

MANUAL DO DESENVOLVEDOR.

SISTEMA DE INFORMAÇÃO HOME CONTROL
CONTROLE DE AMBIENTES

Porto Velho 13 / 06/ 2016

Introdução

Manual do desenvolvedor do sistema SisHomeControl, nesse manual está todas as informações sobre os procedimentos que o sistema faz para seu perfeito funcionamento, aqui o desenvolvedor pode ter uma base de cada atributo métodos entre outros.

1	CAMADA MODELO.....	4
1.1	Usuário.....	4
1.2	NivelAcesso.	4
1.3	Unidade.....	4
1.4	Divisões.	5
1.5	Dispositivo.....	5
1.6	Controles.....	6
1.7	Tipo de Controle.....	6
1.8	Comando.....	6
1.9	Evento.	7
1.10	Agendamento.....	7
1.11	Históricos.	8
1.12	Leitor.	8
1.13	Leitor.	9
2	CAMADA NEGÓCIO.....	9
2.1	UsuárioBO.	9
2.2	UnidadeBO.	10
2.3	LeitorBO.....	11
2.4	HistoricoBO.....	12
2.5	EventoBO.....	13
2.6	DivisãoBO.....	14
2.7	DispositivoBO.	15
2.8	ControleBO.	16
2.9	ComandoBO.	17
2.10	AgendamentoBO.	18
3	CAMADA DAO.....	18
3.1	UsuárioDAO.	18
3.2	UnidadeDAO.....	19
3.3	LeitorDAO.....	20
3.4	HistoricoDAO.....	20
3.5	HistoricoDAO.....	21
3.6	HistoricoDAO.....	22
3.7	HistoricoDAO.....	22
3.8	ControleDAO.	23
3.9	ComandoDAO.	23
3.10	AgendamentoDAO.....	24

1 CAMADA MODELO

Essa camada está relacionada as abstrações, em que foi feita a análise de requisitos e gerada os modelos principais desse sistema.

1.1 Usuário.

Responsável pela autenticação do usuário.

- **int** id – identificação do usuário.
- **List<Unidade>** objUnidades – lista de unidade do usuário
- **string** login – identificação de acesso do usuário.
- **string** senha – chave para entrar no sistema.
- **string** cpfCnpj – documento do usuário.
- **string** nome – nome do usuário.
- **string** telefone – telefone de contato.
- **string** email – email para contato
- **NivelAcesso** nível – o tipo de cliente, administrador, cliente ou técnico
- **Boolean** ativação – determina se o acesso do usuário está ativo ou desativo.

1.2 NivelAcesso.

É um enum que determina o acesso do usuário.

- **NULL** = 0 – não especificado
- **CLIENTE** = 1 – são do tipo clientes.
- **TECNICO** = 2 – são do tipo técnico.
- **ADMINISTRADOR** = 3 – são do tipo administradores.

1.3 Unidade.

É uma abstração de uma ambiente seja ele uma casa, comercio, indústria, bairro ou qualquer local para cadastrar e fazer a automatização.

- **int** id – identificação da unidade
- **Usuario** objUsuario – identifica para qual usuário essa unidade pertence.

- **List<Divisao>** objsDivisoões – identifica quantas divisões o usuário possuem.
- **string** cep – CEP referente a localização do ambiente.
- **string** numero – número da casa, comercio ou industria.
- **string** endereco – endereço de localização da unidade.
- **string** bairro – bairro da unidade.
- **string** cidade – cidade da unidade.
- **string** estado – estado da unidade.
- **string** pais – pais da unidade.
- **int** tempo – tempo de resposta, refere ao ciclo em que o monitor demora para fazer uma nova pesquisa.
- **string** descricao – descrição da unidade.

1.4 Divisões.

É uma abstração das divisões do ambiente, seja um quarto, cozinha, departamentos, salas, ruas ou bairros entre outros.

- **int** id – identidade das divisões.
- **Unidade** objUnidade – diz de qual unidade pertence essa divisão.
- **List<Dispositivo>** objsDispositivos – diz quais são os dispositivos da divisão
- **string** descricao – é usado para descrever uma unidade.

1.5 Dispositivo.

É uma abstração dos dispositivos que serão manipulados, como uma TV, Lâmpada, Ventiladores entre outros.

- **int** id – representa a identidade do dispositivo.
- **string** nome – descreve o nome do dispositivo.
- **Divisao** objDivisao – diz a qual divisão esse dispositivo pertence.
- **Controle** objControle – diz qual o controle essa divisão utiliza.
- **List<Leitor>** objsLeitores – diz os leitores que esse dispositivo tem.
- **List<Evento>** objsEventos – determina uma lista de eventos que o dispositivo pode gravar.

- `string` porta – representa a porta serial do arduino.
- `int` pinoEntrada – qual o pino que o dispositivo deve utilizar.

1.6 Controles.

Essa classe representa um controlador, é aquele que controla um dispositivo dando antes as opções para o cliente escolher

- `List<Comando>` objsComandos – todo controle tem uma lista de comandos.
- `int` id – identificação do controle.
- `string` equipamento – o tipo de equipamento ou dispositivo que esse controle pertence.
- `string` marca – qual a marca que esse controle pertence.
- `string` modelo – qual o modelo que esse controle pertence.
- `TipoControle` tipocontrole – esse é um enum e nele possui tipos de controle conforme está abaixo.

1.7 Tipo de Controle.

Essa variável determina o tipo de comando que esse controle vai aceitar.

- `NULL = 0` – esse valor quer dizer que o campo ficou em branco.
- `SIMPLES = 1` – esse valor é para comandos simples, faz com que o comando só possa ser 0 desligado e 1 ligado e dois que diz que se tiver ligado ele desliga e se tiver desligado ele liga.
- `LOGICO = 2` – esse valor diz que o comando é números hexadecimais utilizado em controle remoto.
- `GRADUAL = 3` – esse valor é do tipo gradual aceitando inteiro de 1 a 100 determinado a intensidade do comando no arduino ele reconhece como 0 a 5 volts.
- `ESPECIFICO = 4` – esse comando deve ser codificado no arduino e pode enviar qualquer caractere.

1.8 Comando

Essa classe possui atributos que;

- **int** id – identificação do comando.
- **string** cmd – comando que é enviado para o arduino.
- **string** nome – descrição do comando.
- **string** estilo – refere a classe do Búfo, a personalização mostrada para o usuário.
- **string** cor – refere a cor do Búfo que será mostrado para o usuário.
- **int** consumoEnergia – é referente ao consumo que o comando dado para o dispositivo gasta.
- **int** consumoAgua – é referente ao consumo que o comando dado para o dispositivo gasta.

1.9 Evento.

Essa classe representa uma solicitação dada para um dispositivo.

- **int** id – identidade do evento
- **Comando** objComando – grava o comando selecionado pelo cliente.
- **Dispositivo** objDispositivo – grava qual o dispositivo receberá o objComando (Comando selecionado)
- **Agenda** objAgenda – Determina que o comando será um agendamento e será solicitado em algum momento determinado.
- **string** potencia – determina que o evento recebe um comando de intensidade variada de 0 a 100.

1.10 Agendamento.

Essa classe diz que se o evento tiver agendamento ele é uma solicitação agendada.

- **int** id – identificador
- **TimeSpan** hora – determina em qual hora o evento entrará em ação.
- **bool** domingo – diz se o evento será efetuada aos domingos.
- **bool** segunda – diz se o evento será efetuada nas segundas.
- **bool** terca - diz se o evento será efetuada nas terças.
- **bool** quarta - diz se o evento será efetuada nas quartas.
- **bool** quinta - diz se o evento será efetuada nas quintas

- **bool** sexta - diz se o evento será efetuada nas sextas.
- **bool** sabado - diz se o evento será efetuada aos sábados.

1.11 Históricos.

Essa classe grava os logs, para fazer o relatório de consumo.

- **int** id – identificador.
- **string** descrição – qual a descrição do histórico dados pelo sistema.
- **Comando** objComando – qual foi o comando executado.
- **Dispositivo** objDispositivo – em qual dispositivo o comando foi solicitado.
- **DateTime** momento – em que momento foi dado o comando.
- **double** consumoEnergia – qual foi o valor do consumo de energia.
- **double** consumoAgua – qual foi o valor do consumo de água.

1.12 Leitor.

Nessa classe, está relacionado a leitura de ações praticadas no dispositivo do cliente, e toda vez que a condição é validada um histórico é gravado.

- **int** id – identificação do leitor.
- **string** nome – nome do leitor
- **int** pinoSaida – qual o pino do arduino é executado a leitura.
- **string** porta; - em qual porta do arduino vai ser executado a leitura.
- **decimal** sensibilidade – qual a sensibilidade do dispositivo da leitura, essa sensibilidade é utilizada para amperímetros que tem por padrão fixos.
- **Condicao** condicao – nesse caso o técnico determina uma condição, se caso ela for validade perante o resultado é gravado um histórico.
- **TipoLeitor** tipoLeitor – qual é o tipo de leitor
- **Dispositivo** objDispositivo – a qual dispositivo esse leitor pertence
- **Comando** objComando – qual comando se o leitor for validade deve ser gravado no histórico.
- **decimal** valor – esse valor deve ser estipulado para ser comparado com condição e resultado.
- **decimal** resultado – o arduino grava o resultado aqui.

- `bool` primeiraLeitura – determina se é a primeira leitura do arduino, pois ele não grava a primeira leitura.

1.13 Leitor.

Esse enum está relacionado ao tipo de leitura que será feita no dispositivo, ela é identificada no arduino.

- `NULL = 0` - sem representação.
- `OPTOACOPLADOR = 5` – fazer leitura de opto acopladores.
- `AMPERIMETRO = 6` – fazer leitura de amperímetros.

2 CAMADA NEGÓCIO.

Essa camada está relacionada a parte em que foi definido as regra de negocio, por onde passa todos os processamento de dados.

2.1 UsuárioBO.

Essa classe está relacionado a buscas, gravações, alterações e as validações do usuário no sistema.

- `public string` Gravar(`Usuario` obj) – esse método grava um usuário.
- `public string` Alterar(`Usuario` obj) – esse método altera um usuário.
- `public string` Deletar(`Usuario` obj) – esse método deleta um usuário.
- `public Usuario` BuscarId(`int` id) – esse método busca um usuário pelo seu identificador.
- `public Usuario` BuscarId(`Usuario` obj) – esse método busca um usuário por objeto usuário.
- `public List<Usuario>` BuscarGeral(`string` valor) – esse método busca todos os usuário relacionado a alguma pesquisa.
- `public List<Usuario>` BuscarLogin(`string` valor) – esse método busca o usuário pelo nome de login.
- `public Usuario` BuscarUnidadesUsuario(`Usuario` obj) – esse método busca todas unidades que pertence ao usuário.
- `public List<Usuario>` BuscarGeralClientes(`string` valor) – busca apenas todos os clientes relacionado a uma pesquisa.

- `public bool ValidarSenha(string senha1, string senha2)` – faz a validação das senhas do formulário.
- `public bool ValidarCamposObrigatorio(Usuario obj)` – confere se todos os campos do usuário foi preenchido corretamente.
- `public bool ValidarCpf(Usuario obj)` – verifica se o CPF do usuário é válido.
- `public bool ValidarMatricula(Usuario obj)` – verifica se o usuário está cadastrado no sistema.
- `public Usuario Logar(Usuario obj)` – faz autenticação do usuário no sistema.
- `public Unidade LogarUnidade(Usuario objUsuario, Unidade objUnidade)` – autentica a unidade do usuário no sistema.

2.2 UnidadeBO

Essa classe está relacionado a todas regra de negócio da unidade.

- `public string Gravar(Unidade objUnidade)` – esse método grava a unidade.
- `public string Alterar(Unidade objUnidade)` – esse método altera unidade.
- `public string Deletar(Unidade objUnidade)` – esse método deleta unidade.
- `public Unidade BuscarId(int id)` – esse método busca unidade por identificado.
- `public Unidade BuscarId(Unidade objUnidade)` – esse método busca unidade por objeto unidade.
- `public List<Unidade> BuscarGeral(string valor)` – esse método busca todas unidade por uma pesquisa específica.
- `public List<Dispositivo> BuscarLeitoresUnidade(Unidade objUnidade)` – esse método busca todos leitores que pertence a unidade.
- `public List<Evento> BuscarEventosUnidade(Unidade obj)` – esse método busca todos eventos que pertence a unidade.

- `public Usuario` BuscarUnidadesUsuario(`Usuario` obj) – esse método busca todas unidade que petence ao usuário.
- `public Unidade` BuscarUnidadeldBotao(`string` valor) – buscar unidade por id do Botão selecionado.
- `public Unidade` BuscarDivisoiesUnidade (`Unidade` obj) – buscar todas divisões que pertencem a unidade.
- `public bool` ValidarCamposObrigatorio(`Unidade` obj) – validar se o objeto unidade foi preenchido corretamente.
- `public bool` ValidarMatricula(`Unidade` obj) – verifica a unidade existe no sistema.
- `public bool` VerificarDivisaoUnidade(`Unidade` obj) – verifica se a divisão está cadastrada.
- `public bool` VerificarUnidadeUsuario(`Usuario` obj) – verifica se a unidade do usuário está cadastrada.

2.3 LeitorBO.

Esse método corresponde a gravar os leitores que pertencem a algum dispositivo, dessa forma toda vez que acontecer algum evento no dispositivo ele grava um histórico dizendo que o dispositivo sofreu alguma ação.

- `public string` Gravar(`Leitor` obj) – gravar um leitor.
- `public string` Alterar(`Leitor` obj) – alterar um leitor.
- `public List<Leitor>` BuscarTodos() – buscar todos leitores.
- `public List<Leitor>` BuscarGeral(`string` valor) – buscar todos leitores por uma pesquisa específica.
- `public bool` ValidarCamposObrigatorio(`Leitor` obj) – validar se todos os campos do leitor foi preenchido corretamente.
- `public bool` ValidarMatricula(`Leitor` obj) – valida se o leitor existe no sistema.
- `public string` Deletar(`Leitor` obj) – deleta o leitor.
- `public Leitor` BuscarId(`int` valor) busca um leitor por identificação.
- `public Leitor` BuscarId(`Leitor` obj) – busca o leitor através de um objeto do tipo leitor.

- `public bool ValidarLeitor(Leitor obj)` – fazer a validação do objeto leitor, e verificar se está preenchido corretamente.
- `public bool ValidarDivisao(Leitor obj)` – fazer a validação da divisão, e verificar se está preenchido corretamente.
- `public List<Leitor> BuscarLeitoresDispositivo(Dispositivo objDispositivo)` – buscar todos os leitores que pertencem a dispositivo.
- `public bool VerificarLeitorDispositivo(Dispositivo obj)` – verificar se o leitor do dispositivo está devidamente cadastrado no sistema.

2.4 HistoricoBO.

Esse método corresponde a gravação de logs e geração de relatórios apresentado através de gráficos para informar o consumo ao usuário.

- `public bool Gravar(Historico objHistorico)` – esse método faz a gravação do histórico para o dispositivo
- `public Historico BuscarId(Historico objHistorico)` – esse método busca um histórico por identificador.
- `public List<Historico> BuscarHistoricoDispositivo(Dispositivo obj)` – esse método busca todos históricos do dispositivo.
- `public List<Historico> BuscarHistorico(Dispositivo obj, DateTime valor)`
- - esse método busca um histórico do dispositivo a partir de uma data.
- `public string[,] ConstruirGraficoEnergiaDispositivo(Dispositivo obj, DateTime valor)` – esse método constrói o gráfico de energia conforme os logs que foram gravados.
- `public string[,] ConstruirGraficoAguaDispositivo(Dispositivo obj, DateTime valor)` – esse método constrói os gráficos de água conforme os logs gravados.
- `public DateTime BuscarPrimeiroDia(DateTime primeiroDiaSemana)` – esse método busca partir de um dia o primeiro dia da semana.

- `public string[,]` MontarGraficoNull(`DateTime` dataInicial, `string` unidade) – esse método monta o gráfico se o valor dado for igual a nullo.
- `public string[,]` MontarGrafico(`double[]` dados, `DateTime` dataInicial, `string` unidade) – esse método monta a parte visual do gráfico.
- `public bool` VerificarHistoricoDispositivo(`Dispositivo` obj) – esse método verifica se o histórico do dispositivo existe no sistema.
- `public double` BuscarMaiorValor(`double[]` valores) – esse método pega o valor dos dados do gráfico o maior.
- `public string` ConvertData(`DateTime` valor) – converte data em caracteres do tipo string.
- `public double` CalcularTotal(`string[,]` dados) – executa o cálculo total dos registros do dia da semana.
- `public Historico` BuscarUltimoHistoricoDispositivo(`Dispositivo` obj) – busca o registro do dispositivo que foi adicionado por último.

2.5 EventoBO.

Esse método corresponde a uma solicitação que será gravada para um usuário dizendo o que o dispositivo deve fazer.

- `public bool` GravarEventoMemoria(`Evento` obj) – grava um evento na memória.
- `public bool` DeletarEventoMemoria(`Evento` obj) – deleta o evento da memória.
- `public Evento` BuscarEventoMemoria(`Unidade` obj) – busca os eventos na memória do computador que pertencem a uma unidade.
- `public List<Dispositivo>` BuscarEventosUsuario(`Usuario` obj) – busca uma lista de dispositivos que pertencem a um usuário.
- `public string` GravarEventoBanco(`Evento` obj) – grava o evento no banco de dados.
- `public string` AlterarEventoBanco(`Evento` obj) – altera o evento no banco de dados.

- `public string` Deletar(`Evento` obj) – deleta o evento que foi especificado.
- `public string` DeletarTodosEventosDispositivo(`Dispositivo` obj) – deleta todos eventos agendado do dispositivo.
- `public Evento` BuscarEventoBancold(`Evento` obj) – busca os eventos por evento.
- `public Evento` BuscarEventoldBotao(`string` valor) – busca os eventos gravado no banco pelo id do Búñ.
- `public Dispositivo` BuscarEventoDispositivo(`Dispositivo` obj) – busca todos eventos do dispositivo.
- `public bool` ValidarMatricula(`Evento` obj) – verifica se ã possui algum evento cadastrado com tal identifiador.
- `public bool` ValidarEvento(`Evento` obj) – verifica se o evento realmente existe.
- `public bool` ValidarAgendamento(`Evento` obj) – esse método verifica se algum dia da semana foi preenchido.
- `public bool` VerificarEventoDispositivo(`Dispositivo` obj) – esse método verifica se existe algum evento para esse dispositivo.

2.6 DivisãoBO.

Esse método corresponde as divisões que uma unidade possui.

- `public string` Gravar(`Divisao` objDivisao) – gravar uma nova divisão.
- `public string` Alterar(`Divisao` objDivisao) – alterar a divisão.
- `public List<Divisao>` BuscarTodos() – buscar todas as divisões.
- `public List<Divisao>` BuscarGeral(`string` valor) – buscar todas as divisões correspondente a um valor.
- `public bool` ValidarCamposObrigatorio(`Divisao` obj) – verificar se os campos da divisão foram preenchidos corretamente.
- `public bool` ValidarMatricula(`Divisao` obj) – verificar se a matricula da divisão existe.
- `public string` Deletar(`Divisao` objDivisao) – deletar uma divisão.
- `public Divisao` BuscarId(`int` valor) – buscar divisão por identificador.

- `public Divisao BuscarId(Divisao objDivisao)` – buscar uma divisão por divisão.
- `public Unidade BuscarDivisoesUnidade(Unidade obj)` – buscar todas as divisões que pertencem a unidade.
- `public Divisao BuscarDispositivosDivisao(Divisao obj)` – busca a divisivos que pertencem a divisão.
- `public Divisao BuscarDivisaoidBotao(string valor)` – esse método busca a divisão por id do botão.
- `public bool VerificarDispositivosDivisao(Divisao obj)` – esse método verifica se a divisão possui dispositivo.

2.7 DispositivoBO.

Esse método corresponde aos dispositivos que são gravados, alterados, excluídos, consultados e validados.

- `public string Gravar(Dispositivo obj)` – gravar um dispositivo.
- `public string Alterar(Dispositivo obj)` – alterar um dispositivo.
- `public string Deletar(Dispositivo obj)` – deletar um dispositivo.
- `public Dispositivo BuscarId(Dispositivo objDispositivo)` – um dispositivo por dispositivo.
- `public Dispositivo BuscarId(int valor)` – buscar o dispositivo por um identificador.
- `public List<Dispositivo> BuscarTodos()` – buscar todos os dispositivos.
- `public Dispositivo BuscarDivisao(Dispositivo obj)` – buscar a divisão do dispositivo.
- `public Dispositivo BuscarControle(Dispositivo obj)` – buscar o controle do dispositivo.
- `public bool ValidarCamposObrigatorio(Dispositivo obj)` – verificar se todos os campos obrigatórios foram preenchidos.
- `public bool ValidarControle(Dispositivo obj)` – validar se o controle informado para esse dispositivo existe.

- `public bool ValidarMatricula(Dispositivo obj)` – verificar se o identificador do dispositivo já está cadastrado no banco.
- `public bool ValidarDispositivo(Dispositivo obj)` – verificar se o dispositivo já está cadastrado no banco.
- `public List<Dispositivo> BuscarGeral(string valor)` – buscar todos os dispositivos cadastrado no sistema.
- `public Divisao BuscarDispositivosDivisao(Divisao obj)` – buscar um dispositivo por objeto da divisão.
- `public Dispositivo BuscarDispositivoIdBotao(string valor)` – buscar um dispositivo por valor do id de botão.

2.8 ControleBO.

Esse método corresponde aos controles que serão gravados, alterados, excluídos, consultados e validados.

- `public string Gravar(Controle obj)` – grava um novo controle
- `public string Alterar(Controle obj)` – altera o controle.
- `public string Deletar(Controle obj)` – deleta um controle
- `public Controle BuscarId(int valor)` – busca controle por identificador.
- `public Controle BuscarId(Controle obj)` – busca controle por objeto.
- `public List<Controle> BuscarGeral(string valor)` – busca todos os controles.
- `public Controle BuscarComandosControle(Controle obj)` – busca todos os comandos que pertencem ao controle
- `public bool ValidarControle(Controle obj)` – verifica se o controle existe no banco.
- `public Comando BuscarComandoIdBotao(string valor)` – busca um comando pelo id do botão.
- `public bool GravarEvento(Evento objEvento)` – grava um evento.

- `public string` ToComandoHistorico(`string` p)- formata comando portencia colocando o sinal de porcentagem.
- `public bool` VerificarComandoControle(`Controle` obj) – verifica se o controle tem algum comando.
- `public bool` VerificarDispositivoControle(`Controle` obj) – verifica se o controle tem algum dispositivo

2.9 ComandoBO.

Esse método corresponde aos comandos que são gravados, alterados, excluídos, consultados e validados.

- `public string` Gravar(`Comando` obj) – grava o comando.
- `public string` Alterar(`Comando` obj) – altera o comando.
- `public string` Deletar(`Comando` obj) – deleta o comando.
- `public Comando` BuscarId(`int` valor) – busca comando por identificador.
- `public Comando` BuscarId(`Comando` obj) – busca os comando por comando.
- `public List<Comando>` BuscarTodos() – busca todos os comandos.
- `public List<Comando>` BuscarComandoControle(`Controle` obj) – busca todos os comandos do controle.
- `public Comando` BuscarComandoldBotao(`string` valor) – busca o comando por id do botão.
- `public bool` ValidarCamposObrigatorio(`Comando` obj) – faz as validações dos campos preenchidos antes de gravar.
- `public bool` ValidarComando(`Comando` obj) – verifica se existe algum comando cadastrado
- `public bool` VerificarHistoricoComando(`Comando` obj) – verifica se o comando possui histórico.
- `public bool` VerificarEventoComando(`Comando` obj)- verifica se o evento possui comando.
- `public bool` VerificarLeitorComando(`Comando` obj) – verifica se o comando possui algum leitor.

2.10 AgendamentoBO.

Esse método corresponde aos comandos que serão gravados, alterados, excluídos, consultados e validados.

- `public int` Gravar(`Agenda` obj) – grava um novo agendamento.
- `public string` Alterar(`Agenda` obj) – altera agendamento.
- `public void` Deletar(`Agenda` obj) – deleta agendamento.
- `public Agenda` BuscarId(`int` valor) busca agendamento por identificador.
- `public Agenda` BuscarId(`Agenda` obj) – busca agendamento por agendamento.

3 CAMADA DAO

Essa camada corresponde a interação com banco de dados, onde o sistema o armazenamento da CAMADA MODELO.

3.1 UsuárioDAO

Essa classe faz o controle de armazenamento do usuário no banco de dados

- `public string` Gravar(`Usuario` obj) – método grava no banco de dados o usuário.
- `public string` Alterar(`Usuario` obj) – método que altera um usuário no banco.
- `public string` Deletar(`Usuario` obj) – método que deleta um valor do banco.
- `public Usuario` BuscarId(`int` valor) – método que busca um usuário no banco por id.
- `public Usuario` BuscarId(`Usuario` obj) – método que busca um usuário no banco por objeto usuário.
- `public Usuario` BuscarLogin(`Usuario` obj) – método que busca usuário por login.
- `public Usuario` BuscarCpf(`Usuario` obj) – método que busca usuário por CPF.

- `public List<Usuario> BuscarTodos()` – método que busca todos os usuários no banco
- `public List<Usuario> BuscarGeralClientes(string valor)` – método que busca todos os clientes por um valor relacionado.
- `public List<Usuario> BuscarGeral(string valor)` – método que busca todos usuários no banco de dados.
- `public Usuario BuscarUnidadeUsuario(Usuario obj)` – método que busca todas as unidades do usuário.

3.2 UnidadeDAO

Essa classe faz o controle de armazenamento da unidade no banco de dados

- `public string Gravar(Unidade obj)` – grava no banco de dados uma unidade.
- `public string Alterar(Unidade obj)` – altera no banco de dados uma unidade.
- `public string Deletar(Unidade obj)` – deleta do banco de dados uma unidade.
- `public Unidade BuscarId(int valor)` – busca uma unidade no banco de dados por identificador.
- `public Unidade BuscarId(Unidade obj)` – busca uma unidade por objeto do tipo unidade no banco de dados.
- `public List<Unidade> BuscarTodos(string valor)` – busca todas as unidades do banco de dados.
- `public Unidade BuscarDivisoesUnidade(Unidade obj)` – busca todas as divisões que pertencem a unidade.
- `public List<Unidade> BuscarUnidadesUsuario(Usuario obj)` – busca todas as unidades que pertencem a um usuário.
- `public bool VerificarUnidadeUsuario(Usuario obj)` – verifica se a unidade possui usuário.
- `public bool VerificarDivisaoUnidade(Unidade obj)` – verifica se a unidade possui divisão.

3.3 LeitorDAO.

Essa classe é responsável por manter o leitor no banco de dados.

- `public string Gravar(Leitor obj)` – grava um leitor no banco de dados.

- `public string` Alterar(`Leitor` obj) – altera um leitor no banco de dados.
- `public string` Deletar(`Leitor` obj) – deleta um leitor no banco de dados.
- `public Leitor` BuscarId(`int` valor)- busca um leitor por identificador.
- `public Leitor` BuscarId(`Leitor` obj) – busca um leitor por leitor.
- `public bool` VerificarLeitorDispositivo(`Dispositivo` obj) – verificar se o dispositivo tem leitor gravado nele.
- `public List<Leitor>` BuscarTodos() – busca todos leitores.
- `public List<Leitor>` BuscarLeitoresDispositivo(`Dispositivo` obj) – busca os leitores do dispositivo.

3.4 HistoricoDAO.

Essa classe é responsável por gravar no banco de dados.

- `public bool` Gravar(`Historico` obj) - grava histórico no banco de dados.
- `public Historico` BuscarId(`int` valor) – busca histórico por identificador.
- `public Historico` BuscarId(`Historico` obj) – busca histórico por objeto histórico.
- `public List<Historico>` BuscarHistoricoDispositivo(`Dispositivo` obj) – busca todos historicos do dispositivo no banco de dados.
- `public List<Historico>` BuscarHistoricoDispositivo(`Dispositivo` obj, `DateTime` dataInicial, `DateTime` dataFinal) – faz consulta do histórico do dispositivo por período.
- `public bool` VerificarHistoricoDispositivo(`Dispositivo` obj) – verificar se o dispositivo tem histórico gravado.
- `public Historico` BuscarUltimoHistoricoDispositivo(`Dispositivo` obj) – buscar o ultimo histórico do dispositivo no banco de dados.

3.5 HistoricoDAO.

Essa classe é responsável por gravar históricos no banco de dados.

- `public string` Gravar(`Evento` obj) – grava um evento no banco de dados.
- `public string` Alterar(`Evento` obj) – altera o evento do banco de dados.
- `public string` Deletar(`Evento` obj) – deleta o evento do banco de dados.
- `public string` DeletarTodosEventosDispositivo(`Dispositivo` obj) – deleta todos os eventos do dispositivo.
- `public Evento` BuscarId(`int` valor) – busca os eventos por identificadores.
- `public Evento` BuscarId(`Evento` obj) – busca todos os eventos por objeto evento.
- `public List<Evento>` BuscarEventoBanco(`Dispositivo` obj) – busca todos os eventos do banco de dados que pertencem ao dispositivo.
- `public Dispositivo` BuscarEventosDispositivo(`Dispositivo` obj) – busca todos os eventos do dispositivo.
- `public IList<Evento>` BuscarEventosUsuario(`Usuario` obj) – busca todos os eventos que pertencem ao usuário.
- `public bool` VerificarEventoDispositivo(`Dispositivo` obj) – verifica se o dispositivo tem evento.

3.6 HistoricoDAO.

Essa classe é responsável por gravar históricos no banco de dados.

- `public string` Gravar(`Divisao` obj) – grava uma divisão no banco de dados.
- `public string` Alterar(`Divisao` obj) – altera uma divisão no banco de dados.
- `public string` Deletar(`Divisao` obj) – deleta uma divisão do banco de dados.
- `public Divisao` BuscarId(`int` valor) – busca a divisão por identificador.

- `public Divisao` BuscarId(`Divisao` obj) – busca a divisão por objeto divisão.
- `public List<Divisao>` BuscarTodos() – busca todas as divisões cadastrada no banco de dados
- `public List<Divisao>` BuscarDivisoesUnidade(`Unidade` obj) – busca todas divisões da unidade no banco de dados.
- `public Divisao` BuscarDispositivosDivisao(`Divisao` obj) – busca todos dispositivos da divisão
- `public bool` VerificarDispositivosDivisao(`Divisao` obj) – verifica se a divisão tem dispositivo.

3.7 HistoricoDAO.

Essa classe é responsável por gravar históricos no banco de dados.

- `public string` Gravar(`Dispositivo` obj) – grava um dispositivo no banco de dados.
- `public string` Alterar(`Dispositivo` obj) – altera um dispositivo no banco de dados.
- `public string` Deletar(`Dispositivo` obj) – deleta um dispositivo no banco de dados
- `public Dispositivo` BuscarId(`int` valor) – busca um dispositivo por identificador.
- `public Dispositivo` BuscarId(`Dispositivo` obj) – dispositivo do tipo objeto dispositivo.
- `public List<Dispositivo>` BuscarTodos() – buscar todos os dispositivos no banco de dados.
- `public List<Dispositivo>` BuscarDispositivosDivisao(`Divisao` obj) – buscar todos os dispositivos da divisão.

3.8 ControleDAO.

Essa classe é responsável por gravar controle no banco de dados.

- `public string` Gravar(`Controle` obj) – grava controle no banco de dados.
- `public string` Alterar(`Controle` obj) – altera controle no banco de dados.
- `public string` Deletar(`Controle` obj) – deleta um controle no banco de dados.
- `public Controle` BuscarId(`int` valor) – busca controle por identificador.

- `public Controle` BuscarId(`Controle` obj) – busca controle por objeto controle.
- `public List<Controle>` BuscarTodos() – busca todos os controles.
- `public Controle` BuscarComandosControle(`Controle` obj) – busca todos os comandos do controle.
- `public bool` VerificarComandoControle(`Controle` obj) – verifica se comando tem controle.
- `public bool` VerificarDispositivoControle(`Controle` obj) – verifica se controle tem dispositivo.

3.9 ComandoDAO.

Essa classe é responsável por gravar comandos no banco de dados.

- `public string` Gravar(`Comando` obj) – grava comando no banco de dados.
- `public string` Alterar(`Comando` obj) – altera comando no banco de dados.
- `public string` Deletar(`Comando` obj) – deleta comando do banco de dados.
- `public Comando` BuscarId(`int` valor) – busca um comando por identidade.
- `public Comando` BuscarId(`Comando` obj) – busca o comando por objeto do tipo comando.
- `public List<Comando>` BuscarTodos() – busca todos os comandos.
- `public List<Comando>` BuscarComandosControle(`Controle` obj) – busca todos os comandos do controle
- `public bool` VerificarHistoricoComando(`Comando` obj) – verifica se comando tem histórico.
- `public bool` VerificarEventoComando(`Comando` obj) – verifica se evento tem comando.
- `public bool` VerificarLeitorComando(`Comando` obj) – verificar se comando tem leitor.

3.10 **AgendamentoDAO.**

Essa classe é responsável por gravar agendamentos no banco de dados.

- `public int Gravar(Agenda obj)` – faz a gravação da agenda no banco de dados.
- `public string Alterar(Agenda obj)` – faz alteração da agenda no banco de dados.
- `public Agenda BuscarId(int valor)` – buscar agenda no banco de dados por identidade.
- `public Agenda BuscarId(Agenda obj)` – buscar um agendamento no banco de dados por objeto do tipo agenda.