

# FITFLEX

Comer bem nunca foi tão facil para  
diabeticos!

Alice Salim Khouri Antunes  
Felipe Henrique Oliveira Diniz  
Matheus de Almeida Moreira  
Theodoro Augusto de Brito Spadinger Motter



# Front conectado com o Back

```
post("/usuario/login", (req, res) -> {
    res.type("application/json");
    Usuario loginReq = gson.fromJson(req.body(), Usuario.class);
    Usuario usuario = usuarioService.autenticarUsuario(loginReq.getEmail(), loginReq.getSenha());
    if (usuario != null) {
        System.out.println(gson.toJson(usuario));
        return gson.toJson(usuario);
    } else {
        res.status(401);
        return "{\"erro\":\"Email ou senha incorretos.\"}";
    }
});

get("/usuario/:id", (req, res) -> {
    res.type("application/json");
    int id = Integer.parseInt(req.params(":id"));
    Usuario usuario = usuarioService.buscarUsuarioId(id);
    if (usuario != null) {
        return gson.toJson(usuario);
    } else {
        res.status(404);
        return "{\"erro\":\"Usuário não encontrado.\"}";
    }
});
```

```
// Rotas Ingrediente (mantidas)
post("/ingrediente", (req, res) -> {
    res.type("application/json");
    Ingrediente ingrediente = gson.fromJson(req.body(), Ingrediente.class);
    boolean sucesso = ingredienteService.cadastrarIngrediente(ingrediente);
    if (sucesso) {
        res.status(201);
        return gson.toJson(ingrediente);
    } else {
        res.status(500);
        return "{\"erro\":\"Erro ao adicionar ingrediente.\"}";
    }
});

get("/ingrediente", (req, res) -> {
    res.type("application/json");
    List<Ingrediente> ingredientes = ingredienteService.listarTodosIngredientes();
    return gson.toJson(ingredientes);
});

put("/ingrediente/:id", (req, res) -> {
    res.type("application/json");
    int id = Integer.parseInt(req.params(":id"));
    Ingrediente ingredienteAtualizado = gson.fromJson(req.body(), Ingrediente.class);
    ingredienteAtualizado.setId(id);
    boolean sucesso = ingredienteService.atualizarIngrediente(ingredienteAtualizado);
    if (sucesso) {
        return gson.toJson(ingredienteAtualizado);
    } else {
        res.status(500);
        return "{\"erro\":\"Erro ao atualizar ingrediente.\"}";
    }
});

delete("/ingrediente/:id", (req, res) -> {
    res.type("application/json");
    int id = Integer.parseInt(req.params(":id"));
    boolean sucesso = ingredienteService.excluirIngrediente(id);
    if (sucesso) {
        return "{\"mensagem\":\"Ingrediente removido com sucesso.\"}";
    } else {
        res.status(404);
        return "{\"erro\":\"Ingrediente não encontrado.\"}";
    }
});

get("/ingrediente/:id", (req, res) -> {
    res.type("application/json");
    int id = Integer.parseInt(req.params(":id"));
    Ingrediente ingrediente = ingredienteService.buscarIngredientePorId(id);
    if (ingrediente != null) {
        return gson.toJson(ingrediente);
    } else {
        res.status(404);
        return "{\"erro\":\"Ingrediente não encontrado.\"}";
    }
});
```

# Back se conecta ao Banco de Dados

```
public class DAO {
    private Connection conexao;

    public DAO() {
        conexao = null;
    }

    public boolean conectar() {
        String driverName = "org.postgresql.Driver";
        String serverName = "localhost";
        String mydatabase = "carro";
        int porta = 5432;
        String url = "jdbc:postgresql://" + serverName + ":" + porta + "/" + mydatabase;
        String username = "ti2cc";
        String password = "ti@cc";
        boolean status = false;

        try {
            Class.forName(driverName);
            conexao = DriverManager.getConnection(url, username, password);
            status = (conexao == null);
            System.out.println("Conexão efetuada com o postgres!");
        } catch (ClassNotFoundException e) {
            System.err.println("Conexão NÃO efetuada com o postgres -- Driver não encontrado -- " + e.getMessage());
        } catch (SQLException e) {
            System.err.println("Conexão NÃO efetuada com o postgres -- " + e.getMessage());
        }

        return status;
    }

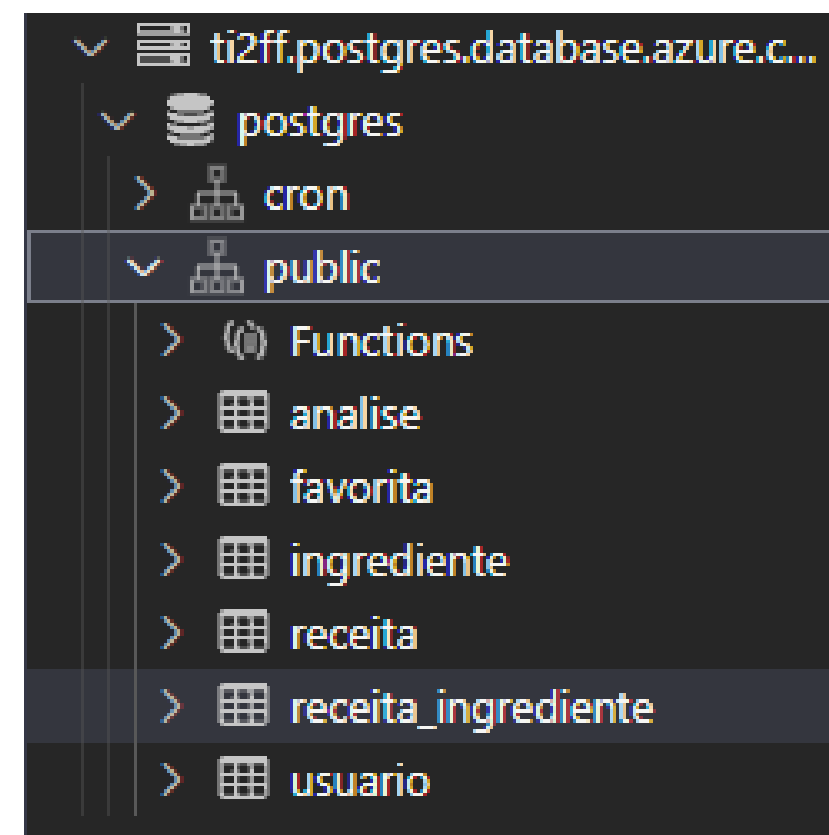
    public boolean close() {
        boolean status = false;

        try {
            conexao.close();
            status = true;
        } catch (SQLException e) {
            System.err.println(e.getMessage());
        }

        return status;
    }

    public boolean inserirCarro(Carro carro) {
        boolean status = false;
        try {
            Statement st = conexao.createStatement();
            st.executeUpdate("INSERT INTO carro (codigo, marca, modelo, ano) "
                + "VALUES (" + carro.getCodigo() + ", '" + carro.getMarca() + "', '"

```



# Organização

```
▼ maven.demo [plmg-cc-ti2-2025-1-g06-fitflex develop]
  ▼ src/main/java
    ▼ app
      ▶ Aplicacao.java
    ▼ dao
      ▶ DAO.java
      ▶ IngredienteDAO.java
      ▶ PossuiDAO.java
      ▶ ReceitaDAO.java
      ▶ UsuarioDAO.java
    ▼ model
      ▶ Ingrediente.java
      ▶ Possui.java
      ▶ Receita.java
      ▶ Usuario.java
    ▼ service
      ▶ IngredienteService.java
      ▶ PossuiService.java
      ▶ ReceitaService.java
      ▶ UserService.java
```

```
▼ src/main/resources
  ▼ frontend
    ▼ public
      ▶ assets
      ▶ index.html
      ▶ manifest.json
      ▶ robots.txt
    ▶ src
      ▶ package.json
      ▶ package-lock.json
      ▶ README.md
      ▶ yarn.lock
  ▼ Scripts
    ▶ fitflex.sql
```

# Classes Java

```
// Construtor completo com todos os atributos
public Ingrediente(int id, String nome, double gordura, double carbo, double cal, double proteínas, double quantidade, double indiceGlicemico, String unidade) {
    this.id = id;
    this.nome = nome;
    this.quantidade = quantidade;
    this.proteínas = proteínas;
    this.carbo = carbo;
    this.gordura = gordura;
    this.cal = cal;
    this.indice_glicemico = indiceGlicemico;
    this.unidade = unidade;
}

public Ingrediente( String nome, double gordura, double carbo, double cal, double proteínas, double quantidade, double indiceGlicemico, String unidade) {
    this.nome = nome;
    this.quantidade = quantidade;
    this.proteínas = proteínas;
    this.carbo = carbo;
    this.gordura = gordura;
    this.cal = cal;
    this.indice_glicemico = indiceGlicemico;
    this.unidade = unidade;
}

// Métodos getters e setters

public int getId() {
    return id;
}

public void setId(int id) {
    this.id = id;
}

public String getNome() {
    return nome;
}

public void setNome(String nome) {
    this.nome = nome;
}
```

# Aplicação Java

```
// Rotas Ingrediente (mantidas)
post("/ingrediente", (req, res) -> {
    res.type("application/json");
    Ingrediente ingrediente = gson.fromJson(req.body(), Ingrediente.class);
    boolean sucesso = ingredienteService.cadastrarIngrediente(ingrediente);
    if (sucesso) {
        res.status(201);
        return gson.toJson(ingrediente);
    } else {
        res.status(500);
        return "{\"erro\":\"Erro ao adicionar ingrediente.\"}";
    }
});

get("/ingrediente", (req, res) -> {
    res.type("application/json");
    List<Ingrediente> ingredientes = ingredienteService.listarTodosIngredientes();
    return gson.toJson(ingredientes);
});

put("/ingrediente/:id", (req, res) -> {
    res.type("application/json");
    int id = Integer.parseInt(req.params(":id"));
    Ingrediente ingredienteAtualizado = gson.fromJson(req.body(), Ingrediente.class);
    ingredienteAtualizado.setId(id);
    boolean sucesso = ingredienteService.atualizarIngrediente(ingredienteAtualizado);
    if (sucesso) {
        return gson.toJson(ingredienteAtualizado);
    } else {
        res.status(500);
        return "{\"erro\":\"Erro ao atualizar ingrediente.\"}";
    }
});

delete("/ingrediente/:id", (req, res) -> {
    res.type("application/json");
    int id = Integer.parseInt(req.params(":id"));
    boolean sucesso = ingredienteService.excluirIngrediente(id);
    if (sucesso) {
        return "{\"mensagem\":\"Ingrediente removido com sucesso.\"}";
    } else {
        res.status(404);
        return "{\"erro\":\"Ingrediente não encontrado.\"}";
    }
});
});
```



# DAO

# MODEL

# SERVICES

- dao
  - DAO.java
  - IngredienteDAO.java
  - PossuiDAO.java
  - ReceitaDAO.java
  - UsuarioDAO.java

- model
  - Ingrediente.java
  - Possui.java
  - Receita.java
  - Usuario.java

- service
  - IngredienteService.java
  - PossuiService.java
  - ReceitaService.java
  - UserService.java

```
// Cadastrar novo ingrediente
public boolean cadastrarIngrediente(Ingrediente ingrediente) throws Exception {
    boolean cadastrado = false;

    String sql = "INSERT INTO ingrediente (nome, quantidade, proteinas, carboidratos, gordura, calorias, indice_glicemico, unidade_medida) " +
        "VALUES (?, ?, ?, ?, ?, ?, ?, ?)";

    try (PreparedStatement stmt = conexao.prepareStatement(sql, Statement.RETURN_GENERATED_KEYS)) {
        stmt.setString(1, ingrediente.getNome());
        stmt.setDouble(2, ingrediente.getQuantidade());
        stmt.setDouble(3, ingrediente.getProteinas());
        stmt.setDouble(4, ingrediente.getCarbo());
        stmt.setDouble(5, ingrediente.getGordura());
        stmt.setDouble(6, ingrediente.getCal());
        stmt.setDouble(7, ingrediente.getIndiceGlicemico());
        stmt.setString(8, ingrediente.getUnidade());

        int affectedRows = stmt.executeUpdate();

        if (affectedRows > 0) {
            try (ResultSet rs = stmt.getGeneratedKeys()) {
                if (rs.next()) {
                    ingrediente.setId(rs.getInt(1));
                    cadastrado = true;
                }
            }
        }
    } catch (SQLException e) {
        e.printStackTrace();
        throw new Exception("Erro ao cadastrar ingrediente.", e);
    }

    return cadastrado;
}
```

```
// Buscar todos os ingredientes
public List<Ingrediente> listarTodos() {
    List<Ingrediente> lista = new ArrayList<>();
    String sql = "SELECT * FROM ingrediente";

    try (PreparedStatement stmt = conexao.prepareStatement(sql);
        ResultSet rs = stmt.executeQuery()) {
        while (rs.next()) {
            Ingrediente ing = new Ingrediente();
            ing.setId(rs.getInt("id"));
            ing.setNome(rs.getString("nome"));
            ing.setQuantidade(rs.getDouble("quantidade"));
            ing.setProteinas(rs.getDouble("proteinas"));
            ing.setCarbo(rs.getDouble("carboidratos"));
            ing.setGordura(rs.getDouble("gordura"));
            ing.setCal(rs.getDouble("calorias"));
            ing.setIndiceGlicemico(rs.getDouble("indice_glicemico"));
            ing.setUnidade(rs.getString("unidade_medida"));

            lista.add(ing);
        }
    } catch (SQLException e) {
        e.printStackTrace();
    }

    return lista;
}
```

```
// Atualizar ingrediente
public boolean atualizarIngrediente(Ingrediente ingrediente) {
    String sql = "UPDATE ingrediente SET nome = ?, quantidade = ?, proteinas = ?, carboidratos = ?, gordura = ?, calorias = ?, indice_glicemico = ?, unidade_medida = ? " +
        "WHERE id = ?";

    try (PreparedStatement stmt = conexao.prepareStatement(sql)) {
        stmt.setString(1, ingrediente.getNome());
        stmt.setDouble(2, ingrediente.getQuantidade());
        stmt.setDouble(3, ingrediente.getProteinas());
        stmt.setDouble(4, ingrediente.getCarbo());
        stmt.setDouble(5, ingrediente.getGordura());
        stmt.setDouble(6, ingrediente.getCal());
        stmt.setDouble(7, ingrediente.getIndiceGlicemico());
        stmt.setString(8, ingrediente.getUnidade());
        stmt.setInt(9, ingrediente.getId());

        return stmt.executeUpdate() > 0;
    } catch (SQLException e) {
        e.printStackTrace();
        return false;
    }
}
```

```
// Excluir ingrediente
public boolean excluirIngrediente(int idIngrediente) {
    String sql = "DELETE FROM ingrediente WHERE id = ?";

    try (PreparedStatement stmt = conexao.prepareStatement(sql)) {
        stmt.setInt(1, idIngrediente);
        return stmt.executeUpdate() > 0;
    } catch (SQLException e) {
        e.printStackTrace();
        return false;
    }
}
```

# CRUD Front

Buscar Receitas





















Análise de Produtos

🔍 Ingredientes

Gerenciamento de Ingredientes

+ Novo Ingrediente

🔍 Buscar ingrediente...

Nome	Unidade	Calorias (kcal)	Proteínas (g)	Gorduras (g)	Carboidratos (g)	IG	Ações
Ovo	Unidade (un)	68	6	5	0.6	0	 
Canela em pó	Gramas (g)	2.47	0.04	0.01	0.81	0.05	 
Fermento Químico	Gramas (g)	0.53	0	0	0.27	0	 
Óleo de coco	Gramas (g)	8.62	0	1	0	0	 
Gengibre	Gramas (g)	0.8	0.02	0.01	0.18	0.1	 
Abóbora cabotia	Gramas (g)	0.4	0.01	0	0.1	0.51	 
Alho	Gramas (g)	1.49	0.06	0.01	0.33	0.1	 
Azeite de Oliva	Gramas (g)	8.84	0	1	0	0	 
Grão de Bico	Gramas (g)	1.64	0.09	0.03	0.27	0.28	 
Tomate	Gramas (g)	0.18	0.01	0	0.04	0.15	 

Novo Ingrediente

Informações Básicas

Nome do Ingrediente

Ex: Arroz Integral

Unidade de Medida

Gramas (g)

Quantidade por Porção

Ex: 100

Informações Nutricionais

Calorias (kcal)

Ex: 130

Proteínas (g)

Ex: 2.7

Gorduras (g)

Ex: 1.0

Carboidratos (g)

Ex: 28.0

Índice Glicêmico

Ex: 55

Cancelar

Salvar

Editar Ingrediente

Informações Básicas

Nome do Ingrediente

teste

Unidade de Medida

Gramas (g)

Quantidade por Porção

1

Informações Nutricionais

Calorias (kcal)

1

Proteínas (g)

1

Gorduras (g)

1

Carboidratos (g)

1

Índice Glicêmico

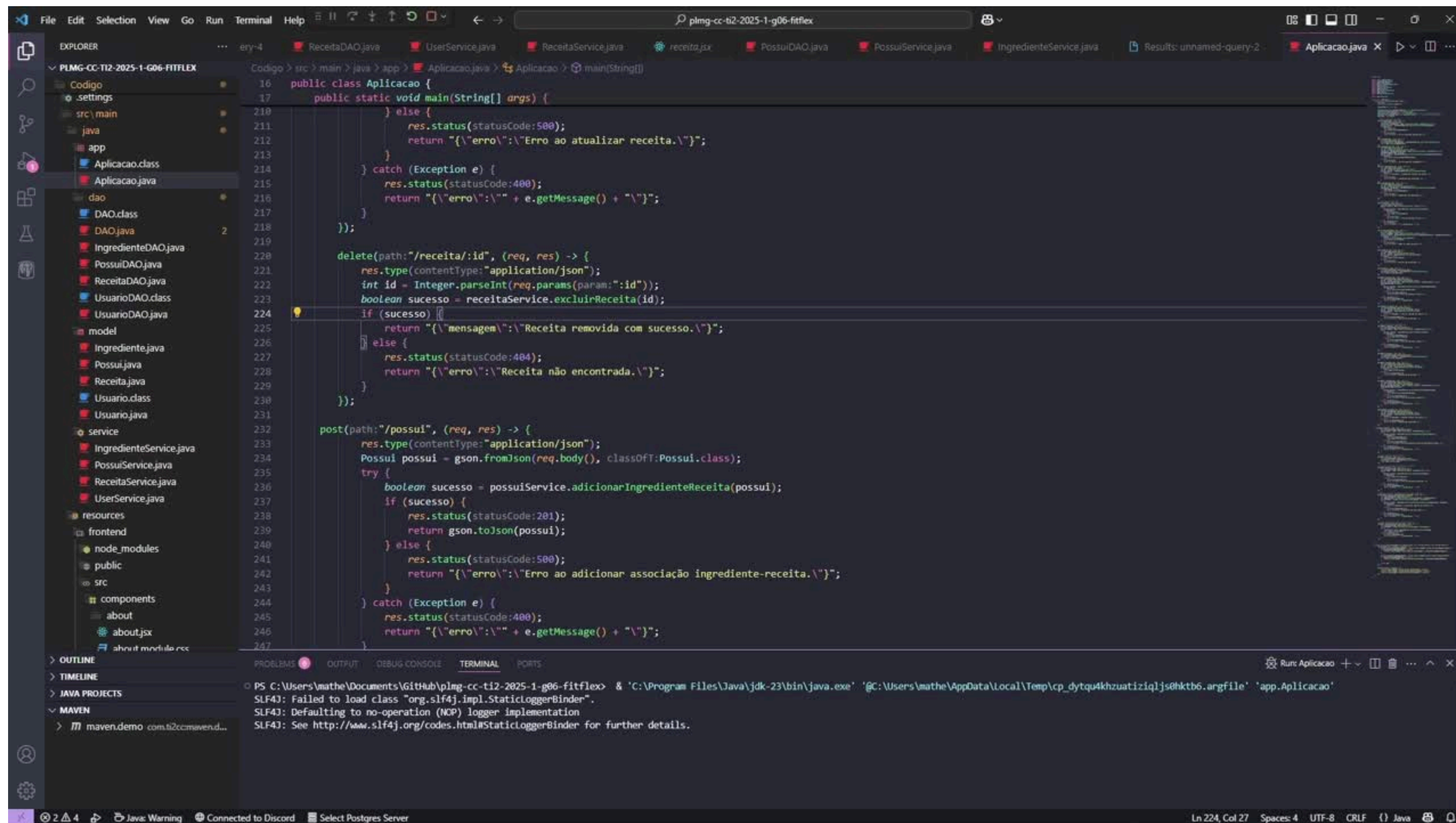
1

Cancelar

Salvar



# Link



```
File Edit Selection View Go Run Terminal Help
PLMG-CC-TI2-2025-1-G06-FITFLEX
Codigo > src > main > java > app > Aplicacao.java > Aplicacao > main(String[])
16 public class Aplicacao {
17     public static void main(String[] args) {
210     } else {
211         res.status(statusCode:500);
212         return "{\n\"erro\":\n\"Erro ao atualizar receita.\n\"}";
213     }
214 } catch (Exception e) {
215     res.status(statusCode:400);
216     return "{\n\"erro\":\n\"\" + e.getMessage() + "\n\"}";
217 }
218 });
219
220 delete(path:"/receita/:id", (req, res) -> {
221     res.type(contentType:"application/json");
222     int id = Integer.parseInt(req.params(param:"id"));
223     boolean sucesso = receitaService.excluirReceita(id);
224     if (sucesso) {
225         return "{\n\"mensagem\":\n\"Receita removida com sucesso.\n\"}";
226     } else {
227         res.status(statusCode:404);
228         return "{\n\"erro\":\n\"Receita não encontrada.\n\"}";
229     }
230 });
231
232 post(path:"/possui", (req, res) -> {
233     res.type(contentType:"application/json");
234     Possui possui = gson.fromJson(req.body(), classOfT:Possui.class);
235     try {
236         boolean sucesso = possuiService.adicionarIngredienteReceita(possui);
237         if (sucesso) {
238             res.status(statusCode:201);
239             return gson.toJson(possui);
240         } else {
241             res.status(statusCode:500);
242             return "{\n\"erro\":\n\"Erro ao adicionar associação ingrediente-receita.\n\"}";
243         }
244     } catch (Exception e) {
245         res.status(statusCode:400);
246         return "{\n\"erro\":\n\"\" + e.getMessage() + "\n\"}";
247     }
248 }

OUTLINE
TIMELINE
JAVA PROJECTS
MAVEN
maven demo com.ti2ccmavend...

PS C:\Users\mathe\Documents\Github\plmg-cc-ti2-2025-1-g06-fitflex> & 'C:\Program Files\Java\jdk-23\bin\java.exe' '@C:\Users\mathe\AppData\Local\Temp\cp_dytqu4khzuatiziq1js0hktb6.argfile' 'app.Aplicacao'
SLF4J: Failed to load class "org.slf4j.impl.StaticLoggerBinder".
SLF4J: Defaulting to no-operation (NOP) logger implementation
SLF4J: See http://www.slf4j.org/codes.html#StaticLoggerBinder for further details.
```

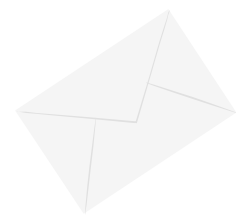
# IS CANVAS

FERRAMENTAL DE IA <ul style="list-style-type: none"><li>• Google Cloud Vision (OCR)</li><li>• Google NLP API</li><li>• Google AutoML</li><li>• OpenCV</li></ul>	ENTRADAS <ul style="list-style-type: none"><li>• Imagem da embalagem com lista de ingredientes</li></ul>	PROPOSIÇÃO DE VALOR <ul style="list-style-type: none"><li>• Verificação rápida e confiável de ingredientes</li><li>• Segurança alimentar para diabéticos</li><li>• Acesso fácil via app ou site</li></ul>	EQUIPE <ul style="list-style-type: none"><li>• Desenvolvedor(a) full-stack</li><li>• Nutricionista (consultoria)</li><li>• Designer UX/UI</li></ul>	CLIENTES <ul style="list-style-type: none"><li>• Diabéticos tipo 1 e tipo 2</li><li>• Familiares e cuidadores</li><li>• Clínicas e nutricionistas</li></ul>
	SAÍDAS <p>Lista de ingredientes com classificação:</p> <ul style="list-style-type: none"><li>• Seguro</li><li>• Potencialmente perigoso</li></ul>		STAKEHOLDERS CHAVES <ul style="list-style-type: none"><li>• Comunidade diabética</li><li>• Organizações de saúde</li><li>• Hospitais e clínicas</li></ul>	
CUSTOS <ul style="list-style-type: none"><li>• APIs do Google Cloud</li><li>• Servidores/infraestrutura</li><li>• Marketing e divulgação</li></ul>			RECEITAS <ul style="list-style-type: none"><li>• Assinaturas premium (planos)</li></ul>	



## Implementações Futuras

- Refatorar design do site
- Incluir fotos
- Favoritar Receitas
- IA para analise de imagens



# OBRIGADO!

Estamos abertos a dúvidas  
e sugestões

**FITFLEX**