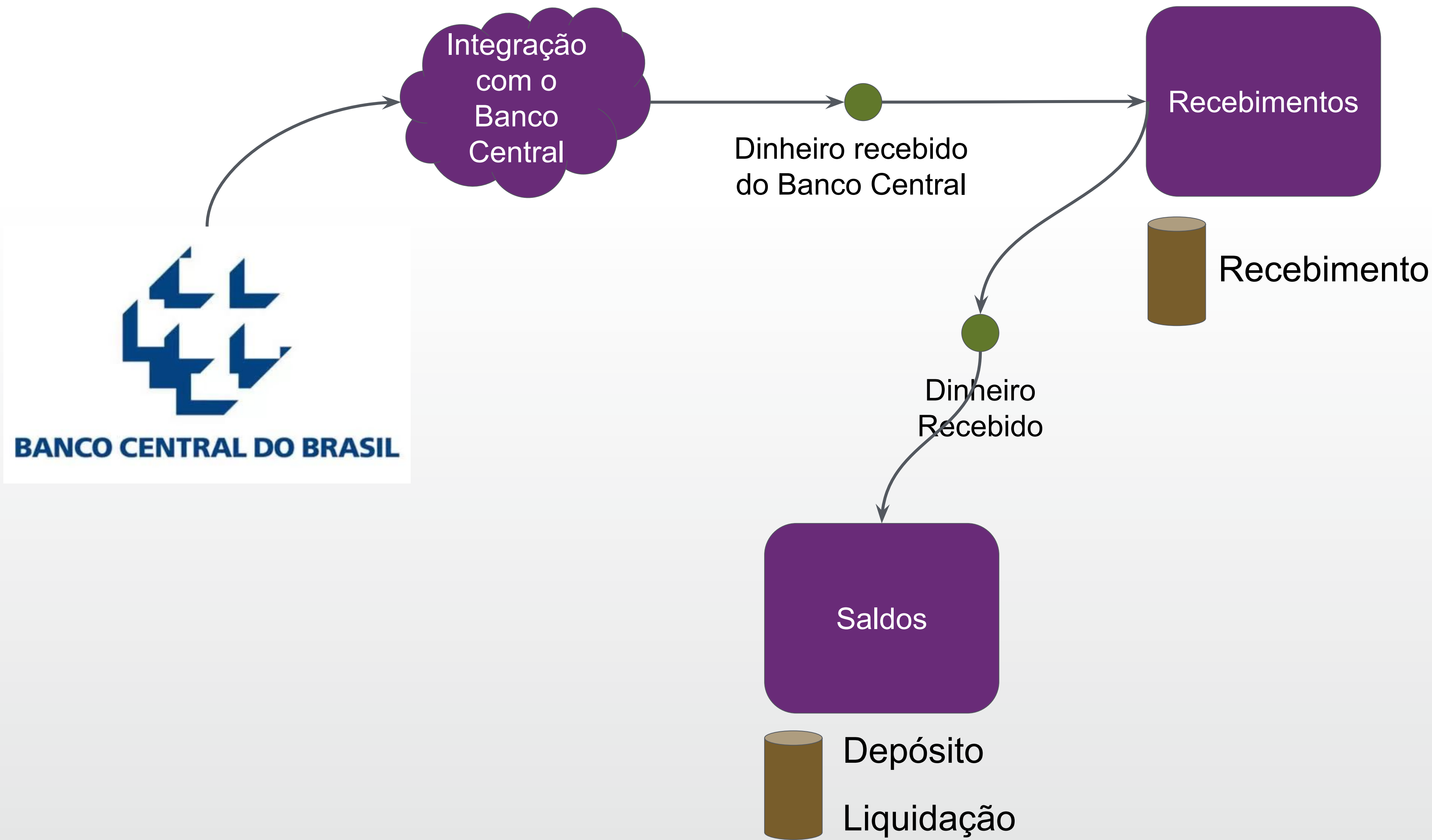


# E se o cliente receber transferências de outra instituição financeira?

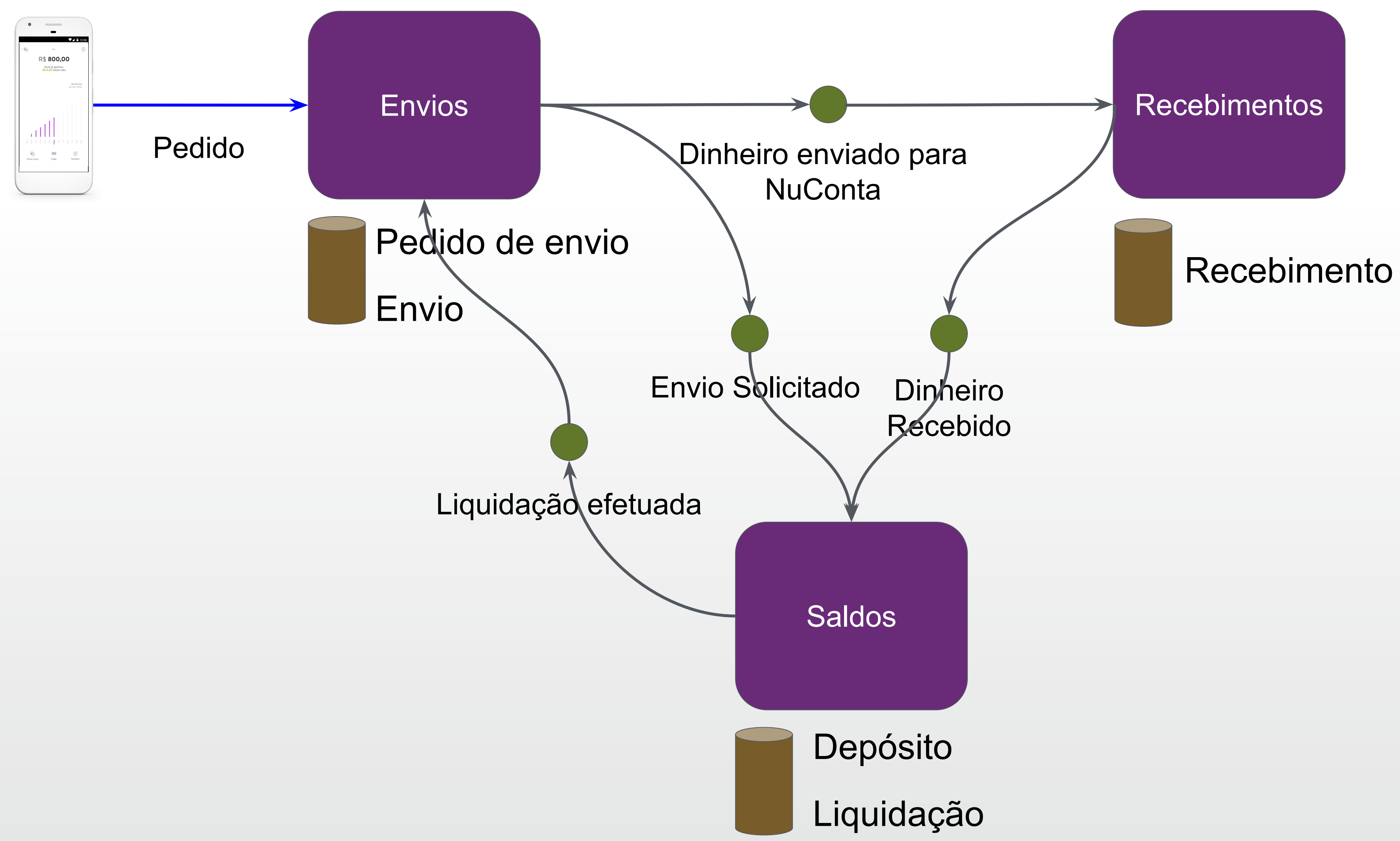




E se o cliente receber transferências de outra instituição financeira?

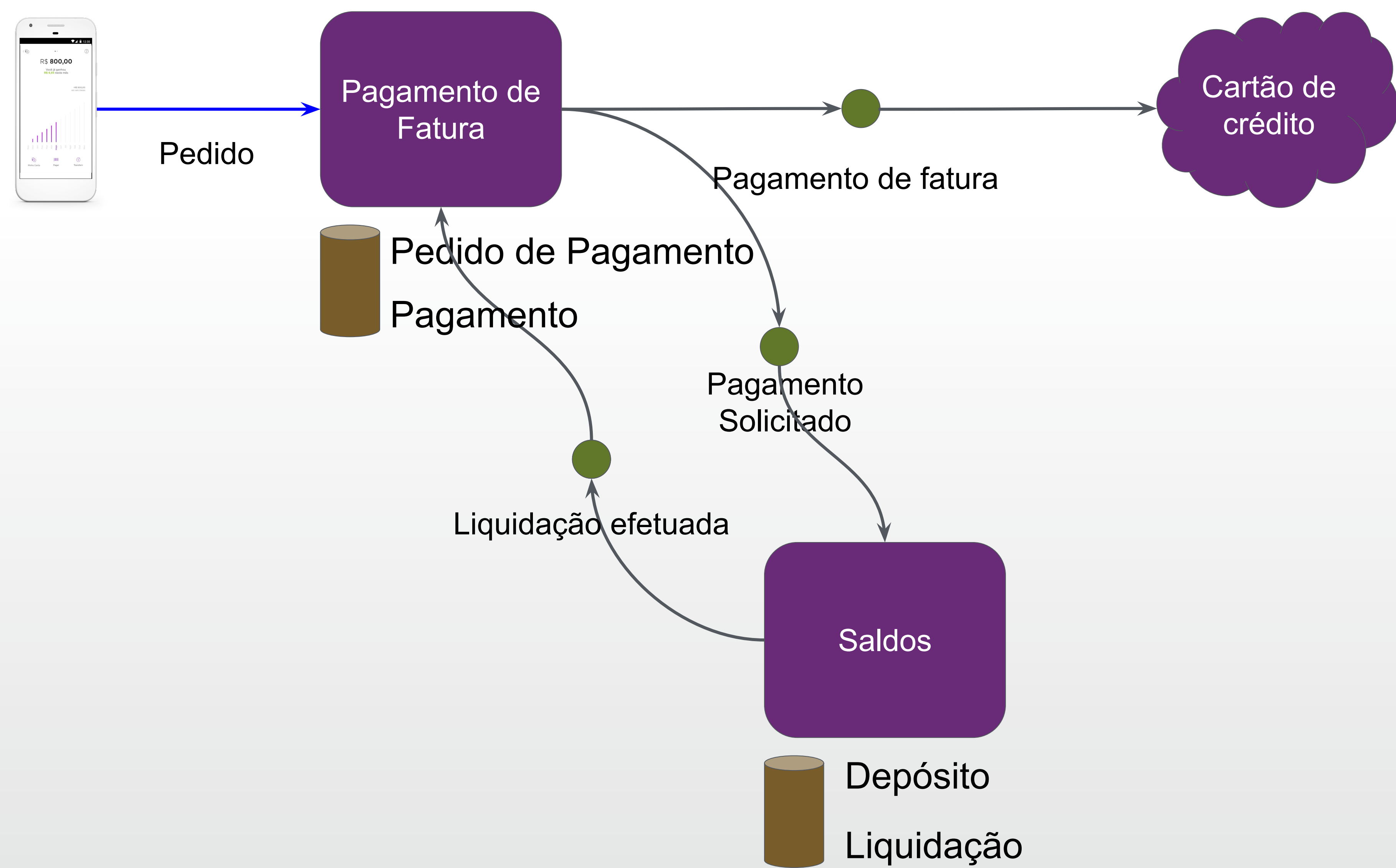


# E se o cliente quiser pagar a fatura do cartão de crédito?

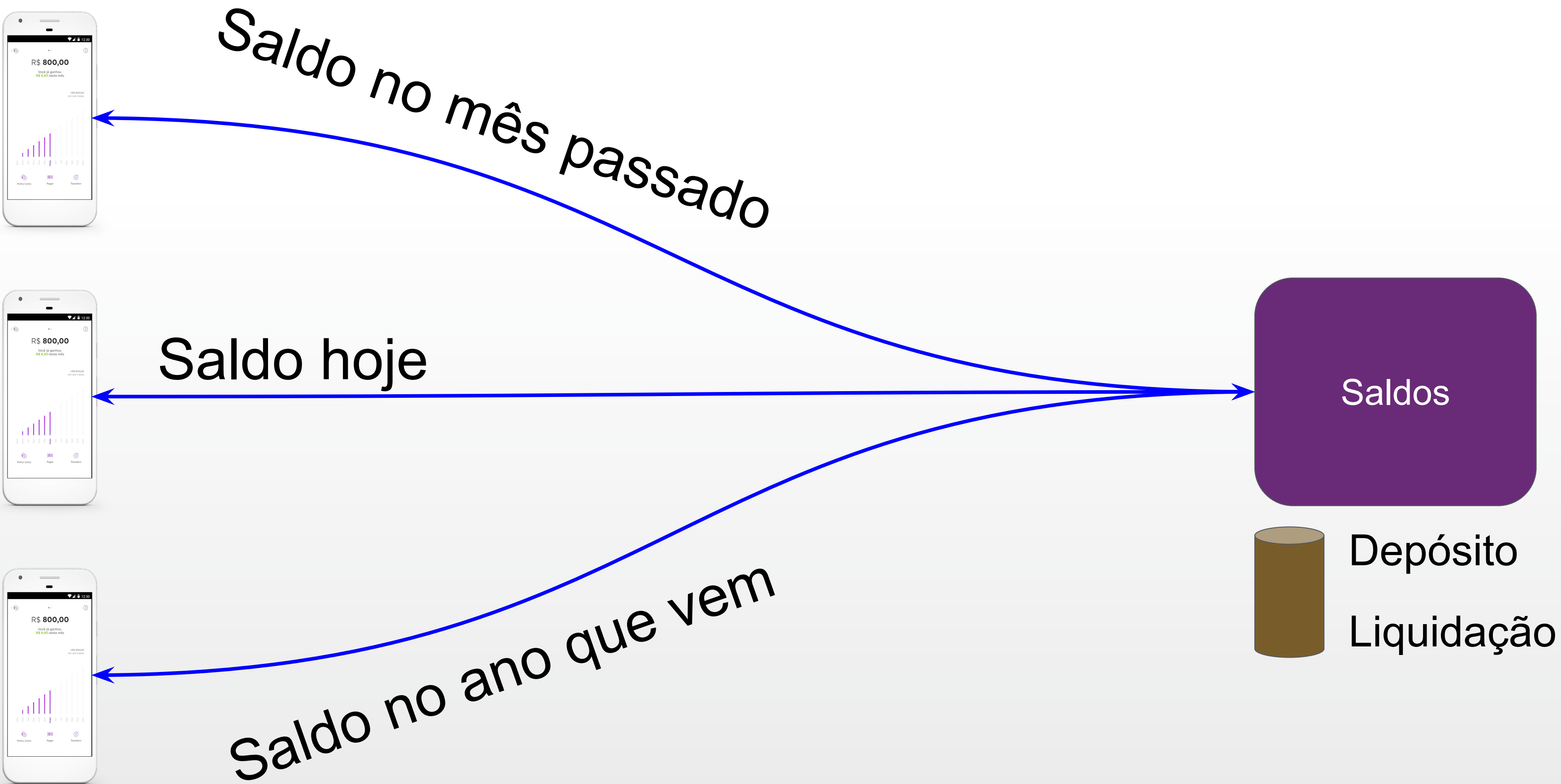




# E se o cliente quiser pagar a fatura do cartão de crédito?

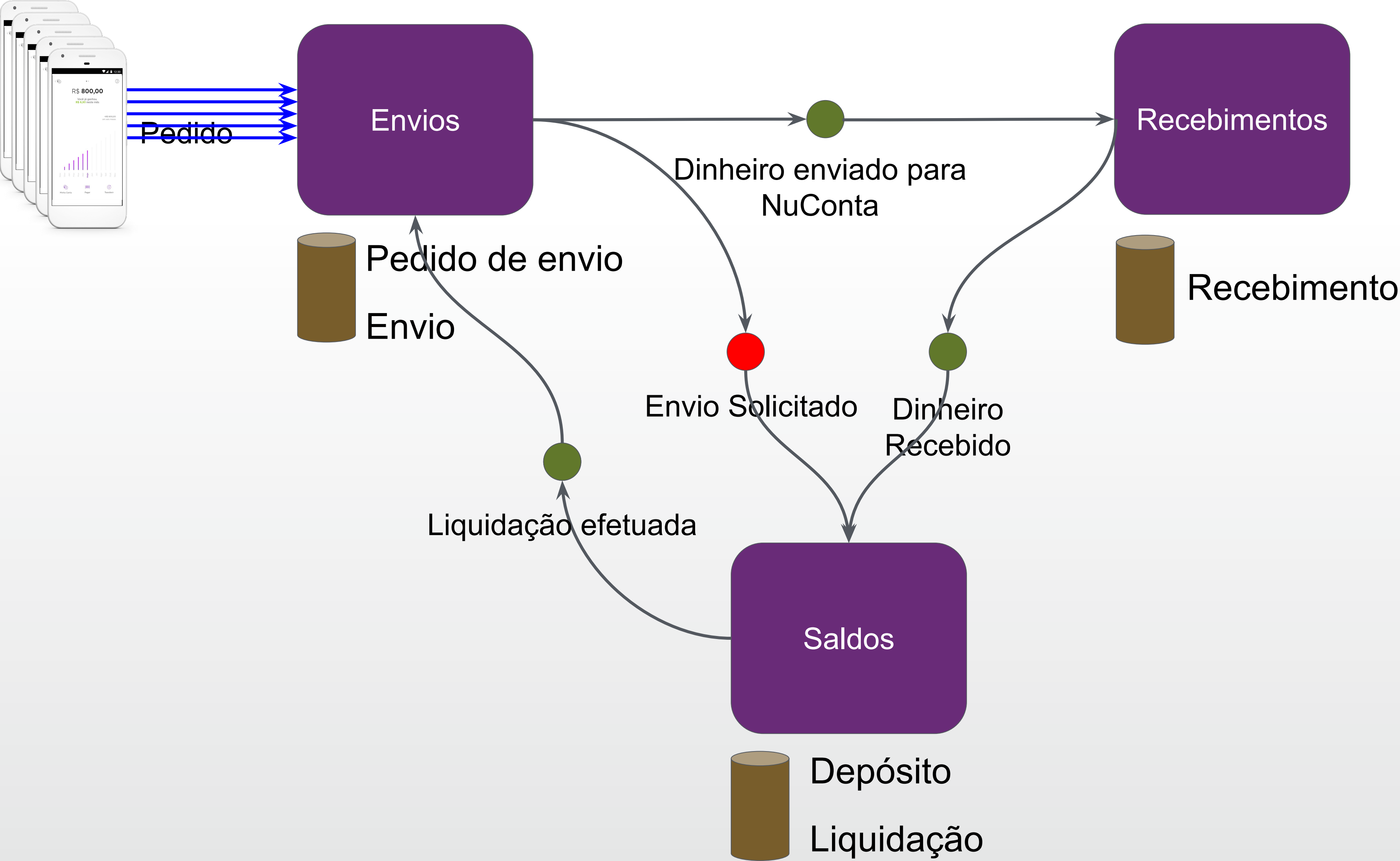


# Event Sourcing





# Event Sourcing



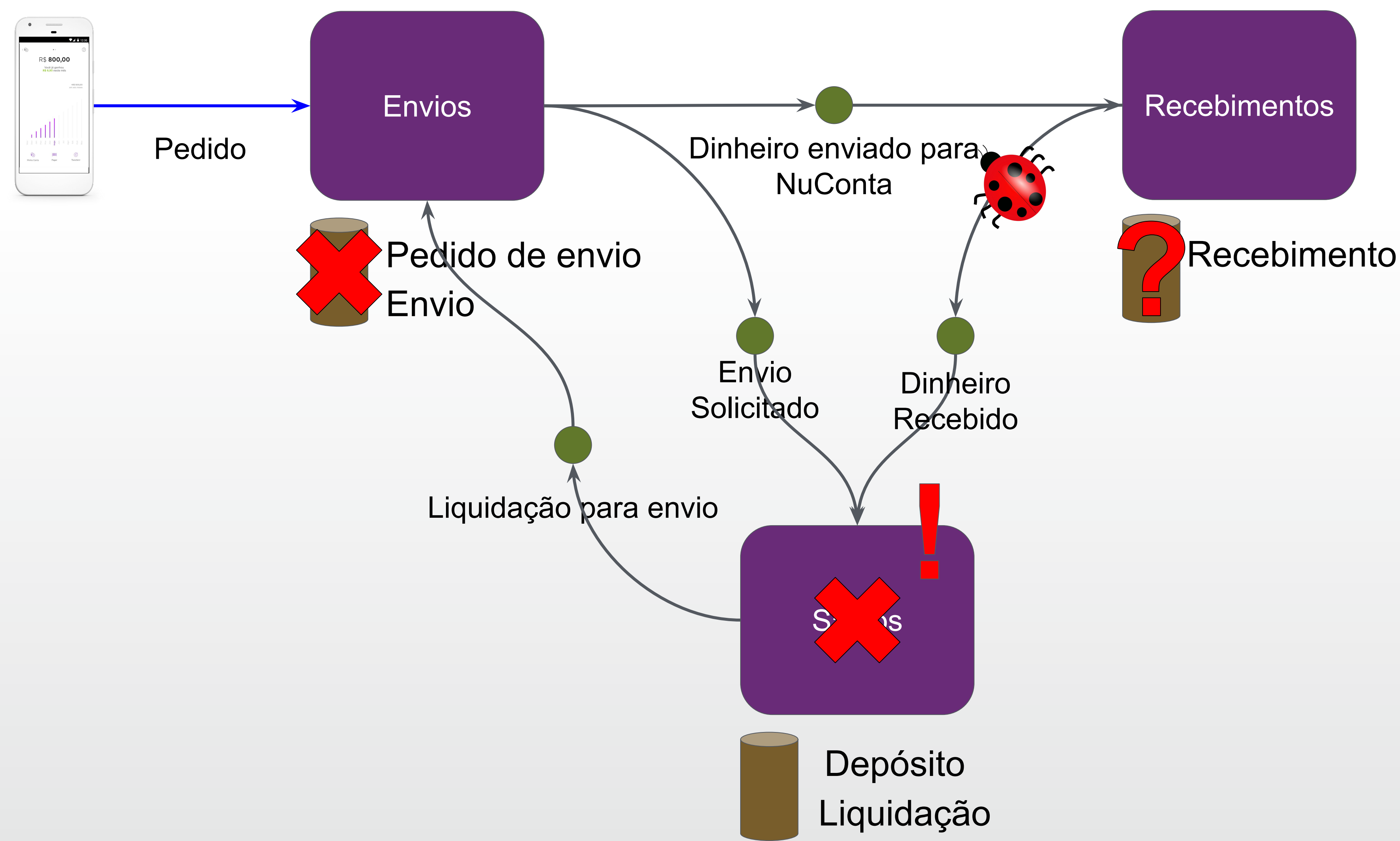


An aerial night photograph of a city, likely Rio de Janeiro, showing the bay and surrounding hills. The city lights are visible, and the water reflects the lights. The text "Consistência em sistemas distribuídos" is overlaid in white.

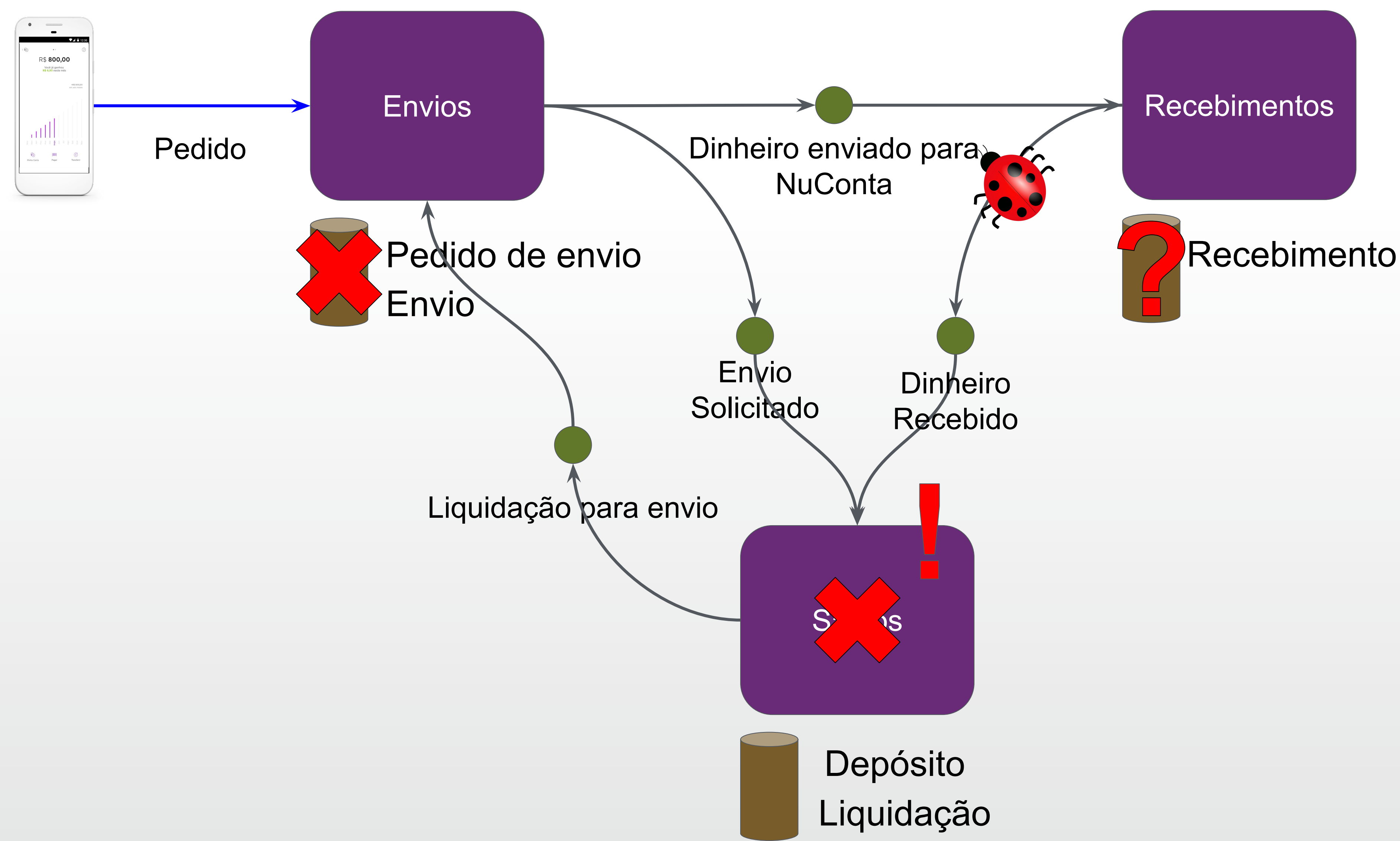
# Consistência em sistemas distribuídos



# O que pode dar errado?



# O que pode dar errado?

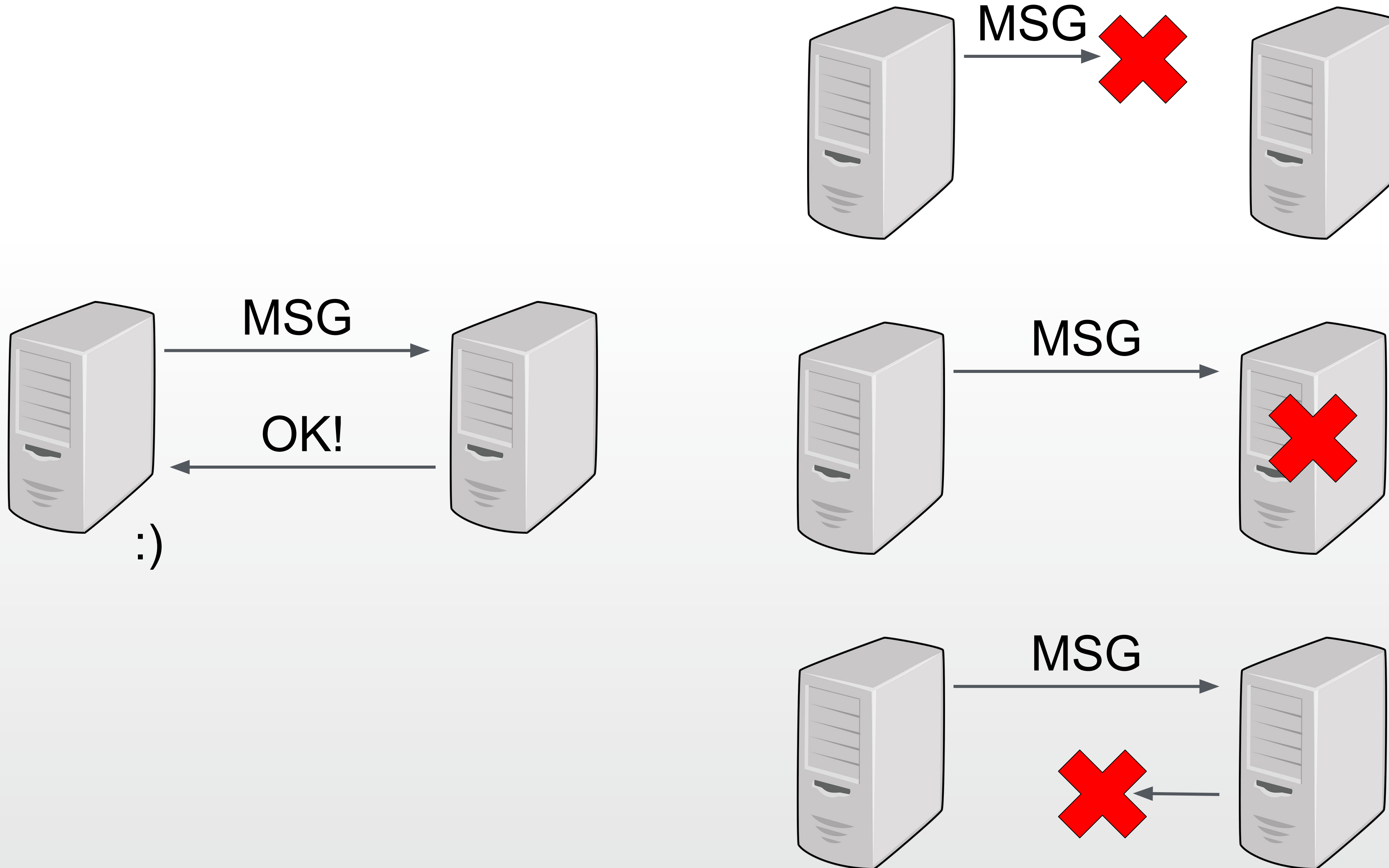




**Primeiro requisito: Processamento *at-least-once***

Todo evento publicado é recebido e processado completamente pelos consumidores **pelo menos uma vez**

## Primeiro requisito: Processamento *at-least-once*





## Primeiro requisito: Processamento *at-least-once*

- Mensagens são persistidas e replicadas no Kafka cluster, podendo ser consumidas a qualquer momento
- Próxima mensagem da fila será entregue de novo até que consumidor confirme que completou seu processamento
- Consumidor só deve confirmar o processamento depois que completar todos os efeitos colaterais

## Segundo requisito: Idempotência

$$f(f(x)) = f(x)$$

- Uma operação é *idempotente* se aplicá-la várias vezes é equivalente a aplicá-la uma vez
- Exemplos:
  - Multiplicação por 0:  $7*0 = 7*0*0 = 7*0*0*0 \dots = 0$
  - `DELETE FROM users WHERE users.id = 186`

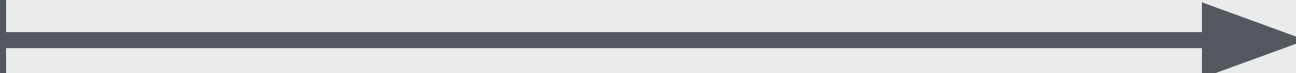
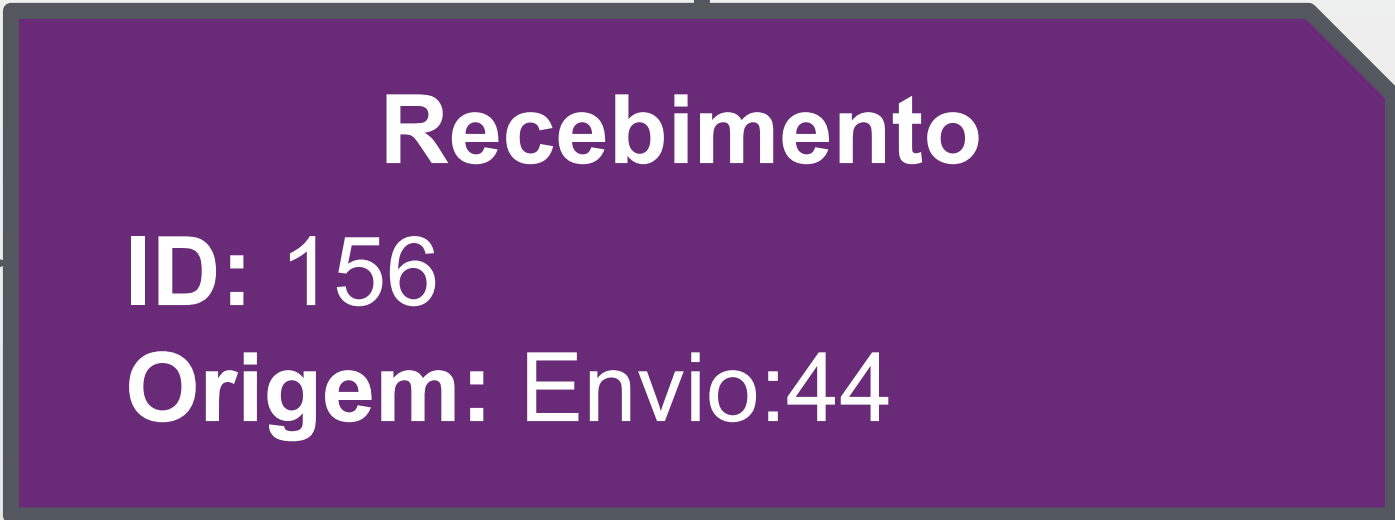
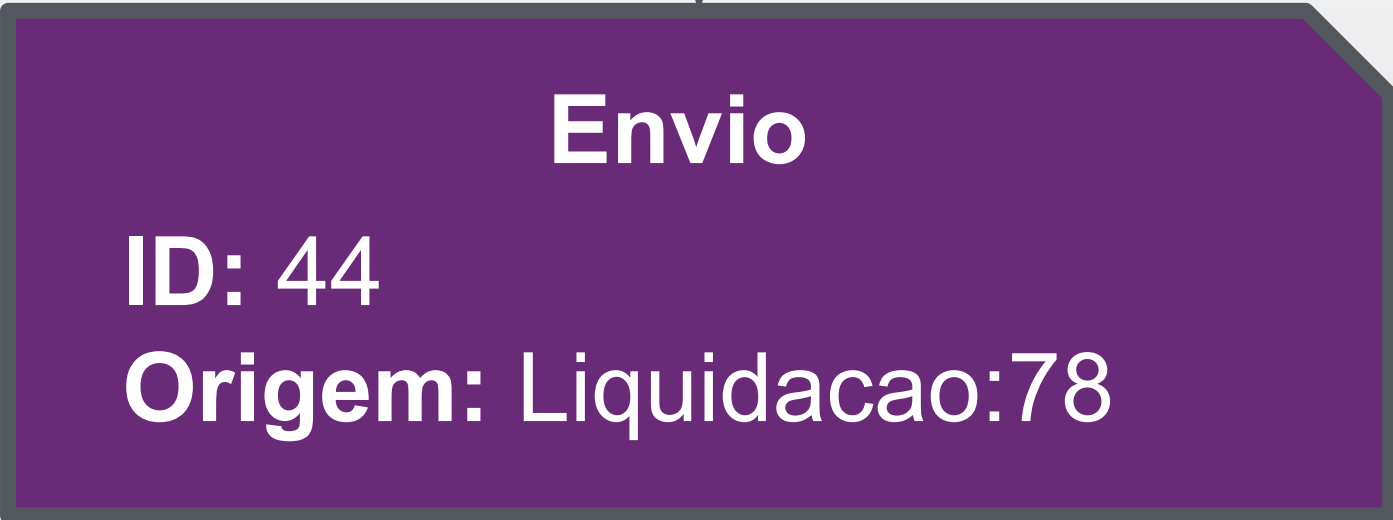
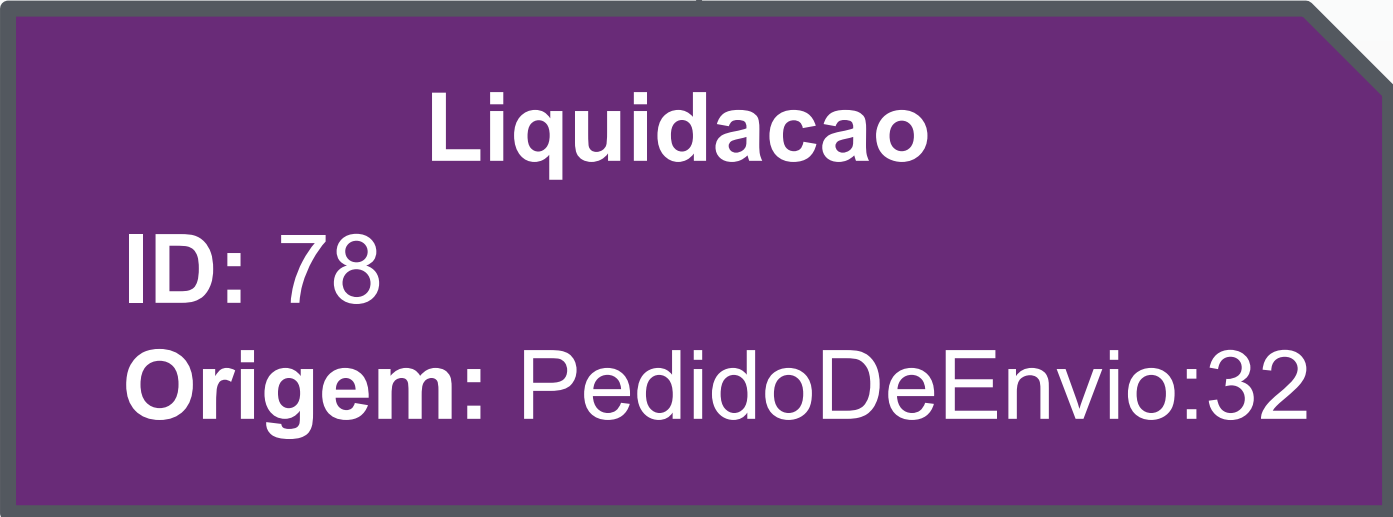


## Segundo requisito: Idempotência

- Cada evento em nossa arquitetura tem um UUID aleatório
- Cada evento derivado usa o UUID do evento anterior (origem) como chave única (**chave de idempotência**)
- Serviços garantem consistência interna: banco de dados local valida a chave única
- Resultado: Criar um evento a partir de outro é uma operação idempotente

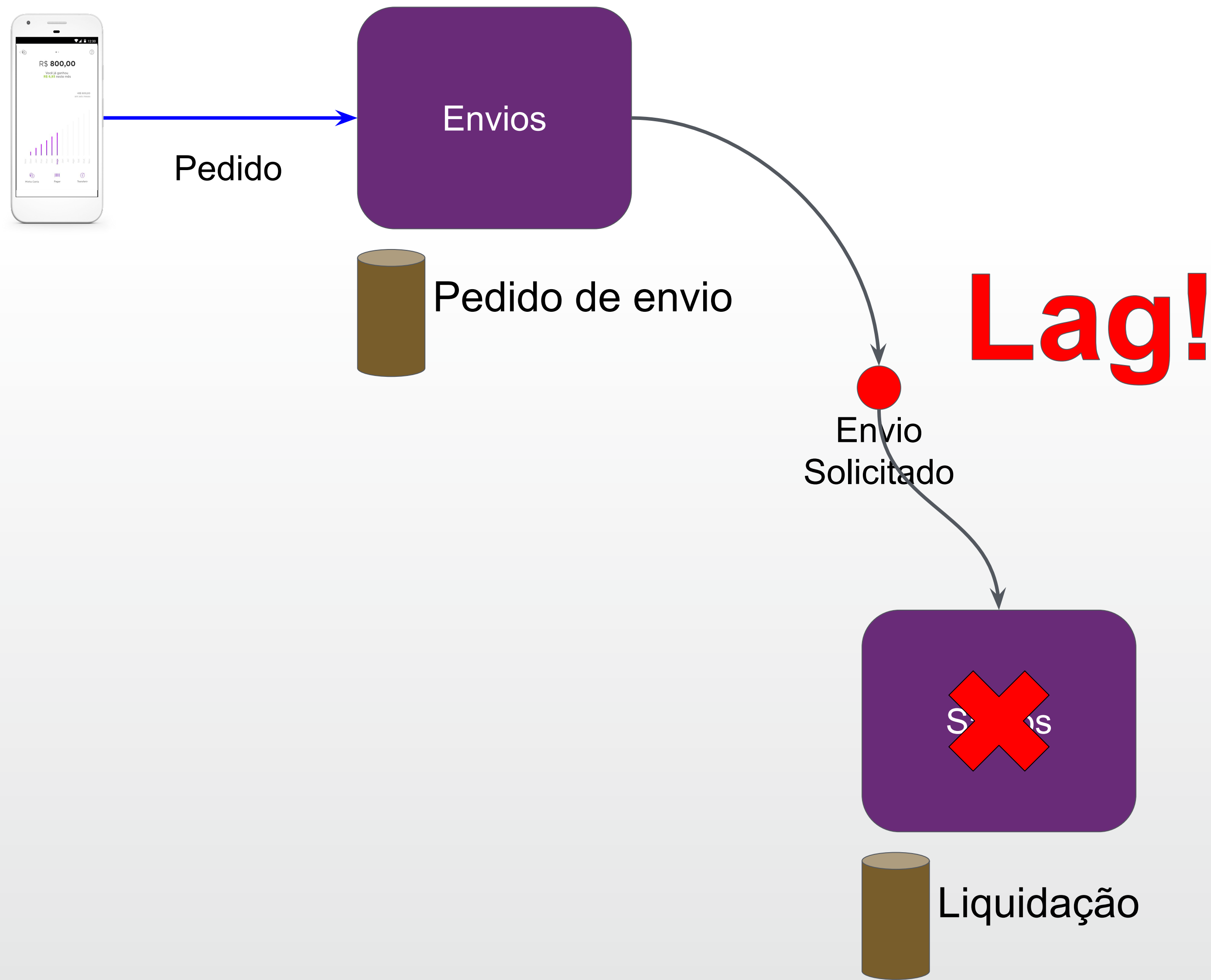


# Segundo requisito: Idempotência

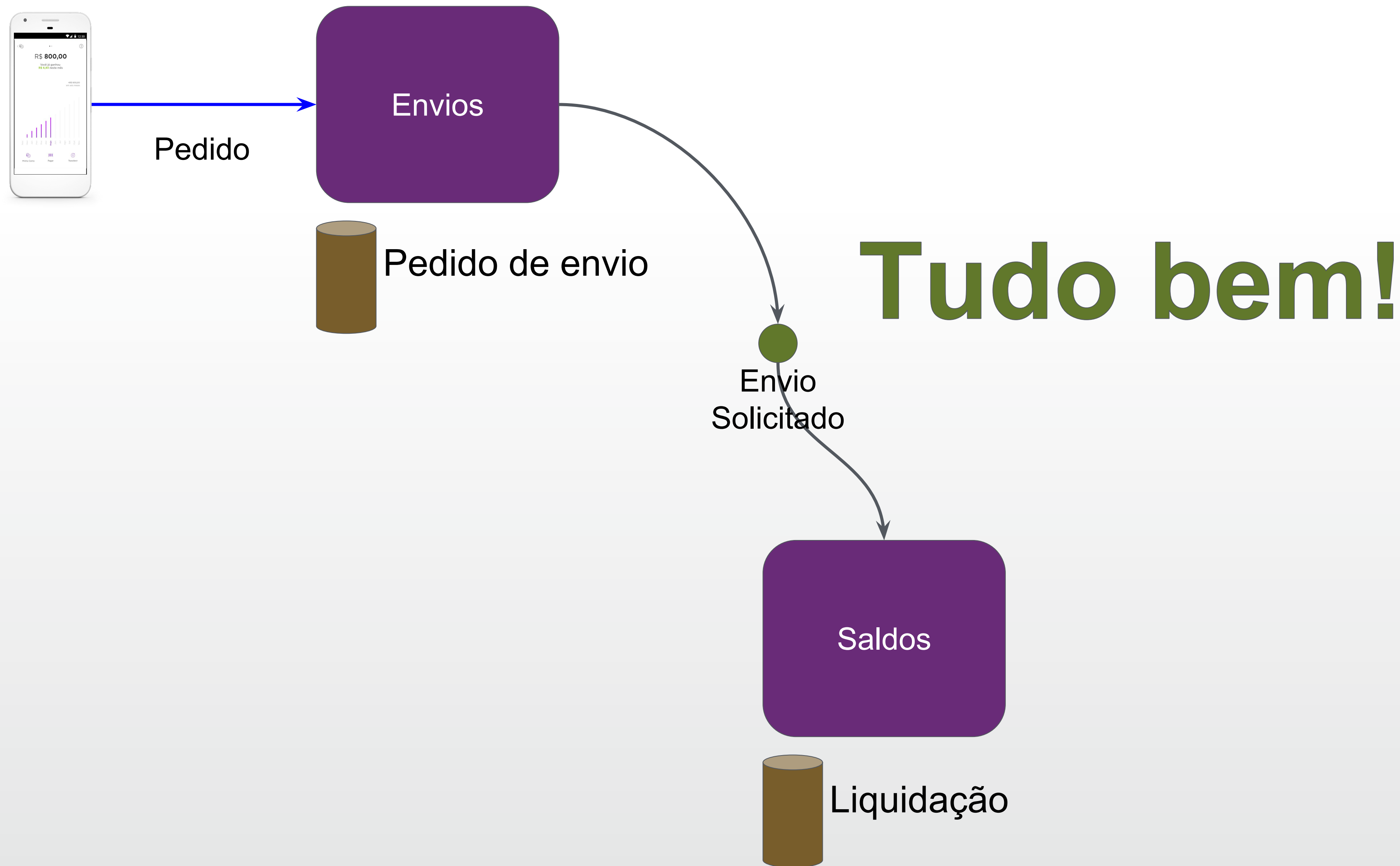




# E quando um serviço cair?

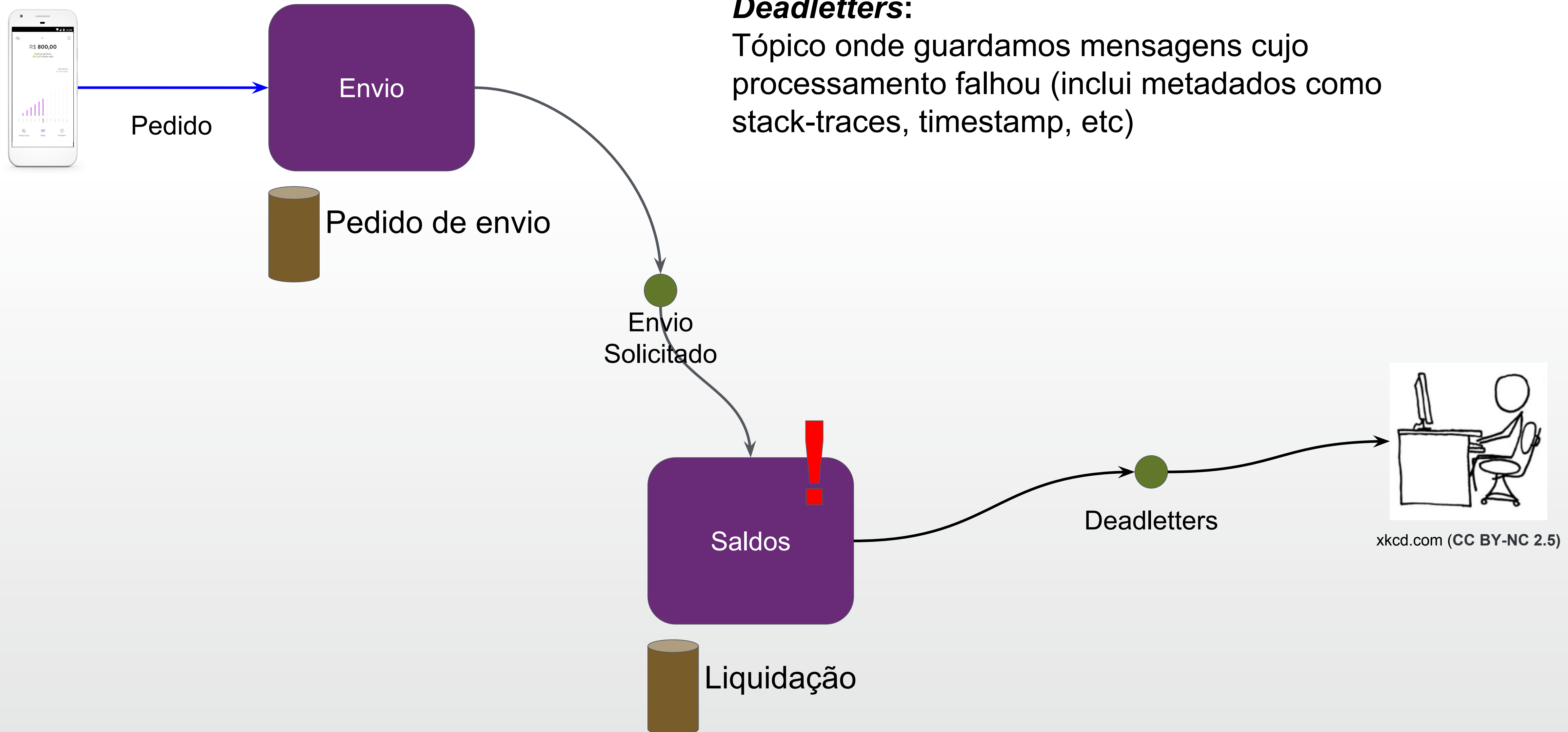


# E quando um serviço cair?



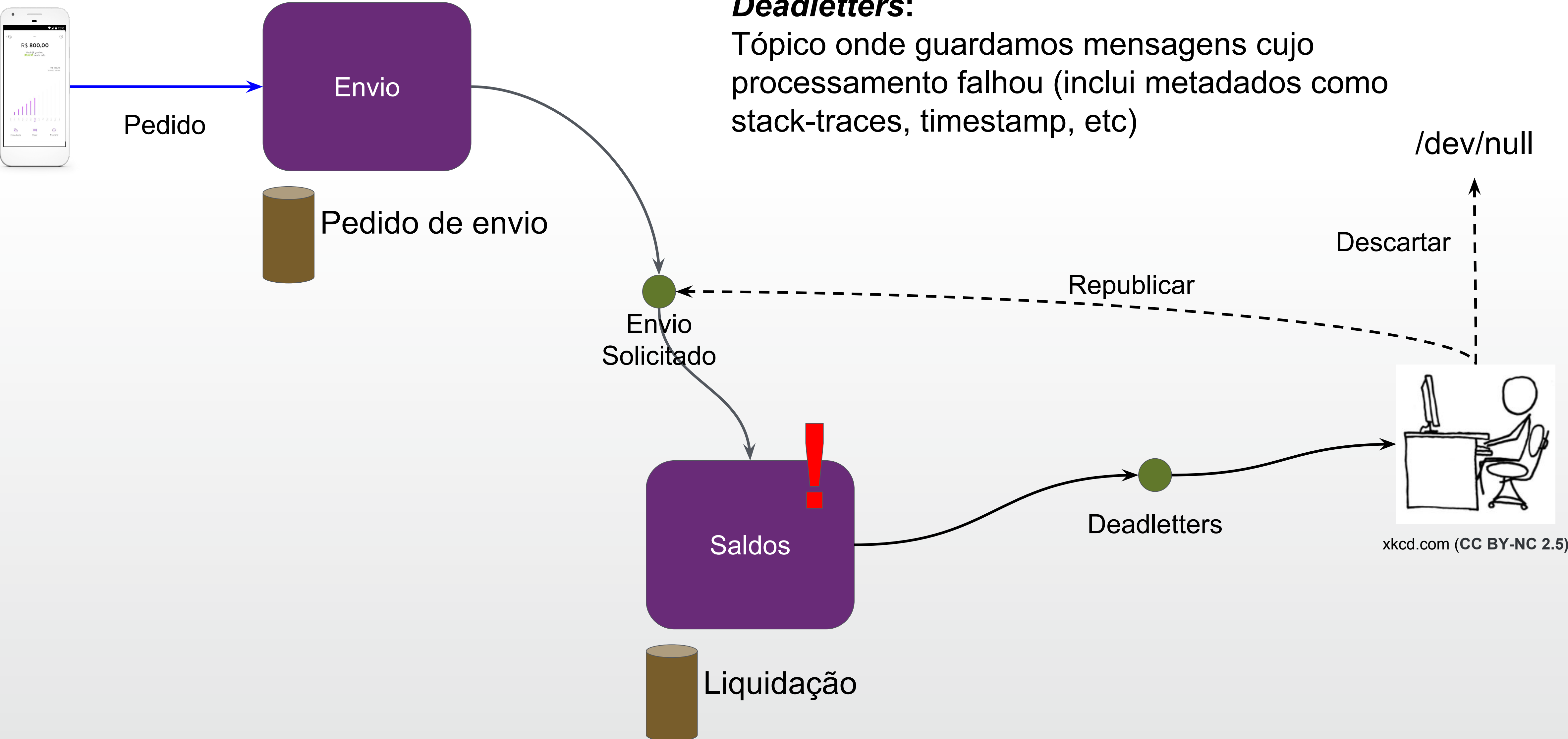


# E se rolar um bug ou mensagem inválida?



# E se rolar um bug ou mensagem inválida?

**Deadletters:**  
Tópico onde guardamos mensagens cujo processamento falhou (inclui metadados como stack-traces, timestamp, etc)

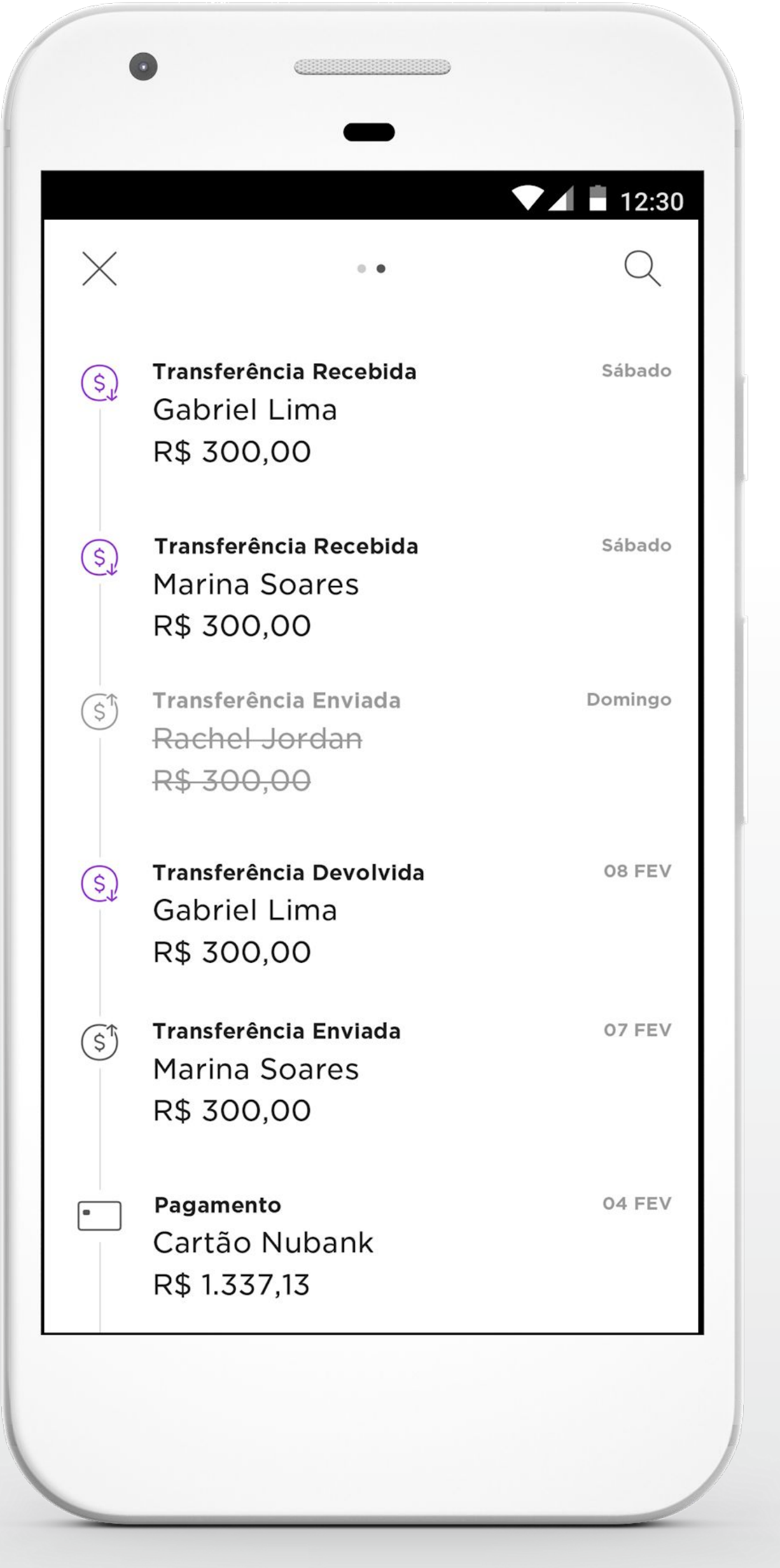




An aerial night photograph of a city, likely Rio de Janeiro, showing a river flowing through the urban landscape and a coastline with a bay. The city lights are visible against the dark terrain and water.

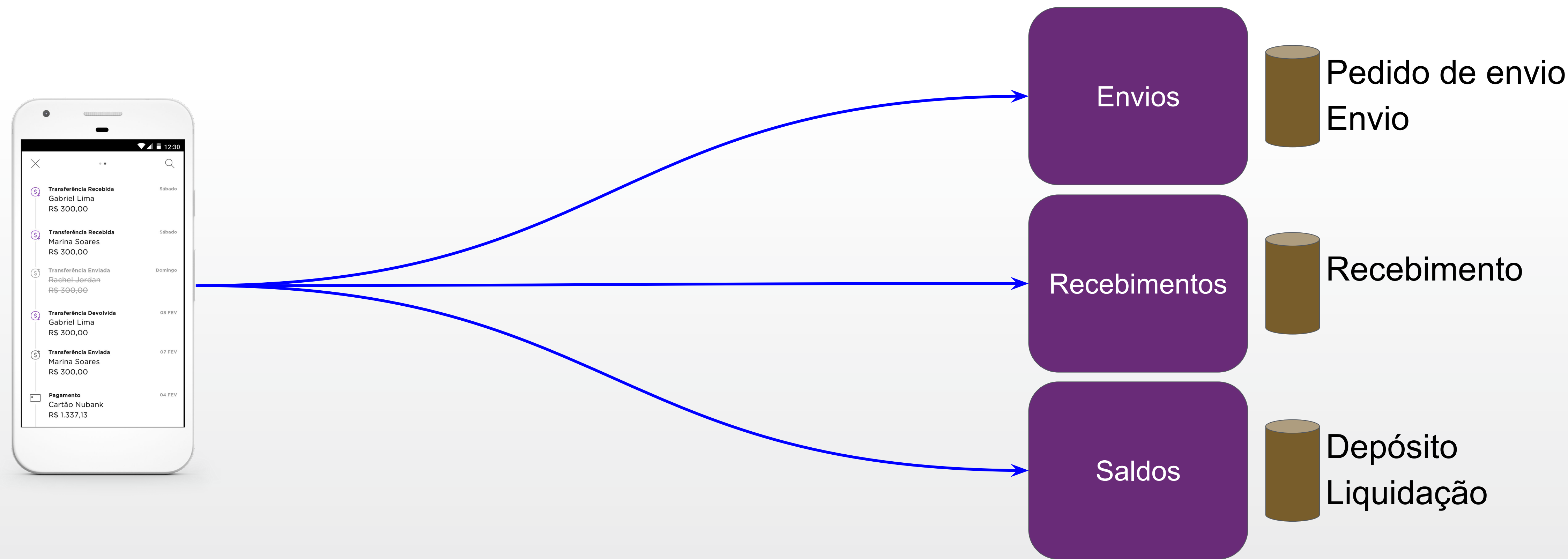
# Feed de movimentações







# Dados distribuídos = muitos requests







# Modelo do backend != visão do cliente

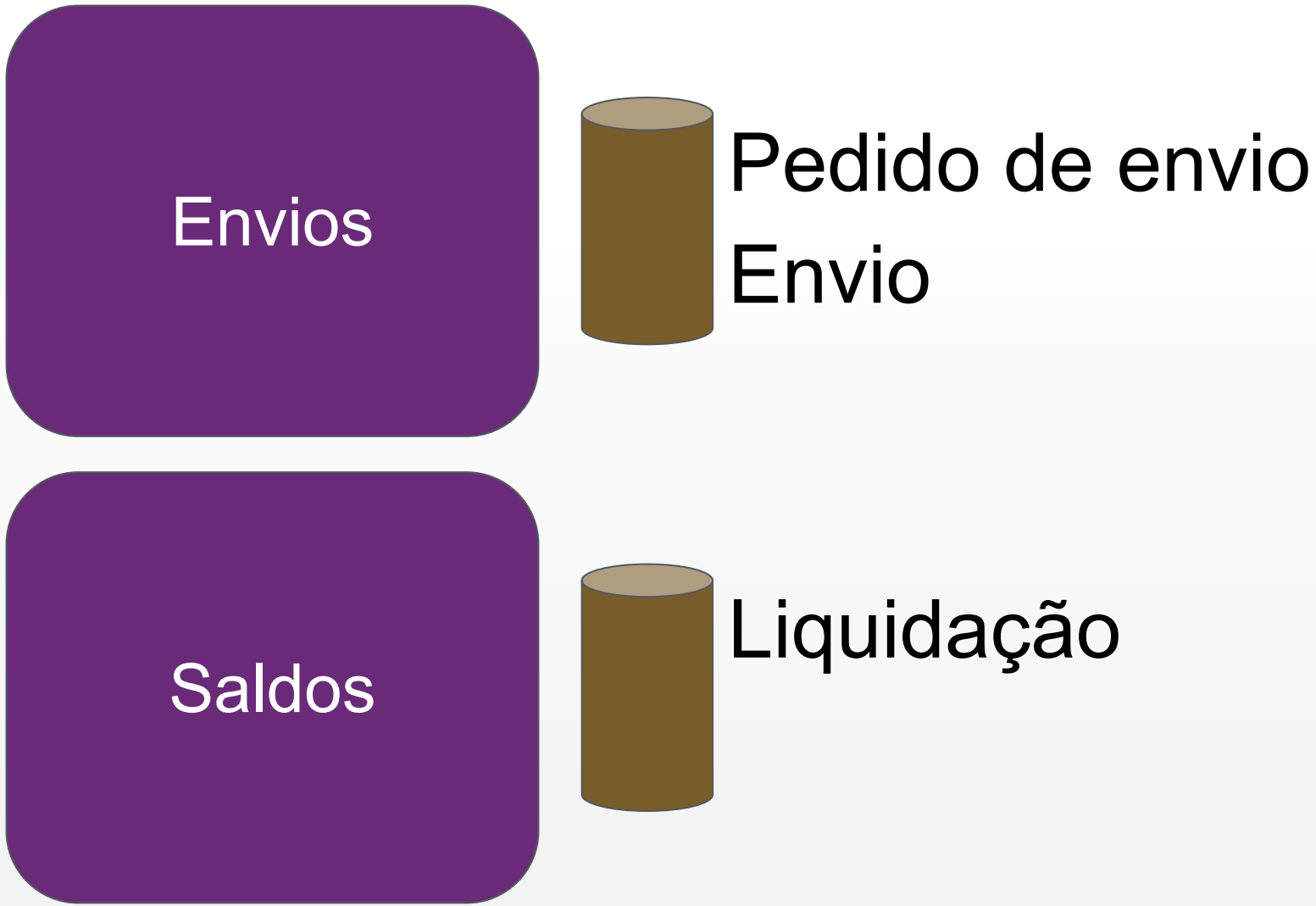
## Evento de envio no feed

- Valor
- Data
- Status: pendente | falha | sucesso

Transferência enviada	24 JUL
Gabriel Lima	
R\$ 23,65	
Transferência enviada	24 JUL
Gabriel Lima	
<del>R\$ 23,65</del>	
Transferência enviada	24 JUL
Gabriel Lima	
R\$ 23,65	

## Liquidação Envio

404   	404
	404
	

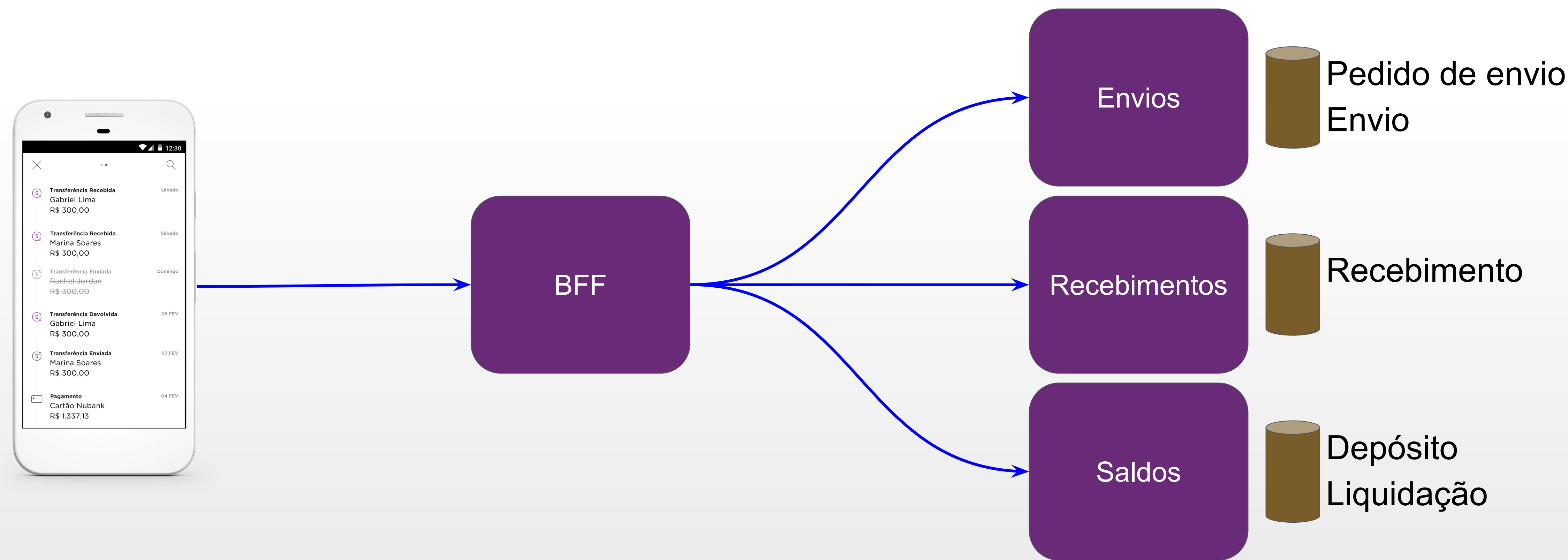




## Ciclos de atualização lentos

- Lançar uma nova versão de um aplicativo numa app store pode demorar dias
- Muitos usuários continuam com versões antigas por muito tempo
- Bugfixes e otimizações demoram para chegar
- Frontend acoplado impede evoluções no backend

# Backend For Frontend



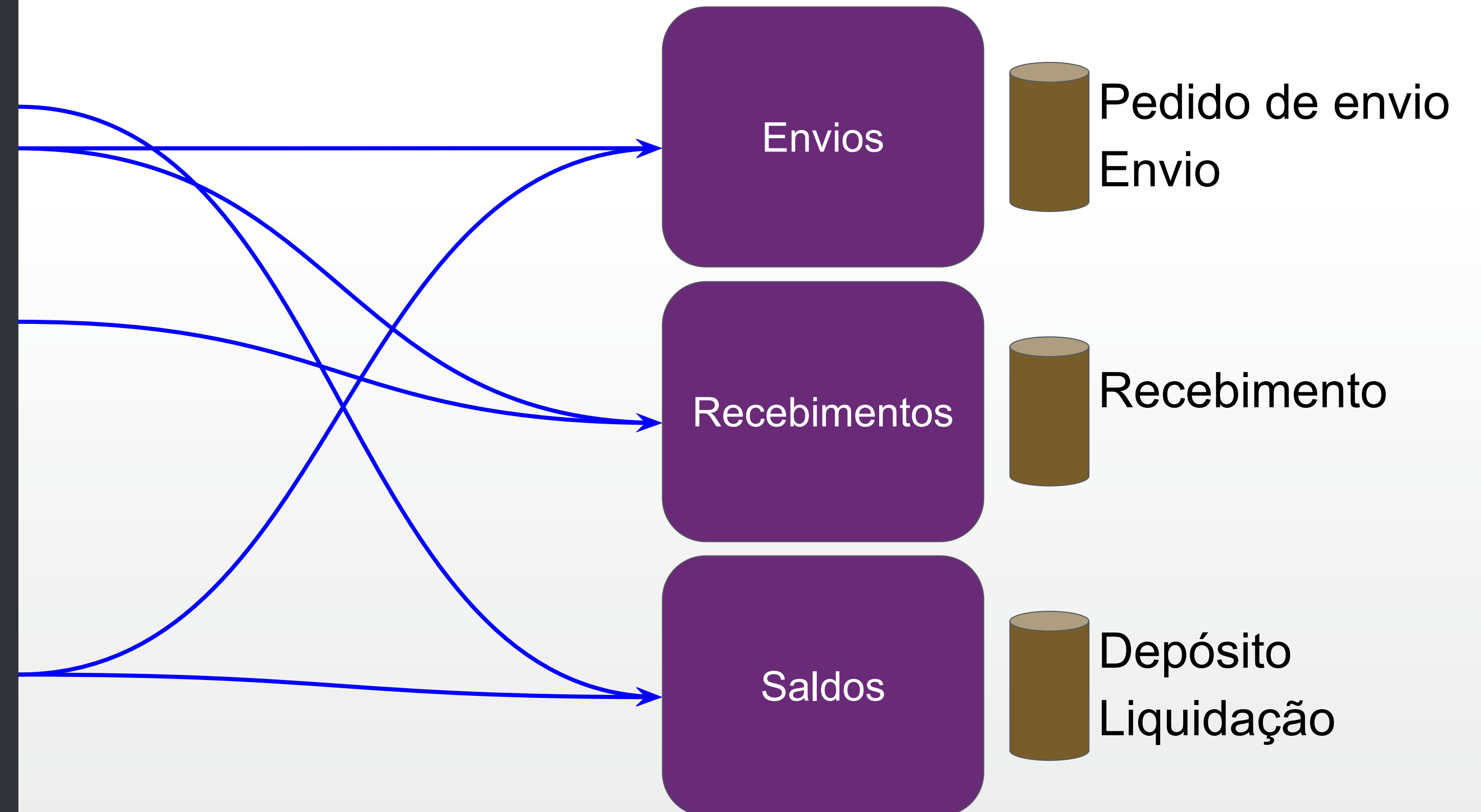


"GraphQL is a query language designed to build client applications by providing an intuitive and flexible syntax and system for describing their data requirements and interactions."

(GraphQL spec: <http://facebook.github.io/graphql/>)

# Schema é o modelo disponível para o frontend

```
schema {  
  query: Query  
}  
  
type Query {  
  saldo(accountId: ID!): Float  
  feed(accountId: ID!):  
    [Envio | Recebimento]  
}  
  
type Recebimento {  
  data: Date  
  valor: Float  
}  
  
enum StatusEnvio {  
  PENDENTE, FALHA, SUCESSO  
}  
  
type Envio {  
  data: Date  
  valor: Float  
  status: StatusEnvio  
}
```





# Query define os campos que o client precisa

```
query feedScreen($accountId: ID!) {  
  feed(accountId: $accountId) {  
    ... on Envio {  
      data  
      valor  
      status  
    }  
    ... on Recebimento {  
      data  
      valor  
    }  
  }  
}
```

POST /api/query



```
schema {  
  query: Query  
}  
  
type Query {  
  saldo(accountId: ID!): Float  
  feed(accountId: ID!):  
    [Envio | Recebimento]  
}  
  
type Recebimento {  
  data: Date  
  valor: Float  
}  
  
enum StatusEnvio {  
  PENDENTE, FALHA, SUCESSO  
}  
  
type Envio {  
  data: Date  
  valor: Float  
  status: StatusEnvio  
}
```

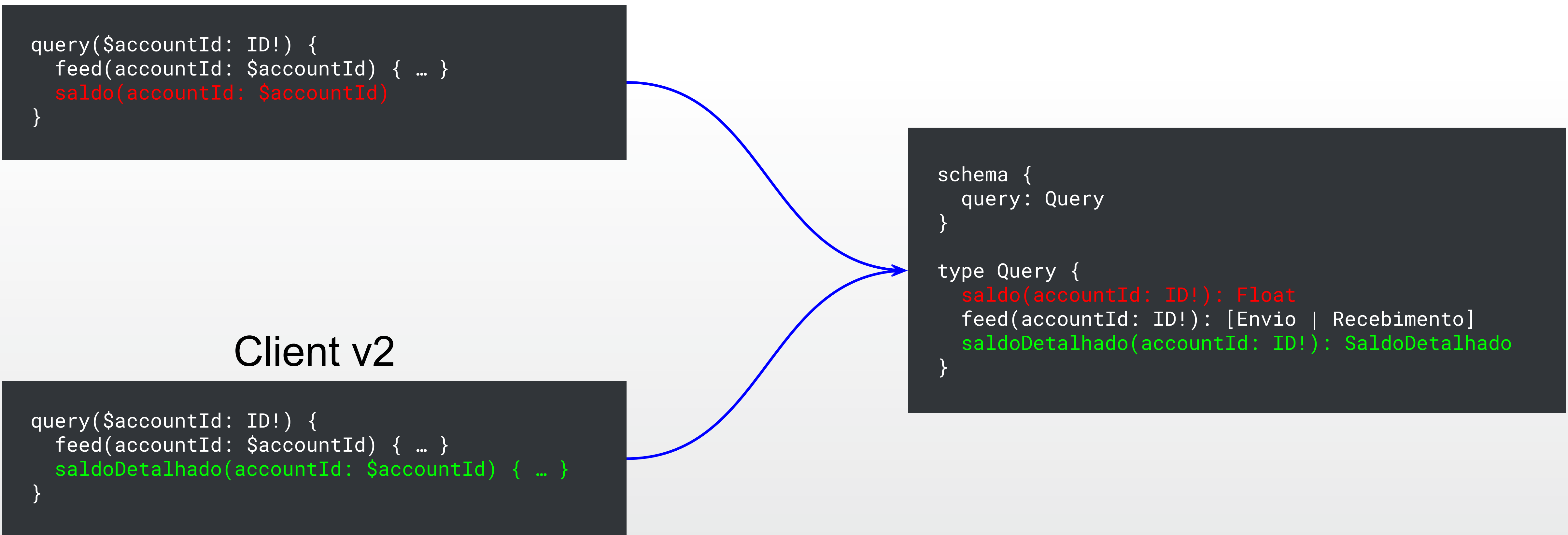
# Query define os campos que o client precisa

## Client v1

```
query($accountId: ID!) {  
  feed(accountId: $accountId) { ... }  
  saldo(accountId: $accountId)  
}
```

## Client v2

```
query($accountId: ID!) {  
  feed(accountId: $accountId) { ... }  
  saldoDetalhado(accountId: $accountId) { ... }  
}
```



```
schema {  
  query: Query  
}  
  
type Query {  
  saldo(accountId: ID!): Float  
  feed(accountId: ID!): [Envio | Recebimento]  
  saldoDetalhado(accountId: ID!): SaldoDetalhado  
}
```

The diagram illustrates the relationship between two client versions and a shared schema. On the left, 'Client v1' and 'Client v2' are shown. Client v1 has a query with fields 'feed' and 'saldo'. Client v2 has a query with fields 'feed' and 'saldoDetalhado'. Two blue curved arrows originate from the 'saldo' field in Client v1 and the 'saldoDetalhado' field in Client v2, both pointing towards a central schema definition on the right. The schema defines a 'Query' type with three fields: 'saldo' (returning a Float), 'feed' (returning a list of Envio or Recebimento), and 'saldoDetalhado' (returning a SaldoDetalhado object).



## Recapitulando

- Event-sourcing
- Comunicação assíncrona via Kafka
- Backend for Frontend com GraphQL

# Obrigado!

Gustavo Bicalho  
Maurício Verardo

<https://sou.nu/vagasnu>

