ỨNG DỤNG LÀM NAIL 4.0

[**I.** **KHẢO SÁT HỆ THỐNG** 3](#_Toc520249725)

[**1.** **Mục đích, yêu cầu của ứng dụng** 3](#_Toc520249726)

[**II.** **PHÂN TÍCH HỆ THỐNG.** 3](#_Toc520249727)

[**1.** **Đối với Ứng dụng** 3](#_Toc520249728)

[**2.** **Đối với người quản trị** 3](#_Toc520249729)

[**III.** **THIẾT KẾ HỆ THỐNG (CƠ SỞ SỮ LIỆU).** 3](#_Toc520249730)

[**1.** **Xác định các thực thể.** 5](#_Toc520249731)

[**IV.** **GIAO DIỆN CHƯƠNG TRÌNH.** 8](#_Toc520249732)

[1. Giao diện đăng nhập 8](#_Toc520249733)

[2. Giao diện chính 10](#_Toc520249734)

[3. Chức năng xem “Hồ sơ cá nhân” 12](#_Toc520249735)

[4. Chức năng “Đăng Tuyển” 13](#_Toc520249736)

[5. Chức năng đăng tìm việc 16](#_Toc520249737)

[6. Chức năng xem danh sách công việc 17](#_Toc520249738)

[7. Chức năng xem danh sách ứng viên. 18](#_Toc520249739)

[8. Chức năng báo cáo lỗi, góp ý. 19](#_Toc520249740)

[**V.** **Quản lý trên server** 20](#_Toc520249741)

[VI. **TỔNG KẾT.** 22](#_Toc520249742)

1. **KHẢO SÁT HỆ THỐNG**
2. **Mục đích, yêu cầu của ứng dụng**

Đây là ứng dụng tuyển dụng và tìm việc thu nhỏ, hay đơn giản hơn là ứng dụng tìm việc làm lĩnh vực “Làm Đẹp”

1. **PHÂN TÍCH HỆ THỐNG.**
2. **Đối với Ứng dụng**

* Hiển thị danh sách công việc
  + Hiển Thị danh sách ứng viên
  + Thêm/sửa/xóa việc làm tuyển dụng.
  + Thêm/sửa xóa thông tin tìm việc.
  + Xem thông tin tuyển dụng/tìm việc

1. **Đối với người quản trị**

* Cập nhật ,Thêm, Sửa, Xóa Công việc/ứng viên.
* Nhận được thông báo lỗi, tìm cách khắc phục.

1. **THIẾT KẾ HỆ THỐNG (CƠ SỞ SỮ LIỆU).**

**FireBase** có thể rất mạnh mẽ đối với ứng dụng backend, nó bao gồm việc lưu trữ dữ liệu, xác thực người dùng, static hosting......Giúp lập trình viên chỉ cần chú tâm đến việc nâng cao trải nghiệm người dùng.

**Firebase Realtime Database**

* Dữ liệu trong cơ sở dữ liệu Firebase của bạn được lưu trữ dưới dạng JSON và đồng bộ realtime đến mọi kết nối client. Khi bạn xây dựng những ứng dụng đa nền tảng như Android, IOS và JavaScrip SDKs, tất cả các client của bạn sẽ chia sẻ trên một cơ sở dữ liệu Firebase và tự động cập nhật với dữ liệu mới nhất.
* Tự động tính toán quy mô ứng dụng của bạn.
* Khi ứng dụng của bạn muốn phát triển, bạn không cần lo lắng về việc nâng cấp máy chủ...Firebase sẽ xử lý việc tự động cho bạn. Các máy chủ của Firebase quản lý hàng triệu kết nối đồng thời và hàng tỉ lượt truy vấn mỗi tháng.
* Các tính năng bảo mật lớp đầu
* Tất cả dữ liệu được truyền qua một kết nối an toàn SSL với một chứng nhận 2048-bit. Cở sở dữ liệu truy vấn và việc xác nhận  được điều khiển tại một cấp độ chi tiết sử dụng theo một số các quy tắc mềm dẻo [security rules language](https://www.firebase.com/docs/security/quickstart.html).  Tất cả các logic bảo mật dữ liệu của bạn được tập trung ở một chỗ để dễ dàng cho việc cập nhật và kiểm thử.
* Làm việc offline
* Ứng dụng Firebase của bạn sẽ duy trì tương tác bất chấp một số các vấn đề về internet xảy ra. Trước khi bất kỳ dữ liệu được ghi đến server thì tất  cả dữ liệu lập tức sẽ được viết vào một cơ sử dữ liệu Firebase ở local. Ngay khi có thể kết nối lại, client đó sẽ nhận bất kỳ thay đổi mà nó thiếu và đồng bộ hoá nó với trạng thái hiện tại server.

**Xác thực người dùng**

* Với Firebase, ta có thể **dễ dàng xác thực người dùng** từ ứng dụng của bạn trên Android, iOS và JavaScript SDKs chỉ với một vài đoạn mã. Firebase đã xây dựng chức năng cho việc xác thực người dùng với Email, Facebook, Twitter, GitHub, Google, và xác thực nặc danh. Các ứng dụng sử dụng chức năng xác thực của FireBase có thể giải quyết được vấn đề khi người dùng đăng nhập, nó sẽ **tiết kiệm thời gian** và rất nhiều các vấn đề phức tạp về phần backend. Hơn nữa bạn có thể tích họp xác thực người dùng với các chức năng backend đã có sẵn sử dụng  [**custom auth tokens**](https://www.firebase.com/docs/web/guide/login/custom.html)**.**

**Firebase Hosting**

* Phát triển ứng dụng web của bạn trong thời gian ngắn với các hosting tĩnh đã được cung cấp sẵn. Tất cả các kết nối được phân phối qua SSL từ CDN trên toàn thể giới của Firebase
* Triểu khai siêu tốc:Việc triển khai sử dụng các công cụ dòng lệnh Firebase  và có thể quay trở lại với phiên bản trước chỉ với một cú click chuột.
* SSL bởi default:Mọi ứng dụng được xử lý thông qua một kết nối an toàn, và Firebase đã cẩn thận cung cấp SSL cert cho bạn.

**Firebase Crashlytics**

* Trong Android, có một xác suất cao cho ứng dụng của bạn bị sự cố ngay cả sau khi bạn đã kiểm thử nó theo cách thủ công và cách tự động. Điều này có thể do các yếu tố như sự không tương thích với các level API khác nhau, kích cỡ màn hình, bộ nhớ điện thoại, cảm biến và sự tùy biến của các nhà cung cấp để cho phù hợp với từng thiết bị. Crashlytics sẽ giúp bạn chẩn đoán các vấn đề này.
* Lợi ích: Dễ dàng để thiết lập. Không giới hạn ứng dụng, người dùng, và số lần treo, cảnh báo theo thời gian thực, báo cáo sự cố theo thời gian thực, phân tích sâu sự cố, theo dõi ngoại lệ, dễ dàng chia sẻ báo cáo sự cố.

**Thiết kế CSDL.**

* Cơ sở dữ liệu nhằm lưu giữ thông tin cần thiết cho chương trình. Với mỗi một cơ sở dữ liệu sẽ thể hiện rõ thông tin tối thiểu cần có của việc quản lý.
* Để lưu trữ CSDL và thực hiện các thao tác, cũng như thuật toán khác nhau vớiứng dụng thì cần phải tìm những cấu trúc dữ liệu thích hợp để mô tả quá trình hoạt động của ứng dụng.
* Để tạo một cơ sở dữ liệu, trước hết phải xác định những thông tin gì cần thiết.
* Sau đó ta thiết kế, tạo các đối tượng sẽ lưu.
* Sau khi tạo ra cấu trúc cơ sở dữ liệu, cơ sở dữ liệu có thể chứa dữ liệu dưới dạng bản ghi.

1. **Xác định các thực thể.**

Bước đầu trong phân tích thiết kế cơ sở dữ liệu là phân tích các yêu cầu của dữ liệu, tức là ta phải xác định được các thông tin cần quản lý để xác đinh ra các đối tượng cần thiết lưu trữ dữ liệu vào bảng.

* Job: Hiển thị danh sách thông tin công việc như : Tên công việc, Số lượng cần tuyển, Nơi làm việc, Mức lương, Ngày bắt đầu, Ngày kết thúc, Mô tả công việc, Hình ảnh.
* Employee: Hiển thị danh sách thông tin ứng viên như : Tên ứng viên, Loại công việc ứng tuyển, Hình ảnh, Mức lương mong muốn, Ngày đăng tìm việc, Kinh nghiệm, Giới thiệu bản thân.
* Customers: Hiển thị thông tin khách hang như: Tên Khách Hàng, Địa Chỉ Khách Hang, Số Điện Thoại, Mail Khách Hàng, Giới Tính, Ngày Sinh.

Cơ sở dữ liệu **NoSQL** (tên gốc là "Non SQL" (phi SQL) hoặc "non relational" (phi quan hệ)) cung cấp một cơ chế để lưu trữ và truy xuất dữ liệu được mô hình hóa khác với các quan hệ bảng được sử dụng trong các cơ sở dữ liệu kiểu quan hệ.

Xuất phát từ quá trình xác định các thực thể như trên ta có thể xây dựng một CSDL trên Firebase có các Collection sau

* 1. Job Collection:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Document | Field | Value | Value trong model | Ghi chú |
| Unique ID | Avatar | String | Bitmap | Hình đại diện |
| tenCongViec | String | String | Tên công việc |
| soLuong | String | int | Số lượng cần tuyển |
| noiLamViec | String | String | Địa điểm làm việc |
| mucLuong | String | String | Mức lương |
| ngayBatDau | String | Calendar | Ngày bắt đầu |
| ngayKetThuc | String | Calendar | Ngày kết thúc |
| moTa | String | String | Mô tả công việc |
| latLng | String | LatLng | Tọa độ địa điểm |
| tieuChuan | String | Int | Tiêu chuẩn công việc |

* 1. Employee Collection:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Document | Field | Value | Value trong model | Ghi chú |
| Unique ID | tenUngVien | String | String | Tên ứng viên |
| loaiCongViec | String | String | Loại công việc |
| diaChiUngVien | String | String | Địa điểm ứng viên |
| kinhNghiem | String | String | Kinh nghiệm |
| mucLuongMongMuon | String | String | Mức lương |
| ngayDangTimViec | String | Calendar | Ngày đăng tìm việc |
| avatar | String | Bitmap | Hình đại diện |
| gioiThieuBanThan | String | String | Mô tả bản thân |
| latLng | String | LatLng | Tọa độ địa điểm |
| tieuChuan | String | Int | Tiêu chuẩn đánh giá |

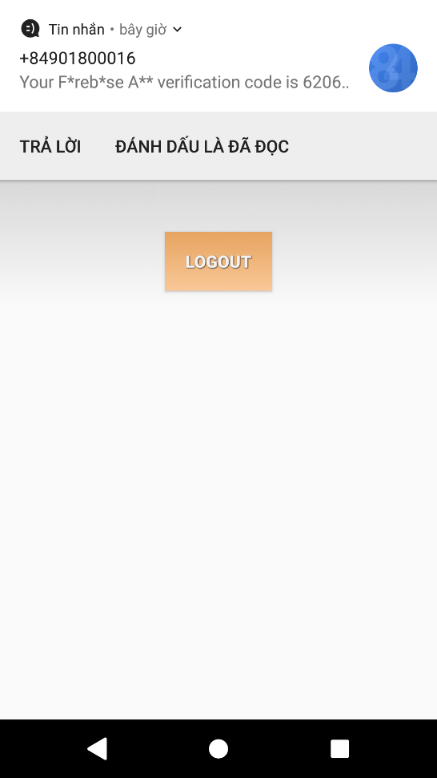
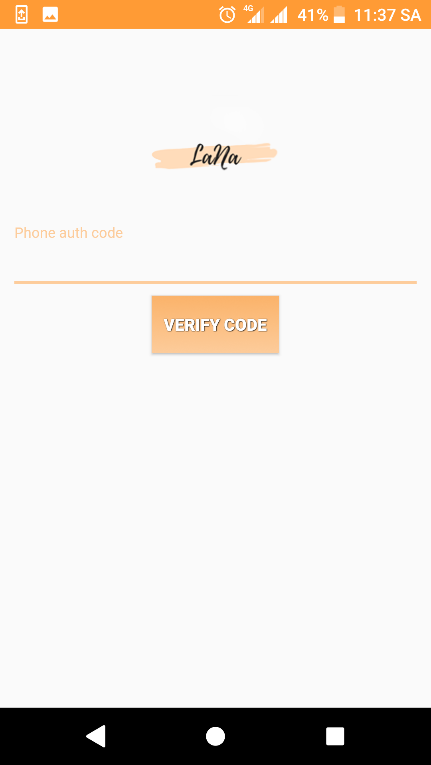
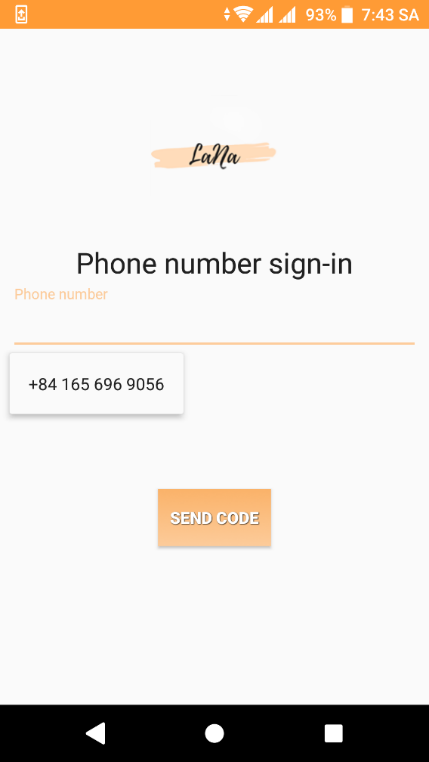
* 1. Report Collection:

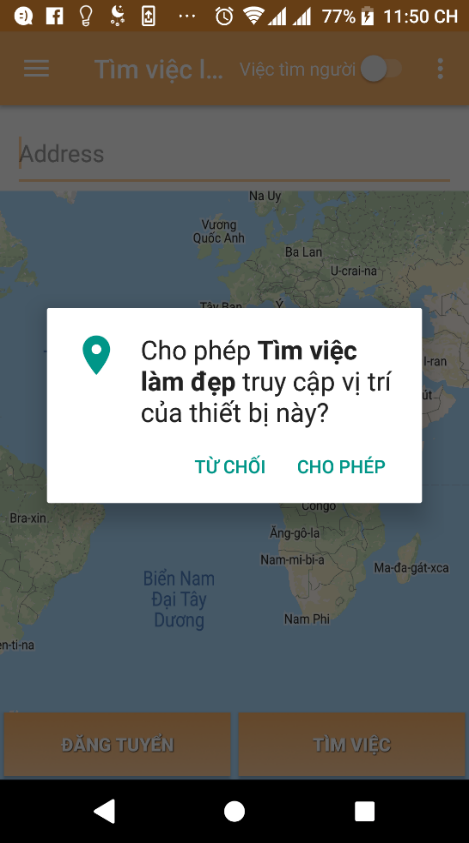
|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Document | Field | Value | Value trong model | Ghi chú |
| Unique ID | thongTinReport | String | String | Thông tin lỗi |
| tenNguoiDung | String | String | Tên người báo cáo |
| sdtNguoiDung | String | String | SDT người báo cáo |
| timeStamp | String | Calendar | Thời gian gửi |

* 1. PhoneAuth Collection:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Document | Field | Value | Value trong model | Ghi chú |
| Unique ID | Phone | String | String | Số điện thoại người dùng |
| timeStamp | String | Calendar | Thời gian đăng nhập |
| VerifycationID | String | String | Mã xác thực lượt truy cập |

1. **GIAO DIỆN CHƯƠNG TRÌNH.**
   1. Giao diện đăng nhập



* Đăng nhập thông qua thư viện FirebaseAuth của Firebase.
* Nhập số điện thoại, nhấn vào nút “SEND CODE” để nhận code Verify.
* Nếu người dùng đang sử dụng chính số điện thoại trong thiết bị thì app sẽ tự nhận tin nhắn và vào luôn trang chủ. Nếu không thì nhập code từ tin nhắn nhận được.
* Dữ liệu gồm số điện thoại và verify code sẽ được lưu vào database vào bảng “phoneAuth”. Lần đầu sử dụng app, yêu cầu người dùng cấp phép sử dụng GPS
* Hàm kiểm tra đăng nhập bằng cách check verify code đã gửi đến số điện thoại

**private void** signInWithPhoneAuthCredential(PhoneAuthCredential credential) {  
 **firebaseAuth**.signInWithCredential(credential)  
 .addOnCompleteListener(**this**, **new** OnCompleteListener<AuthResult>() {  
 @Override  
 **public void** onComplete(@NonNull Task<AuthResult> task) {  
 **if** (task.isSuccessful()) {  
 Log.*d*(***TAG***, **"code verified signIn successful"**);  
 **firebaseUser** = task.getResult().getUser();  
 showSingInButtons();  
  
 *//login successfull* Intent intent = **new** Intent(LoginByPhoneActivity.**this**,MainActivity.**class**);  
 startActivity(intent);  
 } **else** {  
 Log.*w*(***TAG***, **"code verification failed"**, task.getException());  
 **if** (task.getException() **instanceof** FirebaseAuthInvalidCredentialsException) {  
 **verifyCodeET**.setError(**"Invalid code."**);  
 }  
 }  
 }  
 });  
}

* Yêu cầu để sử dụng thư viện fire base là

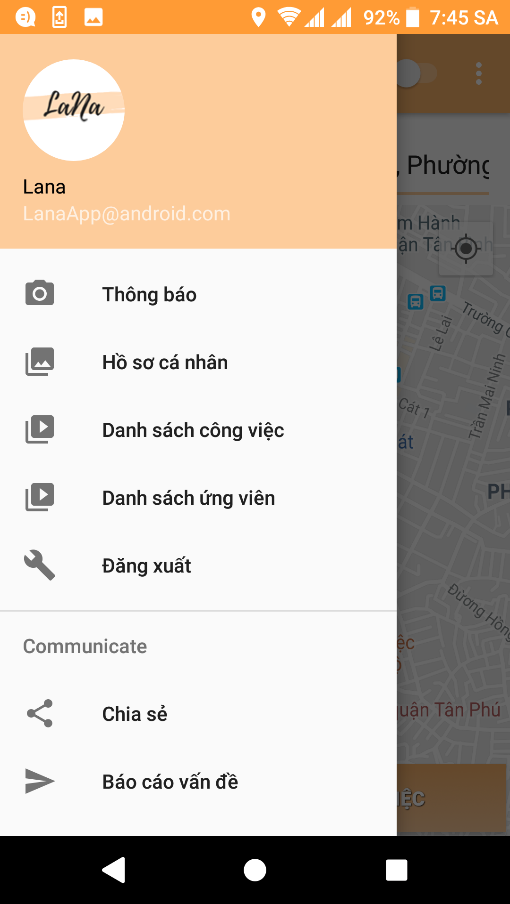
Implement code vào build.gradle(app)

implementation **'com.google.firebase:firebase-core:15.0.0'**implementation **'com.google.firebase:firebase-firestore:16.0.0'**implementation **'com.google.firebase:firebase-auth:15.1.0'**

Implement code vào build.gradle(module)

dependencies {  
 classpath **'com.android.tools.build:gradle:3.1.3'** classpath **'com.google.gms:google-services:3.2.0'** classpath **'io.fabric.tools:gradle:1.25.4'**}

* 1. Giao diện chính



* Màn hình chính hiển thị tọa độ hiện tại của người dùng thông qua API của GoogleMap.
* Có một ‘EditText’ dùng để hiển thị địa chỉ hiện tại thiết bị đang chọn (giữa màn hình). Dữ liệu trong EditText được cập nhật mỗi 5 giây một lần bằng phương thức đa luồng (AsyncTask)
* Trong map hiển thị ra tọa độ các nơi đang tuyển dụng, lấy dữ liệu từ database và vẽ Marker vào GoogleMap trong App.
* Sử dụng Navigation View để hiện các chức năng chính của App
* Trên Action menu có Spinner để chọn hiển thị theo danh sách “Việc tìm người” hay “Người tìm việc”. Khi thay đổi thì các tọa độ (marker) của Googlemap sẽ thay đổi nhờ vào dữ liệu trên database.
* Hàm chuyển dữ liệu tọa độ thành địa chỉ.

@SuppressLint(**"LongLogTag"**)  
**private** String getCompleteAddressString(**double** LATITUDE, **double** LONGITUDE) {  
 String strAdd = **""**;  
 Geocoder geocoder = **new** Geocoder(**this**, Locale.*getDefault*());  
 **try** {  
 List<Address> addresses = geocoder.getFromLocation(LATITUDE, LONGITUDE, 1);  
 **if** (addresses != **null**) {  
 Address returnedAddress = addresses.get(0);  
 StringBuilder strReturnedAddress = **new** StringBuilder(**""**);  
  
 **for** (**int** i = 0; i <= returnedAddress.getMaxAddressLineIndex(); i++) {  
 strReturnedAddress.append(returnedAddress.getAddressLine(i)).append(**"\n"**);  
 }  
 strAdd = strReturnedAddress.toString();  
 Log.*w*(**"My Current loction address"**, strReturnedAddress.toString());  
 } **else** {  
 Log.*w*(**"My Current loction address"**, **"No Address returned!"**);  
 }  
 } **catch** (Exception e) {  
 e.printStackTrace();  
 Log.*w*(**"My Current loction address"**, **"Canont get Address!"**);  
 }  
 **return** strAdd;  
}

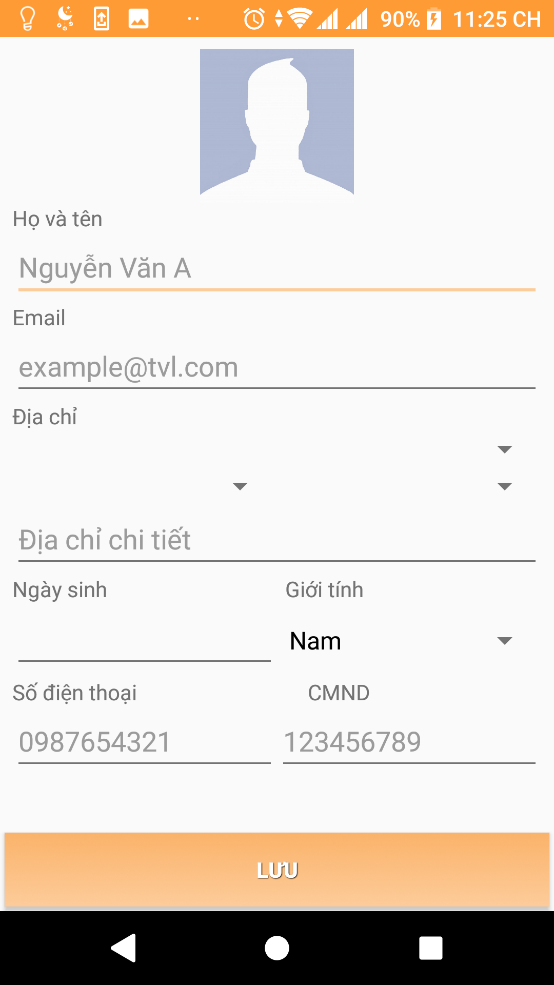
* Để sử dụng GoogleMap API thì ta phải có Secret key do Google cung cấp.

<**string name="google\_maps\_key" templateMergeStrategy="preserve" translatable="false"**>AIzaSyA-NKjK27CqTUUPz2QlCfULwVQhJTY\_M28</**string**>

//Và add vào manifest

<**meta-data  
 android:name="com.google.android.geo.API\_KEY"  
 android:value="@string/google\_maps\_key"** />  
<**meta-data  
 android:name="com.google.android.gms.version"  
 android:value="@integer/google\_play\_services\_version"** />

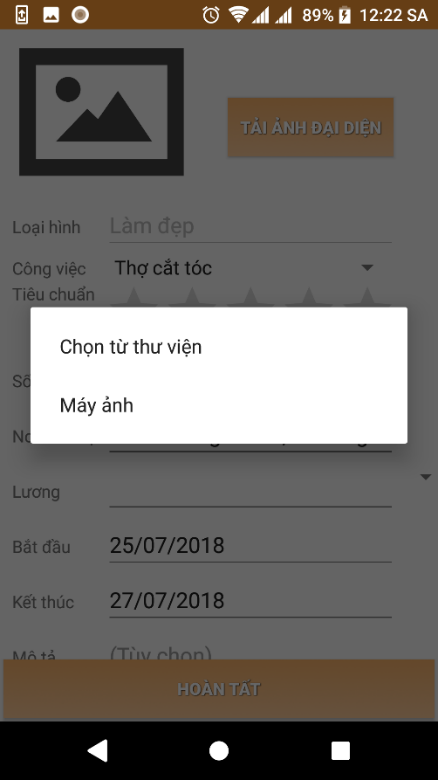
* 1. Chức năng xem “Hồ sơ cá nhân”



* Sử dụng Firebase để load dữ liệu vào app, lấy dữ liệu từ số điện thoại đã đăng nhập. Người dùng có thể cập nhật thông tin cá nhân để tiện cho việc đăng tuyển hoặc ứng tuyển, sau đó update lên firebase thông qua nút “Lưu”.
* Hàm lấy dữ liệu của phiên đăng nhập hiện tại.

FirebaseUser currentUser = **firebaseAuth**.getCurrentUser();  
**try** {  
 **tvDienThoai**.setText(currentUser.getPhoneNumber());  
 **tvTen**.setText(currentUser.getDisplayName());  
 **tvEmail**.setText(currentUser.getEmail());  
}**catch** (Exception ex){  
 Toast.*makeText*(**this**, **"Lỗi"**, Toast.***LENGTH\_SHORT***).show();  
 Log.*d*(**"currentUSER"**,ex.toString());  
}

* 1. Chức năng “Đăng Tuyển”



* Khi người dùng nhấn vào nút “Đăng Tuyển”, ứng dụng sẽ chuyển sang màn hình đăng tuyển. Kèm theo dữ liệu tọa độ và địa chỉ đang chọn.

**btnDangTuyen**.setOnClickListener(**new** View.OnClickListener() {  
 @Override  
 **public void** onClick(View view) {  
 *//Cần phải tự lấy được địa điểm hiện tại (nơi làm việc)* LatLng center = **mGoogleMap**.getCameraPosition().**target**;  
 Intent intent = **new** Intent(getApplicationContext(), CreateHiringActivity.**class**);  
 intent.putExtra(**"latlng"**, center.**latitude** + **","** + center.**longitude**);  
 intent.putExtra(**"address"**,**edtAddressNow**.getText().toString());  
*// intent.putExtra(")* Log.*d*(**TAG**, **"đã gửi latlng và address:"** + center.**latitude** + **","** + center.**longitude** +**"\n"**+**edtAddressNow**.getText().toString());  
 startActivity(intent);  
 }  
 });

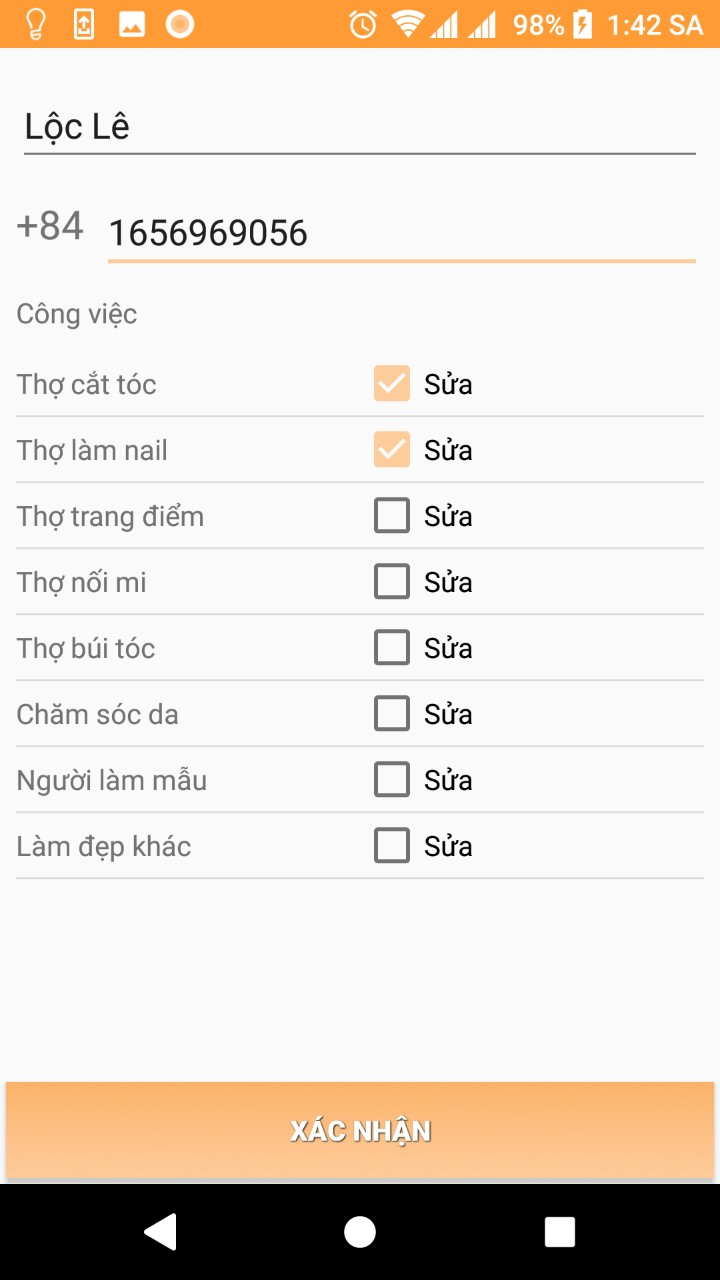
* Trong màn hình đăng tuyển dụng, “Nơi làm việc”, ngày “Bắt đầu” và “Kết thúc” được lấy tự động thông qua màn hình chính và ngày giờ hệ thống.
* Để tải ảnh đại diện, ứng dụng hiển thị context cung cấp 2 lựa chọn là “Chọn từ thư viện” hoặc “Mở camera”.

@Override  
**public void** onCreateContextMenu(ContextMenu menu, View v, ContextMenu.ContextMenuInfo menuInfo) {  
 **super**.onCreateContextMenu(menu, v, menuInfo);  
 **int** itemId=111;  
 menu.add(0,itemId,0,**"Chọn từ thư viện"**);  
 itemId=112;  
 menu.add(0,itemId,1,**"Máy ảnh"**);  
}  
  
**public void** onImageLoadClickEvent(View sender)  
{  
 registerForContextMenu(sender);  
 openContextMenu(sender);  
 unregisterForContextMenu(sender);  
}  
  
@Override  
**public boolean** onContextItemSelected(MenuItem item) {  
 Intent intent;  
 **switch** (item.getItemId()){  
 **case** 111:  
 intent =**new** Intent(Intent.***ACTION\_PICK***);  
 intent.setType(**"image/\*"**);  
 startActivityForResult(intent, **REQUEST\_CODE\_FOLDER**);  
 **break**;  
 **case** 112:  
 intent =**new** Intent(MediaStore.***ACTION\_IMAGE\_CAPTURE***);  
 startActivityForResult(intent, **REQUEST\_CODE\_CAMERA**);  
 **break**;  
 }  
  
 **return super**.onContextItemSelected(item);  
}  
  
@Override  
**protected void** onActivityResult(**int** requestCode, **int** resultCode, Intent data) {  
 **if**(requestCode == **REQUEST\_CODE\_CAMERA** && resultCode == ***RESULT\_OK*** && data !=**null**){  
 Bitmap bitmap = (Bitmap) data.getExtras().get(**"data"**);  
 **imvAvatar**.setImageBitmap(bitmap);  
 }  
 **if**(requestCode == **REQUEST\_CODE\_FOLDER** && resultCode == ***RESULT\_OK*** && data !=**null**){  
 Uri uri = data.getData();  
 **try** {  
 InputStream inputStream = getContentResolver().openInputStream(uri);  
 Bitmap bitmap = BitmapFactory.*decodeStream*(inputStream);  
 **imvAvatar**.setImageBitmap(bitmap);  
 } **catch** (FileNotFoundException e) {  
 e.printStackTrace();  
 }  
 }  
  
 **super**.onActivityResult(requestCode, resultCode, data);  
}

* Khi nhấn nút “Hoàn tất”, tất cả dữ liệu sẽ được ép về kiểu chuỗi, ảnh avatar cũng được dịch sang chuỗi Base64 và lưu như chuỗi thông thường. Lưu vào bảng “job” của database

**btnDangTuyen**.setOnClickListener(**new** View.OnClickListener() {  
 @Override  
 **public void** onClick(View view) {**if**(!KiemTraHopLe()){  
 Toast.*makeText*(CreateHiringActivity.**this**, **"Vui lòng nhập đầy đủ thông tin!"**, Toast.***LENGTH\_SHORT***).show();  
 **return**;  
 }  
 Job cv = **new** Job();  
 cv.setCongViec(**spnCongViec**.getSelectedItem().toString());  
 cv.setSoLuong(**edtSoLuong**.getText().toString());  
 cv.setNoiLamViec(**edtNoiLamViec**.getText().toString());  
 cv.setMucLuong(**edtLuong**.getText().toString());  
 cv.setNgayBatDau(**edtNgayBatDau**.getText().toString());  
 cv.setNgayKetThuc(**edtNgayKetThuc**.getText().toString());  
 cv.setMoTa(**edtMoTa**.getText().toString());  
 cv.setSdtLienHe(**edtSDTLH**.getText().toString());  
 cv.setLatLng(**intent**.getExtras().getString(**"latlng"**));  
 Bitmap avt = ((BitmapDrawable)**imvAvatar**.getDrawable()).getBitmap();  
 cv.setAvatar(Base64.*encodeToString*(getBytesFromBitmap(avt),  
 Base64.***NO\_WRAP***));  
 AddJobToDb(cv);  
 }  
 });

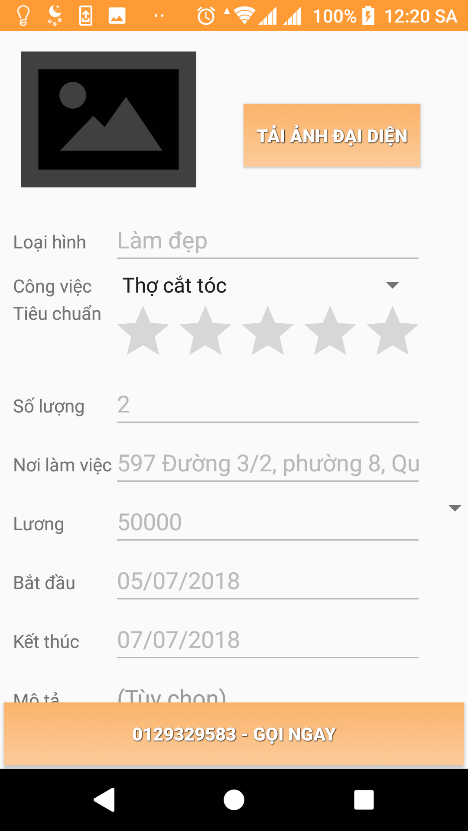
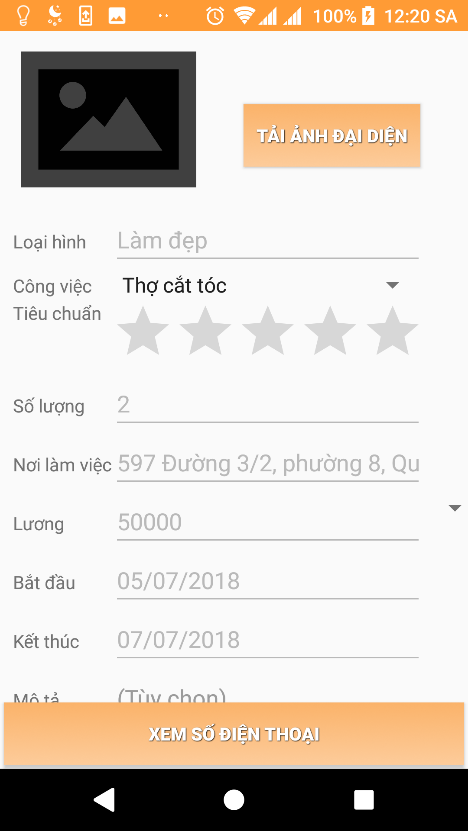
* 1. Chức năng đăng tìm việc

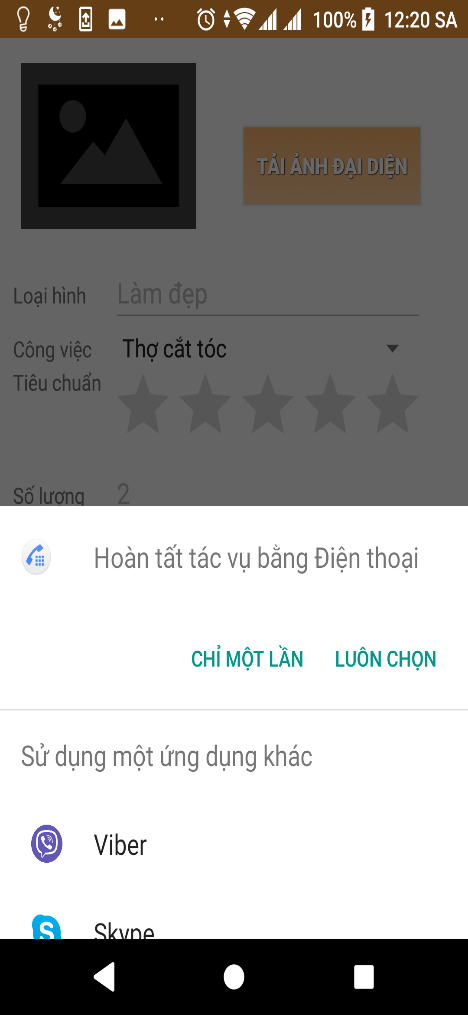


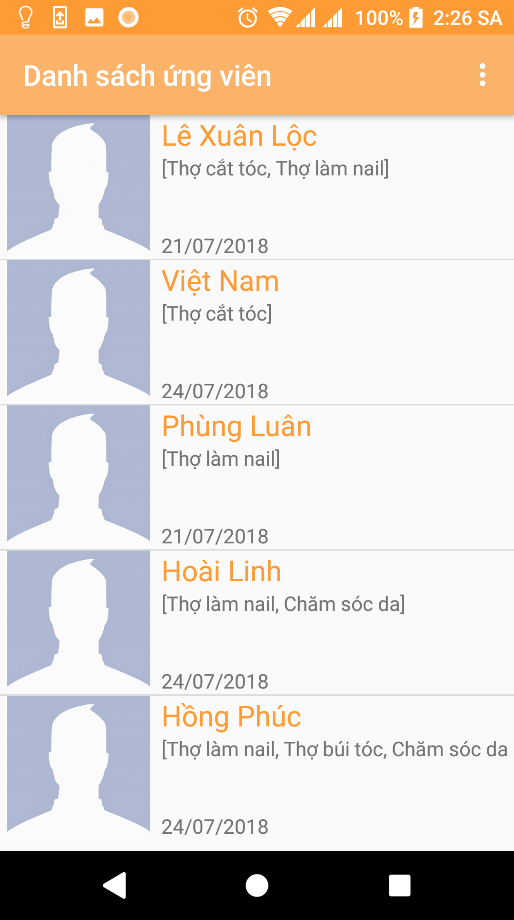
* Tương tự như đăng tìm việc, gửi dữ liệu địa chỉ, tọa độ thông qua intent.
* Chuyển sang màn hình chọn loại công việc. Danh sách công việc được add vào bằng code và custom từng dòng bằng listview.
* Adapter dùng để hiển thị danh sách chọn loại công việc

@Override  
**public** View getView(**int** position, View v, ViewGroup viewGroup) {  
 LayoutInflater inflater = (LayoutInflater) **context**.getSystemService(Context.***LAYOUT\_INFLATER\_SERVICE***);  
 v = inflater.inflate(**myLayout**, **null**);  
  
 TextView title = (TextView) v.findViewById(R.id.***tvTenCongViec***);  
 CheckBox cbCongViec = (CheckBox) v.findViewById(R.id.***cbChonCongViec***);  
 title.setText(**arrayListJob**.get(position).toString());  
 cbCongViec.setTag(title.getText());  
   
 **return** v;  
  
}

* 1. Chức năng xem danh sách công việc



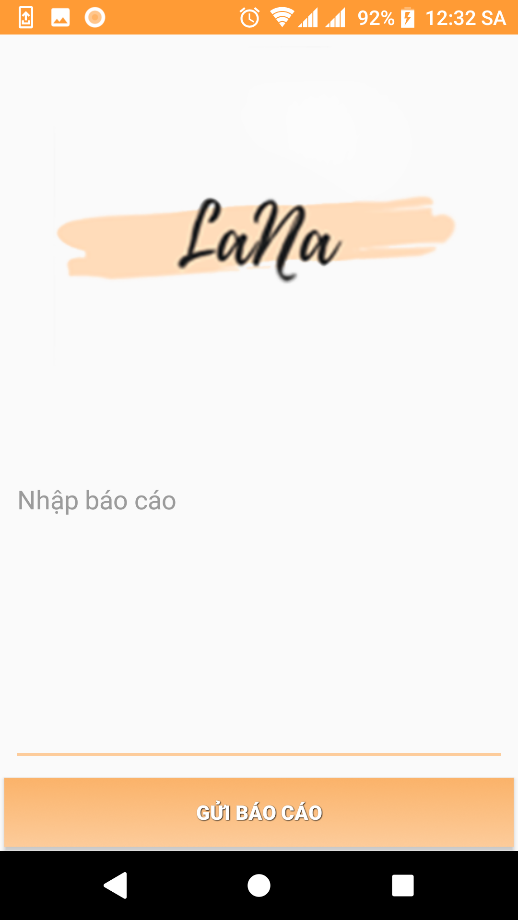
* Danh sách công việc được load từ Firebase. Sau đó được đổ vào listview thông qua adapter hỗ trợ hiển thị.
* Khi người dùng click vào item nào trong listview thì sẽ lấy dữ liệu chi tiết công việc.
* Dữ liệu chi tiết công việc được load từ các key trong document độc lập của collection “job”. Sau đó được gán vào các view.
* Khi người dùng chọn được công việc phù hợp và muốn ứng tuyển, để liên hệ với nhà tuyển dụng. Nhấn vào nút **“Xem số điện thoại”** để lấy số điện thoại.
* Nhấn vào nút “Gọi ngay” để thực hiện cuộc gọi đến nhà tuyển dụng.
  1. Chức năng xem danh sách ứng viên.



* Tương tự như danh sách công việc, danh sách ứng viên được load từ Firebase. Sau đó được đổ vào listview thông qua adapter hỗ trợ hiển thị.
* Khi người dùng click vào item nào trong listview thì sẽ lấy dữ liệu chi tiết của ứng viên.
* Dữ liệu chi tiết của ứng viên được load từ các key trong document độc lập của collection “employee”. Sau đó được gán vào các view.
* Khi nhà tuyển dụng chọn được ứng viên phù hợp và muốn tuyển, để liên hệ với ứng viên. Nhấn vào nút **“Xem số điện thoại”** để lấy số điện thoại.
* Nhấn vào nút “Gọi ngay” để thực hiện cuộc gọi đến ứng viên.

**protected void** call(String phoneNum) {  
*// "phoneNum = +84908412036";* phoneNum.replace(**"+84"**,**"0"**);  
 Intent callIntent = **new** Intent(Intent.***ACTION\_DIAL***);  
 callIntent.setData(Uri.*parse*(**"tel:"** + phoneNum));*//change the number* startActivity(callIntent);  
 }

* 1. Chức năng báo cáo lỗi, góp ý.



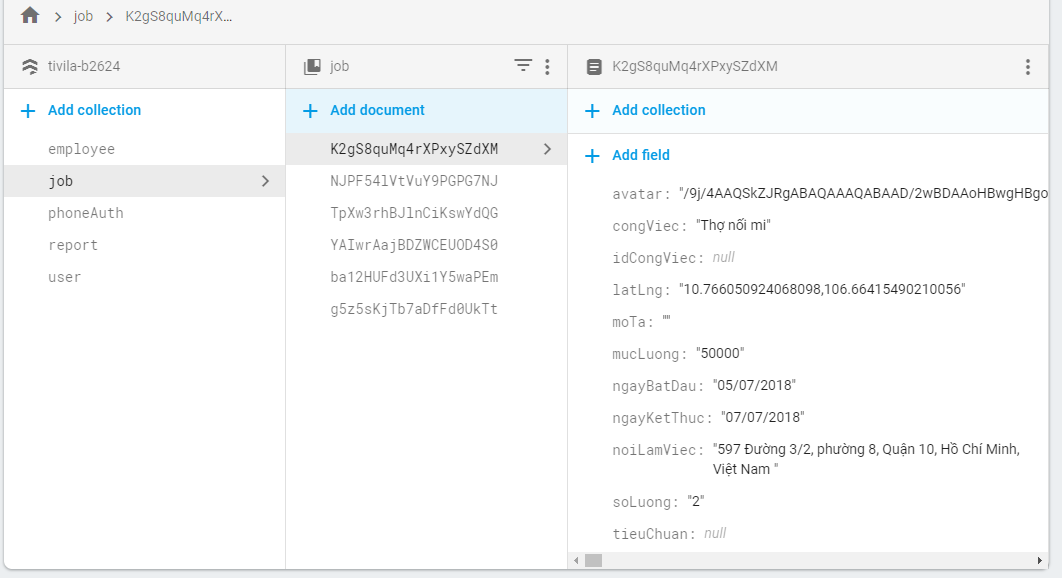
* Người dùng có thể góp ý những điều ứng dụng chưa hoàn thiện thông qua chức năng báo cáo lỗi.
* Thông tin người dùng báo cáo, góp ý sẽ được lưu vào database vào bảng “Report” gồm chi tiết báo cáo, ngày giờ, số điện thoại người dùng. Đồng thời cũng thông báo qua mail admin
* Về vấn đề lỗi hệ thống, ứng dụng sử dụng Thư viện Crashlytics của Firebase để báo cáo lỗi cho admin. Tức là bất cứ người dùng nào đang sử dụng app mà gặp lỗi thì ứng dụng sẽ tự động gửi dữ liệu lỗi về mail cho admin.
* Cài đặt Crashlytics để báo lỗi khi treo app

Fabric.*with*(**this**, **new** Crashlytics());  
*// addContentView(crashButton,  
// new ViewGroup.LayoutParams(ViewGroup.LayoutParams.MATCH\_PARENT,  
// ViewGroup.LayoutParams.WRAP\_CONTENT));* **final** Fabric fabric = **new** Fabric.Builder(**this**)  
 .kits(**new** Crashlytics())  
 .debuggable(**true**)

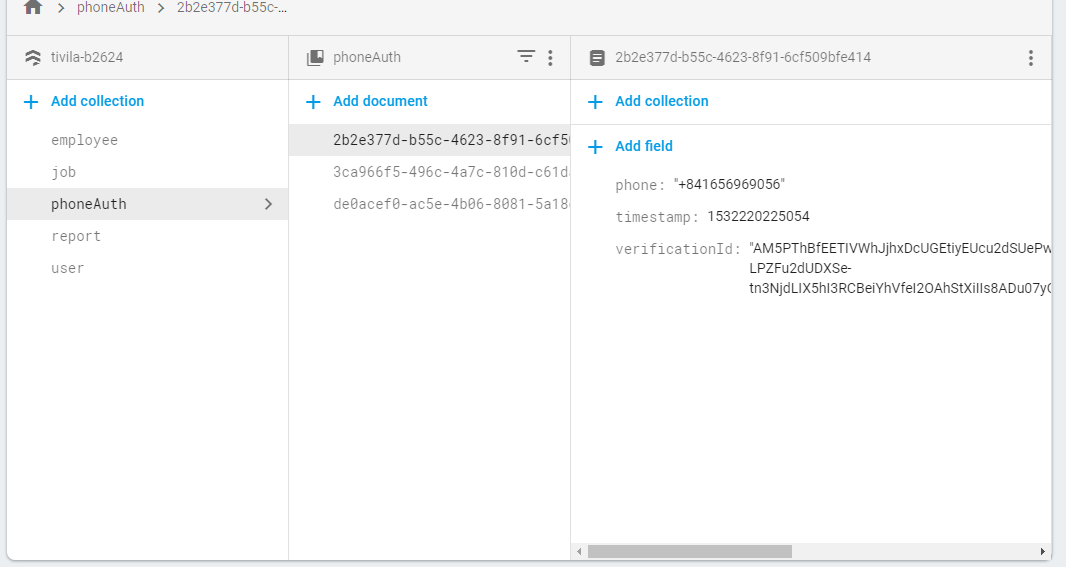
*// Enables Crashlytics debugger* .build();  
 Fabric.*with*(fabric);  
  
  
 }

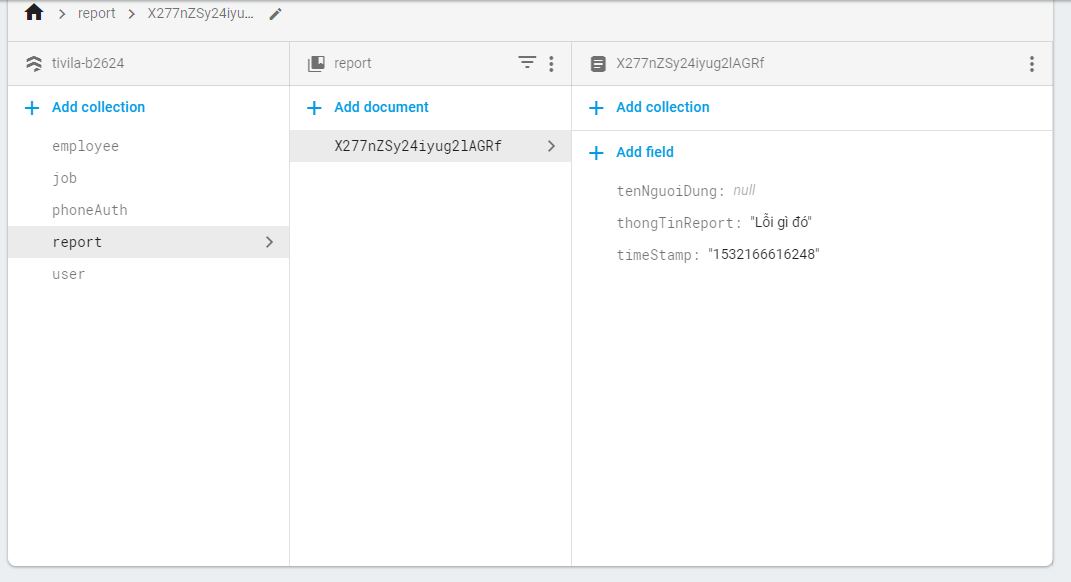
1. **Quản lý trên server**

Quản trị viên có thể kiểm tra thông tin việc làm, ứng viên, các lượt đăng nhập, góp ý của người dùng bằng panel quản lý của firebase.

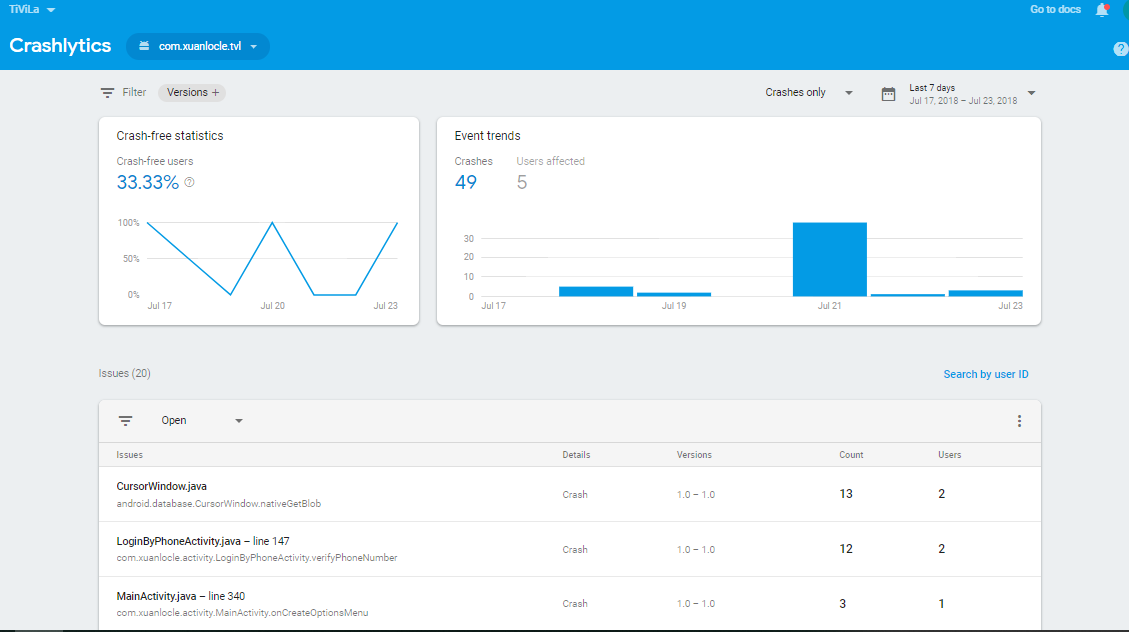








Sử dụng công cụ **Crashlytics**, báo lỗi về cho admin khi người dùng bị lỗi trong quá trình sử dụng. Khi người dùng bị một lỗi bất kì làm treo ứng dụng. Crashlytics sẽ thông qua firebase và gửi mail về cho admin.



1. **TỔNG KẾT.**

* Ứng dụng có những tính năng phù hợp
* Tìm hiều được nền tảng cơ bản của hệ điều hành trên thiết bị di động Android.
* Nắm bắt được cách tạo giao diện từ những công cụ Android cung cấp và cách tùy chỉnh các công cụ này: Linear Layout, Relative Layout, … các control :Listview, Button, Checkbox, ….
* Làm quen với các khái niệm: Activity, Intent, MapView,
* Geocoder, Services, … và sử dụng phần cứng cơ bản của Android đó là Camera để quay phim, chụp ảnh