



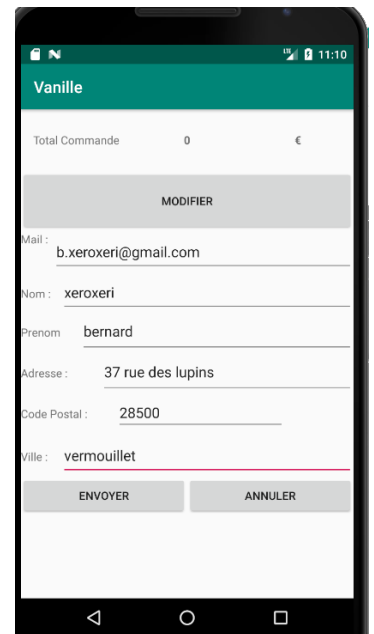
Contexte : Vanille

Les principales fonctionnalités sont :

- Import du catalogue des bonbons depuis le serveur web,
- Affichage des bonbons et sélection,
- **Saisie de la quantité et ajout au panier,**
- **Saisie des informations du client,**
- Export des données sur le serveur web.

Voici le premier écran que nous allons élaborer lors de ce TP : visualisation du montant de la commande et saisie des informations du client

- Lancer Android Studio en tant qu'**Administrateur** (toujours).
- Après chargement complet de votre projet, lancer votre émulateur ou brancher votre téléphone.

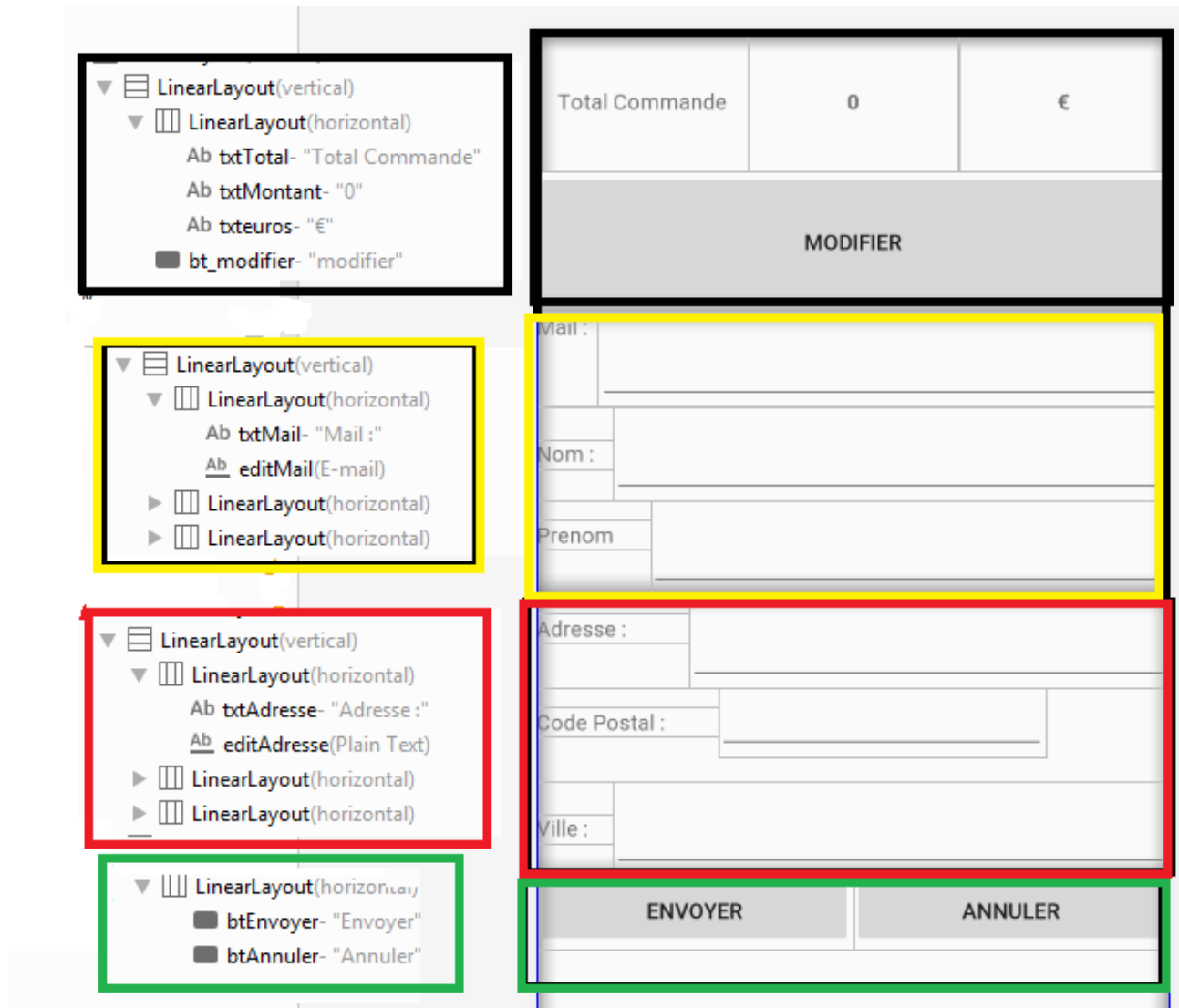


## A. Elaboration d'une deuxième vue : facture du client

- Ajouter une nouvelle activité vide au projet nommée FactureActivity en cochant Launcher Activity afin de pouvoir lancer directement l'activité.

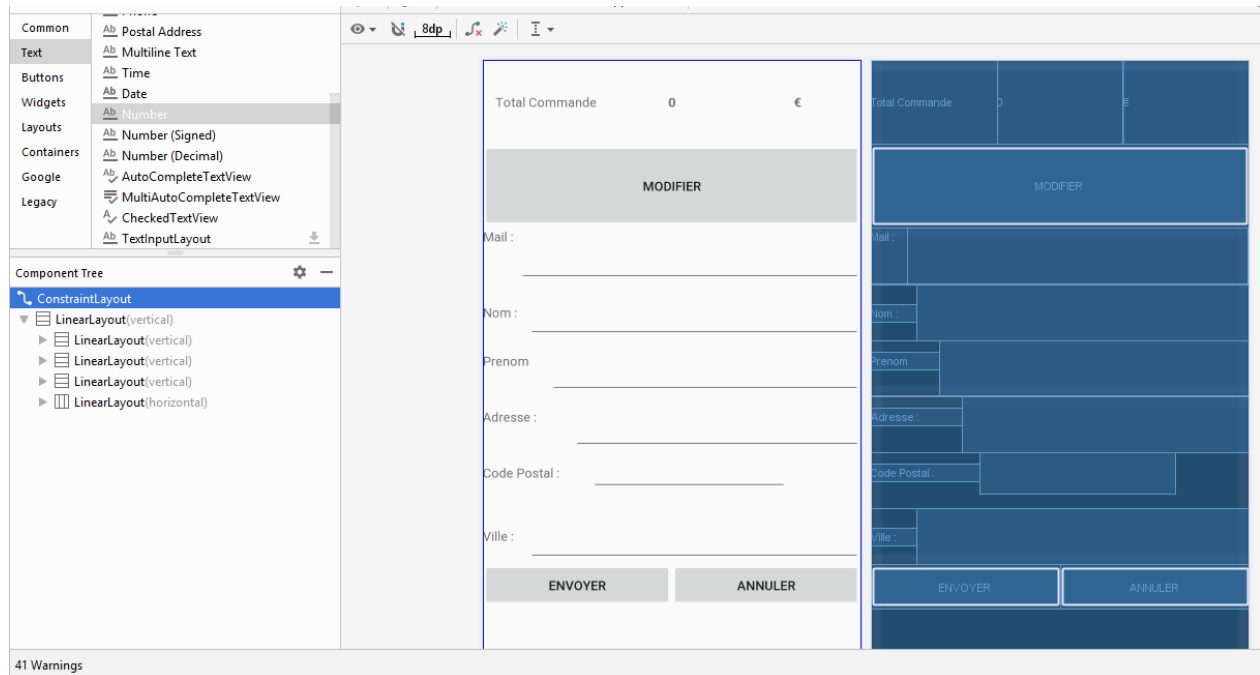
### 1. Création de l'interface

- Dessiner l'interface correspondant à cette activité à l'aide du fichier activity\_facture.xml.
- Ajouter 4 layouts au layout principal linéaire vertical:



Vous devez vous intéresser plus particulièrement aux propriétés `Id`, `layout_width` et `layout_height` (`match_parent` ou `wrap_content`), `Gravity`, `textStyle`,

On obtient



## II. Implémentation de l'activité correspondante

### A. Lien entre l'activité et le Layout

Rappel : Un écran se caractérise par

- Une classe .java héritant de la classe activity : `FactureActivity.java` (contrôleur)
- Un fichier .XML étant le layout de l'activité : `activity_facture.xml` (vue)

Dans la méthode `onCreate()` de la classe , la méthode `setContentView(R.layout.activity_facture)` permet d'associer la classe au layout.

### B. Lien avec les Widgets de la vue

Nous allons créer des vues (objets) afin de les lier aux widgets du fichier xml,

- Déclarer les attributs privés de la classe `FactureActivity` de type `EditText` correspondant aux zones saisies par l'utilisateur (nom, prénom...) et `TextView` à la zone d'affichage comme ci-dessous

Il est classique de reprendre le même nom :

```
public class FactureActivity extends AppCompatActivity {  
  
    //déclaration des editText permettant de récupérer la saisie de l'utilisateur  
    private EditText editNom;  
    private EditText editPrenom;  
    private EditText editAdresse;  
    private EditText editCP;  
    private EditText editVille;  
    private EditText editMail;  
    //zone d'affichage du montant de la facture  
    private TextView txtMontant;
```

- Dans une procédure init() ,Créer un lien entre les variables de la classe et les widgets du fichier xml grâce à leur identifiant :

```
private void init(){  
    editAdresse = findViewById(R.id.Facture_editTextAdresse);  
    editCP = findViewById(R.id.Facture_editTextCode);  
    editMail = findViewById(R.id.Facture_editTextEmailAddress);  
    editNom = findViewById(R.id.Facture_edittextNom);  
    editPrenom = findViewById(R.id.Facture_editTextPrenom);  
    editVille = findViewById(R.id.Facture_editTextVille);  
    txtMontant = findViewById(R.id.Facture_textMontant);  
    bt_envoyer = findViewById(R.id.Facture_bt_envoyer);  
    txtMontant.setText("100.20");
```

- Appeler la méthode init() lors de la création de la vue (onCreate).
- Tester.

Que permet la dernière ligne de la méthode init() ?

### C. Lien avec la commande

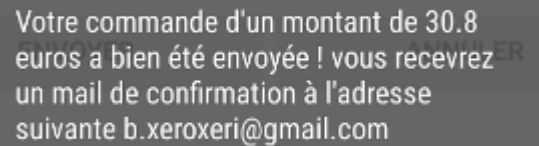
Afin de faire le lien de cette interface avec la commande , nous allons instancier, si ce n'est déjà fait, notre singleton.

- Déclarer un attribut privé de type Commande nommé commandeEnCours ;
- Dans la méthode init(), initialiser cet attribut à l'aide de la méthode getInstance() ;
- Afficher le montant de la commande dans le txtMontant.
- Tester, le montant de la facture doit être égal à 52.0

## D. Gestion du bouton ENVOYER

Pour l'instant un clic sur le bouton ENVOYER va :

- Afficher un Toast signalant à l'utilisateur que la commande a bien été envoyée
  - Créer un client à l'aide des informations saisies et enregistrer ce client dans l'attribut client de la commande.
- Déclarer un attribut privé de type Button.
  - Dans la méthode init(), créer le lien avec le bouton du fichier xml.



Votre commande d'un montant de 30.8 euros a bien été envoyée ! vous recevrez un mail de confirmation à l'adresse suivante b.xeroxeri@gmail.com

Nous allons à présent gérer le clic du bouton,

**Pour vous aider :**

Duration est un entier 1 (LENGTH\_LONG) 0 pour (LENGTH\_SHORT)  
editNom.getText() ne retourne pas un String il faut rajouter editNom.getText().toString()

- Toujours dans méthode init(), écrire la méthode suivante et compléter :

```
bt_envoyer.setOnClickListener(new View.OnClickListener() {  
    @Override  
    public void onClick(View view) {  
        //création d'un client  
        //à vous de jouer  
  
        //ajout du client à la commande  
  
        //Créer un Toast et le lancer  
        //Toast.makeText(context,text,duration).show();  
        //à vous de jouer...  
  
        Toast.makeText(view.getContext(), text: "Votre commande d'un montant de "  
            " a bien été envoyée! vous recevrez un mail de confirmation à l  
    }  
});
```

- Relancer l'application afin de tester.

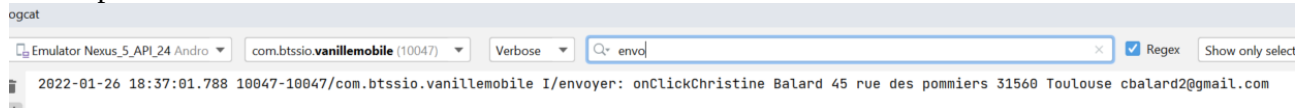
Afin de vérifier que le client a bien été créé, nous allons nous servir de la console Logcat

- Ajouter cette ligne de commande dans le gestionnaire du clic du bouton ENVOYER :

```
Log.i("envoyer", "onClick "+ commandeEnCours.getClient().toString());
```

➤ Tester

Voici un exemple de résultat :

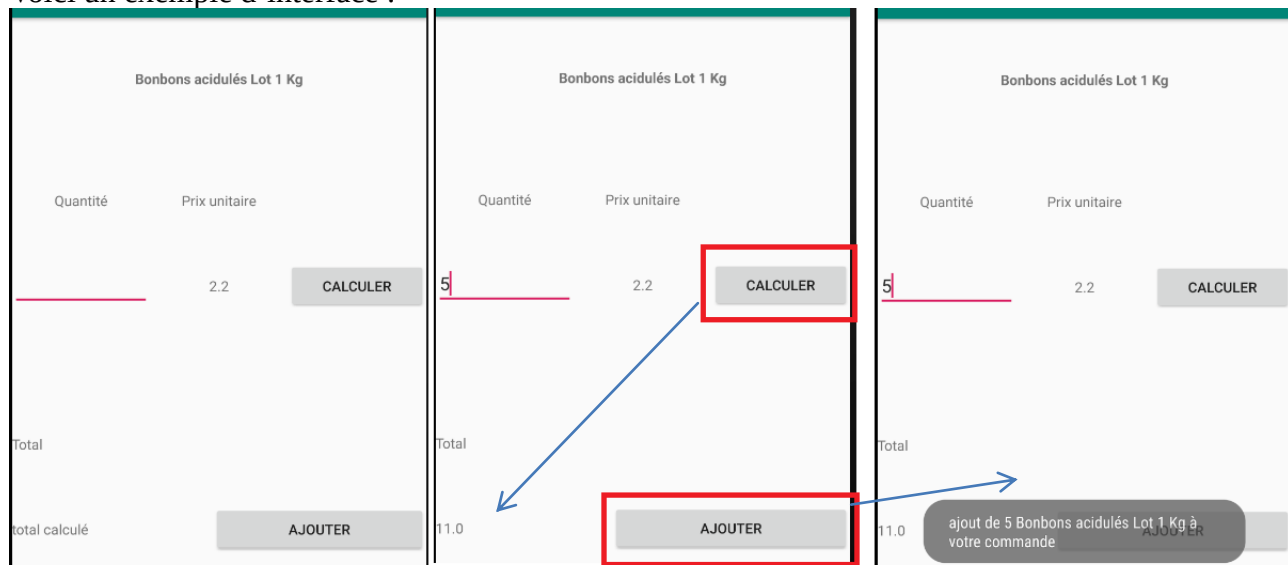


Vous pouvez ajouter des filtres pour ne visualiser que les logs qui vous intéressent.

### III. Elaboration de la vue saisie d'un bonbon

Vous allez créer une activité pour permettre à l'utilisateur de commander un produit. Ce produit a déjà été sélectionné par l'utilisateur dans une ListView (que nous réaliserons plus tard). Nous simulerons cette sélection par la création du produit dans la méthode init().

Voici un exemple d'interface :



Les éléments à respecter :

- Affichage de la description et du prix unitaire (actuel) du bonbon,
- Saisie de la quantité par l'utilisateur,
- Calcul du montant de la ligne lors du clic sur le bouton CALCULER
- Affichage du Toast et ajout de la ligne de commande lors du clic sur le bouton AJOUTER

- Créer une activité vide nommée ProduitActivity, elle doit pouvoir se lancer seule (LAUNCHER)
- Elaborer la partie xml de la vue (activity\_produit.xml)
- Implémenter la classe ProduitActivity (regarder la suite pour vous aider...)

```
public class ProduitActivity extends AppCompatActivity {  
    //déclaration des attributs de la vue afin de faire le lien avec le fichier XML  
    private EditText editQte;  
    .....  
    //déclaration de la commande en cours  
    private Commande commandeEnCours;  
    //déclaration du produit sélectionné  
    private Produit produitSelectionne;  
  
    @Override  
    protected void onCreate(Bundle savedInstanceState) {  
        super.onCreate(savedInstanceState);  
        setContentView(R.layout.activity_produit);  
        init();  
    }  
  
    private void init(){  
        //lien avec les widgets de la vue  
        editQte = findViewById(R.id.Produit_editQte);  
        .....  
  
        //récupération de la commande  
        commandeEnCours = Commande.getInstance();  
        //simulation de la selection du Produit  
        Categorie chocolats = new Categorie( id: "cho", libelle: "Chocolats");  
        produitSelectionne = new Produit( id: "B003", description: "Bonbons chocolat Lot 3 Kg", image: " ", (float)3.0, chocolats);  
  
        //affichage des informations sur le produit  
        textPU.setText(String.valueOf(produitSelectionne.getPrixActuel(LocalDate.now())));  
        .....  
  
        //gestion du clic du bouton calculer  
        btCalculer.setOnClickListener(new View.OnClickListener() {  
            @Override  
            public void onClick(View view) {  
                int qte;  
                float prix;  
                qte = Integer.parseInt(editQte.getText().toString());  
                prix = Float.parseFloat(textPU.getText().toString());  
                textTotalC.setText(String.valueOf(qte*prix));  
            }  
        });  
  
        //gestion du clic du bouton Ajouter  
        btAjouter.setOnClickListener(new View.OnClickListener() {  
            @Override  
            public void onClick(View view) {  
                int qte =Integer.parseInt(editQte.getText().toString());  
                .....  
            }  
        });  
    }  
}
```

Afin de tester le bouton ajout, afficher dans un log toute la commande. Pour cela :

- Générer la méthode toString() dans la classe Commande.
- Ajouter ces 2 lignes de commandes dans le gestionnaire du clic du bouton AJOUTER :

```
Log.i( tag: "ajouter", msg: "onClick: ");  
Log.i( tag: "ajouter", instance.getCommande().toString());
```

- Tester.