Visão Geral do Sistema

O **Sistema de Gestão de Biblioteca** é uma plataforma integrada projetada para ajudar bibliotecas a gerenciar de forma eficiente seu acervo de livros, registrar e acompanhar empréstimos, gerenciar a devolução de livros e manter o controle dos clientes que utilizam os serviços (alunos, professores, funcionários). As funcionalidades principais incluem:

- **Cadastro e Gerenciamento de Livros e Autores**: Adição, consulta e atualização de livros e informações dos autores.
- **Registro de Empréstimos e Devoluções**: Controle completo sobre os empréstimos de livros, incluindo datas de início e retorno.
- **Gerenciamento de Clientes**: Cadastro e manutenção de dados de clientes da biblioteca.
- **Notificações de Empréstimos**: Envio de alertas e lembretes para os clientes sobre devoluções próximas e atrasadas.

Componentes e Módulos

Interface do Usuário (Frontend):

- **Papel**: Oferece interfaces intuitivas para bibliotecários e clientes, possibilitando operações como cadastro de livros, registro de empréstimos e consultas.
- **Interação**: Criada automaticamente com a estrutura de Scaffold, a UI permite que os usuários enviem e visualizem dados em tempo real, utilizando requisições HTTP/REST para interagir com o backend.

API/Backend:

- **Papel**: Gerencia a lógica de negócios e as regras de acesso, processando as requisições recebidas do frontend. Com a estrutura de Scaffold, as operações de CRUD são geradas rapidamente para facilitar a manipulação de dados e funcionalidades essenciais.
- **Interação**: Responde às requisições da UI, faz chamadas aos modelos para acessar o banco de dados e interage com serviços externos, como notificações.

Banco de Dados:

- **Papel**: Armazena dados sobre livros, autores, clientes e registros de empréstimos. O Scaffold gera modelos e migrações, facilitando a definição e atualização das tabelas.

- **Interação**: O backend, por meio de `ActiveRecord`, realiza consultas e manipulações no banco de dados, tornando o processo de desenvolvimento mais rápido e confiável.

Serviço de Notificações:

- **Papel**: Responsável pelo envio de notificações para clientes sobre prazos de devolução e alertas de atraso.
- **Interação**: O backend usa bibliotecas específicas para enviar notificações, garantindo a comunicação com APIs externas.

Interação entre os Componentes

- **Comunicação Frontend e Backend**: O Scaffold cria automaticamente as rotas e controladores necessários, permitindo que a UI envie requisições HTTP/REST para endpoints como `/livros`, `/clientes` e `/emprestimos`.
- **Comunicação Backend e Banco de Dados**: O uso de `ActiveRecord`, gerado pelo Scaffold, facilita as operações de CRUD com comandos simples como `Livro.create`, `Emprestimo.find`, etc.
- **Comunicação Backend e Serviço de Notificações**: A interação com o serviço externo é feita através de chamadas HTTP, utilizando bibliotecas como `HTTParty` para integrar APIs de e-mail/SMS.

Diagrama de Sequência (Fluxo de Empréstimo de Livro):

1. O **bibliotecário** aciona a UI para registrar um empréstimo.
2. A **UI** faz uma requisição POST para o endpoint `/emprestimos`.
3. O **backend** válida os dados e registra o empréstimo no **banco de dados**.
4. O **backend** chama o **serviço de notificações** para enviar uma confirmação.
5. A operação é confirmada no **banco de dados**, e a resposta é enviada de volta para a **UI**.
6. A **UI** exibe a confirmação ao bibliotecário.

Decisões da Arquitetura (ADR)

Decisão: Uso de Arquitetura Monolítica com Ruby on Rails

- **Justificativa**: A escolha da arquitetura monolítica foi feita por sua simplicidade e eficiência em projetos de médio porte, como um sistema de gerenciamento de biblioteca. Essa arquitetura centraliza todos os componentes (frontend, backend, lógica de negócios e banco de dados) em uma única aplicação, o que facilita o desenvolvimento, teste e implantação inicial do sistema.

- **Impacto**: A arquitetura monolítica permite um desenvolvimento mais ágil e reduz a complexidade de integração entre diferentes módulos, garantindo que todas as funcionalidades estejam disponíveis de forma coesa. Embora possa haver desafios de escalabilidade no futuro, para o escopo atual do sistema, essa abordagem é vantajosa por ser mais fácil de gerenciar e manter.

Decisão: Integração com Serviço de Notificações

- **Justificativa**: A integração com um serviço externo foi necessária para garantir uma comunicação eficaz com os clientes em relação aos prazos de devolução.
- **Impacto**: Simplificou a implementação e melhorou a entrega de notificações.

Decisão: Escolha do Ruby on Rails como Framework

- **Justificativa**: Ruby on Rails foi escolhido como o framework principal devido à sua eficiência no desenvolvimento rápido de aplicações web, sua filosofia de “convention over configuration” e a robusta estrutura de Scaffold, que permite a criação automática de componentes de CRUD. Rails oferece um ecossistema maduro com ferramentas integradas que agilizam o desenvolvimento e a implementação de funcionalidades essenciais.
- **Impacto**: Com Ruby on Rails, a equipe de desenvolvimento pode manter um ciclo de desenvolvimento rápido e sustentável, aproveitando recursos como `ActiveRecord` para interação simplificada com o banco de dados e bibliotecas como `Devise` para autenticação e segurança. Rails também facilita futuras expansões do sistema, mantendo um código organizado e fácil de manter.

Frameworks de Frontend: DevExtreme e Bootstrap

- **DevExtreme**: DevExtreme oferece uma ampla gama de widgets que podem ser usados para enriquecer a UI, melhorando a experiência do usuário com tabelas interativas e relatórios detalhados. Embora Ruby on Rails já facilite a geração de views com componentes padrões, usar DevExtreme permitiria uma experiência mais rica e personalizável para os bibliotecários e clientes da biblioteca.
- **Bootstrap**: Ruby on Rails pode ser integrado facilmente ao **Bootstrap**, um dos frameworks de CSS mais populares para desenvolvimento responsivo. Com a inclusão do Bootstrap, o sistema de gerenciamento de biblioteca se beneficiaria de uma interface moderna e responsiva, que ajusta automaticamente a visualização em diferentes dispositivos. Isso é importante para garantir que bibliotecários e usuários possam acessar o sistema de forma eficiente tanto em desktops quanto em dispositivos móveis. A integração com Bootstrap também acelera o desenvolvimento de páginas

esteticamente agradáveis e consistentes, utilizando classes pré-definidas que simplificam o design da interface.

Requisitos Não Funcionais

1. **Desempenho**: O sistema deve suportar até 1000 usuários simultâneos sem afetar a performance, garantindo operações fluidas.
2. **Segurança**: Implementação de autenticação usando `Devise` para garantir a proteção de dados e acesso controlado às funcionalidades.
3. **Escalabilidade**: Estrutura modular do Rails, permitindo a adição de novos módulos, como gestão de reservas, de forma simples.
4. **Confiabilidade**: O uso do Scaffold ajuda a manter um código limpo e testado, garantindo a consistência nas operações básicas.
5. **Manutenibilidade**: O código gerado pelo Scaffold segue as melhores práticas de Rails, facilitando a manutenção e futuras evoluções do sistema.