DOUGLAS DE AQUINO SANCHES
FELIPE MATSUDA KUNIOKA
GIOVANNI ALEIXO DE SOUZA
RONNI APARECIDO FERNANDES DE OLIVEIRA

ATIVIDADE INTEGRADORA [AVA3]:

ALGORITIMOS E ESTRUTURAS DE DADOS

Londrina
2022

DOUGLAS DE AQUINO SANCHES
FELIPE MATSUDA KUNIOKA
GIOVANNI ALEIXO DE SOUZA
RONNI APARECIDO FERNANDES DE OLIVEIRA

ATIVIDADE INTEGRADORA [AVA3]:

ALGORITIMOS E ESTRUTURAS DE DADOS

Atividade Integradora da disciplina Algorítimos e Estruturas de Dados apresentada ao Curso de Análise e Desenvolvimento de Sistemas do Centro Universitário Filadélfia - UniFil.

Londrina
2022

Contextualização

O aluno deverá integrar os conhecimentos adquiridos nas disciplinas que compõem o módulo de oferta de modo a demonstrar que consegue não apenas compreender, mas também aplicar os conceitos e conhecimentos inerentes às disciplinas do módulo.

A proposta é construir a parte lógica da StartUP que está sendo trabalhada na disciplina de extensão.

Em qualquer linguagem de programação, o uso das estruturas de dados é um recurso amplamente utilizado.

Resumo da atividade

Pense em um módulo ou na tela de seu aplicativo e construa a implementação utilizando uma ou várias estruturas de dados apresentadas na disciplina.

1- Explicar qual parte do aplicativo será trabalhada nesta atividade. (Ex.: cadastro de cliente, cadastro de equipamento etc.).

2- Definir a(s) estrutura(s) de dado(s) que estará(ão) incluída(s) nesta atividade. (Ex.: fila, pilha etc.).

3- Apresentar a codificação

1. PROJETO INTEGRADOR

A atividade desenvolvida na disciplina de Extensão será sobre a Startup “VIAJANTES” – Grupo 9.

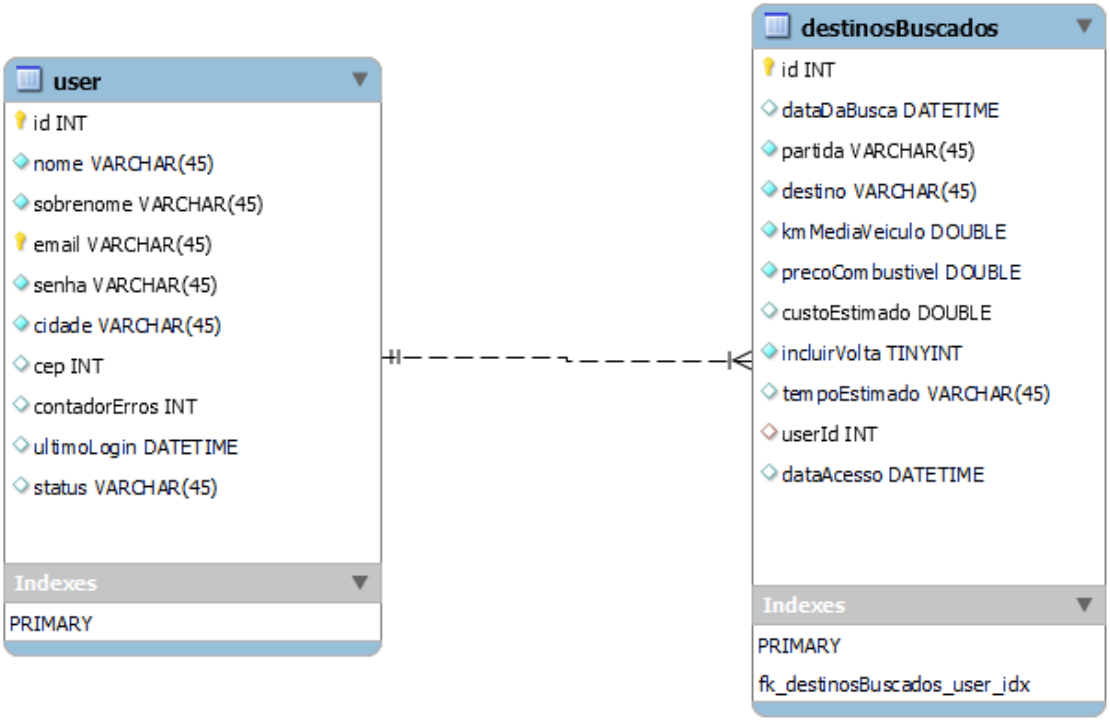
O projeto baseia-se em desenvolver um aplicativo para planejamento de viagens de carro.

Espera-se com o aplicativo poder calcular o custo estimado da viagem, a distância e o tempo a ser incorrido no trajeto.

A estrutura do banco de dados no *App* é bem simples, pois, como seu uso será pelo celular, um banco de dados robusto com informações de destinos e mapas não comportaria ser instalado em dispositivos off-line.

Espera-se buscar as informações mais complexas relacionadas ao destino buscado através do acesso à uma API Web pública (Google Maps, por exemplo).

A figura abaixo mostra o diagrama de entidade e relacionamento do projeto.

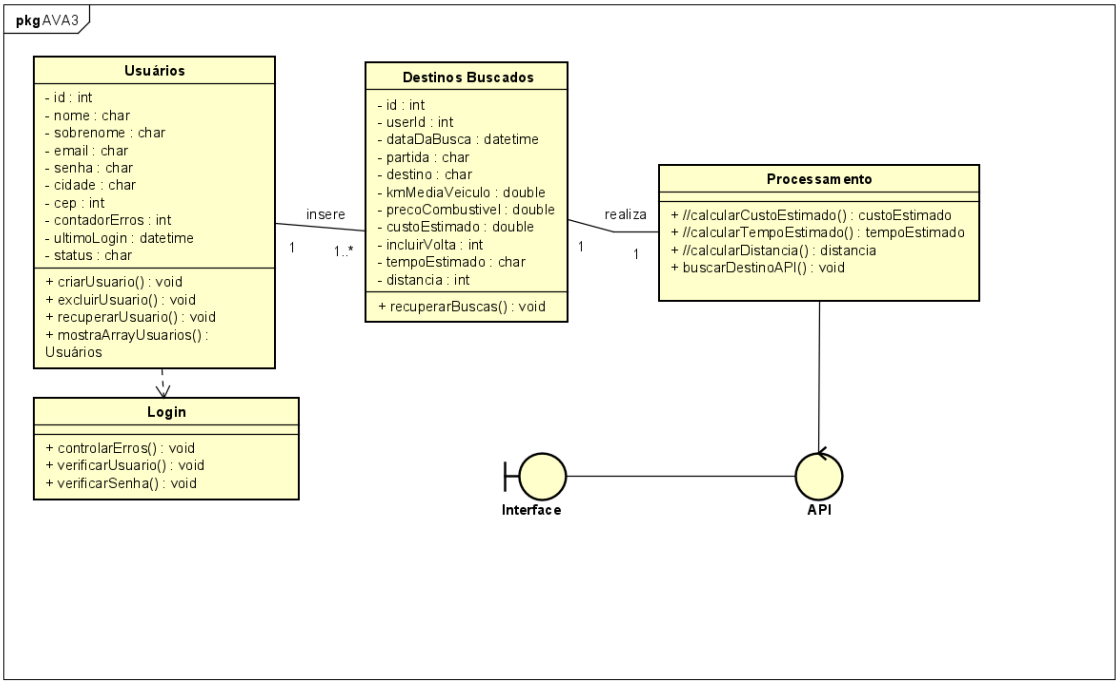


Para o exemplo, foi criado uma estrutura de *array* **Usuarios** populada com as seguintes informações:

ID | NOME | SOBRENOME | EMAIL | SENHA | CIDADE | CEP | CONTADOR ERROS | ULTIMO LOGIN | STATUS
221454113 | Douglas | Sanches | douglsanches@gmail.com | 456 | Londrina | 80010010 | 0 | 10/09/2022 | Desbloqueado |
221454060 | Felipe | Kunioka | felipematsuda17@gmail.com | 789 | Londrina | 80010010 | 0 | 10/09/2022 | Desbloqueado |
222454038 | Giovanni | Souza | giovanni_aleixo@edu.unifil.br | abc | Londrina | 80010010 | 0 | 10/09/2022 | Desbloqueado |
222454140 | Ronni | Oliveira | ronni.fernandes@uol.com.br | 123 | Apucarana | 86810390 | 0 | 10/09/2022 | Desbloqueado |

Nesta atividade será implementada a classe **Login**, e os métodos **<<controlarErros()>>**, **<<verificarUsuario()>>** e **<<verificarSenha()>>**.

Com base nisto, a figura a seguir mostra o diagrama de classes (inicial) do projeto:



Ao executar a classe **main**, o usuário é levado a se autenticar no sistema preenchendo seu login - que é o número de identificação da matrícula Unifil, e com a chamada do método <<verificarUsuario()>> ocorrerá sua validação .

Se o usuário for inválido, ou seja, não pertencer ao *array*, o sistema será encerrado.

Se o usuário for válido, o sistema registrará a data do acesso no array e será solicitada a senha do usuário.

A senha digitada é comparada com a senha armazenada no array, através da chamada do método <<verificarSenha()>>.

Caso a senha inserida seja correta (<<isSenhaValida>>), o sistema irá registrar no a*rray* a data do acesso e o status do usuário para “Ativo”. E para comprovação didática do funcionamento do código, será exibido o array com as alterações pertinentes através do método <<mostrarArrayUsuarios>>.

Caso a senha informada seja incorreta (<<isSenhaInvalida>>), o contador de erros é incrementado e o vetor ***Usuários****[contadorErros]* é atualizado. O usuário poderá realizar mais outras 02 tentativas de acesso, sendo obrigado a reinserir a senha, que

será novamente verificada com a chamada do método <<verificarSenha()>>.

Neste caso, se a senha inserida for correta, o sistema irá registrar no a*rray* o status do usuário para “Ativo”.

Se a senha for incorreta, após 03 tentativas, o sistema registrará o usuário como “Bloqueado”. Com isso, também será exibido o array com as alterações pertinentes através do método <<mostrarArrayUsuarios>>.

A seguir, apresentamos o algoritmo implementado na linguagem Java, para atendimento do proposto nesta atividade integradora.

2. ALGORITMO STARTUP “VIAJANTES” – CLASSE LOGIN

```
import java.util.Calendar;
import java.text.SimpleDateFormat;
import java.util.Scanner;

public class Login {

   public static void main(String[] args) {

      String[][] Usuarios = { //ID, NOME , SOBRENOME, EMAIL, SENHA, CIDADE, CEP, CONTADOR ERROS, ULTIMO LOGIN, STATUS
            {"221454113", "Douglas", "Sanches", "douglsanches@gmail.com", "456", "Londrina", "80010010", "0", "10/09/2022", "Desbloqueado"},
            {"221454060", "Felipe", "Kunioka", "felipematsuda17@gmail.com", "789", "Londrina", "80010010", "0", "10/09/2022", "Desbloqueado"},
            {"222454038", "Giovanni", "Souza", "giovanni_aleixo@edu.unifil.br", "abc", "Londrina", "80010010", "0", "10/09/2022", "Desbloqueado"},
            {"222454140", "Ronni", "Oliveira", "ronni.fernandes@uol.com.br", "123", "Apucarana", "86810390", "0", "10/09/2022", "Desbloqueado"}};

      Scanner entrada = new Scanner(System.in);
      boolean usuarioEncontrado = false;
      int erro = 0;

      String dataHoraAtual = new SimpleDateFormat("dd/MM/yyyy HH:mm:ss").format(Calendar.getInstance().getTime());

      System.out.print("\nDigite o usuario (MATRICULA UNIFIL): ");
      String usuarioDigitado = entrada.next();
      verificarUsuario(Usuarios, entrada, usuarioEncontrado, usuarioDigitado, erro, dataHoraAtual);
   }

   private static void verificarUsuario(String[][] Usuarios, Scanner entrada, boolean usuarioEncontrado,
                                        String usuarioDigitado, int erro, String dataHoraAtual) {
      for (int i = 0; i < Usuarios.length; i++) {
         String usuario = Usuarios[i][0];
         String senha = Usuarios[i][4];

         //SE O USUARIO FOR ENCONTRADO
         if (usuario.equals(usuarioDigitado)) {
            usuarioEncontrado = true;
            Usuarios[i][8] = dataHoraAtual;
            System.out.println("\nBem vindo " + Usuarios[i][1]);
            System.out.println(Usuarios[i][8]);
            System.out.print("\nDigite a senha: ");
            String senhaDigitada = entrada.next();
            verificarSenha(senha, senhaDigitada, erro, Usuarios, i);

         }
      }
      // APÓS PERCORRER O ARRAY, SE O USUARIO NÃO FOR ENCONTRADO
      if (usuarioEncontrado == false)
         System.out.println("Usuario " + usuarioDigitado + " não encontrado");
   }

   private static void verificarSenha(String Senha, String senhaDigitada, int erro, String[][] Usuarios, int i) {
      boolean senhaValida = false;
      String senha = Senha;

      //VERIFICA SE A SENHA É VALIDA OU INVALIDA
      senhaValida = isSenhaValida(senhaDigitada, senhaValida, senha, Usuarios, i);
      isSenhaInvalida(senha, erro, senhaValida, Usuarios, i);
   }

   private static boolean isSenhaValida(String senhaDigitada, boolean senhaValida, String senha, String[][] Usuarios, int i) {
      // SE A SENHA FOR VÁLIDA
      if (senha.equals(senhaDigitada)) {
         senhaValida = true;
         Usuarios[i][9] = "Ativo";
         System.out.println("Acesso concedido.\n\n");
         mostrarArrayUsuarios(Usuarios);
      }
      return senhaValida;
   }
```

```
    private static void isSenhaInvalida(String senha, int erro, boolean senhaValida, String[][] Usuarios, int i) {
        //SE A SENHA FOR INVALIDA
        if (senhaValida == false) {
            System.out.println("Senha incorreta");
            erro++;
            Usuarios[i][7] = Integer.toString(erro); // CONTAGEM DE ERROS ATRIBUIDA AO VETOR

            //VERIFICA AS TENTATIVAS ERRADAS
            if (erro < 3) {
                System.out.println("\n" + (erro + 1) + "ª tentativa");
                System.out.print("Digite a senha novamente: ");
                Scanner entrada = new Scanner(System.in);
                String senhaDigitada = entrada.next();
                verificarSenha(senha, senhaDigitada, erro, Usuarios, i);
            } else {
                System.out.println("Número de tentativas excedidas");
                System.out.println("Usuário bloqueado");
                Usuarios[i][9] = "Bloqueado"; // ALTERA O STATUS DO USUARIO PARA "Bloqueado"
                mostrarArrayUsuarios(Usuarios); // MOSTRA O ARRAY ALTERADO
            }
        }
    }


    private static void mostrarArrayUsuarios(String[][] Usuarios) {
        System.out.println("\nMostrando o Array USUARIOS. " +
                "\nVerifique as alterações nos dados do ultimo login, status do usuário, bem como a contagem de erros, se for o caso:");
        System.out.println("\nID              | NOME | SOBRENOME | EMAIL | SENHA | CIDADE | CEP | CONTADOR ERROS | ULTIMO LOGIN | STATUS
USUARIO");
        for (int i = 0; i < Usuarios.length; i++) {
            for (int j = 0; j < Usuarios[0].length; j++) {
                System.out.print(Usuarios[i][j] + " | ");
            }
            System.out.println("");
        }
    }
}
```

Referências

Utilizando o básico do Workbench Disponível em: https://www.youtube.com/watch?v=P6P1a4TpmvY Acesso em: 06/09/2022

Material do curso EAD – Introdução à Banco de Dados UNIFIL EAD. Acesso em 06 set. 2022

Material do curso EAD – Algoritmos e Estrutura de Dados UNIFIL EAD. Acesso em 09 set. 2022

Material do curso EAD – Análise e Projeto Orientado a Objetos UNIFIL EAD. Acesso em 12 set. 2022