Tecnologias Utilizadas:

1. Linguagem de Programação

Python: É a linguagem de programação principal escolhida devido à sua versatilidade, rica biblioteca padrão e grande comunidade de suporte. O Django, um framework web em Python, será utilizado para o desenvolvimento do back-end do sistema.

2. Banco de Dados:

PostgreSQL: Escolhido como o banco de dados relacional principal, o PostgreSQL é reconhecido por sua robustez, escalabilidade e recursos avançados de segurança. Ele é adequado para o armazenamento de dados relacionais.

3. Framework Web:

- **Django:** O Django é um framework web de alto nível que oferece um conjunto abrangente de ferramentas para desenvolvimento rápido e seguro. Ele inclui um sistema de administração, autenticação, ORM, e facilita a criação de APIs RESTful, permitindo o desenvolvimento eficiente do back-end do sistema.

4. Front-end:

- React: O React é uma biblioteca JavaScript popular para criar interfaces de usuário interativas e responsivas. Ele será usado para desenvolver o front-end do sistema, permitindo uma experiência do usuário rica e dinâmica.

5. Controle de Versão:

- Git: O Git será utilizado como sistema de controle de versão para rastrear alterações no código-fonte do projeto e facilitar a colaboração entre membros da equipe de desenvolvimento. O repositório será hospedado no GitHub para controle de versão e colaboração.

6. Ferramentas de Automação:

- Docker: O Docker será usado para empacotar o aplicativo e suas dependências em contêineres, proporcionando portabilidade e facilitando a implantação consistente em ambientes diferentes.

- **Jenkins:** Jenkins pode ser utilizado para automação de integração contínua (CI) e implantação contínua (CD), permitindo a automação de testes e implantação do aplicativo em ambientes de produção.

7. **Ferramentas de Teste:**

   - **PyTest:** PyTest será utilizado para escrever e executar testes de unidade, garantindo a qualidade do código e a detecção de erros.

   - **Selenium:** Selenium será usado para testes de interface de usuário, simulando a interação do usuário com a aplicação e verificando se a interface está funcionando corretamente.

Arquitetura do Projeto:

A arquitetura do Sistema de Gestão de Biblioteca é baseada em um padrão de arquitetura em camadas (MVC - Modelo-Visão-Controlador). A seguir, estão os detalhes das camadas:

1. **Modelo (Model):**

   - Representa a camada de dados do aplicativo.

   - O Django ORM é utilizado para definir e interagir com os modelos de dados, que representam entidades como Livros, Usuários, Empréstimos, Editoras, etc.

   - Os modelos gerenciam a estrutura do banco de dados e as operações de dados, como inserção, atualização e consulta.

2. **Visão (View):**

   - Responsável pela interface do usuário e pela apresentação de dados aos usuários.

   - As visualizações são construídas usando Django para gerar templates HTML e React para criar interfaces de usuário interativas.

   - As visualizações permitem aos usuários interagir com o sistema, pesquisar livros, fazer solicitações de empréstimo e acessar informações sobre os livros.

3. **Controlador (Controller):**

   - Lida com a lógica de negócios do sistema.

   - As views do Django funcionam como controladores, processando as solicitações HTTP, executando a lógica de negócios (como validação de empréstimo, cálculo de multas) e gerando respostas para as visualizações.

   - O controlador garante o funcionamento eficiente do sistema de acordo com as regras da biblioteca.

Interação entre Componentes:

- Os usuários interagem com a interface de usuário desenvolvida em React.

- As solicitações dos usuários são tratadas pelas visualizações do Django, que processam as solicitações, interagem com o banco de dados PostgreSQL e fornecem respostas apropriadas.

- As respostas das visualizações são usadas pelo React para atualizar a interface do usuário, proporcionando uma experiência interativa.

Benefícios da Arquitetura:

- **Organização:** A arquitetura em camadas separa as preocupações e organiza o código de forma clara.

- **Reutilização:** Componentes podem ser reutilizados em diferentes partes do aplicativo, tornando o desenvolvimento mais eficiente.

- **Escalabilidade:** A arquitetura permite adicionar novas funcionalidades e expandir o sistema com facilidade.

- **Testabilidade:** Cada camada pode ser testada independentemente, garantindo a qualidade e a robustez do sistema.

Essas tecnologias e a arquitetura em camadas permitem o desenvolvimento de um Sistema de Gestão de Biblioteca eficaz, seguro e de alta qualidade. Certifique-se de documentar o projeto adequadamente, implementar testes e seguir as melhores práticas de desenvolvimento durante todo o ciclo de vida do projeto.