ESCOLA POLITÉCNICA DA UNIVERSIDADE DE SÃO PAULO

Departamento de Engenharia de Computação e Sistemas Digitais

Sistema NIR

Navegação Interna com RFID

Especificação Funcional

Aluno

Rodrigo Monteiro de Aquino

Orientador

Prof. Dr. Carlos Eduardo Cugnasca

São Paulo

2015

Catalogação-na-publicação

Aquino, Rodrigo
NIR - Navegação Interna com RFID / R. Aquino -- São Paulo, 2014.
2 p.

Trabalho de Formatura - Escola Politécnica da Universidade de São Paulo. Departamento de Engenharia de Computação e Sistemas Digitais.

1.Dispositivos Eletrônicos 2.Wireless I.Universidade de São Paulo. Escola Politécnica. Departamento de Engenharia de Computação e Sistemas Digitais II.t.

Sumário

1 INTRODUÇÃO

O projeto tem como objetivo a criação de um sistema de navegação *indoor* que seja capaz de localizar uma pessoa dentro de um ambiente, e com esta informação, seja capaz de disponibilizar conteúdo informativo (texto, áudio e vídeo) relevante para ela e coerente à localização. A localização será fornecida por um *gadget* que ficará acoplado ao tênis do usuário e se conectará via *Bluetooth* ao dispositivo Android do usuário (ou fornecido a ele).

2 O PROBLEMA

O mapeamento e navegação em ambientes fechados é um desafio que já possui uma série de soluções, como: triangulação por antenas internas (*beacon*), cálculo de distância via potência de sinal *Bluetooth* ou rede *Wireless*.

Estas soluções acabam acarretando alguns custos, como: alto custo na compra, manutenção cara devido à mão de obra especializada e alto consumo energético das antenas internas e dos dispositivos sem fio. Estes custos acabam encarecendo as soluções a médio/longo prazo, e ainda podem impactar na experiência do usuário, caso não sejam feitas as manutenções adequadas.

3 REQUISITOS

3.1 FUNCIONAIS

- A solução deve ser capaz de localizar a posição atual do usuário em um ambiente
- A solução deve exibir informação relevante ao usuário de acordo com a sua localização
- A informação deve ser apresentada por meio de uma interface em um aplicativo de *smartphone*
- Possuir um custo de implantação e manutenção menor do que outras soluções similares

3.2 NÃO FUNCIONAIS

- Autonomia energética que permita que o dispositivo funcione por ao menos 3 horas sem precisar recarregar;
- Possuir medidas reduzidas, da ordem de 5 cm cada

4 A SOLUÇÃO

A solução proposta possui um gasto inicial mínimo, com a utilização das *tags* RFID de 125kHz e do dispositivo leitor. Além disso, a utilização do dispositivo Android do próprio usuário torna o a experiência mais agradável, pois acaba se tornando uma solução pessoal e customizada.

Será instalado no dispositivo Android um aplicativo que será capaz de apresentar informações armazenadas em um servidor, baseado na localização do usuário.

As tags RFID serão espalhadas pelo ambiante a ser mapeado, por exemplo em um museu onde cada obra possui informações relevantes ou em uma feira de ciências, onde cada projeto pode oferecer um resumo de seu projeto ou informações adicionais dele.

No caso de uma mudança das obras ou atualização das informações que devem ser apresentadas, basta atualizar estas informações no sistema que é apresentado ao usuário, uma vez que as tags armazenam apenas seu valor identificador, e não informação referente ao que elas mapeiam.

Este projeto é importante também pois o produto resultante dele pode ser utilizado como um dispositivo de inclusão para pessoas com deficiências visuais em ambientes como museus.

4.1 ARQUITETURA

O dispositivo projetado lê *tags* RFID que estão posicionados em lugares relevantes no chão de um museu. Esta leitura determina em qual posição a pessoa está e permite que o aplicativo instalado no *smartphone* seja capaz de apresentar conteúdo específico para o usuário.

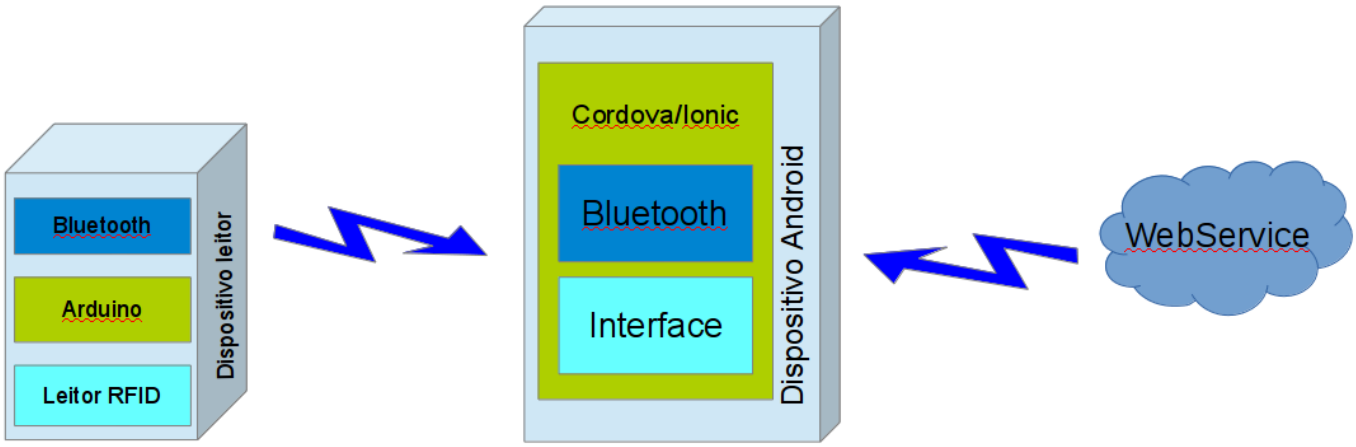


Figura 1: Modelo lógico da solução

A função do *hardware* é de ler o valor de uma *tag* RFID *enviar via Bluetooth* para um dispositivo sincronizado.

O *software* estará instalado no dispositivo Android, e terá como função o tratamento do valor recebido via *Bluetooth* e apresente a informação relativa a este valor para o usuário.

A aquisição destas informações pode ser feita por meia de um *webservice* que proverá um arquivo formatado com todas as informações relevantes.

4.2 HARDWARE

4.2.1 Componentes

- Leitor RFID ID12-LA;
- Arduino Micro Pro;
- Módulo bluetooth 2.0 HC-05;
- Resistores de 10kΩ e 20kΩ;
- Dispositivo Android.

4.2.2 Design do Circuito

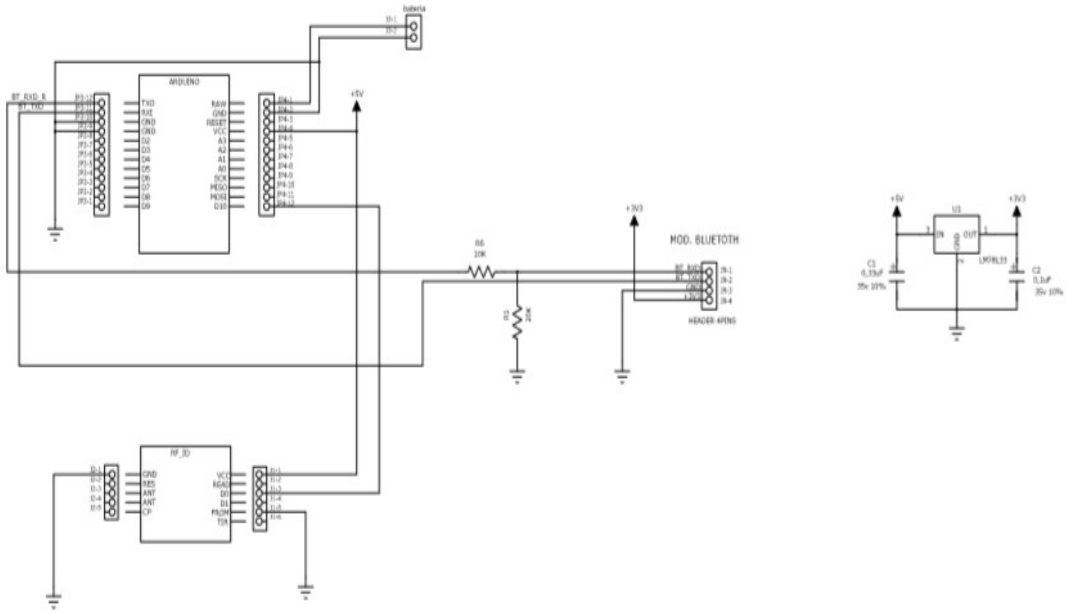
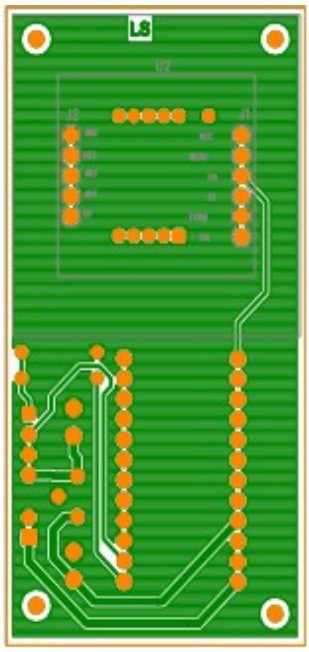


Figura 2: Detalhe do circuito do dispositivo

4.2.3 Design da placa



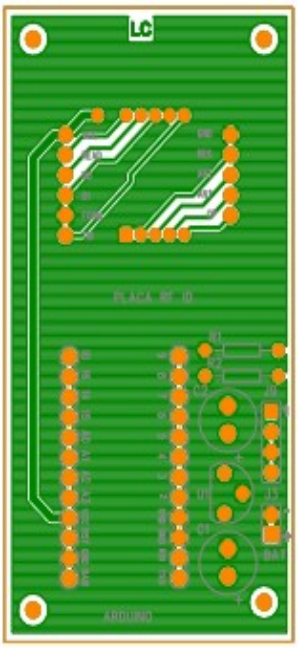


Figura 4: Detalhe da parte superior da placa criada

Figura 3: Detalhe da parte inferior da placa criada

4.3 SOFTWARE

4.3.1 Componentes

- Sistema Android;
- Framework Cordova e Ionic;
- Interface em Html5, CSS3 3 Javascript
- *Webservice* NodeJS
- *Middleware* Express

No desenvolvimento da interface, que apresentará as informações ao usuário, será utilizado *frameworks* que facilitam na criação de aplicações mobile.

Estes *frameworks* possuem *plugins* que são facilmente acopláveis a qualquer projeto e permitem que se possa utilizar itens de *hardware* disponíveis no *smartphone*.

Neste projeto, utilizou-se o plugin de bluetooth, que é capaz de utilizar o bluetooth do smartphone para localizar e parear com um outro dispositivo bluetooth.

Como medida de segurança e controle, apenas dispositivos conhecidos podem parear com este aplicativo, portanto apenas os dispositivos que foram projetados para o projeto podem ser pareados. Isto é possível devido à necessidade do bluetooth de utilizar o MAC Address para o pareamento. Deste modo, apenas os MAC Addresses cadastrados na aplicação serão pareados.

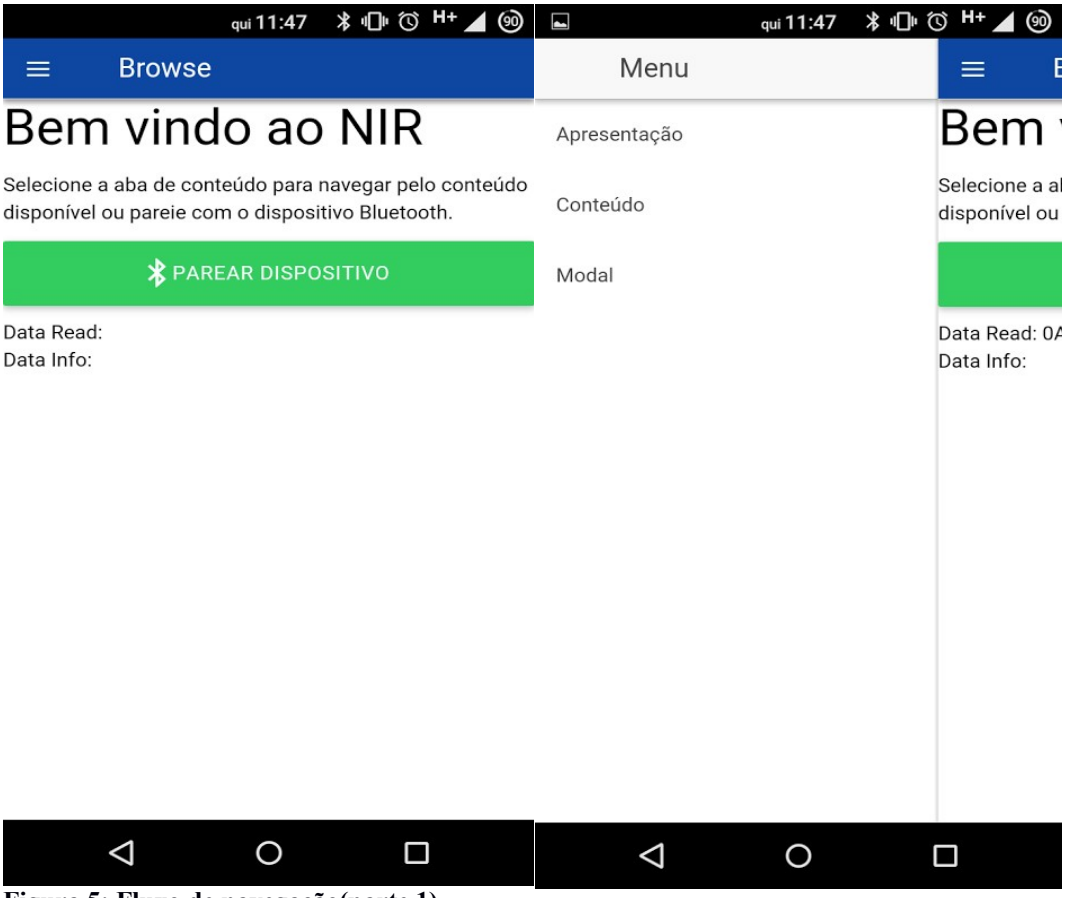


Figura 5: Fluxo de navegação(parte 1)

Figura 6: Fluxo de navegação(parte 2)

Na Figura 6 observa-se que algumas informações são apresentadas em uma lista, e a partir dela pode-se obter informações detalhadas de uma obra, por exemplo.

Estas informações são adquiridas por meio de um serviço que será consumido pelo aplicativo, como apresentado na Figura 7 abaixo.

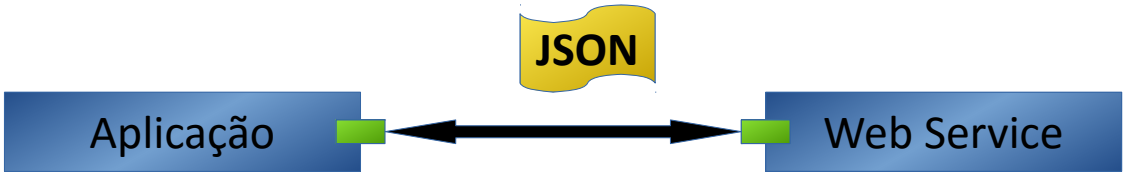


Figura 7: Webservice de conteúdo

O *webservice* é implementado com o *runtime* NodeJS, utilizando o *Middleware* Express.

Por meio dele o aplicativo é capaz de acessar dados previamente cadastrados no servidor utilizando-se de rotas definidas.

Cada rota é um *link* o qual é interpretado pelo NodeJS como algo a ser tratado e respondido ao cliente que acessa o serviço.

5 MODELO DO PROTÓTIPO

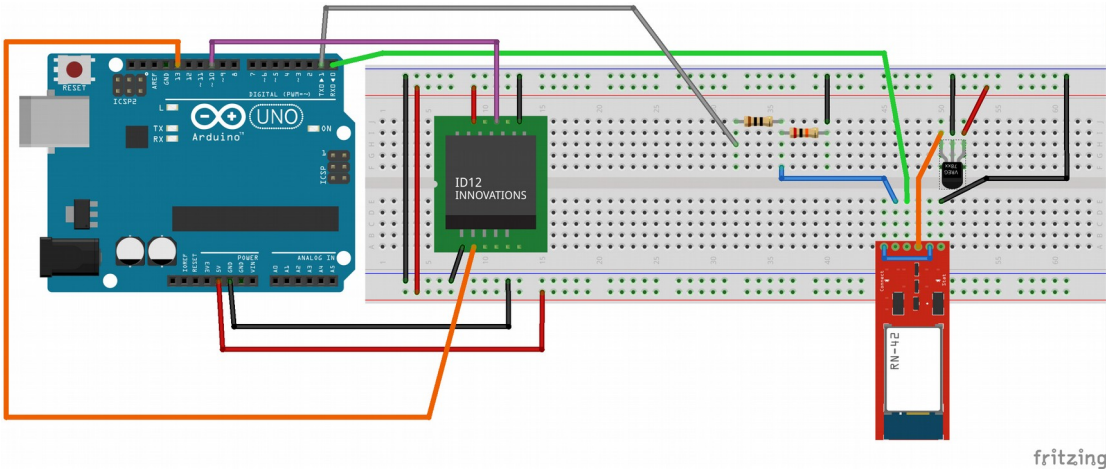


Figura 8: Ligação física dos componentes

Na Figura 8 está a representação da ligação dos componentes do circuito utilizado. No circuito final, no entanto, foi utilizado um arduino micro pro devido às suas dimensões reduzidas e o módulo bluetooth foi o HC-05, mas as ligações mantiveram-se iguais.

Deve-se notar que o arduino possui uma alimentação de 5 V e o módulo bluetooth de 3.3V, portanto teve-se que usar um regulador de tensão nos componentes necessários.

A bateria utilizada possui 9 V de tensão, e pode ser colocada na entrada *raw* do arduino sem problemas, pois ele converte esta entrada para 5 V internamente.

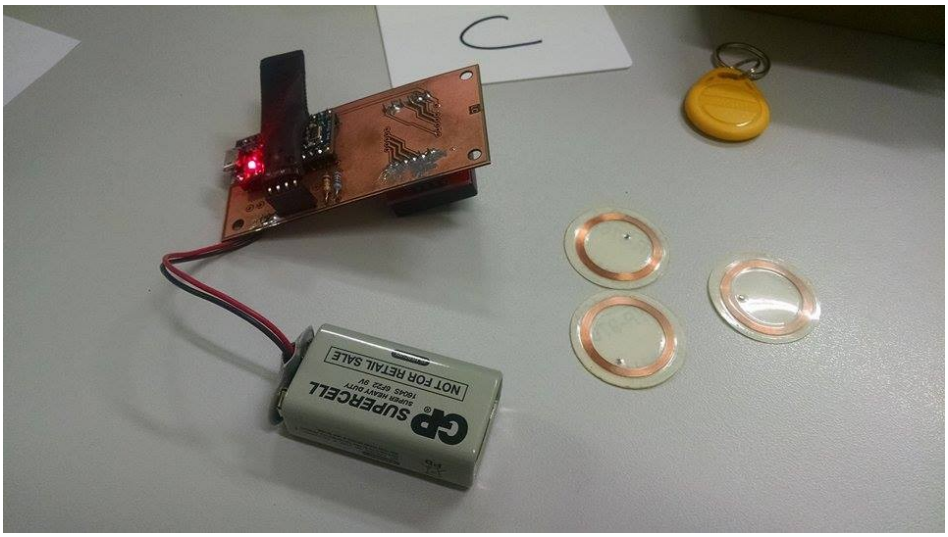


Figura 9: Protótipo do módulo de controle e Tags RFID

O arduino executa uma leitura a cada 1 segundo por meio do leitor RFID, e envia o valor lido via bluetooth para o dispositivo Android pareado previamente.

No dispositivo Android, a aplicação esperará um valor pela porta do *Bluetooth*, que atuará em sua interface com o usuário para apresentar a informação relativa àquele valor lido.

Dada a necessidade do dispositivo ter dimensões reduzidas, foi projetada uma nova versão da placa de circuito impresso, com os componentes ordenados de uma melhor forma e utilizando as duas faces da placa.

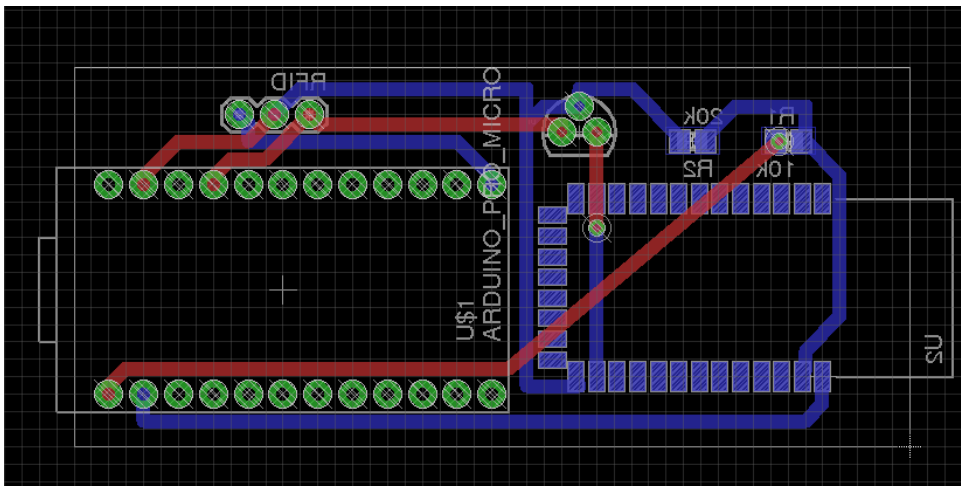


Figura 10: Circuito v2

6 CONCLUSÃO

O projeto foi bem-sucedido, de modo que foi possível desenvolver um dispositivo funcional com medidas reduzidas que permite a implantação de um sistema de oferta de informações utilizando-se de um mapeamento de um ambiente interno com um baixo custo de investimento e manutenção.

Ocorreram alguns empecilhos no decorrer do desenvolvimento, como dificuldades na confecção da segunda versão da placa dupla face e a complexidade na solda de componentes pequenos nela. Estes problemas foram impactantes apenas pela falta de tempo decorrente do pouco tempo disponível para corrigi-los.

7 BIBLIOGRAFIA

1. Trussel, Mark. Reading RFID Tags with an Arduino. Acessado em: 29/11/2015. URL: http://www.instructables.com/id/Reading-RFID-Tags-with-an-Arduino/
2. Laboratório de Garagem. Tutorial Acionamentos com tags RFID 125kHz. Acessado em 29/11/2015. URL: http://labdegaragem.com/profiles/blogs/tutorial-acionamentos-com-tags-rfid-125khz
3. Coleman, Don. Bluetooth Serial Plugin for Phonegap. Acessado em 29/11/2015. URL: https://github.com/don/BluetoothSerial
4. Ionic. Ionic Book. Acessado em 29/11/2015. URL: http://ionicframework.com/docs/guide/
5. Sevilleja, Chris. Build a RESTful API using NODE and Express 4. Acessado em 29/11/2015. URL: https://scotch.io/tutorials/build-a-restful-api-using-node-and-express-4

8 ANEXOS

8.1 CÓDIFICAÇÃO DO DISPOSITIVO (ARDUINO)

```
#include <SoftwareSerial.h>

char ID ;

SoftwareSerial mySerial(10, 11); // RX, TX

void setup()
{
 // Open serial communications and wait for port to open:
 Serial.begin(9600);
 while (!Serial) {
  ; // wait for serial port to connect. Needed for Leonardo only
 }

 Serial.println("Serial port started");

 Serial1.begin(9600);
 while (!Serial1) {
  ; // wait for serial port to connect. Needed for Leonardo only
 }

 Serial.println("BT started");

 // set the data rate for the SoftwareSerial port
 mySerial.begin(9600);

 Serial.println("RFID Serial port started");

}

void loop() // run over and over
{
 if (mySerial.available()){
   ID = mySerial.read();
   Serial.write(ID);
   Serial1.write(ID);
  }
}
```

8.2 CÓDIFICAÇÃO DO APLICATIVO (ANDROID – FRAMEWORK IONIC)

8.2.1 index.html

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8">
    <meta name="viewport" content="initial-scale=1, maximum-scale=1, user-scalable=no,
width=device-width">
    <meta http-equiv="Content-Security-Policy" content="default-src *; style-src 'self'
'unsafe-inline'; script-src 'self' 'unsafe-inline' 'unsafe-eval'">
    <title></title>

    <link href="lib/ionic/css/ionic.css" rel="stylesheet">
    <link href="css/style.css" rel="stylesheet">
    <link href="lib/ionic-material/dist/ionic.material.min.css" rel="stylesheet">

    <!-- IF using Sass (run gulp sass first), then uncomment below and remove the CSS
includes above
    <link href="css/ionic.app.css" rel="stylesheet">
    -->

    <!-- ionic/angularjs js -->
    <script src="lib/ionic/js/ionic.bundle.js"></script>

    <!-- cordova script (this will be a 404 during development) -->
    <script src="cordova.js"></script>

    <!-- your app's js -->
    <script src="js/app.js"></script>
    <script src="js/controllers.js"></script>
    <script src="js/services.js"></script>
    <script src="lib/ngCordova/dist/ng-cordova.js"></script>
    <script src="lib/ionic-material/dist/ionic.material.min.js"></script>

  </head>
  <body ng-app="nir">
    <ion-nav-view><ion-nav-view>
  </body>
</html>
```

8.2.2 app.js

```
angular.module('nir', ['ionic', 'nir.controllers','nir.services','ionic-material'])

.run(function($ionicPlatform) {
  $ionicPlatform.ready(function() {

    // Hide the accessory bar by default (remove this to show the accessory bar above the keyboard
    // for form inputs)
    if (window.cordova && window.cordova.plugins.Keyboard) {
      cordova.plugins.Keyboard.hideKeyboardAccessoryBar(true);
    }
    if (window.StatusBar) {
      // org.apache.cordova.statusbar required
      StatusBar.styleDefault();
    }
  });
})

.directive( 'showData', function ( $compile ) {
  return {
    scope: true,
    link: function ( scope, element, attrs ) {
      var el;

      attrs.$observe( 'template', function ( tpl ) {
        if ( angular.isDefined( tpl ) ) {
          // compile the provided template against the current scope
          el = $compile( tpl )( scope );

          // stupid way of emptying the element
          element.html("");

          // add the template content
          element.append( el );
        }
      });
    }
  };
})

.config(function($stateProvider, $urlRouterProvider) {
  $stateProvider

  .state('app', {
    url: "/app",
```

```
    abstract: true,
    templateUrl: "templates/menu.html",
    controller: 'AppCtrl'
  })

  .state('app.browse', {
    url: "/browse",
    views: {
      'menuContent': {
        templateUrl: "templates/browse.html",
        controller: 'BrowseCtrl'
      }
    }
  })
    .state('app.contents', {
      url: "/contents",
      views: {
        'menuContent': {
          templateUrl: "templates/contents.html",
          controller: 'ContentsCtrl'
        }
      }
    })

  .state('app.content', {
    url: "/contents/:contentId",
    views: {
      'menuContent': {
        templateUrl: "templates/content.html",
        controller: 'ContentCtrl'
      }
    }
  });
  // if none of the above states are matched, use this as the fallback
  $urlRouterProvider.otherwise('/app/browse');
});
```

8.2.3 controllers.js

```
angular.module('nir.controllers', ['ionic'])

.controller('AppCtrl', function($scope,$rootScope, $timeout, informationService,$stateParams,$q) {

    // With the new view caching in Ionic, Controllers are only called
    // when they are recreated or on app start, instead of every page change.
    // To listen for when this page is active (for example, to refresh data),
    // listen for the $ionicView.enter event:
    //$scope.$on('$ionicView.enter', function(e) {
    //});

    informationService.all().then(function(response){
        $rootScope.contents = response;
    },function(error){
        console.log("Error on response " + response)
    });
})

.controller('ContentsCtrl', function($rootScope, informationService, ionicMaterialInk, ionicMaterialMotic
    // The json data will now be in scope.
    ionicMaterialInk.displayEffect();
    ionicMaterialMotion.ripple()
    //informationService.all().then(function(response){
        //$rootScope.contents = response;
        //$rootScope.apply();
    //},function(error){
        //console.log("Error on response " + response)
    //});
})

.controller('BrowseCtrl', function($scope,$rootScope,informationService,$ionicModal, ionicMaterialInk,

    ionicMaterialInk.displayEffect();
    ionicMaterialMotion.ripple()
    // The json data will now be in scope.
    //var macAddress= "20:15:04:10:13:40";
    //var macAddress= "00:11:12:06:03:59";
    $scope.macAddress= "98:D3:31:B1:D1:22";

    $scope.pairBT = function () {

        bluetoothSerial.enable(function() {
            console.log("Bluetooth is enabled");
            $scope.info = "Bluetooth is enabled";
```

```
            $scope.$apply();
        },
        function() {

            console.log("The user did *not* enable Bluetooth");
            $scope.info = "The user did *not* enable Bluetooth";
            $scope.$apply();

        });


        bluetoothSerial.connect($scope.macAddress,function() {

            console.log("Connected to device");
            $scope.info = "Connected to device";
            $scope.$apply();
        },
        function() {

            console.log("Error connecting to device");
            $scope.info = "Error connecting to device";
            $scope.$apply();

        });

        bluetoothSerial.subscribe('\r',function(data) {

            $rootScope.data = data.substring(1).replace(/\s+/g, '');;

            $rootScope.callModal();

            //$rootScope.$apply();

            bluetoothSerial.clear();

        },
        function(data) {

            console.log("Error Subscribing");
            $scope.info = "Error Subscribing";

        });

    };
```

```
$scope.getBT = function() {
        bluetoothSerial.read(function(data) {
            console.log(data);
            $scope.data = data.substring(1);
            $scope.$apply();
        },function(data) {
            console.log("Error Reading");
            $scope.info = "Error Reading";

        });

    };


    $rootScope.callModal = function(){
        $rootScope.content = informationService.get($rootScope.data);
        console.log($rootScope.data);
        $rootScope.modal.show();
    };

    $ionicModal.fromTemplateUrl('templates/project.html', {scope: $rootScope})
        .then(function(modal) {
            $rootScope.modal = modal;
        });

    // Triggered in the modal to close it
    $rootScope.closeModal = function() {
        $rootScope.modal.hide();
    };

    // Open the login modal
    $rootScope.project = function() {
        $rootScope.modal.show();
    };

})

.controller('ContentCtrl', function($scope, $rootScope, $stateParams, informationService, $compile) {
    $rootScope.data = $stateParams.contentId;
    $rootScope.content = informationService.get($stateParams.contentId);
    //$scope.name = $scope.localContent.name;
    //$scope.description = $scope.content.description;

})
```

8.2.4 services.js

```
angular.module('nir.services', ['ngCordova'])

.factory('informationService', function($http) {
  var contentsList = [];

  return {
    all: function() {
      if (contentsList.length == 0){
        //return $http.get('http://192.168.0.106:4730/api/projects').then(function (response) {
        return $http.get('appdata/contents.json').then(function (response) {
          // Json loaded on service list
          //contentsList = response.data;
          contentsList = response.data["contents"];
          return contentsList;
        });
      };

      return contentsList;
    },

    get: function(contentId) {
     for (var i = 0; i < contentsList.length; i++) {
      if (contentsList[i].tag === contentId) {
        console.log(i);
        return contentsList[i];
      }
     }
     return null;
    },
  };
});
```

8.3 CÓDIFICAÇÃO DO SERVEIDOR (NODEJS)

8.3.1 app.js

```
var express = require('express'),
mongoose = require('mongoose'),
path = require('path'),
bodyParser = require('body-parser'),
routes = require('./routes/index'),
projects = require('./routes/projects');
var app = express();

mongoose.connect('mongodb://localhost/projectsTCC');

app.use(bodyParser.json());
app.use(bodyParser.urlencoded({ extended: false }));

app.use(function(req, res, next) {
   res.header('Access-Control-Allow-Origin', "*");
   res.header('Access-Control-Allow-Methods', 'GET,PUT,POST,DELETE');
   res.header('Access-Control-Allow-Headers', 'Content-Type');
   next();
});

app.use('/api', routes);
app.use('/api/projects', projects);


app.listen(process.env.PORT || 4730);

module.exports = app;
```

8.3.2 index.js

```
var express = require('express');
var router = express.Router();

/* GET home page. */
router.get('/', function(req, res, next) {
   res.json(200, { message: "Welcome to projectsTCC"});
});

module.exports = router;
```

8.3.3 projects.js

```
var express = require('express');
var router = express.Router();
var projectModel = require('../models/project');

// middleware to use for all requests
router.use(function(req, res, next) {
    // do logging
    console.log('Someone is accessing projects');
    next(); // make sure we go to the next routes and don't stop here
});

// return all projects
router.get('/', function(req, res, next) {
    projectModel.find(function(err, projects) {
        if (err)
            res.send(err);

        res.json(projects);
    });
    console.log('Showed all projects');
});


//create a new project
router.post('/',function(req, res) {
        console.log('Creating a new project');
        console.log(req.body.description);
        var project = new projectModel();      // create a new instance of the project model
        project.name = req.body.name;  // set the project name (comes from the request)
        project.description = req.body.description;  // set the project description (comes from the request)

        // save the project and check for errors
        project.save(function(err) {
            if (err)
                res.send(err);
            else
                res.json({ message: 'project created!' });
        });
});

// get specific project
router.get('/:project_id',function(req, res) {

    projectModel.findById(req.params.project_id, function(err, project) {
```

```
        if (err)
            res.send(err);
        else
            res.json(project);
    });
});

// update a project
router.put('/:project_id',function(req, res) {

    // use our project model to find the project we want
    projectModel.findById(req.params.project_id, function(err, project) {

        if (err)
            res.send(err);
        else{
            project.name = req.body.name;  // update the projects info
            project.description = req.body.description;  // update the projects info

            // save the project
            project.save(function(err) {
                if (err)
                    res.send(err);
                else
                    res.json({ message: 'project updated!' });
            });
        }


    });
});


module.exports = router;
```