

PROZ EDUCAÇÃO
DESENVOLVIMENTO DE SISTEMAS

Bruno Costa
João Victor Camile

LINGUAGENS DE PROGRAMAÇÃO
EMPRESA CBJ

CONTAGEM

2025

SUMÁRIO

1 INTRODUÇÃO	3
2 PRINT DO SISTEMA (DESCRIÇÃO DETALHADA)	4
2.1 TEMPLATES	4
2.1.1 Home	5
2.1.2 Lista	6
2.1.3 Estilo	7
2.2 DATABASE	9
2.2.1 Db.go	9
2.2.2 Script.sql	10
2.3 GO.MOD	11
2.3.1 Go.mod.....	11
2.3.2 Go.sum	12
2.4 MAIN.GO.....	13
1 INTRODUÇÃO	2
1 INTRODUÇÃO	2
1 INTRODUÇÃO	2
1 INTRODUÇÃO	2

1 INTRODUÇÃO

A CBJ é uma empresa inovadora para o desenvolvimento de soluções tecnológicas que otimizam a gestão empresarial. Este trabalho apresenta o desenvolvimento de um sistema web para a leitura e armazenamento de ativos empresariais, com o objetivo de tornar a administração desses recursos mais eficiente e acessível.

A gestão de ativos é um fator crítico para o sucesso operacional de qualquer empresa, pois permite o controle adequado de equipamentos, reduzindo custos, otimizando recursos e prevenindo falhas. Com isso em mente, a CBJ propõe um sistema intuitivo e de fácil implementação, que possibilita o registro, consulta e monitoramento dos ativos de uma empresa.

2 PRINT DO SISTEMA (DESCRIÇÃO DETALHADA)

2.1.1 HOME

A página inicial do sistema web da CBJ foi projetada para oferecer uma experiência intuitiva e eficiente aos usuários. Ao acessar o sistema, o usuário é recebido por uma interface moderna e responsiva, com um layout limpo que facilita a navegação.

```
home.html X
template > home.html > html > body > thead > h1
1  <!DOCTYPE html>
2  <html lang="en">
3  <head>
4    <meta charset="UTF-8">
5    <meta name="viewport" content="width=device-width, initial-scale=1.0">
6    <title>Home</title>
7    <link rel="stylesheet" href="estilo/style.css">
8  </head>
9
10
11
12 <body>
13   <thead>
14
15     <h1>
16       Sistemas de ativos
17     </h1>
18
19     <h2>
20       CBJ
21     </h2>
22
23   <div class="centerbutton">
24     <button type="submit" value="submit">Consultar Ativos</button>
25   </div>
26
27
28
```

```
home.html X
template > home.html > html > body > thead > h1
2  <html lang="en">
3  <head>
4
5  </head>
6
7
8
9
10
11
12 <body>
13   <thead>
14
15     <h1>
16       Sistemas de ativos
17     </h1>
18
19     <h2>
20       CBJ
21     </h2>
22
23   <div class="centerbutton">
24     <button type="submit" value="submit">Consultar Ativos</button>
25   </div>
26
27
28   </thead>
29
30
31 </body>
```

2.1.2 LISTA

A página lista de ativos do sistema web da CBJ foi desenvolvida para fornecer uma visão organizada e detalhada dos equipamentos cadastrados na empresa. Seu objetivo é facilitar a consulta, o monitoramento e a gestão dos ativos, garantindo acesso rápido às informações essenciais.

```

<? lista.html >
template > <? lista.html > html > body > table.table.table-striped.
1 <!DOCTYPE html>
2 <html lang="en">
3 <head>
4 <meta charset="UTF-8">
5 <meta name="viewport" content="width=device-width, initial-scale=1.0">
6 <title>Lista de Ativos</title>
7 <link rel="stylesheet" href="estilo/style.css">
8 </head>
9 <body>
10 <form method="post" action="salvaativo"></form>
11 <table class="table table-striped">
12 <tr>
13 <th>ID</th>
14 <th>Ativo</th>
15 </tr>
16
17 <tbody>
18 {{ range . }}
19 <tr>
20 <td>{{.Id}}</td>
21 <td>{{.Nome}}</td>
22 </tr>
23 {{end}}
24 </tbody>
25 </table>
26
27 <div class="input">
28 <input type="text" name="nome">
29 </div>

```

```

<? lista.html >
template > <? lista.html > html > body > table.table.table-striped. > tr > th
2 <html lang="en">
9 <body>
11 <table class="table table-striped">
17 <tbody>
19 <tr>
22 </tr>
23 {{end}}
24 </tbody>
25 </table>
26
27 <div class="input">
28 <input type="text" name="nome">
29 </div>
30
31
32 <div class="centerbutton">
33 <button type="submit" value="salvar">Adicionar</button>
34 <button type="reset" value="limpar">Remover</button>
35 </div>
36
37 </form>
38
39 </body>
40 </html>

```

2.1.3 ESTILO

O sistema web da CBJ adota uma abordagem moderna e responsiva para o design da interface, utilizando CSS para garantir uma experiência intuitiva e agradável para garantir uma experiência intuitiva e agradável para os usuários. A estrutura de estilo segue os princípios de simplicidade, acessibilidade e eficiência, permitindo fácil manutenção e escalabilidade do sistema.

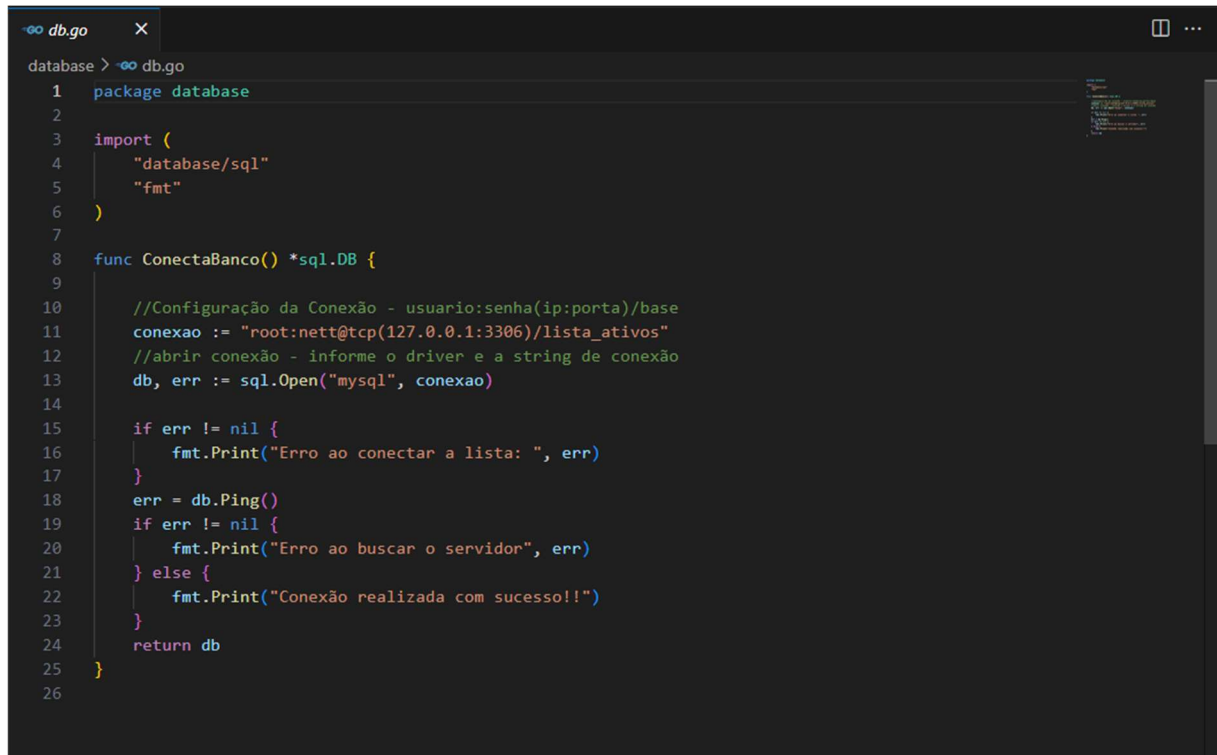
```
# style.css X
template > estilo > # style.css > body
1  body{
2    background-color: rgb(61, 129, 192);
3  }
4
5  h1{
6    text-align: center;
7    color: white;
8    font-size: 80px;
9    font-style: oblique;
10 }
11 h2{
12   text-align: center;
13   color: white;
14   font-size: 80px;
15   font-style: oblique;
16 }
17 button{
18   border-radius: 12px;
19   padding: 10px 26px;
20   text-align: center;
21   font-size: 16px;
22 }
23 button:hover{
24   padding: 15px 39px;
25 }
26 .centerbutton{
27   position: absolute;
28   top: 55%;
29   left: 50%;
```

```
# style.css X
template > estilo > # style.css > body
25 }
26 .centerbutton{
27   position: absolute;
28   top: 55%;
29   left: 50%;
30   -ms-transform: translate(-50%, -50%);
31   transform: translate(-50%, -50%);
32 }
33 .inp A two-dimensional transformation is applied to an element through the 'transform' property. This property contains a
34 list of transform functions similar to those allowed by SVG.
35 (IE 9)
36 -ms-transform: translate(-50%, -50%);
37 transform: translate(-50%, -50%);
38 }
39
40 table{
41   border: 5px solid black;
42   text-align: center;
43   width: 50%;
44   border-radius: 12px;
45   padding: 10px 26px;
46   position: absolute;
47   top: 10%;
48   left: 50%;
49   -ms-transform: translate(-50%, -50%);
50   transform: translate(-50%, -50%);
51 }
52 th{
```

```
# style.css X
template > estilo > # style.css > body
40 table{
41     -ms-transform: translate(-50%, -50%);
42     transform: translate(-50%, -50%);
43 }
44 th{
45     border:3px solid black;
46     text-align: center;
47     width: 50%;
48     border-radius: 12px;
49     padding: 10px 26px;
50     color: white;
51     font-style: oblique;
52     font-size: x-large;
53 }
54 td{
55     border:3px solid black;
56     text-align: center;
57     width: 50%;
58     border-radius: 12px;
59     padding: 10px 26px;
60     color: black;
61     font-style: normal;
62     font-weight: bold;
63     font-size: larger;
64 }
65
66
67
68
69
70
71
72
73
74
75
```

2.2.1 DB.GO

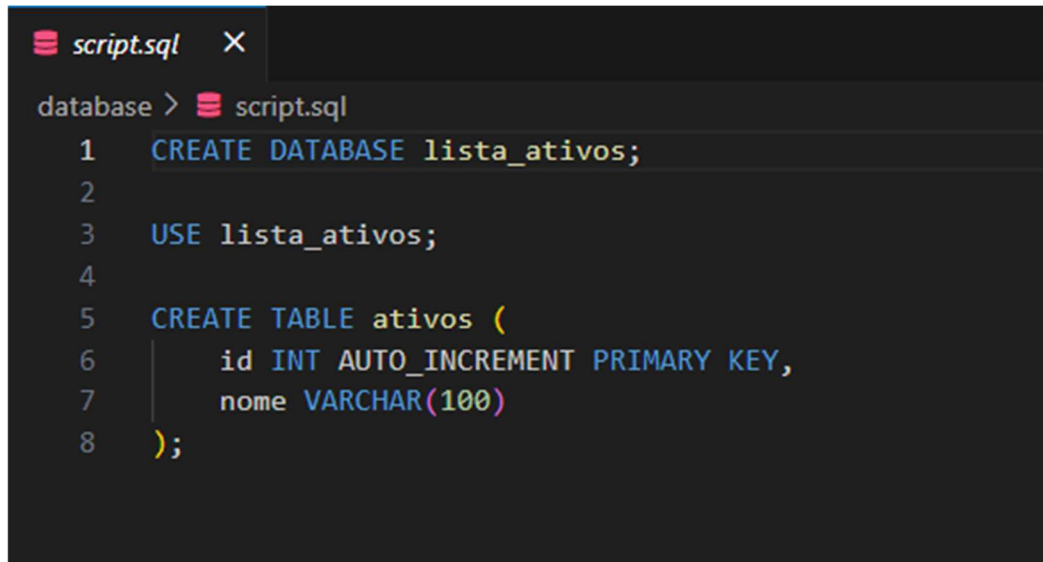
Gerencia a conexão com o banco de dados e executa operações como inserção, consulta, atualização e exclusão de dados relacionados aos ativos empresariais.



```
db.go x
database > db.go
1 package database
2
3 import (
4     "database/sql"
5     "fmt"
6 )
7
8 func ConectaBanco() *sql.DB {
9
10     //Configuração da Conexão - usuario:senha(ip:porta)/base
11     conexao := "root:nett@tcp(127.0.0.1:3306)/lista_ativos"
12     //abrir conexão - informe o driver e a string de conexão
13     db, err := sql.Open("mysql", conexao)
14
15     if err != nil {
16         fmt.Println("Erro ao conectar a lista: ", err)
17     }
18     err = db.Ping()
19     if err != nil {
20         fmt.Println("Erro ao buscar o servidor", err)
21     } else {
22         fmt.Println("Conexão realizada com sucesso!!")
23     }
24     return db
25 }
26
```


2.2.2 SCRIPT.SQL

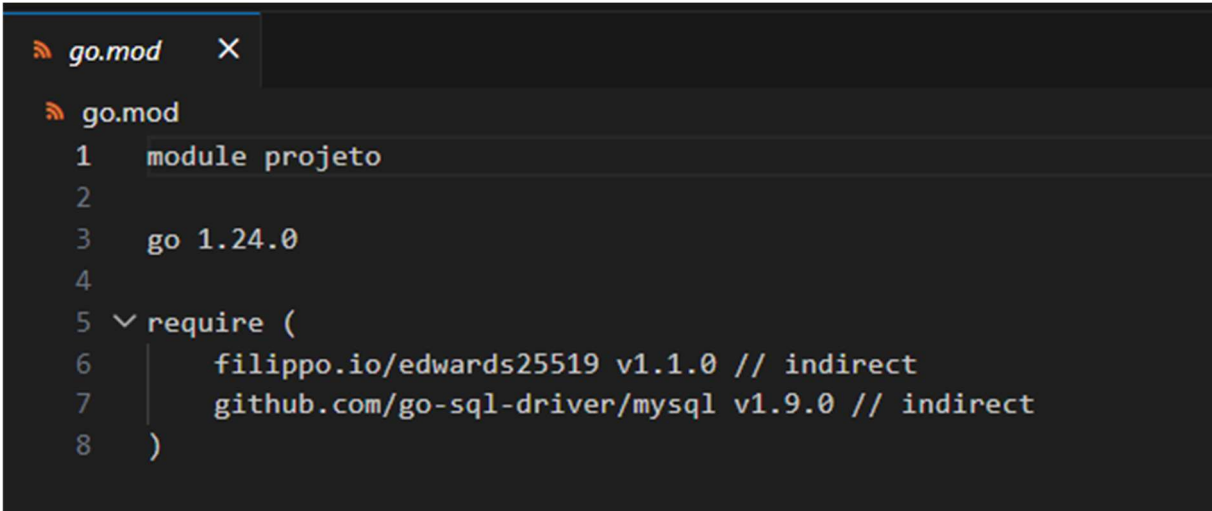
Contém comandos SQL responsáveis por criar e estruturar o banco de dados, além de inserir dados iniciais, caso necessário.



```
script.sql X
database > script.sql
1 CREATE DATABASE lista_ativos;
2
3 USE lista_ativos;
4
5 CREATE TABLE ativos (
6     id INT AUTO_INCREMENT PRIMARY KEY,
7     nome VARCHAR(100)
8 );
```

2.3 GO.MOD

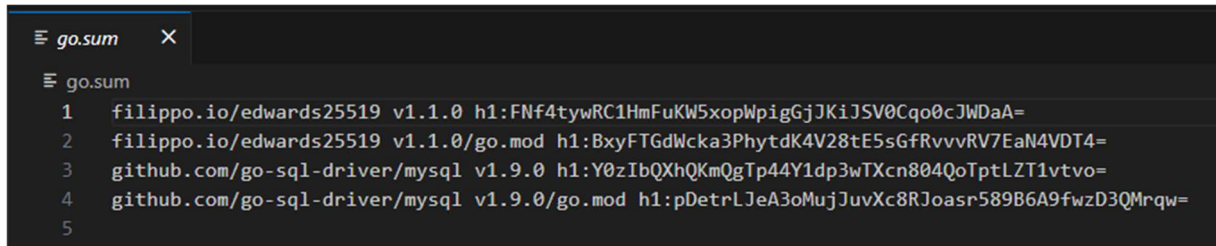
O arquivo `go.mod` é responsável por gerenciar as dependências do seu projeto em Golang . Ele define o nome do módulo e especifica as bibliotecas externas utilizadas, garantindo que todas as versões dos pacotes sejam consistentes.



```
go.mod X
go.mod
1  module projeto
2
3  go 1.24.0
4
5  require (
6      filippo.io/edwards25519 v1.1.0 // indirect
7      github.com/go-sql-driver/mysql v1.9.0 // indirect
8  )
```

2.3 GO.SUM

O arquivo `go.sum` é gerado automaticamente pelo Go para registrar as versões exatas das dependências do seu projeto. Ele funciona como um "comprovante de integridade", garantindo que todas as bibliotecas utilizadas sejam baixadas corretamente e não sofram alterações inesperadas.

A screenshot of a code editor window with a dark theme. The window has a tab labeled 'go.sum' with a close button (X). The editor shows the content of the 'go.sum' file, which lists four dependencies with their exact versions and hashes. The lines are numbered 1 through 5 on the left margin.

```
1 filippo.io/edwards25519 v1.1.0 h1:FNf4tywRC1HmFuKW5xopWpigGjJKiJSV0Cqo0cJWDaA=  
2 filippo.io/edwards25519 v1.1.0/go.mod h1:BxyFTGdWcka3PhytdK4V28tE5sGfRvvvRV7EaN4VDT4=  
3 github.com/go-sql-driver/mysql v1.9.0 h1:Y0zIbQXhQKmQgTp44Y1dp3wTXcn804QoTptLZT1vtvo=  
4 github.com/go-sql-driver/mysql v1.9.0/go.mod h1:pDetrLJeA3oMujJuvXc8RJoasr589B6A9fwzD3QMrgw=  
5
```

2.4 MAIN.GO

O arquivo main.go é o ponto de entrada principal do seu projeto em Golang. Ele contém o código responsável por inicializar a aplicação, configurar dependências e iniciar o servidor web, caso haja um.

```
main.go X
main.go
1 package main
2
3 import (
4     "fmt"
5     "html/template"
6     "net/http"
7     "projeto/database"
8 )
9
10 /*Função para abrir a pagina principal (Home)*/
11 func abreHome(resposta http.ResponseWriter, requisicao *http.Request) {
12     pagina, erro := template.ParseFiles("template/home.html")
13
14     /*Condição para mostrar se ocorreu um erro e o tipo do erro*/
15     if erro != nil {
16         fmt.Println("O erro foi", erro)
17         return
18     }
19     /*Caso nao houver erro a pagina ira executar normalmente*/
20     pagina.Execute(resposta, nil)
21 }
22
23 func abreLista(resposta http.ResponseWriter, requisicao *http.Request) {
24     pagina, erro := template.ParseFiles("template/lista.html")
25
26     if erro != nil {
27         fmt.Println("O erro foi", erro)
28         return
```

```
main.go X
main.go
23 func abreLista(resposta http.ResponseWriter, requisicao *http.Request) {
26     if erro != nil {
27         fmt.Println("O erro foi", erro)
28         return
29     }
30     /*Caso nao houver erro a pagina ira executar normalmente*/
31     pagina.Execute(resposta, nil)
32 }
33
34
35 func salvaAtivo(resposta http.ResponseWriter, requisicao *http.Request) {
36
37     /*Variavel pagina e erro. Carrega o arquivo cadastro.html e se ocorrer
38     algum erro, ele é armazenado na variavel erro.*/
39     pagina, erro := template.ParseFiles("template/lista.html")
40
41     /*Se houver um erro, erro!= nil "erro diferente de um valor nulo",
42     o que foi armazenado na variavel erro sera indicado aqui.*/
43     if erro != nil {
44         fmt.Println("Erro em cadastro", erro)
45         return
46     }
47
48     /*Verificação para armazenar os dados de um cliente.*/
49     if requisicao.Method == http.MethodPost {
50
51         /*Variaveis para pegar dados do cliente como: cep, renda, nome.
52         Baseado nos termos do arquivo html, no caso, cadastro.html.
```

```

main.go X
main.go
35 func salvaAtivo(resposta http.ResponseWriter, requisicao *http.Request) {
49     if requisicao.Method == http.MethodPost {
50
51         /*Variaveis para pegar dados do cliente como: cep, renda, nome.
52         Baseado nos termos do arquivo html, no caso, cadastro.html.
53         Esses termos ("nome", "tel", etc) devem corresponder ao
54         formulario feito no arquivo cadastro.html*/
55         nomeativo := requisicao.FormValue("nome")
56
57         /*Responsavel por imprimir os dados recebidos do usuario*/
58         fmt.Println("O nome do Ativo é: ", nomeativo)
59
60         /*Função para conectar ao banco de dados e armazenar as
61         informações recebidas.*/
62         database.ConectaBanco()
63     }
64     /*Envia a resposta ao usuario. Representa a resposta
65     HTTP (http.ResponseWriter) enviada ao navegador */
66     pagina.Execute(resposta, nil)
67 }
68
69 func main() {
70     //Criando um EndPoint
71     http.HandleFunc("/home", abreHome)
72     http.HandleFunc("/lista", abreLista)
73
74     fmt.Print("Iniciando Servidor!!!")
75
76     erro := http.ListenAndServe(":8080", nil)

```

```

main.go X
main.go
35 func salvaAtivo(resposta http.ResponseWriter, requisicao *http.Request) {
49     if requisicao.Method == http.MethodPost {
50         fmt.Println("O nome do Ativo é: ", nomeativo)
51
52         /*Função para conectar ao banco de dados e armazenar as
53         informações recebidas.*/
54         database.ConectaBanco()
55     }
56     /*Envia a resposta ao usuario. Representa a resposta
57     HTTP (http.ResponseWriter) enviada ao navegador */
58     pagina.Execute(resposta, nil)
59 }
60
61 func main() {
62     //Criando um EndPoint
63     http.HandleFunc("/home", abreHome)
64     http.HandleFunc("/lista", abreLista)
65
66     fmt.Print("Iniciando Servidor!!!")
67
68     erro := http.ListenAndServe(":8080", nil)
69     /*Condição para mostrar se ocorreu um erro e o tipo do erro*/
70     if erro != nil {
71         fmt.Print("Servidor com Problemas")
72     }
73 }
74
75

```