Projeto Alterdata

Candidato: Leandro Silva Azevedo de Almeida

Para o desenvolvimento do projeto tentei construir a maior parte das rotinas evitando alguns componentes de terceiros. Foi adotado o padrão MVC (*Model- View-Controller*). Desta forma a disposição dos pacotes, classes e métodos ficaram da seguinte Maneira:

- Modelo (Pacote referente à *"Model"* do MVC):
    - Neste Pacote ficaram apenas a representação do Funcionário com os atributos Nome e Função, assim como as rotinas responsáveis por lidar com a alteração dos valores do objeto.
    - Neste Pacote também temos a Representação do Ponto marcado e a lista de pontos marcados. Adotei a representação das horas e minutos utilizando o tipo inteiro para facilitar as contas.

- Visão (Pacote referente à *"View"* do MVC):
    - Neste pacote temos todas as telas desenvolvidas. Todas as telas possuem um ponteiro para o funcionário e para a Lista de pontos batidos.
      Dentre as telas, em especial tem-se a tela "TelaRegPont". Nesta tela utilizei atributos do tipo *"MaskEdit"* para exibição das horas no formato indicado em análise. Foi necessário implementar um tratamento especial para horas inseridas, assim como a chamada de rotinas da camada de controle para a validação dos horários.
      Em análise em primeiro momento foi indicado que "As marcações podem ser no intervalo que quiser, desde que ao final do dia tenha completado 8 horas de trabalho". Em outro momento foi indicado a requisição de verificação de horas extras e atrasos logo permiti a marcação dos pontos para valores maiores e menores que 8 horas de trabalho para a geração de horas extras e atrasos.

- Controle (Pacote referente à *"Controller"* do MVC):
    - Este pacote temos duas *"units"* de controle. A primeira delas (ControlePonto) refere-se ao controle das operações sobre a marcação de Ponto e também à validação e formatação dos pontos marcados:

- **InicializaPontos:** Inicializa a lista de pontos, verificando se existe o arquivo “Pontos.csv” que armazena localmente a lista de pontos já antes registrados.
- **AdicionaPonto:** Adiciona um novo ponto a lista de pontos já marcados
- **Const_str:** Gera uma repetição concatenada de uma string, dado uma quantidade de repetições e uma string informadas.
- **inttostr2:** Formata um inteiro menor que 9 e maior que 0 para uma string no formato hora ou minutos, ex:”01”.
- **BateMesReferencia:** Compara se duas datas possuem o mesmo mês/ano.
- **VerificaSeJaBateuDia:** Verifica se um ponto já foi marcado para algum dia.
- **TotalSegEntreHorarios:** Calcula o total de segundos entre duas horas informadas.
- **FormataHoras:** Formata a hora informada para uma string na forma: “00:00”
- **SegEntre2Horarios:** Calcula o total de segundos entre duas horas informadas.
- **SalvaListaDePontos:** Salva o todos os Pontos marcados num arquivo local com extensão “.csv” que pode ser exportado para outras plataformas.

- A outra *“unit”* (ControleFuncionario), refere-se ao controle das operações sobre o funcionário com as seguintes rotinas:
  - **InicializaFuncionario**: Como o nome indica, apenas inicializa uma instância de funcionário, verificando se existe o arquivo “Arq.cnf” que armazena localmente as informações do funcionário.
  - **SalvaFuncionario:** Salva localmente as informações do Funcionário no arquivo.

Assim que o programa é iniciado tenta-se ler do arquivo de configuração o nome do funcionário e a função. Também tenta-se ler do arquivo “csv” a lista de pontos já batidas. Em seguida O ponteiro de todas as telas são ajustados para o mesmo objeto funcionário e Lista de Pontos. Ao final da Execução do programa a lista de pontos é atualizada no arquivo, assim como o nome do funcionário e a função.
Outros tratamentos referentes a nomes do funcionário e função não foram implementados nessa versão.