GESTÃO DE ABRIGOS

GLOBAL SOLUTION – 2TDSPC

FIAP

GABRIEL TORRES FERNANDES – RM553635

Contexto Geral do Projeto – Gestão de Abrigos e Recursos em Tempo Real

Em um cenário impactado por eventos climáticos extremos (enchentes, tempestades intensas ou terremotos), comunidades podem ficar sem acesso a serviços essenciais e correntes logísticas ficam comprometidas. A disponibilidade rápida e precisa de informações sobre abrigos temporários e estoques de suprimentos torna-se crítica para salvar vidas e organizar o socorro humanitário. Neste contexto, propõe-se o desenvolvimento de um sistema integrado que permita:

1. **Registrar e monitorar abrigos temporários**
    - **Cada abrigo (por exemplo, ginásios, escolas ou centros comunitários) deve possuir um cadastro com identificação, localização geográfica (endereço e coordenadas aproximadas), capacidade máxima de pessoas e status de disponibilidade em tempo real (vagas restantes).**
    - **Deve ser possível atualizar automaticamente (ou manualmente) o número de usuários dentro do abrigo por meio da API, simulando ou conectando sensores IoT que contam entradas/saídas.**
2. **Gerenciar estoques de recursos essenciais**
    - **Os itens considerados (alimentos, água potável, colchões, cobertores, kits de primeiros socorros, medicamentos básicos) devem ser cadastrados com categorias e níveis mínimos críticos de estoque.**
    - **O sistema precisa alertar administradores via dashboard se algum recurso atingir limite crítico, para acionar reabastecimento imediato.**
3. **Exibir dashboards interativos para tomada de decisão**
    - **Um painel web (Java ou React Native) que consuma a API em .NET Core e apresente, em tempo real:**
        - **Lista dinâmica de abrigos ativos, organizados por proximidade (baseado em geolocalização aproximada fornecida pelo usuário ou coordenadas fixas).**
        - **Gráfico de ocupação de cada abrigo (vagas disponíveis x ocupadas).**
        - **Tabela de níveis de estoque por tipo de recurso e alerta visual para recursos em risco de esgotamento.**

- **O dashboard deve permitir filtragem por região, status do abrigo (aberto/fechado) e tipo de recurso.**

4. **Automatizar fluxo de CI/CD e testes em ambiente Azure**
    - **Configurar um pipeline no Azure DevOps para construir e publicar a API ASP .NET Core (CI), executar testes automatizados (unit e integration tests) e fazer deploy contínuo (CD) em um App Service ou Function App.**
    - **Utilizar Azure SQL (ou Cosmos DB com API MongoDB) para hospedar o banco de dados, garantindo requisitos de compliance e segurança conforme as melhores práticas estudadas em Compliance, Quality Assurance & Tests.**
5. **Modelagem de dados em bancos relacionais e não-relacionais**
    - **Criar um modelo de dados relacional em Oracle (por exemplo, tabelas Abrigo, Recurso, RegistroOcupacao, Usuário) e, paralelamente, um modelo de documento em MongoDB (collections Abrigos, Recursos), permitindo escolher o armazenamento mais adequado para operações de leitura/gravação em tempo real.**
    - **Implementar operações CRUD para cadastro, atualização e consulta de abrigos e recursos, utilizando bibliotecas ORM (Entity Framework Core para Oracle ou driver oficial para MongoDB).**
6. **Integração de componentes IoT (bônus opcional)**
    - **Simular ou conectar um "sensor" que monitora a ocupação do abrigo (por exemplo, contador de pessoas via dispositivo IoT).**
    - **Enviar dados periodicamente à API (usando MQTT ou HTTP) para atualizar automaticamente o número de ocupantes e disparar alertas quando atingir 80–90% da capacidade.**
    - **Demonstrar, ao menos por meio de uma simulação local (um script C# ou Node.js), como o fluxo IoT se conecta ao back-end.**

Objetivos Gerais

- **Desenvolver uma API RESTful em ASP .NET Core para gerenciar abrigos e recursos, utilizando conhecimentos de Advanced Business Development with .NET.**
- **Implementar pipelines de build, testes e deploy no Azure DevOps, atendendo às boas práticas de Compliance, Quality Assurance & Tests e DevOps Tools & Cloud Computing.**

- **Modelar e persistir dados em banco relacional (Oracle) e não-relacional (MongoDB), aplicando conceitos da disciplina de Mastering Relational and Non-relational Database.**
- **Criar um front-end (Java Web ou React Native) que apresente dashboards em tempo real, alinhado às matérias Java Advanced ou Mobile Application Development.**
- **Demonstrar integração básica de IoT ou predição/alerta (opcional), aplicando conceitos de Disruptive Architectures: IoT, IoB & Generative IA.**

Escopo Funcional

1. **Cadastro e consulta de abrigos**
    - **Campos principais: nome, localização (endereço + coordenadas), capacidade total, status (aberto/fechado), vagas atuais.**
    - **Endpoints:**
        - **POST /api/abrigos – criar novo abrigo.**
        - **GET /api/abrigos – listar todos os abrigos.**
        - **GET /api/abrigos/{id} – detalhes de um abrigo específico.**
        - **PUT /api/abrigos/{id} – atualizar dados ou status.**
        - **DELETE /api/abrigos/{id} – remover abrigo (opcional).**
2. **Cadastro e consulta de recursos**
    - **Campos principais: tipo (alimento, água, medicamento etc.), quantidade disponível, localização vinculada a um abrigo ou ponto de distribuição.**
    - **Endpoints:**
        - **POST /api/recursos – criar novo recurso.**
        - **GET /api/recursos – listar recursos (com filtros por abrigo).**
        - **PUT /api/recursos/{id} – atualizar quantidade ou detalhes.**
        - **DELETE /api/recursos/{id} – remover recurso.**
3. **Atualização de ocupação (manual ou via IoT)**
    - **Endpoint específico para registrar "check-in" e "check-out" de pessoas em um abrigo:**

        - **POST /api/abrigos/{id}/ocupacao – corpo JSON contendo { "tipoOperacao": "entrada" | "saida", "quantidade": número }.**
    - **A API ajusta vagasDisponiveis = capacidadeTotal – ocupacaoAtual e retorna alerta se vagasDisponiveis <= x%.**
4. **Dashboard Web/App Móvel**
    - **Web App em Java: página principal com tabelas e gráficos (por exemplo, usando Chart.js).**
    - **App Mobile em React Native: telas para:**
        - **Listagem de abrigos ordenados por proximidade (próximo/mais longe).**
        - **Visão gráfica do status de cada abrigo.**
        - **Consulta rápida de estoques por tipo de recurso.**
    - **Ambos consomem a API REST e exibem informações em tempo real (requisições a cada X segundos/minutos).**
5. **Pipelines e Deploy no Azure**
    - **Build Pipelines: restaurar pacotes NuGet, compilar solução, executar testes unitários (xUnit ou NUnit).**
    - **Release Pipelines: deploy automático para Azure App Service (API) e Azure SQL ou Cosmos DB.**
    - **Automação de Testes: criar testes de integração que validem endpoints críticos antes do deploy.**

Benefícios Educacionais

- **.NET & API: Aplicar os conceitos de design de API, injeção de dependência e práticas RESTful.**
- **Azure DevOps & Testes: Experimentar fluxo completo de CI/CD, configurar testes automatizados, validar requisitos de compliance e segurança no ciclo de vida.**
- **Bancos Oracle & MongoDB: Comparar vantagens de um modelo relacional vs. não-relacional para dados de desastres em tempo real; trabalhar com migrations, índices e consultas de alto desempenho.**
- **Front-end Web/React Native: Praticar consumo de APIs, gerenciamento de estado (React Hooks ou Java Beans), estilização responsiva e criação de dashboards simples.**

- **IoT/IA (Extra opcional): Explorar como um dispositivo IoT (ou simulação) envia dados para atualizar automaticamente o sistema, além de prever necessidades futuras (por exemplo, usar histórico de ocupação para sugerir distribuição de recursos).**