# Cahier des charges

L’application pour tablette *Android* à réaliser doit permettre à l’utilisateur de prendre une photo. Elle sera ensuite transmise au programme de reconnaissance de visage, qui déterminera ensuite si la photo appartient à une personne connue par la base de données. Par la suite, l’application pourra aussi permettre à l’utilisateur d’ajouter une personne connue dans la base de données de visages, en renseignant son nom et son prénom.

# Prototype de l’application

Dans un premier temps, nous nous sommes concentrés sur la fonctionnalité principale, à savoir la prise de photo et son envoi au programme de reconnaissance de visage.

Nous avons d’abord déterminé les différents éléments qui devront apparaître sur la tablette, et leur agencement.

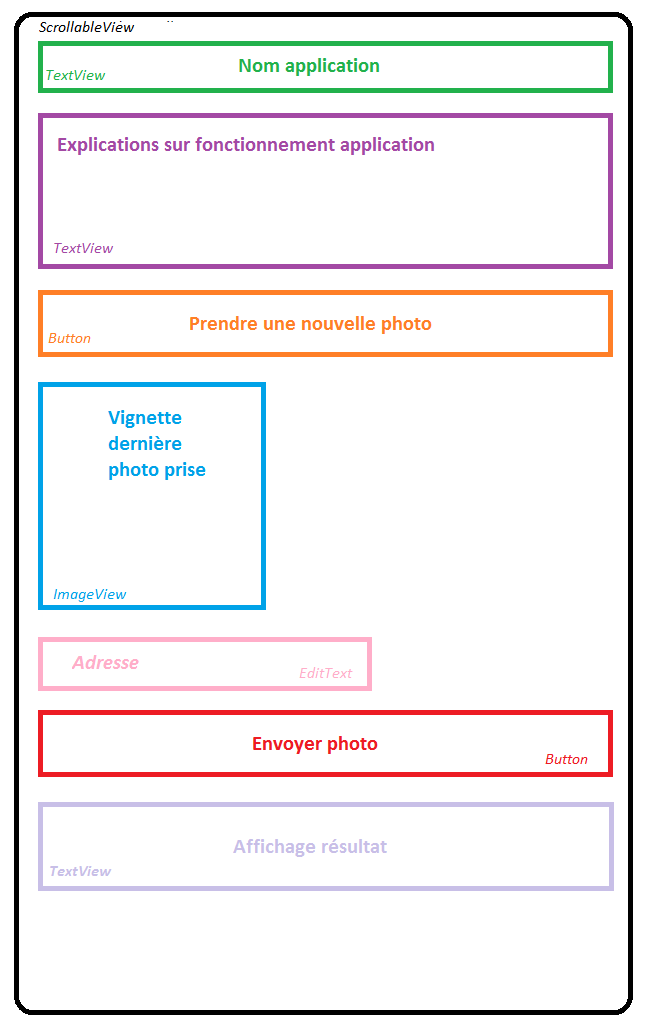
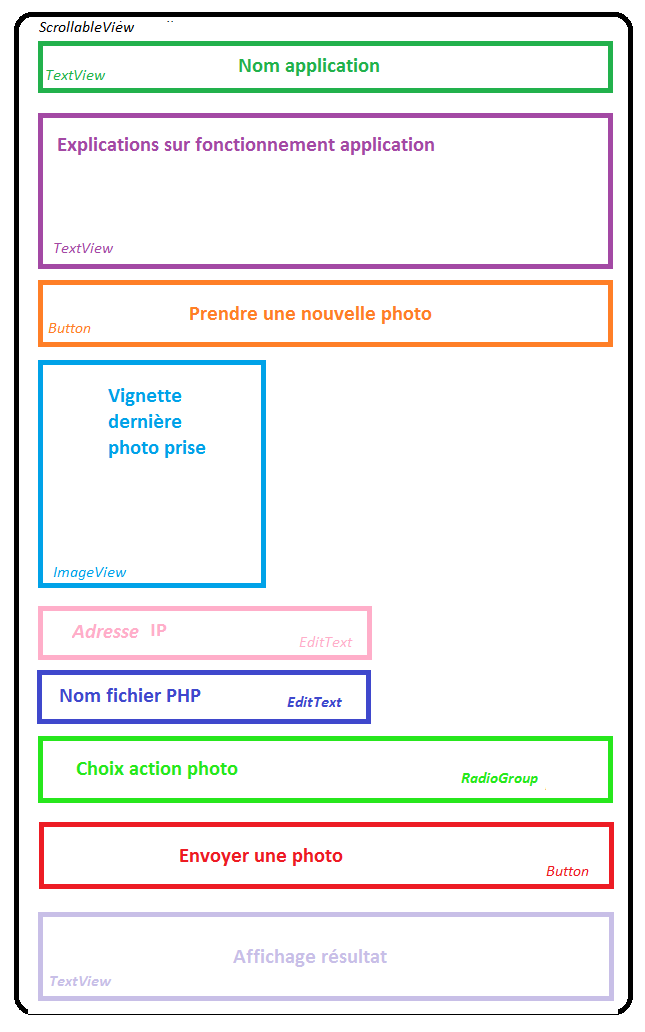


Figure 1 : Première version de l'agencement de l'application

L’application comportera donc :

* Des éléments de texte pour expliquer le fonctionnement de l’application à l’utilisateur.
* Un bouton permettant de prendre une nouvelle photo.
* Une vignette affichant un aperçu de la dernière photo, et qui par la suite permettra d’ouvrir la photo en cliquant dessus.
* Une partie « connexion » au serveur, dans laquelle l’adresse du serveur sera pré-rempli, et un bouton permettant de tester celle-ci.
* Un bouton pour envoyer la photo.
* Une zone de texte permettant d’afficher le résultat de la reconnaissance de visage.

Par la suite, nous avons décidé de changer le champ « Adresse IP » en quatre champs de saisie, pour les différents octets composants l’adresse IP du serveur. Nous avons aussi ajouté un champ de texte permettant de saisir le nom du fichier PHP correspondant au serveur. La mise en page de l’application est donc la suivant :



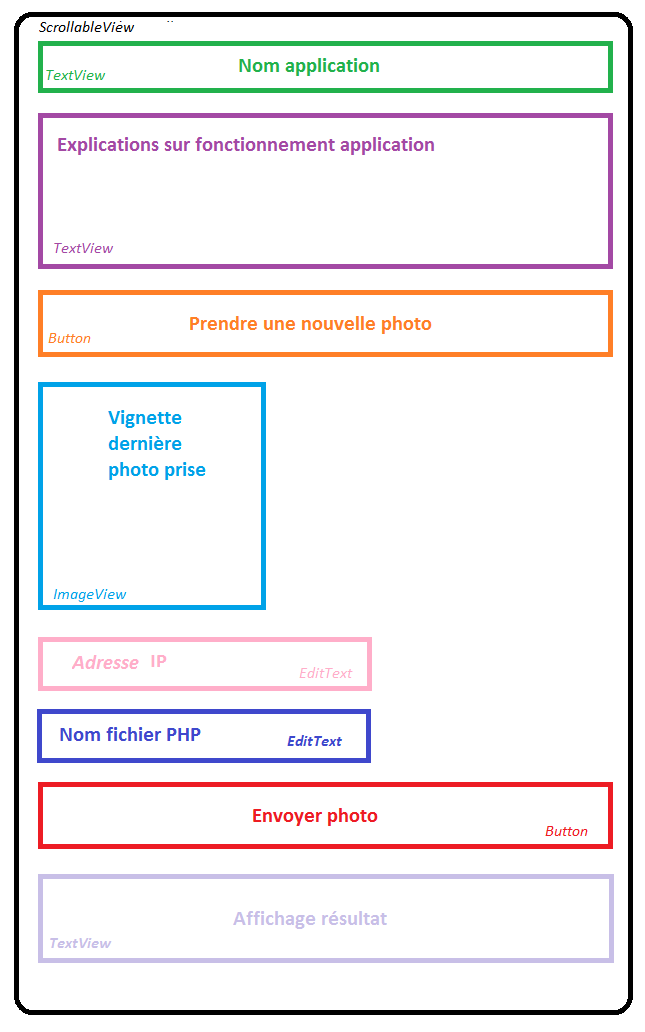
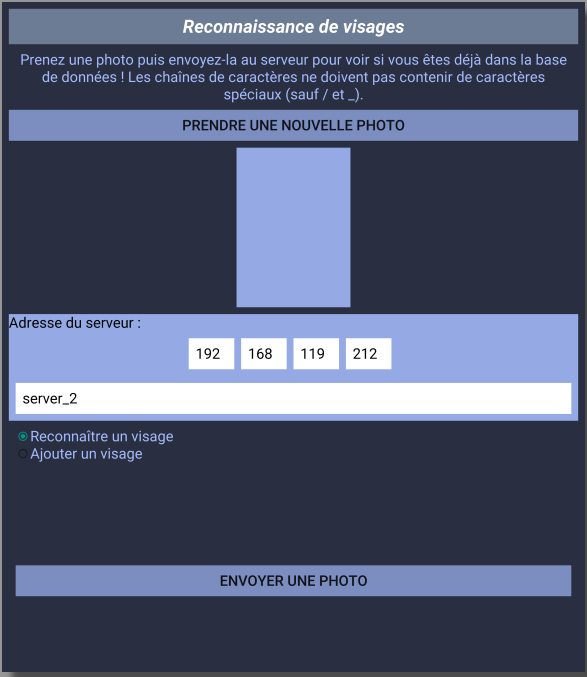
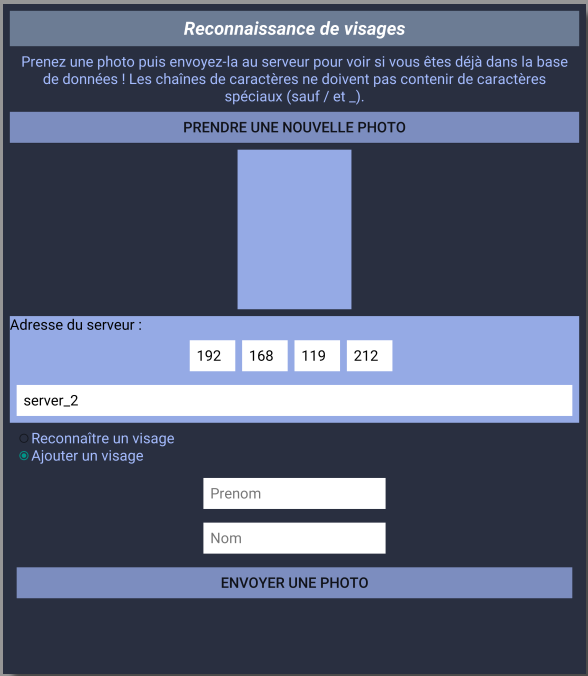


Figure 2 : Deuxième version de l’agencement. . Figure 3 : Dernière version de l’agencement.

Par la suite, la taille des différents éléments composants dépendra de l’appareil sur lequel sera lancée notre application. Nous avons aussi prévu la mise à jour des textes, selon la langue utilisée par la tablette.

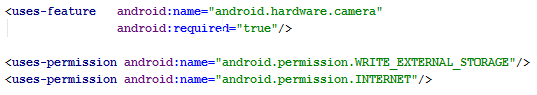
Etant donné que notre application doit aussi permettre l’ajout de nouvelles photos dans la base de données de visages (*Figure 3)*, nous avons ajouté une zone de sélection. Cela permet à l’utilisateur de choisir si la photo qu’il enverra devra lancer la procédure de reconnaissance de visage, ou bien celle d’ajout d’un visage dans la base. Si l’utilisateur choisit d’ajouter un visage, deux champs de texte apparaitront, ce qui permettra e saisir le nom et le prénom de la personne qui a été prise en photo.

La version finale de l’application tablette ressemble donc à cela :



# Implémentation

Nous avons choisi de développer notre application pour Android 4.0, afin de pouvoir effectuer des tests sur le matériel mis à notre disposition par Polytech, mais aussi sur notre matériel personnel. Afin de permettre l’accès à l’appareil photo et de sauvegarder celles-ci, il est nécessaire d’ajouter les permissions suivantes le fichier *AndroidManifest.xml* :



## Développement de l’interface graphique

Nous avons dans un premier temps placé les différents composants de notre interface graphique, selon le schéma que nous avons défini auparavant. Comme nous avons choisi, dès le départ, de gérer plusieurs tailles d’écran, nous avons créé trois fichiers de ressources Android différents, chacun contenant les tailles à appliquer aux composants :

|  |  |
| --- | --- |
| Nom du fichier | Description |
| value-small | Fichier de dimensions pour les écrans d’au moins 426dp x 320dp |
| values-normal | Fichier de dimensions pour les écrans d’au moins 470dp x 320dp |
| values-large | Fichier de dimensions pour les écrans d’au moins 640dp x 480dp |

Selon la taille de l’écran, le fichier de ressources correspondant est chargé lors de l’ouverture de l’application. De plus, les couleurs et les textes à utiliser seront définis dans les fichiers *styles.xml* et *strings.xml*, ce qui nous permet de changer plus facilement les valeurs des différents paramètres et de créer plusieurs fichiers de ressources, selon le thème ou la langue de l’application.

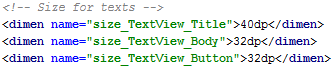


Figure 4 : Exemple de définition dans le fichier values-large.xml

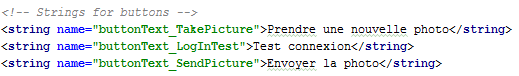


Figure 5 : Exemple de définition dans le fichier strings.xml

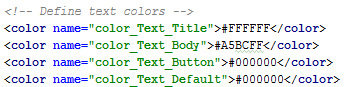


Figure 6 : Exemple de définition dans le fichier styles.xml

Afin de contrôler les choix de l’utilisateur, les champs de saisie d’adresses IP ont été paramétrés de façon à n’accepter que des nombres composés de 3 entiers au maximum. De même, les champs de saisie des paramètres ont eux aussi été paramétrés. De plus, en fonction du EditText dans lequel on se trouve, soit le bouton « suivant » soit le bouton « ok » est affiché.

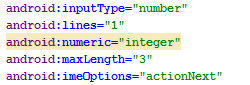


Figure 7 : Propriétés du EditText pour l'adresse IP.

## Développement des fonctionnalités

### Classe Toolbox

Nous avons créé une classe *ToolBox*, contenant des méthodes outils utilisées pour tester les données renseignées par l’utilisateur.

La méthode *checkIP()* vérifie si les quatre octets renseignés par l’utilisateur ont bien une valeur comprise entre 0 et 255, et qu’aucun champ n’est laissé vide.

De même, la méthode *checkName()* vérifie que les chaînes de caractères renseignées par l’utilisateur ne contiennent pas de caractères spéciaux.

Nous avons aussi créé une fonction *translateHTTPAnswer()*  afin de mettre en forme les réponses http reçue, avant leur affichage dans l’application. Pour cela, nous supprimons les balises HTML présentes en début et fin de la chaîne reçue.

### Classe MainActivity

Nous avons choisi de mettre en place une série de tests avant l’envoi de la photo au serveur. Pour cela, nous vérifions que :

* Il y a bien une photo à envoyer.
* L’adresse IP renseignée est valide.
* Le nom du serveur ne contient pas de caractères spéciaux.
* Le nom et prénom renseignés pour un ajout dans la base ne contiennent pas de caractères spéciaux.

L’envoi de la photo au serveur se fait grâce à une requête http post, l’image étant préalablement convertie au format Base64. En effet, ce format permet de coder la photo en utilisant un alphabet de 64 caractères, ce qui est plus pratique pour l’envoi au serveur.

Afin de conserver l’image lors de la rotation de l’écran, la méthode *onSaveInstanceState()* enregistre dans le chemin où est stocké l’image. Le chemin est ensuite récupéré et l’image affichée grâce à la méthode *onRestoreInstanceState()*.

# Quelques liens utiles

[Gestion de plusieurs tailles d’écran](http://developer.android.com/guide/practices/screens_support.html)

[Sauvegarder des données pour les recharger dans l’activity](http://www.techotopia.com/index.php/Saving_and_Restoring_Activity_State_in_Android_Studio#Saving_Dynamic_State) (par exemple pour conserver l’image lorsque l’écran change d’orientation)

[Gestion du clavier sous Android](http://www.tutos-android.com/utilisation-clavier-android)

[Utilisation des radios boutons](http://javatechig.com/android/android-radio-button-example)

[Site du développeur Android](http://developer.android.com/guide/index.html)

[Répertoire Git](https://github.com/PROJ942/Tablette2)