

- janela de login
- janela de cadastro
- registro de categoria
- registro de empresa

moving box
false

start position
(center)

Janela de login

- groupbox
- labels
- textboxes
- picture box
- buttons

moving box
false

start position
(center)

mudar cor azul

centrelight

Janela cadastro

- Groupbox
- text boxes
- label
- button

Nome:	<input type="text"/>
EMAIL:	<input type="text"/>
SENHA:	<input type="password"/>
REPETIR:	<input type="password"/>
<input type="button" value="CADASTRO"/>	

Registro categoria

- groupbox
- textbox
- label
- groupbox
- button
- data grid view

Categoria:	
NOME DA CAT <input type="text"/>	
<input checked="" type="checkbox"/>	<input type="checkbox"/> EXC <input type="checkbox"/> CAM <input type="checkbox"/> SLO
CAT. REG.	

Registro Empresa

Nome Emp.	
End.	
Telefone	III - + → formatar n. digitar DNP na primeira casa
<input checked="" type="checkbox"/> Cat. regis.	<input type="checkbox"/> Exc <input type="checkbox"/> Com <input checked="" type="checkbox"/> SLV
	" " "

* Função MessageBox: aparece recado na tela

Tipos de variáveis

String = função com palavra

double / int = função com valor numérico

boolean = true / false

* Fazer mensagem aparecer na tela

MessageBox.Show ("mensagem");

↳ seleciona a função  mostra a msg.

* Converter variável

Valor = Convert.ToString (txt1.Text)

ou

converte o s digitado toDouble (txt1.Text)

em s numéricamente.

txt1

é
txt1

de
txt1
var da te

label1.Text = Convert.ToString (valor)

↳ pega o valor numérico e
transforma em alfabeto

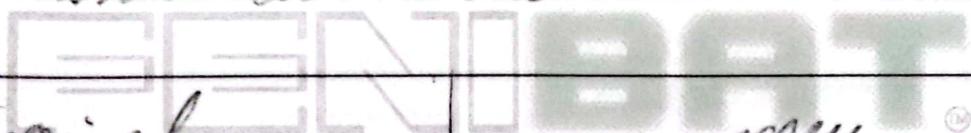
*ordem = if }
else if } { }
else } }
} }
trim () pq é func

*validar campo (ver se inseriram alguma coisa)
if (txtbox.Text.Trim() == "") {
↳ acara a função de tório
do txt box

*truncar cor

label1.ForeColor = Color.red

espelehoamento das cores



original

meu

- Cadastrar	✓	- Login
- login	✓	- cadastrar
- categoria	✓	- categoria
- centro	✓	- empresas
- empresa	✓	
- movimento	✓	
- principal		

Gerenciamento de contas

nome do banco

número da conta

saldo



excluir cancelar sair

contas cadastrar

Movimento.

Realizar Movimento

data [date] hora [time] Empr: " "

tipo : membre box. conta: " "

categoria: " " valor: []

des:

5ª Feira Nacional e Internacional de

Baterias de Chumbo + diferença

Excl. can. salvo

Alugar ou vender

filtrar pelo tipo / membre

[data inicial] [data final]

Alugar

PRINCIPAL

Categorias impressas dentro movimento, na

* menu item strip.

application.exe

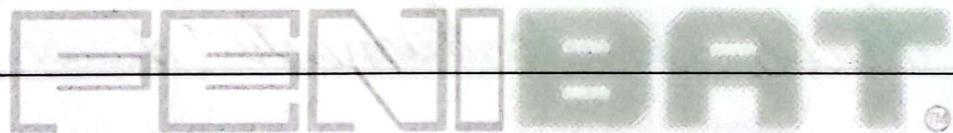
→ para mandar pra categorias *

```
FrmNome VAR1 = new FrmNome();
var1.ShowDialog();
```

LISTA de exercícios

1 - num 1. 2 = \$0 ✓

2 - ✓



3 -

4 - 5a Feira Nacional e Internacional de
Baterias de Chumbo + Conferência

5 -

6 -

7 -

8 -

9 - Peso

10 -

func(15)

func(20)

Funções: Quando você cria determinada função com uma fórmula, o aplica no algoritmo a qualquer momento apenas citando ela e passando os parâmetros.

null = caso a função não retorne nada.

ex: {
 double pere; → define a variável
 pere = convert.ToDouble(txtPere.Text); →
 regrava o input
 reto o parâmetro.
 obgnes.tint = convert.ToString(CalcularAumento(pere));
 → executa a função
 pra acionar.

private double CalcularAumento(double pere)
 {
 double resposta;
 resposta = pere - (pere * 20) / 100;
 → pega
 a info
 aqui { return resposta; → calcula
 → devolve o resultado
 pra método.

TAREAS

- copiar coeduno markington
- trabajos
- exercícios vbacl

3) Tercírio (?)

ai = 8

↑ reno, irre

if ai >= 7 ? aprovado : reprovado;

↳ corre / aprovado

* se a condição for
verdadeira, irá renomear
aquele.Arrays / listas / vetores.

* sempre criar lista antes.

List<double> lstAluno = new List<double>()

declaração de classe

private void CarregarLista()

lstBanc.DataSource = null;

* renderiza a

lstBanc.DataSource = lstNome

lista.

* adicionar algo na lista

private void AddNumber(double x) {

lstNome.Add(x);

Switch (exemplo de filtrar metas) e For

Exemplo:

index do
centro
box

Private Double Nome Função (int excella)

{ int contar = 0; ^{gerenciar novo}

* for (int i = 0; i < lstNome.Count; i++)

^{caso tal coisa, i++ é ~~uma~~ ^{uma} nova}
prox

função bruta

Private Double CentarSituacão (int excella)

{

int contar = 0;

for (int i = 0; i < lstMedias.Count; i++) {

switch (excella)

{

case 0:

if (lstMedias[i] <= 39)

{ contar++; }

break;

case 1:

if (lstMedias[i] > 39 & & lstMedias[i] <= 59)

{ contar++; }

break; }

return contar;

VC IMC

- criar lista
 - validar campo
 - pegar os valores
 - criar função pra calcular IMC
 - criar função pra carregar lista
 - criar função pra calcular situações
- " "

* O que fazer com tal opção da combobox

```
if (cb.SelectedIndex == -1)  
{ valida }
```

l arrem vai.

Criando Classes e objetos

#1. Criar uma classe VO

```
public class NomeVO  
{}
```

```
public string Nome { get; set; }  
ou double
```

* coloquei na classe as informações necessárias

Quando o objeto

List<NomeVO> lstneme = new List<NomeVO>();

no clima

string nome, idade;

double n1, n2;

AlunoVO objaluno;

tips da var nome da var

(onde var jogar a info)

Quando funções com obj:

private NomeVO CriaObj (string nome, etc)

{ AlunoVO objAluno = new AlunoVO(); * declarando
obj. característica 1 = nome
↳ var tal caract de obj ↳ atribuir a ela o parâm
return objAluno }

alterações na grid

-selecten mode = full row select

-multi select = false

-read only = true

-column headers height style mode = auto size.

Quando cumpre de pesquisar retrátil

```
ex: private void PesquisarNome (string Nome)
{ grdNomes.DataSource = null;
if (nome != "") {
    { grdNomes.DataSource = lstAlunos.Where (pesr => pesr.Nome.
        Contains (nome)).ToList();
    }
else { grdNomes.DataSource = lstAlunos; }
}
```

* no clique da text box

```
private void txtNome_TextChanged (object sender,
{ PesquisarNome (txtNome.Text.Trim()); }
}
```

* Adicionar Aluno na lista

```
private void AdicionarNaLista (NomeVO objNome)
{ lstNomes.Add (objNome);
}
```

Filtrar valores de variáveis de tipo Double

Ex: private void Filtrar imc (double imc inicial, double...)

```

    {
        gridName.DataSource = null;           → onde
        gridName.DataSource = lstName.Valor (client => →
        → associado.Cliente.Img >= imc.inicial && cliente.Imc >
        → <= imc.final).ToList();            → acima o valor imc
    }                                     do cliente
                                         ↳ caso verdadeiro: listar.
    
```

Create Read Update Delete

Novidades:

- 1º passo: implementar no fim um Estado Init();

- 2º passo:

private void gridName_CellClick(...)

```

    {
        if (gridName.Rows.Count >= 1)
    }
```

{
 PessoaVO objLinhaCruada;

objLinha... = (PessoaVO) gridName.CurrentRow.DataBound;

txtCedigal.TEXT = objLinhaCruada.Cedigal.ToString();

txtNome.TEXT = objLinhaCruada.Nome;

txtEndereco.TEXT = objLinhaCruada.Endereco;

txtLinha.TEXT = Convert.ToString(c.RowCount);

ativar e desativar os buttons

}

}

- 3º passo: Excluir.

```
private void btnExcluir_Click(object sender, EventArgs e)
```

```
if (MessageBox.Show("Deseja excluir?", "Confirm.", MessageBoxButtons.YesNo) = DialogResult.Yes)
```

```
int pos = Convert.ToInt32(txtLinha.Text)
```

```
lstPessoas.RemoveAt(pos);
```

```
CarregarLista();
```

```
EstoqueInicial();
```

```
}
```

```
}
```

- 4º passo: Alterar

```
private void btnAlterar_Click(object sender, EventArgs e)
```

```
int pos = Convert.ToInt32(txtLinha.Text)
```

```
lstPessoas[pos].Nome = txtNome.Text;
```

```
lstPessoas[pos].Endereco = txtEndereco.Text;
```

```
CarregarLista();
```

```
EstoqueInicial();
```

```
}
```

*método que valida campos

private bool ValidarCampos()

{
bool ret = true;

string campos = " ";

if (txtNome.Text.Trim() == "")

{

ret = false;

campos += "- Nome\n" → quebra a linha
}

if (txtEndereço.Text.Trim() == "")

{

ret = false;

campos += "- Endereço"

}

if (!ret)

{

Messagebox.Show ("Preencher os seguintes campos \n" +

→ + campos, "Atenção"; MessageBoxButtons.OK, MessageBoxIcon.

}

return ret;

Warning

- no clique:

if (ValidarCampos())

{

SQL

* Criar em SQL

Create:

```
insert into tb_Nome  
(id_usuario, nome_categoria)  
values (1, "categoria")  
       ↳ passar as info. estrangeiras  
       necessárias.
```

Read:

```
select * from tb_Nome  
       ↳ everything
```

Update

```
update tb_Nome → onde quero mudar  
set info_usuario = 'Pedro' → o que quero mudar  
where id_usuario = 1 → qual usuario
```

Delete

```
delete from tb_Nome → de qual tabela quero deletar  
where id_Nome = 1 → qual id quero deletar
```

Select avançado

Ex:

Select : nome_banco,
numero_conta,
nome_categoria,
nome_empresa,
data_movimento,
valor_movimento

info's que eu quero

from tb_movimento → sempre relacionar a
table mais ampla peri-
od (mantendo)

↳ tabela de onde
vou tirar uma info
que falta

inner join tb_usuario

on tb_movimento.id_usuario = tb_usuario.id_usuario

inner join tb_conta

on tb_movimento.id_conta = tb_conta.id_conta
↳ atribuir que era info = ↗

inner join tb_categoria

on tb_movimento.id_categoria = tb_categoria.id_cat.

inner join tb_empresa

on tb_movimento.id_empresa = tb_empresa.id_ens.

where tipo_movimento = 0

and tb_movimento.id_usuario in (1, 2)

↳ filtragem mais específica

Procedimentos para linkar o db com o C#

- Criar um projeto chamado DAO (tipos, Net)
- Copiar < connectionSettings> do app.config do db
- Colar no app config do projeto
- Implementar Using DAO na form que for usar
 opcional

(para o compilador startar legado)

- Criar classe chamada util

```
public static class Util {
```

```
    public static int CodigeLegacy = 1;
```

```
}
```

*Brcher IT's
do model

||

||

Ex: em DAO: crie uma classe Empresa DAO

```
public class EmpresaDAO
```

```
{
```

```
    public void Create (tb_empresa objempresa)
```

```
{
```

↳ instancia Banco

```
        Banco objBanco = new Banco();
```

```
        objBanco.Add(tb_empresa objempresa)
```

```
        objBanco.SaveChanges();
```

```
}
```

↓
relação

obj.
o objeto no
Banco

```
{
```

2ª etapa no proj. seg.

Em sua Empresa no projeto original

- Adicionar referência ao DAO
- Implementar Usng DAO
- dentro do clique de reg. na primeira cond:

`if (ValidarCampos())` → instanciar e setar
quem for a info (obj DAO)

`{ EmpresaDAO objDAO = new EmpresaDAO();`

`tb_empresa objEmpresa = new tb_empresa();`

↳ objeto por onde vou acionar os métodos
da tb

atributos }
`objEmpresa.id_usuario = util.GeradorLogar();`
`objEmpresa.nome_empresa = txtNome.Text; trim`
`objEmpresa.endereco_empresa = " " "`
`objEmpresa.telefone_empresa = " " "`

`try` → se der certo

↳ pegue o obj que acabou
de criar na função create

`objDAO.Create(objEmpresa);`

`EstoqueIncial();`

`MessageBox... ("Ação realizada com sucesso")`

`}`

`catch` → se não der certo

`{`

`Message... ("Ocorreu um erro na operação
tente nov. mais tarde")`

`}`

SQL + C#.NET FRAMEWORK

- Exemplo: crud em categoria DAO → parâmetro é um
→ obj de tb-categoria

```
public void Create(tb_categoria objCategoria)
{
    → instance → banco
```

banco objBanco = new banco();

objBanco.AddToTb_categoria(objCategoria)

objBanco.SaveChanges(); → acesso a banco

→ Salva o banco para add o obj
→ tipo → nome func. → parâmetro

```
public List<tb_categoria> Select(int cod_legado)
```

→ instance → banco

banco objBanco = new Banco(); → recebe...

List<tb_categoria> lstCats = objBanco.Tb_categoria.
where(cats => cats.id_usuario == cod_legado).ToList();

→ apelido da tb → cria uma lstCats do tipo tb-ca

return lstCats;

→

→ passa um obj do tipo tb

```
public void Update(tb_categoria objCategoria)
```

banco objBanco = new Banco() → atribuir isso ao obj

tb_categoria objPergata = objBanco.Tb_categoria.Where,

(cat => cat.id_categoria == id_categoria).FirstOrDefault;

objPergata.nome_categoria = objCategoria.nome_cat;

~~public void Delete(int id - categoria)~~

[] → instance o banco

banco obj Banco = new Banco(); → Crie um obj que rec.

tb-categoria obj Resgate = obj Banco.tb_categoria.

where (cat => cat.id - categoria == id - categ).First()

obj Banco.DeleteObject(obj Resgate)

(obj Banco.SaveChanges();

[] → salvar

→ deleta o obj que eu acabei de criar

NA TELA...

Create: - em salvar (btn Salvar)

if (validarCampos())

[] Empresario obj Dao = new EmpresarioDAO()

tb_empresa obj Empresa = novo tb_Empresa()

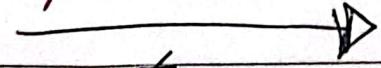
→ instance o banco e o obj.emp.

objEmp.id_usuario = util.adigetLogado;

(objEmp.nome_empresa = txtNome.Text.Trim());

→ atribuir

valores às infes



continua

* autosize coluna made na grid

D S T Q Q S S

```
try
[   if (txtLinha.Text.Trim() == "")  
    [      objDAO.Create(objEmpresa)  
    ]  else  
    [       objEmpresa.id_emp = Convert.ToInt32(txtLinha.  
        objDAO.Update(objEmpresa);  
    ]  EstabeInicial();  
    MessageBox.Show("...")  
]  
catch  
[   MessageBox.Show("...")  
]
```

*(→ se prentura com
estiver selecionada)*

*(→ crie a empresa paranco
o obj)*

*(→ joga o obj da Emp no
Update.)*

Read

```
private void CarregarGrid()  
[   gridEmpresas.DataSource = new EmpDAO().Read(util.cod)  
    gridEmpresas.Columns["id-emp"].Visible = false  
    gridEmpresas.Columns["name-emp"].Header = "Nome"  
]  
(→ altera o header)
```

inst - a classe

↑ acusa essa funç

seleto

(→ para m

private void gridEmp_CellClick

{

if (gridEmp.RowCount > 0)

{ tipo obj de regata tipo }

tb_empr obj LinkLabel = (tb_empresa) →

→ gridEmpresas.CurrentRow.DataBoundItem

txt nome.Text = obj linka clic. nome;

↳ pega a info no campo de teste

btn excluir.Visible = true;

btn Salvar.Text = "Alterar";

}

}

Delete

private void btnExcluir_Click ...

{

↳ mensagem de aviso

if (MessageBox.Show("....")

{ ↳ atribuir info av excl.

int codEmp = Convert.ToInt32(txtLinka.Text.Trim())

Try

{

↳ acusa era funç.

new EmpresDAO().Delete(codEmp)

Estado inicial()

↳ parando era pr.

MessageBox("...")

}

Obj

* o catch é no

Transactions

- Adicionar referência
- Em assemblies → framework → system.Transactions
- Using System.Transactions

- O que é uma transaction scope? É como se fizesse uma "camada" da função que só executa caso a camada num bloco de etapas, se seja, se apenas uma das etapas for salva, o processo não continua, mas se todas as etapas dentro do escopo forem completas, a informação é passada adiante.

Ex: Movimento DAO

→ tipo de mov. → para o obj

public void TrançarMov(tb_mov objMov)

com o par

+ banco objBanco = new Banco();

objBanco.AddToTBMov(objMov)

→ cria um mov. com base no objeto

Using (TransactionScope Trans = new TransactionScope())

+ objBanco.SaveChanges(); → salvo o mov. na banco

tbCenta objContaResgate = objBanco.TbCenta

.Where (conta => conta.idCen == objMov.idCen).First();

new
um
erro

→ especifica qual conta

new. pag

if (objNew.typeNew == 0)

The diagram consists of a horizontal line with seven rectangular boxes underneath it. The boxes are labeled from left to right as D, S, T, Q, Q, S, S.

Obj. Conta Resgate. Saldo Conta + = Obj. Mov. valor

↳ soma à tal Cesta

else

↳ Soma à tal Cenário

1

Subtrai à tal Cesta

7

Ally Baner. Save Changes();

tran. complete \hookrightarrow Salix

3

↳ Se tude dentro delle parentesi dei certi) GG

public void ExclusiveMove(int idMove)

(retorna o diretor à cena)

```
[ Banco elyBanco = new Banco();
```

✓ TB-Movimento objResgate = objBanc. TB-mov.

→ specific qual move where (move \Rightarrow move).
 $id\text{-move} == id\text{-move} \cdot \text{last}$

tb-Conta Conta Rengate = obj Banco . Tb-Conta . where

→ especifica a conta do (Cuenta \Rightarrow Cuenta. idCuenta == merv. Obj Pergate. idCuenta). First

decimal value reserve = obj. Resg. values max;

C. rugata e valver



if (objContaReng.maldoCuenta == 0)

{

ContaReng.Saldo_Cuenta -= valorResarc.

}-

→ tira o \$ da conta

else

{

→ devolve o \$ da conta

ContaReng.Saldo_Cuenta += valorResarc.

}-

→ liga o except.

using (TransactionScope tran = new TransactionScope())

{

→ salva as contas

ObjBanco.SaveChanges(); → delete o mov do

ObjBanco.DeleteObject(objRengate); banco

ObjBanco.SaveChanges() → salva o banco

tran.Complete(); → gg

}

→ parâmetros

public list<MovVO> FiltrarMov(int ced, legado, int tipo)

{

dateTime dtInicial, dateTime final)

banco objBanco = new Banco(); → lista de todos os movs d

list<tb-mov> lstMovsFiltrados = new List<tb-mov>();

list<movimento VO> lstRetorno = new List<mov VO>();

if (tipo == 2) → lista dos movs filtrados

lstMovsFiltrados = objBanco.tb_mov . include("tb-cat")

pegar todos os movs dentro da data
independente do tipo

. include("tb-emp")

. where(mov => mov.id_user == getLog && mov.dtMov

else

[

lstMovsfilt = obj Banco.tb_mov

pega todos os movs dentro da dt de alteração am & tipo selecionado

- include ("tb_cat")
- include ("tb_emp")
- include ("tb-conta")

- where (mov > mov.id_user == wedlog && mov.dataMov >= dataIncial && mov.dataMov <= dataFinal).ToList();

] ↳ && mov.tipo == tipo;

for (int i = 0; i < lstMovsfilt.Count; i++)

[↳ instancia o movVO

MovVO vo = new MovVO();

"propriedades pra exibir na grid"

{ vo.categoria = lstMovs[i].tb_cat.NomeCat
 vo.Centa = lstMovs[i].tbConta.nome_banco + ...
 vo.empresa = lstMovs[i].tb_emp.nome_emp
 vo.Data = lstMovs[i].dataMov.ToString("dd/MM/yyyy")
 vo.tipo = lstMovs[i].TipoMov == 0 ? "Entrada" : "Saida";
 vo.slr = lstMovs[i].slrMov; ↳ if terminal

vo.objMov = lstMovments[i];

list lstRetorno.add(vo)

return lstRetorno ↳ adiciona o obj na lista

return lstRetorno ↳ devolve essa lista

Movimento VO

public class Movimento VO
[

// prop. pra saber na grid.

public string Data [get; set]

public decimal Valor [get; set]

public string Categoria [get; set]

public string Empresa [get; set]

public string Conta [get; set]

public string tipo [get; set]

public string Obs [get; set]

// prop. para guarda a info do obj mov (tbl-mov)

public tb-mov vby Mov [get; set]

]

↳ diz que o que aparece é

public void configurar ~~Categoria~~ ^{Categoria} () {
cb-Categoria.DisplayMember = "nome-categoria";
cb-Categoria.ValueMember = "id-categoria";
}

↳ diz que valor é ↗

private void carregarCategorias () {

CategoriaDAO vby DaoCat = new CategoriaDAO();

List<IB-Cat> lstCats = vby DaoCat.select(util.ectlog);

cb-Categoria.DataSource = lstCats ↗

JAN FEV MAR ABR MAI JUN JUL AGO SET OUT NOV DEZ ANO