APACHE
CORDOVA™

Hugo Deiró

/hdeiro

/in/hdeiro

hugodeiro@gmail.com

/hdeiro

hdeiro.github.io

Carlos Alício

/CarlosAlicio

in/carlosalicio-developer

carlosalicio.developer@gmail.com

/alicio.andradedecarvalho

cordova-test-app

https://hugodeiro.me/app.apk

O *smartphone* é algo essencial para o nosso cotidiano. Com ele nos informamos, trocamos mensagens, nos divertimos, trabalhamos, e realizamos outras mil e uma atividades.

Fonte: Revista Exame

Quase todo mundo usa. Sua mãe, seu pai, seu irmão, inclusive você (que provavelmente está com ele na mão agora).

Por consequência disso, é necessário prover aplicações que auxiliem o usuário na realização de suas tarefas. É aí que entram os famigerados

Aplicativos

Existem diversas tecnologias para trabalhar com o desenvolvimento de aplicações *mobile*, Android, Switft, Kiwi, NativeScript, Qt, React Native, **Cordova**, e várias outras.

O **Apache Cordova** é um *framework open source* que permite a criação de aplicativos móveis para múltiplas plataformas utilizando basicamente código HTML, CSS e JavaScript.

O *framework* é originado de uma ferramenta criada pela Nitobi, o Phonegap. Contudo, em 2011 a empresa foi adquirida pela Adobe.

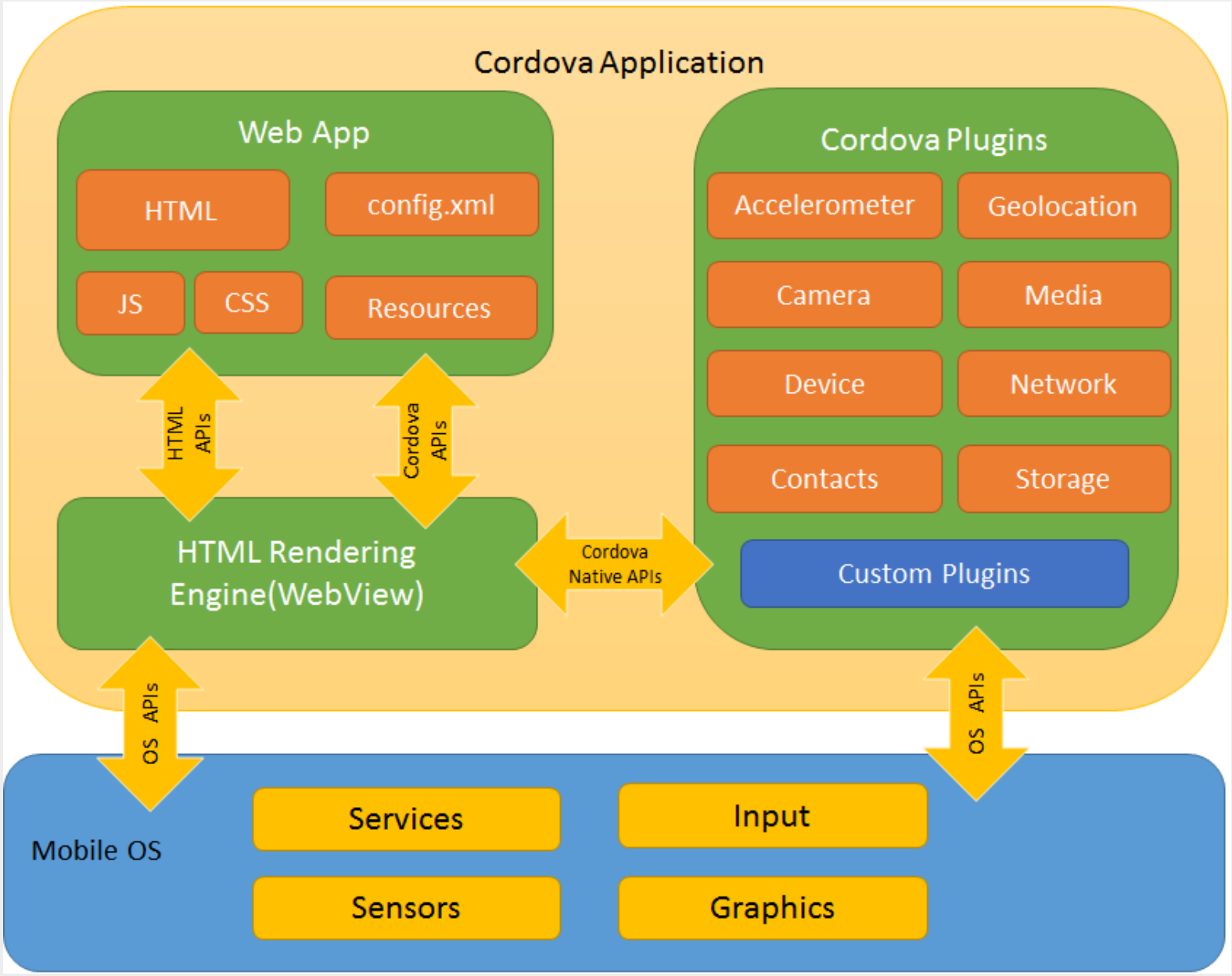
Algum tempo depois da transação, o código do Phonegap foi disponibilizado para a Apache, que criou o Cordova.

O Cordova foi desenvolvido utilizando JavaScript, Java, Objective-C, C++, C# e Node.JS, já teve 132 releases e se encontra na versão 8.X.X (Estável).

Seu código-fonte pode ser encontrado nos repositórios da Apache Foundation do Github.

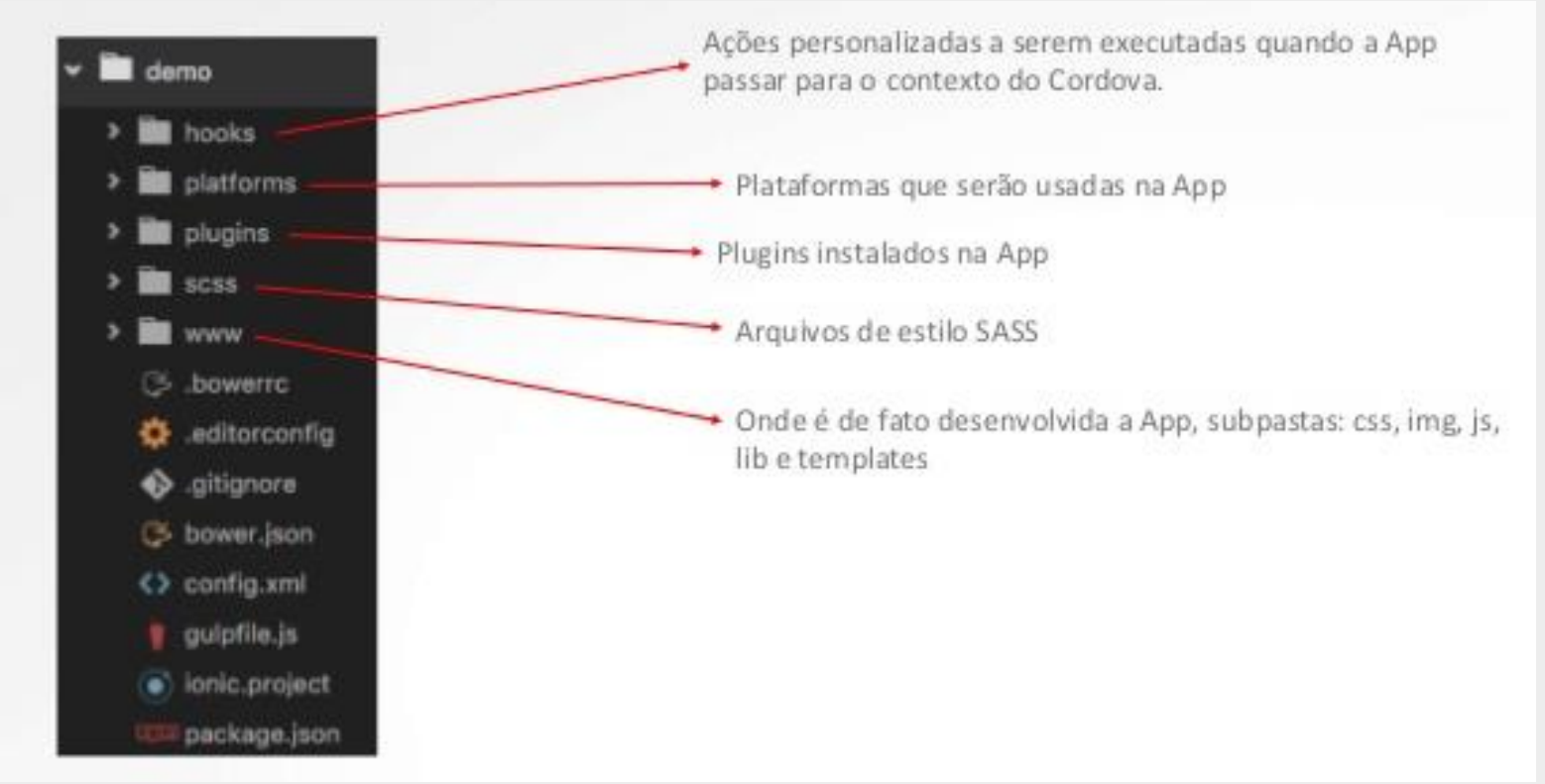
Seu funcionamento se dá a partir da renderização do código-fonte produzido em HTML, CSS e JavaScript em uma *Webview* nativa que pode se comunicar com recursos de *hardware* a partir de plugins.

A arquitetura do *framework* segue o estilo arquitetural *Virtual Machine* de forma que há uma abstração dos componentes do SO *Mobile* e do próprio Cordova que podem se conectar das mais diversas formas.

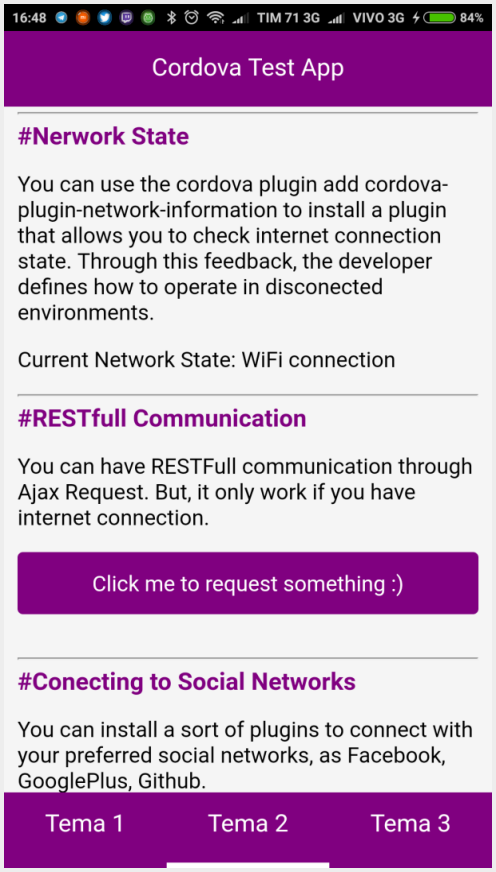


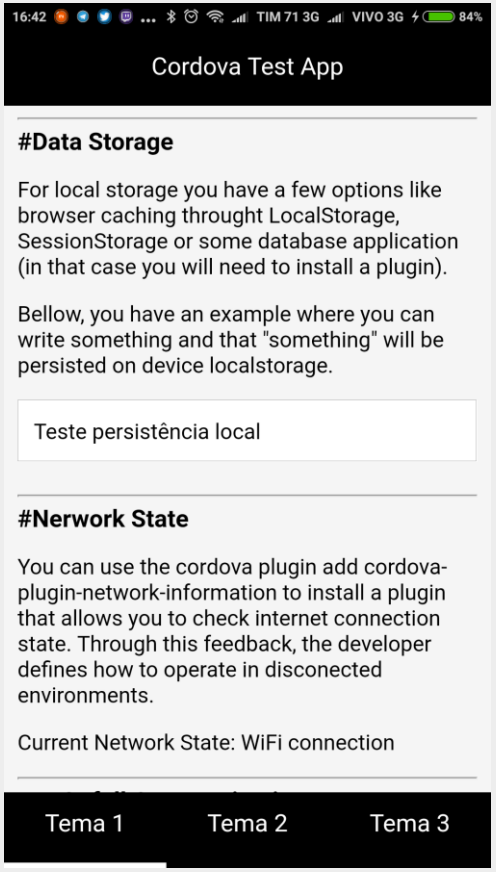
Fonte: Documentação do Apache Cordova

Estrutura de Diretórios



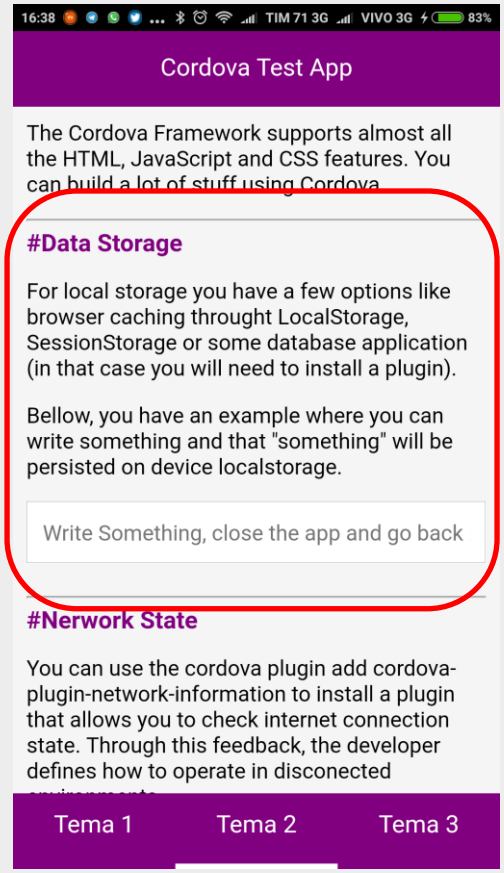
Suporte a Elementos de GUI e Temas

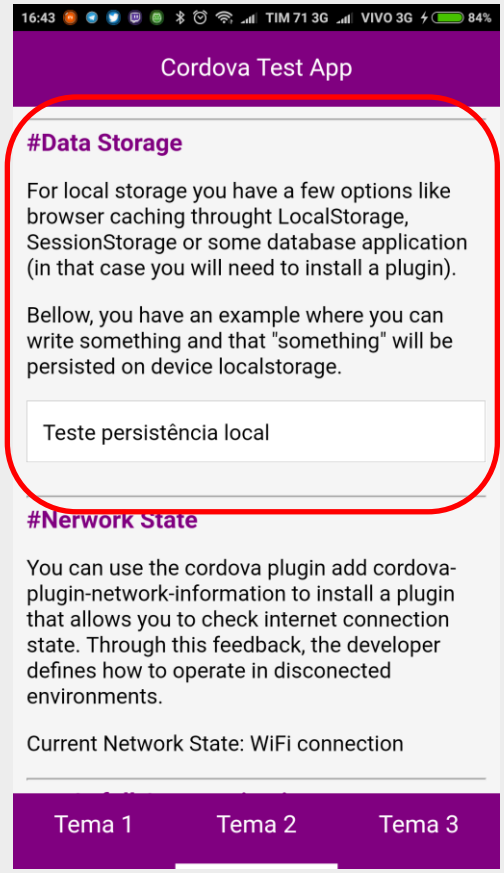





```
const tablist = $$('.tab');

tablist.forEach(tab => tab.addEventListener('click', event => {
    let tab = event.srcElement;

    //Add Theme to body
    body.classList.remove('theme-1', 'theme-2', 'theme-3');
    let themeToBeAdded = tab.attributes['data-theme'].value;
    body.classList.add(themeToBeAdded);

    //Add active to tab
    tablist.forEach(tab => tab.classList.remove('active'));
    tab.classList.add('active');
}));
```

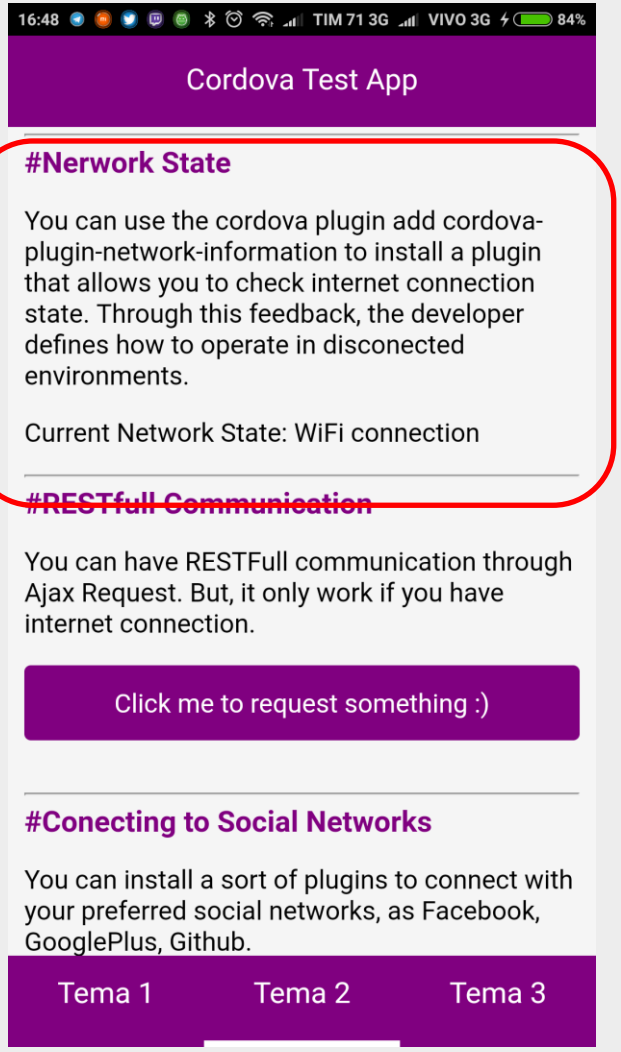
Cache/Persistência local





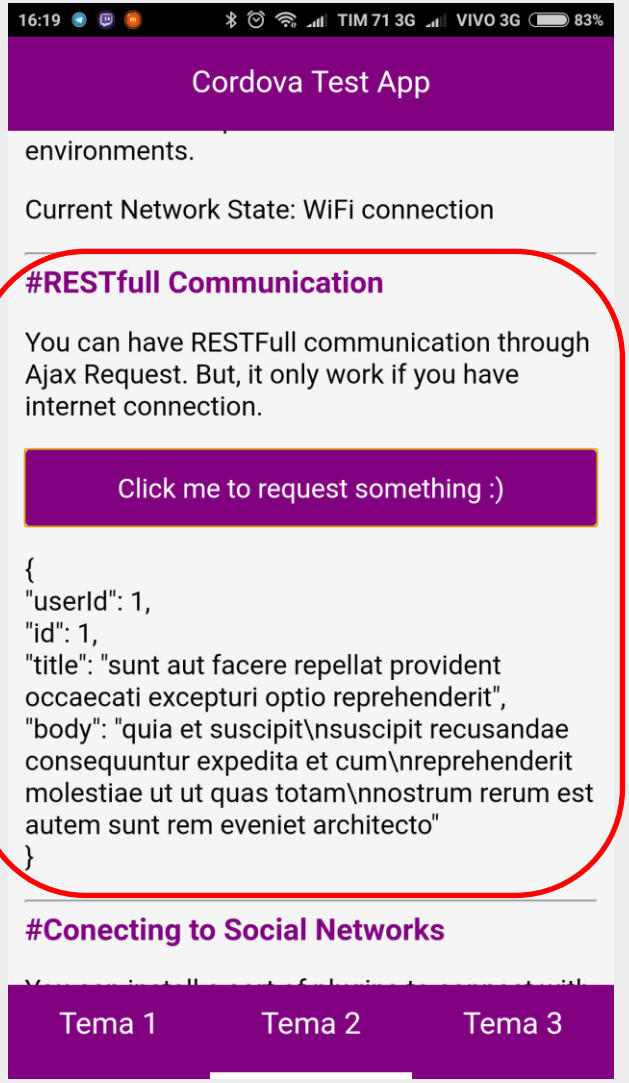

```
const cachable = $('#cachable');
let controlCachableTimeout = null;

cachable.value = localStorage.getItem('cachable');

cachable.addEventListener('input', event => {
    clearTimeout(controlCachableTimeout);

    controlCachableTimeout = setTimeout(() => {
        localStorage.setItem('cachable', cachable.value);
    }, 300);
});
```

Operações desconectadas




```
const networkInfo = $('#networkInfo');

function checkConnection() {
    try {
        var networkState = navigator.connection.type;

        var states = {};
        states[Connection.UNKNOWN]  = 'Unknown connection';
        states[Connection.ETHERNET] = 'Ethernet connection';
        states[Connection.WIFI]     = 'WiFi connection';
        states[Connection.CELL_2G]  = 'Cell 2G connection';
        states[Connection.CELL_3G]  = 'Cell 3G connection';
        states[Connection.CELL_4G]  = 'Cell 4G connection';
        states[Connection.CELL]     = 'Cell generic connection';
        states[Connection.NONE]     = 'No network connection';

        networkInfo.innerText = states[networkState];
    } catch(exception) {
        networkInfo.innerText = 'Undetected network state';
    }
}

setInterval(checkConnection, 3000);

document.addEventListener("offline", () => {
    checkConnection();
    body.classList.add('theme-offline');
}, false);

document.addEventListener("online", () => {
    checkConnection();
    body.classList.remove('theme-offline');
}, false);
```

Comunicação via RESTful




```
const ajaxButton = $('#ajaxButton');
ajaxButton.addEventListener('click', event => {
    var xhttp = new XMLHttpRequest();
    xhttp.overrideMimeType("application/json");
    xhttp.open("GET", "https://jsonplaceholder.typicode.com/posts/1", true);

    xhttp.onreadystatechange = function() {
        if(xhttp.readyState == 4 && xhttp.status == "200") {
            $('#jsonResponse').innerText = xhttp.responseText;
        } else {
            $('#jsonResponse').innerText = 'Something went wrong :(';
        }
    }
    xhttp.send();
});
```

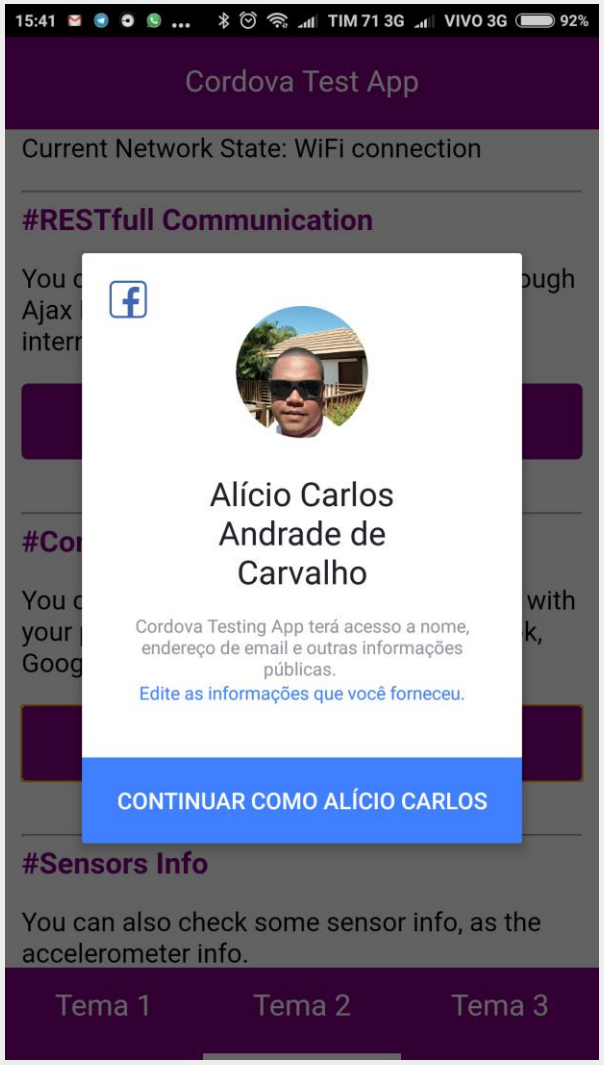
Push notification

A entrega de *push notifications* pode ser feita através da utilização de serviços como o OneSignal, que normalmente disponibilizam um SDK e um *endpoint* para uma API Restfull que permite você enviar e receber as notificações.

Cordova

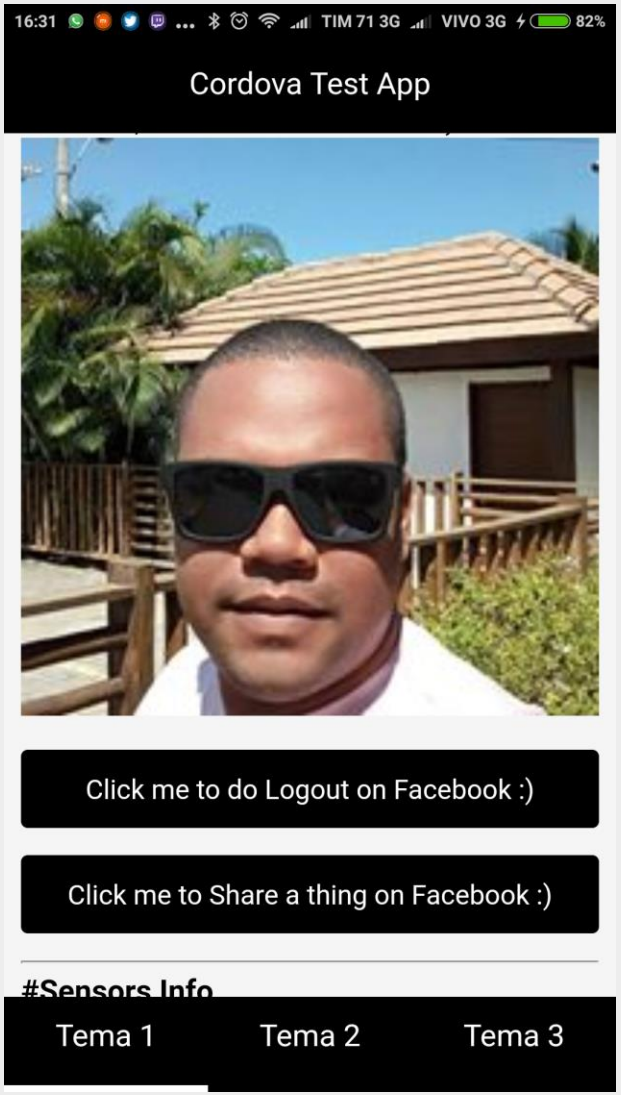
```
// Add to index.js or the first page that loads with your app.
// For Intel XDK and please add this to your app.js.

document.addEventListener('deviceready', function () {
  // Enable to debug issues.
  // window.plugins.OneSignal.setLogLevel({logLevel: 4, visualLevel: 4});

  var notificationOpenedCallback = function(jsonData) {
    console.log('notificationOpenedCallback: ' + JSON.stringify(jsonData));
  };

  window.plugins.OneSignal
    .startInit("YOUR_APPID")
    .handleNotificationOpened(notificationOpenedCallback)
    .endInit();

  // Call syncHashedEmail anywhere in your app if you have the user's email.
  // This improves the effectiveness of OneSignal's "best-time" notification scheduling feature.
  // window.plugins.OneSignal.syncHashedEmail(userEmail);
}, false);
```

Integração com redes sociais




```
const facebookButton = $('#fb');
facebookButton.addEventListener('click', event => {
    facebookConnectPlugin.login(
        ['email', 'public_profile'],
        success => {
            facebookConnectPlugin.api(
                `/me?fields=email,name&access_token=${success.authResponse.accessToken}`,
                null,
                success => {
                    $('#fbResponse').innerHTML = `${JSON.stringify(success)}<img src="http:
                    body.classList.add('facebook-logged');
                },
                error => console.log(error)
            );
        },
        error => console.log(error)
    );
});
```

Integração com redes sociais




```
const facebookShareButton = $('#fbshare');
facebookShareButton.addEventListener('click', event => {
    facebookConnectPlugin.showDialog({
        method: "share",
        href:'https://sdtimes.com/wp-content/uploads/2015/10/1001.sdt-apache-cordova.png',
        name:'Testing post on facebook from Cordova App (name)',
        message:'Testing post on facebook from Cordova App (message)',
        caption: 'Testing post on facebook from Cordova App (caption)',
        description: 'Testing post on facebook from Cordova App (description)'
    }, function (response) {
        console.log(response)
    }, function (response) {
        console.log(response)
    });
});

const facebookLogoutButton = $('#fblogout');
facebookLogoutButton.addEventListener('click', event => {
    facebookConnectPlugin.logout(success => {
        console.log(success);
        body.classList.remove('facebook-logged');
        $('#fbResponse').innerHTML = ``;
    }, error => {
        console.log(error);
    });
});
```

Integração com outros aplicativos

A integração de aplicativos pode acontecer através de plugins que intermediam esta comunicação.

Exemplos: cordova-facebook4 e InAppBrowser

Acesso a dados locais

É possível acessar dados locais, como contatos, por exemplo, através de plugins.

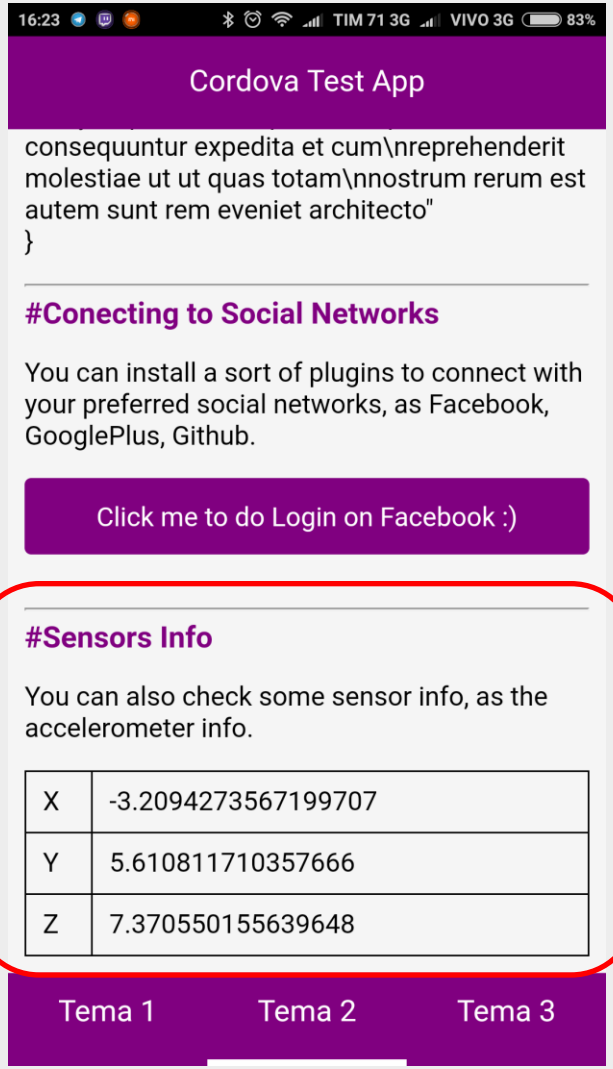
Ex: cordova-plugin-contacts

Example

```
    var myContact = navigator.contacts.create({"displayName": "Test User"});
```

```
function onSuccess(contacts) {
    alert('Found ' + contacts.length + ' contacts.');
};

function onError(contactError) {
    alert('onError!');
};

// find all contacts with 'Bob' in any name field
var options      = new ContactFindOptions();
options.filter   = "Bob";
options.multiple = true;
options.desiredFields = [navigator.contacts.fieldType.id];
options.hasPhoneNumber = true;
var fields       = [navigator.contacts.fieldType.displayName, navigator.contacts.fieldType.name];
navigator.contacts.find(fields, onSuccess, onError, options);
```

Acesso a sensores




```
//#######################################################
// SENSORS
//#######################################################
setInterval(() => {
    if(!navigator.accelerometer)
        return;

    navigator.accelerometer.getCurrentAcceleration(
        success => {
            $('#xval').innerText = success.x;
            $('#yval').innerText = success.y;
            $('#zval').innerText = success.z;
        },
        error => console.log(error)
    );
}, 300);
```

Comunicação via NFC

Existem plugins que disponibilizam interação via NFC.

Ex: phonegap-nfc

nfc.addNdefListener

Registers an event listener for any NDEF tag.

```
nfc.addNdefListener(callback, [onSuccess], [onFailure]);
```

nfc.addTagDiscoveredListener

Registers an event listener for tags matching any tag type.

```
nfc.addTagDiscoveredListener(callback, [onSuccess], [onFailure]);
```

Methods

- nfc.addNdefListener
- nfc.removeNdefListener
- nfc.addTagDiscoveredListener
- nfc.removeTagDiscoveredListener
- nfc.addMimeTypeListener
- nfc.removeMimeTypeListener
- nfc.addNdefFormatableListener
- nfc.write
- nfc.makeReadOnly
- nfc.share
- nfc.unshare
- nfc.erase
- nfc.handover
- nfc.stopHandover
- nfc.enabled
- nfc.showSettings
- nfc.beginSession
- nfc.invalidateSession

Comunicação via Bluetooth

Existem plugins que disponibilizam interação via Bluetooth.

Ex: cordova-plugin-bluetooth-serial

connect

Connect to a Bluetooth device.

```
bluetoothSerial.connect(macAddress_or_uuid, connectSuccess, connectFailure);
```

discoverUnpaired

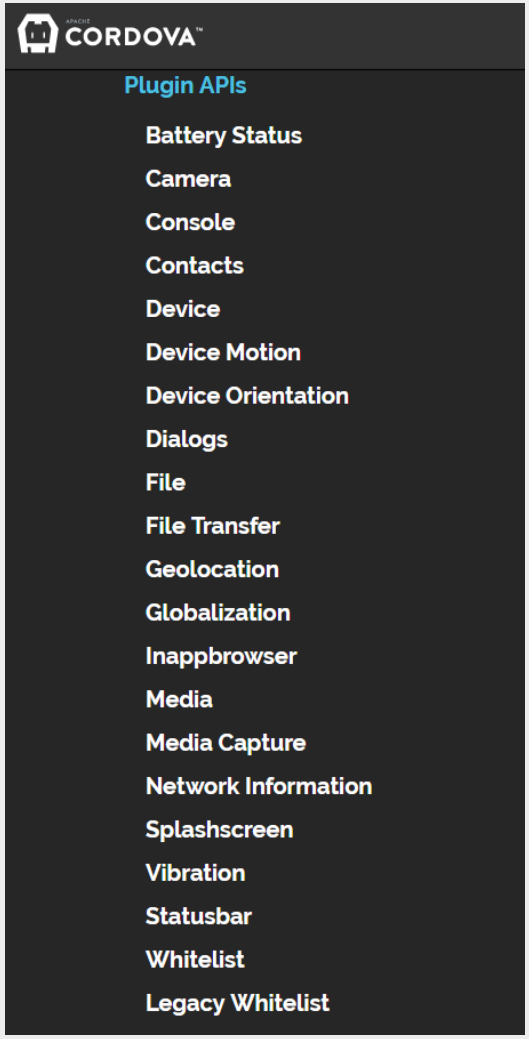
Discover unpaired devices

```
bluetoothSerial.discoverUnpaired(success, failure);
```

Methods

- bluetoothSerial.connect
- bluetoothSerial.connectInsecure
- bluetoothSerial.disconnect
- bluetoothSerial.write
- bluetoothSerial.available
- bluetoothSerial.read
- bluetoothSerial.readUntil
- bluetoothSerial.subscribe
- bluetoothSerial.unsubscribe
- bluetoothSerial.subscribeRawData
- bluetoothSerial.unsubscribeRawData
- bluetoothSerial.clear
- bluetoothSerial.list
- bluetoothSerial.isEnabled
- bluetoothSerial.isConnected
- bluetoothSerial.readRSSI
- bluetoothSerial.showBluetoothSettings
- bluetoothSerial.enable
- bluetoothSerial.discoverUnpaired
- bluetoothSerial.setDeviceDiscoveredListener
- bluetoothSerial.clearDeviceDiscoveredListener
- bluetoothSerial.setName
- bluetoothSerial.setDiscoverable

Outros recursos



Através de plugins o cordova dá suporte a vários outros recursos.

Pontos a serem considerados

- Experiência Mobile / User Experience (UX)
  - Material Design
  - iOS Human Interface Guidelines
- Single Page Application (SPA)
- Frameworks
  - Angular JS
  - Ionic

Conclusão

O Cordova é um framework de fácil extensão o e conceito simples, que busca facilitar o trabalho dos desenvolvedores durante a criação de um aplicativo que precise operar sob diferentes sistemas operacionais móveis.

Com tudo deve-se ter atenção na utilização de determinados recurso para não prejudicar a usabilidade devido a problemas de performance.

Outro aspecto a ser considerado é a preocupação com a experiência mobile que será entregue ao usuário.


That's all Folks!

Referências

Phonegap NFC, disponível em https://github.com/chariotsolutions/phonegap-nfc;

BluetoothSerial, disponível em https://github.com/don/BluetoothSerial#connect;

Cordova-Facebook4, disponível em https://github.com/jeduan/cordova-plugin-facebook4;

Documentação do Apache Cordova, disponível em https://cordova.apache.org;