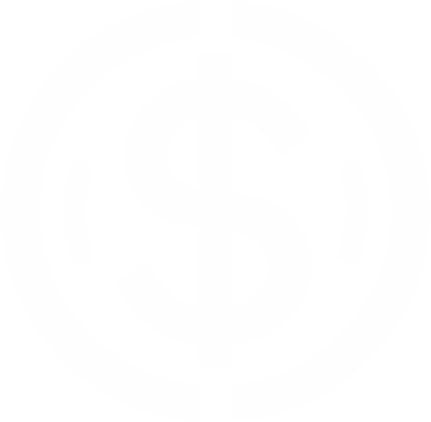
**SICODE**



Sistema de Controle de Débitos - Sistema de Parcelamento

O **SICODE** é um sistema desenvolvido em **Ruby** com o objetivo de otimizar o processo de parcelamento de guias protestadas na Central de Recuperação de Ativos (CRA). Atualmente, o controle de pagamento e a emissão dessas guias parceladas são realizados manualmente. A cada mês, um servidor é responsável por emitir a guia e enviá-la ao devedor, enquanto o controle de pagamentos e baixas é mantido por meio de uma planilha Excel, compartilhada entre os servidores da **Central de Arrecadação Judicial (CAJ)**.

Este documento tem como finalidade detalhar o processo de desenvolvimento do **SICODE** e seu funcionamento. Todas as funcionalidades do sistema serão descritas, juntamente com as regras de negócio que regem o parcelamento de guias protestadas, o passo a passo dos procedimentos e as respectivas partes do código que implementam cada funcionalidade. Ao seguir este guia, você obterá uma visão holística do sistema, o que facilitará a identificação de possíveis melhorias e ajustes. Cada função será abordada separadamente, permitindo o estudo e a análise de forma individualizada. Fique atento ao longo do documento para os itens marcados como **"Passo a Passo"**, que o guiarão através da execução prática das funcionalidades descritas.

# Cadastro

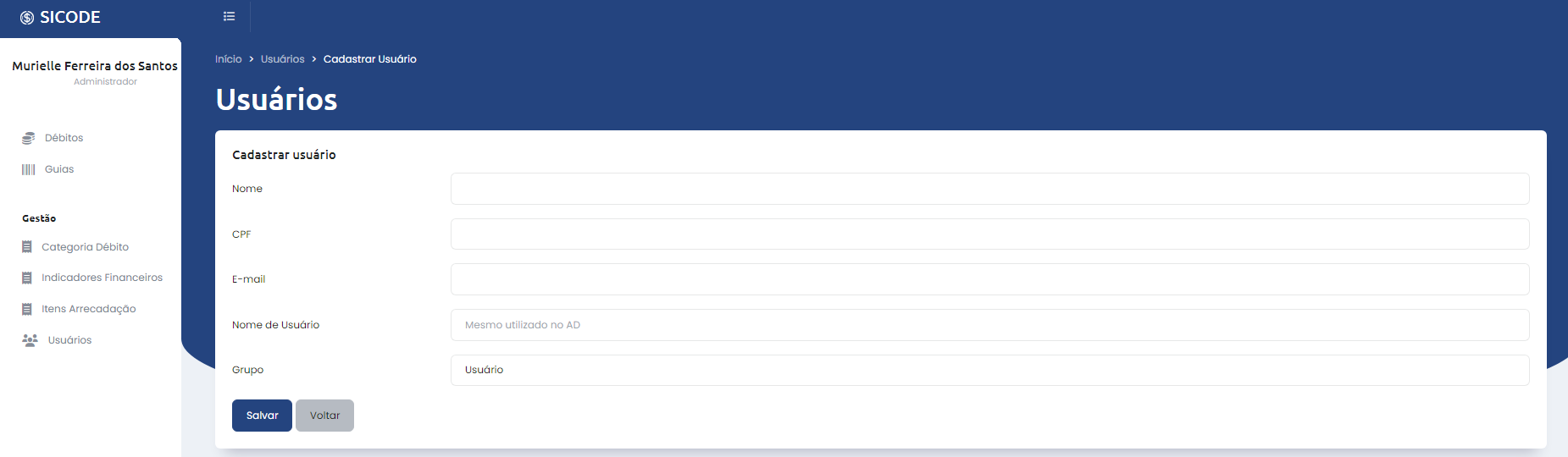
O cadastro é o primeiro passo para o acesso ao SICODE. Esse cadastro é feito pelo apenas pelo perfil de *administrador*. É necessário a seguintes informações, para prosseguir com esse procedimento:

* Nome
* CPF
* E-mail
* Nome de usuário (é o mesmo utilizado no usuário AD)
* Grupo (definição do perfil que esse usuário terá)

Só existe 2 perfis atualmente: usuário (usuário de outra unidade/externo). O perfil do usuário será definido pelo chefe imediato desta unidade. Ele é passível de ser mudado, sem a necessidade de um novo cadastro.

Ainda não existe separação por unidade ou serventia, todos os usuários compartilham um mesmo ambiente (como se fosse uma serventia única do Projudi).

Passo a passo: Preencher as informações solicitadas e definir o perfil de acesso. E salvar.



Observação: Não é necessário a inserção de todas as informações para finalizar o cadastro, porém, é recomendável que sim. Pois garante um melhor controle das informações inseridas de cada usuário no sistema, e possibilita alterações no cadastro, caso necessário.

Código:

# Login

O login é feito no url: <https://sicode-stag-frontend.apps.ocp-prd.tjgo.jus.br/login>.



O usuário e senha são os mesmos utilizados para o login no computador, ou seja, o usuário AD e a senha.

|  |  |
| --- | --- |
| Usuário | = Usuário AD |
| Senha | = Senha do computador |

Passo a passo: Informar o usuário AD e a senha é a do computador. Em seguida, clicar em ‘Entrar’.

*Observação: Existe um botão ‘Início’ nessa primeira tela, e caso ele seja clicado, o usuário irá ser direcionado para uma janela na qual é informado o hash (um código de acesso) para identificação referente ao débito a ser consultado. Para essa consulta, não é necessário login no sistema.*



Código:

# frozen\_string\_literal: true

class SessionsController < Devise::SessionsController

layout 'login'

before\_action :configure\_sign\_in\_params, only: [:create]

# GET /resource/sign\_in

def new

super

end

# POST /resource/sign\_in

def create

super

end

# DELETE /resource/sign\_out

def destroy

super

end

# protected

# If you have extra params to permit, append them to the sanitizer.

def configure\_sign\_in\_params

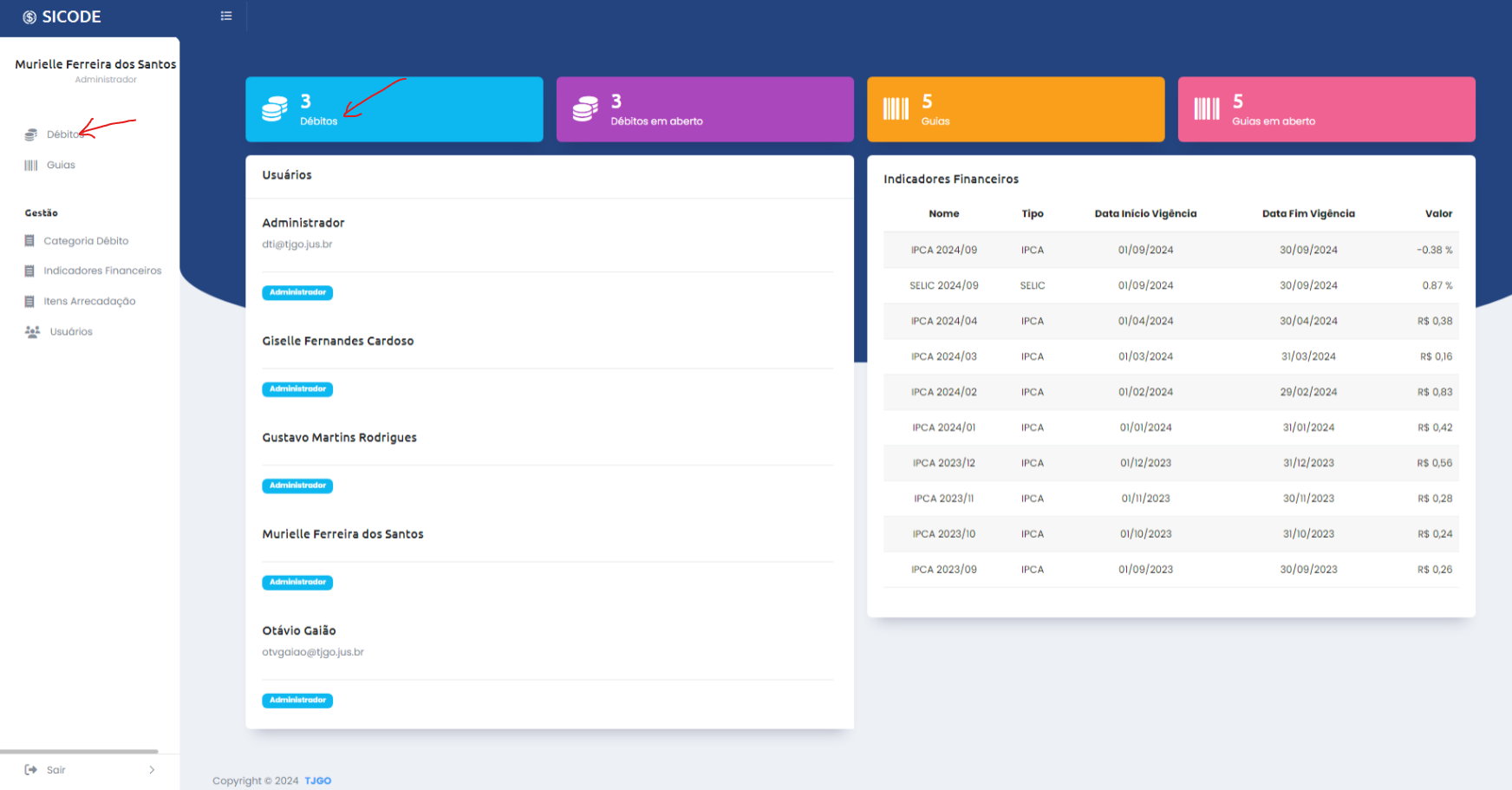
devise\_parameter\_sanitizer.permit(:sign\_in, keys: [:attribute, :email, :username, :password])

end

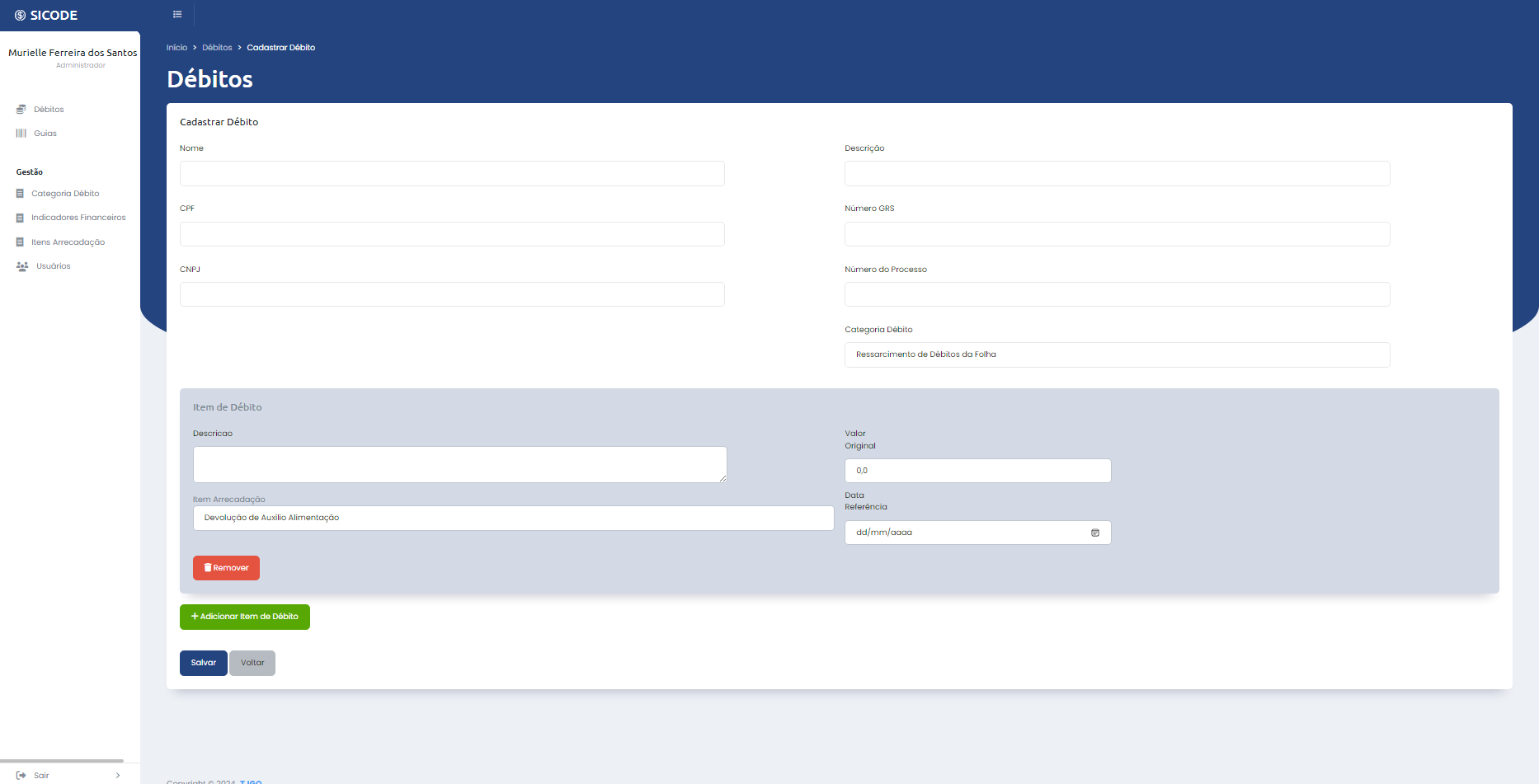
end

# Débitos

Essa funcionalidade pode ser acessada pelo botão azul ou pelo menu lateral presentes na tela. Ela serve para o cadastro de novos débitos, visualização dos débitos cadastrados e edição ou exclusão desses débitos.



Passo a passo para o Cadastro de Novos débitos: Clique no botão ‘+ Adicionar’. Insira as informações do devedor, e em seguida, insira os dados referentes ao débito.



Lembre-se de selecionar no item ‘Categoria Débito’ uma informação compatível a um dos itens da opção ‘Item de Arrecadação’.

Código:

class Debito < ApplicationRecord

has\_many :debito\_itens, dependent: :destroy

has\_many :parcelas, dependent: :destroy

belongs\_to :categoria\_debito

validates :nome, :valor\_ref, presence: true

validate :formatocpfcnpj

enum situacao: { aguardando\_pagamento: 'aguardando\_pagamento', pago: 'pago',

vencido: 'vencido', protesto: 'protesto', protesto\_pago: 'protesto\_pago', cancelado: 'cancelado', divida\_ativa: 'divida\_ativa' }

scope :em\_aberto, -> { where(situacao: [:aguardando\_pagamento, :vencido]) }

before\_save :sanitize\_cpf\_cnpj

after\_save :atualizar, unless: -> { data\_atualizacao == Date.current}

accepts\_nested\_attributes\_for :debito\_itens, reject\_if: :all\_blank , allow\_destroy: true

def registrar

return false if debito\_itens.empty?

return false unless self.valid?

self.hash\_acesso = SecureRandom.hex(10)

debito\_itens.each do |debito\_item|

debito\_item.atualizar\_valores

end

self.data\_atualizacao = Date.current

self.situacao = :aguardando\_pagamento

self.valor\_ref = debito\_itens.sum(:valor\_ref)

self.valor\_atualizado = debito\_itens.sum(:valor\_atualizado)

self.data\_ref = Date.current

self.data\_vencimento = Date.current + 30.days

self.save

end

def atualizar

debito\_itens.each do |debito\_item|

debito\_item.atualizar\_valores

end

self.data\_atualizacao = Date.current

self.valor\_atualizado = debito\_itens.sum(:valor\_atualizado)

self.save

end

def atualizar\_valor

update(valor\_atualizado: debito\_itens.sum(:valor\_atualizado))

end

def parcelar(data\_primeira\_parcela:, numero\_parcelas:, valor\_primeira\_parcela: 0.0)

atualizar\_valor

atualizar if data\_atualizacao < Date.current

Parcelamento.new(debito: self, data\_primeira\_parcela: data\_primeira\_parcela, numero\_parcelas: numero\_parcelas, valor\_primeira\_parcela: valor\_primeira\_parcela)

rescue => e

false

end

def confirmar\_parcelamento

parcelas.map(&:confirmar)

end

def negar\_parcelamento

self.transaction do

self.parcelas.each do |parcela|

raise ActiveRecord::Rollback if parcela.guia.present?

parcela.destroy

end

end

end

def self.search(params={})

debitos = Debito.all

debitos = debitos.where("nome ilike ?", "%#{params[:nome]}%") if params[:nome].present?

debitos = debitos.where("cpf ilike ?", "%#{params[:cpf]}%") if params[:cpf].present?

debitos = debitos.where("cnpj ilike ?", "%#{params[:cnpj]}%") if params[:cnpj].present?

debitos = debitos.where("situacao = ?", params[:situacao]) if params[:situacao].present?

debitos = debitos.where("valor\_ref = ?", params[:valor\_ref]) if params[:valor\_ref].present?

debitos = debitos.where("data\_ref = ?", params[:data\_ref]) if params[:data\_ref].present?

debitos = debitos.where("data\_vencimento = ?", params[:data\_vencimento]) if params[:data\_vencimento].present?

debitos = debitos.where("data\_atualizacao = ?", params[:data\_atualizacao]) if params[:data\_atualizacao].present?

debitos = debitos.where("categoria = ?", params[:categoria]) if params[:categoria].present?

debitos = debitos.where("hash\_acesso = ?", params[:hash\_acesso]) if params[:hash\_acesso].present?

debitos = debitos.order(params[:order]) if params[:order].present?

debitos = debitos.page(params[:page]).per(params[:per])

debitos

end

private

def sanitize\_cpf\_cnpj

self.cpf = cpf.gsub(/[^0-9]/, '') if cpf.present?

self.cnpj = cnpj.gsub(/[^0-9]/, '') if cnpj.present?

end

def formatocpfcnpj

if cpf.present?

errors.add(:cpf, 'CPF inválido') unless CPF.valid?(cpf)

end

if cnpj.present?

errors.add(:cnpj, 'CNPJ inválido') unless CNPJ.valid?(cnpj)

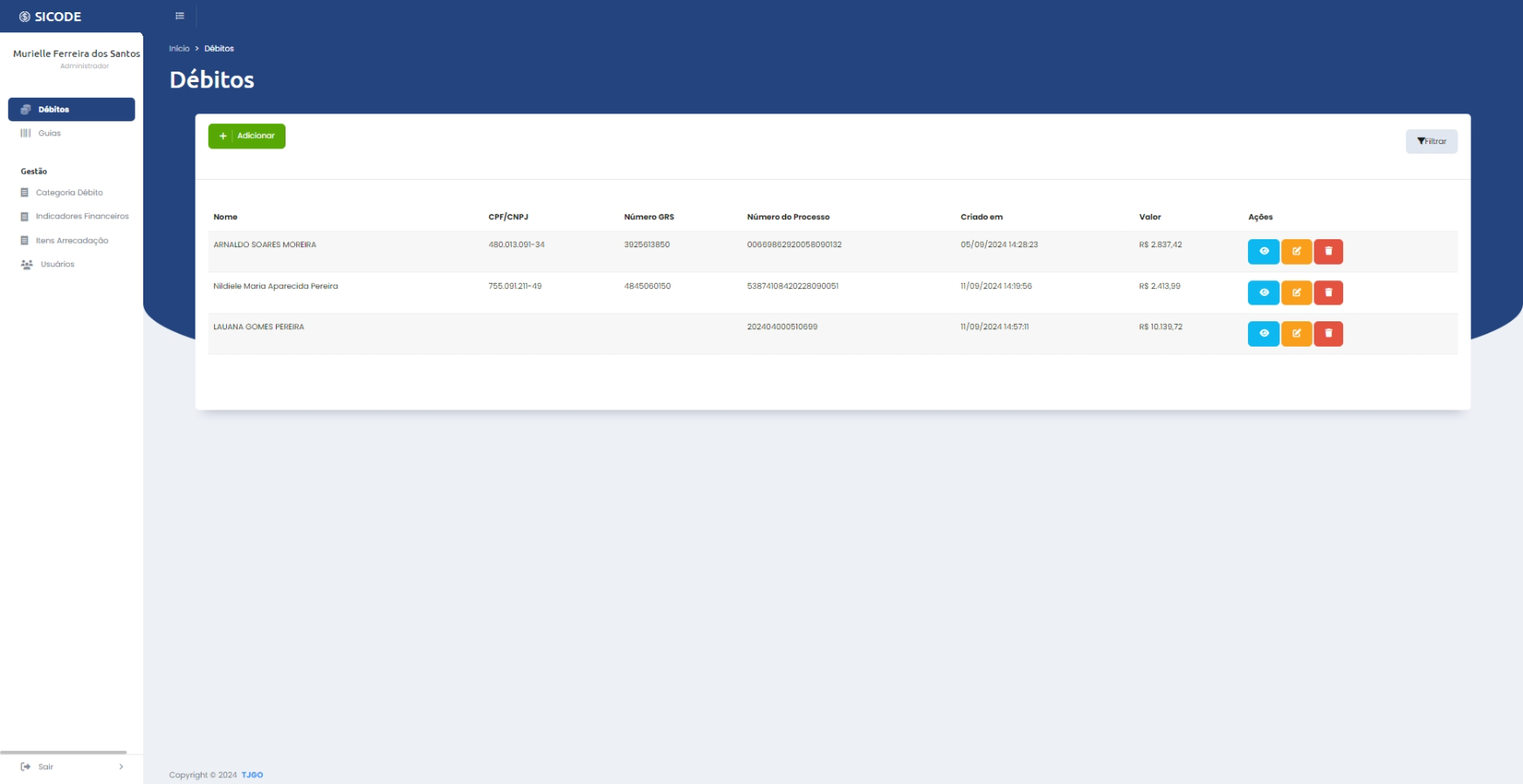
end

end

end

# Guias

Nesta funcionalidade, é mostrada todas as guias que já foram emitidas no sistema SICODE.



Código:

module Admin

class GuiasController < Admin::ApplicationController

def index

@guias = Guia.search(search\_params)

@guias = GuiaDecorator.decorate\_collection(@guias)

end

def show

@guia = Guia.find(params[:id])

@guia = GuiaDecorator.new(@guia)

end

private

def search\_params

params.permit(:numero, :data\_vencimento, :valor, :situacao, :tipo, :ativa, :cpf, :nome, :page, :per\_page, :sort\_by)

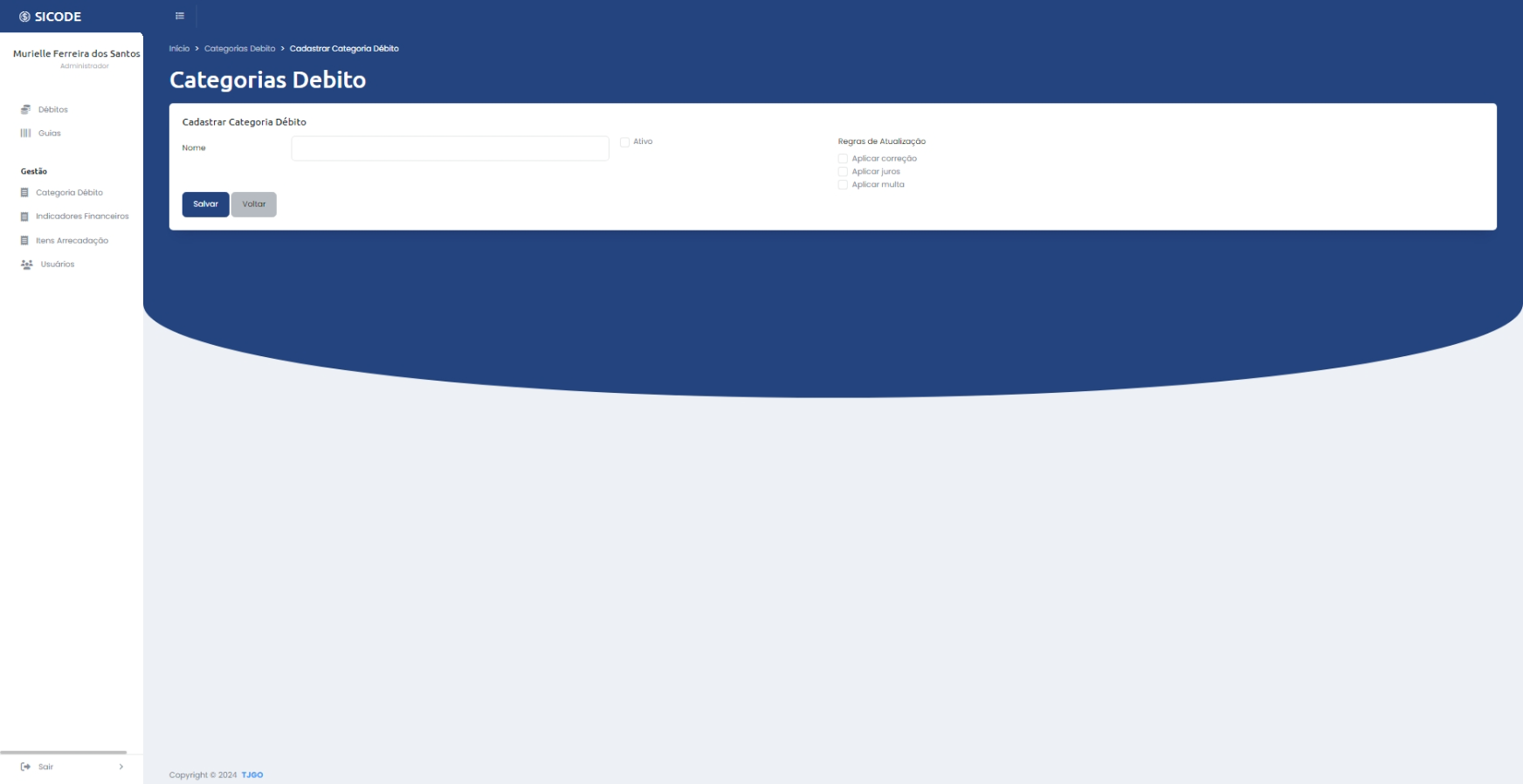
end

end

end

# Gestão - Categoria Débito

Aqui é onde as regras de juros serão aplicadas (Selic e IPCA). Essa função permite o cadastro, edição e exclusão de categorias de débito. Pode-se também ativar ou não a categoria cadastrada. Além disso, é aqui que são escolhidas as regras de atualização.



Passo a passo: Clique no botão **+Adicionar**, preencha as informações solicitadas. E salvar.

Código:

module Admin

class CategoriasDebitoController < Admin::ApplicationController

def index

@categorias\_debito = CategoriaDebito.search(search\_params)

@categorias\_debito = CategoriaDebitoDecorator.decorate\_collection(@categorias\_debito)

respond\_to do |format|

format.html

format.turbo\_stream { render turbo\_stream: turbo\_stream.replace('categorias\_debito', partial: 'admin/categorias\_debito/categoria\_debito', locals: { categorias\_debito: @categorias\_debito }) }

end

end

def show

@categoria\_debito = CategoriaDebito.find(params[:id])

@categoria\_debito = CategoriaDebitoDecorator.decorate(@categoria\_debito)

end

def new

@categoria\_debito = CategoriaDebito.new

authorize @categoria\_debito

end

def edit

@categoria\_debito = CategoriaDebito.find(params[:id])

authorize @categoria\_debito

end

def create

@categoria\_debito = CategoriaDebito.new(categoria\_debito\_params)

authorize @categoria\_debito

if @categoria\_debito.save

redirect\_to admin\_categorias\_debito\_path, notice: 'Categoria de débito criada com sucesso'

else

flash.now[:alert] = 'Não foi possível criar a categoria de débito'

render :new

end

end

def update

@categoria\_debito = CategoriaDebito.find(params[:id])

authorize @categoria\_debito

if @categoria\_debito.update(categoria\_debito\_params)

redirect\_to admin\_categorias\_debito\_path, notice: 'Categoria de débito atualizada com sucesso'

else

flash.now[:alert] = 'Não foi possível atualizar a categoria de débito'

render :edit

end

end

def destroy

@categoria\_debito = CategoriaDebito.find(params[:id])

authorize @categoria\_debito

@categoria\_debito.destroy

respond\_to do |format|

format.html { redirect\_to admin\_categorias\_debito\_path, notice: 'Categoria de débito removida com sucesso' }

format.turbo\_stream { render turbo\_stream: turbo\_stream.remove(@categoria\_debito) }

end

end

private

def categoria\_debito\_params

params.require(:categoria\_debito).permit(:nome, :ativo, :aplicar\_correcao, :aplicar\_juros, :aplicar\_multa)

end

def search\_params

params.permit(:nome, :ativo, :sort\_by, :page)

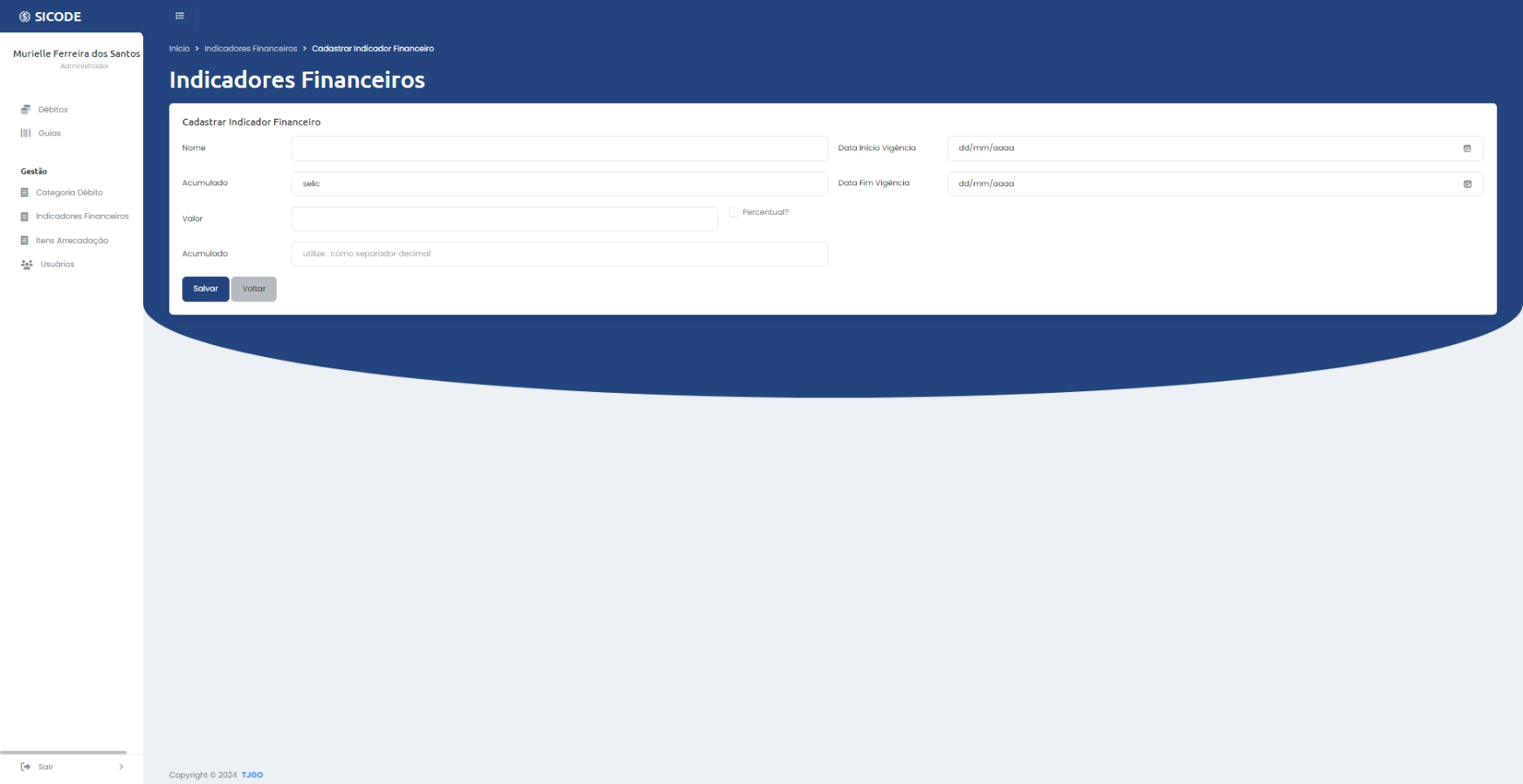
end

end

end

# Gestão - Indicadores Financeiros

Local em que é feito o cadastro dos indicadores financeiros, o início as datas de vigência.



Código:

module Admin

class IndicadoresFinanceirosController < Admin::ApplicationController

def index

@indicadores\_financeiros = IndicadorFinanceiro.all.order(params[:sort\_by] || 'id desc').page(params[:page])

@indicadores\_financeiros = IndicadorFinanceiroDecorator.decorate\_collection(@indicadores\_financeiros)

respond\_to do |format|

format.html

format.js

format.turbo\_stream { render turbo\_stream: turbo\_stream.replace('indicadores\_financeiros', partial: 'admin/indicadores\_financeiros/indicador\_financeiro', locals: { indicadores\_financeiros: @indicadores\_financeiros }) }

end

end

def show

@indicador\_financeiro = IndicadorFinanceiro.find(params[:id])

@indicador\_financeiro = IndicadorFinanceiroDecorator.decorate(@indicador\_financeiro)

end

def new

@indicador\_financeiro = IndicadorFinanceiro.new

end

def edit

@indicador\_financeiro = IndicadorFinanceiro.find(params[:id])

end

def create

@indicador\_financeiro = IndicadorFinanceiro.new(indicador\_financeiro\_params)

if @indicador\_financeiro.save

redirect\_to admin\_indicadores\_financeiros\_path, notice: 'Indicador financeiro criado com sucesso'

else

flash.now[:alert] = 'Não foi possível criar o indicador financeiro'

render :new

end

end

def update

@indicador\_financeiro = IndicadorFinanceiro.find(params[:id])

if @indicador\_financeiro.update(indicador\_financeiro\_params)

redirect\_to admin\_indicadores\_financeiros\_path, notice: 'Indicador financeiro atualizado com sucesso'

else

flash.now[:alert] = 'Não foi possível atualizar o indicador financeiro'

render :edit

end

end

def destroy

@indicador\_financeiro = IndicadorFinanceiro.find(params[:id])

@indicador\_financeiro.destroy

respond\_to do |format|

format.html { redirect\_to admin\_indicadores\_financeiros\_path, notice: 'Indicador financeiro removido com sucesso' }

format.turbo\_stream { render turbo\_stream: turbo\_stream.remove(@indicador\_financeiro) }

end

end

private

def indicador\_financeiro\_params

params.require(:indicador\_financeiro).permit(:nome, :descricao, :valor, :data\_referencia, :tipo, :inicio,

:final, :percentual, :acumulado)

end

end

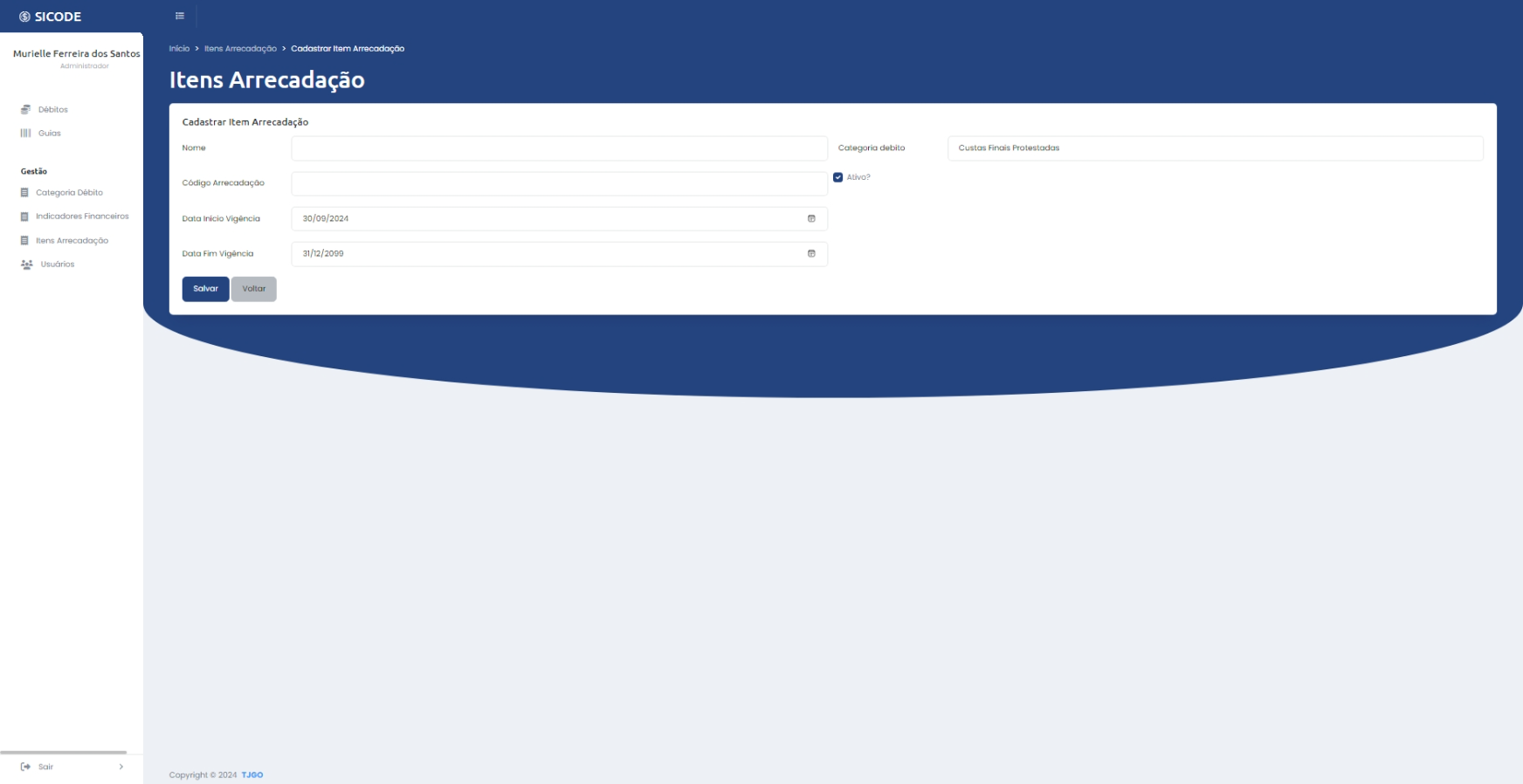
end

# https://projudi.tjgo.jus.br/GerarBoleto?PaginaAtual=2&numeroGuiaConsulta=21816930202

# adicionar numero de guia e numero de processo no debito

# Gestão – Itens de arrecadação

Neste item são visualizados e cadastrados os itens de arrecadação (com seu respectivo código e categoria de débito). É possível editar ou excluir itens já cadastrados.



Código:

module Admin

class ItensArrecadacaoController < Admin::ApplicationController

skip\_before\_action :authenticate\_user!, only: [:autocomplete]

def index

@itens\_arrecadacao = ItemArrecadacao.search(params)

end

def show

@item\_arrecadacao = ItemArrecadacao.find(params[:id])

@item\_arrecadacao = ItemArrecadacaoDecorator.new(@item\_arrecadacao)

end

def new

@item\_arrecadacao = ItemArrecadacao.new

end

def create

@item\_arrecadacao = ItemArrecadacao.new(item\_arrecadacao\_params)

if @item\_arrecadacao.save

redirect\_to admin\_itens\_arrecadacao\_path, notice: 'Item de arrecadação criado com sucesso'

else

render :new

end

end

def edit

@item\_arrecadacao = ItemArrecadacao.find(params[:id])

end

def update

@item\_arrecadacao = ItemArrecadacao.find(params[:id])

if @item\_arrecadacao.update(item\_arrecadacao\_params)

redirect\_to admin\_itens\_arrecadacao\_path, notice: 'Item de arrecadação atualizado com sucesso'

else

render :edit

end

end

def destroy

@item\_arrecadacao = ItemArrecadacao.find(params[:id])

@item\_arrecadacao.destroy

redirect\_to admin\_itens\_arrecadacao\_path, notice: 'Item de arrecadação excluído com sucesso'

end

def autocomplete

@itens\_arrecadacao = ItemArrecadacao.all

@itens\_arrecadacao = @itens\_arrecadacao.where("nome ilike ?", "%#{params[:q]}%") if params[:q].present?

render json: @itens\_arrecadacao.map { |item| { id: item.id, text: item.nome } }

end

private

def item\_arrecadacao\_params

params.require(:item\_arrecadacao).permit(:nome, :codigo, :valor)

end

end

end

**Código das parcelas:**

module Admin

class ParcelasController < Admin::ApplicationController

before\_action :set\_debito

def new

@parcela = @debito.parcelas.build

end

def show

@parcela = @debito.parcelas.find(params[:id])

end

def create

@parcela = @debito.parcelar(data\_primeira\_parcela: parcela\_params[:data\_primeira\_parcela],

numero\_parcelas: parcela\_params[:numero\_parcelas], valor\_primeira\_parcela: parcela\_params[:valor\_primeira\_parcela] )

respond\_to do |format|

format.js

format.html do

if @parcela

redirect\_to admin\_debito\_path(@debito), notice: 'Parcela criada com sucesso'

else

flash.now[:alert] = 'Não foi possível criar a parcela'

redirect\_to admin\_debito\_path(@debito)

end

end

format.turbo\_stream { render turbo\_stream: turbo\_stream.replace('parcelas', partial: 'admin/debitos/parcela', locals: { debito: @debito }) }

end

end

def atualizar

@parcela = @debito.parcelas.find(params[:id])

respond\_to do |format|

format.js

format.html do

if @parcela.atualizar

redirect\_to admin\_debito\_path(@debito), notice: 'Parcela atualizada com sucesso'

else

flash.now[:alert] = 'Não foi possível atualizar a parcela'

redirect\_to admin\_debito\_path(@debito)

end

end

format.turbo\_stream { render turbo\_stream: turbo\_stream.replace('parcelas', partial: 'admin/debitos/parcela', locals: { debito: @debito }) }

end

end

def confirmar

if @debito.confirmar\_parcelamento

redirect\_to admin\_debito\_path(@debito), notice: 'Parcelamento confirmado com sucesso'

else

redirect\_to admin\_debito\_path(@debito), alert: 'Não foi possível confirmar o parcelamento'

end

end

def negar

if @debito.negar\_parcelamento

redirect\_to admin\_debito\_path(@debito), notice: 'Parcelamento negado com sucesso'

else

redirect\_to admin\_debito\_path(@debito), alert: 'Não foi possível negar o parcelamento'

end

end

private

def set\_debito

@debito = Debito.find(params[:debito\_id])

end

def parcela\_params

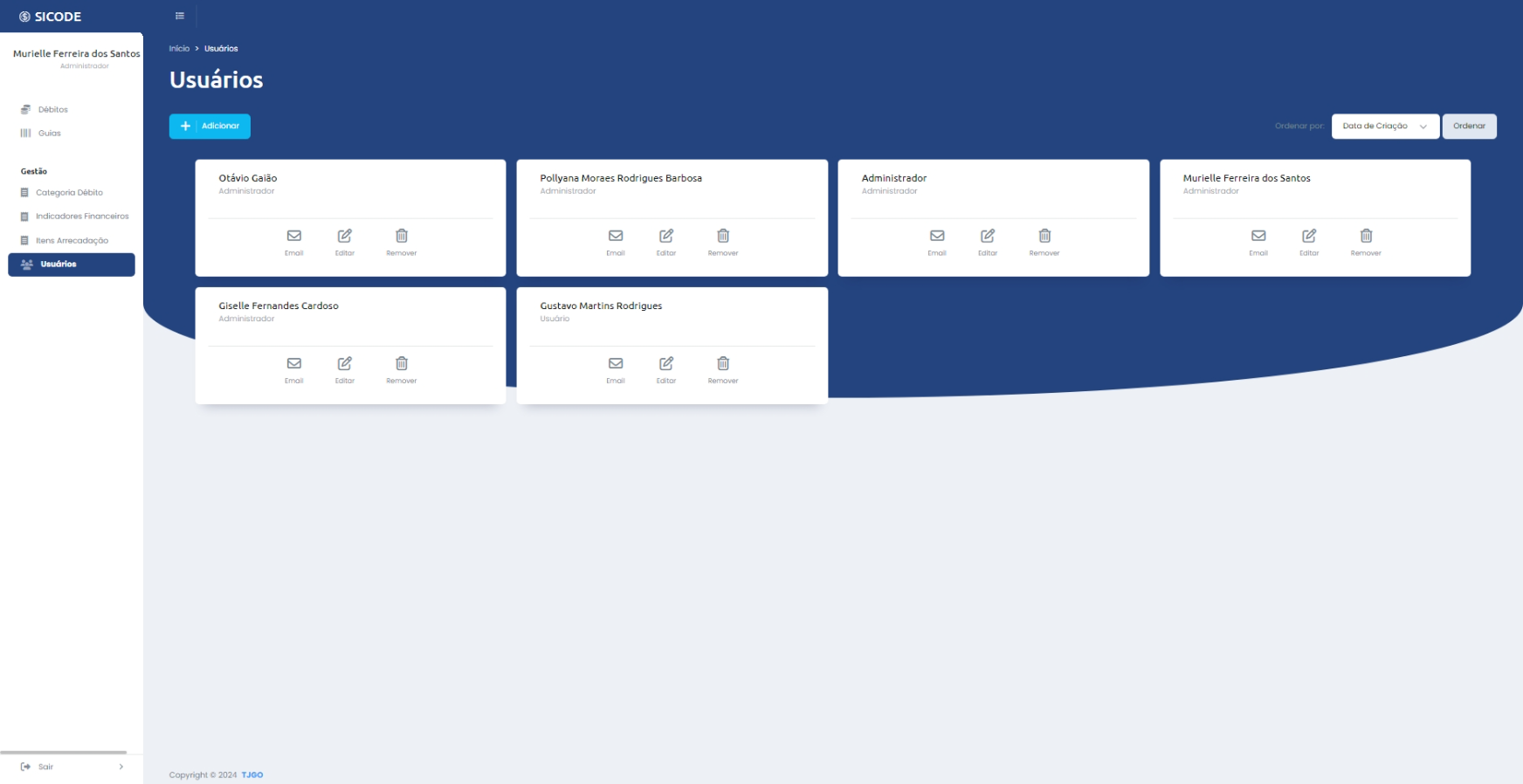
params.permit(:data\_primeira\_parcela, :numero\_parcelas, :valor\_primeira\_parcela)

end

end

end

# Gestão - Usuários

Nesta aba, é possível visualizar todos os usuários cadastrados no SICODE. Além disso, é aqui que podemos realizar novos cadastros, excluir ou editar cadastros já existentes. 

Código:

module Admin

class UsersController < Admin::ApplicationController

def index

@users = User.order(params[:sort\_by] || :id).page(params[:page])

authorize @users

respond\_to do |format|

format.html

format.js

format.turbo\_stream { render turbo\_stream: turbo\_stream.replace('users', partial: 'admin/users/collection', locals: { users: @users }) }

end

end

def show

@user = User.find(params[:id])

authorize @user

end

def new

@user = User.new

authorize @user

end

def create

@user = User.new(user\_params)

authorize @user

if @user.save

redirect\_to admin\_users\_path, notice: 'Usuário criado com sucesso'

else

flash.now[:alert] = 'Não foi possível criar o usuário'

render :new

end

end

def edit

@user = User.find(params[:id])

authorize @user

end

def update

@user = User.find(params[:id])

authorize @user

if @user.update(user\_params)

redirect\_to admin\_users\_path, notice: 'Usuário atualizado com sucesso'

else

flash.now[:alert] = 'Não foi possível atualizar o usuário'

render :edit

end

end

def destroy

@user = User.find(params[:id])

authorize @user

@user.destroy

redirect\_to admin\_users\_path, notice: 'Usuário removido com sucesso'

end

private

def user\_params

params.require(:user).permit(:email, :password, :password\_confirmation, :role, :username, :nome, :cpf, :phone)

end

end

end

# EXTRA

Código: