
Utilizando o back-end como um serviço(BaaS) em uma aplicação Android

— Introdução —

O que é BaaS(Back-end as a Service)?

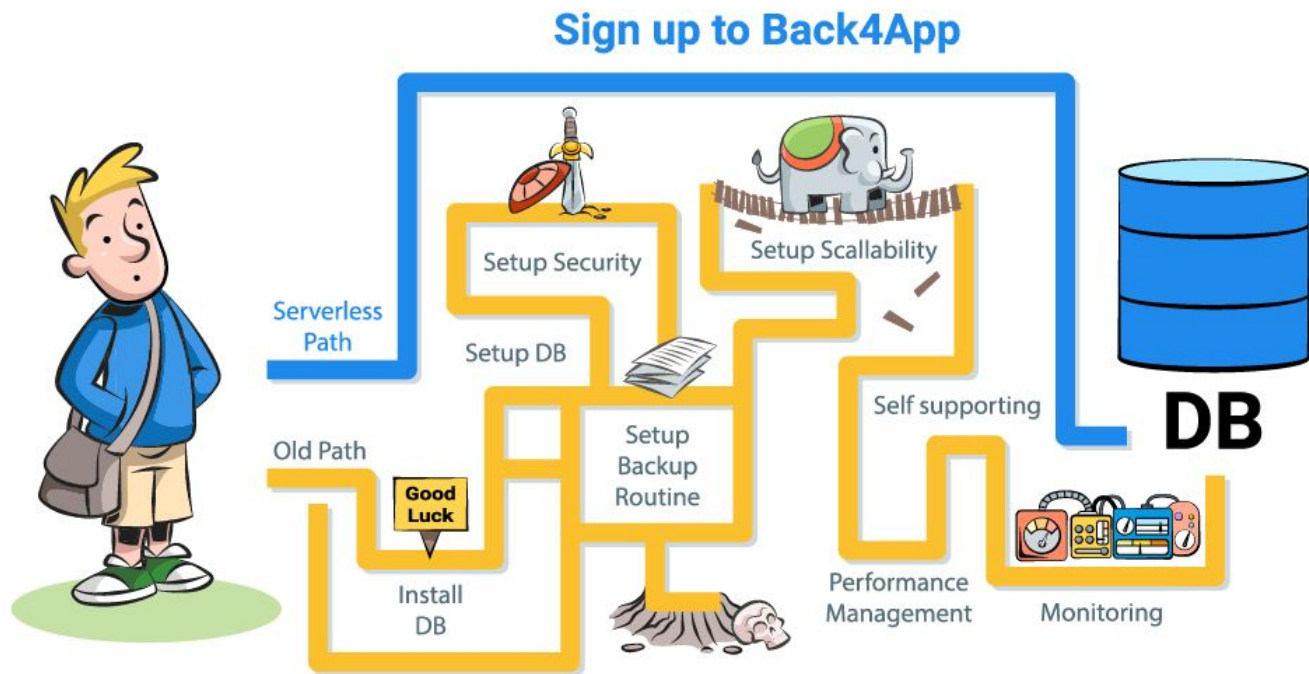
Uma maneira simples de ligar a sua aplicação a um back-end em nuvem

- Conceito recente de back-end
- Ajuda a acelerar o desenvolvimento de software
- Simplifica a criação e utilização de uma API
- Trata especificamente as necessidades de computação em nuvem de desenvolvedores de aplicativos da Web e móveis

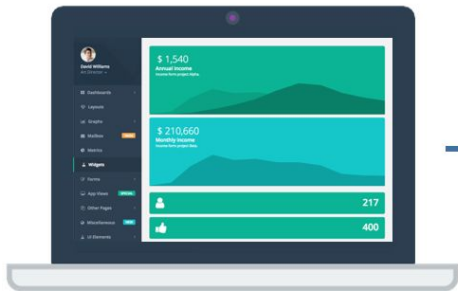


Por que utilizar?

- Conhecimento do desenvolvedor
- Custo
- Velocidade
- Prototipagem
- Escalabilidade
- Segurança



Exemplos



Backend-as-a-Service

 Firebase

 kinvey

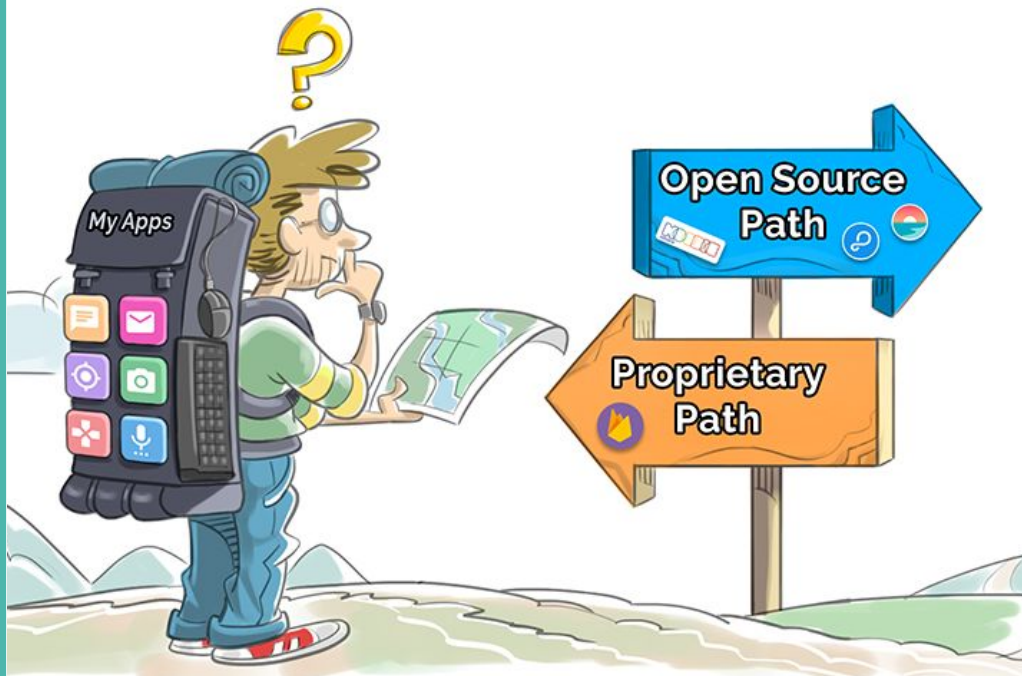
Microsoft Azure

 Parse

 BaQend

Observações

É preciso ler atentamente os termos de uso de cada fornecedor pois alguns bloqueiam os dados e/ou código fonte, tomando o controle total do desenvolvedor. Impossibilitando a migração da base de dados para outra infraestrutura, sem que o desenvolvedor tenha que começar novamente.



Parse



- Surgiu em 2011
- Comprado pelo Facebook em 2013
- Descontinuado pelo Facebook em 2016
- Problemas com a migração
- Parse Server é mantido pela comunidade
- **Back4App** cria uma solução para o problema com a migração



Back4App



Com amadurecimento do Parse Server pela comunidade foi possível a criação de um novo BaaS, o Back4App.

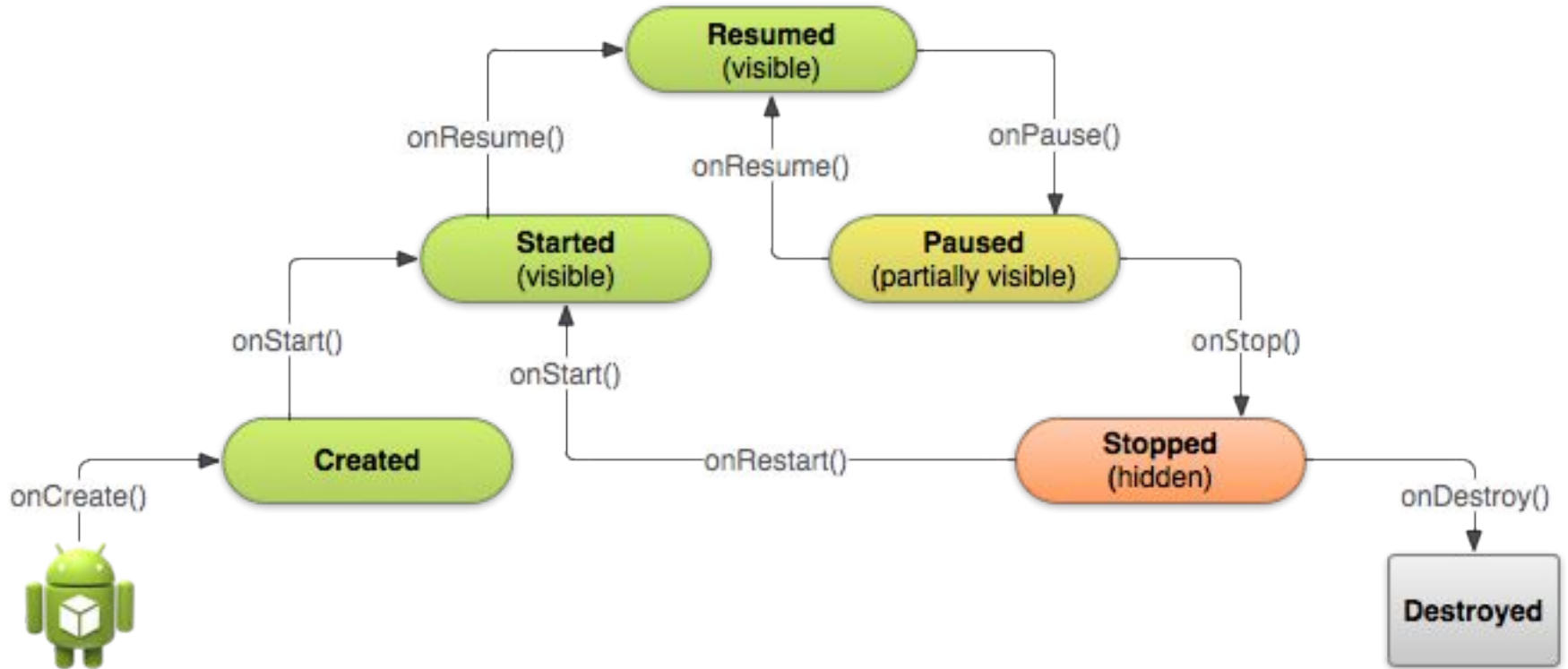
Back4App

BaaS para criar APIs e back-end automaticamente.

- Boa infraestrutura (Servidores da Amazon)
- Seguro
- Escalável



Activity: ciclo de vida



Utilizando o back-end como um serviço(BaaS) em uma aplicação Android

— Desenvolvimento —

Configuração

- Deixe o Android Studio abrindo
- Entre no site www.back4app.com
- Crie sua conta
- Crie um novo App

**Parse Server
made simple**

Build the backend with no code, host with no infrastructure hassles and scale with no technical locks.

**Sign up and build your App
in minutes**

E-mail

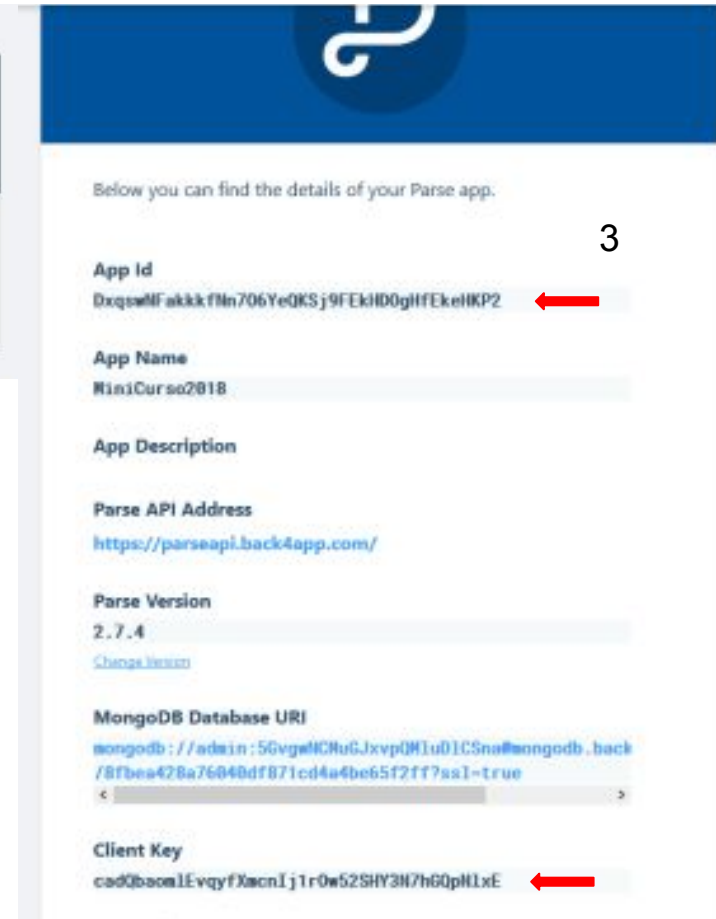
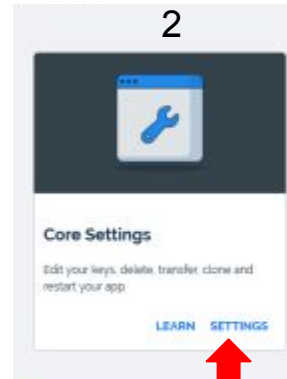
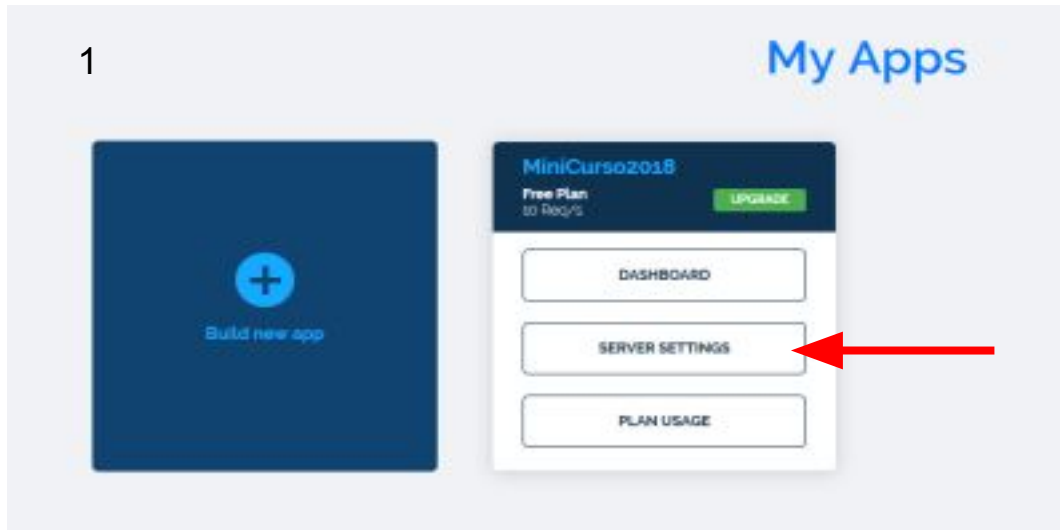
Password

GET STARTED!

By signing up you agree to Back4App's [Terms of Service](#) and [Privacy Policy](#)

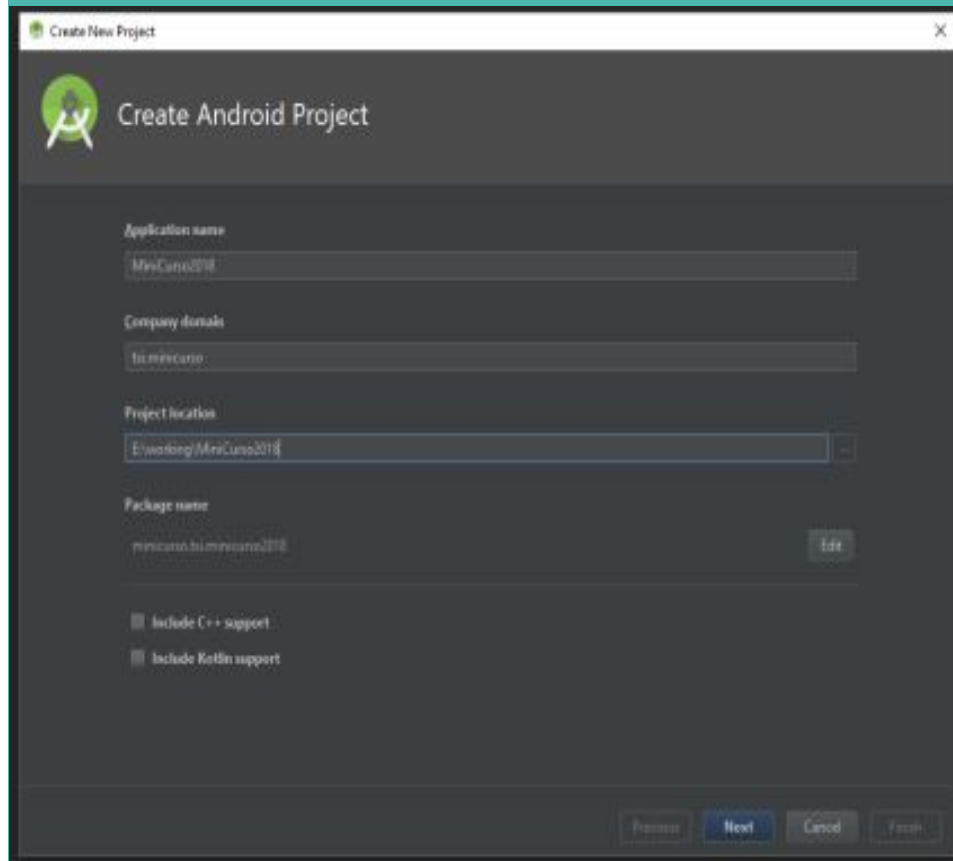
Configuração

- Server Settings
- Core Settings > Settings
- Salve as chaves App Id e Cliente Key

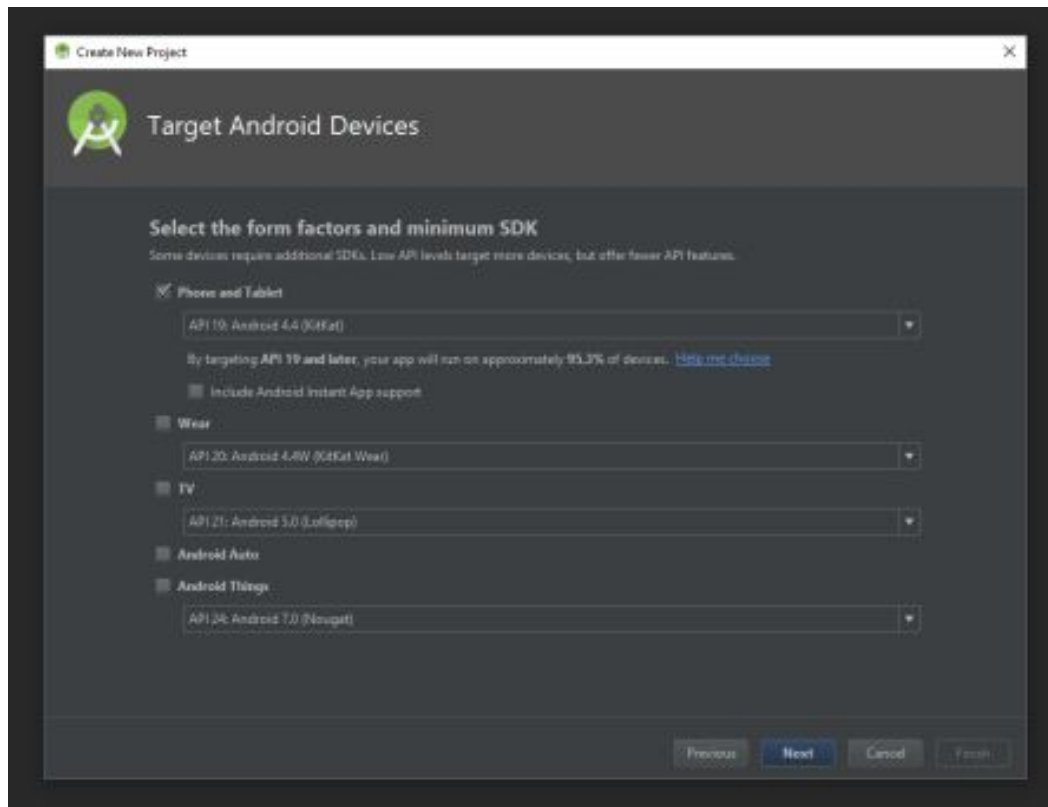


Configuração

- Voltando ao Android Studio
- Create new Project
- Dê um nome ao projeto
- Um nome ao pacote
- Escolha um local
- Next



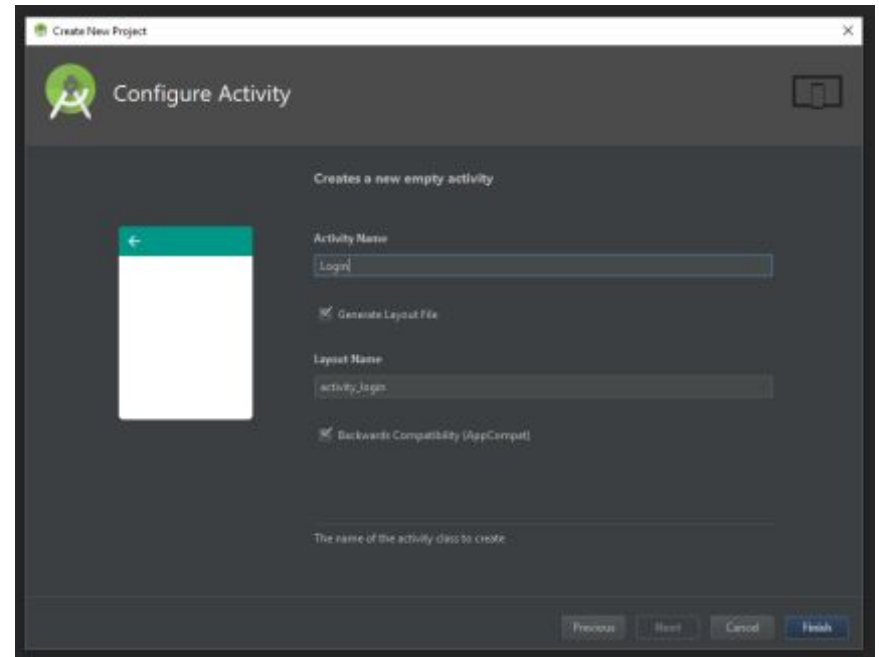
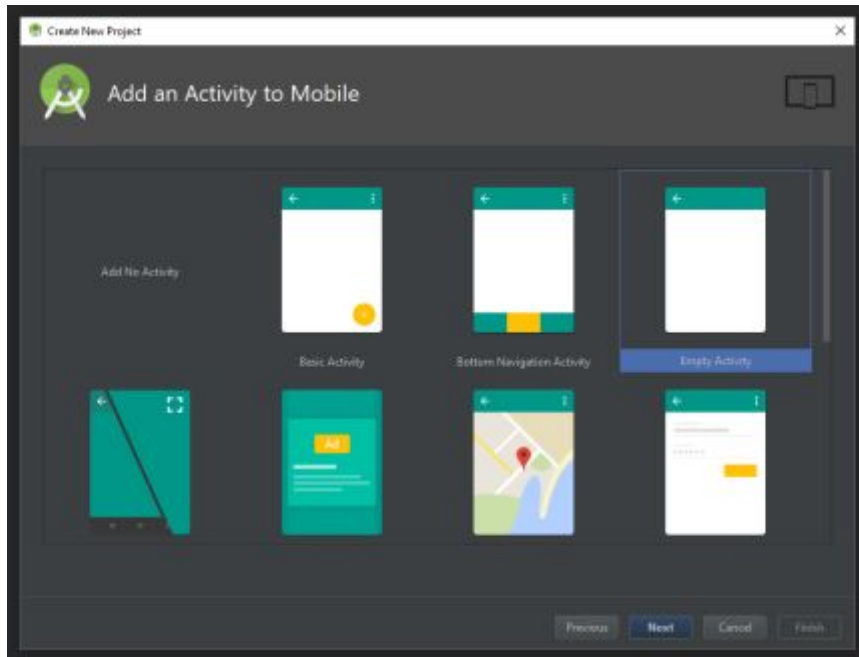
Configuração



- Nessa tela escolha Phone and Tablet
- Marque a API 19, recomendada pela Google por conter o maior número de celulares rodando essa API ou posterior
- Next

Configuração

- Crie um Activity em branco e em seguida nomeia para MainActivity
- Finish



Configuração

- No Gradle no nível do projeto
- Adicione as linhas à tag `allprojects`

```
allprojects{  
    repositories{  
        google()  
        mavenCentral()  
        jcenter()  
        maven{  
            url 'http://maven.google.com'  
        }  
    }  
}
```

```
// NOTE: Do not place your application dependent  
// in the individual module build.gradle files  
  
}  
  
allprojects {  
    repositories {  
        google()  
        mavenCentral()  
        jcenter()  
        maven {  
            url "http://maven.google.com"  
        }  
    }  
}  
  
task clean(type: Delete) {  
    delete rootProject.buildDir  
}
```


Configuração

- No Gradle no nível do app
- Adicione essa linha a tag dependencies

compile 'com.parse:parse-android:1.16.3'

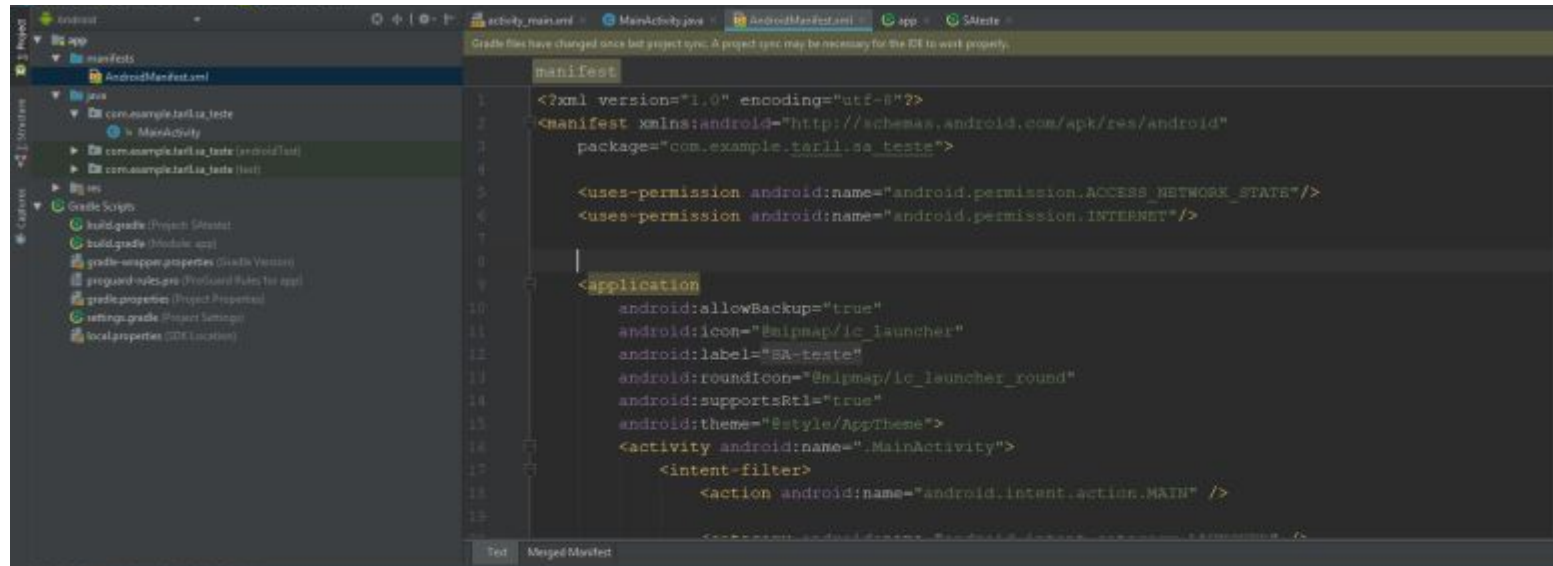
```
dependencies {  
    implementation fileTree(dir: 'libs', include: ['*.jar'])  
    implementation 'com.android.support:appcompat-v7:27.1.1'  
    implementation 'com.android.support.constraint:constraint-layout:1.1.3'  
    testImplementation 'junit:junit:4.12'  
    androidTestImplementation 'com.android.support.test:runner:1.0.2'  
    androidTestImplementation 'com.android.support.test.espresso:espresso-core:3.0.2'  
  
    compile 'com.parse:parse-android:1.16.3' //atualize para a versão mais recente  
}
```

Configuração

- No arquivo AndroidManifest.xml
- Adicione permissões para acesso a Internet

`<uses-permission android:name="android.permission.ACCESS_NETWORK_STATE"/>`

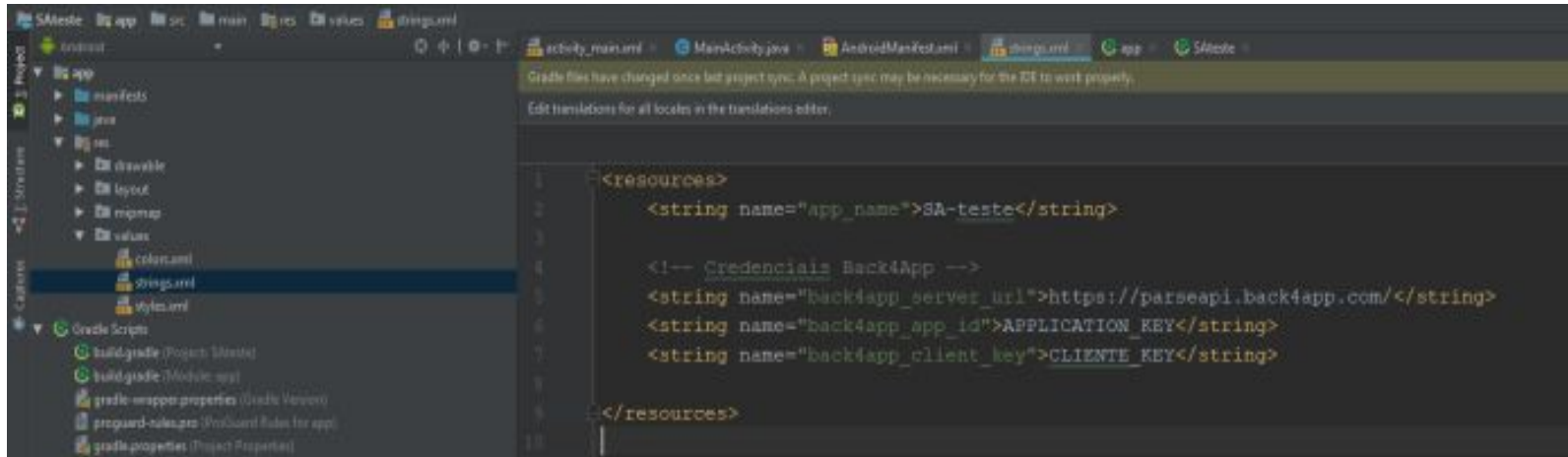
`<uses-permission android:name="android.permission.INTERNET"/>`



Configuração

- Abra res > values > strings.xml
- Insira as tags substituindo pelas chaves obtidas anteriormente nos locais indicados

```
<string name="back4app_server_url">https://parseapi.back4app.com/</string>  
<string name="back4app_app_id">APPLICATION_KEY</string>  
<string name="back4app_client_key">CLIENT_KEY</string>
```



Configuração

- Voltando para AndroidManifest.xml
- Dentro de <application> insira:

```
<meta-data
    android:name="com.parse.SERVER_URL"
    android:value="@string/back4app_server_url" />
<meta-data
    android:name="com.parse.APPLICATION_ID"
    android:value="@string/back4app_app_id" />
<meta-data
    android:name="com.parse.CLIENT_KEY"
    android:value="@string/back4app_client_key" />
```

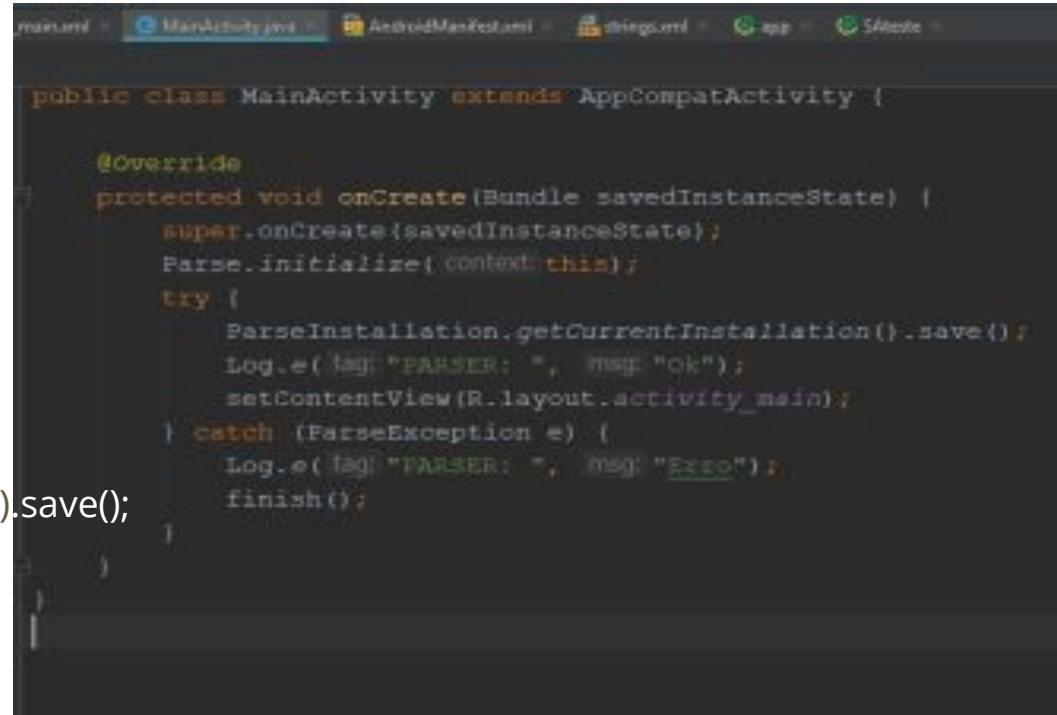


```
13 android:icon="@mipmap/ic_launcher_round"
14 android:supportRtl="true"
15 android:theme="@style/AppTheme">
16
17
18
19
20 <meta-data
21     android:name="com.parse.SERVER_URL"
22     android:value="@string/back4app_server_url" />
23 <meta-data
24     android:name="com.parse.APPLICATION_ID"
25     android:value="@string/back4app_app_id" />
26 <meta-data
27     android:name="com.parse.CLIENT_KEY"
28     android:value="@string/back4app_client_key" />
29
30
31 <activity android:name=".MainActivity">
32     <intent-filter>
```

Configuração

- Vá para a sua classe MainActivity.java
- Dentro do método sobrecarregado onCreate() insira as linhas

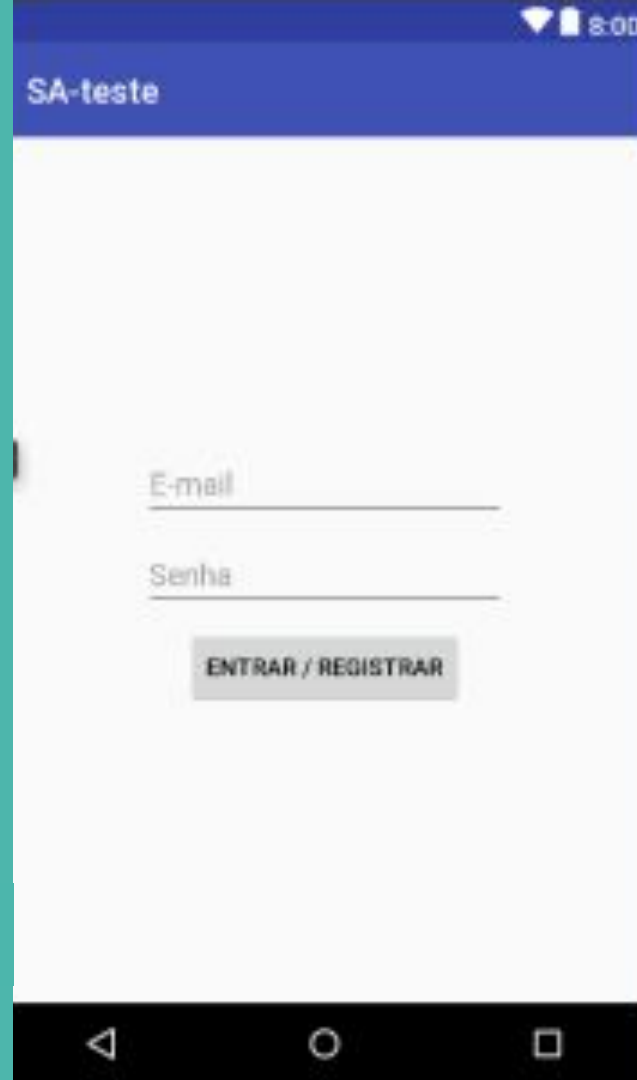
```
Parse.initialize(this);  
try {  
    ParseInstallation.getCurrentInstallation().save();  
    Log.e("PARSER: ", "Ok");  
    setContentView(R.layout.activity_main);  
} catch (ParseException e) {  
    Log.e("PARSER: ", "Erro");  
    finish();  
}
```



```
public class MainActivity extends AppCompatActivity {  
  
    @Override  
    protected void onCreate(Bundle savedInstanceState) {  
        super.onCreate(savedInstanceState);  
        Parse.initialize(context, this);  
        try {  
            ParseInstallation.getCurrentInstallation().save();  
            Log.e(tag, "PARSER: ", msg, "Ok");  
            setContentView(R.layout.activity_main);  
        } catch (ParseException e) {  
            Log.e(tag, "PARSER: ", msg, "Erro");  
            finish();  
        }  
    }  
}
```


Login/Signup

- Em res > layout > activity_main.xml
- Retire o label "HelloWorld"
- Adicione 2 EditText
- Um para Nome(PlainText)
- Outro para Senha(Password)
- Adicione um Button Entrar/Registrar
- O resultado deve ser um tela parecida com a ao lado



Login/Signup

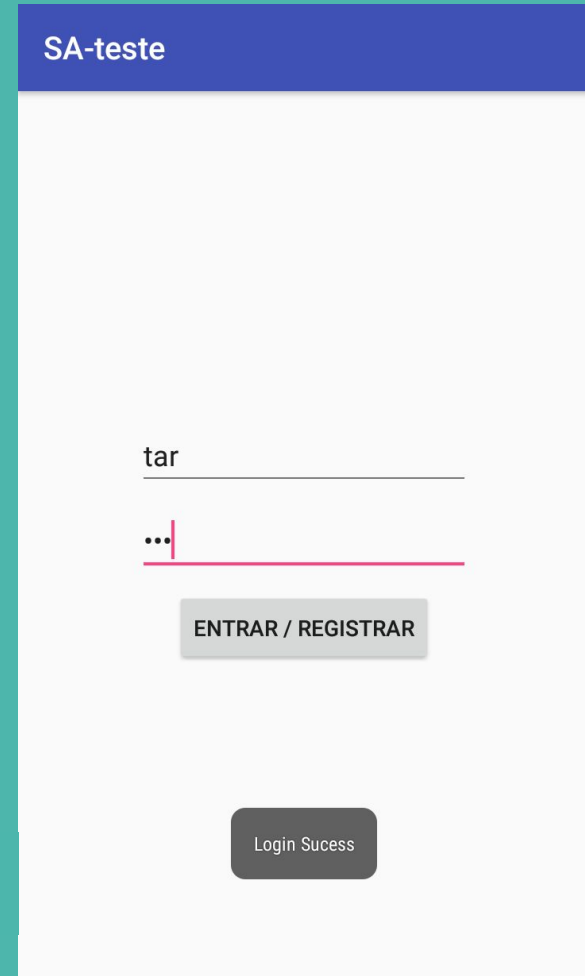
- Vá em LoginActivity.java
- Implemente a interface View.OnClickListener
- Adicione as variáveis para os componentes criados anteriormente
- No método onCreate() encontre as views pelo id e set o onClick do botão para essa classe
- Implemente o método onClick()
- Teste o aplicativo e verifique o Logcat para obter os resultados

```
public class LoginActivity extends AppCompatActivity implements View.OnClickListener {  
  
    private Button btn_entrar;  
    private EditText nomeUsuario;  
    private EditText senha;  
  
    @Override  
    public void onClick(View view) {  
        //Tenta realizar o login, Se não for possível tenta fazer um registro  
        try {  
            ParseUser.logIn(nomeUsuario.getText().toString(),senha.getText().toString());  
            Log.e( tag: "USUARIO: ", msg: "Entrou");  
        } catch (ParseException e) {  
            Log.e( tag: "USUARIO: ", msg: "Falhou");  
  
            //Cria um novo objeto ParseUser e coloca os dados obrigatórios para fazer o login  
            ParseUser usuario = new ParseUser();  
            usuario.setUsername(nomeUsuario.getText().toString());  
            usuario.setPassword(senha.getText().toString());  
  
            //Tenta realizar o cadastro do novo usuário  
            try {  
                usuario.signUp();  
                Log.e( tag: "USUARIO: ", msg: "Criou");  
            } catch (ParseException f){  
                Log.e( tag: "USUARIO: ", msg: "Não Criou");  
            }  
        }  
  
        @Override  
        protected void onCreate(Bundle savedInstanceState) {  
            super.onCreate(savedInstanceState);  
            Parse.initialize( context: this);  
            try {  
                ParseInstallation.getCurrentInstallation().save();  
                Log.e( tag: "PARSER: ", msg: "Ok");  
                setContentView(R.layout.login_activity);  
            } catch (ParseException e) {  
                Log.e( tag: "PARSER: ", msg: "Erro");  
                finish();  
            }  
        }  
  
        btn_entrar = findViewById(R.id.btn_entrar);  
        nomeUsuario = findViewById(R.id.txt_usuario);  
        senha = findViewById(R.id.txt_senha);  
  
        btn_entrar.setOnClickListener(this);  
    }  
}
```


Login/Signup

- Ainda no mesmo arquivo troque os Log.e por Toast.makeText()
- Rode o aplicativo
- Dessa vez o resultado deverá ser como ao lado

```
Toast.makeText(this,"Login Sucess",  
              Toast.LENGTH_SHORT).show();
```



Usuários

- File > New > Activity > Basic Activity
- Dê o nome de UsuariosActivity
- Volte para LoginActivity.java
- Substitua o Toast de "Login Sucess" por uma intenção para abrir a nova activity
- Chame o método startActivity passando como parâmetro a intenção criada

```
Intent intent = new Intent(this, UsuariosActivity.class);  
startActivity(intent);
```

```
@Override  
public void onClick(View view) {  
    //Tenta realizar o login, Se não for possível tenta fazer um registro  
    try {  
        ParseUser.logIn(nomeUsuario.getText().toString(), senha.getText().toString());  
        Intent intent = new Intent(this, UsuariosActivity.class);  
        startActivity(intent);  
    } catch (ParseException e) {  
        Log.e( tag: "USUARIO: ", msg: "Falhou");  
        //Cria um novo objeto ParseUser e coloca os dados obrigatórios para fazer o login  
        ParseUser usuario = new ParseUser();  
        usuario.setUsername(nomeUsuario.getText().toString());  
        usuario.setPassword(senha.getText().toString());  
        //Tenta realizar o cadastro do novo usuário  
        try {  
            usuario.signUp();  
            Log.e( tag: "USUARIO: ", msg: "Criou");  
        } catch (ParseException f){  
            Log.e( tag: "USUARIO: ", msg: "Não Criou");  
        }  
    }  
}
```

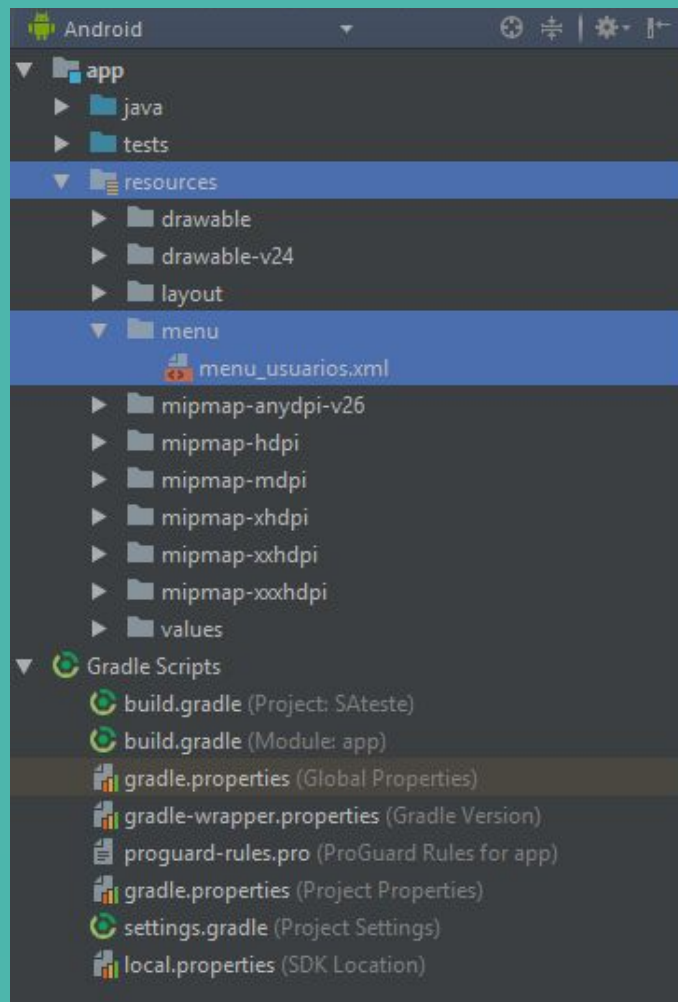
Usuários

- Aderindo as boas práticas de programação
- Crie 2 pacotes: modelo e dao
- Crie a classe Usuario
- Adicione os campos:
 - ◆ String username, password, email, curso id;
 - ◆ int periodo
- Adicione os GET/SET correspondentes(alt + insert)
- Crie a classe UsuarioDAO
- Na classe DAO implemente o método getCurrentUser que tem como retorno um Usuario

```
public class UsuarioDAO {  
    public static Usuario getCurrentUser() {  
        ParseUser pUser = ParseUser.getCurrentUser();  
  
        Usuario user = null;  
  
        if(pUser != null) {  
            user = new Usuario();  
  
            user.setId(pUser.getObjectId());  
            user.setUsername(pUser.getUsername());  
            user.setPassword(pUser.getString( key: "password"));  
            user.setEmail(pUser.getEmail());  
            user.setTelefone(pUser.getString( key: "telefone"));  
        }  
  
        return user;  
    }  
}
```

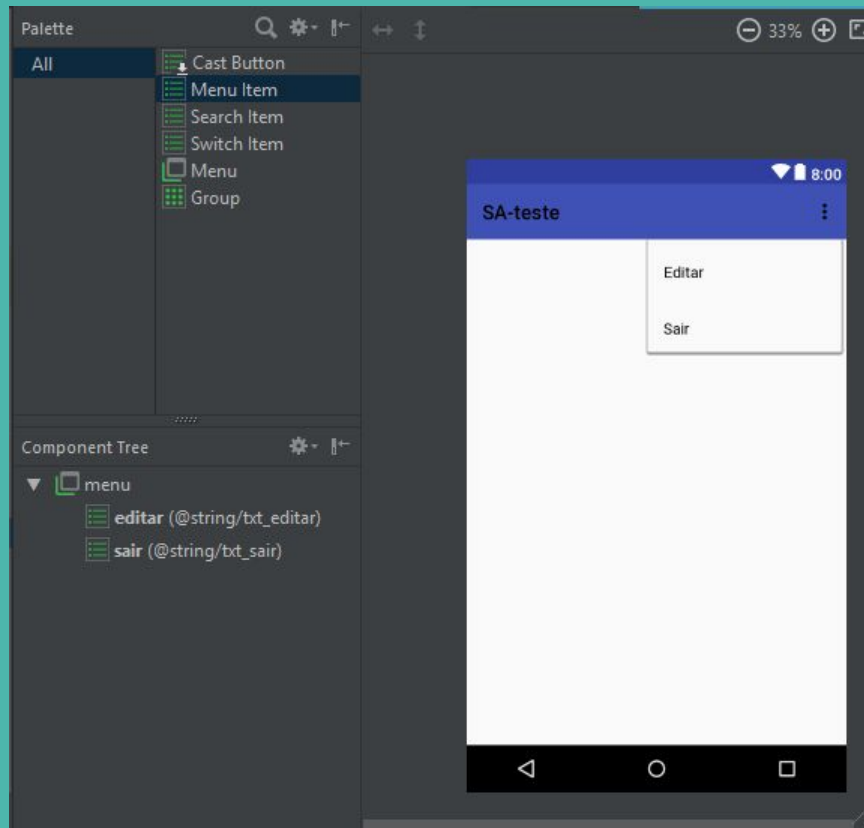
Usuários

- Botão direito em res > new > Android Resource Directory
 - Directory name: menu
 - Resource type: menu
-
- Botão direito no diretório criado > new > Menu resource file
 - File name: menu_usuarios



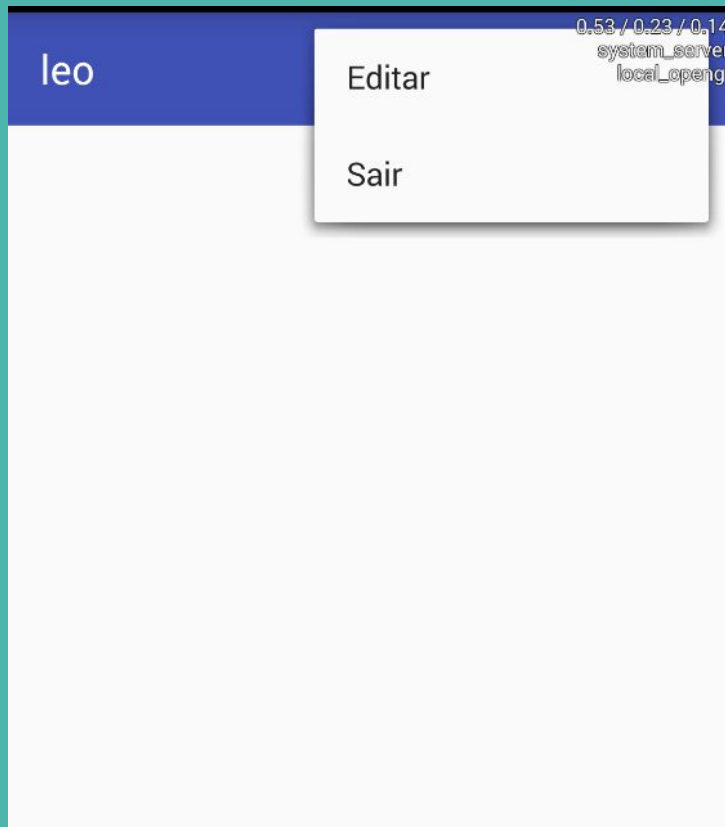
Usuários

- No menu criado adicione dois Menu Item
- Altere seus nomes para Editar e Sair
- Volte ao Arquivo UsuariosActivity
- Em onCreate()
- Encontre o Toolbar pelo id
- Altere o título do Toolbar para o nome do usuário corrente



Usuários

- Sobrescreva os métodos `onCreateOptionsMenu()`, `onOptionsItemSelected()`
- No primeiro adicione a linha:
 - ◆ `getMenuInflater().inflate(R.menu.menu_usuarios, menu);`
- No segundo verifique qual é o id do item clicado e realiza uma ação de acordo



Usuários

- No arquivo content_usuarios.xml
 - ◆ Adicione um ListView com o id lista_usuarios
- Volte para UsuariosActivity
- Crie a variável de instância listaUsuarios do tipo ListView
- Faça a referência através do método findViewById
- Crie um List<String> e a popule com nomes
- Crie o Adapter passando o contexto, layout e a lista criada
- Set o adapter da lista

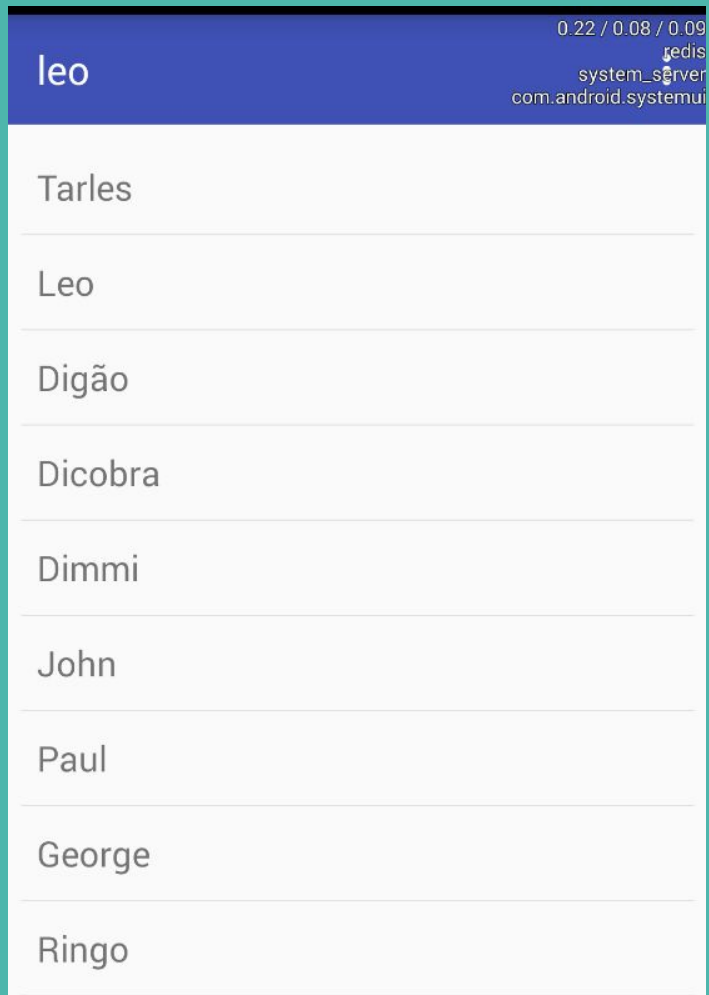
```
listaUsuarios = findViewById(R.id.lista_usuarios);

List<String> usuarios = new ArrayList<>();
usuarios.add("Tarles");
usuarios.add("Leo");
usuarios.add("Digão");
usuarios.add("Dicobra");
usuarios.add("Dimmi");
usuarios.add("John");
usuarios.add("Paul");
usuarios.add("George");
usuarios.add("Ringo");

ArrayAdapter<String> itens = new ArrayAdapter<>(
    context: UsuariosActivity.this,
    android.R.layout.simple_list_item_1,
    usuarios);
listaUsuarios.setAdapter(itens);
```

Usuários

- Implemente a Interface
 - ◆ AdapterView.OnItemClickListener
- Sobrescreva o método:
 - ◆ onItemClick
- Ele informa a posição do item clicado portanto, por enquanto, iremos exibir em um Log a posição clicada
- O resultado será uma tela como a ao lado



Usuários

→ Implemente o método listaUsuarios no arquivo UsuarioDAO

- ◆ Retorna um List<Usuario>

→ Volte em UsuariosActivity

- ◆ Substitua a lista criada anteriormente por uma lista retornada pelo método criado acima
- ◆ Percorra a lista de usuários salvando seus nomes em uma segunda lista
- ◆ Altere o adapter, agora ele deve receber a lista de nomes criada em seu construtor

→ Salve seu código e execute

```
//referencia a lista no arquivo xml
listaUsuarios = findViewById(R.id.lista_usuarios);

//Cria uma lista de usuarios com os usuarios do banco de dados
List<Usuario> usuarios = UsuarioDAO.listaUsuarios();

//Cria uma lista de nomes a partir da lista de usuarios
List<String> nomesUsuarios = new ArrayList<>();
if(usuarios != null)
    for (Usuario usuarioCorrente : usuarios)
        nomesUsuarios.add(usuarioCorrente.getUsername());

//Cria um ArrayAdapter de Strings
ArrayAdapter<String> itens = new ArrayAdapter<>(
    context: UsuariosActivity.this,
    android.R.layout.simple_list_item_1,
    nomesUsuarios);

//adiciona o adapter povoado com nomes ao listView listaUsuarios
listaUsuarios.setAdapter(itens);

listaUsuarios.setOnItemClickListener(this);
```

Usuários

- Agora no método `onOptionsItemSelected()`
- Quando o item clicado for a opção sair:
 - ◆ Realize o logout
 - ◆ Adicione a intenção de abrir o login
 - ◆ Inicie a intenção
 - ◆ Finalize essa activity

```
//Menu
@Override
public boolean onOptionsItemSelected(MenuItem item) {
    //Verifica qual é o id do item clicado
    if(item.getItemId() == R.id.editar){
        Log.e(tag: "Clicou :", msg: "editar");
    }

    if(item.getItemId() == R.id.sair){
        Log.e(tag: "Clicou :", msg: "sair");
        ParseUser.logout();
        Intent intent = new Intent(
            packageContext: UsuariosActivity.this,
            LoginActivity.class);
        startActivity(intent);
        finish();
    }

    return true;
}
```

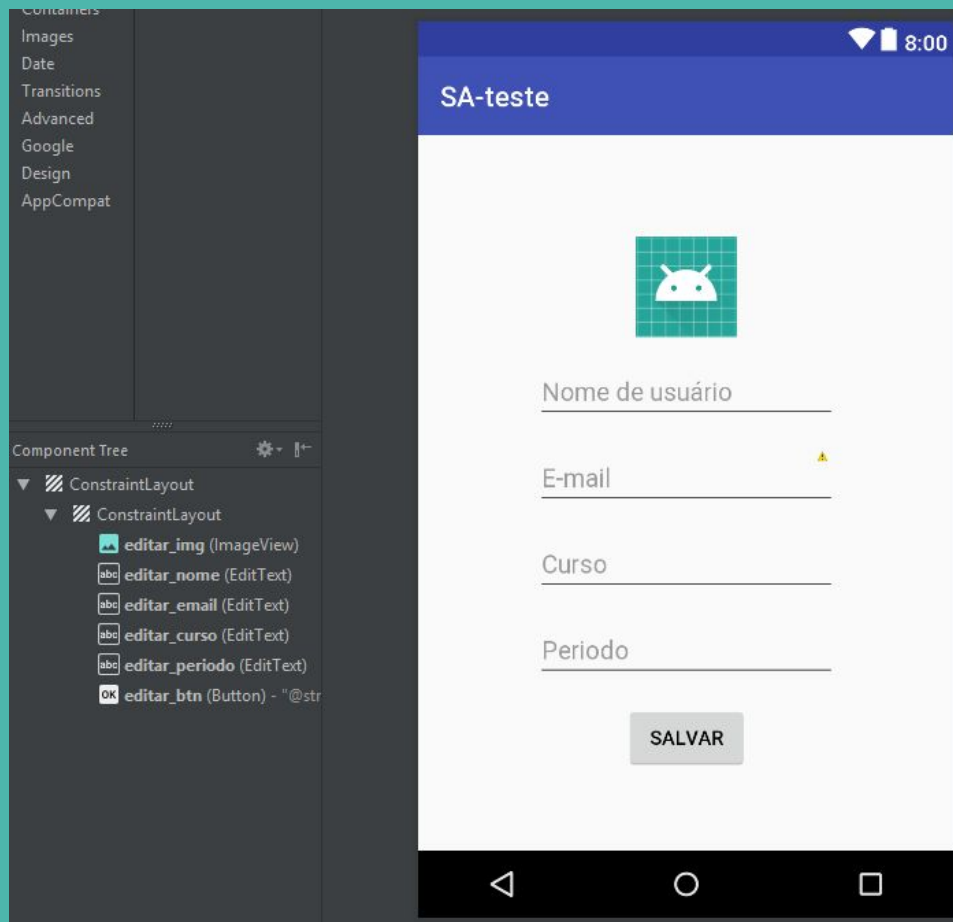
Usuários

- No arquivo usuarioDAO, adicione o método alterar, que retorna boolean e recebe como parâmetro um usuário.
- Recupere um objeto ParseUser do usuário corrente, atualize os seus atributos.
- Salve o usuario e retorne true
- Retorne false caso seja disparada uma exceção

```
public static boolean alterar(Usuario usuario) {  
    try {  
        //Recupera o usuario corrente  
        ParseUser parseUser = ParseUser.getCurrentUser();  
  
        //Setando os atributos  
        parseUser.setUsername(usuario.getUsername());  
        parseUser.setEmail(usuario.getEmail());  
        parseUser.put("curso", usuario.getCurso());  
        parseUser.put("periodo", usuario.getPeriodo());  
  
        //Salvando  
        parseUser.save();  
        return true;  
    } catch (ParseException e) {  
        e.printStackTrace();  
    }  
  
    return false;  
}
```

Editar Usuário

- File > New > Activity > Empty Activity
- Dê o nome de EditarActivity
- Abra o arquivo layout/activity_editar.xml
- Insira uma imagem e os campos como na figura ao lado
- Edite os ids dos elementos para identificarmos na classe controladora



Editar Usuário

- Volte ao arquivo UsuariosActivity.java
- Agora, quando o usuário clicar em Editar
 - ◆ Adicione a intenção de abrir a activity EditarActivity
 - ◆ Inicie a activity

```
//Menu
@Override
public boolean onOptionsItemSelected(MenuItem item) {

    //Verifica qual é o id do item clicado

    if(item.getItemId() == R.id.editar){
        Intent intent = new Intent(
            packageContext: UsuariosActivity.this,
            EditarActivity.class);
        startActivity(intent);
        Log.e(tag: "Clicou :", msg: "editar");
    }

    if(item.getItemId() == R.id.sair){
        Log.e(tag: "Clicou :", msg: "sair");
        ParseUser.logOut();
        Intent intent = new Intent(
            packageContext: UsuariosActivity.this,
            LoginActivity.class);
        startActivity(intent);
        finish();
    }

    return true;
}
```

Editar Usuário

- Referencie todos os elementos criados no xml desta Activity.
- Recupere uma instância de Usuario com o usuário corrente através do método `getCurrentUser` da classe `UsuarioDAO`
- Essa activity não possui um Toolbar, então para mudarmos o Título, podemos recuperar a `ActionBar`. Que por padrão está ativa neste tema.

```
@Override
protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_editar);

    //Encontrando as views
    editarNome = findViewById(R.id.editar_nome);
    editarEmail = findViewById(R.id.editar_email);
    editarCurso = findViewById(R.id.editar_curso);
    editarPeriodo = findViewById(R.id.editar_periodo);
    editarBtn = findViewById(R.id.editar_btn);

    //Recuperando o usuario corrente
    usuario = UsuarioDAO.getCurrentUser();

    //Alterando o titulo do toolbar
    ActionBar actionBar = getSupportActionBar();
    actionBar.setTitle(usuario.getUsername());
}
```

Editar Usuários

- Preencha os EditText com os dados do usuário recuperado.
- Implemente a interface OnClickListener e sobrescreva o método onClick().
- No método onCreate, altere a referência do btnEditar.SetOnClickListener para esta activity, passando *this* como parâmetro.

```
public class EditarActivity extends AppCompatActivity
    implements View.OnClickListener {

    private EditText editarNome, editarEmail, editarCurso, editarPeriodo;
    private Button editarBtn;
    private Usuario usuario;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_editar);

        editarNome = findViewById(R.id.editar_nome);
        editarEmail = findViewById(R.id.editar_email);
        editarCurso = findViewById(R.id.editar_curso);
        editarPeriodo = findViewById(R.id.editar_periodo);
        editarBtn = findViewById(R.id.editar_btn);

        usuario = UsuarioDAO.getCurrentUser();

        ActionBar actionBar = getSupportActionBar();
        actionBar.setTitle(usuario.getUsername());

        //Preenchendo os campos
        editarNome.setText(usuario.getUsername());
        editarEmail.setText(usuario.getEmail());
        editarCurso.setText(usuario.getCurso());
        editarPeriodo.setText(String.valueOf(usuario.getPeriodo()));


        //Alterando ação do botão
        editarBtn.setOnClickListener(this);
    }
}
```


Editar Usuários

- No método OnClick, sobrecarregado, atualize os atributos do usuario com os valores dos respectivos campos.
- Utilize o método alterar da classe UsuarioDAO, passando o usuario atualizado.
- Exiba um Toast de acordo com o retorno do método.
- Caso o retorno seja true, finalize esta activity.

leo

0.05 / 0.06 / 0.09
mediaserver
surfaceflinger
com.example.taril.sa_teste
local_opengl
system_server
com.android.systemui
com.android.phone



leo

leu@emaiu.com

TSI

4

SALVAR

Informações alteradas

Mensagem

- No pacote modelo crie a classe Mensagem, com os atributos:
 - ◆ String id, conteudo
 - ◆ ParseObject parseObject
 - ◆ ParseUser remetente
 - ◆ ParseObject pai
 - ◆ Implemente a interface Parcelable
- Iremos usar os objetos Parse(ParseObject e ParseUser) para criar ponteiros nas tabelas
- O atributo pai indica se essa mensagem é resposta de alguma mensagem ou não

```
public class Mensagem implements Parcelable{
    private String id;

    /**
     * Parseobject referente ao objeto
     */
    private ParseObject parseObject;

    private String conteudo;
    private ParseUser remetente;
    private ParseObject pai;

    public Mensagem() { }

    public Mensagem(ParseObject parseObject, ParseObject pai){
        this.parseObject = parseObject;

        conteudo = parseObject.getString( key: "conteudo");
        remetente = parseObject.getParseUser( key: "remetente");
        this.pai = pai;
    }
}
```

Mensagem

- Ainda na classe mensagem crie um construtor padrão e um sobrecarregado que receba dois parseObject
- Crie também os métodos get/set
- Agora crie a classe MensagemDAO com um método para inserir uma mensagem e um método para retornar todas as mensagens com determinado pai

```
MensagemDAO

public static ParseObject inserir(Mensagem mensagem) {
    //Cria um objeto para a tabela mensagem
    ParseObject object = new ParseObject( theClassName: "Mensagem");

    //Insere as informações no objeto
    object.put("conteudo",mensagem.getConteudo());
    if(mensagem.getPai() != null)
        object.put("pai",mensagem.getPai());
    object.put("remetente",mensagem.getRemetente());

    try { //Salva o objeto
        object.save();
        return object.fetch();
    } catch (ParseException e) { e.printStackTrace(); }
    return null;
}

///Busca todas as mensagens cujo o pai seja igual ao parametro
public static List<Mensagem> getAll(Mensagem msgPai) {
    List<Mensagem> mensagens = new ArrayList<>();

    //Buscando a query da tabela 'Mensagem'
    ParseQuery<ParseObject> query = ParseQuery.getQuery("Mensagem");

    //Informando à query para incluir os dados da coluna 'remetente'
    // que é um ponteiro para um User
    query.include("remetente");

    //Filtrando a busca para só retornar onde a coluna pai seja igual ao objeto
    query.equalTo("pai", msgPai.getParseObject());

    try {
        //Buscando os resultados para a query
        List<ParseObject> parseList = query.find();

        //Convertendo a lista de ParseObject para um lista de Mensagem
        for(ParseObject obj : parseList)
            mensagens.add(new Mensagem(obj,msgPai.getParseObject()));
    } catch (ParseException e) { e.printStackTrace(); }
    return mensagens;
}
```

Posts

- Vamos substituir nossa lista de usuários agora por uma lista de posts, aprofundando um pouco nossas consultas na API.
- Comente as linhas dentro do método onCreate, como ao lado
- Renomeie(Refatorando) os arquivos
 - ◆ UsuariosActivity.java
 - ◆ activity_usuarios.xml
 - ◆ content_usuarios.xml
- Substitua “usuarios” por “posts”
- renomeie as referências dentro dos arquivos também

```
protected void onCreate(Bundle savedInstanceState) {  
    super.onCreate(savedInstanceState);  
    setContentView(R.layout.activity_posts);  
  
    Usuario usuario = UsuarioDAO.getCurrentUser();  
  
    //Cria um objeto da classe Toolbar para referenciar a toolbar nesta activity  
    Toolbar toolbar = (Toolbar) findViewById(R.id.toolbar);  
    toolbar.setTitle(usuario.getUsername());  
    setSupportActionBar(toolbar);  
  
    listaPosts = findViewById(R.id.lista_posts);  
  
    /*  
  
    //referencia a lista no arquivo xml  
    listaUsuarios = findViewById(R.id.lista_usuarios);  
  
    //Cria uma lista de usuarios com os usuarios do banco de dados  
    List<Usuario> usuarios = UsuarioDAO.listaUsuarios();  
  
    //Cria uma lista de nomes a partir da lista de usuarios  
    List<String> nomesUsuarios = new ArrayList<>();  
    if(usuarios != null)  
        for (Usuario usuarioCorrente : usuarios)  
            nomesUsuarios.add(usuarioCorrente.getUsername());  
  
    //Cria um ArrayAdapter de Strings  
    ArrayAdapter<String> itens = new ArrayAdapter<>(  
        UsuariosActivity.this,  
        android.R.layout.simple_list_item_1,  
        nomesUsuarios);  
  
    //adiciona o adapter povoado com nomes ao listView listaUsuarios  
    listaUsuarios.setAdapter(itens);  
  
    listaUsuarios.setOnItemClickListener(this);  
  
    */  
}
```

Posts

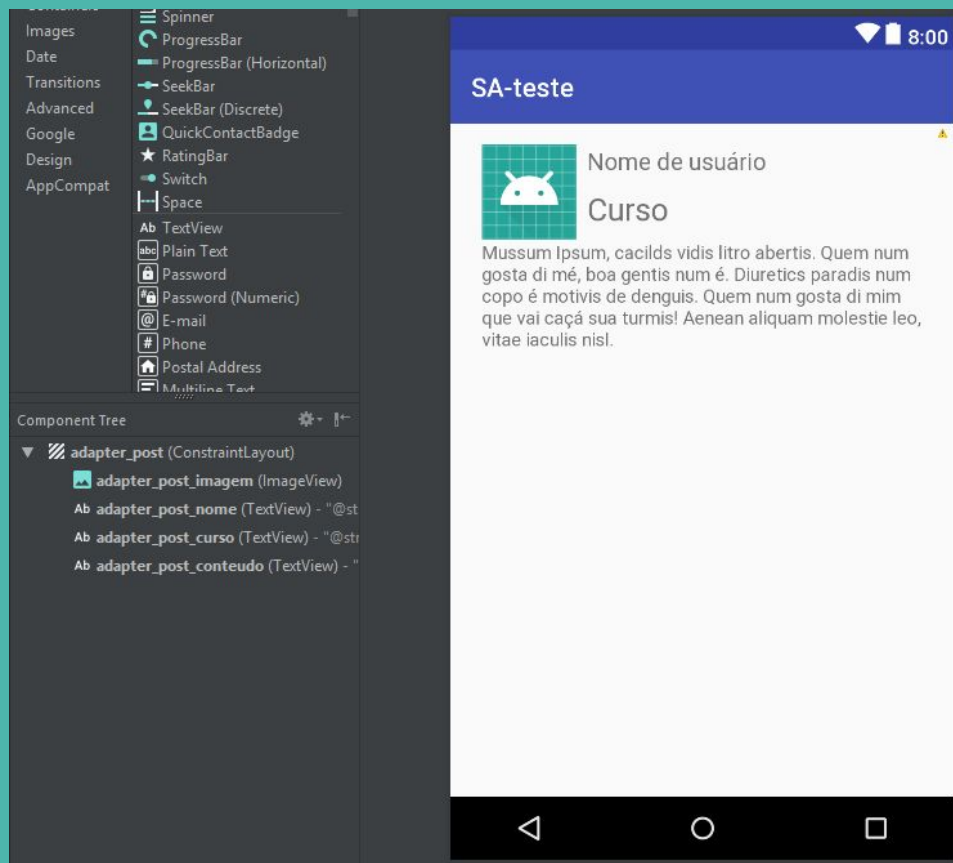
- Vá até seu dashboard no site do Back4App e crie manualmente uma tabela mensagem
- Adicione as mesmas colunas do modelo Mensagem, porém nas linhas adicionadas deixe o ponteiro pai vazio

MiniCurso2018		CLASS Mensagem 10 objects • Public Read and Write enabled					
Core		objectId String	createdAt Date	updatedAt Date	ACL ACL	remetente Pointer ↩	conteudo String
Browser Installation 7 Role 0 Session 10 User 7 Mensagem 10 Webhooks Jobs Logs Config API Console	Create a class	<input type="checkbox"/> W8BD3djmzp	9 Oct 2018 at 20:4...	9 Oct 2018 at 20:4...	Public Read + Write	01kCQJabi2	dont let me down
		<input type="checkbox"/> 9bhc5U0W4o	9 Oct 2018 at 20:4...	9 Oct 2018 at 20:4...	Public Read + Write	01kCQJabi2	dont let me down
		<input type="checkbox"/> EDIosQ9wLj	9 Oct 2018 at 20:3...	9 Oct 2018 at 20:3...	Public Read + Write	01kCQJabi2	dont let me down
		<input type="checkbox"/> bTnwABUYzU	9 Oct 2018 at 20:3...	9 Oct 2018 at 20:3...	Public Read + Write	01kCQJabi2	dont let me down
		<input type="checkbox"/> GXl6CP1Fi7	9 Oct 2018 at 20:2...	9 Oct 2018 at 20:2...	Public Read + Write	01kCQJabi2	dont let me down
		<input type="checkbox"/> HY0JgG10vC	9 Oct 2018 at 20:2...	9 Oct 2018 at 20:2...	Public Read + Write	01kCQJabi2	dont let me down
		<input type="checkbox"/> W0teQb59JS	9 Oct 2018 at 20:2...	9 Oct 2018 at 20:2...	Public Read + Write	01kCQJabi2	dont let me down
		<input type="checkbox"/> dPVG17vMUu	9 Oct 2018 at 20:2...	9 Oct 2018 at 20:2...	Public Read + Write	01kCQJabi2	dont let me down
		<input type="checkbox"/> NXFBdAEbzW	9 Oct 2018 at 20:2...	9 Oct 2018 at 20:2...	Public Read + Write	01kCQJabi2	dont let me down
		<input type="checkbox"/> AWbhMfdav	9 Oct 2018 at 20:2...	9 Oct 2018 at 20:2...	Public Read + Write	01kCQJabi2	hey jude

Posts

→ Vamos trocar o adapter da nossa listView, de String para um customizado

- ◆ Crie o arquivo adapter_mensagem.xml
- ◆ Vamos exibir o Nome de Usuario, o Curso e o Conteúdo da mensagem atribua um ID para recuperarmos o recurso no codigo java



Posts

- Crie um arquivo java MensagemAdapter que estende ArrayAdapter<Mensagem> para controlar o arquivo xml que acabamos de criar
- Crie um construtor que recebe o Context e um ArrayList<Mensagem> e passe-os para a super classe, junto com recurso(R.layout.adapter_mensagem.xml)
- Sobrescreva o método getView

```
public class MensagemAdapter extends ArrayAdapter<Mensagem>{  
  
    public MensagemAdapter(Context context, List<Mensagem> mensagens){  
        super(context, R.layout.adapter_mensagem, mensagens);  
    }  
  
    @NonNull  
    @Override  
    public View getView(int position, @Nullable View convertView,  
                        @NonNull ViewGroup parent) {  
  
        return convertView;  
    }  
}
```

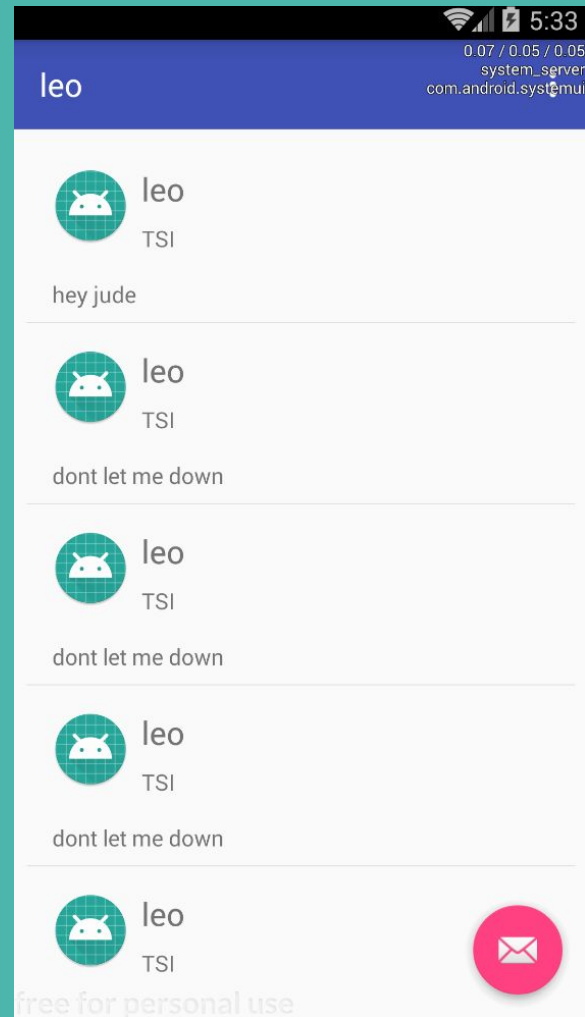
Posts

- Dentro do método getView:
 - ◆ Infle a view, passando o retorno do método inflater.inflate(...) para o convertView que recebeu nos parâmetros desse método
 - ◆ Referencie os componentes da convertView as variáveis que declaramos
 - ◆ Altere o valor através do método setText(...)
- A View retornada será adicionada ao ArrayAdapter

```
public class MensagemAdapter extends ArrayAdapter<Mensagem>{  
  
    ImageView adapterImagem;  
    TextView adapterNome, adapterCurso, adapterConteudo;  
  
    public MensagemAdapter(Context context, List<Mensagem> mensagens) {  
        super(context, R.layout.adapter_mensagem, mensagens);  
    }  
  
    @NonNull  
    @Override  
    public View getView(int position, @Nullable View convertView, @NonNull ViewGroup parent) {  
  
        //Infla a view  
        LayoutInflater inflater = LayoutInflater.from(getContext());  
        convertView = inflater.inflate(R.layout.adapter_mensagem, parent, attachToRoot: false);  
  
        //Recupera o conteudo da view  
        adapterImagem = convertView.findViewById(R.id.adapter_post_imagem);  
        adapterNome = convertView.findViewById(R.id.adapter_post_nome);  
        adapterCurso = convertView.findViewById(R.id.adapter_post_curso);  
        adapterConteudo = convertView.findViewById(R.id.adapter_post_conteudo);  
  
        //Altera os atributos da view  
        Usuario remetente = new Usuario(getItem(position).getRemetente());  
        adapterNome.setText(remetente.getUsername());  
        adapterCurso.setText(remetente.getCurso());  
        adapterConteudo.setText(getItem(position).getConteudo());  
  
        //Retorna a view pronta  
        return convertView;  
    }  
}
```


Posts

- Na classe agora nomeada PostsActivity
- ◆ Mude o nome da variável ListView para listaPosts
 - ◆ Busque uma lista de mensagens usando o MensagemDAO
 - ◆ Crie uma referência do MensagemAdapter implementado
 - ◆ Set o adapter do list view
 - ◆ Set o onClickListener do listview



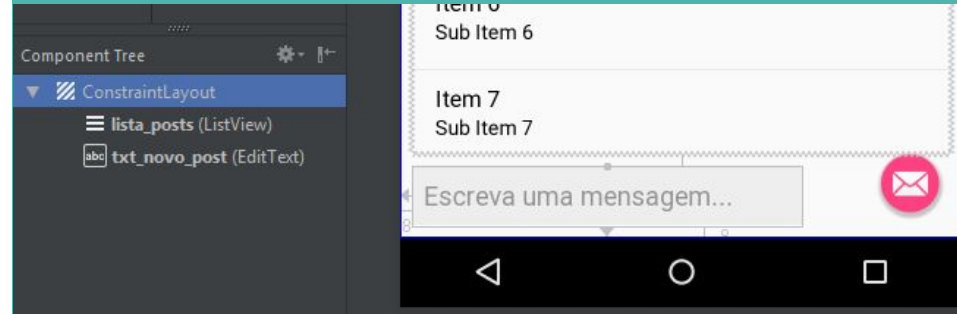
Posts

→ No arquivo content_posts.xml

- ◆ Adicione um MultilineText
- ◆ Link ao final da lista
- ◆ Adicione um hint
- ◆ Adicione um id

→ Na classe PostsActivity

- ◆ Implemente a interface OnClickListener
- ◆ Declare o EditText e o FloatingActionButton adicionado ao xml
- ◆ Set o onClickListener do BtnEnviar para essa classe



```
public class PostsActivity extends AppCompatActivity
    implements AdapterView.OnItemClickListener,
        View.OnClickListener{

    private ListView listaPosts;
    private FloatingActionButton BtnEnviar;
    private EditText txtNovoPost;
```

Posts

→ Sobrescreva o método onClick()

- ◆ Recupere o conteúdo do EditText
- ◆ Monte um objeto Mensagem
- ◆ Insira a mensagem no banco de dados através do MensagemDAO
- ◆ Exiba um Toast informando que a mensagem foi enviada
- ◆ Altere o conteúdo do EditText para vazio

```
@Override
public void onClick(View v) {

    String novoPost = txtNovoPost.getText().toString();

    Mensagem mensagem = new Mensagem();
    mensagem.setRemetente(ParseUser.getCurrentUser());
    mensagem.setConteudo(novoPost);

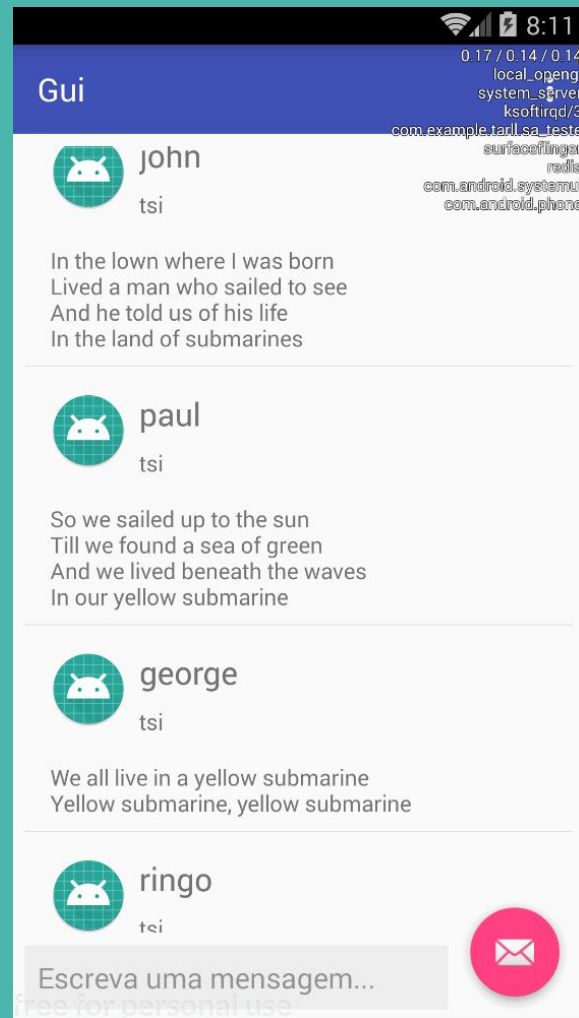
    MensagemDAO.inserir(mensagem);

    Toast.makeText(context, this, text: "Post enviado!!!",
        Toast.LENGTH_LONG).show();

    txtNovoPost.setText("");
}
```

Posts

- Transforme as variáveis locais abaixo em variáveis de classe
 - ◆ ArrayAdapter<Mensagem> itens
 - ◆ List<Mensagem> mensagens
- No método onClick()
 - ◆ Adicione um if para verificar se o campo está vazio



Posts

- Ainda no mesmo método
 - ◆ Através do retorno do método inserir verifique se a mensagem foi enviada
- Se a mensagem foi enviada adicione a mensagem na lista
 - ◆ Notifique que uma alteração ocorreu
 - ◆ Mova o ListView para a última posição.
- Se a mensagem não foi enviada apenas exiba um Toast com a informação

```
@Override
public void onClick(View v) {

    String novoPost = txtNovoPost.getText().toString();

    if(novoPost.isEmpty())
        return;

    Mensagem mensagem = new Mensagem();
    mensagem.setRemetente(ParseUser.getCurrentUser());
    mensagem.setConteudo(novoPost);

    ParseObject parseMsg = MensagemDAO.inserir(mensagem);
    if(parseMsg != null) {
        mensagem.setParseObject(parseMsg);
        mensagens.add(mensagem);
        itens.notifyDataSetChanged();
        listaPosts.smoothScrollToPosition(mensagens.size());

        Toast.makeText(context, this, text: "Post enviado!!!",
            Toast.LENGTH_LONG).show();

        txtNovoPost.setText("");
    }else
        Toast.makeText(context, this, text: "Post não enviado!!!",
            Toast.LENGTH_LONG).show();
}
```

Posts

→ Em LoginActivity no método onCreate() antes de iniciarmos a nova activity

- ◆ Cria uma instância de Mensagem
- ◆ Inclua essa mensagem como um extra na intenção

→ Faça o mesmo no método onClick() quando o login é realizado com sucesso

```
LoginActivity

@Override
protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    Parse.initialize(context, this);

    try {
        ParseInstallation.getCurrentInstallation().save();
        Log.e(tag: "PARSER: ", msg: "Ok");
        setContentView(R.layout.activity_login);

        if(ParseUser.getCurrentUser() != null){
            Intent intent = new Intent(packageContext, this,
                                     PostsActivity.class);

            //Criando uma mensagem
            Mensagem msg = new Mensagem();

            //Adicionando extras à intenção
            intent.putExtra(name: "MSG_PAI", msg);

            startActivity(intent);
            finish();
        }
    } catch (ParseException e) {
        Log.e(tag: "PARSER: ", msg: "Erro");
        finish();
    }

    btn_entrar = findViewById(R.id.btnEntrar);
    nomeUsuario = findViewById(R.id.txtUsuario);
    senha = findViewById(R.id.txtSenha);

    btn_entrar.setOnClickListener(this);
}
```

Posts

→ Em PostsActivity altere a variável msg que está dentro do método onCreate(), para:

- ◆ Variável de classe
- ◆ Nome: msgPai

→ Verifique se a intenção possui um extra

→ Verifique se o extra possui a chave MSG_PAÍ

→ Se sim recupero o extra

→ Se não instancie uma Mensagem

```
PostsActivity onCreate()  
  
//Cria uma mensagem 'nula'  
Mensagem msg = new Mensagem();  
msg.setParseObject(null);
```

```
PostsActivity onCreate()  
  
private Mensagem msgPai;  
  
@Override  
protected void onCreate(Bundle savedInstanceState) {  
    super.onCreate(savedInstanceState);  
    setContentView(R.layout.activity_posts);  
  
    //Buscando extras na intenção  
    Bundle extras = getIntent().getExtras();  
  
    //Verificando e buscando o extra caso possua  
    if(extras != null && getIntent().hasExtra( name: "MSG_PAÍ")){  
        msgPai = (Mensagem) extras.getSerializable( key: "MSG_PAÍ");  
    }else  
        msgPai = new Mensagem();  
  
    Usuario usuario = UsuarioDAO.getCurrentUser();
```

Posts

- No método `setOnClick` crie uma intenção para abrir uma nova activity, passando este contexto e esta classe
- Adicione no extra "MSG_PAI" o `parseObject` da mensagem clicada.
 - ◆ A posição da mensagem na lista de mensagem é a mesma posição no `ArrayAdapter`

```
PostsActivity onItemClick()  
  
//Lista de usuarios  
@Override  
public void onItemClick(AdapterView<?> parent, View view, int position, long id) {  
    Log.e( tag: "Item: ", String.valueOf(position));  
  
    Intent intent = new Intent( packageContext: this, PostsActivity.class);  
  
    //Adicionando extras à intenção  
    intent.putExtra( name: "MSG_PAI", mensagens.get(position));  
  
    startActivity(intent);  
}
```

Posts

- No método onClick antes de salvar a mensagem, passe o parseObject da mensagem atual como o pai da mensagem que será criada.

Assim essa mensagem será como uma resposta a mensagem principal(pai) dessa activity

```
@Override
public void onClick(View v) {

    String novoPost = txtNovoPost.getText().toString();

    if(novoPost.isEmpty())
        return;

    Mensagem mensagem = new Mensagem();
    mensagem.setRemetente(ParseUser.getCurrentUser());
    mensagem.setConteudo(novoPost);

    //Transformando o msgPai em pai dessa resposta
    mensagem.setPai(msgPai.getParseObject());

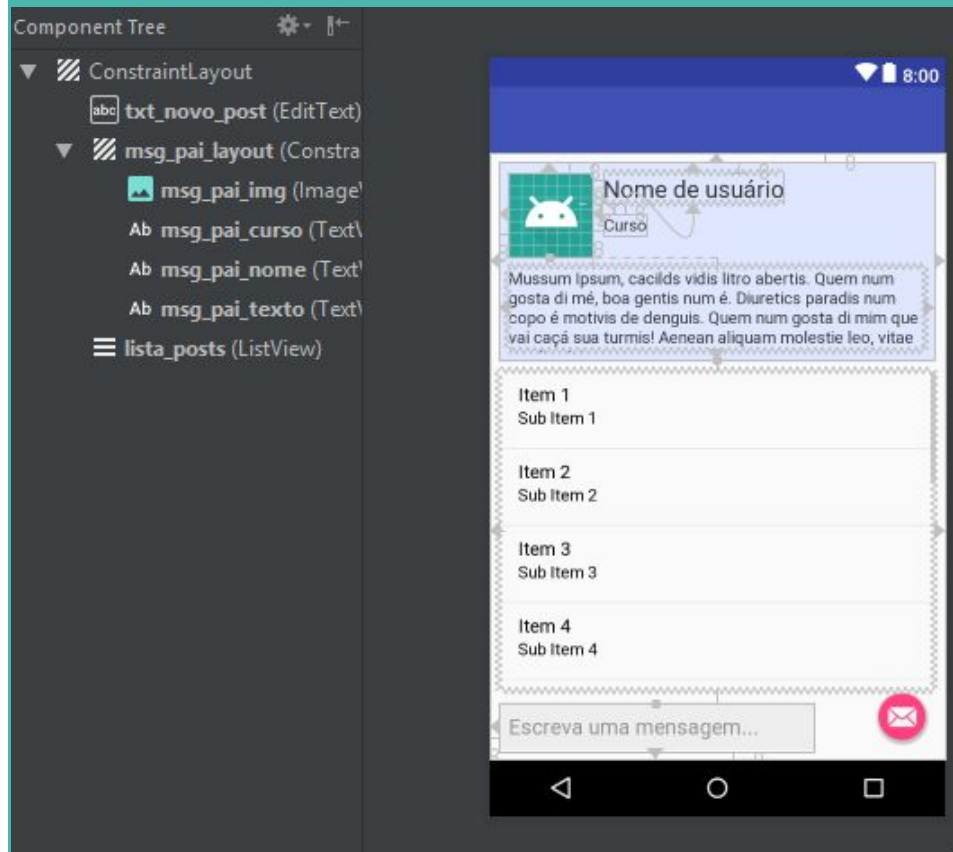
    ParseObject parseMsg = MensagemDAO.inserir(mensagem);
    if(parseMsg != null) {
        mensagem.setParseObject(parseMsg);
        mensagens.add(mensagem);
        itens.notifyDataSetChanged();
        listaPosts.smoothScrollToPosition(mensagens.size());

        Toast.makeText(context, this, text: "Post enviado!!!",
            Toast.LENGTH_LONG).show();

        txtNovoPost.setText("");
    }else
        Toast.makeText(context, this, text: "Post não enviado!!!",
            Toast.LENGTH_LONG).show();
}
```


Posts

- Em content_post.xml adicione um ConstraintLayout no topo da activity.
 - ◆ Dê uma cor de fundo de sua preferência
- Copie os componentes de adapter_mensagem.xml para dentro do layout supracitado
 - ◆ Reposicione-os
 - ◆ Altere os id's



Posts

- Em PostsActivity.java, declare os componentes adicionados
- Se a mensagem principal dessa activity tiver um conteúdo encontre suas referências e atualize o conteúdo dos componentes para seus respectivos valores
- Se não tiver encontre a referência do ConstraintLayout e altere sua visibilidade para View.GONE

```
PostsActivity onCreate()

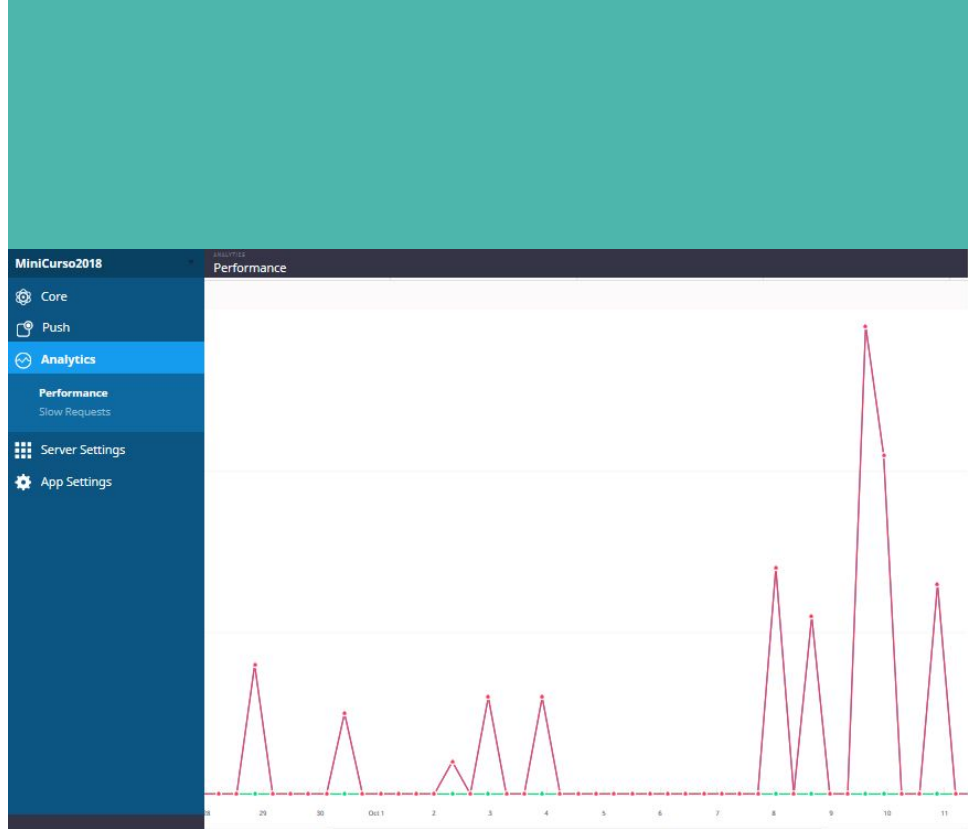
    BtnEnviar = findViewById(R.id.btn_enviar);
    txtNovoPost = findViewById(R.id.txt_novo_post);
    BtnEnviar.setOnClickListener(this);

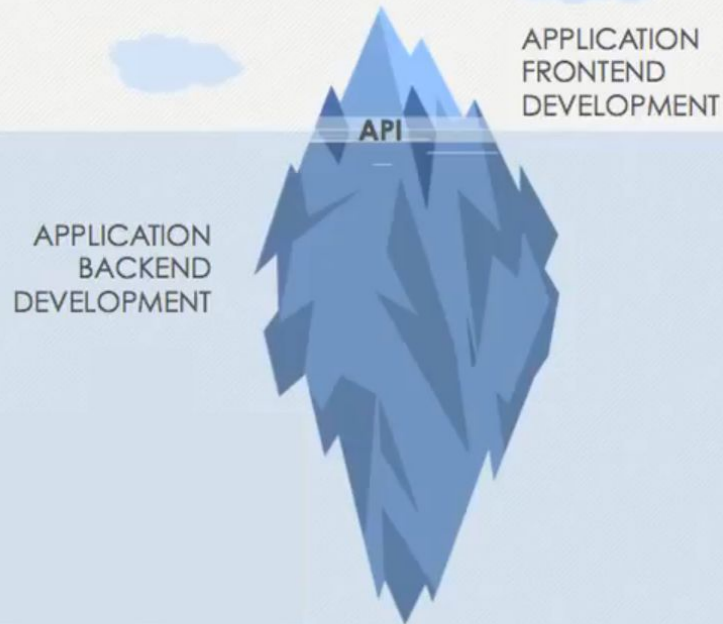
    if(msgPai.getConteudo() != null) {
        msgPaiImg = findViewById(R.id.msg_pai_img);
        msgPaiNome = findViewById(R.id.msg_pai_nome);
        msgPaiCurso = findViewById(R.id.msg_pai_curso);
        msgPaiTexto = findViewById(R.id.msg_pai_texto);

        Usuario remetente = new Usuario(msgPai.getRemetente());
        msgPaiNome.setText(remetente.getUsername());
        msgPaiCurso.setText(remetente.getCurso());
        msgPaiTexto.setText(msgPai.getConteudo());
    } else {
        msgPaiLayout = findViewById(R.id.msg_pai_layout);
        msgPaiLayout.setVisibility(View.GONE);
    }
}
```

Mais

- Parabéns você acaba de desenvolver um aplicativo de fórum para Android utilizando o back-end como um serviço.
- Agora podemos voltar ao site e observarmos informações como a quantidade de requisições realizadas durante os testes.
- Podemos também visualizar serviços como: Login através do Facebook/Gmail. Validação de e-mail, cloud functions, etc.





Leonardo M. R. Carvalho

Tarlles Roman