Fechamento da Folha

Bruno, David Hildebrandt, Enzo, Gabriel Simon, Lucas Walim

Senior

Senior Go DEV_

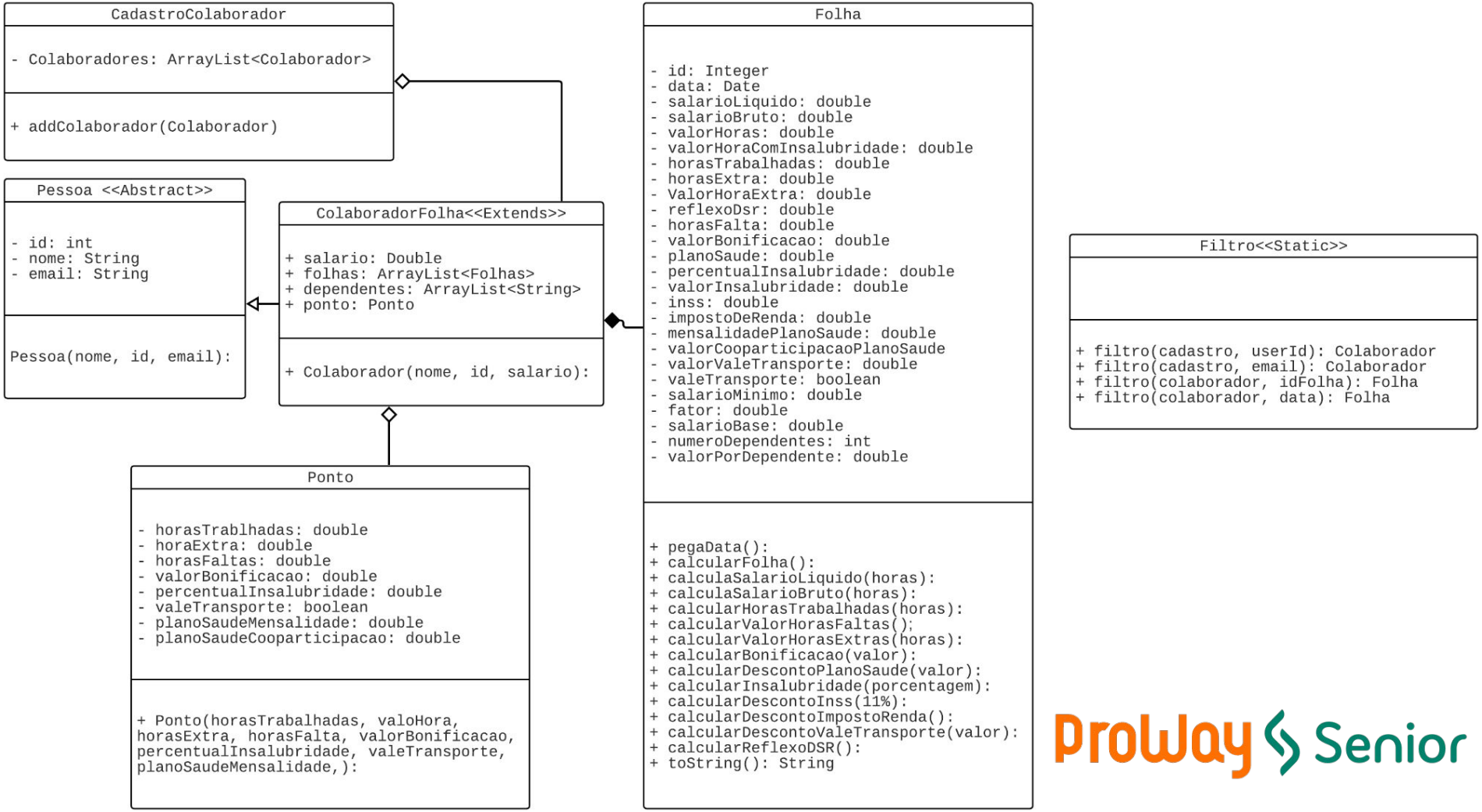
ProWay

Gestão de
Pessoas | HCM
Folha de Pagamento

Diagrama de Classes

ProWay

Senior



ProWay Senior

Classe Pessoa

Pessoa <<Abstract>>
- id: int - nome: String - email: String
Pessoa(nome, id, email):

```java
protected int id;
protected String nome;
protected String email;

public Pessoa(String nome, int id, String email) {
    this.nome = nome;
    this.id = id;
    this.email = email;
}
```

Senior ProWay

Classe Colaborador

ColaboradorFolha<<Extends>>
+ salario: Double + folhas: ArrayList<Folhas> + dependentes: ArrayList<String> + ponto: Ponto
+ Colaborador(nome, id, salario):

```java
private double salario;
private ArrayList<Folha> totalFolhas = new ArrayList<Folha>();
private ArrayList<String> dependentes = new ArrayList<String>();
private Ponto ponto;

public Colaborador(String nome, int id, String email, double salario) {
    super(nome, id, email);
    this.salario = salario;
}
```

Senior ProWay

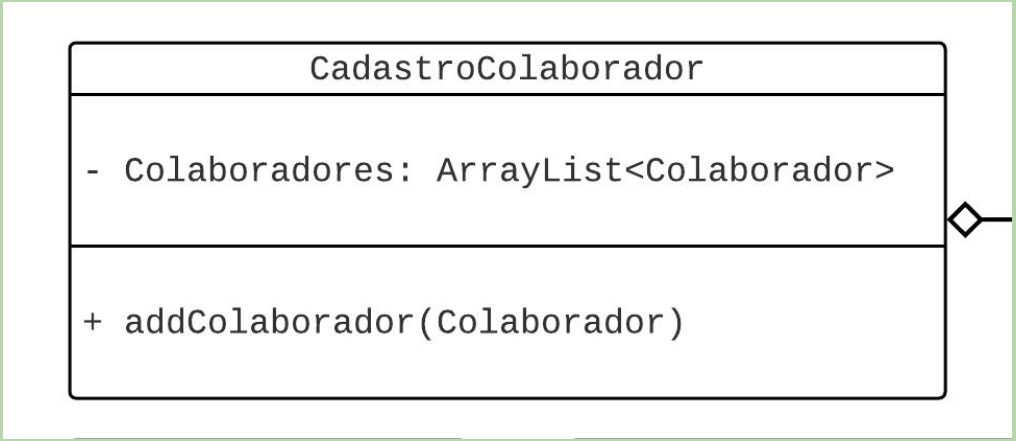
Cadastro de Colaboradores

ProWay Senior

```java
public final class CadastroColaborador {

    private ArrayList<Colaborador> colaboradores = new ArrayList<Colaborador>();

    public ArrayList<Colaborador> getColaboradores() {
        return colaboradores;
    }
```



Classe Ponto

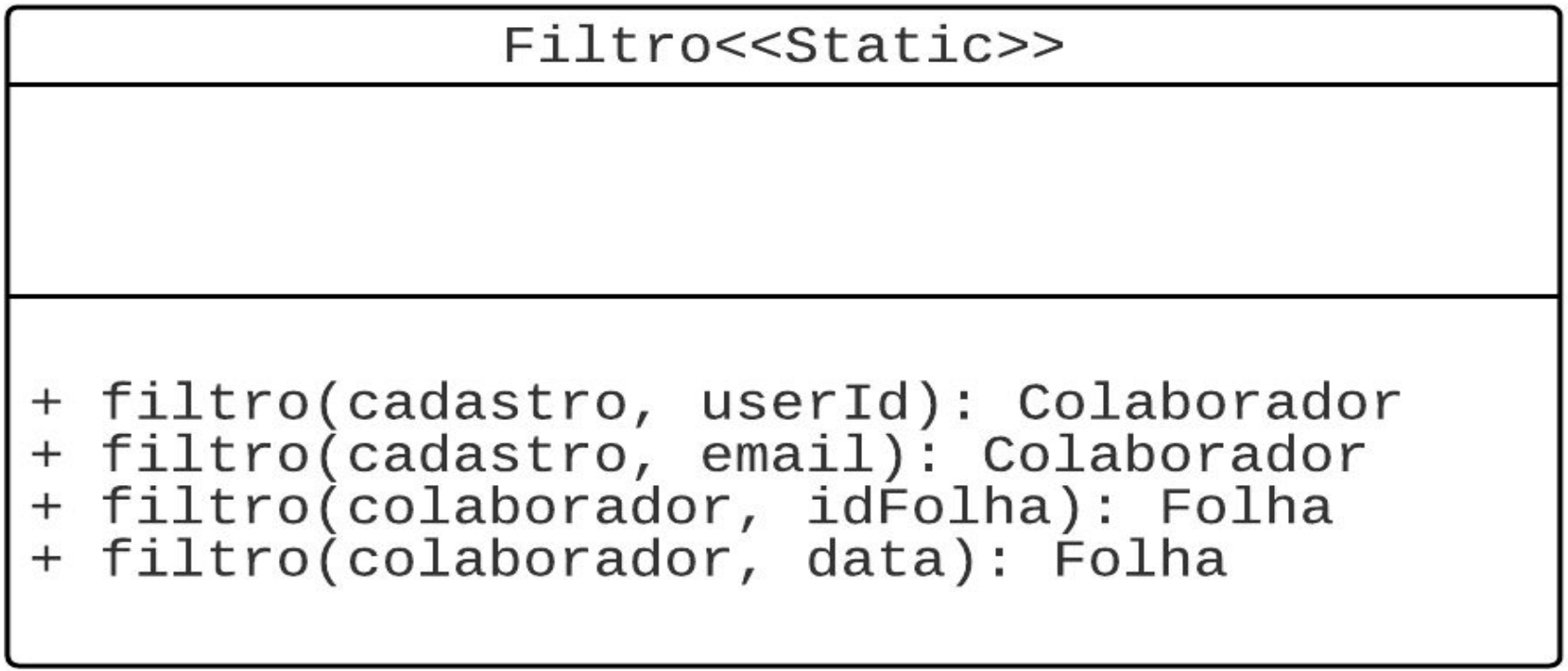
Classe contendo somente o construtor e os atributos que serão passados pelo controle de pontos utilizados para realizar o cálculo da folha.

```java
public Ponto(double horasTrabalhadas, double horasExtra,
        double horasFaltas, double valorBonificacao,
        double percentualInsalubridade, boolean valeTransporte,
        double mensalidadePlanoSaude, double valorCooparticipacaoPlanoSaude) {
    this.horasTrabalhadas = horasTrabalhadas;
    this.horasExtra = horasExtra;
    this.horasFaltas = horasFaltas;
    this.valorBonificacao = valorBonificacao;
    this.percentualInsalubridade = percentualInsalubridade;
    this.valeTransporte = valeTransporte;
    this.mensalidadePlanoSaude = mensalidadePlanoSaude;
    this.valorCooparticipacaoPlanoSaude = valorCooparticipacaoPlanoSaude;
}
```

Ponto
- horasTrablhadas: double - horaExtra: double - horasFaltas: double - valorBonificacao: double - percentualInsalubridade: double - valeTransporte: boolean - planoSaudeMensalidade: double - planoSaudeCooparticipacao: double
+ Ponto(horasTrabalhadas, valoHora, horasExtra, horasFalta, valorBonificacao, percentualInsalubridade, valeTransporte, planoSaudeMensalidade,):

Senior ProWay

Planejamento



Filtro:

Classe com único objetivo de possibilitar o saber do Cadastro de determinado Colaborador e Folha...

- Questão interna e natural...

Método de filtro das Folhas por Data...

```java
/**
 * Filtra os Colaboradores por ID
 *
 * Recebe os parâmetros para verficar se o Colaborador existe e o retorna
 *
 * @param userId; ID referente ao Colaborador que o usuário deseja saber sobre
 * @param cadastro; lista de todos os colaboradores cadastrados
 *
 * @return Colaborador/null Colaborador desejado/Colaborador não existe
 */
static public Colaborador filtro(int userId, CadastroColaborador cadastro) {
    for(Colaborador c : cadastro.getColaboradores()) {
        if(c.getId() == userId) {
            return c;
        }
    }
    return null;
}
```

```java
static public Folha filtro(Colaborador c, String data) {
    for(Folha folha : c.getTotalFolhas()) {
        if(data == folha.getData()) {
            return folha;
        }
    }
    return null;
}
```

Método de filtro dos Colaboradores por ID, e sua devida documentação ...

final

ProWay Senior

Filtro-testes

Opções base do esperado de ocorrência com os métodos, sendo elas, 2...

Testes de filtro do Colaborador requerido, por ID... (Cadastrado ou não)

```java
@Test
public void testFiltroUsuarioId() {
    cadastro.addColaboradores(colaborador0);
    cadastro.addColaboradores(colaborador1);
    Colaborador c = Filtro.filtro(0, cadastro);
    assertEquals(0, c.getId());

}

@Test
public void testFiltroUsuarioInexistenteId() {
    cadastro.addColaboradores(colaborador0);
    cadastro.addColaboradores(colaborador1);
    Colaborador c = Filtro.filtro(5, cadastro);
    assertEquals(null, c);
}
```

```java
@Test
public void testFiltroFolhaData() {
    colaborador0.addTotalFolhas(folha0);
    colaborador0.addTotalFolhas(folha1);
    colaborador0.addTotalFolhas(folha2);
    Folha f = Filtro.filtro(colaborador0, "data2");
    assertEquals("data2", f.getData());
}

@Test
public void testFiltroFolhaInexistenteData() {
    colaborador0.addTotalFolhas(folha0);
    colaborador0.addTotalFolhas(folha1);
    colaborador0.addTotalFolhas(folha2);
    Folha f = Filtro.filtro(colaborador0, "data4");
    assertEquals(null, f);
}
```

Testes de filtro da Folha requerida, por Data... (Cadastrada ou não)

As instâncias de Colaborador, Folha e CadastroColaborador já foram criadas...

ProWay Senior

Classe Folha

ProWay Senior

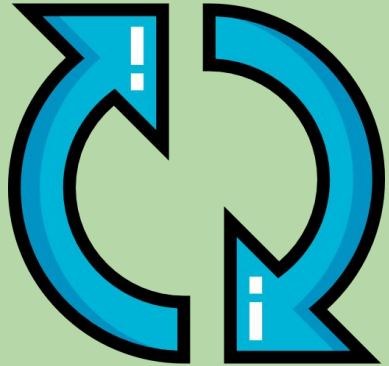
```java
public Folha(ColaboradorFolha colaborador) {
    this.horasTrabalhadas = colaborador.getPonto().getHorasTrabalhadas();
    this.horasExtra = colaborador.getPonto().getHorasExtra();
    this.horasFalta = colaborador.getPonto().getHorasFaltas();
    this.valorBonificacao = colaborador.getPonto().getValorBonificacao();
    this.percentualInsalubridade = colaborador.getPonto().getpercentualInsalubridade();
    this.mensalidadePlanoSaude = colaborador.getPonto().getMensalidadePlanoSaude();
    this.valorCooparticipacaoPlanoSaude = colaborador.getPonto().getvalorCooparticipacaoPlanoSaude();
    this.salarioBase = colaborador.getSalario();
    this.valeTransporte = colaborador.getPonto().isValeTransporte();
    this.numeroDependentes = colaborador.getDependentes().size();
}
```

Folha

- id: Integer
- data: Date
- salarioLiquido: double
- salarioBruto: double
- valorHoras: double
- valorHoraComInsalubridade: double
- horasTrabalhadas: double
- horasExtra: double
- ValorHoraExtra: double
- reflexoDsr: double
- horasFalta: double
- valorBonificacao: double
- planoSaude: double
- percentualInsalubridade: double
- valorInsalubridade: double
- inss: double
- impostoDeRenda: double
- mensalidadePlanoSaude: double
- valorCooparticipacaoPlanoSaude
- valorValeTransporte: double
- valeTransporte: boolean
- salarioMinimo: double
- fator: double
- salarioBase: double
- numeroDependentes: int
- valorPorDependente: double

+ pegaData():
+ calcularFolha():
+ calculaSalarioLiquido(horas):
+ calculaSalarioBruto(horas):
+ calcularHorasTrabalhadas(horas):
+ calcularValorHorasFaltas():
+ calcularValorHorasExtras(horas):
+ calcularBonificacao(valor):
+ calcularDescontoPlanoSaude(valor):
+ calcularInsalubridade(porcentagem):
+ calcularDescontoInss(11%):
+ calcularDescontoImpostoRenda():
+ calcularDescontoValeTransporte(valor):
+ calcularReflexoDSR():
+ toString(): String

Como Era x Como Ficou



Senior ProWay

```java
static double valorReferenciaEvento = 15.00;
static double valorSalarioInicial;
static double valorSalarioFinal;
static double valorDiferencaSalario;
static double valorInsalubridade = 220.00;
static double valorMinimo = 1100.00;
static int colabId = 101;
static String nomeColab = "Maria";
static double quantidadeHorasTrabalhadas = 220;
static double valorHoraColab = 12.81;
static boolean possuiInsalubridade = true;
static double quantidadeHorasExtrasColab = 15;
static double percentualInsalubridadeColab = 20;
static double quantidadeHorasFaltas = 20;
static double valorMensalidadePlanoSaude = 100.00;
static double valorCoparticipacaoPlano = 236.25;
static double valorBonificacaoColab = 250.00;
static double valorHoraComInsalubridade = 13.81;

@Test
public void testFolhaFinal1() {
    double resultado = Calculos.calculaFolhaFinal(colabId);
    assertEquals(resultado, 2451.80, 0.10);
}
```

ProWay Senior

```java
public static double calculaFolhaFinal(int colabId) {
    double horaComInsalubridade = calculaHoraComInsalubridade(valorHoraColab, quantidadeHorasTrabalhadas, percentualInsalubridadeColab);
    double valorSalarioBruto = calculaHorasTrabalhadas(quantidadeHorasTrabalhadas, horaComInsalubridade);
    valorSalarioBruto += valorHorasExtras(quantidadeHorasExtrasColab, horaComInsalubridade, 0.5);
    valorSalarioBruto += adicionaBonificacao(valorBonificacaoColab);
    double salarioDescontos = valorHorasFaltas(horaComInsalubridade, quantidadeHorasFaltas)
            + descontaPlanoSaude(valorMensalidadePlanoSaude, valorCoparticipacaoPlano)
            + calculaImpostoRenda(valorSalarioBruto)
            + descontoInss(valorSalarioBruto);
    double salarioFinal = valorSalarioBruto - salarioDescontos;
    return salarioFinal;
}
```

```java
@Test
public void testeCalculaFolhaComDependente() {

    ColaboradorFolha jorge = new ColaboradorFolha("Jorge", 0, "jorge@gmail.com", 2751.0);
    Ponto pontoJorge = new Ponto(220, 19.37, 12.89, 153, 10, true, 100, 127);
    jorge.setPonto(pontoJorge);
    jorge.addDependentes("Tiburcio");
    Folha folha = new Folha(jorge);
    jorge.addTotalFolhas(folha);

    double valor = folha.calcularFolha();
    assertEquals(valor, 2481.50, 0.01);
}
```

```java
public double calcularFolha() {

    this.salarioBruto += this.calcularHorasTrabalhadas();
    this.salarioBruto -= this.calcularValorHorasFaltas();
    this.salarioBruto += this.calcularValorHorasExtras();
    this.salarioBruto += this.calcularDSR();
    this.salarioBruto += this.calcularBonificacao();
    this.salarioBruto -= this.calcularDescontoInss();
    this.salarioBruto -= this.calcularDescontoImpostoRenda();
    this.salarioBruto -= this.calcularDescontoPlanoSaude();
    this.salarioBruto -= this.calcularDescontoValeTransporte();
    this.salarioLiquido = this.salarioBruto;
    this.setDataEmissao();

    return this.salarioLiquido;
}
```

Senior

ProWay

```java
public static double descontaPlanoSaude(double valorMensalidade, double valorCoparticipacao) {
    double totalDescontoPlanoSaude = valorMensalidade + valorCoparticipacao;
    return totalDescontoPlanoSaude;
}
```

```java
public double calcularDescontoPlanoSaude() {
    if (this.mensalidadePlanoSaude >= 0) {
        if (this.valorCooparticipacaoPlanoSaude >= 0) {
            this.planoSaude = this.mensalidadePlanoSaude + this.valorCooparticipacaoPlanoSaude;
        } else {
            this.valorCooparticipacaoPlanoSaude = 0;
            this.planoSaude = this.mensalidadePlanoSaude + this.valorCooparticipacaoPlanoSaude;
        }
    } else {
        this.mensalidadePlanoSaude = 0;
        this.valorCooparticipacaoPlanoSaude = 0;
        this.planoSaude = this.mensalidadePlanoSaude + this.valorCooparticipacaoPlanoSaude;
    }
    return this.planoSaude;
}
```

```java
@Test
public void testeCalculoIRFeriasComDependente1() {
    ColaboradorFolha jorge = new ColaboradorFolha("Jorge", 0, "jorge@gmail.com", 3000.0);
    Ponto pontoJorge = new Ponto(220, 0, 0, 0, 0, true, 100, 50);
    jorge.setPonto(pontoJorge);
    jorge.addDependentes("Filho 1");
    jorge.addDependentes("Filho 2");
    Folha folha = new Folha(jorge);
    jorge.addTotalFolhas(folha);

    double valor = folha.calcularFerias(20, 10);
    assertEquals(valor, 3437.67, 0.01);
}
```

Senior ProWay

O que foi feito

- Todo o backlog da sprint
- Diagrama de Classes
- Refatoração de testes e códigos

Senior ProWay

Problemas contornados

- Números decimais envolvendo arredondamento
- Biblioteca JUnit
- Git / Github

Senior ProWay

O que não foi feito

- Rodízio Pair programming

Senior ProWay

O que foi adicionado

- **Orientação à objetos**
- Classe CadastroColaborador
- Classe Pessoa
- Classe Colaborador
- Classe Ponto
- Classe Folha
- Cálculo de férias
- Cálculo Reflexo DSR
- Filtro de Folhas
- Integração de Classes

Senior ProWay

O que foi Retirado / Bloqueado

- Interface no console
- Tela para report
- Login

Senior ProWay

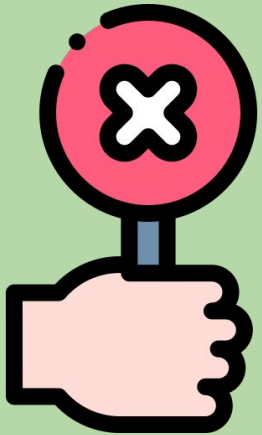
O que deu certo?

- Integração das classes
- Testes unitários
- Desenvolvimento
- Planejamento
- Pair programming
- Trello

Senior ProWay

O que deu errado?

- Implementação de uma classe Main
- GitHub



Senior ProWay

O que nós aprendemos ?

- Testes unitários
- Integração de classes / refatoração
- Orientação a Objeto

Senior ProWay

O que devemos fazer diferente

- Planejar melhor nome de variáveis e métodos
- Fazer rodízio
- Questionar o assunto que tem dificuldade
- Antes do planejamento conversar

Senior ProWay

Sprint Retrospective

- Continuar fazendo
  - Pair programming
  - Testes unitários
  - Diagrama de classes
- Começar a fazer
  - Daily do grupo diária
  - Padrão de atributos métodos
  - Uma pessoa começar a intermediar entre os grupos
- Fazer mais
  - Planejamento
- Parar de fazer
  - Começar programando sem antes conversar
  - Documentar no final
- Fazer menos
  - Programar em casa
  - Programar métodos antes do teste

Senior ProWay

Dúvidas?

Senior ProWay