# Aplicando Padrões de Arquitetura em Qt / QML

Leonnardo Verol

# Um Oferecimento

Inatel
Instituto Nacional de Telecomunicações

TDC TRANSFORMATION
COMPETÊNCIAS PARA A EVOLUÇÃO
DAS PESSOAS E NEGÓCIOS

# Leonnardo Verol

Engenheiro de Computação & de Software

Especialista em Sistemas
Inatel Competence Center

Inatel

github.com/LeonnardoVerol

linkedin.com/in/leonnardoverol

# Hoje - QML

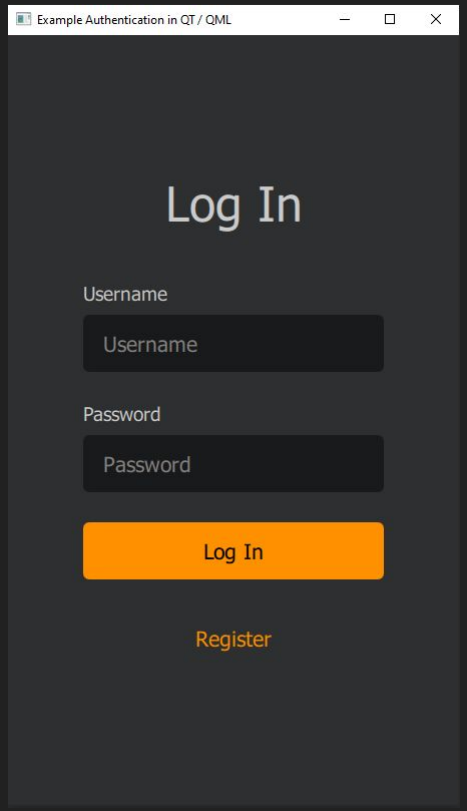
- Components (HTML, React & Others)
- Modules
- Flux Pattern (Facebook, React, VueJS)
- Navigation (VueJS, React)
- Responsive Breakpoints (Web)
- Demonstração

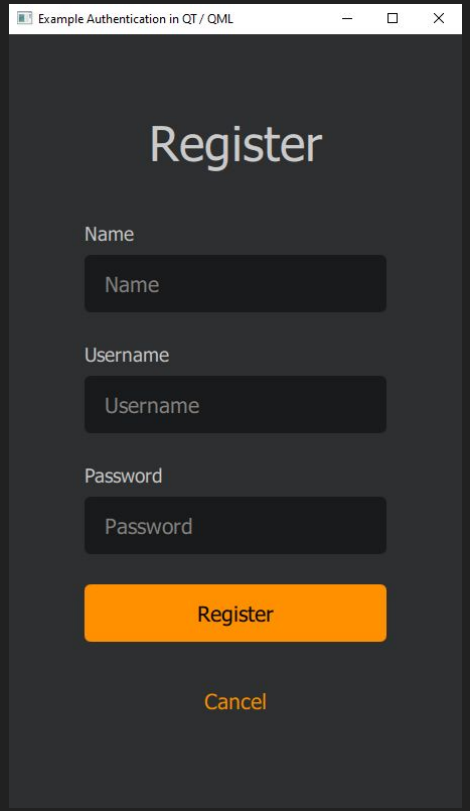
# Casos de Estudo

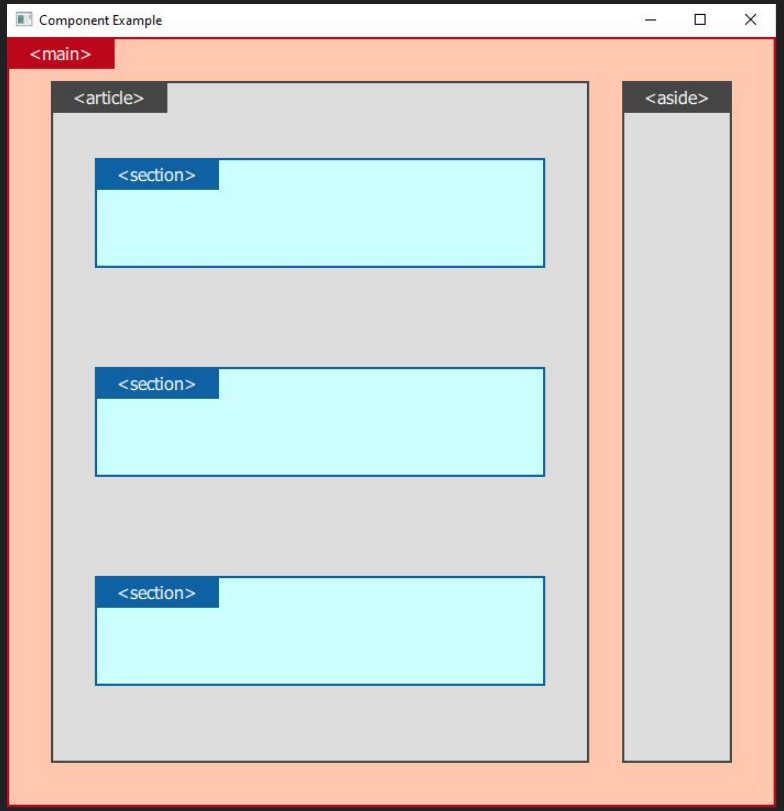
# Casos de Estudo

~/example-authentication-qt-qml

~/example-components-qt-qml



# Components

(Introdução & Definição)

# Components - Definição

- ➔ Trechos de código que serão reutilizados
- ➔ Encapsular uma lógica específica

Essencialmente, são “funções” de Interface

# Components - O Custo de um Código Confuso

“Alguma vez um código ruim já lhe atrasou consideravelmente?

(...) é como se caminhássemos penosamente por um lamaçal de arbustos emaranhados com armadilhas ocultas. Pelejamos para encontrar nosso caminho, esperando avistar alguma dica, alguma indicação do que está acontecendo; mas tudo o que vemos é um código cada vez mais sem sentido.”

- Código Limpo (Pag. 3)

# Components - O Custo de um Código Confuso

- ➔ Manutenibilidade
- ➔ Produtividade
- ➔ Custo

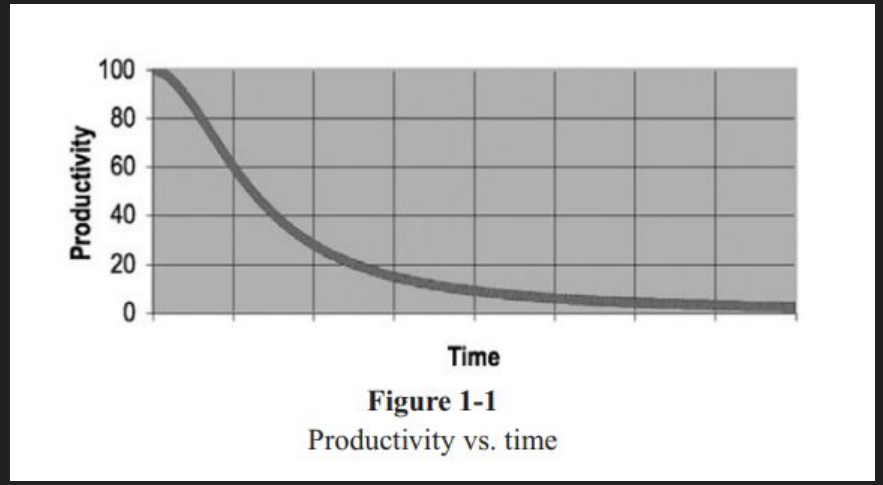



**Figure 1-1**
Productivity vs. time

Código Limpo (Pag. 4)

# Components - A Regra de Escoteiro

“Deixe a área do acampamento mais limpa do que como você a encontrou”

- Boy Scouts of America

# Components - A Regra de Escoteiro

“Se todos deixássemos nosso código mais limpo do que quando o começamos, ele simplesmente nao degradaria”

“Não basta escrever um código bom. ***Ele precisa ser mantido sempre limpo***.”

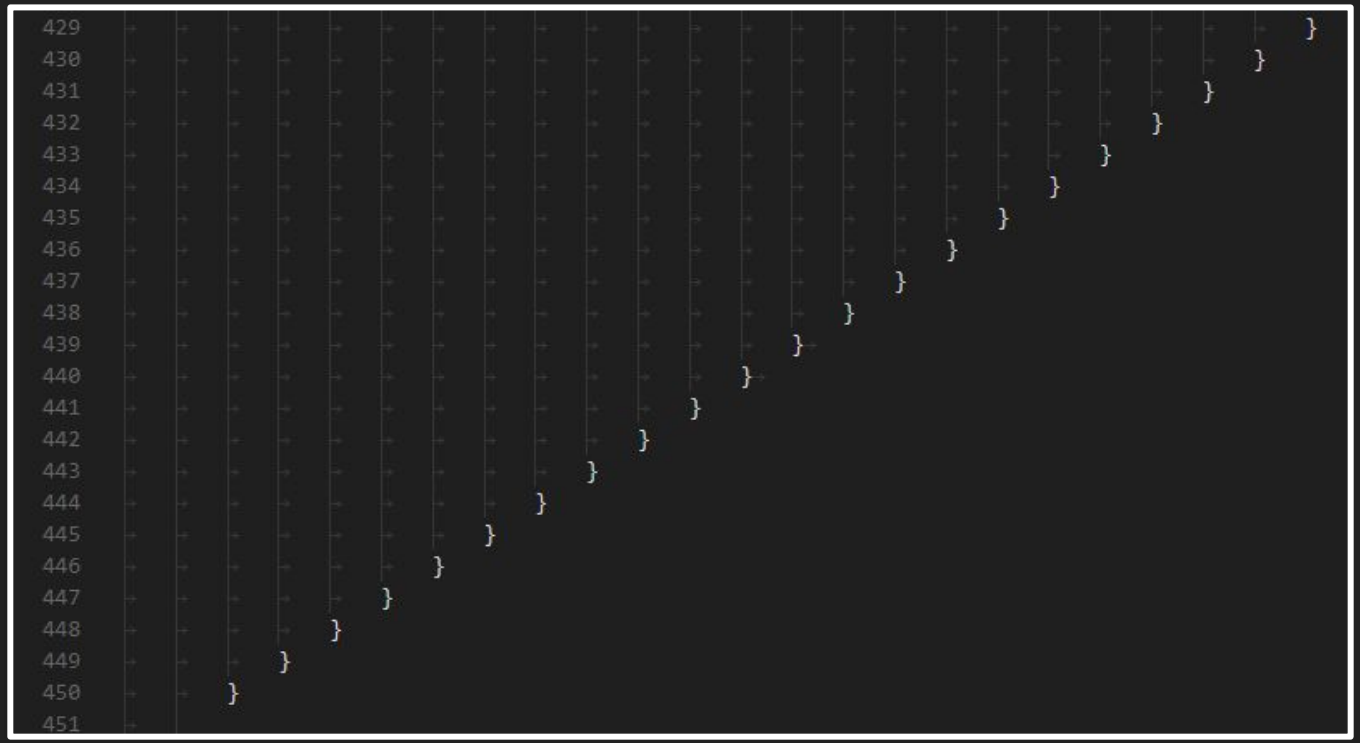
- Código Limpo (Pag. 14)

# Components

(“Bad Code”)

# Components - “Bad Code”

```
5879        }
5880
5881    }
5882
5883    #endif
5884
```



# Components - “Bad Code”

```
Rectangle {
    width: parent.width
    height: 80
    Rectangle {
        width: 50
        height: 50
        anchors.leftMargin: 20
        anchors.rightMargin: 20
        anchors.left: parent.left
        anchors.verticalCenter: parent.verticalCenter
    }
    Rectangle{
        anchors.left: logo.right
        anchors.right: parent.right
        height: parent.height
        RowLayout {
            spacing: 20
            height: parent.height
            anchors.right: parent.right
            Layout.alignment: Qt.AlignVCenter
            Layout.minimumWidth: 100

            Rectangle {
```

# Components

(Local Components)

# Components - Não me faça pensar

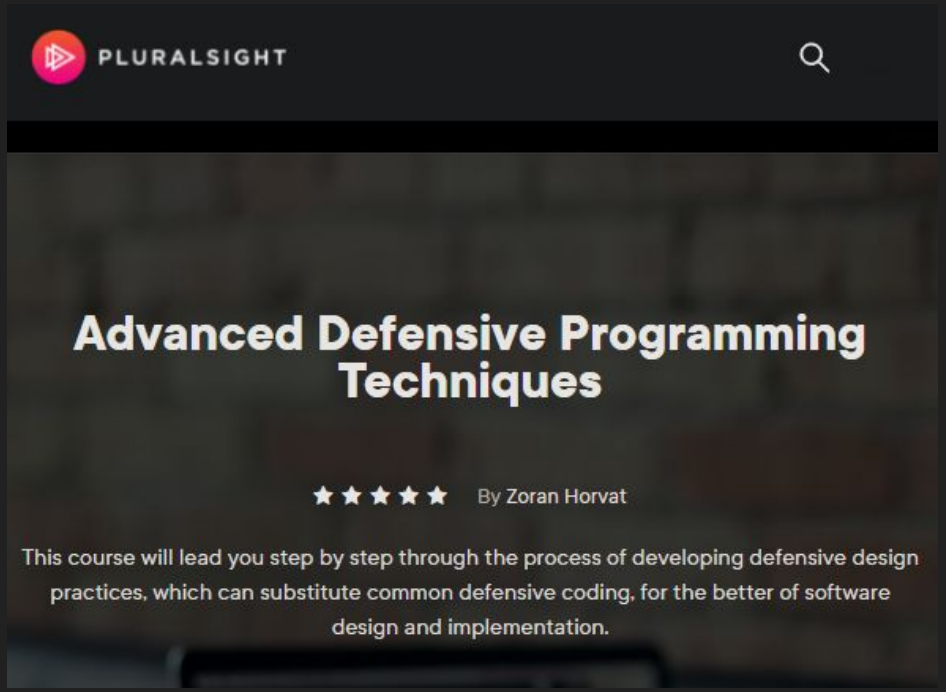
*“As coisas devem ser autoexplicativas”*

*“Não me faça pensar (e nem os outros)”*

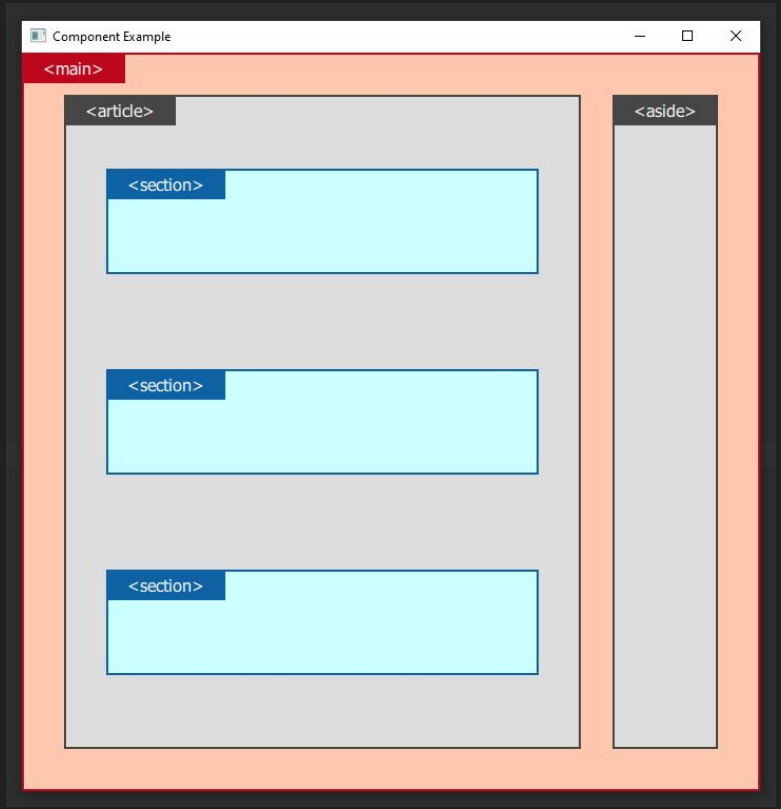
# Components - Defensive Programming

“Avoid Primitive Types”

“Primitive types don’t convey any meaningful domain knowledge”



# Components - A Semântica do HTML

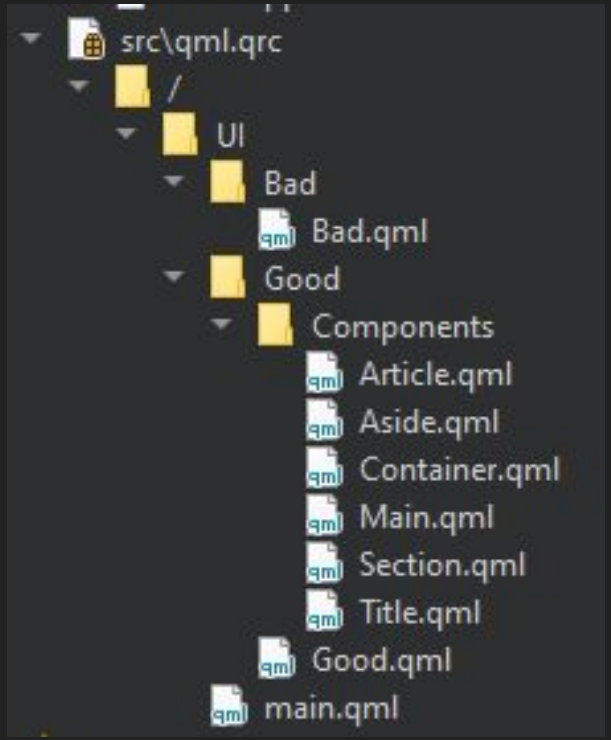


# Components - A Semântica do HTML

```
194             color: "#DDDDDD"
195
196             border.width: 2
197             border.color: "#464646"
198
199             anchors.left: article.right
200             anchors.leftMargin: 30
201
202             y: article.y
203
204             height: 450
205             width: name3.implicitWidth
206
207             Rectangle {
208                 color: "#464646"
209                 anchors.left: parent.left
210                 anchors.top: parent.top
211
212                 height: name3.implicitHeight
213                 width: name3.implicitWidth
214
215                 Text {
216                     id: name3
217                     text: qsTr("<aside>")
218                     color: "white"
219                     anchors.fill: parent
220                     anchors.centerIn: parent
221                     topPadding: 5
222                     bottomPadding: 5
223                     leftPadding: 20
224                     rightPadding: 20
225                     font.pointSize: 12
226                 }
227             }
228         }
229     }
230 }
231
232
```

```
25
26         Article {
27             id: article
28             Layout.fillHeight: true
29             Layout.fillWidth: true
30             clip: true
31
32             ColumnLayout {
33                 anchors.fill: parent
34                 anchors.margins: 40
35
36                 spacing: 30
37
38                 Section {
39                     height: 100
40                     Layout.fillWidth: true
41                 }
42
43                 Section {
44                     height: 100
45                     Layout.fillWidth: true
46                 }
47
48                 Section {
49                     height: 100
50                     Layout.fillWidth: true
51                 }
52             }
53         }
54
55         Aside {
56             id: aside
57             Layout.preferredWidth: 100
58             Layout.fillHeight: true
59         }
60     }
61 }
62
```

# Components - Local Components

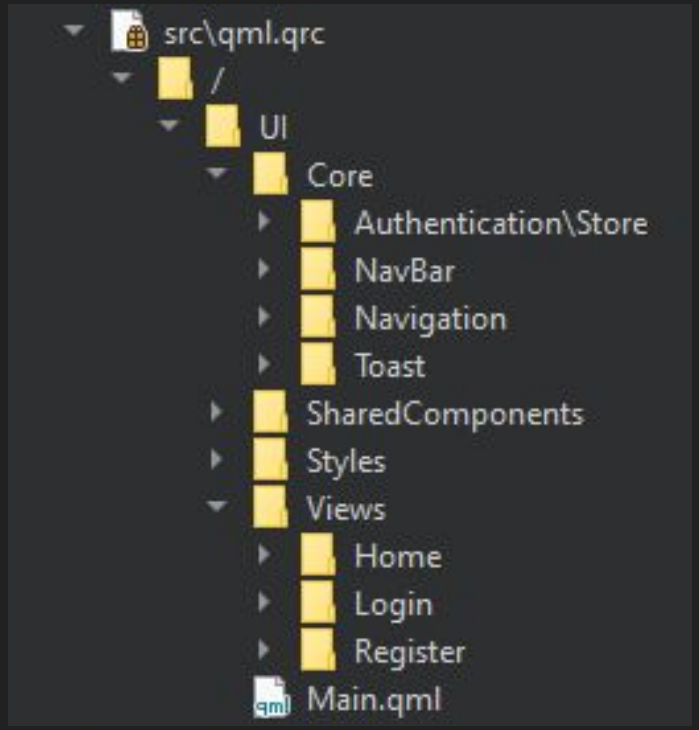


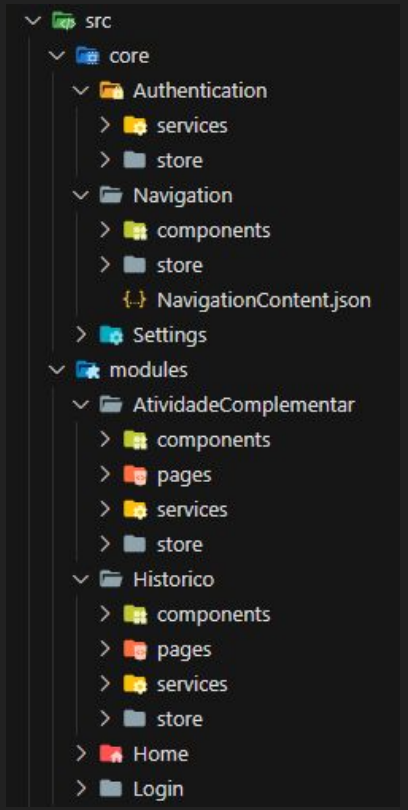
# Modules

# Modules

Crie módulos e organize-os de acordo com seu domínio.

Existem várias abordagens, não se prenda a uma.

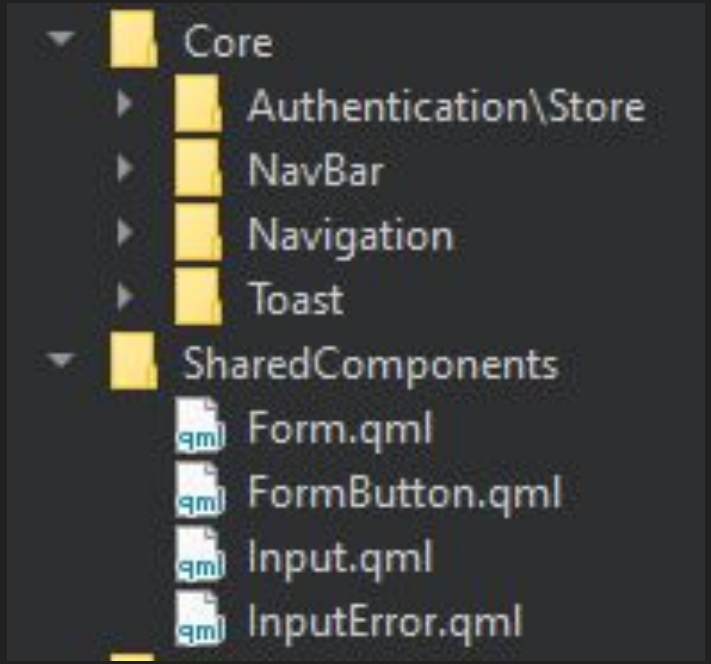


# Components

(Global Components)

# Components - Global Components



# Components

(Styled Components - Work in Progress Ideia)

# Components - Styled Components (WIP Ideia)

```
    return (
        <Container>
            <Content>
                <form onSubmit={handleSubmit}>
                    <h1>Log In</h1>

                    0 references
                    <input type="text" placeholder="Usuário" value={username} onChange={(event) => setUsername(event.target.value)} />
                    0 references
                    <input type="password" placeholder="Senha" value={password} onChange={(event) => setPassword(event.target.value)} />

                    <button type="submit">Entrar</button>
                </form>
            </Content>
        </Container>
    )
}
```

# Components - Styled Components (WIP Ideia)

```
export const Container = styled.div`
    height: 100vh;

    display: flex;
    align-items: center;
    justify-content: center;
`;
```

```
export const Content = styled.div`
    display: flex;
    justify-content: center;
    flex-direction: column;
    align-items: center;

    width: 100%;
    max-width: 700px;

    form {
        margin: 80px 0;
        width: 340px;
        text-align: center;

        h1 {
            margin-bottom: 24px;
        }

        input {
            background: #232129;
            border-radius: 10px;
            border: 2px solid #232139;
            padding: 16px;
            width: 100%;
            color: #F4EDE8;

            & + input {
                margin-top: 8px;
            }
        }
        button {
            background: #FF9000;
            color: #312E38;
            border-radius: 10px;
            border: 0;
            padding: 16px;
            width: 100%;

            font-weight: 500;
            margin-top: 16px;
        }
    }
`;
```

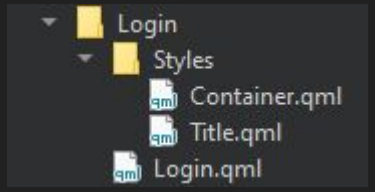
# Components - Styled Components (WIP Ideia)

```
Container {
    Title { text: qsTr("Log In") }

    Form {
        Input {
            id: username
            label.text: qsTr("Username")
            placeholderText: qsTr("Username")
        }

        Input {
            id: password
            label.text: qsTr("Password")
            placeholderText: qsTr("Password")
            echoMode: TextInput.Password
        }

        FormButton {
            id: submit
            text: qsTr("Log In")
            backgroundColor: Colors.button.primary.background
            backgroundHoverColor: Colors.button.primary.hovered
            Layout.fillWidth: true

            onClicked: { ...}
        }

        FormButton {
            text: qsTr("Register")
            textColor: Colors.button.secondary.text
            backgroundHoverBorderColor: Colors.button.secondary.hoveredBorder
            Layout.fillWidth: true

            onClicked: Navigation.push(ScreenTypes.REGISTER_SCREEN)
        }
    }
}
```

# Components - Styled Components (WIP Ideia)

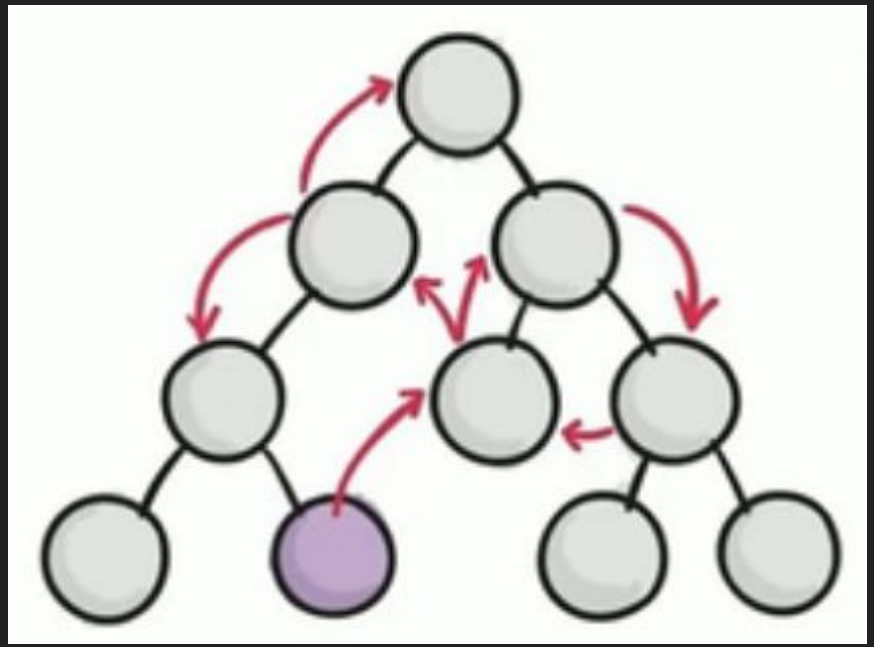
```
Container {
    Title { text: qsTr("Log In") }
```

```
ColumnLayout {
    anchors.centerIn: parent
    width: 300
    spacing: 50
}
```




```
Text {
    Layout.alignment: Qt.AlignHCenter
    font.pointSize: 36
    color: "#CBCBCB"
}
```

# Components - Comunicação entre os componentes



fonte: https://www.bilibili.com/s/video/BV1rE411A7uk

# Flux Pattern

(Introdução & Definição)

# Flux Pattern

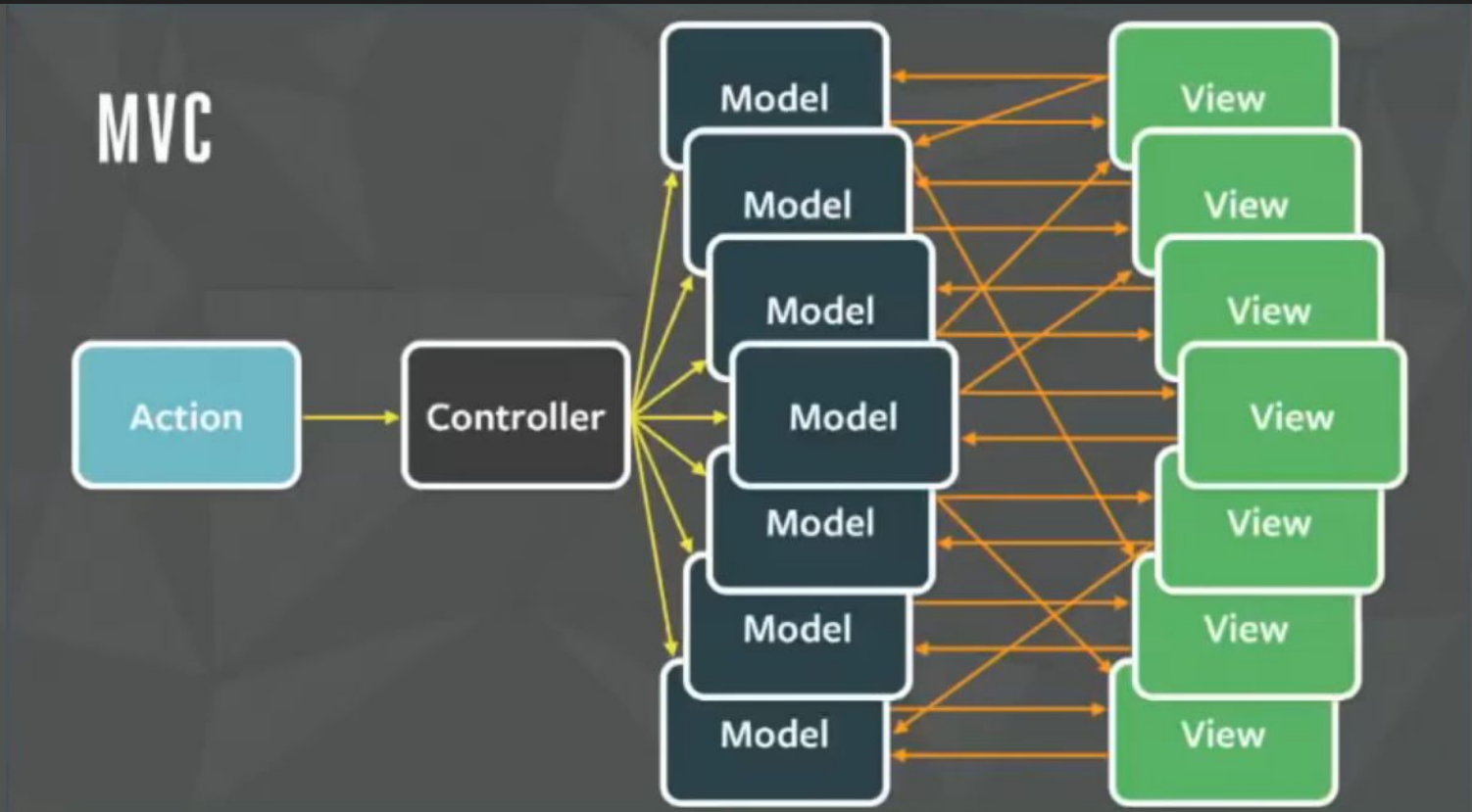
“Flux é a arquitetura que o Facebook usa, atualmente, para construir aplicações Front-End. Ele complementa os componentes do React, utilizando um fluxo de dados unidirecional. É mais um padrão (Pattern) do que um Framework, e você pode começar a usar o Flux imediatamente, sem muito código adicional. ”

Fonte

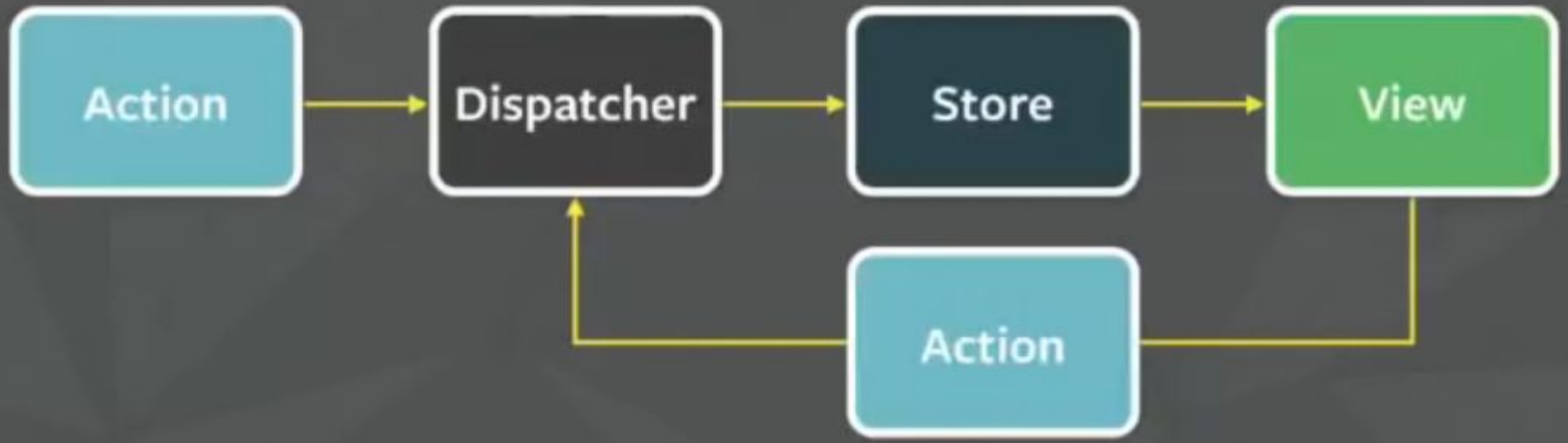
https://facebook.github.io/flux/docs/in-depth-overview

# Flux Pattern

# Flux Pattern



# Flux Pattern

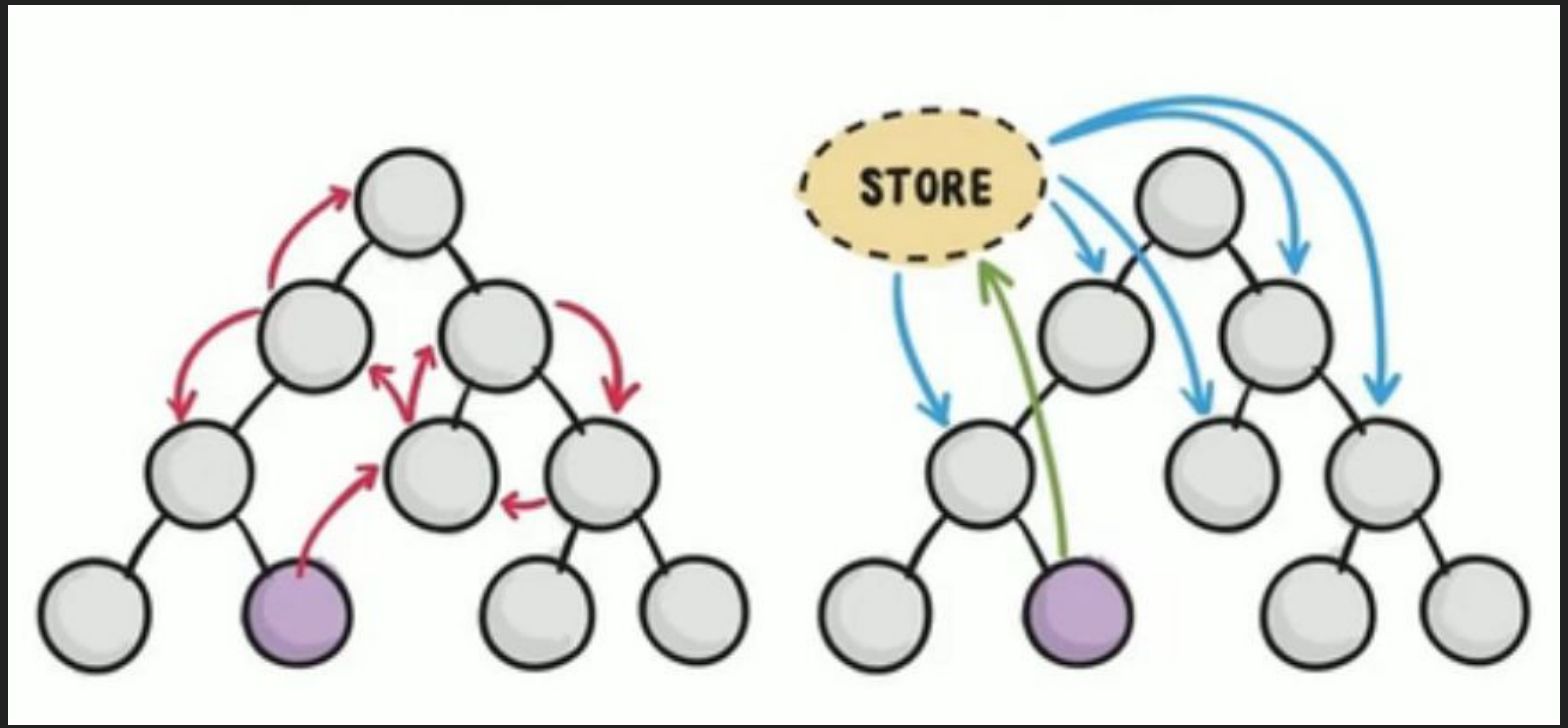


# Flux Pattern

## INCREASE PREDICTABILITY

- Improved data consistency
- Easier to pinpoint root of a bug
- More meaningful unit tests

# Flux Pattern




fonte: https://www.bilibili.com/s/video/BV1rE411A7uk

# Flux Pattern

(Authentication Store)

# Flux Pattern - Authentication Store

```
1   import QtQuick 2.15
2   import "Types.js" as Types
3
4   pragma Singleton
5   Item {
6       property var fakeDatabase: [
7           {name:"admin",username:"admin",password:"admin"}
8       ]
9       property string status: ""
10      property string errorMessage: ""
11
12      property QtObject user: QtObject {
13          property string name
14      }
15
16
17      property var commit: (function(state, payload = undefined) { ...
64      })
65
66      function register(payload)
67      { ...}
80
81      function login(payload)
82      { ...}
101
102     function logoff()
103     { ...}
106 }
107
```

# Flux Pattern - Authentication Store

```
property var commit: (function(state, payload = undefined) {
    const mutations = { [...} 

    mutations[state](payload);
})
```

Types.js

```
const REGISTER_REQUEST = "REGISTER_REQUEST"
const REGISTER_ERROR = "REGISTER_ERROR"
const REGISTER_SUCCESS = "REGISTER_SUCCESS"

const AUTH_REQUEST = "AUTH_REQUEST"
const AUTH_ERROR = "AUTH_ERROR"
const AUTH_SUCCESS = "AUTH_SUCCESS"
const AUTH_LOGOFF = "AUTH_LOGOFF"
```

```
[Types.REGISTER_REQUEST]: function (payload)
{
    status = "Fake API Request";
    errorMessage = "";
},
[Types.REGISTER_ERROR]: function (payload)
{
    status = "Register Failed"
    errorMessage = payload
},
[Types.REGISTER_SUCCESS]: function (payload)
{
    status = "Register Successful";
    errorMessage = "";

    fakeDatabase.push(payload)
    console.log(JSON.stringify(fakeDatabase))
},
[Types.AUTH_REQUEST]: function (payload)
{
    status = "Fake API Request";
    errorMessage = "";
},
[Types.AUTH_ERROR]: function (payload)
{
    status = "Login Failed";
    errorMessage = payload;
},
[Types.AUTH_SUCCESS]: function (payload)
{
    status = "Login Successful";
    errorMessage = "";

    user.name = payload.name
},
[Types.AUTH_LOGOFF]: function (payload)
{
    status = "Log Off Successful";
    errorMessage = "";

    user.name = ""
},
```

# Flux Pattern - Authentication Store

```
function register(payload)
{
    commit(Types.REGISTER_REQUEST);

    if(fakeDatabase.find(user => user.username === payload.username))
    {
        commit(Types.REGISTER_ERROR, "Username Already Exists!");
        throw { status, errorMessage };
    }
    else
    {
        commit(Types.REGISTER_SUCCESS, payload);
    }
}
```

```
[Types.REGISTER_REQUEST]: function (payload)
{
    status = "Fake API Request";
    errorMessage = "";
},
[Types.REGISTER_ERROR]: function (payload)
{
    status = "Register Failed"
    errorMessage = payload
},
[Types.REGISTER_SUCCESS]: function (payload)
{
    status = "Register Successful";
    errorMessage = "";

    fakeDatabase.push(payload)
},
```

# Flux Pattern - Authentication Store

```
function login(payload)
{
    commit(Types.AUTH_REQUEST)

    const userDB = fakeDatabase.find(user => user.username === payload.username);

    if( userDB === undefined)
    {
        commit(Types.AUTH_ERROR, "User Not Found!");
        throw { status, errorMessage };
    }

    if( userDB.password !== payload.password)
    {
        commit(Types.AUTH_ERROR, "Invalid Password!");
        throw { status, errorMessage };
    }

    commit(Types.AUTH_SUCCESS, userDB);
}

function logoff()
{
    commit(Types.AUTH_LOGOFF)
}
```

```
[Types.AUTH_REQUEST]: function (payload)
{
    status = "Fake API Request";
    errorMessage = "";
},
[Types.AUTH_ERROR]: function (payload)
{
    status = "Login Failed";
    errorMessage = payload;
},
[Types.AUTH_SUCCESS]: function (payload)
{
    status = "Login Successful";
    errorMessage = "";

    user.name = payload.name
},
[Types.AUTH_LOGOFF]: function (payload)
{
    status = "Log Off Successful";
    errorMessage = "";

    user.name = ""
},
```

# Navigation

# Navigation

```
signal pushSignal(string url)
signal popSignal(string url);

property var breadcrumbs: [initialPage]

property string initialPage: ScreenTypes.LOGIN_SCREEN

readonly property var routes: [
    {
        id: ScreenTypes.LOGIN_SCREEN,
        name: "Login",
        path: "qrc:/UI/Views/Login/Login.qml",
    },
    {
        id: ScreenTypes.REGISTER_SCREEN,
        name: "Register",
        path: "qrc:/UI/Views/Register/Register.qml",
    },
    {
        id: ScreenTypes.HOME_SCREEN,
        name: "Home",
        path: "qrc:/UI/Views/Home/Home.qml",
    }
]

function getActiveScreenName()
{ ...}

function router(id)
{ ...}

function push(id)
{ ...}

function goBack()
{ ...}

function pop()
{ ...}
```

```
Navigation.push(ScreenTypes.HOME_SCREEN)
```

```
Navigation.push(ScreenTypes.REGISTER_SCREEN)
```

```
Navigation.goBack();
```

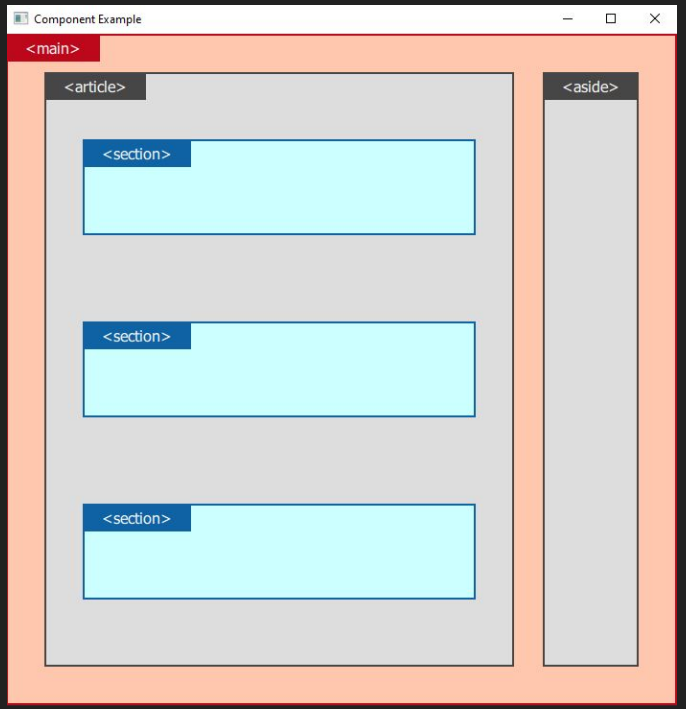
# Navigation - main.qml

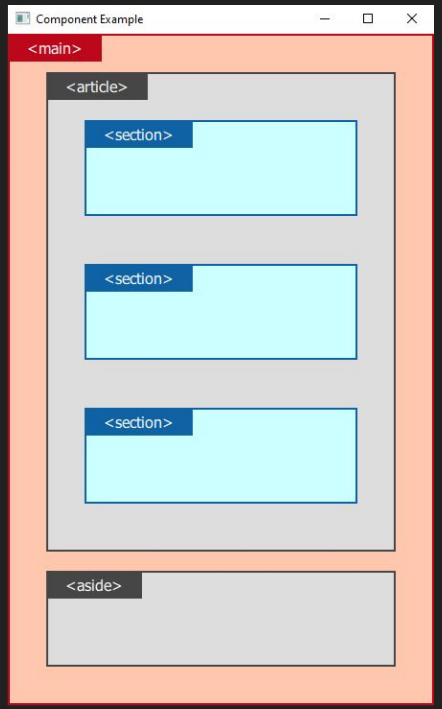
main.qml Window

```qml
import QtQuick 2.15
import QtQuick.Window 2.15

import Styles.Colors 1.0
import "Core/Navigation"
import "Core/Toast"

Window {
    minimumWidth: 450
    width: 450
    minimumHeight: 768
    height: 768
    color: Colors.application.background

    visible: true
    title: qsTr("Example Authentication in QT / QML")

    Navigation {}

    ToastManager {}
}
```

# Responsive Breakpoints

# Responsive Breakpoints



# Responsive Breakpoints

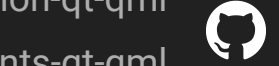
```
states: [
    State {
        name: "MOBILE_BREAKPOINT"
        when: width < 450
        PropertyChanges { target: gridLayout; columns: 1 }
        PropertyChanges {
            target: aside;
            Layout.fillWidth: true;
            Layout.fillHeight: false;
            Layout.preferredHeight: 100
        }
        PropertyChanges { target: article; Layout.bottomMargin: 15 }
    }
]
```

# Demonstração

(Código Disponível no Github)

~/example-authentication-qt-qml

~/example-components-qt-qml



# Obrigado


Inatel
Instituto Nacional de Telecomunicações


TDC TRANSFORMATION
COMPETÊNCIAS PARA A EVOLUÇÃO
DAS PESSOAS E NEGÓCIOS