

Sistema de Histórico de falhas do CCO

Arthur Ramos Ribeiro (560417)

João Vinicius Alves (559369)

Juan Pablo Coelho (560445)

Challenge CCR 2024

10/1/24

Sumário

Descritivo	2
Objetivos	2
Justificativa	2
Descrição das Funcionalidades	2
1. Login e Controle de Acesso:	2
2. Registro de Falhas:	2
3. Consulta ao Histórico de Falhas:	2
4. Geração de Relatórios:	3
5. Alertas Automáticos:	3
6. Análise de Desempenho:	3
Código Fonte	3
Acessar Projeto	10

Descritivo

Objetivos

O projeto visa automatizar o monitoramento e o registro de falhas no Centro de Controle de Operações (CCO), utilizado pela CCR para gerenciar o tráfego ferroviário. O objetivo é aumentar a eficiência nas operações e reduzir a necessidade de intervenções manuais, garantindo uma resposta mais ágil a incidentes e melhorando a precisão dos dados gerados. Além disso, o sistema oferece funcionalidades de análise e geração de relatórios sobre o histórico de falhas, permitindo que decisões estratégicas sejam tomadas com base em informações detalhadas e em tempo real.

Justificativa

A operação do CCO envolve um grande volume de dados e processos manuais, o que pode resultar em falhas humanas, atrasos e respostas inadequadas a eventos críticos. A automação dessas atividades é essencial para garantir uma operação mais eficiente, reduzindo custos e melhorando a qualidade do serviço oferecido aos usuários dos sistemas de transporte. A proposta deste projeto é criar um sistema que integre a captura e gestão de dados em tempo real, permitindo monitoramento contínuo e geração de alertas automáticos, além de uma interface para o registro e consulta de históricos, possibilitando uma visão ampla e precisa do desempenho do CCO.

Descrição das Funcionalidades

O sistema será dividido em diferentes módulos, cada um focado em uma funcionalidade essencial para o CCO:

1. Login e Controle de Acesso: O sistema permitirá que usuários façam login com diferentes permissões (administradores e operadores). Administradores terão controle total sobre os dados, enquanto operadores poderão apenas consultar informações.

2. Registro de Falhas: Quando uma falha ocorrer no sistema, os administradores poderão registrar as informações relevantes, como a natureza do problema, o horário e o status de resolução.

3. Consulta ao Histórico de Falhas: Será possível consultar registros de falhas anteriores com filtros específicos, como data, tipo de falha e responsável pela resolução.

4. Geração de Relatórios: O sistema terá uma funcionalidade para gerar relatórios com base no histórico de falhas, permitindo a visualização de padrões e recorrências, facilitando a tomada de decisões estratégicas.

5. Alertas Automáticos: Com base nas falhas registradas, o sistema enviará alertas automáticos para o time de operações, permitindo uma rápida ação em tempo real para evitar incidentes mais graves.

6. Análise de Desempenho: O sistema permitirá que administradores visualizem gráficos e dados analíticos sobre a frequência e o impacto das falhas, ajudando a priorizar melhorias na operação.

Esse conjunto de funcionalidades está alinhado com o objetivo de reduzir a necessidade de intervenção manual, aumentar a precisão e fornecer dados robustos para análise e tomada de decisões estratégicas no contexto do CCO da CCR.

Código Fonte

```
from datetime import datetime

menuLogin = """\033[9m
=====
\033[1m\033[4mTela de login.\033[22m\033[24m
1. Administrador
2. Operador Geral
0. Sair
=====
"""

menuSistemaAdm = """\033[9m
=====
\033[1m\033[4mBem vindo ao sistema de histórico.\033[22m\033[24m
1. Registrar nova falha
2. Exibir histórico de falhas
3. Gerar relatório de falhas
4. Voltar para os logins
0. Sair
=====
"""
```

```
menuSistemaGeral = """\033[9m
=====
\033[1m\033[4mBem-vindo ao sistema de histórico.\033[22m\033[24m
1. Exibir histórico de falhas
2. Gerar relatório de falhas
3. Voltar para os logins
0. Sair
====="""
```

#Acima os menus principais a serem mostrados | Abaixo as funções do sistema

```
listaFalhas = []
permissaoAdm = False
```

```
def opcaoInvalida():
    return "Opção inválida"
```

```
def opcaoSair():
    return "Agradeço por usar. Saindo..."
```

```
def logarAdm():
    global permissaoAdm
    permissaoAdm = True
    return "\033[93m" + "Logado como Administrador."
```

```
def logarGeral():
    global permissaoAdm
    permissaoAdm = False
```

```

        return "\033[92m" + "Logado como Operador Geral."

def voltarLogin():
    return "Logging off..." + "\033[0m"

def registrarFalha():
    falha = {
        "idFalha": len(listaFalhas) + 1,
        "data": datetime.today().strftime("%d/%m/%Y - %H:%M"),
        "tipo": tipoFalha(),
        "descricao": input("Digite a descricao:\n")}
    listaFalhas.append(falha)
    return f"Falha #{falha["idFalha"]} adicionada ao sistema."

def exibeHistorico():

    historico = ""

    for falha in listaFalhas:
        idFalha = falha["idFalha"]
        dataFalha = falha["data"]
        tipo = falha["tipo"]
        descricaoFalha = falha["descricao"]

        historico += f"#{idFalha} ({dataFalha}) : {tipo} - {descricaoFalha}\n"

    if historico == "":
        historico = "Não há registros"

    return "Histórico de falhas:\n" + historico

```

```

def exibeRelatorio():

    listaTipos = []

    for falha in listaFalhas:

        listaTipos.append(falha["tipo"])

    return (f"Relatório de falhas:\n"
            f"Número de falhas: {len(listaFalhas)}\n"
            f"Falha mais frequente:
{max(listaTipos,key=listaTipos.count)}")

def tipoFalha():

    menuTipoFalha = """
=====
Tipos de falhas:
1.MECANICA
2.ELETRICA
3.SOFTWARE
0.OUTRO"""

    def tipoFalhaOutro():

        return "OUTRO"

    def tipoFalhaMecanica():

        return "MECANICA"

    def tipoFalhaEletrica():

        return "ELETRICA"

    def tipoFalhaSoftware():

        return "SOFTWARE"

```

```
opcoesTipoFalha = {  
    0:tipoFalhaOutro,  
    1:tipoFalhaMecanica,  
    2:tipoFalhaEletrica,  
    3:tipoFalhaSoftware  
}
```

```
print(menuTipoFalha)  
escolha = int(input("Digite o número da opção desejada:\n"))  
if not escolha in [0, 1, 2, 3]:  
    print(opcaoInvalida())  
    return tipoFalha()  
else:  
    resposta = opcoesTipoFalha.get(escolha)()  
    return resposta
```

#Acima funções do sistema | Abaixo organização e lógica dos menus principais

```
opcoesLogin = {  
    0:opcaoSair,  
    1:logarAdm,  
    2:logarGeral  
}
```

```
opcoesSistemaAdm = {  
    0:opcaoSair,  
    1:registrarFalha,  
    2:exibeHistorico,  
    3:exibeRelatorio,  
    4:voltarLogin  
}
```



```
opcoesSistemaGeral = {  
    0:opcaoSair,  
    1:exibeHistorico,  
    2:exibeRelatorio,  
    3:voltarLogin  
}
```

```
# Acima organização | Abaixo lógica do menu
```

```
"""
```

```
Login
```

```
=> Adm
```

```
    => Add falha (adiciona falha ao sistema)
```

```
    => Ver falhas (mostra um historico de falhas)
```

```
    => Relatorio falhas (mostra o numero de falhas e o maior tipo de  
falha)
```

```
    => Voltar login (volta a tela de login)
```

```
    => Sair (sai do programa)
```

```
=> Geral
```

```
    => Ver falhas
```

```
    => Relatorio falhas
```

```
    => Voltar login
```

```
    => Sair
```

```
=> Sair
```

```
"""
```

```
opcao = -1
```

```
while not opcao == 0:
```

```
    try:
```

```
        print(menuLogin)
```

```
        opcao = int(input("Digite o número da opção desejada:\n"))
```

```

if not opcao in [0,1,2]:
    print(opcaoInvalida())
else:
    resultado = opcoesLogin.get(opcao) ()
    print(resultado)
    if opcao in [1,2]:
        opcao = -1
        while not opcao == 0:
            try:
                if permissaoAdm:
                    print(menuSistemaAdm)
                    opcao = int(input("Digite o número da
opção desejada:\n"))

                    if not opcao in [0,1,2,3,4]:
                        print(opcaoInvalida())
                    else:
                        resultado =
opcoesSistemaAdm.get(opcao) ()

                        print(resultado)
                        if opcao == 4:
                            break

                else:
                    print(menuSistemaGeral)
                    opcao = int(input("Digite o número da
opção desejada:\n"))

                    if not opcao in [0,1,2,3]:
                        print(opcaoInvalida())
                    else:
                        resultado =
opcoesSistemaGeral.get(opcao) ()

                        print(resultado)
                        if opcao == 3:
                            break

            except ValueError:
                print("Valor inválido")

```

```
except ValueError:  
    print("Valor inválido")
```

Acessar Projeto

[CHALLENGE-MM/Challenge-1sem-2024-Python: Entregável de python para o challenge \(github.com\)](#)