



IN01

Programmation Android

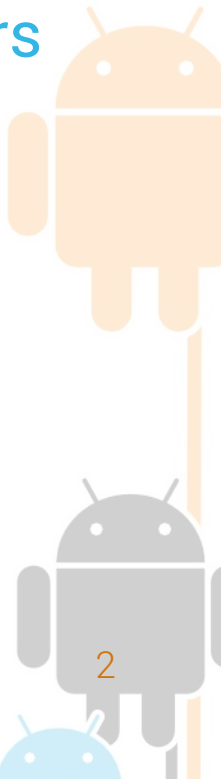
Travaux pratiques

Yann Caron

le cnam

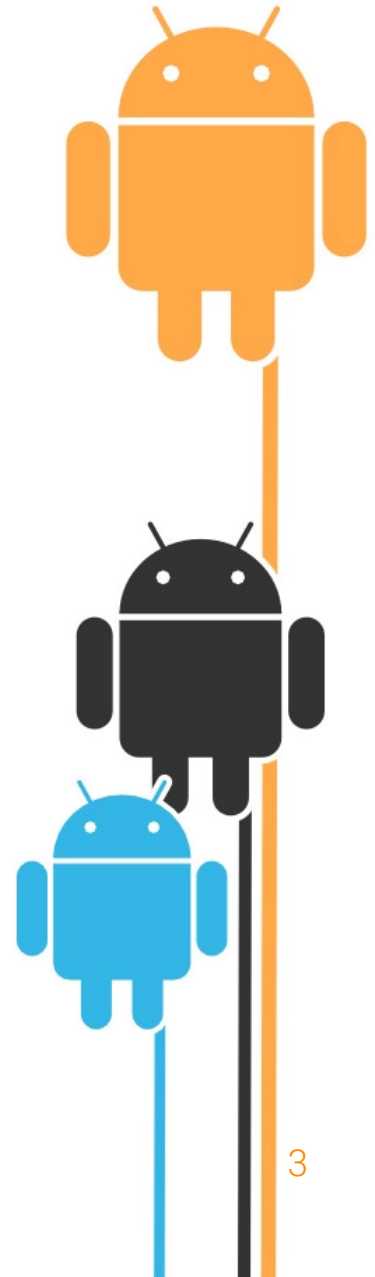
Au programme

- Une application complète : Pêcheur du Léman, c'est de saison !
- Seul ou en binôme (c'est libre !)
- On peut aussi implémenter par groupe et par fonctionnalité et on croise !
- Une application, comme son nom l'indique, pour les pêcheurs
 - ➔ Une base de donnée d'appâts, de poissons
 - ➔ Les spots de pêches, les prises
 - ➔ Des outils, comme la boussole, la météo
 - ➔ Et tout ce que nous pourrions imaginer



Sommaire - TP

- TP 1 :: Prise en main des outils :: Hi Android :: Eclipse, ADT, AVD, ADB
- TP 2 :: Pêcheurs du Léman : Partie I :: WYSIWYG, persistence, logcat
- TP 3 :: Pêcheurs du Léman : Partie II :: HMI, View et Layout, menus
- TP 4 :: Pêcheurs du Léman : Partie III :: SGBD
- TP 5 :: Pêcheurs du Léman : Partie IV :: Webservice Météo, Map
- TP 6 :: Pêcheurs du Léman : Partie V :: Boussole, Photo



IN01 – Travaux pratiques

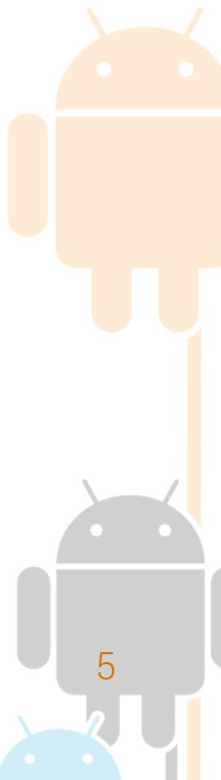
TP 1

Prise en main des outils :: Hi Android
Eclipse, ADT, AVD, ADB



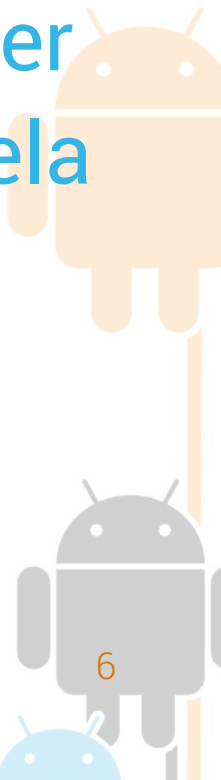
TP 1 – Prise en main

- Très facile !
- Ouvrir l'ADT Bundle (Android Development Tool – alias Eclipse + ADT Plugins) et créer un nouveau projet appelé `fr.cnam.in01.HiAndroid` !
- Remarque :: l'ADT crée 2 projets dont `appCompat_v7` c'est une bibliothèque (lib) qui contient des éléments pour assurer la retrocompatibilité des appareils



TP 1 – Prise en main

- Maintenant créons un virtual device (AVD) avec l'AVD Manager
- Lançons l'application, ça fonctionne déjà !
- Allons un peu plus loin. Nous voulons changer le text “Hello World” en “Hello Android”, où cela se passe-t-il ?
- Changeons le nom de l'application



IN01 – Travaux pratiques

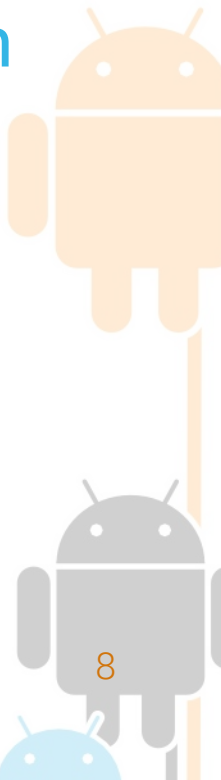
TP 2

Pêcheurs du Léman :: Partie I
WYSIWYG, persistence, logcat



TP 2 – PDL – Partie I

- Dans Eclipse, créons notre projet
`fr.cnam.in01.pecheurDuLeman`
(LakeGenevaFisherman marche aussi !!!!)
- Créons une première Activity :: Bait (les leurres de pêche)
- Dans le manifest, déclarons là et faisons d'elle la main activity
- Utilisons le WYSIWYG disponible pour créer la vue
- Attention !! Il nous faut utiliser le fichier `string.xml` pour nos labels



TP 2 – PDL – Partie I

- Ce qui est attendu =>
- Les vues utilisées ::
 - ➔ LinearLayout (vertical)
 - ➔ TextView (pour les labels)
 - ➔ EditText (pour les champs de saisie)

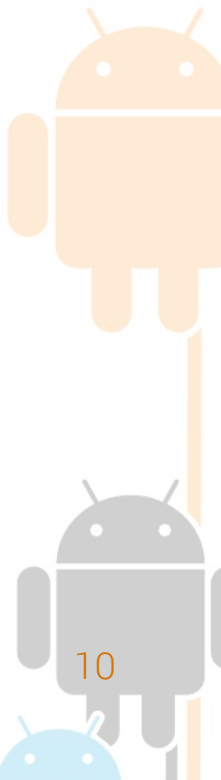


The screenshot shows an Android application interface with a black background. At the top, there is a status bar with a Wi-Fi icon and a battery icon. Below the status bar is a header bar with a blue fish icon and the text "Pêcheur Du Leman". The main content area contains five labels and corresponding input fields, all in white text on a black background:

- Bait name: [input field]
- Brand: [input field]
- Price (\$): [input field]
- Size (in): [input field]
- Weight (oz): [input field]

TP 2 – PDL – Partie I

- Maintenant nous aimerions comprendre le cycle de vie de l'application ::
 - ➔ Dans l'activity nous allons créer un log dans les méthodes onCreate, onDestroy, onPause, onResume
 - ➔ Observons le logcat



TP 2 – PDL – Partie I

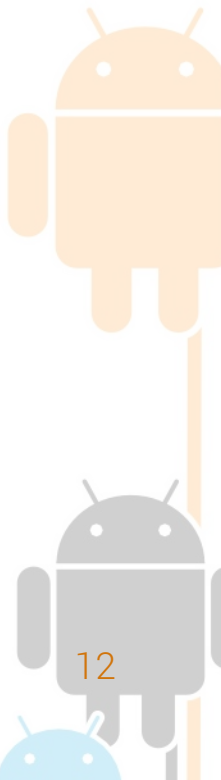
- Notre premier Pojo nous est fournis
- Nous devons le créer dans le package dto (Data Transfert Object)

```
public class Bait {  
  
    private String name; private Brand brand;  
    private float price, size, weight;  
  
    public String getName() {  
        return name;  
    }  
  
    public void setName(String name) {  
        this.name = name;  
    }  
  
    // accesseurs etc ....  
  
    public Bait() {  
        super();  
    }  
  
    public Bait(String name, Brand brand,  
                float price, float size, float weight) {  
  
        super();  
        this.name = name;  
        // etc ....  
    }  
}
```

TP 2 – PDL – Partie I

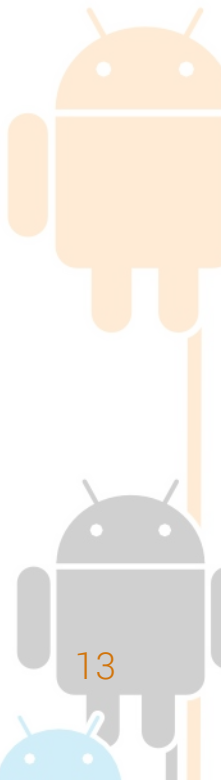
- Maintenant, lorsque nous tournons l'écran, nous aimerions conserver les données saisies par l'utilisateur. Pour passer les données ::
 - Il faut, dans la méthode onCreate, récupérer références des objets graphiques

```
public class BaitActivity extends Activity {  
  
    private EditText textName, textBrand;  
  
    @Override  
    protected void onCreate(Bundle savedInstanceState) {  
        super.onCreate(savedInstanceState);  
        setContentView(R.layout.bait);  
  
        textName = (AutoCompleteTextView) findViewById(R.id.textName);  
        textBrand = (AutoCompleteTextView) findViewById(R.id.textBrand);  
    }  
}
```



TP 2 – PDL – Partie I

- Dans la méthode `onSaveInstanceState`, on crée un objet `dto.Bait` avec les données du formulaire
- Passer l'objet sérialiser au `Bundle`
- Dans Récupérer les données ::
 - Dans la méthode `onRestoreInstanceState`, on déserialise l'objet `dto.Bait`
 - On restaure les données du formulaire



IN01 – Travaux pratiques

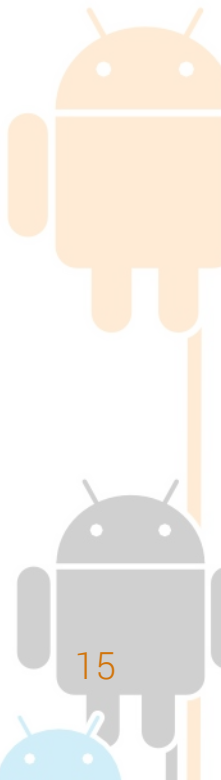
TP 3

Pêcheurs du Léman :: Partie II
WYSIWYG, persistence, logcat



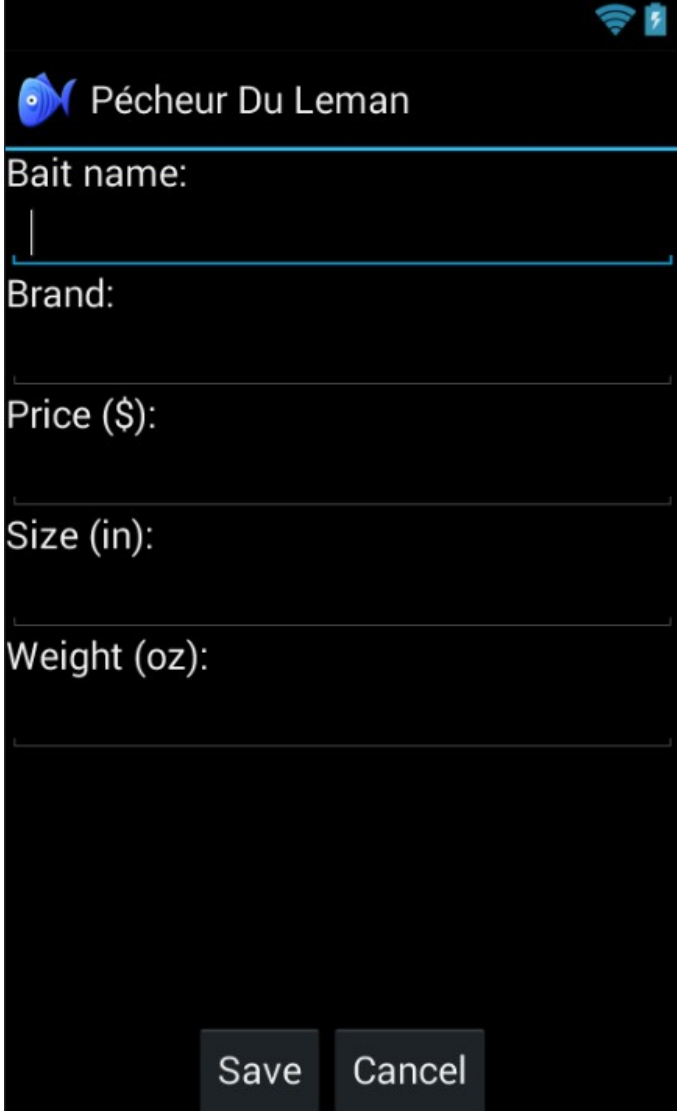
TP 3 – PDL – Partie II

- Améliorons notre formulaire
- Utilisons des `AutoCompleteTextView` pour le nom et la marque



TP 3 – PDL – Partie II

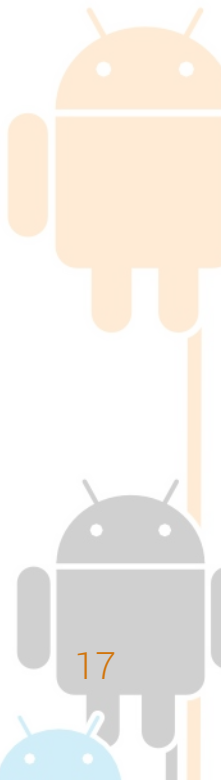
- Utilisons les layout pour ajouter les boutons d'action Save et Cancel comme ceci =>
- On peut utiliser des LinearLayout ou des RelativeLayout
- Récupérons les évènements de ces boutons :: c'est pour plutard



The screenshot shows an Android application interface with a black background. At the top, there is a status bar with a Wi-Fi icon and a battery icon. Below the status bar, the app title "Pêcheur Du Lemman" is displayed next to a blue fish icon. The main form consists of five text input fields, each preceded by a label: "Bait name:", "Brand:", "Price (\$):", "Size (in):", and "Weight (oz):". At the bottom of the form, there are two buttons: "Save" and "Cancel".

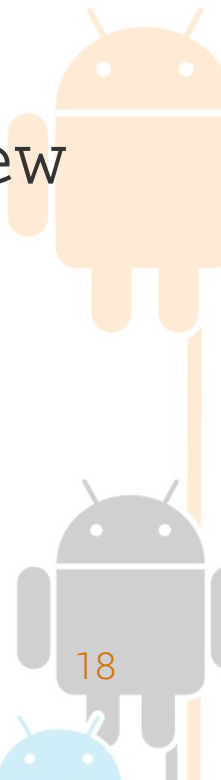
TP 3 – PDL – Partie II

- Créons un menu ::
 - ➔ Fisheries
 - Statistics
 - ➔ Data
 - Fishes
 - Baits
 - ➔ Tools
 - Compass
 - Meteo
 - Solunar Calendar

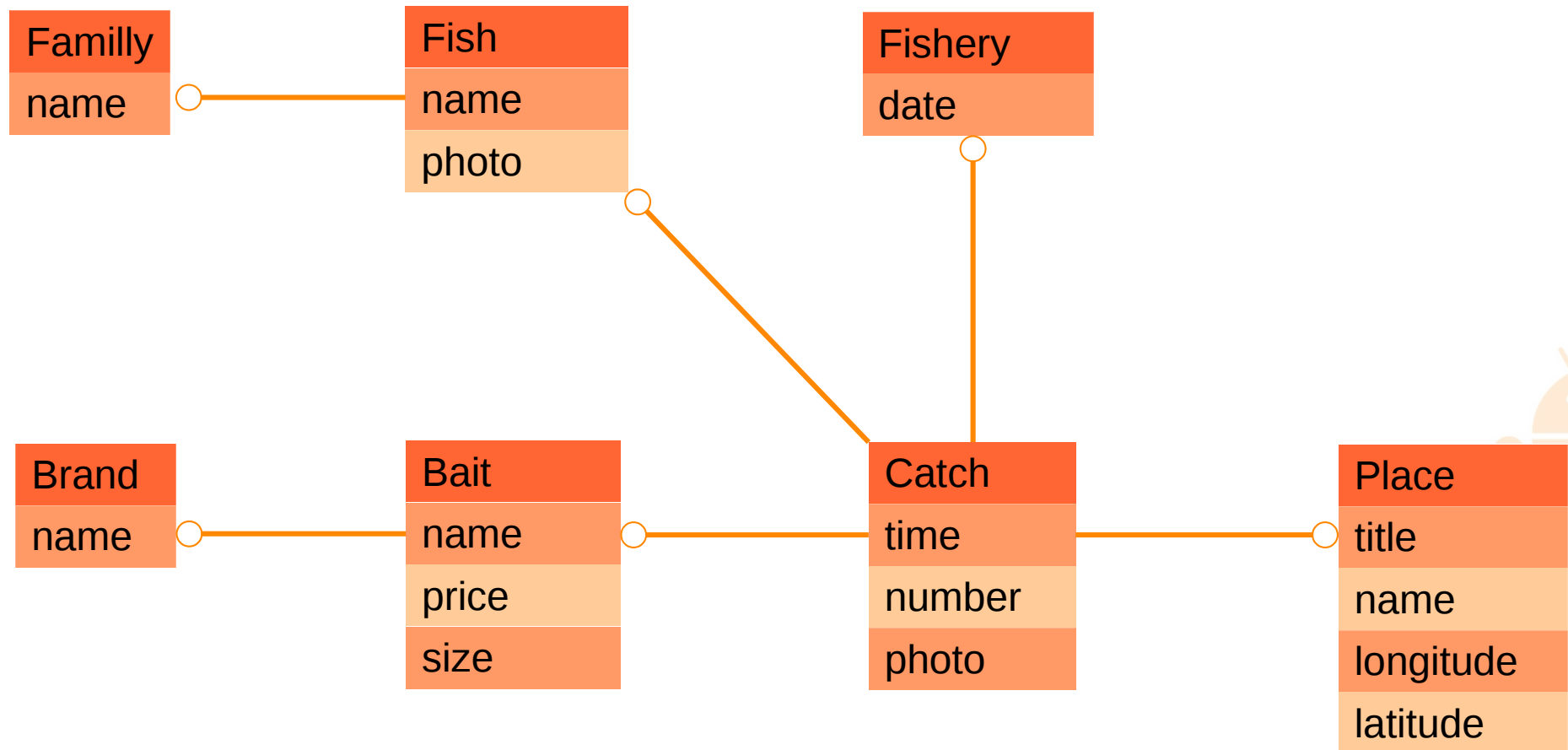


TP 3 – PDL – Partie II

- Chargeons ce menu dans notre activity
- Optionnel, sur le même modèle, créons l'activity Fish et Fisherie (MCD au slide suivant)
- Attention :: nous utiliserons les composants Spinner **plutôt qu'**`AutoCompleteTextView`
- Pour plus de fun, on pourrait en créer une de toute pièce (sans XML juste du Java)



TP 3 – PDL – Partie II



IN01 – Travaux pratiques

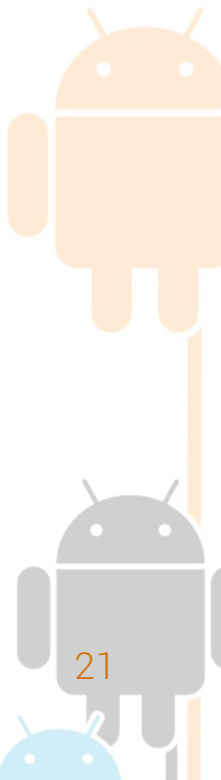
TP 4

Pêcheurs du Léman :: Partie III
SGBD

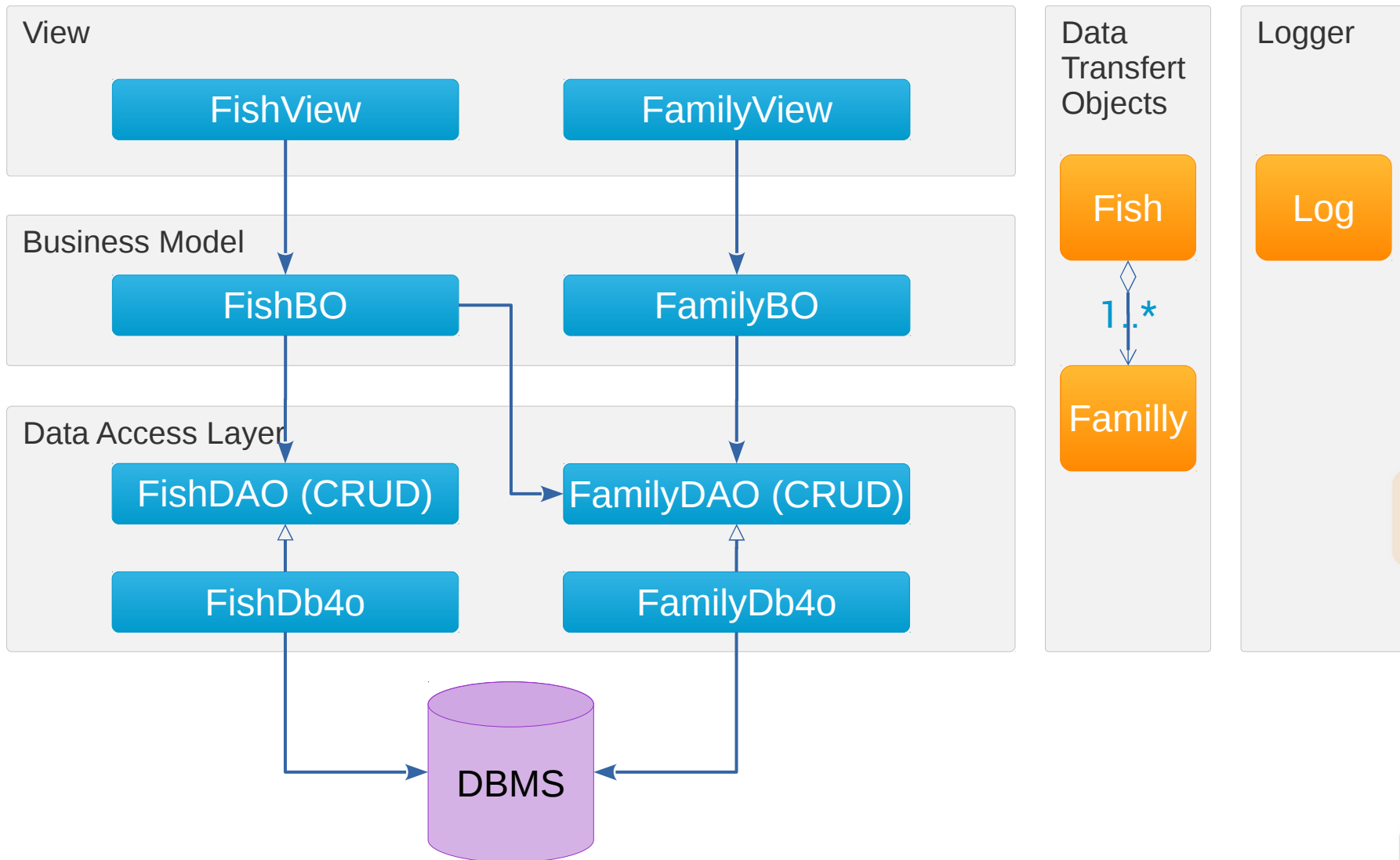


TP 4 – PDL – Partie III

- Choix du modèle de persistance :: SQLite, ORMLite ou DB4Object
- Un conseil, faire 3 groupes pour les 3 approches, puis on croise les expériences
- Optionnel :: on utilise les couches ou la MudBox ??



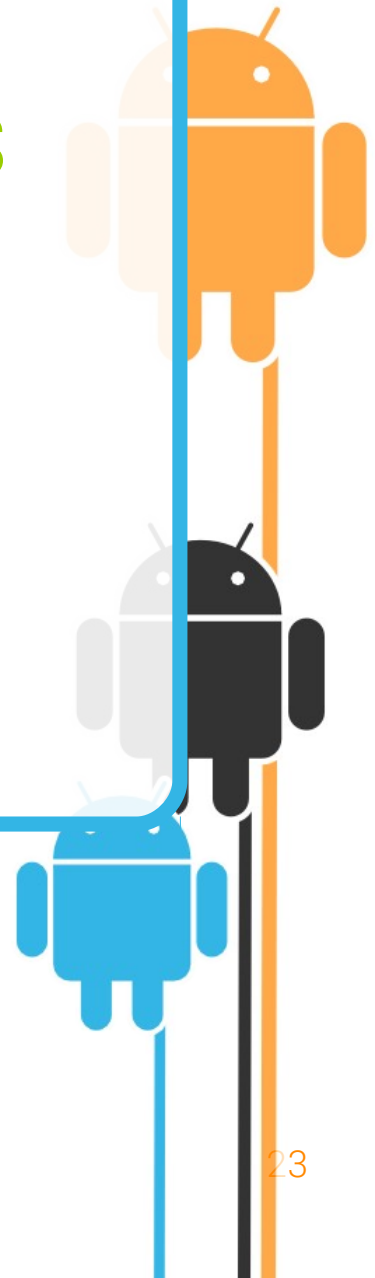
TP 4 – PDL – Partie III - Architecture



IN01 – Travaux pratiques

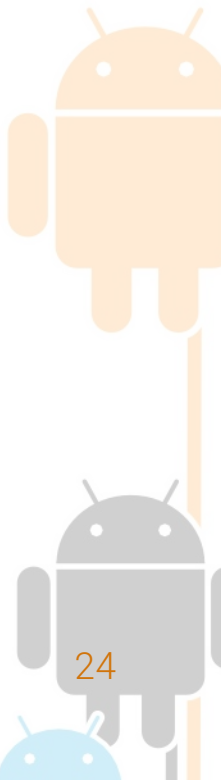
TP 5

Pêcheurs du Léman :: Partie IV Webservices météo, Map



TP 5 – PDL – Partie IV

- Pour les plus courageux ::
 - ➔ On peut aller chercher les données météo sur <http://weather.yahooapis.com/forecastrss?w=782538> et les afficher dans une vue personnalisée.
 - ➔ Ou créer une vue personnalisée pour créer un calendrier Solunaire
 - ➔ On peut aussi récupérer la vue google map et épingler le spot de pêche (géolocalisation) en prenant ::
 - la dernière position connue de l'appareil
 - les coordonnées CellID ou GPS.
 - ➔ Soyons fou !! On pourrait afficher les données météo sur la map !!
 - ➔ L'idée serait ensuite de pouvoir tout sauver dans la base de donnée
- Là encore, on peut travailler en divisant les groupes et on fusionne.



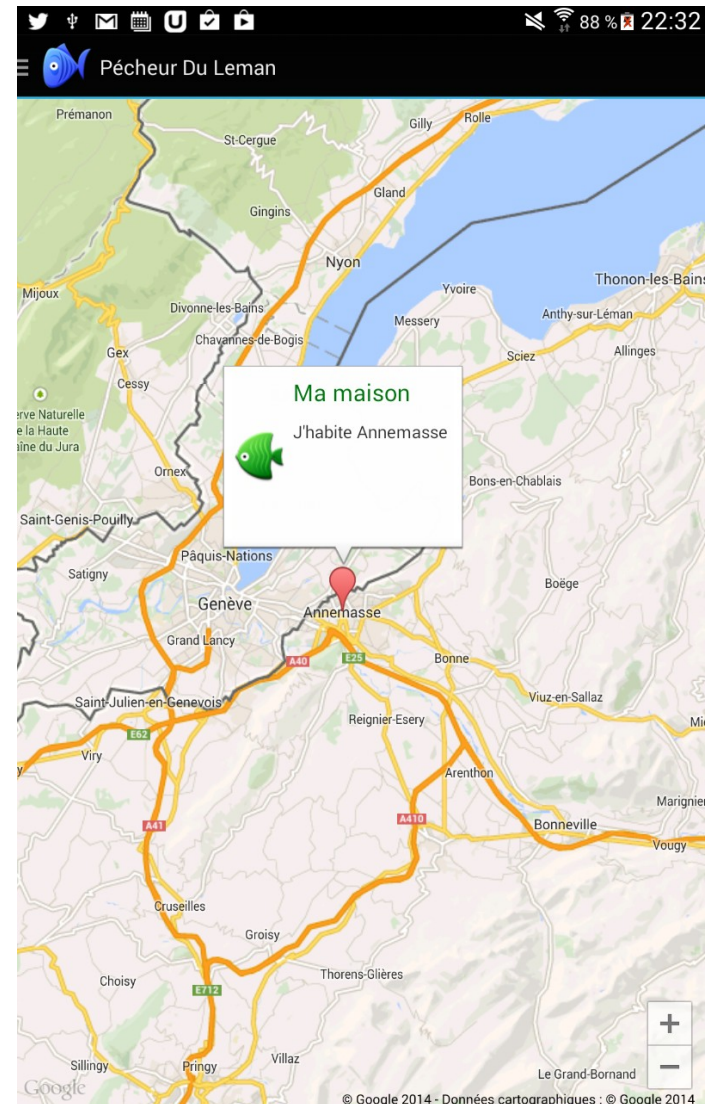
TP 5 – PDL – Partie IV



Weather app interface for Melbourne, Victoria, Australia. The app shows the current temperature as 68°F (73°/54°) and feels like 68°F. The sky is clear (Ciel dégagé). The app also displays sunrise and sunset times, day length, wind speed, and humidity. A forecast for the next four days (JEU., VEN., SAM., DIM.) is shown at the bottom.

Melbourne, Victoria, Australia
Heure locale: 2:58 AM, nov. 07 (GMT+11)
68°F 73°/54°
Feels like 68°F
Ciel dégagé
Lever du soleil: 6:07 AM
Coucher du soleil: 8:00 PM
Durée du jour: 13 h, 53 m
Vent: 32 km/h (8,9 m/s) N
Humidité: 24%
Dernière mise à jour: il ya 6 min.

JEU.	VEN.	SAM.	DIM.
Partiellement nuageux	Averses	Averses/ciel dégagé	Averses/ciel dégagé



Map app interface showing a location in Annemasse. The app displays a map of the region around Geneva, Switzerland, with a red pin marking the location. A pop-up window shows the text "Ma maison" and "J'habite Annemasse". The app also displays the name of the location, "Annemasse", and the name of the app, "Pêcheur Du Lemman".

Ma maison
J'habite Annemasse

Annemasse

Pêcheur Du Lemman

IN01 – Travaux pratiques

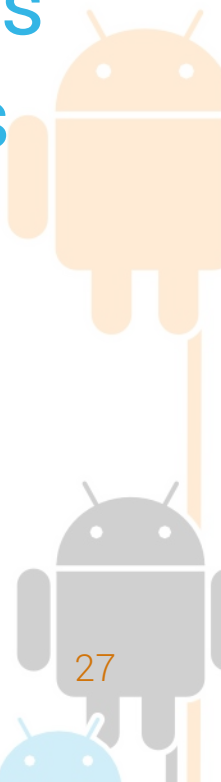
TP 6

Pêcheurs du Léman :: Partie V Boussole, Photos

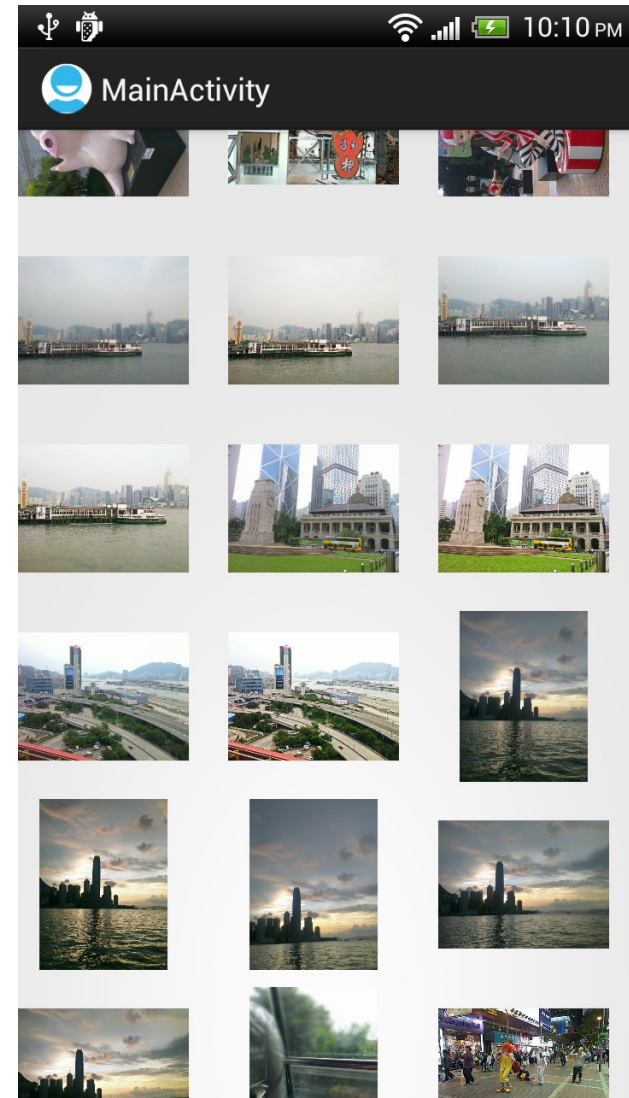
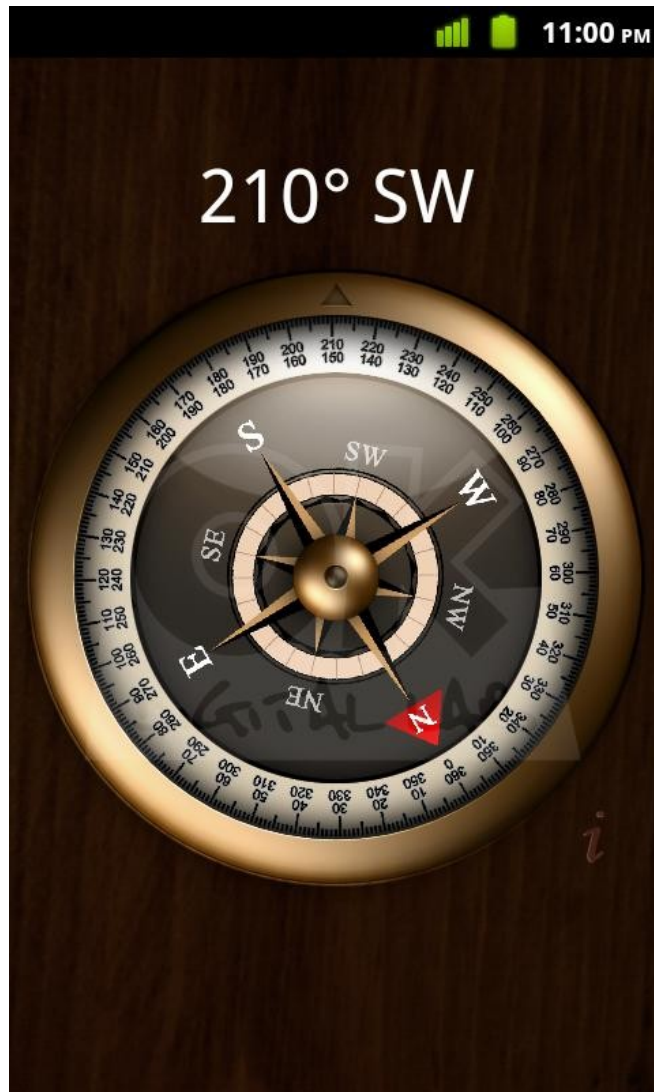


TP 6 – PDL – Partie V

- Grâce aux vues personnalisées (onDraw est notre ami), on veut créer une boussole. Il va falloir utiliser un capteur !!
- On aimerait aussi pouvoir prendre des photos des poissons, des prises et des leurres et les ajouter à nos formulaires
- On peut travailler en 2 groupes !!

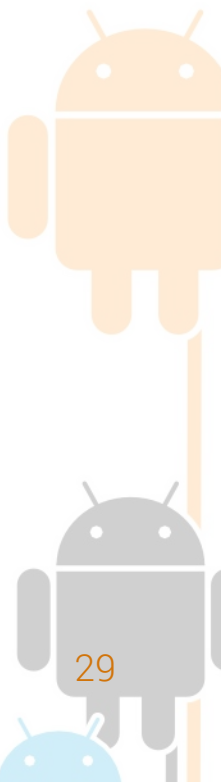


TP 6 – PDL – Partie V



Liens

- Retrouvez tous les cours sur SlideShare ::
<http://www.slideshare.net/YannCaron1/>
- Ou sur BitBucket ::
https://bitbucket.org/yann_caron/in01/src
- Les sources du projet sont sous
[/ADTWorkspace](#)



Fin

- Merci de votre participation :-)
- Finissez l'app chez vous et envoyez nous là.
- N'hésiter pas à en parler autour de vous !!
- Un sondage ?? C'est le moment de dire ce que vous en avez pensé !!

