***Visão do Sistema***

**Wise Control**

**Janeiro/2023**

# 

# Introdução

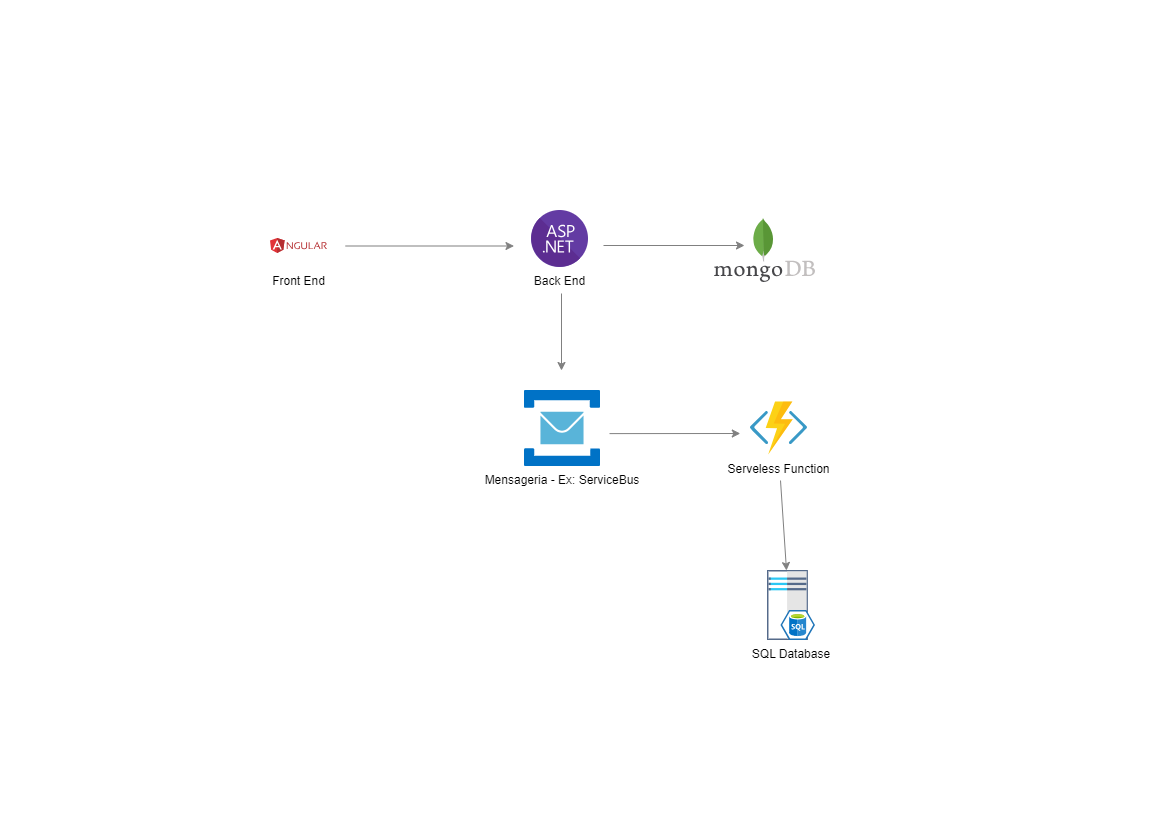
O Wise Control é um sistema de lançamentos de valores classificados em créditos e débitos que tem como principal objetivo a validação técnica para a vaga de Arquiteto de Software

# Requisitos não funcionais

Os requisitos não funcionais principais são:

* Aplicação das principais tecnologias descritas na vaga
* Aplicação de Testes Unitários
* Restrições em relação a segurança, ponto limitado ao tempo de desenvolvimento do teste
* Uso de Design Patterns, ponto também limitado ao tempo de desenvolvimento do teste

# Diagrama de Arquitetura do Sistema



# Premissas e Restrições

* O teste técnico desenvolvido focou no desenvolvimento de back-end e front-end para alimentação das informações do MongoDB
* Não foi realizado o desenvolvimento do carregamento dos dados do SQL Server através de uma servless function, que ficaria para uma etapa posterior. Porém, o mesmo manteve-se no diagrama para ilustrar a extensão da arquitetura para distribuição dos dados para um ETL, banco de dados SQL, envio de notificações.

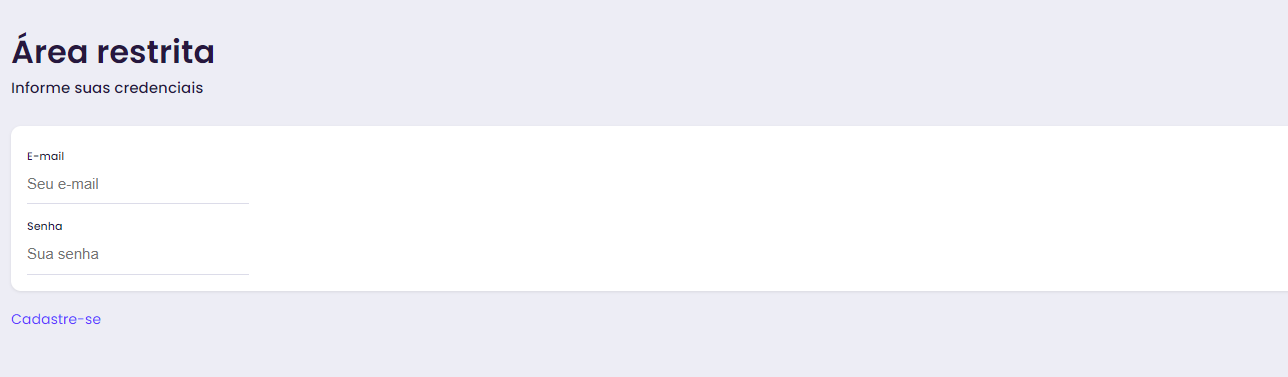
# Funcionalidades Gerais do Sistema

A seguir uma descrição resumida das funcionalidades implementadas assim como evidências da mesma dentro da aplicação.

## Login

Para uso da aplicação o usuário deverá fazer Login fornecendo as credenciais de e-mail e senha.

Após fornecer E-mail e Senha, a aplicação deverá validar o Login do Usuário após consumo de API e mantê-lo Logado para uso das demais funcionalidades.

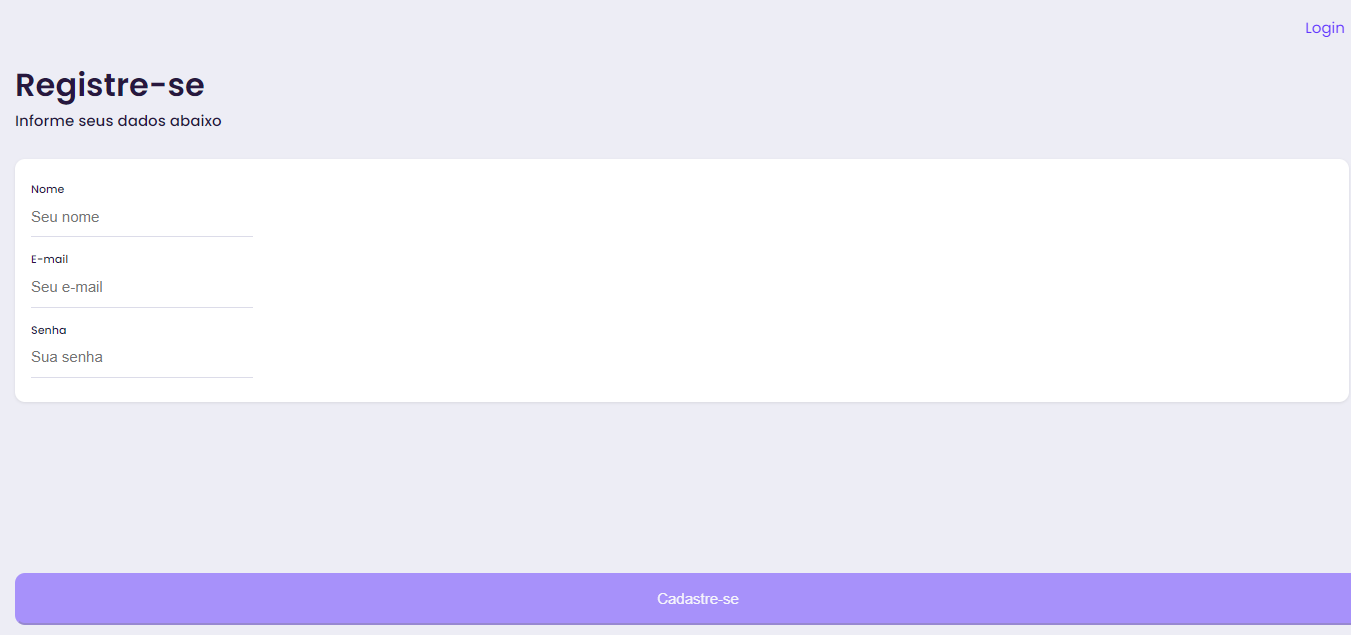


Ex: Tela de Login Do Sistema

## Registro de Usuário

Para realização da identificação, o usuário deve previamente realizar um cadastro fornecendo dados como nome do usuário, e-mail e senha a ser utilizada em momento posterior.

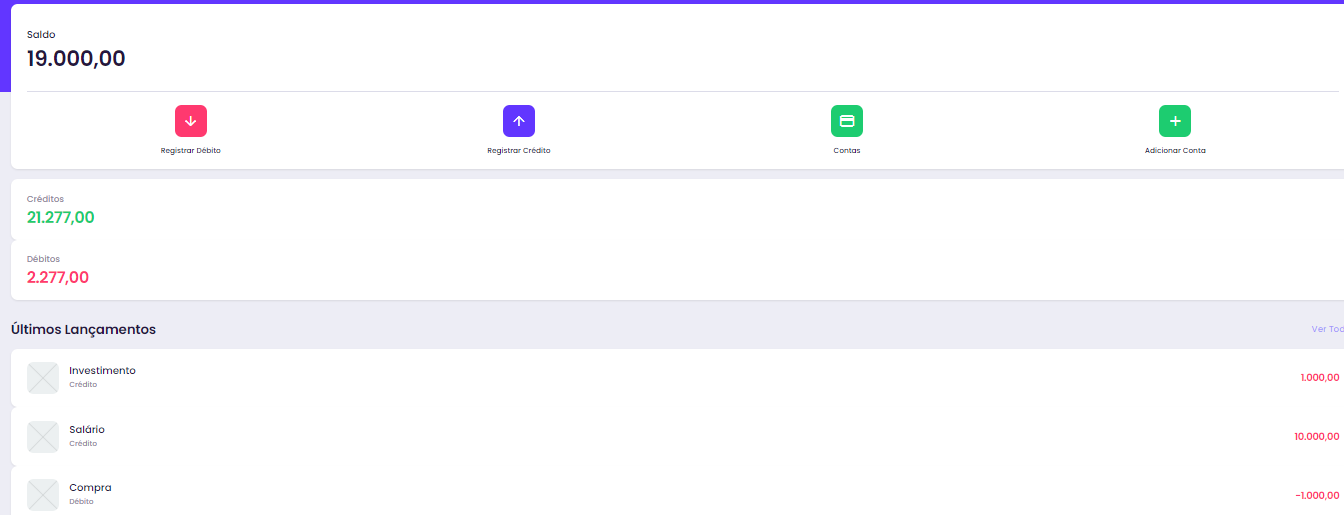
Esses dados são armazenados dentro de uma collection Users em um banco de dados do MongoDB



## Dashboard

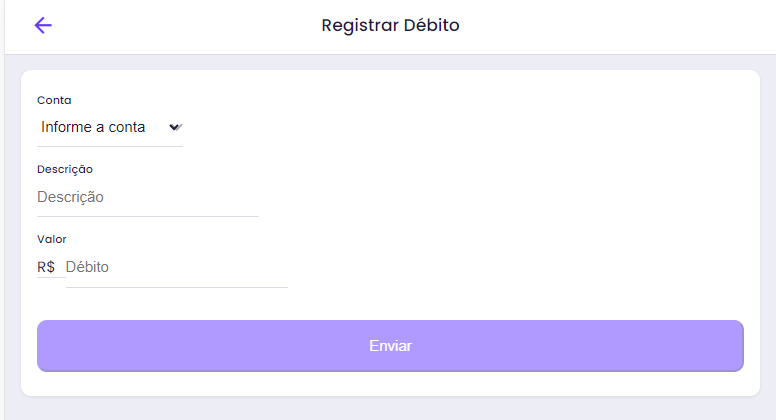
Após a realização da identificação do usuário, o sistema passa a exibir informações de contas e transações, assim como o saldo geral para o usuário.

Os lançamentos, sendo créditos e débitos, são agrupados em contas que são visualizáveis a todos os usuários.



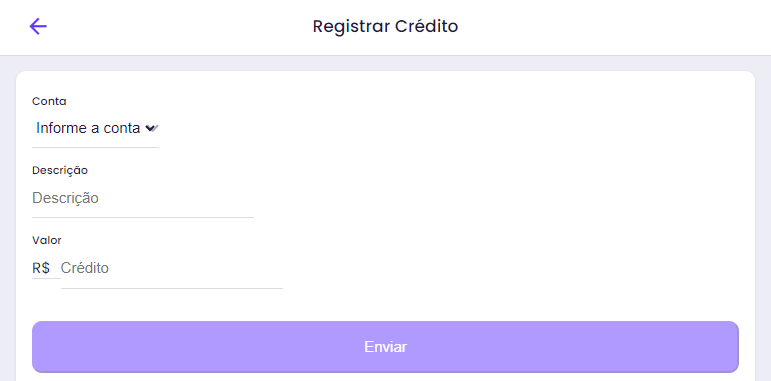
## Registro de Débito

Funcionalidade criada para registro de lançamento do tipo débito. Para esses pontos o usuário deve fornecer a conta da operação, a descrição da mesma e seu valor monetário.



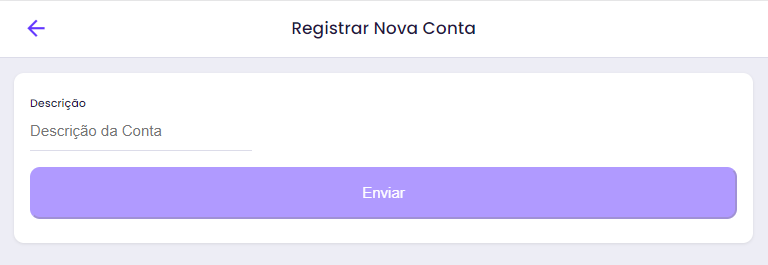
## Registro de Crédito

Funcionalidade criada para registro de lançamento do tipo crédito. Para esses pontos o usuário deve fornecer a conta da operação, a descrição da mesma e seu valor monetário.



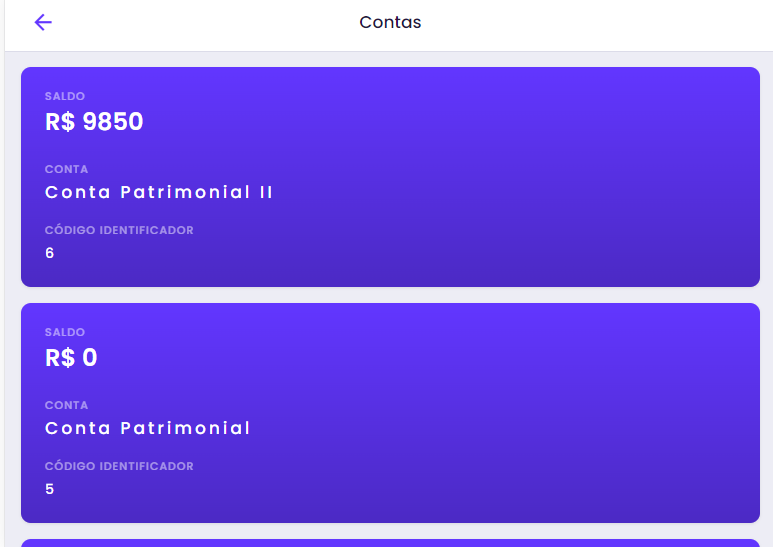
## Registro de Conta

Registro de Conta dentro do sistema. Para isso, o usuário deve fornecer uma identificação que auxilia na identificação da mesma.



## Visualização de Contas Cadastradas

Listagem e visualização das contas cadastradas dentro do sistema para agrupamento de lançamentos.



## Modelo de Dados

Foi considerado o uso de um banco de dados NoSQL, MongoDB e portanto, foi utilizado coleções de documentos ao invés de entidades relacionais.

Os dados podem ser agrupados nas seguintes coleções: Transactions, Accounts e Users.

As informações são armazenadas em JSON e são recuperadas através da biblioteca MongoDB.Driver.

Os seguintes dados são gravados para cada documento registrado

**Accounts Collection**

* AccountId
* Description
* Balance

**Transactions Collection**

* AccountId
* TransactionId
* Type
* Description
* Date
* Value

**Users Collection**

* UserId
* Name
* Email
* Password

## Sugestão de Futuros Desenvolvimentos

## Existem várias melhorias a serem realizadas dentro do código implementado.

Do ponto de vista de arquitetura de sistemas o uso de CQRS permitiria uma melhor separação entre operações de consulta e escrita do banco de dados.

Em relação as funcionalidades, é possível criar um módulo de microserviço que faça o envio de notificações lendo as mensagens disponibilizadas dentro de serviços como Amazon SQS, Azure Service Bus. Serviços como Amazon SNS podem realizar o envio dessas mensagens.

A criação de um microserviço para alimentação de um banco de dados relacional como um SQL Server também pode ser utilizado e é ilustrado no diagrama de arquitetura da aplicação.

Dentro do cenário de Devops a criação de GitHub Actions e Pipelines também são considerados crucias em uma aplicação real, providenciando integração e entrega continua.

Do ponto de vista de segurança, realizar o refresh dos tokens JWT, a implementação de mecanismos anti-forgery e o uso de uma política Cors apropriada para ambientes em produção são requisitos críticos.

Também poderá haver uma maior granularidade em relação as funcionalidades e dados que determinado usuário pode acessar, como a implementar contas confidenciais as quais somente determinados usuários teriam acesso.

Em relação aos testes unitários, foram criados classes base para esta finalidade, mas o nível de cobertura para um cenário em produção precisaria aumentar consideravelmente. Uma documentação de plano de testes também seria desejável.