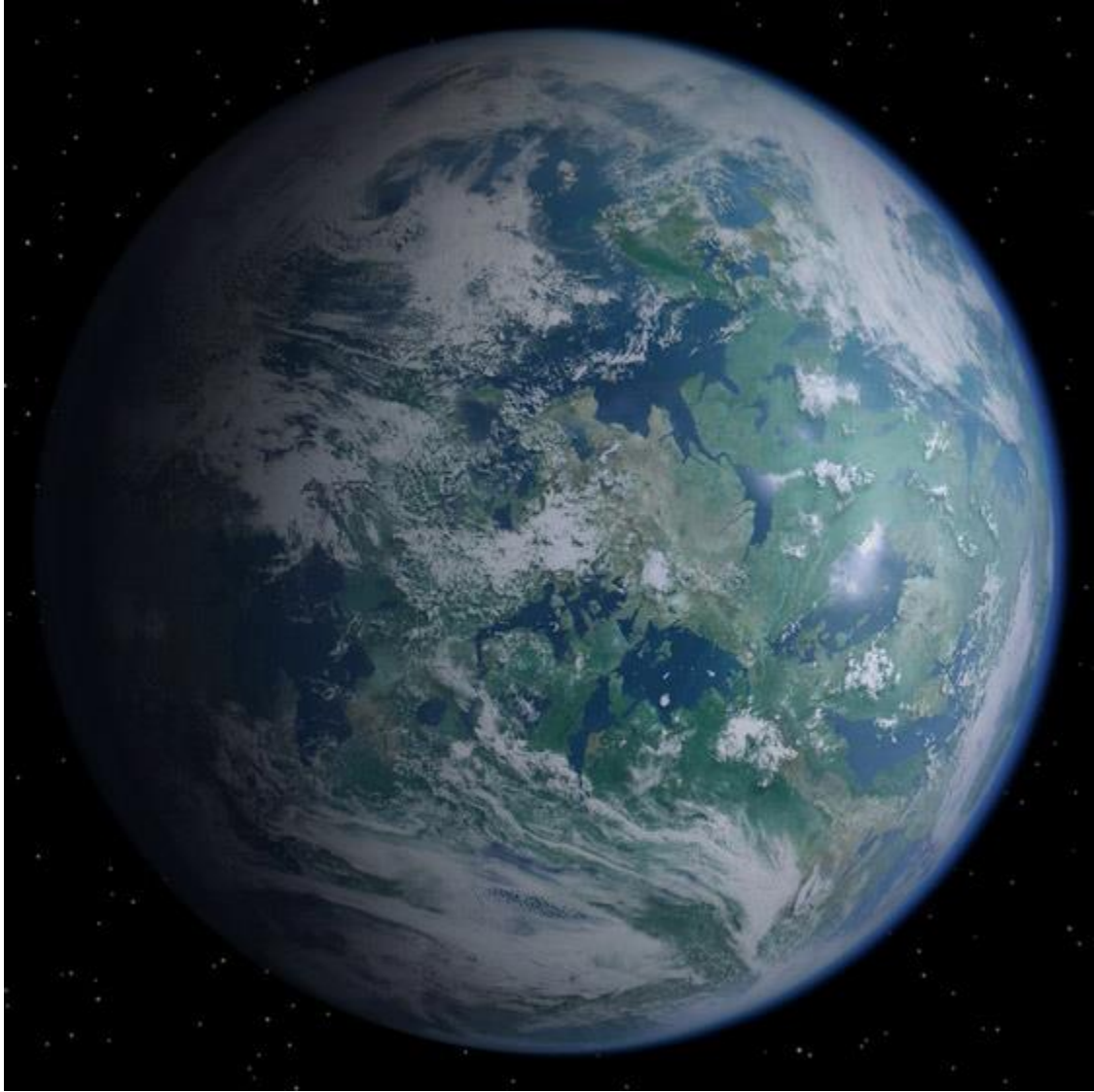

PLANETA ALDERAAN



Sumário

- Descrição do projeto 3
- Modelo ER 5
- CRUDs 6
- Consultas..... 10
- 11

Descrição do projeto

Integrantes:

- Lucas de Matos da Silveira;
- Raul Steffen Lacerda;
- Gabriel Nunes de Siqueira.

Problema:

Em um planeta recém-colonizado certos produtos não estão à disposição de toda a população de forma livre, pois são escassos e necessitam de controle para que se consiga viver de forma harmônica durante o período de colonização.

Solução:

O grupo pensou em fazer um grande sistema de controle de estoque, com localização dos produtos e controle de acesso de pessoas, para que não haja problemas de roubo nem extravio dos produtos ofertados.

Projeto:

Sistema de controle de estoque, com a localização do produto no estoque e o controle de acesso de pessoas integrado ao mesmo.

Requisitos funcionais:

ADM:

- Gerenciar produto (extend)
 - Consultar produto
 - Retirar produto
- Gerenciar usuário

Usuário:

- Gerenciar produto (extend)
 - Consultar produto
 - Retirar produto

Requisitos não funcionais:

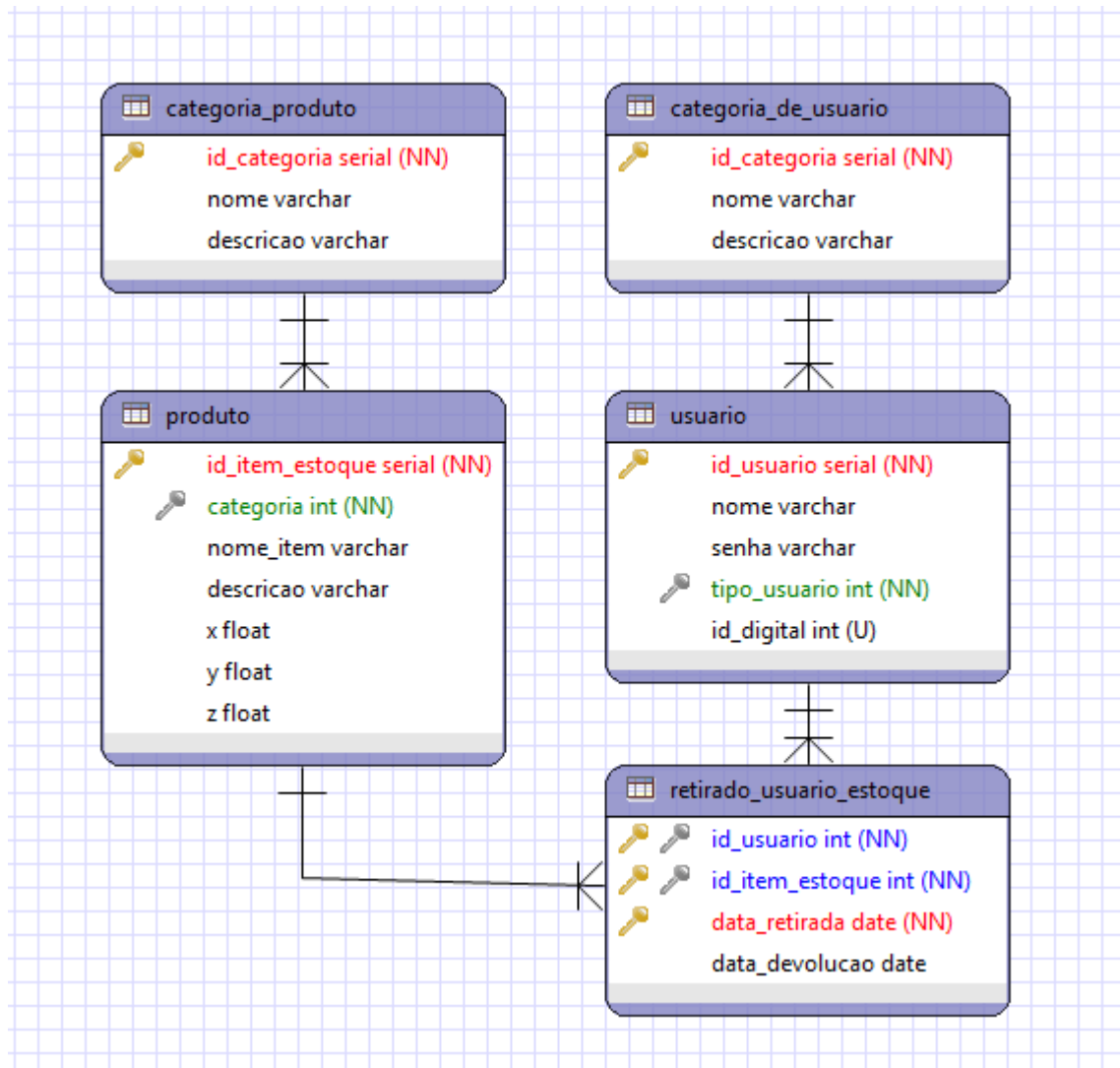
- Ler digitais dos usuários
- Funcionar em plataforma web
- Responder em, no máximo, 2 segundos as consultas

Regra de negócio:

- Avisar aos adms do sistema se alguma pessoa tentar entrar no sistema sem ter cadastro/permissão
- Caso não haja o produto será feita a encomenda do mesmo

-Caso não seja encontrado o produto, deverá ser relatado ao gerenciador que está sob controle do estoque

Modelo ER



CRUDs

Localização

```
45 public Localizacao load(int id) {
46     String selectSQL = "SELECT * FROM localizacao WHERE id_localizacao = " + id + ";";
47     Localizacao local = new Localizacao();
48     Conexao c = new Conexao();
49     Connection dbConnection = c.getConexao();
50     Statement ps;
51     try {
52         ps = dbConnection.createStatement();
53         ResultSet rs = ps.executeQuery(selectSQL);
54         while (rs.next()) {
55             local.setIdLocalizacao(rs.getInt("id_localizacao"));
56             local.setX(rs.getDouble("x"));
57             local.setY(rs.getDouble("y"));
58             local.setZ(rs.getDouble("z"));
59         }
60     } catch (SQLException f) {
61         f.printStackTrace();
62     }
63     return local;
64 }
65
66 public void update(int id) {
67     Conexao c = new Conexao();
68     Connection dbConnection = c.getConexao();
69     PreparedStatement preparedStatement = null;
70     String insertTableSQL = "UPDATE localizacao SET x = ?, y = ?, z = ? WHERE id = ?";
71     try {
72         preparedStatement = dbConnection.prepareStatement(insertTableSQL);
73         preparedStatement.setDouble(1, this.getIdLocalizacao());
74         preparedStatement.setDouble(2, this.x);
75         preparedStatement.setDouble(3, this.y);
76         preparedStatement.setDouble(4, this.z);
77         preparedStatement.executeUpdate();
78     } catch (SQLException e) {
79         e.printStackTrace();
80     }
81 }
82
83 public void insert() {
84     Conexao c = new Conexao();
85     Connection dbConnection = c.getConexao();
86     PreparedStatement ps = null;
87     String insertSQL = "INSERT INTO localizacao(id_localizacao, x, y, z) VALUES (?, ?, ?, ?)";
88     try {
89         ps = dbConnection.prepareStatement(insertSQL);
90         ps.setInt(1, this.getIdLocalizacao());
91         ps.setDouble(2, this.x);
92         ps.setDouble(3, this.y);
93         ps.setDouble(4, this.z);
94         ps.executeUpdate();
95     } catch (SQLException e) {
96         e.printStackTrace();
97     }
98 }
99
100 public void delete(int id) {
101     Conexao C = new Conexao();
102     Connection dbConnection = C.getConexao();
103     PreparedStatement preparedStatement = null;
104     String deleteData = "DELETE FROM localizacao WHERE id_localizacao = " + id;
105     try {
106         preparedStatement = dbConnection.prepareStatement(deleteData);
107         preparedStatement.executeUpdate();
108     } catch (SQLException e2) {
109         e2.printStackTrace();
110     }
111 }
112
113 }
```

Usuário

```
72 public void insert() {
73     Conexao c = new Conexao();
74     Connection dbConnection = c.getConnection();
75     PreparedStatement ps = null;
76     String insertSQL = "INSERT INTO usuario (nome,senha,tipo_usuario) VALUES (?,?,?)";
77     try {
78         ps = dbConnection.prepareStatement(insertSQL);
79         ps.setString(1, this.nome);
80         ps.setString(2, this.senha);
81         ps.setInt(3, this.tipoUsuario);
82         ps.executeUpdate();
83         c.desconecta();
84     } catch (SQLException e) {
85         e.printStackTrace();
86     }
87 }
88
89 public void updateUser(int idUser, int tipo) {
90     Conexao c = new Conexao();
91     Connection dbConnection = c.getConnection();
92     PreparedStatement ps = null;
93     String stringSQL = null;
94     try {
95         switch (tipo) {
96             case 1:
97                 stringSQL = "UPDATE usuario SET nome = ? WHERE id_usuario = " + idUser;
98                 ps = dbConnection.prepareStatement(stringSQL);
99                 ps.setString(1, this.nome);
100                 break;
101             case 2:
102                 stringSQL = "UPDATE usuario SET senha = ? WHERE id_usuario = " + idUser;
103                 ps = dbConnection.prepareStatement(stringSQL);
104                 ps.setString(1, this.senha);
105                 break;
106             case 3:
107                 stringSQL = "UPDATE usuario SET tipo_usuario = ? WHERE id_usuario = " + idUser;
108                 ps = dbConnection.prepareStatement(stringSQL);
109                 ps.setInt(1, this.tipoUsuario);
110                 break;
111         }
112         ps.executeUpdate();
113         c.desconecta();
114     } catch (SQLException e) {
115         e.printStackTrace();
116     }
117 }
118
119 public void delete(int id) {
120     Conexao C = new Conexao();
121     Connection dbConnection = C.getConnection();
122     PreparedStatement preparedStatement = null;
123     String deleteData = "DELETE FROM usuario WHERE id_usuario= " + id;
124
125     try {
126         preparedStatement = dbConnection.prepareStatement(deleteData);
127
128         preparedStatement.executeUpdate();
129         C.desconecta();
130     } catch (SQLException e2) {
131         e2.printStackTrace();
132     }
133 }
134
135 public static ArrayList<Usuario> getAllUsers() {
136     String selectSQL = "SELECT * FROM usuario";
137     Conexao c = new Conexao();
138     ArrayList<Usuario> lista = new ArrayList<>();
139     Connection dbConnection = c.getConnection();
140     Statement ps;
141     try {
142         ps = dbConnection.createStatement();
143         ResultSet rs = ps.executeQuery(selectSQL);
144         while (rs.next()) {
145             Usuario usr = new Usuario();
146             usr.setIdUsuario(rs.getInt("id_usuario"));
147             usr.setNome(rs.getString("nome"));
148             usr.setSenha(rs.getString("senha"));
149             usr.setTipoUsuario(rs.getInt("tipo_usuario"));
150             lista.add(usr);
151         }
152     } catch (SQLException f) {
153         f.printStackTrace();
154     }
155     c.desconecta();
156     return lista;
157 }
158
159 @Override
160 public String toString() {
161     ArrayList<CategoriaUsuario> ctUsr = getAll();
162     String aux = "";
163     for (CategoriaUsuario ctUsr1 : ctUsr) {
164         if (tipoUsuario == ctUsr1.getIdCategoria()) {
165             aux += ctUsr1.getNome();
166         }
167     }
168
169     return "\nId:" + idUsuario + "\nNome: " + nome + "\nTipo: " + aux;
170 }
171
172
173 public static Usuario loadUser(int id) {
174     String selectSQL = "SELECT * FROM usuario WHERE id_usuario= " + id;
175     Usuario user = new Usuario();
176     Conexao c = new Conexao();
177     Connection dbConnection = c.getConnection();
178     Statement ps;
179
180     try {
181         ps = dbConnection.createStatement();
182         ResultSet rs = ps.executeQuery(selectSQL);
183         while (rs.next()) {
184             user.setIdUsuario(rs.getInt("id_usuario"));
185             user.setNome(rs.getString("nome"));
186             user.setSenha(rs.getString("senha"));
187             user.setTipoUsuario(rs.getInt("tipo_usuario"));
188         }
189     }
190
191     } catch (SQLException f) {
192         f.printStackTrace();
193     }
194     c.desconecta();
195     return user;
196 }
197 }
```

Produto

```
103 public ArrayList<Produto> load(String nome) {
104     String selectSQL = "SELECT * FROM produto WHERE nome_item = '" + nome + "'";
105     Produto prod = new Produto();
106     Conexao c = new Conexao();
107     Connection dbConnection = c.getConnection();
108     Statement ps;
109     ArrayList<Produto> lista = new ArrayList<>();
110     try {
111         ps = dbConnection.createStatement();
112         ResultSet rs = ps.executeQuery(selectSQL);
113         while (rs.next()) {
114             prod.setIdItemEstoque(rs.getInt("id_item_estoque"));
115             prod.setNome(rs.getString("nome_item"));
116             prod.setDescricao(rs.getString("descricao"));
117             prod.setCategoria(rs.getInt("categoria"));
118             prod.setX(rs.getDouble("x"));
119             prod.setY(rs.getDouble("y"));
120             prod.setZ(rs.getDouble("z"));
121             lista.add(prod);
122         }
123     } catch (SQLException f) {
124         f.printStackTrace();
125     }
126     c.desconecta();
127     return lista;
128 }
129
130 public static ArrayList<Produto> getAllProducts() {
131     String selectSQL = "SELECT * FROM produto ORDER BY id_item_estoque ASC";
132     Conexao c = new Conexao();
133     ArrayList<Produto> lista = new ArrayList<>();
134     Connection dbConnection = c.getConnection();
135     Statement ps;
136     try {
137         ps = dbConnection.createStatement();
138         ResultSet rs = ps.executeQuery(selectSQL);
139         while (rs.next()) {
140             Produto pdt = new Produto();
141             pdt.setIdItemEstoque(rs.getInt("id_item_estoque"));
142             pdt.setNome(rs.getString("nome_item"));
143             pdt.setCategoria(rs.getInt("categoria"));
144             pdt.setDescricao(rs.getString("descricao"));
145             pdt.setX(rs.getDouble("x"));
146             pdt.setY(rs.getDouble("y"));
147             pdt.setZ(rs.getDouble("z"));
148             lista.add(pdt);
149         }
150     } catch (SQLException f) {
151         f.printStackTrace();
152     }
153     c.desconecta();
154     return lista;
155 }
156
157 public static Produto loadID(int id) {
158     String selectSQL = "SELECT * FROM produto WHERE id_item_estoque = '" + id + "' ORDER BY id_item_estoque ASC";
159     Produto prod = new Produto();
160     Conexao c = new Conexao();
161     Connection dbConnection = c.getConnection();
162     Statement ps;
163     try {
164         ps = dbConnection.createStatement();
165         ResultSet rs = ps.executeQuery(selectSQL);
166         while (rs.next()) {
167             prod.setIdItemEstoque(rs.getInt("id_item_estoque"));
168             prod.setNome(rs.getString("nome_item"));
169         }
170     }
```

```
172         prod.setDescricao(rs.getString("descricao"));
173         prod.setCategoria(rs.getInt("categoria"));
174         prod.setX(rs.getDouble("x"));
175         prod.setY(rs.getDouble("y"));
176         prod.setZ(rs.getDouble("z"));
177     }
178 }
179
180 } catch (SQLException f) {
181     f.printStackTrace();
182 }
183 c.desconecta();
184 return prod;
185 }
186
187 public void update(int id, int tipo) {
188     Conexao c = new Conexao();
189     Connection dbConnection = c.getConnection();
190     PreparedStatement ps = null;
191     String insertTableSQL = null;
192     try {
193         switch (tipo) {
194             case 1:
195                 insertTableSQL = "UPDATE produto SET nome_item = ? WHERE id_item_estoque = " + id;
196                 ps = dbConnection.prepareStatement(insertTableSQL);
197                 ps.setString(1, this.nome);
198                 break;
199             case 2:
200                 insertTableSQL = "UPDATE produto SET descricao = ? WHERE id_item_estoque = " + id;
201                 ps = dbConnection.prepareStatement(insertTableSQL);
202                 ps.setString(1, this.descricao);
203                 break;
204             case 3:
205                 insertTableSQL = "UPDATE produto SET categoria = ? WHERE id_item_estoque = " + id;
206                 ps = dbConnection.prepareStatement(insertTableSQL);
207                 ps.setInt(1, this.categoria);
208                 break;
209             case 4:
210                 insertTableSQL = "UPDATE produto SET x = ? WHERE id_item_estoque = " + id;
211                 ps = dbConnection.prepareStatement(insertTableSQL);
212                 ps.setDouble(1, this.x);
213                 break;
214             case 5:
215                 insertTableSQL = "UPDATE produto SET y = ? WHERE id_item_estoque = " + id;
216                 ps = dbConnection.prepareStatement(insertTableSQL);
217                 ps.setDouble(1, this.y);
218                 break;
219             case 6:
220                 insertTableSQL = "UPDATE produto SET z = ? WHERE id_item_estoque = " + id;
221                 ps = dbConnection.prepareStatement(insertTableSQL);
222                 ps.setDouble(1, this.z);
223                 break;
224         }
225         ps.executeUpdate();
226     } catch (SQLException e) {
227         e.printStackTrace();
228     }
229     c.desconecta();
230 }
231
232 public void insert() {
233     Conexao c = new Conexao();
234     Connection dbConnection = c.getConnection();
235     PreparedStatement ps = null;
236     String insertSQL = "INSERT INTO produto(nome_item, descricao, categoria, x, y, z) VALUES (?, ?, ?, ?, ?, ?)";
237     try {
238         ps = dbConnection.prepareStatement(insertSQL);
239         ps.setString(1, this.nome);
240         ps.setString(2, this.descricao);
241         ps.setInt(3, this.categoria);
242         ps.setDouble(4, this.x);
243         ps.setDouble(5, this.y);
244         ps.setDouble(6, this.z);
245         ps.executeUpdate();
246     } catch (SQLException e) {
247         e.printStackTrace();
248     }
249     c.desconecta();
250 }
251
252 public void delete(int id) {
253     Conexao c = new Conexao();
254     Connection dbConnection = c.getConnection();
255     PreparedStatement preparedStatement = null;
256     String deleteData = "DELETE FROM produto WHERE id_item_estoque = " + id;
257     try {
258         preparedStatement = dbConnection.prepareStatement(deleteData);
259         preparedStatement.executeUpdate();
260     } catch (SQLException e2) {
261         e2.printStackTrace();
262     }
263     c.desconecta();
264 }
```


Categoria Usuário

```
46 public CategoriaUsuario load(String nome) {
47     String selectSQL = "SELECT * FROM categoria_usuario WHERE nome = '" + nome + "'";
48     CategoriaUsuario usrCat = new CategoriaUsuario();
49     Conexao c = new Conexao();
50     Connection dbConnection = c.getConnection();
51     Statement ps;
52     try {
53         ps = dbConnection.createStatement();
54         ResultSet rs = ps.executeQuery(selectSQL);
55         while (rs.next()) {
56             usrCat.setidCategoria(rs.getInt("id_categoria"));
57             usrCat.setNome(rs.getString("nome"));
58             usrCat.setDescricao(rs.getString("descricao"));
59         }
60     } catch (SQLException f) {
61         f.printStackTrace();
62     }
63     c.desconecta();
64     return usrCat;
65 }
66
67
68 public static ArrayList<CategoriaUsuario> getAll() {
69     String selectSQL = "SELECT * FROM categoria_usuario";
70     Conexao c = new Conexao();
71     ArrayList<CategoriaUsuario> lista = new ArrayList<>();
72
73     Connection dbConnection = c.getConnection();
74     Statement ps;
75     try {
76         ps = dbConnection.createStatement();
77         ResultSet rs = ps.executeQuery(selectSQL);
78         while (rs.next()) {
79             CategoriaUsuario usrCat = new CategoriaUsuario();
80             usrCat.setidCategoria(rs.getInt("id_categoria"));
81             usrCat.setNome(rs.getString("nome"));
82             usrCat.setDescricao(rs.getString("descricao"));
83             lista.add(usrCat);
84         }
85     } catch (SQLException f) {
86         f.printStackTrace();
87     }
88     c.desconecta();
89     return lista;
90 }
91
92
93 public void update(int id) {
94     Conexao c = new Conexao();
95     Connection dbConnection = c.getConnection();
96     PreparedStatement preparedStatement = null;
97     String insertTableSQL = "UPDATE categoria_usuario SET nome = '?', descricao = '?' WHERE id_categoria = (?)";
98     try {
99         preparedStatement = dbConnection.prepareStatement(insertTableSQL);
100         preparedStatement.setString(1, this.nome);
101         preparedStatement.setString(2, this.descricao);
102         preparedStatement.setInt(3, this.idCategoria);
103
104         preparedStatement.executeUpdate();
105
106         c.desconecta();
107     } catch (SQLException e) {
108         e.printStackTrace();
109     }
110 }
111
112 public void insert() {
113     Conexao c = new Conexao();
114     Connection dbConnection = c.getConnection();
115     PreparedStatement ps = null;
116     String insertSQL = "INSERT INTO categoria_usuario(nome, descricao) VALUES (?,?)";
117     try {
118         ps = dbConnection.prepareStatement(insertSQL);
119         ps.setString(1, this.nome);
120         ps.setString(2, this.descricao);
121         ps.executeUpdate();
122         c.desconecta();
123     } catch (SQLException e) {
124         e.printStackTrace();
125     }
126 }
127
128 public void delete(int id) {
129     Conexao c = new Conexao();
130     Connection dbConnection = c.getConnection();
131     PreparedStatement preparedStatement = null;
132     String deleteData = "DELETE FROM categoria_usuario WHERE id_categoria = " + id;
133
134     try {
135         preparedStatement = dbConnection.prepareStatement(deleteData);
136
137         preparedStatement.executeUpdate();
138         c.desconecta();
139     } catch (SQLException e2) {
140         e2.printStackTrace();
141     }
142 }
143
144 }
```


Categoria produto

```
46 public static ArrayList<CategoriaProduto> getLista() {
47     String selectSQL = "SELECT * FROM categoria_produto";
48
49     Conexao c = new Conexao();
50     Connection dbConnection = c.getConexao();
51     Statement ps;
52     CategoriaProduto catProd;
53     ArrayList<CategoriaProduto> listProd = new ArrayList<>();
54     try {
55         ps = dbConnection.createStatement();
56         ResultSet rs = ps.executeQuery(selectSQL);
57         while (rs.next()) {
58             catProd = new CategoriaProduto();
59             catProd.setIdCategoria(rs.getInt("id_categoria"));
60             catProd.setNome(rs.getString("nome"));
61             catProd.setDescricao(rs.getString("descricao"));
62             listProd.add(catProd);
63         }
64     } catch (SQLException e) {
65         e.printStackTrace();
66         //System.out.println("TÁ COM ERRO CLASSE CATEGORIAPRODUTO");
67     }
68     c.desconecta();
69     return listProd;
70 }
71
72 public void insert() {
73     Conexao c = new Conexao();
74     Connection dbConnection = c.getConexao();
75     PreparedStatement ps = null;
76     String insertSQL = "INSERT INTO categoria_produto(nome, descricao) VALUES (?,?)";
77     try {
78         ps = dbConnection.prepareStatement(insertSQL);
79         ps.setString(1, this.nome);
80         ps.setString(2, this.descricao);
81
82         ps.executeUpdate();
83         c.desconecta();
84     } catch (SQLException e) {
85         e.printStackTrace();
86     }
87 }
88
89 }
90
```

Consultas

Navegação

Administração


**Re: Planeta Alderaan**
por [Amanda Costa das Almas](#) - segunda, 31 Out 2016, 14:33

Querys

1. Todos os produtos que tenham nome igual a alimento.
2. Todas as categorias que tenham banana.
3. Todas os produtos que fazem parte de uma categoria.
4. Todos os produtos que não possuem categoria.
5. Todas as categorias que não possuem produto.
6. Todos os produtos e todas as categorias.
7. Todos os produtos que não possuem categoria e Todas as categorias que não possuem produto.

67 palavras

[Mostrar principal](#) | [Responder](#)

**Re: Planeta Alderaan**
por [Gabriel Nunes de Siqueira](#) - segunda, 31 Out 2016, 15:08

A proposta 4 possui um problema, o sistema foi modelado para não existir produtos sem categoria, sendo obrigatório a inserção da coluna no cadastro dos dados, o mesmo vale para a proposta 7. É possível que exista uma categoria sem produtos porém não existe um produto sem categoria.


A proposta 6 também acaba se igualando a 3. Não ficou claro a especificação na terceira, deve listar UMA categoria e TODOS seus produtos ou TODAS as categorias com TODOS os produtos? Caso seja de uma específica, qual categoria deverá ser listada?

90 palavras

[Mostrar principal](#) | [Responder](#)

Navegação


Administração

**Re: Planeta Alderaan**
por [Amanda Costa das Almas](#) - segunda, 31 Out 2016, 15:23

Para efetuar a query 6 é necessário que a relação retirado_usuario_estoque seja realmente uma N..N
Da forma como foi modelado ela é 1..N mesmo com a tabela de relação visto que id_item_estoque é UNN, ou seja Unico.

42 palavras

[Mostrar principal](#) | [Responder](#)

**Re: Planeta Alderaan**
por [Lucas de Matos da Silveira](#) - segunda, 7 Nov 2016, 13:27

1 --todos os alimentos que tem como categoria = alimento

```
SELECT nome_item FROM produto p LEFT JOIN categoria_produto cp ON cp.id_categoria = p.categoria WHERE cp.nome = 'Alimento'
```

2--Todas as categorias que tenham banana.

```
SELECT nome FROM categoria_produto cp RIGHT JOIN produto p ON cp.id_categoria = p.categoria WHERE p.nome_item = 'Banana'
```

3--Todas os produtos que fazem parte de uma categoria

```
SELECT nome, nome_item FROM produto p INNER JOIN categoria_produto c ON p.categoria = c.id_categoria ORDER BY nome ASC;
```

5--Todas as categorias que não possuem produto.

```
SELECT nome FROM categoria_produto cp LEFT JOIN produto p ON p.categoria = cp.id_categoria WHERE p.categoria IS NULL
```

112 palavras

[Mostrar principal](#) | [Responder](#)