Software Engineering Conference Russia

October 2017, St. Petersburg

# Improvement of hybrid solutions for the development of cross-platform mobile applications

Dmitry Soldatenkov & Alexander Epifanov

TAU TECHNOLOGIES

## About presenters

**Dmitry Soldatenkov**

Co-founder,CTO TAU TECHNOLOGiES

In high school I understood that programming is my main interest
I worked in one of the first Russian companies to develop computer games. Then work on a large product for developers (TogetherSoft / Borland). During my career my interests shifted to the mobile platforms, and currently I work in this area more than 13 years.

I worked at Vivendi, TogetherSoft, Borland, LG Electronics, TWP, RhoMobile, Motorola Solutions, Zebra Technologies.

Details : https://www.linkedin.com/in/dsoldatenkov

E-Mail: dsoldatenkov@tau-technologies.com

**Alexander Epifanov**

Co-founder,VP Technology TAU TECHNOLOGiES

Mobile and embedded expert and manager.
More than 11 years of experience in mobile development.

Worked at TWP, RhoMobile, Motorola Solutions, Zebra Technologies and other companies.

Details : https://www.linkedin.com/in/aepifanov

E-Mail: aepifanov@tau-technologies.com

## Remark

**Time limit**

Since this presentation has strong time limit we will skip some parts, but you can download full presentation from our web site http://tau-technologies.com

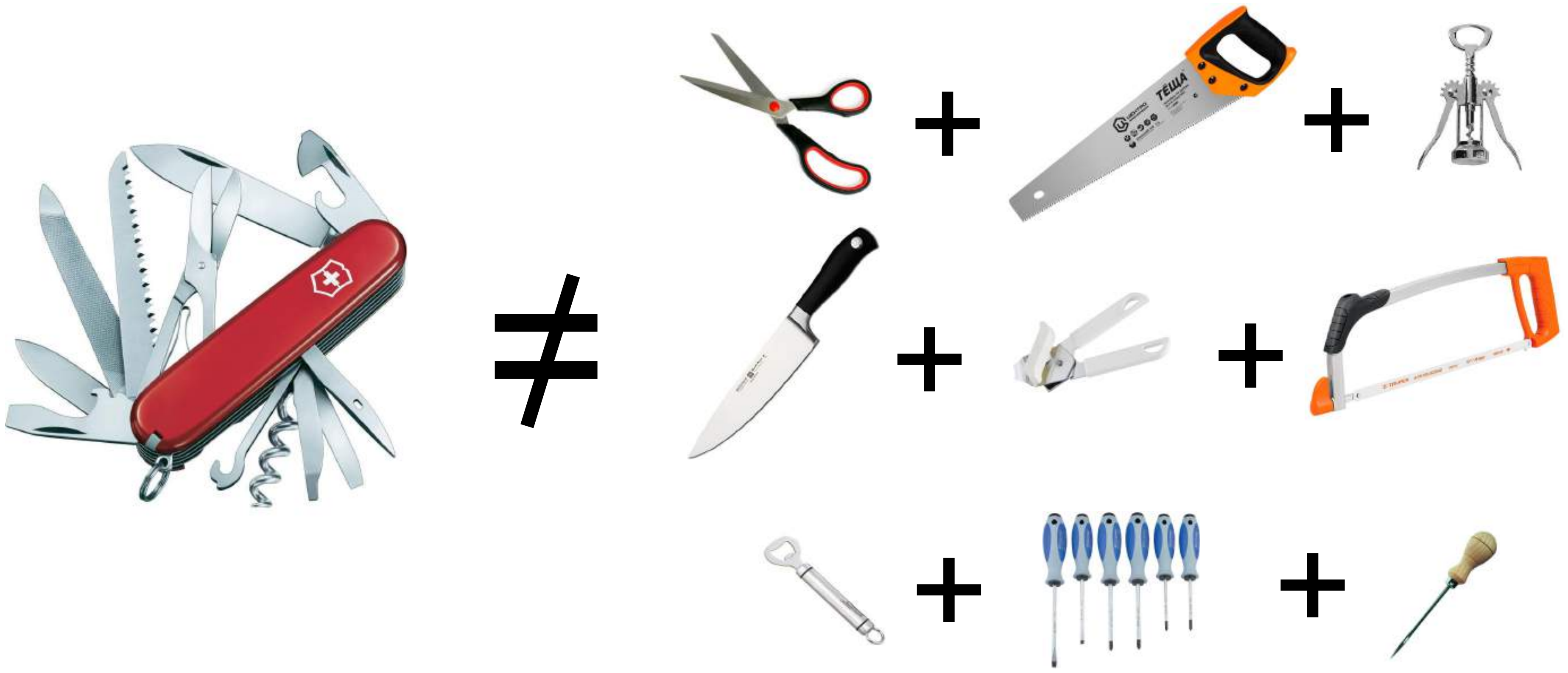
**Previous presentation**

On previous SECR 2016 we also presented our lecture about cross-platform solutions:
*“Current state and future of solutions for develop enterprise cross-platform mobile applications.”* You can download this presentation in Russian with this link:
http://files.tau-technologies.com/Events/2016_10_CEE_SECR/TAU_Technologies_CEE_SECR_2016_RUS.pdf

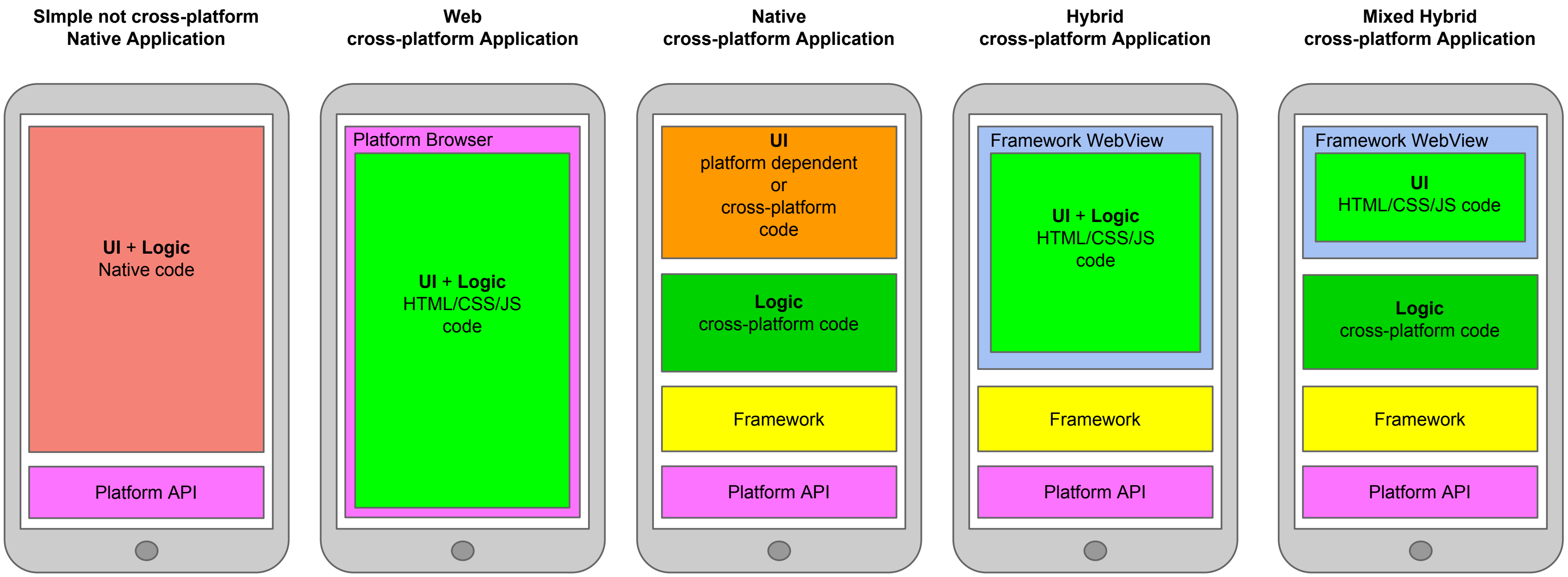
In previous lecture we spoke about existing solutions on market and compared them more detailed than this time - see previous presentation to know more.

## Contents

## 1. Cross-platform development is always a tradeoff!



# 1. Mobile cross-platform development architectures


SImple not cross-platform Native Application
UI + Logic Native code
Platform API
Web cross-platform Application
Platform Browser
UI + Logic HTML/CSS/JS code
Native cross-platform Application
UI platform dependent or cross-platform code
Logic cross-platform code
Framework
Platform API
Hybrid cross-platform Application
Framework WebView
UI + Logic HTML/CSS/JS code
Framework
Platform API
Mixed Hybrid cross-platform Application
Framework WebView
UI HTML/CSS/JS code
Logic cross-platform code
Framework
Platform API

iOS: ObjC, Swift, C++
Android: Java, C++
WinCE/WM: C#, C++

a lot of
HTML/CSS/JS frameworks

Xamarin (C#)
Appcelerator (JS)
React Native (JS)
NativeScript (JS)
QT (C++, QML)
RubyMotion (Ruby)
CodenameOne (Java)
Corona (Lua)

Cordova/PhoneGap RhoMobile
+ a lot of
HTML/CSS/JS frameworks

Cordova/PhoneGap with
jxCore plugin,
RhoMobile (Ruby/JS)
+ a lot of
HTML/CSS/JS frameworks

## Native

Qt

RubyMotion

kony

Xamarin

CODENAME ONE

Corona SDK

appcelerator® titanium

Telerik® Develop experiences

React Native

NativeScript

## Hybrid

APACHE CORDOVA™

PhoneGap

RHOMOBILE SUITE

## 2. Extended Browser - solution for lightweight web applications

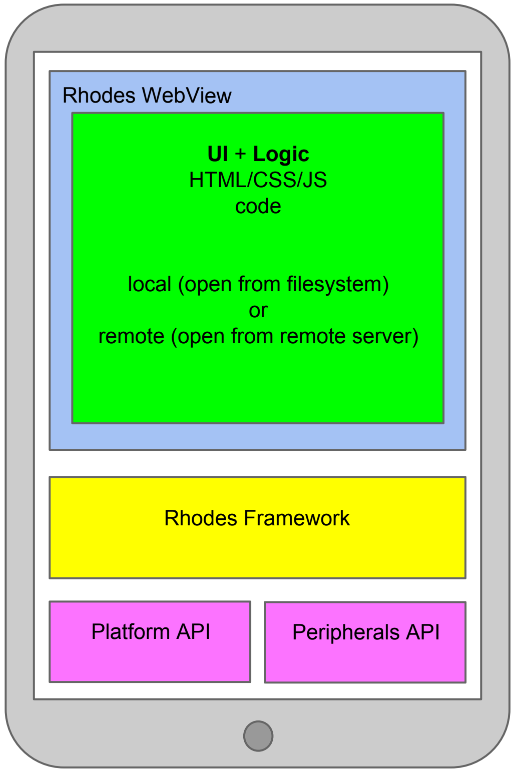
Sometimes we just need a lightweight web application with access to some H/W capabilities of device or some native API.

Adding a set of APIs into device browser we get a solution to execute our lightweight web applications (HTML/CSS/JS).

Some of device vendors already have this kind of solution. For instance, Zebra Technologies has **Enterprise Browser** (based on RhoMobile, not on Cordova, and supporting WM/WinCE):
https://www.zebra.com/us/en/products/software/mobile-computers/mobile-app-utilities/enterprise-browser.html

Honeywell also has similar product:
http://www.intermec.com/products/ib/index.aspx

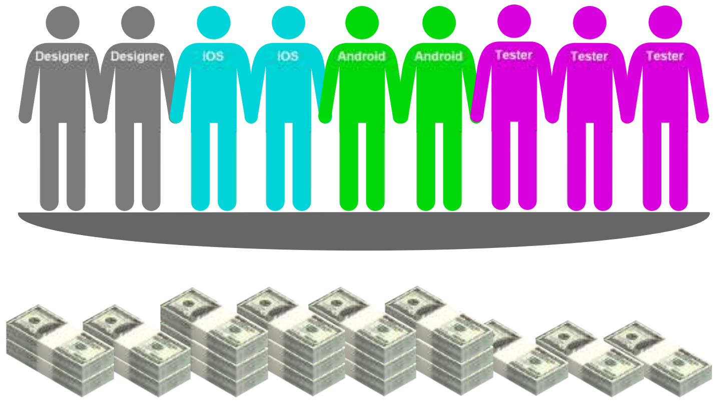
**RhoBrowser**

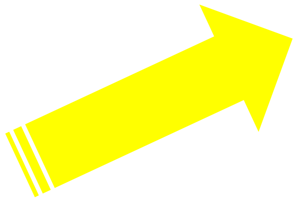


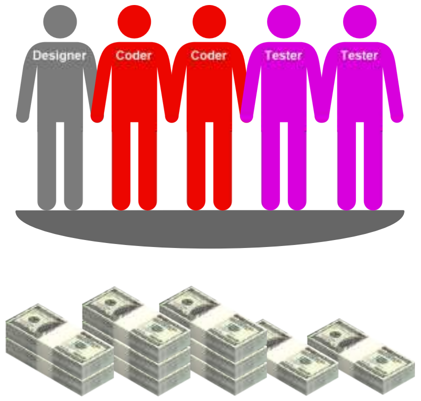
Tau Technologies has similar product - RhoBrowser, which runs on iOS, Android, WinCE/WM and supporting Zebra’s enterprise devices (WM and Android) H/W capabilities like Barcode scanner, RFID scanner etc.

## 3. Benefits of cross-platform development

- Cross-platform native solutions can decrease development resources up to two times !





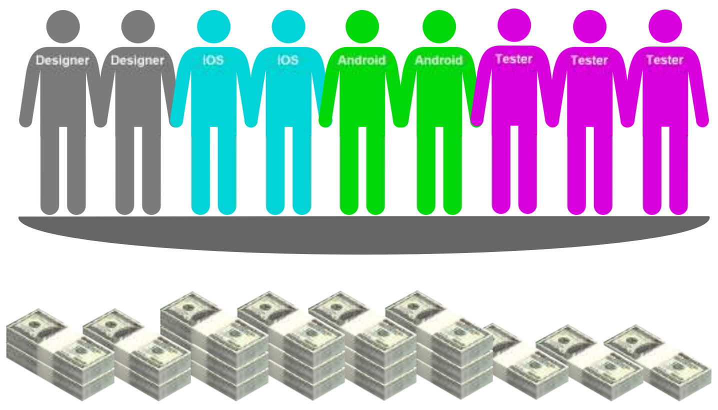


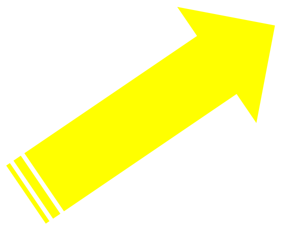
Problems of not-hybrid solutions :

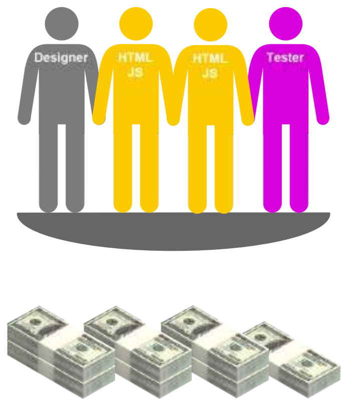
- Developers must learn specific API/Language
- Application’s code can not be used out of solution

## 3. Benefits of hybrid solutions

- Hybrid Cross-platform solutions (like Cordova or RhoMobile) can decrease development resources up to three times !







Important benefits :

- You can use existing web developers !
- You can use already developed code from web applications (HTML/CSS/JS) !
- You can transfer your code from web portal to application and from one hybrid solution to another without changes !

## 4. Problems of hybrid solutions

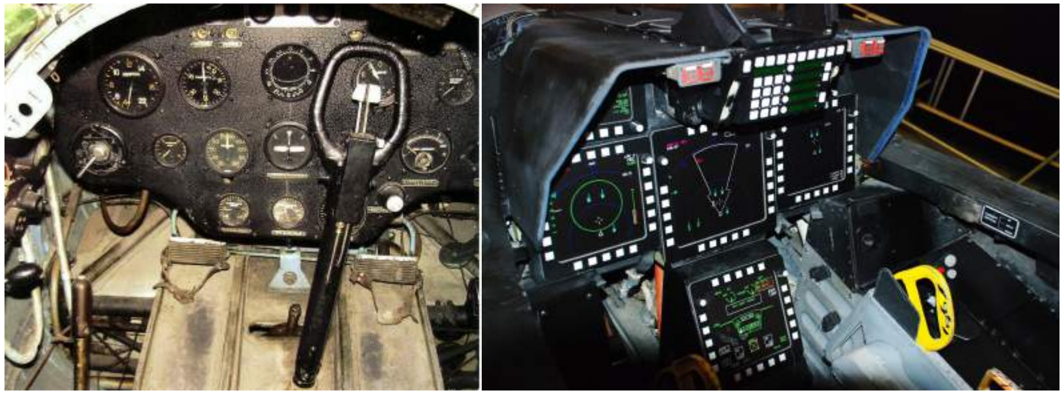
**Main problem is all code and data is inside WebView**

Big application like big web Application will have low performance will run in single thread etc.

**Web-based UI does not look and feel like native one**
Not a big deal for in-house enterprise apps;
Javascript UI frameworks can look almost like Native UI

**Different web browser versions**

Use own browser, Crosswalk for instance (https://crosswalk-project.org/)
RhoMobile has own WebKit port for WinCE/WM.
JS Frameworks also help.



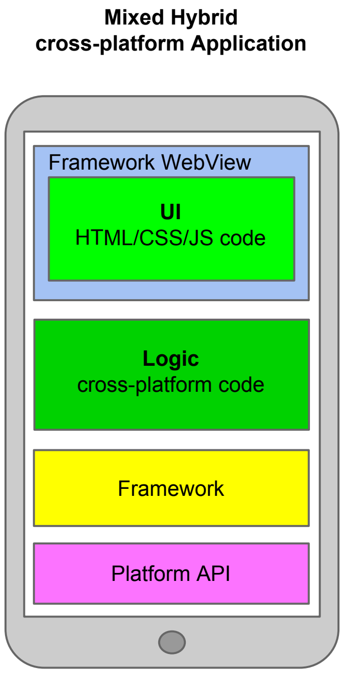
## 5. Mixed-hybrid architecture

How we can solve main problem of hybrid solutions ?

Extract not-UI code from WebView into separate threads and container !

Mixed-Hybrid solution do it !

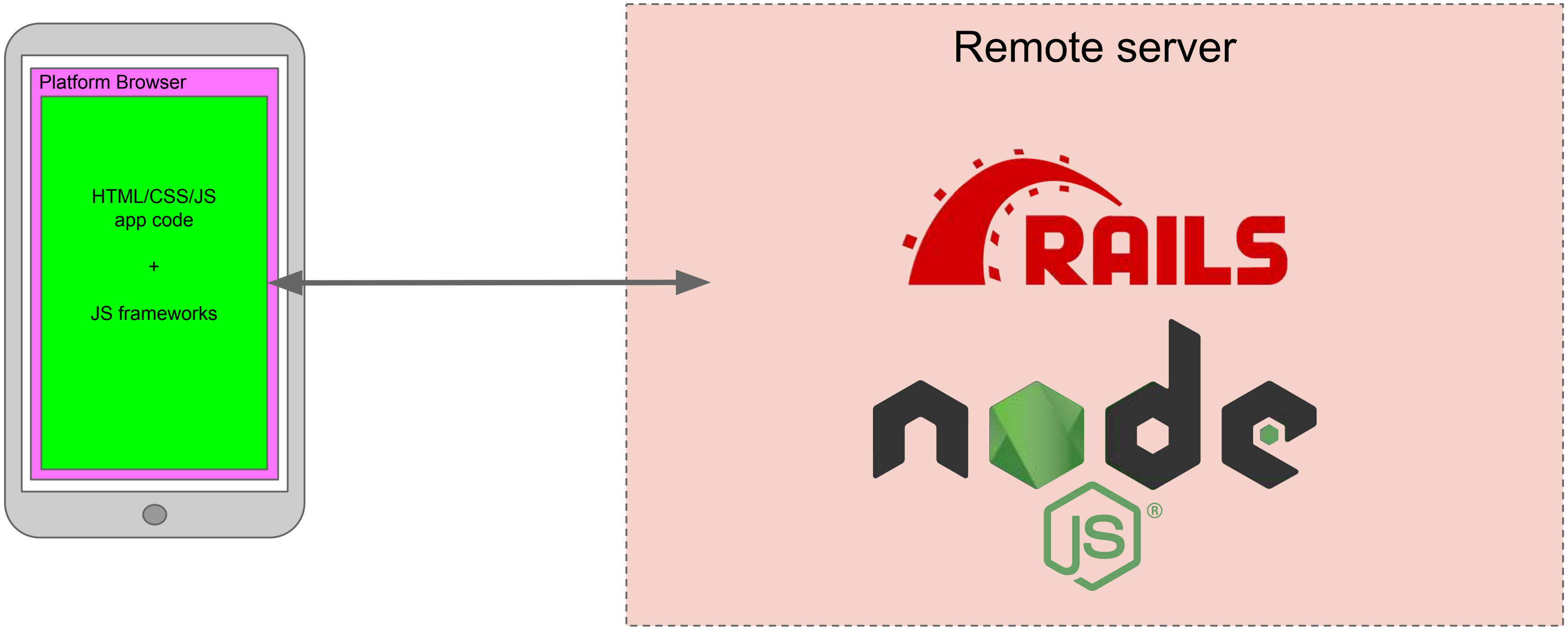
But how make it ?
What language should we use ?
How UI in WebView should be connected to logic and data ?



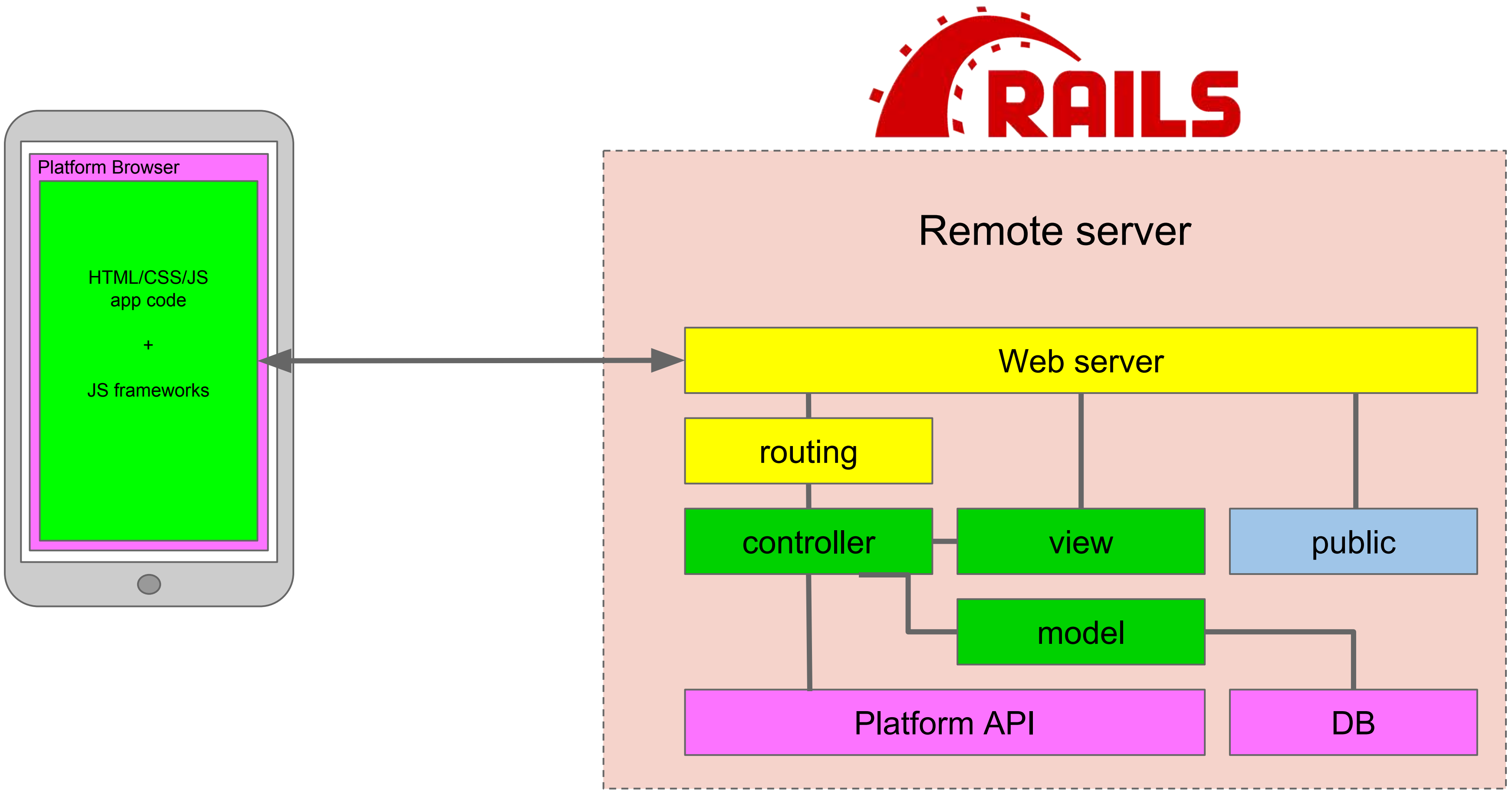
## 6. Client-Server web application

Wait a minute.
We have a web browser with HTML/CSS/JS code and separate logic/DB ?
But this is well known client-server architecture for web applications !

We have a lot of already existing code for this platforms.
We have a lot of experienced developers for this platforms.
We already used this in our web applications etc.



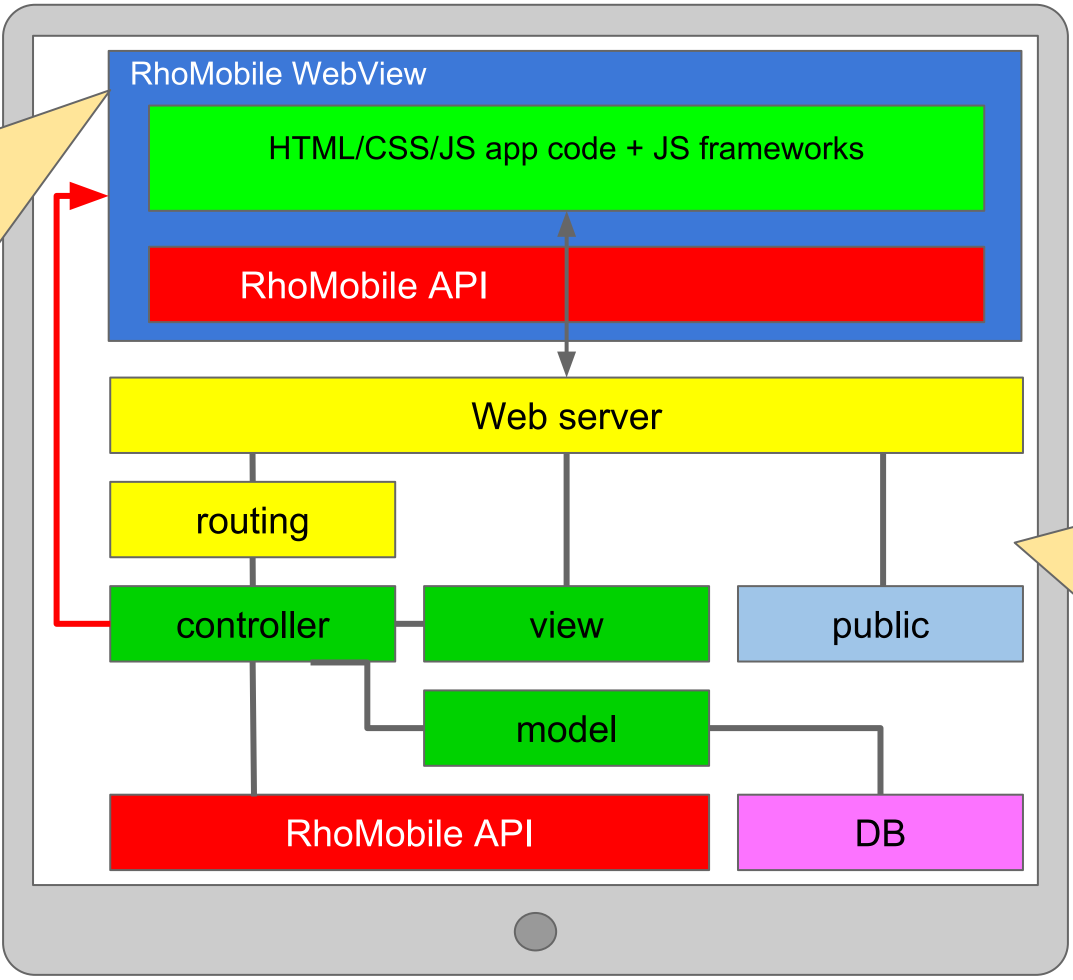
## 7. Ruby on Rails from Client-Server with remote server



## 7. Ruby on Rails to Client-Server with local server on device !

Hybrid architecture.
UI implemented with web technologies inside WebView.

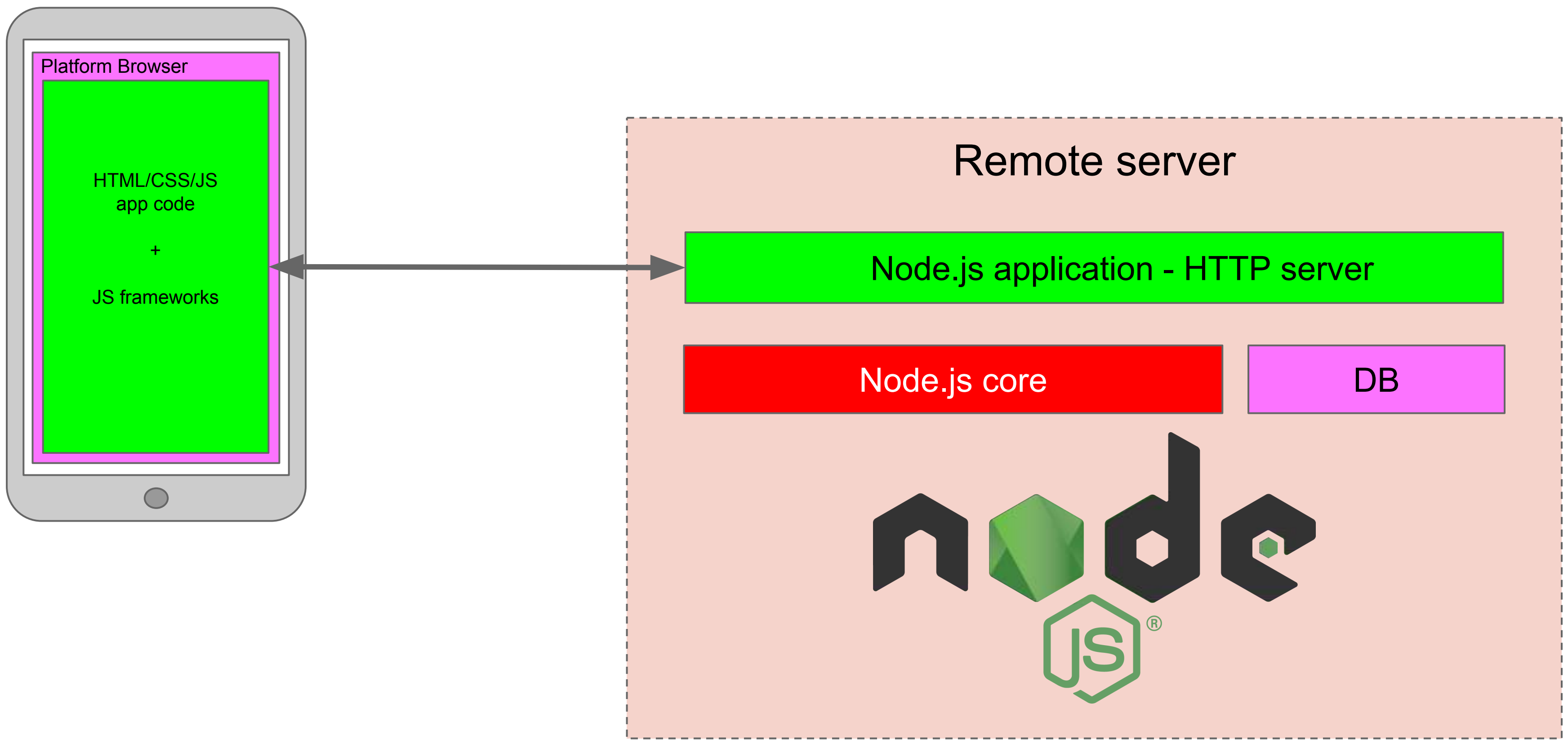
Developers can use any JS frameworks.



RHODES

We have local HTTP server with Ruby VM on our mobile device. Ruby code executed on mobile device.

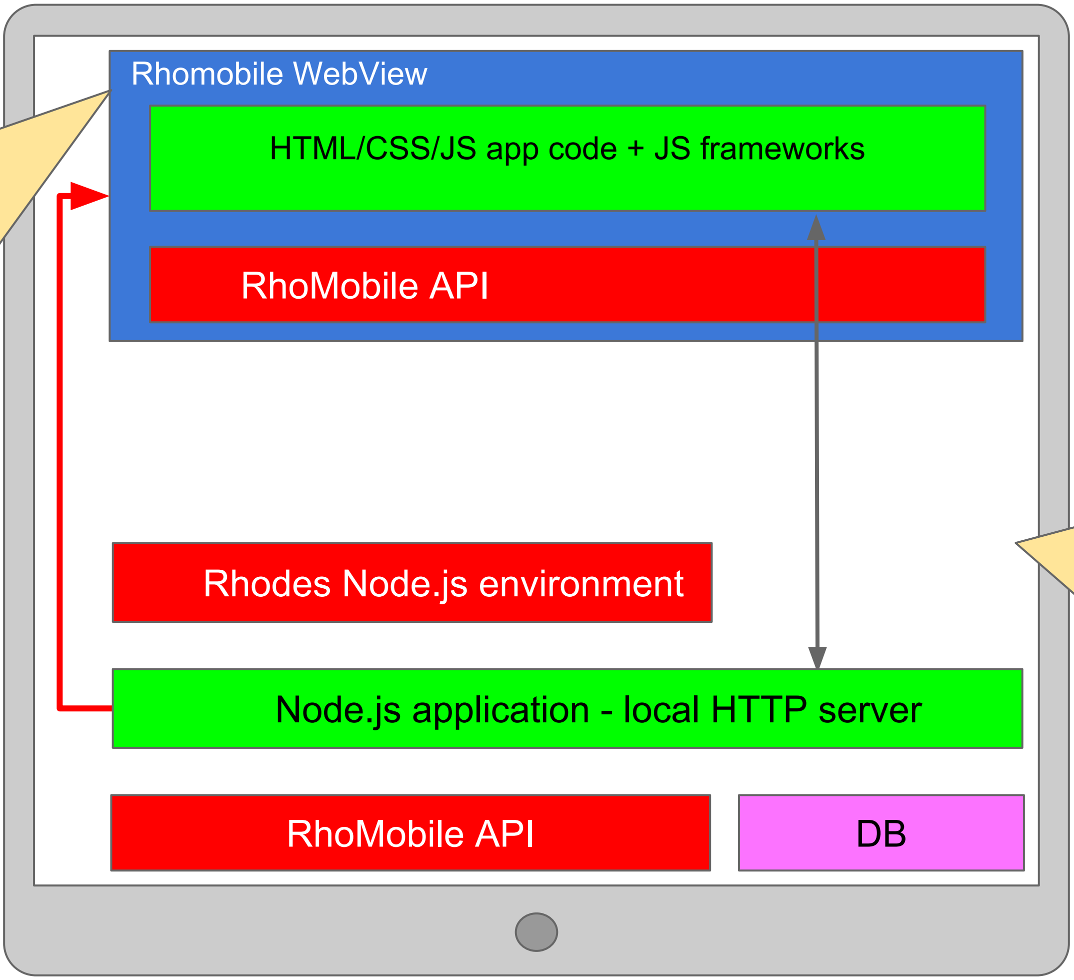
## 8. Node.js from Client-Server with remote server



## 8. Node.js to Client-Server with local server on device

Hybrid architecture.
UI implemented with web technologies inside WebView.

Developers can use any JS frameworks.



Rhomobile WebView
HTML/CSS/JS app code + JS frameworks
RhoMobile API
Rhodes Node.js environment
Node.js application - local HTTP server
RhoMobile API
DB

RHODES

We have local JS VM and Node.js environment.
Node.js application starts local HTTP server.

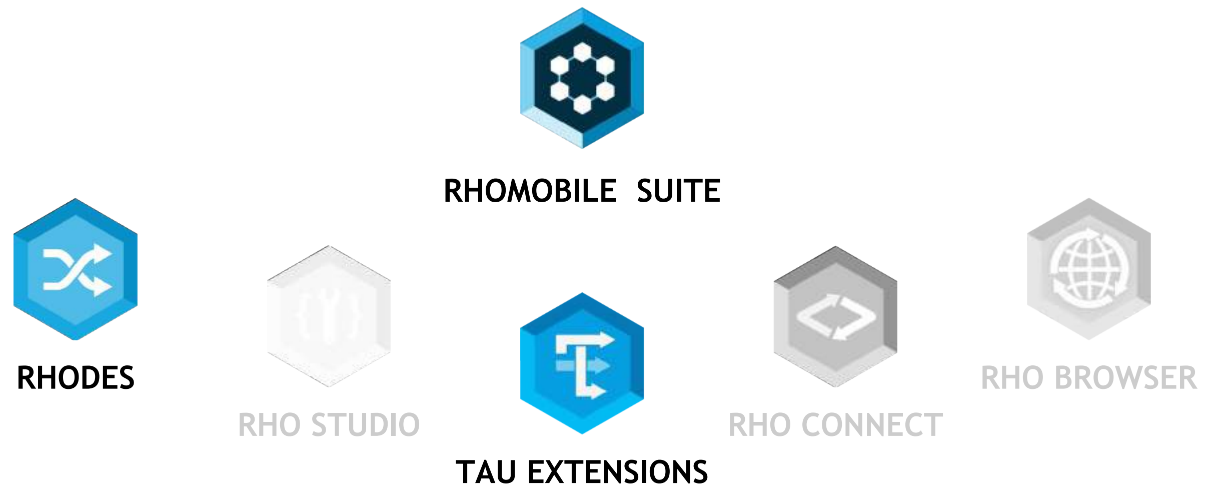
## 9. Node.js based desktop solutions

Hybrid solutions to make desktop applications based on Node.js

| | |
|---|---|
| name | **NW.js** |
| developer | **NW.js community** |
| type | hybrid |
| source code | full open source |
| price | FREE |
| website | [http://nwjs.io/](http://nwjs.io/) |
| Supported platforms | Linux, Mac OS X, Windows |

| | |
|---|---|
| name | **Electron** |
| developer | **Electron community** |
| type | hybrid |
| source code | full open source |
| price | FREE |
| website | [http://electron.atom.io/](http://electron.atom.io/) |
| Supported platforms | Linux, Mac OS X, Windows |

## 10. RhoMobile Solution



**Rhodes**

- **Solution for development of mobile cross-platform hybrid and mixed-hybrid applications**
- Developers can use just **HTML/CSS/JS** (like Cordova), and also use **Ruby** in Ruby on Rails like environment
- Includes a lot of modules with support for different APIs like Barcode, Printing etc.
- Support **iOS**, **Android**, **WinCE/WM**, **WP**
- **Own port of Ruby 2.3.3 VM**

**Tau Extensions**

Additional modules include:

- Crosswalk WebView for Android (replaces system WebView in application)
- Own port of WebKit for WinCE/WM (replaces system IE in application)
- improved OpenSSL for Android
- **module with Node.js based on jxCore**

## 10. Installation of RhoMobile

Before installation please install all prerequisites into your system. Details:
http://docs.tau-technologies.com/en/6.0/guide/rhomobile-install
http://docs.tau-technologies.com/en/6.0/guide/nativesdksetup

There are three ways to install RhoMobile - please install our latest release 6.0 :

- Download and install our all-in-one installation package.
  http://tau-technologies.com/developers/downloads/

RECOMMENDED

- Install gems manually

```
$ gem install rhodes
$ gem install rho-tau-extensions
$ gem install rhoconnect
$ gem install rhoconnect-client
```

- Download source code from GitHub(you should manually define path to rhodes in applications)
  Source code : https://github.com/rhomobile/rhodes

After install you should set up paths to mobile SDKs:

```
$ rhodes-setup
```

## 11. How it works. Mobile cross-platform application with Ruby

Let’s make our application:
(see details: http://docs.tau-technologies.com/en/6.0/guide/creating_a_project) :

```
$ rhodes app MyApp
```

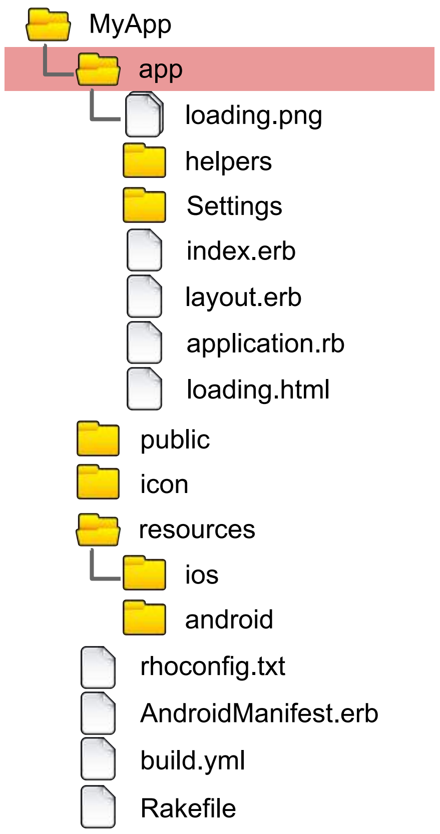
**rhodes** - command line tool for generating :
applications, models, extension.
Generated code is fully workable and can be built and run.

## 11. How it works. Mobile cross-platform application with Ruby

```
MyApp
└ app
    └ loading.png
      helpers
      Settings
      index.erb
      layout.erb
      application.rb
      loading.html
  public
  icon
  resources
    └ ios
      android
  rhoconfig.txt
  AndroidManifest.erb
  build.yml
  Rakefile
```

We get MyApp folder where located generated application’s code, resources etc.

## 11. How it works. Mobile cross-platform application with Ruby



**“app”** folder contains application’s code - ***.ruby** and ***.erb** (templates) files

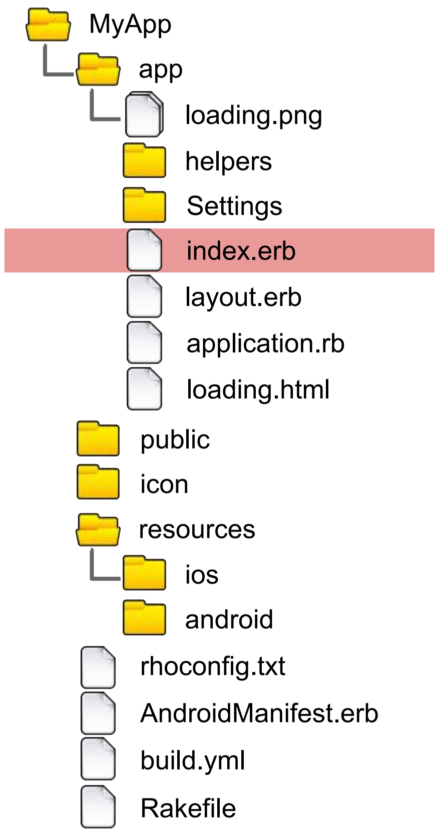
In runtime this folder is located under root of local HTTP server.

## 11. How it works. Mobile cross-platform application with Ruby

```
MyApp
└ app
   └ loading.png
     helpers
     Settings
     index.erb
     layout.erb
     application.rb
     loading.html
  public
  icon
  resources
   └ ios
     android
  rhoconfig.txt
  AndroidManifest.erb
  build.yml
  Rakefile
```

Folders with helper ruby files

# 11. How it works. Mobile cross-platform application with Ruby



```
MyApp
└ app
   └ loading.png
     helpers
     Settings
     index.erb
     layout.erb
     application.rb
     loading.html
  public
  icon
  resources
   └ ios
     android
  rhoconfig.txt
  AndroidManifest.erb
  build.yml
  Rakefile
```

**erb** template for index page (start page in application by default)
Used two steps HTML generation :
Page template contains only **content**, and shared other parts and loading of **CSS**,**JS** in separate **erb** template **layout**

```
<div class="container-fluid">



  <div class="row">
    <div class="list-group">
      <a href="#" class="list-group-item">
        <span class="glyphicon glyphicon-chevron-right pull-right"
aria-hidden="true"> </span>
        Add link here...
      </a>
    </div>
  </div>

</div>
```

## 11. How it works. Mobile cross-platform application with Ruby

**erb** template for all pages (can be overriden in each controller)

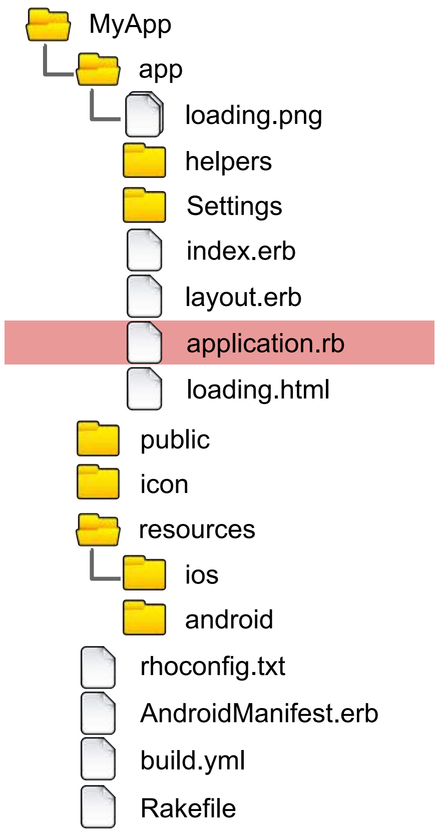
```
MyApp
└ app
   └ loading.png
     helpers
     Settings
     index.erb
     layout.erb
     application.rb
     loading.html
  public
  icon
  resources
   └ ios
     android
  rhoconfig.txt
  AndroidManifest.erb
  build.yml
  Rakefile
```

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
        "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">

<head>
  <title>MyApp</title>
  <meta name="viewport" content="width=device-width, initial-scale=1.0,
maximum-scale=1.0, user-scalable=0"/>

loading of CSS and JS ...

</head>

<body data-platform="<%= Rho::System.getProperty('platform') %>">
<%= @content %>
</body>

</html>
```

framework places generated page content here

## 11. How it works. Mobile cross-platform application with Ruby

application class code - activation, deactivation etc.



```
require 'rho/rhoapplication'

class AppApplication < Rho::RhoApplication

  def initialize
    # Tab items are loaded left->right, @tabs[0] is leftmost tab in the tab-bar
    # Super must be called *after* settings @tabs!
    @tabs = nil
    #To remove default toolbar uncomment next line:
    #@@toolbar = nil
    super

  end


end
```

# 11. How it works. Mobile cross-platform application with Ruby

folder with static files of local HTTP server: CSS, JS, images etc.

```
MyApp
└ app
   └ loading.png
     helpers
     Settings
     index.erb
     layout.erb
     application.rb
     loading.html
  public
  icon
  resources
   └ ios
     android
  rhoconfig.txt
  AndroidManifest.erb
  build.yml
  Rakefile
```

```
public
└ css
  images
  jqmobile
  jquery
  js
```

## 11. How it works. Mobile cross-platform application with Ruby

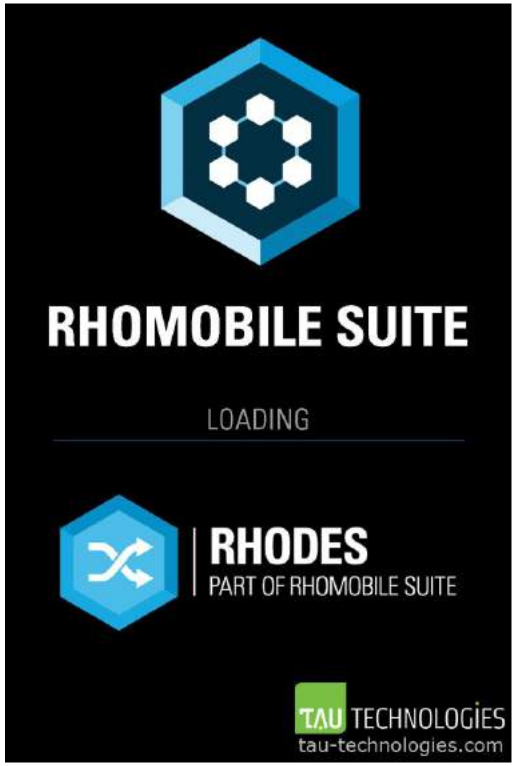
```
MyApp
└ app
   └ loading.png
     helpers
     Settings
     index.erb
     layout.erb
     application.rb
     loading.html
  public
  icon
  resources
   └ ios
     android
  rhoconfig.txt
  AndroidManifest.erb
  build.yml
  Rakefile
```

folder with resources used for application build: icon, splash image, iTunes image etc.

See details in documentation:

[http://docs.tau-technologies.com/en/6.0/guide/app_icon_splash](http://docs.tau-technologies.com/en/6.0/guide/app_icon_splash)

[http://docs.tau-technologies.com/en/6.0/guide/build_ios](http://docs.tau-technologies.com/en/6.0/guide/build_ios)



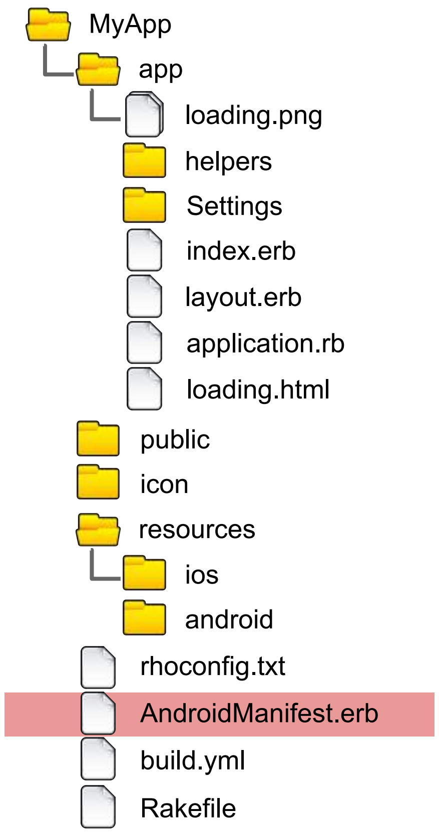
## 11. How it works. Mobile cross-platform application with Ruby

application’s configuration - used in run-time

```
MyApp
└ app
  └ loading.png
    helpers
    Settings
    index.erb
    layout.erb
    application.rb
    loading.html
  public
  icon
  resources
  └ ios
    android
  rhoconfig.txt
  AndroidManifest.erb
  build.yml
  Rakefile
```

```
# startup page for your application
start_path = '/app'

options_path = '/app/Settings'

# Rhodes log properties
MinSeverity  = 1
LogToOutput = 1
MaxLogFileSize=50000
logserver = 'http://rhologs.heroku.com'
logname='MyApp'

syncserver = ''
sync_poll_interval=0

...
```

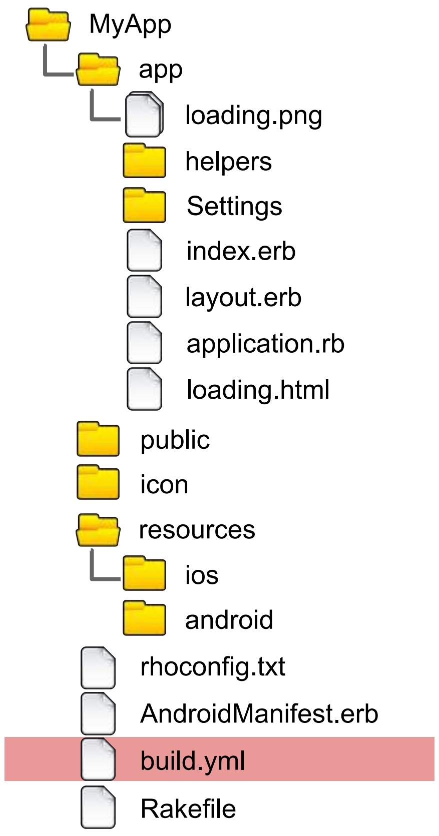
## 11. How it works. Mobile cross-platform application with Ruby



template for application’s Android manifest.
See details:
[http://docs.tau-technologies.com/en/6.0/guide/build_android](http://docs.tau-technologies.com/en/6.0/guide/build_android)

## 11. How it works. Mobile cross-platform application with Ruby

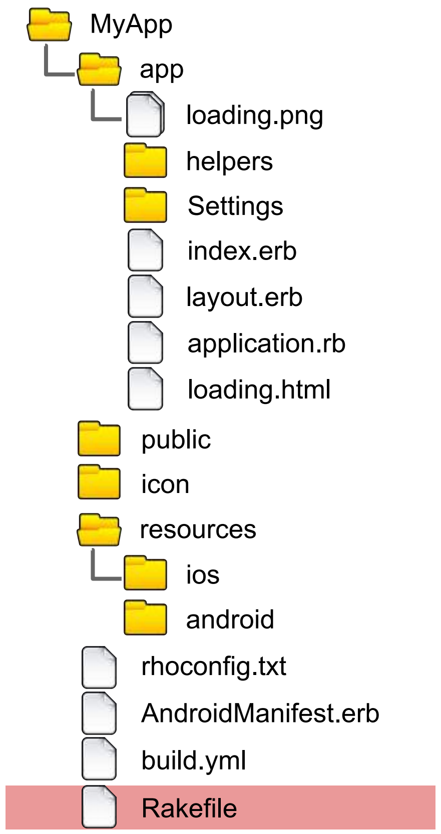
Build configuration - settings used for application building, enable/disable capabilities, link extensions etc.



```
name: MyApp
version: 1.0
vendor: rhomobile
build: debug
applog: rholog.txt

capabilities:
  - camera

iphone:
  configuration: Release
  sdk: latest
  BundleIdentifier: com.rhomobile.myapp
  BundleURLScheme: myapp

android:
  version: 4.1.0
  logcatFilter: APP:I StrictMode:I DEBUG:I *:E

extensions: []
```

## 11. How it works. Mobile cross-platform application with Ruby

Standard Ruby script for rake commands (build, run, etc.)



```
MyApp
└ app
  └ loading.png
    helpers
    Settings
    index.erb
    layout.erb
    application.rb
    loading.html
  public
  icon
  resources
  └ ios
    android
  rhoconfig.txt
  AndroidManifest.erb
  build.yml
  Rakefile
```

## 11. How it works. Mobile cross-platform application with Ruby

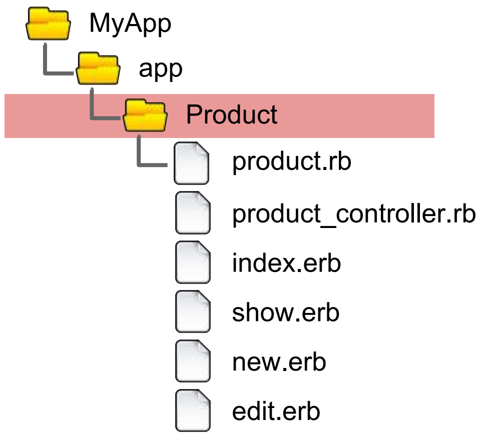
Let’s add simple DB model to our application - Rhodes generator makes model Ruby file and set of views (erb files) for view, edit, delete etc.
(details: http://docs.tau-technologies.com/en/6.0/guide/rhom_ruby)

Run Rhodes generator from application’s folder :

```
$ rhodes model Product name,brand,price
```

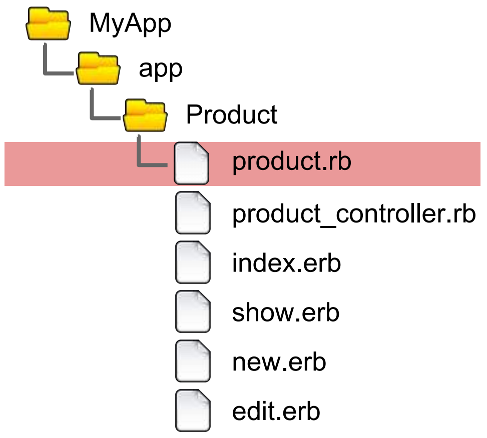
## 11. How it works. Mobile cross-platform application with Ruby

You can see some new content in **app** folder: **Product** folder with set of files



## 11. How it works. Mobile cross-platform application with Ruby

This file contains model definition code. PropertyBag scheme is used by default.



```
# The model has already been created by the framework, and extends Rhom::RhomObject
# You can add more methods here
class Product
  include Rhom::PropertyBag

  # Uncomment the following line to enable sync with Product.
  # enable :sync

  #add model specific code here
end
```

# 11. How it works. Mobile cross-platform application with Ruby

```
MyApp
└ app
  └ Product
    └ product.rb
      product_controller.rb
      index.erb
      show.erb
      new.erb
      edit.erb
```

product_controller.rb - Ruby code for controller class.
Contoller’s methods are called from WebView by URL like /app/Controller/**method** and return HTML page generated from erb template named **method**.erb by default ( can be overridden by controller’s code ).
If method is not exists then simple content generation used by erb template.

```ruby
require 'rho/rhocontroller'
require 'helpers/browser_helper'

class ProductController < Rho::RhoController
  include BrowserHelper

  def index
    @products = Product.find(:all)
    render :back => '/app'
  end

  def edit
    @product = Product.find(@params['id'])
    if @product
      render :action => :edit, :back => url_for(:action => :index)
    else
      redirect :action => :index
    end
   end

 ...
```

Find all object from **Product** model

**@params** - hash with all parameters of request (query part of URL)

# 11. How it works. Mobile cross-platform application with Ruby

```
MyApp
└ app
  └ Product
    └ product.rb
      product_controller.rb
      index.erb
      show.erb
      new.erb
      edit.erb
```

set of **views** - **erb** templates to generate main model access pages.
**index.erb** is default page. Controller's **index** functions also executed by default, when local server answer request like **/app/Controller/**

```
<div class="container-fluid">

  <div class="row page-title">
…
  </div>

  <div class="row">
    <div class="list-group">
      <% @products.each do |product| %>

        <a href="<%= url_for :action => :show, :id => product.object %>" class="list-group-item">
          <span class="glyphicon glyphicon-chevron-right pull-right" aria-hidden="true"></span>
          <%= product.name %>
        </a>

      <% end %>
    </div>
  </div>

</div>
```

enumerate all objects from model and make HTML list with links to view each object

## 11. How it works. Mobile cross-platform application with Ruby

FInal change - modify our application start URL in **rhoconfig.txt** to Product page:

```
# startup page for your application
start_path = '/app/Product'
```

# 11. How it works. Mobile cross-platform application with Ruby

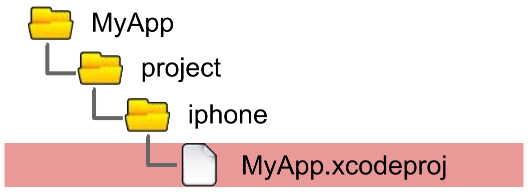
Let’s run our application on iPhone Simulator:

```
$ rake run:iphone
```

Also you can generate XCode project and use XCode for build/run etc.
Generate XCode project :

```
$ rake rake build:iphone:setup_xcode_project
```

Generated XCode project located in:



Details: [http://docs.tau-technologies.com/en/6.0/guide/build_ios](http://docs.tau-technologies.com/en/6.0/guide/build_ios)

## 11. How it works. Mobile cross-platform application with Ruby

Run application on Android Emulator:

```
$ rake run:android
```

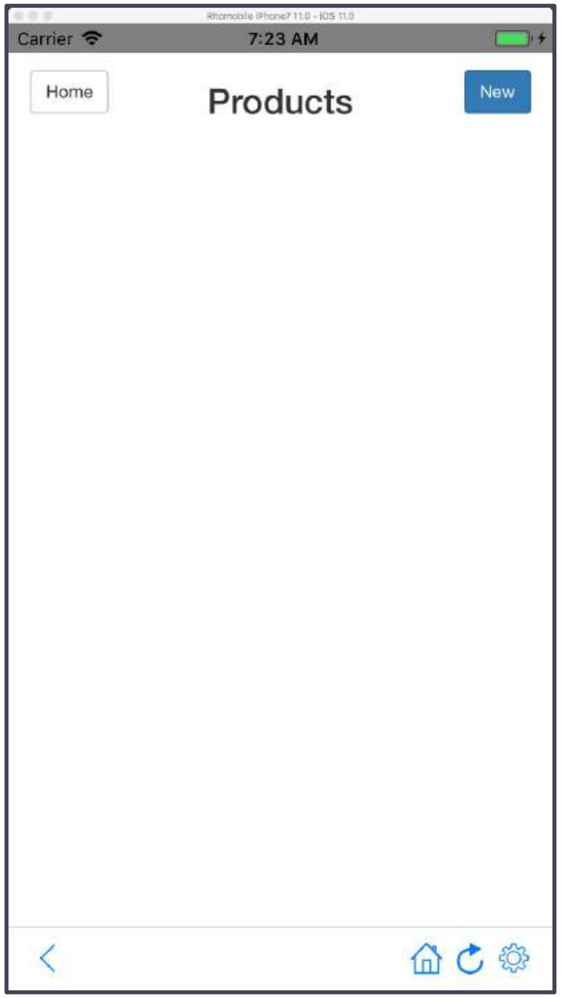
Build and run application on USB-linked Android device :

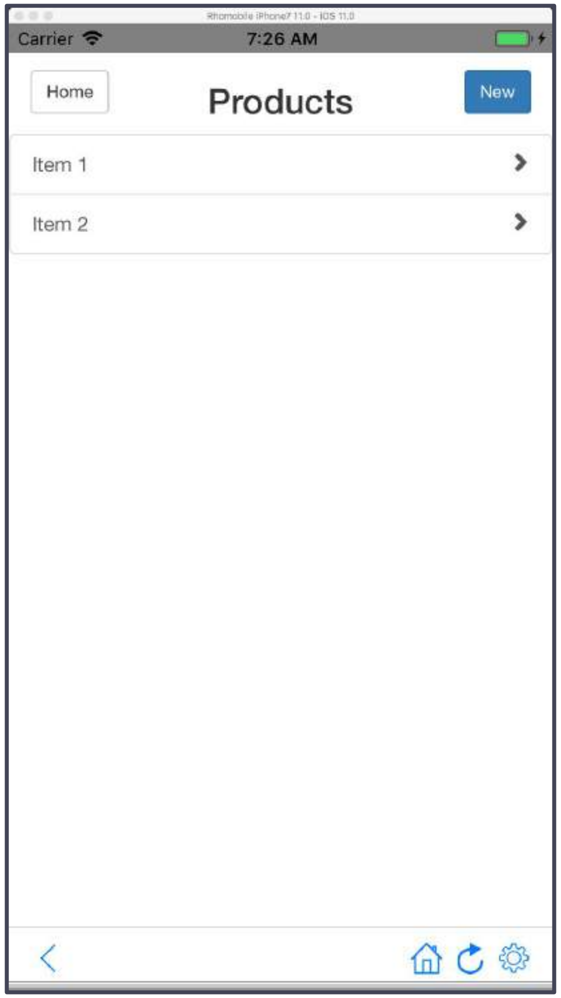
```
$ rake run:android:device
```

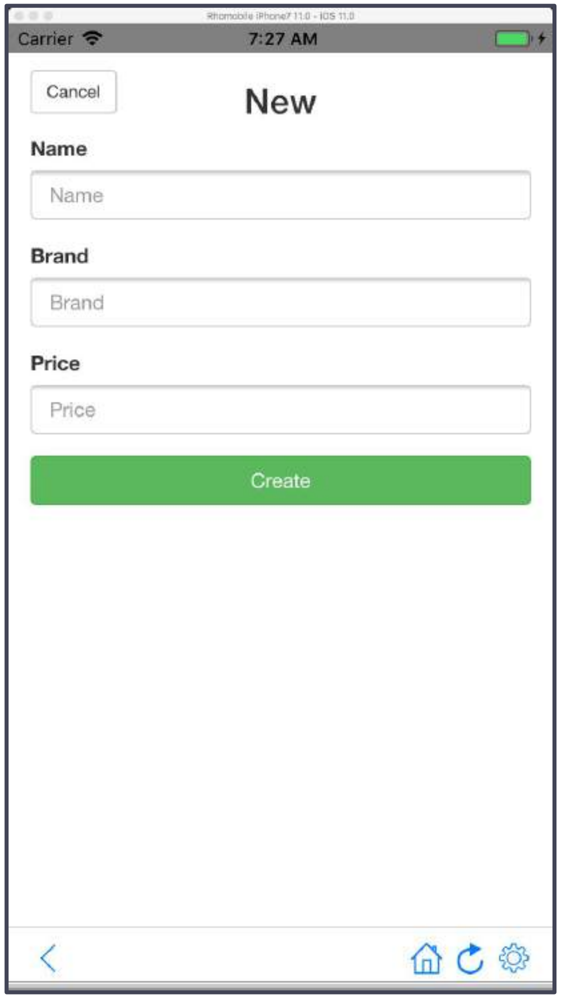
Details: http://docs.tau-technologies.com/en/6.0/guide/build_android

## 11. How it works. Mobile cross-platform application with Ruby

Application's screenshots:







## 12. How it works. Mobile cross-platform application with Node.js

Rhodes use jxCore Node.js port for iOS and Android platforms.
JXCore - already closed project. Currently still alive fork - Thali Project (sponsored by Microsoft)
JXCore FAQ: http://www.goland.org/jxcore/
Also jxCore has Cordova plugin: https://github.com/jxcore/jxcore-cordova

| | |
|---|---|
| название | **JXCore** |
| разработчик | **Nubisa** (прекращено) |
| тип | Node.js порт |
| исходный код | full open source |
| доступность | FREE |
| сайт | https://github.com/jxcore/jxcore |
| Платформа | iOS |

| | |
|---|---|
| название | **Thali** |
| разработчик | **Thali** (спонсирует Microsoft) |
| тип | гибридный (Cordova +JXCore) |
| исходный код | full open source |
| доступность | FREE |
| сайт | http://thaliproject.org/ |
| Платформа | iOS |

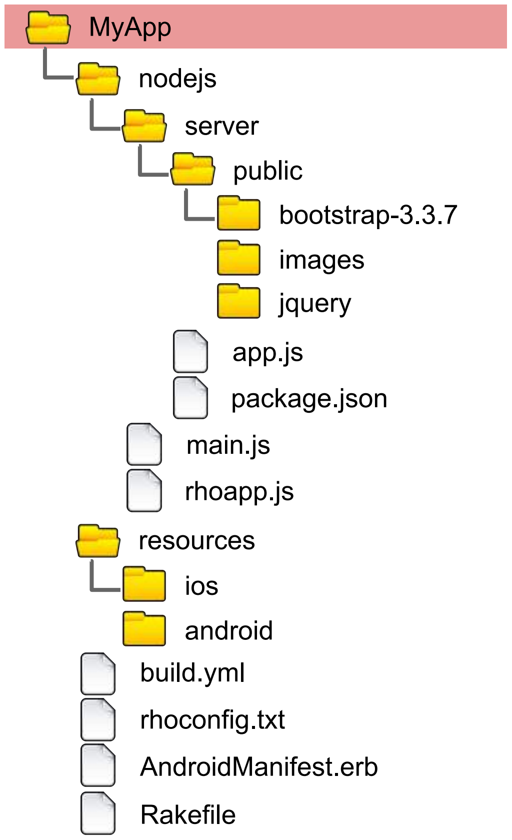
## 12. How it works. Mobile cross-platform application with Node.js

Let’s make our application with Node.js:
(see details: http://docs.tau-technologies.com/en/6.0/guide/creating_a_project) :

```
$ rhodes nodejsapp MyApp
```

**rhodes** - command line tool for generating :
applications, models, extension.
Generated code is fully workable and can be built and run.

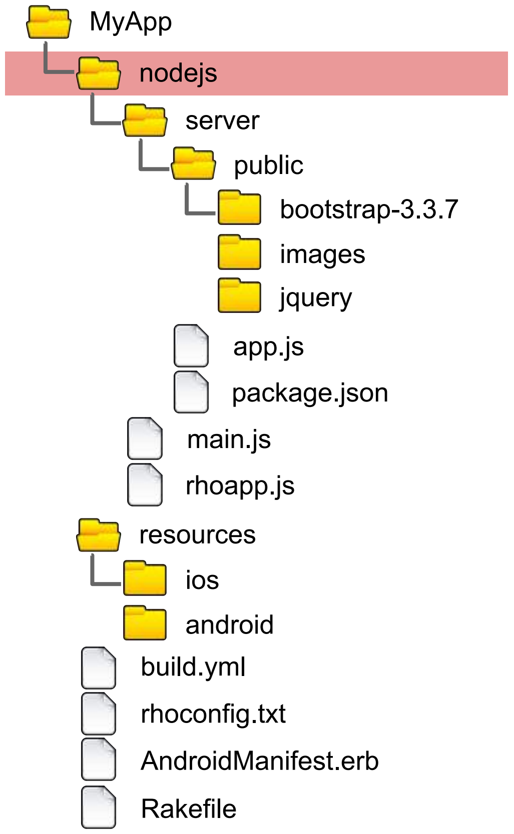
# 12. How it works. Mobile cross-platform application with Node.js



```
MyApp
└ nodejs
  └ server
    └ public
      └ bootstrap-3.3.7
        images
        jquery
      app.js
      package.json
    main.js
    rhoapp.js
  resources
  └ ios
    android
  build.yml
  rhoconfig.txt
  AndroidManifest.erb
  Rakefile
```

We get MyApp folder with application’s code, resources etc.

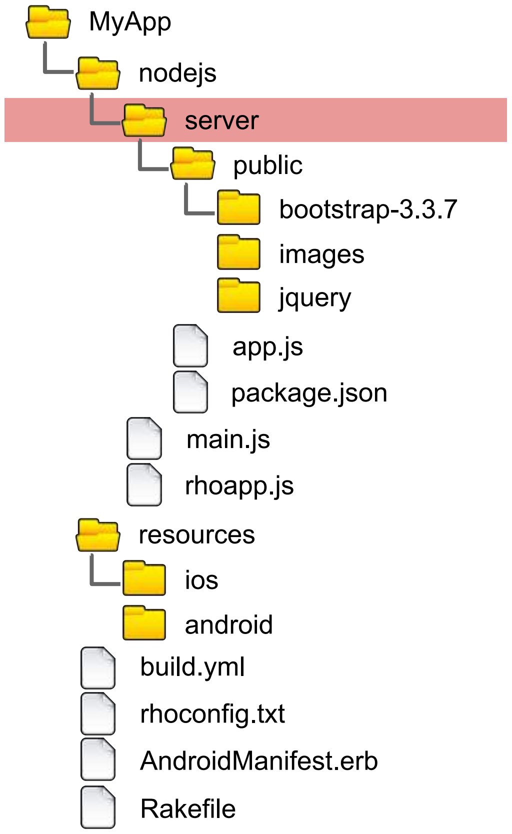
## 12. How it works. Mobile cross-platform application with Node.js



```
MyApp
    nodejs
        server
            public
                bootstrap-3.3.7
                images
                jquery
            app.js
            package.json
        main.js
        rhoapp.js
    resources
        ios
        android
    build.yml
    rhoconfig.txt
    AndroidManifest.erb
    Rakefile
```

This folder contains Node.js application main files and additional files

## 12. How it works. Mobile cross-platform application with Node.js



This folder contains Node.js application files
This folder is also root folder of local Node.js HTTP server

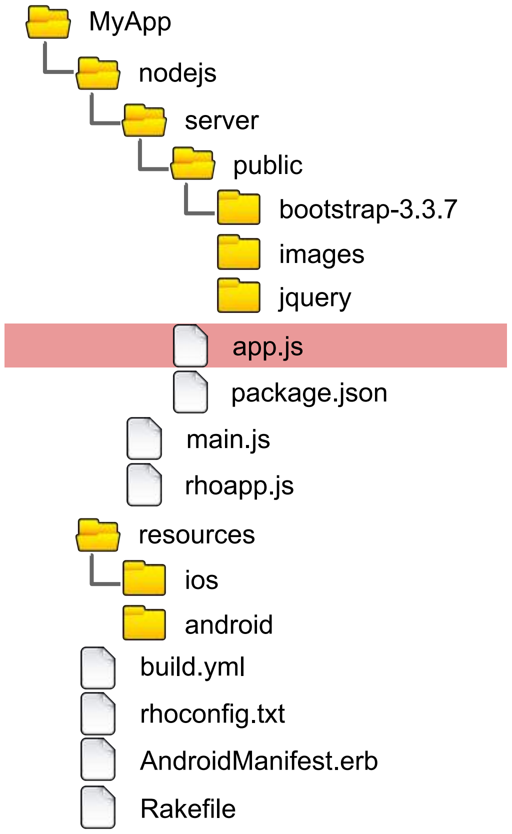
## 12. How it works. Mobile cross-platform application with Node.js

```
MyApp
└ nodejs
   └ server
      └ public
         └ bootstrap-3.3.7
           images
           jquery
        app.js
        package.json
     main.js
     rhoapp.js
  resources
   └ ios
     android
  build.yml
  rhoconfig.txt
  AndroidManifest.erb
  Rakefile
```

This folder contains static files of local Node.js HTTP server - CSS, JS, images etc.

At build time an 'api' folder will be added here with Rhodes API JS files.

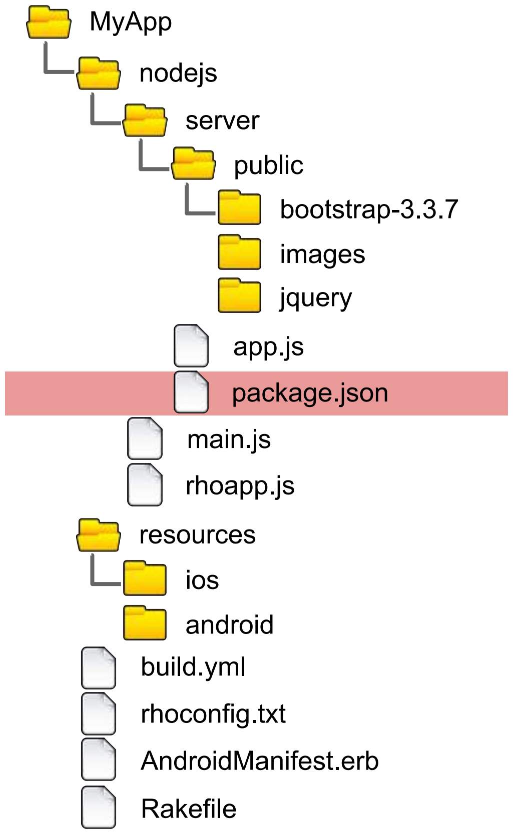
# 12. How it works. Mobile cross-platform application with Node.js



**app.js** main file of Node.js application - this file will be executed on start. Developer should start local HTTP server with predefined port here and call notify of Rhodes API.

```
var server_port = Rho.System.NodejsServerPort

var path = require('path');
var express = require('express');
var app = express();

app.use('/public', express.static(path.join(__dirname, 'public')));

app.get('/', function (req, res) {
  res.send('Hello World! (' + Date.now() + ")");
});

var server = app.listen(server_port, function () {
  Rho.Log.info("Express server is started. (port: "+server_port+")", "Node.js JS");
  // application must be inform RHomobile platform about starting of http server !
  Mobile.httpServerStarted();
});
```

# 12. How it works. Mobile cross-platform application with Node.js



**package.json** - file with Node.js application’s main properties.
Same with usual Node.js application.
Contains all used modules.

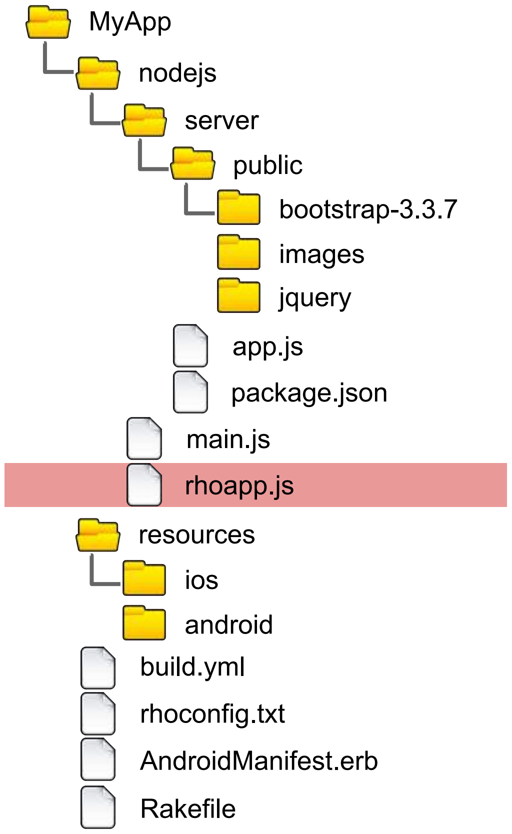
```
{
  "name": "rhonodejsapplication",
  "version": "0.0.1",
  "private": true,
  "dependencies": {
    "express": "*"
  }
}
```

## 12. How it works. Mobile cross-platform application with Node.js

```
MyApp
└ nodejs
    └ server
        └ public
            └ bootstrap-3.3.7
              images
              jquery
          app.js
          package.json
      main.js
      rhoapp.js
  resources
    └ ios
      android
  build.yml
  rhoconfig.txt
  AndroidManifest.erb
  Rakefile
```

**main.js** - core init code, also includes init of Rhodes API.
Developer must not change this file!

# 12. How it works. Mobile cross-platform application with Node.js



**rhoapp.js** - this file is executed during application initialisation. By default contains code for setup of main application's event processing - activate, deactivate etc. Developer can change this file.
Below you can see part of code where we make Native Toolbar in our application when application is activated (it is native platform Toolbar)

```
…

function onRhomobileApplicationActivated() {
   Rho.Log.info("Node.js event : APP_EVENT_ACTIVATED", "Node.js JS App");

   var native_toolbar = [
      {"action": "back", "icon": "/nodejs/server/public/images/bar/back_btn.png"},
      {"action": "separator"},
      {"action": "home", "icon": "/nodejs/server/images/bar/home_btn.png"},
      {"action": "refresh"},
      {"action": "options", "icon": "/nodejs/server/images/bar/gears.png"}
   ];
   Rho.NativeToolbar.create(native_toolbar);
}

...
```

# 12. How it works. Mobile cross-platform application with Node.js

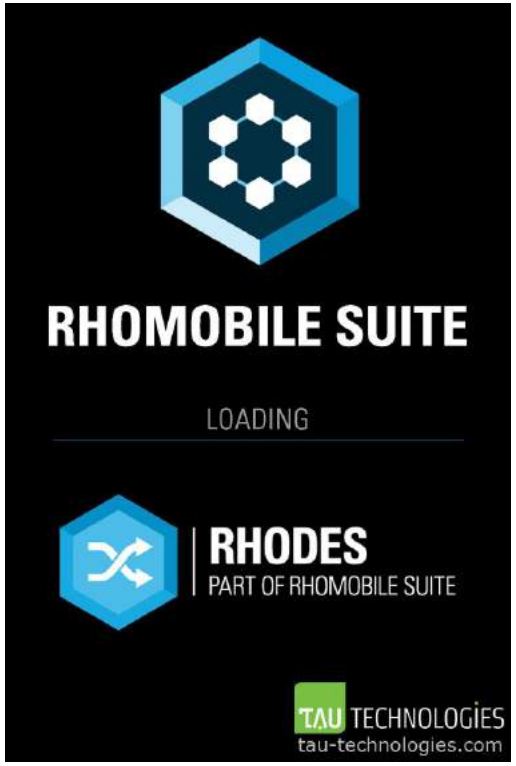
```
MyApp
└ nodejs
  └ server
    └ public
      └ bootstrap-3.3.7
        images
        jquery
      app.js
      package.json
    main.js
    rhoapp.js
  resources
  └ ios
    android
  build.yml
  rhoconfig.txt
  AndroidManifest.erb
  Rakefile
```

folder with resources used for application building: icon, splash image, iTunes image etc.

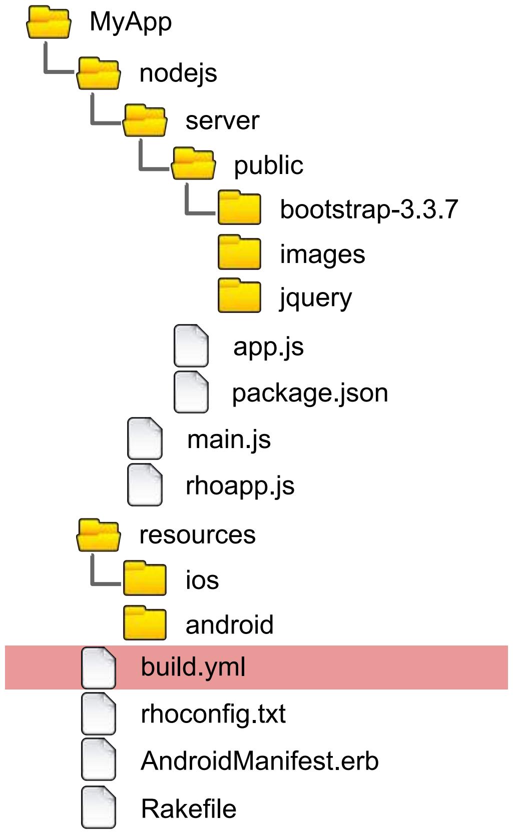
See details in documentation:

http://docs.tau-technologies.com/en/6.0/guide/app_icon_splash

http://docs.tau-technologies.com/en/6.0/guide/build_ios



## 12. How it works. Mobile cross-platform application with Node.js



```
MyApp
└ nodejs
  └ server
    └ public
      └ bootstrap-3.3.7
        images
        jquery
      app.js
      package.json
    main.js
    rhoapp.js
  resources
  └ ios
    android
  build.yml
  rhoconfig.txt
  AndroidManifest.erb
  Rakefile
```

**build.yml** - application’s build config

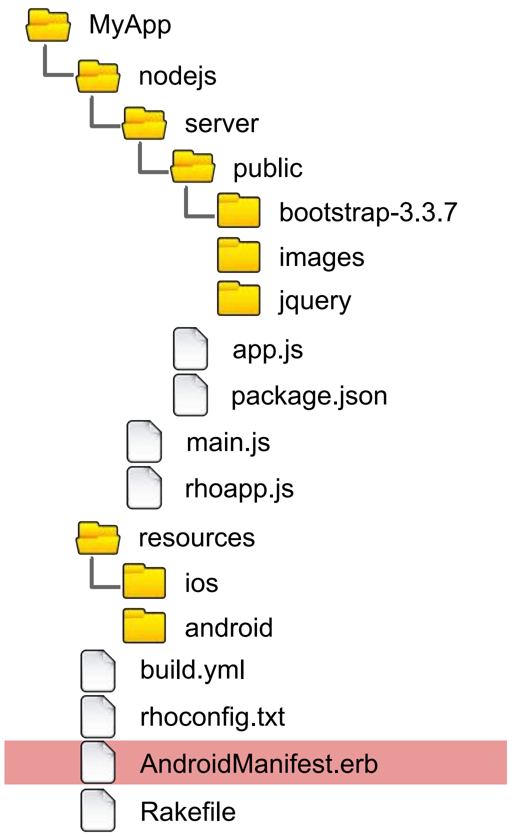
## 12. How it works. Mobile cross-platform application with Node.js

application’s runtime configuration

```
MyApp
└ nodejs
  └ server
    └ public
      └ bootstrap-3.3.7
        images
        jquery
      app.js
      package.json
    main.js
    rhoapp.js
  resources
  └ ios
    android
  build.yml
  rhoconfig.txt
  AndroidManifest.erb
  Rakefile
```

```
# startup page for your application
start_path = '/'

options_path = '/app/Settings'

# Rhodes log properties
MinSeverity  = 1
LogToOutput = 1
MaxLogFileSize=50000
logserver = 'http://rhologs.heroku.com'
logname='MyApp'

syncserver = ''
sync_poll_interval=0

...
```

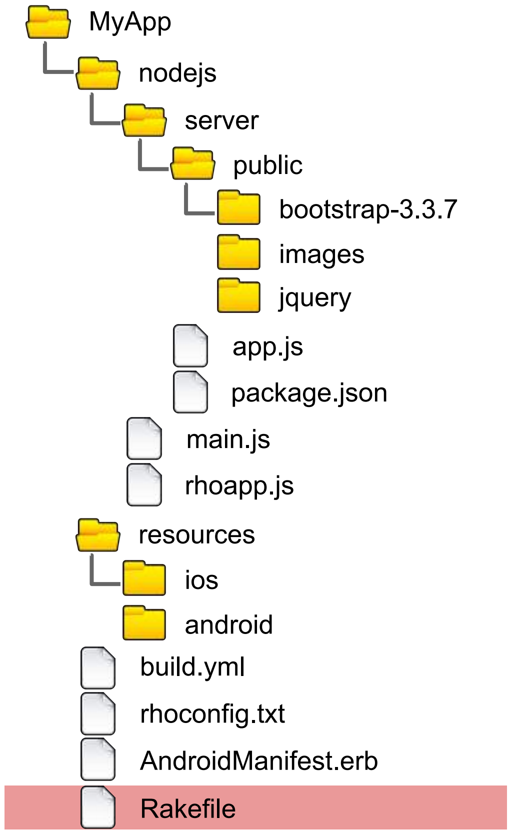
## 12. How it works. Mobile cross-platform application with Node.js



template for application’s Android manifest.
See details:
[http://docs.tau-technologies.com/en/6.0/guide/build_android](http://docs.tau-technologies.com/en/6.0/guide/build_android)

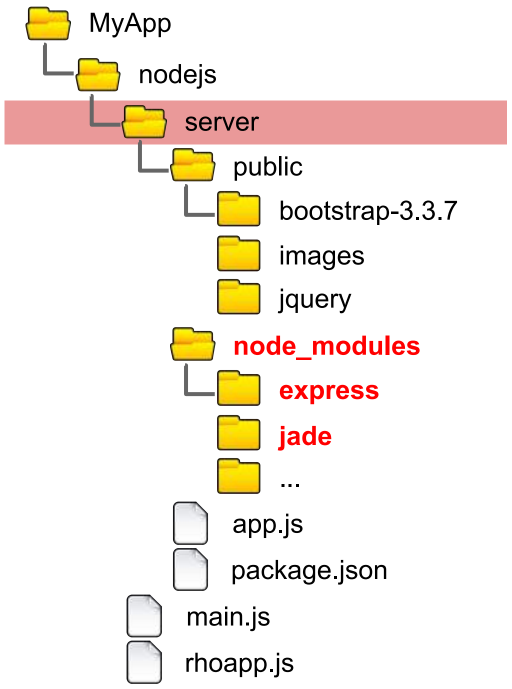
# 12. How it works. Mobile cross-platform application with Node.js

Standard Ruby script for rake commands (build, run, etc.)



```
MyApp
└─ nodejs
   └─ server
      └─ public
         └─ bootstrap-3.3.7
            images
            jquery
         app.js
         package.json
      main.js
      rhoapp.js
   resources
   └─ ios
      android
   build.yml
   rhoconfig.txt
   AndroidManifest.erb
   Rakefile
```

## 12. How it works. Mobile cross-platform application with Node.js



You should install node modules before building application. We should do the same with usual Node.js app - go to **nodejs/server** folder and run standard command :

```
$ npm install
```

After this command is done you can see new folder **node_modules** with all Node.js modules used.

## 12. How it works. Mobile cross-platform application with Node.js

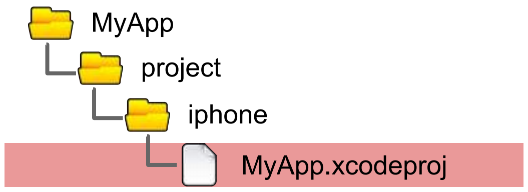
Let’s run our application on iPhone Simulator:

```
$ rake run:iphone
```

Also you can generate XCode project and use XCode for build/run etc.
Generate XCode project :

```
$ rake rake build:iphone:setup_xcode_project
```

Generated XCode project located in:



Details: [http://docs.tau-technologies.com/en/6.0/guide/build_ios](http://docs.tau-technologies.com/en/6.0/guide/build_ios)

## 12. How it works. Mobile cross-platform application with Node.js

Run application on Android Emulator:

```
$ rake run:android
```

Build and run application on linked via USB Android device :

```
$ rake run:android:device
```

Details: http://docs.tau-technologies.com/en/6.0/guide/build_android

# 12. How it works. Mobile cross-platform application with Node.js

**Simple example - NodeJSApp**

```
NodeJSApp
└─ nodejs
   └─ server
      └─ public
         └─ bootstrap-3.3.7
            images
            jquery
            index.html
         app.js
         package.json
      main.js
      rhoapp.js
   resources
   └─ ios
      android
   build.yml
   rhoconfig.txt
   AndroidManifest.erb
   Rakefile
```

Source code - https://github.com/tauplatform/NodeJSApp

In run time configuration file **rhoconfig.txt** we define start URL:
**/public/index.html**
this is URL on our local HTTP server, started in **/app.js**

So in our application's WebView following URL will be opened:
**http://127.0.0.1:port/public/index.html**

```
# startup page for your application
start_path = '/public/index.html'

...
```

## 12. How it works. Mobile cross-platform application with Node.js

**Simple example - NodeJSApp**

```
NodeJSApp
└ nodejs
   └ server
      └ public
         └ bootstrap-3.3.7
           images
           jquery
           index.html
        app.js
        package.json
     main.js
     rhoapp.js
  resources
  └ ios
    android
  build.yml
  rhoconfig.txt
  AndroidManifest.erb
  Rakefile
```

Source code - https://github.com/tauplatform/NodeJSApp

In main Node.js application file (app.js) we register global function, where we write some into application's log and execute some Javascript code in application's WebView:

```
…

global.myfunc = function() {
    Rho.Log.info("$$$$$$$$$$$$$$$$$ RUN NODEJS CODE !!!", 'Node.js JS');
    Rho.WebView.executeJavascript("showAlert();");
};

...
```

# 12. How it works. Mobile cross-platform application with Node.js

**Simple example - NodeJSApp**

```
NodeJSApp
└ nodejs
   └ server
      └ public
         └ bootstrap-3.3.7
           images
           jquery
           index.html
        app.js
        package.json
     main.js
     rhoapp.js
  resources
  └ ios
    android
  build.yml
  rhoconfig.txt
  AndroidManifest.erb
  Rakefile
```

Source code - [https://github.com/tauplatform/NodeJSApp](https://github.com/tauplatform/NodeJSApp)

**onExecuteNodejsCode()** is called on button press

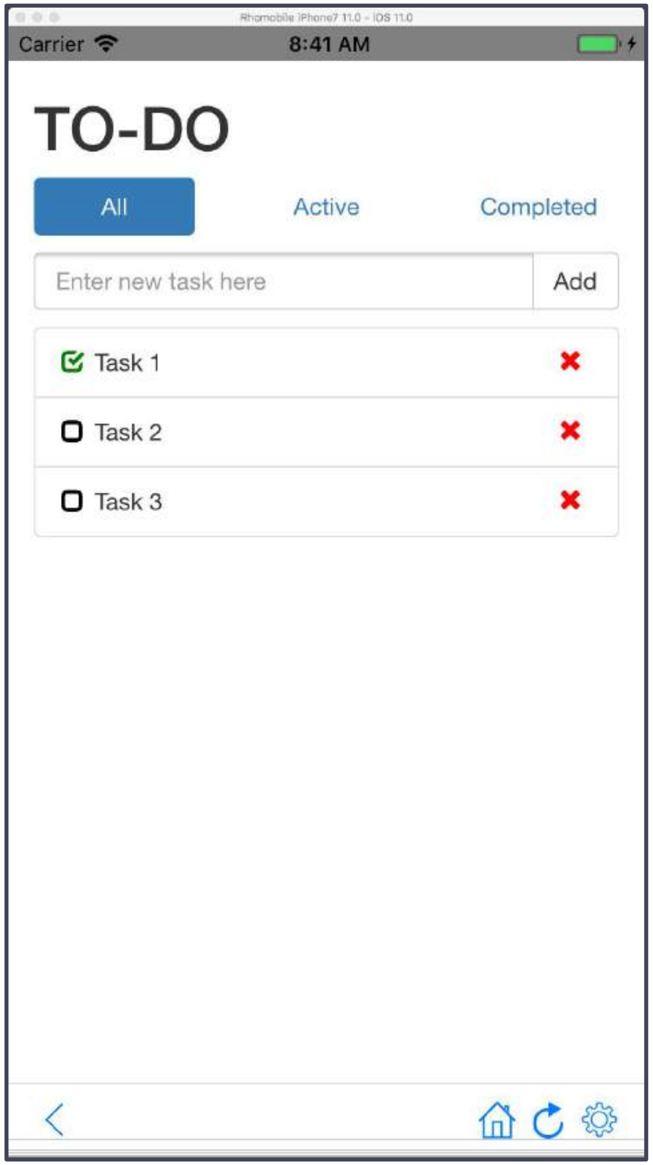
**showAlert()** will be called from Node.js code via Rhodes API - **Rho.WebView.executeJavascript()**

When user presses button :

1. JS code is executed from WebView, and this function invokes JS code in Node.js context via Rhodes API - **Rho.Nodejs.executeJavascript()**
2. Our JS code in Node.js context will execute JS code in WebView and we will see Alert.

```
<script type="text/javascript">

function onExecuteNodejsCode() {
    Rho.Nodejs.executeJavascript("myfunc();");
}

function showAlert() {
    alert("Alert !!!");
}

</script>
```

## Complex example - todo-nodejs

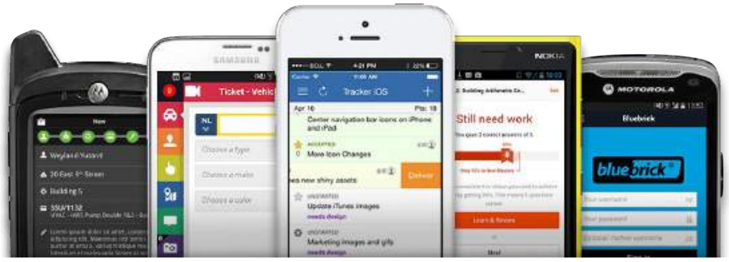


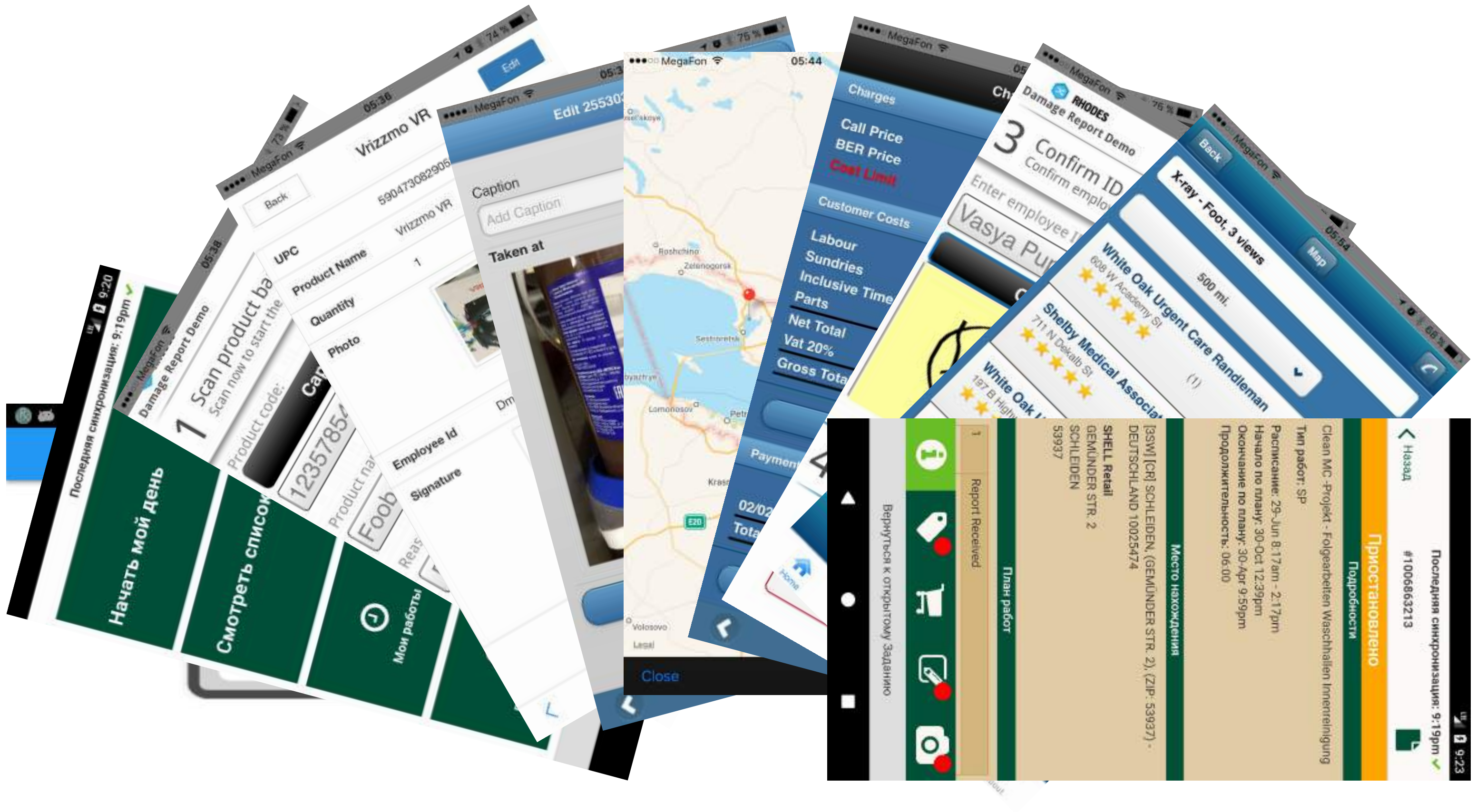
Source code - https://github.com/tauplatform/todo-nodejs

This is complex example of Node.js application - todo list of tasks
Uses following components:

- express (Node.js web framework)
- bootstrap (CSS)
- hbs (Express.js view engine for handlebars.js)
- sequelize (Node.js ORM)
- sqlite3 (DB)

# 12. Examples of real applications based on RhoMobile solution





## 13. How choose solution for development - Tau Quadrant

| Platform | type | Server code based on | | | |
|---|---|---|---|---|---|
| | | .NET | Java | Ruby | Javascript |
| iOS, Android | lightweight | Cordova/PhoneGap | Cordova/PhoneGap | Cordova/PhoneGap, Rhodes | Cordova/PhoneGap |
| | big | Xamarin | Xamarin, Codename One | Rhodes | Appcelerator, Native Script/Telerik, React Native, Kony |
| WinCE, WM and iOS, Android | Extended Browser is enough | Rhodes Browser Zebra Enterprise Browser (only for Zebra) | Rhodes Browser Zebra Enterprise Browser (only for Zebra) | Rhodes Browser Zebra Enterprise Browser (only for Zebra) | Rhodes Browser Zebra Enterprise Browser (only for Zebra) |
| | lightweight or big one | Rhodes | Rhodes | Rhodes | Rhodes |

## 14. Questions ?

http://tau-technologies.com