01

Lendo código de barra na App

Transcrição

Estamos desenvolvendo o Garçonapp, um aplicativo Cordova que será usado pelos funcionários do restaurante para anotarem os pedidos. Já implementamos várias funcionalidades, mas nos pediram para incluir uma nova: uma maneira mais fácil de registrar o número da mesa que fez o pedido.

No app, quando confirmamos o pedido, ele abrirá uma janelinha na qual anotaremos qual mesa fez o pedido. Iremos implementar uma forma, em que não será necessário digitar o número da mesa. Com um ícone que ficará no topo do app, teremos a opção de fotografar um código de barras (um QR Code) que irá revelar o número da mesa. Criaremos está funcionalidade.

Primeiramente, vamos colocar o ícone que fará a ação desejada.

Na parte de HTML do projeto, temos um trecho do código referente ao Menu, onde usamos o ícone `more_vert` . Vamos incluir outro ícone. No Materialize usamos ícones com nome de classe (`material-icons`), junto com alguns efeitos e o nome `camera` .

```
<div>
    <i class="material-icons waves-effect waves-light waves-circle">camera</i>
    <i data-activates="submenu" datas-gutter="5"
       data-contrainwidth="false"
        class="material-icons waves-effect waves-ligth waves-circle dropdown-button">more_vert<,

</div>
```

Se abrirmos o projeto no navegador, já teremos o ícone de câmera na parte superior. Agora precisamos que o ícone funcione e que, ao clicarmos nele, a câmera seja aberta. Para fazermos isto, usaremos um plugin que adicionará este tipo de funcionalidade.

Vamos procurar no Google um plugin para ler QR Code. Usarei como termo de pesquisa: "phonegap barcode". Encontraremos no resultado, uma página do GitHub (https://github.com/phonegap/phonegap-plugin-barcodescanner) onde poderemos baixar o `phonegap-plugin-barcodesscanner` . Se lermos a documentação, veremos que ele suporta diversas plataformas (Android, iOS, Windows, BlackBerry e Browser). Também trabalha com diferentes tipos de código, entre eles QR_CODE, DATA_MATRIX e outros. Em seguida, instalaremos o plugin.

No Terminal, usaremos o comando `cordova plugin add` e o nome do plugin.

```
garconapp $ cordova plugin add phonegap-plugin-barcodescanner --save
```

Vamos instalá-lo com `--save` , para aproveirá-lo no `xml` como fizemos anteriormente. O plugin será instalado. Agora, como iremos usá-lo? Na documentação, encontraremos um código que devemos incluir no JS do projeto.

```
cordova.plugins.barcodeScanner.scan(
     function (result) {
```

```
            alert("We got a barcode\n" +
                  "Result: " + result.text + "\n" +
                  "Format: " + result.format + "\n" +
                  "Cancelled: " + result.cancelled);
        },
        function (error) {
            alert("Scanning failed: " + error);
        }
    );
```

Mas, como queremos que o escaneamento seja feito apenas quando o usuário toque no ícone da câmera, vamos incluir a classe `scan-qrcode` no HTML.

```
<i class="<material-icons waves-effect waves-light waves-circle scan-qrcode">camera</i>
```

Depois, na parte de baixo do arquivo Java Script, especificaremos que quando o usuário clicar no `scan-qrcode` iremos escanear o QR Code.

```
$('.scan-qrcode').click(function(){
    cordova.plugins.barcodeScanner.scan();
});
```

Ele irá avisar qual arquivo foi escaneado através de um codec, que iremos receber com `resultado`. Adicionaremos `alert` para o resultado.

```
$('.scan-qrcode').click(function(){
    cordova.plugins.barcodeScanner.scan(function(resultado));
    alert(resultado.text);
    });
});
```

Será que poderemos testar esta funcionalidade no navegador? Como precisaremos de uma câmera, só poderemos testar os plugins nativos do Cordova, se rodarmos em um dispositivo móvel.

Antes, vamos observar um ponto. O usamos no código o `cordova.plugins.barcodeScanner`. De onde veio este objeto Cordova? Se abrirmos o arquivo HTML, teremos elementos JS, mas faltou um `script` que faça referência ao objeto `cordova`. Não fizemos esta ação antes, porque não tinhamos nenhum plugin JS antes. Mas precisamos adicionar o arquivo `cordova.js`.

```
<script src="cordova.js"></script>
<script src="js/jquery.min.js"</script>
<script src="js/materialize.min.js"</script>
<script src="js/app.js"</script>
```

Quando fizermos a exportação do app, ele irá gerar um arquivo chamado `cordova.js` com todas as interfaces dos plugins instalados. É lá encontraremos o `cordova.plugins.barcodeScanner`.

Se tentarmos abrir o HTML no navegador, aparecerá uma mensagem de erro, porque ainda não temos o arquivo `cordova.js` criado localmente. Ele só irá existir, quando gerarmos o aplicativo nativo. Vamos então, rodá-lo no

dispositivo.

No Terminal, usaremos o comando `cordova run android`. Com um celular conectado ao computador, ele irá "buildar" o projeto.

Para fazer o teste com o scanner, precisaremos de um QR Code. No Google, iremos buscar uma página de criação de QR Code. Quando digitar o número da mesa, por exemplo "17", ele irá o código. Com o celular, faremos o escaneamento. Clicamos no ícone da câmera, em seguida, ele abrirá o dispositivo, assim poderemos escanear o QR Code. Ele realizou o processo rapidamente e mostrou o *Alert* com o número 17.

Nosso objetivo não é que ele mostre o *Alert*, mas sim, que ele implemente a funcionalidade de preencher o número da mesa no Garçonapp. Então, vamos substituir no código a linha do `alert` por `Materialize.toast` que irá imprimir a mensagem "Mesa" e o `resultado.text`.

```
$('.scan-qrcode').click(function(){
    cordova.plugins.barcodeScanner.scan(function(resultado));
    Materialize.toast('Mesa ' + resultado.text, 2000);
    });
});
```

Faremos outra alteração: havíamos incluído um `input` no HTML, para que o usuário pudesse preencher manualmente o número da mesa.

```
<div class="modal modal-fixed-footer" id="confirmacao">
    <div class="modal-content">
    <h5>Resumo do pedido</h5>
    <input type="number" class="validade" placeholder="Número da mesa" id=
    </div>
</div>
```

Com o QR Code, este `input` deverá ser preenchido automaticamente, sem que alguém precise digitar o número. Faremos isto, no arquivo JS, o `input` será `numero-mesa` e queremos que o valor dele seja o que `resultado.text` escaneado. Adicionaremos um `if` por garantia, porque às vezes, ele não traz o `text` preenchido.

```
$('.scan-qrcode').click(function(){
    cordova.plugins.barcodeScanner.scan(function (result){
        if (resultado.text) {
            Materialize.toast('Mesa ' + resultado.text, 2000);
            $('#numero-mesa').val(resultado.text);
        }
    });
});
```

Adicionaremos também uma segunda função de *callback*, para quando ocorrer um erro. Será um outro `Materialize.toast`, que irá mostrar o `Erro`. Vamos incluir outra propriedade com a cor do texto que queremos que seja exibido o erro (`red-text`).

```
function(erro{
    Materialize.toast('Erro ' + erro, 2000, 'red-text');
    });
```

Ao integrarmos o plugin, no app, poderemos fazer o escaneamento.

Vamos rodar de novo na linha de comando. Ele fará a instalação da versão mais recente no aparelho e poderemos escanear o QR Code.

Voltamos ao site de geração de QR Code e testaremos com um outro número, "51".

No app, após fazer o pedido e confirmá-lo, a janela de resumo terá o campo "Número da mesa" vazio. Cancelaremos o pedido e selecionaremos novamente os produtos. Em seguida, vamos clicar no ícone da câmera e escanear o QR Code. Quando confirmarmos o pedido, já veremos no resumo, o número da mesa. Ele preencheu automaticamente a informação sobre a mesa, com o número do QR Code.

Resumindo, o nosso app possui esses poderes nativos com a instalação dos plugins. Basta pesquisar o plugin com a funcionalidade desejada, depois instalá-lo, e verificar na documentação o código para o fazer funcioná-lo. Assim, o plugin vai funcionar no iOS, Android e Windows, de maneira multiplataforma. Muito simples.