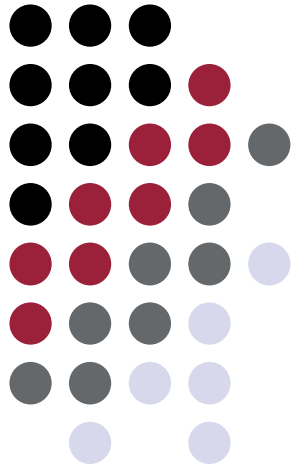
A07 Persistência de Dados

CSI436 – Computação Móvel



Prof. Dr. George H. G. Fonseca
Universidade Federal de Ouro Preto

Introdução

- Há variadas formas de persistir dados no Android
  - Shared Preferences
    - Armazena dados primitivos em pares chave-valor
  - Armazenamento Interno
    - Armazena dados em arquivos internos no dispositivo
  - SQLite
    - Armazena e acessa dados pelo SQLite
  - NoSQL database
    - Usualmente armazena dados na nuvem e mantem um cache no dispositivo (ex.: Firebase)

Shared Preferences

- Persiste um dado primitivo java na aplicação
  - Ex. de uso: boolean indicando se é o primeiro uso do app para mostrar tutorial ou não
  - Recupera um valor (de id “text”) salvo como SharedPreference:

```
SharedPreferences pref = getSharedPreferences(NOME, MODE_PRIVATE);
String text = pref.getString("text", "");
```

Valor padrão caso não haja nenhum valor salvo ainda

  - Salva valor (String text) como SharedPreference sob o id “text”

```
SharedPreferences pref = getSharedPreferences(NOME, MODE_PRIVATE);
SharedPreferences.Editor editor = pref.edit();

editor.putString("text", text);
editor.commit();
```

Armazenamento Interno

- Criação/escrita:

```
FileOutputStream fos;
try {
    fos = this.openFileOutput("t.tmp", Context.MODE_PRIVATE);
    ObjectOutputStream oos = new ObjectOutputStream(fos);
    oos.writeObject(alunos);
    oos.close();
} catch (Exception e) {
    e.printStackTrace();
}
```

Abre/cria arquivo com nome t.tmp.
O segundo parâmetro é o modo de escrita:
- MODE_PRIVATE apaga os dados anteriores
- MODE_APPEND adiciona dados ao arquivo

writeObject escreve qualquer tipo de objeto.
Há varias outras possibilidades:
writeInt
writeString
writeDouble
...

Objeto tem que implementar a interface Serializable!

Armazenamento Interno

- Leitura

```
FileInputStream fis;
try {
    fis = this.openFileInput("t.tmp");
    ObjectInputStream ois = new ObjectInputStream(fis);
    alunos = (ArrayList<Aluno>) ois.readObject();
    ois.close();
} catch (Exception e) {
    e.printStackTrace();
}
```

Abre abre o arquivo t.tmp para leitura.

Lê qualquer tipo de objeto (fazer cast).
Analogamente, tipos nativos também podem ser lidos:

- readInt
- readString
- readDouble

SQLite

- O SGBD SQLite permite aproveitar de todas as características de um Banco de Dados tradicional de aplicações desktop

- Pode não parecer, mas é extremamente simples usá-lo no Android =)

- No exemplo a seguir são persistidos os dados dos alunos do projeto StudentManager
  - Você precisa mexer apenas na classe DatabaseController para criar um banco de dados diferente e outras SQLs
  - A classe SQLiteHelper não precisa ser alterada

SQLite

- A classe DatabaseController e criada como Singleton para ser facilmente acessada de qualquer parte do projeto e manter uma referência única ao BD

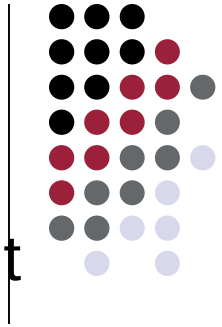
```
public static DatabaseController controller = null;

public static DatabaseController getInstance(Context context) {
    if(controller == null) {
        controller = new DatabaseController(context);
    }
    return controller;
}
```

- Scripts de criação e deleção do banco de dados

```
private static final String SCRIPT_DATABASE_DELETE = "DROP TABLE IF EXISTS student";

private static final String[] SCRIPT_DATABASE_CREATE = new String[]{
        "create table if not exists student(_id integer primary key autoincrement, " +
                "idMat integer not null, name text not null, course text not null, " +
                "period integer not null, coefficient double not null, phone text not null);",
//              "insert into student(idMat, name, course, period, coefficient, phone) " +
//                   "values(1, 'George Fonseca', 'SJM', 6, 9.3, '988189164');",
};
```



- Função para inserir um novo registro na tabela student

```
public long insert(Student student) {
    ContentValues values = new ContentValues();
    values.put("idMat", student.getIdMat());
    values.put("name", student.getName());
    values.put("course", student.getCourse());
    values.put("period", student.getPeriod());
    values.put("coefficient", student.getCoefficient());
    values.put("phone", student.getPhone());
    long id = db.insert("student", "", values);
    Log.i("CSI436", "Data inserted " + id);
    return id;
}
```

- Poderia ser feita via SQL puro:

```
public long insert(Student student) {
    String sql = "insert into student(idMat, name, course, period, coefficient) " +
        "values(" + student.getIdMat() + ", '" + student.getName() + "', '" +
        student.getCourse() + "', " + student.getPeriod() + ", " +
        student.getCoefficient() + ");";
    db.execSQL(sql);
    return 1;
}
```

- Função para atualizar um registro na tabela student

```
public int update(Student student) {
    ContentValues values = new ContentValues();
    values.put("idMat", student.getIdMat());
    values.put("name", student.getName());
    values.put("course", student.getCourse());
    values.put("period", student.getPeriod());
    values.put("coefficient", student.getCoefficient());
    String _id = String.valueOf(student.getId());
    String where = "_id=?";
    String[] whereArgs = new String[] {_id};
    int count = db.update("student", values, where, whereArgs);
    Log.i("CSI436", "Updated " + count + " records.");
    return count;
}
```

- Via SQL puro:

```
public int update(Student student) {
    String sql = "update student set idMat = " +student.getIdMat()
            + ", name = '" +student.getName()
            + "', course = '" + student.getCourse()
            + "', period = " + student.getPeriod()
            + ", coefficient = " + student.getCoefficient()
            + ", phone = '" + student.getPhone()
            + "' where _id = " + student.getId() + ";";
    db.execSQL(sql);
    return 1;
}
```

- Função para deletar um registro da tabela student

```
public int delete(Student student) {
    String _id = String.valueOf(student.getId());
    String where = "_id=?";
    String[] whereArgs = new String[] {_id};
    int count = db.delete("student", where, whereArgs);
    Log.i("CSI489", "Deleted " + count + " records.");
    return count;
}
```

- Via SQL puro:

```
public int delete(Student student) {
    String sql = "delete from student where _id = " + student.getId();
    db.execSQL(sql);
    return 1;
}
```

- Função para buscar um registro da tabela student por id

```
public Student searchById(long id) {
    //select * from aluno where _id=id
    Cursor c = db.query(
        "student", //from
        new String[]{"_id","idMat","name","course","period","coefficient","phone"},//select
        "_id=" + id, //where
        null, //selection args (use ?s at where clause e specify the values here)
        null, //group by
        null, //having
        null, //orderby
        null); //limit
    if(c.getCount() > 0) {
        c.moveToFirst();
        long _id = c.getLong(0);
        int idMat = c.getInt(1);
        String name = c.getString(2);
        String course = c.getString(3);
        int period = c.getInt(4);
        double coefficient = c.getDouble(5);
        String phone = c.getString(6);
        return new Student(_id, idMat, name, course, period, coefficient, phone);
    }
    return null;
}
```

- Função para listar todos os registros da tabela student

```
public ArrayList<Student> listStudents() {
    Cursor c = db.query(
        "student", //from
        new String[]{"_id","idMat","name","course","period","coefficient","phone"}, //select
        null, //where
        null, //section arguments (use ?s at where clause and pass values here)
        null, //group by
        null, //having
        null, //orderby
        null); //limit
    ArrayList<Student> students = new ArrayList<Student>();
    if(c.moveToFirst()) {
        do {
            long id = c.getLong(0);
            int idMat = c.getInt(1);
            String name = c.getString(2);
            String course = c.getString(3);
            int period = c.getInt(4);
            double coefficient = c.getDouble(5);
            String phone = c.getString(6);
            students.add(new Student(id, idMat, name, course, period, coefficient, phone));
        } while(c.moveToNext());
    }
    return students;
}
```

- Para acessar essas funções é muito simples via Singleton em qualquer classe do projeto:

```
DatabaseController.getInstance(this).insert(student);

DatabaseController.getInstance(this).update(student);

DatabaseController.getInstance(this).delete(student);

DatabaseController.getInstance(this).searchById(id);

DatabaseController.getInstance(this).listStudents();
```

Firebase

- Firebase é uma ferramenta poderosíssima disponibilizada pela Google
  - Pode ser usada para Android, iOS e Web

- Principais funcionalidades:
  - Analytics
  - Authentication
  - Realtime Database
  - Crash report
  - Admob
  - ...

Firebase

- Para algumas funcionalidades basta seguir o tutorial em Tools → Firebase

- Para outras precisaremos de um pouco de código

Firebase

- Autenticação via conta Google
  - 1. Adicione as dependências de autenticação Firebase ao app.
  - 2. Habilite um provedor de login no console do Firebase
  - 3. Configure as dependências do provedor de login escolhido
  - No caso do Google esse tutorial explica as dependências (você não precisa de arquivo de configuração pois o Firebase já resolve)

    https://developers.google.com/identity/sign-in/android/start-integrating
  - Em caso de dúvidas, veja e replique as dependências dos arquivos gradlle do projeto A08 Student Firebase

Firebase

- Realtime Database
  - 1. Configure as dependências conforme tutorial em Tools → Firebase → Realtime Database
  - 2. Para salvar dados, use:

```
// Write a message to the firebase
firebase = FirebaseDatabase.getInstance();
DatabaseReference myRef = firebase.getReference("message");
myRef.setValue("Hello, World!");
```

- Realtime Database
  - 3. Para carregar dados, use:

```
// Read from the firebase
myRef.addValueEventListener(new ValueEventListener() {
    @Override
    public void onDataChange(DataSnapshot dataSnapshot) {
        // This method is called once with the initial value and again
        // whenever data at this location is updated.
        String value = dataSnapshot.getValue(String.class);
        Log.d("CSI436", "Value is: " + value);
    }

    @Override
    public void onCancelled(DatabaseError error) {
        // Failed to read value
        Log.w("CSI436", "Failed to read value.", error.toException());
    }
});
```

Firebase

- Realtime Database
  - E para operações mais complexas, por exemplo, manipular dados de cada usuário do app?
  - Atributos da class (activity)

```
private DatabaseReference firebase;
private FirebaseUser user;
private FirebaseAuth mAuth;
```

  - Inicializar atributos no onCreate

```
mAuth = FirebaseAuth.getInstance();
auth = mAuth.getCurrentUser();
```

  - Salvar estudante no firebase (dentro do diretório do usuário);

```
private boolean saveStudent(Student student){
    try {
        firebase = FirebaseDatabase.getInstance().getReference()
                .child("user").child(auth.getUid().toString()).child("students");
        firebase.child("" + student.getId()).setValue(student);
        Toast.makeText(this, "Student registered successfully!", Toast.LENGTH_LONG).show();
        return true;
    } catch (Exception e){
        e.printStackTrace();
        return false;
    }
}
```

- Realtime Database
  - Para recuperar lista de students de um usuário específico:
  - Atributos da classe (activity):

```
private DatabaseReference firebase;
private FirebaseUser user;
private ValueEventListener valueEventListener;
private ArrayList<Student> students;
```

- Realtime Database
  - Inicializar atributos no onCreate

```
students = new ArrayList<>();
user = FirebaseAuth.getInstance().getCurrentUser();
firebase = FirebaseDatabase.getInstance().getReference().child("user")
                .child(user.getUid().toString()).child("students");
```

valueEventListener onDataChange é acionado toda vez que algum dado é alterado!

```
valueEventListener = new ValueEventListener() {
    @Override
    public void onDataChange(DataSnapshot dataSnapshot) {
        students.clear();
        for (DataSnapshot data : dataSnapshot.getChildren()){
            Student newStudent = data.getValue(Student.class);
            students.add(newStudent);
        }
        Log.d("CSI436", "valueEventListener Students: " + students);
        adapter.notifyDataSetChanged();
    }

    @Override
    public void onCancelled(DatabaseError databaseError) {

    }
};
```

- Realtime Database
  - No método onStart adicionar o valueEventListener e no onStop removê-lo

```
@Override
protected void onStop() {
    super.onStop();
    firebase.removeEventListener(valueEventListener);
}

@Override
protected void onStart() {
    super.onStart();
    firebase.addValueEventListener(valueEventListener);
}
```

Adiciona o listener à referência do Firebase e executa onDataChange

- Realtime Database
  - Os dados são persistidos off-line? Por padrão não, mas podemos facilmente configurar essa funcionalidade:

  - Adicione a tag name em application no AndroidManifest.xml para uma classe MyApplication a ser criada:

```
<application
    android:allowBackup="true"
    android:icon="@mipmap/ic_launcher"
    android:label="@string/app_name"
    ...
    android:name=".MyApplication">
```

Dessa forma, a classe MyApplication é iniciada antes de qualquer activity, inclusive da Main!

- Realtime Database
  - Os dados são persistidos off-line? Por padrão não, mas podemos facilmente configurar essa funcionalidade:

  - A classe MyApplication simplesmente define a propriedade setPersistenceEnabled para true

```
public class MyApplication extends Application {

    @Override
    public void onCreate() {
        super.onCreate();
        FirebaseDatabase.getInstance().setPersistenceEnabled(true);
    }
}
```

Firebase

- AdMob
  - AdMod é um dos gerenciadores daquelas propagandas (irritantes cof cof) que aparecem nos apps
  - É uma forma de receita a ser explorada
  - Pode ainda diferenciar uma versão Free de uma Premium de um app por exemplo

- AdMob
  - Para integar em um projeto Android:
  - 1. Configure as dependências conforme tutorial em Tools → Firebase → Admob
  - 2. Adicione o banner de propaganda no arquivo de layout:

```
<com.google.android.gms.ads.AdView
    android:id="@+id/adView"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_centerHorizontal="true"
    ads:adSize="BANNER"
    ads:adUnitId="@string/banner_ad_unit_id">
</com.google.android.gms.ads.AdView>
```

O Firebase escolhe automaticamente as propagandas mais relevantes e as que se ajustam melhor ao espaço disponibilizado no layout!

- AdMob
  - 3. Carregue o Admob na activity em que será exibido:

```
//Load adMob
mAdView = (AdView) findViewById(R.id.adView);
AdRequest adRequest = new AdRequest.Builder().build();
mAdView.loadAd(adRequest);
```

  - 4. Adicione o seu Ad Unit ID:

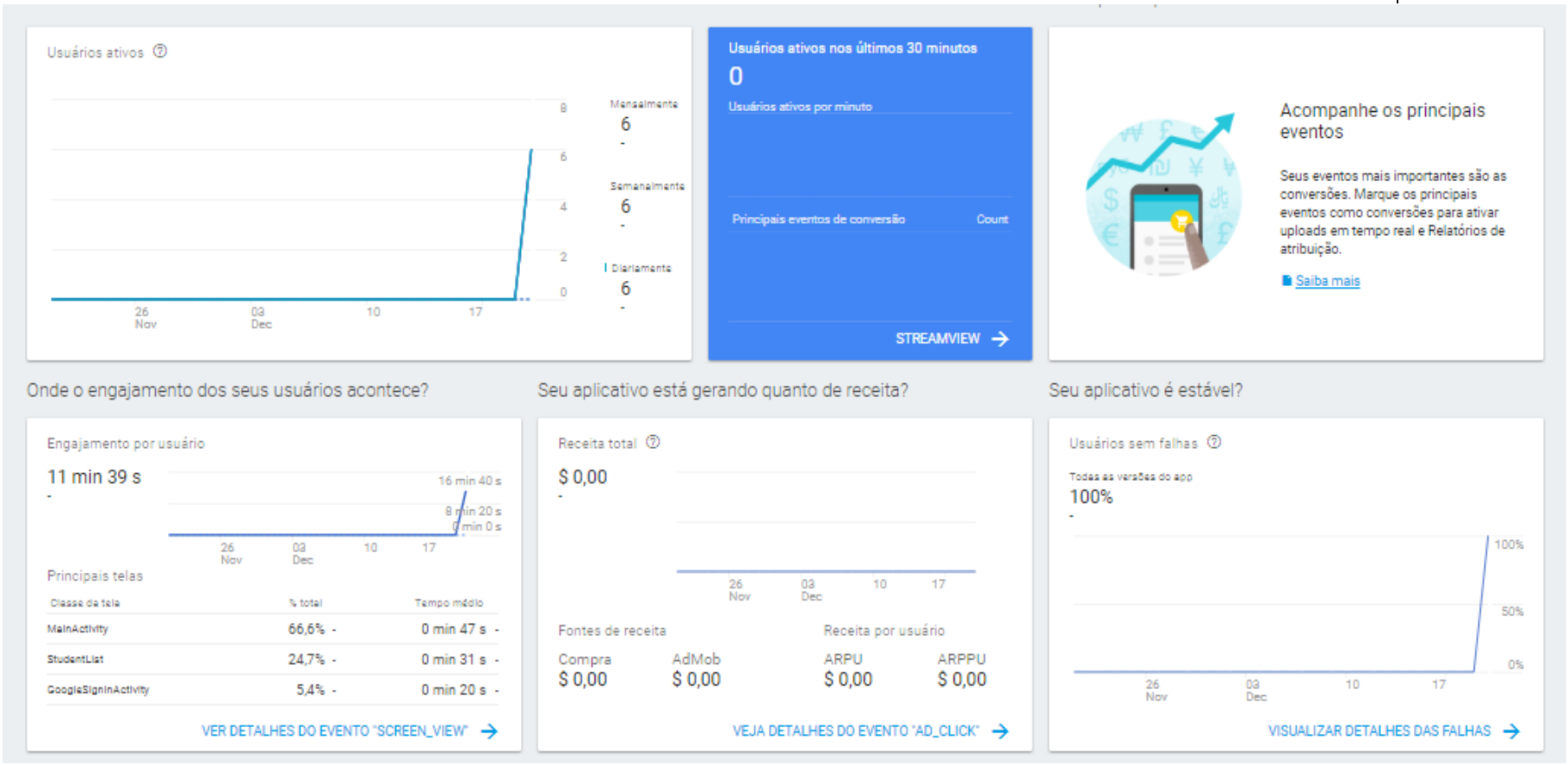
```
<resources>
    ...
    <string name="banner_ad_unit_id">ca-app-pub-3940256099942544/6300978111</string>
</resources>
```

O ID acima é apenas para teste, assegure-se de mudar antes de publicar o app!

Firebase

- Console (https://console.firebase.google.com):
  - Responsável por visualizar/gerenciar os dados do app
  - Podemos ver, por exemplo:
    - Quais usuários foram autenticados
    - O que está armazenado no Database (e alterar!)
    - Quanto tempo se passa em cada activity
    - De qual país são os usuários
    - Receita total do app e por fonte
    - Relatório de crashes
    - etc

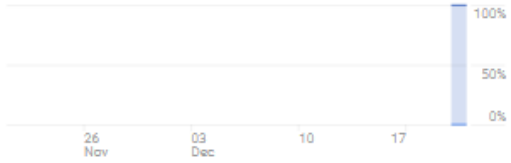
Firebase


Usuários ativos
8
6
4
2
0
26 Nov
03 Dec
10
17
Mensalmente
6
-
Semanalmente
6
-
Diariamente
6
-
Usuários ativos nos últimos 30 minutos
0
Usuários ativos por minuto
Principais eventos de conversão
Count
STREAMVIEW
Acompanhe os principais eventos
Seus eventos mais importantes são as conversões. Marque os principais eventos como conversões para ativar uploads em tempo real e Relatórios de atribuição.
Saiba mais
Onde o engajamento dos seus usuários acontece?
Seu aplicativo está gerando quanto de receita?
Seu aplicativo é estável?
Engajamento por usuário
11 min 39 s
-
16 min 40 s
8 min 20 s
0 min 0 s
26 Nov
03 Dec
10
17
Principais telas
Classe da tela
% total
Tempo médio
MainActivity
66,6% -
0 min 47 s -
StudentList
24,7% -
0 min 31 s -
GoogleSignInActivity
5,4% -
0 min 20 s -
VER DETALHES DO EVENTO "SCREEN_VIEW"
Receita total
$ 0,00
-
26 Nov
03 Dec
10
17
Fontes de receita
Compra
$ 0,00
AdMob
$ 0,00
Receita por usuário
ARPU
$ 0,00
ARPPU
$ 0,00
VEJA DETALHES DO EVENTO "AD_CLICK"
Usuários sem falhas
Todas as versões do app
100%
-
100%
50%
0%
26 Nov
03 Dec
10
17
VISUALIZAR DETALHES DAS FALHAS

Firebase

Qual é o nível de aceitação da versão mais recente entre os usuários?

Adoção da versão do aplicativo — De a dez 21



Última versão	% de usuários ativos	Usuários sem falhas
1.0	100%	100%

VER A VERSÃO MAIS RECENTE →

Como você conquista novos usuários?

Aquisição — De 120 dias a dez 20

Origem	Conversões first_open	LTV
(direct)	5	$ 0,00

VER ATRIBUIÇÃO DE FIRST_OPEN →

Qual é seu desempenho na retenção de usuários?

Coortes de retenção — De 6 semanas a dez 20

	Semana 0	Semana 1	Semana 2	Semana 3	Semana 4	Semana 5
Todos os usuários	0%	0%	0%	0%	0%	0%
5 de nov - 11 de nov						
12 de nov - 18 de nov						
19 de nov - 25 de nov						
26 de nov - 2 de dez						
3 de dez - 9 de dez						
10 de dez - 16 de dez						

26 de nov - 2 de dez
Semana 1
0
Retenção de 0%

VIS… RETENÇÃO →

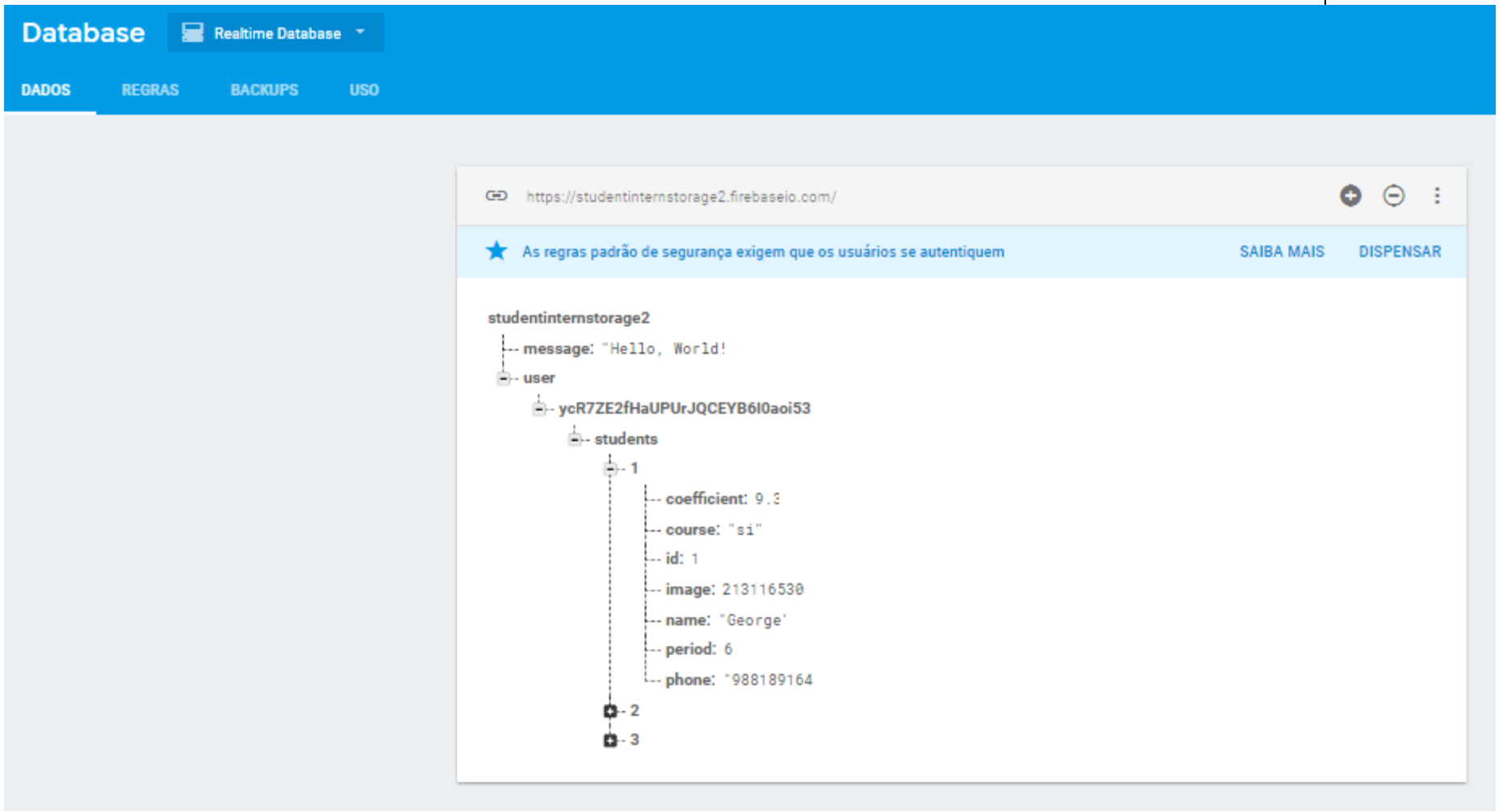
Como é seu público?

Local | Dispositivos | Informações demográficas

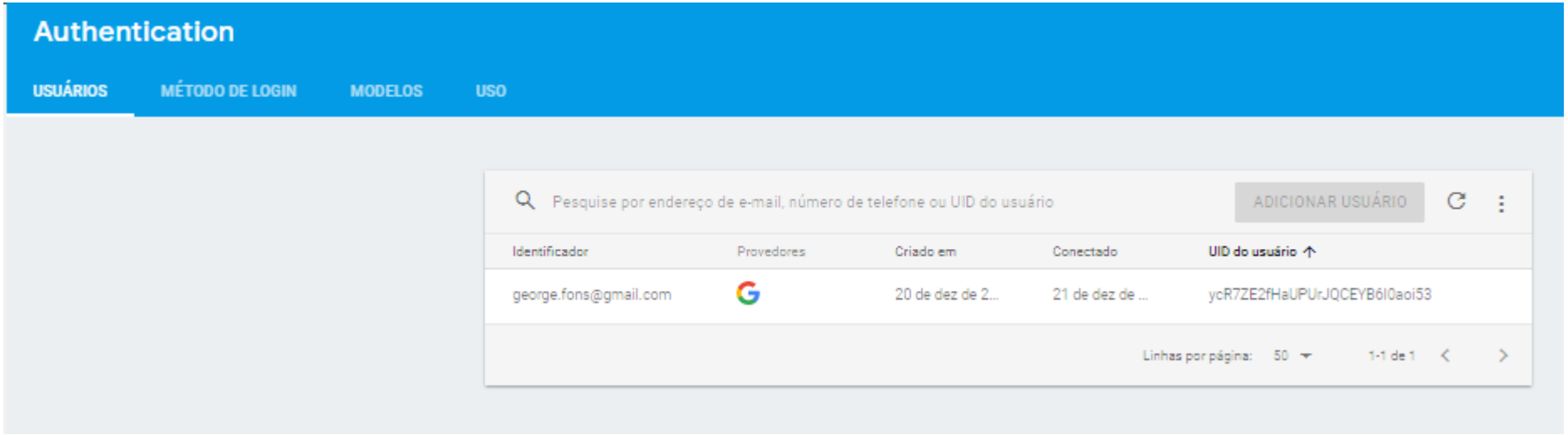
País/região	Sessões	% total
Brasil	8	100%

VISUALIZAR O PÚBLICO "TODOS OS USUÁRIOS" →

Firebase


Database
Realtime Database
DADOS
REGRAS
BACKUPS
USO
https://studentinternstorage2.firebaseio.com/
As regras padrão de segurança exigem que os usuários se autentiquem
SAIBA MAIS
DISPENSAR
studentinternstorage2
message: "Hello, World!
user
ycR7ZE2fHaUPUrJQCEYB6l0aoi53
students
1
coefficient: 9.3
course: "si"
id: 1
image: 213116530
name: "George"
period: 6
phone: "988189164
2
3

Firebase


Authentication
USUÁRIOS
MÉTODO DE LOGIN
MODELOS
USO
Pesquise por endereço de e-mail, número de telefone ou UID do usuário
ADICIONAR USUÁRIO
Identificador
Provedores
Criado em
Conectado
UID do usuário
george.fons@gmail.com
20 de dez de 2...
21 de dez de ...
ycR7ZE2fHaUPUrJQCEYB6l0aoi53
Linhas por página: 50
1-1 de 1

Bibliografia

- Google and Open Handset Alliance n.d. **Android API Guide**. http://developer.android.com/guide/index.html. Acessado em Maio de 2017.

- Google and Open Handset Alliance n.d. **Android training guide**. http://developer.android.com/training/index.html. Acessado em Maio de 2017.

- Lecheta, R. R. **Google Android: Aprenda a criar aplicações para dispositivos móveis com o Android SDK**. 3ª edição. São Paulo: Novatec Editora, 2013.