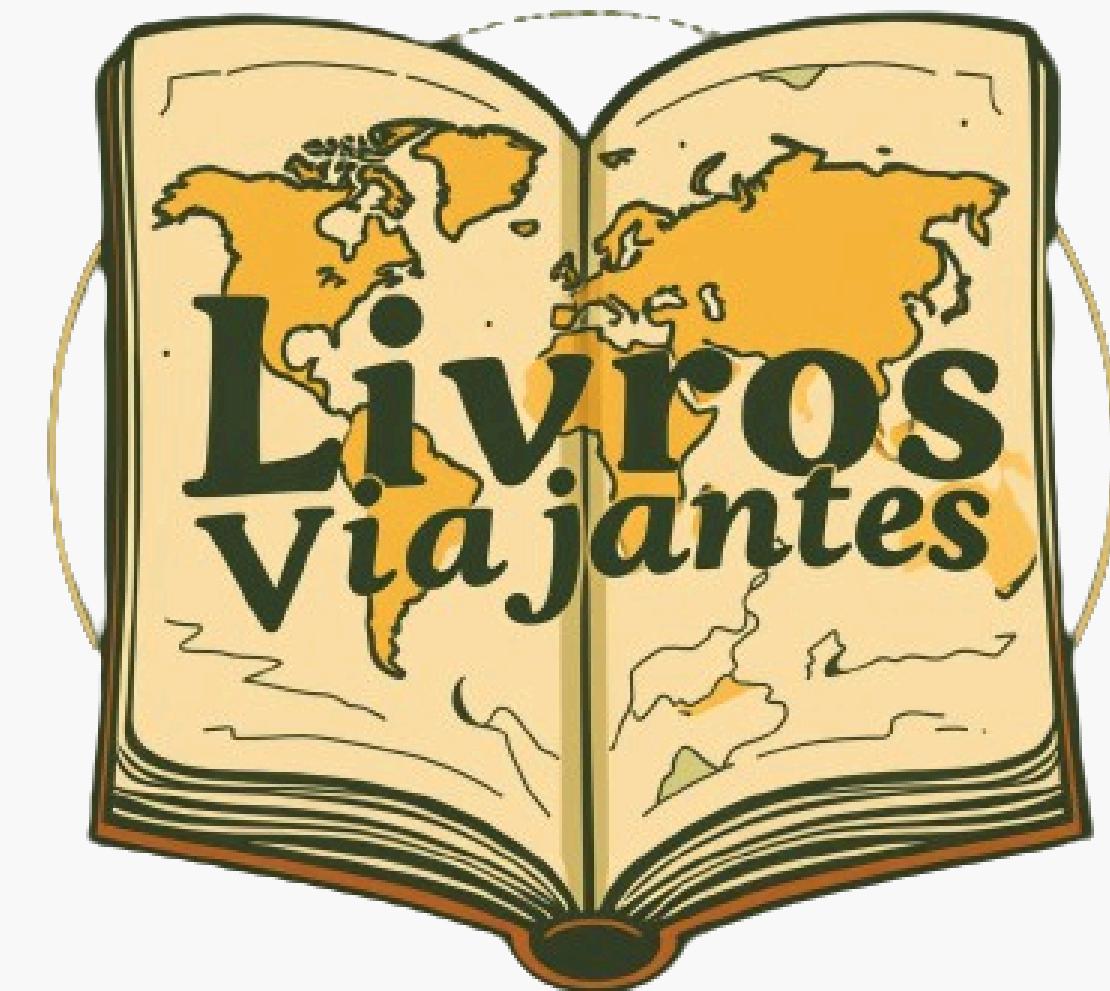




Trabalho Interdisciplinar II: Back-End - Grupo 5



LIVROS QUE VIAJAM, HISTÓRIAS QUES FICAM.



Objetivo do projeto



Uma plataforma de **troca de livros**, onde os usuários podem cadastrar livros que não utilizam mais e trocá-los com outros membros da comunidade. A proposta incentiva o compartilhamento de conhecimento e promove o acesso a novos livros sem a necessidade de compra.

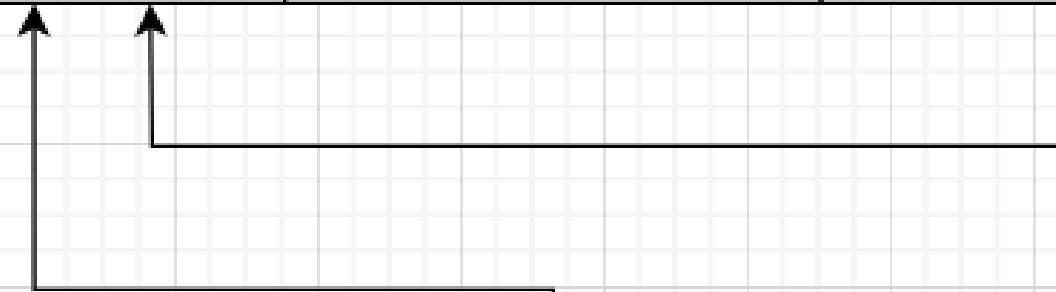


Modelo de Implementação do Banco de Dados



USUARIO

<u>id_usuario</u>	nome	email	senha	telefone	endereco
-------------------	------	-------	-------	----------	----------



OFERTA

<u>id_oferta</u>	usuario_ofertante	usuario_recebedor	livro_recebido	livro_ofertado	<u>id_usuario</u>
------------------	-------------------	-------------------	----------------	----------------	-------------------

LIVROS

<u>id_livro</u>	titulo	autor	genero	sinopse	<u>id_oferta</u>
-----------------	--------	-------	--------	---------	------------------

Diagrama Entidade-Relacionamento (DER)

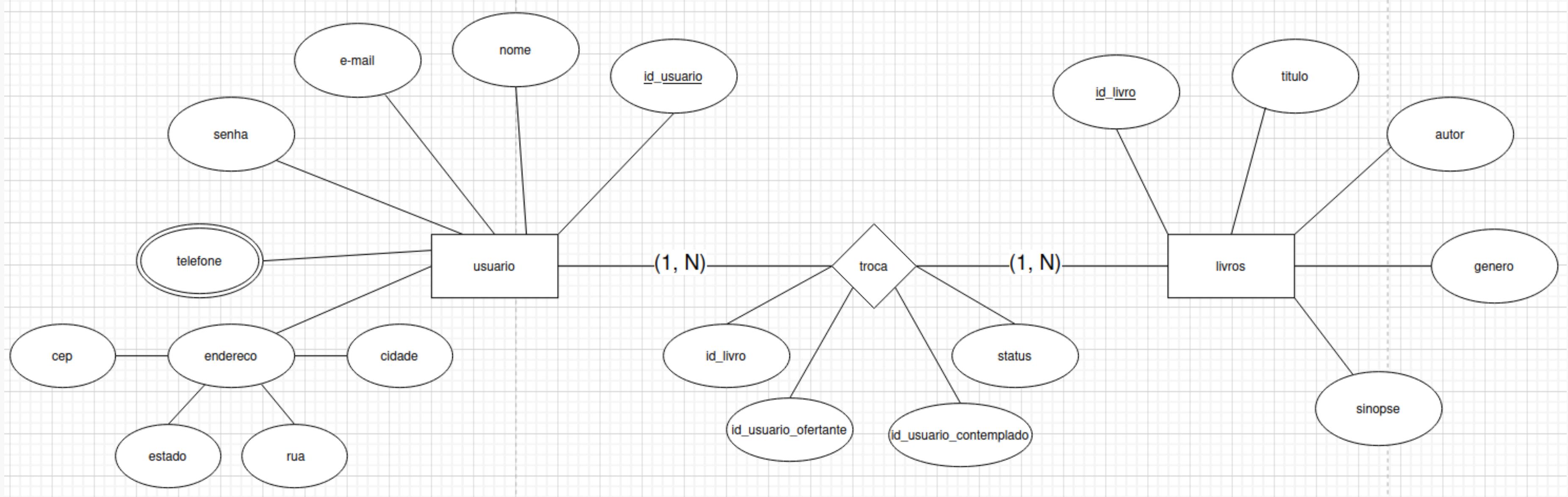
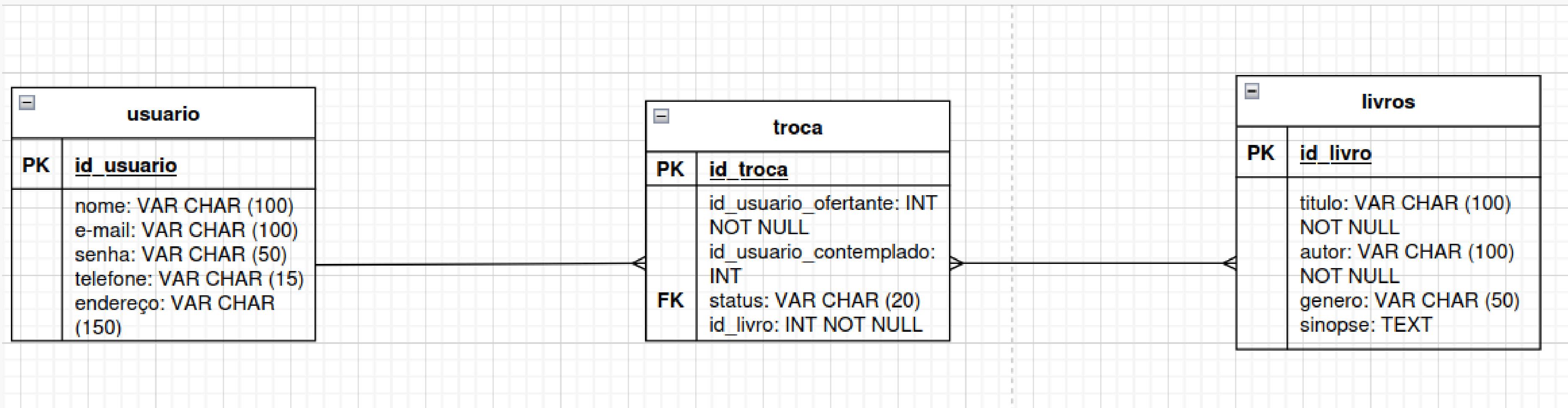


Diagrama ER (pé de galinha)



Scripts para Criação do Banco de Dados



Restrições

Tabelas Principais:

- Usuários:

chave primária: id_usuario

- Troca:

chave primária: id_troca

- Livros:

chave primária: id_livro

- Chaves Primárias (PRIMARY KEY):

Definem identificadores únicos para cada tabela.

- Chaves Estrangeiras (FOREIGN KEY):

Relacionam as tabelas, associando livros aos usuários e categorias, e trocas aos livros e usuários envolvidos.

- Restrições de Validação (CHECK):

Garantem que os valores de status_oferta sejam limitados a 'Pendente', 'Trocado' ou 'Cancelado'.



Scripts SQL



-- Tabela Usuario

```
CREATE TABLE Usuario (
    id_usuario SERIAL PRIMARY KEY,
    nome VARCHAR(100) NOT NULL,
    email VARCHAR(100) NOT NULL
    UNIQUE,
    senha VARCHAR(255) NOT NULL,
    telefone VARCHAR(15),
    rua VARCHAR(100),
    cidade VARCHAR(100),
    estado VARCHAR(2),
    cep VARCHAR(10)
);
```

- -Tabela Livro

```
CREATE TABLE Livro (
    id_livro SERIAL PRIMARY KEY,
    titulo VARCHAR(255) NOT NULL,
    autor VARCHAR(100) NOT NULL,
    genero VARCHAR(50),
    sinopse TEXT
);
```

- -Tabela Troca

```
CREATE TABLE Troca (
    id_troca SERIAL PRIMARY KEY,
    id_livro INT NOT NULL,
    usuario_ofertante INT NOT NULL,
    usuario_contemplado INT,
    status VARCHAR(20) NOT NULL, -- Ex: "pendente", "concluída",
    "cancelada"
    FOREIGN KEY (id_livro) REFERENCES Livro (id_livro),
    FOREIGN KEY (usuario_ofertante) REFERENCES Usuario (id_usuario),
    FOREIGN KEY (usuario_contemplado) REFERENCES Usuario
    (id_usuario)
```



```
scripts.sql
-- Tabela Livro
CREATE TABLE Livro (
    id_livro SERIAL PRIMARY KEY,
    titulo VARCHAR(255) NOT NULL,
    autor VARCHAR(100) NOT NULL,
    genero VARCHAR(50),
    sinopse TEXT
);
```

Livro	
id_livro	serial primary key
titulo	varchar(255)
autor	varchar(255)
genero	varchar(50)
sinopse	text



Back-end - Código de Aplicação



The image shows a screenshot of a code editor with a dark theme. The tab bar at the top has three tabs: 'App.java' (selected), 'Livro.java', and 'LivroDAO.java'. The code in 'App.java' is as follows:

```
// ----- Rotas de Livros -----

post("/cadastrar-livro", (req, res) -> {
    String body = req.body();
    Livro livro = new Gson().fromJson(body, Livro.class);
    // Salvar livro no banco de dados
    livroService.cadastrarLivro(req, res);
    res.status(201); // Código de sucesso
    return "Livro cadastrado com sucesso!";
});

// Obter livro por ID
get("/livro/:id", (request, response) -> livroService.getLivroById(request, response));

// Atualizar livro
put("/livro/:id", (request, response) -> livroService.updateLivro(request, response));

// Deletar livro
delete("/livro/:id", (request, response) -> livroService.deletarLivro(request, response));
```

At the bottom left of the code editor window, there is a decorative footer consisting of five small orange circles.

Front-end – Envio de Requisição POST



The screenshot shows a web form titled "Cadastrar Novo Livro" set against a background of floating books. The form includes fields for "Título" (livro1), "Autor(a)" (sarah j.mas), "Gênero" (Fantasia), and "Sinopse" (apenas um exemplo de sinopse aqui.). A large purple button at the bottom right is labeled "Cadastrar Livro". A yellow curved arrow points from the "Cadastrar Livro" button towards the "Livro.js" code block on the right.

Cadastrar Novo Livro

Título:
livro1

Autor(a):
sarah j.mas

Gênero:
Fantasia

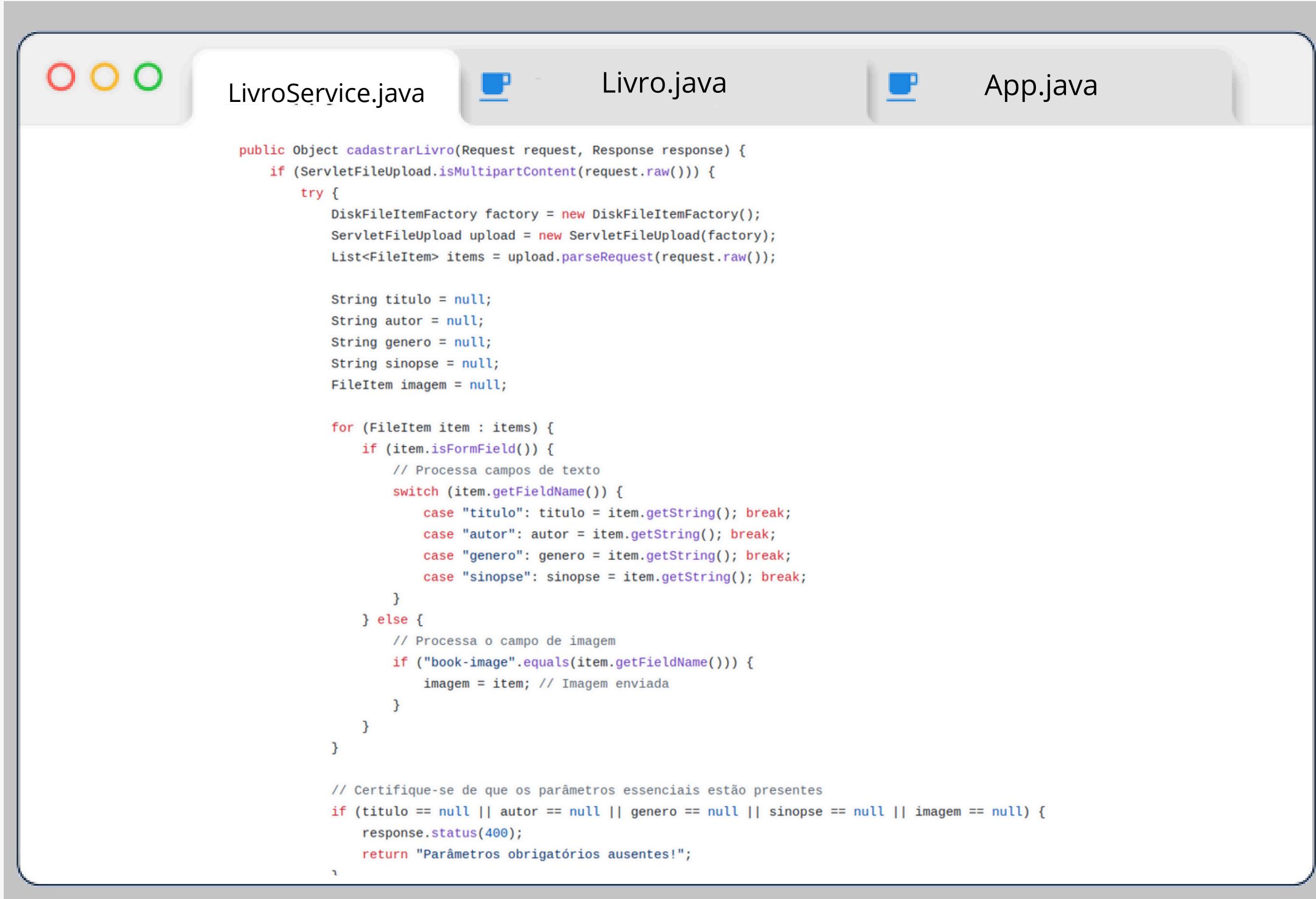
Sinopse:
apenas um exemplo de sinopse aqui.

Cadastrar Livro

The screenshot shows a code editor window with a tab labeled "Livro.js". The code is a fetch request for a POST method to "http://localhost:4567/livro". It includes headers for Content-Type: "application/json" and a body created by JSON.stringify(livro). The .then() block handles the response, throwing an error if it's not ok, and returning the JSON data. An alert is shown for successful data insertion, followed by a reset of the form. The .catch() block logs errors to the console and alerts the user. A yellow curved arrow points from the "Cadastrar Livro" button in the screenshot on the left towards this code block.

```
fetch("http://localhost:4567/livro", {
  method: "POST",
  headers: {
    "Content-Type": "application/json"
  },
  body: JSON.stringify(livro)
})
.then(response => {
  if (!response.ok) {
    throw new Error("Erro ao cadastrar o livro: " + response.statusText);
  }
  return response.json();
})
.then(data => {
  alert("Livro cadastrado com sucesso! ID: " + data);
  // Limpar o formulário após o cadastro
  document.getElementById("book-form").reset();
})
.catch(error => {
  console.error(error);
  alert("Erro: " + error.message);
});
```

Integração Banco de dados e Back-end



The image shows a screenshot of a Java code editor with three tabs visible: `LivroService.java`, `Livro.java`, and `App.java`. The `LivroService.java` tab is active, displaying the following code:

```
public Object cadastrarLivro(Request request, Response response) {
    if (ServletFileUpload.isMultipartContent(request.raw())) {
        try {
            DiskFileItemFactory factory = new DiskFileItemFactory();
            ServletFileUpload upload = new ServletFileUpload(factory);
            List<FileItem> items = upload.parseRequest(request.raw());

            String titulo = null;
            String autor = null;
            String genero = null;
            String sinopse = null;
            FileItem imagem = null;

            for (FileItem item : items) {
                if (item.isFormField()) {
                    // Processa campos de texto
                    switch (item.getFieldName()) {
                        case "titulo": titulo = item.getString(); break;
                        case "autor": autor = item.getString(); break;
                        case "genero": genero = item.getString(); break;
                        case "sinopse": sinopse = item.getString(); break;
                    }
                } else {
                    // Processa o campo de imagem
                    if ("book-image".equals(item.getFieldName())) {
                        imagem = item; // Imagem enviada
                    }
                }
            }

            // Certifique-se de que os parâmetros essenciais estão presentes
            if (titulo == null || autor == null || genero == null || sinopse == null || imagem == null) {
                response.status(400);
                return "Parâmetros obrigatórios ausentes!";
            }
        } catch (Exception e) {
            response.status(500);
            return "Ocorreu um erro ao processar o formulário.";
        }
    }
}
```

The code implements a `cadastrarLivro` method that handles file uploads. It uses the `ServletFileUpload` class to parse the multipart request and extract file items. The method then processes form fields (title, author, genre, synopsis) and checks if a book image was uploaded. If any required parameter is missing, it returns a 400 Bad Request response with an error message. Otherwise, it returns a 500 Internal Server Error response with a generic error message.

Back-end – Processamento da Requisição



The screenshot shows a Java code editor with three tabs at the top: App.java, Livro.java, and LivroDAO.java. The LivroDAO.java tab is active, displaying the following code:

```
public boolean inserirLivro(Livro livro) {
    String sql = "INSERT INTO livro (titulo, autor, genero, sinopse, imagem) VALUES (?, ?, ?, ?, ?)"

    try (Connection conn = this.conectarLivro();
        PreparedStatement pstmt = conn.prepareStatement(sql, Statement.RETURN_GENERATED_KEYS)) {

        pstmt.setString(1, livro.getTitulo());
        pstmt.setString(2, livro.getAutor());
        pstmt.setString(3, livro.getGenero());
        pstmt.setString(4, livro.getSinopse());
        pstmt.setBytes(5, livro.getImagen());

        int rowsAffected = pstmt.executeUpdate();

        if (rowsAffected > 0) {
            ResultSet rs = pstmt.getGeneratedKeys();
            if (rs.next()) {
                livro.setIdLivro(rs.getInt(1));
            }
            return true;
        }

    } catch (SQLException e) {
        System.err.println("Erro ao inserir livro: " + e.getMessage());
    }

    return false;
}
```

Decorative footer element consisting of five yellow dots is located at the bottom left.

Visualizar livro cadastrado



A screenshot of a web-based library management system. The interface includes a header with 'Login', 'Cadastrar', 'Meus Livros', 'Em Alta', 'Buscar livro...', 'Buscar', and 'Registrar Novo Livro'. Below the header, there's a section titled 'Livros de Programação' displaying five book cards:

- Programming in C, 3e** by Andrei Nechaev and Anton Kachanov. Description: C is one of the most popular programming languages. It runs on most software... Add to list.
- Core Python Programming** by Asa G. Kachanov and Brian W. Kernighan. Description: Experts and novices alike will be able to find information about every command they'll... Add to list.
- Conceitos de Linguagens de Programação** by Sem descrição. Add to list.
- A Linguagem de Programação Go** by A. A. Díaz. Description: A linguagem de programação Go é a fonte mais confiável para qualquer programador... Add to list.
- Programação em Baixo Nível** by Brian W. Kernighan. Description: Conheça a linguagem Assembly e a arquitetura Intel 64, torne-se proficiente... Add to list.

An orange curved arrow points from the 'Listar Todos Livros' button in the screenshot to the 'listarTodosLivros' method in the code editor.

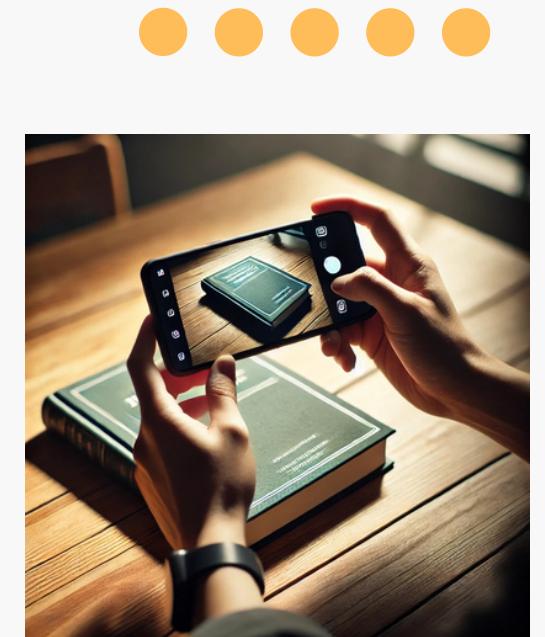
livroService.java

```
124 // Método para listar todos os livros (com imagem)
125 public Object listarTodosLivros(Response response) {
126     List<Livro> livros = livroDAO.buscarTodosLivros();
127     JSONArray jsonResponse = new JSONArray();
128     for (Livro livro : livros) {
129         JSONObject jsonLivro = new JSONObject();
130         jsonLivro.put("titulo", livro.getTitulo());
131         jsonLivro.put("autor", livro.getAutor());
132         jsonLivro.put("genero", livro.getGenero());
133         jsonLivro.put("sinopse", livro.getSinopse());
134
135         // Converte a imagem para base64 para retornar como JSON
136         if (livro.getImagen() != null) {
137             String imagemBase64 = java.util.Base64.getEncoder().encodeToString(livro.getImagen());
138             jsonLivro.put("imagem", imagemBase64);
139         }
140
141         jsonResponse.put(jsonLivro);
142     }
143
144     response.type("application/json");
145     return jsonResponse.toString();
146 }
147 }
```



Especificação - Sistemas Inteligentes

- **API GOOGLE VISION:** A partir de uma foto tirada do usuário, a API vai realizar o reconhecimento das capas de livros e extrair o autor e o título.
- **ENTRADA:** imagem do livro.
- **SAÍDAS:** Título, autor, sinopse, gênero, editora e ano de publicação.
- **INTEGRAÇÃO COM OUTROS SISTEMAS:** Google Vision API, Google Books API e Banco de Dados Interno.





Obrigado!

Filipe Lorenzato Cardoso Rodrigues
Gabryelle Franco Xavier
Iuri Saad Furtunato Fialho
Rodrigo Oliveira Andrade de Vasconcelos

