République Algérienne Démocratique et Populaire

Ministère de l'Enseignement Supérieur et de la Recherche Scientifique

Université M'Hamed Bougara – Boumerdès

Projet de module développement d'application mobile

L3 SI Grp03

Théme « application liste d'employés »

Encadré par :

BETIT

Réalisé par :

BOUABDALLAH Kheira 191931098370

KHELIFAOUI Djamel-Eddine 191931029343

CHERGUI Abd-El-Kader 171831061441

Introduction :

Les entreprises utilisent souvent des listes en papiers manuscrites ou des fichiers Excel pour gérer les employés et leurs informations.

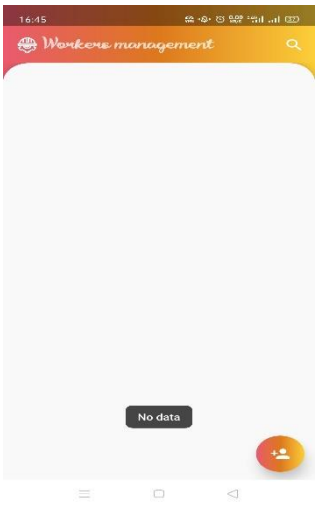
Dans ce mini projet on va développer une application mobile Android qui sert à gérer les employés au sein d'une entreprise pour faciliter la tâche pour l'utilisateur et aider le pour réduire les appréhensions et les charges mentales et aussi pour gagner du temps.

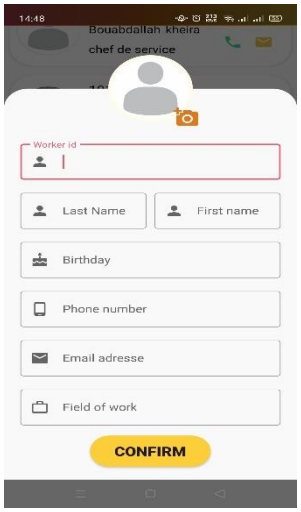
On a utilisé une base de données SQLite pour sauvegarder les informations de l'employé et l'application permettra à l'utilisateur de :

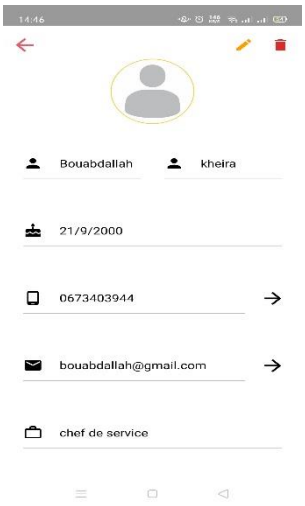
- Consulter la liste des employés.
- Ajouter, modifier ou supprimer un employé.
- Contacter un employé (par SMS, email, appel téléphonique…).
- Chercher dans la liste le nom ou l'identifiant de l'employé.

Pour cette application on a créé un design UX/UI qui s'adapte aux usages de l'utilisateur et améliore son expérience et fait une excellente première impression. Car cette conception de l'interface utilisateur de l'application mobile vise des interactions simples, agréables et efficaces entre les utilisateurs et l'application.

1. Présentation de l'application :

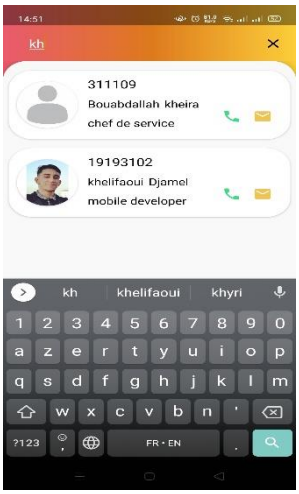


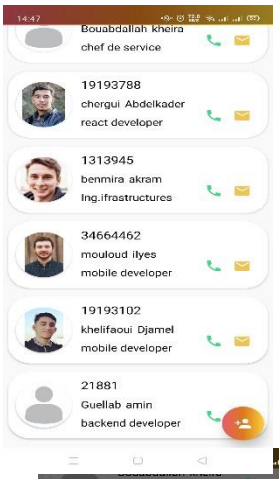




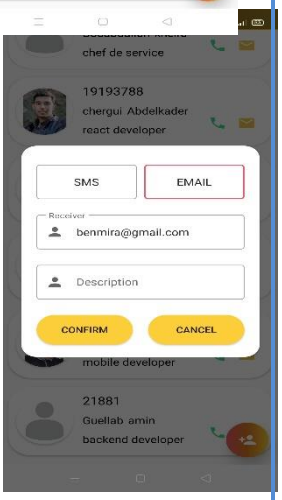
Notre application se présente comme illustrer sur les captures d'écrans à côté. Le premier affichage dés quand en rentrant dans l'application est une liste vide avec un buton « plus + » juste en bas.

Pour ajouter un employé en cliquant sur le button il s'affiche une boite dialogue à remplir avec les informations de l'employé "nom, prénom, email, photo, …" Qu'on veut l'ajouter et en bas un buton « confirm » pour confirmer ces informations et l'ajouter dans la lise cela implique l'ajout des informations dans la base de donnée.



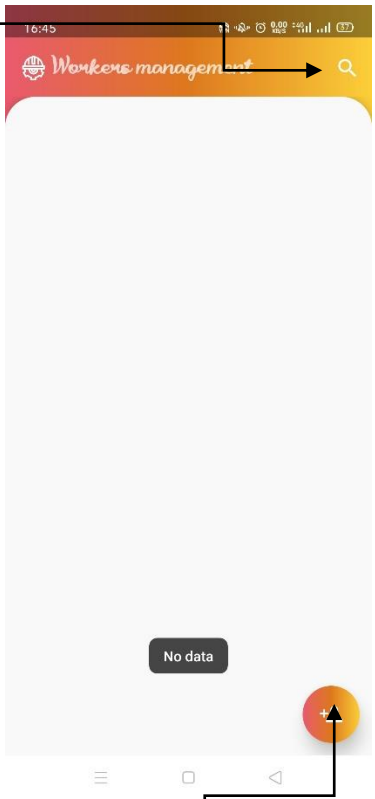


En ajoutant plusieurs employées ; on a plusieurs options qu'on peut les faire à volonté chercher avec le nom, prénom ou bien l'identifiant de l'employé dans la barre à recherche juste en haut de la liste, cliquer sur les icônes pour effectuer un appel téléphonique, envoyer un sms ou un e-mail, et aussi on peut cliquer sur la barre de l'employé pour afficher les informations même modifier ou supprimé à l'aide de deux icones un stylo pour modifier et une poubelle pour supprimer.



2. Présentation de chaque affichage :

❖ La fonction de recherche est codée en java par :




```
searchView.setOnSearchClickListener(view -> {
            appname.setVisibility(View.INVISIBLE);
            app_icone.setVisibility(View.INVISIBLE);
        });
        searchView.setOnQueryTextListener(new SearchView.OnQueryTextListener() {
            @Override
            public boolean onQueryTextSubmit(String query) {
                return false;
            }

            @Override
            public boolean onQueryTextChange(String text) {
                search_for(text);
                return true;
            }
        })
```

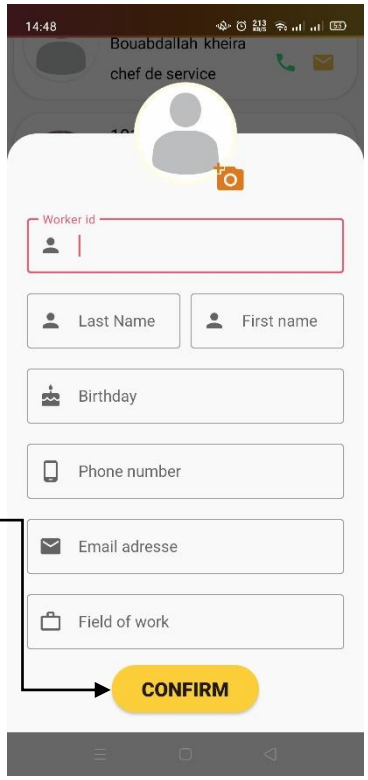
```
private void search_for(String query) {
    ArrayList<Worker> search_result=new ArrayList<>();
    for(Worker worker:workerList){
        if(worker.getId().toString().contains(query.toLowerCase()) ||
                worker.getFirst_name().toLowerCase().contains(query.toLowerCase()) ||
                worker.getLast_name().toLowerCase().contains(query.toLowerCase()) ){
            search_result.add(worker);
        }
    }
    if(search_result.isEmpty()){
        Toast.makeText(MainActivity.this,"sorry no exact matches found",Toast.LENGTH_SHORT);
    }
    else {
        adapter.setSearchResult(search_result);
    }
}
```

❖ La fonction utilisée pour ajouter un employé est codée en java par :

```
public void add_worker(View v) {
        scaled.setAnimationListener(new Animation.AnimationListener() {
            @Override
            public void onAnimationStart(Animation animation) {

            }

            @RequiresApi(api = Build.VERSION_CODES.N)
            @Override
            public void onAnimationEnd(Animation animation) {
                if (dialog != null) if (dialog.isShowing()) return;
                dialog = new BottomSheetDialog(MainActivity.this);
                LinearLayout layout = new LinearLayout(MainActivity.this);
                layout.addView(View.inflate(MainActivity.this, R.layout.add_worker, null));
                dialog.setContentView(layout);
                dialog.show();
                worker_id = dialog.findViewById(R.id.worker_id);
                last_name = dialog.findViewById(R.id.last_name);
                first_name = dialog.findViewById(R.id.first_name);
                email = dialog.findViewById(R.id.email_adresse);
                add_picture=dialog.findViewById(R.id.add_picture);
                picture=dialog.findViewById(R.id.profile);

                birthday= dialog.findViewById(R.id.birthday);
                birthday.setInputType(InputType.TYPE_NULL);
                birthday.setOnClickListener( view ->  {...});
                birthday.setOnFocusChangeListener((v1, hasFocus) ->  {...});
                phone_number = dialog.findViewById(R.id.phone_numberr);
                field_of_work = dialog.findViewById(R.id.field_of_work);
                confirm_btn = dialog.findViewById(R.id.confirm);
                confirm_btn.setOnClickListener(view ->  {...});
                add_picture.setOnClickListener(view ->  {...});
            }
```

❖ La fonction confirmer est codée en java par :

```
  confirm_btn.setOnClickListener(view -> {
                    if(!worker_id.getText().toString().isEmpty() &&
!last_name.getText().toString().isEmpty() && !first_name.getText().toString().isEmpty()&&
                            !email.getText().toString().isEmpty() &&
!phone_number.getText().toString().isEmpty() && !field_of_work.getText().toString().isEmpty() &&
                            !birthday.getText().toString().isEmpty()){
                        add();
                            workerList.clear();
                        storeDataInArrays();
                        adapter.notifyDataSetChanged(); }
                        else {
                            Toast.makeText(MainActivity.this, "u must enter all fields",
Toast.LENGTH_SHORT).show();
                           }

                });
```

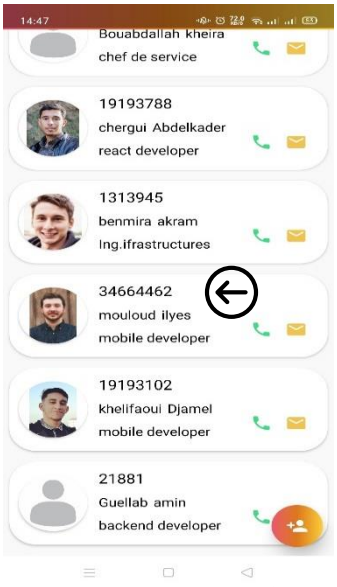


❖ cette fonction ajoute les informations de l'employé à la base de donnée utilisant la méthode addWorker :

```
   //ajouter l'employe dans la base de donner
    private void add() {
         worker=new
Worker(Long.parseLong(worker_id.getText().toString().trim()),last_name.getText().toString().trim(),firs
t_name.getText().toString().trim()

,phone_number.getText().toString().trim(),email.getText().toString().trim(),field_of_work.getText().toS
tring().trim(),birthday.getText().toString().trim(),ImageToBitmap(picture));
        mDBhandler.addWorker(worker);
        dialog.cancel();

    }
```

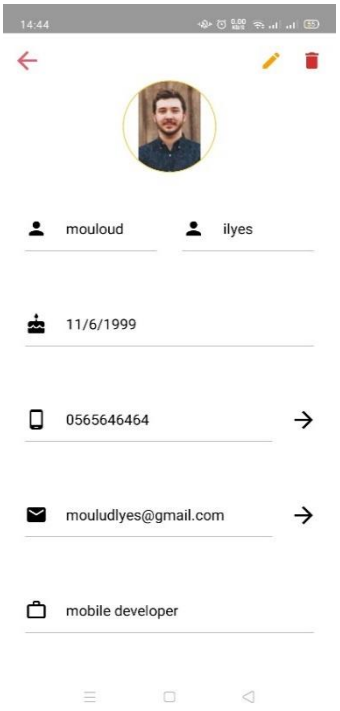
- Affecter le recyclerview adapter dans le recyclerview layout:

```
RecyclerView recyclerView = findViewById(R.id.rv);
LinearLayoutManager manager = new LinearLayoutManager(MainActivity.this);
recyclerView.setLayoutManager(manager);
recyclerView.setAdapter(adapter);
```



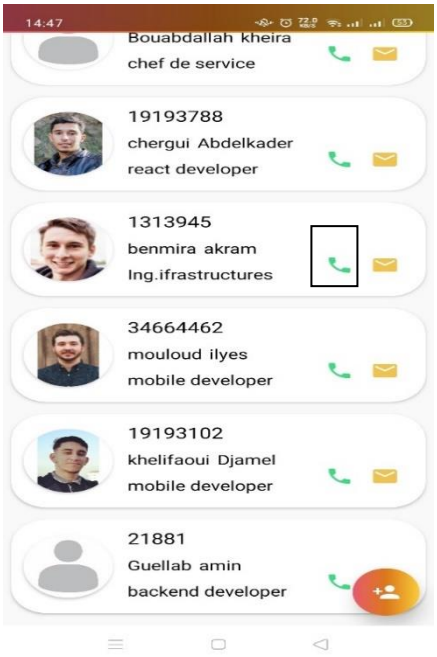
- la fonction quand on clique sur l'un des emplyer l'application méne vers le profil d'employé :

```
// instancier database
      mDBhandler = new DBhandler(MainActivity.this);
      workerList = new ArrayList<>();
      // remplir les donnée dans workerLIst avec la fonction storeDataArrays
      storeDataInArrays();
      // instancier l'adapter
      adapter = new RecyclerViewAdapter(workerList, new RecyclerViewAdapter.OnWorkerListener() {
          @Override
          public void onWorkerClick(int position,View v) {
              // affecter une animation dans floating button
              scaled.setAnimationListener(new Animation.AnimationListener() {
                  @Override
                  public void onAnimationStart(Animation animation){...}
                 @Override
                  public void onAnimationEnd(Animation animation) {
                      // deplacer vers l'activity worker profile
                      Intent i = new Intent(MainActivity.this, Worker_profile.class);
                      i.putExtra("id", String.valueOf(workerList.get(position).getId()));
                      startActivity(i);
                  }

                  @Override
                public void onAnimationRepeat(Animation animation) {}
                 });
              v.startAnimation(scaled);

          }
```



❖ La fonction qui fait un appel téléphonique à l'employé On clique sur l'icône de téléphone :

```
/*recupérer le numéro de telephone est appelé l'employé
a partir de recycle view adapter */
@Override
 public void onWorkerCall(int position) {
     Uri uri = Uri.parse("tel:"+workerList.get(position).getPhone_number());
     Intent intent = new Intent(Intent.ACTION_DIAL, uri);
     startActivity(intent);
 }
```



❖ Le code de la fonction suivante gère à la fois les appels téléphoniques et les sms, en cliquant sur la flèche de numéro de téléphone :

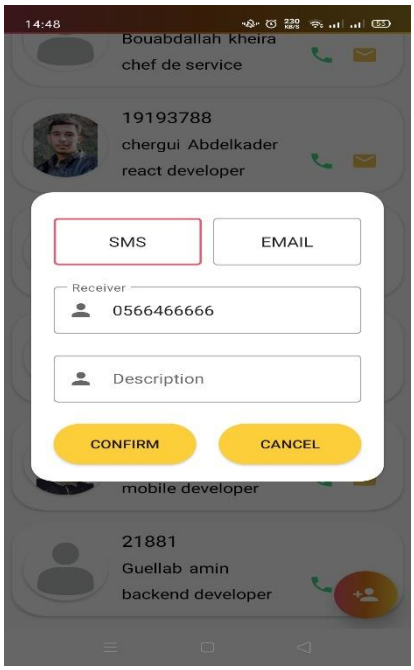
```
// afficher un dialog pour manipuler les action d'appel est sms
   phone_arrow.setOnClickListener(v-> {
              AlertDialog.Builder dialogBuilder = new AlertDialog.Builder(Worker_profile.this);
              LayoutInflater inflater = Worker_profile.this.getLayoutInflater();
              View dialogView = inflater.inflate(R.layout.dialog_sms_call, null);
              dialogBuilder.setView(dialogView);
              AlertDialog alertDialog = dialogBuilder.create();
              alertDialog.getWindow().setBackgroundDrawable(new ColorDrawable(Color.TRANSPARENT));
              alertDialog.show();
              sms_receiver=dialogView.findViewById(R.id.sms_receiver);
              sms_description=dialogView.findViewById(R.id.sms_content);
              sms_receiver.setText(phone_number.getText().toString());
              call=dialogView.findViewById(R.id.confirm_call);
              send_sms=dialogView.findViewById(R.id.confirm_sms);
              exit_dialog=dialogView.findViewById(R.id.exit_dialog);
              exit_dialog.setOnClickListener(view ->{
                  alertDialog.cancel();
              });
              //recupérer le numéro de telephone est appelé
              call.setOnClickListener(view ->{
                  Uri uri = Uri.parse("tel:"+phone_number);
                  Intent intent = new Intent(Intent.ACTION_DIAL, uri);
                  startActivity(intent)  });
              //recupérer le numéro de telephone est envoyer un sms
              send_sms.setOnClickListener(view ->{
                  Uri uri=Uri.parse("sms:"+phone_number.getText().toString());//xxxxxxxx représente
                    un numéro de téléphone
                  Intent intent = new Intent(Intent.ACTION_VIEW, uri);
                  intent.putExtra("sms_body",sms_description.getText().toString());
                  startActivity(intent);
              });
          });
```

❖ Explication de l'importation de l'image à partir de Storage et l'affectation au imageView :

```
//afficher l'activity de crop image pour recupérer l'image
    private void PickImage() {
        CropImage.activity()
                .start(this);
    }
   //convertir l'image au tableau de byte
    private byte[] ImageToBitmap_toArray(ImageView picture) {

        Bitmap bitmap=((BitmapDrawable) picture.getDrawable()).getBitmap();
        ByteArrayOutputStream stream=new ByteArrayOutputStream();

        bitmap.compress(Bitmap.CompressFormat.PNG,50,stream);
        byte[] bytes=stream.toByteArray();
        //bitmap.recycle();
        return bytes;
    }
    //recupérer l'image apartir l'activity de crop image
    @Override
    public void onActivityResult(int requestCode, int resultCode, Intent data) {
        super.onActivityResult(requestCode, resultCode, data);
        if (requestCode == CropImage.CROP_IMAGE_ACTIVITY_REQUEST_CODE) {
            CropImage.ActivityResult result = CropImage.getActivityResult(data);
            if (resultCode == RESULT_OK) {
                Uri resultUri = result.getUri();
                try {
                    InputStream stream=getContentResolver().openInputStream(resultUri);
                    Bitmap bitmap= BitmapFactory.decodeStream(stream);
                    profile_picture.setImageBitmap(bitmap);
                } catch (FileNotFoundException e) {
                    e.printStackTrace();
                }
            } else if (resultCode == CropImage.CROP_IMAGE_ACTIVITY_RESULT_ERROR_CODE) {
                Exception error = result.getError();
            }
        }
    }
```

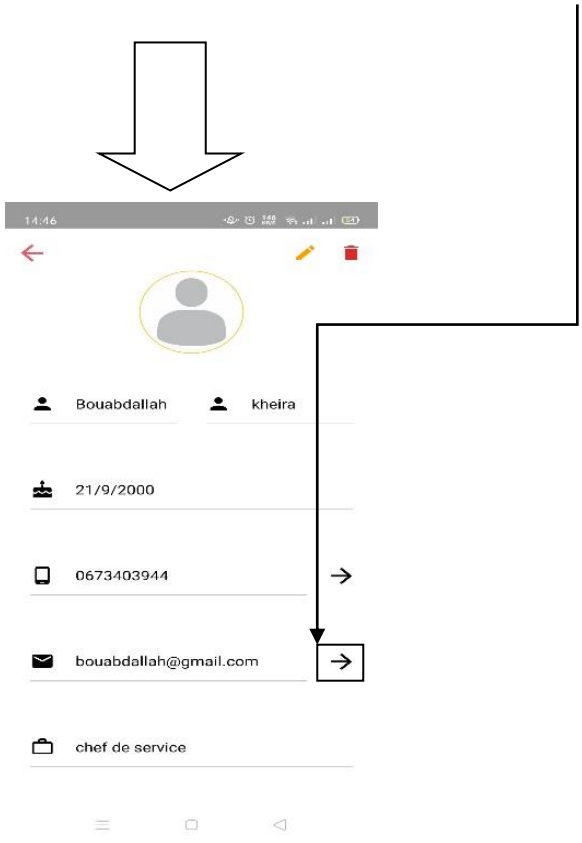
❖ La fonction qui envoie un sms ou e-mail vers l'employé en cliquant sur l'icône d'une lettre :

```
        //recupérer le numéro de telephone est envoyes un sms
        //redéfinir la fonction de l'interface qu'on a deja implémenté dans le recycle view adapter
        @Override
        public void onWorkerSend(int position) {
            AlertDialog.Builder dialogBuilder = new AlertDialog.Builder(MainActivity.this);

            LayoutInflater inflater = MainActivity.this.getLayoutInflater();
            View dialogView = inflater.inflate(R.layout.send_dialog, null);
            dialogBuilder.setView(dialogView);

            sms_choice = dialogView.findViewById(R.id.sms_choice);
            email_choice = dialogView.findViewById(R.id.email_choice);
            receiver=dialogView.findViewById(R.id.receiver);
            email_choice.setInputType(InputType.TYPE_NULL);
            sms_choice.setInputType(InputType.TYPE_NULL);
            receiver.setInputType(InputType.TYPE_NULL);
            change_focus(position);
            Button cancel_sending=dialogView.findViewById(R.id.cancel_sending);
            Button confirm_sending=dialogView.findViewById(R.id.confirm_sending);
            EditText description=dialogView.findViewById(R.id.content);
            change_sender(position);
            AlertDialog alertDialog = dialogBuilder.create();
            alertDialog.getWindow().setBackgroundDrawable(new ColorDrawable(Color.TRANSPARENT));
            alertDialog.show();
            confirm_sending.setOnClickListener(v ->{
                if(sms_choice.hasFocus()){
                    Uri uri=Uri.parse("sms:"+receiver.getText().toString());
                  //xxxxxxxx représente un numéro de téléphone
                    Intent intent = new Intent(Intent.ACTION_VIEW, uri);
                    String message=description.getText().toString();
                    intent.putExtra("sms_body",message );
                    startActivity(intent);
                }
                else if(email_choice.hasFocus()){
                    Intent email = new Intent(Intent.ACTION_SEND);
                    email.putExtra(Intent.EXTRA_EMAIL, new String[]{receiver.getText().toString()});
                    email.putExtra(Intent.EXTRA_TEXT, description.getText().toString());

                    //need this to prompts email client only
                    email.setType("message/rfc822");

                    startActivity(Intent.createChooser(email, "Choose an Email client :"));
                }
            });
            cancel_sending.setOnClickListener(view -> {
                alertDialog.cancel();
            });
        }
    });
```



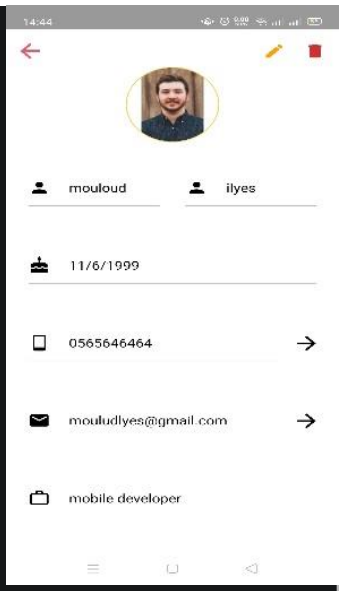
❖ La fonction de l’envoie d’e-mail par la clique sur la flèche de l’adresse mail :

```
// afficher un dialog pour manipuler l'action d'envoyer email
email_arrow.setOnClickListener(v-> {
    AlertDialog.Builder dialogBuilder = new AlertDialog.Builder(Worker_profile.this);

    LayoutInflater inflater = Worker_profile.this.getLayoutInflater();
    View dialogView = inflater.inflate(R.layout.dialog_email, null);
    dialogBuilder.setView(dialogView);
    AlertDialog alertDialog = dialogBuilder.create();
    alertDialog.getWindow().setBackgroundDrawable(new ColorDrawable(Color.TRANSPARENT));
    alertDialog.show();
    email_receiver=dialogView.findViewById(R.id.email_receiver);
    email_description=dialogView.findViewById(R.id.email_content);
    email_receiver.setText(email.getText().toString());
    subject=dialogView.findViewById(R.id.subject);
    send_email=dialogView.findViewById(R.id.confirm_email);
    cancel_email=dialogView.findViewById(R.id.cancel_email);
    cancel_email.setOnClickListener(view ->{  alertDialog.cancel(); });
    send_email.setOnClickListener(view ->{
        Intent intent = new Intent(Intent.ACTION_SEND);
        intent.putExtra(Intent.EXTRA_EMAIL, new String[]{email.getText().toString()});
        intent.putExtra(Intent.EXTRA_SUBJECT, subject.getText().toString());
        intent.putExtra(Intent.EXTRA_TEXT, email_description.getText().toString());
        //need this to prompts email client only
        intent.setType("message/rfc822");
        startActivity(Intent.createChooser(intent, "Choose an Email client :"));
    });
});
```



❖ La fonction dédiée à remplir les informations de l'employé :

```
//initialisation de l'instance "db" de la clasee DBhandler
        db = new DBhandler(Worker_profile.this);
        //recupérer l'employé selectionné par "id"
        String id = getIntent().getStringExtra("id");

        worker = db.getWorker(id);   //getWorker est la méthode de la classe DBhandler

        //remplir les champs de profile d'employé

        phone_number.setText(worker.getPhone_number());
        name.setText(worker.getLast_name() );
        first_name.setText(worker.getFirst_name());
        birthday.setText(worker.getDate());
        work_field.setText(worker.getField());
        email.setText(worker.getEmail());
        /*recupérer l'image apartir de sqlite sous la forme blob est tranformé la au bitmap
        pour l'afficher */
        profile_picture.setImageBitmap(BitmapFactory.decodeByteArray(worker.getImage(), 0, worker.getImage().length));
```

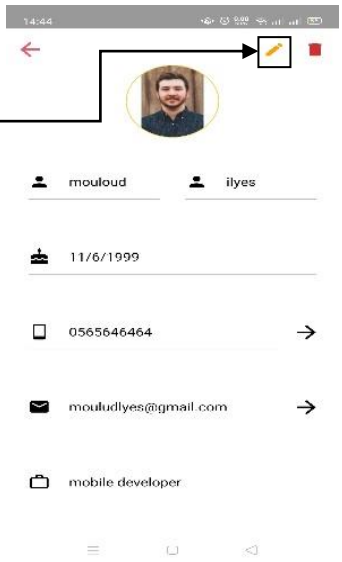


❖ La fonction qui supprime un employer de la liste en cliquant sur l'icône « poubelle » :

```
//supprime un employé avec la fonction deleteWorker de la classe DBhandler
        supprimer.setOnClickListener(v -> {
            db.deleteWorker(id);
            finish();
        });
```

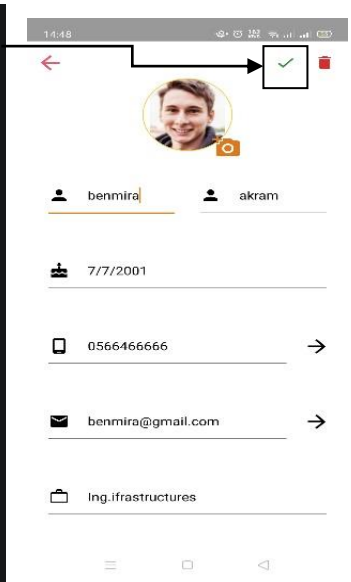
- La fonction d'activation de l'état de modification quand l'utilisateur clique sur l'icône « stylo » :

```
private void change0() {
      modifier.setOnClickListener(view -> {

          add_picture.setVisibility(View.VISIBLE);
          first_name.setEnabled(true);
          name.setEnabled(true);
          email.setEnabled(true);
          phone_number.setEnabled(true);
          work_field.setEnabled(true);
          birthday.setEnabled(true);
          add_picture.setEnabled(true);

          modifier.setImageResource(R.drawable.ic_check);
          change();

      });
  }
```



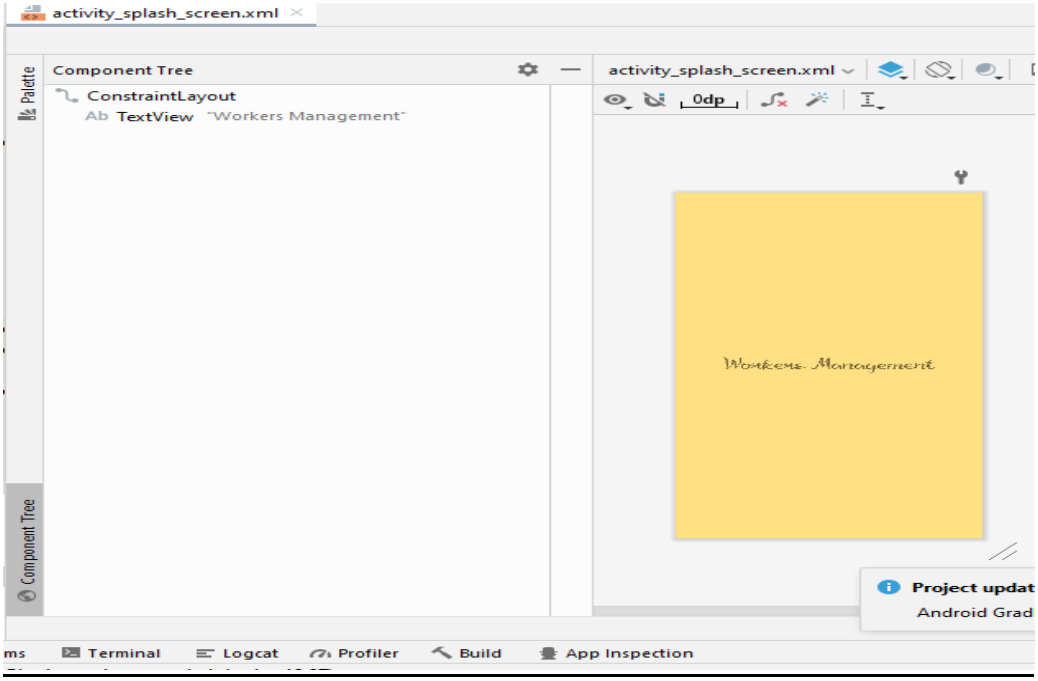
❖ cette fonction vient juste après le clique sur le stylo de modifier elle modifie les informations de l'employé et les confirmer par la méthode apdateWorker de BDD.

```
private void change() {
      add_picture.setOnClickListener(view -> PickImage());
      modifier.setOnClickListener(v ->{
        if (!name.getText().toString().isEmpty() && !first_name.getText().toString().isEmpty() &&
                !email.getText().toString().isEmpty() && !phone_number.getText().toString().isEmpty()
                && !work_field.getText().toString().isEmpty() &&
                !birthday.getText().toString().isEmpty()) {
            birthday.setInputType(InputType.TYPE_NULL);
            birthday.setOnClickListener(view -> {
                DatePickerDialog dialog = new DatePickerDialog(Worker_profile.this);
                dialog.show();
                dialog.setOnDismissListener(dialogInterface ->
                        birthday.setText(dialog.getDatePicker().getDayOfMonth() + "/"
                        + (dialog.getDatePicker().getMonth() + 1) + "/"
                        + dialog.getDatePicker().getYear()));

            });
            birthday.setOnFocusChangeListener((v1, hasFocus) -> {...});
            worker = new Worker(worker.getId(), name.getText().toString().trim(),
            first_name.getText().toString().trim()
                    , phone_number.getText().toString().trim(), email.getText().toString().trim(),
                      work_field.getText().toString().trim(),
                       birthday.getText().toString().trim(), ImageToBitmap(profile_picture));
            db.updateWorker(worker);
            phone_number.setText(worker.getPhone_number());
            name.setText(worker.getLast_name());
            first_name.setText(worker.getFirst_name());
            birthday.setText(worker.getDate());
            email.setText(worker.getEmail());
            work_field.setText(worker.getField());
            Bitmap bm = BitmapFactory.decodeByteArray(ImageToBitmap(profile_picture), 0,
            ImageToBitmap(profile_picture).length);
            profile_picture.setImageBitmap(bm);
            modifier.setImageResource(R.drawable.ic_edit);
            first_name.setEnabled(false);
            name.setEnabled(false);
            email.setEnabled(false);
            phone_number.setEnabled(false);
            work_field.setEnabled(false);
            birthday.setEnabled(false);
            add_picture.setEnabled(false);
            add_picture.setVisibility(View.INVISIBLE);


        } else {
            Toast.makeText(Worker_profile.this, "u must enter all fields", Toast.LENGTH_SHORT).show();
        }
       change0();
    });
  }
```

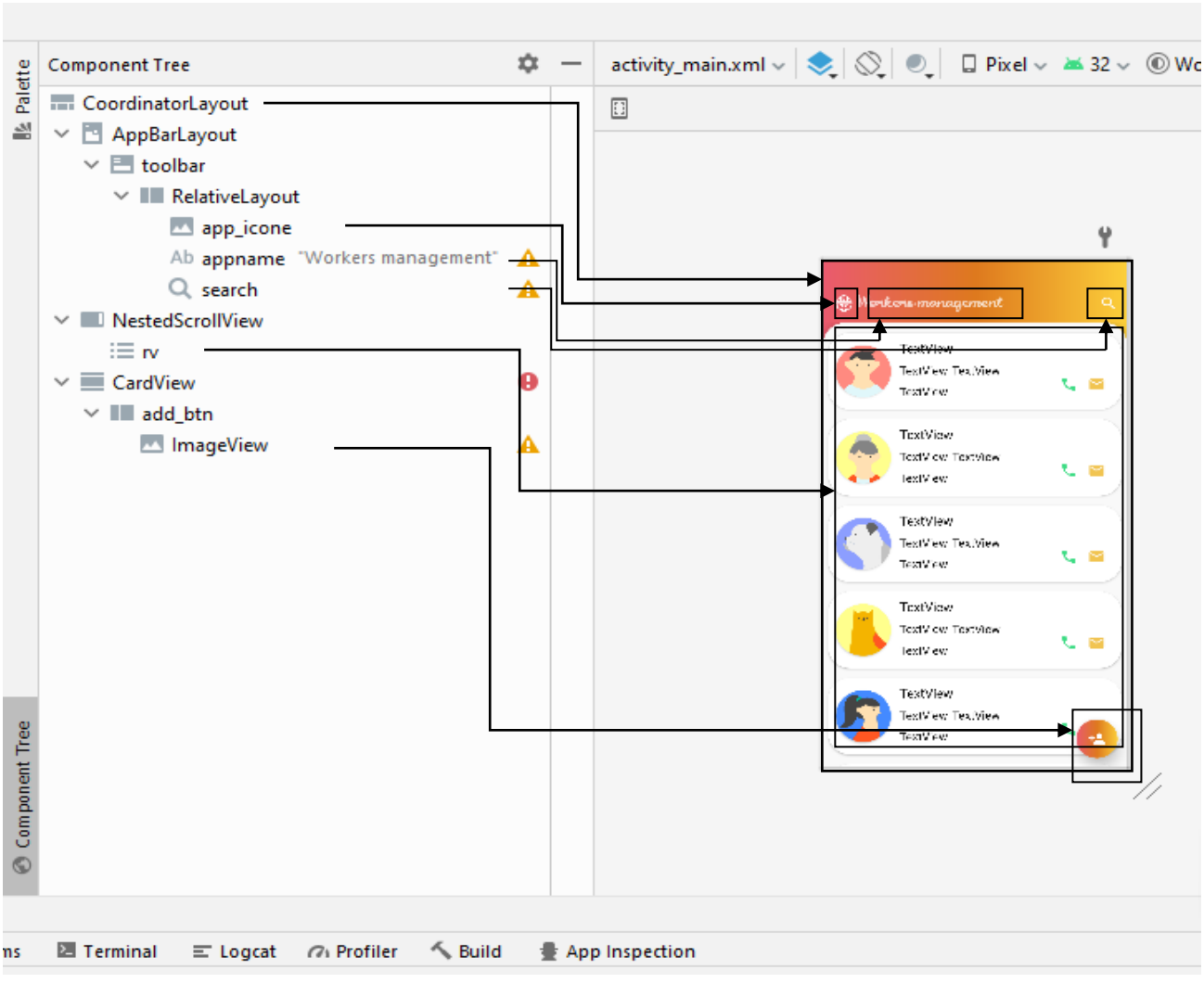


3. Xml :

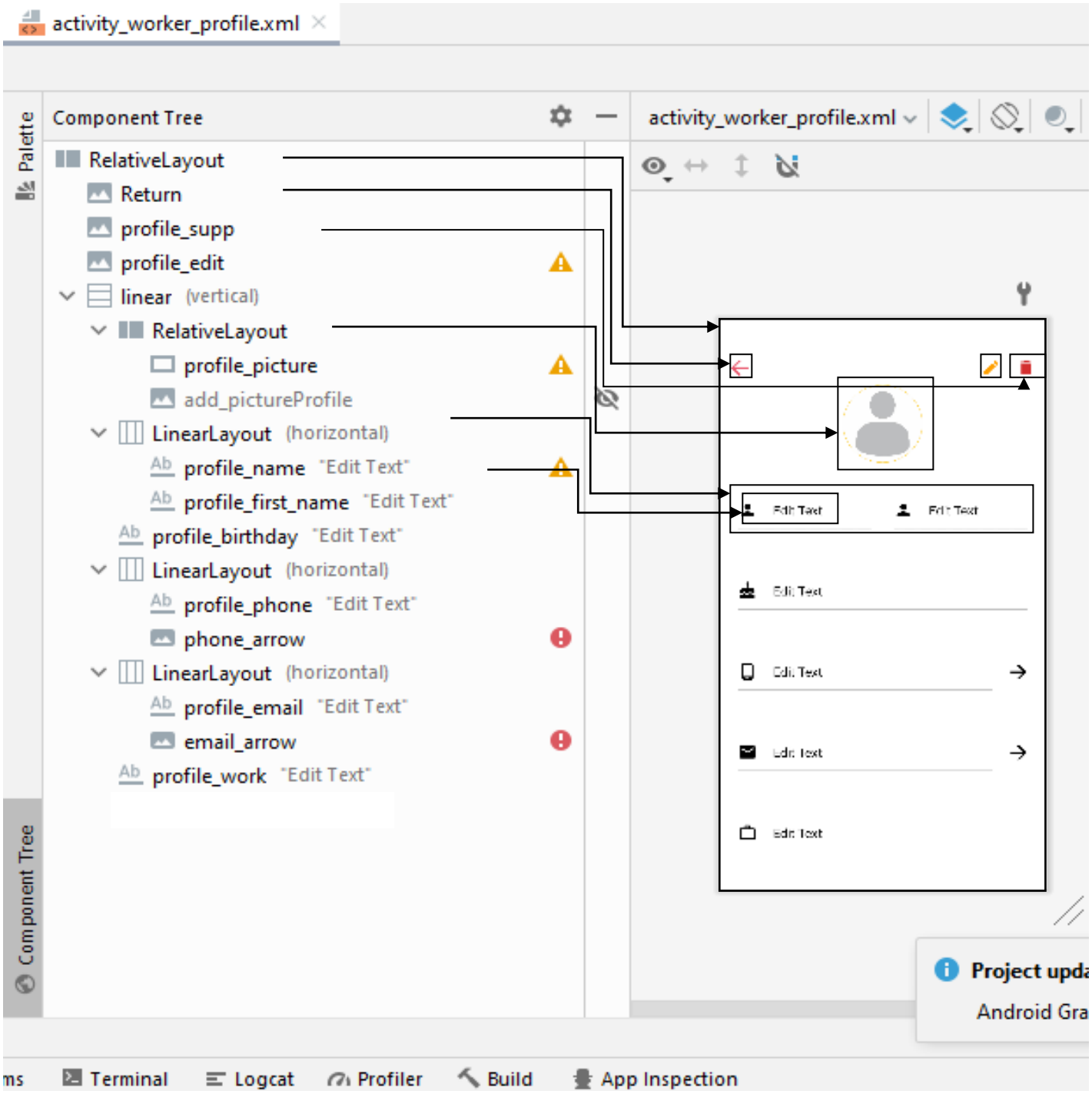
1er affichage



2éme affichage :



3éme affichage :



4éme affichage :