

## Exercício para fixação

Vamos criar um sistema de atendimento de suporte. Quando o usuário necessita de ajuda sobre algum equipamento que não está funcionando corretamente é aberto um chamado para o suporte técnico.

Para começar vamos criar um banco de dados no mysql

Para guardar estes dados será necessário criar um banco de dados no mysql.

**Crie um banco chamado suportedb e uma tabela com o nome de **chamado****

```
create database suportedb charset=utf8mb4 collate=utf8mb4_general_ci;

create table chamado(

idchamado bigint auto_increment primary key,

solicitacao varchar(200) not null,

departamentosolicitado varchar(200) not null,

descricao Chamado text not null,

dataabertura varchar(20),

dataresolucao varchar(20),

statuschamado varchar(20),

observacoes text,

atendente varchar(200)

)engine innodb charset=utf8mb4 collate=utf8mb4_general_ci;
```

**Cadastre uma linha de registro de um problema que tenha ocorrido**

**Agora, crie um projeto java e adicione os seguintes pacotes:**

br.com.dominio

br.com.dao

br.com.view

**No pacote domínio crie um arquivo java com o nome de Chamado.java. Acompanhe o código**

```
public class Chamado {
    private Long idChamado;
    private String solicitacao;
    private String depSolicitado;
    private String descChamado;
    private String dataAbertura;
    private String dataResolucao;
    private String statusChamado;
    private String observacoes;
    private String atendente;

    public Chamado() {}
}
```

```

public Chamado(Long idChamado, String solicitacao, String depSolicitado, String descChamado,
String dataAbertura,
                String dataResolucao, String statusChamado, String observacoes, String
atendente) {
    this.idChamado = idChamado;
    this.solicitacao = solicitacao;
    this.depSolicitado = depSolicitado;
    this.descChamado = descChamado;
    this.dataAbertura = dataAbertura;
    this.dataResolucao = dataResolucao;
    this.statusChamado = statusChamado;
    this.observacoes = observacoes;
    this.atendente = atendente;
}

public Long getIdChamado() {
    return idChamado;
}

public void setIdChamado(Long idChamado) {
    this.idChamado = idChamado;
}

public String getSolicitacao() {
    return solicitacao;
}

public void setSolicitacao(String solicitacao) {
    this.solicitacao = solicitacao;
}

public String getDepSolicitado() {
    return depSolicitado;
}

public void setDepSolicitado(String depSolicitado) {
    this.depSolicitado = depSolicitado;
}

public String getDescChamado() {
    return descChamado;
}

public void setDescChamado(String descChamado) {
    this.descChamado = descChamado;
}

public String getDataAbertura() {
    return dataAbertura;
}

public void setDataAbertura(String dataAbertura) {
    this.dataAbertura = dataAbertura;
}

public String getDataResolucao() {
    return dataResolucao;
}

public void setDataResolucao(String dataResolucao) {
    this.dataResolucao = dataResolucao;
}

public String getStatusChamado() {
    return statusChamado;
}

public void setStatusChamado(String statusChamado) {
    this.statusChamado = statusChamado;
}

public String getObservacoes() {
    return observacoes;
}

```

```

    public void setObservacoes(String observacoes) {
        this.observacoes = observacoes;
    }

    public String getAtendente() {
        return atendente;
    }

    public void setAtendente(String atendente) {
        this.atendente = atendente;
    }
}

```

**Agora, vamos criar os arquivos do DAO. Essa camada é responsável por persistir os dados em banco. Será realizada as seleções, cadastros, atualização e deleção dos dados.**

**No pacote br.com.dao crie o arquivo Conexao.java**

```

public abstract class Conexao {

    protected Connection conn = null;
    protected PreparedStatement pst = null;
    protected ResultSet rs = null;

    public void abrirConexao() {
        try {
            Class.forName("com.mysql.cj.jdbc.Driver").newInstance();

            conn =
DriverManager.getConnection("jdbc:mysql://127.0.0.1:6556/dignadb?useTimeZone=true", "root",
"senac@123");
        }
        catch(SQLException se) {
            se.printStackTrace();
        }
        catch(Exception e) {
            e.printStackTrace();
        }
    }

    public void fecharConexao() {
        try {
            conn.close();
        }
        catch(SQLException se) {
            se.printStackTrace();
        }
        catch(Exception e) {
            e.printStackTrace();
        }
    }
}

```

**Agora, crie a interface CRUD.java ainda no pacote br.com.dao**

```

public interface CRUD<T> {
    String registrar(T obj);
    List<T> listar();
    List<T> listar(T obj);
    T atualizar(T obj);
    String apagar(T obj);
}

```

**Vamos criar um arquivo para cadastrar, atualizar, deletar selecionar os dados no banco. Crie o arquivo CRUDChamado**

```

public class CRUDChamado extends Conexao implements CRUD<Chamado>{

    @Override
    public String registrar(Chamado obj) {
        String txt = "";

        try {
            abrirConexao();
            String sql = "INSERT INTO
chamado(solicitacao,depSolicitado,descChamado)values(?,?,?)";
            pst = conn.prepareStatement(sql);

            pst.setString(1, obj.getSolicitacao());
            pst.setString(2, obj.getDepSolicitado());
            pst.setString(3, obj.getDescChamado());

            int i = pst.executeUpdate();
            if(i > 0) {
                txt = "Chamado solicitado com sucesso.";
            }
            else {
                txt = "Não foi possível realizar o chamado.";
            }
        }
        catch(SQLException se) {
            txt = "Erro de SQL -> " + se.getMessage();
        }
        catch(Exception e) {
            txt = "Erro inesperado -> " + e.getMessage();
        }
        finally {
            fecharConexao();
        }
        return txt;
    }

    @Override
    public List<Chamado> listar() {
        List<Chamado> lstChamado = new ArrayList<Chamado>();
        try {
            abrirConexao();
            String sql = "SELECT * FROM chamado";
            pst = conn.prepareStatement(sql);
            /*
             * O comando executeQuery é utilizado para executar comandos
             * de SELECT. O retorno do executeQuery é um ResultSet,
             * portanto é necessário que haja uma variável do tipo ResultSet
             * para guardar o retorno da execução. Estamos usando a variável
             * rs
             * que foi declarada na classe Conexao
             */
            rs = pst.executeQuery();

            while(rs.next()) {
                Chamado chamado = new Chamado();
                chamado.setIdChamado(rs.getLong(1));
                chamado.setSolicitacao(rs.getString(2));
                chamado.setDepSolicitado(rs.getString(3));
                chamado.setDescChamado(rs.getString(4));
                chamado.setDataAbertura(rs.getString(5));
                chamado.setDataResolucao(rs.getString(6));
                chamado.setStatusChamado(rs.getString(7));
                chamado.setAtendente(rs.getString(8));

                lstChamado.add(chamado);
            }

        }
        catch(SQLException se) {
            se.printStackTrace();
        }
        catch(Exception e) {
            e.printStackTrace();
        }
    }
}

```

```

        }
        finally {
            fecharConexao();
        }

        return lstChamado;
    }

    @Override
    public List<Chamado> listar(Chamado obj) {
        List<Chamado> lstChamado = new ArrayList<Chamado>();

        try {
            abrirConexao();
            String sql = "";

            if(obj.getIdChamado() != 0) {
                sql = "Select * from chamado WHERE statusChamado = 'Pendente'
and idChamado=" + obj.getIdChamado();
            }
            else if(obj.getDepSolicitado() != null) {
                sql = "Select * from chamado WHERE statusChamado = 'Pendente'
and depSolicitado like " + obj.getDepSolicitado();
            }
            else if(obj.getSolicitacao() != null) {
                sql = "Select * from chamado WHERE statusChamado = 'Pendente'
and solicitacao like " + obj.getSolicitacao() + "%";
            }
            else if(obj.getDescChamado() != null) {
                sql = "Select * from chamado WHERE statusChamado = 'Pendente'
and descChamado like %" + obj.getDescChamado() + "%";
            }
            else if(obj.getDataAbertura() != null) {
                sql = "Select * from chamado WHERE statusChamado = 'Pendente'
and dataAbertura like " + obj.getDataAbertura();
            }
            else if(obj.getStatusChamado() != null) {
                sql = "Select * from chamado WHERE statusChamado = 'Pendente'
and statusChamado like " + obj.getStatusChamado();
            }
            else {
                sql = "Select * from chamado";
            }

            pst = conn.prepareStatement(sql);
            rs = pst.executeQuery();

            /*
             * Enquanto houver dados na tabela de banco de dados o laço
             * continua a buscar os dados. O comando next() do rs captura
             * dados da tabela e adiciona em novo curso.
             */
            while(rs.next()) {
                Chamado chamado = new Chamado();
                chamado.setIdChamado(rs.getLong(1));
                chamado.setSolicitacao(rs.getString(2));
                chamado.setDepSolicitado(rs.getString(3));
                chamado.setDescChamado(rs.getString(4));
                chamado.setDataAbertura(rs.getString(5));
                chamado.setDataResolucao(rs.getString(6));
                chamado.setStatusChamado(rs.getString(7));
                chamado.setAtendente(rs.getString(8));

                lstChamado.add(chamado);
            }
        }
        catch(SQLException se) {
            se.printStackTrace();
        }
        catch(Exception e) {
            e.printStackTrace();
        }
        finally {
            fecharConexao();
        }
    }

```

```

        }
        return lstChamado;
    }

    @Override
    public Chamado atualizar(Chamado obj) {
        Chamado chamado = new Chamado();
        try {
            abrirConexao();
            String sql = "UPSTRING chamado SET dataResolucao=?, statusChamado=?,
atendente=?, observacoes=? WHERE idChamado=?";
            pst = conn.prepareStatement(sql);
            pst.setString(1, obj.getDataResolucao());
            pst.setString(2, obj.getStatusChamado());
            pst.setString(3, obj.getAtendente());
            pst.setString(4, obj.getObservacoes());
            pst.setLong(5, obj.getIdChamado());

            int i = pst.executeUpdate();
            if(i > 0) {
                chamado = obj;
            }
            else {
                throw new Exception("Não foi possível atualizar os dados");
            }
        }

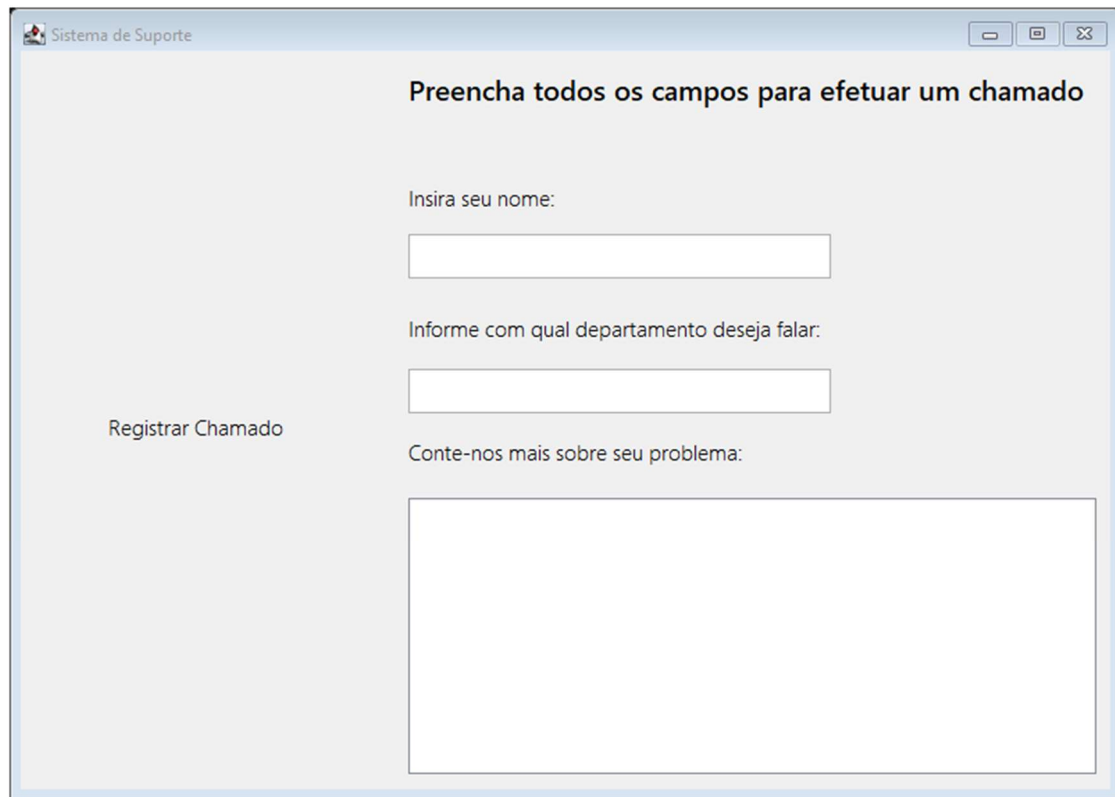
        catch(SQLException se) {
            se.printStackTrace();
        }
        catch(Exception e) {
            e.printStackTrace();
        }
        finally {
            fecharConexao();
        }
        return chamado;
    }

    @Override
    public String apagar(Chamado obj) {
        String txt = "";
        try {
            abrirConexao();
            String sql = "DELETE FROM chamado WHERE idChamado=?";
            pst = conn.prepareStatement(sql);
            pst.setLong(1, obj.getIdChamado());

            int i = pst.executeUpdate();
            if(i > 0) {
                txt = "Chamado apagado.";
            }
            else {
                txt = "Não foi possível apagar o chamado.";
            }
        }
        catch(SQLException se) {
            txt = "Erro na consulta de SQL -> " + se.getMessage();
        }
        catch(Exception e) {
            txt = "Ocorreu um erro inesperado durante a consulta -> " +
e.getMessage();
        }
        finally {
            fecharConexao();
        }
        return txt;
    }
}

```

Construa a de abertura de chamado da seguinte forma, em java usando o window builder. Isso será feito no pacote `br.com.view`



Sistema de Suporte

Preencha todos os campos para efetuar um chamado

Insira seu nome:

Informe com qual departamento deseja falar:

Conte-nos mais sobre seu problema:

Registrar Chamado

### Código para Registrar o Chamado

```
Chamado soliChamado = new Chamado();

if(txtNome.getText().trim().equals("") ||
txtDepartamento.getText().trim().equals("") ||
txtDescricao.getText().trim().equals("")) {
    JOptionPane.showMessageDialog(null, "Todos os
campos devem ser preenchidos.", "Erro 4000765x" , JOptionPane.ERROR_MESSAGE);
}
else {

    soliChamado.setSolicitacao(txtNome.getText());

    soliChamado.setDepSolicitado(txtDepartamento.getText());

    soliChamado.setDescChamado(txtDescricao.getText());

    JOptionPane.showMessageDialog(null, cc.registrar(soliChamado));
}
```

Agora, crie outra tela de atendimento ao chamado como segue abaixo

### Codigo do botão atualizar chamado:

```

Chamado cr = new Chamado();

if(txtResponsavel.getText().trim().equals("") || txtStatus.getText().trim().equals("") ||
txtId.getText().trim().equals("") || txtDataResolucao.getText().trim().equals("")) {
    JOptionPane.showMessageDialog(null, "Os campos Responsável Chamado, Id do Chamado,
Status do Chamado e Data de Resolução devem ser preenchidos",
    "Erro 4000765x" , JOptionPane.ERROR_MESSAGE);
}
else {

    cr.setDataResolucao(Date.valueOf(txtDataResolucao.getText()));
    cr.setStatusChamado(txtStatus.getText());
    cr.setAtendente(txtResponsavel.getText());
    cr.setObservacoes(txtObservacoes.getText());
    cr.setIdChamado(id);
    JOptionPane.showMessageDialog(null, cc.atualizar(cr));
}

```

### Codigo do botão Excluir chamados

```

if(id.equals(null)) {
    JOptionPane.showMessageDialog(null,"Selecione o chamado a ser excluído.", "Erro
4000770x",JOptionPane.ERROR_MESSAGE);
}
else {
    if(JOptionPane.showConfirmDialog(null, "Você tem certeza desta ação? \nEstá ação é
permanente "+ "e não pode ser desfeita", "ATENÇÃO", JOptionPane.YES_NO_OPTION,
JOptionPane.QUESTION_MESSAGE) == 0) {
        Chamado cr = new Chamado();
        cr.setIdChamado(id);
        JOptionPane.showConfirmDialog(null, cc.apagar(cr));
    }
}

```



**Antes de testar não esqueça de adicionar o driver de comunicação do mysql conector**