UNIVERSIDADE FEDERAL DE UBERLÂNDIA

SISTEMAS DE INFORMAÇÃO

GABRIEL AUGUSTO PAIVA

PEDRO HENRIQUE LOPES DUARTE

RAYANE REIS MOTA

TRABALHO PRÁTICO:

QUIZ APP COLABORATIVO COM FIREBASE

UBERLÂNDIA

2025

SUMÁRIO

1 - INTRODUÇÃO

Este relatório documenta as decisões de projeto, arquitetura, cronograma, responsabilidades, implementação e validação do aplicativo Quiz App desenvolvido para Android. O app oferece login e autenticação via Google/Firebase, sincronização de questões do Firestore, execução de quizzes por categoria e registro do desempenho do usuário tanto localmente (Room) quanto na nuvem (Firebase).

2 - OBJETIVOS

Os objetivos do projeto foram implementar autenticação individual por meio do *Firebase Auth*; baixar e armazenar localmente as questões seguindo uma abordagem *offline-first* com *Room*; executar quizzes dinâmicos com controle de pontuação e tempo; registrar e exibir o histórico do usuário com as sessões concluídas e suas estatísticas; disponibilizar um ranking acompanhado de um pequeno painel de resumo; e preparar todos os artefatos de entrega, incluindo código versionado, *APK*, relatório, vídeo curto e slides de apresentação.

3 - ESCOPO E REQUISITOS ATENDIDOS

No que diz respeito ao escopo funcional, o aplicativo realiza login com Google por meio do *Firebase Authentication*. Na primeira execução — e sempre que o usuário solicitar atualização — as categorias e as questões são baixadas do Firestore e aplicadas ao banco local através de operações de upsert, preservando a consistência. Cada sessão de quiz é composta por cinco perguntas sorteadas a partir da categoria escolhida; durante a execução, o sistema acompanha o tempo e contabiliza os acertos. Ao finalizar, o desempenho é persistido localmente em *Room* e remotamente em uma subcoleção do usuário no Firestore. O histórico apresenta as sessões em ordem de conclusão e o ranking global utiliza a soma das pontuações para ordenar os usuários e gerar um pequeno painel de indicadores. Além das funcionalidades, a entrega contempla um repositório Git documentado, um APK instalável, um vídeo de três a cinco minutos demonstrando o fluxo principal e um conjunto de slides.

4 - ARQUITETURA E DECISÕES DE PROJETO

A arquitetura adota o padrão MVVM com *ViewModel* e *StateFlow*, orquestrado por repositórios. O *Room* desempenha o papel de única fonte da verdade nas leituras em tempo de execução. A sincronização de dados segue uma estratégia *pull-based* acionada pela interface, reduzindo a complexidade de estados no contexto acadêmico; o campo *updatedAt* é preservado para permitir evolução com sincronização incremental. A composição de dependências é feita por um *AppContainer* que inicializa *FirebaseAuth*, *FirebaseFirestore*, Room e os repositórios, evitando a necessidade de frameworks de injeção. Os mapeamentos entre DTO, entidades e modelos de domínio concentram-se em data/mapper para manter a coerência e facilitar manutenção.

5 - MODELOS DE DADOS

No Firestore, mantemos documentos de usuários em */users/{uid}* com informações básicas como *e-mail*, *nome*, *foto*, *pontuação* e *carimbos de criação/atualização*. O desempenho por sessão é armazenado em */users/{uid}/quiz_history/{sessionId}*, com *categoria*, *acertos*, *tempo* e *horário de finalização*. As categorias residem em */categories/{categoryId}* com o respectivo nome, enquanto as perguntas ficam em */categories/{categoryId}/questions/{questionId}* contendo o *texto*, a *lista de opções*, a *resposta* e a *data de atualização*.

No armazenamento local, o *Room* define as tabelas de categorias, questões e a sessão de quiz jogada pelo usuário.

6 - IMPLEMENTAÇÃO POR HISTÓRIAS

A construção do aplicativo seguiu um percurso contínuo de amadurecimento. Começamos pela fundação do projeto — criação do repositório, estrutura inicial do aplicativo e integração ao Firebase — ainda com protótipos de interface que orientaram as primeiras decisões. Com a base pronta, evoluímos para o fluxo de identificação do usuário: a tela de login, a navegação após a autenticação e a preservação da sessão consolidaram o ponto de partida de toda a experiência.

Em seguida, o foco passou para o uso real do produto. Modelamos os dados e demos vida à lista de categorias com estados claros de carregamento, erro e vazio; na mesma levada, implementamos o quiz e a tela de resultados que sustentam a dinâmica das partidas. A partir desse núcleo, o histórico ganhou forma: definimos as estruturas responsáveis por registrar cada sessão, automatizamos o envio do resultado ao término do jogo e preparamos consultas locais que alimentam as estatísticas apresentadas ao usuário.

Com esse fluxo estabilizado, abrimos espaço para a comparação entre jogadores. O ranking e um pequeno painel de resumo passaram a ler a pontuação global e a apresentar o desempenho de maneira direta. Na etapa final, revisamos as rotinas de sincronização, tratando leituras e escritas, intermitência de rede e a atualização no servidor após mudanças locais. Essa revisão deixou preparado o terreno para uma sincronização incremental completa baseada em updatedAt. Encerramos reunindo os materiais de entrega: capturas de tela, este relatório, o vídeo demonstrativo e os slides.

Do ponto de vista técnico, a autenticação parte do Google Sign-In integrado ao *FirebaseAuth* e persiste o perfil do usuário tanto localmente quanto na nuvem. As questões são buscadas no Firestore, convertidas por mapeadores e inseridas no Room por meio de operações de upsert, mantendo o banco local como fonte de verdade. Durante a partida, o ViewModel seleciona aleatoriamente cinco perguntas da categoria, controla o cronômetro e contabiliza acertos de forma segura ao ciclo de vida; ao fim, a sessão é registrada no Room e replicada na subcoleção *quiz_history*. O ranking deriva de consultas sobre *users.totalScore*, garantindo que a visualização reflita o desempenho agregado.

7 - PAPÉIS E RESPONSABILIDADES

Ao longo do desenvolvimento, as frentes de trabalho caminharam de forma integrada para que cada entrega chegasse coesa à tela. Rayane Reis Mota conduziu a experiência do usuário do rascunho aos layouts finais, definindo hierarquia visual, estados de carregamento e microinterações que guiaram a navegação entre login, categorias, quiz, resultados, histórico e ranking. Em paralelo, Pedro Henrique Lopes Duarte estruturou a comunicação com o Firebase, alinhando a autenticação com Google, modelando as coleções do Firestore e garantindo leituras e escritas consistentes — inclusive o cálculo e a exposição da pontuação utilizado no ranking. Dando sustentação a tudo isso, Gabriel Augusto Paiva modelou o armazenamento local com Room, implementando entidades, DAOs e repositórios, além de organizar a composição de dependências no AppContainer e estabelecer a sincronização pull que mantém o banco local como fonte de verdade.

O trabalho aconteceu em ciclos curtos de validação: protótipos de interface orientaram contratos de dados e casos de uso; a base local consolidou a leitura estável para os ViewModels; e as integrações com o Firebase confirmaram os estados previstos pela UI. Sempre que o fluxo principal evoluía, os três revisavam juntos o impacto no histórico e no ranking para preservar coerência entre o que o usuário vê, o que se persiste e o que se sincroniza. Esse arranjo evitou retrabalho e manteve a experiência alinhada do primeiro login à visualização do desempenho.

8 - DIFICULDADES TÉCNICAS E USO DE LLM

Enfrentamos desafios típicos de Android: manter estado durante rotações e mudanças de configuração, sincronizar dados entre offline e online e integrar a autenticação *Google/Firebase* sem acoplamento com a interface. Resolvemos com *ViewModel* e *StateFlow*, *corrotinas* no escopo correto e *Room* como fonte única de verdade; a sincronização segue *pull* do *Firestore* com mapeadores e upsert, já preparada para deltas por *updatedAt*. Na autenticação, configuramos as chaves necessárias, preservamos a sessão e isolamos o cliente de login. O histórico exigiu modelo normalizado para permitir a mesma pergunta em partidas diferentes, com opções serializadas; o ranking usa a pontuação total e considera, para a próxima iteração, operações atômicas no servidor para evitar a condição de corrida.

As LLMs ajudaram a acelerar decisões e a reduzir retrabalho: sustentaram o recorte *MVVM* com repositórios, a criação do *AppContainer*, o isolamento do *GoogleSignInClient* e o uso de *upsert* com *updatedAt*; também guiaram ajustes em *DAOs* e *mapeadores*. O resultado é um código modular, legível e pronto para escalar com listeners em tempo real, migrações de esquema e testes instrumentados.

9 - CONCLUSÃO

Ao término do ciclo, o projeto entrega um repositório versionado e documentado, um APK instalável e um vídeo de três minutos que percorre login, categorias, partida, resultado, histórico e ranking. Esses materiais refletem um aplicativo que cumpre o escopo proposto: a experiência de uso se mantém estável mesmo sem conectividade imediata, graças ao Room como fonte de verdade, e a integração com o Firebase garante autenticação confiável e publicação dos resultados no histórico e no ranking.

A arquitetura em *MVVM*, apoiada por repositórios e mapeadores consistentes, deixa o código legível e modular, facilitando a manutenção e evolução. Do ponto de vista pedagógico, o trabalho consolidou boas práticas de ciclo de vida, persistência local e sincronização com serviços remotos. Como desdobramento natural, apontamos três frentes de continuidade: concluir a sincronização incremental com listeners em tempo real e deltas guiados por *updatedAt*; ampliar a cobertura de testes (*DAOs*, *repositórios* e *fluxos de UI)* para reduzir regressões; e iniciar a migração gradual para Jetpack Compose, em paralelo ao endurecimento das regras de segurança do Firestore e à instrumentação de métricas de uso. Em síntese, o resultado é um produto funcional e uma base tecnológica pronta para escalar com previsibilidade.