Flutter Pizzeria - TD 3

# Le détail d’une pizza

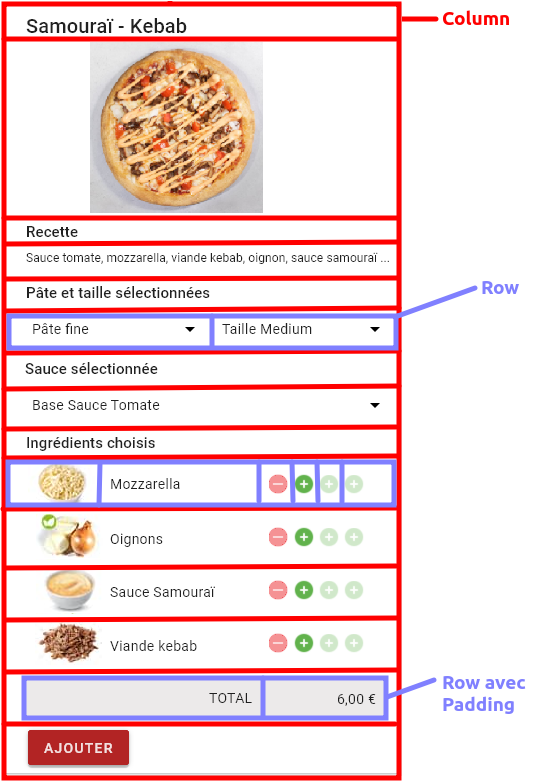
Nous allons nous préoccuper de l'affichage des détails d’une pizza.

## L'affichage d’une pizza

Cette classe va afficher les détails d’une pizza. La classe recevra donc en paramètre une pizza.

### Le design

Le design doit ressembler à ceci :



*Remarque : La partie “Ingrédients choisis” n’est pas traitée dans le TD.*

### Le code de départ

Comme c’est une nouvelle fenêtre, nous allons créer la classe dans le répertoire “ui”.

Dans ce répertoire, créez un nouveau fichier nommé “pizza\_details”.

Ce Widget va être un widget avec état car, plus tard, ses éléments seront modifiés avec l’interaction utilisateur.

Dans le fichier vide, tapez “stful” puis entrée, nommez la classe “PizzaDetails”.

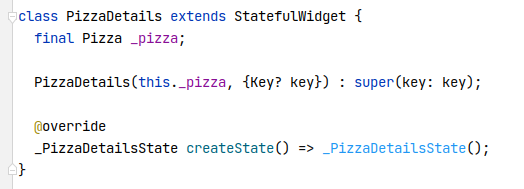
Comme c’est une nouvelle fenêtre, il nous faut ajouter les Widgets Scaffold et AppBar.

Dans un premier temps, nous allons afficher :

* le titre
* l’image
* la recette (exactement, ce que nous avons appelé la garniture)
* la pâte et la taille sélectionnées (mais par les dropdown). Les 2 éléments doivent prendre la même largeur
* la sauce sélectionnée (mais par le dropdown)
* le total (le prix de la pizza pour l’instant)
* le bouton Ajouter identique au TD 2

Le code sera le suivant :

#### Le stateful Widget



Explication :

* Nous passerons la pizza sélectionnée au constructeur de la classe

#### La classe state associée au stateful Widget



Explication :

* Dans le widget Row, la valeur MainAxisAlignment.spaceEvenly permet à tous les éléments d’avoir la même largeur.



Explications:

* Dans le widget Row principal (ligne 3), la valeur MainAxisAlignment.end permet d’aligner les éléments en partant de la fin.
* Le bouton ElevatedButton a pour enfant un widget Row permettant d’aligner un icône, un espacement (SizedBox(width: 5)) et le texte.

Il est préférable de tester visuellement le résultat donc il faut pouvoir afficher cette nouvelle fenêtre.

# 

# La navigation

Pour aller de la liste des pizzas au détail d’une pizza, il faut que l’utilisateur clique sur une Pizza. La gestion d’un “click” se fait avec différents Widget dont [GestureDetector](https://flutter.dev/docs/cookbook/gestures/handling-taps) ou [InkWell](https://flutter.dev/docs/cookbook/gestures/ripples).

Pour varier les plaisirs, nous allons utiliser GestureDectector lorsque l’utilisateur clique sur une pizza.

Nous allons modifier le code de l’affichage de la liste des pizzas (**fichier “pizza\_list.dart”**).

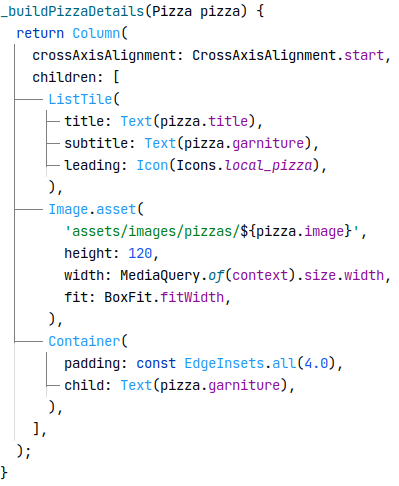
La méthode, pour afficher un élément, est “\_buildRow”. Afin de garder un minimum de clarté dans notre code, nous allons scinder cette méthode comme ceci :



Explications :

* Le widget GestureDetector a la méthode “onTap” qui réagit au clic de l'utilisateur
* Au clic, l’application va vers la fenêtre PizzaDetails, nous passons au constructeur la pizza sélectionnée

Puis la méthode “\_buildPizzaDetails” :



Explications :

* Pour l’affichage de l’image de la pizza
  + nous fixons une hauteur arbitraire
  + nous fixons la largeur en fonction du téléphone : MediaQuery.*of*(context).size.width
  + l’image sera entièrement affichée sur la largeur mais tronquée sur la hauteur

Et la méthode “\_buildBuyButton” identique à la page 4.

# 

# Factorisation

Comme vous l’avez remarqué, le code de la méthode “\_buildBuyButton” est dupliqué. Nous allons donc créer une classe pour afficher le bouton.

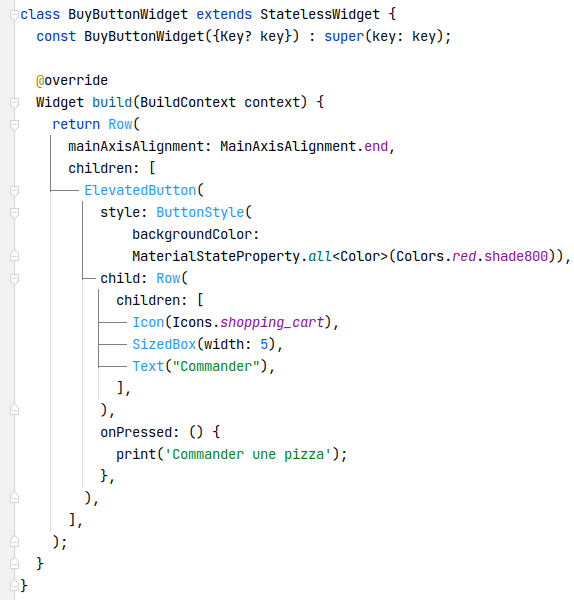
## La classe BuyButtonWidget

Dans le répertoire “ui”, créer un répertoire “share”. Dans le répertoire “share”, créez un nouveau fichier nommé “buy\_button\_widget”.

Ce Widget n’ayant aucune interaction utilisateur, il sera stateless.

Dans le fichier vide, tapez “stless” puis entrée, nommez la classe “BuyButtonWidget”.

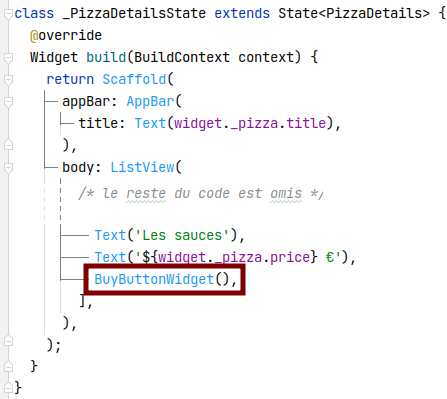
Le code de la méthode “build” doit être celui de la méthode “\_buildBuyButton” de la page 4 :



## Utilisation de la classe

### Dans la classe PizzaDetails

Modifiez la classe \_PizzaDetailsState pour intégrer le widget :



### 

### Dans la classe PizzaList

Vous devez modifiez la méthode “\_buildRow” comme ceci :



# L'affichage d’une pizza (suite)

Comme vous l’aurez remarqué, il n’y a aucune mise en forme.

## Le format du texte

Le texte est formaté de la même façon à plusieurs endroits.

Le code ne devant exister qu’à un seul endroit, nous allons créer la classe PizzeriaStyle pour définir les différents styles utilisés.

Dans le répertoire “ui/share”, créez un nouveau fichier nommé “pizzeria\_style”.

Créer la classe PizzeriaStyle comme ceci :



Explications :

* un style de base est fixé : baseTextStyle
* les autres style sont construit à partir de ce style grâce à la méthode copyWith qui a des paramètres nommés

## Utilisation des styles

Nous allons modifier l’affichage d’une pizza pour intégrer les styles de police.

Il nous faut modifier la méthode “build” de la classe “\_PizzaDetailsState” comme ceci (extrait) :



## L’affichage des boîtes de sélections (DropDownButton)

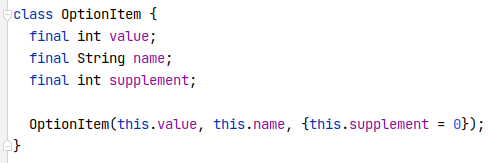
L'utilisateur doit sélectionner la pâte, la taille et la sauce. Pour ce faire, nous allons utiliser le widget [DropdownButton](https://api.flutter.dev/flutter/material/DropdownButton-class.html) qui permet d’afficher une liste déroulante.

Ce widget a un constructeur typé : DropdownButton<**T**>(...) où **T** représente la classe utilisée pour un élément.

Nous allons donc créer une classe pour représenter un élément de la classe DropDownButton.

### La classe OptionItem

Dans le répertoire “lib/models”, créez le fichier “option\_item” dans lequel vous aurez la classe “OptionItem” définie ainsi :



Cette classe définit un attribut “supplement” car la pâte, la taille et la sauce auront un supplément en fonction de ce qui sera choisi.

### Modification de la classe Pizza

Il faut ajouter les différentes options possibles (pâte, taille et sauce) dans la classe Pizza. Nous allons le faire en dur mais il faudrait le faire d’après le format JSON.

De plus, il faut savoir quelles options sont sélectionnées par l'utilisateur.

Le code de la classe Pizza sera alors le suivant :



### Modification de la classe PizzaDetails

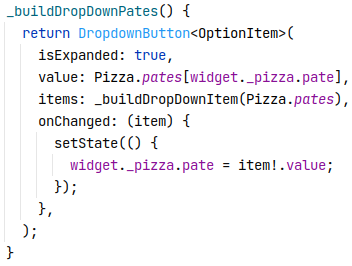
Comme il y a 3 listes déroulantes nous allons créer des méthodes pour les afficher.

Nous allons remplacer le texte par les méthodes comme ceci :



Pour chaque méthode, nous allons utiliser le widget DropDownButton.

La méthode “\_buildDropDownPates” va s’écrire comme ceci :

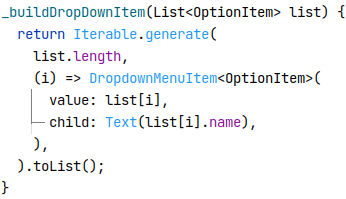


Explications :

* la classe “DropDownButton” est typée avec la classe OptionItem. La liste des éléments sera donc du type OptionItem.
* la valeur “true” pour l’attribut “isExpanded” permet d’afficher la liste sur toute la largeur disponible
* l’attribut “value” permet de fixer l’élément sélectionné au départ
* l’attribut “items” contient donc la liste des éléments. La construction est déléguée à la méthode “\_buildDropDownItem” définie ci-après.
* La méthode “onChange” est obligatoire, elle va permet de modifier l’attribut “value”

Vous devez écrire les méthodes “\_buildDropDownTailles” et “\_buildDropDownSauces” de la même manière.

La méthode “\_buildDropDownItem” permet donc de remplir les éléments du widget “DropDownButton”. Le code est le suivant :



Explications :

* la classe Iterable a un constructeur nommé “generate” qui permet de générer une collection d’objet DropdownMenuItem
* le constructeur “generate” doit avoir la taille de la collection à générer (list.length)
* pour chaque index, défini d’après la taille, la classe Iterable renvoie un objet DropdownMenuItem
* un objet DropdownMenuItem est typé avec OptionItem
* la collection est converti au format liste (.toList())

Testez !!!

# 

# Le calcul du prix d’une pizza

Pour l’instant nous avons juste afficher le prix de base de la pizza. Mais nous voulons ajouter des options telles que la pâte, la taille et la sauce.

De plus l’affichage doit ressemble à cela :



Cet affichage sera réutilisé dans le panier, nous allons donc créer un widget.

## La classe TotalWidget

Dans le répertoire “lib/ui/share”, créez le fichier “total\_widget”. Ce widget sera sans état. Dans le fichier vide, tapez “stless” puis entrée, nommez la classe “TotalWidget”.

Nous voulons avoir un fond gris avec une bordure noire et un widget de type Row pour afficher les éléments. Le code sera le suivant :

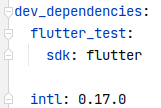


Le widget Row permet d’afficher le texte Total et le prix, ces 2 éléments devant prendre la même largeur et avoir un padding. De plus nous voulons formater le prix avec une virgule et le symbole euro.

Le code sera alors le suivant :

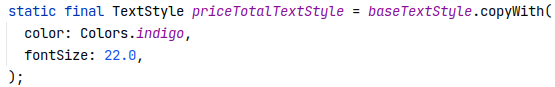


Nous utilisons la classe NumberFormat présente dans le package [intl](https://pub.dev/packages/intl). Vous devez l’ajouter dans le fichier pubspec.yaml :



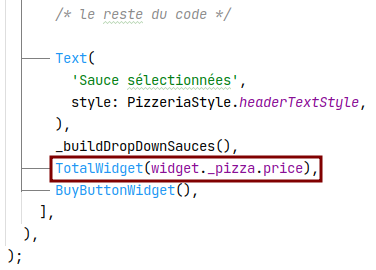
Puis lancez la commande ”flutter pub get”.

Nous utilisons aussi un nouveau style que vous devez ajouter à la classe PizzeriaStyle :



## Intégration de la classe TotalWidget

La classe TotalWidget va être utilisé dans la classe \_PizzaDetailsState, méthode build comme ceci :



Si vous testez le prix de base s’affiche bien mais les options (pâtes, sauces, …) ne sont pas prises en compte.

## 

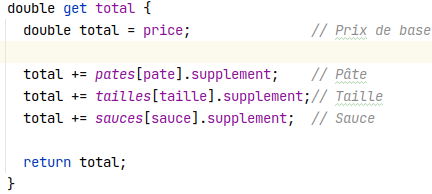
## Le calcul d’après les options

Une pizza a un prix de base et des options.Au lieu de renvoyer le prix de base, il faut renvoyer le total du prix en incluant les options :

* chaque option a un supplément (0 par défaut)
* les attributs pate, taille et sauce permettent de savoir quelle option est sélectionnée.

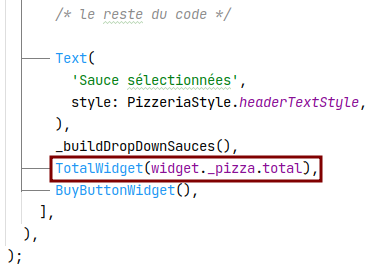
### Modification de la classe Pizza

Nous allons ajouter le “getter total” dans la classe Pizza comme ceci :



### Modification de la classe PizzaDetails

Dans la classe \_PizzaDetailsState, méthode build, nous allons passer le prix total de la pizza au widget TotalWidget comme ceci :

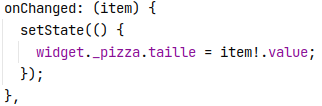


Testez. Le prix est mis à jour !

### 

### Comment cela fonctionne-t-il ?

Lorsqu’une option est changée, le code exécute la méthode setState, exemple pour la sauce :



Flutter sait que tous les objets utilisant la taille doivent être mis à jour. Or le getter total de la classe Pizza l’utilise, Flutter met alors à jour les objets utilisant ce getter.

Donc Flutter met à jour le prix total :



# Résumé

Vous devez maitriser les éléments suivants :

* Utilisation de [Expanded](https://api.flutter.dev/flutter/widgets/Expanded-class.html) et [SizedBox](https://api.flutter.dev/flutter/widgets/SizedBox-class.html)
* Utilisation de [MediaQuery](https://api.flutter.dev/flutter/widgets/MediaQuery-class.html)