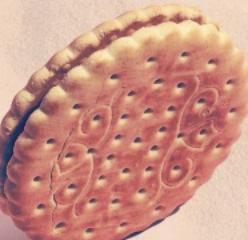


Introducción



Lucky Cookie

Tu aplicación ideal
para
Inspirarte y Aconsejarte

Introducción

Descripción de la App

Lucky Cointe es una aplicación que proporciona frases diarias, bromas y consejos. Utiliza tres API's diferentes para obtener contenido dirigido que los usuarios buscan y almacenar en su base de datos local (SQLite).

Objetivo de la aplicación

Lucky Cointe intenta alegrar el día a los usuarios mediante su variado contenido. La aplicación busca formar una comunidad entre personas que comparten interesantes frases y consejos útiles que se pueden tanto escuchar como guardar.

Público objetivo

Lucky Cointe está dirigida a personas que buscan inspiración diaria y humor en sus ratos libres. Es ideal para estudiantes y profesionales que utilizan dispositivos móviles y buscan mejorar su bienestar emocional sin costearse demasiado.



Descripción de la App

Lucky Cookie es una aplicación que proporciona frases diarias, bromas y consejos. Utiliza tres APIs diferentes para generar contenido dinámico que los usuarios pueden guardar y gestionar en su base de datos local (SQLite).

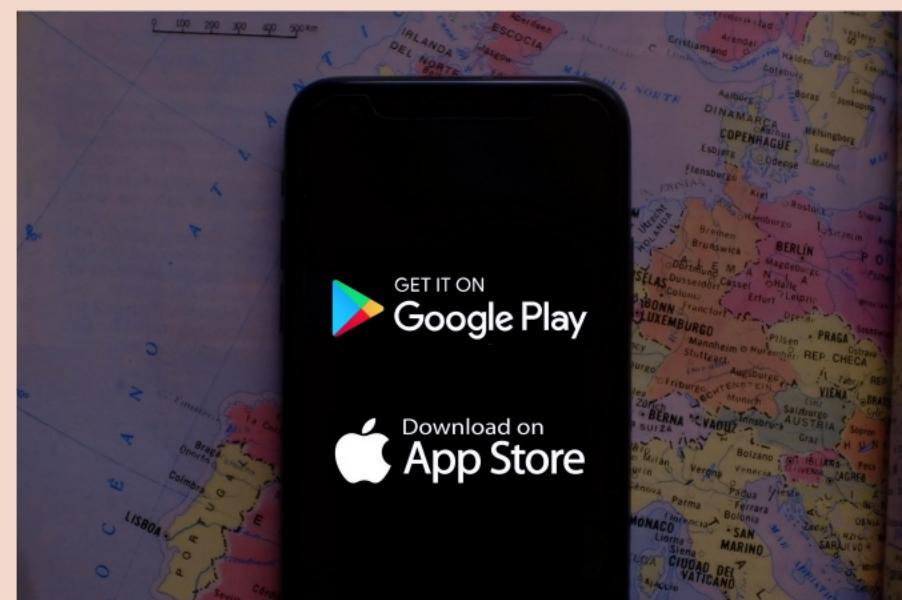
Objetivo de la aplicación

Lucky Cookie pretende alegrar el día a los usuarios mediante su variado contenido. La aplicación busca fomentar un ambiente positivo al ofrecer interesantes frases y consejos útiles que se pueden tanto escuchar como guardar.

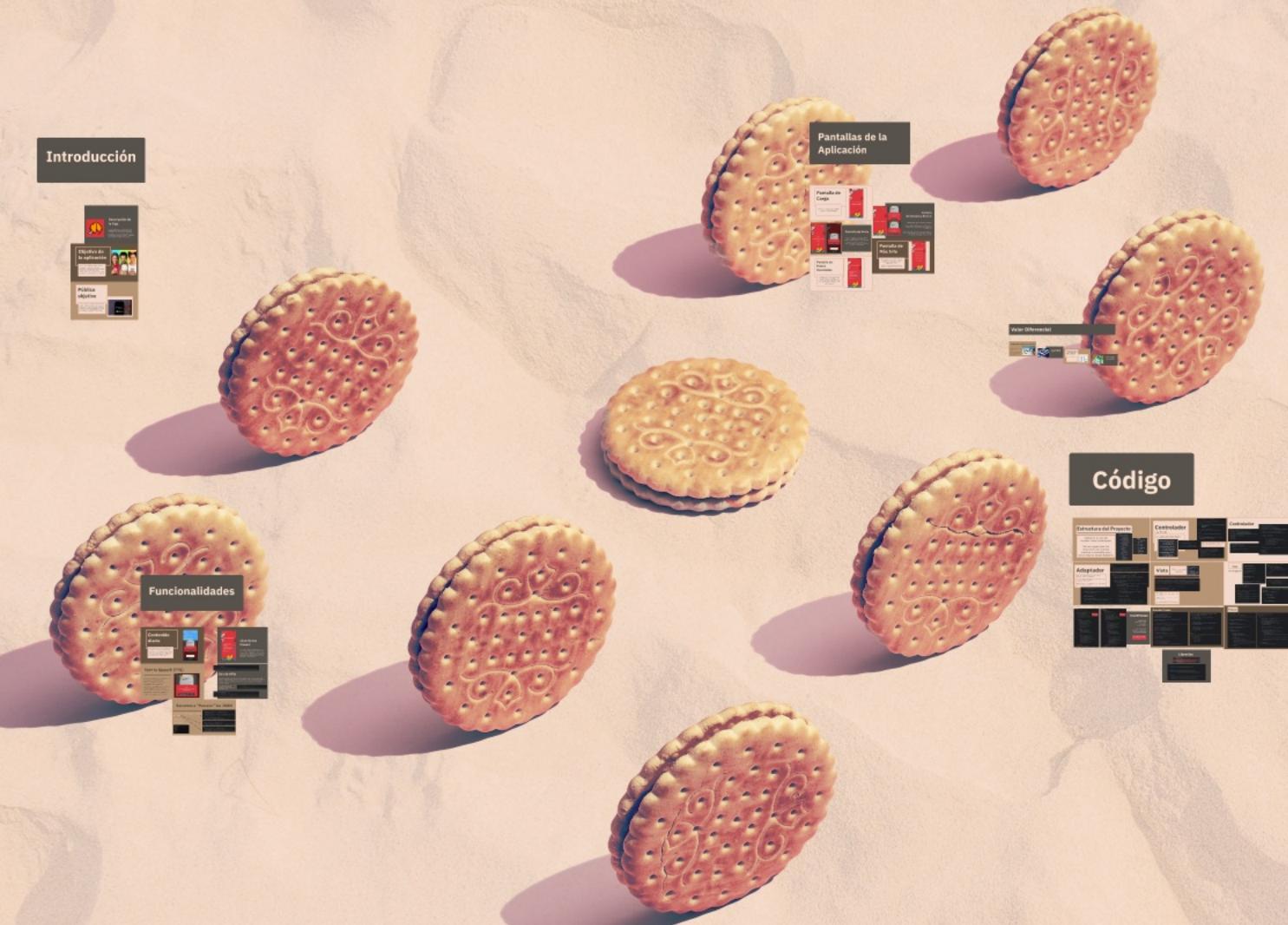


Público objetivo

Lucky Cookie está dirigida a personas que buscan inspiración diaria y humor en sus vidas. La aplicación es ideal para jóvenes y adultos que utilizan dispositivos móviles y buscan mejorar su bienestar emocional con contenido accesible y entretenido.



Introducción



Lucky Cookie

Tu aplicación ideal
para
Inspirarte y Aconsejarte

Funcionalidades

Contenido diario

Lucky Cookie proporciona frases dianas que ofrecen inspiración y motivación. Estas frases se actualizan constantemente a través de una API dedicada, asegurando que los usuarios siempre encuentren contenido fresco y estimulante.

Text to Speech (TTS)

En el apartado de consejos, se incluye una función de Text to Speech (TTS) que permite a los usuarios escuchar consejos en voz alta. Esta característica no solo hace que el contenido sea más accesible, sino que también mejora la experiencia de uso.

Uso de APIs

Lucky Cookie utiliza tres APIs diferentes para ofrecer un contenido variado: frases, consejos y bromas. Estas APIs permiten que el contenido se actualice regularmente y adapte el interés del usuario a través de información fresca y relevante.

Sacamos a "Parsear" los JSON

Las 3 APIs tienen algo en común: Sacamos la información de la misma manera. Solo cambian los datos.

```
try {  
    const response = await fetch('https://api.adviceslip.com/advice');  
    const data = await response.json();  
    console.log(data);  
}  
catch (error) {  
    console.error(error);  
}
```

```
const response = await fetch('https://api.adviceslip.com/advice');  
const data = await response.json();  
const quote = data.slip.quote;  
const author = data.slip.author;
```

```
const response = await fetch('https://api.adviceslip.com/advice');  
const data = await response.json();  
const quote = data.slip.quote;  
const author = data.slip.author;
```

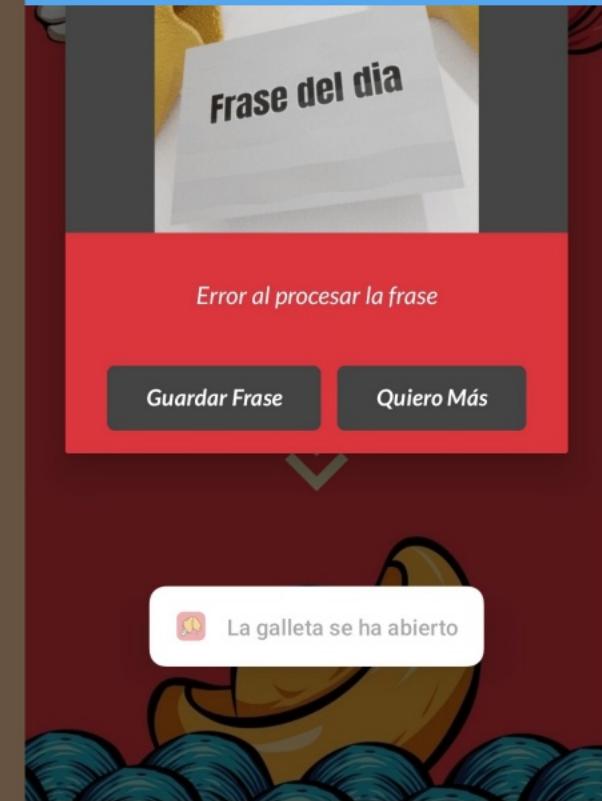
Contenido diario

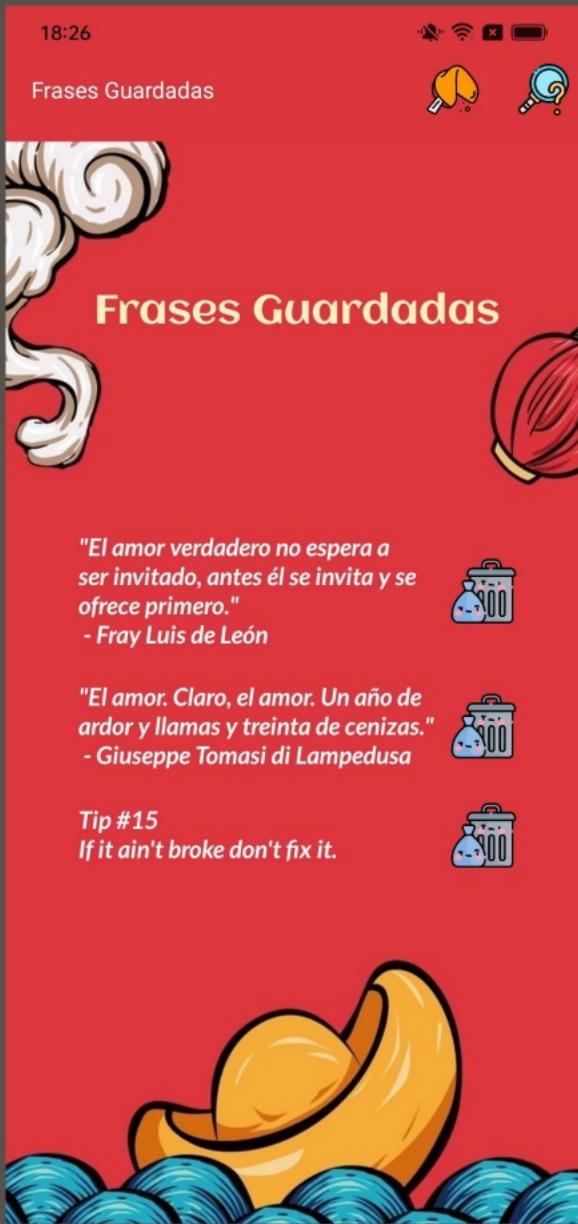
Lucky Cookie proporciona frases diarias que ofrecen inspiración y motivación. Estas frases se actualizan diariamente a través de una API dedicada, asegurando que los usuarios siempre encuentren contenido fresco y estimulante.

Error 403 - This web app is stopped.

The web app you have attempted to reach is currently stopped and does not accept any requests. Please try to reload the page or visit it again soon.

If you are the web app administrator, please find the common 403 error scenarios and resolution [here](#). For further troubleshooting tools and recommendations, please visit [Azure Portal](#).





¡Guarda tus Frases!

Los usuarios pueden guardar sus frases, consejos y bromas favoritos en una base de datos local (SQLite). Esta funcionalidad permite acceder fácilmente a contenido previamente guardado y disfrutar de una experiencia personalizada.

Text to Speech (TTS)

En el apartado de consejos, se incluye una función de Text to Speech (TTS) que permite a los usuarios escuchar consejos en voz alta. Esta característica no solo hace que el contenido sea más accesible, sino que también mejora la experiencia de uso.



```
public class ApiBromas { 2 usages
    // URL para conectarnos a la API de bromas
    private static final String URLBromas = "https://v2.jokeapi.dev/joke/Any?lang=es&type=twopart";
```

Uso de APIs

Lucky Cookie utiliza tres APIs diferentes para ofrecer un contenido variado: frases, consejos y bromas. Estas APIs permiten que el contenido se actualice dinámicamente y mantenga el interés del usuario a través de información fresca y relevante.

Las 3 vienen en formato JSON, lo cual permite que la extracción de la información, sea más sencilla

```
public class ApiConsejos { 2 usages
    // URL para conectarnos a la API de consejos
    private static final String URLConsejos = "https://api.adviceslip.com/advice";
```

```
public class ApiFrasediaria { 2 usages
    // URL para conectarnos a la API de frases diarias
    private static final String URLFrasediaria = "https://frasedeldia.azurewebsites.net/api/phrase";
```

Sacamos a "Parsear" los JSON

Las 3 APIs tienen algo en común
El JSON

Sacamos la
información de la
misma manera
Solo cambian los datos

```
{  
    "error": false,  
    "category": "Programming",  
    "type": "twopart",  
    "setup": "¿Por qué C consigue todas las chicas y Java no tiene ninguna?",  
    "delivery": "Porque C no las trata como objetos.",  
    "flags": {  
        "nsfw": false,  
        "religious": false,  
        "political": false,  
        "racist": false,  
        "sexist": false,  
        "explicit": false  
    },  
    "safe": true,  
    "id": 6,  
    "lang": "es"  
}
```

```
try {  
    JSONObject json0bject = new JSONObject(response);  
    // Nodo raíz  
    JSONObject slip0bject = json0bject.getJSONObject( name: "slip");  
    // Elementos del JSON  
    int id = slip0bject.getInt( name: "id");  
    String advice = slip0bject.getString( name: "advice");
```

```
try {  
    JSONObject json0bject = new JSONObject(response);  
    String frase = json0bject.getString( name: "phrase"); // Extraemos la frase diaria  
    String autor = json0bject.getString( name: "author"); // Extraemos el autor de esta
```

```
try {  
    JSONObject json0bject = new JSONObject(response);  
    String bromaparte1 = json0bject.getString( name: "setup"); // Parte 1  
    String bromaparte2 = json0bject.getString( name: "delivery"); // Parte 2
```

Introducción



Lucky Cookie

Tu aplicación ideal
para
Inspirarte y Aconsejarte

Pantallas de la Aplicación

Pantalla de Carga

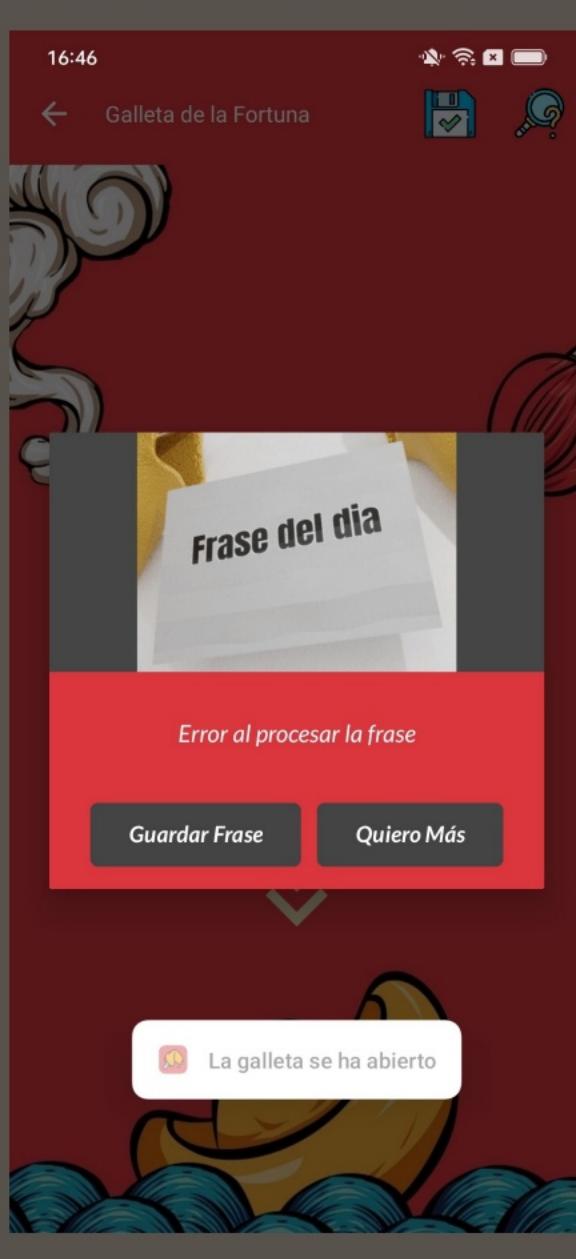
Contamos con el estilo y una progresión oculta en la parte inferior mientras cargamos la siguiente actividad.



Pantalla de Carga

Contamos con el título y una progressbar
oculta en la parte inferior mientras
cargamos la siguiente actividad





Pantalla de Inicio

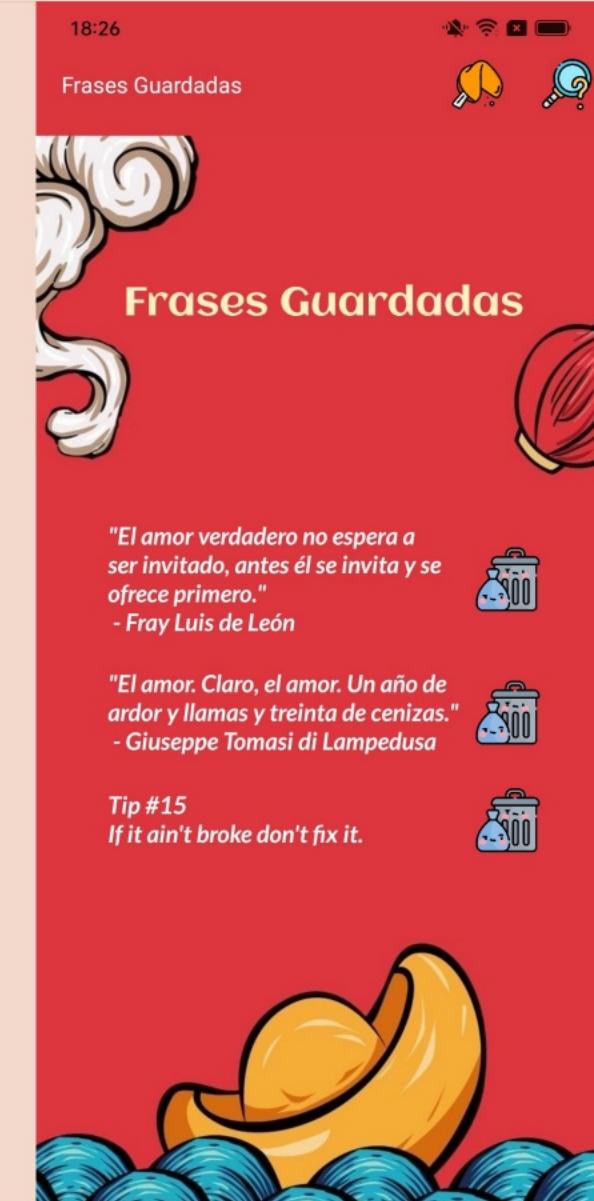
Nos encontramos con el título y abajo del todo un ImageButton una frase del día aleatoria, obtenida a través de la API de frases.

Incorpora un botón que permite abrir un popup animado junto a opciones para guardar o consultar más contenido.

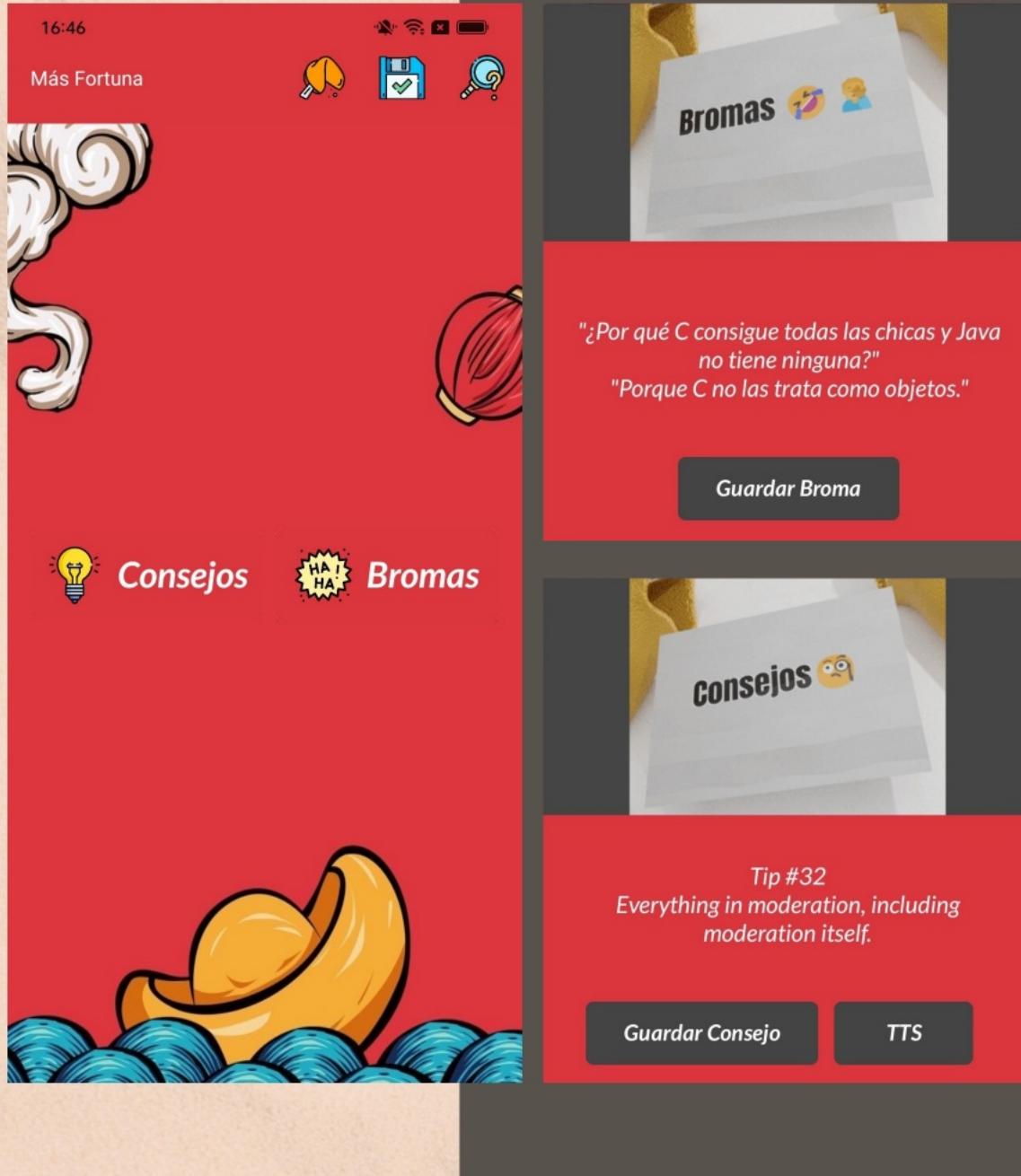
Pantalla de Frases Guardadas

Aquí gestionamos las frases, consejos y bromas favoritas almacenadas por el usuario en SQLite.

Utilizamos un RecyclerView que imprime esta información para presentar estos elementos de forma organizada, facilitando su visualización y acceso. Tenemos la posibilidad de borrar cada elemento con el botón a la derecha



Pantalla de Consejos y Bromas



Puedes elegir entre consejos y bromas.

Ambas están codificadas en dentro de un CardView, lo cual lo hace más estético e interactivo.

Estas opciones permiten al usuario optar por bromas aleatorias o por reproducir consejos en voz alta mediante la función TTS.

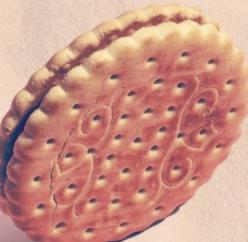
Pantalla de Más Info

Esta sección ofrece detalles sobre el desarrollo de Lucky Cookie, incluido el creador, horas de trabajo, líneas programadas y costos.

Aumenta la transparencia del proyecto, generando confianza entre los usuarios.

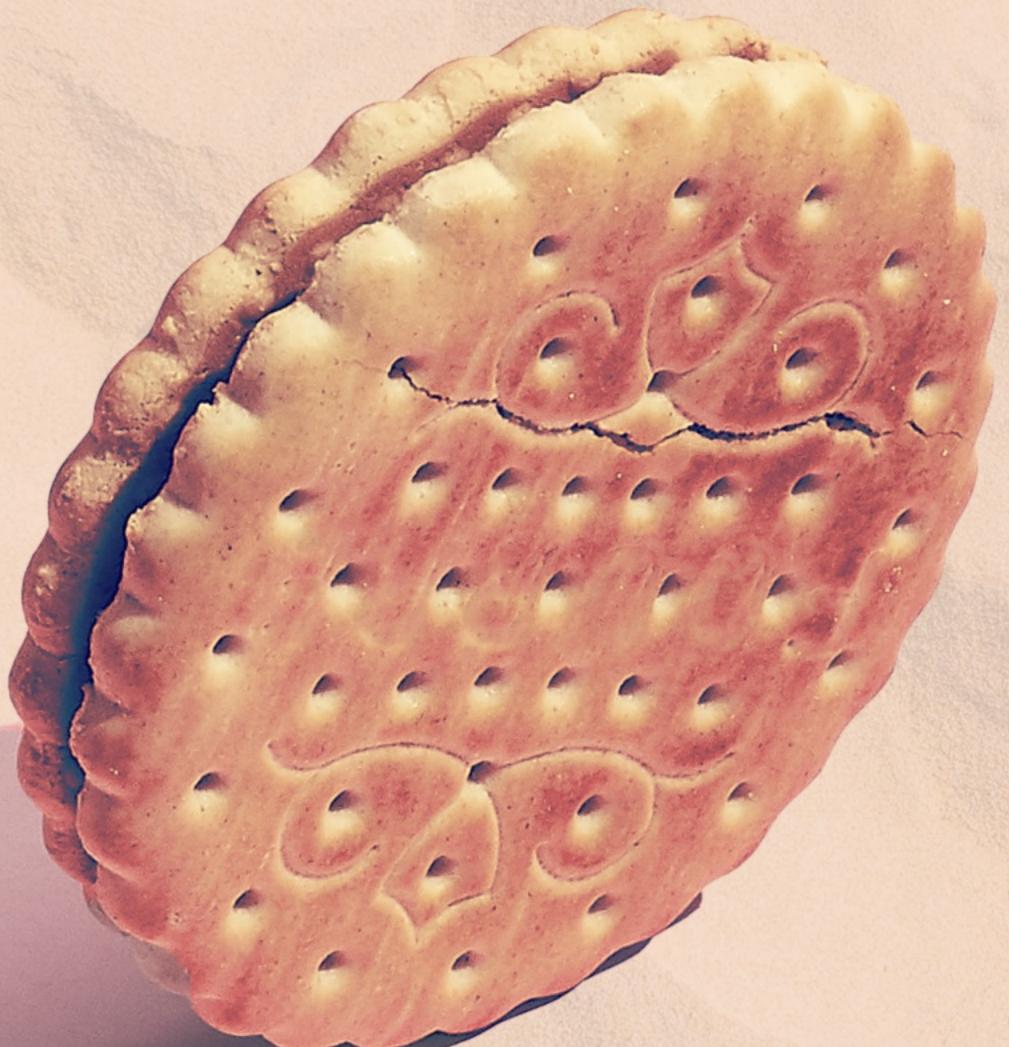


Introducción



Lucky Cookie

Tu aplicación ideal
para
Inspirarte y Aconsejarte



Código

Estructura del Proyecto

Mediante el uso del Modelo Vista-Controlador

Se han organizado los directorios de manera intuitiva y accesible para otros futuros desarrolladores

Controlador

Crear & Operar

Con la Base de Datos SQLite

- ControladorGeneral
- CrearAlumnos
- CrearCursos
- CrearMaterias
- CrearNotas
- CrearProfesores
- CrearTareas
- CrearUsers
- CrearVideos
- OperarAlumnos
- OperarCursos
- OperarMaterias
- OperarNotas
- OperarProfesores
- OperarTareas
- OperarUsers

Controlador

Adaptador

Este patrón sirve al convertir dentro de un sistema de datos que no se maneja bien.

En el adaptador debe tener las acciones donde se van a dar las tareas o operaciones.

Controlador -> Redirecciona las acciones con información.

Vista

Parte que ve el usuario

- FormsController
- Modelo
- OpeningScreen
- StartScreen

Vista

Text To Speech

Guardar Frases

CardViews

Una buena alternativa a los cards tradicionales

Permite juntar imágenes y textos en un solo layout

Popups

Librerías

- Algoritmos
- Base de Datos
- Controladores
- Modelos
- Operaciones
- Patrones
- Plataformas
- Prácticas
- Procesos
- Programación
- Redes
- Sistemas
- UI
- Utilidades

Estructura del Proyecto

Mediante el uso del
Modelo Vista-Controlador

Se han organizado las
directorios de manera
intuitiva y accesible para
otros futuros desarrolladores

The diagram illustrates the project structure. It features two main vertical columns on a dark background. The left column is labeled 'java' at the top and contains a tree view of Java packages and classes. The right column is labeled 'res' at the top and contains a list of resource types. A horizontal line connects the two columns.

- java
 - com.example.galleta
 - ControladorBromas
 - CrearDBBromas
 - OperarDBBromas
 - ControladorConsejos
 - CrearDBConsejos
 - OperarDBConsejos
 - ControladorFrases
 - CrearDBFrases
 - OperarDBFrases
 - Helpers
 - Adaptador
 - Modelo
 - ApiBromas
 - ApiConsejos
 - ApiFraseDiaria
 - Vista
 - FortuneCookie
 - Help
 - MoreAPIs
 - OpeningScreen
 - SavePhrase

- res
 - drawable
 - font
 - layout
 - menu
 - mipmap
 - raw
 - values
 - xml

Controlador

Crear & Operar

Con la Base de Datos SQLite

- ✓ ControladorBromas
 - © CrearDBBromas
 - © OperarDBBromas
- ✓ ControladorConsejos
 - © CrearDBConsejos
 - © OperarDBConsejos
- ✓ ControladorFrases
 - © CrearDBFrases
 - © OperarDBFrases

```
public class OperarDBConsejos { 9 usages

    private CrearDBConsejos crearDB; 5 usages

    public OperarDBConsejos(Context contexto) { 3 usages
        crearDB = new CrearDBConsejos(contexto);
    }
```

```
public class CrearDBConsejos extends SQLiteOpenHelper { 2 usages

    public static final String DATABASE_NAME = "galleta_consejos.db"; 1 usage
    public static final int DATABASE_VERSION = 1; 1 usage

    // Sentencia para crear la tabla de consejos
    public static final String CREATE_TABLE = "CREATE TABLE IF NOT EXISTS Consejos (" +
        "id INTEGER PRIMARY KEY, " +
        "advice TEXT);";

    public CrearDBConsejos(Context contexto) { 1 usage
        super(contexto, DATABASE_NAME, factory: null, DATABASE_VERSION);
    }

    @Override
    public void onCreate(SQLiteDatabase db) { db.execSQL(CREATE_TABLE); }

    @Override 10 usages
    public void onUpgrade(SQLiteDatabase db, int oldVersion, int newVersion) {
        db.execSQL("DROP TABLE IF EXISTS Consejos");
        onCreate(db);
    }
}
```

getWritableDatabase() es un método que abre la base de datos en modo escritura.
Inserción, actualización y eliminación

```
// Insertar un nuevo consejo en la base de datos
public void insertarConsejo(int id, String advice) { 1 usage
    SQLiteDatabase db = crearDB.getWritableDatabase();
    ContentValues values = new ContentValues();
    values.put("id", id);
    values.put("advice", advice);
    db.insert( table: "Consejos", nullColumnHack: null, values);
    db.close();
}
```

Controlador

```
// Comprobar si el consejo ya existe para evitar duplicados
public boolean existeConsejo(int id) { 3 usages
    SQLiteDatabase db = crearDB.getReadableDatabase();
    String query = "SELECT id FROM Consejos WHERE id = ?";
    Cursor cursor = db.rawQuery(query, new String[]{String.valueOf(id)});
    boolean existe = cursor.getCount() > 0;
    cursor.close();
    db.close();
    return existe;
}
```

```
// Método para eliminar un consejo de la base de datos
public void borrarConsejo(int id) { 1 usage
    SQLiteDatabase db = crearDB.getWritableDatabase();
    int filasAfectadas = db.delete(table: "Consejos", whereClause: "id = ?", new String[]{String.valueOf(id)});
    db.close();

    if (filasAfectadas > 0) {
        Log.d(tag: "SQLite", msg: "Consejo eliminado correctamente: " + id);
    } else {
        Log.e(tag: "SQLite", msg: "Error: No se encontró el consejo con ID " + id);
    }
}
```

```
// Obtener los consejos guardados
public List<String> obtenerConsejos() { 1 usage
    List<String> consejos = new ArrayList<>();
    SQLiteDatabase db = crearDB.getReadableDatabase();
    Cursor consulta = db.rawQuery(sql: "SELECT id, advice FROM Consejos", selectionArgs: null);

    int idIndex = consulta.getColumnIndex(columnName: "id");
    int adviceIndex = consulta.getColumnIndex(columnName: "advice");

    if (idIndex != -1 && adviceIndex != -1) {
        if (consulta.moveToFirst()) {
            do {
                int id = consulta.getInt(idIndex);
                String advice = consulta.getString(adviceIndex);
                String consejoFormatizado = "Tip #" + id + "\n" + advice;
                consejos.add(consejoFormatizado);
                //consejos.add("Consejo #: " + id + "\n" + advice);
            } while (consulta.moveToNext());
        }
    } else {
        Log.e(tag: "DatabaseError", msg: "Las columnas 'id' y/o 'advice' no se encuentran en la consulta.");
    }
    consulta.close();
    db.close();
    return consejos;
}
```

Adaptador

Nos permite adaptar el contenido dentro de un RecyclerView. Tiene métodos como:

- OnCreateViewHolder: Crear los espacios donde se van a poner los items o información (pasillo de congelados)
- OnBindViewHolder: Rellena los espacios con información (Pizza carbonara congelada)

```
    } else if (item.startsWith("Broma #")) {  
        // Si el texto empieza con "Broma #", se trata de una broma  
  
        // Extraer el ID de la broma  
        int idBroma = Integer.parseInt(item.replace("Broma #", "").  
            split(regex: "\n")[0].trim());  
  
        // Eliminar la broma de la base de datos  
        dbOpBromas.borrarBroma(idBroma);  
  
        // Comprobar si realmente se eliminó  
        if (!dbOpConsejos.existeConsejo(idBroma)) { // Posible error: debería ser dbOpBromas  
            Log.d(tag: "Adaptador", msg: "Broma eliminada de la BD correctamente.");  
        } else {  
            Log.e(tag: "Adaptador", msg: "Error: La broma sigue existiendo en la BBDD.");  
        }  
    }
```

```
// Método que se ejecuta cuando se necesita crear una nueva vista para un elemento  
@NotNull  
@Override  
public ViewHolder onCreateViewHolder(@NotNull ViewGroup parent, int viewType) {  
    // Infla el diseño XML de un ítem en un objeto View  
    View itemView = LayoutInflater.from(context).inflate(R.layout.item_frase, parent, attachToRoot: false);  
    return new ViewHolder(itemView); // Retorna una instancia de ViewHolder con la vista creada  
}  
  
// Método que asigna datos a una vista cuando aparece en pantalla  
@Override  
public void onBindViewHolder(ViewHolder holder, int position) {  
    String item = items.get(position); // Obtiene el elemento en la posición actual  
    holder.textoItem.setText(item); // Muestra el texto en el TextView de la vista  
  
    // Establece la acción cuando el usuario presiona el botón de borrar  
    holder.borrar.setOnClickListener(v -> {  
        if (item.contains(".")) {  
            // Si la cadena contiene comillas, se asume que es una frase y se elimina de la BD  
            dbOpFrases.borrarFrase(item);  
            Log.d(tag: "Adaptador", msg: "Frase eliminada: " + item);  
        }  
    });  
}  
  
} else if (item.startsWith("Tip #")) {  
    // Si el texto empieza con "Tip #", se trata de un consejo  
  
    // Extraer el ID del consejo  
    int id = Integer.parseInt(item.replace("Tip #", "").  
        split(regex: "\n")[0].trim());  
  
    // Eliminar de la base de datos  
    dbOpConsejos.borrarConsejo(id);  
  
    // Comprobar si realmente se eliminó  
    if (!dbOpConsejos.existeConsejo(id)) {  
        Log.d(tag: "Adaptador", msg: "Consejo eliminado de la BD correctamente.");  
    } else {  
        Log.e(tag: "Adaptador", msg: "Error: El consejo sigue existiendo en la BBDD.");  
    }  
}
```

Vista

Parte que ve el usuario

- © FortuneCookie
- © Help
- © MoreAPIs
- © OpeningScreen
- © SavePhrase

```
public class MoreAPIs extends AppCompatActivity implements TextToSpeech.OnInitListener {  
  
    // Ejecutores y manejadores para realizar operaciones en segundo plano  
    private final ExecutorService ejecutor = Executors.newSingleThreadExecutor();  
    private final Handler handler = new Handler(Looper.getMainLooper()); 2 usages  
  
    // Variables para el motor de Text-to-Speech (TTS)  
    private TextToSpeech tts; 6 usages  
    private boolean ttsReady = false; 2 usages
```

ExecutorService:

- Ejecuta tareas pesadas en segundo plano para evitar bloqueos en la UI.

Handler

- Permite actualizar la UI desde un hilo en segundo plano de manera segura.

Ambas garantizan que la app funcione de manera fluida y no se congele mientras obtiene datos de una API.

Vista

Text To Speech

```
// Método para inflar el menú de la toolbar
@Override
public boolean onCreateOptionsMenu(Menu menu) {
    getMenuInflater().inflate(R.menu.menu_moreapis, menu);
    return true;
}

// Método que maneja la selección de opciones en la barra de herramientas
@Override
public boolean onOptionsItemSelected(MenuItem item) {
    if (item.getItemId() == R.id.ir_savePhrase) {
        startActivity(new Intent(packageContext: MoreAPIs.this, SavePhrase.class));
        return true;
    } else if (item.getItemId() == R.id.inicio) {
        startActivity(new Intent(packageContext: MoreAPIs.this, FortuneCookie.class));
        return true;
    } else if (item.getItemId() == R.id.help) {
        startActivity(new Intent(packageContext: MoreAPIs.this, Help.class));
        return true;
    } else {
        return super.onOptionsItemSelected(item);
    }
}
```

```
@Override
protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    EdgeToEdge.enable($this$enableEdgeToEdge: this);
    setContentView(R.layout.activity_more_apis);

    // Inicializar el motor de síntesis de voz (TTS)
    tts = new TextToSpeech(context: this, listener: this);

    // CardView para abrir popups de consejos
    CardView consejos = findViewById(R.id.Consejos);
    consejos.setOnClickListener(v -> {
        Toast.makeText(context: this, text: "Consejos en Inglés", Toast.LENGTH_SHORT).show();
        popupConsejos();
    });

    // CardView para abrir popups de bromas
    CardView bromas = findViewById(R.id.Bromas);
    bromas.setOnClickListener(v -> {
        Toast.makeText(context: this, text: "Bromas", Toast.LENGTH_SHORT).show();
        popupBromas();
    });

    // Configuración de la toolbar
    androidx.appcompat.widget.Toolbar toolbar = findViewById(R.id.toolbar);
    setSupportActionBar(toolbar);
}

// Método para inicializar el TTS
private void initTTS() {
    tts = new TextToSpeech(context: this, listener: this) {
        @Override
        protected void onInit(int status) {
            if (status == TextToSpeech.SUCCESS) {
                int result = tts.setLanguage(Locale.US);
                if (result == TextToSpeech.LANG_MISSING_DATA || result == TextToSpeech.LANG_NOT_SUPPORTED) {
                    Log.e(tag: "TTS", msg: "Idioma no soportado");
                } else {
                    ttsReady = true;
                }
            } else {
                Log.e(tag: "TTS", msg: "Error en la inicialización");
            }
        }
    };
}
```

```
// Método que se ejecuta cuando el TTS se inicializa
@Override 1 usage
public void onInit(int status) {
    if (status == TextToSpeech.SUCCESS) {
        int result = tts.setLanguage(Locale.US);
        if (result == TextToSpeech.LANG_MISSING_DATA || result == TextToSpeech.LANG_NOT_SUPPORTED) {
            Log.e(tag: "TTS", msg: "Idioma no soportado");
        } else {
            ttsReady = true;
        }
    } else {
        Log.e(tag: "TTS", msg: "Error en la inicialización");
    }
}
```

```
// Método para hacer que el TTS lea un texto en voz alta
private void speakText(String text) { 1 usage
    if (ttsReady && !text.isEmpty()) {
        tts.speak(text, TextToSpeech.QUEUE_FLUSH, params: null, utteranceId: null);
    }
}
```

```
| // Método que se ejecuta cuando la actividad se destruye
@Override
protected void onDestroy() {
    if (tts != null) {
        tts.stop();
        tts.shutdown();
    }
    super.onDestroy();
}
```

Popups

```
// Método para mostrar el popup de consejos
public void popupConsejos() { 1 usage
    LayoutInflator inflater = LayoutInflator.from( context: this);
    View popupView = inflater.inflate(R.layout.popupconsejos_layout,  root: null);

    AlertDialog alertDialog = new AlertDialog.Builder( context: this).setView(popupView).create();

    TextView TV = popupView.findViewById(R.id.ConsejosRecibir);
    Button btnSpeak = popupView.findViewById(R.id.btnTTS);
    Button guardarConsejo = popupView.findViewById(R.id.GuardarConsejo);

    // Método para mostrar el popup de bromas
public void popupBromas() { 1 usage
    LayoutInflator inflater = LayoutInflator.from( context: this);
    View popupView = inflater.inflate(R.layout.popupbromas_layout,  root: null);

    AlertDialog alertDialog = new AlertDialog.Builder( context: this).setView(popupView).create();

    TextView TV = popupView.findViewById(R.id.BromasRecibir);
    Button guardarBroma = popupView.findViewById(R.id.GuardarBroma);

    // Llamar a la API de bromas en segundo plano
ejecutor.execute() -> {
    String response = ApiBromas.getBroma();
    handler.post() -> {
        try {
            JSONObject jsonObject = new JSONObject(response);
            // Extraen las partes de la broma del JSON
            String bromaparte1 = jsonObject.getString( name: "setup");
            String bromaparte2 = jsonObject.getString( name: "delivery");
            String bromaformateada = "\n" + bromaparte1 + "\n" + bromaparte2 + "\n";
            TV.setText(bromaformateada);

            // Botón para guardar la broma en la base de datos
guardarBroma.setOnClickListener(v -> {
            OperarDBBromas dbBromas = new OperarDBBromas( contexto: this);
            if (!dbBromas.existeBroma(bromaparte1, bromaparte2)) {
                dbBromas.insertarBroma(bromaparte1, bromaparte2, source: "ApiBromas");
                Toast.makeText( context: this,  text: "Broma guardada correctamente", Toast.LENGTH_SHORT).show();
            } else {
                Toast.makeText( context: this,  text: "Esta broma ya está guardada", Toast.LENGTH_SHORT).show();
            }
            startActivity(new Intent( packageName: MoreAPIs.this, SavePhrase.class));
            alertDialog.dismiss();
        });
    };
} catch (JSONException e) {
    TV.setText("Error al procesar la broma");
}}
```

```
// Llamar a la API de consejos en segundo plano
ejecutor.execute() -> {
    String response = ApiConsejos.getConsejo();
    handler.post() -> {
        try {
            JSONObject jsonObject = new JSONObject(response);
            // Extraer datos del JSON recibido
            JSONObject slipObject = jsonObject.getJSONObject( name: "slip");
            int id = slipObject.getInt( name: "id");
            String advice = slipObject.getString( name: "advice");
            String consejoFormatizado = "Tip # " + id + "\n" + advice;
            TV.setText(consejoFormatizado);

            // Botón para leer el consejo en voz alta
btnSpeak.setOnClickListener(v -> speakText(consejoFormatizado));

            // Botón para guardar el consejo en la base de datos
guardarConsejo.setOnClickListener(v -> {
            OperarDBConsejos dbOp = new OperarDBConsejos( contexto: this);
            if (!dbOp.existeConsejo(id)) {
                dbOp.insertarConsejo(id, advice);
                Toast.makeText( context: this,  text: "Consejo guardado correctamente.", Toast.LENGTH_SHORT).show();
            } else {
                Toast.makeText( context: this,  text: "Este consejo ya está guardado.", Toast.LENGTH_SHORT).show();
            }
            startActivity(new Intent( packageName: MoreAPIs.this, SavePhrase.class));
            alertDialog.dismiss();
        });

    } catch (JSONException e) {
        TV.setText("Error al procesar el consejo");
    };
};
```

Guardar Frases

```
@Override  
protected void onCreate(Bundle savedInstanceState) {  
    super.onCreate(savedInstanceState);  
    setContentView(R.layout.activity_save_phrase);  
    EdgeToEdge.enable($this$enableEdgeToEdge: this);  
  
    androidx.appcompat.widget.Toolbar toolbar = findViewById(R.id.toolbar);  
    setSupportActionBar(toolbar);  
  
    // Reproducir música al entrar en esta activity  
    sonido = MediaPlayer.create(context: this, R.raw.soundeffectchina); // Cargamos el sonido  
    sonido.start(); // Reproduce automáticamente el sonido  
  
    // Inicializar las operaciones de la base de datos  
    dbOpFrases = new OperarDBFrases(contexto: this);  
    dbOpConsejos = new OperarDBConsejos(contexto: this);  
    dbOpBromas = new OperarDBBromas(contexto: this);  
  
    // Configurar el RecyclerView  
    recyclerViewFrases = findViewById(R.id.listViewFrases);  
  
    // Obtener y combinar los datos de frases, consejos y bromas  
    items = new ArrayList<>();  
    obtenerDatosUnificados();  
  
    // Configurar el Adapter y LayoutManager  
    adapter = new Adaptador(context: this, items);  
    recyclerViewFrases.setAdapter(adapter);  
    recyclerViewFrases.setLayoutManager(new LinearLayoutManager(context: this));  
}  
}
```

```
// Método para obtener y combinar frases, consejos y bromas  
private void obtenerDatosUnificados() { 1 usage  
    // Obtener frases  
    List<String> frases = dbOpFrases.obtenerFrases();  
    items.addAll(frases);  
  
    // Obtener consejos  
    List<String> consejos = dbOpConsejos.obtenerConsejos();  
    items.addAll(consejos);  
  
    // Obtener bromas  
    List<String> bromas = dbOpBromas.obtenerBromas();  
    items.addAll(bromas);  
}  
  
@Override  
// Método para liberar y destruir el sonido al salir de la activity  
protected void onDestroy() {  
    super.onDestroy();  
    if (sonido != null) {  
        sonido.release();  
        sonido = null;  
    }  
}
```

CardViews

Una buena alternativa a los botones tradicionales

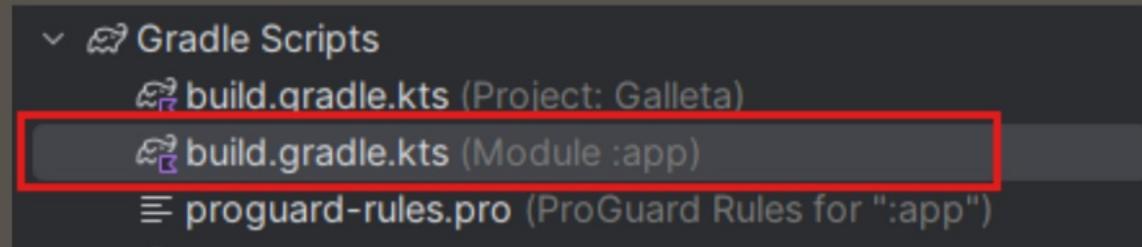
Permite juntar imágenes y botones en un solo layout



```
<LinearLayout  
    android:orientation="vertical">  
  
    <!-- CardView como alternativa a los botones convencionales -->  
    <androidx.cardview.widget.CardView  
        android:id="@+id/Consejos"  
        android:layout_width="wrap_content"  
        android:layout_height="wrap_content"  
        android:background="@android:color/white">  
  
        <LinearLayout  
            android:id="@+id/linearcosejos"  
            android:layout_width="match_parent"  
            android:layout_height="match_parent"  
            android:background="#DB363E"  
            android:gravity="center_vertical"  
            android:orientation="horizontal"  
            android:padding="10dp">  
  
            <ImageView  
                android:layout_width="40dp"  
                android:layout_height="40dp"  
                android:src="@drawable/consejos_icon" />  
  
            <TextView  
                android:layout_width="wrap_content"  
                android:layout_height="wrap_content"  
                android:layout_marginStart="10dp"  
                android:fontFamily="@font/lato_bold_italic"  
                android:text="Consejos"  
                android:textColor="@color/white"  
                android:textSize="25sp" />  
  
        </LinearLayout>  
    </androidx.cardview.widget.CardView>  
</LinearLayout>
```

```
<LinearLayout  
    android:gravity="center"  
    android:orientation="vertical">  
  
    <androidx.cardview.widget.CardView  
        android:id="@+id/Bromas"  
        android:layout_width="wrap_content"  
        android:layout_height="wrap_content"  
        android:background="@android:color/white"  
        app:cardCornerRadius="5dp">  
  
        <LinearLayout  
            android:layout_width="match_parent"  
            android:layout_height="match_parent"  
            android:background="#DB363E"  
            android:gravity="center_vertical"  
            android:orientation="horizontal"  
            android:padding="13dp">  
  
            <ImageView  
                android:layout_width="40dp"  
                android:layout_height="40dp"  
                android:src="@drawable/bromas_icon" />  
  
            <TextView  
                android:layout_width="wrap_content"  
                android:layout_height="wrap_content"  
                android:layout_marginStart="10dp"  
                android:fontFamily="@font/lato_bold_italic"  
                android:text="Bromas"  
                android:textColor="@color/white"  
                android:textSize="25sp" />  
  
        </LinearLayout>  
    </androidx.cardview.widget.CardView>  
</LinearLayout>
```

Librerías

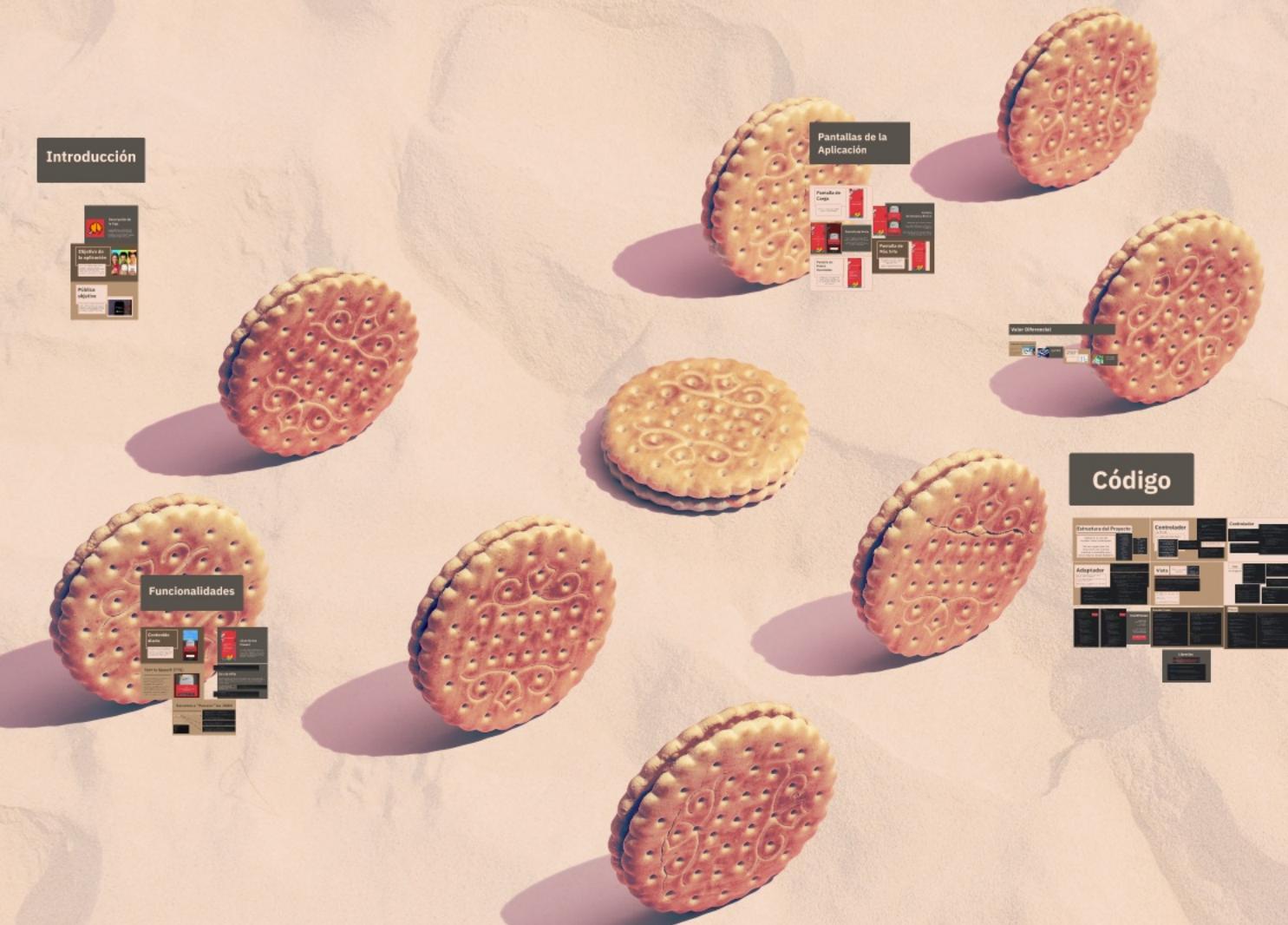


```
// APIs
implementation ("com.squareup.okhttp3:okhttp:4.9.3")

//Gif
implementation("com.github.bumptech.glide:glide:4.15.1")
implementation ("pl.droidsonroids.gif:android-gif-drawable:1.2.23")

//Gemini API
//    implementation("com.google.ai.client.generativeai:generativeai:0.9.0")
//    implementation("com.google.guava:guava:31.0.1-android")
//    implementation("org.reactivestreams:reactive-streams:1.0.4")
```

Introducción



Lucky Cookie

Tu aplicación ideal
para
Inspirarte y Aconsejarte

Valor Diferencial

Contenido dinámico y variado

Lucky Cloud ofrece contenido actualizado y diverso, con frases inspiradoras y consejos aleatorios.



El uso de tres APIs distintas permite obtener un constante de nuevas frases y bromas manteniendo la frescura y el interés del usuario.



Interactividad y personalización

Los usuarios pueden guardar y eliminar sus favoritos para tenerlos en el inicio.

La función de "Mejor Juego PTC" permite ver los resultados de los mejores jugadores y sus estadísticas.

La aplicación incluye una sección de "Mejores" que muestra videos sobre el desarrollo de las habilidades y la motivación.

La interfaz es muy intuitiva y fácil de usar.



Transparencia del proyecto

Los desarrolladores ofrecen una visión detallada de su trabajo.

La aplicación incluye una sección de "Mejores" que muestra videos sobre el desarrollo de las habilidades y la motivación.

La interfaz es muy intuitiva y fácil de usar.



Diseño intuitivo y accesibilidad

Lucky Cloud es un sistema pensado para facilitar la interacción y la accesibilidad a todos los usuarios.

La aplicación tiene un diseño sencillo y moderno.

La interfaz es muy intuitiva y fácil de usar.

Contenido dinámico y variado

Lucky Cookie ofrece contenido actualizado y diverso, con frases inspiradoras y consejos aleatorios.

El uso de tres APIs distintas permite la generación constante de nuevas frases y bromas, manteniendo la frescura y el interés del usuario.





Interactividad y personalización

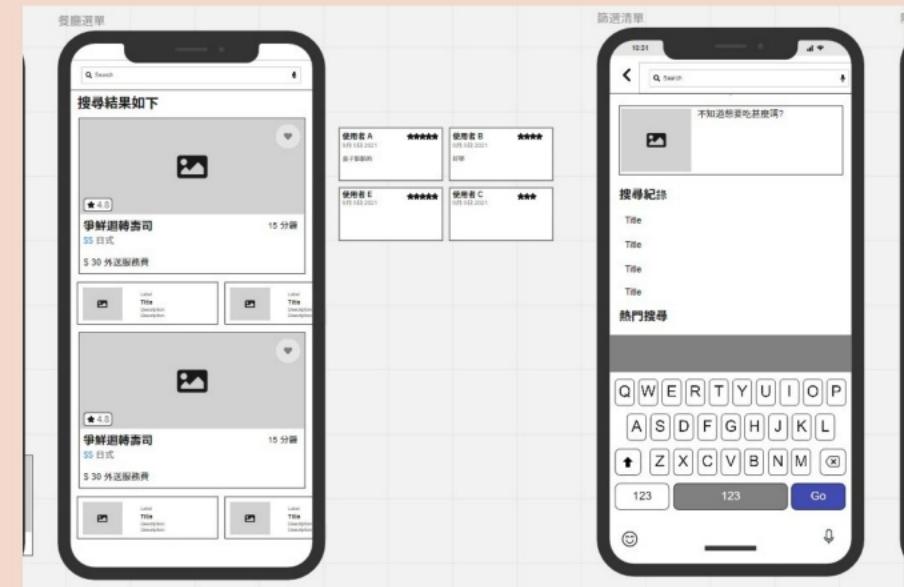
Los usuarios pueden guardar y eliminar su contenido favorito, lo que les permite personalizar su experiencia.

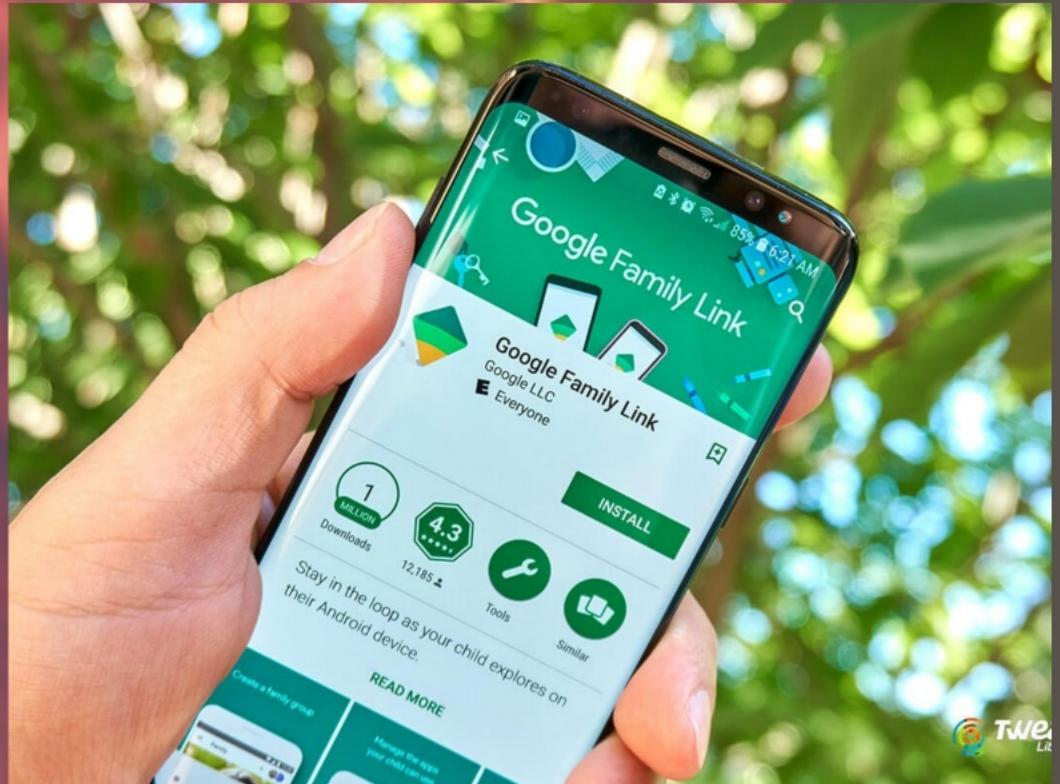
La función de Text to Speech (TTS) añade un nivel de interactividad, permitiendo a los usuarios escuchar consejos, aumentando así la accesibilidad.

Transparencia del proyecto

La aplicación incluye una pantalla de 'Más Info' que proporciona detalles sobre el proyecto, incluyendo el creador, las horas trabajadas y los costos asociados.

Esta transparencia genera confianza entre los usuarios y promueve una conexión más cercana con la aplicación.





Diseño intuitivo y accesibilidad

Lucky Cookie se caracteriza por su diseño intuitivo, facilitando la navegación a través de pantallas interconectadas.

La aplicación busca ser accesible, permitiendo que usuarios de diferentes edades y habilidades disfruten de su contenido sin complicaciones.

Introducción



Lucky Cookie

Tu aplicación ideal
para
Inspirarte y Aconsejarte