

Qt



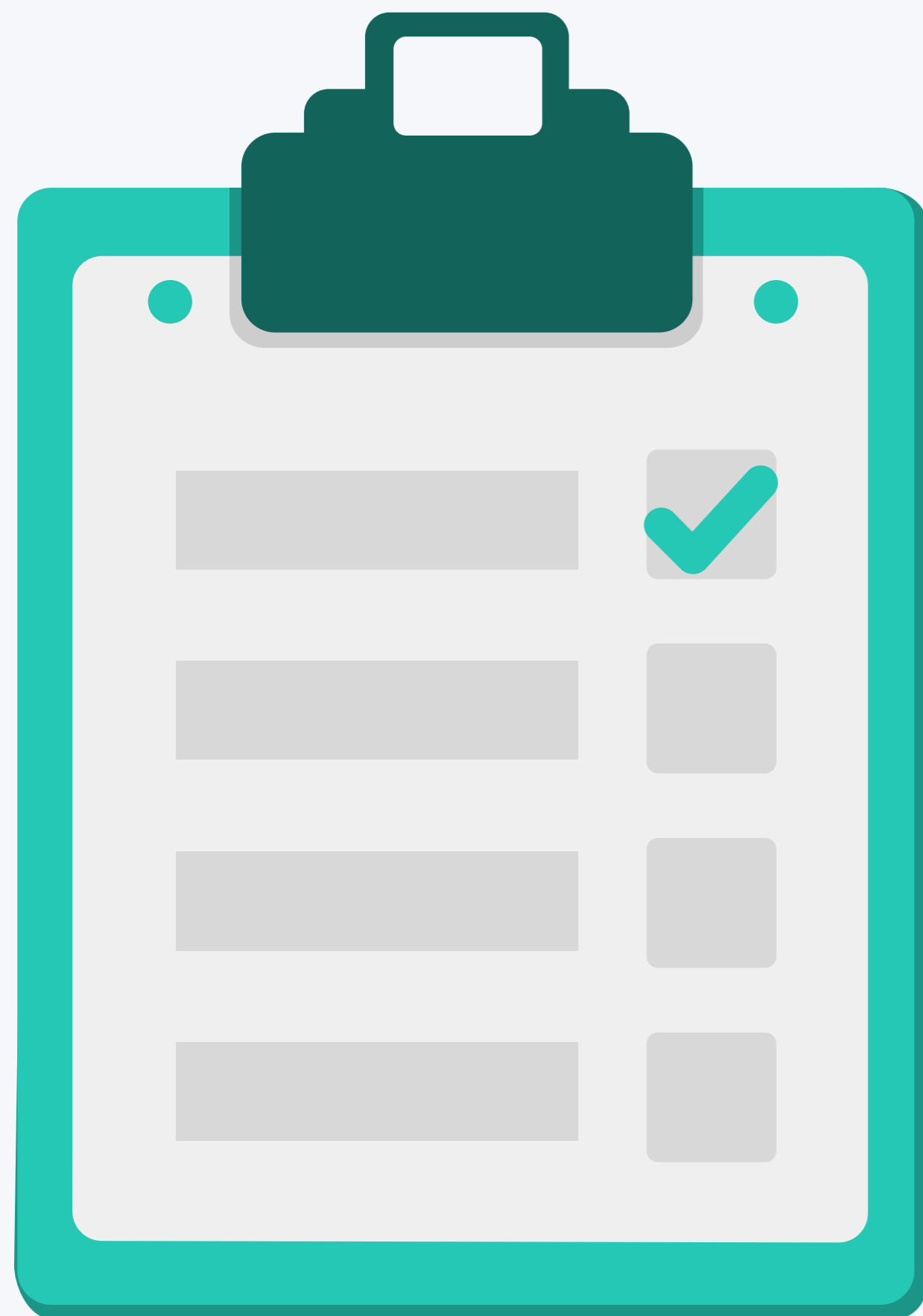
# Desenvolvendo Aplicações Android com

---

**Sandro S. Andrade**  
sandroandrade@kde.org

**IFBA/KDE**

# Objetivos



- 1 Apresentar os principais conceitos e fundamentos do Qt e do QML.
- 2 Apresentar os principais recursos do Qt para desenvolvimento de aplicativos móveis utilizando QML.
- 3 Apresentar os principais recursos do Qt para acesso a sensores, câmeras e comunicação cliente-servidor via RESTful.
- 4 Proporcionar vivências práticas sobre os tópicos acima.



**whoareyou?**



# whoami?

Professor no Instituto Federal de Educação, Ciência e Tecnologia da Bahia (IFBA)  
Colaborador nas comunidades Qt e KDE há 10 anos  
Desenvolvedor/Arquiteto C++ e Qt há 18 anos



[sandroandrade@kde.org](mailto:sandroandrade@kde.org)



[sandroandrade.org](http://sandroandrade.org)



[@andradesandro](https://twitter.com/andradesandro)

# Agenda



01

## INTRODUÇÃO AO Qt E AO QML

O que é o Qt? Porque utilizar o Qt no desenvolvimento para mobile? Módulos do Qt voltados para mobile.

02

## ANDROID, QML E Qt QUICK CONTROLS 2

Anatomia de uma aplicação Qt para Android Hello world com QML e QtQuickControls 2.

03

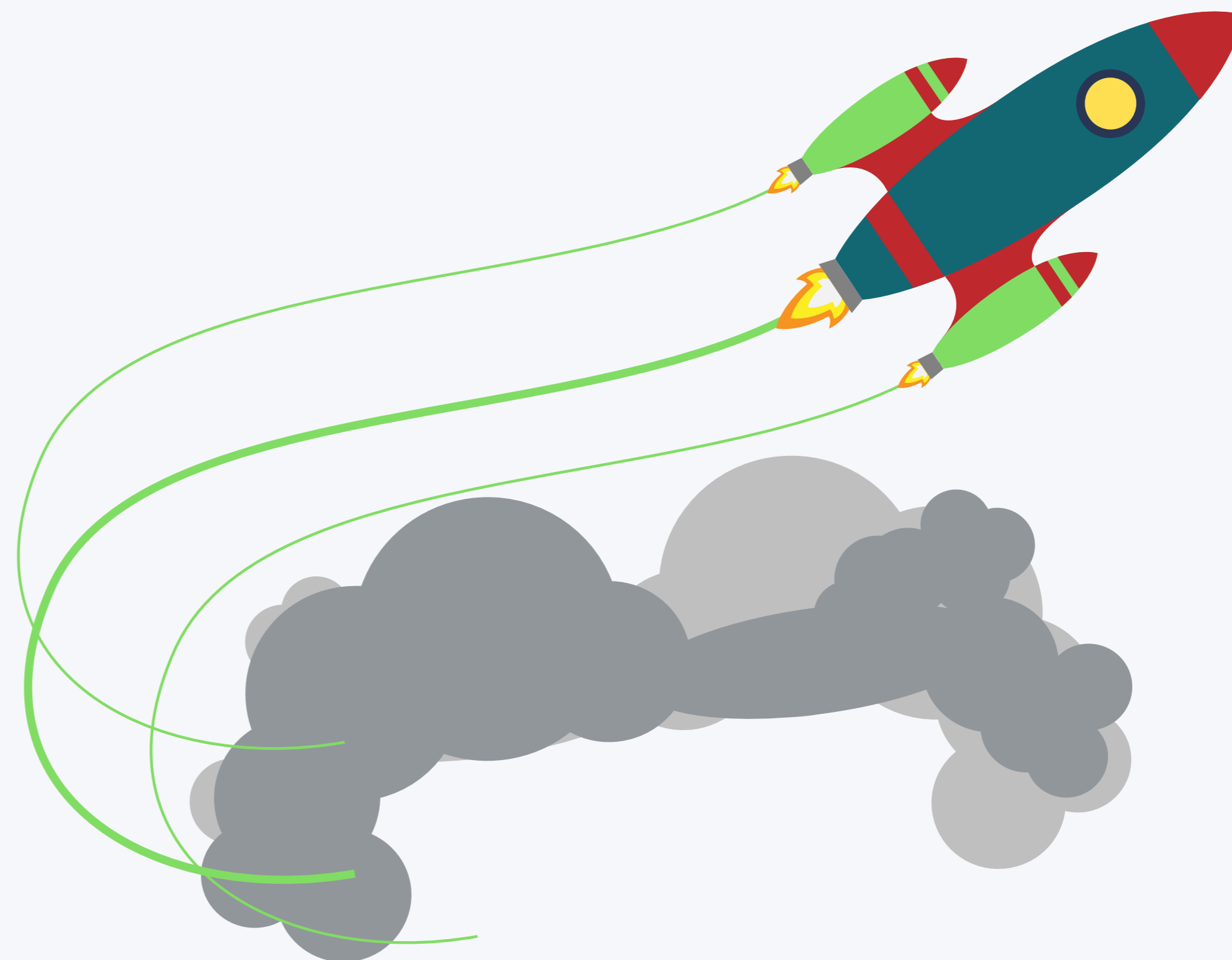
## SENSORES E MULTIMÍDIA

Utilizando sensor de proximidade e acelerômetro. Acessando a câmera do smartphone.

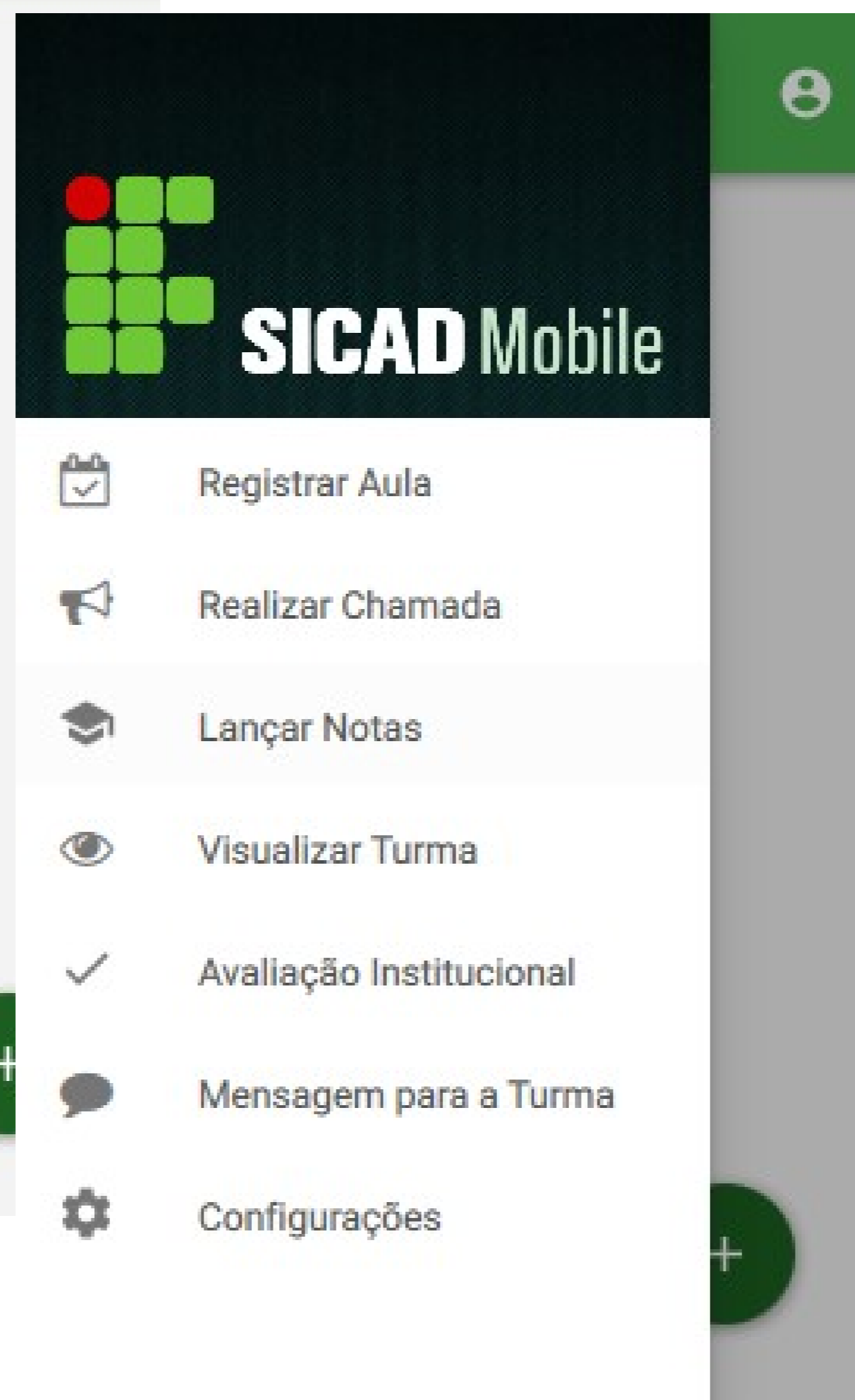
04

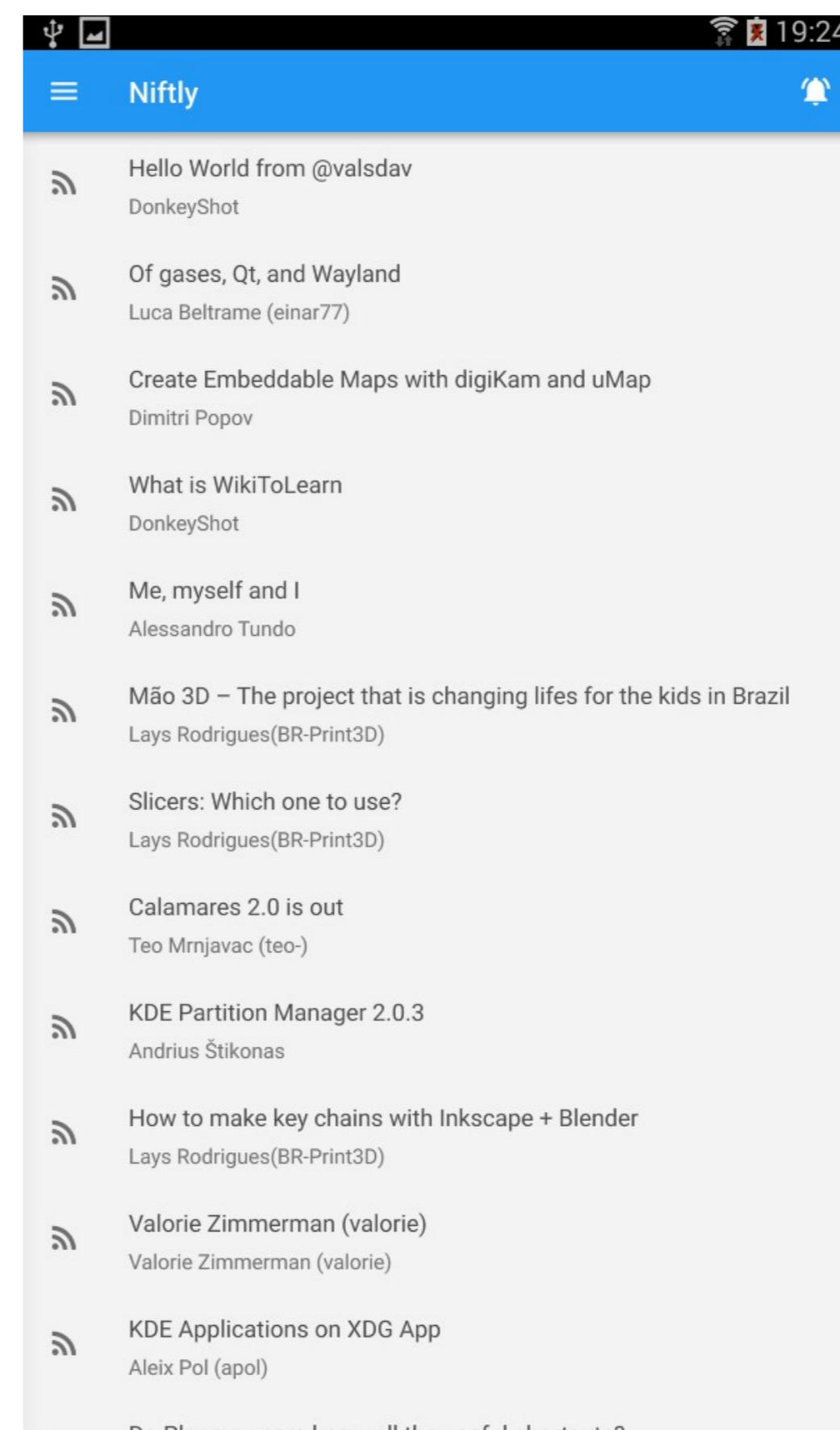
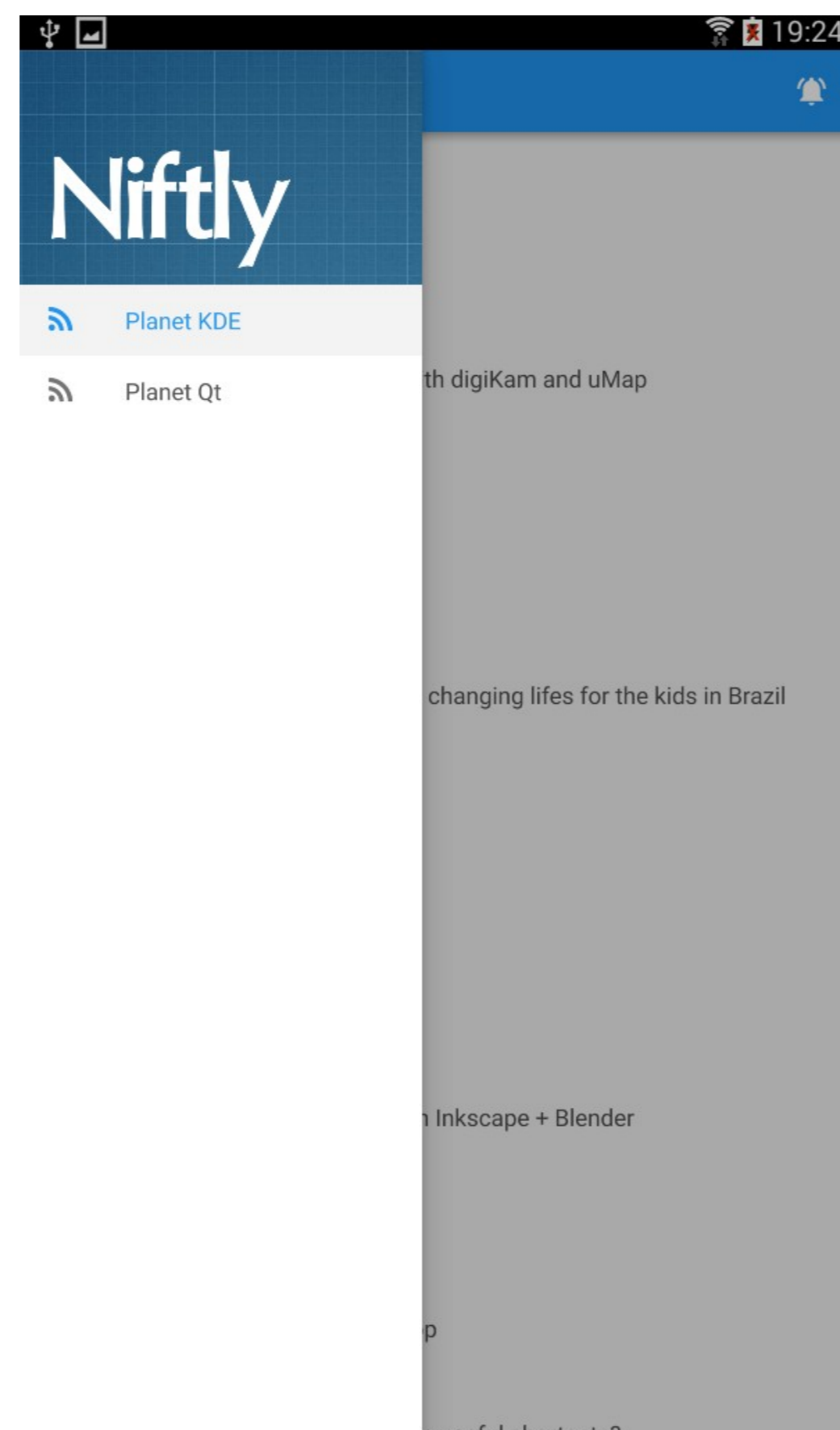
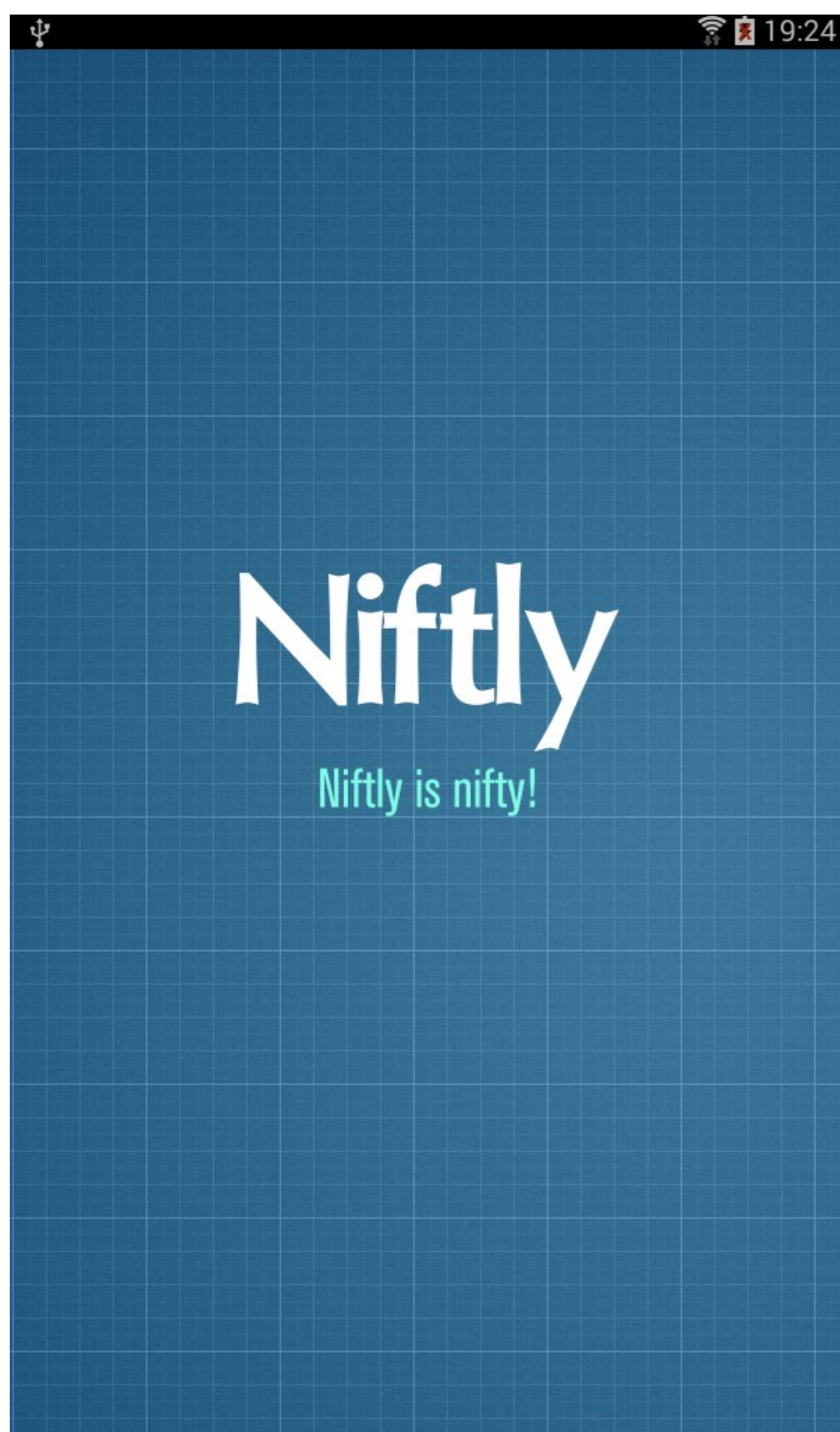
## APLICATIVOS EM MODO CLIENTE-SERVIDOR

Comunicação via RESTful. Caches locais.

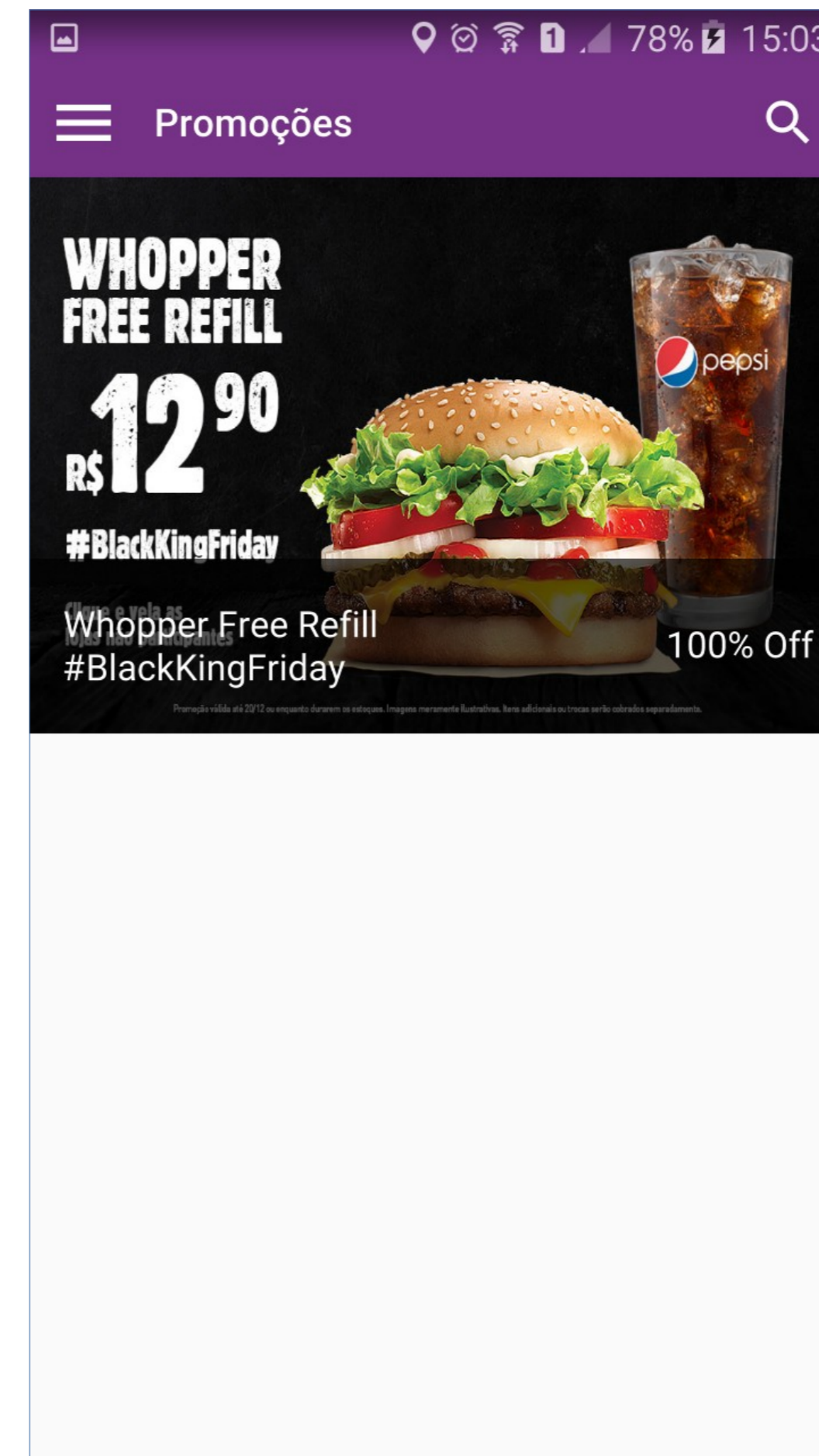
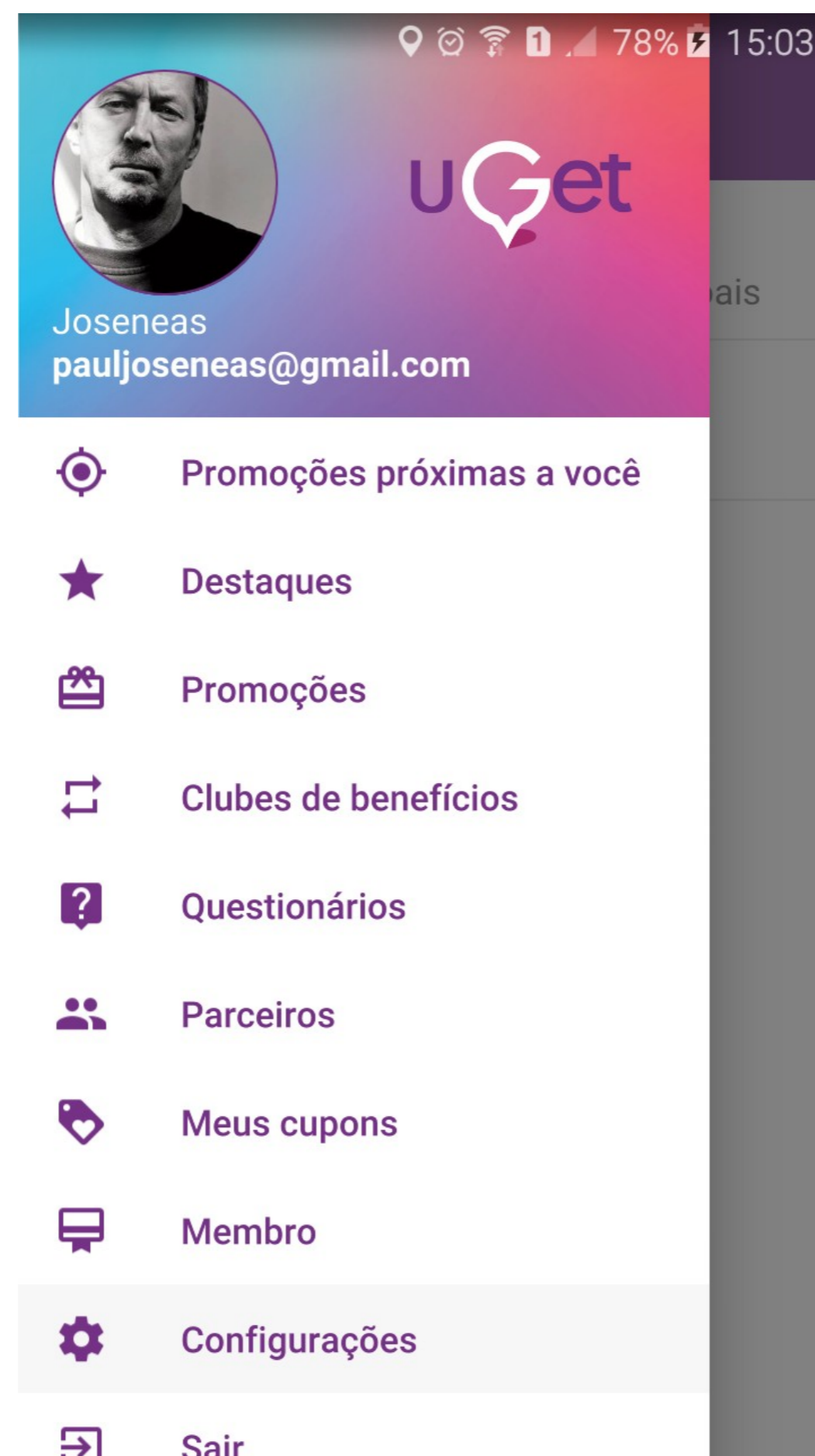
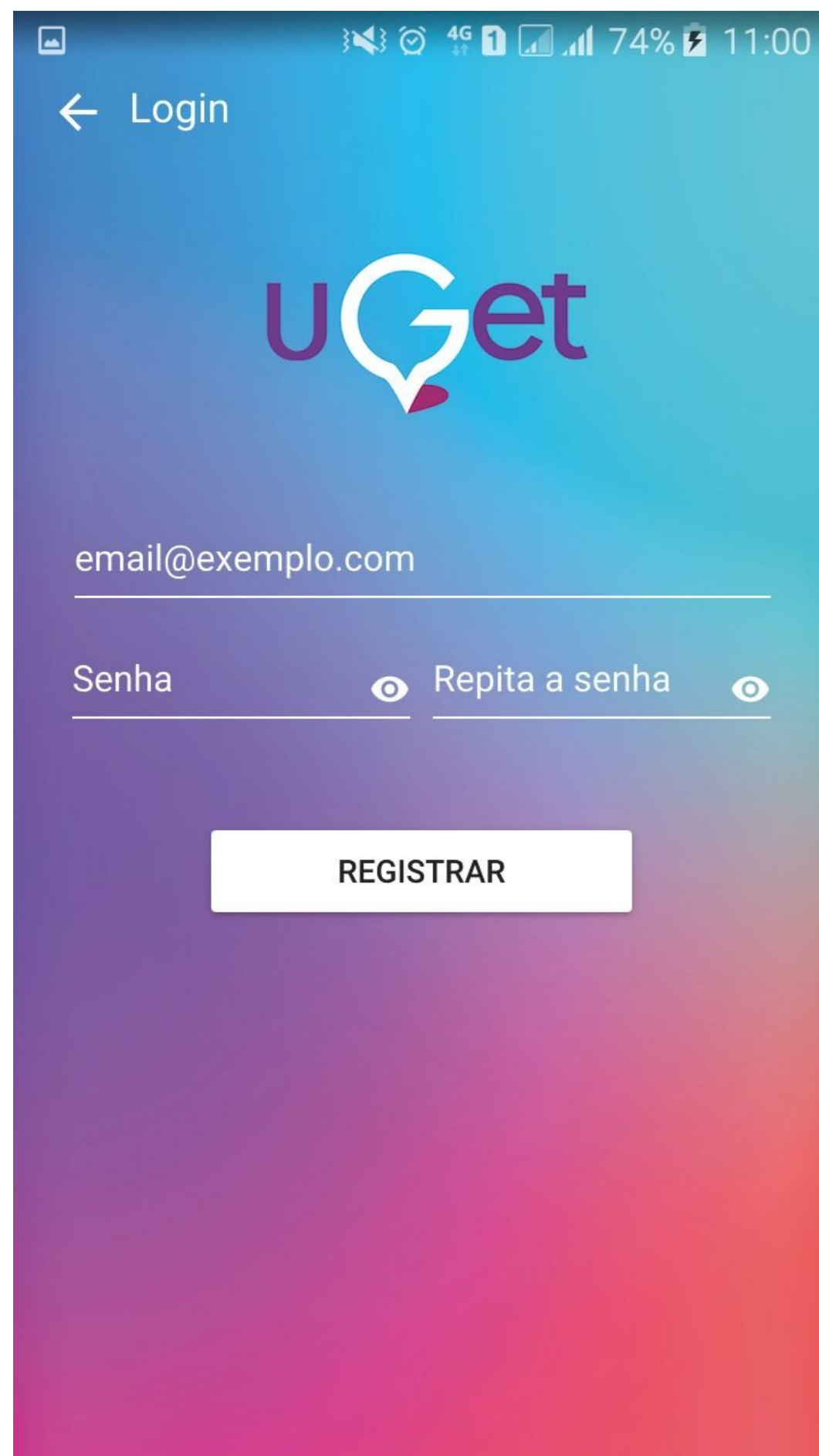















10:30 76%



email

senha

**ENTRAR**

**CRIAR CONTA**

**RECUPERAR SENHA**

10:29 76%

### Notificações

→ Você Todos os professores do curso de None

Em design gráfico e editoração, Lorem ipsum é um texto utilizado para preencher o espaço de texto em publicações (jornais, revistas, e websites), com a finalidade de verificar o lay-out, tipografia e formatação antes de utilizar conteúdo real. Muitas vezes este texto também é utilizado em catálogos de tipografia para demonstrar textos e títulos escritos com as fontes [...]

sábado, 22 de julho de 2017 10:28:13

→ Você Todos os alunos do curso de None


Pessoal, amanhã é o último dia para a inscrição no ADS DAY. Quem ainda não fez a inscrição, passe na CADS e solicite um formulário. Importante lembrar que é requerido 1 KG de alimento.

sábado, 22 de julho de 2017 10:27:4

→ Você Todos os alunos do curso de None


Prezados alunos, a aula de hoje do professor Manoel será no lab5, pavilhão 7.

sábado, 22 de julho de 2017 10:24:35



10:24 77%

### Meu perfil



**Nome completo**  
Renato

**Sexo**  
Masculino

**Email**  
renato@ifba.edu.br

**Endereço**  
Rua Teste

**Data de nascimento**  
08-03-1989

Qt



Minuet

Hear the chord and then choose an answer from options below

Play Question Give Up Start Test

Available Answers

Minor	Major	Minor 7	Dominant 7	Diminished	Augmented	Minor 9
Major 9	Major maj7	Major maj7(9)	Major Seventh	Diminished Seventh	Half Diminished Seventh	Minor maj7
Major maj7(b5)	Major 7	Major 7(b5)	Major 7(#5)	Major 7(#9)	Major 7(b9)	Major 7(#5/b9)

Your Answer(s)



Minuet

by Qt

Minuet

Hear the chord and then choose an answer from options below

PLAY QUESTI... GIVE UP

Available Answers

Minor	Major
Minor 7	Dominant 7
Diminished	Augmented

Your Answer(s)



Ágora Mobile



Desenvolvido em Qt



Sandro Andrade

Programação

Palestrantes

Tags

Logout

Sobre

QtCon Brasil 2017

18/08/2017

08:00 - 08:30  
Foyer - Térreo  
Credenciamento

08:30 - 12:30  
Sala 9 - 4º andar  
Desenvolvendo Aplicações Embarcadas com Qt e Toradex - Turma 1  
Cleiton Bueno (B2Open)

08:30 - 12:30  
Sala 10 - 4º andar  
Desenvolvendo Aplicações Android com Qt Turma 1  
Sandro Andrade (IFBA/KDE)

12:30 - 14:00  
Almoço

PROGRAMAÇÃO

PALESTRANTES

QtCon Brasil 2017

Qt

Resumo

Neste treinamento serão explorados alguns recursos para desenvolvimento utilizando a IDE Qt Creator e compilação-cruzada de aplicações, desenvolvendo uma aplicação abordando tópicos comumente usados no desenvolvimento de sistemas embarcados. Tópicos a serem abordados: introdução ao Qt para Linux Embarcado, introdução às placas Toradex, IDE Qt Creator - configuração de toolchain e acesso à placa Toradex, manipulação de entradas e saídas via GPIO, acionamento de led via GPIO, leitura de status de botão via GPIO, criação de temporizador para acesso ao led, obtenção de informações do hardware e do sistema, transporte destas informações para uma interface grafica amigável e fluida com o QML, mecanismo de troca de dados entre C++ e o QML e geração de log.

Biografia do Palestrante

**Cleiton Bueno** - Formando em Engenharia da Computação, proprietário da B2Open, empresa com foco em desenvolvimento, consultoria e treinamentos de Linux, Linux Embarcado, Qt5 e Python. Cleiton trabalha há mais de 10 anos com Linux e todo seu ecossistema e tem o prazer em compartilhar cada experiência e vivência open-source no seu bloa pessoal ou no site Embarcados.

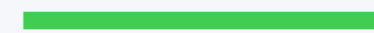
PROGRAMAÇÃO

PALESTRANTES





# INTRODUÇÃO AO Qt E AO QML



O que é o Qt? Porque utilizar o Qt no desenvolvimento para mobile? Módulos do Qt voltados para mobile.

# O Qt

...



O Qt é um toolkit para desenvolvimento multiplataforma de aplicações em diversos domínios, com foco em execução nativa, excelente desempenho e produtividade.



O Qt  
...



BMW  
GROUP



Ableton



OPERA™  
software



AUTODESK®



Google

KDAB

last.fm™  
the social music revolution



Adobe

MENDELEY

basysKom



# O Qt



## Por que usar?

- Tecnologia madura (desenvolvido há 23 anos).
- Alta produtividade (mesmo com C++, melhor ainda com QML/JS).
- Rico em funcionalidades (47 módulos, 1647 classes).
- Efetivo para desenvolvimento multiplataforma.
- Excelente documentação e comunidade ativa.
- Excelente desempenho (aceleração via hardware no QML).
- Diversas bibliotecas de terceiros (KF5, include.org).
- Open Governance com licença dual (LGPL e comercial).

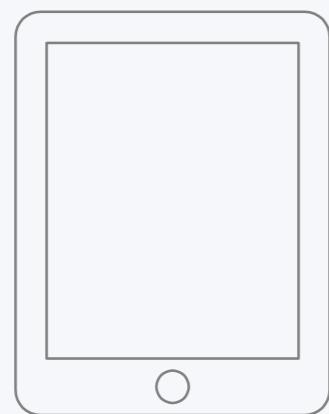


# O Qt



## Tecnologias para UI/UX:

- QtWiddgets: C++ (oficial), Python, C#, Go Haskell, Ruby
- QtQuick: QML + JavaScript
- QtWebEngine: HTML + CSS + JavaScript
- QtCharts/QtDataVisualization

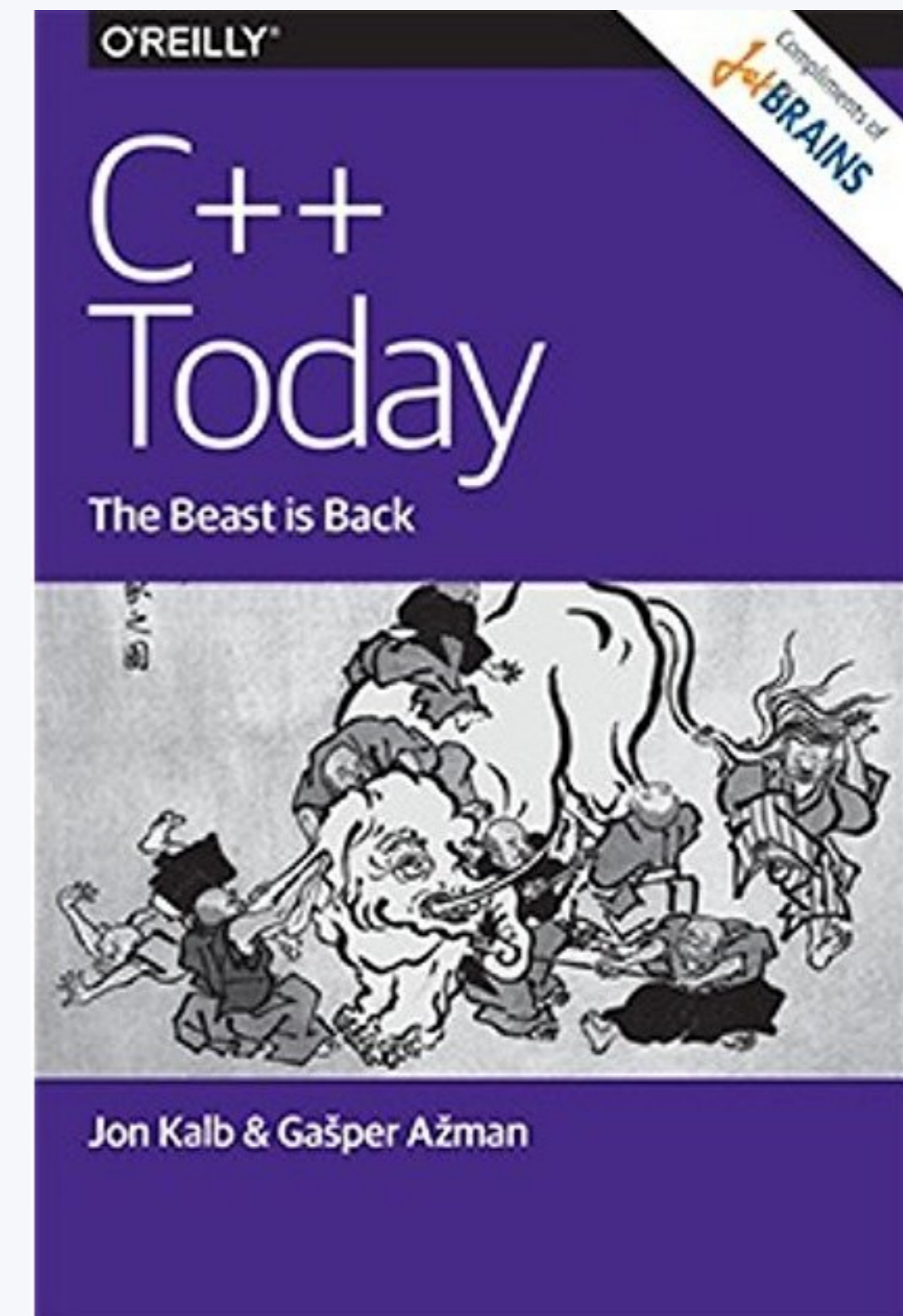


# O Qt



## Tecnologias para UI/UX:

- QtWiddgets: C++ (oficial), Python, C#, Go Haskell, Ruby
- QtQuick: QML + JavaScript
- QtWebEngine: HTML + CSS + JavaScript
- QtCharts/QtDataVisualization





# O Qt



## QtQuick x QtWidgets x QtWebEngine

	QtQuick	QtWidgets	QtWebEngine
Linguagem	QML/JS	C++	HTML/CSS/JS
Look'n'feel nativo	✓	✓	
Look'n'feel customizado	✓	✓	✓
UI animadas e fluidas	✓		✓
Suporte a touch screen	✓		✓

# O Qt



## QtQuick x QtWidgets x QtWebEngine

	QtQuick	QtWidgets	QtWebEngine
Widgets padrão da indústria		✓	
Model/View	✓	✓	
Prototipagem rápida de UX	✓✓	✓	✓
Aceleração via hardware	✓	✓	✓

# O Qt



## QtQuick x QtWidgets x QtWebEngine

	QtQuick	QtWidgets	QtWebEngine
Efeitos gráficos (partículas, etc)	✓		
Rich text	✓	✓	
Integração de conteúdo web existente			✓



# Qt e Mobile



2006

Qt/Embedded + Qtopia



2006

Qtopia em milhares de dispositivos  
(Sharp/Motorola)

2009

Lançamento do QML

2010-2011

Qt no Symbian e MeeGo



# Qt e Mobile



**2011**

Projeto Necessitas (KDE) e o primeiro port do KDE para Android

**2015**

QML Qt Location e QtQuick Controls for Embedded

**2013**

Primeiro tech-preview oficial do suporte a Android e iOS (BlackBerry, Sailfish/Jolla e Ubuntu Mobile)

**2016**

QtQuickControls 2, KDE Kirigami e Android services com Qt

**2014**

API QtPurchasing multiplataforma e suporte a Bluetooth LE

# O Qt



## Por que usar **em plataformas móveis?**

- Um codebase, múltiplas plataformas.

<b>Mobile Platforms: <a href="#">Android</a>, <a href="#">iOS</a>, <a href="#">WinRT</a></b>		
Windows Phone 8.1 (arm)	MSVC 2013	Hosts: <b>Windows 8.1</b> , Windows 10
Windows Runtime (x86, x86_64, arm)	MSVC 2013	Hosts: <b>Windows 8.1</b> , Windows 10
Universal Windows Platform (UWP) (x86, x86_64, arm)	MSVC 2015	Hosts: <b>Windows 10</b>
iOS 7 and above	Clang as provided by Apple	<b>macOS host</b>
Android (API Level: 16)	GCC as provided by Google	Hosts: <b>Ubuntu 14.04 (64-bit)</b> , macOS, Windows



# O Qt



## Por que usar **em plataformas móveis?**

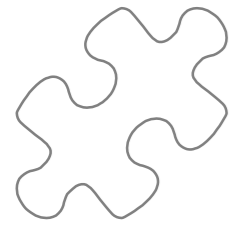
- Um codebase, múltiplas plataformas.
- Alto desempenho (nativo + aceleração via GPU).
- Boa documentação.
- Está em constante evolução, com foco nestas plataformas.
- Melhor gerenciamento de memória.
- Mesma API e funcionalidades em várias versões do Android.



# ANDROID, QML E Qt QUICK CONTROLS 2

---

Anatomia de uma aplicação Qt para Android Hello world com QML e QtQuickControls 2



## MÓDULOS DO Qt ESPECIFICAMENTE CRIADOS PARA MOBILE

Qt Bluetooth (QML e C++)

Android, iOS, Linux (BlueZ 4.x/5.x) e OS X

Qt Graphical Effects (QML)

Qt Positioning (QML e C++)

Android, iOS, Linux (com GeoClue) e WinRT

Qt Sensors (QML e C++)

Android, iOS, SailFish e WinRT

Qt Quick Extras

Qt Android Extras

Qt Notifier

Qt Mac Extras

Qt









## ANATOMIA DE UMA APLICAÇÃO Qt PARA ANDROID

- Solução: QPA + JNI
- Uma aplicação Qt para Android é formada por duas partes:
  - A aplicação em si, criada pelo desenvolvedor
  - Launcher da aplicação Android, gerada automaticamente pela IDE oficial do Qt (Qt Creator) O Qt Creator automatiza todo o processo de geração (e assinatura) do .apk.



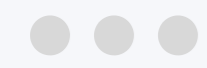


## ANATOMIA DE UMA APLICAÇÃO Qt PARA ANDROID

- Três métodos de implantação:
  - Todas as dependências empacotadas no .apk.
  - Implantação baseada no serviço Ministro.
  - Implantação das dependências em um diretório temporário (para fins de debugging).



# A Linguagem QML



- O QML é uma linguagem declarativa para especificação e programação de interfaces gráficas de usuário.
- O QtQuick é a biblioteca padrão de tipos e funcionalidades principais do QML:
  - Tipos visuais e interativos, animações, models, views, efeitos de partículas, etc.



# Hello QML



```
1. import QtQuick 2.3
2.
3. Rectangle {
4.     width: 200
5.     height: 100
6.     color: "red"
7.
8.     Text {
9.         anchors.centerIn: parent
10.        text: "Hello, World!"
11.    }
12. }
```

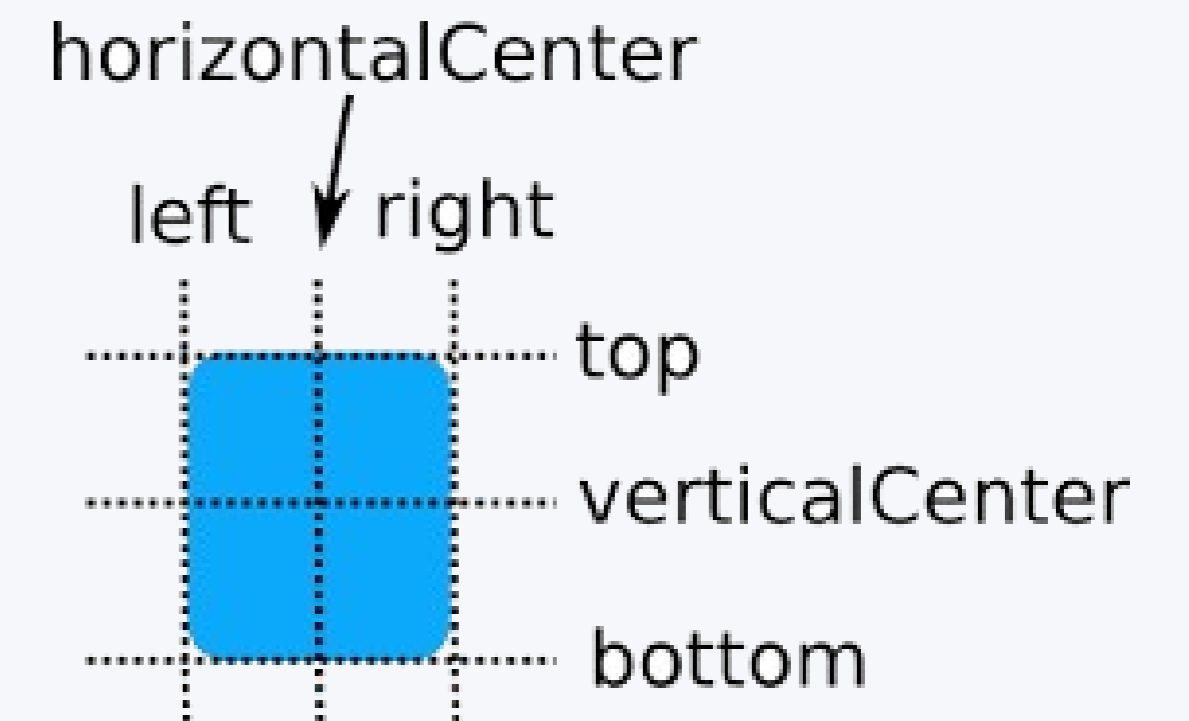
Atividade Prática



# Usando um ApplicationWindow



```
1. import QtQuick 2.3
2. import QtQuick.Controls 1.2
3. import QtQuick.Window 2.2
4. ApplicationWindow {
5.     title: qsTr("Hello World")
6.     width: 640; height: 480
7.     menuBar: MenuBar {
8.         Menu { title: qsTr("File")
9.             MenuItem { text: qsTr("&Open"); onTriggered: console.log("Open") }
10.            MenuItem { text: qsTr("Exit"); onTriggered: Qt.quit() }
11.        }
12.    }
13.    Button {
14.        text: qsTr("Hello World")
15.        anchors.horizontalCenter: parent.horizontalCenter
16.        anchors.verticalCenter: parent.verticalCenter
17.    }
18. }
```



# Capturando Ações do Mouse



```
1. Rectangle {
2.     width: 200
3.     height: 100
4.     color: "red"
5.
6.     Text {
7.         anchors.centerIn: parent
8.         text: "Hello, World!"
9.     }
10.
11.     MouseArea {
12.         anchors.fill: parent
13.         onClicked: parent.color = "blue"
14.     }
15. }
```

# Property Bindings



```
1. Rectangle {
2.     width: 400
3.     height: 200
4.
5.     Rectangle {
6.         width: parent.width / 2
7.         height: parent.height
8.     }
9.
10.    Rectangle {
11.        width: parent.width / 2
12.        height: parent.height
13.        x: parent.width / 2
14.    }
15. }
```



# Definindo Tipos Customizados



...

## 1. `MyButton.qml`

```
2. import QtQuick 2.3
3. Rectangle {
4.     width: 100; height: 100
5.     color: "red"
6.
7.     MouseArea {
8.         anchors.fill: parent
9.         onClicked: console.log("Clicked!")
10.    }
11. }
```

## 1. `main.qml`

```
2. import QtQuick 2.3
3. Column {
4.     MyButton { width: 50; height: 50 }
5.     MyButton {
6.         x: 50; width: 100
7.         height: 50; color: "blue"
8.     }
9.     MyButton {
10.        width: 50; height: 50
11.        radius: 8
12.    }
13. }
```

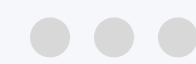
# MiniBrowser



## Metas:

- Uso do módulo webview.
- Prática com âncoras e layouts.
- Property bindings.
- Signals e handlers.
- Ícones e fontes.

# Atributos de Objetos QML



- Objetos QML podem ter atributos de diferentes tipos:
  - O atributo id.
  - Atributos do tipo property.
  - Atributos do tipo signal.
  - Atributos do tipo signal handler.
  - Atributos do tipo método.
  - Atributos do tipo attached properties/signal handlers.

# Atributos de Objetos QML



...

– 0 atributo id:

```
1. import QtQuick 2.0
2.
3. Column {
4.     width: 200; height: 200
5.
6.     TextInput { id: myTextInput; text: "Hello World" }
7.     Text { text: myTextInput.text }
8. }
```



# Atributos de Objetos QML



...

– Atributos do tipo property:

```
1. main.qml
2. Rectangle {
3.     property color previousColor
4.     property color nextColor
5.     onNextColorChanged: console.log("Next color: " + nextColor.toString())
6. }
7.
8. main.qml
9. Rectangle {
10.    color: "red"
11.    property color nextColor: "blue" // declaration + initialization
12. }
```

# Atributos de Objetos QML



...

– Property alias:

```
1. import QtQuick 2.0
2.
3. Rectangle {
4.     property alias buttonText: textItem.text
5.
6.     width: 100; height: 30; color: "yellow"
7.     Text { id: textItem }
```

# Atributos de Objetos QML



...

– Atributos do tipo signal handler:

```
1. import QtQuick 2.0
2.
3. Item {
4.     width: 100; height: 100
5.
6.     MouseArea {
7.         anchors.fill: parent
8.         onClicked: {
9.             console.log("Click!")
10.        }
11.    }
12. }
```

# Atributos de Objetos QML



...

– Definindo atributos do tipo signal:

## 1. `SquareButton.qml`

```
2. Rectangle {
3.     id: root
4.
5.     signal activated(real xPos, real yPos)
6.     signal deactivated
7.     property int side: 100
8.     width: side; height: side
9.
10.    MouseArea {
11.        anchors.fill: parent
12.        onPressed: root.activated(mouse.x, mouse.y)
13.        onReleased: root.deactivated()
14.    }
15. }
```

## 1. `main.qml`

```
2. SquareButton {
3.     onActivated: console.log("Activated at " +
4.         xPos + ", " + yPos)
5.     onDeactivated: console.log("Deactivated!")
6. }
```



# Atributos de Objetos QML



...

– Signal handlers de mudança de propriedades:

```
1. import QtQuick 2.0
2.
3. TextInput {
4.     text: "Change this!"
5.
6.     onTextChanged: console.log("Text has changed to:", text)
7. }
```

# Atributos de Objetos QML



...

– Atributos do tipo método:

```
1. Item {
2.     width: 200; height: 200
3.     MouseArea {
4.         anchors.fill: parent
5.         onClicked: label.moveTo(mouse.x, mouse.y)
6.     }
7.     Text {
8.         id: label
9.         function moveTo(newX, newY) {
10.            label.x = newX; label.y = newY;
11.        }
12.        text: "Move me!"
13.    }
14. }
```

# Atributos de Objetos QML



...

– Conectando sinais a funções:

```
1. Rectangle {
2.     id: relay
3.     signal messageReceived(string person, string notice)
4.     Component.onCompleted: {
5.         relay.messageReceived.connect(sendToPost)
6.         relay.messageReceived.connect(sendToTelegraph)
7.         relay.messageReceived("Tom", "Happy Birthday")
8.     }
9.     function sendToPost(person, notice) {
10.        console.log("Sending to post: " + person + ", " + notice)
11.    }
12.    function sendToTelegraph(person, notice) {
13.        console.log("Sending to telegraph: " + person + ", " + notice)
14.    }
15. }
```

# Sistema de Tipos do QML



- Os tipos usados na definição de hierarquias de objetos QML podem ser:
  - Disponibilizados nativamente pela linguagem QML.
  - Registrados via C++.
  - Disponibilizados como documentos QML.

# Sistema de Tipos do QML



– Tipos disponibilizados nativamente:

<code>bool</code>	Binary true/false value
<code>double</code>	Number with a decimal point, stored in double precision
<code>enumeration</code>	Named enumeration value
<code>int</code>	Whole number, e.g. 0, 10, or -20
<code>list</code>	List of QML objects
<code>real</code>	Number with a decimal point
<code>string</code>	Free form text string
<code>url</code>	Resource locator
<code>var</code>	Generic property type



# Sistema de Tipos do QML



– Tipos disponibilizados nativamente:

<code>date</code>	Date value
<code>point</code>	Value with x and y attributes
<code>rect</code>	Value with x, y, width and height attributes
<code>size</code>	Value with width and height attributes
<code>color</code>	ARGB color value. The type refers to an ARGB color value. It can be specified in a number of ways:
<code>font</code>	Font value with the properties of QFont. The type refers to a font value with the properties of QFont
<code>matrix4x4</code>	A matrix4x4 type is a 4-row and 4-column matrix
<code>quaternion</code>	A quaternion type has scalar, x, y, and z attributes
<code>vector2d</code>	A vector2d type has x and y attributes
<code>vector3d</code>	Value with x, y, and z attributes
<code>vector4d</code>	A vector4d type has x, y, z and w attributes

# Sistema de Tipos do QML



...

– Tipos disponibilizados via JavaScript:

```
1. import QtQuick 2.0
2.
3. Item {
4.     property var theArray: new Array()
5.     property var theDate: new Date()
6.
7.     Component.onCompleted: {
8.         for (var i = 0; i < 10; i++)
9.             theArray.push("Item " + i)
10.        console.log("There are", theArray.length, "items in the array")
11.        console.log("The time is", theDate.toUTCString())
12.    }
13. }
```

# Acelerômetro e Sensor de Proximidade



Metas:

- Prática com sensores.
- Tratando diferentes densidades de pixel.
- Animações de propriedades.
- Attached properties e signal handler.

# Câmera



Metas:

- Uso do módulo de multimídia.

# Leitor de Feeds RSS



Metas:

- Prática com Model-View.
- Acesso remoto via XmlListModel.
- Navigation Drawer.



# Cliente-Servidor via RESTful



Metas:

- WebServices com RESTful.
- Acesso remoto via JsonListModel.
- SwipeView e StackView.

# Integrando QML com C++



Porque integrar QML com C++?

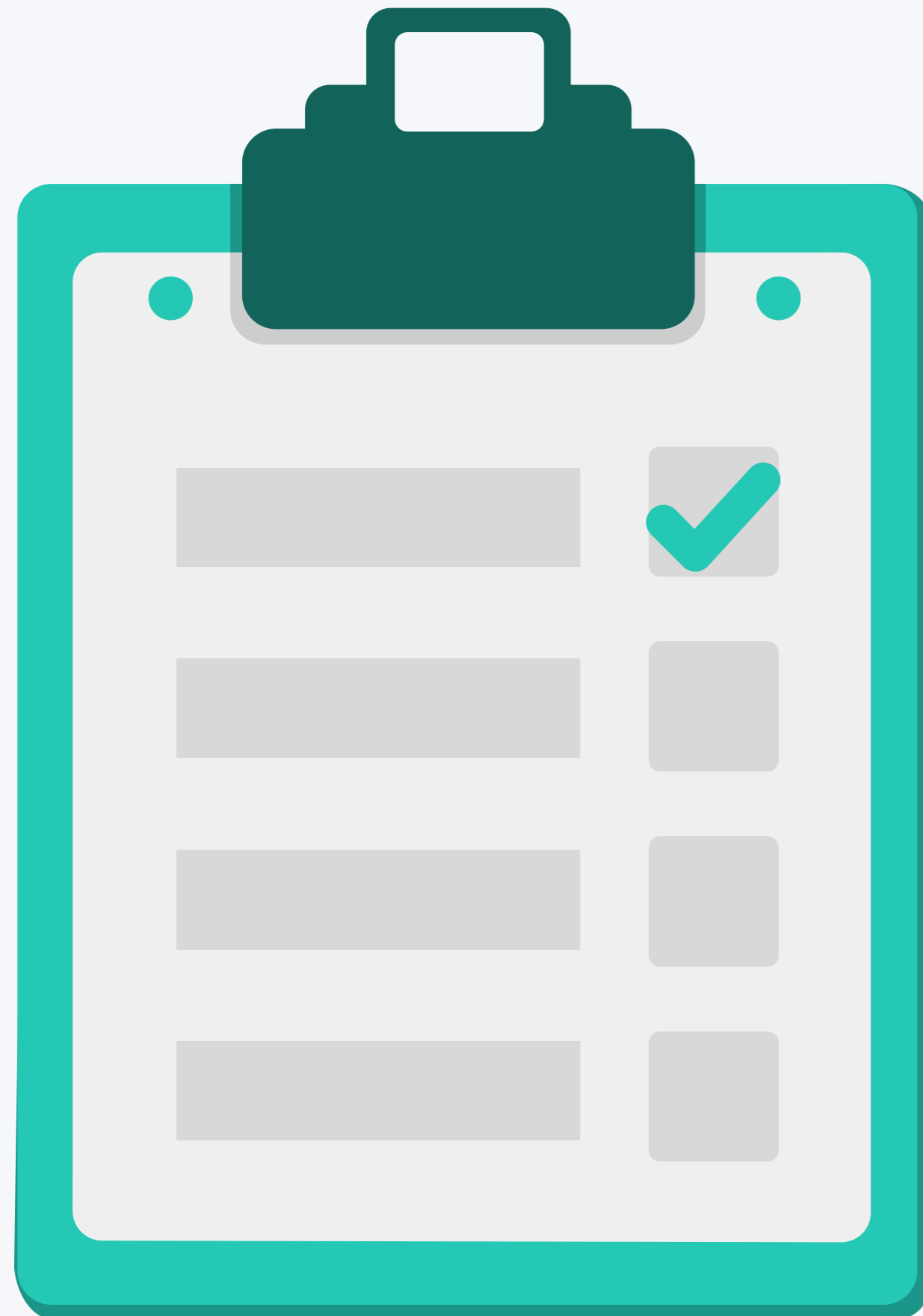
- Para separar código de interface (QML+JS) de código da lógica da aplicação (C++).
- Para usar funcionalidade C++ a partir de código QML.
- Para acessar objetos QML a partir do código C++.
- Para criar novos tipos de objetos QML a partir do C++.

# Ágora Mobile



## DEMO

# Conclusão



1

Consulte e confie na documentação do Qt. Aprenda os fundamentos (bindings, signals, handlers, properties)

2

Dê tempo ao tempo, sempre praticando. Demora um pouquinho para se acostumar ao modo declarativo de projetar software.

3

Arquitetura de software é importante. Muitos projetos QML sofrem de problemas arquiteturais.

4

QML é uma tecnologia para UI/UX. Não abuse de bindings e código JavaScript.

Qt



Obrigado!

---

**Sandro S. Andrade**  
sandroandrade@kde.org

**IFBA/KDE**