PONTIFÍCIA UNIVERSIDADE CATÓLICA DE GOIÁS
CURSO DE ANÁLISE E DESENVOLVIMENTO DE SISTEMAS

CAMILA MARTINS SOUSA
GIOVANNA CASTRO FREITAS
PABLO LEANDRO DE OLIVEIRA GLÓRIA

PROJETO INTEGRADOR: EVEX

Goiânia - GO
2025

CAMILA MARTINS SOUSA
GIOVANNA CASTRO FREITAS
PABLO LEANDRO DE OLIVEIRA GLÓRIA

PROJETO INTEGRADOR: EVEX

Trabalho apresentado à disciplina de Projeto Integrador do 5° período do Curso de Análise e Desenvolvimento de Sistemas da Pontifícia Universidade Católica de Goiás, como requisito parcial para a obtenção de nota.

Orientador: Prof. Rafael Leal Martins

Goiânia - GO
2025

LISTA DE FIGURAS

LISTA DE TABELAS

SUMÁRIO

1 INTRODUÇÃO

Este documento tem como objetivo integrar e aplicar os conhecimentos adquiridos ao longo do curso, relacionado às disciplinas do 5° período do curso de Análise e Desenvolvimento de Sistemas, abrangendo desde o planejamento até a entrega final de um software. O tema deste documento é o desenvolvimento do Evex, um sistema de software projetado para gerenciar eventos acadêmicos, com foco em palestras e congressos realizados para a comunidade acadêmica da PUC-GO.

O Evex visa facilitar a organização, inscrição e disseminação de conteúdos de eventos. A escolha do tema se justifica pela crescente dificuldade apresentada pelos próprios estudantes à plataforma de eventos da instituição, que apresenta falta de praticidade devida à navegação por múltiplas abas para confirmação de inscrições e à ausência de uma centralização clara das informações sobre eventos, congressos e palestras disponíveis.

Os objetivos da pesquisa são desenvolver um protótipo robusto para a gestão de eventos, especificando seus requisitos, arquitetura e ciclo de vida, além de propor uma solução que atenda às necessidades dos organizadores e participantes. O propósito é criar um software que melhore a eficiência na administração de eventos e proporcione uma experiência de usuário intuitiva.

2 DESENVOLVIMENTO

2.1 ESCOPO DO PROJETO

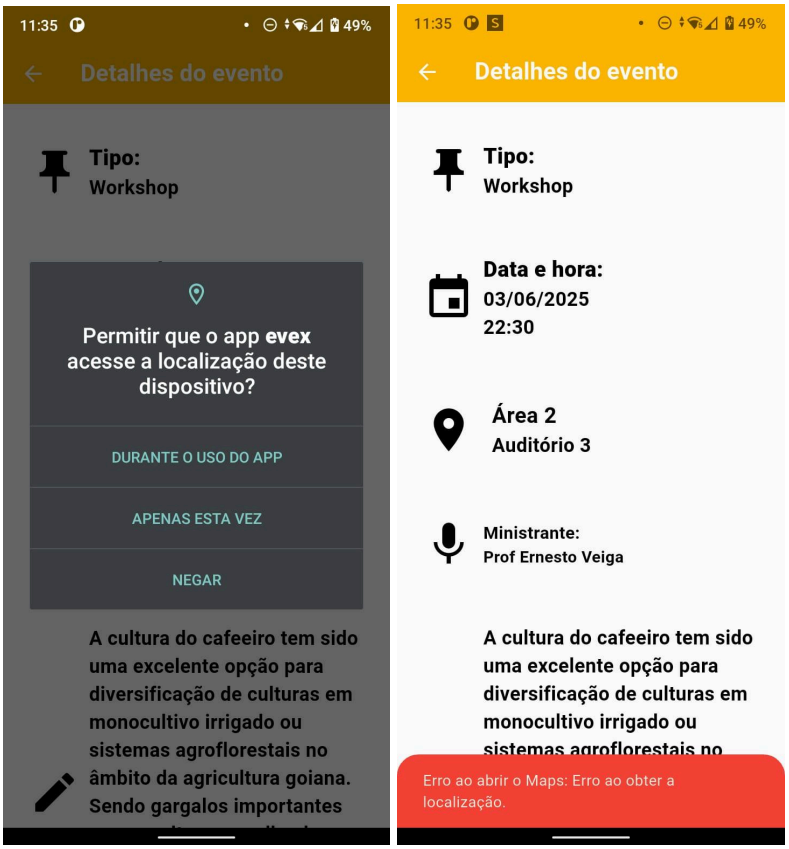
O projeto Evex tem como objetivo desenvolver um aplicativo mobile para gerenciar e divulgar eventos acadêmicos (como palestras, workshops e congressos) na Pontifícia Universidade Católica de Goiás, integrando e aplicando os conhecimentos adquiridos ao longo do curso de desenvolvimento de software. O sistema visa resolver a fragmentação e a falta de praticidade na plataforma atual de eventos da instituição, oferecendo uma solução centralizada, intuitiva e personalizada. Os objetivos específicos incluem:

- Simplificar o acesso a informações sobre eventos acadêmicos, permitindo busca e inscrição de forma prática.
- Aumentar o engajamento da comunidade acadêmica, reduzindo barreiras de acesso e incentivando a participação em eventos.
- Demonstrar a aplicação prática de conceitos de análise, modelagem e desenvolvimento de software em um protótipo completo.

O Evex apresenta as seguintes funcionalidades:

- **Sistema de Cadastro e Login:** Autenticação via e-mail institucional da PUC-GO (implementada como protótipo, devido falta de acesso real ao banco de dados da instituição) para garantir identificação do usuário por meio de matrícula (alunos) e nome(professor) trazendo praticidade.
- **Gestão de Eventos:** Criação, edição e categorização de eventos (ex. palestras, workshops, congressos), com informações sobre data, horário, local (integrado ao Google Maps) e uma breve descrição.

FIGURA 1 - LOCALIZAÇÃO E INTEGRAÇÃO



- **Recursos Sociais:** Funcionalidades de envio de notificações push e lembretes automáticos sobre eventos inscritos.
- **Interface Intuitiva:** Design mobile-first, com navegação simplificada, evitando múltiplas abas e centralizando informações para melhorar a experiência do usuário.

O público-alvo do Evex são os estudantes e professores da PUC-GO. Os estudantes, especialmente, se beneficiarão da facilidade de acesso a eventos oferecidos pela faculdade, enquanto os professores poderão divulgar eventos e gerenciar inscrições de forma eficiente, também podendo ser uma ferramenta para criar e gerenciar atividades.

O escopo do Evex abrange:

- Desenvolvimento de um aplicativo mobile funcional, com interface otimizada somente para Android.

- Implementação de um protótipo de autenticação via e-mail institucional, sem integração real com o banco de dados da PUC-GO.
- Gestão de eventos acadêmicos restrita ao contexto da PUC-GO, incluindo palestras, workshops e congressos.
- Integração com Google Maps para exibição de locais de eventos.
- Funcionalidades sociais básicas, como lembretes e notificações.

O projeto apresenta as seguintes restrições:

- **Ausência de Integração com Banco de Dados Institucional:** Devido à falta de acesso ao sistema da PUC-GO, a autenticação será simulada como protótipo, utilizando dados fictícios.
- **Escopo limitado à PUC-GO:** O Evex é projetado exclusivamente para eventos acadêmicos da instituição, não sendo aplicável a outras universidades ou contextos sem adaptações.
- **Recursos Sociais Básicos:** Funcionalidades como lembretes e notificações serão limitadas a canais básicos (ex. push notifications), sem integração com sistemas avançados de comunicação.
- **Dependência de Tecnologias Externas:** A integração com Google Maps depende de APIs externas, sujeitas a limitações de uso gratuito.

2.2 REQUISITOS DO PROJETO

Abaixo, são apresentados os requisitos funcionais e não funcionais do sistema, detalhando as funcionalidades esperadas e as características de desempenho, usabilidade, segurança, entre outros, necessárias para garantir a qualidade do aplicativo.

Os requisitos funcionais descrevem as funcionalidades que o sistema deve oferecer para atender às necessidades dos usuários.

TABELA 1 – REQUISITOS FUNCIONAIS

Identificador	Nome	Descrição
RF01	Cadastro	O usuário deve se cadastrar no aplicativo informando obrigatoriamente e-mail institucional, nome completo e senha.
RF02	Autenticação e Autorização	O login deve ser realizado com o e-mail institucional do aluno ou professor e a senha cadastrada, simulando autenticação (protótipo, sem integração com o banco de dados da PUC-GO).
RF03	Gerenciamento de usuário	O sistema deve categorizar usuários em 2 perfis: Aluno (Discente) e Professor (Docente), com base nas informações de cadastro, determinando permissões especificadas (ex. criação de eventos para docentes).
RF04	Criação de Eventos	Usuários do tipo docente devem poder criar e editar eventos, informando nome, tipo (ex. palestra, congresso), modalidade (presencial ou online), ministrante, local (integrado ao Google Maps), capacidade, data e horário.
RF05	Detalhamento de eventos	O usuário deve acessar uma tela com descrição detalhada de um evento selecionado, contendo nome, tipo, ministrante, modalidade, local, capacidade, data e horário.
RF06	Inscrição em Eventos	O usuário deve poder se inscrever diretamente em eventos de seu interesse pelo aplicativo.

RF07	Validação de conflito de horário	O sistema não deve permitir inscrição em eventos com horários conflitantes.
RF08	Alerta de conflito de horário	Caso o usuário tente se inscrever em eventos com conflito de horário, o sistema deve exibir uma mensagem de alerta informando a existência de uma inscrição no mesmo horário.
RF9	Validação de capacidade	O sistema não deve permitir inscrição em eventos que atingiram a capacidade máxima.
RF10	Alerta de capacidade máxima	Caso o usuário tente se inscrever em um evento com capacidade máxima atingida, o sistema deve exibir um alerta informando que não há vagas disponíveis.
RF11	Cancelamento de inscrição	O usuário deve poder cancelar sua inscrição em um evento até uma hora antes do início.
RF12	Desativação de cancelamento	Após o limite de uma hora antes do evento, a opção de cancelamento deve ser desativada, exibida em cinza, indicando que a ação não está disponível.
RF13	Notificações e Lembretes	O sistema deve enviar notificações no dia do evento e uma hora antes do início, como lembretes de participação.
RF14	Relatório de Eventos	O sistema deve gerar relatórios com informações sobre inscrições, presenças confirmadas e inscrições canceladas para cada evento.

Os requisitos não funcionais especificam as características de desempenho, usabilidade, segurança e outros aspectos que garantem a qualidade e a adequação do sistema às necessidades dos usuários.

TABELA 2 – REQUISITOS NÃO FUNCIONAIS

Identificador	Nome	Categoria	Prioridade	Descrição
RNF01	Interface	Usabilidade	Alta	O aplicativo deve oferecer uma interface simples, intuitiva e responsiva, compatível com dispositivos móveis Android (versão 8.0 ou superior).
RNF02	Tempo de Resposta	Desempenho	Alta	O aplicativo deve carregar a lista de eventos em no máximo 15 segundos, sob condições normais de rede.
RNF03	Capacidade de acessos simultâneos	Escalabilidade	Alta	O sistema deve suportar pelo menos 100 usuários simultâneos, especialmente em períodos de alta demanda, como semanas acadêmicas.
RNF04	Criptografia de Segurança	Segurança	Alta	Os dados dos usuários devem ser armazenados com autenticação protegida e criptografia de

				dados sensíveis, garantindo segurança.
RNF05	Compatibilidade	Compatibilidade	Alta	O aplicativo deve funcionar em sistemas Android (versão 8.0 ou superior), garantindo compatibilidade com os dispositivos utilizados pelos usuários.
RNF06	Desenvolvimento de Software	Manutenibilidade	Alta	O código deve seguir boas práticas de engenharia de software, com estrutura modular e documentação clara, para facilitar futuras atualizações e manutenções.
RNF07	LGPD	Privacidade	Alta	O sistema deve cumprir as normas da Lei Geral de Proteção de Dados (LGPD), garantindo a privacidade e o uso responsável dos dados pessoais dos usuários.
RNF08	Disponibilidade do Sistema	Confiabilidade	Alta	O sistema deve manter uma disponibilidade mínima de 80%, funcionando adequadamente mesmo em horários de alto acesso.

Os requisitos de interface complementam os requisitos funcionais e não funcionais, focando na experiência do usuário e na usabilidade do aplicativo.

TABELA 3 – REQUISITOS DE INTERFACE

Identificador	Categoria	Descrição
RI01	Layout Responsivo	A interface deve se adaptar a diferentes tamanhos de tela (smartphones e tablets), garantindo boa visualização em orientação vertical.
RI02	Navegação Intuitiva	O aplicativo deve ter menus claros e ícones identificáveis, com acesso às principais funcionalidades (visualizar eventos, consultar agenda, configurações) em até três toques.
RI03	Diferenciação de Usuário	A interface deve diferenciar os perfis de usuário: Alunos (Discentes) podem visualizar e se inscrever em eventos; Professores (Docentes) têm um ícone adicional no menu para criação e edição de eventos.
RI04	Destaque para eventos abertos	Eventos devem ser exibidos com etiquetas coloridas: verde (inscrições abertas com vagas), amarela (vagas limitadas), vermelha (vagas esgotadas), cinza (evento encerrado).
RI05	Mensagens de Erro e confirmação	A interface deve exibir mensagens claras para ações como inscrição realizada, conflito de horário, inscrição cancelada, limite de participantes atingido e presença confirmada.

RI06	Cancelar inscrição	O botão de cancelamento deve ser desativado (visual cinza, sem clique) automaticamente uma hora antes do início do evento, indicando que o prazo expirou.

FIGURA 2 - ERROS E VALIDAÇÃO DE ENTRADA DE DADOS

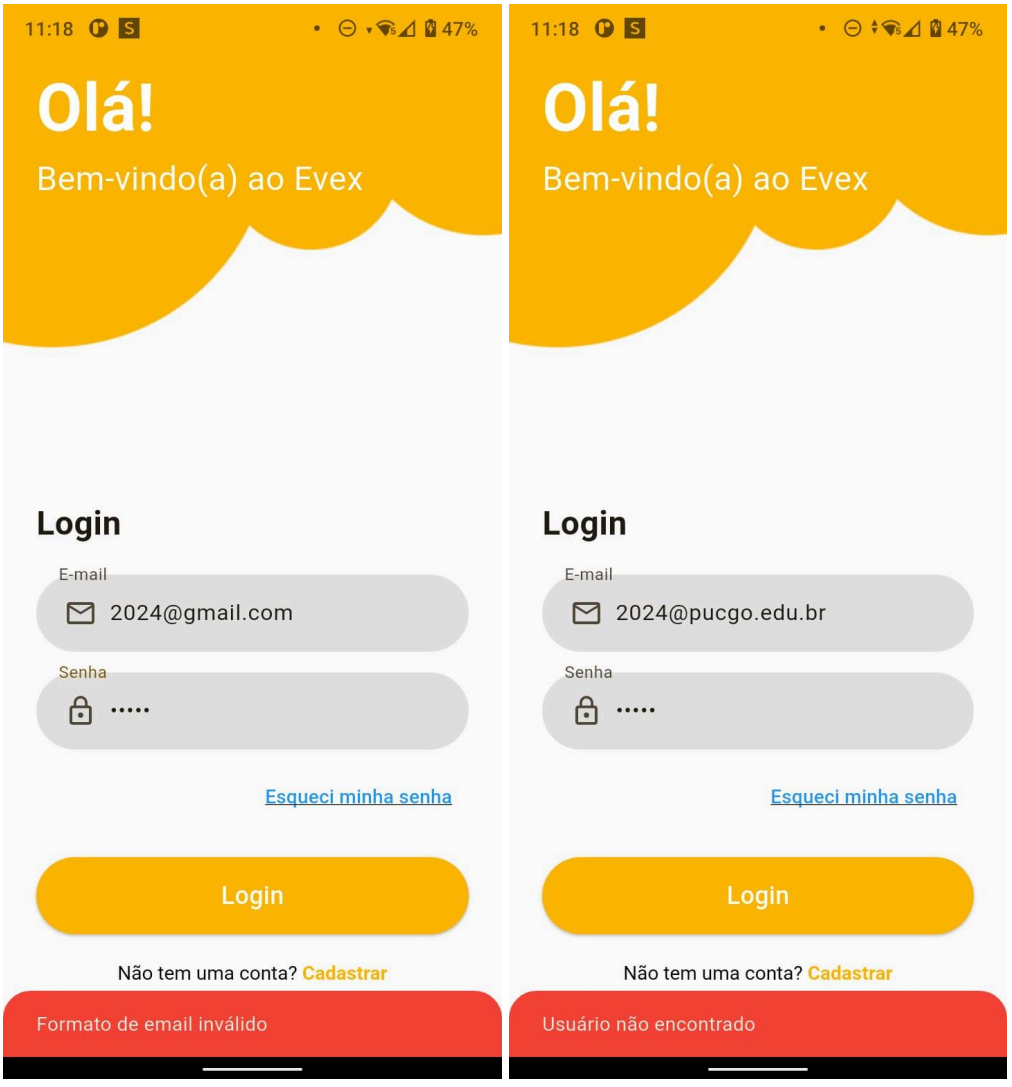
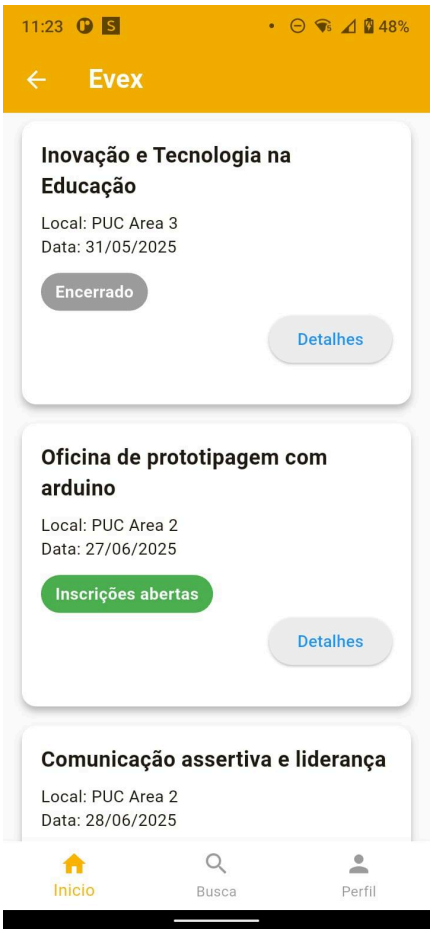


FIGURA 3 - EVENTOS ETIQUETADOS



2.3 ARQUITETURA DE SOFTWARE

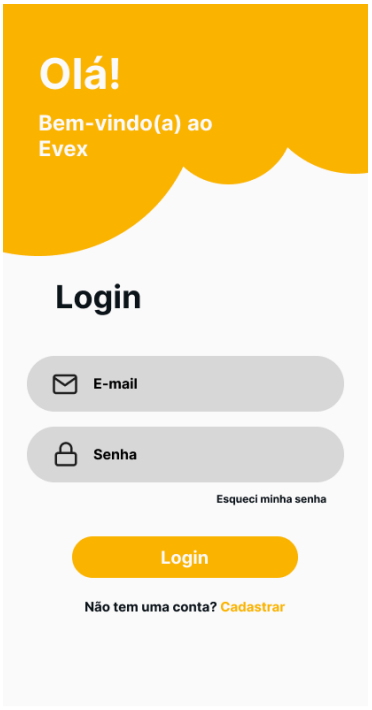
O aplicativo Evex adota a arquitetura 3-Tier (ou arquitetura em três camadas), uma abordagem modular que separa as responsabilidades do sistema em três camadas distintas: Apresentação (Frontend), Lógica de Negócio (Backend) e Dados (Data Layer). Essa arquitetura foi escolhida por sua capacidade de promover escalabilidade, manutenibilidade e separação clara de responsabilidades, facilitando o desenvolvimento e a evolução do sistema.

O Evex utiliza Flutter/Dart para a camada de apresentação, Firebase (Firestore, Authentication, Storage e Cloud Functions) para a lógica de negócio e armazenamento de dados, e Firestore Security Rules para gerenciar permissões. A arquitetura suporta o objetivo do Evex de centralizar a gestão e divulgação de eventos acadêmicos da PUC-GO, garantindo uma experiência de usuário intuitiva, sincronização em tempo real e segurança dos dados.

Camada de Apresentação (Frontend)

- **Tecnologia:** Flutter/Dart
- **Responsabilidades:**
  - Exibir dados recuperados do Firestore, como listas de eventos e detalhes de eventos;
  - Gerenciar a navegação entre telas, como login, cadastro, lista de eventos e perfil do usuário.
  - Capturar entradas do usuário (ex. credenciais de login, formulários de inscrição) e enviá-las à camada de Lógica de negócio.
- **Principais Telas:**
  - **Tela de Login:** Permite autenticação via e-mail institucional e senha.

FIGURA 4 - TELA LOGIN

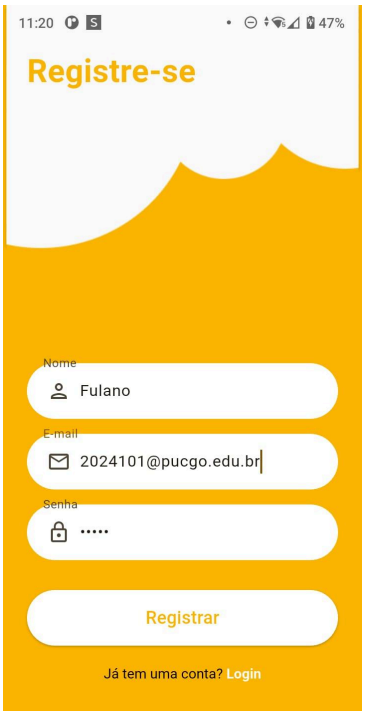


  - **Tela de Eventos Principais:** Exibe eventos filtrados.

FIGURA 5 - TELA DE EVENTOS

- **Tela de Cadastro:** Coleta informações do usuário (e-mail, nome, senha).

FIGURA 6 - TELA DE CADASTRO



- **Tela de Perfil do Usuário:** Exibe informações do usuário e eventos inscritos.

FIGURA 7 - PERFIL DO USUÁRIO

- **Exemplos de Fluxo:**
  1. O usuário insere credenciais na tela de login.
  2. As credenciais são enviadas ao Firebase Authentication via camada de Lógica de Negócio.
  3. Após autenticação, a tela de eventos principais exibe dados do Firestore.

Camada de Lógica de Negócio (Backend)

- **Tecnologias:** Dart, Firebase Cloud Functions, Firestore Security Rules, API de Gerenciamento de E-mails (Node.js).
- **Responsabilidades:**

- Processar autenticação de usuários via Firebase Authentication.
- Validar dados, como verificar se o e-mail segue o padrão institucional da PUC-GO.
- Aplicar regras de negócio, como impedir inscrições em eventos com conflito de horário ou capacidade máxima atingida.
- Gerenciar permissões de acesso com base no perfil do usuário (Aluno ou Professor).

- **Exemplo de Fluxo:**
  1. Durante o cadastro, a API verifica o e-mail institucional.
  2. A matrícula é extraída do e-mail para determinar o perfil (aluno ou professor).
  3. Regras de segurança do Firestore restringem o acesso a dados conforme o nível de permissão.

Camada de Dados (Data Layer)

- **Tecnologia:** Firestore (banco de dados NoSQL).
- **Responsabilidades:**
  - Armazenar dados de eventos (nome, tipo, modalidade, ministrante, local, capacidade, data, horário) e usuários (e-mail, nome, perfil).
  - Gerenciar permissões de acesso via Firestore Security Rules, garantindo que apenas usuários autenticados acessem dados relevantes.
  - Fornecer sincronização em tempo real para atualizar a interface com alterações nos dados (ex. novos eventos ou inscrições).
- **Exemplos de Fluxo:**
  1. A aplicação consulta a coleção “eventos” no Firestore para exibir a lista de eventos.

2. Regras de segurança garantem que apenas docentes possam criar/editar eventos.
3. Alterações (ex. nova inscrição) são refletidas em tempo real na interface.

Fluxo de Comunicação

1. O usuário interage com a interface (ex. clica para inscrever-se em um evento).
2. A camada de apresentação (Flutter) envia a solicitação à camada de lógica de negócio (Dart ou Firebase Cloud Functions).
3. A lógica de negócio valida a solicitação (ex. verifica conflito de horário ou capacidade) e interage com o Firestore.
4. O Firestore processa a solicitação (salva ou recupera dados) e retorna o resultado.
5. A camada de lógica de negócio processa o resultado e o envia à camada de apresentação.
6. A interface é atualizada com os dados retornados (ex. confirmação de inscrição).

Diagrama de Componentes

FIGURA 8 – DIAGRAMA DE COMPONENTES

Benefícios da Arquitetura

- **Escalabilidade:** O Firestore e as Firebase Cloud Functions, escalam automaticamente com o aumento de usuários, suportando períodos de alta demanda (RNF03).
- **Manutenibilidade:** A separação em camadas facilita a manutenção e atualização do código (RNF06).
- **Tempo Real:** A sincronização em tempo real do Firestore melhora a experiência do usuário, refletindo alterações instantaneamente (RNF01).
- **Segurança:** As Firestore Security Rules garantem acesso controlado aos dados, em conformidade com a LGPD (RNF04, RNF07).

Exemplo de Código

A seguir, um trecho de código que demonstra a interação entre a camada de apresentação (captura de dados via controladores) e a camada de dados (Firestore), com tratamento de erros para garantir robustez.

FIGURA 9 – EXEMPLO DE CÓDIGO

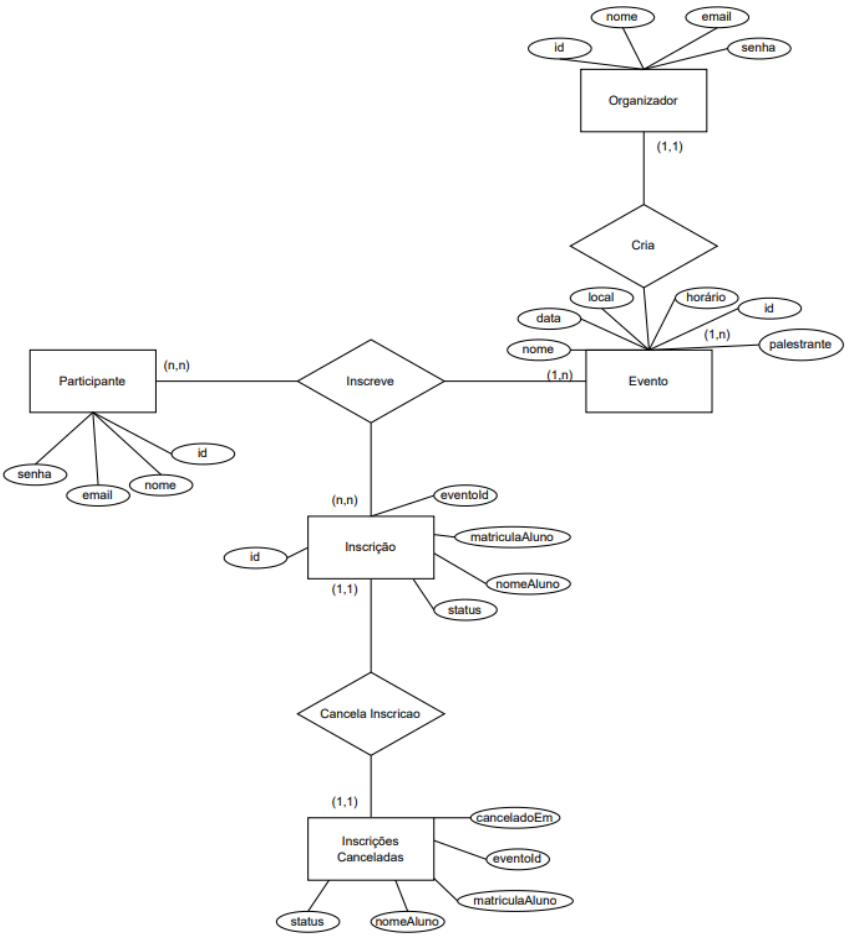
```
try {
  final docRef = firestore.collection("eventos").doc();
  final evento = {
    "id": docRef.id,
    "nome": nomeController.text,
    "data": dataController.text,
    "local": localController.text,
    "horario": horarioController.text,
    "descricao": descricaoController.text,
    "palestrante": palestranteController.text,
    "criado_em": FieldValue.serverTimestamp(),
  };
  await docRef.set(evento);
  mostrarMensagem("Evento cadastrado com sucesso!", Colors.green);
  // Limpar campos
  nomeController.clear();
  dataController.clear();
  localController.clear();
  horarioController.clear();
  descricaoController.clear();
  palestranteController.clear();
} catch (e) {
  mostrarMensagem("Erro ao cadastrar evento!", Colors.red);
  print("Erro ao cadastrar evento: $e");
}
```

2.4 CRIAÇÃO DO BANCO DE DADOS

O aplicativo Evex utiliza o Firestore, um banco de dados NoSQL do Firebase, para armazenar e gerenciar dados de eventos acadêmicos da PUC-GO. A estrutura do banco de dados foi projetada para atender aos requisitos funcionais (RF01-RF15) e não funcionais (RNF04, RNF07) do sistema, garantindo eficiência, segurança e escalabilidade. A estrutura foi modelada usando um Diagrama Entidade-Relacionamento (DER) adaptado ao modelo NoSQL do Firestore, que organiza dados em coleções e documentos.

Nele são descritas as entidades principais como: **Evento**, **Participante** e **Organizador**. A entidade associativa **Inscrição** representa o relacionamento muitos-para-muitos entre **Evento** e **Participante**. O atributo id está presente em todas as entidades como chave primária, garantindo a identificação única de cada registro.

FIGURA 10 – DIAGRAMA ENTIDADE-RELACIONAMENTO (DER)



Relacionamentos

- **Evento ↔ Participante (muitos-para-muitos) via inscrição:**
  - Um evento pode ter vários participantes, e um participante pode se inscrever em vários eventos (RF07).
  - Validações:
    - Não permitir inscrição se houver conflito de horário (RF08, RF09).
    - Não permitir inscrição se a capacidade máxima do evento for atingida (RF10, RF11).
- **Evento → Organizador (muitos-para-um):**
  - Cada evento é criado por um organizador (usuário com perfil “Professor”) (RF14).

- **Inscrição ↔ Inscrições canceladas (um-para-um)**
  - Cada inscrição pode ter no máximo um cancelamento associado, e cada registro de cancelamento está ligado a uma única inscrição, respeitando o prazo de cancelamento (RF11).

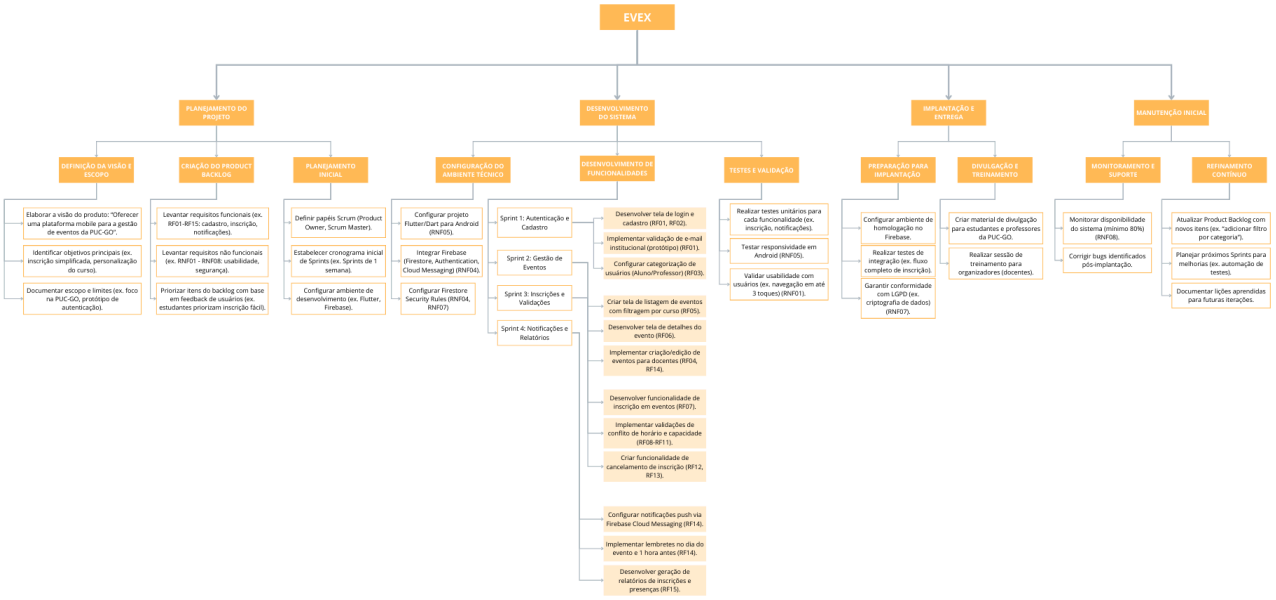
Regras de Segurança (Firestore Security Rules)

- Apenas usuários autenticados podem acessar dados (RNF04).
- Usuários com perfil “Professor” podem criar/editar eventos (RF14).
- Participantes só podem visualizar eventos e suas próprias inscrições (RF05, RF06).
- Dados sensíveis (ex. e-mail) são protegidos em conformidade com a LGPD (RNF07).

2.5 ESTRUTURA ANALÍTICA DO PROJETO (EAP)

A Estrutura Analítica do Projeto (EAP) organiza as atividades necessárias para o desenvolvimento do Evex em uma hierarquia com pelo menos três níveis, detalhando pacotes de trabalho e tarefas. O projeto segue a metodologia Scrum, com Sprints semanais para entregas incrementais. A EAP abrange desde o planejamento até a implantação e manutenção inicial do aplicativo.

FIGURA 11 – ESTRUTURA ANALÍTICA DO PROJETO (EAP)



2.6 CRONOGRAMA DE ENTREGAS

O cronograma de entregas do Evex organiza as atividades e marcos principais em um período de 11 semanas, de 20 de março de 2025 a 09 de junho de 2025. O projeto segue a metodologia Scrum, com Sprints semanais, garantindo entregas incrementais e adaptação contínua. O cronograma é apresentado em um Gantt Chart, que detalha as fases do projeto (Planejamento, Desenvolvimento, Implantação e Entrega), Manutenção Inicial, as atividades correspondentes, e os marcos importantes, como a conclusão de cada Sprint, a entrega do documento (02 de junho), e a apresentação do projeto (09 de junho).

Datas e Prazos

O cronograma foi ajustado para iniciar em 20/03/25, com a entrega do documento marcada para 02/06/25 e a apresentação do projeto para 09/06/25. Cada atividade tem uma duração estimada com base na complexidade e nos recursos disponíveis. As Sprints são organizadas para entregar incrementos funcionais (ex. autenticação, gestão de eventos) a cada semana, com revisões e ajustes contínuos.

FIGURA 12 – GRÁFICO GANTT

NÚMERO DA EAP	TÍTULO DA TAREFA	DURAÇÃO	INÍCIO	FIM
	Execução completa do projeto	74 dias	20/03/25	02/06/25
1	Planejamento	7 dias	20/03/25	26/03/25
1.1	Definição da visão e Escopo	3 dias	20/03/25	22/03/25
1.2	Criação do Product Backlog	3 dias	22/03/25	24/03/25
1.3	Planejamento Inicial	3 dias	24/03/25	26/03/25
2	Desenvolvimento do sistema	42 dias	27/03/25	07/05/25
2.1	Configuração do Ambiente Técnico	7 dias	27/03/25	02/04/25
2.2	Sprint 1: Autenticação e Cadastro	7 dias	03/04/25	09/04/25
2.3	Sprint 2: Gestão de Eventos	7 dias	10/04/25	16/04/25
2.4	Sprint 3: Inscrições e Validações	7 dias	17/04/25	23/04/25
2.5	Sprint 4: Notificações e Relatórios	7 dias	24/04/25	30/04/25
2.6	Teste e Validação	7 dias	01/05/25	07/05/25
3	Implantação e Entrega	7 dias	08/05/25	14/05/25
3.1	Preparação para implantação	3 dias	08/05/25	10/05/25
3.2	Divulgação e Treinamento	2 dias	13/05/25	14/05/25
4	Manutenção inicial	7 dias	15/05/25	21/05/25
4.1	Monitoramento e Suporte	4 dias	15/05/25	18/05/25
4.2	Refinamento continuo	3 dias	19/05/25	21/05/25
5	Entrega Final			
5.1	Entrega do Documento e código	1 dia	02/06/25	02/06/25
5.2	Apresentação do projeto	1 dia	09/06/25	02/06/25

Detalhamento das Fases e Marcos

Fase 1: Planejamento do Projeto (20 de março a 26 de março de 2025)

- Definição da Visão e Escopo (20 de março a 22 de março): Elaboração da visão do produto e identificação de objetivos e limites.
- Criação do Product Backlog (22 de março a 24 de março): Levantamento e priorização de requisitos (RF01-RF15, RNF01-RNF09).
- Planejamento Inicial (24 de março a 26 de março): Definição de papéis Scrum e configuração do ambiente.
- **Marco**: Product Backlog inicial concluído (26 de março).

Fase 2: Desenvolvimento do Sistema (27 de março a 7 de maio de 2025)

- Configuração do Ambiente Técnico (27 de março a 2 de abril): Integração com Firebase e configuração do Flutter.
- Sprint 1: Autenticação e Cadastro (3 de abril a 9 de abril): Implementação de RF01-RF03 (cadastro, login, categorização de usuários).

    - **Marco**: Incremento com autenticação funcional (9 de abril).

- Sprint 2: Gestão de Eventos (10 de abril a 16 de abril): Implementação de RF04-RF06, RF14 (listagem, detalhes, criação de eventos).
    - **Marco**: Incremento com gestão de eventos funcional (16 de abril).
- Sprint 3: Inscrições e Validações (17 de abril a 23 de abril): Implementação de RF07-RF13 (inscrições, validações, cancelamento).
    - **Marco**: Incremento com inscrições funcional (23 de abril).
- Sprint 4: Notificações e Relatórios (24 de abril a 30 de abril): Implementação de RF14-RF15 (notificações, relatórios).
    - **Marco**: Incremento com notificações e relatórios funcional (30 de abril).
- Testes e Validação (1 de maio a 7 de maio): Testes unitários, de responsividade (RNF05), e usabilidade (RNF01).
    - **Marco**: Aplicativo testado e validado (7 de maio).

Fase 3: Implantação e Entrega (8 de maio a 14 de maio de 2025)

- Preparação para Implantação (8 de maio a 10 de maio): Testes de integração e conformidade com LGPD (RNF07).
- Divulgação e Treinamento (13 de maio a 14 de maio): Divulgação e treinamento para usuários da PUC-GO.

Fase 4: Manutenção Inicial (15 de maio a 21 de maio de 2025)

- Monitoramento e Suporte (15 de maio a 18 de maio): Monitoramento de disponibilidade (RNF08) e resposta a feedbacks.
- Refinamento Contínuo (19 de maio a 21 de maio): Atualização do Product Backlog para futuras melhorias.

Fase 5: Entrega Final e Apresentação (2 de junho a 9 de junho de 2025)

- Entrega do Documento (2 de junho): Submissão do documento final do projeto.

- Apresentação do Projeto (9 de junho): Apresentação final do projeto para as partes interessadas.

2.7 DESCRIÇÃO DO SOFTWARE DESENVOLVIDO

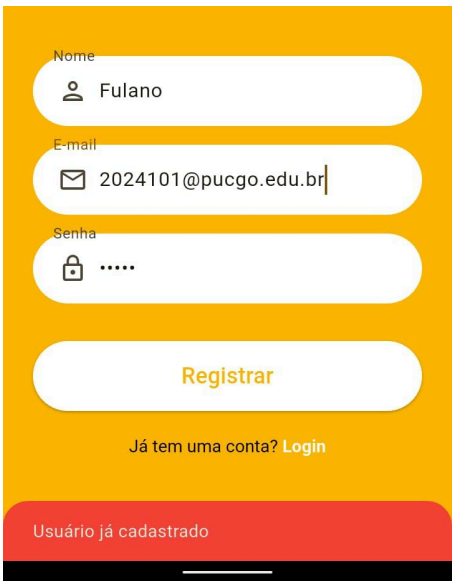
Visão Geral do Software

O Evex é um aplicativo mobile desenvolvido para centralizar a gestão de eventos acadêmicos da PUC-GO, como palestras e congressos, com o objetivo de facilitar a organização e participação da comunidade acadêmica. Inspirado na plataforma Sympla, o software promove engajamento ao melhorar a comunicação entre organizadores e participantes, oferecendo uma solução intuitiva para divulgação e inscrição em eventos. O protótipo foi projetado para atender aos requisitos funcionais (RF01-RF14) e não funcionais (RNF01-RNF08) do projeto, com foco exclusivo em eventos da PUC-GO e acesso restrito à comunidade acadêmica via e-mail institucional.

Principais Funcionalidades Implementadas

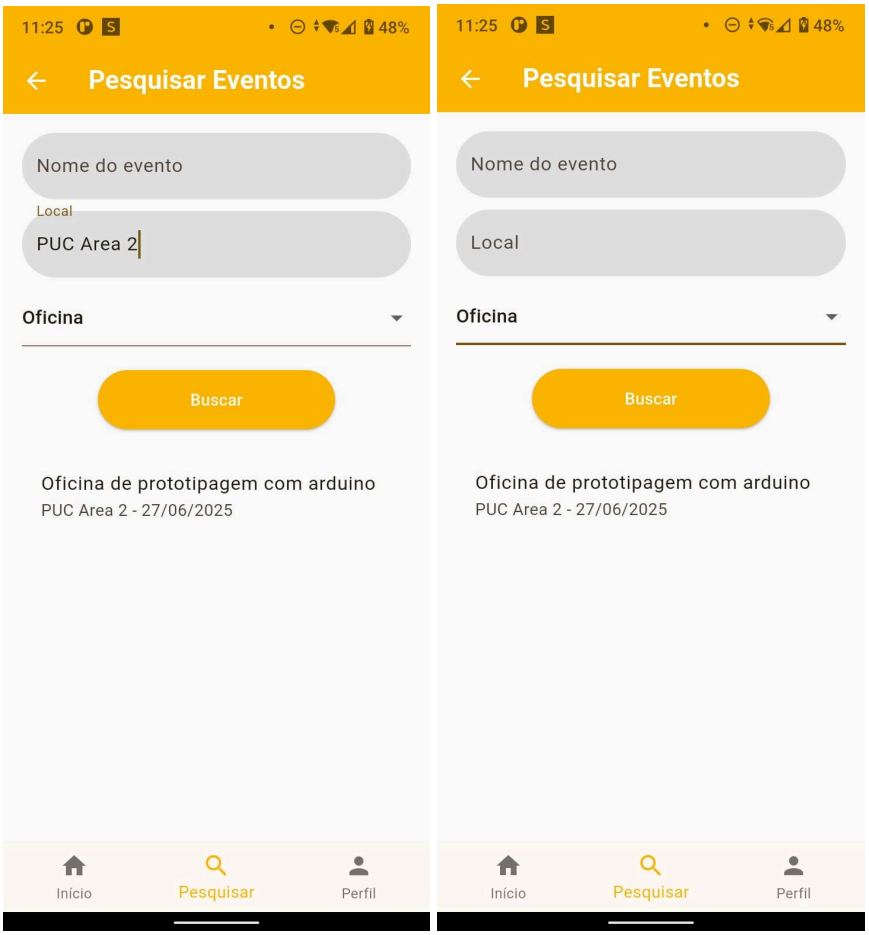
- **Autenticação e Gerenciamento de Usuários (RF01-RF03):** Login via Firebase Authentication com e-mail institucional, categorização de usuários (Aluno/Professor), e gerenciamento de perfil (ex. matrícula extraída do e-mail).

FIGURA 13 - FUNCIONALIDADES DE AUTENTICAÇÃO DO SISTEMA



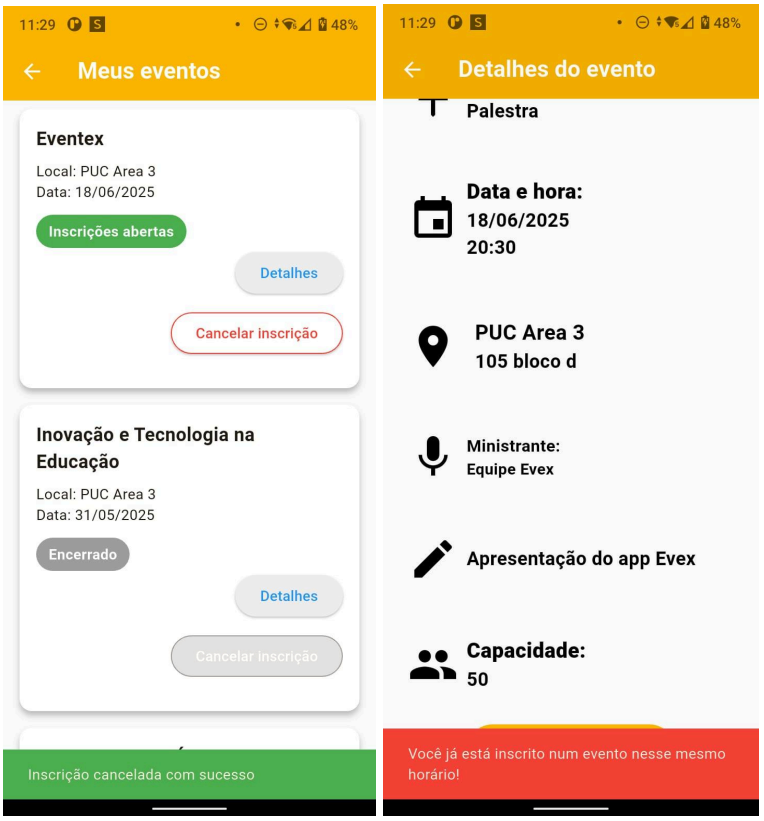
- **Gestão de Eventos (RF04-RF05, RF13):** Organizadores podem criar/editar eventos com informações como nome, tipo, modalidade (presencial/online), ministrante, local, capacidade, data e horário. Estudantes podem listar eventos, filtrar por curso, e visualizar detalhes.

FIGURA 14 - FILTRAGEM POR NOME, LOCAL E TIPO DE EVENTO



- **Inscrições e Validações (RF06-RF12):** Inscrição em eventos com validações de conflito de horário e capacidade máxima. Cancelamento de inscrição até 1 hora antes do evento, com notificação ao usuário.

FIGURA 15 - FUNCIONALIDADE DE CANCELAMENTO DE INSCRIÇÃO



- **Notificações e Relatórios (RF13-RF14):** Notificações push via Firebase Cloud Messaging com lembretes no dia do evento e 1 hora antes. Geração de relatórios de inscrições e presenças para organizadores.
- **API RESTful:** Uma API RESTful foi implementada usando Node Js para manipular requisições ao banco de dados de forma segura e escalável. A API gerencia operações como inscrição e validação de usuários, validação do domínio do e-mail, verificando se o usuário autenticado existe no banco de dados e retorna as informações que a aplicação necessita, protegida por regras de segurança do Firestore e autenticação via token.

Funcionamento do Software

O Evex opera em uma arquitetura 3-Tier (Apresentação, Lógica de Negócio, Dados). A camada de apresentação utiliza Flutter/Dart para uma interface Android. A camada de lógica de negócio é gerenciada por Firebase Cloud Functions, que implementa uma API RESTful para processar requisições, como **POST /eventos** para criar eventos e **GET /inscrições** para listar inscrições. A camada de dados utiliza o

Firestore para armazenamento NoSQL, com coleções como **Eventos**, **Participantes**, **Organizadores** e **Inscrição** (relacionamento muitos-para-muitos).

O fluxo de funcionamento do software é o seguinte:

- O usuário faz login com e-mail institucional.
- Estudantes visualizam eventos disponíveis e podem se inscrever, com validações automáticas.
- Organizadores podem criar eventos e acessar relatórios.
- Notificações push são enviadas para lembrar os participantes.

Limitações do Protótipo O protótipo do Evex apresenta as seguintes limitações:

- Foco exclusivo em eventos da PUC-GO.
- Acesso restrito à comunidade acadêmica, utilizando e-mail institucional.
- Ausência de processamento de pagamentos, integração com sistemas acadêmicos externos, versão desktop, streaming de eventos online e emissão de certificados.
- Suporte técnico limitado ao horário comercial.
- Capacidade inicial de 1000 usuários simultâneos (RNF03).

Desenvolvimento da API RESTful A API RESTful foi desenvolvida utilizando Node.js e está hospedada no Render, visando garantir segurança e escalabilidade na manipulação do banco de dados Firestore. Exemplos de endpoints incluem:

- **POST /identificarUsuario**: Este endpoint identifica se o formato de e-mail fornecido é válido, extrai a matrícula e retorna os dados do usuário (nome, tipo, matrícula).
- **POST /cadastrarUsuario**: Este endpoint verifica se o e-mail fornecido está no formato válido e identifica o tipo de usuário (professor pelo nome ou aluno pela matrícula) a partir do e-mail. Se as verificações forem bem-sucedidas, o usuário é cadastrado e adicionado à sua respectiva coleção no Firebase.

Desafios Enfrentados

- **Integração com Firebase:** Erros de Gradle e Java (versões 11, 17, 23) no Visual Studio Code e no Intellij devido a conflitos de dependências.

Resolvidos com tutoriais no Youtube e orientações do professor Rafael Leal, ajustando configurações de build e versões do JDK.

- **Alinhamento de Agendas:** Membros da equipe conciliam trabalho e estudos, causando atrasos. Superado com reuniões virtuais via Google Meet e comunicação diária pelo whatsapp.

Tecnologias Utilizadas

As principais tecnologias empregadas incluem:

- Flutter e Dart: Desenvolvimento para Android, garantindo portabilidade (RNF05) e usabilidade (RNF01).
- Firebase:
    - Firestore: Banco de dados NoSQL para armazenamento de eventos, participantes e inscrições.
    - Firebase Authentication: Autenticação segura com e-mail institucional.
    - Firebase Storage: Armazenamento de imagens de eventos.
    - NodeJs e Render: Implementação da API RESTful para lógica de negócio.
    - Flutter Local Notifications: Envio de notificações push.
- Java (versões 11, 17, 23): Utilizado para configurações específicas no ambiente de desenvolvimento (ex. ajustes de Gradle).
- Visual Studio Code e Intellij: Ambiente de desenvolvimento, com extensões para Flutter e Dart.
- Figma: Design de interfaces, garantindo navegação intuitiva.

Todas as ferramentas foram escolhidas por sua compatibilidade com o desenvolvimento ágil e pela capacidade de atender aos requisitos de escalabilidade (RNF03) e segurança (RNF04, RNF07).

2.8 ESPECIFICAÇÃO DE OBJETIVOS E REQUISITOS

Objetivos do Software

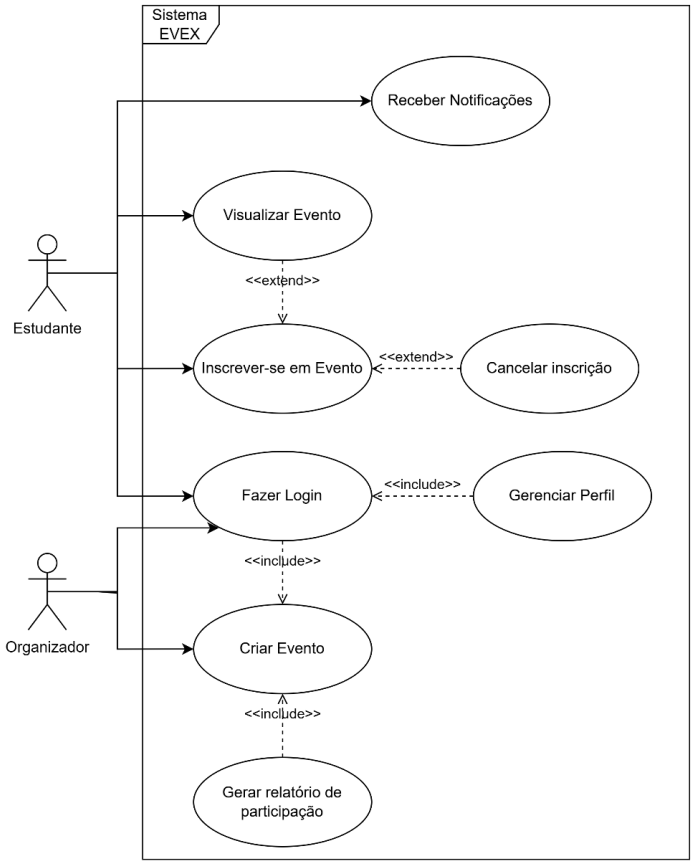
O Evex tem como objetivo centralizar a gestão de eventos acadêmicos da PUC-GO, promovendo engajamento entre estudantes e professores ao facilitar a organização, divulgação e participação em eventos como palestras e congressos. O software busca:

- Simplificar o processo de inscrição em eventos para estudantes.
- Oferecer uma interface intuitiva para organizadores (docentes) gerenciarem eventos e participantes.
- Melhorar a comunicação com notificações automáticas e relatórios de participação.
- Atender à comunidade acadêmica da PUC-GO com acesso restrito via e-mail institucional.

Casos de Uso e Atores

Os casos de uso descrevem as interações principais entre os atores e o sistema Evex, mapeando as suas funcionalidades. Dentro do sistema Evex temos dois atores: Estudante (Usuário com perfil “Aluno”, que se inscreve e participa de eventos) e Organizador (Usuário com perfil “Professor”, que cria e gerencia eventos.

FIGURA 16 – DIAGRAMA DE CASO DE USO



2.9 DESIGN DE SOFTWARE

Padrões de Design

O Evex utiliza os seguintes padrões de design para garantir modularidade, escalabilidade e manutenibilidade:

- **MVC (Model-View-Controller):**
    - **Model:** Representado pelo Firestore (coleções como Eventos, Participantes) e pela API RESTful (NodeJs), que gerencia os dados e a lógica de negócio.
    - **View:** Implementada com Flutter/Dart, exibindo a interface para usuários (telas de login, listagem de eventos, etc).
    - **Controller:** Gerenciado por Flutter (controladores Dart) e API Rest, que processam interações do usuário e atualizam o modelo.

    - **Justificativa:** Separação clara entre dados, interface e lógica, facilitando manutenção e testes.

- **Singleton:**
    - Usado para gerenciar instâncias únicas de serviços como autenticação (Firebase Auth) e notificações (Flutter Local Notifications) no Flutter.
    - **Justificativa:** Garante que apenas uma instância do serviço seja criada, economizando recursos e evitando conflitos.
- **Factory:**
    - Utilizado para criar objetos de eventos e usuários com base no perfil (Aluno/Professor) no lado do cliente (Flutter).
    - **Justificativa:** Simplifica a criação de objetos com configurações específicas, melhorando a extensibilidade.

Decisões de Design

- **Arquitetura 3-Tier:** Adotada para separar Apresentação (Flutter), Lógica de negócio (NodeJs) e Dados (Firestore), garantindo escalabilidade (RNF03) e manutenibilidade.
- **NoSQL com Firestore:** Escolhido por sua flexibilidade para adicionar campos dinâmicos (ex. novas categorias de eventos) e suporte a sincronização em tempo real (RNF06).
- **API RESTful Serverless:** Implementada com NodeJs para manipulação segura do banco de dados, usando autenticação por token e regras de segurança do Firestore (RNF04, RNF07).
- **Flutter para Android:** Escolhido para garantir compatibilidade com o Android (RNF05).
- **Prototipagem com Figma:** Usado para validar fluxos e design antes da implementação, assegurando usabilidade (RNF01).

Essas decisões priorizam a qualidade do software (ex. usabilidade, segurança) e a praticidade de manutenção, permitindo futuras expansões (ex. novas funcionalidades ou suporte a mais usuários).

2.10 CICLO DE VIDA DO SOFTWARE

Metodologia Adotada

O Evex foi desenvolvido utilizando a metodologia Scrum, uma abordagem ágil que enfatiza entregas incrementais e adaptação a mudanças. O Scrum foi escolhido por sua adequação a projetos com requisitos dinâmicos (ex. feedback contínuo de estudantes e organizadores) e pela necessidade de entregas rápidas dentro de um cronograma de 11 semanas (20/03/25 a 09/06/25).

Etapas do Ciclo de Vida

O ciclo de vida foi estruturado em Sprints semanais, com as seguintes fases:

1. **Planejamento (20/03/25 a 26/03/25):**
    - Atividades: Definição do tema (“centralizar eventos acadêmicos da PUC-GO”), criação do Product Backlog com itens como “inscrição em eventos” e “notificações push”, e priorização com base em feedback de usuários (ex. estudantes priorizam inscrição fácil).
    - Artefatos: Visão do produto, Product Backlog inicial.
2. **Planejamento do Sprint:**
    - Atividades: Reunião para definir o objetivo (ex. implementar autenticação) e dividir as tarefas (ex. criar tela de login, integrar Firebase Auth).
    - Artefatos: Sprint Backlog com tarefas detalhadas.
    - Exemplo: Sprint 1 (03/04/25 a 09/04/25) teve como objetivo “permitir login com e-mail institucional”.
3. **Execução do Sprint (Sprints de 03/04/25 a 30/04/25):**

- Atividades: Desenvolvimento incremental (ex. codificação de telas em Flutter, integração com API RESTful), Daily Scrum (15 minutos diários) para alinhamento e resolução de impedimentos.
- Exemplo: Durante o Sprint 2, a Camila falou no Daily Scrum: “Terminei a tela de listagem de eventos; vou começar a fazer a filtragem dos eventos”.

4. Revisão do Sprint (Fim de cada Sprint):

- Atividades: Sprint Review para demonstrar incrementos (ex. tela de inscrição funcional).
- Artefatos: Incremento do Produto (ex. versão com autenticação utilizável).
- Exemplo: No Sprint 3, o feedback foi adicionar validação de capacidade máxima.

5. Retrospectiva do Sprint (Fim de Cada Sprint):

- Atividades: Sprint Retrospective (45 minutos) para refletir sobre o que funcionou e o que melhorar (ex. atrasos na integração com Firebase).
- Exemplo: Ação definida: "Automatizar testes de interface no próximo Sprint."

6. Testes e Validação (01/05/25 a 07/05/25):

- Atividades: Testes unitários, de responsividade (Android), e usabilidade (navegação em até 3 toques).
- Exemplo: Validação confirmou que o tempo de resposta foi ≤ 3 segundos (RNF06).

7. Implantação e Entrega (08/05/25 a 14/05/25):

- Atividades: Testes de integração e treinamento para usuários da PUC-GO.

8. Manutenção Inicial (15/05/25 a 21/05/25):

- Atividades: Monitoramento de disponibilidade (80% mínimo, RNF08), resposta a feedbacks, e refinamento do Product Backlog.

- Exemplo: Giovanna sugeriu "adicionar filtro por categoria".

9. Entrega Final e Apresentação (02/06/25 a 09/06/25):

- Atividades: Entrega do documento final (02/06/25) e apresentação do projeto (09/06/25).
- Exemplo: Documento entregue com 100% de conclusão; apresentação pendente (em 09/06/25).

3 CONCLUSÃO

O projeto Evex buscou alcançar os objetivos propostos de centralizar a gestão de eventos acadêmicos da PUC-GO e facilitar a organização e participação da comunidade acadêmica. A implementação de funcionalidades como autenticação segura, gestão de eventos, inscrições com validações, notificações automáticas e relatórios de participação, aliada à API RESTful desenvolvida com Node.js, demonstrou uma solução prática e escalável para atender às necessidades de estudantes e professores. Apesar de desafios como a integração com o Firebase, implementação da API e o próprio tempo, o protótipo atendeu aos requisitos de usabilidade e segurança.

Os resultados indicam que o uso de tecnologias como Flutter, Dart e Firebase, combinado ao ciclo de vida Scrum, permitiu entregas incrementais e adaptação contínua, promovendo engajamento ao melhorar a comunicação entre os atores envolvidos. Embora o protótipo apresente limitações, os avanços observados reforçam a viabilidade de expandir o sistema para integrar sistemas acadêmicos externos e suportar mais funcionalidades, alinhando-se à proposta inicial de otimizar a gestão de eventos acadêmicos da instituição.

REFERÊNCIAS

ASANA. *Work breakdown structure*. Disponível em: https://asana.com/pt/resources/work-breakdown-structure. Acesso em: 20 maio 2025.

CANALTECH. *Como citar ou referenciar imagens e fotos em ABNT*. Disponível em: https://canaltech.com.br/educacao/como-citar-ou-referenciar-imagens-e-fotos-em-abnt/. Acesso em: 21 maio 2025.

LEAL, Rafael. Aula: *Acesso a APIs REST com Flutter*. [S.l.]: Microsoft Teams, 2025.

DRAW.IO. *Draw.io*. Disponível em: https://draw.io. Acesso em: 22 maio 2025.

FIGMA. *Figma.io*. Disponível em: https://figma.io. Acesso em: 23 maio 2025.

GOOGLE. *Firebase database*. Disponível em: https://firebase.google.com/docs/database?hl=pt-br. Acesso em: 24 maio 2025.

GOOGLE. *Firebase authentication*. Disponível em: https://firebase.google.com/docs/auth?hl=pt-br. Acesso em: 25 maio 2025.

LEAL, Rafael. *Aula: Programação Orientada a Objetos com Flutter*. [S.l.]: Microsoft Teams, 2025.

LEAL, Rafael. *App: Gerenciador de Tarefas Aprimorado com Firebase*. [S.l.]: Microsoft Teams, 2025.

LEAL, Rafael. *Exemplo de um aplicativo Flutter básico que utiliza Firebase Authentication e Firestore*. [S.l.]: Microsoft Teams, 2025.

LEAL, Rafael. *Firebase – Introdução*. [S.l.]: Microsoft Teams, 2025.

LEAL, Rafael. *Flutter com Firebase*. [S.l.]: Microsoft Teams, 2025.

IBM. *Three-tier architecture*. Disponível em: https://www.ibm.com/br-pt/topics/three-tier-architecture. Acesso em: 26 maio 2025.

INOVELI LICITAÇÕES. *O que é cronograma de entregas?*. Disponível em: https://inovelicitacoes.com.br/glossario/o-que-e-cronograma-de-entregas/. Acesso em: 20 maio 2025.

LUCIDCHART. *O que é diagrama entidade-relacionamento*. Disponível em: https://www.lucidchart.com/pages/pt/o-que-e-diagrama-entidade-relacionamento. Acesso em: 21 maio 2025.

LUCIDCHART. *Diagrama de componentes UML*. Disponível em: https://www.lucidchart.com/pages/pt/diagrama-de-componentes-uml. Acesso em: 22 maio 2025.

QUERO BOLSA. *Requisitos funcionais e não funcionais*. Disponível em: https://querobolsa.com.br/revista/requisitos-funcionais-e-nao-funcionais. Acesso em: 23 maio 2025.

REDEX. *Firebase: descubra para que serve, como funciona e como usar*. Disponível em: https://www.remessaonline.com.br/blog/firebase-descubra-para-que-serve-como-funciona-e-como-usar/. Acesso em: 24 maio 2025.

ASANA. *Project scope*. Disponível em: https://asana.com/pt/resources/project-scope. Acesso em: 25 maio 2025.

ATLASSIAN. *Guia do Scrum*. Disponível em: https://www.atlassian.com/br/agile/scrum. Acesso em: 26 maio 2025.

ATLASSIAN. *O que é um gráfico de Gantt?*. Disponível em: https://www.atlassian.com/br/agile/project-management/gantt-chart. Acesso em: 20 maio 2025.