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

Contents

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

menu_login = ("\033[9m"

              "\n====================================\n"

              "\033[1;4;29m"

              "Tela de login.\n"

              "\033[22;24m"

              "1. Administrador\n"
              "2. Operador\n"
              "0. Sair\n"
              "====================================")

menu_sistema_adm = ("\033[9m"

                    "\n====================================\n"

                    "\033[1;4;29m"

                    "Bem vindo ao sistema de histórico.\n"

                    "\033[22;24m"

                    "1. Registrar nova falha\n"
                    "2. Exibir histórico de falhas\n"
                    "3. Gerar relatório de falhas\n"
                    "4. Voltar para os logins\n"
                    "0. Sair\n"
                    "====================================")

menu_sistema = ("\033[9m"

                "\n====================================\n"

                "\033[1;4;29m"
```

```
                "Bem-vindo ao sistema de histórico.\n"

                "\033[22;24m"

                "1. Exibir histórico de falhas\n"
                "2. Gerar relatório de falhas\n"
                "3. Voltar para os logins\n"
                "0. Sair\n"
                "====================================")

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

lista_falhas = []
permissao_adm = False


def valor_invalido():
    return ("\033[1m"
            "Valor inválido"
            "\033[22m")


def opcao_invalida():
    return ("\033[1m"
            "Opção inválida"
            "\033[22m")


def opcao_sair():
    return ("\033[7m"
            "Agradeço por usar. Saindo..."
            "\033[27m")


def logar_adm():
    global permissao_adm
    permissao_adm = True
    return ("\033[93m"
            "Logado como Administrador.")


def logar_operador():
    global permissao_adm
    permissao_adm = False
    return ("\033[92m"
            "Logado como Operador.")


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


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


def exibe_historico():
    historico = ""

    for falha in lista_falhas:
```

```
        id_falha = falha["id_falha"]
        data_falha = falha["data"]
        tipo = falha["tipo"]
        descricao_falha = falha["descricao"]
        historico += f"#{id_falha} ({data_falha}) : {tipo} - {descricao_falha}\n"

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

    return "Histórico de falhas:\n" + historico


def exibe_relatorio():
    lista_tipos = []

    if len(lista_falhas) == 0:
        return "Não há falhas para o relatório"

    for falha in lista_falhas:
        lista_tipos.append(falha["tipo"])

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


def tipo_falha():
    menuTipoFalha = ("\n====================================\n"
                     "Tipos de falhas:\n"
                     "1.MECANICA\n"
                     "2.ELETRICA\n"
                     "3.SOFTWARE\n"
                     "0.OUTRO")

    def tipo_falha_outro():
        return "OUTRO"

    def tipo_falha_mecanica():
        return "MECANICA"

    def tipo_falha_eletrica():
        return "ELETRICA"

    def tipo_falha_software():
        return "SOFTWARE"

    opcoes_tipo_falha = {
        0: tipo_falha_outro,
        1: tipo_falha_mecanica,
        2: tipo_falha_eletrica,
        3: tipo_falha_software
    }

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


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

opcoes_login = {
    0: opcao_sair,
```

```
    1: logar_adm,
    2: logar_operador
}

opcoes_sistema_adm = {
    0: opcao_sair,
    1: registrar_falha,
    2: exibe_historico,
    3: exibe_relatorio,
    4: voltar_login
}

opcoes_sistema = {
    0: opcao_sair,
    1: exibe_historico,
    2: exibe_relatorio,
    3: voltar_login
}

# 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(menu_login)
        opcao = int(input("Digite o número da opção desejada:\n"))
        if not opcao in [0, 1, 2]:
            print(opcao_invalida())
        else:
            resultado = opcoes_login.get(opcao)()
            print(resultado)
            if opcao in [1, 2]:
                opcao = -1
                while not opcao == 0:
                    try:
                        if permissao_adm:
                            print(menu_sistema_adm)
                            opcao = int(input("Digite o número da opção
desejada:\n"))
                            if not opcao in [0, 1, 2, 3, 4]:
                                print(opcao_invalida())
                            else:
                                resultado = opcoes_sistema_adm.get(opcao)()
                                print(resultado)
                                if opcao == 4:
                                    break
                        else:
                            print(menu_sistema)
                            opcao = int(input("Digite o número da opção
desejada:\n"))
                            if not opcao in [0, 1, 2, 3]:
                                print(opcao_invalida())
```

```
                        else:
                            resultado = opcoes_sistema.get(opcao)()
                            print(resultado)
                            if opcao == 3:
                                break
                except ValueError:
                    print(valor_invalido())
except ValueError:
    print(valor_invalido())
```

Mudanças das Sprints

Sprint 1

- Projeto Original;

Sprint 2

- Mudança do nome das variáveis e funções de camelCase para snake_case;

- Atulização da formatação dos menus;

Acessar Projeto

https://github.com/MMChallengeMM/Challenge-Python