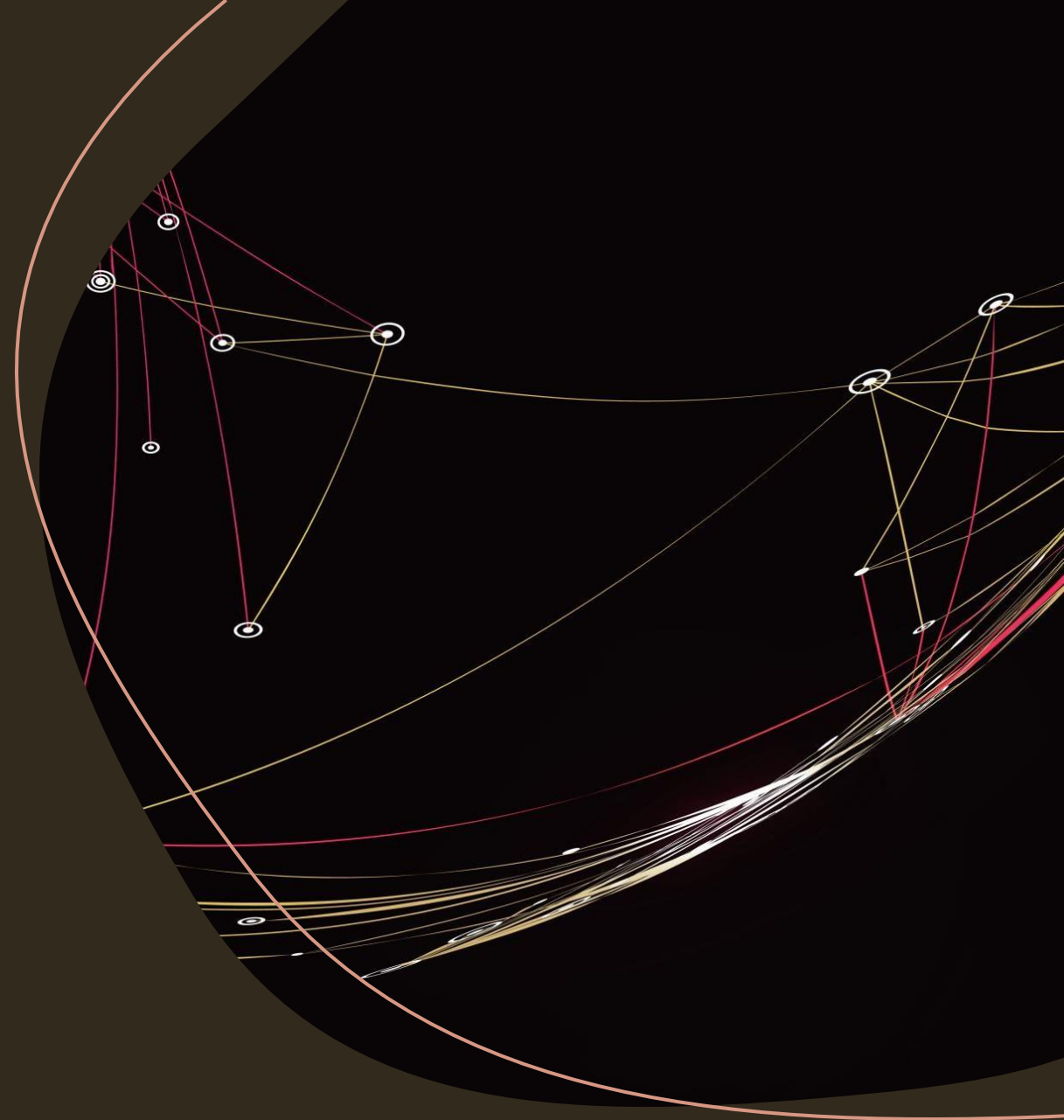


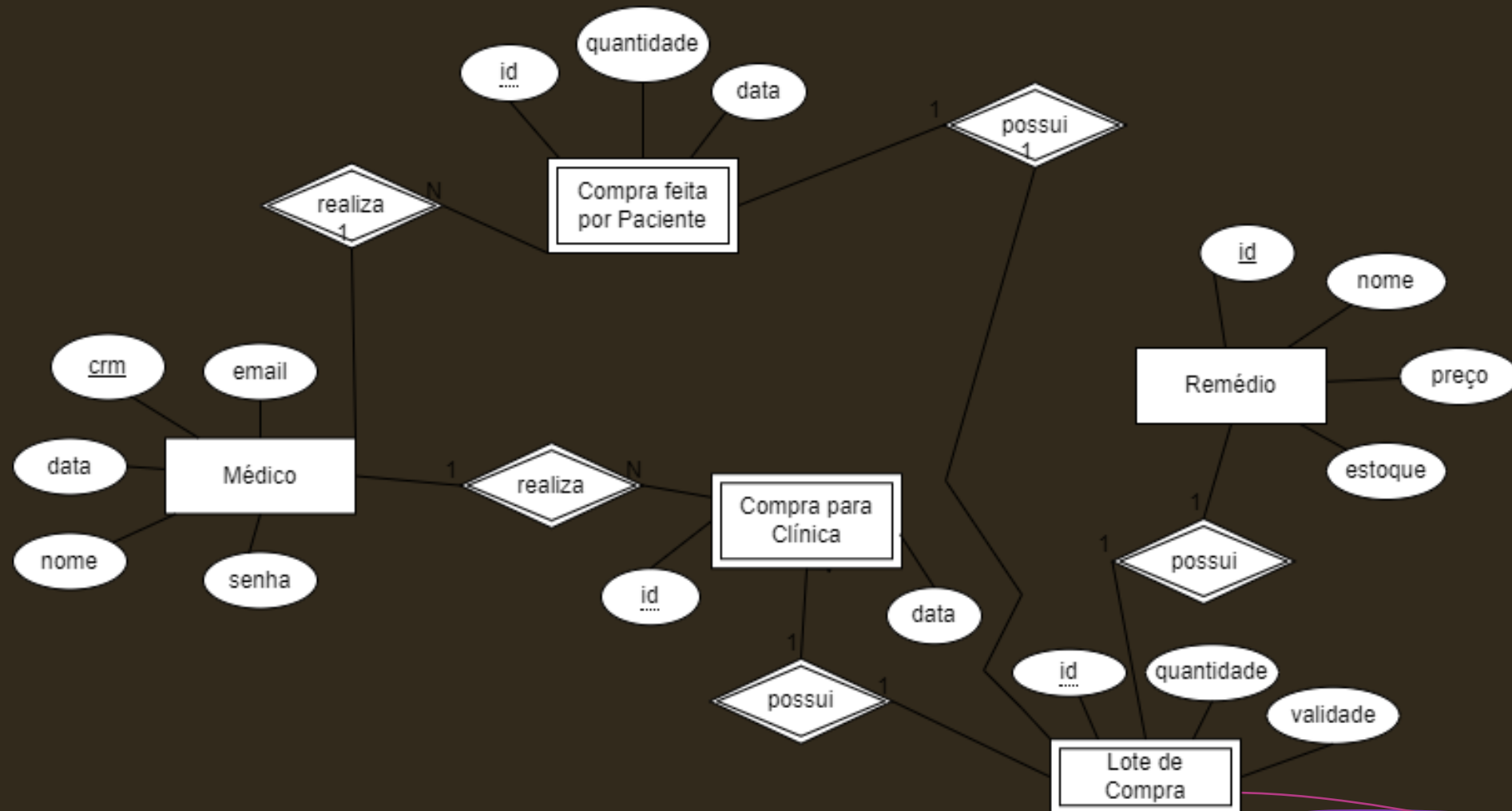
Medstock

Integrantes:

- Felipe Rivetti Mizher
- Pedro Hosken Fernandes Guimarães
- Rafael Rehfeld Martins de Oliveira
- Vitor Augusto Viana Azevedo



Diagrama



Conexão back-end ao Banco de dados

```
public class Aplicacao {  
    Run | Debug  
    public static void main(String[] args) {  
        // Configuração de porta  
        port(4567);  
  
        // Configuração do banco de dados  
        String jdbcUrl = "jdbc:postgresql://medstocker.postgres.database.azure.com:5432/postgres?sslmode=require";  
        String username = "vitor";  
        String password = "Medstocker@";  
  
        try {  
            Connection connection = DriverManager.getConnection(jdbcUrl, username, password);  
  
            // Criando instâncias dos DAOs  
            MedicoDAO medicoDAO = new MedicoDAO(connection);  
            RemedioDAO remedioDAO = new RemedioDAO(connection);  
            LoteDAO loteDAO = new LoteDAO(connection);  
            CompraMedicoDAO compraMedicoDAO = new CompraMedicoDAO(connection);  
            CompraPacienteDAO compraPacienteDAO = new CompraPacienteDAO(connection);  
  
            // Criando instâncias dos Services  
            MedicoService medicoService = new MedicoService(medicoDAO);  
            RemedioService remedioService = new RemedioService(remedioDAO);  
            LoteService loteService = new LoteService(loteDAO, remedioDAO);  
            CompraMedicoService compraMedicoService = new CompraMedicoService(compraMedicoDAO, medicoDAO, loteDAO);  
            CompraPacienteService compraPacienteService = new CompraPacienteService(compraPacienteDAO, remedioDAO);  
        }  
    }  
}
```

Conexão back-end ao Banco de dados

```
INSERT INTO Remedio (nome, preco)
VALUES ('Rivotril', 22.28);
SELECT * FROM "remedio";
```

4 rows returned

	id integer	nome character varying	preco numeric	estoque integer
1	4	Naproxeno	100.50	0
2	3	Tylenol	12.50	10
3	6	Foxis	10.50	0
4	1	Paracetamol	10.50	80

localhost:4567/remedios

Estilos de formatação ☐

```
[{"id":4,"nome":"Naproxeno","preco":100.5,"estoque":0}, {"id":3,"nome":"Tylenol","preco":12.5,"estoque":10}, {"id":6,"nome":"Foxis","preco":10.5,"estoque":0}, {"id":1,"nome":"Paracetamol","preco":10.5,"estoque":80}]
```

Conexão back-end ao Banco de dados

```
INSERT INTO Remedio (nome, preco) Untitled-1 • J II ↺ ↻ ⬇ ⬆ ↺ ⬆ untitle:untitled-1 X
```

```
1 INSERT INTO Remedio (nome, preco)
2 VALUES ('Rivotril', 22.28);
3 SELECT * FROM "remedio";
```

1 row inserted

5 rows returned

	id integer	nome character varying	preco numeric	estoque integer
1	4	Naproxeno	100.50	0
2	3	Tylenol	12.50	10
3	6	Foxis	10.50	0
4	1	Paracetamol	10.50	80
5	9	Rivotril	22.28	0

← → ↺ ⓘ localhost4567/remedios ☆ 🔔 🌱 📄 V ⋮

Estilos de formatação ☐

[{"id":4,"nome":"Naproxeno","preco":100.5,"estoque":0},{ "id":3,"nome":"Tylenol","preco":12.5,"estoque":10},{ "id":6,"nome":"Foxis","preco":10.5,"estoque":0},{ "id":1,"nome":"Paracetamol","preco":10.5,"estoque":80},{ "id":9,"nome":"Rivotril","preco":22.28,"estoque":0}]

Pastas do projeto

```

└─ Codigo
  └─ Medstock
    └─ src
      └─ main
        └─ java
          ├── app
          ├── dao
          ├── model
          ├── service
          └─ resources
            ├── Cadastro
            ├── Home
            ├── images
            ├── Login
            ├── Notificacoes
            └─ Remedios
              └─ application.properties
        └─ test
      └─ target
        ├── classes
        ├── test-classes
        └─ pom.xml
```

```

└─ Codigo
  └─ Medstock
    └─ src
      └─ main
        └─ java
          ├── app
          ├── dao
          ├── model
          └─ service
            ├── CompraMedicoDAO.java
            ├── CompraPacienteDAO.java
            ├── DAO.java
            ├── LoteDAO.java
            ├── MedicoDAO.java
            ├── RemedioDAO.java
            ├── CompraMedico.java
            ├── CompraPaciente.java
            ├── Lote.java
            ├── Medico.java
            ├── Remedio.java
            ├── CompraMedicoService.java
            ├── CompraPacienteService.java
            ├── LoteService.java
            ├── MedicoService.java
            └── RemedioService.java
```

```

└─ Codigo
  └─ Medstock
    └─ src
      └─ main
        ├── java
        ├── resources
        └─ images
          ├── cadastro.css
          ├── cadastro.html
          ├── densis.css
          ├── densis.html
          ├── home.css
          ├── home.html
          ├── suprah.css
          ├── suprah.html
          ├── medStockerLogo.png
          ├── remedio1.png
          ├── remedio2.png
          └── remedio3.png
        ├── Login
        ├── Notificacoes
        └─ Remedios
          ├── login.css
          ├── login.html
          ├── notificacoes.css
          ├── notificacoes.html
          ├── remedio.css
          ├── remedio.html
          ├── remedios.css
          ├── remedios.html
          └─ application.properties
```

Princípios de desenvolvimento java (get/set)

```
1 package model;
2
3 public class Medico {
4     private int crm;
5     private String nome;
6     private String sobrenome;
7     private String email;
8     private String dataNascimento;
9
10    // Construtor sem parâmetros
11    public Medico() {
12    }
13
14    // Construtor completo
15    public Medico(int crm, String nome, String sobrenome, String email, String dataNascimento) {
16        this.crm = crm;
17        this.nome = nome;
18        this.sobrenome = sobrenome;
19        this.email = email;
20        this.dataNascimento = dataNascimento;
21    }
22
23    // Getters e Setters
24    public int getCrm() {
25        return crm;
26    }
27
28    public void setCrm(int crm) {
29        this.crm = crm;
30    }
31
32    public String getNome() {
33        return nome;
34    }
```

```
36    public void setNome(String nome) {
37        this.nome = nome;
38    }
39
40    public String getSobrenome() {
41        return sobrenome;
42    }
43
44    public void setSobrenome(String sobrenome) {
45        this.sobrenome = sobrenome;
46    }
47
48    public String getEmail() {
49        return email;
50    }
51
52    public void setEmail(String email) {
53        this.email = email;
54    }
55
56    public String getDataNascimento() {
57        return dataNascimento;
58    }
59
60    public void setDataNascimento(String dataNascimento) {
61        this.dataNascimento = dataNascimento;
62    }
63 }
64
```

Código da Aplicação.java

```
1 package app;
2
3 import static spark.Spark.*;
4 import com.google.gson.Gson;
5 import service.MedicoService;
6 import service.RemedioService;
7 import service.LoteService;
8 import service.CompraMedicoService;
9 import service.CompraPacienteService;
10 import model.Medico;
11 import model.Remedio;
12 import model.Lote;
13 import model.CompraMedico;
14 import model.CompraPaciente;
15 import dao.MedicoDAO;
16 import dao.RemedioDAO;
17 import dao.LoteDAO;
18 import dao.CompraMedicoDAO;
19 import dao.CompraPacienteDAO;
20
21 import java.sql.Connection;
22 import java.sql.DriverManager;
23 import java.sql.SQLException;
```

```
25 public class Aplicacao {
26     public static void main(String[] args) {
27         // Configuração de porta
28         port(4567);
29
30         // Configuração do banco de dados
31         String jdbcUrl = "jdbc:postgresql://medstocker.postgres.database.azure.com:5432/postgres?sslmode=require";
32         String username = "vitor";
33         String password = "Medstocker@";
34
35         try {
36             Connection connection = DriverManager.getConnection(jdbcUrl, username, password);
37
38             // Criando instâncias dos DAOs
39             MedicoDAO medicoDAO = new MedicoDAO(connection);
40             RemedioDAO remedioDAO = new RemedioDAO(connection);
41             LoteDAO loteDAO = new LoteDAO(connection);
42             CompraMedicoDAO compraMedicoDAO = new CompraMedicoDAO(connection);
43             CompraPacienteDAO compraPacienteDAO = new CompraPacienteDAO(connection);
44
45             // Criando instâncias dos Services
46             MedicoService medicoService = new MedicoService(medicoDAO);
47             RemedioService remedioService = new RemedioService(remedioDAO);
48             LoteService loteService = new LoteService(loteDAO, remedioDAO);
49
50             CompraMedicoService compraMedicoService = new CompraMedicoService(compraMedicoDAO, medicoDAO, loteDAO);
51             CompraPacienteService compraPacienteService = new CompraPacienteService(compraPacienteDAO, remedioDAO);
52
53             // Usando Gson para conversão de JSON
54             Gson gson = new Gson();
55
56             // ----- Rotas para Médico -----
57
58             // Rota para listar todos os médicos (GET)
59             get("/medicos", (req, res) -> {
60                 res.type("application/json");
61                 return gson.toJson(medicoService.listarTodosMedicos());
62             });
63
64             // Rota para buscar um médico pelo CRM (GET)
65             get("/medicos/:crm", (req, res) -> {
66                 res.type("application/json");
67                 int crm = Integer.parseInt(req.params("crm"));
68                 Medico medico = medicoService.buscarMedicoPorCRM(crm);
69                 if (medico != null) {
70                     return gson.toJson(medico);
71                 } else {
72                     res.status(404);
73                 }
74             });
75         } catch (SQLException e) {
76             e.printStackTrace();
77         }
78     }
79 }
```



```

72         return "Médico não encontrado";
73     }
74 });
75
76 // Rota para cadastrar um médico (POST)
77 post("/medicos", (req, res) -> {
78     Medico medico = gson.fromJson(req.body(), Medico.class);
79     medicoService.cadastrarMedico(medico);
80     res.status(201); // Created
81     return "Médico cadastrado com sucesso!";
82 });
83
84 // Rota para atualizar um médico (PUT)
85 put("/medicos/:crm", (req, res) -> {
86     Medico medico = gson.fromJson(req.body(), Medico.class);
87     medicoService.atualizarMedico(medico);
88     return "Médico atualizado com sucesso!";
89 });
90
91 // Rota para deletar um médico (DELETE)
92 delete("/medicos/:crm", (req, res) -> {
93     int crm = Integer.parseInt(req.params("crm"));
94     medicoService.deletarMedico(crm);

```

```

118
119 // Rota para cadastrar um remédio (POST)
120 post("/remedios", (req, res) -> {
121     Remedio remedio = gson.fromJson(req.body(), Remedio.class);
122     remedioService.cadastrarRemedio(remedio);
123     res.status(201); // Created
124     return "Remédio cadastrado com sucesso!";
125 });
126
127 // Rota para atualizar um remédio (PUT)
128 put("/remedios/:id", (req, res) -> {
129     Remedio remedio = gson.fromJson(req.body(), Remedio.class);
130     remedioService.atualizarRemedio(remedio);
131     return "Remédio atualizado com sucesso!";
132 });
133
134 // Rota para deletar um remédio (DELETE)
135 delete("/remedios/:id", (req, res) -> {
136     int id = Integer.parseInt(req.params("id"));
137     remedioService.deletarRemedio(id);
138     return "Remédio deletado com sucesso!";
139 });
140

```

```

95         return "Médico deletado com sucesso!";
96     });
97
98     // ----- Rotas para Remédios -----
99
100 // Rota para listar todos os remédios (GET)
101 get("/remedios", (req, res) -> {
102     res.type("application/json");
103     return gson.toJson(remedioService.listarTodosRemedios());
104 });
105
106 // Rota para buscar um remédio por ID (GET)
107 get("/remedios/:id", (req, res) -> {
108     res.type("application/json");
109     int id = Integer.parseInt(req.params("id"));
110     Remedio remedio = remedioService.buscarRemedioPorId(id);
111     if (remedio != null) {
112         return gson.toJson(remedio);
113     } else {
114         res.status(404);
115         return "Remédio não encontrado";
116     }
117 });

```

```

141 // ----- Rotas para Lotes -----
142
143 // Rota para listar todos os lotes (GET)
144 get("/lotes", (req, res) -> {
145     res.type("application/json");
146     return gson.toJson(loteService.listarTodosLotes());
147 });
148
149 // Rota para cadastrar um novo lote (POST)
150 post("/lotes", (req, res) -> {
151     Lote lote = gson.fromJson(req.body(), Lote.class);
152     loteService.cadastrarLote(lote);
153     res.status(201); // Created
154     return "Lote cadastrado com sucesso!";
155 });
156
157 // ----- Rotas para CompraMedico -----
158
159 // Rota para cadastrar uma nova compra de médico (POST)
160 post("/compramedico", (req, res) -> {
161     CompraMedico compraMedico = gson.fromJson(req.body(), CompraMedico.class);
162     compraMedicoService.cadastrarCompraMedico(compraMedico);
163     res.status(201); // Created

```

```

164         return "Compra do médico registrada com sucesso!";
165     });
166
167 // ----- Rotas para CompraPaciente -----
168
169 // Rota para cadastrar uma nova compra de paciente (POST)
170 post("/comrapaciente", (req, res) -> {
171     CompraPaciente compraPaciente = gson.fromJson(req.body(), CompraPaciente.class);
172     compraPacienteService.cadastrarCompraPaciente(compraPaciente);
173     res.status(201); // Created
174     return "Compra do paciente registrada com sucesso!";
175 });
176
177 } catch (SQLException e) {
178     e.printStackTrace();
179 }
180 }
181 }

```

Pacote DAO (insert/update/remove/get/

dao

- CompraMedicoDAO.java
- CompraPacienteDAO.java
- DAO.java
- LoteDAO.java
- MedicoDAO.java
- RemedioDAO.java

```
11 public class MedicoDAO {
12     private Connection connection;
13
14     public MedicoDAO(Connection connection) {
15         this.connection = connection;
16     }
17
18     // Método para inserir um médico no banco
19     public void inserir(Medico medico) throws SQLException {
20         String sql = "INSERT INTO Medico (crm, nome, sobrenome, email, data_nascimento) VALUES (?, ?, ?, ?, ?)";
21         PreparedStatement statement = connection.prepareStatement(sql);
22         statement.setInt(1, medico.getCrm());
23         statement.setString(2, medico.getNome());
24         statement.setString(3, medico.getSobrenome());
25         statement.setString(4, medico.getEmail());
26         statement.setString(5, medico.getDataNascimento());
27         statement.executeUpdate();
28     }
29
30     // Método para buscar um médico por CRM
31     public Medico buscarPorCRM(int crm) throws SQLException {
32         String sql = "SELECT * FROM Medico WHERE crm = ?";
33         PreparedStatement statement = connection.prepareStatement(sql);
34         statement.setInt(1, crm);
35         ResultSet resultSet = statement.executeQuery();
36
37         if (resultSet.next()) {
38             return new Medico(
39                 resultSet.getInt("crm"),
40                 resultSet.getString("nome"),
41                 resultSet.getString("sobrenome"),
42                 resultSet.getString("email"),
43                 resultSet.getString("data_nascimento"));
44         }
45         return null; // Retorna null se o médico não for encontrado
46     }
47 }
```

```
48 // Método para atualizar um médico
49 public void atualizar(Medico medico) throws SQLException {
50     String sql = "UPDATE Medico SET nome = ?, sobrenome = ?, email = ?, data_nascimento = ? WHERE crm = ?";
51     PreparedStatement statement = connection.prepareStatement(sql);
52     statement.setString(1, medico.getNome());
53     statement.setString(2, medico.getSobrenome());
54     statement.setString(3, medico.getEmail());
55     statement.setString(4, medico.getDataNascimento());
56     statement.setInt(5, medico.getCrm());
57     statement.executeUpdate();
58 }
59
60 // Método para deletar um médico
61 public void deletar(int crm) throws SQLException {
62     String sql = "DELETE FROM Medico WHERE crm = ?";
63     PreparedStatement statement = connection.prepareStatement(sql);
64     statement.setInt(1, crm);
65     statement.executeUpdate();
66 }
67
68 // Método para listar todos os médicos
69 public List<Medico> listarTodos() throws SQLException {
70     String sql = "SELECT * FROM Medico";
71     PreparedStatement statement = connection.prepareStatement(sql);
72     ResultSet resultSet = statement.executeQuery();
73
74     List<Medico> medicos = new ArrayList<>();
75     while (resultSet.next()) {
76         Medico medico = new Medico(
77             resultSet.getInt("crm"),
78             resultSet.getString("nome"),
79             resultSet.getString("sobrenome"),
80             resultSet.getString("email"),
81             resultSet.getString("data_nascimento"));
82         medicos.add(medico);
83     }
84     return medicos;
85 }
86
87 }
```

Pacote DAO (método para abrir e fechar a conexão com o banco de dados)

```
1 package dao;
2 import java.sql.*;
3
4 public class DAO {
5     private String url = System.getenv("medstocker.postgres.database.azure.com");
6     private String usuario = System.getenv("vitor");
7     private String senha = System.getenv("MedStocker!");
8     protected Connection conexao;
9
10    public DAO() {
11        try {
12            conexao = DriverManager.getConnection(url, usuario, senha);
13        } catch (SQLException e) {
14            throw new RuntimeException("Erro ao conectar ao banco de dados: ", e);
15        }
16    }
17
18    public ResultSet executeQuery(String sql) {
19        try {
20            Statement stmt = conexao.createStatement();
21            return stmt.executeQuery(sql);
22        } catch (SQLException e) {
23            throw new RuntimeException("Erro ao executar consulta: ", e);
24        }
25    }
```

```
26
27    public int executeUpdate(String sql) {
28        try {
29            Statement stmt = conexao.createStatement();
30            return stmt.executeUpdate(sql);
31        } catch (SQLException e) {
32            throw new RuntimeException("Erro ao executar atualização: ", e);
33        }
34    }
35
36    public PreparedStatement prepareStatement(String sql) throws SQLException {
37        return conexao.prepareStatement(sql);
38    }
39
40    public void close() {
41        try {
42            if (conexao != null && !conexao.isClosed()) {
43                conexao.close();
44            }
45        } catch (SQLException e) {
46            throw new RuntimeException("Erro ao fechar conexão com o banco de dados: ", e);
47        }
48    }
49 }
```

Pacote Model

▼ model

- J CompraMedico.java
- J CompraPaciente.java
- J Lote.java
- J Medico.java
- J Remedio.java

```
1 package model;
2
3 public class Medico {
4     private int crm;
5     private String nome;
6     private String sobrenome;
7     private String email;
8     private String dataNascimento;
9
10    // Construtor sem parâmetros
11    public Medico() {
12    }
13
14    // Construtor completo
15    public Medico(int crm, String nome, String sobrenome, String email, String dataNascimento) {
16        this.crm = crm;
17        this.nome = nome;
18        this.sobrenome = sobrenome;
19        this.email = email;
20        this.dataNascimento = dataNascimento;
21    }
22
23    // Getters e Setters
24    public int getCrm() {
25        return crm;
26    }
27
28    public void setCrm(int crm) {
29        this.crm = crm;
30    }
31
32    public String getNome() {
33        return nome;
34    }
35
```

```
39
40    public String getSobrenome() {
41        return sobrenome;
42    }
43
44    public void setSobrenome(String sobrenome) {
45        this.sobrenome = sobrenome;
46    }
47
48    public String getEmail() {
49        return email;
50    }
51
52    public void setEmail(String email) {
53        this.email = email;
54    }
55
56    public String getDataNascimento() {
57        return dataNascimento;
58    }
59
60    public void setDataNascimento(String dataNascimento) {
61        this.dataNascimento = dataNascimento;
62    }

```

Pacote SERVICE

▼ service

- J CompraMedicoService.java
- J CompraPacienteService.java
- J LoteService.java
- J MedicoService.java
- J RemedioService.java

```
9  ▼ public class MedicoService {  
10      private MedicoDAO medicoDAO;  
11  
12      public MedicoService(MedicoDAO medicoDAO) {  
13          this.medicoDAO = medicoDAO;  
14      }  
15  
16      // Método para cadastrar um novo médico  
17      public void cadastrarMedico(Medico medico) throws SQLException {  
18          medicoDAO.inserir(medico);  
19      }  
20  
21      // Método para buscar médico pelo CRM  
22      public Medico buscarMedicoPorCRM(int crm) throws SQLException {  
23          return medicoDAO.buscarPorCRM(crm);  
24      }  
25
```

```
26      // Método para atualizar informações do médico  
27      public void atualizarMedico(Medico medico) throws SQLException {  
28          medicoDAO.atualizar(medico);  
29      }  
30  
31      // Método para deletar um médico  
32      public void deletarMedico(int crm) throws SQLException {  
33          medicoDAO.deletar(crm);  
34      }  
35  
36      // Método para listar todos os médicos  
37      public List<Medico> listarTodosMedicos() throws SQLException {  
38          return medicoDAO.listarTodos();  
39      }  
40  }
```

Inteligência do Sistema

- A inteligência do sistema reside na capacidade de capturar a imagem do rótulo do remédio, processá-la para extrair informações, e verificar automaticamente se o remédio está dentro da data de validade.

Recursos e técnicas do sistema Inteligente



O sistema captura uma foto do rótulo do medicamento.



Processa a imagem e extrai informações textuais (como data de validade).



Verifica automaticamente se o medicamento está dentro da validade.

Proposições de valores

- **Eficiência:** Automatização do processo de verificação da validade, eliminando a necessidade de entrada manual de dados.
- **Precisão:** Redução de erros humanos na identificação de datas de validade.



Aprendizado da inteligência

O sistema usará um sistema de aprendizagem offline

Foco na natureza estável dos rótulos e datas, sem necessidade de constante adaptação.

Menor necessidade de recursos computacionais em tempo real.

Entradas e saídas

Entrada: Imagem do rótulo do remédio.

Saída: Texto extraído (incluindo a data de validade) e status de validade (dentro ou fora da validade).

Fornecedor de tecnologia

- Fornecedor: **Google Cloud** ou **Microsoft Azure**.
- Ambos possuem robustas soluções de IA como serviços.



Serviços desse fornecedor

- **Google Cloud Vision API** ou **Azure Computer Vision API**.
- Reconhecimento de texto (OCR) para processamento das imagens capturadas.

