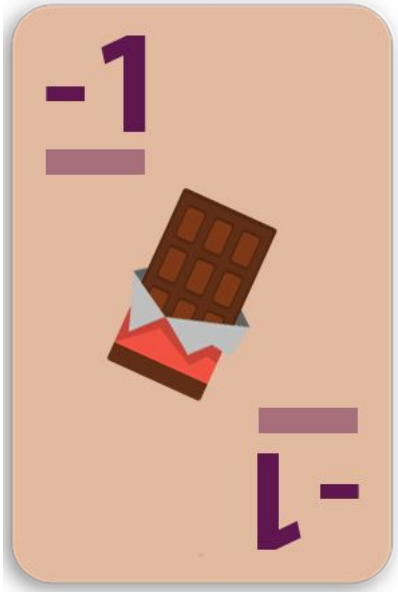




# CABO

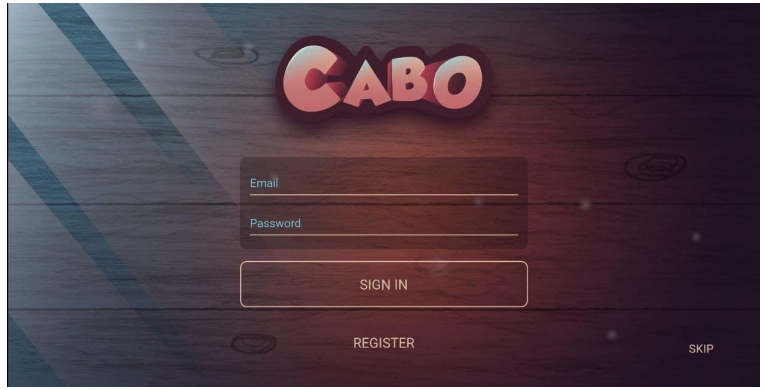
Online-Multiplayer-Kartenspiel für
2-4 Spieler

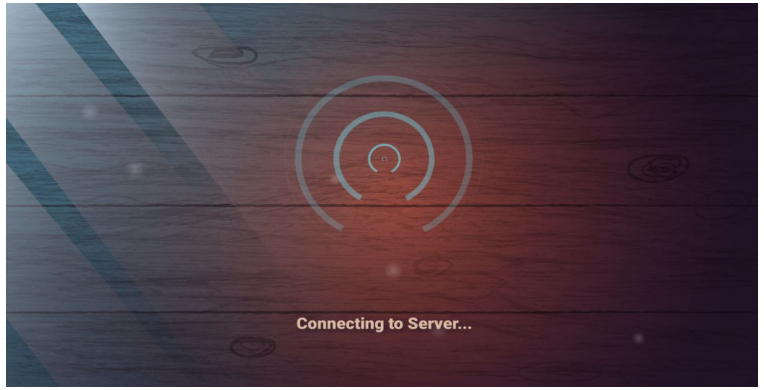
# Aufgabenbereich - Pauline

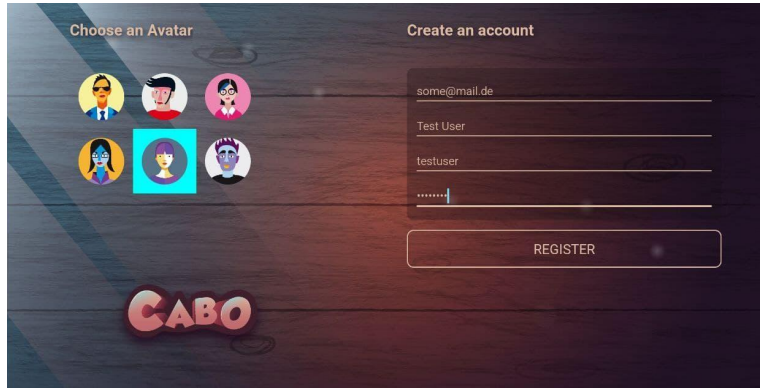
- Front-End Entwicklung
- Layouts
- **Umsetzung des Spielablaufs und anderen Logikbereichen in der GUI**
- Gestalten eines optisch ansprecheneden und interaktiven Nutzererlebnisses

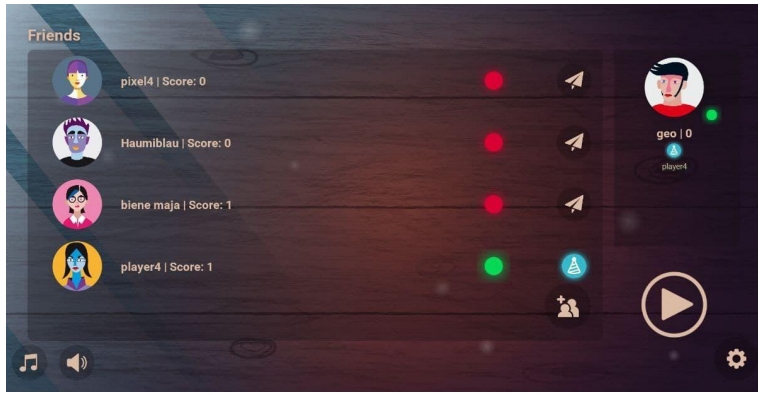
# Layout-Anpassungen

- Umgestaltung und Anpassung an “Cabo-Theme”
- Optimierung für Querformat
- Beachtung von Guidelines für z.B. Button-Design (Platzierung und Optik nach Wichtigkeit, Feedback wenn geklickt
  → Custom Button mit verschiedenen States)
  → konsistent durchgezogen
- Optimierung und custom Text-input-fields



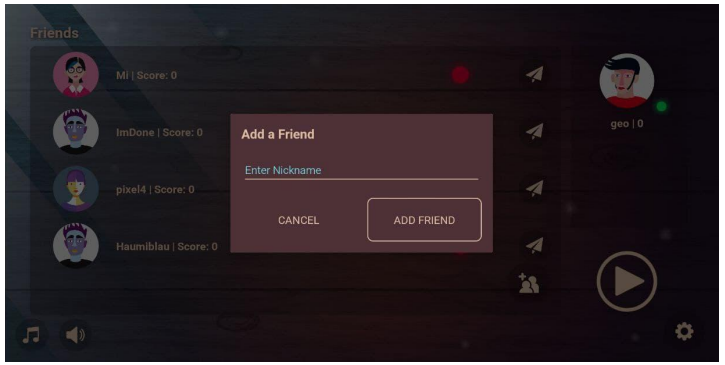



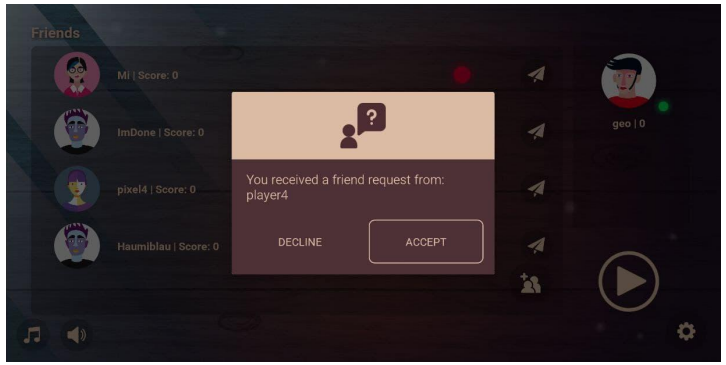




Optimierung der Statusanzeigen der Player und Party-Anzeigen

# Layout-Anpassungen

- Custom Dialog-Gestaltung → Erweiterung von Georgs Dialogen
- Anlegen eigener Klasssen und XMLs
- Wiederverwendbarkeit (insbes. Requestdialog → Freundschaftsanfrage annehmen, Partyeinladung annehmen, Spiel beenden)



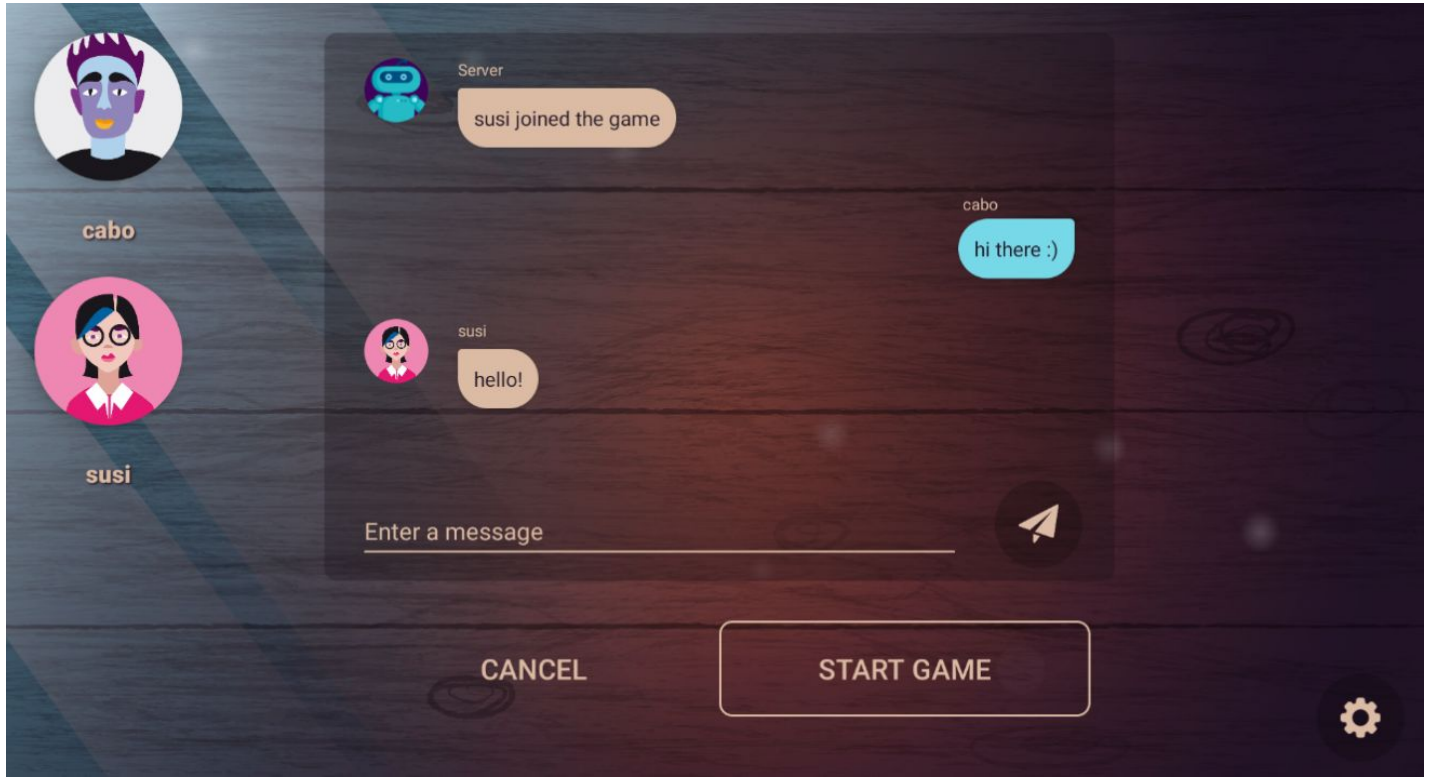




```
public class CustomSearchDialog {
    public void showDialog(Activity activity, Player me, ArrayList<Player> allUsers, Communicator communicator){
        final Dialog dialog = new Dialog(activity);
        dialog.requestWindowFeature(Window.FEATURE_NO_TITLE);
        dialog.setCancelable(true);
        dialog.setContentView(R.layout.search_dialog);

        TextView title = (TextView) dialog.findViewById(R.id.text_dialog_title);
        EditText searchNick = (EditText) dialog.findViewById(R.id.search_friend_input);

        Button dialogButtonSearch = (Button) dialog.findViewById(R.id.btn_dialog_search);
        dialogButtonSearch.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                if (!searchNick.getText().toString().isEmpty()) {
                    if (!(me.getFriendsNicknames().contains(searchNick.getText().toString()))) {
                        if (allUsers.size() > 0) {
                            for (Player user : allUsers) {
                                if (user.getNick().toLowerCase().equals(
                                        searchNick.getText().toString().toLowerCase().trim())) {
                                    Toast.makeText(activity,
                                            text: "Friendrequest sent to " + user.getDbID(), Toast.LENGTH_LONG);
```

```
public class RequestDialog {

    private boolean wantToAccept = false;
    private Button dialogButtonAccept;
    private Button dialogButtonDecline;
    private Dialog dialog = null;
    private TextView text;
    private ImageView image;

    @SuppressLint("SetTextI18n")
    public void showDialog(Activity activity, Player sender){
        dialog = new Dialog(activity);
        dialog.requestWindowFeature(Window.FEATURE_NO_TITLE);
        dialog.setCancelable(true);
        dialog.setContentView(R.layout.request_dialog);

        image = (ImageView) dialog.findViewById(R.id.dialog_image);
        text = (TextView) dialog.findViewById(R.id.text_dialog);
        if(sender!=null){
```

# Chat im Waiting Room (und im Game)

- im Chat-Bubble Design
- Unterscheidung zwischen Server-Message, eigener Message und Message von anderem Spieler → je nachdem anderes Layout von Listview-Adapter inflated



# Chat im Waiting Room (und im Game)

showText wird in **WaitingRoomActivity** aufgerufen, wenn vom Server eine Nachricht kommt, es werden die Message als String, ein boolean, ob es sich um eine Servermessage handelt und der sendende Player mitgeschickt (null falls von Server)

→ je nach sender wird ein anderes ChatMessage-Objekt zur messageList hinzugefügt und der Adapter wird geupdated

```java
/**
 * this method shows the received or typed in messages in the UI
 * the object ChatMessage is created depending on the sender
 *
 * @param message
 */
private void showText(String message, boolean serverMsg, Player sender) {

    Log.d( tag: "----------SHOW TEXT", msg: "trying to show text");
    runOnUiThread(new Runnable() {
        @Override
        public void run() {
            if (sender == null && serverMsg) {
                messageList.add(new ChatMessage( name: "Server", message, myMessage: false, R.drawable.robot));
            }
            if (sender != null) {
                if (sender.getId() == me.getId()) {
                    messageList.add(new ChatMessage(me.getName(), message, myMessage: true, me.getAvatarIcon()));
                } else {
                    messageList.add(new ChatMessage(sender.getName(), message, myMessage: false, sender.getAvatarIcon()));
                }
            }
            messagesListView.setAdapter(null);
            messagesListView.setAdapter(adapter);
            adapter.notifyDataSetChanged();
        }
    });
}
```

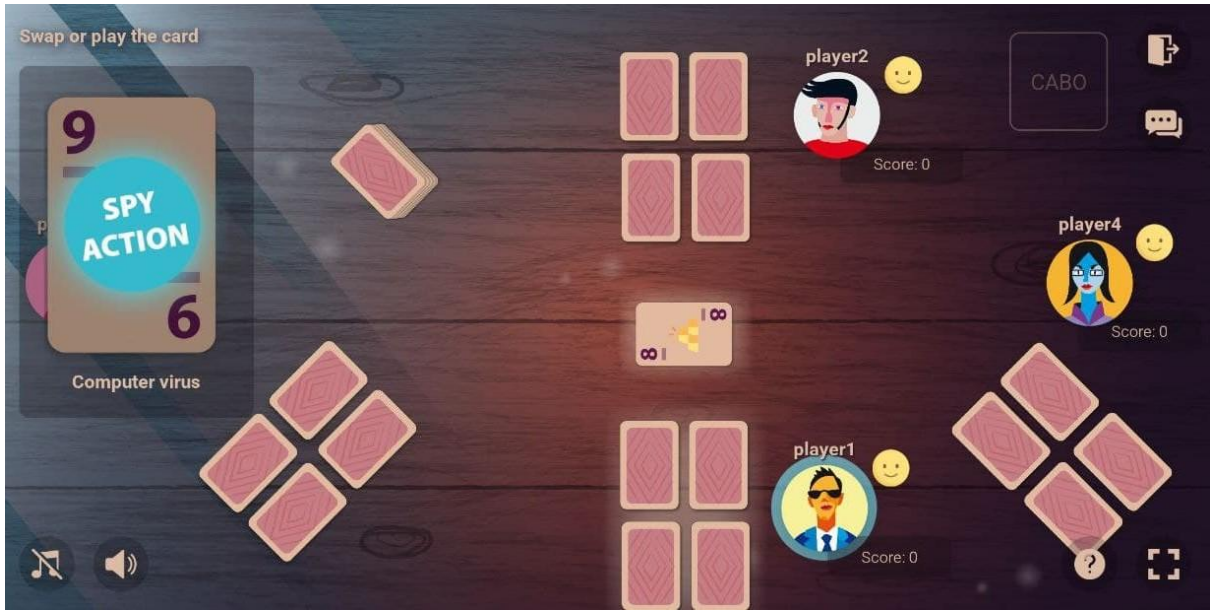
```java
public class ChatMessage {

    String name;
    String message;
    boolean myMessage;
    int avatar;

    public ChatMessage(String name, String message, boolean myMessage, int avatar){
        this.name = name;
        this.message = message;
        this.myMessage = myMessage;
        this.avatar = avatar;
    }

    public String getName() { return this.name; }

    public String getMessage() { return this.message; }

    public boolean getIfMyMessage() { return this.myMessage; }

    public int getAvatar(){ return this.avatar; }
}
```

Im **ChatAdapter** (extends BaseAdapter) wird je nachdem, ob es sich um eine eigene Message handelt oder eine fremde, ein anderes Layout inflated

```java
public View getView(int position, View convertView, ViewGroup parent) {
    ViewHolder holder;
    ChatMessage msg = this.chatMessageList.get(position);
    holder = new ViewHolder();


    if (convertView == null) {
        if(msg.getIfMyMessage()){
            convertView = layoutinflater.inflate(R.layout.my_chat_message, root: null);

        }
        else{
            convertView = layoutinflater.inflate(R.layout.their_chat_message, root: null);
            holder.avatar = (CircleImageView) convertView.findViewById(R.id.avatar);
            holder.avatar.setImageResource(msg.getAvatar());
        }

        holder.senderNameView = (TextView) convertView.findViewById(R.id.name);
        holder.chatMessageView = (TextView) convertView.findViewById(R.id.message_body);
        convertView.setTag(holder);

    } else {
        holder = (ViewHolder) convertView.getTag();
    }

    holder.senderNameView.setText(msg.getName());
    holder.chatMessageView.setText(msg.getMessage());

    return convertView;
}
```

# InGameActivity → von Grund auf aufgebaut

- Layout erstellt
- Brainstorming über Client-Server-Kommunikation zusammen mit Sophia → Protokoll
  - wann braucht die GUI welche Infos vom Server, um etwas anzuzeigen, eine Aktion einzuleiten
  - was muss innerhalb diese Aktion wieder geantwortet werden
- Umsetzung der Nutzer-Aktionen
- WICHTIG: Restriktionen für Nutzer → nur das klickbar was “erlaubt” und auch klar ersichtlich
- alle Eventualitäten abfangen

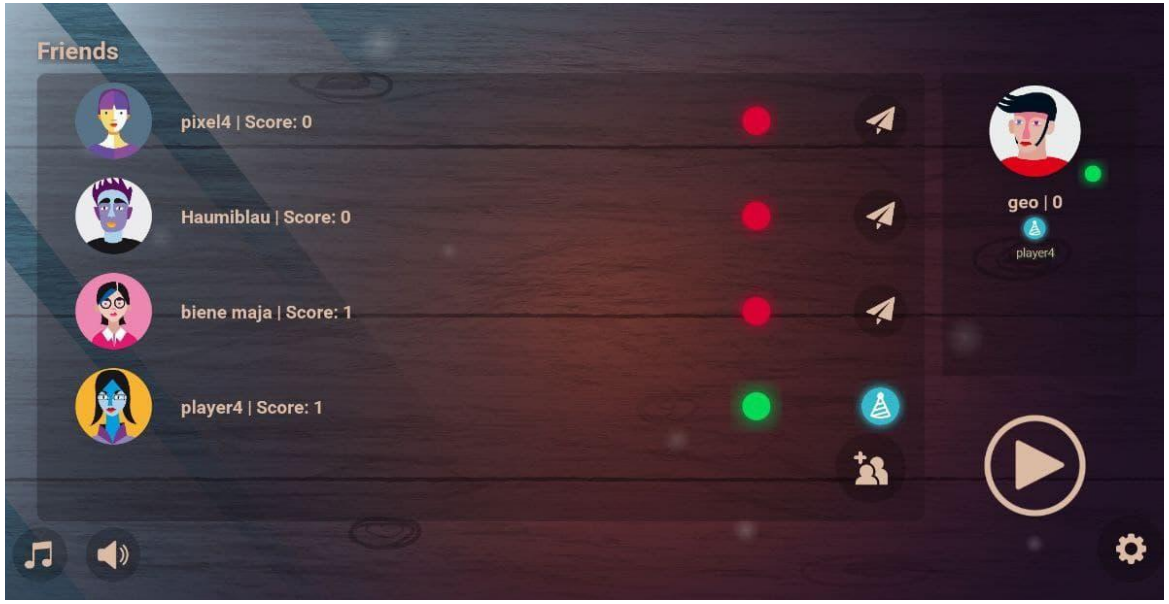


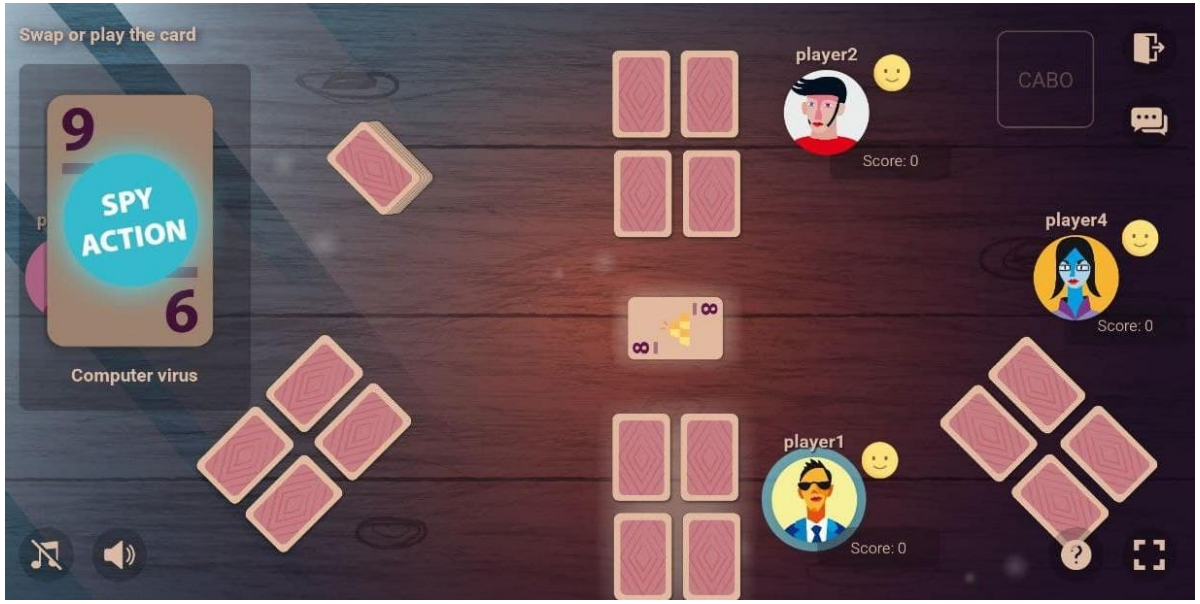
# InGameActivity → Live Code Review zu

## Reagieren mit Interaktionsmethoden auf Servermessages

# “Der Teufel liegt im Detail”

- jeder Bereich im Spiel soll ansprechend und liebevoll gestaltet sein
- User soll sich immer klar orientieren können und sich der Interaktionsmöglichkeiten bewusst sein → viele hints und visual cues eingebaut
- Textfeldeingaben angenehm handlen
- Notification bubble im Chat
- Chat in echtem Chat-design
- Custom Pop-Up Dialoge
- kontinuierliche Erweiterungen und Einbauen neuer Anregungen



# Bonus

SWAP ACTION

- Logo Design
- Erstellung eigener Assets in Photoshop und Illustrator (Hintergrund, Karten, Overlays)
- Game Changer LottieAnimation -> Einbinden von herunterladbaren Animationen
- Erstellung eigener Animation (card swap) in Adobe After Effects


CABO


# Game Changer: LottieAnimation

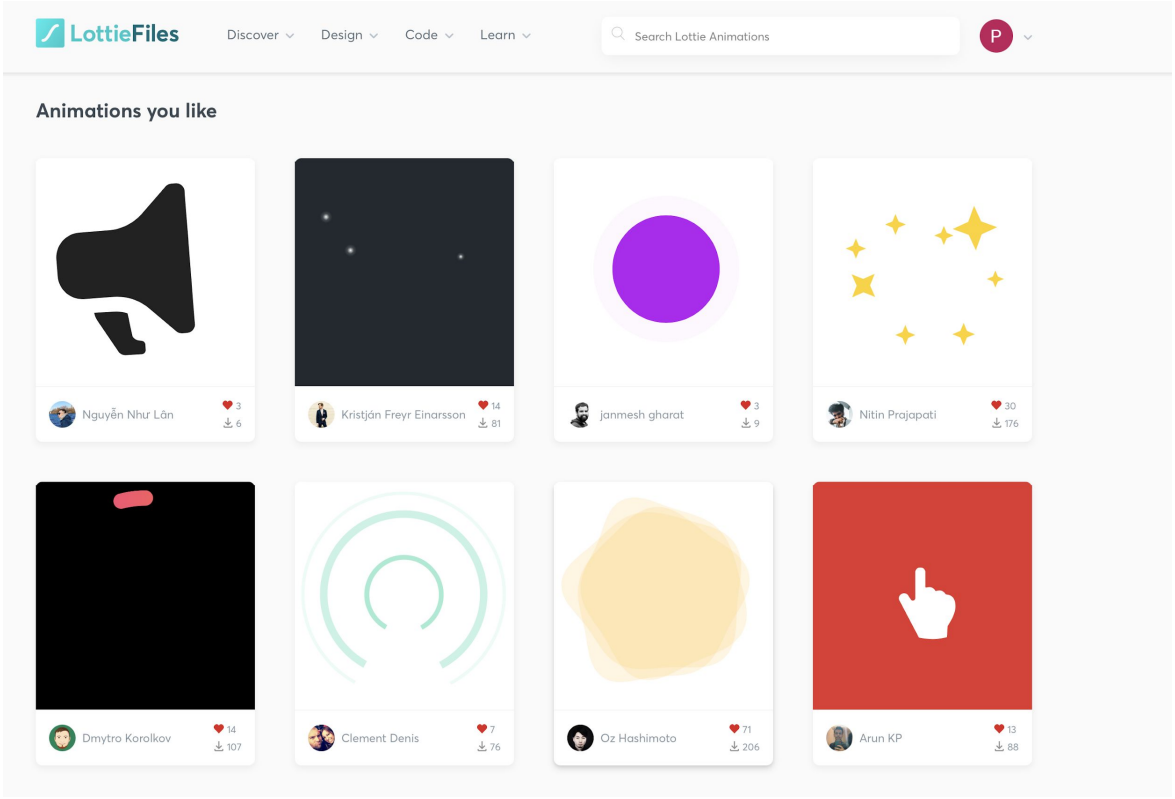
## Lottie forAndroid, iOS, Web, React Native, and Windows

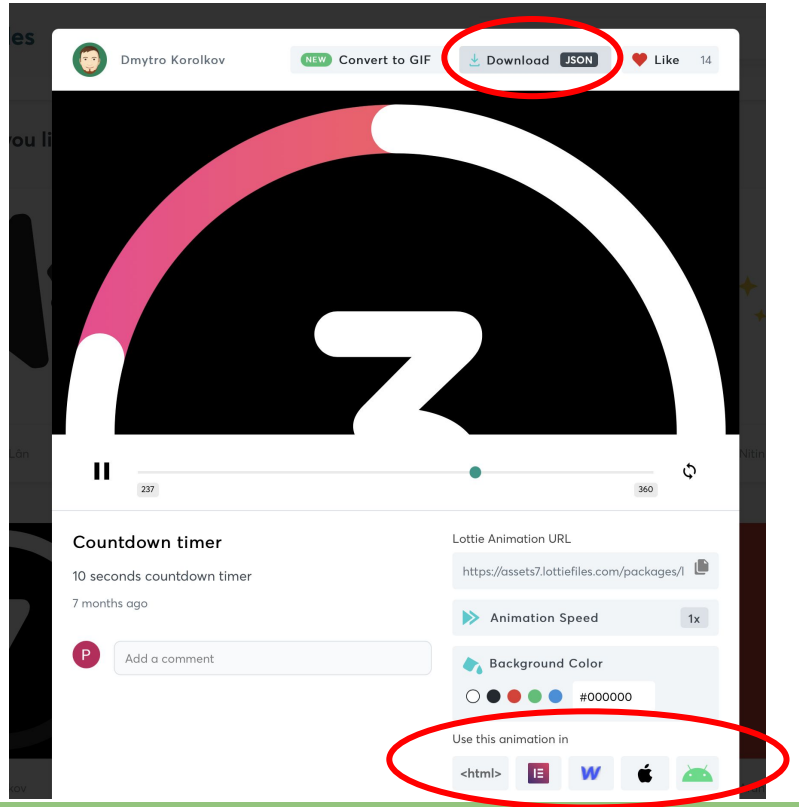

Lottie is a library for Android, iOS, Web, and Windows that parses Adobe After Effects animations exported as json with Bodymovin and renders them natively on mobile and on the web!

For the first time, designers can create **and ship** beautiful animations without an engineer painstakingly recreating it by hand. They say a picture is worth 1,000 words so here are 13,000:

# Game Changer: LottieAnimation

- Animationen aus großer kostenloser library als json-Datei herunterladen und so einfach wie ein Bild einbinden
- selbst mit After Effects erstellte Animationen in json umwandeln und einbinden
  → besonders für komplexere Animationen cool
  → wertet Anwendung optisch sehr auf

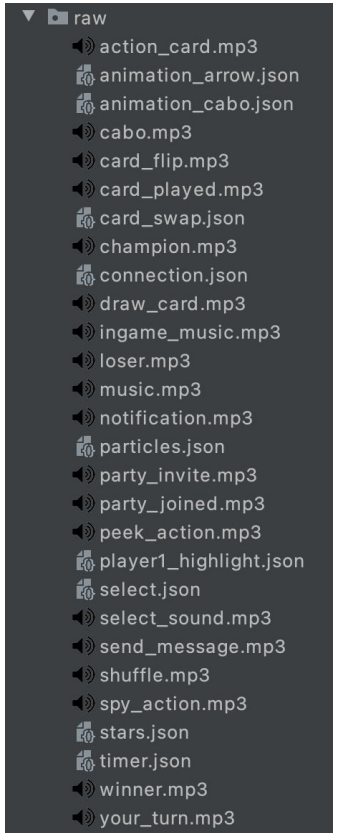


# Game Changer: LottieAnimation

```
dependencies {
    implementation 'androidx.appcompat:appcompat:1.2.0'
    implementation "androidx.fragment:fragment:1.2.5"
    implementation 'androidx.constraintlayout:constraintlayout:2.0.4'
    implementation 'androidx.lifecycle:lifecycle-extensions:2.1.0'
    implementation 'androidx.annotation:annotation:1.1.0'
    implementation 'com.google.android.material:material:1.2.1'
    implementation 'com.google.code.gson:gson:2.8.6'
    implementation 'com.google.android.gms:play-services-auth:19.0.0'
    implementation 'com.google.firebase:firebase-analytics'
    implementation 'com.google.firebase:firebase-database:19.2.1'
    implementation 'com.google.firebase:firebase-auth'
    implementation platform('com.google.firebase:firebase-bom:26.1.1')
    implementation 'com.github.AppIntro:AppIntro:6.0.0'
    implementation "org.java-websocket:Java-WebSocket:1.3.0"
    implementation 'com.squareup.okhttp3:okhttp:3.10.0'
    implementation 'org.codehaus.groovy:groovy-all:2.4.15'
    implementation 'de.hdodenhof:circleimageview:3.1.0'
    implementation 'com.otaliastudios:zoomlayout:1.8.0'
    def lottieVersion = "3.6.0"
    implementation "com.airbnb.android:lottie:$lottieVersion"

    testImplementation 'junit:junit:4.+'

    androidTestImplementation 'androidx.test.ext:junit:1.1.2'
    androidTestImplementation 'androidx.test.espresso:espresso-core:3.3.0'

}
```

Lottie in build.gradle dependencies angeben

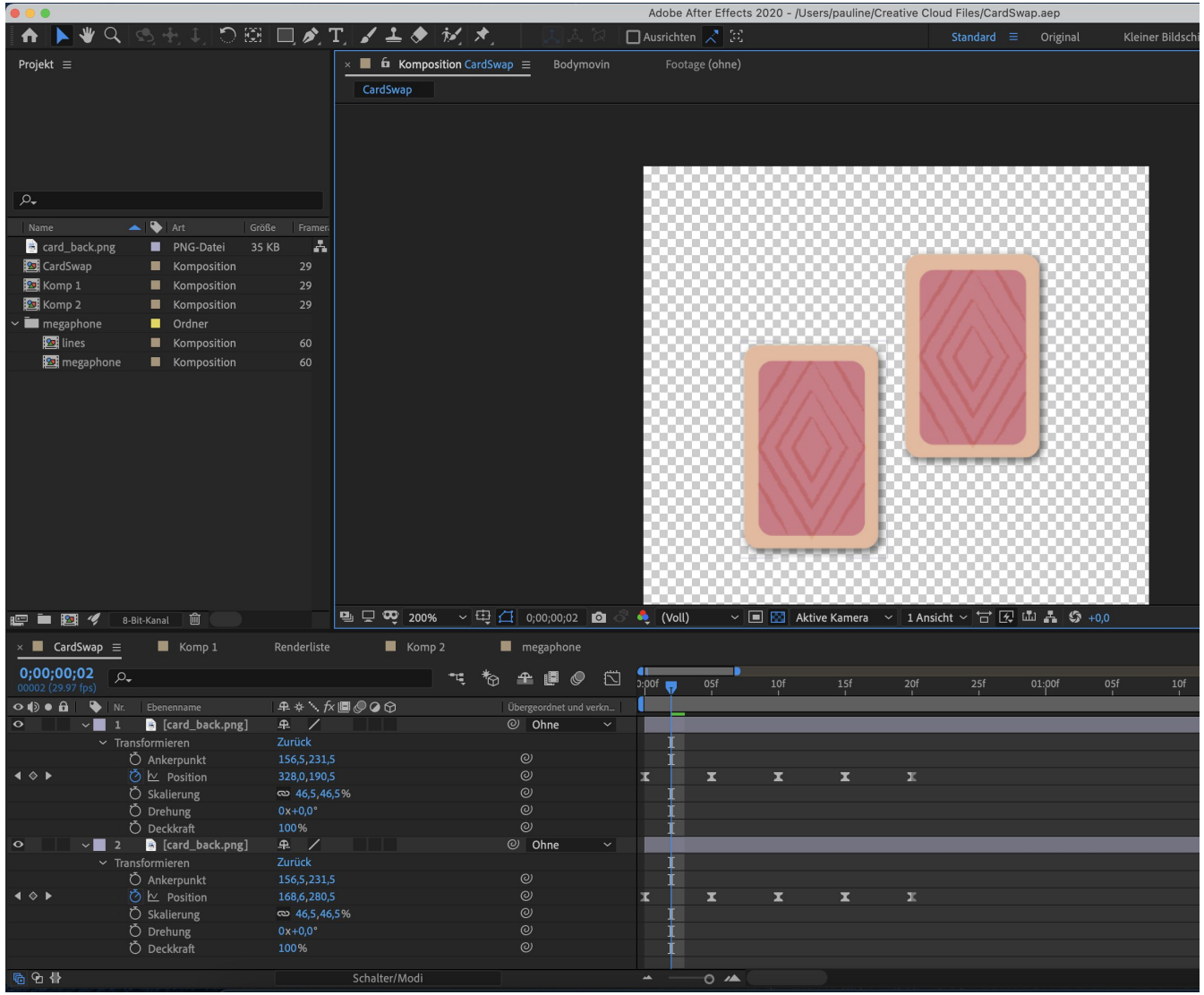



JSON Dateien
in raw-Ordner
ablegen

```
<com.airbnb.lottie.LottieAnimationView
    android:id="@+id/timer_animationView"
    android:layout_width="70dp"
    android:layout_height="60dp"
    app:lottie_rawRes="@raw/timer"
    app:lottie_autoPlay="true"
    app:lottie_loop="true"
    android:translationX="-100dp"
    app:layout_constraintBottom_toTopOf="@id/peek_button"
    app:layout_constraintLeft_toLeftOf="@id/peek_button"
    app:layout_constraintRight_toRightOf="@id/peek_button" />
```

In XML so einfach wie ein Bild einbinden

# Game Changer: LottieAnimation




selbst erstellte Animationen in Adobe After-Effects erstellen und mit Plugin auch in Lottie-JSON umwandeln

# Fragen?