Beleza! Vou criar um projeto Java completo com Spark Java + Gson para API e Swing para frontend, incluindo as funcionalidades que conversamos (listar, cadastrar, editar, excluir suprimentos) — pronto para você subir no GitHub.

**Projeto: SuprimentosHospitalares**

**Estrutura simplificada:**

SuprimentosHospitalares/

├── src/

│ ├── main/

│ │ ├── java/

│ │ │ ├── app/

│ │ │ │ ├── Main.java # Spark Java API

│ │ │ │ ├── model/

│ │ │ │ │ └── Suprimento.java

│ │ │ │ └── ui/

│ │ │ │ ├── CadastroSuprimentoFrame.java

│ │ │ │ └── TelaSuprimentos.java

│ └── resources/

│ └── (se quiser configs)

├── pom.xml

└── README.md

**Como montar e subir no GitHub**

Eu vou gerar o conteúdo principal do projeto aqui, você só precisa copiar os arquivos para a estrutura e depois criar o repositório no GitHub.

**1. pom.xml**

<project xmlns="http://maven.apache.org/POM/4.0.0"

xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"

xsi:schemaLocation="http://maven.apache.org/POM/4.0.0

https://maven.apache.org/xsd/maven-4.0.0.xsd">

<modelVersion>4.0.0</modelVersion>

<groupId>com.seu.nome</groupId>

<artifactId>suprimentos-hospitalares</artifactId>

<version>1.0-SNAPSHOT</version>

<properties>

<maven.compiler.source>17</maven.compiler.source>

<maven.compiler.target>17</maven.compiler.target>

</properties>

<dependencies>

<!-- Spark Java -->

<dependency>

<groupId>com.sparkjava</groupId>

<artifactId>spark-core</artifactId>

<version>2.9.4</version>

</dependency>

<!-- Gson -->

<dependency>

<groupId>com.google.code.gson</groupId>

<artifactId>gson</artifactId>

<version>2.10.1</version>

</dependency>

<!-- SQLite JDBC (opcional, caso queira banco real) -->

<dependency>

<groupId>org.xerial</groupId>

<artifactId>sqlite-jdbc</artifactId>

<version>3.42.0.0</version>

</dependency>

</dependencies>

<build>

<plugins>

<!-- Plugin para gerar jar com dependências -->

<plugin>

<groupId>org.apache.maven.plugins</groupId>

<artifactId>maven-assembly-plugin</artifactId>

<version>3.5.0</version>

<configuration>

<descriptorRefs>

<descriptorRef>jar-with-dependencies</descriptorRef>

</descriptorRefs>

<archive>

<manifest>

<mainClass>app.Main</mainClass>

</manifest>

</archive>

</configuration>

<executions>

<execution>

<id>make-assembly</id>

<phase>package</phase>

<goals><goal>single</goal></goals>

</execution>

</executions>

</plugin>

</plugins>

</build>

</project>

**2. app/model/Suprimento.java**

package app.model;

public class Suprimento {

public int id;

public String nome;

public String miligramagem;

public String tipo;

public int quantidade;

public double valor\_unitario;

}

**3. app/Main.java (API Spark Java simples com dados em memória)**

package app;

import static spark.Spark.\*;

import com.google.gson.\*;

import app.model.Suprimento;

import java.util.\*;

import java.util.concurrent.atomic.AtomicInteger;

public class Main {

private static List<Suprimento> banco = new ArrayList<>();

private static AtomicInteger idGen = new AtomicInteger(1);

private static Gson gson = new Gson();

public static void main(String[] args) {

port(4567);

before((req, res) -> res.type("application/json"));

// Listar todos

get("/api/suprimentos", (req, res) -> gson.toJson(banco));

// Cadastrar novo

post("/api/suprimentos", (req, res) -> {

Suprimento s = gson.fromJson(req.body(), Suprimento.class);

s.id = idGen.getAndIncrement();

banco.add(s);

res.status(201);

return gson.toJson(s);

});

// Atualizar

put("/api/suprimentos/:id", (req, res) -> {

int id = Integer.parseInt(req.params(":id"));

Suprimento atualizado = gson.fromJson(req.body(), Suprimento.class);

for (int i = 0; i < banco.size(); i++) {

if (banco.get(i).id == id) {

atualizado.id = id;

banco.set(i, atualizado);

return gson.toJson(atualizado);

}

}

res.status(404);

return "{\"error\":\"Suprimento não encontrado\"}";

});

// Deletar

delete("/api/suprimentos/:id", (req, res) -> {

int id = Integer.parseInt(req.params(":id"));

boolean removed = banco.removeIf(s -> s.id == id);

if (removed) {

return "{\"message\":\"Suprimento removido\"}";

} else {

res.status(404);

return "{\"error\":\"Suprimento não encontrado\"}";

}

});

}

}

**4. app/ui/CadastroSuprimentoFrame.java (form para cadastro)**

package app.ui;

import javax.swing.\*;

import java.awt.\*;

import java.io.OutputStream;

import java.net.HttpURLConnection;

import java.net.URL;

import com.google.gson.Gson;

import app.model.Suprimento;

public class CadastroSuprimentoFrame extends JFrame {

private JTextField txtNome, txtMiligramagem, txtTipo, txtQuantidade, txtValor;

private JButton btnSalvar;

private JTextArea txtStatus;

public CadastroSuprimentoFrame() {

setTitle("Cadastro de Suprimento");

setSize(400, 350);

setDefaultCloseOperation(EXIT\_ON\_CLOSE);

setLayout(new GridLayout(7, 2));

add(new JLabel("Nome:"));

txtNome = new JTextField();

add(txtNome);

add(new JLabel("Miligramagem:"));

txtMiligramagem = new JTextField();

add(txtMiligramagem);

add(new JLabel("Tipo:"));

txtTipo = new JTextField();

add(txtTipo);

add(new JLabel("Quantidade:"));

txtQuantidade = new JTextField();

add(txtQuantidade);

add(new JLabel("Valor Unitário:"));

txtValor = new JTextField();

add(txtValor);

btnSalvar = new JButton("Salvar");

txtStatus = new JTextArea();

txtStatus.setEditable(false);

JScrollPane scroll = new JScrollPane(txtStatus);

add(btnSalvar);

add(new JLabel());

add(new JLabel("Status:"));

add(scroll);

btnSalvar.addActionListener(e -> enviarCadastro());

}

private void enviarCadastro() {

try {

Suprimento s = new Suprimento();

s.nome = txtNome.getText();

s.miligramagem = txtMiligramagem.getText();

s.tipo = txtTipo.getText();

s.quantidade = Integer.parseInt(txtQuantidade.getText());

s.valor\_unitario = Double.parseDouble(txtValor.getText());

Gson gson = new Gson();

String json = gson.toJson(s);

URL url = new URL("http://localhost:4567/api/suprimentos");

HttpURLConnection conn = (HttpURLConnection) url.openConnection();

conn.setRequestMethod("POST");

conn.setDoOutput(true);

conn.setRequestProperty("Content-Type", "application/json");

try (OutputStream os = conn.getOutputStream()) {

os.write(json.getBytes());

os.flush();

}

int status = conn.getResponseCode();

if (status == 201) {

txtStatus.setText("✅ Suprimento cadastrado com sucesso!");

limparCampos();

} else {

txtStatus.setText("❌ Erro: código " + status);

}

} catch (Exception e) {

txtStatus.setText("❌ Erro: " + e.getMessage());

}

}

private void limparCampos() {

txtNome.setText("");

txtMiligramagem.setText("");

txtTipo.setText("");

txtQuantidade.setText("");

txtValor.setText("");

}

public static void main(String[] args) {

SwingUtilities.invokeLater(() -> new CadastroSuprimentoFrame().setVisible(true));

}

}

**5. app/ui/TelaSuprimentos.java (listagem, edição e exclusão)**

package app.ui;

import javax.swing.\*;

import javax.swing.table.DefaultTableModel;

import java.awt.\*;

import java.io.\*;

import java.net.\*;

import java.util.List;

import com.google.gson.reflect.TypeToken;

import com.google.gson.\*;

import app.model.Suprimento;

public class TelaSuprimentos extends JFrame {

private JTable tabela;

private DefaultTableModel modelo;

private JTextField txtNome, txtMiligramagem, txtTipo, txtQuantidade, txtValor;

private JButton btnAtualizar, btnRemover;

private Gson gson = new Gson();

private List<Suprimento> lista;

public TelaSuprimentos() {

setTitle("Gerenciar Suprimentos");

setSize(800, 400);

setDefaultCloseOperation(EXIT\_ON\_CLOSE);

setLayout(new BorderLayout());

modelo = new DefaultTableModel(new String[]{"ID", "Nome", "Tipo", "Qtd", "Valor"}, 0);

tabela = new JTable(modelo);

add(new JScrollPane(tabela), BorderLayout.CENTER);

btnAtualizar = new JButton("Salvar Alterações");

btnRemover = new JButton("Remover Suprimento");

JPanel botoes = new JPanel();

botoes.add(btnAtualizar);

botoes.add(btnRemover);

JPanel form = new JPanel(new GridLayout(6, 2));

txtNome = new JTextField();

txtMiligramagem = new JTextField();

txtTipo = new JTextField();

txtQuantidade = new JTextField();

txtValor = new JTextField();

form.add(new JLabel("Nome:"));

form.add(txtNome);

form.add(new JLabel("Miligramagem:"));

form.add(txtMiligramagem);

form.add(new JLabel("Tipo:"));

form.add(txtTipo);

form.add(new JLabel("Quantidade:"));

form.add(txtQuantidade);

form.add(new JLabel("Valor Unitário:"));

form.add(txtValor);

form.add(botoes);

add(form, BorderLayout.SOUTH);

carregarSuprimentos();

tabela.getSelectionModel().addListSelectionListener(e -> {

int i = tabela.getSelectedRow();

if (i >= 0) {

Suprimento s = lista.get(i);

txtNome.setText(s.nome);

txtMiligramagem.setText(s.miligramagem);

txtTipo.setText(s.tipo);

txtQuantidade.setText(String.valueOf(s.quantidade));

txtValor.setText(String.valueOf(s.valor\_unitario));

}

});

btnAtualizar.addActionListener(e -> atualizarSuprimento());

btnRemover.addActionListener(e -> removerSuprimento());

}

private void carregarSuprimentos() {

try {

URL url = new URL("http://localhost:4567/api/suprimentos");

HttpURLConnection conn = (HttpURLConnection) url.openConnection();

conn.setRequestMethod("GET");

lista = gson.fromJson(new InputStreamReader(conn.getInputStream()),

new TypeToken<List<Suprimento>>() {

}.getType());

modelo.setRowCount(0);

for (Suprimento s : lista) {

modelo.addRow(new Object[]{s.id, s.nome, s.tipo, s.quantidade, s.valor\_unitario});

}

} catch (Exception e) {

JOptionPane.showMessageDialog(this, "Erro ao carregar: " + e.getMessage());

}

}

private void atualizarSuprimento() {

int row = tabela.getSelectedRow();

if (row < 0) {

JOptionPane.showMessageDialog(this, "Selecione um suprimento para atualizar.");

return;

}

Suprimento s = lista.get(row);

s.nome = txtNome.getText();

s.miligramagem = txtMiligramagem.getText();

s.tipo = txtTipo.getText();

try {

s.quantidade = Integer.parseInt(txtQuantidade.getText());

s.valor\_unitario = Double.parseDouble(txtValor.getText());

} catch (NumberFormatException e) {

JOptionPane.showMessageDialog(this, "Quantidade e Valor Unitário devem ser numéricos.");

return;

}

try {

URL url = new URL("http://localhost:4567/api/suprimentos/" + s.id);

HttpURLConnection conn = (HttpURLConnection) url.openConnection();

conn.setRequestMethod("PUT");

conn.setRequestProperty("Content-Type", "application/json");

conn.setDoOutput(true);

String

json = gson.toJson(s);

try (OutputStream os = conn.getOutputStream()) {

os.write(json.getBytes());

}

int status = conn.getResponseCode();

if (status == 200) {

JOptionPane.showMessageDialog(this, "Atualizado com sucesso!");

carregarSuprimentos();

} else {

JOptionPane.showMessageDialog(this, "Erro ao atualizar: código " + status);

}

} catch (Exception e) {

JOptionPane.showMessageDialog(this, "Erro: " + e.getMessage());

}

}

private void removerSuprimento() {

int row = tabela.getSelectedRow();

if (row < 0) {

JOptionPane.showMessageDialog(this, "Selecione um suprimento para remover.");

return;

}

Suprimento s = lista.get(row);

int confirm = JOptionPane.showConfirmDialog(this,

"Deseja realmente remover o suprimento '" + s.nome + "'?",

"Confirmação", JOptionPane.YES\_NO\_OPTION);

if (confirm != JOptionPane.YES\_OPTION) return;

try {

URL url = new URL("http://localhost:4567/api/suprimentos/" + s.id);

HttpURLConnection conn = (HttpURLConnection) url.openConnection();

conn.setRequestMethod("DELETE");

int status = conn.getResponseCode();

if (status == 200) {

JOptionPane.showMessageDialog(this, "Removido com sucesso!");

carregarSuprimentos();

limparCampos();

} else {

JOptionPane.showMessageDialog(this, "Erro ao remover: código " + status);

}

} catch (Exception ex) {

JOptionPane.showMessageDialog(this, "Erro: " + ex.getMessage());

}

}

private void limparCampos() {

txtNome.setText("");

txtMiligramagem.setText("");

txtTipo.setText("");

txtQuantidade.setText("");

txtValor.setText("");

}

public static void main(String[] args) {

SwingUtilities.invokeLater(() -> new TelaSuprimentos().setVisible(true));

}

}

---

## 6. `README.md`

```markdown

# Suprimentos Hospitalares

Projeto simples de API REST e interface Swing para gerenciar suprimentos hospitalares.

## Como rodar

1. Execute a API:

```bash

mvn clean package

java -jar target/suprimentos-hospitalares-1.0-SNAPSHOT-jar-with-dependencies.jar

1. Rode a interface:

java -cp target/suprimentos-hospitalares-1.0-SNAPSHOT-jar-with-dependencies.jar app.ui.TelaSuprimentos

1. Ou abra a tela de cadastro:

java -cp target/suprimentos-hospitalares-1.0-SNAPSHOT-jar-with-dependencies.jar app.ui.CadastroSuprimentoFrame

**Funcionalidades**

* Listar suprimentos
* Cadastrar suprimentos
* Editar suprimentos
* Remover suprimentos

**Tecnologias**

* Java 17
* Spark Java
* Gson
* Swing (JFrame)
* Dados em memória (ArrayList)

---

# Como criar seu repositório no GitHub

1. Crie um novo repositório no [github.com](https://github.com/new) — ex: `suprimentos-hospitalares`

2. Clone localmente:

```bash

git clone https://github.com/seuusuario/suprimentos-hospitalares.git

cd suprimentos-hospitalares

1. Copie os arquivos e pastas conforme estrutura acima
2. Commit e push:
3. git add .
4. git commit -m "Primeiro commit: API + Swing frontend"
5. git push origin main

Se preferir, posso gerar um repositório online e te passar o link, mas preciso que você autorize acesso (via GitHub OAuth), que infelizmente não é possível aqui.

Se precisar, posso ajudar a preparar o ZIP do projeto para download também.

Quer?