**Projeto de Agenda Simples**  
**Requisitos Funcionais e Não Funcionais**

**Ferramentas Utilizadas**

**Frontend (Interface do Usuário):**

* **HTML5**: Estrutura da interface.
* **CSS3**: Estilização e design responsivo.
* **JavaScript**: Interatividade e comunicação com o backend.
* **Bibliotecas/Frameworks (opcional):**
  + **Bootstrap**: Para design responsivo.
  + **React.js** ou **Vue.js**: Para interfaces dinâmicas.

**Backend (Lógica do Sistema):**

* **PHP**: Processamento de requisições e interação com o banco de dados.
* **MySQL**: Armazenamento de dados de datas e compromissos.
* **APIs RESTful**: Comunicação entre frontend e backend.

**Ferramentas de Desenvolvimento:**

* **Visual Studio Code**: Editor de código.
* **XAMPP/WAMP**: Servidor local para testes.
* **Git**: Controle de versão.
* **Postman/Insomnia**: Testes de APIs.

**Outras Ferramentas:**

* **Figma**: Prototipagem da interface.
* **Trello/Notion**: Gerenciamento de tarefas.

**Requisitos Funcionais**

**1. Cadastro de Datas**

* **Ferramentas**: PHP, MySQL.
* Armazenar todos os dias de um ano específico.
* Cada data deve ter: data completa, dia, mês e ano.

**2. Cadastro de Compromissos**

* **Ferramentas**: PHP, MySQL, HTML5, JavaScript.
* Adicionar compromissos a uma data.
* Cada compromisso tem: descrição, horário (opcional), categoria e prioridade.

**3. Consulta de Compromissos**

* **Ferramentas**: PHP, MySQL, JavaScript.
* Consultar compromissos por:
  + Data específica.
  + Intervalo de datas.
  + Categoria ou prioridade.
* Exibir detalhes dos compromissos.

**4. Edição e Exclusão de Compromissos**

* **Ferramentas**: PHP, MySQL, JavaScript.
* Editar ou excluir compromissos existentes.
* Confirmação necessária para exclusão.

**5. Interface Amigável**

* **Ferramentas**: HTML5, CSS3, JavaScript.
* Calendário para seleção de datas.
* Formulários simples para adicionar/editar compromissos.
* Lista de compromissos organizada por data.

**Requisitos Não Funcionais**

**1. Desempenho**

* **Ferramentas**: MySQL (otimização de consultas), PHP (caching).
* Consultas rápidas (menos de 2 segundos).
* Suportar até 100.000 compromissos sem lentidão.

**2. Segurança**

* **Ferramentas**: PHP (prepared statements), MySQL (hash de senhas).
* Prevenir injeção de SQL e outros ataques.
* Senhas armazenadas com hash (se houver autenticação).

**3. Usabilidade**

* **Ferramentas**: HTML5, CSS3, JavaScript.
* Interface intuitiva e responsiva.
* Funcionar bem em dispositivos móveis e desktops.

**4. Disponibilidade**

* **Ferramentas**: Servidor confiável (ex: Apache, Nginx).
* Sistema disponível 24/7, com manutenção programada.

**5. Escalabilidade**

* **Ferramentas**: MySQL (replicação), PHP (balanceamento de carga).
* Suportar até 1.000 usuários simultaneamente.

**6. Compatibilidade**

* **Ferramentas**: PHP 7.x, MySQL 5.7, navegadores modernos.
* Funcionar em navegadores modernos (Chrome, Firefox, etc.).
* Backend compatível com PHP 7.x e MySQL 5.7.

**7. Manutenibilidade**

* **Ferramentas**: Git (controle de versão), PHP (boas práticas de código).
* Código modular e bem documentado.
* Logs de erros para facilitar depuração.

**8. Backup e Recuperação**

* **Ferramentas**: MySQL (ferramentas de backup), scripts PHP.
* Backups periódicos do banco de dados.
* Restauração fácil em caso de falha.

**9. Testabilidade**

* **Ferramentas**: PHPUnit (testes automatizados).
* Testes automatizados para 80% do código.
* Suíte de testes para funcionalidades críticas.

**10. Acessibilidade**

* **Ferramentas**: HTML5 (semântica), CSS3 (contraste).
* Interface acessível, seguindo diretrizes WCAG.

**Resumo**

**Requisitos Funcionais:**

* Cadastro e consulta de datas e compromissos.
* Edição, exclusão e lembretes de compromissos.
* Interface simples e exportação de dados.
* Autenticação de usuário (opcional).

**Requisitos Não Funcionais:**

* Desempenho rápido e segurança robusta.
* Usabilidade e acessibilidade priorizadas.
* Sistema escalável e de fácil manutenção.
* Compatibilidade com tecnologias modernas.