Proyecto integrado

Babyguard

2016/2017

Yeray Ruiz Juárez

Introducción

Babyguard es una aplicación destinada a todas las guarderías y padres de la misma que les gusta tener una buena relación padre-profesor, con el fin de que el padre sepa en todo momento cómo se encuentra su hijo.
Podrás ser informado de cómo ha dormido, su actitud,... entre otras cosas, e incluso mantener una conversación con el profesor y ser informado de futuros eventos.

Babyguard dispone de una interfaz móvil para los padres y los profesores y una interfaz web para los gerentes de la guardería, desde la cual podrá gestionar la licencia, las clases y el tipo de cuenta que dispone, así como la posibilidad de actualizar su licencia.

Hay pocas aplicaciones que aportan lo mismo, pero en lo que se diferencia BabyGuard es en tener un diseño moderno, rápido, con un manejo intuitivo para usuarios sin conocimientos informáticos y mejores precios que la competencia.

La idea surge al ver que mi anterior idea para el proyecto me sobrepasaba, decidí cambiarlo por éste cuando Lourdes comentó que la aplicación que utilizan en su guardería era muy lenta.

Objetivos del proyecto

El objetivo final del proyecto es mantener en contacto entre dos personas. Para ello hay que cumplir otros objetivos:

- Crear un chat entre el padre y el profesor, siendo capaces de intercambiar mensajes en tiempo real y ser notificado en caso de no estar conectado
- Tener un calendario con los eventos de la clase y que el profesor pueda crear, editar y eliminar eventos.
- Tener una lista con eventos relacionados con el seguimiento del niño de ese día y que el profesor pueda crearlos, editarlos y eliminarlos.
- Sistema de notificaciones al recibir un mensaje, evento de seguimiento o de calendario, y si un evento ocurre mañana u hoy.
- Crear una web de presentación de la aplicación, con pasarelas de pago para automatizar la creación de cuentas.
- Crear un sistema de gestión de usuarios, tanto para el administrador de la aplicación, como para los administradores de las guarderías que permitan crear cuentas en función del plan contratado.
- Personalización de los campos del usuario desde la aplicación, cuya sesión esté iniciada.

Planificación del proyecto

Tareas

- App android
  - Interfaz de profesor
  - Interfaz del padre
  - Firebase
- Web de presentación
  - Front end

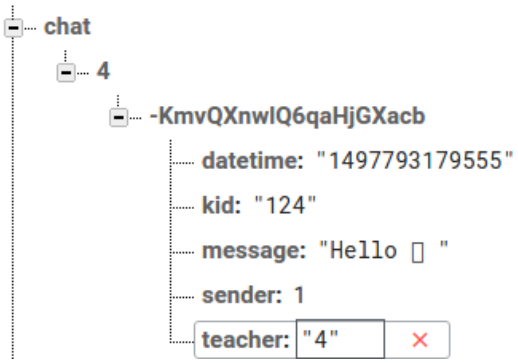
Actividad	Inicio	Fin	Horas
Android			164
Interfaz del padre	8/12/16	28/12/16	80
Firebase: Documentación	15/04/17	15/04/17	4
Firebase: Implementación	17/04/17	20/04/17	12
Cambios menores en la interfaz	22/04/17	24/04/17	6
Interfaz de “Acerca de” de la guardería	24/04/17	24/04/17	2
Chat: Interfaz y funcionamiento	14/05/17	15/05/17	8
Base de datos SQLite: mensajes	14/05/17	14/05/17	3
Interfaz del profesor	23/05/17	27/05/17	16
Arreglado un fallo de los chats	01/06/17	01/06/17	4
Manejar eventos y seguimiento	04/06/17	11/06/17	10
Preferencias del usuario	12/06/17	13/06/17	5
Firebase storage: subida de imágenes de perfil	13/06/17	14/06/17	3
Notificaciones	16/06/17	17/06/17	7
Otras cosas: Internacionalización, refactorización (leve), correción de fallos	17/06/17	18/06/17	4
Web			
Interfaz	13/03/17	26/03/17	20

Análisis y diseño del sistema

Modelado relacional o bien orientado a objetos de la base de datos

Utiliza Firebase, que es una base de datos nosql.
Además, en Android, utiliza una base de datos creada con el formato “Babyguard_idusuario.db” según el usuario que está logueado, la cual sólo contiene una tabla mensajes, con el formato de la tabla ‘Chat’ de Firebase.

Chat



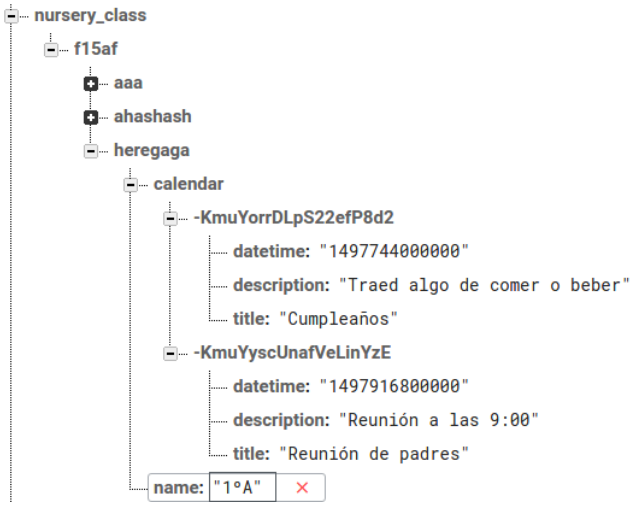
Contiene el chat asociado a cada profesor y a cada niño.
Cada mensaje contiene la fecha en la que se recive (UnixTime), el mensaje, el id del profesor, del niño y quién lo envía: 0 es el padre y 1 es el profesor. En cuanto el usuario lo recibe lo borra de la tabla y se almacena en la base de datos del teléfono

Kid

Contiene el perfil de un niño: Su id, guardería y clase a la que va, padre asociado, imagen del niño, imagen del padre, nombre, información y el token de Firbase Cloud Messaging.

Nursery

Contiene el perfil de la guardería: Nombre, dirección, correo electrónico, web y teléfono(s)



Nursery_class

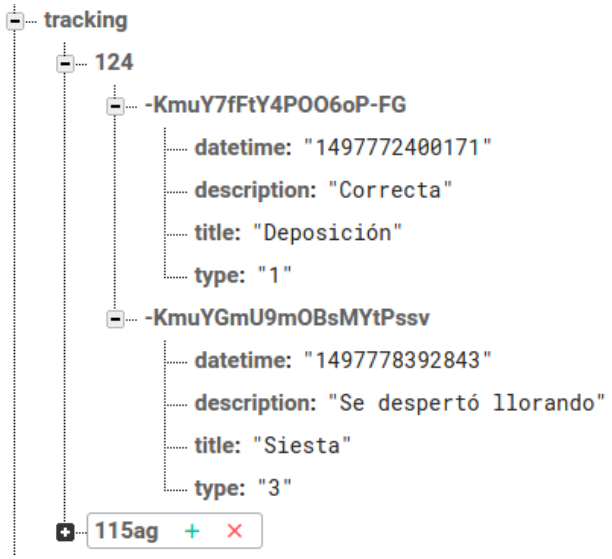
Contiene los datos de las clases de una guardería: Nombre identificativo y el calendario asociado a una clase.

El calendario se organiza por eventos, los cuales tienen la fecha, un título y una descripción.

Tracking

Funciona de forma parecida al calendario.

Contiene un título, descripción, fecha, hora y el tipo de dato que puede ser: Deposición, comida, sueño u otro.



User

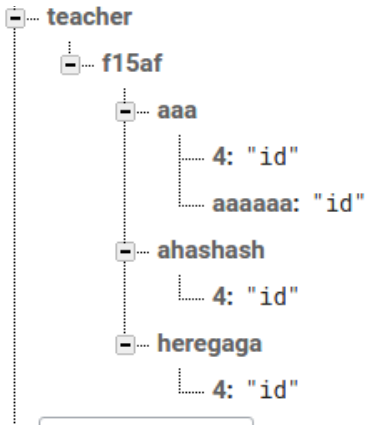
Hay cuatro tipos:

- Administrador (1): Cuenta que permite la creación de cualquier cuenta. Usa una aplicación web.
- Gestor (2): Cuenta asociada al gestor de la guardería. Permite la gestión de clases, cuentas y niños. Usa una aplicación web.
- Profesor (3): Permite añadir, modificar y eliminar elementos relacionados con la cuenta del padre (calendario, seguimiento). Utiliza una aplicación Android.
- Padre (4): Permite modificar datos relacionados con sus niños. Utiliza una aplicación Android.

Las cuentas tipo profesor y padre tienen un campo para guardar la contraseña que encripta la base de datos sqlite, el token de firebase para recibir notificaciones, la imagen de perfil, el correo, el nombre, el teléfono y el tipo de usuario. Además, los profesores guardan cuáles son sus clases asociadas.

Teacher

Es un nodo auxiliar, el cual tiene cuáles son los profesores asociados a cada clase de una guardería



Consultas a la base de datos

Android

Inicio de sesión y recuperación de contraseña: No es una consulta como tal, se encarga Firebase de ello.

Insertar, editar y eliminar eventos al calendario (profesor) y lectura de los mismos (profesor y padre).

Insertar, editar y eliminar eventos al seguimiento del niño(profesor) y lectura de los mismos (profesor y padre).

Enviar y recibir mensajes de chat, y eliminarlos una vez son leídos (profesor y padre).

Actualización y lectura de datos del perfil.

SQLite: Añadir y leer mensajes

Para añadir listeners en firebase utilizo el mismo método para todos: Añado el listener a la referencia de la base de datos y lo añado a una lista para que al cerrar sesión se liberen esos listeners.

```
private void addListener(Query reference, ValueEventListener
listener) {
  reference.addValueEventListener(listener);
  ArrayList<ValueEventListener> aux =
activeValueListeners.get(reference);
  if (aux == null)
    aux = new ArrayList<>();
  aux.add(listener);
  activeValueListeners.put(reference, aux);
}
```

Para eliminar datos consigo la referencia que quiero eliminar y llamo a removeValue()

```
public DiaryCalendarEvent removeEvent(String nurseryId,
String classId, String eventId) {

  DiaryCalendarEvent result =
Repository.getInstance().removeEvent(nurseryId, classId, eventId);
database.getReference().child(NURSERY_CLASS_REFERENCE).child(nursery
Id).child(classId).child("calendar").child(eventId).removeValue();

  return result;

}
```

Para actualizar datos creo un HashMap con los campos que quiera subir, declaro la referencia y uso setValue(hashmap)

```
public void updateEvent(String nurseryId, String classId,
DiaryCalendarEvent event) {
  HashMap<String, String> eventPush = new HashMap<>();
  String datetime = Utils.parseDateToUnix(event.getDate());
  eventPush.put("datetime", datetime);
```

```
  eventPush.put("description", event.getDescription());
  eventPush.put("title", event.getTitle());

database.getReference().child(NURSERY_CLASS_REFERENCE).child(nursery
Id).child(classId).child("calendar").child(event.getId()).setValue(e
ventPush);
}
```

Para añadir eventos es similar a actualizar, sólo que antes hay que hacer push() para que firebase genere un nodo con un key aleatorio. Entonces obtengo ese key y ya hago setValue()

```
public DiaryCalendarEvent addEvent(String nurseryId, String classId,
DiaryCalendarEvent event) {
  HashMap<String, String> eventPush = new HashMap<>();
  String datetime = Utils.parseDateToUnix(event.getDate());
  eventPush.put("datetime", datetime);
  eventPush.put("description", event.getDescription());
  eventPush.put("title", event.getTitle());
  DatabaseReference ref =
database.getReference().child(NURSERY_CLASS_REFERENCE).child(nursery
Id).child(classId).child("calendar").push();
  event.setId(ref.getKey());
  ref.setValue(eventPush);
  return event;
}
```

Especificaciones del sistema

Tecnologías usadas

Firebase

Firebase es una plataforma desarrollada por Google con una multitud de herramientas, la mayoría gratuitas o con un limitado pero amplio plan gratuito. Entre las herramientas que dispone, las que he utilizado son:

- Su base de datos en tiempo real, la cual sustituye el back-end de una API web, que se implementa de forma más sencilla y más rápida.
- Cloud messaging, que permite enviar notificaciones en tiempo real.
- Storage, es una “nube”, donde se guardan las imágenes de perfil de los usuarios.
- Su sistema de autenticación, el cual es muy fácil de implementar y hasta trae métodos para cambiar contraseña y recuperarla.

SQLCipher

SQLCipher es una librería que permite cifrar con una contraseña la base de datos en Android, evitando así que, aún siendo usuario root, se pueda acceder al contenido de la base de datos.

Materialize

Materialize es un framework css y js que añade un estilo Material design a una web.

Además he usado otras librerías que aportaban nuevos diseños de interfaces.

Instalación y configuración de la aplicación

Para la instalación de la aplicación basta con ir al play store y descargarla:

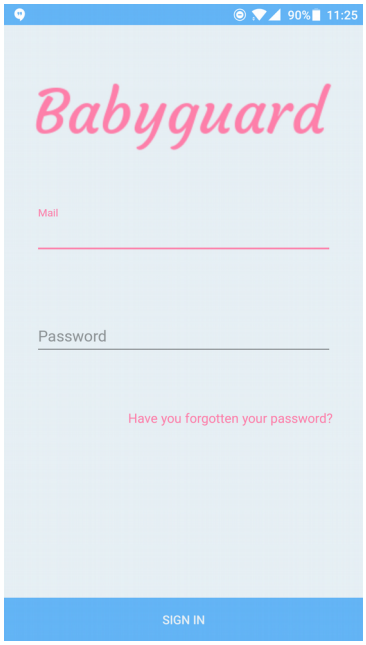
Para la web hay que tener un dominio (o un servidor local con acceso a internet y PHP), y subir los archivos.

Especificaciones hardware del sistema

No hay especificaciones para la app de Android, cualquier móvil que se pueda permitir Android 6.0 tiene el hardware necesario para utilizar la app.

Especificaciones del software

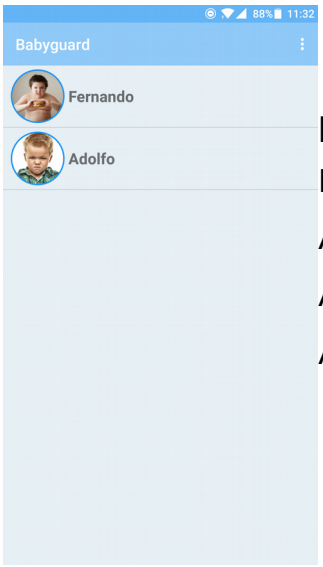
Interfaz y navegación de la aplicación



Login

Al pulsar “¿Has olvidado la contraseña?” redirige a una web para recuperarla.

Al pulsar “INICIA SESIÓN” entra en la siguiente Activity.

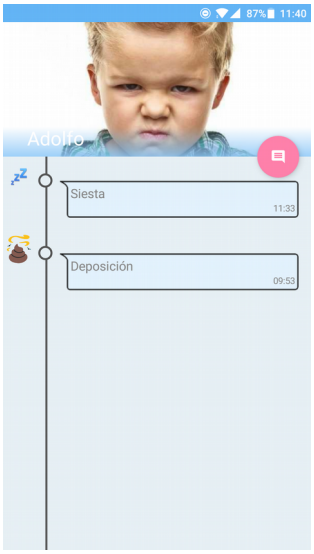


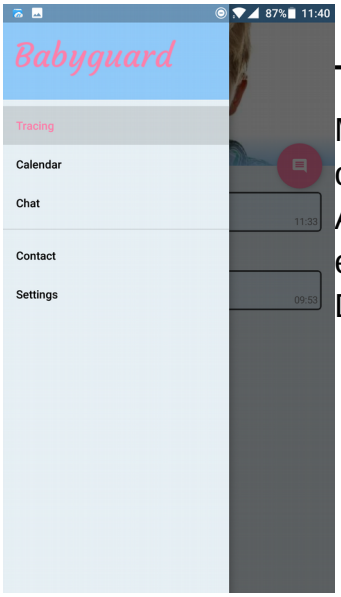
Lista de niños

Muestra los niños que tiene.

Al pulsar la foto se amplia.

Al pulsar en él entra en la siguiente Activity



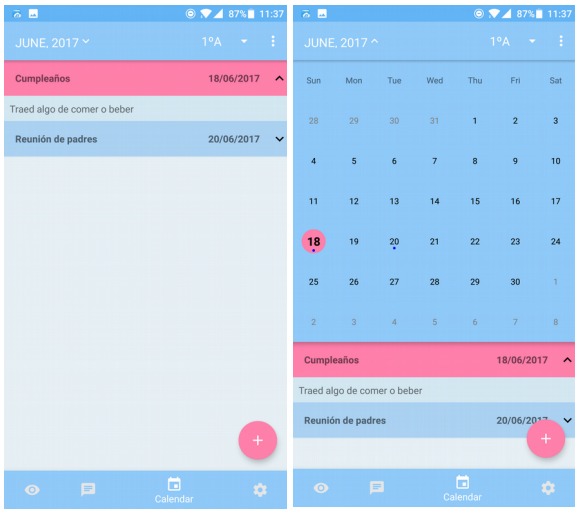


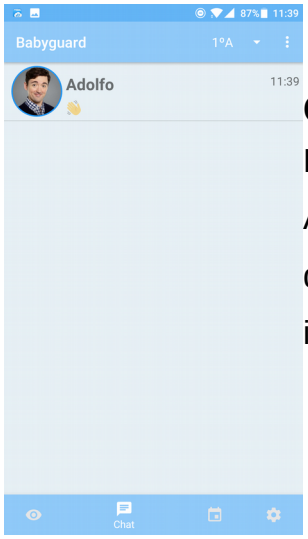
Tracking

Muestra el seguimiento del niño, ordenado por la hora.
Al pulsar el botón de chat se abre el fragment de chat.
Dispone de un Navigation Drawer.

Calendario

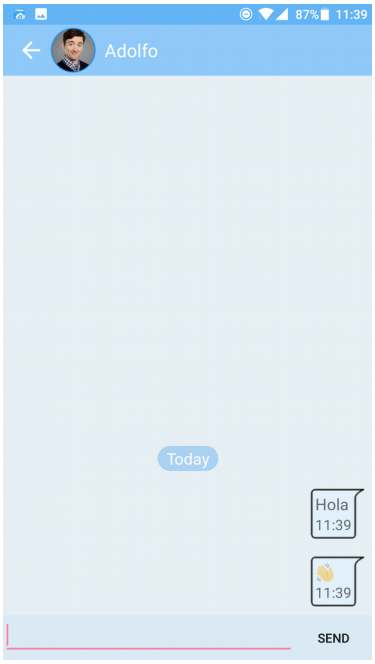
El calendario muestra los eventos de la clase del niño.
Dispone de animaciones para ocultarse y expandirse.
Remarca el día actual.
Al pulsar un día con un evento, la lista de abajo hace scroll hasta donde se encuentra y lo remarca en azul más oscuro.

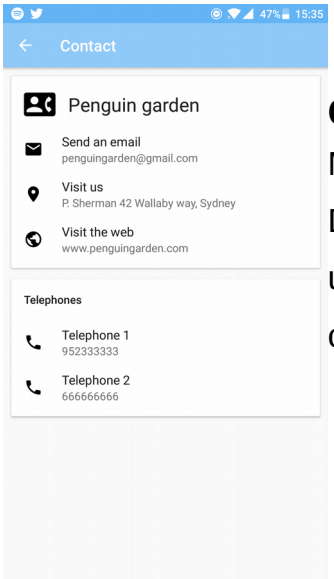




Chat

Permite el chat con el profesor.
Al recibir los mensajes se borran del servidor, y se almacenan internamente.



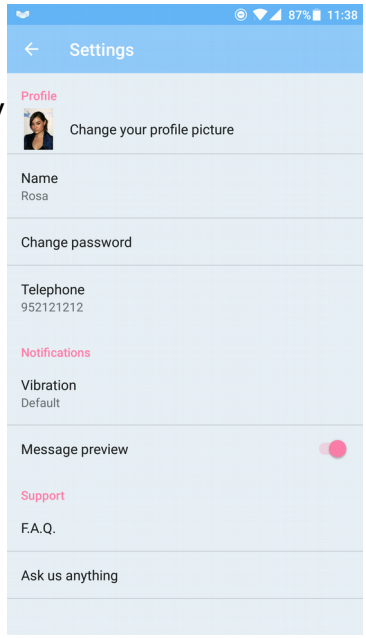


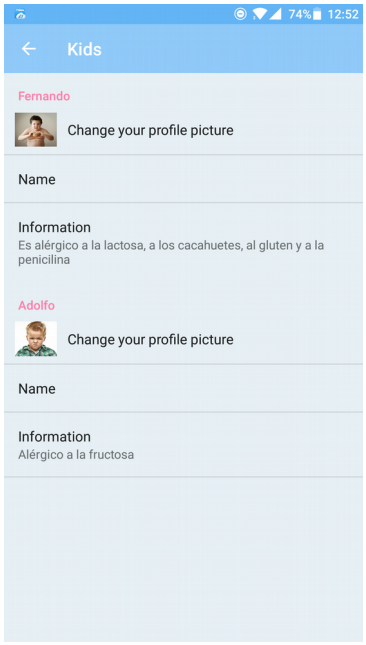
Contacto

Muestra los datos de contacto de la guardería.
Dependiendo del tipo de dato, al pulsar sobre ella realiza una acción u otra: Ver la guardería en el mapa, enviar un correo…

Ajustes

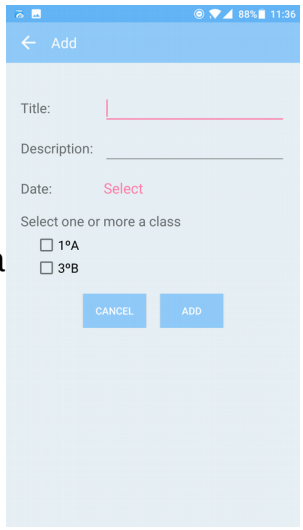
Muestra el aspecto configurable de la aplicación, links de interés y opciones para cambiar datos del perfil del usuario, y de los niños si fuese un padre.

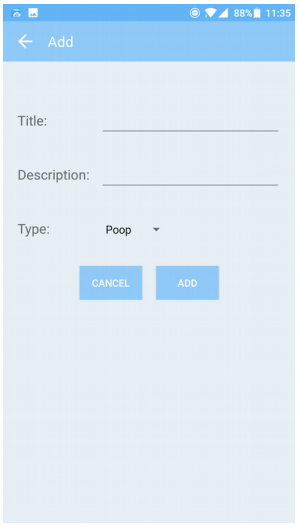




La interfaz del profesor se diferencia en:

- El padre tiene un drawer layout y el profesor un bottom navigation bar.
- El profesor tiene un spinner para cambiar entre clases.
- El profesor dispone de FloatingActionButtons para añadir eventos y elementos de seguimiento. Por ello, tiene además ventanas adicionales para añadir y editarlos.





Código fuente relevante

Enviar notificaciones

Para enviar notificaciones a otros usuarios uso una clase llamada PushNotifications que contiene los campos a enviar, y llamo a pushNotification() pasándole el token del dispositivo al que va dirigida.

En él, convierto el objeto a json con la estructura

{

data{

type=”tipo de notificación”

calendar{//Dependiendo del tipo de notificación será null o no

}

tracking{//Dependiendo del tipo de notificación será null o no

}

message{//Dependiendo del tipo de notificación será null o no

}

fromUser = “Id de usuario”

toUser = “Id de usuario”

},

to=deviceId

}

Después hago una petición post al servidor de Firebase, pasándol el el API Key de mi proyecto Firebase en el header y el json resultante.

```
public void pushNotification(String deviceId) {
  try {
    JSONObject root = new JSONObject();
    try {
      JSONObject j = toJSON();
      root.put("data", j);
      root.put("to", deviceId);
    } catch (JSONException e) {
      e.printStackTrace();
    }
```

```
    StringEntity entity = new StringEntity(root.toString());
    RestClient.post(FCM_URL, FCM_APIKEY, entity, new
AsyncHttpResponseHandler() {
      @Override
      public void onSuccess(int statusCode, Header[] headers, byte[]
responseBody) {
      }
      @Override
      public void onFailure(int statusCode, Header[] headers, byte[]
responseBody, Throwable error) {
      }
    });
  } catch (UnsupportedEncodingException e) {
    e.printStackTrace();
  }
}
private JSONObject toJSON() {
  JSONObject jsonArray = null;
  Gson gson = new Gson();
  String jsonString = gson.toJson(this);
  try {
    jsonArray = new JSONObject(jsonString);
    //jsonObject = new JSONObject(jsonString);
  } catch (JSONException e) {
    // TODO Auto-generated catch block
    e.printStackTrace();
  }
  return jsonArray;
}
```

SQLite

SQLite funciona con la librería SQLCipher, la cuál proporciona una base de datos protegida por contraseña. La base de datos es creada en función del usuario que está logueado, de forma que al cerrar sesión e iniciar sesión otro usuario no se perderían los datos del otro, pero no usarían la misma base de datos.
Cuando el usuario se loguea, la inicializo con el constructor pasándole el id. Después, desbloqueo la base de datos pasándole la contraseña

```
public DatabaseHelper(Context context, String userId) {
  super(context, DATABASE_NAME + "_" + userId + DATABASE_EXTENSION, null,
DATABASE_VERSION);
  SQLiteDatabase.loadLibs(context);
  ((Babyguard_Application)
context.getApplicationContext()).setDatabaseLoaded(false);
  this.context = context;
}
static public synchronized DatabaseHelper getInstance(String userId) {
  if (instance == null) {
    instance = new DatabaseHelper(Babyguard_Application.getContext(),
userId);
  }
  return instance;
}
static public synchronized DatabaseHelper getInstance() throws Exception {
  if (instance == null) {
    throw new Exception("DB not initialized");
  }
  return instance;
}
```

CustomToolbar

Se trata de una toolbar personalizada que “colorea” el texto y los iconos que aparecen en la toolbar, de forma que se puede cambiar el color cambiando simplemente una variable.
Para ello, se recorre las vistas que contiene la toolbar, y dependiendo de qué tipo sea la vista hace una cosa u otra.
Utiliza los siguientes métodos:

```
public static void colorizeToolbar(Toolbar toolbarView, int
toolbarIconsColor) {
  final PorterDuffColorFilter colorFilter
      = new PorterDuffColorFilter(toolbarIconsColor,
PorterDuff.Mode.SRC_IN);
  for (int i = 0; i < toolbarView.getChildCount(); i++) {
    final View v = toolbarView.getChildAt(i);
    if (v != null)
```

```
      doColorizing(v, colorFilter, toolbarIconsColor);
  }
  toolbarView.setTitleTextColor(toolbarIconsColor);
  toolbarView.setSubtitleTextColor(toolbarIconsColor);
}
public static void doColorizing(View v, final ColorFilter colorFilter, int
toolbarIconsColor) {
  try {
    if (v instanceof ImageButton) {
      ((ImageButton) v).getDrawable().setAlpha(255);
      ((ImageButton) v).getDrawable().setColorFilter(colorFilter);
    }
    if (v instanceof ImageView) {
      ((ImageView) v).getDrawable().setAlpha(255);
      ((ImageView) v).getDrawable().setColorFilter(colorFilter);
    }
    if (v instanceof AutoCompleteTextView) {
      ((AutoCompleteTextView) v).setTextColor(toolbarIconsColor);
    }
    if (v instanceof TextView) {
      ((TextView) v).setTextColor(toolbarIconsColor);
    }
    if (v instanceof EditText) {
      ((EditText) v).setTextColor(toolbarIconsColor);
    }
    if (v instanceof ViewGroup) {
      for (int lli = 0; lli < ((ViewGroup) v).getChildCount(); lli++) {
        doColorizing(((ViewGroup) v).getChildAt(lli), colorFilter,
toolbarIconsColor);
      }
    }
    if (v instanceof ActionMenuView) {
      for (int j = 0; j < ((ActionMenuView) v).getChildCount(); j++) {
        final View innerView = ((ActionMenuView) v).getChildAt(j);
        if (innerView instanceof ActionMenuItemView) {
          int drawablesCount = ((ActionMenuItemView)
innerView).getCompoundDrawables().length;
          for (int k = 0; k < drawablesCount; k++) {
            if (((ActionMenuItemView) innerView).getCompoundDrawables()
[k] != null) {
              final int finalK = k;
              ((ActionMenuItemView) innerView).getCompoundDrawables()
[finalK].setColorFilter(colorFilter);
            }
          }
        }
      }
    }
  } catch (Exception e) {
  }
}
```

Subida de imágenes

Para la subida de imágenes se utiliza Firebase Storage.
Antes de la subida, la imagen se redimensiona y se comprime para que ocupe menos.

```
public static Bitmap shrinkImage(Bitmap bitmap, int maxDimensionSize) {
  int originalWidth = bitmap.getWidth();
  int originalHeigth = bitmap.getHeight();
  Bitmap shrinked;
  if (originalHeigth > maxDimensionSize || originalWidth >
maxDimensionSize) {
    int newWidth, newHeight;
    if (originalHeigth > originalWidth) {
      newHeight = maxDimensionSize;
      newWidth = maxDimensionSize * originalWidth / originalHeigth;
    } else {
      newWidth = maxDimensionSize;
      newHeight = maxDimensionSize * originalHeigth / originalWidth;
    }
    float scaleWidth = ((float) newWidth) / originalWidth;
    float scaleHeight = ((float) newHeight) / originalHeigth;
    Matrix matrix = new Matrix();
    matrix.postScale(scaleWidth, scaleHeight);
    shrinked = Bitmap.createBitmap(bitmap, 0, 0, originalWidth,
originalHeigth, matrix, false);
  } else {
    shrinked = bitmap;
  }
  return shrinked;
}
public static Bitmap uriToBitmap(Uri uri) {
  Bitmap bitmap = null;
  try {
    bitmap =
MediaStore.Images.Media.getBitmap(Babyguard_Application.getContext().getCon
tentResolver(), uri);
  } catch (IOException e) {
    e.printStackTrace();
  }
  return bitmap;
}
public static byte[] bitmapToByteArray(Bitmap bitmap) {
  ByteArrayOutputStream baos = new ByteArrayOutputStream();
  bitmap.compress(Bitmap.CompressFormat.JPEG, 50, baos);
  return baos.toByteArray();
}
public static byte[] prepareImageToUpload(Uri uri, int maxSize) {
  Bitmap bitmap = uriToBitmap(uri);
  byte[] image = null;
  if (bitmap != null) {
    bitmap = shrinkImage(bitmap, maxSize);
    image = bitmapToByteArray(bitmap);
  }
```

```
  return image;
}
```

Para la subida se crea una referencia hacia donde se va a subir y se le pasa el archivo. Tiene la posibilidad de añadir listeners para saber si se ha subido correctamente o no.

```
UploadTask uploadTask = ref.putBytes(imageBytes);
```

Seguridad

Firebase

No necesita una configuración adicional ya que utiliza un certificado SSL con llaves de 2048 bit, con lo que previene que se intercepten los datos (de forma legible).

Las contraseñas se guardan en firebase con una función hash llamada Bcrypt, la cual utiliza una contraseña generada aleatoriamente, la cuál es combinada con la contraseña.

Utiliza tokens, usando el estándar JSON Web Token (JWT), el cuál es un JSON que tiene tres partes:

El Header, que indica el tipo de token y el algoritmo usado, codificado en base64.

El Payload, que contiene información sobre el token, como fecha de expiración, para quién va dirigido, … cualquier cosa que queramos poner, codificado también en base64.

La firma (Signature), que utiliza el método de encriptar indicado en el Header, codificando el Header + . + Payload + clave.

Con esto se consigue que si no se puede ver la firma significa que el token no es suyo y puede haber sido robado.

SQLCipher

Como ya se ha comentado anteriormente, SQLCipher proporciona una base de datos SQLite protegida por contraseña.

Ya que al ofuscar una aplicación se podría seguir encontrando la contraseña usada, decidí crear contraseñas en función de cada usuario, las cuales se crean al registrar a un usuario. Cuando el usuario se loguea se obtiene del servidor la contraseña y se desbloquea la base de datos.

Ofuscación

Ofuscar una aplicación sirve para optimizar y reducir el código renombrando variables, entre otras cosas. Con ello además se consigue que si alguien decompila la aplicación le cueste ver qué hace cada cosa, y ralentice, por ejemplo, la copia del código.

Conclusiones del proyecto

Aún quedan muchas por implementar y mejorar, como el sistema de notificaciones de eventos según la hora, que ahora mismo se ejecuta cada vez que se abre la aplicación, el sistema de badges indicando que hay mensajes nuevos, uso de WebCam a través de internet, entre otra cosas.

Considero que lo peor del proyecto ha sido no haber planificado del todo bien cómo estructurarlo, lo que ha hecho que me retrase al tener que reestructuras la base de datos, la interfaz y cómo funciona internamente.
De hecho, empecé creando una API, pero al ver que en un finde de semana sólo hice el login y los métodos de seguridad decidí cambiar a Firebase, con el que hice casi todo en el mismo tiempo.

Si tuviese que decir algo a los que están pensando aún qué proyecto elegir, les diría que no cogiesen un proyecto que les apasione, y a poder ser que no sea gigante, ya que con el tiempo acabas odiando el proyecto.
Además, es una situación algo irreal ya que al hacer un proyecto tan grande no vas a trabajar sólo, o en caso de que sea así, sería por un salario o como proyecto personal, por lo que no tendrías tiempo límite.

El tema de realizar las prácticas durante la realización del proyecto estaba regular planteada, menos mal que este año se realizaron las interfaces a lo largo del curso.
Pienso esto ya que el hecho de echar 6 horas diarias hace que llegues a tu casa a las 3-4 de la tarde, y si tienes otras actividades obligatorias, como sacar al perro, limpiar, etc. te parte la tarde haciéndola mucho menos productiva.
Por ello, a mitad de las prácticas decidí cambiar el horario, aumentándolo media hora más, con lo que conseguí acabar dos días antes, lo cual se agradeció bastante ya que como casi todo el mundo, se trabaja mejor bajo presión de entrega, y esos dos días completos extra se agradecieron bastante.

Bibliografía

https://stackoverflow.com

https://firebase.google.com/docs/android/setup

https://masonwebdev.wordpress.com/2016/03/22/getting-started-with-security-in-firebase/

https://www.taringa.net/