Ionic

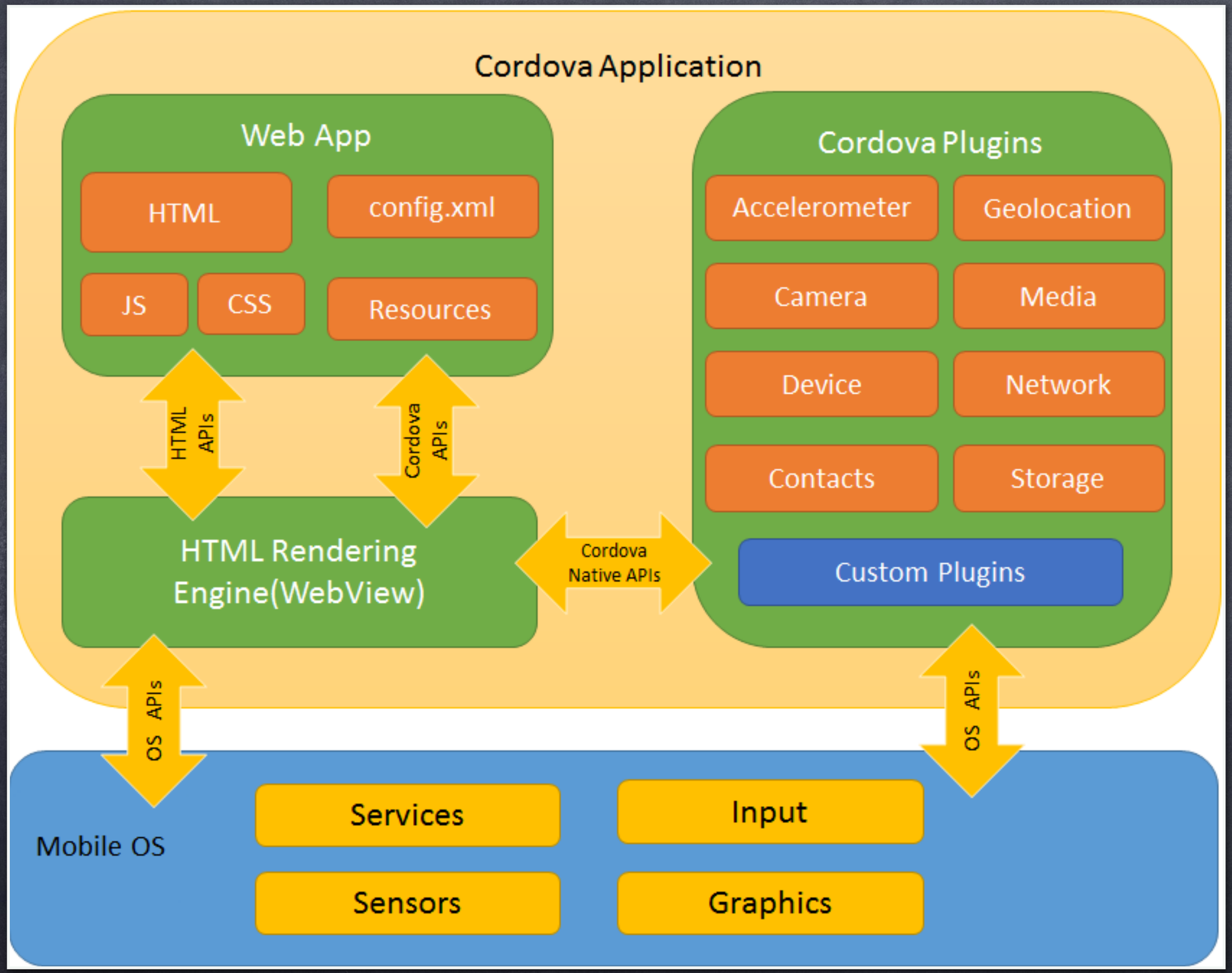
Présentation

Ionic est un framework HTML5 qui permet de développer des applications hybrides pour les appareils mobiles.

- Composants CSS et javascript pour simuler le comportement et l'apparence des applications natives
- Collection d'icônes
- Commandes CLI (en partie reprise des commandes Cordova)

Architecture

- Basé sur le framework Angular 2
- Utilise TypeScript (dérivé de Javascript)
- Utilise Sass (pré-processeur CSS)
- Offre une interface vers les plugins Cordova (Ionic Native)


Cordova Application
Web App
HTML
config.xml
JS
CSS
Resources
HTML APIs
Cordova APIs
HTML Rendering Engine(WebView)
Cordova Native APIs
Cordova Plugins
Accelerometer
Geolocation
Camera
Media
Device
Network
Contacts
Storage
Custom Plugins
OS APIs
OS APIs
Mobile OS
Services
Input
Sensors
Graphics

Installation

Installer Git

https://git-scm.com/downloads

3. bash

```
pc12:~ seb$ git --version
git version 2.6.4
pc12:~ seb$
```

Installer Node-js

https://nodejs.org/en/download/

3. bash

```
pc12:~ seb$ node --version
v4.2.2
pc12:~ seb$ npm -- version
{ npm: '2.14.7',
  ares: '1.10.1-DEV',
  http_parser: '2.5.0',
  icu: '56.1',
  modules: '46',
  node: '4.2.2',
  openssl: '1.0.2d',
  uv: '1.7.5',
  v8: '4.5.103.35',
  zlib: '1.2.8' }
pc12:~ seb$
```

Installer les outils

```
npm install -g ionic/cli
```

Avec Mac OS, il faut avoir les droits en écriture sur les dossiers de npm pour installer globalement. Le plus simple est de changer le propriétaire de ces dossier en exécutant les lignes de commandes suivantes :

```
sudo chown -R $(whoami) /usr/local/{lib/node_modules,bin,share}
sudo chown -R $(whoami) ~/.npm ~/.npmrc
```

Ligne de commande

myApp — bash — 109×24

bash | bash | bash

```
MacBook-Pro-de-Sebastien-6:myApp seb$ ionic --list

 _              _
(_)            (_)
 _  ___  _ __   _  ___
| |/ _ \| '_ \| |/ __|
| | (_) | | | | | (__
|_|\___/|_| |_|_|\___|  CLI v2.1.17

Usage: ionic task args


========================
Available tasks:
(use --help or -h for more info)


   start  ..........  Starts a new Ionic project in the specified PATH
   serve  ..........  Start a local development server for app dev/testing
   setup  ..........  Configure the project with a build tool (beta)
   generate  .......  Generate pages and components
   platform  .......  Add platform target for building an Ionic app
   run  ............  Run an Ionic project on a connected device
   emulate  ........  Emulate an Ionic project on a simulator or emulator
   build  ..........  Build (prepare + compile) an Ionic project for a given platform.
```

Créer un projet

```
ionic start chemin-du-projet template
```

modèles intégrés

```
blank
sidemenu
tabs
```

Tester un projet dans un navigateur

```
ionic serve
```

```
-c : affiche les logs de l'application dans la console
-s : affiche les logs du serveur dans la console
-p : détermine le port (8100 par défaut)
-w : détermine le navigateur à lancer (firefox, chrome, safari)
-l : simule la taille d'un smartphone
-b : désactive le lancement automatique du navigateur
```

Générateur

```
//Génère un élément
ionic g type NomDeLaClasse

//Génère une page
ionic g page MaPage
```

component	module
page	class
service	guard
pipe	interface
directive	enum

Debug sur un périphérique

Etapes

- Lancer le projet sur un émulateur ou un périphérique physique
- Activer le debug USB sur le périphérique
- Lancer Chrome en mode inspection chrome://inspect

Inspect with Chrome Develope

chro...

Applications | phpStorm | m2i-dev | web local | misc | » Autres favoris

DevTools

- Devices
- Pages
- Extensions
- Apps
- Shared workers
- Service workers
- Other

Devices

- [x] Discover USB devices — Port forwarding...
- [x] Discover network targets — Configure...

Remote Target #LOCALHOST

Custom Phone - 7.0.0 - API 24 - 768x1280 #192.168.58.101:5555

WebView in com.ionicframework.notebookapp892141 (51.0.2704.91)

- [ ] Ionic App file:///android_asset/www/index.html
  inspect

Developer Tools - file:///android_asset/www/index.html

file:///android_asset/www/index.html

Elements Console Sources Network Timeline Profiles Resources Security Audits

Consider disabling Chrome Data Saver while debugging. more never show

Ionic Blank

Un titre

AJOUTER UNE NOTE

Sources Content scripts Snippets

/AppModule/Note…r.ngfactory.js

- top
  - file://
    - android_asset/www
      - build
      - plugins
      - index.html
      - cordova.js
  - (no domain)

```
(function(jit_NoteFormPage0,jit_setBindingDebugI
/**/) {
function Wrapper_NoteFormPage(p0,p1,p2) {
  var self = this;
  self._changed = false;
  self._changes = {};
  self.context = new jit_NoteFormPage0(p0,p1,p2)
}
Wrapper_NoteFormPage.prototype.ngOnDetach = func
};
Wrapper_NoteFormPage.prototype.ngOnDestroy = fun
};
Wrapper_NoteFormPage.prototype.ngDoCheck = funct
  var self = this;
  var changed = self._changed;
  self._changed = false;
  if (!throwOnChange) { if (changed) {
    jit_setBindingDebugInfoForChanges1(view.rend
    self._changes = {};
  } }
  return changed;
};
Wrapper_NoteFormPage.prototype.checkHost = funct
};
Wrapper_NoteFormPage.prototype.handleEvent = fun
  var self = this;
  var result = true;
  return result;
};
Wrapper_NoteFormPage.prototype.subscribe = funct
  var self = this;
  self._eventHandler = _eventHandler;
};
return Wrapper_NoteFormPage
})
```

{} Line 1, Column 1

Pause On Caught Exceptions

Watch

```
running: <not available>
total: <not available>
input: <not available>
n: <not available>
items: <not available>
```

Call Stack

Not Paused

Scope

Not Paused

Breakpoints

No Breakpoints

DOM Breakpoints

XHR Breakpoints

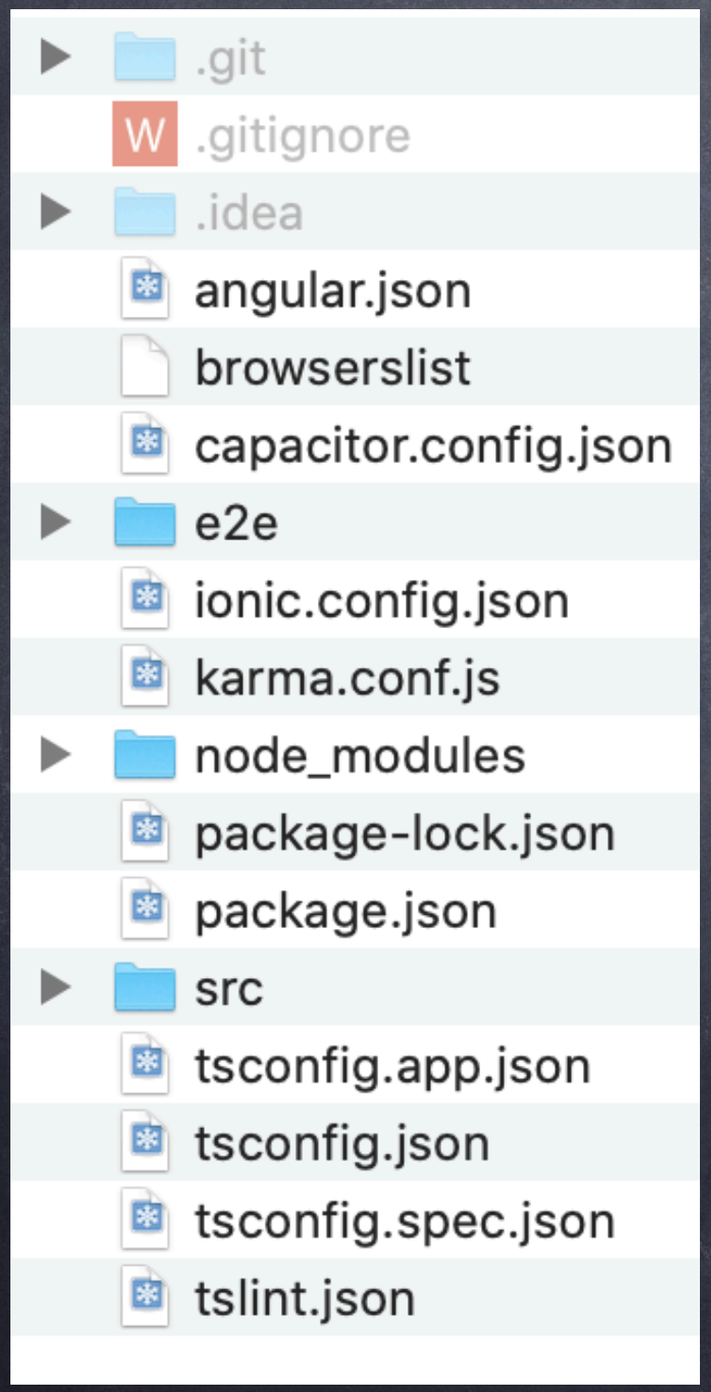
Event Listener Breakpoints

Event Listeners

Console

Architecture

Quelques principes



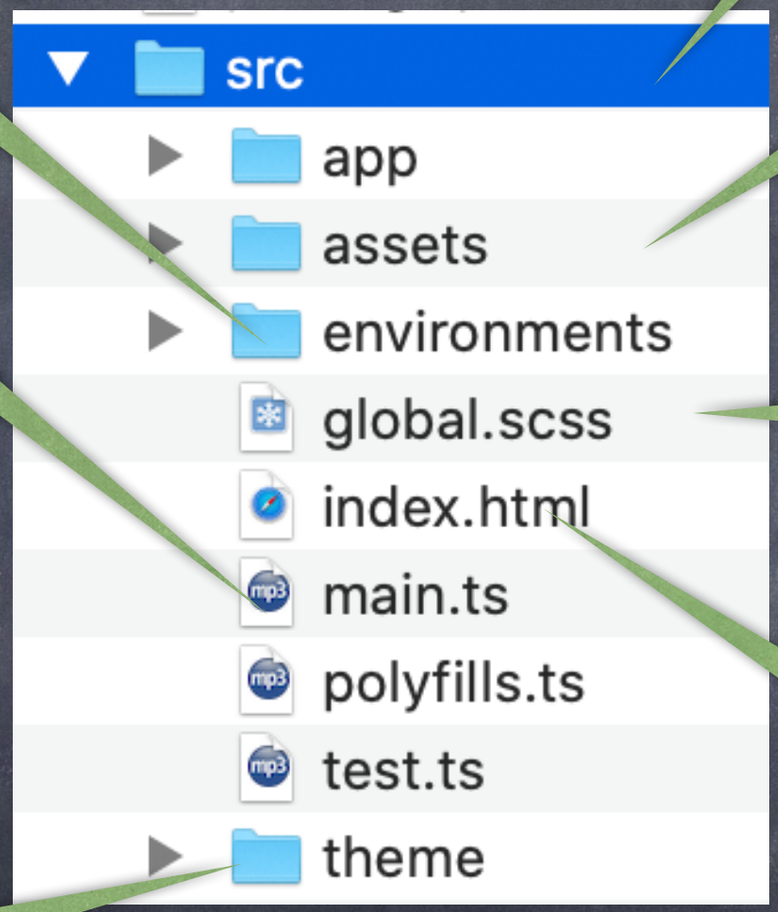
- On ne travaille que dans le dossier src
- Le code et les ressources sont copiés
  et/ou transpilés en mémoire vive pour tester l'application

Variables d'environnement

Lanceur de l'application

Sources de l'application

Resources (images, sons, etc)



Styles qui s'appliquent à toutes les pages

Point d'entrée de l'application

Thèmes (couleurs, variables CSS4)

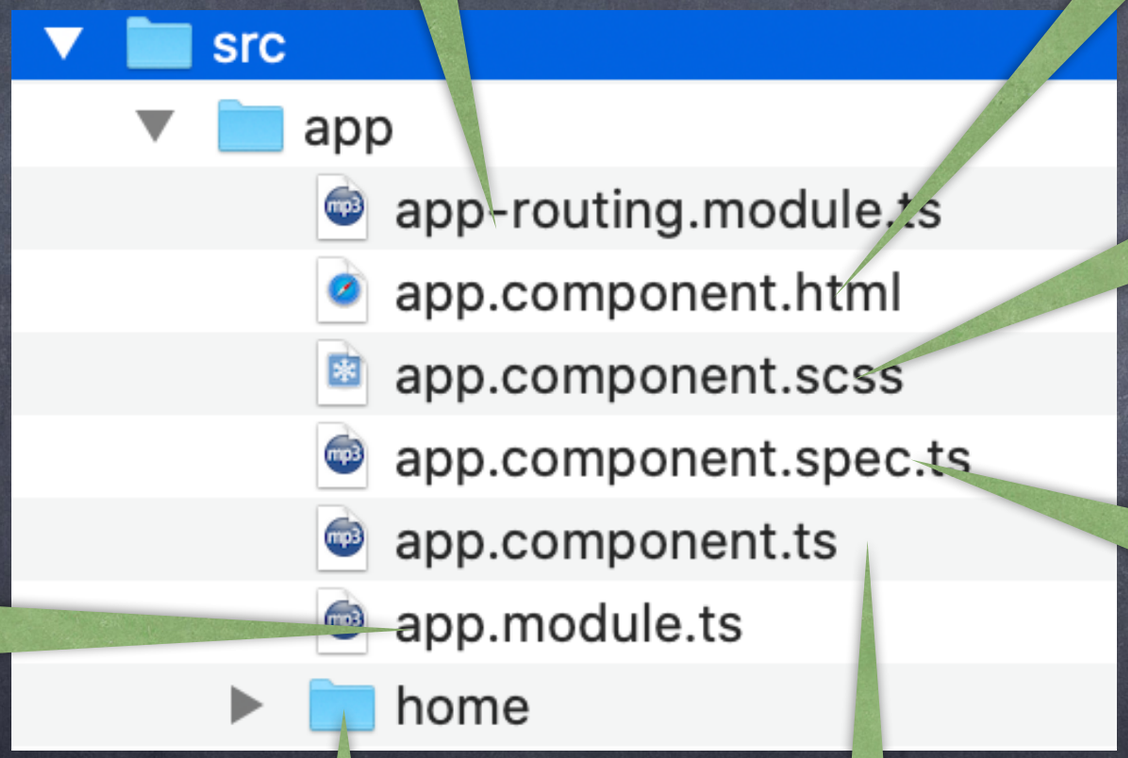
L'application

Définition des routes

Gabarit html de l'application

Styles de l'application

Paramètrage et définition des dépendances



Tests unitaires

Un écran

Classe principale

app.component.ts

```
import { Component } from '@angular/core';

import { Platform } from '@ionic/angular';
import { SplashScreen } from '@ionic-native/splash-screen/ngx';
import { StatusBar } from '@ionic-native/status-bar/ngx';
```

Imports

```
@Component({
  selector: 'app-root',
  templateUrl: 'app.component.html',
  styleUrls: ['app.component.scss']
})
```

Décorateur

```
export class AppComponent {
  constructor(
    private platform: Platform,
    private splashScreen: SplashScreen,
    private statusBar: StatusBar
  ) {
    this.initializeApp();
  }

  initializeApp() {
    this.platform.ready().then(() => {
      this.statusBar.styleDefault();
      this.splashScreen.hide();
    });
  }
}
```

Classe

app.module.ts

Imports

```
import { NgModule } from '@angular/core';
import { BrowserModule } from '@angular/platform-browser';
import { RouteReuseStrategy } from '@angular/router';

import { IonicModule, IonicRouteStrategy } from '@ionic/angular';
import { SplashScreen } from '@ionic-native/splash-screen/ngx';
import { StatusBar } from '@ionic-native/status-bar/ngx';

import { AppComponent } from './app.component';
import { AppRoutingModule } from './app-routing.module';
```

Injection de dépendance

```
@NgModule({
  declarations: [AppComponent],
  entryComponents: [],
  imports: [BrowserModule, IonicModule.forRoot(), AppRoutingModule],
  providers: [
    StatusBar,
    SplashScreen,
    { provide: RouteReuseStrategy, useClass: IonicRouteStrategy }
  ],
  bootstrap: [AppComponent]
})
export class AppModule {}
```

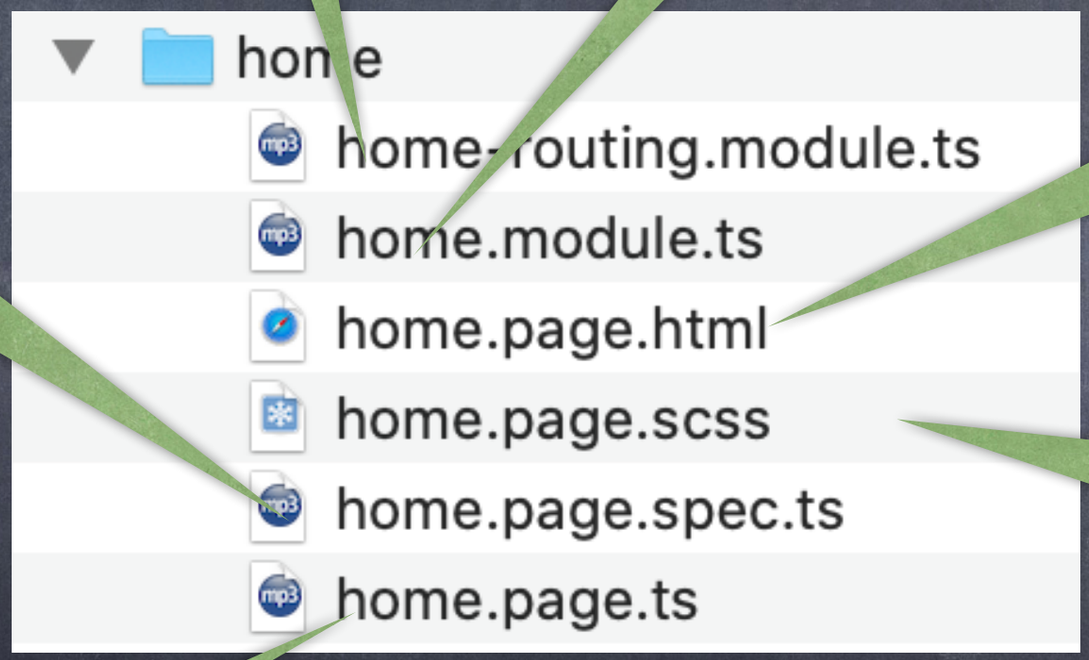
Un écran

Routage de la page

Dépendances de la page

Interface graphique

Tests unitaires



Styles

Code de la page

home.ts

```
//Importation des dépendance de cet écran
import { Component } from '@angular/core';
import { NavController } from 'ionic-angular';

//Décorateur, un écran est un type de Component
@Component({
  //Inutile pour un écran
  selector: 'page-home',
  //Définition de la vue à utiliser
  templateUrl: 'home.html'
})

//Un écran est une classe
export class HomePage {

  constructor(public navCtrl: NavController) {

  }

}
```

home.html

```
<ion-header>
  <ion-navbar>
    <ion-title>
      Ionic Blank
    </ion-title>
  </ion-navbar>
</ion-header>

<ion-content padding>
  The world is your oyster.
  <p>
    If you get lost, the
    <a href="http://ionicframework.com/docs/v2">docs</a>
    will be your guide.
  </p>
</ion-content>
```

Interface graphique

Page

```
<ion-header>
</ion-header>

<ion-content padding>
 contenu de la page
</ion-content>

<ion-footer>
</ion-footer>
```

élément
balise perso

directive
attribut perso

En-tête

barre fixe en haut de l'écran

```
<ion-header>
  <ion-navbar>
    <ion-title>
      Mon app
    </ion-title>
  </ion-navbar>
</ion-header>
```

Pied

barre fixe en bas de l'écran

```
<ion-footer>
   <ion-title>
     Mon app
   </ion-title>
</ion-footer>
```

Icônes

police de caractères contenant des pictogrammes

```
<ion-icon name="search"></ion-icon>
```

Boutons

color	couleur du bouton
fill	clear | outline | solid
size	large | small | default
shape	round
expand	Full | block
block	bouton sur une ligne
mode	ios | md

```
<ion-button
  color="danger"
  shape="round"
  expand=« block"
>
  OK
</ion-button>
```

Bouton avec texte et icône

```
<ion-button
  color="danger"
>
  OK
  <ion-icon name="star">
  </ion-icon>
</ion-button>
```

Bouton avec icône uniquement

```
<ion-button
 color=« danger">

 <ion-icon name="star"
  slot="icon-only">
 </ion-icon>

</ion-button>
```

Groupes de boutons

```
<ion-buttons slot="start">
    <ion-button color="danger">
        <ion-icon slot="icon-only"
                  name="star">
       </ion-icon>
    </ion-button>
</ion-buttons>
```

Les listes

List

Liste d'éléments avec défilement

```
<ion-list>
  <ion-list-header>fruits</ion-list-header>
  <ion-item>pommes</ion-item>
  <ion-item>poires</ion-item>
</ion-list>
```

Elément multi ligne

Il faut encadrer le contenu avec une balise ion-label

```
<ion-list>
 <ion-item>
  <ion-label>
    <h3>Tycho Brahé</h3>
    <p>Un astronome aux moustaches
    audacieuses</p>
  </ion-label>
 </ion-item>
</ion-list>
```

List dividers

Liste avec regroupements

```
<ion-item-group>
  <ion-item-divider>A</ion-item-divider>
  <ion-item>abricots</ion-item>
  <ion-item>ananas</ion-item>
</ion-item-group>

<ion-item-group>
  <ion-item-divider>B</ion-item-divider>
  <ion-item>bananes</ion-item>
  <ion-item>brugnons</ion-item>

</ion-item-group>
```

Liste avec icône

la position est déterminée par la directive slot (start | end)

```
<ion-list>
  <ion-item>
    pommes
    <ion-icon name="apple" slot="end">
    </ion-icon>
  </ion-item>

</ion-list>
```

Liste avec image

la position est déterminée par la directive slot (start | end)

```
<ion-list>
  <ion-item>
    pommes
    <ion-avatar slot=« start">
        <img src="/assets/img/apple.png">
    </ion-avatar>
  </ion-item>
</ion-list>
```

Liste avec image

la position est déterminée par la directive slot (start | end)

```
<ion-list>
  <ion-item>
    pommes
    <ion-thumbnail slot="end">
        <img src="/assets/img/apple.png">
    </ion-thumbnail>
  </ion-item>
</ion-list>
```

Liste avec bouton

```
<ion-list>
  <ion-item>
    pommes
    <ion-button color="danger" slot="end">
      Supprimer
    </ion-button>
  </ion-item>
</ion-list>
```

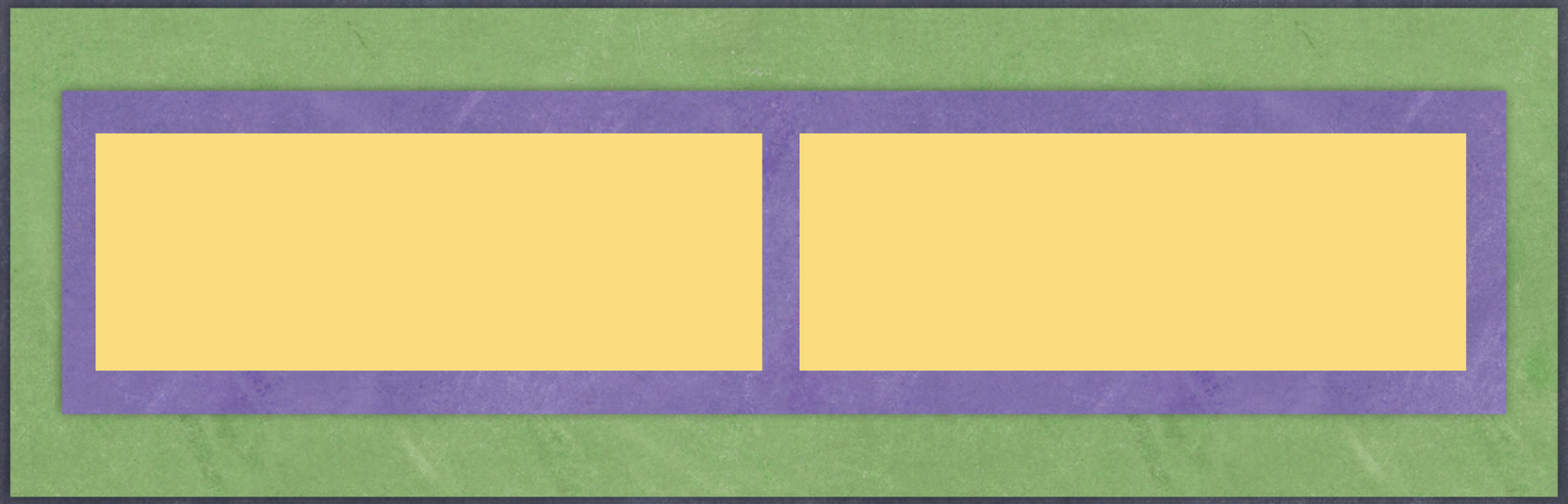
Swipe to reveal

```
<ion-list>
  <ion-item-sliding>
    <ion-item>pommes</ion-item>

    <ion-item-options side="left">
      <ion-item-option>
        <button ion-button>delete</button>
      </ion-item-option>
    </ion-item-options>

  </ion-item-sliding>

</ion-list>
```

Grilles

Principes

Un composant <ion-grid> définit une grille de mise en page, cette grille contient des lignes et des colonnes



Dimensions

Ionic utilise une grille de 12 colonnes
L'attribut size permet de spécifier
la taille de chaque colonne

Exemple

Une barre de boutons
L'espace est distribué entre les trois colonnes

```
<ion-grid>
  <ion-row>
    <ion-col>
      <ion-button expand="block">bt 1</ion-button>
    </ion-col>
    <ion-col>
      <ion-button expand="block">bt 2</ion-button>
    </ion-col>
    <ion-col>
      <ion-button expand="block">bt 3</ion-button>
    </ion-col>
  </ion-row>
</ion-grid>
```

Exemple

La deuxième colonne prend 50% de l'espace
les deux autres se partagent le reste

```
<ion-grid>
  <ion-row>
    <ion-col>
      <ion-button expand="block">bt 1</ion-button>
    </ion-col>
    <ion-col size="6">
      <ion-button expand="block">bt 2</ion-button>
    </ion-col>
    <ion-col>
      <ion-button expand="block">bt 3</ion-button>
    </ion-col>
  </ion-row>
</ion-grid>
```

Grille

La dernière colonne est trop grande,
elle passe à la ligne

```
<ion-row>
    <ion-col size="4">
        <ion-button expand="block">bt 1</ion-button>
    </ion-col>
    <ion-col size="4">
        <ion-button expand="block">bt 2</ion-button>
    </ion-col>
    <ion-col size="8">
        <ion-button expand="block">bt 3</ion-button>
    </ion-col>
</ion-row>
```

Formulaires

Input

Contrôle de saisie de texte monoligne, soit les types suivant :

text, password, email, number, search, tel, url

```
<ion-list>
  <ion-item>
    <ion-label>Nom</ion-label>
    <ion-input type="text"></ion-input>
  </ion-item>
</ion-list>
```

Pour le texte multiligne on utilisera <ion-textarea>

Label

fixed	le label apparait sur le côté gauche du contrôle
floating	le label apparait à côté du contrôle lorsque celui-ci est vide. Il se placera au dessus si le contrôle a le focus ou n'est pas vide
stacked	le label se place toujours au dessus du contrôle

```
<ion-label
   position="stacked">
  Nom
</ion-label>
<ion-input>
</ion-input>
```

Select

Une combo box, fonctionne comme la balise select du langage html, mais affiche les options dans une autre fenêtre.

```
<ion-item>
  <ion-label>profession</ion-label>
  <ion-select>
    <ion-select-option>Développeur</ion-option>
    <ion-select-option>Graphiste</ion-option>
    <ion-select-option>Chef de projet</ion-option>
    <ion-select-option selected >Architecte</ion-
option>
  </ion-select>
</ion-item>
```

Select à choix multiple

L'attribut muliple permet les choix multiples au sein de la liste.

```
<ion-item>
  <ion-label>Langages</ion-label>
  <ion-select multiple="true">
    <ion-select-option>Java</ion-select-option>
    <ion-select-option>PHP</ion-select-option>
    <ion-select-option>Python</ion-select-option>
    <ion-select-option selected >C++
   </ion-select-option>
  </ion-select>
</ion-item>
```

Libellé des boutons

Les attributs okText et cancelText définissent les libellés de boutons

```
<ion-item>
  <ion-label>Langages</ion-label>
  <ion-select okText="Valider"
      cancelText="Annuler">
    <ion-select-option>Java</ion-select-option>
    <ion-select-option>PHP</ion-select-option>
    <ion-select-option>Python</ion-select-option>
    <ion-select-option selected >C++</ion-select-
option>
  </ion-select>
</ion-item>
```

Boutons radio

Les éléments de type ion-radio ont un comportement de bascule s'ils sont enfants d'un élément ion-radio-group

```
<ion-list>
  <ion-radio-group>
     <ion-list-header>Sexe</ion-list-header>

    <ion-item>
      <ion-label>Femme</ion-label>
      <ion-radio value=« F" slot="start"></ion-radio>
    </ion-item>

    <ion-item>
      <ion-label>Homme</ion-label>
      <ion-radio value=« M" slot="start"></ion-radio>
    </ion-item>
  </ion-radio-group>
</ion-list>
```

Case à cocher

```
<ion-item>
   <ion-label>Java</ion-label>
   <ion-checkbox slot="start"></ion-checkbox>
</ion-item>
<ion-item>
   <ion-label>PHP</ion-label>
   <ion-checkbox slot="start"></ion-checkbox>
</ion-item>
```

Range

```
<ion-list>
 <ion-item>
    <ion-range min="20" max="80" step="2">

      <ion-icon small range-left name="sunny">
      </ion-icon>

      <ion-icon range-right name="sunny">
      </ion-icon>

    </ion-range>
  </ion-item>
<ion-list>
```

Attributs de range

min, max	valeur minimale et maximale
step	pas de l'incrémentation. si step est différent de 1, l'attribut snaps(bool) permet de limiter le déplacement aux incréments
dualKnobs	booléen, affiche deux pointeurs et retourne un objet possédant deux propriétés lower et upper
pin	booléen, affiche la valeur correspondant au curseur lors du clique

Toggle

Un bouton qui retourne true ou false
et est plus facile à cliquer qu'une case à cocher

```
<ion-list>
  <ion-item>
    <ion-label>remember me</ion-label>
    <ion-toggle slot=« end"></ion-toggle>
  </ion-item>
</ion-list>
```

Styles

Où placer le code css ?

- Dans le fichier global.scss
- Dans le fichier variables.scss
- Dans le fichier scss de chaque page

Les sélecteurs

- Classes
- Balises (ion-something comprises)

Les css utilities

ion-text-[align]	Alignement du texte (left | right | start | end | center | justify)
ion-hide	Masque un élément
ion-padding	Padding de 16 px, possibilité de préciser la position (ex ion-padding-left)
ion-margin	Marge de 16 px possibilité de préciser la position (ex ion-margin-top)
ion-no-border	Désactive les bordures sur les éléments ion-footer et ion-header

Les positions

top	en haut
bottom	en bas
start	à gauche sur les systèmes configurés pour une lecture de gauche à droite
end	à droite sur les systèmes configurés pour une lecture de gauche à droite
vertical	en haut et en bas
horizontal	à droite et à gauche

Les variables

Ces variables s'appliquent à l'ensemble de l'application

Les sélecteurs indiquent la portée des variables

```
/* portée globale */
:root {
  --ion-background-color: #ff3700;
}

/* portée limitée à iOS */
.ios {
  --ion-text-color: #000;
}

/* portée limitée à Android */
.md {
  --ion-text-color: #222;
}
```

variables.scss

Variables pour les components

Chaque composant dispose d'une série de variables qui déterminent son apparence (voir la doc)

```
// variables pour les ion-button
ion-button {
  --border-radius: 20px;
}

// ion-button uniquement sur Android
.md {
  ion-button {
    --background-hover: #FFFF00;
  }
}
```

variables.scss

Variables perso

On peut définir et utiliser une variable dans n'importe quel fichier scss, cependant la définition est souvent centralisée dans variables.scss

```
// Uniquement sur Android
.md {
  --big-font: 28px;
}

// Uniquement sur Android
.ios {
  --big-font: 25px;
}

// utilisation de la variable
ion-button {
  font-size: var(--big-font);
}
```

variables.scss

Les couleurs

variables.scss contient une liste de couleurs par défaut, le générateur permet de les modifier

```
:root {
  --ion-color-primary: #3880ff;
  --ion-color-primary-rgb: 56,128,255;
  --ion-color-primary-contrast: #ffffff;
  --ion-color-primary-contrast-rgb: 255,255,255;
  --ion-color-primary-shade: #3171e0;
  --ion-color-primary-tint: #4c8dff;

   …

}
```

variables.scss

Nouvelle couleur

```
:root {
  --ion-color-green: #008D80;
  --ion-color-green-rgb: 0, 141, 128;
  --ion-color-green-contrast: #ffffff;
  --ion-color-green-contrast-rgb: 255, 255, 255;
  --ion-color-green-shade: #007c71;
  --ion-color-green-tint: #1a988d;
}

/* En dehors de :root */
.ion-color-green {
  --ion-color-base: var(--ion-color-green) !important;
  --ion-color-base-rgb: var(--ion-color-green-rgb) !important;
  --ion-color-contrast: var(--ion-color-green-contrast) !important;
  --ion-color-contrast-rgb: var(--ion-green-gold-contrast-rgb) !important;
  --ion-color-shade: var(--ion-color-green-shade) !important;
  --ion-color-tint: var(--ion-color-green-tint) !important;
}
```

variables.scss

Angular

Liaison du modèle vers la vue

home.ts

```
export class HomePage {
  name = "Seb";

  constructor() {}

}
```

home.html

```
<ion-content class="ion-padding">
 <h1>Hello {{name}}</h1>
</ion-content>
```

Liaison avec les attributs

home.ts

```
export class HomePage {
  name = "Seb";
  visible = true;

  constructor() {}

}
```

home.html

```
<ion-content padding>
 <h1 [hidden]="!visible">
    Hello {{name}}
  </h1>
</ion-content>
```

Liaison bidirectionnelle du modèle vers la vue

home.ts

```
export class HomePage {
  name = "Seb";

  constructor() {}

}
```

home.html

```
<ion-content>
  <ion-input
  [(ngModel)]="name"
  >
  </ion-input>

  <h1>{{name}}</h1>
</ion-content>
```

Gestion des évènements

home.ts

```
export class HomePage {
  name = "Seb";

  constructor() {}

  doClick(){
    console.log("click")
  }

}
```

home.html

```
<ion-content>
  <ion-button
    (click)="doClick()">
    Cliquez moi
  </ion-button>
</ion-content>
```

Transmission de l'évènement

home.ts

```
export class HomePage {
  name = "Seb";

  constructor() {}

  doClick(evt){
    console.log(evt);
  }

}
```

home.html

```
<ion-content padding>
  <ion-button
    (click)="doClick($event)">
    Cliquez moi
  </ion-button>
</ion-content>
```

```
▶ MouseEvent {isTrusted: true, screenX: 567, screenY: 297, clientX: 74, clientY: 98…}
```

NgIf

home.ts

```
export class HomePage {
  name = "Seb";

  constructor() {}
  showMessage=false;

  doClick(){
    this.showMessage = true;
  }

}
```

home.html

```
<ion-content>
  <ion-button
    (click)="doClick()">
    Afficher le message
  </ion-button>

  <p *ngIf="showMessage">
    Hello
  </p>


</ion-content>
```

selon un expression booléenne
bascule entre un display block et none

Ngfor

home.ts

```
export class HomePage {
  name = "Seb";

  constructor() {}
  fruits = ["pommes","poires",
            "abricots"];
}
```

home.html

```
<ion-content>
  <ion-list>
    <ion-item
      *ngFor="let item of fruits">

      {{item}}

    </ion-item>
  </ion-list>
</ion-content>
```

boucle pour chaque élément d'une collection

Variables d'itération

index	index de l'itération (commence par 0)
first	booléen
last	booléen
even	booléen
odd	booléen

home.html

```
<ion-content>
  <ion-list>
    <ion-item
      *ngFor="let item of fruits;
      let pos = index">
      {{pos +1}} - {{item}}
    <ion-item>
  </ion-list>
</ion-content>
```

NgClass

home.ts

```
export class HomePage {
  name = "Seb";

  constructor() {}
  className="open";

  doClick(){
    this.className = "close";
  }

}
```

home.html

```
<ion-content>
  <ion-button
    (click)="doClick()">
    Changer la classe
  </ion-button>

  <p *ngClass="className">
    Hello
  </p>


</ion-content>
```

affecte un classe css à l'élément

NgClass

home.ts

```
export class HomePage {
  name = "Seb";

  constructor() {}
  type="all";

  filter(choice){
    this.type = choice;
  }

}
```

home.html

```
<ion-content padding>
<ion-button [ngClass]="{active: type =='all'}"
  (click)="filter('all')">all</ion-button>

<ion-button [ngClass]="{active: type =='red'}"
  (click)="filter('red')">red</ion-button>

<ion-button [ngClass]="{active: type
=='blue'}"
  (click)="filter('blue')">blue</ion-button>

</ion-content>
```

affectation de la la classe css "active"
en fonction d'une expression

Pipes

- Filtres appliqués sur l'affichage d'une expression dans une vues

Exemple

home.ts

```
export class HomePage {
  name = "Seb";
  constructor() {}

}
```

home.html

```
<ion-content padding>
 <h1>
    Hello {{name | uppercase}}
  </h1>
</ion-content>
```

Avec des paramètres

home.ts

```
export class HomePage {
  now = new Date();
  constructor() {}

}
```

home.html

```
<ion-content padding>
 <h1>
    {{now | date:'d/M/yy'}}
  </h1>
</ion-content>
```

Liste des pipes

Pipe	Description
lowercase	conversion en minuscule
uppercase	conversion en majuscule
number	affichage des nombres
percent	affichage de pourcentages
currency	affichage de monétaires
date	affichage de date
json	affichage de données json
slice	extrait une partie d'une chaîne ou d'un tableau

Date

format	Description
d ou D	jour
M ou MM	mois
y ou yy	année
EEE ou EEEE	jour de la semaine
j ou jj	heure sur 24h
h ou hh	heure sur 12h
m ou mm	minutes
s ou ss	secondes
z	fuseau horaire

```
<ion-content padding>
 <h1>
    {{now | date:'d/M/yy'}}
  </h1>
</ion-content>
```

Number

```
<ion-content padding>
 <h1>
  {{15.568 | number:'2.1-2'}}
 </h1>
</ion-content>
```

nombre de chiffres avant la virgule

nombre minimum de chiffres après la virgule

nombre maximum de chiffres après la virgule

Percent

```
<ion-content padding>
 <h1>
  {{.56 | percent:'2.1-2'}}
 </h1>
</ion-content>
```

nombre de chiffres avant la virgule

nombre minimum de chiffres après la virgule

nombre maximum de chiffres après la virgule

Currency

monnaie

```
<ion-content padding>
 <h1>
  {{.56 | currency:’EUR’:true:'2.1-2'}}
 </h1>
</ion-content>
```

affichage du symbole monétaire

format du nombre

Slice

```
<ion-content padding>
 <h1>
  {{'bonjour' | slice: '0:3'}}
 </h1>
</ion-content>
```

position de début

position de fin (non inclus)

Extrait les 3 premiers caractères de la chaîne (de la position 0 à la position 3 non incluse)

Combinaison des pipes

```
<ion-content padding>
 <h1>
   {{'bonjour' | slice: '0:3' | uppercase}}
  </h1>
</ion-content>
```

Formulaires avec Angular

Template driven forms

Principes

1. Un formulaire html classique
2. Un id à la sauce Angular
3. Envoi du formulaire avec l'événement onSubmit qui transmet le formulaire hydraté
4. Les champs ont un directive ngModel sans valeur
5. D'autres directives permettent de spécifier les règles de validation
6. Les champs ont un attribut name (comme en html)
7. Le formulaire est envoyé grâce à un bouton de type submit

```
<form #myForm="ngForm" (ngSubmit)="onSubmit(myForm)">
  <ion-list>
    <ion-item>
      <ion-label position="floating">email</ion-label>
      <ion-input type="text" ngModel required email name="email" >
      </ion-input>
    </ion-item>
    <ion-item>
      <ion-label position="floating">Mot de passe</ion-label>
      <ion-input type="password" ngModel required
                 minlength="6" maxlength="12" name="password">
      </ion-input>
    </ion-item>
  </ion-list>
  <ion-button type="submit" expand="block" color="primary">
    Connexion
  </ion-button>
</form>
```

Intérêt

- La validation est automatique et les champs en erreur apparaissent en rouge
- On récupère un objet de type NgForm qui contient des informations sur la saisie et son contexte

Le traitement

```
onSubmit(form: NgForm){
  console.log(form);
}
```

```
▼ NgForm                                                    E onSubmit — home.page.ts:14
  ▶ O __ngContext__: LComponentView_HomePage [<app-home class="ion-page">, TView, 147, LR…
  ▶ O _directives: [NgModel, NgModel] (2)
  ▼ O form: FormGroup
    ▶ f _onCollectionChange: function()
    ▶ O _onDisabledChange: [] (0)
      N asyncValidator: null
    ▶ O controls: {email: FormControl, password: FormControl}
      N errors: null
      B pristine: false
      S status: "VALID"
    ▶ O statusChanges: EventEmitter {_isScalar: false, observers: [], closed: false, isS…
      B touched: true
      N validator: null
    ▼ O value: Object
        S email: "aaaa@aa.com"
        S password: "12345678"
      ▶ Prototype Object
    ▶ O valueChanges: EventEmitter {_isScalar: false, observers: [], closed: false, isSt…
  ▶ Prototype FormGroup
  ▼ O ngSubmit: EventEmitter
      B __isAsync: false
      B _isScalar: false
      B closed: false
      B hasError: false
      B isStopped: false
    ▶ O observers: [Subscriber] (1)
      N thrownError: null
  ▶ Prototype EventEmitter
    B submitted: true
▶ Prototype NgForm
```

Afficher le label en rouge en cas d'erreur

globals.scss

```
.ion-invalid.ion-touched ion-label {
  color: var(--ion-color-danger);
}
```

Change la propriété color des balises ion-label enfants d'une balise qui possède les classes ion-invalid et ion-touched

Afficher un message d'erreur

```
<ion-item>
  <ion-label position="floating">email</ion-label>
  <ion-input type="text" ngModel required email
             name="email"
             #emailControl="ngModel">

  </ion-input>
  <ion-text *ngIf="! emailControl.valid && emailControl.touched"
            color="danger" slot="end">
    Vous devez saisir un email valide
  </ion-text>
</ion-item>
```

Touched indique que le champ a été modifié, car un formulaire non encore remplit est invalide hors on ne veut pas afficher les erreurs sans avoir donné à l'utilisateur l'occasion de faire une saisie

Désactiver le bouton tant que le formulaire est invalide

```
<ion-button type="submit" expand="block" color="primary"
[disabled]="! myForm.valid">
  Connexion
</ion-button>
```

Traitement du formulaire

```
export class HomePage {

    constructor() {
    }

    private checkCredentials(user){
      return user.email === 'moi@moi.com' && user.password === '123456';
    }

    onSubmit(form: NgForm) {
        if (!form.valid) {
            return;
        }
        // Les données sont dans la clef value de l'objet form
        const formData = form.value;

        if (this.checkCredentials(formData)){
          // Todo rediriger vers une page
        }
    }
}
```

Reactive forms

Imports

home.module.ts

```
import { NgModule } from '@angular/core';
import { CommonModule } from '@angular/common';
import { IonicModule } from '@ionic/angular';
import {FormsModule, ReactiveFormsModule} from '@angular/forms';
import { HomePage } from './home.page';

import { HomePageRoutingModule } from './home-routing.module';


@NgModule({
  imports: [
    CommonModule,
    FormsModule,
    ReactiveFormsModule,
    IonicModule,
    HomePageRoutingModule
  ],
  declarations: [HomePage]
})
export class HomePageModule {}
```

Le formulaire

home.page.html

```
<form>
    <ion-list>
        <ion-item>
            <ion-label position="floating">Nom</ion-label>
            <ion-input type="text"
                       formControlName="name">
            </ion-input>
        </ion-item>

        <ion-item>
            <ion-label position="floating">Date de naissance</ion-label>
            <ion-datetime min="1900-01-01" max="2015-01-01"
                          formControlName="birthDate">
            </ion-datetime>
        </ion-item>
    </ion-list>
</form>
```

home.page.ts

Le contrôleur

```
import {Component, OnInit} from '@angular/core';
import {FormControl, FormGroup, Validators} from '@angular/forms';

@Component({
  selector: 'app-home',
  templateUrl: 'home.page.html',
  styleUrls: ['home.page.scss'],
})
export class HomePage implements OnInit{

  public myForm: FormGroup;

  constructor() {}

  ngOnInit(): void {
    this.myForm = new FormGroup({
      name: new FormControl(null, {
        updateOn: 'blur',
        validators: [Validators.required, Validators.pattern('[a-zA-Z]+')]
      }),
      birthDate: new FormControl(null, {
        updateOn: 'blur',
        validators: [Validators.required]
      })
    });
  }
}
```

Imports

Cycle de vie

Variable publique pour pouvoir être affichée dans la vue

Instanciation du formulaire

FormControl(valeur initiale, options)

La clef ici correspond à l'attribut formControlName de la vue

Liaison entre la vue et le contrôleur

home.page.html

```
<form [formGroup]="myFor
    <ion-list>
       // ici les champs du formulaire
    </ion-list>
    <ion-button (click)="onCreatePerson()"
                [disabled]="!myForm.valid"
                expand="block">
        Valider
    </ion-button>
</form>
```

```
onCreatePerson(){
  console.log(this.myForm);
}
```

home.page.ts

Afficher les erreurs

```
<ion-item *ngIf="! myForm.get('name').valid && myForm.get('name').touched">
    <ion-label color="danger">
        Le nom ne peut comporter que des caractères alphabétiques
    </ion-label>
</ion-item>
```

home.page.html

Les Validators

```
class Validators {
  static min(min: number): ValidatorFn
  static max(max: number): ValidatorFn
  static required(control: AbstractControl): ValidationErrors | null
  static requiredTrue(control: AbstractControl): ValidationErrors | null
  static email(control: AbstractControl): ValidationErrors | null
  static minLength(minLength: number): ValidatorFn
  static maxLength(maxLength: number): ValidatorFn
  static pattern(pattern: string | RegExp): ValidatorFn
}
```

Il est également possible de créer nos propres classes de validation

Directive Reorder

home.html

```
<ion-list>
  <ion-list-header>Les fruits</ion-list-header>

  <ion-reorder-group
    disabled="false"
    (ionItemReorder)="reorderFruits($event)">

    <ion-item *ngFor="let item of fruits">
      <ion-label>{{ item }}</ion-label>
      <ion-reorder slot="end"></ion-reorder>
    </ion-item>

  </ion-reorder-group>
</ion-list>
```

home.ts

```
reorderFruits(ev){
  //élément à déplacer
  let item = this.fruits[ev.detail.from];

  //suppression de l’élément à la position de départ
  this.fruits.splice(ev.detail.from, 1);

  //insertion de l’élément à la position d’arrivée
  this.fruits.splice(ev.detail.to, 0, item);

   //Finalisation du déplacement de l'élément
   ev.detail.complete();
}
```

Navigation

Ionic 5

Principes

- Chaque écran est associé à une route (comme dans une application web)
- Une route peut recevoir des paramètres
- Toutes les routes sont déclarées dans un seul fichier de routage

Définition des routes

app-routing.modules.ts

```
const routes: Routes = [
  { path: '', redirectTo: 'home', pathMatch: 'full' },
  { path: 'home', loadChildren: './home/home.module#HomePageModule' }
];

@NgModule({
  imports: [
    RouterModule.forRoot(routes,
        { preloadingStrategy: PreloadAllModules })
  ],
  exports: [RouterModule]
})
export class AppRoutingModule { }
```

Trois types de routes

```
const routes: Routes = [
  //Redirection
  { path: '', redirectTo: 'home', pathMatch: 'full' },
  //lazy
  { path: 'home', loadChildren: './home/home.module#HomePageModule' }
  //eager
  { path: 'contact', component: contactPage}
];
```

Lien vers une route

```
<a href="/home">Accueil</a>

<ion-button href="/home">Accueil</ion-button>
```

```
<a routerLink="/home">Accueil</a>

<ion-button routerLink="/home">Accueil</ion-button>
```

On préférera routerLink car il n'induit pas le rechargement de la page

Lien vers une route dans un contrôleur

```
import { Component } from '@angular/core';
import { Router } from '@angular/router';

@Component({ ... })
export class HomePage {
  constructor(private router: Router) {}

  go() {
    this.router.navigateByUrl('/animals');
  }
}
```

Lien vers une route dans un contrôleur

```
import { Component } from '@angular/core';
import { Router } from '@angular/router';

@Component({ ... })
export class HomePage {
  constructor(private router: Router) {}

  go() {
    this.router.navigate(['animals']);
  }
}
```

Passage de paramètre dans la route

```
const routes: Routes = [
  //Route avec paramètre
  { path: 'items/:id', component: ItemPage },
];
```

app-routing.modules.ts

```
<a href="/items/5">Lien vers item 5</a>
```

Autre méthode

```
const routes: Routes = [
  //Route avec paramètre
  { path: 'items/:id', component: ItemPage },
];
```

app-routing.modules.ts

```
<ion-button
   [routerLink]="['/items/',{id: '6'}]">
   contact 6
</ion-button>
```

Dans un contrôleur

```
import { Component } from '@angular/core';
import { Router } from '@angular/router';

@Component({ ... })
export class HomePage {
  constructor(private router: Router) {}

  go() {
    Const id = 5;
    this.router.navigate(['items', id]);
  }
}
```

Récupération des paramètres

```
constructor(private route: ActivatedRoute) {
  this.route.paramMap.subscribe(
      (data: ParamMap) => {
        this.id = data.get('id');
      ));
}
```

*.page.ts

Une meilleure méthode

```
export class FirstPage implements OnInit, OnDestroy {

  public name;
  private paramSubscription: Subscription;

  constructor(private activatedRoute: ActivatedRoute) { }

  ngOnInit() {
    this.paramSubscription = this.activatedRoute.paramMap
    .subscribe(
        (data: ParamMap) => this.name = data.get('name')
    );
  }

  ngOnDestroy(): void {
    this.paramSubscription.unsubscribe();
  }
}
```

stockage de la souscription

injection de la dépendance

abonnement aux données

Désabonnement pour éviter les fuite de mémoire

Récupération des paramètres

```
constructor(private route: ActivatedRoute) {
  this.id = this.activatedRoute.snapshot
                .paramMap.get('id');
}
```

contact.page.ts

Sans observable, cela veut dire que si le paramètre change la page ne sera pas notifiée de ce changement

Envoi de paramètres sans route

```
constructor(private router: Router) { }

goToPage() {
  const user = {name: 'Joe', role: 'user'};
  const extras: NavigationExtras = {
    queryParams: {
      name: user.name,
      role: user.role
    }
  };
  this.router.navigate(['details'], extras);
}
```

list.ts

Récupération des paramètres

```
constructor( private activatedRoute: ActivatedRoute) {
  this.activatedRoute.queryParams.subscribe(
      (data: any) => {
        console.log(data);
      }
  );
}
```

details.ts

Bouton retour

contact.ts

```
<ion-header>
 <ion-toolbar color="primary">
   <ion-buttons slot="start">
     <ion-back-button defaultHref="/dashboard"></ion-back-button>
   </ion-buttons>
   <ion-title>Details</ion-title>
 </ion-toolbar>
</ion-header>
```

L'attribut defaultHref détermine la page sur laquelle pointe le bouton. Si ce paramètre est omis alors le bouton pointe vers la page précédente dans l'historique

Cycle de vie

Evénement	Description
ngOnInit	Premier chargement de la page
ionViewWillEnter	Lorsque la page est sur le point de devenir la page active
ionViewDidEnter	Lorsque la page est devenue la page active qu'il s'agisse d'un premier chargement ou d'un retour de navigation
ionViewWillLeave	Lorsque la page est sur le point d'être quittée
ionViewDidLeave	Lorsque la page a été quittée est n'est plau la page active
ngOnDestroy	Lorsque la page est sur le point d'être détruite (fermeture de l'application)

Widgets

Toast

```
import { ToastController } from 'ionic-angular';

export class MyPage {
  constructor(public toastCtrl: ToastController) {
  }

  async showToast() {
    //options
    const options = {
     message: 'Hello', duration: 1000,
     position: bottom
    };

    //Création du toast
    const toast = await this.toastCtrl.create(options);

    //Affichage
    toast.present();
  }
}
```

Avec une promesse

```
import { ToastController } from 'ionic-angular';

export class MyPage {
  constructor(public toastCtrl: ToastController) {
  }

  showToast() {
    //options
    const options = {
     message: 'Hello', duration: 1000,
     position: bottom
    };

    //Création du toast
    this.toastCtrl.create(options)
                  .then( toast => toast.present());
  }
}
```

Scroll

Un conteneur permettant de scroller le contenu sur un ou deux axes

```
<ion-scroll scrollX="true" scrollY="true" zoom="true">
  <img src="…">
</ion-scroll>
```

Loading

Une animation indiquant une opération en cours

```
import {LoadingController} from 'ionic-angular';

constructor(public loadingCtrl: LoadingController) {}

async launchProcess(){
  const options = {content: 'chargement en cours'};
  const loading = await this.loadingCtrl.create(options);

  loading.present();

  setTimeout(()=>{
     loading.dismiss();
   },
   5000);
}
```

Alert

Alertes

Une fenêtre popup affichant des informations

Il faut importer la classe AlertController

```
async showSimpleAlert(){
    Const options = {
      title: 'Attention',
      subTitle: 'soyez attentifs',
      message: 'Vous êtes prévenu',
      buttons: ['OK']
    };

    Const alert = await this.alertCtrl.create(options)

    alert.present();
}
```

Boutons personnalisés

```
getAlertButtons(){
  let noButton = {
      text: 'non',
      role: 'cancel',
      handler: ()=> {
        console.log('non');
      }
    };

  let okButton = {
      text: 'oui',
      handler: ()=> {
        console.log('oui');
      }
    };

  return [noButton, okButton];
}
```

Confirmation

```
async showConfirmAlert(){
    Const options = {
      title: 'Voulez-vous continuer',
      buttons: this.getAlertButtons()
    };

    Const alert = await this.alertCtrl.create(options)

    alert.present();
}
```

Zones de saisie

```
getAlertInputs(){
    let emailInput = {
      name: 'email', //nom de la variable
      placeholder: 'votre email'
    }

    let passwordInput = {
      name: 'password',
      placeholder: 'Votre mot de passe',
      type: 'password'
    }

    return [emailInput, passwordInput];
  }
```

Boutons

```
getAlertButtons(){
  let cancelButton = {
      text: 'Annuler',role: 'cancel',
      handler: ()=> { console.log('annulation');}
    };

  let okButton = {
      text: 'Valider',
      handler: (data)=> {
        console.log('Votre email : ' + data.email);
      }
    };

  return [cancelButton, okButton];
}
```

Prompt

```
async showPromptAlert(){
    Const options = {
      title: 'Login',
      inputs: this.getAlertInputs(),
      buttons: this.getAlertButtons()
    };

    Const alert = await this.alertCtrl.create(options)

    alert.present();
}
```

Cases à cocher

```
getFlavoursInputs(){

      let choices = [
        {type: 'checkbox', label: 'chocolat', value: '1'},
        {type: 'checkbox', label: 'café', value: '2'},
        {type: 'checkbox', label: 'ananas', value: '3'},
        {type: 'checkbox', label: 'fraise', value: '4'}
      ];

      return choices;
}
```

Les types checkbox ou radio ne sont pas compatibles avec les autres champs de saisie

FAB
Floating Action Buttons

Boutons flottants au dessus du contenu de l'écran

```
<!-- un seul bouton -->
<ion-fab top right>
    <button ion-fab><ion-icon name="help"></ion-icon></button>
</ion-fab>

<!-- liste de boutons -->
<ion-fab bottom right>
    <button ion-fab> <ion-icon name="arrow-dropup"></ion-icon></button>
    <ion-fab-list side="top">
        <button ion-fab><ion-icon name="star"></ion-icon></button>
        <button ion-fab><ion-icon name="heart"></ion-icon></button>
        <button ion-fab><ion-icon name="settings"></ion-icon></button>
    </ion-fab-list>
</ion-fab>
```

ActionSheet

Menu d'options affiché au dessus du contenu de l'écran

```
async showActionSheet(){
  const actionSheet = await this.actionSheetCtrl.create({
    title: "Que faire ?",
    buttons: [
      {text: 'Aller à la plage', icon: 'star',
         handler: ()=>{console.log('plage')} },
      {text: 'Supprimer', role: 'destructive',
         handler: ()=>{console.log('supprimer')} },
      {text: 'Annuler', role: 'cancel',
         handler: ()=>{console.log('annuler')} }
    ]
  });

  actionSheet.present();
}
```

Popover

Une page affichée au dessus du contenu de l'écran

Importer la classe PopoverController

```
async showPopover(evt){
      const popover = await this.popoverCtrl.create(PopoverPage);
      console.log(evt);
      popover.present({ev: evt});
    }
```

```
<ion-navbar>
 <ion-title>alerts</ion-title>
  <ion-buttons end>
    <button ion-button (click)="showPopover($event)">…</button
  </ion-buttons>
</ion-navbar>
```

Le popover

```
import { ViewController } from 'ionic-angular';
import { Component } from '@angular/core';

@Component({
  templateUrl: 'popover.html',
})

export class PopoverPage {
  constructor(public viewCtrl: ViewController) {}

  close(){
    this.viewCtrl.dismiss();
  }
}
```

```
<ion-list>
    <ion-item (click)="close()">Option A</ion-item>
    <ion-item (click)="close()">Option B</ion-item>
    <ion-item (click)="close()">Option C</ion-item>
</ion-list>
```

Modal

- Même principe que popover
  - Utilise la classe ModalController
  - Affiche un écran qui peut se fermer avec un bouton

Services

Définition

- Un service est un classe qui fournit des données ou de la logique de traitement
- Un même service peut être utilisé par plusieurs autres classes
- Les services déclarés dans un module sont des singletons pour les autres classes de ce module

Création

- Créer une classe
- Lui attribuer le décorateur @Injectable

Génération avec la ligne de commande

```
ionic g service nom-du-service
```

Utilisation

- Déclarer la classe comme provider dans un module (app.module.ts)
- Importer le service dans la classe utilisatrice
- Injecter le service dans le constructeur de la classe utilisatrice

```
import { Injectable } from '@angular/core';

@Injectable()

export class DataService {

  private data = [];

  constructor(){}

  getData(){
    return data;
  }

  addData(item){
    this.data.push(item);
  }
}
```

app.module.ts

```
import { DataService } from ./../services/data-service';

import { NgModule, ErrorHandler } from '@angular/core';
import { IonicApp, IonicModule, IonicErrorHandler }
  from 'ionic-angular';
import { MyApp } from ./app.component';
import { HomePage } from ../pages/home/home';

@NgModule({
  declarations: [MyApp, HomePage],
  imports: [ IonicModule.forRoot(MyApp)],
  bootstrap: [IonicApp],
  entryComponents: [MyApp,HomePage],
  providers: [
    {provide: ErrorHandler, useClass: IonicErrorHandler},
    DataService
  ]
})
export class AppModule {}
```

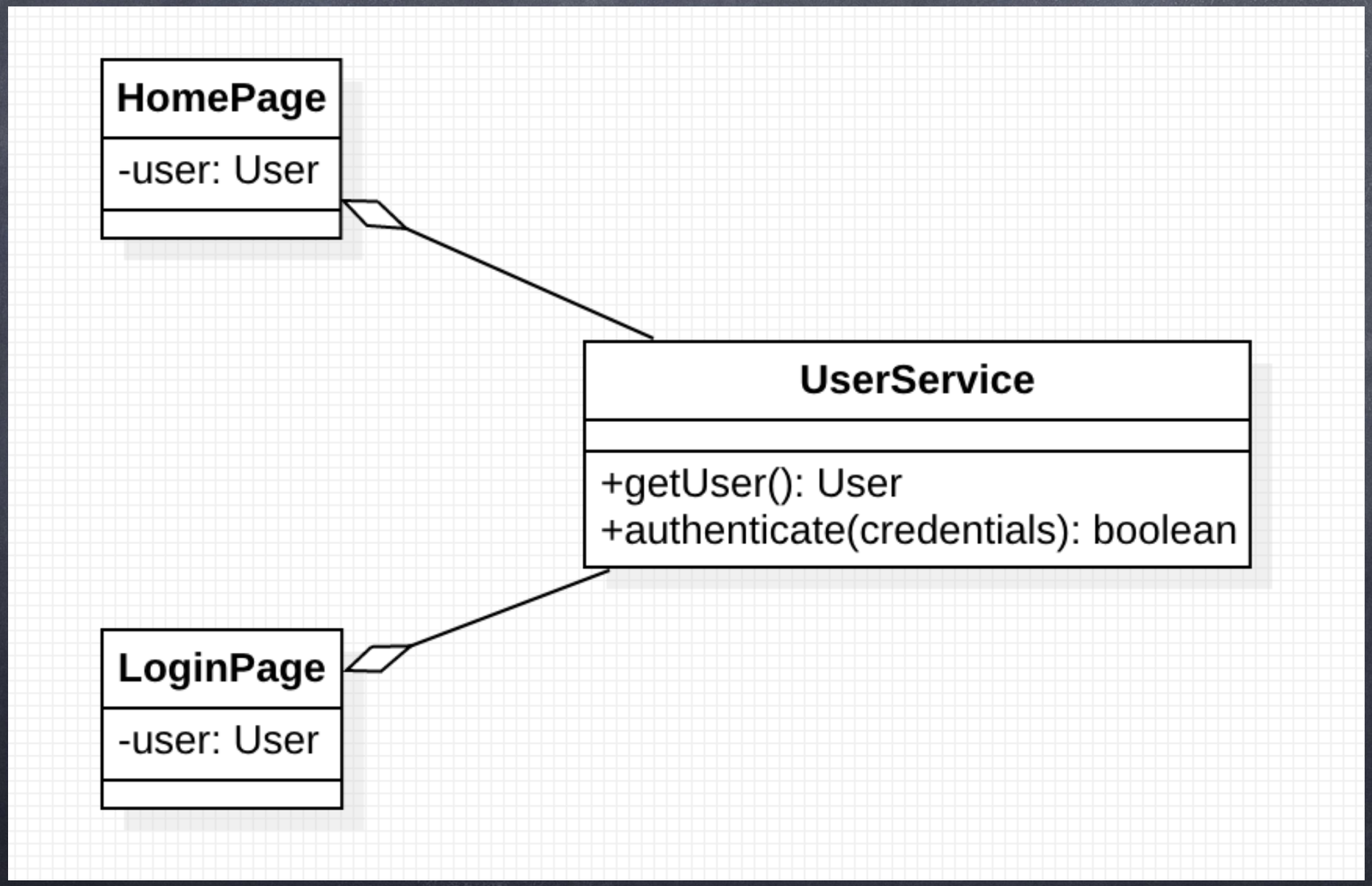
```
import { DataService } from ./../services/data-service';

  …

@Component({
  selector: 'page-home',
  templateUrl: 'home.html'
})
export class HomePage {

  constructor(private provider:DataService) {

  }

  insertData(item){
    this.provider.addData(item);
  }

  getOne(index){
    let data = this.provider.getData();
    return data[index];
  }

}
```

home.ts

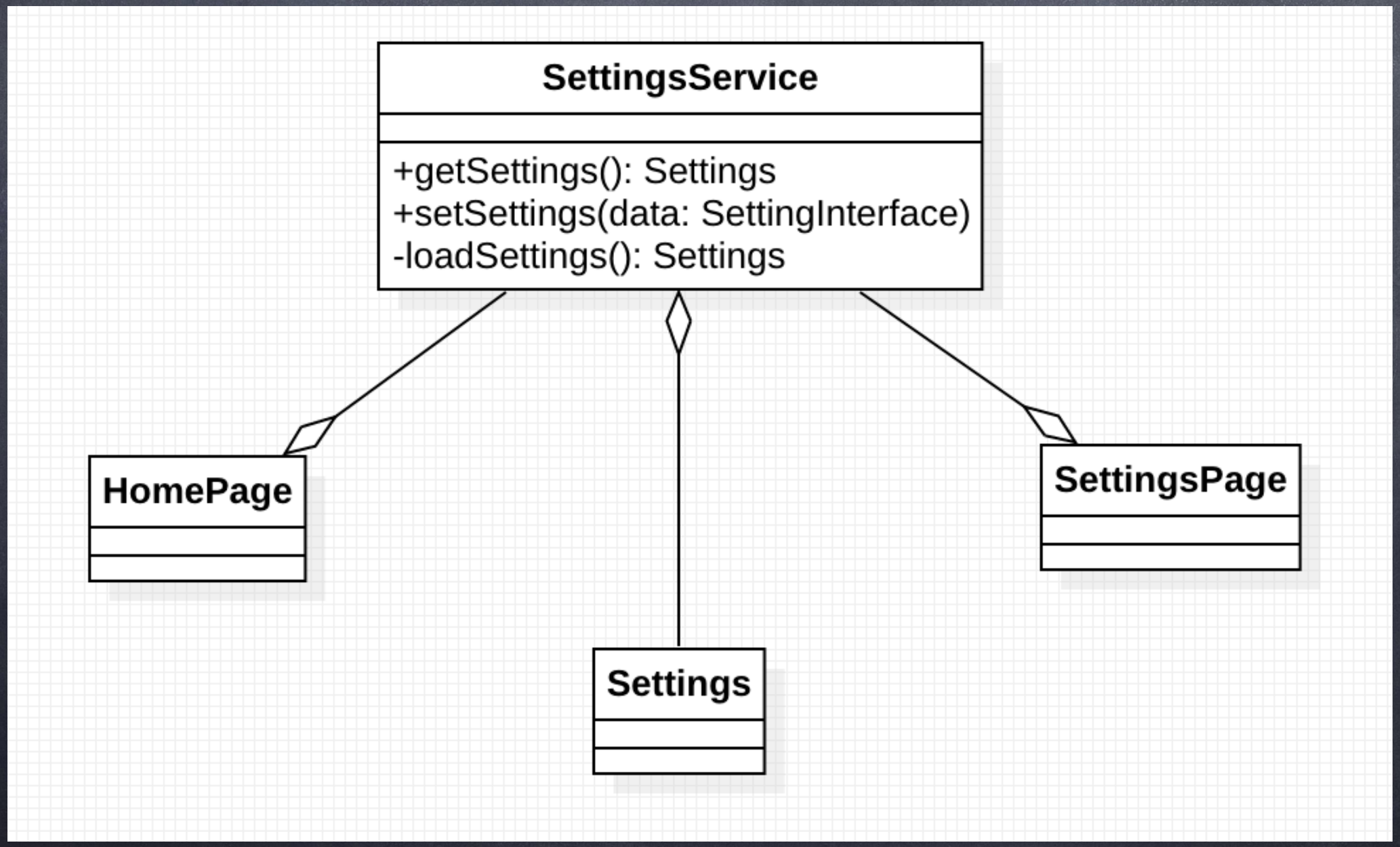
A quoi ça sert

- Partager des données entre plusieurs pages
- Factoriser des comportements
- Mieux organiser le code

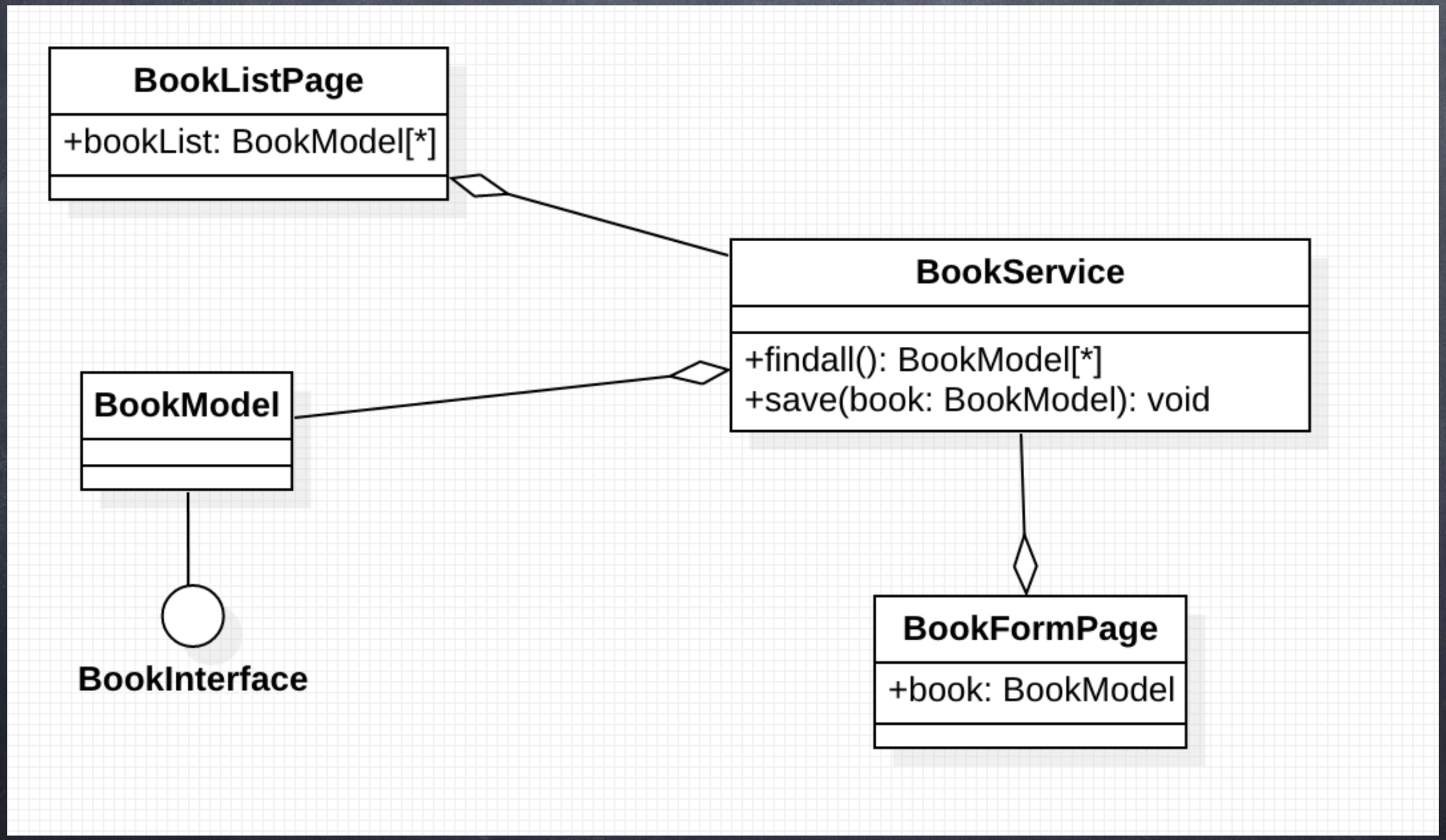
Partage de données



Paramètrage



Accès aux données



Service non partagé

Si le service est déclaré directement dans la classe utilisatrice et non dans le module alors chaque appel au service retournera une nouvelle instance

Il ne sera donc pas possible de partager des données avec cette méthode

```
import { MonProvider } from './../providers/note-provider';

  …

@Component({
  selector: 'page-home',
  templateUrl: 'home.html',
  providers: [MonProvider]
})
export class HomePage {

  constructor(private provider:MonProvider) {

  }

  insertData(item){
    this.provider.addData(item);
  }

  getOne(index){
    let data = this.provider.getData();
    return data[index];
  }

}
```

home.ts

Pipes personnalisés

```
Ionic g pipe pipes/capitalize
```

```
import { Pipe, PipeTransform } from '@angular/core';

@Pipe({
  name: 'capitalize'
})
export class CapitalizePipe implements PipeTransform {

  transform(value: string, ...args: string[]): string {
    return value.charAt(0).toUpperCase()
           + value.substring(1).toLowerCase();
  }
}
```

capitalize.ts

```
…

import {CapitalizePipe} from '../pipes/capitalize.pipe';


@NgModule({
    imports: [
        CommonModule,
        FormsModule,
        IonicModule,
        HomePageRoutingModule,
    ],
  declarations: [HomePage, CapitalizePipe]
})
export class HomePageModule {}
```

home.module.ts

home.html

```
<ion-list>
    <ion-item *ngFor="let user of userList">
        <h2>
          {{user.name| capitalize}}
        </h2>
    </ion-item>
</ion-list>
```

Rassembler les pipes dans un module

```
import { NgModule } from '@angular/core';
import { CommonModule } from '@angular/common';
import { CapitalizePipe } from './capitalize.pipe';
import { TruncatePipe } from './truncate.pipe';

@NgModule({
  declarations: [CapitalizePipe, TruncatePipe],
  imports: [
    CommonModule
  ],
  exports: [CapitalizePipe, TruncatePipe]
})
export class SharedPipesModule { }
```

pipes/shared-pipes.module.ts

Import du module dans une page

```
…

import { SharedPipesModule } from '../pipes/shared-pipes.module';

@NgModule({
    imports: [
        CommonModule,
        FormsModule,
        IonicModule,
        HomePageRoutingModule,
        SharedPipesModule
    ],
  declarations: [HomePage]
})
export class HomePageModule {}
```

home.module.ts

Storage Cordova

Principes

- Une classe qui fournit une abstraction pour la persistance des données
- En fonction du contexte, le moteur de persistance peut varier (SQLite, indexDB, WebSQL, LocalStorage)

Utilisation avec SQLite

Il faut tout d'abord installer le plugin avec la ligne de commande

```
ionic cordova plugin add cordova-sqlite-storage
```

```
npm install --save @ionic-native/storage
```

Référencer le plugin
dans app/app.module.ts
directive @NgModule, clef imports

Utilisation simple

app.module.ts

```
import { IonicStorageModule } from '@ionic/storage';

//…

@NgModule({
  declarations: [MyApp, HomePage],
  imports: [ IonicModule.forRoot(MyApp),
    IonicStorageModule.forRoot()
  ],
  bootstrap: [IonicApp],
  entryComponents: [MyApp,HomePage],
  providers: [
  ]
})
export class AppModule {}
```

home.ts

```
import { Storage } from '@ionic/storage';

  //…

@Component({
  selector: 'page-home',
  templateUrl: 'home.html'
})
export class HomePage {

  constructor(private storage:Storage) {
    this.storage.set('name','bob');

    this.storage.get('name').then((value)=>{
      console.log(value)
    }
  }
}
```

Avec un provider

app.module.ts

```
import { UserService } './../providers/user-service';;

//…

@NgModule({
  declarations: [MyApp, HomePage],
  imports: [ IonicModule.forRoot(MyApp)],
  bootstrap: [IonicApp],
  entryComponents: [MyApp,HomePage],
  providers: [
    //…
    UserService
  ]
})
export class AppModule {}
```

user-provider.ts

```
import { Injectable } from '@angular/core';
import {Storage} from '@ionic/storage';

@Injectable()

export class UserService {

  constructor(public storage:Storage){}

  getUsers(){
    return this.storage.get('users');
  }

  addUser(user){
    this.getUsers().then((data) => {
      let userList = JSON.parse(data);
       userList.push(user);
      this.storage.set('users',
           JSON.stringify(userList));
    });
  }
}
```

```
import { UserService } from './../services/user-service';

  …

@Component({
  selector: 'page-home',
  templateUrl: 'home.html'
})
export class HomePage {

  users = [];

  constructor(public userSrv: UserService) {
    loadUsers();
  }

  loadUsers(index){
    this.userSrv.getUsers().then(
      (data) => {
        this.users = JSON.stringify(data);
      }
    );
  }

}
```

home.ts

Storage Capacitor

Principes

- Une classe qui fournit une abstraction pour la persistance des données
- Utilise LocaleStorage sur une navigateur web et les moteurs UserDefaults ou SharedPreferences sur iOS ou Android

Installation

Il faut tout d'abord installer Capacitor

```
npm install --save @capacitor/core @capacitor/cli
```

Les méthodes

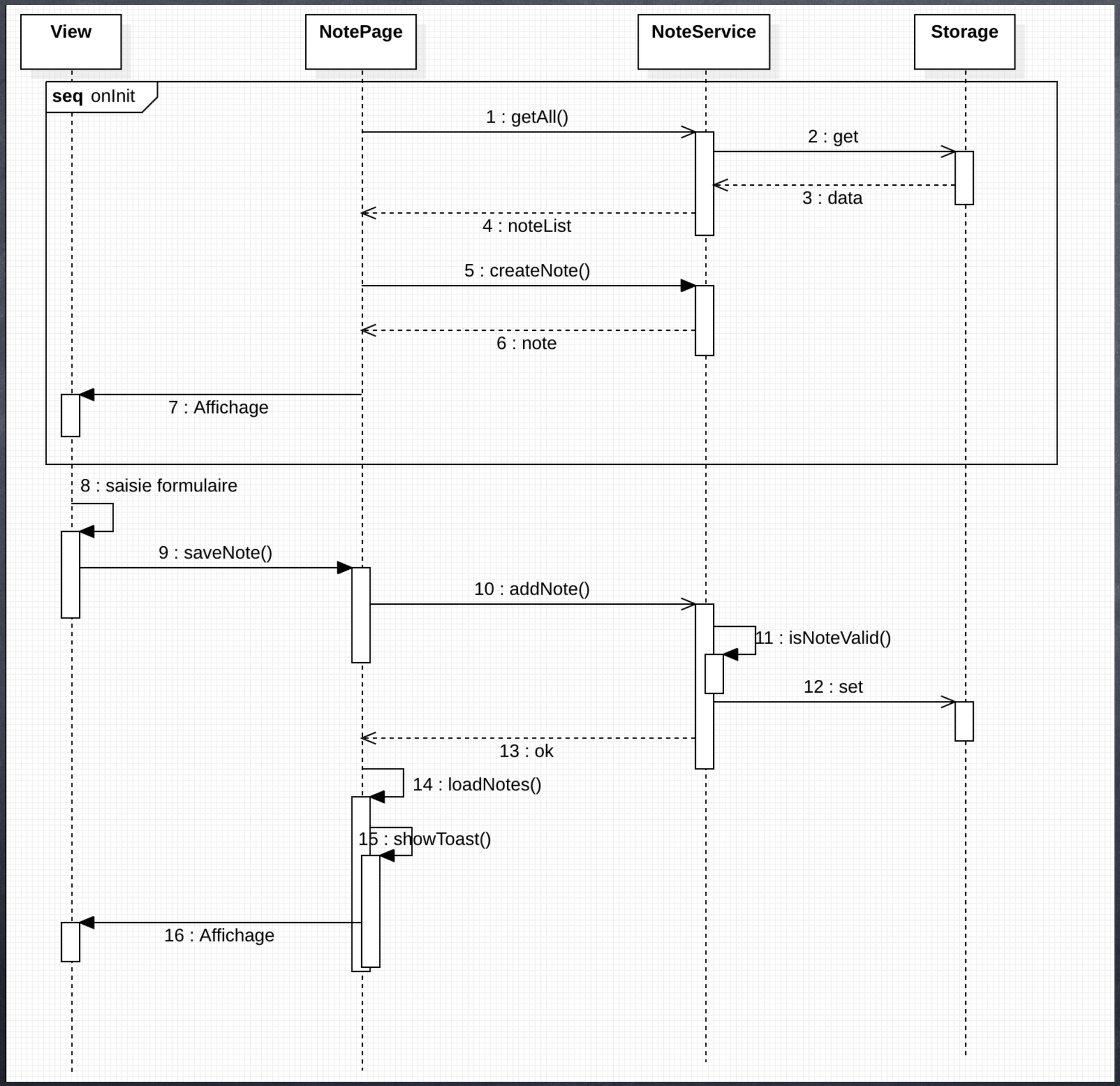
clear	Efface toutes les clefs
get	Retourne la valeur d'une clef
set	Ajoute ou modifie une clef
remove	Supprime une clef
keys	Retourne la liste des clefs

Utilisation simple

home.ts

```
import { Plugins } from '@capacitor/core';

const { Storage } = Plugins;

const STORAGE_KEY = 'user';

  //…

  async setUser() {
    await Storage.set({
      key: STORAGE_KEY,
      value: JSON.stringify({
        id: 1,
        name: 'Paul'
      })
    });
  }

  async getUser() {
    const data: any = await Storage.get({ key: STORAGE_KEY });
     return JSON.parse(data.value);
  }
}
```

Un crud avec Storage


View
NotePage
NoteService
Storage
seq onInit
1 : getAll()
2 : get
3 : data
4 : noteList
5 : createNote()
6 : note
7 : Affichage
8 : saisie formulaire
9 : saveNote()
10 : addNote()
11 : isNoteValid()
12 : set
13 : ok
14 : loadNotes()
15 : showToast()
16 : Affichage

Le service

```
import { Injectable } from '@angular/core';
import { Plugins } from '@capacitor/core';

const { Storage } = Plugins;

const STORAGE_KEY = 'noteList';

/**
 * Structure de données d'une note
 */
export interface NoteInterface {
  title: string;
  id: number;
  text: string;
}

/**
 * Structure de données de la réponse de Storage
 */
interface StorageResponse {
  value: string;
}


@Injectable({
  providedIn: 'root'
})
export class NoteService {

  constructor() { }

  /**
   * Lecture des notes depuis Storage
   */
  async getAll(): Promise<Array<NoteInterface>>{
    const data: StorageResponse = await Storage.get({key:
STORAGE_KEY});
    return JSON.parse(data.value) || [];
}

/**
 * Validation de la saisie d'une note
 */
private isNoteValid(note: NoteInterface){
  return  note.title && note.title.trim() !== ''
          && note.text && note.text.trim() !== '';
}

/**
 * Ajout d'une note
 */
async addNote(note: NoteInterface){
  const isValid = this.isNoteValid(note);
  if (isValid){
    const noteList: Array<NoteInterface> = await
this.getAll();
    note.id = new Date().getTime();
    noteList.push(note);
    await Storage.set({key: STORAGE_KEY, value:
JSON.stringify(noteList)});
  }
  return isValid;
}

/**
 * Création d'une note vide
 */
createNote(): NoteInterface{
  return  {
    title: '',
    text: '',
    id: null
  };
}
}
```

1

2

3

4

5

6

1

Importation de Storage

```
import { Plugins } from '@capacitor/core';

const { Storage } = Plugins;

const STORAGE_KEY = 'noteList';
```

2

Interfaces

```
/**
 * Structure de données d'une note
 */
export interface NoteInterface {
  title: string;
  id: number;
  text: string;
}


/**
 * Structure de données de la réponse de Storage
 */
interface StorageResponse {
  value: string;
}
```

3

Lecture des données

async indique que la fonction contient un appel asynchrone et donc retourne une promesse

Retourne une promesse de résultat

```
/**
 * Lecture des notes depuis Storage
 */
async getAll(): Promise<Array<NoteInterface>>{
  const data: StorageResponse = await Storage.get({key: STORAGE_KEY});
  return JSON.parse(data.value) || [];
}
```

Attention, on récupère un objet, les données sont dans la clef value. Il faut également déssérialiser pour récupérer un tableau

Valeur par défaut si la clef n'existe pas

La clef est passée dans un objet

4

Validation

```
/**
 * Validation de la saisie d'une note
 */
private isNoteValid(note: NoteInterface){
  return  note.title && note.title.trim() !== ''
          && note.text && note.text.trim() !== '';
}
```

Le titre et le texte ne peuvent être vides

5

Ajout d'une note

Uniquement si la note est valide

Lecture de la liste des notes

```
async addNote(note: NoteInterface){
  const isValid = this.isNoteValid(note);
  if (isValid){
    const noteList: Array<NoteInterface> = await this.getAll();
    note.id = new Date().getTime();
    noteList.push(note);
    await Storage.set({key: STORAGE_KEY, value: JSON.stringify(noteList)});
  }
  return isValid;
}
```

Création de l'id

Ajout de la nouvelle note

Sauvegarde du tableau dans le Storage. Attention il faut sérialiser (convertir l'objet en string)

6

Création d'une note vide

```
createNote(): NoteInterface{
  return  {
    title: '',
    text: '',
    id: null
  };
}
```

Le contrôleur

```
import {Component, OnDestroy, OnInit} from '@angular/core';
import {NoteInterface, NoteService} from '../services/
note.service';
import {ToastController} from '@ionic/angular';
import {Subscription} from 'rxjs';

@Component({
  selector: 'app-home',
  templateUrl: 'home.page.html',
  styleUrls: ['home.page.scss'],
})
export class HomePage implements OnInit, OnDestroy{

  // La liste des notes
  public noteList: Array<NoteInterface> = [];

  // Un objet note
  public note: NoteInterface;

  private noteSubscription: Subscription;

  constructor(
      private noteService: NoteService,
      private toastCtrl: ToastController) {}

  ngOnInit(): void {
    this.note = this.noteService.createNote();
    // Abonnement aux notes
    this.noteSubscription =
this.noteService.noteChanged.subscribe(
        (data: NoteInterface[]) => this.noteList = data
    );
  }

  ngOnDestroy(): void {
    // Désabonnement
    this.noteSubscription.unsubscribe();
  }

  /**
   * Sauvegarde d'une note
   */
  saveNote(){
    let message = '';
    // Sauvegarde de la note
    const ok = this.noteService.addNote(this.note);
    if (ok){
      message = 'Votre note a été enregistrée';
      this.note = this.noteService.createNote();
    } else {
      // message d'erreur
      message = 'Impossible d\'enregistrer la note';
    }
    // Affichage du message
    this.showToast(message);
  }

  /**
   * Création et affichage du message
   * sous la forme d'un toast
   */
  async showToast(text: string){
    const toast = await this.toastCtrl.create({
      message: text,
      duration: 1000,
      position: 'top'
    });

    await toast.present();
  }

}
```

1 2 3 4 5 6 7

1

Importations

```
import {Component, OnInit} from '@angular/core';
import {NoteInterface, NoteService} from '../services/note.service';
import {ToastController} from '@ionic/angular';
```

2

Les variables

```
// La liste des notes
public noteList: Array<NoteInterface> = [];

// Un objet note
public note: NoteInterface;
```

3 Injection des dépendances

La variable

La classe

```
constructor(
    private noteService: NoteService,
    private toastCtrl: ToastController) {}
```

Raccourcis, évite la déclaration et l'affectation explicite d'une variable

4 Initialisation de l'application

```
ngOnInit(): void {
  // Chargement des notes
  this.loadNotes();
}
```

Il faut passer par une autre méthode (loadNotes) car ngOnInit doit retourner void et ne peut donc être asynchrone
(une méthode asynchrone retourne toujours une promesse)

5

Chargement des notes

On attend le résultat avant de faire l'affectation

```
private async loadNotes(){
  // Lecture de la liste des notes depuis le service
  this.noteList = await this.noteService.getAll();
  // Création d'une note
  this.note = this.noteService.createNote();
}
```

Création d'une note vide pour le formulaire

6

Sauvegarde d'une note

addNote retourne un booléen

```
async saveNote(){
  let message = '';
  // Sauvegarde de la note
  const ok = await this.noteService.addNote(this.note);
  if (ok){
    await this.loadNotes();

    message = 'Votre note a été enregistrée';
  } else {
    // message d'erreur
    message = 'Impossible d\'enregistrer la note';
  }
  // Affichage du message
  this.showToast(message);
}
```

Après l'ajout on met à jour la liste des notes

7

Affichage du message

```
async showToast(text: string){
  const toast = await this.toastCtrl.create({
    message: text,
    duration: 1000,
    position: 'top'
  });

  await toast.present();
}
```

Avec un observable

Principes

- La classe service fournit un Subject
- La classe Page s'abonne au Subject
- La classe service émet un message
- La classe page reçoit le message

Comme dans la vie réelle, lorsque je m'abonne, je ne reçois pas immédiatement la revue, mais je recevrai chaque nouveau numéro émis tant que mon abonnement sera actif

Intérêts

- La classe émettrice n'a pas connaissance des abonnés, cela réduit le couplage
- La classe réceptrice s'abonne une fois et reçoit les messages à chaque émission, elle n'a pas à s'occuper du cycle de vie du service.
  - exemple : je m'abonne à la liste des notes, lors de l'ajout je n'ai rien à faire, je reçois les nouvelles données, c'est le principe de la programmation reactive

Le service

```
import { Injectable } from '@angular/core';
import { Plugins } from '@capacitor/core';
import {Subject} from 'rxjs';

const { Storage } = Plugins;

const STORAGE_KEY = 'noteList';

/**
 * Structure de données d'une note
 */
export interface NoteInterface {
  title: string;
  id: number;
  text: string;
}

/**
 * Structure de données de la réponse de Storage
 */
interface StorageResponse {
  value: string;
}


@Injectable({
  providedIn: 'root'
})
export class NoteService {

  public noteChanged: Subject<NoteInterface[]>;

  private noteList: NoteInterface[] = [];

  constructor() {
    this.noteChanged = new Subject<NoteInterface[]>();
  }
```

1

2

3

```
/**
 * Lecture des notes depuis Storage
 */
getAll(): void{
  Storage.get({key: STORAGE_KEY}).then(
      (data: StorageResponse) => {
        this.noteList = JSON.parse(data.value) || [];
        this.noteChanged.next(this.noteList);
      }
  );
}

/**
 * Validation de la saisie d'une note
 */
private isNoteValid(note: NoteInterface){
  return  note.title && note.title.trim() !== ''
          && note.text && note.text.trim() !== '';
}

/**
 * Ajout d'une note
 */
addNote(note: NoteInterface){
  const isValid = this.isNoteValid(note);
  if (isValid){
    note.id = new Date().getTime();
    this.noteList.push(note);
    this.noteChanged.next(this.noteList);
  }
  return isValid;
}

/**
 * Création d'une note vide
 */
createNote(): NoteInterface{
  return  {
```

4

5

1

Importation de Subject

```
import { Injectable } from '@angular/core';
import { Plugins } from '@capacitor/core';
import {Subject} from 'rxjs';
```

2

Les variables

Un subject émet un événement, il faut préciser le type des données émises lors de cet événement.

```
public noteChanged: Subject<NoteInterface[]>;

private noteList: NoteInterface[] = [];
```

La liste des notes est stockée dans la classe

3

L'instanciation

Instanciation du Subject à la construction de l'objet, il faut à nouveau préciser le type des données émises.

```
constructor() {
  this.noteChanged = new Subject<NoteInterface[]>();
}
```

4

Chargement des notes

Interrogation du Storage

Stockage des données dans la classe

```
getAll(): void{
  Storage.get({key: STORAGE_KEY}).then(
      (data: StorageResponse) => {
        this.noteList = JSON.parse(data.value) || [];
        this.noteChanged.next(this.noteList);
      }
  );
}
```

Notification du changement aux abonnés

5

Ajout d'une note

Ajout de la note dans la variable de la classe

```
addNote(note: NoteInterface){
  const isValid = this.isNoteValid(note);
  if (isValid){
    note.id = new Date().getTime();
    this.noteList.push(note);
    this.noteChanged.next(this.noteList);
    Storage.set({
      key: STORAGE_KEY,
      value: JSON.stringify(this.noteList)
    });
  }
  return isValid;
}
```

Persistence dans le Storage

Notification des changements

Le contrôleur

```
import {Component, OnDestroy, OnInit} from '@angular/core';
import {NoteInterface, NoteService} from '../services/
note.service';
import {ToastController} from '@ionic/angul
import {Subscription} from 'rxjs';

@Component({
  selector: 'app-home',
  templateUrl: 'home.page.html',
  styleUrls: ['home.page.scss'],
})
export class HomePage implements OnInit, OnDestroy{

  // La liste des notes
  public noteList: Array<NoteInterface> = [];

  // Un objet note
  public note: NoteInterface;

  private noteSubscription: Subscription;

  constructor(
      private noteService: NoteService,
      private toastCtrl: ToastController) {}

  ngOnInit(): void {
    this.note = this.noteService.createNot
    // Abonnement aux notes
    this.noteSubscription =
this.noteService.noteChanged.subscribe(
        (data: NoteInterface[]) => this.noteList = data
    );
  }

  ngOnDestroy(): void {
    // Désabonnement
    this.noteSubscription.unsubscribe();
  }
```

1

2

3

4

```
  /**
   * Sauvegarde d'une note
   */
  saveNote(){
    let message = '';
    // Sauvegarde de la note
    const ok = this.noteService.addNote(this.note);
    if (ok){
      message = 'Votre note a été enregistrée';
      this.note = this.noteService.createNote();
    } else {
      // message d'erreur
      message = 'Impossible d\'enregistrer la note';
    }
    // Affichage du message
    this.showToast(message);
  }

  /**
   * Création et affichage du message
   * sous la forme d'un toast
   */
  async showToast(text: string){
    const toast = await this.toastCtrl.create({
      message: text,
      duration: 1000,
      position: 'top'
    });

    await toast.present();
  }


}
```

5

1

Importations

```
import {Component, OnDestroy, OnInit} from '@angular/core';
import {NoteInterface, NoteService} from '../services/note.service';
import {ToastController} from '@ionic/angular';
import {Subscription} from 'rxjs';
```

2

Les variables

```
// La liste des notes
public noteList: Array<NoteInterface> = [];

// Un objet note
public note: NoteInterface;

// L'abonnement aux données
private noteSubscription: Subscription;
```

3

L'abonnement

La variable stockant l'abonnement

Abonnement aux données

```
ngOnInit(): void {
  this.note = this.noteService.createNote();
  // Abonnement aux notes
  this.noteSubscription = this.noteService
      .noteChanged.subscribe(
          (data: NoteInterface[]) => this.noteList = data
  );
}
```

callback exécuté à chaque émission de noteChanged

4

Le désabonnement

```
ngOnDestroy(): void {
  // Désabonnement
  this.noteSubscription.unsubscribe();
}
```

Désabonnement à la destruction de l'objet pour libérer la mémoire

5

Ajout d'une note

```
saveNote(){
  let message = '';
  // Sauvegarde de la note
  const ok = this.noteService.addNote(this.note);
  if (ok){
    message = 'Votre note a été enregistrée';
    this.note = this.noteService.createNote();
  } else {
    // message d'erreur
    message = 'Impossible d\'enregistrer la note';
  }
  // Affichage du message
  this.showToast(message);
}
```

Ajout de la note, le service émettra un nouvel événement qui sera capturé par l'abonnement et permettra la mise à jour de la liste des notes

Création d'une note vide pour le formulaire

Constantes et configurations

Plusieurs solutions

- Définir un provider qui expose les constantes et/ou variables
- Définir des variables ou des constantes dans app.module.ts et les exporter

Première solution

```
import { Injectable } from '@angular/
core';

@Injectable()

export class ConfigProvider {

  private APIKEY = '123';

  constructor(){}

  getApiKey(){
    return this.APIKEY;
  }
}
```

app.module.ts

```
import { ConfigProvider } from ./../providers/config-provider';

import { NgModule, ErrorHandler } from '@angular/core';
import { IonicApp, IonicModule, IonicErrorHandler }
  from 'ionic-angular';
import { MyApp } from ./app.component';
import { HomePage } from ../pages/home/home';

@NgModule({
  declarations: [MyApp, HomePage],
  imports: [ IonicModule.forRoot(MyApp)],
  bootstrap: [IonicApp],
  entryComponents: [MyApp,HomePage],
  providers: [
    {provide: ErrorHandler, useClass: IonicErrorHandler},
    ConfigProvider
  ]
})
export class AppModule {}
```

home.ts

```
import { ConfigProvider }
  from './../providers/config-provider';

  …

@Component({
  selector: 'page-home',
  templateUrl: 'home.html'
})
export class HomePage {

  constructor(public config:ConfigProvider) {
    console.log(config.getApiKey());
  }
}
```

Deuxième solution

app.module.ts

```
import …

export const config = {
  APIKEY: 'F34A67843BCE45'
};

@NgModule({
 …
})
export class AppModule {}
```

home.ts

```
import { config } from ./../../app.module';

  …

@Component({
  selector: 'page-home',
  templateUrl: 'home.html',
})
export class HomePage {

  constructor(public provider:MonProvider) {
    console.log(config.APIKEY);
  }
}
```

Déploiement

Cibles

- Android (il faut installer Android Studio)
- iOS (il faut un mac et installer Xcode)
- PWA (Progressive Web Application)
- Desktop (il faut installer Electron)

Transpilation

Pour déployer, il faut obtenir la transpilation de notre projet dans un dossier
Pour cela il faut exécuter une des instructions suivantes dans un terminal

```
ng build --prod
ou
ionic build --prod
```

Activer Capacitor

Si Capacitor n'a pas été intégré à la création du projet, il faut le faire maintenant
avec la commande console suivante

```
ionic integrations enable capacitor
```

Initialisation du projet

L'initialisation permet de définir le nom de l'application et surtout son identifiant qui doit être unique. Il est possible de le changer par la suite dans la configuration du projet (capacitor.config.json)

```
npx cap init nom-du-projet id-du-projet
```

```
npx cap init myApp fr.smaloron.myapp
```

Ajouter une plateforme

L'ajout de la plateforme génère un projet Android ou iOS que vous pourrez ouvrir avec Android Studio ou Xcode

```
npx cap add android
npx cap add ios
```

Ouvrir le projet

Vous pouvez ouvrir le projet depuis le terminal, il est bien entendu possible d'ouvrir le projet directement depuis l'IDE

```
npx cap open android
npx cap open iOS
```

A chaque build

A chaque fois que votre code change et que vous faites un nouveau build, il faut copier les fichiers dans le projet Android ou iOS

```
npx cap copy
```

Si vous avez ajouté des plugins ou des dépendances avec npm il faut synchroniser les projets. Cette commande réalise également une copie du build

```
npx cap sync
```

Android

Lancement du projet

cible du lancement

Gestion des téléphones virtuels

Gestion des versions d'Android

android [~/Documents/formation/ionic/training-playground/demo-app/android] ~/Documents/formation/ionic/training-playground/demo-app/node_modules/@capacitor/android/ca...

capacitor 〉 src 〉 main 〉 AndroidManifest.xml | app | Nexus 5 API 29 | Git:

Android

- app
- capacitor-android
  - manifests
    - AndroidManifest.xml
  - java
  - res
    - drawable
    - layout
    - values
      - attrs.xml
      - colors.xml
      - strings.xml
      - styles.xml
- capacitor-cordova-android-plugins
- Gradle Scripts

AndroidManifest.xml | strings.xml

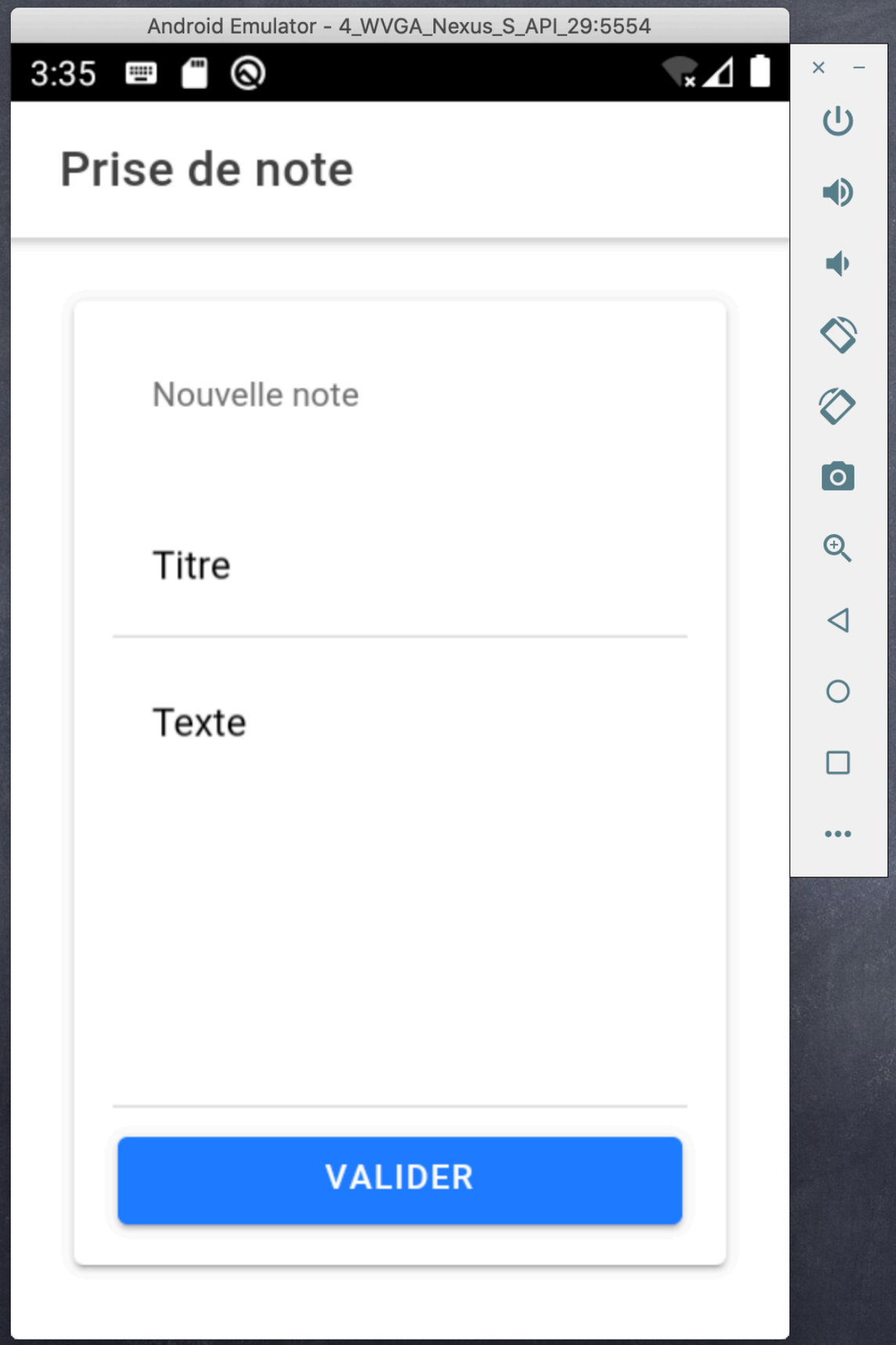
```
1  <?xml version="1.0" encoding="utf-8"?>
2  <manifest xmlns:android="http://schemas.android.com/apk/res/android"
3      package="com.getcapacitor.android">
4      <uses-feature android:name="android.hardware.camera"
5          android:required="false" />
6
7      <application>
8          <service android:name="com.getcapacitor.plugin.background.BackgroundTaskService" and
9          <receiver android:name="com.getcapacitor.plugin.notification.TimedNotificationPublis
10         <receiver android:name="com.getcapacitor.plugin.notification.NotificationDismissRece
11         <meta-data android:name="firebase_messaging_auto_init_enabled" android:value="false"
12         <service android:name="com.getcapacitor.CapacitorFirebaseMessagingService" android:s
13             <intent-filter>
14                 <action android:name="com.google.firebase.MESSAGING_EVENT" />
15             </intent-filter>
16         </service>
17     </application>
18 </manifest>
```

Project configurations files can be added to Git
View Files Always Add Don't Ask Again

Text | Merged Manifest

9: Version Control | Terminal | Build | 6: Logcat | TODO | Event Log

Project configurations files can be added to Git // View Files // Always Add // Don't Ask Again (moments ago) 1:1 LF UTF-8 4 spaces Git: storage-capacitor


Android Emulator - 4_WVGA_Nexus_S_API_29:5554
3:35
Prise de note
Nouvelle note
Titre
Texte
VALIDER

Quelques modifications

- Dans manifest.xml, changer l'id de l'application dans l'attribut packages de la balise <manifest>
- Dans build.gradle (Module: app) changer également l'id et éventuellement la version
- Dans res/values/string.xml, changer la valeur de app_name

SDK Manager

Preferences for New Projects

- Appearance & Behavior
  - Appearance
  - Menus and Toolbars
  - System Settings
    - Passwords
    - HTTP Proxy
    - Data Sharing
    - Updates
    - Memory Settings
    - Android SDK
    - Notifications
    - Quick Lists
    - Path Variables
- Keymap
- Editor
- Plugins 1
- Build, Execution, Deployment
- Kotlin
- Tools

Appearance & Behavior › System Settings › Android SDK

Manager for the Android SDK and Tools used by Android Studio

Android SDK Location: /Users/seb/frameworks/android-sdk Edit Optimize disk space

SDK Platforms | SDK Tools | SDK Update Sites

Each Android SDK Platform package includes the Android platform and sources pertaining to an API level by default. Once installed, Android Studio will automatically check for updates. Check "show package details" to display individual SDK components.

Name	API Level	Revision	Status
Android API 30			
Android SDK Platform 30	30	1	Not installed
Google APIs Intel x86 Atom System Image	30	5	Not installed
Google APIs Intel x86 Atom_64 System Image	30	5	Not installed
Google Play Intel x86 Atom System Image	30	5	Not installed
Google Play Intel x86 Atom_64 System Image	30	5	Not installed
Android 10.0 (Q)			
Android SDK Platform 29	29	4	Installed
Sources for Android 29	29	1	Not installed
Intel x86 Atom System Image	29	7	Not installed
Intel x86 Atom_64 System Image	29	7	Not installed
Google APIs Intel x86 Atom System Image	29	10	Not installed
Google APIs Intel x86 Atom_64 System Image	29	10	Not installed
Google Play Intel x86 Atom System Image	29	8	Installed
Google Play Intel x86 Atom_64 System Image	29	8	Not installed
Android Q Preview			
Android TV Intel x86 Atom System Image	Q	1	Not installed
Android 9.0 (Pie)			
Android SDK Platform 28	28	6	Installed
Sources for Android 28	28	1	Not installed
Android TV Intel x86 Atom System Image	28	9	Not installed

☑ Hide Obsolete Packages ☑ Show Package Details

Cancel Apply OK

SDK Manager

Preferences for New Projects

- Appearance & Behavior
  - Appearance
  - Menus and Toolbars
  - System Settings
    - Passwords
    - HTTP Proxy
    - Data Sharing
    - Updates
    - Memory Settings
    - Android SDK
    - Notifications
    - Quick Lists
    - Path Variables
- Keymap
- Editor
- Plugins (1)
- Build, Execution, Deployment
- Kotlin
- Tools

Appearance & Behavior › System Settings › Android SDK

Manager for the Android SDK and Tools used by Android Studio

Android SDK Location: /Users/seb/frameworks/android-sdk Edit Optimize disk space

SDK Platforms | SDK Tools | SDK Update Sites

Below are the available SDK developer tools. Once installed, Android Studio will automatically check for updates. Check "show package details" to display available versions of an SDK Tool.

Name	Version	Status
[-] Android SDK Build-Tools 30		Update Available: 30.0.0
[ ] NDK (Side by side)		Not Installed
[ ] Android SDK Command-line Tools (latest)		Not Installed
[ ] CMake		Not Installed
[x] Android Auto API Simulators	1.0.0	Installed
[ ] Android Auto Desktop Head Unit emulator	1.1	Not installed
[x] Android Emulator	30.0.12	Installed
[-] Android SDK Platform-Tools	29.0.2	Update Available: 30.0.3
[x] Android SDK Tools	26.1.1	Installed
[x] Android Support Library, rev 23.2	23.2.0	Installed
[x] Documentation for Android SDK	1	Installed
[x] Google Play APK Expansion library	1.0.0	Installed
[ ] Google Play Licensing Library	1	Not installed
[-] Google Play services	48	Update Available: 49
[x] Google Web Driver	2	Installed
[-] Instant Apps Development SDK	1.1.0	Update Available: 1.9.0
[x] Intel x86 Emulator Accelerator (HAXM installer)	7.5.1	Installed
[ ] Skia Parser Server	1	Not installed

[x] Hide Obsolete Packages [ ] Show Package Details

? | Cancel | Apply | OK

SDK Manager

Preferences for New Projects

- Appearance & Behavior
  - Appearance
  - Menus and Toolbars
  - System Settings
    - Passwords
    - HTTP Proxy
    - Data Sharing
    - Updates
    - Memory Settings
    - Android SDK
    - Notifications
    - Quick Lists
    - Path Variables
- Keymap
- Editor
- Plugins 1
- Build, Execution, Deployment
- Kotlin
- Tools

Appearance & Behavior › System Settings › Android SDK

Manager for the Android SDK and Tools used by Android Studio

Android SDK Location: /Users/seb/frameworks/android-sdk Edit Optimize disk space

SDK Platforms | SDK Tools | SDK Update Sites

Below are the available SDK developer tools. Once installed, Android Studio will automatically check for updates. Check "show package details" to display available versions of an SDK Tool.

Name	Version	Status
Android SDK Build-Tools 30		
30.0.0	30.0.0	Not installed
29.0.3	29.0.3	Installed
29.0.2	29.0.2	Not installed
29.0.1	29.0.1	Not installed
29.0.0	29.0.0	Not installed
28.0.3	28.0.3	Installed
28.0.2	28.0.2	Not installed
28.0.1	28.0.1	Not installed
28.0.0	28.0.0	Not installed
27.0.3	27.0.3	Not installed
27.0.2	27.0.2	Not installed
27.0.1	27.0.1	Not installed
27.0.0	27.0.0	Not installed
26.0.3	26.0.3	Not installed
26.0.2	26.0.2	Not installed
26.0.1	26.0.1	Not installed
26.0.0	26.0.0	Not installed
25.0.3	25.0.3	Not installed
25.0.2	25.0.2	Not installed
25.0.1	25.0.1	Not installed

Hide Obsolete Packages Show Package Details

Cancel Apply OK

AVD Manager

Virtual Device Configuration

Select Hardware

Choose a device definition

Category	Name	Play Store	Size	Resolution	Density
TV	Pixel XL		5.5"	1440x2...	560dpi
Phone	Pixel 3a XL		6.0"	1080x2...	400dpi
Wear OS	Pixel 3a		5.6"	1080x2...	440dpi
Tablet	Pixel 3 XL		6.3"	1440x2...	560dpi
Automotive	Pixel 3		5.46"	1080x2...	440dpi
	Pixel 2 XL		5.99"	1440x2...	560dpi
	Pixel 2		5.0"	1080x1...	420dpi

Pixel 2

1080px

5.0"

1920px

Size: large
Ratio: long
Density: 420dpi

New Hardware Profile | Import Hardware Profiles | Clone Device...

Cancel | Previous | Next | Finish

Choix de l'image

Virtual Device Configuration

System Image

Select a system image

Recommended | x86 Images | Other Images

Release Name	API Level	ABI	Target
API 30 Download	30	x86	Android API 30 (Google APIs)
Q Download	29	x86	Android 10.0 (Google APIs)
Oreo Download	27	x86	Android 8.1 (Google APIs)
Oreo Download	26	x86	Android 8.0 (Google APIs)
Nougat Download	25	x86	Android 7.1.1 (Google APIs)
Nougat Download	24	x86	Android 7.0 (Google APIs)
Marshmallow Download	23	x86	Android 6.0 (Google APIs)
Lollipop Download	22	x86	Android 5.1 (Google APIs)

API Level
30

Android
Google Inc.

System Image
x86

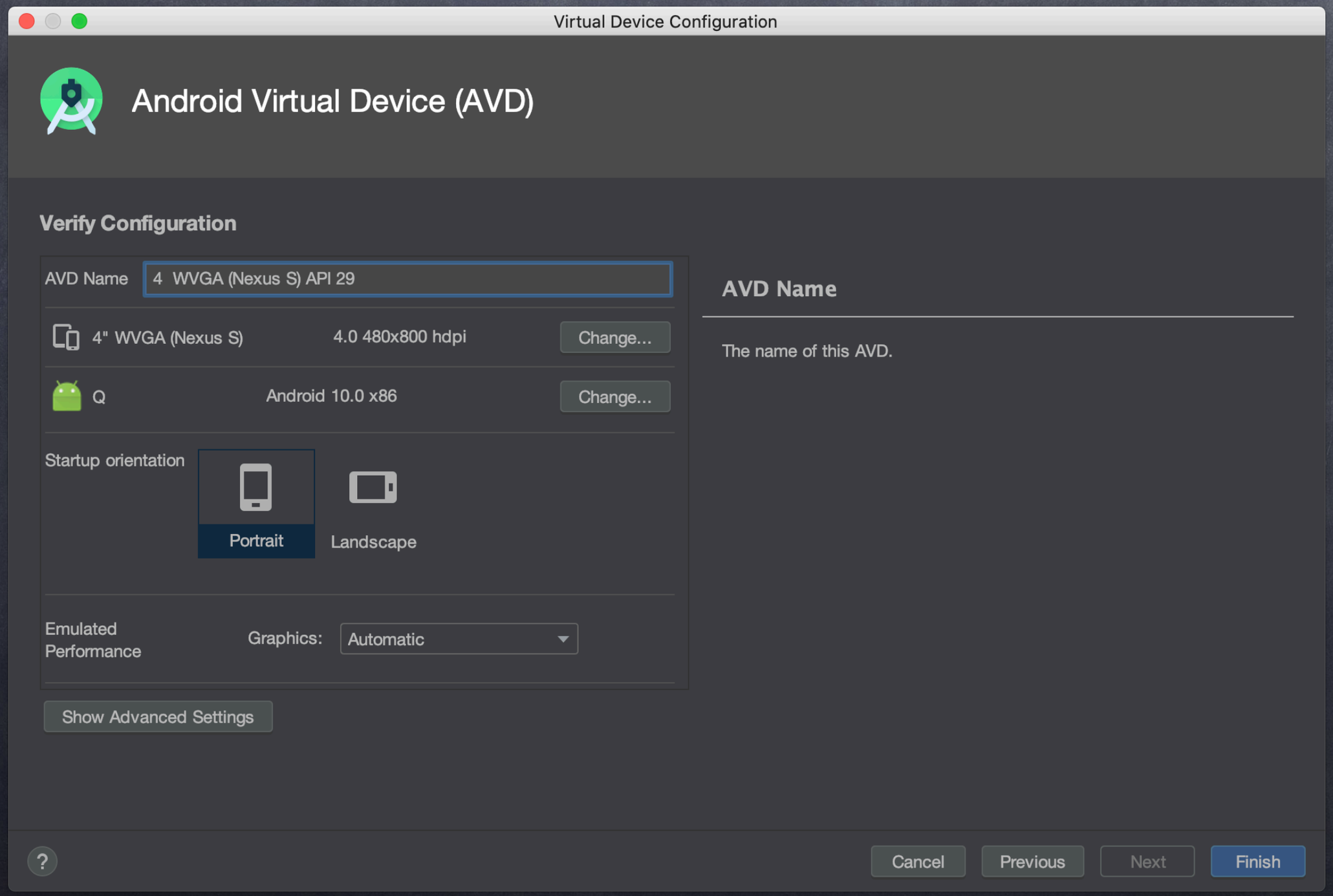
We recommend these images because they run the fastest and support Google APIs.

Questions on API level?
See the API level distribution chart

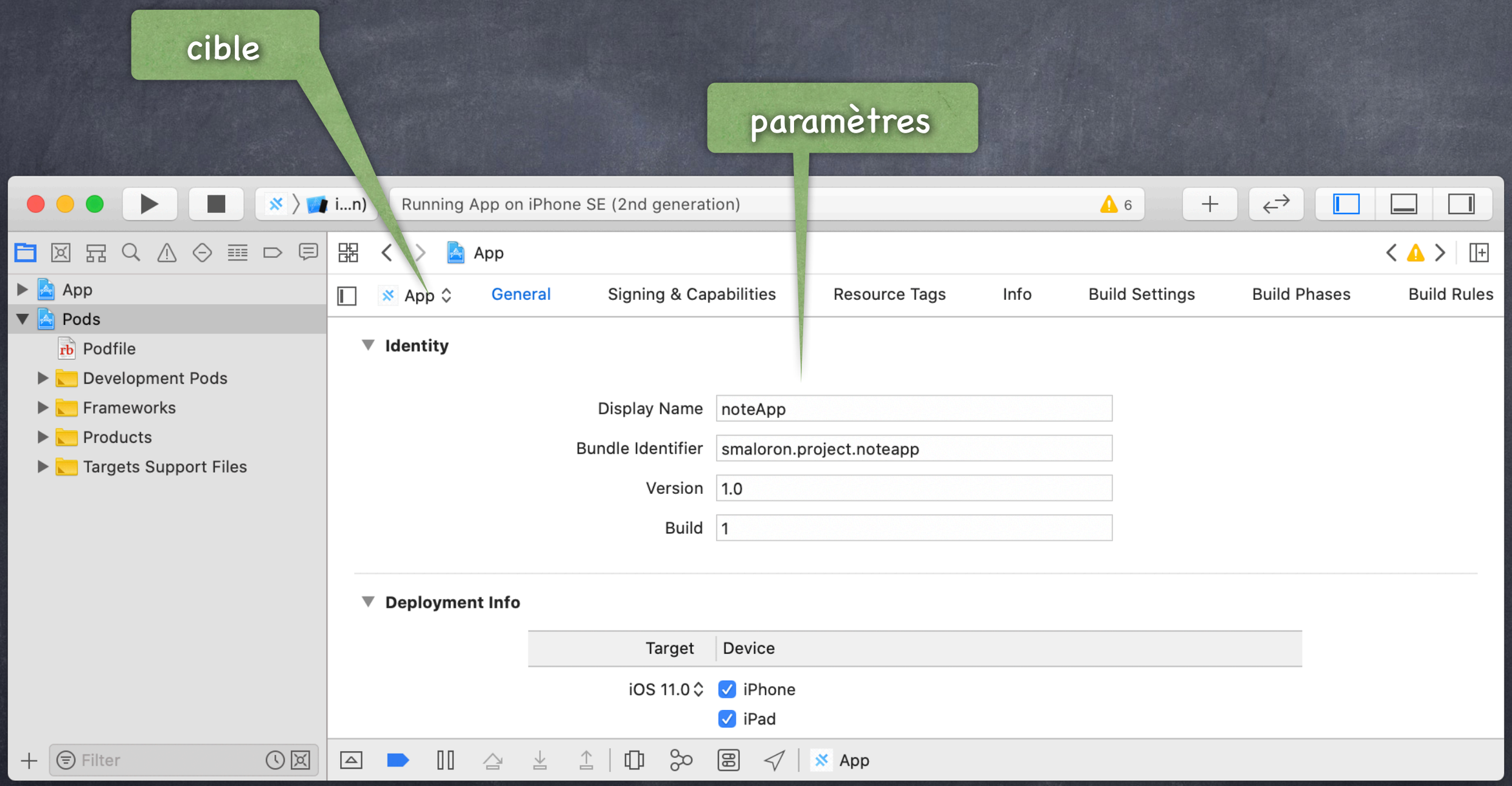
A system image must be selected to continue.

Cancel | Previous | Next | Finish

Configuration



iOS


cible
paramètres
Running App on iPhone SE (2nd generation)
App
Pods
Podfile
Development Pods
Frameworks
Products
Targets Support Files
General
Signing & Capabilities
Resource Tags
Info
Build Settings
Build Phases
Build Rules
Identity
Display Name
noteApp
Bundle Identifier
smaloron.project.noteapp
Version
1.0
Build
1
Deployment Info
Target
Device
iOS 11.0
iPhone
iPad
Filter

Lancer la compilation

Choisir le téléphone

App 〉 iPhone SE (2nd generation) | App | Build App: **Succeeded** | Today at 12:39

Find 〉 Text 〉 Containing

Text

In Workspace | Ignoring Case

Filter

App 〉 App 〉 capacitor.config.json 〉 No Selection

```
1  {
2    "appId": "smaloron.project.noteapp",
3    "appName": "noteApp",
4    "bundledWebRuntime": false,
5    "npmClient": "npm",
6    "webDir": "www",
7    "plugins": {
8      "SplashScreen": {
9        "launchShowDuration": 0
10     }
11   },
12   "cordova": {}
```

Auto | Filter | All Output

iPhone SE (2nd generation) (13...

Carrier 12:55 PM

Prise de note

Nouvelle note

Titre

Texte

Valider

attention

Alertes et infos de debug

json server

simulation de requête http

Principes

- Création d'une API REST en 30 secondes
- La source de données est un fichier JSON

Installation

```
npm install -g json-server
```

La source de données

Le nom de la collection doit être au pluriel

id obligatoire

la clef étrangère repend le nom de la collection au singulier et ajoute Id

```
{
  "posts": [
    {
      "id": 1, "title": "Ionic c'est super",
      "userId": 1, "createdAt": "2018-01-01"
    },
  ],
  "comments": [
    {
      "id": 1, "body": "Vraiment super", "postId": 1
    }
  ],
  "users": [{ "id":1, "name": "seb", "role": "user"}]
}
```

Lancement du serveur

```
json-server --watch db.json
```

Lance un serveur sur http://localhost:3000

Les routes

verbe	action	url
GET	lister tous les posts	/posts
GET	obtenir le post dont l'id est 1	/posts/1
POST	ajouter un post	
DELETE	supprimer un post	/posts/1
PUT	modifier un post	/posts/1

Les filtres

action	url
tous les posts d'un auteur	/posts?authorId=1
multi critères	/posts?authorId=1&createdAt= 2018-01-01
filtre sur une relation	/posts?author.name=seb
filtre sur plusieurs id	/posts?id=1&id=5

Pagination

- Le paramètre _page permet d'accéder à une page donnée.
- Par défaut 10 résultats par page
- Le nombre de résultats est fixé par le paramètre _limit

```
//Affiche les articles de 11 à 20
http://posts?_page=2

//Affiche les articles de 21 à 40
http://posts?_page=2&_limit=20
```

Tri

- Les critères de tris sont gérés par le paramètre _sort
- L'ordre de tri est géré par le paramètre _order

```
//Trie les résultats par id en ordre descendant
http://posts?_sort=id&_order=desc

//Trie les résultats par authorId et id
http://posts?_sort=authorId,id&_order=desc,asc
```

Les relations

action	url
Obtenir les données d'une collection de références	/posts?_embed=comments
Obtenir les données d'une référence	/posts?_expand=author
Combinaison	/posts?_embed=comments&_expand=author
Tous les commentaires du post id=1	/posts/1/comments

HTTP

Principes

- Requête asynchrone
- Retourne un Observable
  (même principe qu'une promesse)

Mise en place

- Importer la bibliothèque HttpClientModule depuis @angular/common/http (clef imports dans app.module.ts)
- Injecter une instance de HttpClient dans le constructeur de la page
- Utiliser la méthode subscribe pour résoudre l'appel asynchrone

La méthode subscribe

```
this.http.get(url).subscribe(
  //Succès
   (response)=> {
     console.log(response);
     console.log(response.json());
   },
  //Echec
  (err) => {
    console.log(err);
  },
  //Completion en cas de succès uniquement
  ()=>{
   console.log('fin de la requête')
  }
);
```

Utilisation simple

```
import { HttpClient} from '@angular/common/http';

  …

@Component({
  selector: 'page-home',
  templateUrl: 'home.html'
})
export class HomePage {
  private url = "https://randomuser.me/api";
  public userList;

  constructor(public http:HttpClient) { loadUsers(); }

  loadUsers(){
    let url = this.url + '?results=20';

    this.http.get(url).subscribe(
      (response)=> {
        this.userList = response.results;
      }
    );
  }
}
```

Avec passage de paramètres

```
import { HttpClient, HttpParams} from '@angular/common/http';
  …

export class HomePage {
  private url = "https://randomuser.me/api";
  private params: HttpParams;
  public userList;

  constructor(public http:HttpClient) {
    this.params = new HttpParams()
                  .set('results', '20');
    loadUsers();
  }

  loadUsers(){
    this.http.get(this.url, {params: this.params})
      .subscribe(
        (data)=> {
          this.userList = data.results;
        }
      );
  }
}
```

Attention, objet immuable, les setters doivent être invoqués dans la même instruction lors de l'instanciation

Affichage asynchrone

Au niveau du contrôleur on ne fait que définir un objet de type Observable, il n'y a pas de "subscribe" pour résoudre la requête asynchrone

Il faut également importer la bibliothèque Observable

```
import { HttpClient} from '@angular/common/http';
import {Observable} from 'rxjs/Observable';
  …

export class HomePage {
  private url = "https://randomuser.me/api?results=10";
  private request:Observable<any>

  constructor(public http:HttpClient) {
   this.request = http.get(this.url);
  }
}
```

Au niveau de la vue on affiche les données en ajoutant le pipe "async" pour indiquer que l'affichage doit être différé.

Le "?" permet d'éviter une erreur lorsque "request" est undefined et donc ne possède pas de clef results

```
<ion-item *ngFor="let user of (request|async)?.results">
  <p>{{user.name.first}} {{user.name.last}}
</ion-item>
```

Poster des données

```
import { HttpClient} from '@angular/common/http';
  …

export class HomePage {

  private url = "https://mon-api/user";

  constructor(public http:HttpClient) {}

  saveUser(userData){
    let postData = {
       name: 'userName', id: new Date().getTime()
    };

    this.http.post(this.url, postData)
       .subscribe(
         (data)=> {
          console.log(data);
         }
     );
  }
}
```

Les données sont postées sous la forme d'un objet json

Transmettre un en-tête http

```
import { HttpClient, HttpHeaders}
        from '@angular/common/http';
  …

export class HomePage {

  private url = "https://mon-api/user";

  constructor(public http:Http) {}

  deleteUser(userData){
    let postData = {id: userData.id};

    let headers = new HttpHeaders()
                  .set('Authorization': 'XXXXXXXXX');

  this.http.delete(this.url, postData, {header: headers})
     .subscribe(
        (data)=> {
          console.log(data);
        }
     );
  }
}
```

Attention, objet immuable, les setters doivent être invoqués dans la même instruction lors de l'instanciation

Infinite scroll

La vue

```
<ion-content>

  <ion-list>…</ion-list>

  <ion-infinite-scroll (ionInfinite)="loadMore($event)">
    <ion-infinite-scroll-content
      loadingSpinner="bubbles"
      loadingText="Loading more data...">
    </ion-infinite-scroll-content>
  </ion-infinite-scroll>

</ion-content>
```

Le contrôleur

```
loadMore(infiniteScroll){

    this.http.get(this.url)
      .subscribe(
        (data)=> {
          //Chargement de nouvelles données
          this.userList = this.userList.concat(data.results);

          //Envoi d'un message pour indiquer la fin du chargement
          infiniteScroll.target.complete();
        }
      );
  }
```

Pull to refresh

La vue

```
<ion-content>

  <ion-list>…</ion-list>

<ion-refresher (ionRefresh)="doRefresh($event)">
    <ion-refresher-content
      pullingIcon="arrow-dropdown"
        pullingText="Pull to refresh"
        refreshingSpinner="circles"
        refreshingText="Refreshing...">
    </ion-refresher-content>
  </ion-refresher>

</ion-content>
```

Le contrôleur

```
doRefresh(refresher){

    this.http.get(this.url)
       .subscribe(
        (data)=> {
          //Chargement de nouvelles données
           this.userList = data.results.concat(this.userList);

          //Envoi d'un message pour indiquer la fin du chargement
          refresher.target.complete();
        }
     );
}
```

SQLite plugin

Accès à la base de donnée embarquée du périphérique

Installation

```
npm install --save @capacitor/core @capacitor/cli
```

Les méthodes

open({database:string}): Promise<{result:boolean,message:string}>	Ouvre une bd existante ou en créé une nouvelle.
close({database:string}): Promise<{result:boolean,message:string}>	Ferme la connexion à une base de données
execute({statements:string}): Promise<{changes:{changes:number}, message:string}>	Exécute une série de requêtes sans paramètres
run({statement:string,values:[]}) => Promise<{changes: {changes:number,lastId:number},message:string}>	Exécute une seule requête qui peut être paramètrée
query({statement:string,values:[]}) => Promise<{values:Array<any>, message:string}>	Exécute une requête SELECT qui peut être paramètrée

Exemple simple

```
import {Component, OnInit} from '@angular/core';
import {Plugins} from '@capacitor/core';

const {CapacitorSQLite} = Plugins;

@Component({
    selector: 'app-test-sqlite',
    templateUrl: './test-sqlite.page.html',
    styleUrls: ['./test-sqlite.page.scss'],
})
export class TestSqlitePage implements OnInit {

    constructor() {
    }

    ngOnInit() {
        this.testDB();
    }

    private async testDB() {
        const connectionOK = await this.openDB();

        if (connectionOK) {
            await this.createTable();

            await this.wipeData();

            await this.insertData(['joe@mail.com', 'Joe User', 34]);
            await this.insertData(['alice@mail.com', 'Alice la malice', 27]);

            await this.getUsers();

        }
    }
// ... suite
```

La connexion

```
private async openDB() {
    const connection = await CapacitorSQLite.open({database: 'test'});
    if (connection.result) {
        console.log('ouverture de la base ok');
    } else {
        console.error(connection.message);
    }

    return connection.result;
}
```

Création de la table

```
private async createTable() {
 const sql = `
    BEGIN TRANSACTION;
    CREATE TABLE IF NOT EXISTS users (
      id INTEGER PRIMARY KEY NOT NULL,
      email TEXT UNIQUE NOT NULL,
      name TEXT,
      age INTEGER
    );
    PRAGMA user_version = 1;
    COMMIT TRANSACTION;
    `;
 const queryResult = await CapacitorSQLite
                          .execute({statements: sql}
 );
 console.log(queryResult);
}
```

Suppression des données

```
private async wipeData(){
    const sql = 'DELETE FROM users;';
    const queryResult = await CapacitorSQLite
                                .run({statement: sql, values: []});

    console.log(queryResult);
}
```

Insertion des données

```
private async insertData(data: string[]) {
    const sql = `INSERT INTO users (email, name, age)
                                VALUES (?,    ?,    ?);`;
    const queryResult = await CapacitorSQLite
                                .run({statement: sql, values: data});

    console.log(queryResult);
}
```

Récupération des données

```
private async getUsers() {
    const sql = 'SELECT * FROM users;';
    const queryResult = await CapacitorSQLite
                                .query({statement: sql, values: []});

    console.log(queryResult);
}
```

Récupération d'un utilisateur

```
private async getOneUserById(id: number) {
    const sql = 'SELECT * FROM users WHERE id=?;';
    const queryResult = await CapacitorSQLite
                                .query({statement: sql, values: [id]});

    console.log(queryResult);
}
```

Camera plugin

Installation

```
npm install --save @capacitor/core @capacitor/cli
```

Pour tester dans un navigateur web

```
npm install --save @ionic/pwa-elements
```

```
import { enableProdMode } from '@angular/core';
import { platformBrowserDynamic } from '@angular/platform-browser-dynamic';
import { AppModule } from './app/app.module';
import { environment } from './environments/environment';

import { defineCustomElements } from '@ionic/pwa-elements/loader';

if (environment.production) {
  enableProdMode();
}

platformBrowserDynamic().bootstrapModule(AppModule)
  .catch(err => console.log(err));

defineCustomElements(window)
    .catch(err => console.log(err));
```

main.ts

home.page.ts

```
import {Component} from '@angular/core';
import {
    CameraOptions, CameraResultType,
    CameraSource, Plugins
} from '@capacitor/core';

const {Camera} = Plugins;

@Component({
    selector: 'app-home',
    templateUrl: 'home.page.html',
    styleUrls: ['home.page.scss'],
})
export class HomePage {

    public imgData;

    constructor() {
    }

    async cliclac() {
        const options: CameraOptions = {
            resultType: CameraResultType.Base64,
            source: CameraSource.Camera,
            quality: 70,
            correctOrientation: true
        };

        this.imgData = await Camera.getPhoto(options);
    }

    getImgSource() {
        return '' + this.imgData.base64String;
    }
}
```

home.page.html

```
<ion-content [fullscreen]="true">
  <ion-button (click)="cliclac()" expand="block">
    Prendre une photo
  </ion-button>

  <ion-img *ngIf="imgData" [src]="getImgSource()"></ion-
img>
</ion-content>
```

Geolocation

Installation

```
npm install --save @capacitor/core @capacitor/cli
```

home.page.ts

```
import {Component} from '@angular/core';
import {
    GeolocationOptions,
    GeolocationPosition, Plugins
} from '@capacitor/core';

const {Geolocation} = Plugins;

@Component({
    selector: 'app-home',
    templateUrl: 'home.page.html',
    styleUrls: ['home.page.scss'],
})
export class HomePage {

    position: GeolocationPosition;

    private options: GeolocationOptions;

    constructor() {
        this.options = {
            enableHighAccuracy: true
        };
    }

    async getPosition() {
        this.position = await Geolocation
            .getCurrentPosition(this.options);
        console.log(this.position.coords);
    }

    getDate() {
        return new Date(this.position.timestamp);
    }
}
```

Les imports

Options de Géolocalisation

La géolocalisation est asynchrone

Conversion du timestamp en objet Date

home.page.html

```
<ion-content [fullscreen]="true">
  <ion-button (click)="getPosition()">
      Obtenir la position
  </ion-button>

  <div *ngIf="position">
    <ion-item>Latitude {{ position.coords.latitude }}</ion-item>
    <ion-item>Longitude {{ position.coords.longitude }}</ion-item>
    <ion-item>horodatage {{ getDate() | date: 'hh:mm:ss' }}</ion-item>
  </div>

</ion-content>
```

Cartographie sans Google

Installation

```
npm install --save leaflet
```

La documentation de Leaflet

Styles

Ajouter au fichier global.scss

```
@import "~leaflet/dist/leaflet.css";
```

Ajouter au fichier scss
de la page

```
#mapView {
  width: 100%;
  height: 100%;
}
```

La vue

```
<ion-header>
  <ion-toolbar>
    <ion-title>
      Blank
    </ion-title>
  </ion-toolbar>
</ion-header>

<ion-content>
  <div id="mapView"></div>
</ion-content>
```

Le contrôleur

```
import { Map, tileLayer } from 'leaflet';
// ...
export class HomePage implements OnDestroy{

  public map: Map;

  constructor() {}

  ionViewDidEnter(){
    this.leafLetInit();
  }

  leafLetInit(){
    this.map = new Map('mapView');
    this.map.setView([33.6396965, -84.4304574], 23);

    tileLayer('https://{s}.tile.openstreetmap.org/{z}/{x}/{y}.png', {
      attribution: '&copy; <a href="https://www.openstreetmap.org/copyright">OpenStreetMap</a> contributors'
    }).addTo(this.map);

    this.map.on('click', (even) => console.log(even));

  }

  ngOnDestroy(): void {
    this.map.remove();
  }
}
```

Il faut que le DOM soit chargé donc pas ngOnInit

l'id du conteneur dans la vue

centrage de la carte et zoom initial

Le fond de carte

Un clic sur la carte

Components

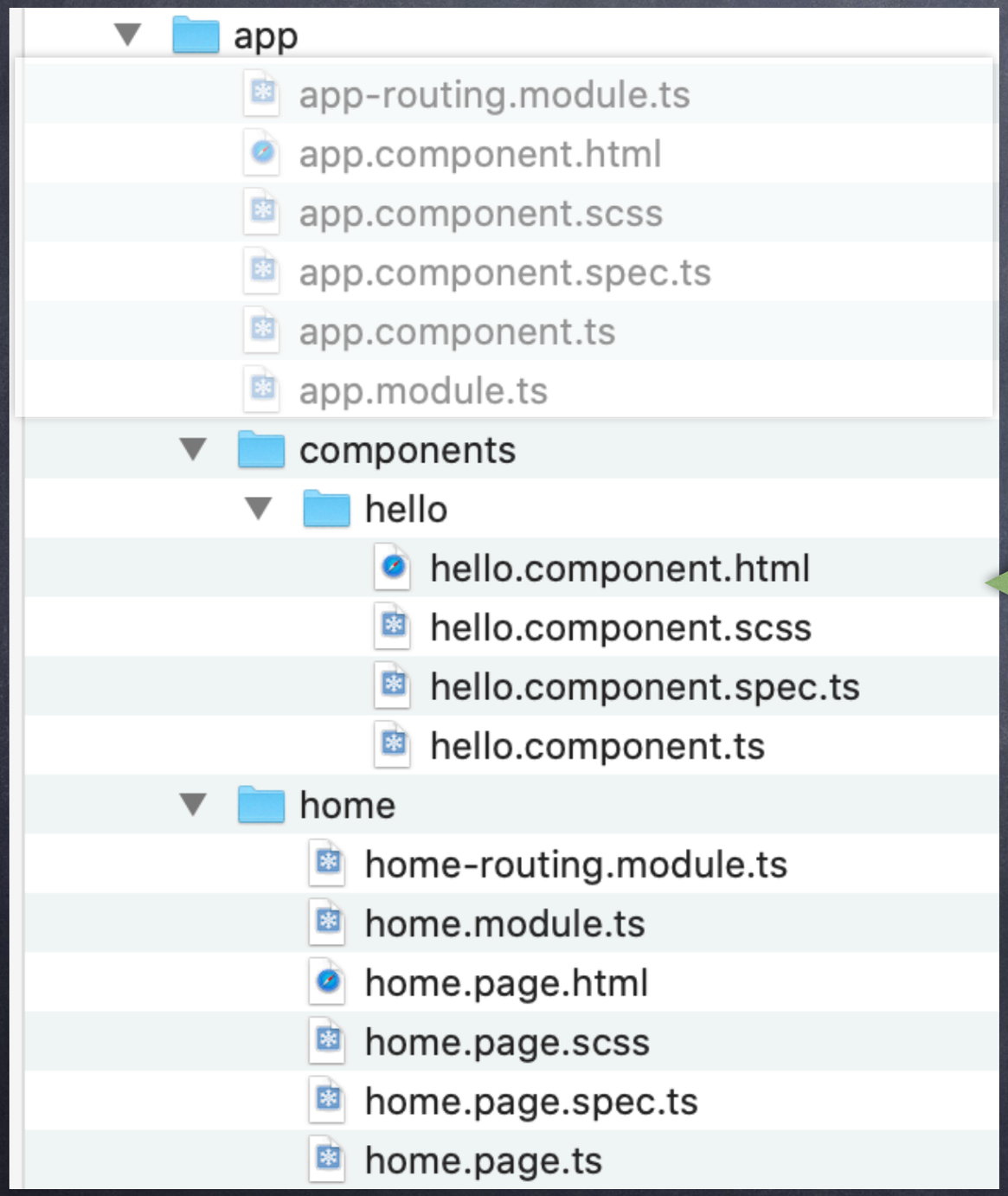
Création

```
ionic g component [chemin et nom du component]
```

```
ionic g component components/hello
```

Créé un dossier hello dans le dossier components qui contient l'ensemble des fichiers du composant dont une classe HelloComponent

Les fichiers



Un component ressemble beaucoup à une page sans module ni routage

Le code

```
import { Component, OnInit } from '@angular/core';

@Component({
  selector: 'app-hello',
  templateUrl: './hello.component.html',
  styleUrls: ['./hello.component.scss'],
})
export class HelloComponent implements OnInit {

  constructor() { }

  ngOnInit() {}
}
```

nom de la balise pour insérer le composant dans une page

hello.component.ts

```
<p>
  hello works!
</p>
```

hello.component.html

Déclaration dans un module

```
import { NgModule } from '@angular/core';
import { CommonModule } from '@angular/common';
import { IonicModule } from '@ionic/angular';
import { FormsModule } from '@angular/forms';
import { HomePage } from './home.page';

import { HomePageRoutingModule } from './home-routing.module';
import {HelloComponent} from '../components/hello/hello.component';


@NgModule({
  imports: [
    CommonModule,
    FormsModule,
    IonicModule,
    HomePageRoutingModule
  ],
  declarations: [HomePage, HelloComponent]
})
export class HomePageModule {}
```

home.module.ts

Utilisation dans une vue

```
<ion-header [translucent]="true">
  <ion-toolbar>
    <ion-title>
      Blank
    </ion-title>
  </ion-toolbar>
</ion-header>

<ion-content class="ion-padding">

  <app-hello></app-hello>

</ion-content>
```

Ajout d'un paramètre

```
import {Component, Input, OnInit} from '@angular/core';

@Component({
    selector: 'app-hello',
    templateUrl: './hello.component.html',
    styleUrls: ['./hello.component.scss'],
})
export class HelloComponent implements OnInit {

    @Input() name: string;

    constructor() {
    }

    ngOnInit() {
    }
}
```

hello.component.ts

```
<p>
  hello {{ name }}
</p>
```

hello.component.html

Utilisation dans une vue

```
<ion-header [translucent]="true">
  <ion-toolbar>
    <ion-title>
      Blank
    </ion-title>
  </ion-toolbar>
</ion-header>

<ion-content class="ion-padding">

  <app-hello name="Seb">
  </app-hello>

</ion-content>
```

Ajout d'une méthode

```
import {Component, EventEmitter, Input, Output, OnInit} from
'@angular/core';

@Component({
    selector: 'app-hello',
    templateUrl: './hello.component.html',
    styleUrls: ['./hello.component.scss'],
})
export class HelloComponent implements OnInit {

    @Input() name: string;

    @Output() helloClick: EventEmitter<number>;

    private nbClicks = 0;

    constructor() {
        this.helloClick = new EventEmitter<number>();
    }

    ngOnInit() {
    }

    private personClick() {
        this.nbClicks++;
        this.helloClick.emit(this.nbClicks);
    }
}
```

hello.component.ts

```
<p (click)="personClick()">
  hello {{ name }}
</p>
```

hello.component.html

Utilisation dans une vue

```
<ion-header [translucent]="true">
  <ion-toolbar>
    <ion-title>
      Blank
    </ion-title>
  </ion-toolbar>
</ion-header>

<ion-content class="ion-padding">

  <app-hello name="Seb"
     (helloClick)="clickHandler($event)">
  </app-hello>

</ion-content>
```

home.page.html

Le contrôleur

```
import { Component } from '@angular/core';

@Component({
  selector: 'app-home',
  templateUrl: 'home.page.html',
  styleUrls: ['home.page.scss'],
})
export class HomePage {

  constructor() {}

  clickHandler(event){
    console.log(event);
  }
}
```

home.page.ts

Un composant star

Le code du composant

```
import {Component, OnInit, Input, Output, EventEmitter} from '@angular/core';

@Component({
    selector: 'app-star',
    templateUrl: './star.component.html',
    styleUrls: ['./star.component.scss'],
})
export class StarComponent implements OnInit {

    @Input() max = 5;
    @Input() activeColor = 'danger';
    @Input() inactiveColor = 'medium';

    @Output() starValueChange: EventEmitter<number>;

    @Input() starValue;

    values: Array<number> = [];

    constructor() {
        this.starValueChange = new EventEmitter<number>();
    }

    ngOnInit() {
        for (let i = 1; i <= this.max; i++) {
            this.values.push(i);
        }
    }

    setStarValue(n) {
        this.starValue = n;
        this.starValueChange.emit(n);
    }

}
```

les inputs

Le nom de l'output doit être le nom de la propriété suivi de "Change"

Création du tableau des valeurs

Modification de la valeur et transmission de l'événement

star.component.ts

la vue du composant

```
<ion-label>
  <ion-icon *ngFor="let n of values"
            name="star"
            [color]="n<=starValue? activeColor: inactiveColor"
            (click)="setStarValue(n)"></ion-icon>
</ion-label>
```

star.component.html

Déclaration dans le module de la page

```
import { CommonModule } from '@angular/common';
import { IonicModule } from '@ionic/angular';
import { FormsModule } from '@angular/forms';
import { HomePage } from './home.page';

import { HomePageRoutingModule } from './home-routing.module';
import {HelloComponent} from '../components/hello/hello.component';
import {StarComponent} from '../components/star/star.component';



@NgModule({
  imports: [
    CommonModule,
    FormsModule,
    IonicModule,
    HomePageRoutingModule
  ],
  declarations: [HomePage, HelloComponent, StarComponent]
})
export class HomePageModule {}
```

home.module.ts

Le code de la page

```
import { Component } from '@angular/core';

@Component({
  selector: 'app-home',
  templateUrl: 'home.page.html',
  styleUrls: ['home.page.scss'],
})
export class HomePage {

  rating = 3;

  constructor() {}

  clickHandler(event){
    console.log(event);
  }
}
```

home.page.ts

La vue

home.page.html

```
<ion-content class="ion-padding">

  <app-hello name="Seb" (helloClick)="clickHandler($event)"></app-hello>

  <app-star max="6" [(starValue)]="rating"></app-star>

  <p>{{ rating }}</p>

</ion-content>
```

Liaison bidirectionnelle entre le composant et la page.
On a créé notre propre ngModel