**# Arquivos HTML**

**Código para index.html (página principal):**

<!DOCTYPE html>

<html lang="pt-BR">

<head>

<meta charset="UTF-8" />

<meta name="viewport" content="width=device-width, initial-scale=1" />

<title>Zeca Eventos - Bem-vindo</title>

<link rel="stylesheet" href="css/style.css" />

</head>

<body>

<header>

<h1>Bem-vindo ao Zeca Eventos</h1>

</header>

<main>

<nav>

<ul>

<li><a href="login.html">Acessar Minha Conta</a></li>

<li><a href="cadastro.html">Cadastrar-se</a></li>

</ul>

</nav>

<section>

<h2>Conheça nossos eventos</h2>

<p>Após login, você poderá escolher o tipo de evento e solicitar orçamento personalizado.</p>

<a href="eventos.html" class="btn">Ver Tipos de Eventos</a>

</section>

</main>

<footer>

<p>&copy; 2025 Zeca Eventos. Todos os direitos reservados.</p>

</footer>

</body>

</html>

**login.html**

<!DOCTYPE html>

<html lang="pt-BR">

<head>

<meta charset="UTF-8" />

<meta name="viewport" content="width=device-width, initial-scale=1.0"/>

<title>Login - Zeca Eventos</title>

<link rel="stylesheet" href="style.css" />

</head>

<body>

<header>

<h1>Zeca Eventos</h1>

</header>

<main>

<h2>Acessar minha conta</h2>

<form action="/login" method="post">

<label for="email">E-mail:</label>

<input type="email" id="email" name="email" required />

<label for="senha">Senha:</label>

<input type="password" id="senha" name="senha" required />

<button type="submit">Entrar</button>

</form>

<p><a href="cadastro.html">Não tem conta? Cadastre-se aqui.</a></p>

</main>

</body>

</html>

**cadastro.html**

<!DOCTYPE html>

<html lang="pt-BR">

<head>

<meta charset="UTF-8" />

<meta name="viewport" content="width=device-width, initial-scale=1.0"/>

<title>Cadastro - Zeca Eventos</title>

<link rel="stylesheet" href="style.css" />

</head>

<body>

<header>

<h1>Zeca Eventos</h1>

</header>

<main>

<h2>Cadastre-se</h2>

<form action="/cadastro" method="post">

<label for="nome">Nome completo:</label>

<input type="text" id="nome" name="nome" required />

<label for="email">E-mail:</label>

<input type="email" id="email" name="email" required />

<label for="senha">Senha:</label>

<input type="password" id="senha" name="senha" required />

<button type="submit">Cadastrar</button>

</form>

</main>

</body>

</html>

**eventos.html**

<!DOCTYPE html>

<html lang="pt-BR">

<head>

<meta charset="UTF-8" />

<meta name="viewport" content="width=device-width, initial-scale=1.0"/>

<title>Tipos de Evento - Zeca Eventos</title>

<link rel="stylesheet" href="style.css" />

</head>

<body>

<header>

<h1>Zeca Eventos</h1>

</header>

<main>

<h2>Escolha o tipo de evento</h2>

<form action="/orcamento.html" method="get">

<select name="tipoEvento" required>

<option value="">Selecione...</option>

<option value="casamento">Casamento</option>

<option value="aniversario">Aniversário</option>

<option value="formatura">Formatura</option>

<option value="corporativo">Corporativo</option>

</select>

<button type="submit">Avançar para orçamento</button>

</form>

</main>

</body>

</html>

**orcamento.html**

<!DOCTYPE html>

<html lang="pt-BR">

<head>

<meta charset="UTF-8" />

<meta name="viewport" content="width=device-width, initial-scale=1.0"/>

<title>Orçamento - Zeca Eventos</title>

<link rel="stylesheet" href="style.css" />

</head>

<body>

<header>

<h1>Zeca Eventos</h1>

</header>

<main>

<h2>Solicitar Orçamento</h2>

<form action="/orcamento" method="post">

<label for="tipoEvento">Tipo de Evento:</label>

<input type="text" id="tipoEvento" name="tipoEvento" value="(vindo de eventos.html)" required />

<label for="data">Data do evento:</label>

<input type="date" id="data" name="data" required />

<label for="convidados">Número de convidados:</label>

<input type="number" id="convidados" name="convidados" required />

<label for="observacoes">Observações:</label>

<textarea id="observacoes" name="observacoes"></textarea>

<button type="submit">Solicitar Orçamento</button>

</form>

</main>

</body>

</html>

**# Arquivo CSS**

/\* style.css \*/

body {

font-family: Arial, sans-serif;

background-color: #f9f9f9;

margin: 0;

padding: 0;

color: #333;

}

header {

background-color: #004080;

color: white;

padding: 1rem;

}

header h1 {

margin: 0;

}

nav a {

color: white;

margin-right: 1rem;

text-decoration: none;

font-weight: bold;

}

nav a:hover {

text-decoration: underline;

}

main {

padding: 1rem;

max-width: 900px;

margin: 0 auto;

}

section {

margin-bottom: 2rem;

background: white;

padding: 1rem;

border-radius: 8px;

box-shadow: 0 0 5px #ccc;

}

form input, form button {

display: block;

margin: 0.5rem 0;

padding: 0.6rem;

width: 100%;

box-sizing: border-box;

font-size: 1rem;

}

form button {

background-color: #004080;

color: white;

border: none;

cursor: pointer;

border-radius: 4px;

transition: background-color 0.3s ease;

}

form button:hover {

background-color: #003060;

}

#output {

margin-top: 1rem;

}

# **JavaScript**

// app.js

// Função para exibir mensagens na área de saída

function mostrarMensagem(msg) {

const output = document.getElementById("output");

output.innerHTML = `<p>${msg}</p>`;

}

// Função para enviar dados de um formulário via fetch (simulação)

async function enviarFormulario(url, dados) {

try {

// Simulando envio, no futuro conectar com backend real

console.log(`Enviando para ${url}`, dados);

mostrarMensagem(`Dados enviados para ${url} com sucesso!`);

return true;

} catch (error) {

mostrarMensagem(`Erro ao enviar dados: ${error}`);

return false;

}

}

// Evento de submissão do formulário Clientes

document.getElementById("formCliente").addEventListener("submit", function (e) {

e.preventDefault();

const dados = {

nome: this.nome.value,

email: this.email.value,

telefone: this.telefone.value,

};

enviarFormulario("/api/clientes", dados);

this.reset();

});

// Evento de submissão do formulário Agendamentos

document.getElementById("formAgendamento").addEventListener("submit", function (e) {

e.preventDefault();

const dados = {

data: this.data.value,

descricao: this.descricao.value,

};

enviarFormulario("/api/agendamentos", dados);

this.reset();

});

// Evento de submissão do formulário Fornecedores

document.getElementById("formFornecedor").addEventListener("submit", function (e) {

e.preventDefault();

const dados = {

nome: this.nome.value,

servico: this.servico.value,

};

enviarFormulario("/api/fornecedores", dados);

this.reset();

});:root {

    --azul-principal: #1f5a84;

    --azul-claro: #6bb7d6;

    --laranja: #f2921b;

    --fundo-claro: #f9f9f9;

    --texto: #333;

}

body {

    font-family: Arial, sans-serif;

    background-color: var(--fundo-claro);

    color: var(--texto);

    margin: 0;

    padding: 0 20px;

}

.top-header {

    display: flex;

    justify-content: space-between;

    align-items: center;

    background-color: var(--fundo-claro); /\* Agora usa o azul claro \*/

    color: white;

    padding: 8px 20px;

}

/\* Container para logo + slogan \*/

.logo-slogan {

    display: flex;

    align-items: center;

    gap: 15px;

}

.logo {

    max-width: 150px;

    height: auto;

    display: block;

    margin: 0;

}

/\* Estilo do slogan \*/

.slogan {

    font-size: 1.1rem;

    font-weight: normal;

    color: white;

}

/\* Menu do usuário \*/

.user-links {

    display: flex;

    gap: 20px;

}

.user-links a {

    color: #f2921b;

    text-decoration: none;

    font-weight: bold;

    padding: 6px 10px;

    font-size: 0.95rem;

    border-radius: 4px;

    transition: background-color 0.3s ease;

}

.user-links a:hover {

    background-color: var(--azul-principal); /\* Contraste com o azul claro \*/

    color: white;

}

/\* Evitar que links fiquem grudados na borda direita \*/

body {

    padding-right: 40px; /\* aumenta espaço à direita para o menu \*/

}

/\* Se quiser, pode ajustar o padding do header também \*/

.top-header {

    padding-left: 20px;

    padding-right: 40px;

}

/\* Outras seções mantêm seu estilo original \*/

section {

    padding: 20px 0;

}

.btn {

    display: inline-block;

    padding: 10px 20px;

    background-color: var(--laranja);

    color: white;

    text-decoration: none;

    border-radius: 5px;

    margin-top: 10px;

}

.hero {

    background-color: var(--azul-claro);

    color: white;

    padding: 40px 20px;

    text-align: center;

}

footer {

    text-align: center;

    padding: 15px;

    background-color: var(--azul-claro);

    color: white;

    margin-top: 20px;

}

.hero-banner {

    background-image: url("../Imagens/casamento.png"); /\* ajuste conforme necessário \*/

    background-size: cover;

    background-position: center;

    position: relative;

    height: 400px;

    display: flex;

    align-items: center;

    justify-content: center;

    color: white;

}

.hero-overlay {

    background-color: rgba(0, 0, 0, 0.4); /\* escurece a imagem \*/

    padding: 30px;

    text-align: center;

    width: 100%;

    max-width: 1000px;

    border-radius: 8px;

}

.hero-overlay h1 {

    font-size: 2rem;

    margin-bottom: 10px;

}

.hero-overlay p {

    font-size: 1.1rem;

    margin-bottom: 25px;

}

.search-form {

    display: flex;

    flex-wrap: wrap;

    background-color: white;

    border-radius: 6px;

    overflow: hidden;

    box-shadow: 0 2px 8px rgba(0,0,0,0.2);

}

.search-field {

    display: flex;

    flex-direction: column;

    padding: 10px;

    flex: 1;

    min-width: 150px;

}

.search-field label {

    font-weight: bold;

    color: red;

    font-size: 0.9rem;

    margin-bottom: 5px;

}

.search-field input,

.search-field select {

    padding: 8px;

    border: 1px solid #ccc;

    border-radius: 4px;

}

.search-btn {

    background-color: #d13b3b;

    color: white;

    border: none;

    padding: 0 20px;

    font-size: 1.2rem;

    cursor: pointer;

    display: flex;

    align-items: center;

    justify-content: center;

}

**# Java**

**Classe principal**

package com.zecaeventos;

import org.springframework.boot.SpringApplication;

import org.springframework.boot.autoconfigure.SpringBootApplication;

@SpringBootApplication

public class ZecaEventosApplication {

public static void main(String[] args) {

SpringApplication.run(ZecaEventosApplication.class, args);

}

}

**Classe Cliente.java**

package com.zecaeventos.model;

import jakarta.persistence.\*;

@Entity

@Table(name = "clientes")

public class Cliente {

@Id

@GeneratedValue(strategy = GenerationType.IDENTITY)

private Long id;

private String nome;

private String email;

private String telefone;

// Getters e setters

public Long getId() { return id; }

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

public String getNome() { return nome; }

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

public String getEmail() { return email; }

public void setEmail(String email) { this.email = email; }

public String getTelefone() { return telefone; }

public void setTelefone(String telefone) { this.telefone = telefone; }

}

**ClienteRepository.java** (repositório)

package com.zecaeventos.repository;

import com.zecaeventos.model.Cliente;

import org.springframework.data.jpa.repository.JpaRepository;

import org.springframework.stereotype.Repository;

@Repository

public interface ClienteRepository extends JpaRepository<Cliente, Long> {

}

**ClienteController.java** (controller REST)

package com.zecaeventos.controller;

import com.zecaeventos.model.Cliente;

import com.zecaeventos.repository.ClienteRepository;

import org.springframework.beans.factory.annotation.Autowired;

import org.springframework.web.bind.annotation.\*;

import java.util.List;

@RestController

@RequestMapping("/api/clientes")

@CrossOrigin(origins = "\*")

public class ClienteController {

@Autowired

private ClienteRepository repository;

@GetMapping

public List<Cliente> listarTodos() {

return repository.findAll();

}

@PostMapping

public Cliente salvar(@RequestBody Cliente cliente) {

return repository.save(cliente);

}

@DeleteMapping("/{id}")

public void deletar(@PathVariable Long id) {

repository.deleteById(id);

}

}

**Colaborador.java** (entidade)

package com.zecaeventos.model;

import jakarta.persistence.\*;

@Entity

@Table(name = "colaboradores")

public class Colaborador {

@Id

@GeneratedValue(strategy = GenerationType.IDENTITY)

private Long id;

private String nome;

private String funcao;

private String telefone;

private String email;

// Getters e setters

public Long getId() { return id; }

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

public String getNome() { return nome; }

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

public String getFuncao() { return funcao; }

public void setFuncao(String funcao) { this.funcao = funcao; }

public String getTelefone() { return telefone; }

public void setTelefone(String telefone) { this.telefone = telefone; }

public String getEmail() { return email; }

public void setEmail(String email) { this.email = email; }

}

**ColaboradorRepository**.**java**

package com.zecaeventos.repository;

import com.zecaeventos.model.Colaborador;

import org.springframework.data.jpa.repository.JpaRepository;

import org.springframework.stereotype.Repository;

@Repository

public interface ColaboradorRepository extends JpaRepository<Colaborador, Long> {

}

**ColaboradorController.java**

package com.zecaeventos.controller;

import com.zecaeventos.model.Colaborador;

import com.zecaeventos.repository.ColaboradorRepository;

import org.springframework.beans.factory.annotation.Autowired;

import org.springframework.web.bind.annotation.\*;

import java.util.List;

@RestController

@RequestMapping("/api/colaboradores")

@CrossOrigin(origins = "\*")

public class ColaboradorController {

@Autowired

private ColaboradorRepository repository;

@GetMapping

public List<Colaborador> listarTodos() {

return repository.findAll();

}

@PostMapping

public Colaborador salvar(@RequestBody Colaborador colaborador) {

return repository.save(colaborador);

}

@DeleteMapping("/{id}")

public void deletar(@PathVariable Long id) {

repository.deleteById(id);

}

}

**Fornecedor.java** (entidade)

package com.zecaeventos.model;

import jakarta.persistence.\*;

@Entity

@Table(name = "fornecedores")

public class Fornecedor {

@Id

@GeneratedValue(strategy = GenerationType.IDENTITY)

private Long id;

private String nome;

private String servico;

private String telefone;

private String email;

// Getters e setters

public Long getId() { return id; }

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

public String getNome() { return nome; }

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

public String getServico() { return servico; }

public void setServico(String servico) { this.servico = servico; }

public String getTelefone() { return telefone; }

public void setTelefone(String telefone) { this.telefone = telefone; }

public String getEmail() { return email; }

public void setEmail(String email) { this.email = email; }

}

**FornecedorRepository.java**

package com.zecaeventos.repository;

import com.zecaeventos.model.Fornecedor;

import org.springframework.data.jpa.repository.JpaRepository;

import org.springframework.stereotype.Repository;

@Repository

public interface FornecedorRepository extends JpaRepository<Fornecedor, Long> {

}

package com.zecaeventos.controller;

import com.zecaeventos.model.Fornecedor;

import com.zecaeventos.repository.FornecedorRepository;

import org.springframework.beans.factory.annotation.Autowired;

import org.springframework.web.bind.annotation.\*;

import java.util.List;

@RestController

@RequestMapping("/api/fornecedores")

@CrossOrigin(origins = "\*")

public class FornecedorController {

@Autowired

private FornecedorRepository repository;

@GetMapping

public List<Fornecedor> listarTodos() {

return repository.findAll();

}

@PostMapping

public Fornecedor salvar(@RequestBody Fornecedor fornecedor) {

return repository.save(fornecedor);

}

@DeleteMapping("/{id}")

public void deletar(@PathVariable Long id) {

repository.deleteById(id);

}

}

**Terceirizada.java** (entidade)

package com.zecaeventos.model;

import jakarta.persistence.\*;

@Entity

@Table(name = "terceirizadas")

public class Terceirizada {

@Id

@GeneratedValue(strategy = GenerationType.IDENTITY)

private Long id;

private String nome;

private String cnpj;

private String telefone;

private String email;

private String servicoOferecido;

// Getters e setters

public Long getId() { return id; }

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

public String getNome() { return nome; }

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

public String getCnpj() { return cnpj; }

public void setCnpj(String cnpj) { this.cnpj = cnpj; }

public String getTelefone() { return telefone; }

public void setTelefone(String telefone) { this.telefone = telefone; }

public String getEmail() { return email; }

public void setEmail(String email) { this.email = email; }

public String getServicoOferecido() { return servicoOferecido; }

public void setServicoOferecido(String servicoOferecido) { this.servicoOferecido = servicoOferecido; }

}

**TerceirizadaRepository.java**

package com.zecaeventos.repository;

import com.zecaeventos.model.Terceirizada;

import org.springframework.data.jpa.repository.JpaRepository;

import org.springframework.stereotype.Repository;

@Repository

public interface TerceirizadaRepository extends JpaRepository<Terceirizada, Long> {

}

**TerceirizadaController.java**

package com.zecaeventos.controller;

import com.zecaeventos.model.Terceirizada;

import com.zecaeventos.repository.TerceirizadaRepository;

import org.springframework.beans.factory.annotation.Autowired;

import org.springframework.web.bind.annotation.\*;

import java.util.List;

@RestController

@RequestMapping("/api/terceirizadas")

@CrossOrigin(origins = "\*")

public class TerceirizadaController {

@Autowired

private TerceirizadaRepository repository;

@GetMapping

public List<Terceirizada> listarTodos() {

return repository.findAll();

}

@PostMapping

public Terceirizada salvar(@RequestBody Terceirizada terceirizada) {

return repository.save(terceirizada);

}

@DeleteMapping("/{id}")

public void deletar(@PathVariable Long id) {

repository.deleteById(id);

}

}

**OrcamentoEvento.java** (entidade)

package com.zecaeventos.model;

import jakarta.persistence.\*;

import java.math.BigDecimal;

@Entity

@Table(name = "orcamentos\_eventos")

public class OrcamentoEvento {

@Id

@GeneratedValue(strategy = GenerationType.IDENTITY)

private Long id;

private String descricao;

private BigDecimal valorEstimado;

private BigDecimal valorFinal;

@ManyToOne

@JoinColumn(name = "cliente\_id")

private Cliente cliente;

// Getters e setters

public Long getId() { return id; }

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

public String getDescricao() { return descricao; }

public void setDescricao(String descricao) { this.descricao = descricao; }

public BigDecimal getValorEstimado() { return valorEstimado; }

public void setValorEstimado(BigDecimal valorEstimado) { this.valorEstimado = valorEstimado; }

public BigDecimal getValorFinal() { return valorFinal; }

public void setValorFinal(BigDecimal valorFinal) { this.valorFinal = valorFinal; }

public Cliente getCliente() { return cliente; }

public void setCliente(Cliente cliente) { this.cliente = cliente; }

}

**OrcamentoEventoRepository.java**

package com.zecaeventos.repository;

import com.zecaeventos.model.OrcamentoEvento;

import org.springframework.data.jpa.repository.JpaRepository;

import org.springframework.stereotype.Repository;

@Repository

public interface OrcamentoEventoRepository extends JpaRepository<OrcamentoEvento, Long> {

}

**OrcamentoEventoController.java**

package com.zecaeventos.controller;

import com.zecaeventos.model.OrcamentoEvento;

import com.zecaeventos.repository.OrcamentoEventoRepository;

import org.springframework.beans.factory.annotation.Autowired;

import org.springframework.web.bind.annotation.\*;

import java.util.List;

@RestController

@RequestMapping("/api/orcamentos")

@CrossOrigin(origins = "\*")

public class OrcamentoEventoController {

@Autowired

private OrcamentoEventoRepository repository;

@GetMapping

public List<OrcamentoEvento> listarTodos() {

return repository.findAll();

}

@PostMapping

public OrcamentoEvento salvar(@RequestBody OrcamentoEvento orcamento) {

return repository.save(orcamento);

}

@DeleteMapping("/{id}")

public void deletar(@PathVariable Long id) {

repository.deleteById(id);

}

}

**Evento.java** (entidade)

package com.zecaeventos.model;

import jakarta.persistence.\*;

import java.time.LocalDateTime;

@Entity

@Table(name = "eventos")

public class Evento {

@Id

@GeneratedValue(strategy = GenerationType.IDENTITY)

private Long id;

private String descricao;

private LocalDateTime dataHora;

@ManyToOne

@JoinColumn(name = "cliente\_id")

private Cliente cliente;

@ManyToOne

@JoinColumn(name = "orcamento\_id")

private OrcamentoEvento orcamento;

// Getters e setters

public Long getId() { return id; }

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

public String getDescricao() { return descricao; }

public void setDescricao(String descricao) { this.descricao = descricao; }

public LocalDateTime getDataHora() { return dataHora; }

public void setDataHora(LocalDateTime dataHora) { this.dataHora = dataHora; }

public Cliente getCliente() { return cliente; }

public void setCliente(Cliente cliente) { this.cliente = cliente; }

public OrcamentoEvento getOrcamento() { return orcamento; }

public void setOrcamento(OrcamentoEvento orcamento) { this.orcamento = orcamento; }

}

**EventoRepository.java**

package com.zecaeventos.repository;

import com.zecaeventos.model.Evento;

import org.springframework.data.jpa.repository.JpaRepository;

import org.springframework.stereotype.Repository;

@Repository

public interface EventoRepository extends JpaRepository<Evento, Long> {

}

**EventoController.java**

package com.zecaeventos.controller;

import com.zecaeventos.model.Evento;

import com.zecaeventos.repository.EventoRepository;

import org.springframework.beans.factory.annotation.Autowired;

import org.springframework.web.bind.annotation.\*;

import java.util.List;

@RestController

@RequestMapping("/api/eventos")

@CrossOrigin(origins = "\*")

public class EventoController {

@Autowired

private EventoRepository repository;

@GetMapping

public List<Evento> listarTodos() {

return repository.findAll();

}

@PostMapping

public Evento salvar(@RequestBody Evento evento) {

return repository.save(evento);

}

@DeleteMapping("/{id}")

public void deletar(@PathVariable Long id) {

repository.deleteById(id);

}

}

**Script SQL para criação das tabelas (MySQL):**

CREATE TABLE clientes (

id BIGINT AUTO\_INCREMENT PRIMARY KEY,

nome VARCHAR(150) NOT NULL,

email VARCHAR(150) NOT NULL,

telefone VARCHAR(50)

);

CREATE TABLE colaboradores (

id BIGINT AUTO\_INCREMENT PRIMARY KEY,

nome VARCHAR(150) NOT NULL,

funcao VARCHAR(100),

telefone VARCHAR(50),

email VARCHAR(150)

);

CREATE TABLE fornecedores (

id BIGINT AUTO\_INCREMENT PRIMARY KEY,

nome VARCHAR(150) NOT NULL,

servico VARCHAR(150),

telefone VARCHAR(50),

email VARCHAR(150)

);

CREATE TABLE terceirizadas (

id BIGINT AUTO\_INCREMENT PRIMARY KEY,

nome VARCHAR(150) NOT NULL,

cnpj VARCHAR(20),

telefone VARCHAR(50),

email VARCHAR(150),

servico\_oferecido VARCHAR(150)

);

CREATE TABLE orcamentos\_eventos (

id BIGINT AUTO\_INCREMENT PRIMARY KEY,

descricao VARCHAR(255),

valor\_estimado DECIMAL(15,2),

valor\_final DECIMAL(15,2),

cliente\_id BIGINT,

CONSTRAINT fk\_orcamento\_cliente FOREIGN KEY (cliente\_id) REFERENCES clientes(id)

);

CREATE TABLE eventos (

id BIGINT AUTO\_INCREMENT PRIMARY KEY,

descricao VARCHAR(255),

data\_hora DATETIME,

cliente\_id BIGINT,

orcamento\_id BIGINT,

CONSTRAINT fk\_evento\_cliente FOREIGN KEY (cliente\_id) REFERENCES clientes(id),

CONSTRAINT fk\_evento\_orcamento FOREIGN KEY (orcamento\_id) REFERENCES orcamentos\_eventos(id)

);

**Arquivo application.properties para Spring Boot (MySQL):**

# Configuração do banco de dados

spring.datasource.url=jdbc:mysql://localhost:3306/zecaeventos?useSSL=false&serverTimezone=UTC

spring.datasource.username=seu\_usuario

spring.datasource.password=sua\_senha

# Dialeto Hibernate para MySQL

spring.jpa.database-platform=org.hibernate.dialect.MySQL8Dialect

# Atualiza o schema automaticamente (use com cuidado em produção)

spring.jpa.hibernate.ddl-auto=update

# Mostrar SQL no console

spring.jpa.show-sql=true

# Configurações de log para SQL

logging.level.org.hibernate.SQL=DEBUG

logging.level.org.hibernate.type.descriptor.sql.BasicBinder=TRACE

**Passos para usar:**

1. Crie o banco zecaeventos no seu MySQL (ou outro banco que preferir, alterando o dialecto).
2. Execute o script SQL acima para criar as tabelas.
3. Atualize spring.datasource.username e spring.datasource.password com seu usuário/senha do banco.
4. Execute o projeto Spring Boot normalmente.

**Passo a passo para gerar o pom.xml no Spring Initializr**

1. Acesse: https://start.spring.io/
2. Preencha os campos:

* **Project:** Maven Project
* **Language:** Java
* **Spring Boot:** (use a versão estável mais recente, ex: 3.0.5)
* **Project Metadata:**
  + Group: com.zecaeventos
  + Artifact: zecaeventos
  + Name: zecaeventos
  + Package name: com.zecaeventos
  + Packaging: Jar
  + Java: 17 (ou sua versão preferida)

1. Em **Dependencies**, adicione:

* **Spring Web** (para criar APIs REST)
* **Spring Data JPA** (para persistência com JPA/Hibernate)
* **MySQL Driver** (driver para conexão com MySQL)

1. Clique em **Generate** para baixar o projeto ZIP.

**O que vem no pom.xml gerado?**

Aqui está um exemplo resumido do pom.xml gerado por esse procedimento:

xml

CopiarEditar

<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

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

<modelVersion>4.0.0</modelVersion>

<parent>

<groupId>org.springframework.boot</groupId>

<artifactId>spring-boot-starter-parent</artifactId>

<version>3.0.5</version>

<relativePath/>

</parent>

<groupId>com.zecaeventos</groupId>

<artifactId>zecaeventos</artifactId>

<version>0.0.1-SNAPSHOT</version>

<name>zecaeventos</name>

<description>Projeto Zeca Eventos</description>

<properties>

<java.version>17</java.version>

</properties>

<dependencies>

<dependency>

<groupId>org.springframework.boot</groupId>

<artifactId>spring-boot-starter-web</artifactId>

</dependency>

<dependency>

<groupId>org.springframework.boot</groupId>

<artifactId>spring-boot-starter-data-jpa</artifactId>

</dependency>

<dependency>

<groupId>com.mysql</groupId>

<artifactId>mysql-connector-j</artifactId>

<scope>runtime</scope>

</dependency>

<dependency>

<groupId>org.springframework.boot</groupId>

<artifactId>spring-boot-starter-test</artifactId>

<scope>test</scope>

</dependency>

</dependencies>

<build>

<plugins>

<plugin>

<groupId>org.springframework.boot</groupId>

<artifactId>spring-boot-maven-plugin</artifactId>

</plugin>

</plugins>

</build>

</project>