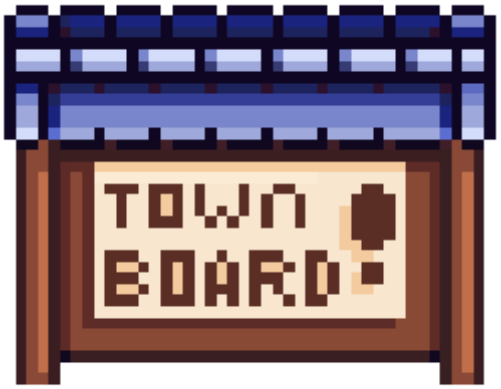
Computação Móvel

Docente: Lourenço Gomes



ESCOLA SUPERIOR DE TECNOLOGIA IPCA

Eduardo Miraldo nº 20768
Rui Cardoso nº 20810
Pedro Saldanha nº 19808

Índice

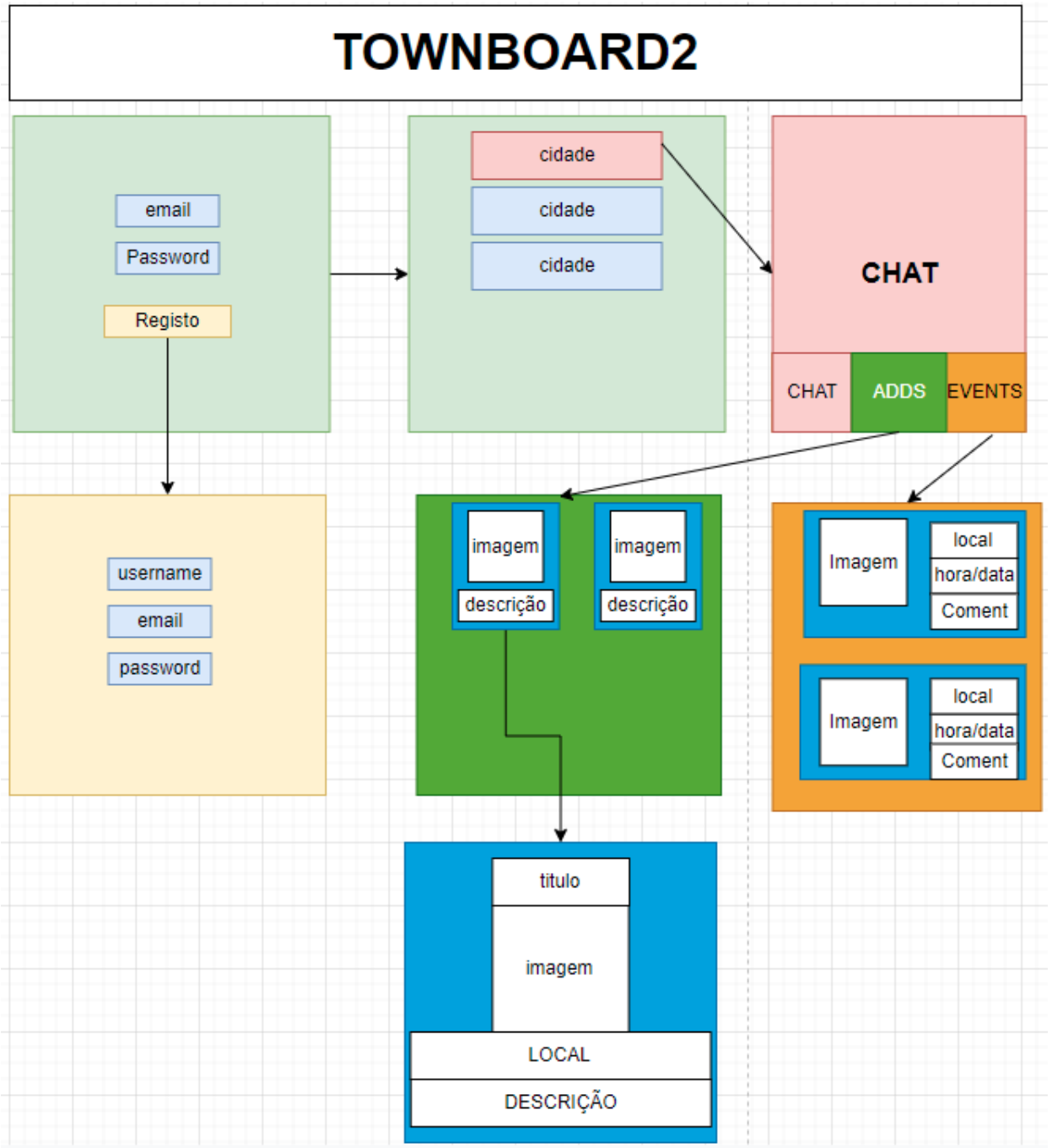
Introdução

Para este projeto, foi pedido que desenvolvêssemos um aplicação para android, com uso do *AndroidStudio,* em *Kotlin.*

Não houve um tema específico para o trabalho, portanto, decidimos escolher algo que fosse útil no quotidiano do nosso país - Uma aplicação que serve para facilitar a comunicação entre co-habitantes de cidades pequenas.

A nossa aplicação consiste num *chat* de grupo em tempo real com a possibilidade de criação e publicidade de eventos e promoções por qualquer utilizador, tudo isto inserido dentro da cidade escolhida pelo utilizador, para evitar *spam* desnecessário e garantindo que toda a informação passada é relevante para o utilizador.

Estrutura


TOWNBOARD2
email
Password
Registo
cidade
cidade
cidade
CHAT
CHAT
ADDS
EVENTS
username
email
password
imagem
descrição
imagem
descrição
Imagem
local
hora/data
Coment
Imagem
local
hora/data
Coment
titulo
imagem
LOCAL
DESCRIÇÃO

Modelos de Dados

- User

```
class User {
   var name:  String? = null
   var email:  String? = null
   var uID:     String? = null
```

- City

```
class City {
   var name : String? = null
```

- Message

```
class Message {
   var message       :  String?  = null
   var senderUID    :  String? = null
   var senderName  :  String? = null
   var date             :  String? = null
   var hour             :  String? = null
```

- Event

```
class Event {

   var name          : String? = null
   var local         : String? = null
   var hora          : String? = null
   var data          : String? = null
   var description  : String? = null
   var photoName : String? = null
```

- Add

```
class Add {
    var photoName  : String? = null
    var name       : String? = null
    var local      : String? = null
    var description : String? = null
```

Funcionalidades e implementação

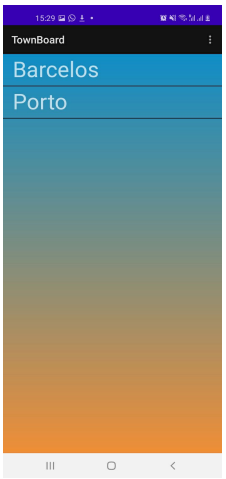
• Registo/Login

Sendo umas das funcionalidades obrigatórias do trabalho, a primeira interação de um utilizador com a aplicação é a inserção de dados válidos nos campos de email/login e password para poder aceder à aplicação. Caso não possua um registo válido, pode clicar no botão para se registar e assim abrir a janela de registo para inserir o seu nome, email e password pretendidos. Esta informação será guardada na base de dados do firebase.

O firebase possui um registo de autenticação pré definido, limitado ao número de campos. Assim sendo, criamos uma classe *User* que guarda todos os campos da autenticação que serão necessários para, por exemplo, identificar o utilizador que enviou uma mensagem no chat. No futuro, poderiam também ser utilizados para guardar os criadores de eventos ou publicidades.

Tecnologias Utilizadas: Firebase Authentication, Firebase Firestore

• City



Após ter feito o login com sucesso, o utilizador é direcionado para uma *activity* onde lhe são apresentadas todas as cidades que fazem parte da aplicação, através de uma *RecyclerView*, podendo escolher com qual deseja interagir. Após escolher essa cidade, o utilizador é endereçado ao núcleo da aplicação, onde pode ter total acesso às funcionalidades de chat, publicidades e eventos.

Tecnologias Utilizadas: Firebase Firestore

• Chat

O chat da cidade contém a informação de quem enviou a mensagem, a data/hora e correspondente ordenação para uma fácil visualização dos eventos a ocorrer. Outra funcionalidade é que, as mensagens enviadas pelo utilizador aparecem do lado direito do ecrã com o fundo roxo, enquanto que as mensagens recebidas dos restantes utilizadores aparecem do lado esquerdo, tendo assim uma certa semelhança com outras aplicações de chat mais conhecidas(whatsapp, facebook messenger, …).

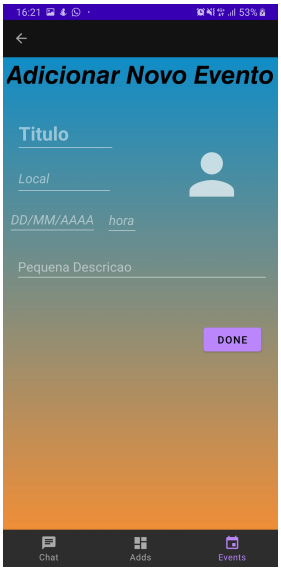
Para a realização desta funcionalidade, foram criados três *layouts*. O *message send*(mensagens de quem envia), *message_receive* (mensagens dos outros utilizadores) e um *fragment_chat* (todas as mensagens). Para que haja uma organização das mensagens e como devem aparecer perante o utilizador, foi necessário criar um adapter que faz essa gestão das views. Este adapter estende para um *RecyclerView Adapter,* cujo *ViewHolde*r encarrega-se em atribuir as mensagens aos lados correspondentes.

Todas as mensagens e informações relevantes são guardadas na nossa base de dados sob a coleção da cidade correspondente.

Sempre que a base de dados muda, ou seja, que uma nova mensagem foi enviada, o nosso *ChatFragment* busca os dados atualizados e notifica o nosso *ChatAdapter* desta mudança que retorna ao utilizador a vista atualizada do estado do *chat.*

Tecnologias Utilizadas: Firebase Authentication, Firebase Firestore

• Events

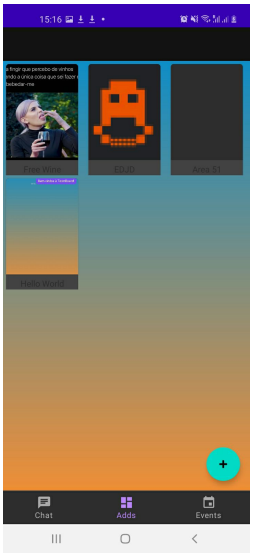


Os eventos contém informação simples e rápida sobre eventos que aconteceram ou que acontecerão. O desenvolvimento desta funcionalidade foi divido em 3 partes, a primeira é seu *row* que é feito com *LinearLayout* que está ligado ao *fragments* onde vai buscar as informações que são inseridas como nome,data,hora,local, photo e uma pequena descrição, todas as informações são colocadas em uma *listView.*

Por fim temos a parte de adicionar um novo evento que está num *fragment* a parte.

Tecnologias Utilizadas: Firebase Firestore, Firebase Storage

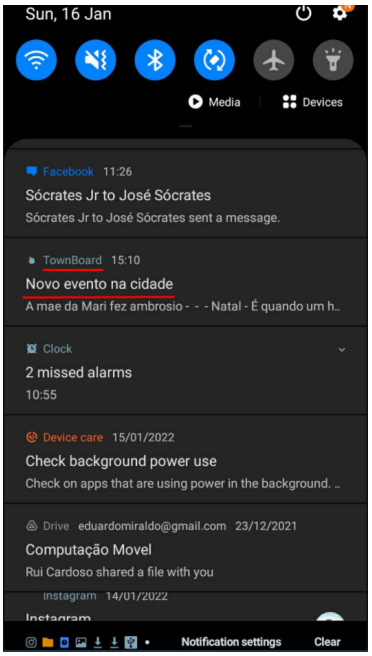
• Adds



Os *adds* são a parte da aplicação destinada a divulgação de promoções, produtos, lançamentos e outras informações que, diferentemente de eventos, não possuem um data e hora em concreto.

Para criar este fragmento utilizamos uma *RecyclerView.* Mas quisemos ir mais longe e ao invés de utilizarmos uma *row* comum como, por exemplo, nas funcionalidades do *chat* e dos *events* criamos um *CardLayout.* O que não apenas torna a aplicação mais dinâmica, mas também ajuda a diferenciar esta parte em relação à sua "irmã", os *events*

Tecnologias Utilizadas: Firebase Firestore, Firebase Storage



Notificações

O *firebase* dispõe de um sistema de *cloud Messaging* para o envio de notificações para os utilizadores. Para que haja ligação entre a aplicação e esse sistema, é obrigatório adicionar os comandos do serviço no *AndroidManisfest* (MyFirebaseMessagingService). É possível enviar uma notificação manualmente pela própria *firebase*, preenchendo o assunto, texto e uma data específica, portanto o que fizemos foi apenas criar uma função que preenchesse esses campos com a informação sobre de um evento acabado de criar.

Conclusão

Durante o desenvolvimento deste projeto, tivemos alguns desafios consideráveis. De todos eles, o mais impactante(e talvez o que mais proporcionou um crescimento académico e profissional) foi, sem sombra de dúvidas, ser a primeira vez que estávamos a trabalhar com uma linguagem recente, sem derivação de C. Sair da zona de conforto do que havíamos aprendido até agora e trabalhar com variáveis dinâmicas, sem sequenciação de operações, sem função *"main"*, foi uma aventura da mesma forma que agora mostra-se gratificante.

É de ressaltar que *Kotlin* foi uma ótima companheira neste projeto, no entanto, a *API* que estávamos a utilizar - *AndroidStudio* - parecia fazer tudo para que não conseguíssemos concluir o projeto. A maior parte do tempo passado a desenvolver, foi gasta em aprender o que era que o compilador nos estava a dizer, fosse em erros de compilação ou em sessões de *debugging* - *"Source code does not match the byte code"* - parecia o monólogo do compilador enquanto passámos horas a mudar as definições do *AndroidStudio* conforme o oráculo(Google) nos recomendava. No final, era porque tínhamos nomeado uma variável errada ou outro tipo de falta de atenção que, ao trabalharmos com C ou C# era imediatamente avisado pelo compilador.

Além disso, compreender corretamente o que é que as várias classes nativas do *AndroidStudio* realmente fazem e para que realmente servem, demora naturalmente um ou mais projetos. Neste momento, temos a crença que compreendemos quase por inteiro o que é um *binding*, o que é fragmento, o que é um *adapter,* como essas classes conectam uma com as outras e como trabalhar com elas, mas certas coisas ainda se mostram como um mistério - "Porque este *setOnClickListener* funciona aqui, mas não ali?"; "Porque esta variável é uma *string*, a função pede uma *string*, mas o compilador diz que é um *bitmap*?"; "Porque a maneira de fazer upload de uma imagem, conforme diz a própria documentação do *AndroidStudio* não funciona?". Estas foram apenas as mais comuns de todos os impasses que tivemos durante o desenvolvimento deste projeto(a vida de um programador *android* não é a mais fácil), mas é ao ultrapassar elas que realmente aprendemos, e é bastante gratificante saber que podemos ter uma ideia e a realizar.