**Passo a Passo da Arquitetura do Sistema e Banco de Dados do Sistema ERP-Central**

A arquitetura do **Sistema ERP-Central** foi projetad

**1. Visão Geral do Sistema ERP-Central**

A arquitetura é baseada em uma **estrutura de múltiplos camadas**, onde cada

* **Camada de Apresentação (Frontend)**: Interface co
* **Camada de Aplicação**: Servidor de aplicação (backend) qu
* **Camada de Dados**: Banco de dado

A seguir, c

**2. Camada de Apresentação (Frontend)**

* **Responsabilidade**: A cama**estoque**, \*\*finan**financeiro**, \*\***contratos**, **RH**, etc.
* **Tecnologias Possíveis**: H
* **Fluxo**:
  1. O usuário acessa a
  2. A interface envia requisições para o **servidor de aplicação**, via **API** (RESTfu

**3. Camada de Aplicação (Backend)**

* **Responsabilidade**: A camada de ap

**Componentes principais:**

1. **API Gateway**:
   * **Função**: Gerencia as requisições que chegam da camada de frontend, roteando-as para os serviços apropriados.
   * **Fluxo**: O usuário envia uma solicitação (por exemplo, criar um novo contrato). A solicitação é direcionada para a API Gateway, que a repassa para o serviço correspondente.
2. **Controladores (Controllers)**:
   * **Função**: Os controladores publicam APIs REST para interagir com o frontend. Eles recebem as requisições, chamam os métodos de negócios correspondentes e retornam os resultados.
   * **Fluxo**: A API Gateway chama o controlador que executa a lógica de negócios e interage com os serviços.
3. **Serviços (Services)**:
   * **Função**: Contêm a lógica de negócios do sistema. Eles manipulam as requisições recebidas pelos controladores, executando as regras de negócios e interagindo com os repositórios para acessar dados.
   * **Fluxo**: O controlador chama o serviço para manipular os dados ou executar ações como calcular um valor financeiro, verificar o estoque ou processar uma folha de pagamento.
4. **Repositórios (Repositories)**:
   * **Função**: Gerenciam a persistência de dados. Eles interagem com o banco de dados para salvar ou recuperar informações.
   * **Fluxo**: O serviço chama o repositório para buscar ou escrever dados no banco de dados. Por exemplo, ao cadastrar um produto, o repositório salvará os dados na tabela **Produtos**.
5. **Modelos (Models)**:
   * **Função**: Representam as entidades de dados no sistema, como **Usuários**, **Contratos**, **Produtos**, etc.
   * **Fluxo**: Os modelos definem como os dados são estruturados e persistidos no banco. Eles são usados pelos repositórios para realizar operações de banco de dados.
6. **Event Bus (Kafka)**:
   * **Função**: Gerencia eventos assíncronos. Ele publica e consome eventos entre diferentes partes do sistema.
   * **Fluxo**: Quando ocorre uma ação importante (como a criação de um novo contrato), o sistema pode gerar um evento que será consumido por outros serviços ou sistemas.

**4. Camada de Dados**

A camada de dados é responsável por armazenar e gerenciar informações persistentes do sistema.

**Componentes principais:**

1. **Banco de Dados Relacional**:
   * **Função**: O banco de dados armazena dados estruturados sobre a empresa e seus processos. Ele usa tabelas para organizar entidades como **Usuários**, **Contratos**, **Produtos**, **Estoque**, **Financeiro**, **RH**, etc.
   * **Tecnologia Sugerida**: **PostgreSQL** ou **MySQL**.
   * **Estrutura**: O banco de dados é organizado em tabelas relacionadas. Por exemplo:
     + **Tabela de Usuários**: armazena informações dos usuários.
     + **Tabela de Contratos**: armazena detalhes sobre os contratos firmados.
     + **Tabela de Produtos**: armazena os itens no estoque.
     + **Tabela de Financeiro**: armazena transações financeiras.
2. **Cache (Redis)**:
   * **Função**: A camada de cache armazena dados temporários para melhorar o desempenho. Isso permite que as informações frequentemente acessadas (como relatórios ou dados de estoque) sejam recuperadas rapidamente.
   * **Fluxo**: Quando o sistema precisa de dados que já foram acessados, ele verifica o cache antes de consultar o banco de dados. Isso reduz a carga no banco de dados e melhora a velocidade de resposta.
3. **ElasticSearch**:
   * **Função**: Utilizado para indexação e pesquisa eficiente de dados. Permite buscar dados de maneira rápida e precisa.
   * **Fluxo**: Quando o sistema precisa realizar uma pesquisa avançada (como buscar todos os contratos assinados em um período específico), ele utiliza o **ElasticSearch** para buscar dados indexados.

**5. Comunicação e Integração**

A comunicação entre os diferentes módulos e serviços é realizada por meio de APIs e eventos.

* **APIs (REST/GraphQL)**: Os controladores publicam APIs para interação entre o frontend e o backend. Isso permite que os dados sejam manipulados e apresentados de forma eficiente.
* **Event Bus (Kafka)**: Usado para comunicar eventos entre diferentes componentes do sistema de maneira assíncrona. Ele ajuda a manter os módulos desacoplados, permitindo que eles se comuniquem sem depender diretamente uns dos outros.

**6. Fluxo Completo**

1. **Interação com a Interface**: O usuário interage com a interface (frontend) para realizar uma ação, como adicionar um novo contrato ou consultar o estoque.
2. **Requisição para API Gateway**: A solicitação é enviada para o **API Gateway**, que a roteia para o **controlador** apropriado.
3. **Lógica de Negócio**: O controlador chama o **serviço** correspondente, que executa a lógica de negócios, como calcular valores ou validar dados.
4. **Persistência de Dados**: O serviço interage com o **repositório** para salvar ou consultar dados no banco de dados.
5. **Armazenamento em Cache**: Dados frequentemente acessados podem ser armazenados no **cache** para melhorar a performance.
6. **Eventos**: Se necessário, eventos são gerados e enviados para o **Event Bus** para comunicação assíncrona entre serviços.
7. **Resultado para o Usuário**: O sistema retorna a resposta para a interface, atualizando os dados apresentados ao usuário.

**Conclusão**

A arquitetura do **Sistema ERP-Central** é estruturada para garantir flexibilidade, eficiência e escalabilidade. Com componentes claramente definidos para a camada de apresentação, aplicação e dados, o sistema é capaz de atender as necessidades empresariais de forma ágil e segura. A utilização de APIs, cache e eventos assíncronos também proporciona uma experiência fluida e de alta performance para os usuários.