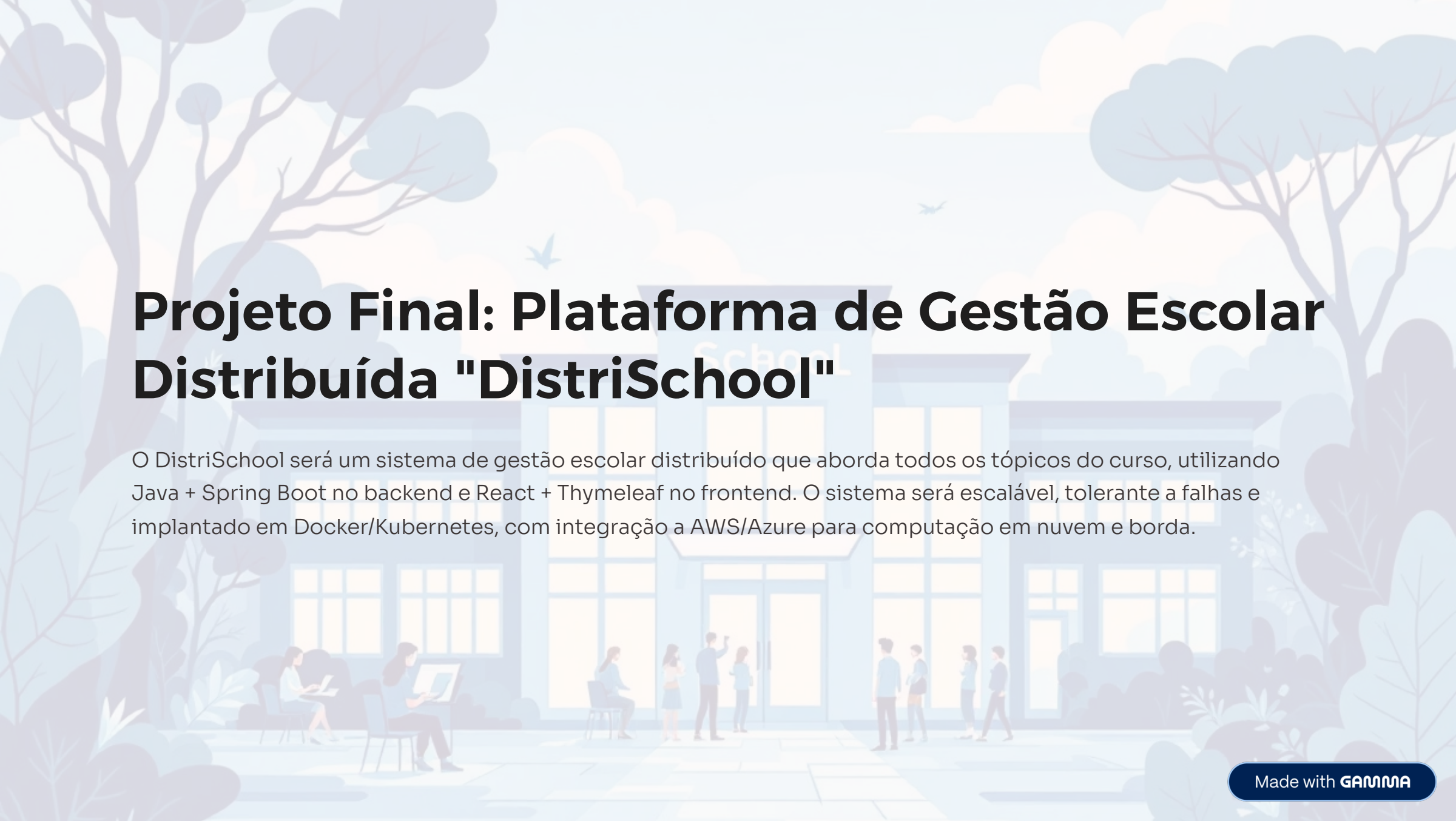
Projeto Final: Plataforma de Gestão Escolar Distribuída "DistriSchool"

O DistriSchool será um sistema de gestão escolar distribuído que aborda todos os tópicos do curso, utilizando Java + Spring Boot no backend e React + Thymeleaf no frontend. O sistema será escalável, tolerante a falhas e implantado em Docker/Kubernetes, com integração a AWS/Azure para computação em nuvem e borda.



Arquitetura do Sistema

O projeto seguirá uma arquitetura de microservices com os seguintes componentes:

Frontend (Cliente)

Tecnologia: React (ou JavaFX para desktop)

Comunicação: REST (Spring Web) + WebSocket (notificações em tempo real)

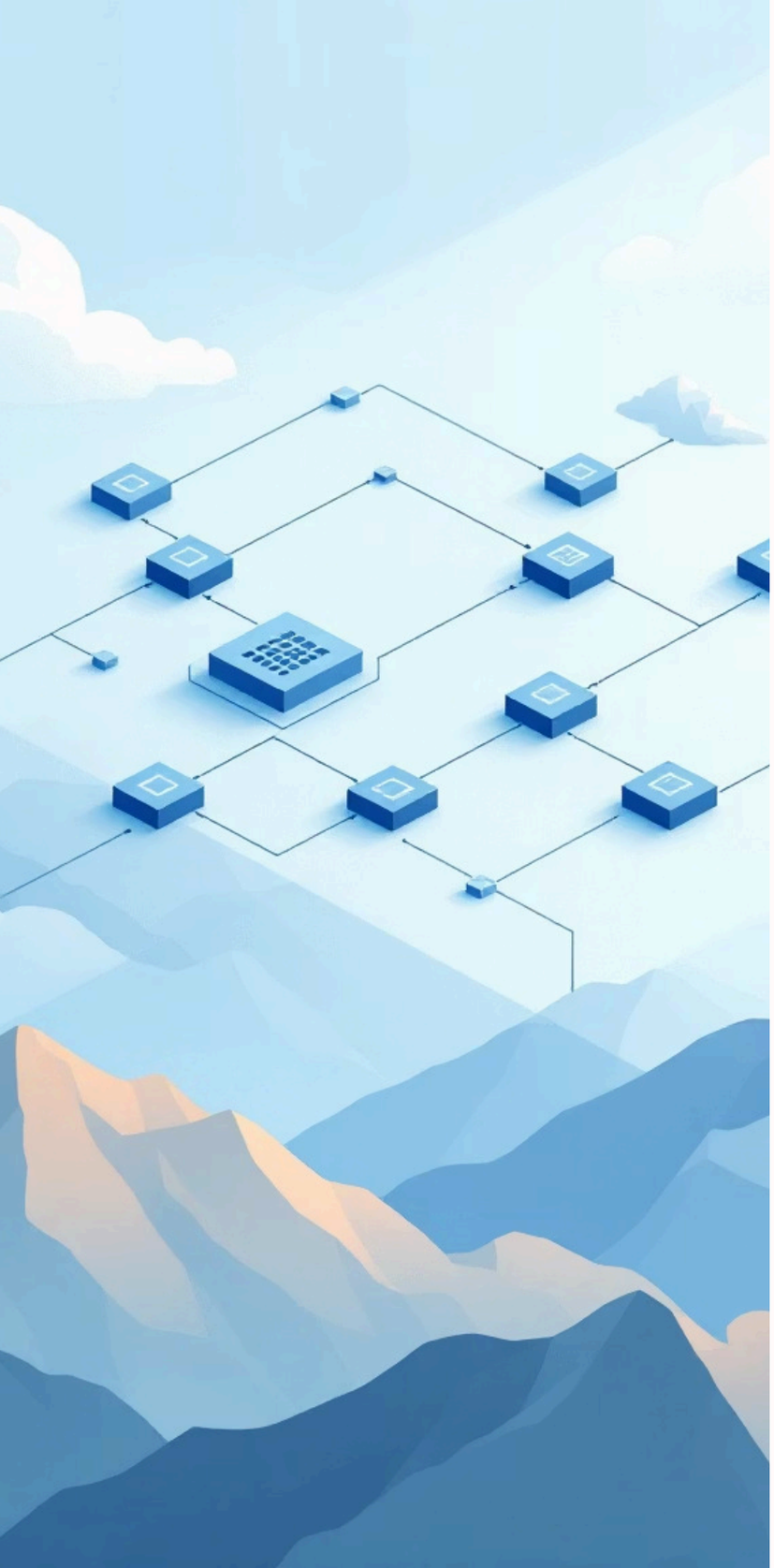
Funcionalidades: Dashboard de gestão, Cadastro de alunos/professores, Registro de presenças/notas, Relatórios e análises, Portal para pais

Backend (Serviços Distribuídos)

- API Gateway Spring Cloud Gateway - Roteamento e balanceamento de carga
- Serviço de Alunos Spring Boot + MongoDB/PostgreSQL - CRUD de alunos, busca e histórico acadêmico
- Serviço de Professores Spring Boot + PostgreSQL - CRUD de professores, atribuição de turmas e horários
- Serviço de Notas e Avaliações Spring Boot + RabbitMQ - Lançamento de notas, cálculo de médias (assíncrono)
- Serviço de Usuários Spring Boot + JWT - Autenticação, autorização e perfis (aluno, professor, admin, pai)
- Serviço de Relatórios gRPC + ML (Python) - Geração de relatórios e análises em tempo real
- Serviço de Comunicação Kafka + Spring Cloud Stream - Notificações e avisos

Infraestrutura

- Docker para conteinerização
- Kubernetes (minikube para desenvolvimento, AWS EKS/AKS em produção)
- Prometheus + Grafana para monitoramento
- ELK Stack (Elasticsearch, Logstash, Kibana) para logs
- RabbitMQ/Kafka para mensageria assíncrona
- Terraform para provisionamento na nuvem (AWS/Azure)



Funcionalidades por Unidade do Curso - Parte 1

Fundamentos de Sistemas Distribuídos

Introdução a Sistemas Distribuídos

- Requisitos de Projeto
- Modelos de Arquitetura

O gestão escolar é um exemplo clássico de sistema distribuído (alta disponibilidade, escalabilidade).

- Escalabilidade horizontal (Kubernetes)
- Tolerância a falhas (Circuit Breaker, Retry Policies)
- Microservices (Spring Boot)
- Cliente-Servidor (Frontend/Backend)
- Peer-to-Peer (Cache distribuído com Redis)

Modelos de Interação

Modelos de Falha

- Síncrona (REST/gRPC)
- Assíncrona (RabbitMQ/Kafka)
- Simulação de falhas (Chaos Engineering)
- Detecção via Health Checks (Spring Actuator)

Funcionalidades por Unidade do Curso - Parte 2

Padrões e Tecnologias

Middleware

- gRPC para comunicação interna entre serviços
- REST para APIs públicas

Padrões de Invocação

- RPC (gRPC para relatórios de desempenho)
- Mensageria (Kafka para notificações de presenças/notas)

Resiliência

- Circuit Breaker (Resilience4J)
- Retry em falhas de lançamento de notas

Computação em Nuvem/Borda

- Implantação na AWS/Azure (IaaS/PaaS)
- Edge Computing para cache regional (Cloudflare)

Virtualização

- Docker para containers
- Kubernetes para orquestração

Funcionalidades por Unidade do Curso - Parte 3

Desenvolvimento e Implantação

Configuração de Ambientes

- Terraform para AWS/Azure
- Ansible para configuração automatizada

CI/CD

- GitHub Actions/GitLab CI
- Deploy contínuo no Kubernetes

Avaliação

- Testes de carga (JMeter)
- Monitoramento (Prometheus)
- Métricas de latência/throughput (Grafana)

Plano de Desenvolvimento (8 Semanas)

Semanas 1-2

1

- Definir requisitos
- Configurar Docker/K8s
- Criar serviços básicos (Alunos, Usuários, Gestão de Professores e Técnicos Administrativos)

Semanas 3-4

2

- Implementar API Gateway
- Integrar RabbitMQ/Kafka
- Criar frontend básico

Semanas 5-6

3

- Circuit Breaker e Resiliência
- Sistema de notificações assíncrono

Semana 7

4

- CI/CD com GitHub Actions
- Monitoramento (Prometheus/Grafana)
- Testes de carga

Semana 8

5

- Documentação
- Ajustes finais
- Vídeo de demonstração

Tecnologias Utilizadas e Execução

Stack Tecnológico

Categoria	Tecnologias
Backend	Java 17+, Spring Boot, Spring Cloud, gRPC
Frontend	React (ou JavaFX)
Banco de Dados	MongoDB, PostgreSQL, Redis
Mensageria	RabbitMQ, Apache Kafka
DevOps	Docker, Kubernetes, Terraform, GitHub Actions
Cloud	AWS (ECR, EKS, S3) ou Azure (AKS, Blob Storage)
Monitoramento	Prometheus, Grafana, ELK Stack

Como Executar o Projeto

01

Pré-requisitos

- Docker + Kubernetes (minikube)
- Java 17+
- Node.js (para frontend React)

02

Clonar repositório

```
git clone https://github.com/seu-usuario/distrischool.git
cd distrischool
```

03

Subir containers

```
docker-compose up -d
```

04

Acessar frontend

http://localhost:3000

Conclusão

O DistriSchool é um projeto completo que abrange todos os tópicos do curso, desde fundamentos de sistemas distribuídos até implantação em nuvem com Kubernetes. Ele demonstra:

Escalabilidade (K8s)

Sistema capaz de crescer horizontalmente conforme a demanda, utilizando Kubernetes para orquestração automática de containers.

Tolerância a Falhas (Circuit Breaker)

Implementação de padrões de resiliência que garantem a continuidade do serviço mesmo em caso de falhas parciais.

Comunicação Síncrona/Assíncrona (REST, gRPC, Kafka)

Múltiplos protocolos de comunicação para diferentes necessidades de integração entre serviços.

Monitoramento (Prometheus/Grafana)

Sistema completo de observabilidade com métricas, logs e alertas para garantir a saúde da aplicação.

Tempo estimado: 2 meses (8 semanas) com uma equipe de 3-5 desenvolvedores.