

## Descrição

### Público-Alvo/Intended User

### Funcionalidades/Features

### Protótipo de Interfaces do Usuário

Tela 1

Tela 2

Tela 3

Tela 4

Tela 5

Tela 6

Tela 7

Tela 8

Tela 9

Widget

### Considerações Chave/Key Considerations

Como seu app vai tratar a persistência de dados?

Descreva qualquer caso de uso específico ("corner case") da experiência do Usuário (UX).

Descreva quais bibliotecas você utilizará e compartilhe a razão de incluí-las.

Descreva como você implementará o Google Play Services.

### Próximos Passos: Tarefas Necessárias

Tarefa 1: Configuração do Projeto/Project Setup

Tarefa 2: Implementar a Interface de Usuário (UI) para cada Activity e Fragment

Tarefa 3: Conectar Firebase analytics e Admob

Tarefa 4: Definição de modelos e conexão com database e storage

Tarefa 5: Configurar Location API e Google maps distance matrix API

Tarefa 6: Implementar Autenticação

Tarefa 7: Implementar FCM

Tarefa 8: Implementar Widget

Tarefa 9: Implementar testes unitários e instrumentais

Tarefa 10: Gerar apk de release do aplicativo

**Usuário do GitHub:** tiagofrbarbosa

# Fleekard

## Descrição

Fleekard é um aplicativo de rede social que permite aos usuários encontrarem outros usuários por perto, navegar por perfis, ver fotos, status, marcar perfis como favoritos e caso tenha interesse mandar uma mensagem pelo bate papo, os usuários podem ver quem os visitou, gostou do seu perfil e mandou mensagem através da tela de notificações, além de filtrar outros perfis por distância, faixa etária e gênero.

## Público-Alvo/Intended User

Este é um app para pessoas que estão em busca de novas amizades, bate-papo e relacionamentos.

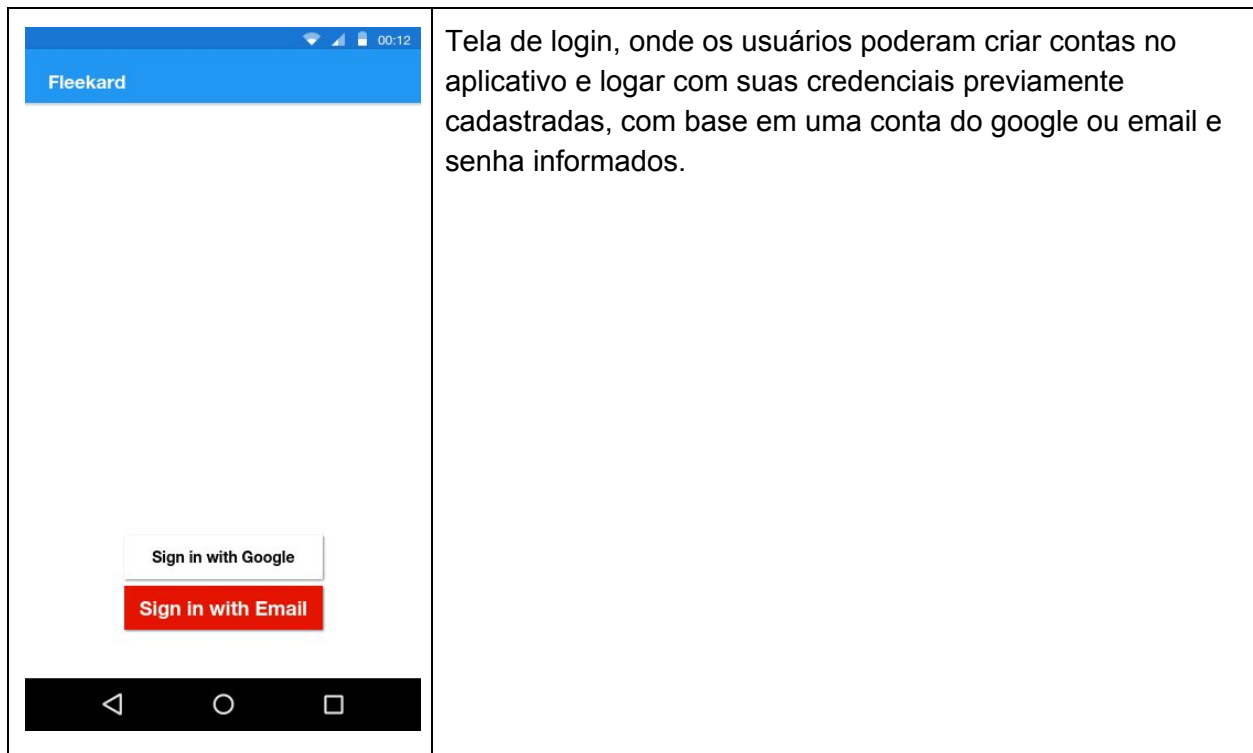
## Funcionalidades/Features

- Mostra outros usuários por perto com base na localização
- Permite que os usuários criem perfis e adicionem foto, status, nome, idade, gênero
- Bate-papo
- Filtro de perfil de outros usuários por distância, gênero e idade
- Opção de curtir outros perfis
- Tela de notificações para exibição de visitas, likes e mensagens

## Protótipo de Interfaces do Usuário

Elas podem ser feitas a mão (tire uma foto dos seus desenhos e os insira neste fluxo), ou usando um programa como o Photoshop ou Balsamiq.

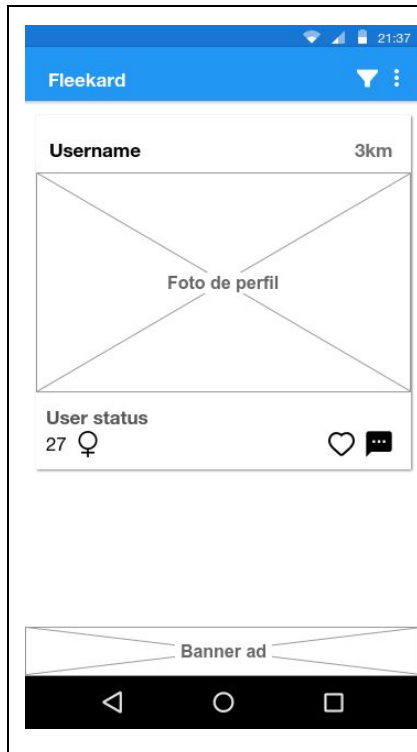
### Tela 1



## Tela 2




### Tela 3

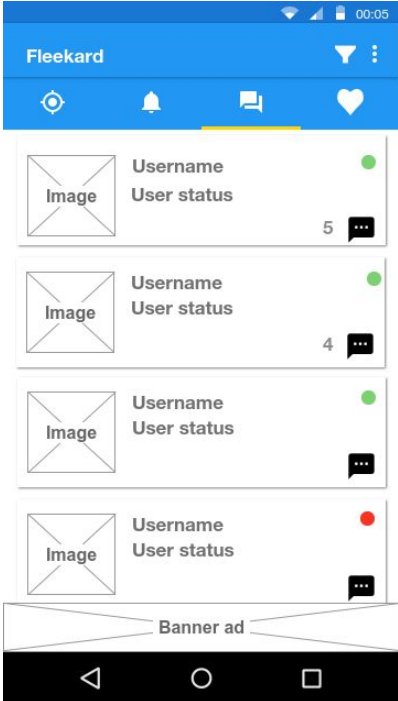


Ao clicar em um perfil na tela anterior, esta tela será mostrada com algumas informações básicas do usuário, além de um botão de like e outro para envio de mensagem.

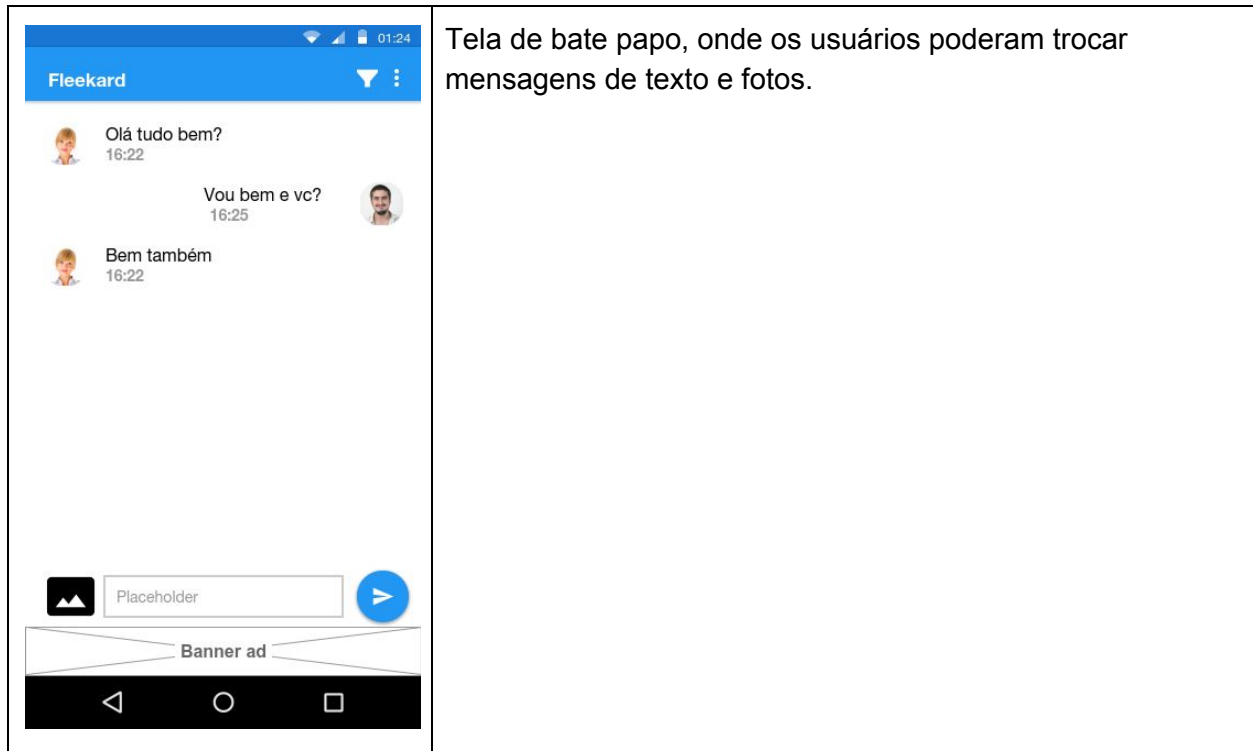
### Tela 4

	<p>Conforme os usuários forem interagindo o app vai registrar e mostrar na tela de notificações os eventos de: visitas, likes e mensagens recebidas.</p>
---	--

## Tela 5

	<p>Esta tela irá mostrar as conversas iniciadas entre usuários, mensagens recebidas, além de status da conexão de outros usuários no chat (online ou offline).</p>
---	--

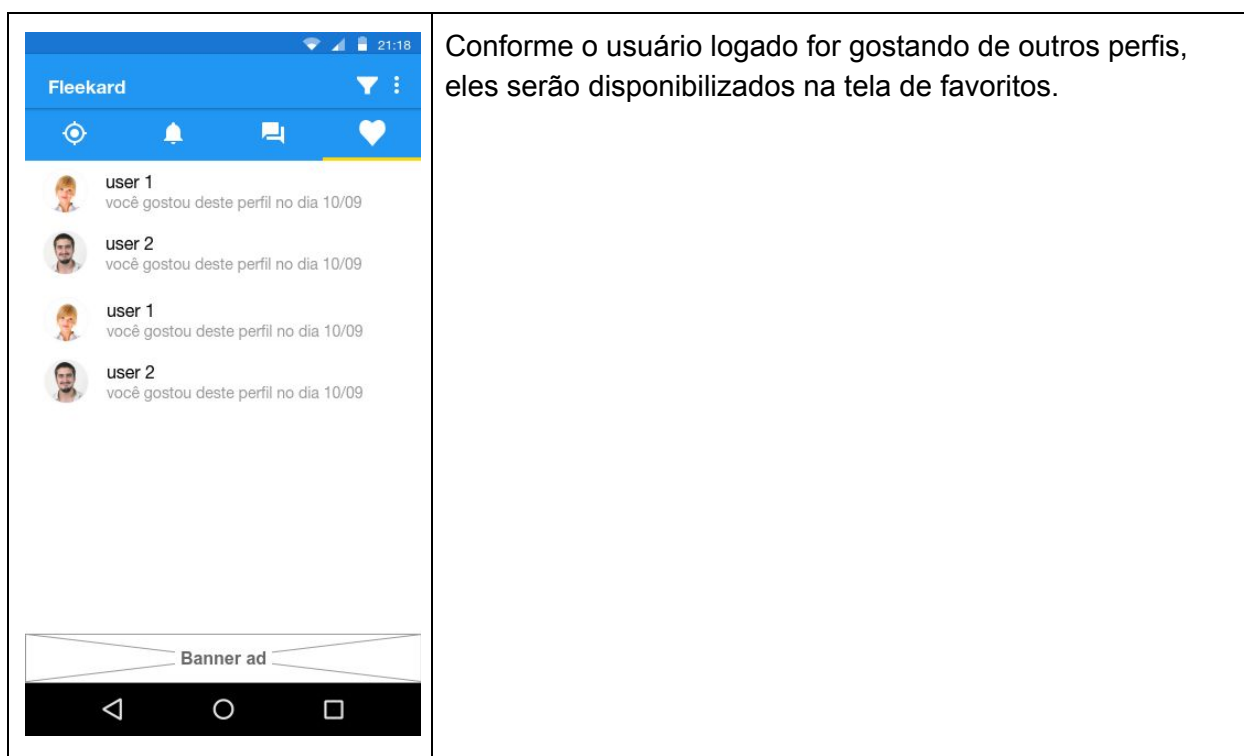
## Tela 6




## Tela 7



## Tela 8




## Tela 9

	<p>Tela de configuração de pesquisa, o usuário poderá alterar valores de distância, faixa etária e gênero para pesquisa de outros usuários.</p>
--	---

## Widget



	<p>O widget do app irá mostrar as notificações de visitas, mensagens e likes.</p>
---	---

Adicione quantas telas achar necessário para demonstrar o fluxo de UI do seu app.

## Considerações Chave/Key Considerations

### Como seu app vai tratar a persistência de dados?

Os dados de texto como informações de usuário, mensagens de bate-papo, notificações e favoritos serão estruturados e armazenados no firebase realtime database e os arquivos de imagem no firebase storage.

- No armazenamento de dados no realtime database a capacidade do firebase em serializar objetos java em json será explorada, a tarefa de armazenamento consistirá em criar cada classe para representar as estruturas na árvore json onde os atributos do objeto farão parte do mesmo nó
- caso o objeto tenha composição esta fará parte do nó filho aninhado, por exemplo será criada a classe User e Location, a classe User terá como atributo uma Location, na

árvore json, Location ficará em um nó filho de User, cada usuário irá possuir apenas uma Location

- no momento da primeira gravação dos dados de usuário (push) será informado o uid obtido através da autenticação, desta forma cada usuário será único na árvore users
- Esta abordagem de serialização será utilizada para definição de todas as classes necessárias para a estrutura, caso as classes tenham relacionamentos bidirecionais será utilizado o conceito de desnormalização e informado os respectivos ids para associação, caso o relacionamento seja unidirecional as estruturas serão aninhadas
- Será ativada a persistência em disco do firebase para que os dados fiquem disponíveis off-line, mesmo que o usuário ou o sistema operacional reinicie o app.
- A tarefa do firebase storage será armazenar fotos do perfil no storage enviadas a partir de arquivos locais e armazená-las em um caminho como este: `/users/{userId}/profilePicture.png`, e armazenar imagens enviadas a partir de conversas de bate-papo em um caminho como: `/chat_photos/{chatId}/Image.png`
- Dados de preferência serão armazenados em SharedPreferences que serão alterados por um preference fragment, este dados serão passados como parâmetros em consultas ao realtime database
- Será explorado o recurso de gerenciamento de presença do realtime database para monitorar os clientes online ou offline mostrando essa informação no chat

### **Descreva qualquer caso de uso específico (“corner case”) da experiência do Usuário (UX).**

O usuário logado poderá acessar perfis de outros usuários de três formas diferentes: pela tela inicial, notificações e favoritos.

### **Descreva quais bibliotecas você utilizará e compartilhe a razão de incluí-las.**

- Dagger2 - para fazer a injeção de dependências.
- ButterKnife - para injetar views e anotar eventos.
- ButterKnife-compiler - processador de annotations do butterknife.
- Glide - para tratamento de imagens.
- Timber - para registro de logs.
- Retrofit2 - para fazer solicitações para a api google maps distance matrix
- Converter-gson - converter para serializar e desserializar json com retrofit2.

- Appcompat-v7 - biblioteca de compatibilidade.
- Design - para usar animações e widgets do material design.
- Recyclerview - para usar recyclerview e o padrão viewholder.
- Cardview - para usar cartões.

**Descreva como você implementará o Google Play Services.**

- Google location API - para captura de latitude e longitude.
- Google maps distance matrix API - para calcular a distância entre duas coordenadas de latitude e longitude.
- Firebase analytics - para captura de métricas e eventos do usuário.
- Firebase realtime database - para armazenamento de dados.
- Firebase storage - para armazenamento de arquivos de imagem.
- Firebase authentication - para autenticação de usuários.
- FirebaseUI - para facilitar o processo de autenticação e criação de usuários.
- Firebase cloud messaging - para envio de mensagens aos usuários pelo console ou por cloud functions.
- Firebase ads - para exibir anúncios em banner.

## Próximos Passos: Tarefas Necessárias

Esta é a parte onde você falará sobre as principais funcionalidades do seu app (mencionadas acima) e as dividirá em tarefas técnicas tangíveis que você pode completar de forma incremental até finalizar o app.

### Tarefa 1: Configuração do Projeto/Project Setup

- Criação do projeto e configuração de sdk mínimo como 16.
- Configurar bibliotecas no gradle.
- Criar modulo dagger.

### Tarefa 2: Implementar a Interface de Usuário (UI) para cada Activity e Fragment

- Criar layout com tabLayout.
- Implementar fragments e layout das telas: (inicial, notificações, chats e favoritos).
- Implementar activity e layout da tela de perfil e edição de perfil.
- Implementar activity e layout da tela de bate papo.
- Criar tela de configurações com preference fragment.
- Implementar recyclerviews.
- Implementar importação de imagens com Glide.
- Implementar acessibilidade.
- Implementar alternância de layout RTL.

### Tarefa 3: Conectar Firebase analytics e Admob

- Configuração e instanciação do firebase analytics na classe application.
- Configuração do firebase admob como banner.

### Tarefa 4: Definição de modelos e conexão com database e storage

- Criar classe para representar usuários.
- Criar classe para representar localização
- Criar classe para representar chats.
- Criar classe para representar mensagens.

- Criar classe para representar notificações.
- Criar classe para representar favoritos.
- Conectar firebase realtime database.
- Implementar sistema de presença do usuário.
- Conectar firebase storage.
- Implementar métodos de upload de imagens.
- Implementar Loaders para movimentação de dados entre as views das activitys e fragments.
- Atualizar adapters das recyclerviews com os novos dados.

### **Tarefa 5: Configurar Location API e Google maps distance matrix API**

- Obter latitude e longitude do usuário com location API.
- Gravar no realtime database.
- Criar consulta na Google maps distance matrix API com retrofit2.
- Criar AsyncTask para rodar a consulta do retrofit2 fora da UI thread.
- Exibir o resultado de distância nas views.

### **Tarefa 6: Implementar Autenticação**

- Configurar autenticação e criação de usuários com firebase ui.
- Configurar regras de leitura e gravação no database.
- Configurar regras de leitura e gravação no storage.

### **Tarefa 7: Implementar FCM**

- Implementar cloud function para envio de mensagem a partir de eventos do database.

### **Tarefa 8: Implementar Widget**

- Criação de widget do aplicativo.
- Criar consultas para alimentar o widget a partir de dados do firebase.

## **Tarefa 9: Implementar testes unitários e instrumentais**

- Implementar testes unitários
- Implementar testes instrumentais

## **Tarefa 10: Gerar apk de release do aplicativo**

- Implementar tarefa installRelease no gradle.
  - Criar keystore do app.
  - Gerar versão de release.
-