Registro de Ponto

Integrantes:
Janaina
Lucas Grijó
Lucas Nunes
Marcelo
Vitor Peres

Sprint Review

O que foi feito;

O que não foi feito;

O que foi adicionado;

Bloqueados e removidos;

O que foi feito ✓

O que foi adicionado ✚

- ✓ Re-escrito histórias de Usuário -> Product Backlog
- ✓ Migração/Centralização da documentação do trello para o G-drive.
- ✚ Especificação dos Caso de Uso (Parcial).
- ✚ Diagrama de Classes
- ✓ Re-estruturação do Código-fonte para OO.
- ✓ Re-organização do Trello
- ✚ Arquitetura em camadas (separar Entidade/Lógica) e interfaces.
- ✚ Controller com rotinas executadas em horários pré-determinados
- ✚ Lógica de abre e fecha ponto

Backlog do Produto

Estórias de Usuário

	A	B
1	**ID**	**Estória/Tarefa**
2	C1	**como** colaborador **quero** bater ponto **a fim de** registrar minha entrada/saída.
3	C2	**como** colaborador **quero** abrir uma justificativa **a fim de** explicar os motivos de uma falta ou ponto batido incorretamente.
4	C3	**como** colaborador **quero** consultar meus pontos **a fim de** gerenciar minha situação de horas trabalhadas.
5	C4	**como** colaborador **quero** ser notificado quando esqueço de bater ponto de saída **a fim de** me lembrar de justificar essa ausência.
6	RH01	**como** RH **quero** analisar uma justificativa aberta (de outro colaborador) **a fim de** aprovar ou recusar a justificativa.
7	RH02	**como** RH **quero** consultar todas as justificativas abertas **a fim de** escolher entre as que devo
8	RH03	**como** RH **quero** cadastrar uma chave de acesso para um determinado colaborador **a fim de** permitir que o mesmo bata o ponto.
9	G01	**como** Gerente **quero** ser notificado quando existem pontos ausentes a mais de 3 dias de funcionários sob minha supervisão **a fim de** .
10	G02	**como** Gerente **quero** visualizar os pontos dos colaboradores que supervisiono **a fim de** gerenciar o batimento do ponto.
11	G03	**como** Gerente **quero** consultar justificativas de colaboradores sob minha supervisão **a fim de** .

Casos de Uso

Estórias/Atores definidos no [Product Backlog]

Introdução
- Classes de Usuários (Atores)
- Definição de Conceitos

Requisitos de Software
- Diagramas de casos de uso UML
- Requisitos Funcionais (Casos de Uso)
  - [C01] Requisitar o Ponto
  - [C02] Abrir Justificativa
  - [C03] Consultar Ponto
  - [C04] Receber notificação de ponto faltante
  - [RH01] Analisar uma justificativa
  - [RH02] Consultar Justificativas Abertas
  - [RH03] Cadastrar Chave de Acesso
  - [G01] Notificar Pontos Ausentes para o Gerente
  - [G02]
  - [G03]
- Requisitos Não-Funcionais
- Rastreabilidade de Requisitos

Requisitos Funcionais (Casos de Uso)

[ID Estória] Nome do Caso de Uso	[C01] Requisitar o Ponto
Pré-condições	O colaborador tem uma chave de acesso cadastrada no sistema de ponto e possui um meio de acessar o sistema (web/app/terminal).
Pós-condições	O sistema registra um ponto do colaborador (seja de entrada ou saída).
Fluxo Principal	1. O colaborador acessa o sistema. a. Por aplicativo/web i. O colaborador autentica usando chaves locais do smartphone (ou a senha cadastrada no sistema). ii. Se for a primeira vez do colaborador acessando o app/web, o sistema pede permissão para receber a localização (obrigatório). iii. O sistema exibe o horário atual do sistema (no fuso horário da empresa) e um botão para bater o ponto. iv. O colaborador pressiona o botão de bater ponto (ou na tela do app ou no site). v. O sistema envia email de confirmação ao usuário. vi. O sistema realiza a batida do ponto. b. Por terminal localizado na empresa (crachá, biometria e senha) i. O colaborador vai até o terminal de ponto e bate o ponto de acordo com o tipo do terminal.. ii. O sistema identifica meio de registro de ponto: crachá, biometria ou senha. iii. O sistema verifica se existe papel na impressora. iv. Se existir papel, o sistema realiza a batida do ponto. v. Se não existir papel, o sistema informa o usuário de que a máquina está sem papel e não finaliza a batida do ponto.

Trello

Ponto

in list Fazendo

MEMBERS

LABELS

Módulo de registro de ponto | Métodos

Description

Edit

[C01] - Como colaborador quero bater ponto a fim de registrar entrada/saída.
[C03] - Como colaborador quero consultar meus pontos.
[RH03] - Eu como colaborador do RH quero cadastrar uma chave de acesso (crachá, senha, biometria) para determinado colaborador a fim de permitir que o mesmo bata o ponto.
[G02] - Como colaborador gerente de uma equipe quero visualizar os pontos dos colaboradores que supervisiono a fim de gerenciar o batimento de ponto.
[G04] - ?? (a decidir se é responsabilidade do nosso grupo implementar)

Tarefas Não Alocadas

Delete

0%

- [ ] [C01] Criar uma forma de registrar a localização
- [ ] [C01] Criar método que registra ip e nome da máquina
- [ ] [RH03] Criar método que gera chave de acesso para bater ponto.
- [ ] [RH03] Criar método que cadastra chave de acesso.

Classe - Histórico Mensal

Hide checked items | Delete

100%

- [x] ~~[C01] Criar classe HistoricoMensal com generics~~

Add an item

Classe - JornadaDeTrabalho

Hide checked items | Delete

50%

- [x] ~~[C01] Criar classe JornadaDeTrabalho com encapsulamento~~
- [ ] [C03] Cria um método que consulta o total de horas trabalhadas por uma pessoa. (A fim de retornar o total de horas para o grupo 6 [folha de pagamento]) calcularHorasTrabalhadasPorJornada()

Add an item

Classe - Turno

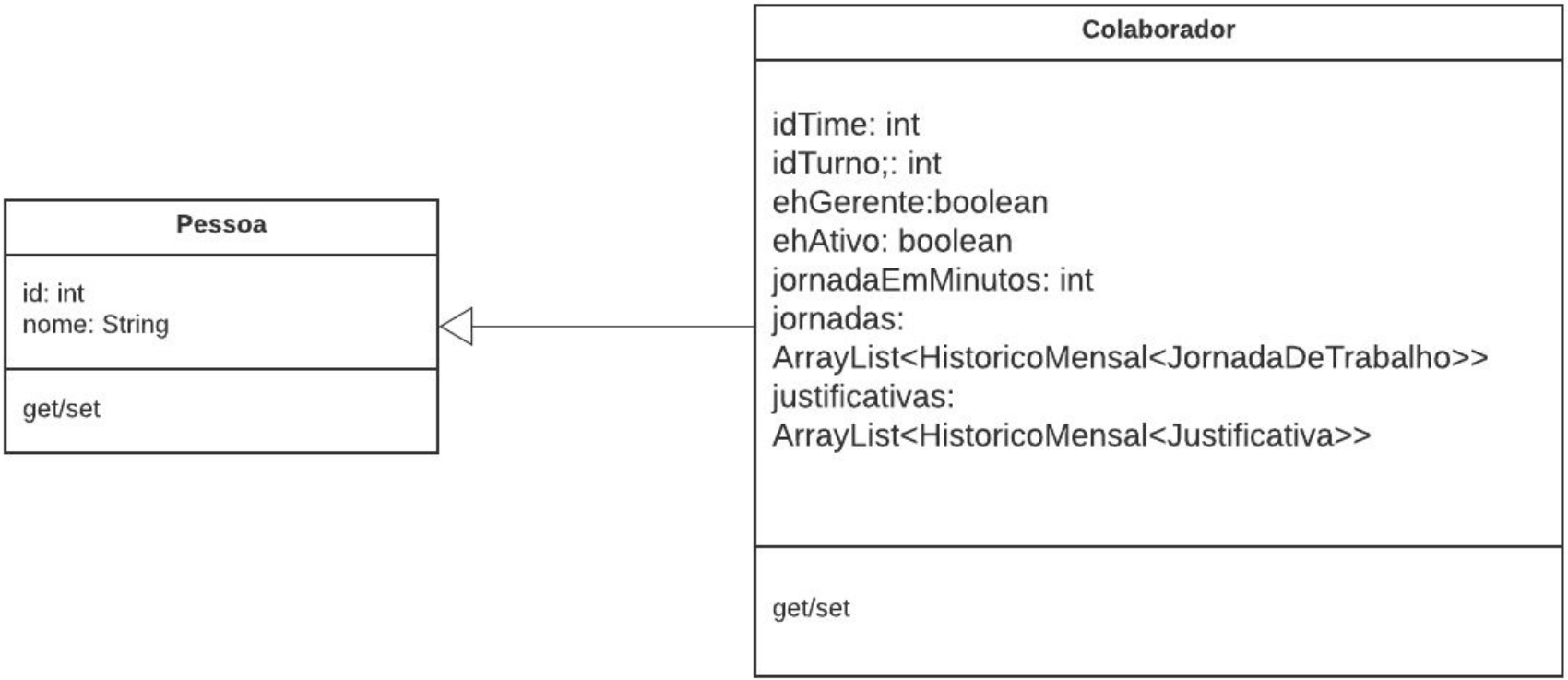
Hide checked items | Delete

100%

- [x] ~~[C01] Criar enum turno: primeiro turno, segundo turno, terceiro turno e comercial.~~

Add an item

Diagrama de Classe



HistoricoMensal

objetos: ArrayList<T>
ano: int
mes: int

get/set

JornadaDeTrabalho

pontos: ArrayList<Ponto>
horas: int
iminutos: int
segundos: int

get/set

Justificativa

Integer id;
String data;
String msg;
String img;
StatusJustificativa status;
ArrayList<String> dados;

verificaData(Justificativa entidadeJustificativa): boolean
verificaImagem(Justificativa entidadeJustificativa): boolean
salvarJustificativa(Justificativa novaJustificativa, Colaborador colaboradorRecebido): ArrayList<HistoricoMensal<Justificativa>>
consultaJustificativaStatus(ArrayList<Justificativa> justificativas, String status): ArrayList<String>
consultaJustificativaPessoa(ArrayList<Justificativa> justificativas, int id): ArrayList<String>
verificaMensagem(Justificativa entidadeJustificativa)
criaEVerificaJustificativa(Integer idRecebido, String dataRecebido, String msgRecebido, String imgRecebido) : Justificativa
Justificativa novaJustificativa, Colaborador colaboradorRecebido

Ponto

id: int
idColaborador: int
tipo: tipoDePonto (enum)
data: LocalDate
localizacao: String (google)
horario: LocalDateTime

// constructor
// getters & setters
//pegarPontosPorPessoa(Colaborador colabrado)
//pegarTodosOsPontos
//salvarPonto(TipoDePonto tipo, String localizacao, Colaborador c)
//cadastrarpontos
// pegarPontosDeUmaJornadaPorColaboradorEData(Colaborador colaborador, LocalDate data)

consultarPonto(Colaborador c)
consultarPonto(Colaborador c, int ano)
consultarPonto(Colaborador c, int ano, int mes)
consultarPontosEmUmIntervalo(Colaborador c, Date inicioIntervalo, Date finalIntervalo)
consultarPontosDoTime(Colaborador c) [apenas para gerente]

Notificações

idNotificacao: int
idDestinatario: int
mensagem: String
enviar: boolean

// enviarNotificacao(Integer idDestinatario, String mensagem)
// retornaColaboradoresASeremNotificados(ArrayList<Colaborador>listaColaborador)
// validarDestinatario(int IdDestinatario): Colaborador
// validarMensagem(String mensagem): boolean

consultarNotificacaoPorPessoa(int idUsuario): ArrayList<Notificacao>

Código

```
58    public Ponto salvarPonto(TipoDePonto tipo, String localizacao, Colaborador c) {
59        HistoricoMensal<JornadaDeTrabalho> ultimoHistoricoMensal = c.getJornadas().get(c.getJornadas().size() -
60        JornadaDeTrabalho ultimaJornadaDoUltimoHistoricoMensal = ultimoHistoricoMensal.getObjetos()
61                .get(ultimoHistoricoMensal.getObjetos().size() - 1);
62        LocalDate dataAtual = LocalDate.now();
63        LocalDateTime horarioAtual = LocalDateTime.now();
64        Ponto novoPonto = new Ponto(c.getId(), tipo, dataAtual, horarioAtual, localizacao);
65        ultimaJornadaDoUltimoHistoricoMensal.addPonto(novoPonto);
66        return novoPonto;
67    }
```

Código

```
public ArrayList<HistoricoMensal<Justificativa>> salvarJustificativa(Justificativa novaJustificativa, Colaborador colaboradorRecebido) {

    Integer idRecebido = colaboradorRecebido.getId();
    ArrayList<HistoricoMensal<Justificativa>> novoHistoricoAtualizado = colaboradorRecebido.getJustificativas();
    if(idRecebido == novaJustificativa.getId()) {

        novoHistoricoAtualizado.get(novoHistoricoAtualizado.size()-1).getObjetos().add(novaJustificativa);
        return novoHistoricoAtualizado;
    }
    return null;
}
/**
```

Entrada -› Saída

Trabalharemos com entrada e saída, dessa forma não importa quantas vezes o funcionário irá bater o ponto, sempre iremos extrair o intervalo de tempo entre as batidas (entrada e saída).

Para manter tudo em ordem, teremos um controller que diariamente fará uma verificação dos pontos batidos de cada colaborador.

Primeiro-turno, segundo-turno e comercial: 23h
Terceiro-turno: 12h

>>> 23 horas

1. O método consulta todas as pessoas cadastradas ativas e insere em uma lista.
2. O método recebe essa lista de pessoas e percorre cada colaborador para validar seus pontos e jornada de trabalho.
3. Verifica se é par ou ímpar:
    a. se for ímpar: o controller notifica o usuário, pois isso significa que temos alguma entrada sem saída. Para evitar maiores problemas referente a esta entrada aberta, o controller irá fechar essa entrada e notificar o usuário para que o mesmo envie uma justificativa, onde o RH receberá essa justificativa para corrigir o horário de saída.
    b. se for par: o controller calcula a jornada de trabalho cumprida e verifica se está de acordo com a jornada de trabalho do colaborador em questão.

```
public ArrayList<Colaborador> retornaColaboradoresASeremNotificados(ArrayList<Colaborador> listaColaborador) {

	ArrayList<Colaborador> colaboradores = new ArrayList<Colaborador>();
	ArrayList<LocalDateTime> listaEntrada = new ArrayList<LocalDateTime>();
	ArrayList<LocalDateTime> listaSaida = new ArrayList<LocalDateTime>();

	for (Colaborador colaborador : listaColaborador) {
		LocalDate dataDeHoje = LocalDate.of(2021, 1, 1);
		PontoLogica pontoL = new PontoLogica();
		ArrayList<Ponto> listaPontos = pontoL.pegarPontosDeUmaJornadaPorColaboradorEData(colaborador,
		dataDeHoje);

		if (listaPontos.size() % 2 == 0) {
			for (Ponto ponto : listaPontos) {
				if (listaPontos.indexOf(ponto) % 2 == 0) {
					listaEntrada.add(ponto.getHorario());
				} else {
					listaSaida.add(ponto.getHorario());
				}
			}
			int totalMinutos = 0;
			for (int i = 0; i < listaEntrada.size(); i++) {
				LocalDateTime entrada = listaEntrada.get(i);
				LocalDateTime saida = listaSaida.get(i);
				LocalDateTime diferencaTempo = LocalDateTime.from(entrada);
				long horas = diferencaTempo.until(saida, ChronoUnit.HOURS);
				diferencaTempo = diferencaTempo.plusHours(horas);
				long minutos = diferencaTempo.until(saida, ChronoUnit.MINUTES);
				diferencaTempo = diferencaTempo.plusMinutes(minutos);
				totalMinutos += (int) ((horas * 60) + minutos);
			}
			if (totalMinutos < colaborador.getJornadaEmMinutos()) {
				colaboradores.add(colaborador);
			}
		} else {
			colaboradores.add(colaborador);
		}
	}
	return colaboradores;
}
```

O que não foi feito 

- Especificação completa dos casos de uso
- Remake do Diagrama de Casos de Uso
- Métodos de Gerência
- Salvar localização geográfica dos pontos batidos.
- Geração da Chave de Acesso (Senha).
- Lista de todos os colaboradores (persistência dos dados).
- Refatoração do código fonte
  - Quebrar métodos maiores em menores
  - Revisão documentação

Bloqueados e removidos 

Bloqueados

- Autenticação.
- Retorno dos dados para visualização (precisa do BD)

Removidos

- Pacote de Ferramentas (Calculo de Horas / Impressora)

RETROSPECTIVA

Fazer mais:

1. TDD
2. Dailies Internas
3. Atualizar o Trello constantemente

Fazer menos:

1. Fugir do foco
2. Planejar circunstâncias futuras (banco de dados, integração)

Parar de fazer:

1. Trabalhar sem planejar o gasto do tempo.

Continuar fazendo:

1. Comunicação em Equipe
2. Teamwork
3. Organização/Divisão de Tarefas

Começar a fazer:

1. Planejamento da Sprint

O que aprendemos

- **Git**
  - Documentação, Diagrama de Classe
  - ArrayList<ArrayList<ArrayList<Ponto>>>
  - Arquitetura em Camadas, Generics <T>

Antes:

Depois:

FIM!

Alguma questão? Vamos conversar!

https://github.com/marcelo-schaefer/senior-proway-controle-ponto