Activitat d’avaluació 1.3 - Crea una pantalla a partir del disseny

Documentació del desenvolupament

Inici de l’aplicació

L’aplicació va iniciar-se desenvolupant els components de la carpeta *styles*, amb l’objectiu d’obtenir la classe *app_styles* demanada, amb la intenció d’obtenir un codi aprofitable per a futurs desenvolupaments. No és probablement el punt habitual per a començar un programa, però recordem que és un programa d’interfície d’usuari, i la base ha de ser sòlida.

> En realitat la meva història laboral dels últims vint anys està principalment centrada en aspectes de *backend* i arquitectura, amb només un contacte mínim amb les capes de *frontend* pels aspectes relacionats amb l’arquitectura de les aplicacions; a més, tampoc els aspectes bàsics relacionats amb el disseny són una de les meves aptituds, i per aquest motiu trobo força interessant l’aproximació de *Flutter* que ofereix els ***temes***.
>
> *Text 1: Context personal*

En classe s’ha objectat que en la pràctica els temes no s’ajusten al llibre d’estils que els dissenyadors estableixen en les empreses que els fan servir, i ***es requereix establir valors constants específics fixats pel llibre d’estils***, en comptes de fer servir un tema de *Flutter*.

> Això és efectivament cert, i és un problema en part degut a la manca d’eines específiques per als dissenyadors, que llavors només treballen amb les seves eines de disseny; l’enfocament anterior a *Flutter* va ser la separació física dels components de disseny i els de programació, i s’han fet intents molt ambiciosos, oblidant de vegades la disponibilitat de les eines més bàsiques. Afortunadament, l’**eina de disseny *Figma*** (i potser alguna altra que desconec) ja compta amb un connector per generar *temes de Flutter*, així com altres per a *HTML*, *React*...
>
> > *Fa uns vint-i-cinc anys els dissenys per a web d’un dissenyador tenien la mida fixa 600x800; es van generalitzar resolucions més altes, i només alguns bons dissenyadors incorporaven múltiples dissenys segons la pantalla (els que en aquella època no feien servir flash, és clar). No sé si foren els dissenyadors desbordats per la gran quantitat de resolucions que estaven fent servir els usuaris, els programadors farts d’aquests dissenys fixos, o els fabricants dels nous dispositius* smart-phones*, però finalment van aparèixer eines per fer el disseny responsiu amb HTML4.*
>
> *Text 2: Observacions personals*

En concret, per a l’exercici s’ens proporciona una paleta estipulada per https://coolors.co/f2f8a0-f896d8-bf63f8-3407da-181528, que ens proporciona un conjunt de cinc colors amb nou tonalitats per a cada color, entre el 100 i el 900 (no hi és el 50, i no entenc alguns sistemes que també treballen amb el valor *zero*:

{ '**mindaro**': { DEFAULT: '#f2f8a0', 100: '#474c06', 200: '#8e980b', 300: '#d6e411', 400: '#e8f254', 500: '#f2f8a0', 600: '#f5f9b3', 700: '#f7fbc6', 800: '#fafcd9', 900: '#fcfeec' }, '**persian_pink**': { DEFAULT: '#f896d8', 100: '#4b0533', 200: '#960a67', 300: '#e00e9a', 400: '#f34bbb', 500: '#f896d8', 600: '#faabdf', 700: '#fbc0e7', 800: '#fcd5ef', 900: '#feeaf7' }, '**heliotrope**': { DEFAULT: '#bf63f8', 100: '#2a0342', 200: '#540684', 300: '#7e09c7', 400: '#a321f5', 500: '#bf63f8', 600: '#cc82f9', 700: '#d8a1fb', 800: '#e5c1fc', 900: '#f2e0fe' }, '**chrysler_blue**': { DEFAULT: '#3407da', 100: '#0a012c', 200: '#150357', 300: '#1f0483', 400: '#2a05ae', 500: '#3407da', 600: '#5021f8', 700: '#7c59fa', 800: '#a790fc', 900: '#d3c8fd' }, '**dark_purple**': { DEFAULT: '#181528', 100: '#050408', 200: '#0a0810', 300: '#0e0d18', 400: '#131120', 500: '#181528', 600: '#3b3463', 700: '#5e539e', 800: '#9289c1', 900: '#c8c4e0' } }

Com a conclusió:

1) Els temes:
    a) Ofereixen una manera senzilla per a ignorants en els aspectes de disseny sense l'asistència d'un dissenyador: triar una combinació de colors compatibles, els modes *dark* / *light*, i també proporciona guies per triar la mida de les fonts.
    b) També és la manera més senzilla per a **establir la família de font per a tota l'aplicació**; fem servir un tema només amb aquesta finalitat.
2) Els dissenys personalitzats que requereixen les empreses requereixen un enfocament més personalitzat. Es construeixen les constants de color fent servir la classe ***MaterialColor***, que permet la definició d'un ***swatch*** (conjunt de tonalitats) per al color:
    a) D'aquesta manera es pot definir la classe *AppStyles* com a una senzilla enumeració de constants, que per conveniència es divideixen en dos apartats (*records*):
        1) *AppStyles.fonts*: un *record* que permet accedir per nom a les diferents fonts requerides; inclou el nom de la font que es farà servir per a tota l'aplicació.
        2) *AppStyles.colors*: un record que proporciona el color *black* i les paletes dels diferents colors (*mindaro*, *persianPink*, *heliotrope*, *chryslerBlue* i *darkPurple*)

El desenvolupament d'aquest apartat ha estat útil per familiaritzar-me amb el components del llenguatge *Dart*, components dels que he llegit i m'han explicat en classe, però en realitat entres a fer-los servir de cap quan fas servir *Flutter*, sense temps per a familiaritzar-te. En realitat gran part del codi que he escrit més tard he llançat, en trobar la manera de fer servir *MaterialColor*, perquè inicialment vaig fallar en construir-lo correctament.

Text 3: Aprofitament personal

Propostes de millora

1. En aquest cas l'enunciat limita el color negre a una única tonalitat; considerar l'opció de tenir **tonalitats també pel color negre** podria ser una opció.
2. Tampoc en cap moment es parla de la **dualitat *dark mode* / *light mode***, que avui dia hi serà en qualsevol llibre d'estils amb un mínim de presència.

- Sense pensar-ho gaire, sense ser coneixedor d'aquests temes, i sense cap prova de concepte, podriem pensar **una opció per canviar entre els modes *dark* i *light*** amb el senzill procediment de canviar tots el valors pel càlcul *1000* - *valor*, mantenint així la paleta i tots els colors; això deixaria la tonalitat per defecte (500) invariant entre els modes *dark* i el *light*; les tonalitats 400 i 600 també s'haurien de considerar invariants, per no tenir prou diferència de tonalitat entre elles.
- S'ha d'insistir que en la pràctica seria una **tasca del dissenyador**, i el llibre d'estils ja hauria d'incorporar ambdós modes, que el dissenyador probablement considerarà com a dos variacions diferents del tema. Ja existeixen eines com a ***Figma*** per als dissenyadors.
- **Fent servir els temes de *Flutter*** es faria servir l'esquema següent:

```
MaterialApp(
      title: 'App Title',
      theme: ThemeData(
        brightness: Brightness.light,
        /* light theme settings */
      ),
      darkTheme: ThemeData(
        brightness: Brightness.dark,
        /* dark theme settings */
      ),
      themeMode: ThemeMode.dark,
      /* ThemeMode.system to follow system theme,
         ThemeMode.light for light theme,
         ThemeMode.dark for dark theme
      */
      debugShowCheckedModeBanner: false,
      home: YourAppHomepage(),
    );
```

- Actualment *Flutter* fa servir *Material-3* de *Google*. Observar que les paletes de color i les ***Material Color Utilities*** (*MCU*) també tenen en consideració aspectes com els dissenys amb **colors corporatius** o l'**accesibilitat**, dues situacions que en els dissenys cal tenir molt presents; de vegades s'oblida l'alt percentatge de **daltonisme** (**superior al 10% entre la població** masculina, tot i que és inferior al 1% en la femenina)

Tipografia

Potser caldria recordar la classificació de les mides de lletra del disseny *Material* de *Google*, que es classifiquen en cinc categories, i podem trobar en *TextTheme*. Cadascuna de les categories incorpora tres mides: *Large*, *Medium* i *Small*, formant un total de quinze diferents mides.

Categories de les mides de lletra de *Material*, per ordre:

1. **Display**: *As the largest text on the screen, display styles are reserved for short, important text or numerals. They work best on large screens*.
   - Entre tT57 i tT36, sense espaiat ni enfatitzat (*FontWeight.w400*).
2. **Headline**: *Headline styles are smaller than display styles. They're best-suited for short, high-emphasis text on smaller screens*.
   - Entre tT32 i tT24, sense espaiat ni enfatitzat (*FontWeight.w400*).

3. **Title**: *Titles are smaller than headline styles and should be used for shorter, medium-emphasis text*.
    - Entre tT22 i tT14, amb espaiat mínim i enfatitzat (*FontWeight.w500*).
4. **Label**: *Label styles are smaller, utilitarian styles, used for areas of the UI such as text inside of components or very small supporting text in the content body, like captions*.
    - Entre tT14 i tT11, amb espaiat més gran que el cas de *Title*, i també amb enfatitzat (*FontWeight.w500*).
5. **Body**: *Body styles are used for longer passages of text*.
    - Entre tT16 i tT12, amb espaiat mínim per millorar llegibilitat (i sense enfatitzat *FontWeight.w400*).

Característiques de la tipografia

S'ha incorporat una funció per a cada tipus de lletra, fent servir noms anàlogs als noms que fan servir els temes de *Material*, amb paràmetres per als valors de *Color*, *FontStyle*, *FontStyle*, *FontWeight* i *TextDecoration*, incorporant valors per defecte per facilitar el seu us.

Importació de la font de Google *Montserrat*

Òbviament, la manera més senzilla d'importar una font de google és fer servir la referència a *Google Fonts*, però si volem fer servir una única font, o, com en aquest cas, una font i la seva variant en cursiva, resulta més convenient incorporar aquesta font directament en el projecte.

Dificultats trobades

L'activitat ha estat útil per familiaritzar-me amb el llenguatge *Dart*, que en general no presenta grans sorpreses, amb la salvetat de no implementar sobrecàrregues i les seves particularitats en les derivacions de classes i interfícies, amb l'afegit del *mixin*. També és novedos fer servir la nomenclatura per determinar la visibilitat, limitada a *privat* i *públic*.; en canvi, no accepta la notació «_» per a la separació de grups de dígits en les xifres, que, amb encert, va introduir Ada, i cada vegada més llenguatges incorporen per facilitar la llegibilitat dels números.

L'entorn de compilació presenta dificultats, particularment quan tens el costum de fer servir entorns avançats. Alguns dels problemes poden ser deguts a la meva versió, doncs treballo amb *Linux Manjaro* (distribució derivada d'*Archlinux*), i el suport per a Linux el limiten a *Debian* i *Ubuntu*.

1. «***Hot Reload***» no mostra molts errors, amagant els **errors que «*només es produeixen en mode debug*»**, la qual cosa fa que no t'adonis del molts problemes fins fer un «*reinici*».
2. Es dona el cas que el «reinici» no salva en disc, tot i que clarament en el butó indica que salva; això estic segur que ***és un problema de la meva distribució***, però també m'ha ocasionat molts inconvenients fins adonar-me del comportament anòmal, perquè el codi que s'executava no era el codi que es veia en el editor; des de la última actualització en aquesta mateixa setmana (Manjaro és una «*rolling-release*») sembla que aquest problema ja no es produeix.
3. El **control de flux de la compilació vinculada a l'editor és molt dolenta**; amb tota la «propaganda» del control dels valors nuls i el «***null-safe***», però resulta que t'obliga a incorporar molts operadors «***!***» perquè el compilador no és capaç de detectar que el valor no és null. En la pràctica això deixa el codi ple d'indicadors que el programador es veu obligat

a escriure, en casos que altres compiladors detecten sense dificultat. Un exemple, on es pot veure que el compilador ha exigit condicions que es dedueixen del control de fluxe:

```
f([x]) {
        if (total.value.$1 == null || profile.goals[total.key]!.$1 == null) {
          if (profile.goals[total.key]!.$1 == null && profile.goals[total.key]!.$2 != null) {
            if (total.value.$2.inMinutes >= profile.goals[total.key]!.$2!.inMinutes) { return 1.0; }
            else { return total.value.$2.inMinutes / profile.goals[total.key]!.$2!.inMinutes; }
          } else { return 0.0; }
        } else {
          if (total.value.$1! >= profile.goals[total.key]!.$1!) { return 1.0; }
          else { return total.value.$1! / profile.goals[total.key]!.$1!; }
        }
}
// En vermell els operadors «!» innecessaris; en aquest tros de codi, el problema s'ha repetit deu vegades.
// En negreta les condicions que els fan innecessaris.
```

4. Un altre problema destacat és que l'editor (*Visual Studio Code*) requereix que no hagi errors en el codi per permetre formatar-lo, i només alguns errors molt lleus escapen a aquesta norma; de fet, la funcionalitat de formatar el codi seleccionat no està operatiu.

Principal dificultat

A més de les «petites» dificultats esmentades, que foren inconvenients vincultats a les eines de treball, sobretot dels dos o tres primers dies, la gran dificultat que trobo és que la explicació referent a la maquetació («*layout*») en realitat escapa força a les explicacions lògiques. He mirat molta documentació i molts tutorials, però encara no tinc clars els mecanismes per a la maquetació, que finalment la solució acaba en «provar» fins trobar una solució.

Una indicació que vaig trobar en la documentació va ser fer servir «*Expanded*» dins de «*Row*» o de «*Column*» per ocupar l'espai, però també em vaig trobar algun cas on es produia un error quan s'incorporava aquest «*Expanded*» (sí, en l'axis principal).

És cert que la gran dificultat venia per la incoporació de la «`SingleChildScrollView`», però tot i conèixer el problema em va ser molt difícil fer la maquetació, i, sobretot, no he tingut del tot clar tots els elements que m'han permés arribar a la solució, i la situació ha estat una mica fustrant.

Codi addicional

El temps de lliurament ha estat molt retardat, però no ha estat degut al codi addicional incorporat, sinó a la principal dificultat que ja he esmentat; el codi addicional ha estat una manera d'escapar de tant en tant dels problemes de la maquetació.