Roteiro A3 programação

Funcionalidade conforme proposta- Murilo

Olá, eu sou Murilo e vou apresentar o nosso projeto . Ele Marketplace de veículos. Na tela inicial o Usuário pode fazer login ou se cadastrar. Após o login abri-se a tela de menu, lá o usuário tem acesso as opções de anunciar um veículo, pesquisar um veículo, excluir o seu perfil da plataforma(mediante cpf e senha fornecidos no seu cadastro), excluir um anúncio desejado(mediante o cpf fornecido no cadastro e o código gerado no anúncio do veículo), atualizar o seu cadastro, atualizar o anúncio de veículo mediante o fornecimento do código do veículo e do cpf do usuário, e ele pode ver o seu perfil e sair da plataforma .

Integração com banco de dados- Nícolas

A integração com o banco de dados foi feita a partir de uma classe com o nome sqlA3, responsável pela conexão com o banco, e é chamada, junto com a connection nomeada por nós como “conectar” , nas classe que fazem alguma ação com o banco, como por exemplo a classe inicial, na opção de login:

```
try (Connection c = sqlA3.conectar()) {
```

vou usar a classe a opção de login da classe inicial como exemplo, mas a lógica se estende para a maneira como realizamos as ações com o banco de dados em outras classes, é claro mudando as ações realizadas e a proposta da classe. Nesse exemplo, estamos executando uma consulta SQL para verificar se há um usuário com o email e senha fornecidos no banco de dados. Usamos um ‘Prepared Statement’ para evitar injeção de SQL. Depois de executar a consulta, iteramos pelos resultados usando um ‘ResultSet’. Contamos o número de resultados e realizamos ações com base no número de resultados encontrados: Se encontrar exatamente um resultado, faz login normalmente . Se encontrar mais de um resultado, é tratado como cadastro duplicado. Se não encontrarmos nenhum resultado, exibimos uma mensagem :”email e senha

errados. Tente novamente”

```
JButton btnNewButton = new JButton("ENTRAR");
btnNewButton.setFont(new Font("Arial Black", Font.PLAIN, 10));
btnNewButton.addActionListener(new ActionListener() {
    @SuppressWarnings("deprecation")
    public void actionPerformed(ActionEvent e) {
        if(TEXTFIELDEMAIL.getText().isEmpty()) {
            JOptionPane.showMessageDialog(null, "preencha o email");
        }
        if(passwordField.getText().isEmpty()) {
            JOptionPane.showMessageDialog(null, "preencha a senha");
        } else {
            String query = "select * from usuarios where email=? and senha=?";
            try (Connection c = sqlA3.conectar()) {
                PreparedStatement ps = c.prepareStatement(query);
                ps.setString(1, TEXTFIELDEMAIL.getText());
                ps.setString(2, passwordField.getText());
                ResultSet rs = ps.executeQuery();

                int count = 0;

                while (rs.next()) {
                    count = count + 1;
                }
                if (count == 1) {

                    userEmail = TEXTFIELDEMAIL.getText();
                    userPassword = passwordField.getText();
                    MENUpri pri = new MENUpri(userEmail, userPassword);
                    pri.setVisible(true);
                    dispose();

                } else if (count > 1) {
                    JOptionPane.showMessageDialog(null, "Duplicado email e senha");
                } else {
                    JOptionPane.showMessageDialog(null, "email e senha não estão corretos Tente Novamente");
                }

                rs.close();
                ps.close();
            } catch (SQLException erro) {
                erro.printStackTrace();
            }
        }
    }
});
```

Vale mencionar que utilizamos o Driver SQL conector ‘’’’’’’sei lá qual versão depois coloca aqui’’’’’’’’’’ .

Utilização dos conceitos POO e Organização do Código -Maurício/Rodrigo

Rodrigo -Herança:

A classe MENUpri estende JFrame, o que é um exemplo de herança, aproveitando funcionalidades e comportamentos de JFrame.

Tratamento de Exceções:

O código trata exceções usando blocos try-catch. Por exemplo, ao lidar com operações de banco de dados, o código envolve o código relevante em um bloco try e captura possíveis exceções.

Maurício- Encapsulamento:

Embora não tão proeminente neste código, a classe MENUpri encapsula o comportamento e os dados relevantes da aplicação dentro dela. As variáveis são privadas e acessadas por métodos públicos.

Instanciação de Objetos:

Vários objetos são instanciados dentro dos listeners de eventos quando certos botões são clicados. Por exemplo, CadCAR, Pesquisa, excluirANU, etc. Isso mostra a criação de objetos para executar ações específicas

Utilização dos componentes visuais- Marcelo

Utilizamos o Window Builder no Eclipse para criar a interface de nosso projeto. Componentes como Jlabel foram usados para armazenar textos estáticos e imagens. Já os TextFields e os Passowrdfields foram implementados para que o usuário possa colocar o texto conforme necessário. Utilizamos em alguns momentos os Jpanel para o carregamento na interface de dados vindos do banco, como os veículos disponíveis na classe de menu e pesquisa. Fala mencionar, que usamos os botões do tipo JButtom para disparar ações quando pressionados como, por exemplo, a interação com o banco .