Considerações Finais

1. Introdução

O principal objetivo do aplicativo(app) é oferecer uma solução completa para o controle e gerenciamento da frequência dos treinos, além de centralizar a comunicação entre personal trainers e seus alunos. Com o aplicativo, os usuários poderão monitorar a assiduidade e desempenho em seus treinos, garantindo um acompanhamento eficiente de suas atividades físicas.

Assim, o aplicativo visa melhorar a organização tanto para alunos quanto para personal trainers, promovendo uma experiência mais integrada e colaborativa no gerenciamento de treinos e no compartilhamento de informações.

Para o desenvolvimento do app, utilizamos as seguintes tecnologias e ambientes:

- **React Native**: Será a principal framework para o desenvolvimento do aplicativo mobile, escolhida pela sua capacidade de criar aplicações nativas para iOS e Android com uma única base de código em JavaScript.
- **Firebase:** Usaremos o Firebase para garantir um serviço de armazenamento de dados em tempo real, autenticação de usuários e gerenciamento de notificações. O Firebase também será responsável pelo banco de dados do projeto e pela gestão de anúncios feitos pelos personal trainers, além de lidar com a sincronização em tempo real das frequências de treino.
- **Expo Dev:** Usamos o Expo como uma plataforma para poder rodar a aplicação mobile de forma nativa.

Os dois principais pontos de desafios foram:

- Dificuldade de todos os integrantes em gerenciar nosso tempo para conseguir coordenar a vida pessoal, profissional e a acadêmica.
- A falta de garantia da base de conhecimentos provida pela universidade. A disciplina que deveria focar no ponto de desenvolvimento em React Native optou por seguir caminhos diferentes.

Contextualização da importância das considerações finais para a avaliação do processo de desenvolvimento.

2. Avaliação dos Frameworks e Tecnologias Utilizadas

Tecnologias e Frameworks Escolhidos:

No desenvolvimento deste projeto, optou-se pelo uso do React Native para a construção da aplicação mobile e Firebase para gerenciamento de backend, autenticação e banco de dados em tempo real. Essas escolhas foram fundamentadas pela necessidade de desenvolver um aplicativo com desenvolvimento ágil e de integração fluida com plataformas móveis.

- **React Native:** Framework JavaScript desenvolvido pelo Facebook, utilizado para construir aplicações móveis nativas utilizando uma única base de código. Ele permite a criação de interfaces altamente responsivas para iOS e Android, com a vantagem de utilizar JavaScript e React para desenvolvimento.
- **Firebase:** Plataforma de desenvolvimento de backend fornecida pelo Google, que oferece uma gama de serviços para aplicações móveis, incluindo autenticação, banco de dados em tempo real (Firestore), armazenamento de arquivos

Pontos Positivos:

- **React Native:**
    - **Desenvolvimento Rápido:** A principal vantagem do React Native é a possibilidade de compartilhar grande parte do código entre as versões de iOS e Android, acelerando o processo de desenvolvimento. Isso resulta em uma economia significativa de tempo e esforço, pois os desenvolvedores não precisam escrever código nativo para cada plataforma.
    - **Comunidade e Suporte:** React Native possui uma comunidade ativa, com vasto conteúdo de aprendizado e suporte. Isso facilita a resolução de problemas e a busca por boas práticas, tornando o desenvolvimento mais eficiente.
    - **Performance:** Embora React Native não ofereça o mesmo nível de desempenho que aplicativos nativos, ele consegue atingir uma performance muito próxima, especialmente com o uso de componentes nativos para operações mais intensivas.
- **Firebase:**
    - **Facilidade de Integração:** A integração do Firebase com o React Native é simples e bem documentada, permitindo que recursos como autenticação, banco de dados em tempo real e armazenamento sejam facilmente implementados.
    - **Banco de Dados em Tempo Real:** O Firestore, um banco de dados NoSQL em tempo real, oferece uma grande vantagem para projetos que exigem a sincronização instantânea de dados entre os usuários, com suporte a escalabilidade automática.
    - **Autenticação Simplificada:** Firebase Authentication proporciona uma solução prática e segura para autenticar usuários através de múltiplos provedores, como e-mail/senha, Google, Facebook, entre outros. Mas nesse processo específico, utilizamos somente o método e-mail/senha.
    - **Escalabilidade e Manutenção:** Firebase oferece soluções escaláveis para aplicações de grande porte sem a necessidade de gerenciar servidores, o que reduz a complexidade operacional.
- Expo Dev

- **Facilidade de Configuração e Desenvolvimento:** Expo simplifica o processo de configuração de um projeto React Native, oferecendo um ambiente de desenvolvimento mais rápido e amigável. Com ele, é possível rodar o aplicativo em dispositivos reais ou emuladores sem a necessidade de configurar Xcode ou Android Studio, o que acelera ainda mais o processo de desenvolvimento.
- **Desenvolvimento e Testes Instantâneos:** O sistema de atualização ao vivo e recarga no Expo permite que os desenvolvedores testem alterações imediatamente em dispositivos reais ou emuladores, sem precisar recompilar o aplicativo. Isso acelera o ciclo de desenvolvimento e torna os testes mais ágeis.

Desafios e Limitações:

- **React Native:**
  - **Desafios com Performance:** Apesar de ser altamente eficiente para a maioria das aplicações, em projetos que exigem alto desempenho gráfico ou cálculos pesados, React Native pode apresentar limitações, especialmente em dispositivos mais antigos.
  - **Dependência de Plugins de Terceiros:** Embora o React Native forneça muitas funcionalidades nativas, algumas funcionalidades específicas requerem o uso de bibliotecas de terceiros, que podem não estar sempre atualizadas ou bem mantidas.
- **Firebase:**
  - **Limitações no Plano Gratuito:** Embora o Firebase ofereça um plano gratuito, ele possui limitações de uso, como a quantidade de armazenamento ou número de requisições. Para aplicações que crescem rapidamente, pode ser necessário migrar para planos pagos, o que pode aumentar os custos operacionais.
  - **Controle Limitado sobre a Infraestrutura:** O Firebase, sendo uma plataforma gerenciada, oferece pouca flexibilidade em termos de personalização da infraestrutura e das configurações avançadas, o que pode ser uma limitação dependendo das necessidades do projeto.
  - **Limitações de Consultas no Firestore:** O Firestore pode apresentar desafios de desempenho em consultas complexas e grandes volumes de dados, devido à forma como os dados são indexados.
- Expo Dev:
  - **Dificuldade de rodar a multiplataforma:** O sistema possui limitações para rodar o aplicativo em diversas plataformas diferentes, como Android, Web e iOS.

3. Análise Crítica da Arquitetura do Projeto

Arquitetura Adotada:

A arquitetura do projeto foi baseada em uma abordagem moderna e ágil, com foco na

escalabilidade e flexibilidade. Utilizamos um sistema de desenvolvimento ágil com a combinação de Scrum e Kanban para gerenciar as etapas do projeto e as tarefas de desenvolvimento. O Scrum foi adotado para sprints, onde estabelecemos metas de curto prazo, enquanto o Kanban foi usado para o gerenciamento de fluxo de trabalho contínuo, permitindo um acompanhamento mais visual e eficiente das atividades.

Além disso, a arquitetura técnica do sistema foi construída em torno do Firebase, uma plataforma de backend como serviço (BaaS) oferecida pelo Google. Optamos pelo Firebase devido à sua integração fácil com React Native e por ser uma solução que oferece escalabilidade e manutenção simplificadas para o backend de aplicações móveis.

Firebase foi utilizado para vários serviços, incluindo:

- **Banco de Dados Firestore:** Um banco de dados NoSQL em tempo real, utilizado para armazenar e sincronizar dados de forma eficiente entre os dispositivos.
- **Autenticação com Firebase Authentication:** Para gerenciar o login de usuários de maneira simples e segura.

Pontos Fortes da Arquitetura:

- **Escalabilidade e Flexibilidade:** A escolha do Firebase como solução de backend proporcionou uma arquitetura escalável, sem a necessidade de gerenciar servidores ou infraestrutura complexa. Como o Firebase oferece serviços como banco de dados em tempo real e autenticação sem servidor, isso permitiu um foco maior na lógica do aplicativo e nas funcionalidades.
- **Desenvolvimento Ágil com Scrum e Kanban:** A combinação do Scrum para sprints e Kanban para gerenciamento contínuo permitiu uma abordagem estruturada, mas flexível. Os sprints facilitaram a definição de metas de curto prazo, enquanto o Kanban proporcionou visibilidade e controle do progresso das tarefas no dia a dia, evitando sobrecarga e permitindo ajustes rápidos quando necessário.
- **Integração de Tecnologias:** A integração entre React Native e Firebase foi um dos principais pontos fortes da arquitetura. A comunicação entre a interface do usuário e o backend foi facilitada pela documentação clara e pela rica comunidade de ambos os frameworks. Isso resultou em um ciclo de desenvolvimento mais rápido e em uma aplicação robusta.
- **Simplicidade de Manutenção:** Como o Firebase é uma plataforma gerenciada, a manutenção do sistema se tornou mais simples, pois não há necessidade de configurar ou manter servidores e bancos de dados fisicamente, reduzindo o overhead operacional.

Pontos de Melhoria na Arquitetura:

- **Desempenho em Consultas Complexas:** Embora o Firestore seja altamente eficiente para consultas simples, ele pode enfrentar desafios de desempenho quando lidando com

consultas complexas ou com grandes volumes de dados. Dependendo do crescimento da base de dados, seria interessante considerar alternativas de otimização de consultas, como a criação de índices ou a divisão de dados em coleções menores.

- **Limitações de Personalização:** Embora a solução Firebase seja robusta, ela impõe algumas limitações em termos de personalização da infraestrutura. Por exemplo, não há um controle total sobre o banco de dados e a forma como ele é escalado, o que pode ser um fator limitante para um sistema que exija configurações altamente específicas ou customizadas de infraestrutura.
- **Gerenciamento de Estado do Cliente:** Em projetos com grande interatividade e múltiplas funcionalidades, o gerenciamento de estado no lado do cliente (no caso, com React Native) pode se tornar complexo. A solução adotada no projeto pode não ser suficiente para gerenciar o estado de forma eficiente à medida que o número de usuários e dados cresce, sugerindo a necessidade de técnicas mais avançadas de gerenciamento de estado.

4. Análise Crítica do Processo de Análise e Desenvolvimento

Metodologia de Desenvolvimento:

Para o desenvolvimento deste projeto, adotou-se uma metodologia ágil combinando Scrum e Kanban. O uso de Scrum facilitou a organização do trabalho em sprints, com ciclos de desenvolvimento curtos e objetivos bem definidos. Isso proporcionou foco e priorização das tarefas mais importantes ao longo do desenvolvimento, além de uma rápida adaptação às mudanças e aos feedbacks.

Já o Kanban foi utilizado via board de issues do GitHub para gerenciar o fluxo contínuo de trabalho. Ele permitiu visualizar e controlar as tarefas do dia a dia, assegurando que todas as atividades fossem monitoradas de perto e evitando que o time se sobrecarregasse com um número excessivo de tarefas simultâneas.

O processo foi bem alinhado com o objetivo de entregar uma versão funcional da aplicação em um tempo mais curto. Cada sprint focava em entregas incrementais e funcionais, enquanto o Kanban facilitava a gestão de atividades contínuas como correções de bugs, ajustes de funcionalidades e implementação de melhorias.

5. Propostas de Melhorias e Conclusões

Propostas de Melhorias para o Projeto:

1. **Refatoração do Código e Melhoria na Modularização:** Durante o desenvolvimento, algumas áreas do código podem se beneficiar de uma maior modularização. A refatoração de componentes, a separação de responsabilidades e a aplicação de princípios de design poderiam melhorar a legibilidade e a manutenção do código. Isso também facilitaria a escalabilidade do sistema à medida que novas funcionalidades fossem adicionadas.
2. **Melhoria na Estratégia de Testes:** Para garantir maior robustez e confiabilidade, é essencial que o projeto invista em uma estratégia de testes automatizados mais

abrangente. A implementação de testes unitários, de integração e end-to-end ajudaria a detectar problemas mais rapidamente, principalmente nas integrações com o Firebase e no gerenciamento de estado da aplicação com React Native. Isso também permitiria acelerar o ciclo de desenvolvimento, proporcionando feedback contínuo sobre a qualidade do código.

3. **Aprimorar o Gerenciamento de Dependências e Integrações:** Uma maior atenção à gestão de dependências entre as diferentes partes do sistema e à organização das integrações entre React Native e Firebase pode evitar bloqueios e melhorar a fluidez do desenvolvimento. Utilizar ferramentas de gerenciamento de dependências mais robustas e realizar reuniões de integração periódicas pode ajudar a minimizar riscos e a melhorar o alinhamento da equipe.
4. **Documentação e Treinamento da Equipe:** Uma documentação mais detalhada sobre a arquitetura do sistema, processos de desenvolvimento e boas práticas adotadas ajudaria a integrar novos membros na equipe de forma mais eficiente. Além disso, promover treinamentos internos sobre as tecnologias utilizadas, como React Native e Firebase, contribuiria para melhorar a expertise técnica da equipe e garantir que todos os membros estejam alinhados com as práticas recomendadas.

Reflexões Finais:

Este projeto foi um desafio em termos de desenvolvimento, com a equipe fazendo grande esforço para integrar eficientemente o React Native com o Firebase. A abordagem ágil, utilizando Scrum e Kanban, foi essencial para garantir o progresso contínuo e o cumprimento de prazos, apesar das adversidades enfrentadas ao longo do caminho.

O aprendizado adquirido durante o desenvolvimento foi significativo, especialmente em relação à escolha das tecnologias, à adaptação do processo de desenvolvimento e ao gerenciamento de fluxo de trabalho. No entanto, como qualquer projeto de software, sempre existem áreas que podem ser aprimoradas.

Por fim, o projeto serviu como um excelente ponto de partida para a equipe aplicar metodologias ágeis de forma prática e experimentar com tecnologias modernas, abrindo portas para melhorias contínuas em futuros projetos. A experiência adquirida será fundamental para a evolução da equipe e para a implementação de melhores práticas nos próximos desafios.