



CONCEPTES D'IONIC

Conceptes bàsic per inicialitzar-se en Ionic

Contenido

Introducció	3
Bloc I: Etiquetes HTML	4
1. Introducció	4
2. Conceptes d'etiqueta	4
3. Arxius xx.page.html	4
4. Etiqueta <ion-content>	4
5. Etiquetes de text	4
5.1. Paràgrafs.....	4
5.2. Llistes.....	4
5.3. Span.....	5
5.4. Negreta i cursiva.....	5
5.5. ion-item	5
6. Formularis	5
6.1. Entrada de text.....	5
7. Botons	6
8. Imatges.....	6
Bloc II: Fulla d'estils	7
1. Introducció	7
2. Relacionant elements amb els estils	7
2.1. Nom de l'element.....	7
2.2. Classes	7
2.3. Identificador	8
Bloc III: TypeScript.....	9
1. Introducció	9
2. Importació de Mòduls.....	9
3. Declaració de variables	9
3.1. Variables globals.....	9
3.2. Variables Locals	10
4. Bucles	10
4.1. Bucle for	10
4.2. Bucle for – each.....	11
4.3. Bucle if – else if.....	11
4.4. Bucle switch.....	11
4.5. Bucle while	12
4.6. Bucle do-while.....	12

5. Mètodes	12
Bloc IV: Ionic/Cli i el seu entorn	14
1. Introducció	14
2. NodeJS.....	14
3. Ionic/CLI.....	14
4. IDE (Integrated Drive Electronics)	14
5. Comadaments d'Ionic	14
Apèndix I: Crear una aplicació.....	16
1. Introducció	16
2. Generant el projecte	16
3. Modificant l'aplicació	17
4. Afegint un encapçalament	17
5. Creant un formulari.....	17
6. Treballant amb els estils.....	19
7. Programant la lògica	22
7.1. Definint les variables	22
7.2. Creant els mètodes	22
7.3. Vinculant les variables.....	23
8. Afegint una nova vista.....	24
9. Enllaçant pàgines.....	24
10. Afegint una imatge	25
11. Afegint llistes	25
11.1. Llista no numerada.....	25
11.2. Llista numerada.....	26
12. Afegint enllaços.....	26
13. Afegint un botó	26
14. Pàgina Nova.....	27
14.1. Afegint una imatge	27
14.2. Afegint un llistat	27
Apèndix II: Lector de pantalla	29
1. Introducció	29
2. Primera pantalla.....	29
3. Segona i Tercer pantalla.....	29

Introducció

Aquest és un petit recull dels conceptes de programació bàsics per a utilitzar Ionic. El recull està dividit en quatre grans blocs:

1. Etiquetes HTML.
2. Estils de CSS.
3. Programació amb TypeScript.
4. Ionic/Cli i el seu entorn.

Als blocs es veuran els conceptes bàsics i mínims per a començar a programar una aplicació utilitzant Ionic i al final del mateix es realitzarà una petita aplicació utilitzant el que s'ha vist.

Bloc I: Etiquetes HTML

1. Introducció

En aquest bloc es veuran les etiquetes bàsiques d'HTML / Ionic per a crear la part visual (User Interface) de l'aplicació. Durant els apunts es veuran tant les etiquetes d'Ionic, que tenen un estil definit pel framework, i les etiquetes HTML que no tenen un estil definit.

Com veurem, hi ha atributs que funcionen a les etiquetes d'Ionic i no a les d'HTML i a l'inrevés.

2. Conceptes d'etiqueta

El llenguatge HTML és un llenguatge d'etiquetes d'hipertext i bàsicament s'utilitzen etiquetes i no cal programació de codi. La gran majoria de les etiquetes d'HTML tenen una etiqueta d'obertura que marca l'inici i una etiqueta de tancament, que marca la finalització de la mateixa tal i com es mostra a continuació.

- Etiqueta d'obertura: `<body>`.
- Etiqueta de tancament: `</body>`.

En cas de que no es tanqui una etiqueta o no s'obri se'ns mostrarà la pantalla en blanc ja que l'aplicació no es mostrarà però no se'ns marcarà cap error durant la compilació de l'aplicació. Per aquest motiu aquest és un error d'execució.

Per evitar aquest tipus d'error s'aconsella que quan s'obri una etiqueta també es tanqui per tenir un mínim control de l'estructura de la part visual.

3. Arxius `xx.page.html`

Les etiquetes HTML estan als fitxers amb finalització `page.html`. cadascuna d'aquest fitxers serà una visualització (pantalla) diferent a la nostra aplicació.

4. Etiqueta `<ion-content>`

Aquesta etiqueta serà la que contendrà la part principal de les etiquetes que utilitzarem per a crear la interfase d'usuari.

5. Etiquetes de text

Hi ha diferents etiquetes per mostrar el text a l'aplicació, des de etiquetes de paràgraf fins a notes o cites.

5.1. Paràgrafs

La etiqueta que utilitzarem per a marcar el diferents paràgrafs serà la etiqueta `<p>`. Amb aquesta etiqueta marcarem l'inici i el final dels paràgrafs, tal i com es mostra a l'exemple.

```
<p>
    Això és un paràgraf.
</p>
```

5.2. Llistes

Per a crear llistes a la nostra aplicació es pot utilitzar les etiquetes ``, `` i `<ion-list>`. Les dos primeres pertanyen a HTML i la darrera a Ionic.

5.2.1. ``

L'etiqueta `` crearà un llistat no numerat i al seu interior contindrà l'etiqueta `` que marcarà cadascuna dels components del llistat tal i com es mostra a l'exemple.

```
<ul>
    <li>Primer element de la llista.</li>
    <li>Segon element de la llista.</li>
</ul>
```

5.2.2.

L'etiqueta **** crearà un llistat amb els seus elements numerats. Aquesta etiqueta també contendrà la seu interior elements ****. A continuació es mostra un exemple.

```
<ol>
    <li>Primer element de la llista.</li>
    <li>Segon element de la llista.</li>
</ol>
```

5.2.3. <ion-list>

Aquesta etiqueta és d'Ionic i el seu comportament serà similar a les altres dues etiquetes. Aquesta etiqueta al seu interior contendrà l'etiqueta **<ion-item>** per cada element del llistat tal i com es mostra a l'exemple.

```
<ion-list>
    <ion-item>Primer element de la llista.</ion-item>
    <ion-item>Segon element de la llista.</ion-item>
</ion-list>
```

5.3. Span

L'etiqueta **** no és pròpiament una etiqueta de text, però ens permet destacar un text amb atributs diferents, com seria un idioma diferent, o fer que aparegui d'un color diferent per destacar sobre la resta.

A continuació es mostra un exemple de l'etiqueta.

```
<p>
    Això es una paràgraf amb una <span>paraula</span> destacada
</p>
```

5.4. Negreta i cursiva

El format de negreta i cursiva es pot imposar mitjançant etiquetes o estils. Per fer-ho amb etiquetes s'utilitzarà **** per negreta i **<i>** per cursiva.

5.5. ion-item

Per a que el text es distribueixi un sota l'altre a Ionic es té tendència a utilitzar les etiquetes **<ion-item>** per fer que un element es situa a sota d'un altre element.

6. Formularis

Els formularis permeten la introducció de dades mitjançant el teclat del dispositiu i al seu interior pot contenir multitud d'etiquetes depenent de la informació que es vulgui recollir.

Per a crear un formulari utilitzarem l'etiqueta **<form>**.

6.1. Entrada de text

Per a introduir un text mitjançant el teclat podem utilitzar l'etiqueta **<input>** o **<ion-input>**. Aquestes etiquetes tindran sempre associada una etiqueta **<label>** on es dirà la informació que es demana.

Exemple amb etiquetes HTML.

```
<form>
  <label>Nom: </label>
  <input>
</form>
```

Exemple amb etiquetes d'Ionic.

```
<form>
  <ion-label>Nom:</ion-label>
  <ion-input></ion-input>
</form>
```

7. Botons

Els botons s'utilitzaran per enviar la informació recollida a un formulari o per obrir una nova vista de l'aplicació. Per aquest elements tenim una etiqueta d'HTML: **<button>** i una d'Ionic **<ion-button>**.

Exemple de botó amb HTML

```
<button>Fes click</button>
```

Exemple amb Ionic

```
<ion-button>Fes click</ion-button>
```

8. Imatges

Per inserir imatges en una vista podem utilitzar una etiqueta HTML **** o la d'Ionic **<ion-img>**. La diferència que hi ha entre una i altre és bàsicament que la primera carrega les imatges al carregar la vista i la segona només carrega les imatges quan aquestes son visibles a la vista.

Un exemple d'ús d'aquesta etiqueta seria;

```

```

L'etiqueta en pel cas d'Ionic s'utilitzarà de la següent manera:

```
<ion-img src="adreça de la imatge" alt="descripció de la imatge"></ion-img>
```

Bloc II: Fulla d'estils

1. Introducció

La fulla d'estils és un arxiu en format css o sass on es donen els estils dels elements de les vistes. Hi ha una fulla d'estils per a cada vista i una fulla d'estils general per tota l'aplicació. La fulla d'estils de la vista està a la mateixa carpeta que la resta d'arxius de la vista i té el nom de `xx.page.css`.

Per contra, la fulla d'estils generals es troba a `src/theme/global.css`. Si obrim aquest últim fitxer veurem que hi ha referències a les fulles d'estils generals.

2. Relacionant elements amb els estils

Per relacionar un element de la part visual (fitxer HTML) amb un estil (fitxer CSS) podem utilitzar alguna de les següents relacions:

- Nom de l'element.
- Classe (Class).
- Identificador (Id).

2.1. Nom de l'element

Possiblement utilitzar el nom de l'element és la forma més senzilla de relacionar un element amb el seu estil. El problema d'utilitzar aquest mètode és que l'estil s'aplicarà a tots els elements d'aquell tipus que contingui la vista.

Per exemple, si es vol que els paràgrafs d'una vista siguin amb una mida de lletra, una lineació i un color determinats a la fulla d'estils escriurem:

```
p {  
    font-size: 12pt;  
    text-align: justify;  
    color: blue;  
}
```

Amb l'exemple anterior hem fet que tots els paràgrafs de la vista tinguin una mida de lletra de 12 punts, una alineació justificada i siguin de color blau. Però això s'aplicarà a tots els elements paràgraf.

2.2. Classes

Les classes permeten que els estils s'apliquin als elements que nosaltres vulguem de la vista, i no han de ser elements del mateix tipus. Així per exemple, podem fer que els estils del punt anterior només s'apliquin a certs paràgrafs i a un enllaç, per exemple.

Per fer això haurem de relacionar els elements amb els estils. Per fer-ho a les etiquetes d'HTML afegirem l'atribut **class** de la següent manera:

```
<p class="important">Paràgraf on aplico estils</p>  
<p>Paràgraf on no aplico estils</p>
```

D'aquesta forma estem lligant a un dels paràgrafs amb un estil de la fulla d'estils. Ara, a la fulla d'estils escriurem:

```
.important{
```



```
font-size: 12pt;
text-align: justify;
color: blue;
}
```

Presteu atenció al . (punt) que hi ha davant la paraula important. Aquest punt especifica que estem davant una classe.

Cal recordar i emfatitzar que una classe pot afectar a més d'un element de la vista i que aquests elements podem ser de diferents tipus.

2.3. Identificador

Els identificadors permeten que l'estil afecti a una única etiqueta de la vista, és a dir, a diferència de les classes només poden afectar a una unitat. Per marcar una etiqueta HTML amb un identificador utilitzarem l'atribut **id** de la següent manera.

```
<p id="important">Únic element afectat</p>
```

D'aquesta manera estem identificant aquest element de la vista amb un identificador únic. A la fulla d'estils escriurem:

```
#important{
    font-size: 12pt;
    text-align: justify;
    color: blue;
}
```

El símbol # (coixinet) davant la paraula important ens marca que estem davant un identificador i que els estils només afectaran a l'element amb aquell indicador.

Cal remarcar que els identificadors son únics i que per tant, només afecten a un element de la vista; a diferència de les classes que poden afectar a diversos elements.

Bloc III: TypeScript

1. Introducció

Es podria dir que TypeScript es una evolució de JavaScript i és el llenguatge que utilitza Ionic per a la lògica de les aplicacions. En aquest bloc es veuran les característiques més bàsiques d'aquest llenguatge com és:

- Importació de Mòduls.
- Declaració de variables.
- Bucles.
- Mètodes.

Els arxius de TypeScript també estan inclosos dins la carpeta de la vista i reben el nom de `XX.page.ts`.

2. Importació de Mòduls

Quan volem importar un mòdul a TypeScript, el que seria una llibreria en altre llenguatges com JavaScript, primer s'ha gairebé sempre es importar el mòdul al fitxer **src/app/app.module.ts**. Per importar ho farem de la següent manera:

```
import { nomDelModul } from 'ubicació del mòdul';
```

Un cop importat el mòdul l'inclourem a l'array d'**imports** o de **providers**. Un cop fet això ja podrem utilitzar el mòdul a la nostre aplicació.

Ara, si volem utilitzar el mòdul simplement l'haurem d'importar a la vista on vol utilitzar-ho escrivint a la zona d'imports el següent.

```
import { nomDelModul } from 'ubicació del mòdul';
```

3. Declaració de variables

Hi ha dos tipus de variables: les globals i les locals. La diferència entre ambdues és que les primeres afecten a tot el fitxer i les segones només a una part del fitxer, normalment un mètode.

3.1. Variables globals

Les variables globals es declararan abans del **constructor()** i es farà de la següent manera:

```
nomDeLaVariable: tipusDeVariable;
```

Si es vol, es pot inicialitzar una variable just en el moment de la seva declaració. Al inicialitzar la variable li estem donant un valor inicial i no ens cal indicar el tipus de variable ja que TypeScript li assignarà un segons el seu valor. Encara que no calgui indicar el tipus s'aconsella fer-ho per deixar ben clar quin tipus de variables hem declarat.

La declaració i inicialització d'una variable global es fa de la següent manera:

```
nomDeLaVariable: tipusDeVariable = valorDeLaVariable;
```

Els tipus de variables de TypeScript son:

- Number.
- String.
- Boolean.

- Any.

Per utilitzar una variable global haurem d'utilitzar la paraula reservada **this** abans del nom de la variable de la següent manera:

```
this.nomDeLaVariable;
```

3.1.1. Variables number

Les variables **number** s'utilitzen amb nombre ja siguin sencer o decimals.

3.1.2. Variables string

Les variable **string** s'utilitzen per declarar cadenes de caràcters o nombres. En cas de declarar un nombre com a **string** no podrem utilitzar la variable per a realitzar operacions matemàtiques.

3.1.2 Variables boolean

Les variables **boolean** s'utilitzen per a declara una variable que només pot tenir com a valors **false** o **true** i s'utilitzen, generalment, per a la presa de decisions en un bucle.

3.1.3. Variables any

Les variables **any** permeten fer que una variable agafi qualsevol tipus de variable i és el que utilitzarem per a declara una variable que sigui un objecte o que no tinguem clar de quin tipus ha de ser.

3.2. Variables Locals

Les variables locals es declaren dins de mètodes i només afecta al mètode on es declara. La forma de declarar una variable local pot ser la mateixa que l'emprada per a declara una global afegint la paraula reservada **let** abans del nom de la variable, o inicialitzant-la al declarar-la de la següent manera:

```
let nomDeLaVariable = valorDeLaVariable;
```

Per utilitzar una variable local no hem d'afegir la paraula reservada **this** abans del nom de la variable.

4. Bucles

Els bucles permeten la presa de decisions segons els valors de les diferents variables. A TypeScript s'utilitzen els següents bucles:

- for.
- for – each.
- if – else if.
- switch.
- while.
- do – while.

4.1. Bucle for

El bucle **for** s'utilitza, generalment, per a recorre elements com els array de forma seqüencial i element a element i la seva declaració i ús és molt similar a la resta de llenguatges.

Un bucle for té el següent aspecte:

```
for(let i = 0; i < 10; i+1) {  
    variable[i] = 2*i+10;
```

```
}
```

La descripció d'aquest bucle seria la següent:

```
for( definició; condició; actualització){  
    execució  
}
```

4.2. Bucle for – each

És un cas especial del bucle **for** vist al punt anterior que es diferencia bàsicament en la seva declaració al no utilitzar-se l'actualització de la variable, ni caldre una definir una variable local.

Aquest tipus de bucle només poden recórrer els array element a element i es declara de la següent manera:

```
for (let element of this.elements) {  
    Element = 0;  
}
```

En aquest cas hem fet un bucle que recorre tots els elements (element) de l'array elements i els igual a 0.

4.3. Bucle if – else if

A diferència dels bucles anteriors, aquest bucle ens permet la presa de decisions dins de l'aplicació. A aquest bucle se l'imposa una condició que en cas de ser certa (**true**) realitzarà una o varies accions i en cas de ser falsa (**false**) realitzarà altres accions.

La seva declaració és de la forma:

```
if(condicio1){  
    accions si condició 1 és certa  
} else if (condicio2){  
    accions si condició 2 és certa  
}  
else {  
    accions si condició 1 i 2 son falses  
}
```

else if i **else** poden o no ser necessàries per al bucle. És a dir, si volen que una acció es realitzi només quan una condició sigui certa podem utilitzar només **if** sense tenir que emprar ni **else if** ni **else**.

4.4. Bucle switch

Aquest bucle és molt similar al bucle anterior i també permet la presa de decisions però la seva declaració és força diferent a la del bucle anterior, encara que també avalua quan una condició és certa.

La forma de declara aquest bucle és la següent:

```
switch (expressió){  
    case valor1:  
        accions si expressió = valor1;  
        break;
```

```

        case valor2:
            accions si expressió = valor2;
            break;
        default:
            accions per defecte;
            break;
    }

```

En aquest bucle s'avalua si la expressió és igual a un dels casos (**case**) i quan ho és es realitzen les accions que indica el cas. Un cop realitzades aquestes accions, si volem que no s'avaluïn més opcions hem d'escriure la paraula reservada **break** per sortir del bucle. En cas de que no s'escriu a **break** es passarà a comprovar si l'expressió és igual al següent valor.

Pel que fa a **default**, és l'acció que es realitzarà al finalitzar el bucle si no es sortirà del mateix. De la mateixa forma que **else** a un bucle **if -else**, no tenim obligació d'utilitzar una secció **default** a un bucle **switch**.

4.5. Bucle while

Un bucle **while** fa que una acció o accions es realitzin de forma continuada fins que la condició es deixa de complir. S'ha de tenir molt de compte en la utilització d'aquest bucle ja que podem imposar una condició que sigui certa sempre i per tant que entrem en un bucle infinit del que no puguem sortir.

Un exemple de declaració d'aquest bucle és el següent:

```

while(index < 5){
    accions;
    index++;
}

```

El bucle de l'exemple realitzarà la mateixa acció fins que la variable `index` agafi el valor 5.

4.6. Bucle do-while

El bucle **do – while** és molt semblant al bucle anterior amb la diferència que les accions dins del bloc **do** es realitzen al menys un cop.

La declaració d'aquest bucle es realitza de la següent manera:

```

do {
    accions;
    index++;
}
while(index < 5)

```

Aquest bucle realitzaria les mateixes accions que l'anterior però amb la diferència, com ja s'ha dit, que tenim la certesa que l'acció es realitzarà al menys un cop.

5. Mètodes

El mètodes o funcions són blocs de codi que s'executen al ser cridats ja sigui per un altre mètode o per un element de la part visual de l'aplicació com podria ser un botó.

Al declarar un mètode podem fer que aquest agafi una variable que se li transfereix al ser cridat. En cas de que el mètode retorni una variable (**return**), no hem d'especificar quin tipus de variable retorna, tal i com es fa en C/C++.

La declaració d'un mètode es fa de la següent manera:

```
nomDelMetodo(variable){  
    accions;  
    return valorVariable;  
}
```

En aquest cas al mètode nomDelMetodo se li ha passat la variable variable i ens retorna la variable valorVariable.

Com veurem més endavant, els mètodes són la part més important de la programació de l'aplicació al poder-se utilitzar com a peces d'un puzzle per a crear una aplicació.

Bloc IV: Ionic/Cli i el seu entorn

1. Introducció

En aquest bloc es mostrarà com instal·lar Ionic/Cli i el seu entorn al nostre equip i quins son els comandaments més importants i que hem de conèixer per poder desenvolupar una aplicació.

2. NodeJS

Per poder instal·lar Ionic primer haurem de descarregar l'aplicació NodeJS des de la seva [pàgina web](#) i instal·lar-la al nostre equip.

La instal·lació de l'aplicació es senzilla ja que simplement hem d'executar l'instal·lador. Un cop instal·lada obrirem un terminal o una línia d comandaments al nostre equip i escriurem:

node -v

Si la instal·lació ha sigut exitosa se'ns retornarà la versió de NodeJS instal·lada. Al instal·lar-se NodeJS també se'ns ha instal·lat **npm**. Per saber la seva versió escriurem:

npm -v

3. Ionic/CLI

Amb **NodeJS** ja instal·lat podem instal·lar Ionic/cli. Per fer-ho, obrirem un terminal o línia de comandaments i escriurem el següent:

npm install -g @ionic/cli

Després d'uns minuts tindrem Ionic instal·lat al nostre equip. Per comprovar que la instal·lació s'ha realitzat de forma correcta escriurem al terminal:

ionic -v

i si tot ha anat bé se'ns retornarà la versió d'Ionic que acabem d'instal·lar.

4. IDE (Integrated Drive Electronics)

Un Entorn de Desenvolupament Integrat, IDE per la seves sigles en anglès, és una aplicació que integra tot el que ens cal per desenvolupar una aplicació. En el cas d'Ionic es pot utilitzar un editor de notes com seria el bloc de notes de Windows, o un entorn més complexa com seria Visual Code.

Per realitzar les aplicacions es recomana utilitzar un IDE, ja sigui [Visual Studio Code](#), [Atom](#), [Angular IDE](#) o qualsevol altre. Aquests tres IDEs es poden descarregar de forma gratuïta de les seves pàgines web.

5. Comadaments d'Ionic

Els comandaments d'ionic més usuals son:

- **Iniciar un nou projecte:** ionic start nomProjecte --type=tipusProjecte frameWorkProjecte.
 - nomPrjecte és el nom del nostre projecte.
 - tipusProjecte pot ser:
 - tabs. Pestanyes.
 - sidemenu. Barra de menú lateral.
 - blank. Projecte buit.
 - my-first-app. Exemple d'aplicació.

- conference. Exemple d'aplicació.
 - frameWorkProjecte pot ser:
 - Angular.
 - React.
- **Executar projecte a un navegador:** ionic serve.
- **Generar un servei, pàgina, ...:** ionic g element element/nomElement.
 - Element pot ser se:
 - page per a generar una pàgina, vista, nova.
 - service per generar un servei.
 - component per generar un component.
 - module per crear un mòdul.
 - class per crear una classe.
 - directive per crear una directiva.
 - guard per crear un àrea privada.
 - pipe per crear un filtre.
 - interface per crear una interfase.
 - enum per crear una enumerador.
- **Crear una pàgina web:** ionic build.
- **Afegir una plataforma:** npx cap add plataforma.
 - ios afegirà el sistema iOS.
 - android afegirà el sistema Android.
- **Obrir el projecte nadiu:** npx cap open plataforma.
- **Executar en un terminal:** npx cap run plataforma.

Apèndix I: Crear una aplicació

1. Introducció

En aquest apèndix posarem en funcionament part del vist fins ara per crear una petita aplicació amb Ionic. L'aplicació mostrarà i amagarà certs valors introduïts en un formulari.

2. Generant el projecte

Per generar el nou projecte obrirem un terminal o línia de comandament i escriurem el següent:

```
ionic start laMevaApp blank --type=angular
```

Després d'uns minuts tindrem creat un nou projecte anomenada laMevaApp. El següent pas serà accedir a la carpeta del projecte amb el comandament:

```
cd laMevaApp
```

Si hem instal·lat Visual Code podem introduir el comandament **code** . per obrir el projecte a l'IDE. En cas contrari obrirem el projecte a l'IDE que tinguem instal·lat.

Un cop obert el nostre projecte introduïrem el comandament per a veure l'aspecte de l'aplicació a la terminal o línia de comandament:

```
ionic serve
```

D'aquesta forma es mostrarà l'aplicació al navegador que tinguem assignat per defecte. Si volem veure com seria l'aplicació en un dispositiu mòbil o tauleta anirem a **Configuració i més > més eines > Eines de desenvolupament**.

Blank

Ready to create an app?

Start with [Ionic UI Components](#)

Il·lustració 1. Aspecte inicial de l'aplicació.

Si volem veure totes les opcions que ens ofereix Ionic per a crear la interfase d'usuari podem pitjar a l'enllaç [Components](#). S'obrirà una nova pestanya del navegador que ens mostrarà els elements definits per crear la interfase d'usuari dels que disposa Ionic.

3. Modificant l'aplicació

Amb el projecte obert anirem a `src/app/home/home.page.html` i modificarem el títol de la vista per **La Meva Primer App**. Per modificar el títol esborrarem Blank, que està dins l'etiqueta `<ion-title>` i escriurem el nou títol. El següent pas serà esborrar tot el que hi ha dins l'etiqueta `<ion-content>`.

4. Afegint un encapçalament

El següent pas serà afegint un encapçalament al contingut de la vista. Per això utilitzarem l'etiqueta d'encapçalament de primer nivell `<h1>` on digui **Formulari d'accés**.

Recordem que aquesta etiqueta ha d'estar dins de l'etiqueta `<ion-content>`. Ara mateix el codi hauria de tenir la següent forma:

```
<ion-header [translucent]="true">
  <ion-toolbar>
    <ion-title>
      La Meva Primera App
    </ion-title>
  </ion-toolbar>
</ion-header>

<ion-content [fullscreen]="true">
  <h1>Formulari d'accés</h1>

</ion-content>
```

El següent pas serà la creació d'un formulari.

5. Creant un formulari

Per crear un formulari utilitzarem les etiquetes d'HTML i d'Ionic per veure les diferències en el seu comportament i visuals.

El primer que hem de fer és escriure l'etiqueta `<form>` a sota de l'encapçalament que acabem de crear. Aquesta etiqueta `<form>` contindrà tots els camps del formulari.

El primer que demanarem serà el nom de l'usuari. Per tant, haurem d'utilitzar una etiqueta `<label>` i una `<input>` de la següent forma:

```
<label for="name">Nom</label>
<input name="name" placeholder="Escriu el teu nom">
```

Aquestes dues etiquetes son d'HTML i les vinculem entre elles mitjançant l'atribut **for** a l'etiqueta `<label>` i **name** a l'etiqueta `<input>`. Si ens fixem, els valor dels dos atributs és el mateix.

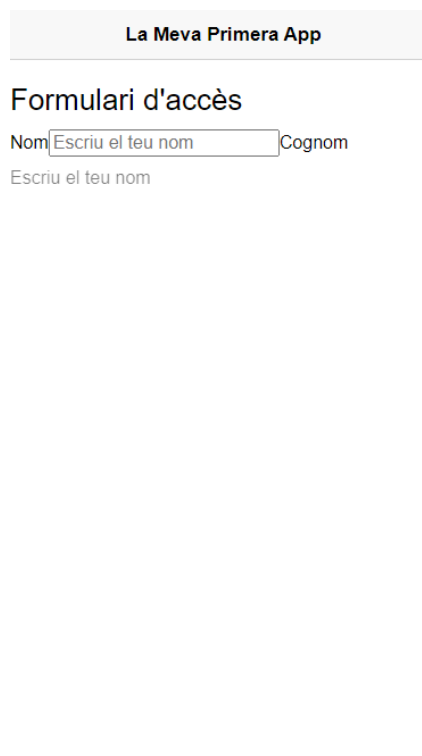
Per últim, l'atribut **placeholder** ens permet escriure un missatge que s'esborrarà de forma automàtica al camp on l'usuari ha d'escriure el seu nom.

A continuació demanarem el primer cognom de l'usuari utilitzant les etiquetes d'Ionic de la següent manera:

```
<ion-label for="surename">Cognom</ion-label>
<ion-input name="surename" placeholder="Escriu el teu cognom"></ion-input>
```

Hi ha poc que comentar sobre aquest codi ja que és molt semblant al de les etiquetes d'HTML. Anem a veure si visualment son igual.

Si obrim el projecte al navegador observarem que visualment son diferents. L'etiqueta **<input>** d'HTML crea un requadre al voltant del camp i la d'Ionic dibuixa una línia inferior.



Il·lustració 2. Aspecte inicial de formulari.

L'aspecte inicial del formulari no és molt atractiu, d'aquí un moment treballarem a la fulla d'estils per fer que sigui més atractiu visualment.

Però abans, afegim els elements d'ionic necessaris per a sol·licitar el segon cognom a l'usuari.

Un formulari no es pot enviar si no afegim un botó al final. Per tant, anem a afegir dos botons, un per a enviar el formulari i un altre per a esborrar totes les dades introduïdes.

El primer botó el crearem amb l'etiqueta **<button>** d'HTML. Just abans de la cloenda del formulari (**</form>**) escriurem el següent:

```
<button (click)="send()">Enviar</button>
```

L'atribut **(click)** permet la crida al mètode **send()** que crearem més endavant al fitxer de TypeScript.

El botó d'esborrar el crearem amb l'etiqueta **<ion-button>** d'Ionic de la següent manera:

```
<ion-button (click)="erase()">Esborrar</ion-button>
```

Igual que per les etiquetes anteriors, la codificació d'aquestes dues etiquetes és molt semblant. Anem a veure el seu aspecte visual.

La Meva Primera App

Formulari d'accès

Nom Cognom

Escriu el teu cognom

Segon Cognom

Escriu el teu cognom

Enviar

Il·lustració 3. Formulari amb botons.

Podem observar que visualment l'aspecte d'un botó i de l'altre és molt diferent.

6. Treballant amb els estils

En aquest apartat anem a treballar amb la fulla d'estils que està a **src/app/home/home.page.css**. Començarem per esborrar tot el contingut de la fulla ja que els elements als que fa referència els hem eliminat.

El primer element a modificar l'estil serà **<input>**. Per fer-ho escriurem:

```
input{
  border: 0;
  border-bottom: solid 0.1em;

  background-color: #eee;

  margin: 0 3vw;
}
```

El que hem fet ha sigut eliminar tot el contorn de l'element **<input>** i després fem que el contorn inferior (**border-bottom**) sigui sòlid i tingui una amplada de 0.1 cops la l'alçada de la lletra m (**em**).

A banda d'això fem que el color de fons de l'element sigui una mica gris per diferenciar-ho de la resta de la web i fem que l'element es separi horitzontalment el 3% de l'amplada de la pantalla per a que pugui respirar.

El següent pas que realitzarem serà que l'etiqueta Cognom passi a la línia inferior. Això es realitza de forma més senzilla afegint al fitxer HTML l'etiqueta **<ion-item>** abans de l'etiqueta **<ion-label>** i tancant-la després del tancament de **<ion-input>**. De la següent manera:

```
<ion-item>
  <ion-label for="surename">Cognom</ion-label>
  <ion-input name="surename" placeholder="Escriu el teu
cognom"></ion-input>
</ion-item>
```

D'aquesta forma hem fet que aquest elements es situïn a sota del Nom.

El següent pas serà donar format al botó creat amb HTML per que tingui més consistència visual. Per això afegirem el següent:

```
button {
  margin: 0;
  padding: 1.5vh 2vw;
  width: 100%;

  text-transform: uppercase;

  background-color: blue;

  color: white;
}
```

Pel botó enviar hem dit que tingui una amplada (**width**) del 100% del seu contenidor; així que hem de fer el mateix pel botó d'esborrar. Això ho podem fer des de la fulla d'estil o des de l'etiqueta **<ion-button>**. Si ho fem des de l'etiqueta, haurem d'afegir l'atribut **expand** de la següent manera:

```
<ion-button expand="block" (click)="erase()">Esborrar</ion-button>
```

Ara mateix, la forma visual de l'aplicació hauria de ser la següent.

La Meva Primera App

Formulari d'accès

Nom

Cognom

Segon Cognom

Il·lustració 4. Vista de l'aplicació.

Encara hem de corregir algunes coses com és la separació entre els diferents elements per que *respirin* una mica i no estigui tot tant junt. Per fer-ho. Anirem a la fulla d'estils, esborrarem la propietat **margin** d'**input** i escriurem:

```
input, ion-input, ion-item{
  margin: 1vh 2vw 2vh 3vw;
}

label, ion-label{
  margin: 1vh 2vw 2vh 3vw;
}
```

Introduint el **margin** amb els seus quatre components estem donant el marge segons el sentit de les agulles d'un rellotge començant pel marge superior, és a dir:

- Primer valor: marge superior.
- Segon valor: marge dret.
- Tercer valor: marge inferior.
- Quart valor: marge esquerra.

Ara ens manca donar un marge a l'encapçalament. En aquest cas el que farem serà centrar el text ja que l'element **<h1>** ocupa tot l'ample de pantalla per defecte. Així doncs escriurem:

```
h1{
  text-align: center;
}
```

Ara l'aspecte de la nostra aplicació ha de ser el següent:

La Meva Primera App

Formulari d'accès

Nom

Cognom

Segon Cognom

ENVIAR

Esborrar

Il·lustració 5. Vista final del formulari.

Els elements del formulari no estan alineats perfectament degut a que estem utilitzant etiquetes d'HTML sense format amb etiquetes d'Ionic que tenen un format per defecte. En un projecte normal sempre utilitzarem etiquetes d'Ionic o d'HTML. Això farà que sigui molt més senzill donar un format comú.

En aquests apunts s'utilitzen les dues per veure les diferències entre elles i el seu comportament amb els lectors de pantalla, cosa que es veurà més endavant.

7. Programant la lògica

Ara que tenim la vista ja creada procedirem a programar la lògica de l'aplicació. En aquest cas farem que el botó **Enviar** reculli la informació, ens la mostri per consola i esborri els camps i el botó **Esborrar** esborri els camps del formulari.

Així que comencem obrint el fitxer **src/app/home/home.page.ts**.

7.1. Definint les variables

El primer pas serà definir les variables. Per això, abans del **constructor()** definirem una variable per cadascun dels camps del formulari de la següent manera:

```
name: string;
surename1: string;
surename2: string;
```

En aquest cas, totes les variables són del tipus **string** (cadena de text). Amb les variables ja definides passarem a la creació dels mètodes.

7.2. Creant els mètodes

Com hem dit a l'inici haurem de crear dos mètodes a realitzar dues accions diferents. El primer mètode que crearem serà el d'esborrar el contingut del formulari.

7.2.1. Mètode *erase()*

Aquest mètode estarà associat al botó d'esborrar i per tant ha de tenir el mateix nom que el l'atribut (**click**) del botó esborrar. Aquest atribut es va definir com: **erase()**.

Així doncs, a sota del **constructor()** escriurem:

```
erase(){
  this.name = null;
  this.surename1 = null;
  this.surename2 = null;
}
```

El que estem fent és senzillament assignar el valor null a cadascuna de les variables quan l'usuari pitgi el botó esborrar. Al assignar qualsevol tipus de variable el valor **null** fa que la variable agafi el valor nul.

Amb aquest mètode creat, anem a definir el mètode **send()**.

7.2.2. Mètode *send()*

Aquest mètode ha d'agafar el valor, mostrar-ho a la consola i esborrar-ho. Per tant serà de la següent manera:

```
send(){
  console.log('El teu nom ', this.name);
  console.log('El teu primer cognom ', this.surename1);
  console.log('El teu segon cognom ', this.surename2);

  this.erase();
}
```

Amb **console.log()** estem fent que les variables es mostrin a la consola de la pàgina web. Per veure la consola hem d'activar les Eines de Desenvolupador des de el menú del nostre navegador.

Del mètode hem de parar especial atenció a la darrera línia, on es fa una crida al mètode **erase()**. Aquí podríem tornar a escriure el mateix codi que al mètode **erase()**, però fer una crida a aquest mètode ens estalvia línies de codi i queda més net i millor.

Si anem a l'aplicació i pitgem els botons veurem que aquests no funcionen. Per què?

7.3. Vinculant les variables

El problema que tenim és que les variables no estan vinculades amb la part visual de l'aplicació. Per fer això hem d'afegir el vincle a les etiquetes on introduïm les dades. Això es fa mitjançant l'atribut **[(ngModel)]** de la següent manera:

```
<input name="name" placeholder="Escriu el teu nom" [(ngModel)]="name">
...
<ion-input name="surename" placeholder="Escriu el teu cognom"
[(ngModel)]="surename1"></ion-input>
...
<ion-input name="surename2" placeholder="Escriu el teu cognom"
[(ngModel)]="surename2"></ion-input>
```


...

Ara si executem l'aplicació i pitgem els botons veurem que aquests realitzen les accions que esperem dels mateixos.

8. Afegint una nova vista

Ja tenim una aplicació funcional, però anem a afegir una segona vista que es mostrarà quan l'usuari premi el botó enviar.

Per crear una nova pàgina hem d'anar al terminal o la línia de comandament i escriure el següent comandament a dins de la carpeta del nostre projecte:

```
ionic g page pages/page2
```

Amb aquest comandament hem generat una segona pàgina a dins de la carpeta **pages**. Es recomana que tot el que creem s'emmagatzemi a dins de carpetes amb els noms del seu tipus. És a dir, les pàgines aniran a dins d'una carpeta **pages**, els serveis a dins d'una carpeta **services**, etcètera.

El següent pas serà enllaçar el nostre formulari amb aquesta segona pàgina.

9. Enllaçant pàgines

Per enllaçar les pàgines obrirem el fitxer **src/app/home/home.page.ts** i importarem el mòdul **NavController**.

```
import { NavController } from '@ionic/angular';
```

Un cop importat el mètode, l'injectarem a dins el **constructor()** de la següent manera:

```
constructor(private navCtrl: NavController) {}
```

Ara que tenim el mètode importat i inserit ja el podem utilitzar per navegar entre les diferents pàgines. Per fer-ho escriurem dins el mètode **send()** i abans del seu tancament:

```
this.navCtrl.navigateForward('/page2');
```

Ara si anem a la nostra aplicació i pitgem el botó **Enviar** veurem com es mostra la segona pàgina de l'aplicació.



Il·lustració 6. Segona vista de l'aplicació.

Ja tenim una nova vista, en blanc, així que anem a donar-li continguts. El que farem serà afegir una imatge i dues llistes amb enllaços a pàgines externes emprant etiquetes d'HTML. Comencem!

10. Afegint una imatge

Per afegir una primer imatge afegirem l'etiqueta `` a dins de l'etiqueta `<ion-content>` a l'arxiu `src/app/pages/page2/page2.page.html`.

```

```

A la propietat `src` introduïm l'adreça on tenim emmagatzemada la imatge que volem mostrar. Cada carpeta es separa per / (barra inclinada) i .. (dos punts) significa que em de pujar de carpeta (anar enrere).

La propietat `alt` fa que es mostri el text quan situem el focus a sobre la imatge i que els lectors de pantalla enuncïin aquest text per descriure la imatge.

11. Afegint llistes

Ara afegirem dos tipus de llistes, una no numerada i una altre numerada.

11.1. Llista no numerada

El llistat no numerat es crea amb una etiqueta `` de la següent manera:

```
<ul>  
  <li>Àfrica</li>  
  <li>Amèrica</li>  
</ul>
```

Més endavant afegirem enllaços a aquest llistat.

11.2. Llista numerada

Una llista numerada es genera amb l'etiqueta `` de forma molt similar al vist al punta anterior.

```
<ol>
  <li>Congo</li>
  <li>Canadà</li>
</ol>
```

12. Afegint enllaços

Per afegir un enllaç s'ha d'utilitzar l'etiqueta `<a>`, on especificarem l'adreça d'internet a la que volem navegar.

Això es fa de la següent manera:

```
<ul>
  <li><a href="https://www.tempsrecord.cat/albums/soundtrack/africa-
rutes-per-a-viatgers-intrepids/" title="Anar a rutes per
Àfrica">Àfrica</a></li>
  <li><a href="https://ca.wikiloc.com/rutes/senderisme/paisos-
baixos/limburg/america" title="Rutes per Amèrica">Amèrica</a></li>
</ul>
```

Es deixa com a exercici realitzar el mateix amb el segon llistat.

Com es volen utilitzar les etiquetes d'Ionic per veure les diferències visuals i en el comportament, afegirem un botó per anar a una tercera pàgina.

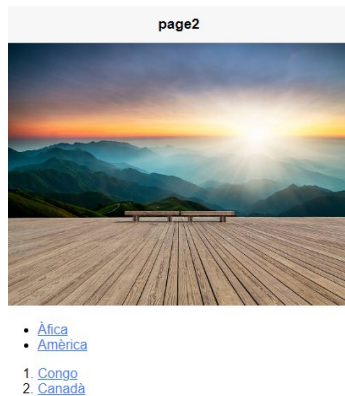
13. Afegint un botó

El botó que afegirem serà un botó que cridi directament a la següent pàgina sense cridar cap mètode. Això ho farem de la següent manera:

```
<ion-footer>
  <ion-button expand="block" routerLink="/page3"
routerDirection="forward">Següent</ion-button>
</ion-footer>
```

Amb l'etiqueta `<ion-footer>` fem que el botó aparegui a la part inferior de la pantalla.

Un cop realitzades les modificacions l'aspecte de l'aplicació hauria de ser semblant al següent.



Següent

Il·lustració 7. Aspecte final de la pàgina 2.

14. Pàgina Nova

El següent pas serà crear una tercera pàgina, cosa que es deixarà per vosaltres, on inserirem una imatge i un llistat amb les etiquetes d'Ionic.

14.1. Afegint una imatge

Per afegir una imatge emprant l'etiqueta **<ion-img>** realitzarem el següent:

```
<ion-img src="https://media.istockphoto.com/photos/winding-road-picture-id1173544006?s=612x612" alt="Carretera entre bosc"></ion-img>
```

En aquest cas, en lloc d'inserir una etiqueta emmagatzemada al nostre equip, hem inserit una imatge d'internet. El costat bo de fer-ho així és que no ocupem espai al dispositiu. Per contra, no controlem la imatge, és adir, si el servidor esborra la imatge o la canvia, per exemple, aquesta no es mostrarà.

14.2. Afegint un llistat

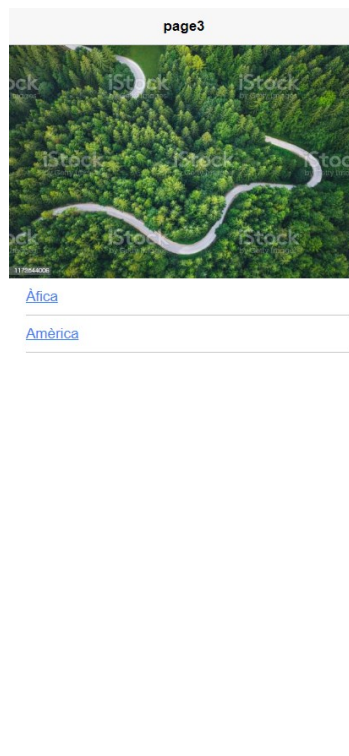
Per afegir un llistat emprant l'etiqueta **<ion-list>** ho farem d'aquesta forma:

```
<ion-list>
  <ion-item>
    <ion-label><a
href="https://www.tempsrecord.cat/albums/soundtrack/africa-rutes-per-a-
viatgers-intrepids/" title="Anar a rutes per Àfrica">Àfrica</a></ion-
label>
  </ion-item>

  <ion-item>
```

```
<a href="https://ca.wikiloc.com/rutes/senderisme/paisos-baixos/limburg/america" title="Anar a rutes per Amèrica">Amèrica</a>
</ion-item>
</ion-list>
```

Si ara mirem la nostre aplicació veurem que hi ha diferències visual entre un llistat i un altre. Quan utilitzem **<ion-list>** observem que apareixen línies horitzontals entre els elements de la llista.



Il·lustració 8. Tercera pàgina de l'aplicació.

Apèndix II: Lector de pantalla

1. Introducció

En aquest apèndix anem a veure les diferències de comportament, si les hi ha entre les etiquetes pures d'HTML i les etiquetes d'Ionic (**ion-**).

Per evitar que el lector de pantalles modifiqui l'idioma de la veu, en cas de que tinguem configurat que ho faci, hem de fer la següent modificació al fitxer **src/app/index.html**.

A la segona línia, on diu

```
<html lang="en">
```

Hem de modificar el contingut de la propietat **lang** per ca, si tenim una veu en català i la volem utilitzar, o es, si volem fer la revisió en castellà.

2. Primera pantalla

Quan utilitzem el lector de pantalla entre els elements de la primera pantalla observem que el comportament dels diferents elements és similar i podem accedir a tots els elements de la pantalla utilitzant el **tabulador**.

La única diferència destacable de comportament és que pel cas on utilitzem l'etiqueta **<ion-item>** el lector de pantalla ens llegeix sempre l'etiqueta **<label>/<ion-label>** associada però en els altres dos casos ens llegeix el text introduït en **placeholder**.

3. Segona i Tercer pantalla

En aquestes pantalles sí que hi ha diferència entre el comportament d'un element: la imatge. Quan utilitzem l'etiqueta d'HTML **** el lector de pantalla ens enuncia el text introduït a l'atribut **alt**; per contra, quan utilitzem l'etiqueta **<ion-img>** això no succeeix, el lector de pantalla no ens avisa que hi ha una imatge. Si la imatge que volem utilitzar és una imatge decorativa que no porti informació a la pantalla, llavors no tindrem cap problema en utilitzar una etiqueta o una altre.

Si volem utilitzar una imatge que porti informació a la pantalla llavors s'ha d'utilitzar l'etiqueta 'HTML **** amb el seu atribut **alt** o crear/inserir una etiqueta **** oculta visualment, però accessible pels lectors de pantalla, que contingui un text on es descriu la imatge.