PLANEJAMENTO DE PROJETO

|  |
| --- |
| **Identificação do(s) Formando(s): Alan Rosa / Maria Clara** |

|  |  |
| --- | --- |
| **Definição do Projeto:** | **Aplicação do Sistema Bancário** |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Cenário de Desenvolvimento** | | | | | |
| **Fases de desenvolvimento do projeto** | | ***PLANEJAR***  Planejamento, pesquisa e recolha de informação. | ***CRIAR***  Criar o protótipo / produto / aplicação | ***COMUNICAR***  Partilhar/Divulgar o projeto ao grupo. | ***AUTOAVALIAÇÃO***  Auto-avaliação / avaliação dos pares |
| **Duração:** | | **3 horas** | **21 horas** | **45 min** | **15 min** |
| **Duração Total:** | | **25 horas** | | | |
| **Ações a desenvolver** | **Formando** | Saque de requisitos;  **DRAFT (Guia)**: Construção da documentação de apoio: aplicação do Sistema Bancário;  Planejamento do funcionamento do programa / script; | Desenvolver o programa;  Testes ao programa / script.  **FINALIZAR (Guia)**: Construção da documentação de apoio: aplicação do Sistema Bancário; | Apresentar o trabalho ao grupo;  Divulgação na plataforma TEAMS; | Autoavaliação do formando. |
| **Recursos:** | | Computador; Aplicação Visual Studio Code, PyCharm; Internet | Computador; Aplicação Visual Studio Code, PyCharm; Internet | Computador; Aplicação Visual Studio Code, PyCharm; Internet | Computador; Internet |

|  |
| --- |
| **Documentos anexos à PLANEJAMENTO** |
| **Planejamento + Guia do projeto + Respetivo Projeto (.py).**  **Observação: O Guia do projeto está devidamente documentado no projeto em si, em forma de comentários.**  OBS: Para conexão à base de dados MySQL eu utilizei user: “root” e senha: “root”.  GUIA DO PROJETO:  O software consiste em um sistema de bancário, ou seja, um sistema que agrega 4 perfis de usuários: Administrador, Gerente, Escriturário e Caixa, acrescido da possibilidade de cadastrar perfis de usuários limitados aos conflitos listados numa matriz SOD.  O perfil Administrador (nível de acesso 4) é capaz de cadastrar usuários de todos os perfis e suas respetivas senhas do sistema. Esse perfil também realiza o cadastro de sistemas, perfis de acessos vinculados a esses sistemas, matriz SOD com os conflitos relacionados aos perfis de sistemas e perfis de usuários para cada usuário cadastrado no Sistema Bancário.  O perfil Gerente (nível de acesso 3) tem o privilégio de gerir cadastro de agências, clientes e contas, incluir, editar e excluir, realizar depósitos e saques, bem como excluir transações de contas.  O perfil Escriturário (nível de acesso 2) tem o privilégio de gerir cadastro de clientes e contas, incluir, editar e excluir, bem como excluir transações de contas.  O perfil Caixa (nível de acesso 1) tem como principais funções realizar depósitos e saques, consultar clientes, contas e transações das contas.  O sistema foi escrito na linguagem Python, utilizando como ferramentas as IDE’s VSCode e PyCharm e utilizando conexão a uma base de dados MySQL.  Foram utilizadas diversas bibliotecas e métodos:   1. random /randint: Para gerar números aleatórios para números de conta; 2. xml.etree.ElementTree: Para importação de arquivos xml; 3. mysql.connector: Para conexão com base de dados MySQL; 4. hashlib /sha256: Para criptografia de senhas; 5. PyQt6.QtWidgets /QMainWindow, QApplication, QTableWidgetItem: Para interface gráfica; 6. PyQt6.QtGui /QPixmap: Para importar imagem de logo; 7. datetime /datetime: Para inserir data à criação de contas e transações; 8. sys: Para iniciar janelas de interface gráfica; 9. cx\_Freeze: Para gerar arquivo executável.   O sistema foi elaborado seguindo o modelo MVC (Model-View-Controller), que é um modelo onde existem as classes de cada entidade:  Agencia, Cliente, Conta, MatrizSOD, PerfilAcesso, PerfilUsuario, Sistema, Transacao e Usuario.  Cada classe possui um respetivo arquivo na pasta model, bem como um arquivo de controle de dados na pasta controller: agenciaControl.py, clienteControl.py, contaControl.py, matrizControl.py, pAcessoControl.py, pUsuarioControl.py, sistemaControl.py, transacaoControl.py e usuarioControl.py.  As interfaces gráficas estão localizadas na pasta view: telaLogin.py, telaAdministrador.py, telaGerente.py, telaEscriturario.py e telaCaixa.py, cada uma destas está relacionada a um perfil e ao login. Cada tela possui um arquivo com uma classe GUI() relacionada para o vínculo às classes e aos controles.  Existe ainda uma pasta “data” que contém um arquivo de conexão à base de dados MySQL com todas as queries de conexão com o MySQL, criação da base de dados e das tabelas, comandos genéricos de seleção, inserção, atualização, remoção e verificação de dados e campos, sendo as mesmas adaptáveis aos campos de cada tabela específica. Existe também uma pasta “xml” contendo oito arquivos: agencias.xml, clientes.xml, limites\_contas.xml, matriz\_sods.xml, perfil\_acessos.xml, sistemas.xml, tipo\_contas.xml e usuarios.xml, cada um destes relacionado a uma tabela criada na base de dados. Perfil de usuários, contas e transações são supostos serem criados nos testes.  Além disso, assim que o sistema é instanciado em cada perfil (na inserção de usuário e senha), o arquivo conexao.py (dentro da pasta “data”) é executado e se não houver base de dados, tabelas ou dados, estes são criados/inseridos automaticamente: para cada tabela são inseridos automaticamente alguns dados para teste provenientes dos arquivos xml, bem como quatro usuários (Administrador, Gerente, Escriturário e Caixa).  Na raiz do projetos ainda existem seis arquivos: guiAdministrador.py, guiEscriturario.py, guiGerente.py e guiCaixa.py, cada um para um perfil e atrelado a uma view, o guiLogin.py, que valida os usuários cadastrados no sistema e ainda o arquivo principal (main.py) vinculado à todas as GUI’s.  Em cada pasta e no diretório raiz existe um arquivo \_\_init\_\_.py, contendo uma variável “all” com uma lista de todas as pastas ou arquivos que deve ser visto especificamente em cada diretório, para uma melhor organização e uma execução mais rápida e eficiente dos arquivos.  O programa executável foi gerado dentro de um ambiente virtual para Python utilizado a biblioteca “cx-Freeze” e um arquivo com os parâmetros setup.py no diretório raiz do projeto.  Com relação ao planejamento do projeto e ao cronograma, pude aprender com pesquisas como desenvolver o projeto dentro do cronograma proposto e assim, obtive êxito em concluir o mesmo dentro do prazo estipulado.  Foram feitos testes e ajustes ao sistema bancário e, ao que parece, está tudo funcionando como o planejado. |