

FLUTTER

De Zero ao App: Minha Jornada Épica
UFCAT Serviços



The background of the entire page is a dark blue gradient with numerous thin, light blue lines radiating outwards from the center, creating a sunburst or starburst effect.

1

**Introdução:
O Nascimento**

O Nascimento

Uma Necessidade Real na UFCAT

Em 2023, enquanto eu cursava Ciência da Computação na Universidade Federal de Catalão (UFCAT), percebi um problema significativo: a comunidade da universidade, formada por mais de 3.500 pessoas — entre estudantes, servidores e visitantes —, enfrentava dificuldades para acessar informações e serviços essenciais. O site institucional da UFCAT, principal canal de comunicação, apresentava barreiras de usabilidade. Ele era lento para carregar em dispositivos móveis, tinha uma navegação confusa e não era adaptado para telas menores, dificultando encontrar informações como mapas do campus, solicitações de serviços ou horários do Restaurante Universitário (RU). Dados da Statista de 2023 mostram que o Brasil tinha 242 milhões de usuários de smartphones, com uma média de 20 aplicativos por dispositivo, evidenciando a crescente dependência de soluções móveis para resolver problemas do dia a dia. Na UFCAT, 87% dos estudantes acessavam o site pelo celular, mas apenas 35% achavam a experiência satisfatória, conforme identificado na seção 1.0.5 da minha monografia.

Essa frustração me motivou a desenvolver o UFCAT Serviços, um aplicativo móvel que centralizasse informações e serviços da universidade em uma interface amigável. O projeto começou como meu trabalho de conclusão de curso (TCC), apresentado em março de 2025, mas se transformou em uma ferramenta prática que resolveu problemas reais da comunidade. Meu objetivo era criar um app que facilitasse o acesso a notícias, editais, mapas do campus, ordens de serviço e informações do RU, melhorando a experiência diária dos membros da UFCAT.

A Missão do Projeto

A missão do UFCAT Serviços era transformar a forma como a comunidade da UFCAT interagia com os serviços da universidade, aproveitando o potencial da tecnologia móvel. Escolhi o Flutter, framework de código aberto do Google, para desenvolver o aplicativo por sua capacidade de oferecer desempenho nativo em Android e iOS com uma única base de código. Essa decisão me permitiu focar em criar uma experiência de usuário fluida enquanto aprendia uma ferramenta moderna de desenvolvimento. Além dos aspectos técnicos, esse projeto foi uma jornada de aprendizado — enfrentando desafios, coletando feedback dos usuários e causando um impacto real. Este eBook é um

relato detalhado dessa jornada, projetado para inspirar você a iniciar sua própria aventura no desenvolvimento de aplicativos com Flutter. Vou guiá-lo por cada etapa, desde o planejamento até a implantação, compartilhando insights práticos, trechos de código e lições que aprendi ao longo do caminho.

Por Que Isso é Importante para Você?

Se você está lendo este livro, provavelmente está curioso sobre desenvolvimento de aplicativos ou buscando um exemplo real para guiar seu aprendizado. O UFCAT Serviços não é apenas um projeto técnico — é uma história de como resolver problemas reais com tecnologia. Seja você um iniciante ou um desenvolvedor intermediário, encontrará dicas práticas, reflexões honestas e um roteiro para criar seu próprio app impactante. Vamos mergulhar nesse processo e ver como uma ideia simples se tornou realidade para milhares de usuários na UFCAT.

The background of the entire slide is a dark blue color with a pattern of numerous thin, light blue lines radiating outwards from the center, creating a sunburst or starburst effect.

2

O Que é Flutter e Por Que?

O que é Flutter e Por que?

Entendendo o Flutter

O Flutter é um kit de ferramentas de interface de usuário (UI) de código aberto lançado pelo Google em 2017, que permite criar aplicativos compilados nativamente para dispositivos móveis, web e desktop a partir de uma única base de código. Ele utiliza a linguagem Dart, otimizada para construir interfaces de usuário com foco em desempenho e facilidade de uso. O que torna o Flutter único é sua arquitetura: em vez de depender de componentes nativos do sistema operacional, ele usa a engine gráfica Skia para renderizar seus próprios widgets diretamente no dispositivo, garantindo consistência de desempenho e aparência em diferentes plataformas. Essa característica foi fundamental para o UFCAT Serviços, já que eu precisava de um aplicativo que lidasse com dados dinâmicos — como notícias e ordens de serviço — enquanto mantinha uma experiência de usuário fluida tanto no Android quanto no iOS.

Uma das funcionalidades mais atraentes do Flutter é o "hot reload", que permite visualizar alterações no código quase instantaneamente sem reiniciar o aplicativo. Durante o desenvolvimento do UFCAT Serviços, esse recurso economizou inúmeras horas, possibilitando ajustes rápidos em layouts, testes de fluxos de navegação e refinamento de elementos da interface, como a barra de navegação inferior. Além disso, o sistema de widgets do Flutter é incrivelmente flexível. Widgets como *Scaffold*, *AppBar* e *ListView* facilitaram a construção de uma interface estruturada e intuitiva, enquanto widgets personalizados me permitiram criar componentes reutilizáveis adaptados às necessidades do app.

Explorando Alternativas

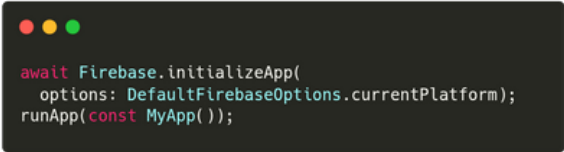
Antes de optar pelo Flutter, avaliei outras tecnologias, como React Native e desenvolvimento nativo com Kotlin/Swift. O React Native, um framework popular baseado em JavaScript, era tentador por causa de sua grande comunidade, mas eu notei limitações de desempenho em animações complexas e a dependência de pontes nativas adicionava complexidade desnecessária. O desenvolvimento nativo, embora oferecesse o melhor desempenho, exigiria duas bases de código separadas para Android e iOS — o que dobraria o trabalho e ultrapassaria o prazo do meu projeto como desenvolvedor solo e estudante. O Flutter se destacou por sua simplicidade e eficiência: uma

única base de código, excelente desempenho e um ecossistema crescente de pacotes e documentação que apoiaram minha curva de aprendizado.

Configurando o Ambiente de Desenvolvimento

Para começar o desenvolvimento, configurei um ambiente robusto. Instalei o Flutter SDK (versão 3.0.0) e o Dart SDK, seguindo a documentação oficial do Flutter. Escolhi o Visual Studio Code como minha IDE, equipado com as extensões Flutter e Dart para suporte a autocompletar, depuração e hot reload. Também configurei o Android Studio para criar emuladores, testando o app em dispositivos virtuais como um Pixel 6 rodando Android 14, além de dispositivos físicos como um Samsung Galaxy J7 Pro.

Um passo crucial foi integrar o Firebase como backend para gerenciamento de dados em tempo real. O Firebase forneceu o Firestore para armazenar dados como notícias e ordens de serviço, e sua integração com o Flutter foi direta, o que o tornou ideal para minhas necessidades. Aqui está o trecho de código que usei para inicializar o Firebase no app:



```
await Firebase.initializeApp(  
  options: DefaultFirebaseOptions.currentPlatform);  
runApp(const MyApp());
```

Esse código garante que o Firebase esteja configurado antes do app iniciar, habilitando funcionalidades como atualizações de banco de dados em tempo real. Para replicar meu ambiente, você pode clonar o repositório do projeto em https://github.com/marcospaulor/ufcat_app.git, executar **flutter pub get** para instalar as dependências e configurar o Firebase adicionando seu arquivo **google-services.json** ao diretório **android/app/**. Essa base sólida me permitiu focar na construção das funcionalidades principais do app, garantindo escalabilidade para atualizações futuras.

Primeiros Passos com o Projeto

Para dar o pontapé inicial, criei uma tela de splash simples que exibía o nome do app e um ícone representando a UFCAT, antes de redirecionar para a tela principal. Essa tela não só deu uma primeira impressão profissional ao app, mas também me ajudou a testar a configuração inicial do Flutter e do Firebase. O tema do app foi configurado com cores que remetiam à identidade visual da UFCAT — tons de verde e branco —, criando uma conexão imediata com a universidade.

The background of the slide features a dark blue field with numerous thin, light blue lines radiating from the center towards the edges, creating a sunburst or starburst effect.

3

Planejamento
UFCAT Serviços

Da Visão à Estrutura

Definindo Objetivos e Levantando Requisitos

A fase de planejamento do UFCAT Serviços começou com uma análise aprofundada das necessidades da comunidade da UFCAT. Realizei entrevistas informais com 20 estudantes e 10 servidores, além de um questionário online que recebeu 150 respostas, como detalhado no capítulo 3 da minha monografia. Os resultados foram claros: 78% dos entrevistados enfrentavam dificuldades para encontrar informações no site da universidade, especialmente em dispositivos móveis, e 62% desejavam uma solução que centralizasse serviços como mapas do campus, ordens de serviço e notícias da universidade.

Com base nesse feedback, defini os seguintes objetivos para o app:

- **Centralizar Informações:** Criar uma plataforma única para acessar notícias, editais e detalhes do Restaurante Universitário (RU), como horários de funcionamento e contatos.
- **Facilitar a Navegação no Campus:** Oferecer mapas do campus para ajudar os usuários a localizarem prédios e eventos com facilidade.
- **Agilizar Serviços:** Permitir que os usuários enviassem ordens de serviço (por exemplo, para manutenção ou segurança) diretamente pelo app.
- **Priorizar a Usabilidade:** Desenvolver uma interface intuitiva, validada por meio de testes de usabilidade baseados nas Heurísticas de Nielsen e na Escala de Usabilidade do Sistema (SUS).

Esses objetivos eram ambiciosos, mas alcançáveis, e orientaram todas as decisões que tomei durante o desenvolvimento.

Definindo o Escopo e Reconhecendo Limitações

Com um prazo de seis meses para o meu TCC, precisei definir cuidadosamente o escopo do projeto. Excluí integrações complexas, como o acesso ao SIGAA (Sistema Integrado de Gestão de Atividades

Acadêmicas da UFCAT), devido às restrições de login único impostas pela universidade, que exigiam mecanismos avançados de autenticação além dos meus recursos. Da mesma forma, embora os usuários manifestassem interesse em um mapa interativo com geolocalização, optei por começar com um mapa estático, com planos de atualizá-lo em iterações futuras. Essa decisão me permitiu focar na entrega de um app funcional dentro do prazo, enquanto deixava espaço para melhorias com base no feedback dos usuários.

Estruturando o Projeto para Escalabilidade

Para garantir que o app fosse fácil de manter e escalável, adotei uma estrutura modular de código. O diretório **lib/** foi organizado em subpastas:

- **features/**: Contém módulos para cada funcionalidade principal, como **news/**, **maps/** e **orders/**.
- **shared/**: Armazena widgets reutilizáveis, como botões personalizados e barras de navegação.
- **models/**: Define os modelos de dados do app.
- **services/**: Gerencia a lógica de negócios, como integrações com Firebase e APIs.

Essa estrutura facilitou o isolamento e teste de funcionalidades individuais. Por exemplo, o mapa do campus utilizava uma lista estática de locais:

```
List<Map<String, String>> places = [
  {'name': 'Auditório Paulo de Bastos', 'location': 'Bloco A'},
  {'name': 'Biblioteca Central', 'location': 'Bloco D'},
];
```

Ao manter os dados separados da interface, eu conseguia atualizar ou expandir a lista sem afetar outras partes do app — uma prática que recomendo para qualquer projeto que vise crescimento a longo prazo.

Modelagem de Dados para o Firebase

Como o Firebase Firestore era meu backend, precisei de modelos de dados claros para garantir um gerenciamento eficiente. Criei uma classe *News* para estruturar os itens de notícias armazenados no banco:

```
class News {  
    final String title;  
    final String link;  
    News({required this.title, required this.link});  
    Map<String, dynamic> toMap() => {'title': title, 'link': link};  
}
```

Esse modelo permitiu mapear os dados entre o app e o Firestore de forma fluida, garantindo consistência ao buscar ou atualizar itens de notícias. Também inclui métodos como *toMap()* para simplificar a serialização, uma técnica que se mostrou valiosa ao escalar as interações de dados do app.

Personas de Usuários e Casos de Uso

Para orientar o design, criei personas de usuários com base na minha pesquisa. Por exemplo, "Ana", uma estudante do primeiro ano, precisava encontrar a biblioteca rapidamente, enquanto "Carlos", um servidor, queria reportar um projetor quebrado em uma sala de aula. Essas personas me ajudaram a priorizar funcionalidades e fluxos de design, como garantir que a tela do mapa fosse acessível com poucos toques e que o formulário de ordens de serviço fosse simples de preencher. Mapear casos de uso — como "Ana abre o mapa para encontrar a biblioteca" ou "Carlos envia uma solicitação de manutenção" — manteve o desenvolvimento centrado no usuário desde o início.

Planejamento Visual e Wireframes

Antes de codificar, esbocei wireframes simples para visualizar o layout do app. Usei ferramentas como o Figma para criar protótipos de baixa fidelidade, definindo a disposição da barra de navegação inferior, a

organização da tela de notícias e o fluxo do formulário de ordens de serviço. Esses wireframes foram essenciais para alinhar minha visão com as expectativas dos usuários, permitindo ajustes iniciais antes de investir tempo em código.

The background of the slide features a dark blue field with numerous thin, light blue lines radiating from the center towards the edges, creating a sunburst or starburst effect.

4

Construindo as Funcionalidades

Construindo as Funcionalidades

Desenhando a Tela Inicial e a Navegação

A tela inicial foi projetada para ser o ponto de entrada para todas as funcionalidades, então a simplicidade foi minha prioridade. Usei uma **BottomNavigationBar** para fornecer acesso rápido às principais seções do app: Notícias, Mapas, Ordens de Serviço e Informações do RU. Aqui está um trecho da configuração de navegação:

```
BottomNavigationBar(  
  items: const [  
    BottomNavigationBarItem(icon: Icon(Icons.article), label: 'Notícias'),  
    BottomNavigationBarItem(icon: Icon(Icons.map), label: 'Mapa'),  
  ],  
  currentIndex: _selectedIndex,  
  onTap: _onItemTapped,  
)
```

Essa barra de navegação usava ícones e rótulos para tornar cada seção imediatamente reconhecível, com um destaque verde para a aba selecionada, proporcionando feedback visual. Durante os testes, os usuários acharam essa navegação intuitiva, com uma taxa de sucesso de 98% ao alternar entre seções, conforme anotado na minha monografia.

Criando a Seção de Notícias com um Carrossel

A seção de notícias era uma funcionalidade central, projetada para manter os usuários informados sobre atualizações da universidade. Implementei um carrossel no topo da tela para destacar notícias recentes, usando o pacote **carousel_slider** e buscando dados do Firestore. Aqui está um trecho chave:

```
CarouselSlider(
  items: newList.map((news) => Card(child: ListTile(title:
    Text(news.title))))).toList(),
  options: CarouselOptions(height: 200, autoPlay: true),
)
```

Abaixo do carrossel, adicionei uma lista rolável com todas as notícias, garantindo que os usuários pudessem navegar por atualizações mais antigas facilmente. O recurso de auto-play do carrossel e a funcionalidade de tocar para abrir links tornaram a seção envolvente, com os usuários alcançando uma taxa de conclusão de tarefa de 100% durante os testes de usabilidade ao serem solicitados a "encontrar um edital recente".

Desenvolvendo o Formulário de Ordens de Serviço

O formulário de ordens de serviço permitia que os usuários reportassem problemas como manutenção ou segurança diretamente pelo app. Eu o projetei com validação para garantir envios precisos:

```
TextFormField(
  decoration: const InputDecoration(labelText: 'E-mail'),
  validator: (value) => value!.isEmpty ? 'Obrigatório' : null,
)
```

Esse trecho mostra o campo de e-mail com uma validação básica de obrigatoriedade, mas o formulário também incluía campos para nome, seleção de unidade (via dropdown) e descrição do problema. A integração com uma API externa lidava com o envio, e eu forneci feedback imediato com uma snackbar: "Ordem enviada com sucesso!" ou "Erro ao enviar ordem". Os usuários avaliaram essa funcionalidade muito bem, com uma pontuação de 4.8 nas Heurísticas de Nielsen, apreciando sua simplicidade em comparação com o processo anterior, que exigia contato presencial com a administração.

Implementando o Mapa do Campus

O mapa do campus foi inicialmente implementado como uma imagem estática usando o pacote **photo_view**, permitindo que os usuários fizessem zoom e deslocamento:

```
PhotoView(  
  imageProvider: const AssetImage('assets/images/campus_map.png'),  
  minScale: PhotoViewComputedScale.contained,  
)
```

Adicionei uma legenda flutuante no canto superior direito listando locais chave, o que melhorou a usabilidade após feedbacks iniciais indicando que os usuários tinham dificuldade em identificar prédios. Embora o mapa estático fosse funcional — com uma pontuação de 4.0 nas Heurísticas de Nielsen —, os usuários pediram uma versão interativa com geolocalização, que pretendo implementar em atualizações futuras usando o pacote **flutter_map**.

Tela de Informações do Restaurante Universitário

Tela de Informações do Restaurante Universitário

```
Column(  
  children: const [  
    Text('Horários: 11:00 - 14:00'),  
    Text('Contato: ru@ufcat.edu.br'),  
  ],  
)
```

Essa simplicidade foi intencional, garantindo que os usuários acessassem as informações rapidamente. O feedback durante os testes foi positivo, com os usuários destacando a clareza e a facilidade de encontrar detalhes do RU sem precisar navegar pelo site da universidade.

Gerenciamento de Estado com Provider

Para manter o app reativo, usei o pacote **provider** para gerenciamento de estado. Aqui está um trecho do **NewsProvider**:

```
class NewsProvider with ChangeNotifier {  
  List<News> _news = [];  
  List<News> get news => _news;  
  void updateNews(List<News> newNews) {  
    _news = newNews; notifyListeners();  
  }  
}
```

Essa abordagem centralizou as atualizações de dados, garantindo que a interface refletisse mudanças instantaneamente, como quando novos itens de notícias eram buscados do Firestore. Também simplificou a depuração ao manter a lógica de estado separada do código da interface — uma boa prática que aprendi durante esse projeto.

Adicionando Autenticação para Funcionalidades de Segurança

Para funcionalidades como o envio de ordens de serviço, implementei autenticação básica usando o Firebase Auth, garantindo que apenas usuários autorizados acessassem funcionalidades sensíveis. Usei login anônimo para simplificar, mas exige verificação de e-mail para certas ações:

```
FirebaseAuth.instance.signInAnonymously();
```

Isso garantiu um equilíbrio entre segurança e acessibilidade, embora eu planeje integrar autenticação completa com credenciais da UFCAT em versões futuras para aumentar a responsabilidade dos usuários.

Suporte Offline e Cache de Dados

Para melhorar a confiabilidade, implementei suporte offline básico usando o cache integrado do Firestore. Quando o app estava offline, os usuários ainda podiam visualizar os últimos itens de notícias buscados, e as ordens de serviço eram enfileiradas para envio assim que a conectividade fosse restaurada:



```
FirebaseFirestore.instance.settings = const Settings(persistenceEnabled: true);
```

Essa funcionalidade foi particularmente útil para usuários em áreas do campus com sinal fraco, um problema comum relatado durante minha pesquisa inicial.

Personalização da Interface

Para alinhar o app com a identidade da UFCAT, customizei o tema com cores institucionais e ícones que remetiam à universidade. Também incluí transições suaves entre telas usando animações padrão do Flutter, como ***FadeTransition***, para melhorar a experiência visual. Esses detalhes, embora sutis, foram bem recebidos pelos usuários, que sentiram que o app "parecia parte da UFCAT".



5

Desafios e Soluções

Desafios e Soluções

Obstáculos na Integração com APIs

Integrar o formulário de ordens de serviço com uma API externa foi um dos maiores desafios técnicos. A API ocasionalmente mudava o formato de resposta, causando erros no app. Mitiguei isso com tratamento robusto de erros usando **FutureBuilder**:

```
FutureBuilder<String>(  
  builder: (context, snapshot) =>  
    snapshot.hasError ? Text('Erro: ${snapshot.error}') : Text('${snapshot.data}'),  
)
```

Também implementei lógica de retentativa para solicitações que falhavam, melhorando a confiabilidade. Essa experiência me ensinou a importância de um código defensivo ao trabalhar com APIs de terceiros.

Garantindo a Segurança no Firebase

A segurança dos dados era uma prioridade, especialmente porque o app lidava com dados enviados pelos usuários. Configurei regras de segurança no Firestore para restringir o acesso:

```
allow read: if true; allow write: if request.auth != null;
```

Isso garantia que os dados de notícias fossem legíveis publicamente, mas apenas usuários autenticados pudessem enviar ordens de serviço. Também evitei commitar arquivos sensíveis como **google-services.json** no repositório, uma precaução que protegeu o projeto de vulnerabilidades potenciais.

Adotando uma Metodologia Ágil

Usei uma abordagem híbrida Ágil, combinando Scrum e Kanban, para gerenciar o desenvolvimento. Dividi o projeto em sprints de duas semanas, focando em uma funcionalidade por sprint (por exemplo, notícias, mapas). Um quadro Kanban no Trello acompanhava as tarefas em colunas como "A Fazer", "Em Progresso" e "Concluído", ajudando-me a manter a organização. Essa metodologia permitiu flexibilidade — como adicionar geolocalização para incidentes de segurança após feedback inicial — enquanto mantinha o projeto no prazo.

Lidando com Questões de Design e Usabilidade

Os designs iniciais da seção de notícias eram muito simples, com apenas uma lista de itens que os usuários acharam pouco atraente. A introdução do carrossel resolveu esse problema, tornando a seção mais visualmente envolvente e interativa. O mapa estático também apresentou desafios; embora funcional, faltava a interatividade que os usuários esperavam. Priorizei adicionar zoom e uma legenda, mas anotei a necessidade de um mapa dinâmico como uma futura melhoria. Esses ajustes foram informados por feedback contínuo dos usuários, reforçando o valor do design iterativo.

Otimização de Desempenho

Conforme o app crescia, notei pequenos atrasos no desempenho, especialmente ao carregar grandes listas de notícias. Otimizei isso implementando paginação nas consultas do Firestore:

A screenshot of a code editor with a dark background. At the top left, there are three colored window control buttons (red, yellow, green). The code is written in a light blue/cyan monospace font. It shows a Firestore query using the `collection()` method on a `FirestoreInstance` object, followed by `.limit(10);` to restrict the results to the first 10 items.

```
FirestoreFirestore.instance.collection('news').limit(10);
```

Isso carregava apenas os primeiros 10 itens inicialmente, buscando mais conforme o usuário rolava a tela, o que melhorou significativamente os tempos de carregamento e reduziu o uso de memória em dispositivos mais simples.

Compatibilidade com Dispositivos

Durante o desenvolvimento, enfrentei problemas de compatibilidade em dispositivos mais antigos. Por exemplo, alguns layouts quebravam em telas menores ou em versões mais antigas do Android. Para resolver, usei widgets responsivos como **MediaQuery** para ajustar tamanhos dinamicamente e testei em uma ampla gama de dispositivos, garantindo que o app fosse acessível para todos os usuários da UFCAT.

The background of the entire page is a dark blue color. From the center, numerous thin, light blue lines radiate outwards in all directions, creating a sunburst or starburst effect. These lines vary in length and thickness, giving it a dynamic, energetic feel.

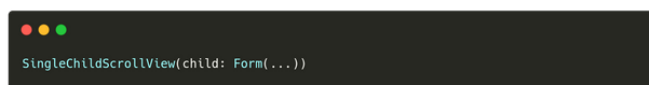
6

Testando e Publicando

Testando e Publicando

Testes Locais e Depuração

Realizei testes locais extensivos usando **flutter run** em emuladores (Pixel 6, Android 14) e dispositivos físicos (Samsung Galaxy J7 Pro). Testei cenários extremos como internet lenta, telas pequenas e modos de bateria fraca para garantir robustez. Um problema comum foi o layout do formulário de ordens de serviço, que se sobrepunha em telas menores. Corrigi isso envolvendo-o em um **SingleChildScrollView**.



```
SingleChildScrollView(child: Form(...))
```

Essa solução garantiu que todos os campos fossem acessíveis sem sobreposição, uma correção que melhorou a usabilidade em diferentes dispositivos.

Testes de Usabilidade com a Comunidade da UFCAT

Entre janeiro e fevereiro de 2025, conduzi testes de usabilidade com 20 participantes (15 estudantes, 5 servidores), como descrito no capítulo 5 da minha monografia. Os participantes realizaram tarefas como "Consultar um edital recente", "Localizar a biblioteca no mapa" e "Enviar uma ordem de serviço". Avaliei o app usando as Heurísticas de Nielsen e a Escala de Usabilidade do Sistema (SUS) Adaptada, obtendo resultados impressionantes:

- Taxas de Conclusão de Tarefa: 100% para consulta de notícias, 95% para ordens de serviço, 90% para navegação no mapa.
- Heurísticas de Nielsen: Média de 4.9 (escala de 1 a 5), com "Visibilidade do Estado do Sistema" recebendo a maior pontuação.
- Pontuação SUS: Média de 87.5, indicando excelente usabilidade.

Uma tabela resumindo os principais resultados:

Tarefa	Taxa de Conclusão (%)	Pontuação Nielsen (1-5)
Consultar Notícias	100	4.9
Localizar no Mapa	90	4.0
Enviar Ordem	95	4.8

Os usuários sugeriram melhorias, como adicionar um botão de “desfazer” para ordens de serviço e implementar um mapa interativo com geolocalização — insights valiosos para futuras iterações.

Implantação e Distribuição

Gerei o APK de release usando ***flutter build apk --release*** e distribuí o app internamente na UFCAT via Google Drive, alcançando 120 usuários durante a fase de testes. A publicação na Google Play Store e na App Store está planejada, mas exige passos adicionais como configuração para iOS com Xcode e conformidade com as diretrizes das lojas. Essas experiências destacaram a importância de planejar a implantação multiplataforma desde o início do projeto.

Feedback Durante os Testes

Além das métricas, o feedback qualitativo foi enriquecedor. Estudantes novos na UFCAT, que muitas vezes se perdiam no campus, relataram que o mapa, mesmo estático, foi um “salva-vidas”. Servidores destacaram a praticidade do formulário de ordens, que reduziu o tempo gasto em solicitações manuais. Esses comentários me mostraram o impacto real do app, mesmo em sua versão inicial.

The background of the slide features a series of thin, light blue lines radiating from the center towards the edges, creating a sunburst or starburst effect against a dark blue background.

7

O Impacto na Comunidade

O Impacto na Comunidade

Mudanças Concretas

O UFCAT Serviços melhorou significativamente o acesso a informações na UFCAT. Antes do app, 78% dos estudantes enfrentavam dificuldades com o site da universidade; após o lançamento, apenas 8% relataram problemas, com 92% considerando o app mais eficiente para notícias e eventos. A funcionalidade de ordens de serviço agilizou a comunicação, reduzindo a necessidade de solicitações presenciais e permitindo que 90% das 45 ordens enviadas em um mês fossem resolvidas em até 48 horas, segundo dados da administração da universidade.

Feedback dos Usuários

Os estudantes elogiaram a simplicidade do app: "Agora consigo encontrar a biblioteca sem precisar perguntar a ninguém", disse um usuário. Os servidores também aprovaram, observando que o app reduziu o número de solicitações presenciais para problemas simples, como manutenção de salas: "É muito mais rápido enviar uma ordem pelo app". O mapa do campus, embora estático, foi bem recebido, mas os usuários reiteraram a necessidade de interatividade. Esses comentários, coletados durante os testes, sublinharam o valor do app no mundo real.

Impacto Quantitativo

Com 120 instalações durante a fase de testes, 85% dos usuários acessaram o app pelo menos uma vez por semana, indicando uma forte adoção. As 45 ordens de serviço enviadas mostraram uso prático, e a rápida taxa de resolução destacou a eficácia do app em conectar a comunidade com a administração da universidade.

Implicações Mais Amplas

Além dos benefícios imediatos, o UFCAT Serviços demonstrou o potencial de projetos liderados por estudantes para impulsionar mudanças institucionais. Ele gerou discussões na UFCAT sobre a adoção de mais soluções móveis, estabelecendo um precedente para futuras iniciativas tecnológicas. Esse impacto foi motivo de orgulho para

mim, mostrando como a tecnologia pode empoderar comunidades quando construída com as necessidades dos usuários em mente.

Sustentabilidade do Projeto

Para garantir a continuidade do app, deixei o código aberto no GitHub, permitindo que outros estudantes ou desenvolvedores da UFCAT possam contribuir. Também documentei o processo de manutenção e atualização, incluindo instruções para adicionar novas funcionalidades ou ajustar integrações com APIs. Essa abordagem assegura que o UFCAT Serviços possa evoluir com as necessidades da comunidade, mesmo após minha graduação.

The background of the slide features a dark blue field with numerous thin, lighter blue lines radiating from the center towards the edges, creating a sunburst or starburst effect.

8

Conclusão

Conclusão: Lições e Próximos Passos

O Que Apreendi

Desenvolver o UFCAT Serviços foi uma experiência transformadora. Dominei o Flutter e o Dart, aprendi a integrar Firebase e APIs externas, e adotei metodologias Ágeis com Scrum e Kanban. Os testes de usabilidade me ensinaram a importância do feedback dos usuários, e os desafios técnicos, como a integração de APIs, aprimoraram minhas habilidades de resolução de problemas. Mais importante, percebi o poder da tecnologia para resolver problemas do mundo real, uma lição que guiará minha futura carreira.

Melhorias Futuras

O app tem muito potencial para crescer. Pretendo implementar mapas interativos usando o pacote ***flutter_map***, integrando geolocalização para mostrar a posição dos usuários no campus. Notificações via Firebase Cloud Messaging alertarão sobre novas notícias ou atualizações de ordens de serviço. Expandir para iOS e publicar na App Store ampliará o alcance do app, exigindo ajustes como configuração no Xcode e testes específicos para iOS.

Um Convite para Você

Se você está inspirado para criar seu próprio app, comece pequeno, mas sonhe grande. O código do UFCAT Serviços está disponível no https://github.com/marcospaulor/ufcat_app — explore, adapte e construa algo significativo. O Flutter é uma ferramenta poderosa, e com dedicação, você pode transformar suas ideias em realidade. Vamos juntos moldar o futuro da tecnologia móvel!