Em cinza: Preenchido pela Fatec SJC Em azul: Preenchido pelo Parceiro

Aprendizagem por Projetos Integrados 2024-2

Parceiro:	Parceiro Externo	
Período / Curso:	4º Semestre do Curso de Banco de Dados	
Professor M2:	Carlos Garcia	garcia.carlos@fatec.sp.gov.br
Professor P2:	Emanuel Mineda	mineda.emanuel@fatec.sp.gov.br
Contato do Parceiro:	Rubsney Nascimento	rubsney.nascimento@ito1.com.br

Tema do Semestre

Produto sem Contexto – Baseado na Matriz de Competências do semestre

A ITO1 é uma empresa voltada para dados. Utilizamos IOTs para coletar informações de diferentes ambientes e com isso inovar nas soluções e produtos dos nossos clientes. Quando falamos sobre o volume de dados gerados por IoTs para geolocalização de pessoas e objetos, estamos falando de grandes quantidades de dados que precisam ser gerenciados de maneira eficiente. De acordo com a matriz de conhecimento do curso, abaixo segue uma descrição sobre como desenvolver uma solução usando um banco de dados NoSQL MongoDB, Spring Boot, e Vue.js para lidar com esses dados.

1. Volume de Dados Gerados

- **Frequência de Atualização**: Dispositivos IoT de geolocalização podem enviar dados em intervalos que variam de segundos a minutos. Quanto menor o intervalo, maior o volume de dados.
- **Quantidade de Dispositivos**: O número de dispositivos em operação simultânea impacta diretamente o volume de dados. Em ambientes de grande escala, como fábricas ou hospitais e cidades, esse volume pode ser extremamente alto.
- **Tipos de Dados**:
  - **Coordenadas GPS**: Latitude, longitude, altitude e timestamps.
  - **Dados Adicionais**: Velocidade, direção, status do dispositivo, distância de um ponto de referência e controle com cerca eletrônica.

2. Desenvolvimento da Solução

Camada IoT

- **Coleta de Dados**: Dispositivos IoT, como beacons ou GPS, enviam dados de localização para um gateway ou diretamente para a nuvem.
- **Protocolos de Comunicação**: Utilização de MQTT para transmissão eficiente de dados de localização em tempo real ou HTTP para comunicação baseada em REST.

Backend com Spring Boot

- **Recepção de Dados**: Implementação de APIs RESTful para receber os dados dos dispositivos IoT.
- **Consulta de dados:** Implementação de endpoints HTTP e WebSocket para disponibilização de dados.
- **Processamento de Dados**:
  - **Geofencing**: Definição de áreas geográficas para monitoramento e envio de alertas quando um dispositivo entra ou sai dessas áreas.
  - **Análise de Trajetória**: Processamento de dados para calcular trajetórias e identificar padrões de movimento.

- **Armazenamento de Dados**:
  - **Banco de Dados NoSQL**:
    - **MongoDB**: Armazena dados de localização como documentos JSON, facilitando a manipulação e consulta.

Frontend com Vue.js

- **Visualização em Tempo Real**: Mapas interativos que mostram a localização atual de pessoas e objetos.
- **Atualizações em Tempo Real**: Implementação de WebSocket para refletir mudanças instantâneas na localização sem precisar recarregar a página.

3. Escalabilidade e Performance

- **Processamento em Lote vs. Streaming**: Implementação de processamento em lote para dados históricos e processamento em streaming para dados em tempo real.
- **Cache de Dados**: Utilização de Redis ou outro sistema de cache para melhorar a velocidade de acesso a dados frequentemente consultados.

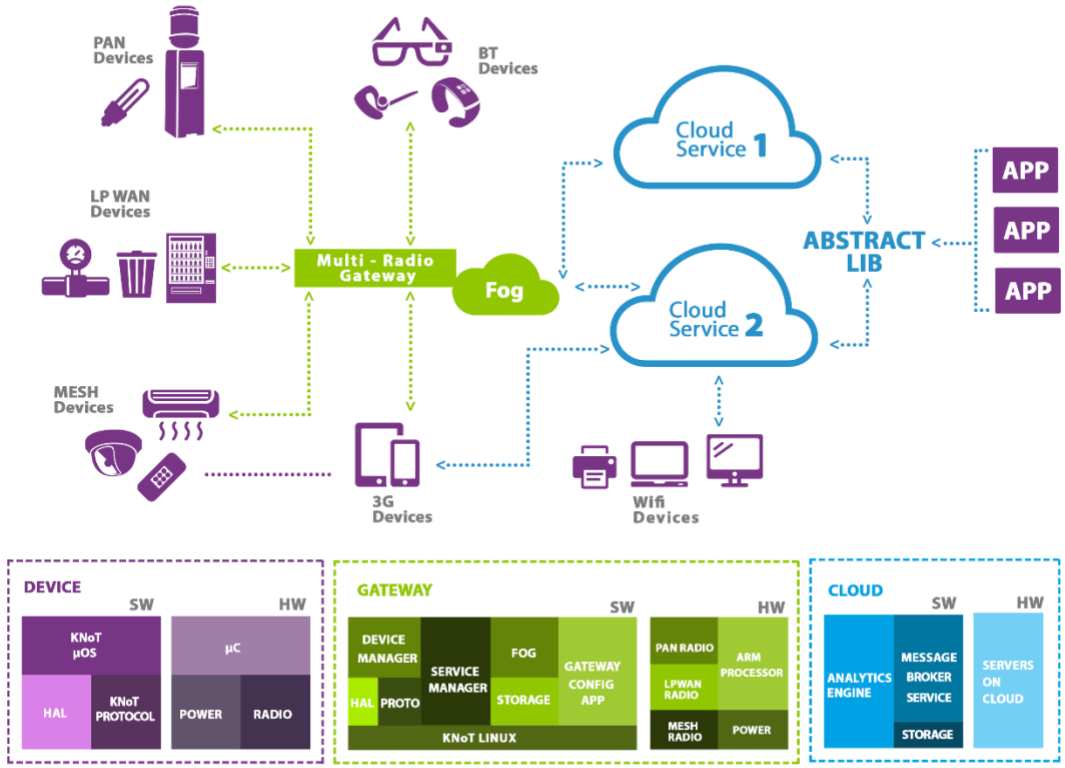
4. Segurança e Privacidade

- **Criptografia de Dados**: Garantir que os dados sejam transmitidos e armazenados de forma segura.
- **Controle de Acesso**: Implementação de autenticação e autorização robustas para proteger o acesso aos dados de geolocalização.

Considerações Finais

Essa arquitetura é capaz de lidar com o grande volume de dados gerados por dispositivos IoT de geolocalização, fornecendo uma solução escalável e eficiente para rastreamento e monitoramento em tempo real. A combinação de um backend robusto com Spring Boot, armazenamento eficiente com NoSQL, e uma interface interativa com Vue.js garante que a solução seja ágil e confiável.

Sobre o tópico **Camada IoT**



Responsabilidades:

- ITO1: DEVICE e GATEWAY
- FATEC Alunos: Solução de Cloud para recebimentos, armazenamento e apresentação das informações.

Conhecimentos ensinados no semestre

Listar todos os conhecimentos e tecnologias ensinadas no semestre – Baseado na Matriz de competências do semestre

- **API Application Programming Interface (Interface de Programação de Aplicação)**
  - Criação de APIs RESTful: Aprender sobre a criação de APIs RESTful utilizando Spring Boot, incluindo o mapeamento de endpoints, manipulação de requisições HTTP e respostas, incluindo WebSocket e integração com o banco de dados para fornecer dados aos clientes frontend.
  - Injeção de Dependências e Arquitetura Modular: Compreender o uso da injeção de dependências e da arquitetura modular oferecida pelo Spring Framework, permitindo a construção de uma aplicação bem organizada e de fácil manutenção.
  - Gerenciamento de Banco de Dados com JPA/Hibernate: Experiência no uso do Spring Data JPA e Hibernate para gerenciar operações de persistência em bancos de dados, facilitando o mapeamento objeto-relacional e simplificando as operações CRUD.
  - Segurança com Spring Security: Adquirir conhecimento na implementação de autenticação e autorização utilizando Spring Security, garantindo que apenas

usuários autenticados e autorizados tenham acesso a determinados recursos da API.

- **BANCO DE DADOS**
  - Arquitetura de Bancos de Dados NoSQL: Compreender os diferentes tipos de bancos de dados NoSQL (documentos, chave-valor, colunar, grafos) e suas respectivas características, vantagens e desvantagens.
  - Modelagem de Dados NoSQL: Saber como projetar e otimizar esquemas de dados semiestruturados para atender aos requisitos de escalabilidade e desempenho, considerando padrões de acesso e consulta.
  - Monitoramento e Tuning de Desempenho: Saber monitorar o desempenho do banco de dados, identificar gargalos e ajustar a configuração para otimizar a performance, especialmente sob cargas variáveis.
- **FRONTEND**
  - **Componentização e Reutilização de Código**: Aprender sobre a criação de componentes reutilizáveis, modularizando a interface do usuário para facilitar a manutenção e a escalabilidade do código.
  - **Gerenciamento de Estado com Vue**: Compreender do uso do Vue para gerenciar o estado global da aplicação, permitindo a sincronização eficiente de dados entre diferentes componentes e garantindo consistência nas interações do usuário.
  - **Integração com APIs RESTful**: Experiência na integração de Vue.js com APIs RESTful e WebSocket, utilizando o Axios para fazer requisições assíncronas, manipular dados e atualizar a interface em tempo real.
  - **Manipulação e Visualização de Dados em Tempo Real**: Desenvolver a habilidade no uso de bibliotecas como Vue Chart.js ou Leaflet, combinadas com Vue.js, para renderizar gráficos, mapas e outros elementos visuais, proporcionando uma experiência interativa e dinâmica.
  - **Diretivas Personalizadas e Renderização Condicional**: Aprender sobre como criar diretivas personalizadas e utilizar a renderização condicional (v-if, v-for, v-bind, etc.) para controlar o DOM, melhorando a performance e a responsividade da aplicação.

Título do Desafio

Definir o problema em uma Frase

Solução para registro e consulta de geolocalização de pessoas, ativos e outros objetos em banco de dados NOSQL escalável e de alta disponibilidade.

Descrição do Desafio

Definir entre 2 e 3 parágrafos

Desenvolver uma solução robusta para o armazenamento e consulta de dados de geolocalização em tempo real de pessoas, ativos e outros objetos em um banco de dados NoSQL escalável e de alta disponibilidade. A solução deve ser capaz de lidar com grandes volumes de dados gerados continuamente por dispositivos IoT, como wearables, tags e smartphones, garantindo que os dados possam ser acessados rapidamente e de forma confiável.

A solução deve suportar operações de leitura e escrita distribuídas por cliente (base de dados), assegurando a integridade e a consistência dos dados. Será essencial considerar aspectos como replicação, particionamento de dados (sharding), otimização de consultas espaciais, e segurança da informação, incluindo controle de acesso e criptografia.

Requisitos Funcionais e Não Funcionais

Listar entre 5 e 7 Itens

Requisitos Funcionais

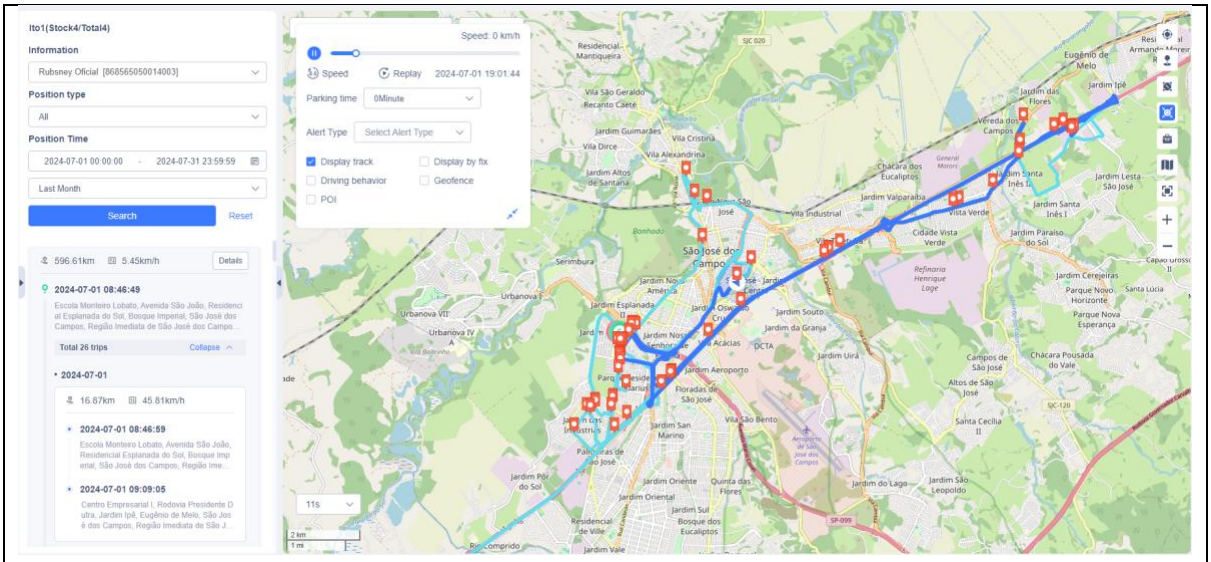
1. API Application Programming Interface (Interface de Programação de Aplicação)
    - Autenticação e Autorização:
        1. A API deve implementar um sistema de autenticação para garantir que apenas usuários autorizados possam acessar os serviços.
        2. A API deve permitir diferentes níveis de acesso (admin, usuário comum, etc.).
    - Criação, Leitura, Atualização e Exclusão (CRUD):
        1. A API deve permitir a criação, leitura, atualização e exclusão de recursos específicos, como usuários, dispositivos IoT, dados de geolocalização, etc.
    - Geolocalização:
    - A API deve fornecer endpoints para registrar e consultar dados de geolocalização de pessoas ou objetos.
        1. A API deve permitir o rastreamento em tempo real e histórico das localizações.
    - Consultas e Relatórios:
        1. A API deve fornecer mecanismos para realizar consultas complexas sobre os dados armazenados, incluindo filtros por tempo, localização, tipo de dispositivo, etc.
        2. A API deve gerar relatórios baseados em consultas definidas pelo usuário.

2. BANCO DE DADOS
    - Armazenamento de Dados de Geolocalização:
        1. O sistema deve armazenar dados de geolocalização em tempo real, incluindo latitude, longitude, e carimbo de tempo.
        2. Suporte ao armazenamento de dados de múltiplos tipos de dispositivos IoT, como wearables, tags e smartphones.
    - Consultas:
        1. Implementar suporte para consultas espaciais eficientes, permitindo a busca por proximidade, região geográfica específica e rotas.
        2. Permitir a consulta de histórico de localização para indivíduos, ativos ou objetos específicos.
    - Processamento em Tempo Real:
        1. O sistema deve processar e armazenar dados em tempo real, com latência mínima, garantindo que os dados mais recentes estejam disponíveis para consulta imediata.
        2. Suporte para alertas em tempo real baseados em eventos de geolocalização, como a entrada ou saída de uma zona geográfica.
    - Gerenciamento de Identidade e Acesso:
        1. O sistema deve fornecer autenticação e autorização robustas para controlar o acesso a dados sensíveis.
        2. Implementar diferentes níveis de acesso com base em perfis de usuários e requisitos de segurança.
3. FRONTEND

- Menus Suspensos com Filtros:
    1. Tipo de Objeto:
        1. O sistema deve oferecer um menu suspenso que permita ao usuário filtrar os resultados com base no tipo de objeto (Pessoa, Veículo).
    2. Pessoa ou Objeto:
        1. O sistema deve permitir que o usuário selecione entre uma lista de pessoas ou objetos, para focar a pesquisa em um indivíduo ou item específico.
    3. Origem do Dado:
        1. O sistema deve incluir um filtro que permita ao usuário selecionar a origem dos dados, como diferentes fontes ou dispositivos de rastreamento (Wearables, Tags e Smartphones)
    4. Período da Pesquisa:
        1. O sistema deve permitir ao usuário selecionar o período de tempo para a pesquisa, definindo uma data de início e fim para refinar os resultados.
        2. Pesquisa Rápida
            1. O sistema deve disponibilizar um campo de pesquisa rápida para que o usuário possa inserir termos específicos e encontrar resultados relevantes de forma mais eficiente. Hoje, Últimos 3 dias, Nessa semana, Último mês, etc..
- Dialog para Seleção de Funções:
    1. Velocidade de Reprodução dos Pontos ou Movimentações
        1. O sistema deve permitir que o usuário selecione a velocidade com que os pontos de parada ou a movimentação são reproduzidos no mapa, para facilitar a análise.
    2. Elemento que Mostra a Movimentação
        1. O sistema deve fornecer uma interface que permita ao usuário avançar ou retroceder no histórico de movimentação, visualizando os pontos de parada e as trajetórias em um intervalo de tempo específico.
- Mapa:
    1. Plotagem de Pontos de Parada e Movimentação
    2. O sistema deve plotar automaticamente no mapa os pontos de parada com um intervalo de 15 minutos, além de mostrar a movimentação entre esses pontos durante o período pesquisado.
    3. Funcionalidades Básicas de Mapa
        1. O sistema deve incluir funcionalidades básicas de interação com o mapa, como arrasto e zoom, para que o usuário possa navegar facilmente pela área geográfica de interesse.

EXPLICAÇÃO EM VÍDEO DO FUNCIONAMENTO DA TELA
https://rubsneynascimento.tinytake.com/msc/MTAwMjExMjdfMjM2OTAyNDA

EXEMPLO DE TELA



Requisitos Não Funcionais

4. Escalabilidade:
    - O sistema deve ser capaz de escalar horizontalmente para suportar um número crescente de dispositivos IoT e volume de dados sem degradação de desempenho.
    - Suporte para aumento e redução de capacidade sem interrupção dos serviços.
    - A API deve ser escalável horizontalmente para lidar com o aumento no número de usuários e no volume de dados de geolocalização.

5. Desempenho:
    - A solução deve garantir tempos de resposta rápidos para operações de leitura e escrita, mesmo sob alta carga.
    - A latência das operações em tempo real deve ser minimizada, mantendo-se dentro de limites aceitáveis (por exemplo, menos de 100ms).
    - A API deve responder às solicitações em tempo hábil, mesmo sob carga alta. Um tempo de resposta aceitável deve ser definido (por exemplo, menos de 200ms por requisição).
    - As funcionalidades de filtro e plotagem no mapa devem ser executadas de forma rápida e eficiente, com tempos de resposta mínimos, mesmo em grandes volumes de dados.
    - A reprodução dos pontos e movimentações deve ser suave, sem atrasos ou travamentos, garantindo uma experiência fluida ao usuário.

6. Segurança:
    - Os dados armazenados devem ser protegidos com criptografia tanto em repouso quanto em trânsito.
    - Implementar auditoria e monitoramento contínuos para detectar e responder a acessos não autorizados ou anomalias de segurança.
    - Todos os dados transmitidos pela API devem ser criptografados usando HTTPS.
    - A API deve proteger contra ataques comuns, como SQL injection, Cross-Site Scripting (XSS), e Cross-Site Request Forgery (CSRF).

7. Consistência:
    - Implementar mecanismos de resolução de conflitos para manter a integridade dos dados em caso de falhas.

8. Usabilidade
    - O sistema deve oferecer uma interface de usuário intuitiva e fácil de usar, com menus suspensos e diálogos claros e responsivos.
    - A experiência de usuário deve ser otimizada para garantir que as funções de filtro, seleção e navegação sejam acessíveis e fáceis de manipular.
9. Compatibilidade e Integração:
    - A solução deve ser compatível com diferentes plataformas de IoT e facilitar a integração com outros sistemas e serviços, como sistemas de análise de dados ou plataformas de visualização geoespacial.
    - Suporte a APIs RESTful para facilitar a integração com sistemas de terceiros e aplicações cliente.
    - O código da API deve ser modular e seguir boas práticas de programação para facilitar a manutenção e a atualização.
    - A API deve incluir documentação clara e abrangente para facilitar o desenvolvimento e a integração. Sugestão Swagger ou Postman Collections.
    - O sistema deve ser compatível com diferentes navegadores web modernos e dispositivos, incluindo desktops e tablets, garantindo acessibilidade e funcionalidade consistente em todas as plataformas.

- Fonte de Dados:
    - Temos os IOTs rodando e coletando os dados dentro do PIT.
    - Temos dados históricos de localização em formato relacional.
    - Temos os três tipos de origens: Wearables, Tags e Smartphones.
    - Infraestrutura de cloud para armazenamento e/ou desenvolvimento da solução.