

LEARNIT

PROGRAMACION DE DISPOSITIVOS MOVILES



UAA
GERARDO CASTAÑEDA MARTIN
MICHELLE MONSERRAT GOMEZ LOPEZ

INTRODUCCION

El proyecto **LearnIt** se desarrolló como una herramienta educativa innovadora, diseñada para facilitar el aprendizaje del idioma inglés de manera interactiva y accesible desde dispositivos móviles. Esta aplicación combina lecciones estructuradas, juegos educativos y herramientas de navegación intuitivas, ofreciendo a los usuarios una experiencia de aprendizaje personalizada.

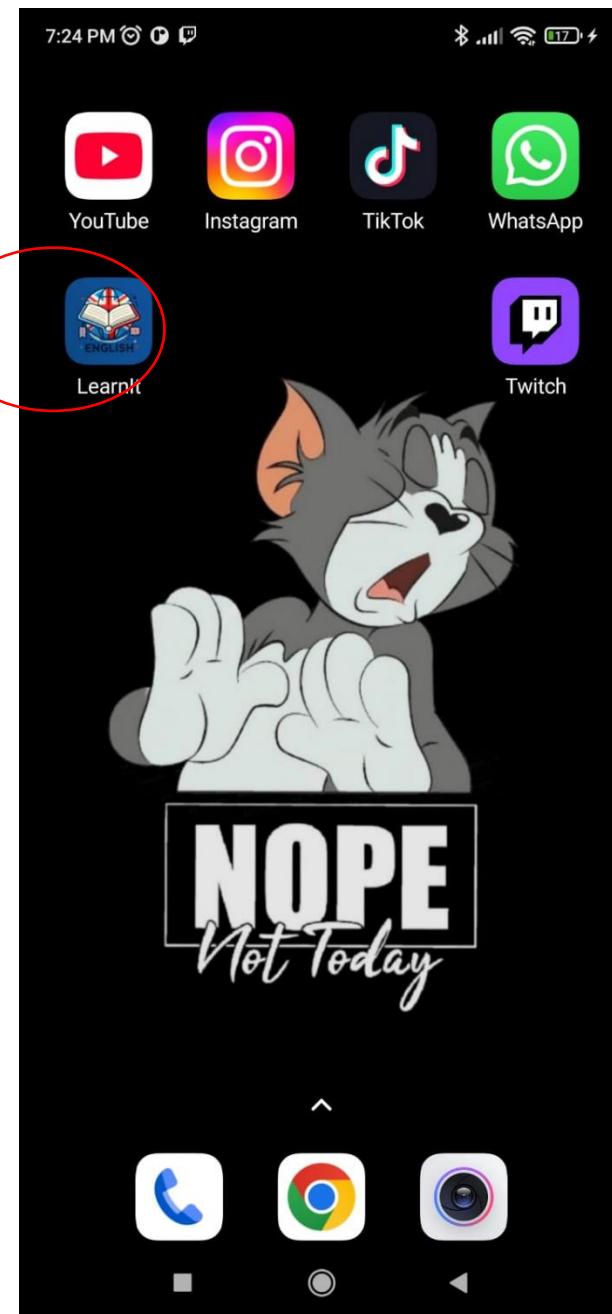
LearnIt se dirige a estudiantes de inglés en todos los niveles, integrando actividades como vocabulario, frases básicas, comprensión lectora y juegos de memoria, todos orientados a fomentar la práctica y la adquisición del idioma. Además, su diseño moderno y amigable asegura que los usuarios disfruten aprendiendo mientras mejoran sus habilidades lingüísticas.

OBJETIVOS

1. **Fomentar el aprendizaje del inglés:** Proveer un entorno educativo que combine actividades didácticas y juegos interactivos para reforzar habilidades lingüísticas básicas y avanzadas.
2. **Desarrollar competencias clave:** Enfatizar la práctica en vocabulario, frases comunes, comprensión lectora y pronunciación mediante ejercicios prácticos y atractivos.
3. **Ofrecer retroalimentación inmediata:** Incorporar sistemas de puntuación, barras de progreso y vidas visuales para motivar al usuario y mostrar su avance en tiempo real.
4. **Diseñar una experiencia accesible:** Garantizar que la aplicación sea fácil de usar y esté disponible para una amplia variedad de usuarios mediante una interfaz intuitiva y adaptable.
5. **Promover la gamificación:** Hacer el aprendizaje más dinámico y atractivo mediante juegos como crucigramas, memorama y cuestionarios interactivos.

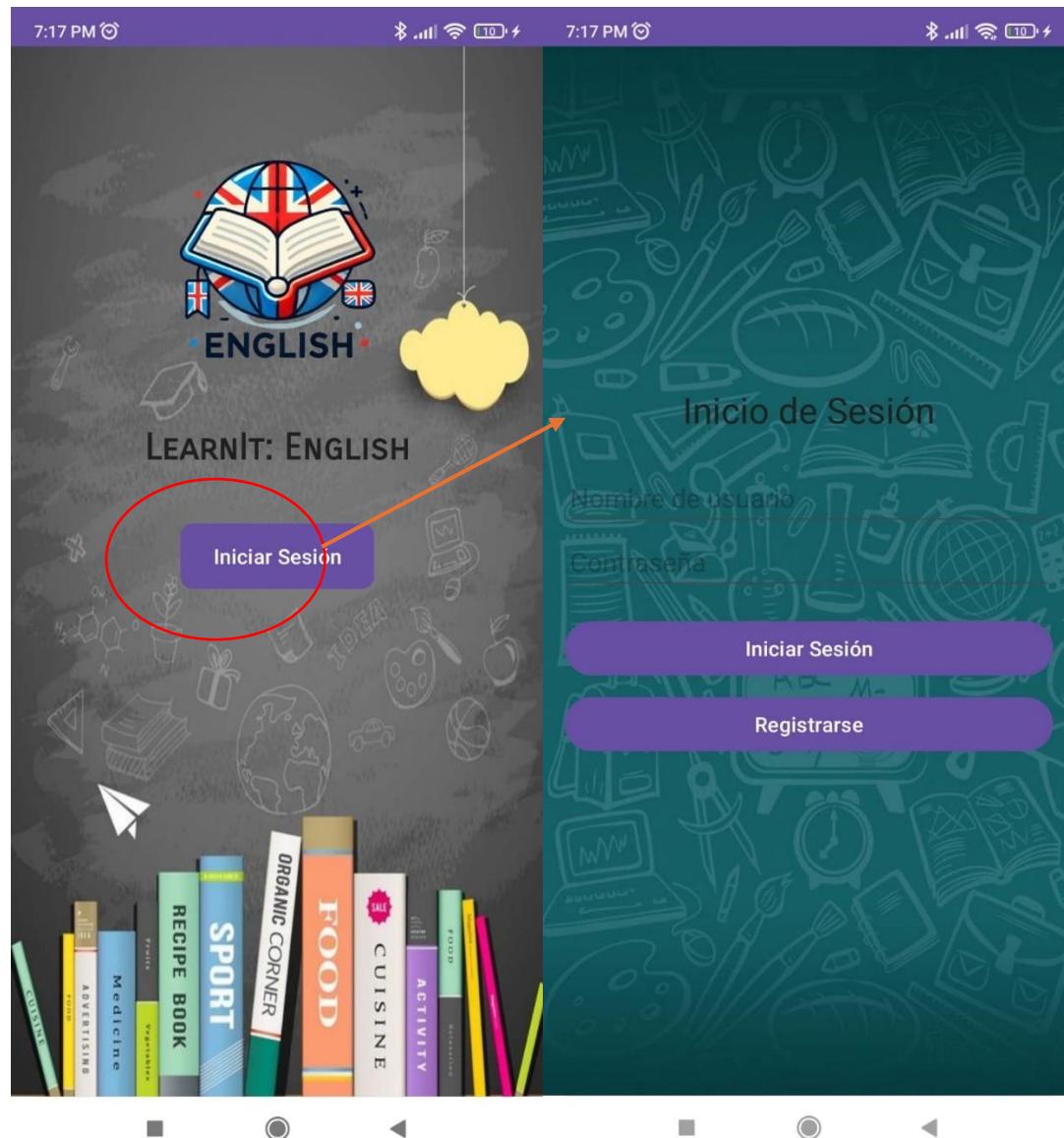
NAVEGACION

Una vez instalada nuestra aplicación de manera correcta nuestra aplicación se mostrará en su dispositivo de la siguiente manera:

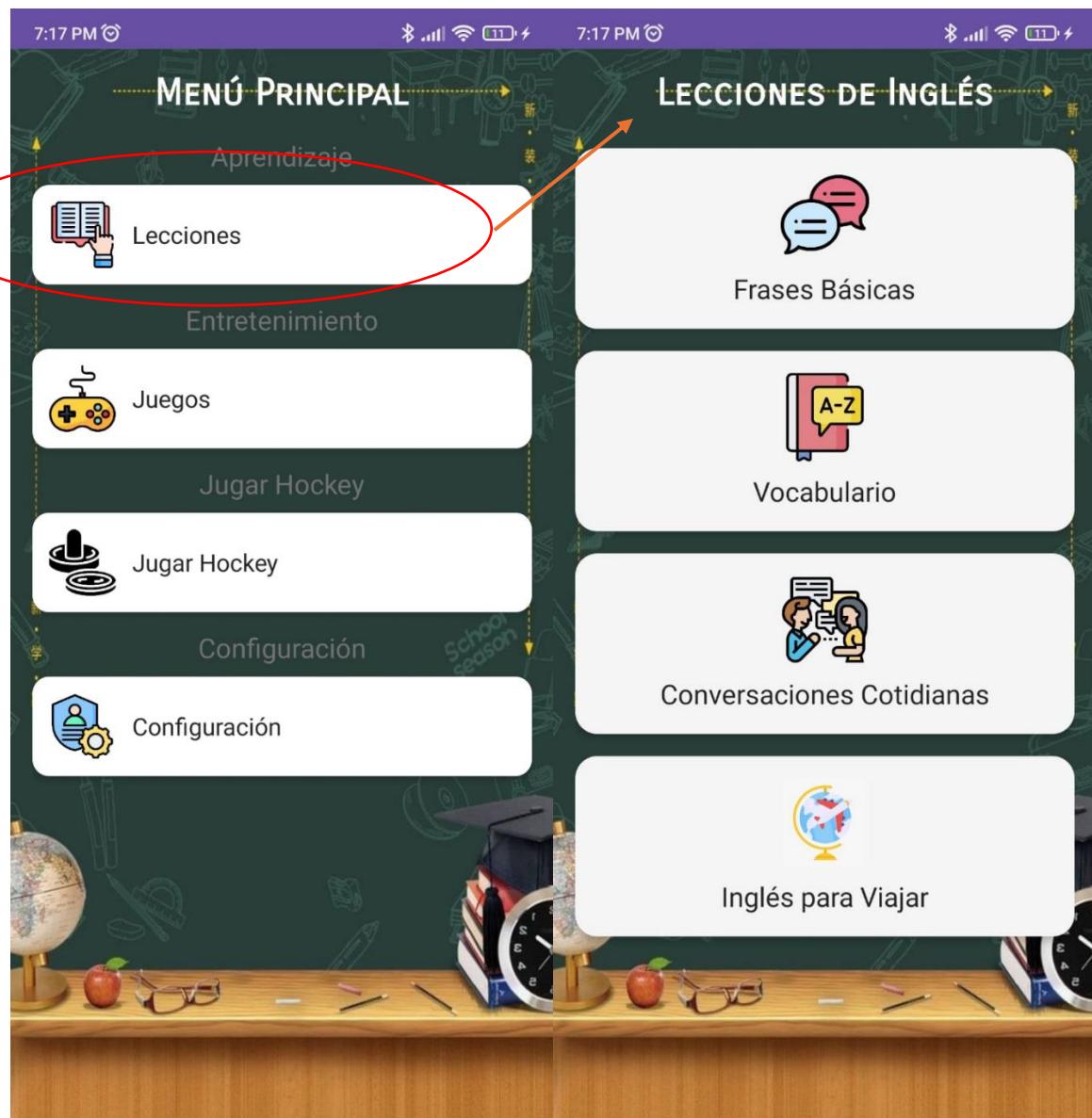


Una vez realizado esto ahora si pasaremos a mostrar con imágenes y indicaciones visuales de como es que se puede navegar en toda nuestra aplicación:

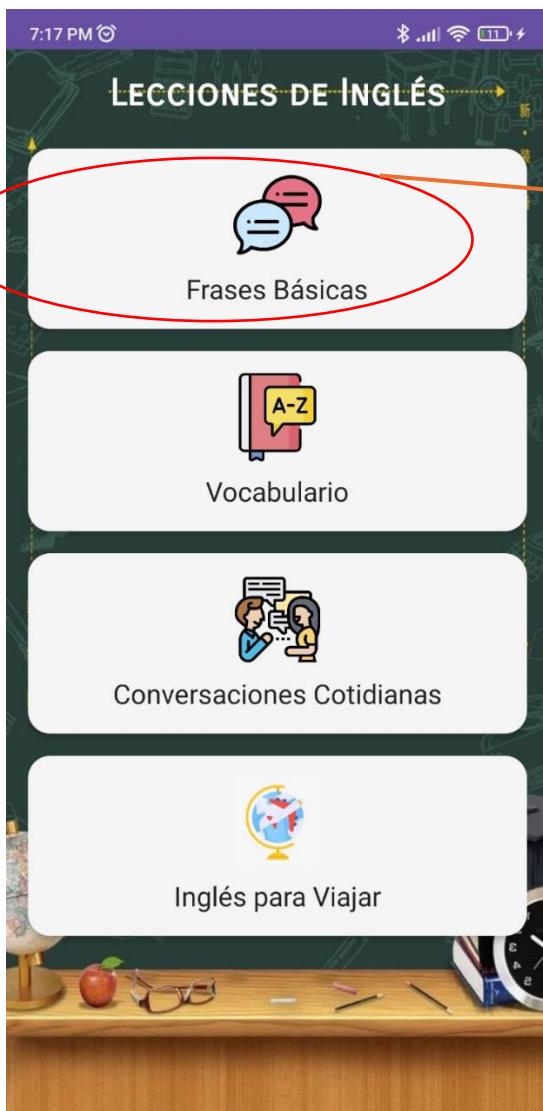
SECCION INICIO DE SESION:



SECCION MENU PRINCIPAL Y PASAMOS A SECCIONES DE INGLES:



SECCION DE FRASES BASICAS EN EL MENU DE LECCIONES DE INGLES:



The image shows a mobile application interface for English lessons. On the left, there's a menu titled "LECCIONES DE INGLÉS" with four options: "Frases Básicas" (highlighted with a red oval), "Vocabulario", "Conversaciones Cotidianas", and "Inglés para Viajar". Each option has an associated icon. On the right, a detailed view of the "Frases Básicas" section is shown. It starts with a header "Basic Phrases" and instructions: "Instructions: Answer the questions by selecting the correct option. You have 3 lives. Good luck!". Below this is a question: "How do you say 'Hola' in English?". Three buttons provide options: "Hello", "Goodbye", and "Please". Further down, there's a "Progress" bar, a "Lives" counter (showing three hearts), and a large orange "Next" button. The bottom of the screen features standard Android navigation buttons.

Basic Phrases

Instructions: Answer the questions by selecting the correct option. You have 3 lives. Good luck!

How do you say 'Hola' in English?

Hello

Goodbye

Please

Progress

Lives

Next

SECCION DE VOCABULARIO EN EL MENU DE LECCIONES DE INGLES:

7:17 PM ⌚

LECCIONES DE INGLÉS

Frases Básicas

Vocabulario

Conversaciones Cotidianas

Inglés para Viajar

7:17 PM ⌚

Vocabulary Practice

Select the correct category for each word shown below:

Current Word:
Taco

Progress:

Choose a category:

- Animals
- Fruits
- Food
- Clothes
- Transport

SECCION DE CONVERSACIONES COTIDIANAS EN EL MENU DE LECCIONES DE INGLES:

The image shows two screenshots of a mobile application for English lessons. The left screenshot displays a menu titled 'LECCIONES DE INGLÉS' with four options: 'Frases Básicas' (Basic Phrases), 'Vocabulario' (Vocabulary), 'Conversaciones Cotidianas' (Everyday Conversations), and 'Inglés para Viajar' (Travel English). A red oval highlights the 'Conversaciones Cotidianas' option, and a red arrow points from it to the right screenshot. The right screenshot shows a reading comprehension practice screen. It includes a welcome message: 'Welcome to Reading Comprehension Practice! Read the story carefully and answer the questions. Choose the best answer to improve your reading skills!', a green 'Score: 0' bar, and a story text: 'John was walking in the park when he saw a dog stuck in a bush. He decided to help the dog and took it to a shelter. The dog was adopted by a family the next day.' Below the story are three heart icons. At the bottom are four purple buttons with the following text: 'John ignored the dog.', 'John helped the dog and took it to a shelter.', 'John took the dog home.', and 'John called the police.'

7:17 PM

LECCIONES DE INGLÉS

Frases Básicas

Vocabulario

Conversaciones Cotidianas

Inglés para Viajar

Welcome to Reading Comprehension Practice!
Read the story carefully and answer the
questions.
Choose the best answer to improve your reading
skills!

Score: 0

John was walking in the park when he saw a dog stuck in a bush. He decided to help the dog and took it to a shelter. The dog was adopted by a family the next day.

John ignored the dog.

John helped the dog and took it to a shelter.

John took the dog home.

John called the police.

SECCION DE INGLES PARA VIAJAR EN EL MENU DE LECCIONES DE INGLES:

7:17 PM

LECCIONES DE INGLÉS

Frases Básicas

Vocabulario

Conversaciones Cotidianas

Inglés para Viajar

Match Spanish words with their English translations!

PLAYA	HOTEL
AEROPUERTO	TRAIN
TAXI	RESTAURANT
RESTAURANTE	MAP
MAPA	TAXI
HOTEL	BEACH
PASAPORTE	PASSPORT
BOLETO	LUGGAGE
TREN	TICKET
EQUIPAJE	AIRPORT

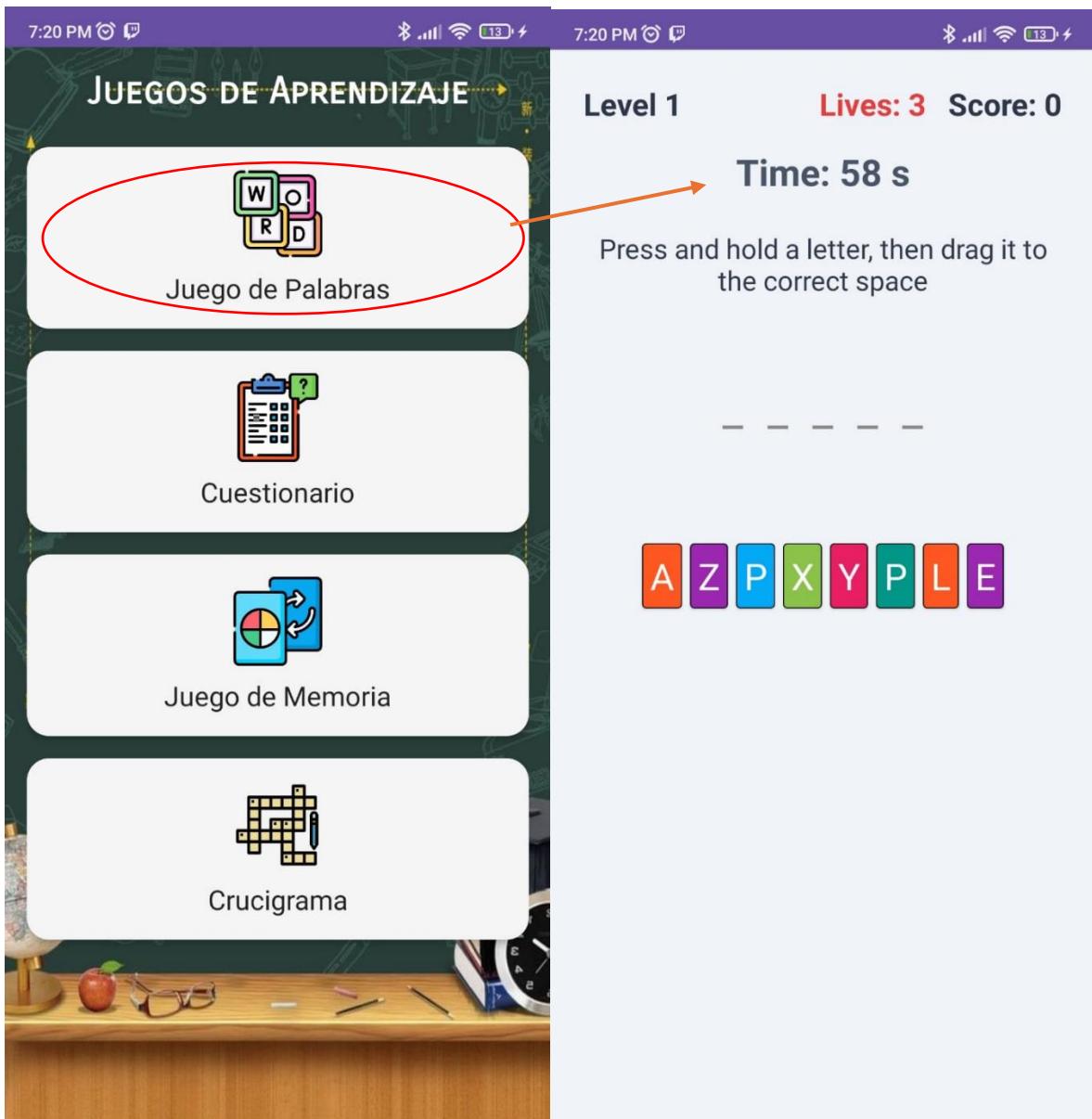
Score: 0

Time: 58

REGRESAMOS AL MENU PRINCIPAL Y VEMOS LA SECCION DE JUEGOS Y SU RESPECTIVO MENU:



SECCION DE JUEGOS DE PALABRAS EN EL MENU DE JUEGOS DE APRENDIZAJE:



SECCION DE CUESTIONARIO EN EL MENU DE JUEGOS DE APRENDIZAJE:

The image shows a mobile application interface for a learning game. At the top, there are two status bars: the left one shows '7:20 PM' and signal strength, while the right one shows '7:20 PM', signal strength, and battery level at 13%. Below the status bars is a title 'JUEGOS DE APRENDIZAJE'. The main content area displays four game options in cards:

- Juego de Palabras**: Represented by a icon of four colored squares (blue, green, yellow, pink) with letters W, O, R, D.
- Cuestionario**: Represented by a icon of a clipboard with a question mark.
- Juego de Memoria**: Represented by a icon of two overlapping blue cards.
- Crucigrama**: Represented by a icon of a crossword puzzle grid.

A red oval highlights the 'Cuestionario' card. An orange arrow points from this highlighted card to the word 'Computer' in the details section on the right. The details section includes:

Level: 1 Score: 0

Computer

An electronic device for storing and processing data

Press to Speak

Next

At the bottom of the screen, there are navigation icons: a square, a circle, a triangle, another square, another circle, and another triangle.

SECCION DE JUEGO DE MEMORIA EN EL MENU DE JUEGOS DE APRENDIZAJE:



SECCION DE CRUCIGRAMA EN EL MENU DE JUEGOS DE APRENDIZAJE:

7:20 PM ⚡ 🔋

JUEGOS DE APRENDIZAJE

WORD

Juego de Palabras

QUESTION

Cuestionario

MEMORY

Juego de Memoria

CRUCIGRAMA

Crucigrama

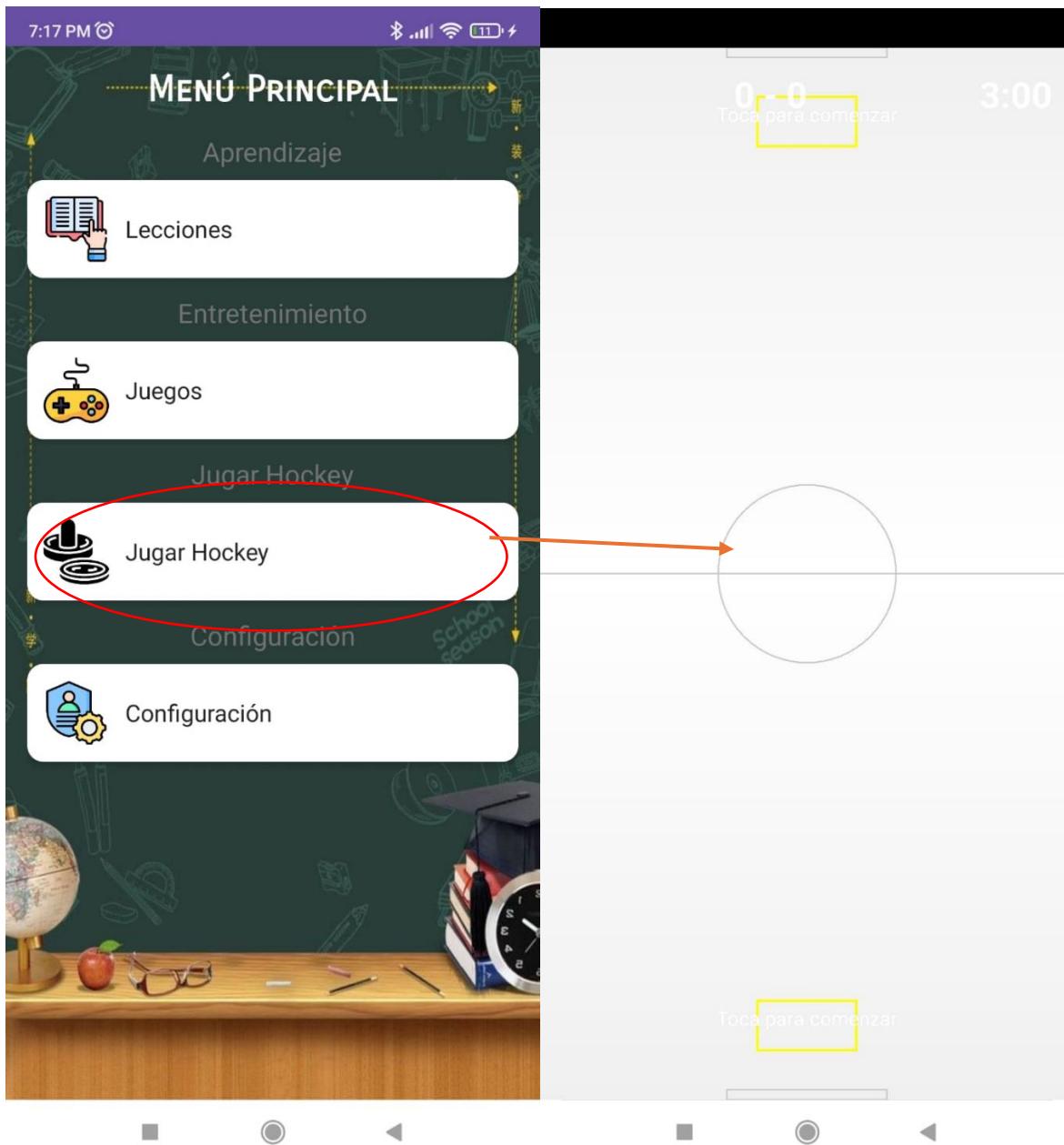
Nature Crossword

Complete the crossword puzzle by filling in nature-related words. Tap a cell to select it, then use your keyboard to type. All words are in English. Good luck!

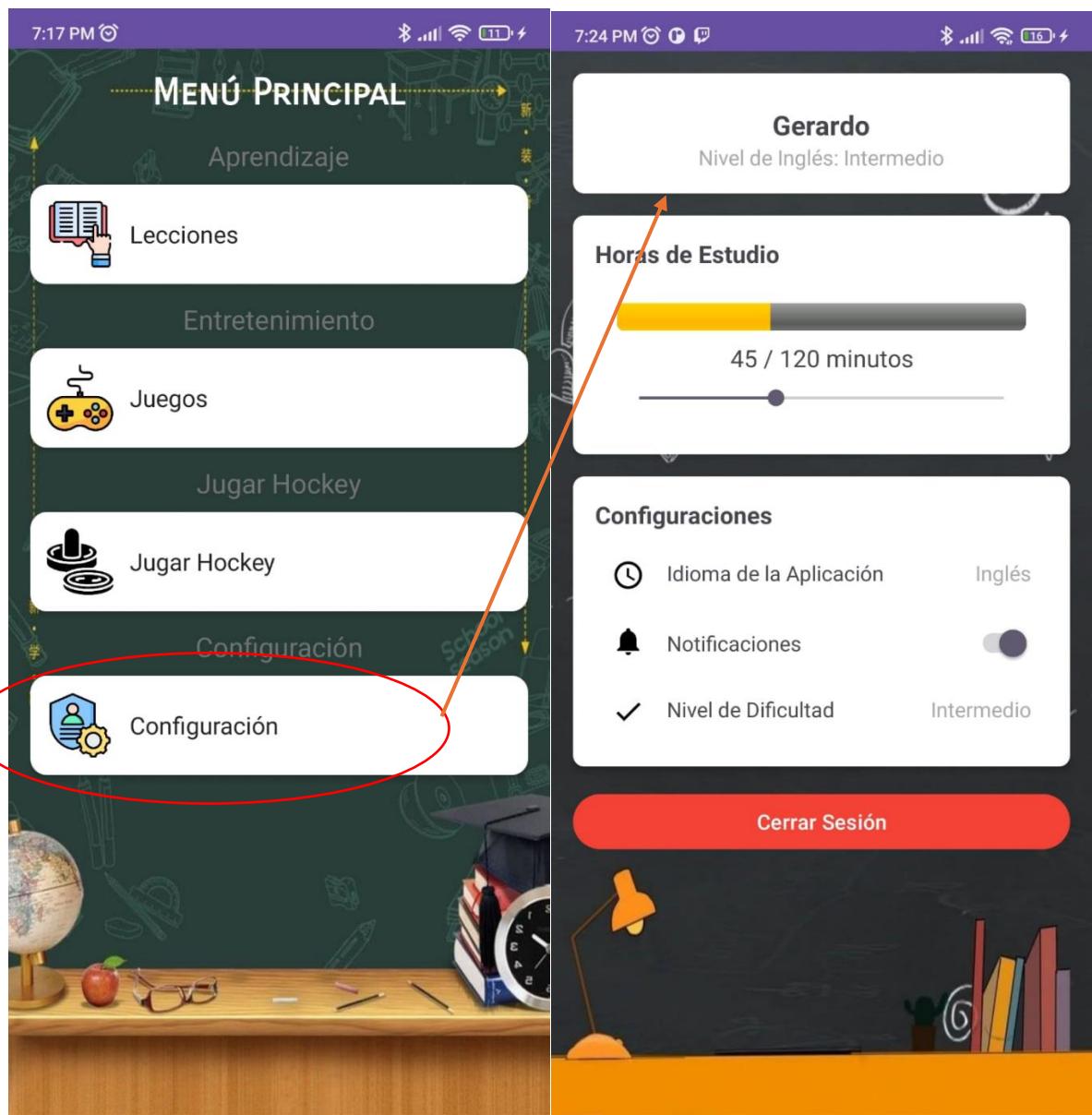
00:01 Score: 0

Clues

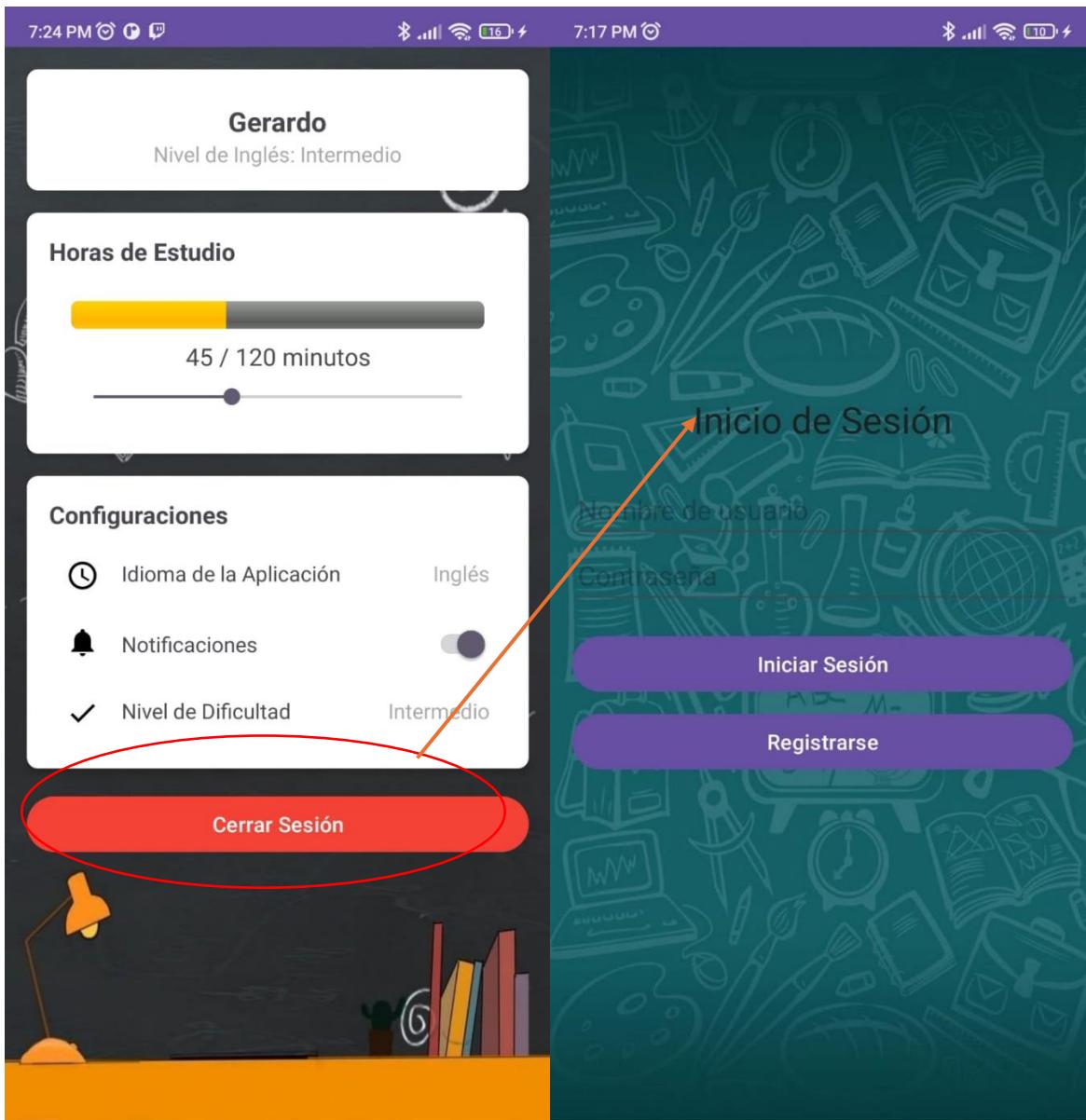
REGRESAMOS AL MENU PRINCIPAL Y VEMOS LA SECCION DE JUGAR HOCKEY:



REGRESAMOS AL MENU Y VEMOS LA SECCION DE CONFIGURACION:



CERRANDO SESIÓN VOLVEMOS A LA SECCION DE INICIO DE SESIÓN:



DESARROLLO

Pantalla Principal - LearnIt App

Descripción General

La pantalla principal de la aplicación LearnIt presenta el logotipo, el título de la aplicación y un botón para iniciar sesión. Está diseñada para ser una interfaz amigable que da la bienvenida al usuario y lo invita a interactuar con la aplicación, facilitando la navegación hacia la funcionalidad de inicio de sesión.

Características Principales

- Fondo personalizado con una imagen (@drawable/background_main).
- Logotipo centrado en la parte superior.
- Título de la aplicación en texto estilizado.
- Botón interactivo para navegar a la pantalla de inicio de sesión.

Diseño de la Interfaz

El diseño está basado en un RelativeLayout, que organiza los elementos de manera sencilla y centrada.

1. Logotipo:

- Un ImageView que muestra el logotipo centrado horizontalmente.
- Tamaño fijo de 200dp x 200dp.

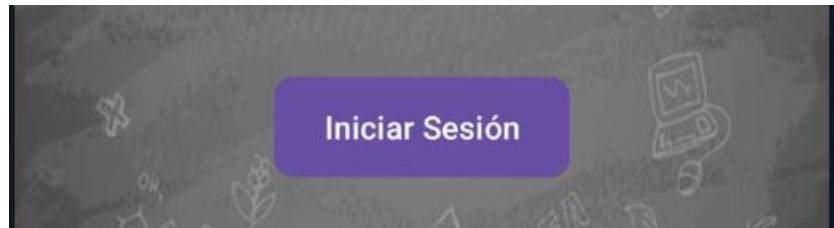
2. Título de la aplicación:

- Un TextView con el texto "LearnIt: English".
- Estilo en negrita y en mayúsculas pequeñas.



3. Botón de inicio de sesión:

- Un Button centrado bajo el título.
- Con fondo personalizado y texto blanco.



Lógica de la Actividad

La lógica es sencilla y se encarga de manejar la interacción con el botón "Iniciar sesión".

Pasos Principales

1. Configuración del layout con setContentView.
2. Vinculación del botón mediante su ID.
3. Configuración de un OnClickListener que redirige al usuario a la pantalla de inicio de sesión (LoginActivity).

```
MemoryGameActivity.java    activity_settings.xml    activity_main.xml    MainActivity.java    activity_menu.xml
1 package com.mmgl.pruebas;
2
3 > import ...
4
5 <></> public class MainActivity extends AppCompatActivity {
6
7     @Override
8     protected void onCreate(Bundle savedInstanceState) {
9         super.onCreate(savedInstanceState);
10        setContentView(R.layout.activity_main);
11
12        // Obtener referencia al botón de Iniciar Sesión
13        Button btnLogin = findViewById(R.id.btnLogin);
14
15        // Configurar el listener para el botón de Iniciar Sesión
16        btnLogin.setOnClickListener(new View.OnClickListener() {
17            @Override
18            public void onClick(View v) {
19                // Crear un Intent para ir a la pantalla de LoginActivity
20                Intent intent = new Intent(getApplicationContext(), LoginActivity.class);
21                startActivity(intent);
22            }
23        });
24    }
25}
```

The code shows the Java implementation for the MainActivity. It overrides the onCreate method to set the main layout. It then finds the button with id btnLogin and sets an OnClickListener. This listener's onClick method creates an Intent to start the LoginActivity.

Pantalla de Inicio de Sesión - LearnIt App

Descripción General

La pantalla de **Iniciar sesión** permite a los usuarios ingresar su nombre de usuario y contraseña para acceder a la aplicación. Si los datos son correctos, el usuario es redirigido a la pantalla principal de la aplicación. También ofrece un botón para registrarse si el usuario aún no tiene cuenta.

Características Principales

- **Campos de texto** para ingresar nombre de usuario y contraseña.
- **Botón de inicio de sesión** para validar los datos y acceder a la aplicación.
- **Validación** de que los campos no estén vacíos.
- **Uso de SharedPreferences** para guardar el nombre de usuario una vez que el inicio de sesión sea exitoso.
- **Notificación al usuario** mediante un Toast para informar si los datos de inicio de sesión son incorrectos o si hay un error.

Diseño de la Interfaz

La pantalla está compuesta por los siguientes elementos visuales:

1. Campos de texto:

- EditText para ingresar el nombre de usuario.
- EditText para ingresar la contraseña.

2. Botones:

- Button para iniciar sesión.
- Button para registrarse (si el usuario no tiene cuenta).

3. Fondo y estructura básica definida por el archivo XML correspondiente.



Lógica de la Pantalla de Inicio de Sesión

La lógica de la pantalla de inicio de sesión se encarga de verificar si los campos de nombre de usuario y contraseña son correctos, y redirigir al usuario a la actividad correspondiente en caso de un inicio de sesión exitoso. Si los campos están vacíos o los datos son incorrectos, se muestra un mensaje de error.

Flujo Principal:

1. Validación de Campos:

- Se verifican los campos de nombre de usuario y contraseña para asegurarse de que no estén vacíos.

2. Verificación de Usuario:

- Se utiliza un método de la base de datos (checkUser()) para validar si el usuario y la contraseña coinciden con los registros.

3. Guardado de Datos:

- Si el inicio de sesión es exitoso, se guarda el nombre de usuario en **SharedPreferences** para futuras sesiones.

4. Redirección:

- Si el inicio de sesión es exitoso, el usuario es redirigido a la siguiente actividad.

```
protected void onCreate(Bundle savedInstanceState) {
    btnLogin.setOnClickListener(new View.OnClickListener() {
        @Override
        public void onClick(View v) {
            String username = etUsername.getText().toString().trim();
            String password = etPassword.getText().toString().trim();

            // Validar que los campos no estén vacíos
            if (username.isEmpty() || password.isEmpty()) {
                Toast.makeText(context, "Por favor, ingresa nombre de usuario y contraseña", Toast.LENGTH_SHORT).show();
            } else {
                if (db.checkUser(username, password)) {
                    // Guardar username en SharedPreferences al iniciar sesión
                    SharedPreferences sharedPreferences = getSharedPreferences("LearnItPrefs", MODE_PRIVATE);
                    SharedPreferences.Editor editor = sharedPreferences.edit();
                    editor.putString("CURRENT_USERNAME", username);
                    editor.apply();

                    Toast.makeText(context, "Inicio de sesión exitoso", Toast.LENGTH_SHORT).show();
                    Intent intent = new Intent(getApplicationContext(), MenuActivity.class);
                    startActivity(intent);
                    finish();
                } else {
                    Toast.makeText(context, "Usuario o contraseña incorrectos", Toast.LENGTH_SHORT).show();
                }
            }
        }
    });
}
```

- **onCreate:** Inicializa los componentes de la interfaz de usuario.
- **btnLogin.setOnClickListener:** Verifica si los campos son válidos y si las credenciales coinciden con la base de datos.
- **SharedPreferences:** Guarda el nombre de usuario para la próxima sesión si el inicio es exitoso.
- **Intent:** Redirige al usuario a la actividad principal (HomeActivity) al iniciar sesión correctamente.

Menú principal (MainActivity)

El menú principal sirve como centro de navegación de la aplicación y ofrece cuatro secciones principales:

Secciones

1. Lecciones

- Propósito: Contenido estructurado para el aprendizaje del inglés
- Destino:LessonsActivity

2. Juegos (Juegos)

- Propósito: Aprendizaje interactivo a través de juegos educativos.
- Destino:GamesActivity

3. Hockey (Sección opcional)

- Propósito: Entretenimiento y desestrese para los niños
- Destino:HockeyActivity

4. Ajustes (Configuración)

- Finalidad: Configuración de la aplicación y preferencias del usuario
- Destino:SettingsActivity

Comportamiento de navegación

- Utiliza CardView elementos para un diseño de interfaz de usuario intuitivo y moderno.
- Implementa openActivity() un método genérico para una navegación optimizada.
- Evita volver a pantallas anteriores cuando se está en el menú principal
- Admite la funcionalidad del botón Atrás

Diseño de interfaz de usuario

Disposición del menú (activity_menu.xml)

- Se utiliza ScrollView para diseño responsive
- Imagen de fondo para atractivo estético.
- Tipografía y combinación de colores consistentes
- Secciones de menú categorizadas con iconos

Elementos visuales

- Título principal: "Menú Principal"
- Encabezados de sección con tipografía distintiva
- 48 iconos de DP para cada categoría del menú
- Diseño basado en tarjetas con elevación y estados en los que se puede hacer clic



Implementación técnica

Clases clave

- `MenuActivity`: Manejo del menú principal
- `LessonsActivity`: Módulo de lecciones
- `GamesActivity`: Módulo de juegos
- `HockeyActivity`: Funcionalidad adicional
- `SettingsActivity`: Configuración de la aplicación

```
public class MenuActivity extends AppCompatActivity {

    // Declaración de las CardViews
    private CardView cardLecciones, cardJuegos, cardProgreso, cardConfiguracion; 2 usages

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_menu);

        // Inicializar las CardViews
        cardLecciones = findViewById(R.id.cardLecciones);
        cardJuegos = findViewById(R.id.cardJuegos);
        cardProgreso = findViewById(R.id.cardHockey);
        cardConfiguracion = findViewById(R.id.cardConfiguracion);

        // Configurar eventos de clic para cada CardView
        cardLecciones.setOnClickListener(v -> openActivity(LessonsActivity.class));
        cardJuegos.setOnClickListener(v -> openActivity(GamesActivity.class));
        cardProgreso.setOnClickListener(v -> openActivity(HockeyActivity.class));
        cardConfiguracion.setOnClickListener(v -> openActivity(SettingsActivity.class));

        // Habilitar el botón de "Atrás" en la barra de acciones
        if (getSupportActionBar() != null) {
            getSupportActionBar().setDisplayHomeAsUpEnabled(true);
        }
    }
}
```

Módulo Lecciones

Descripción general

El módulo Lecciones es un componente fundamental de la aplicación LearnIt, que proporciona contenido de aprendizaje de inglés estructurado en cuatro categorías clave.

Diseño de Interfaz de Usuario

Estructura Visual

- Diseño basado en tarjetas (CardView)
- Iconografía representativa para cada categoría
- Paleta de colores consistente
- Tipografía clara y legible

Elementos Visuales

- Íconos de 64dp centrados
- Títulos en texto grande
- Tarjetas con elevación suave
- Fondo con imagen de menú de fondo



Implementación Técnica

Categorías de Lecciones

```
@Override
protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_lessons);

    // Referencias a los CardViews
    cardBasicPhrases = findViewById(R.id.cardBasicPhrases);
    cardVocabulary = findViewById(R.id.cardVocabulary);
    cardConversations = findViewById(R.id.cardConversations);
    cardTravel = findViewById(R.id.cardTravel);

    // Configurar los Intents para cada CardView
    cardBasicPhrases.setOnClickListener(new View.OnClickListener() {
        @Override
        public void onClick(View v) {
            // Abrir la actividad de frases básicas
            Intent intent = new Intent( mContext: LessonsActivity.this, BasicPhrasesActivity.class);
            startActivity(intent);
        }
    });

    cardVocabulary.setOnClickListener(new View.OnClickListener() {
        @Override
        public void onClick(View v) {
            // Abrir la actividad de vocabulario
            Intent intent = new Intent( mContext: LessonsActivity.this, VocabularyActivity.class);
            startActivity(intent);
        }
    });

    cardConversations.setOnClickListener(new View.OnClickListener() {
        @Override
        public void onClick(View v) {
            // Abrir el juego de lectura (ReadingGameActivity)
            Intent intent = new Intent( mContext: LessonsActivity.this, ReadingGameActivity.class);
            startActivity(intent);
        }
    });

    cardTravel.setOnClickListener(new View.OnClickListener() {
        .. .
    });
}
```

Actividad de Frases Básicas - Leranit App

Descripción General

La actividad de frases básicas dentro de la aplicación LearnIt está diseñada para ayudar a los usuarios a practicar frases comunes en inglés. En esta actividad, los usuarios deben elegir la opción correcta en un ejercicio de opción múltiple. El sistema de puntuación se ajusta en función de la respuesta del usuario, y el jugador tiene un número limitado de intentos (vidas) antes de que el ejercicio termine.

Características Principales

- Ejercicios de opción múltiple para evaluar el conocimiento de frases básicas.
- Sistema de puntuación: Los usuarios ganan puntos por respuestas correctas.
- Número de vidas limitado: Los jugadores tienen 3 vidas que se restan por cada respuesta incorrecta.
- Progreso visual: Un ProgressBar muestra el avance del jugador en el ejercicio.
- Guardado de progreso: Uso de SharedPreferences para almacenar el puntaje y las vidas entre sesiones.

Interfaz de Usuario

La interfaz está compuesta por varios elementos esenciales para interactuar con el ejercicio:

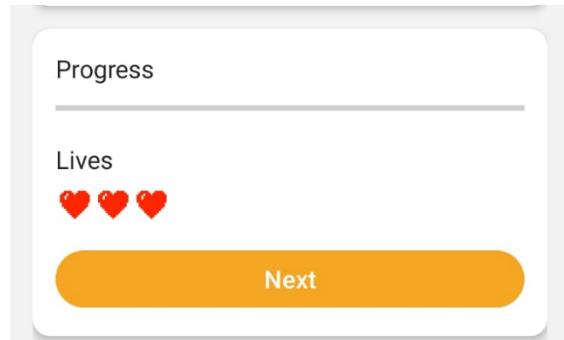
1. Texto de la pregunta: Un TextView que muestra la pregunta o frase que debe completar el jugador.

How do you say 'Hola' in English?

2. Botones de opción múltiple: Tres botones (Button) para que el jugador elija una respuesta.



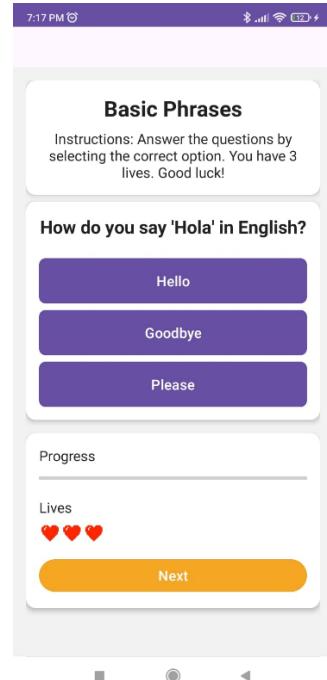
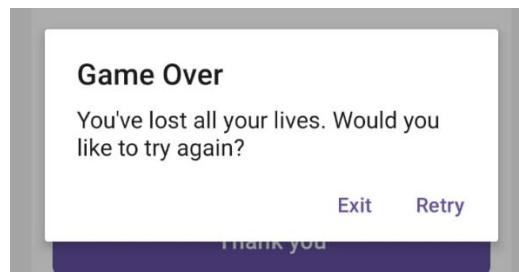
3. Botón para avanzar: Un Button que permite avanzar al siguiente ejercicio.



4. Progreso visual: Un ProgressBar para mostrar el progreso del jugador durante el ejercicio.



5. AlertDialog: Para notificar al jugador sobre el fin del ejercicio o el agotamiento de vidas.



Explicación de la Lógica:

1. **Inicialización:** Se configuran las vistas (TextView, Button, ProgressBar) y se crean los ejercicios.

2. **Carga de ejercicios:** Los ejercicios se cargan en una lista exercises y se muestran en pantalla.
3. **Avance al siguiente ejercicio:** El jugador puede avanzar al siguiente ejercicio usando el botón nextButton.
4. **Verificación de respuestas:** La lógica para verificar respuestas se debe añadir en un futuro, junto con el sistema de puntuación y penalizaciones.

```
@Override
protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_basic_phrases);

    // Inicializar vistas
    questionTextView = findViewById(R.id.questionTextView);
    option1Button = findViewById(R.id.option1Button);
    option2Button = findViewById(R.id.option2Button);
    option3Button = findViewById(R.id.option3Button);
    nextButton = findViewById(R.id.nextButton);
    progressBar = findViewById(R.id.progressBar);
    sharedPrefs = getSharedPreferences("LearnItPrefs", MODE_PRIVATE);

    // Cargar ejercicios y mostrar el primero
    loadExercises();
    displayCurrentExercise();

    // Configurar listener para avanzar al siguiente ejercicio
    nextButton.setOnClickListener(v -> goToNextExercise());

    // Configurar listeners para las opciones
    option1Button.setOnClickListener(v -> checkAnswer(selectedOption: 0));
    option2Button.setOnClickListener(v -> checkAnswer(selectedOption: 1));
    option3Button.setOnClickListener(v -> checkAnswer(selectedOption: 2));
}
```

```
private void loadExercises() { 1 usage
    exercises = new ArrayList<>();
    exercises.add(new BasicPhraseExercise( question: "How do you say 'Hola' in English?",
        new String[]{"Hello", "Goodbye", "Please"}, correctAnswerIndex: 0));
    exercises.add(new BasicPhraseExercise( question: "What does 'Thank you' mean in Spanish?",
        new String[]{"Gracias", "Por favor", "Adiós"}, correctAnswerIndex: 0));
    exercises.add(new BasicPhraseExercise( question: "Translate 'Good Morning' into Spanish.",
        new String[]{"Buenos Días", "Buenas Tardes", "Buenas Noches"}, correctAnswerIndex: 0));
    exercises.add(new BasicPhraseExercise( question: "What is the English word for 'Gracias'?",
        new String[]{"Please", "Sorry", "Thank you"}, correctAnswerIndex: 2));
    exercises.add(new BasicPhraseExercise( question: "How do you say 'Perdón' in English?",
        new String[]{"Please", "Sorry", "Excuse me"}, correctAnswerIndex: 1));
    exercises.add(new BasicPhraseExercise( question: "Translate 'Good Night' into Spanish.",
        new String[]{"Buenas Tardes", "Buenos Días", "Buenas Noches"}, correctAnswerIndex: 2));
    exercises.add(new BasicPhraseExercise( question: "What does 'I need help' mean in Spanish?",
        new String[]{"Necesito ayuda", "Hola", "Gracias"}, correctAnswerIndex: 0));
    exercises.add(new BasicPhraseExercise( question: "Translate 'Excuse me' into Spanish.",
        new String[]{"Por favor", "Perdón", "Disculpe"}, correctAnswerIndex: 2));
    exercises.add(new BasicPhraseExercise( question: "How do you say 'Adiós' in English?",
        new String[]{"Hello", "Goodbye", "Thank you"}, correctAnswerIndex: 1));
    exercises.add(new BasicPhraseExercise( question: "Translate 'I am sorry' into Spanish.",
        new String[]{"Estoy cansado", "Estoy perdido", "Lo siento"}, correctAnswerIndex: 2));
}

private void displayCurrentExercise() { 3 usages
    if (currentExerciseIndex < exercises.size()) {
        BasicPhraseExercise currentExercise = exercises.get(currentExerciseIndex);
        questionTextView.setText(currentExercise.getQuestion());
        option1Button.setText(currentExercise.getOptions()[0]);
        option2Button.setText(currentExercise.getOptions()[1]);
        option3Button.setText(currentExercise.getOptions()[2]);
        progressBar.setProgress(currentExerciseIndex * 100 / exercises.size());
        updateLivesDisplay(); // Llama al método para actualizar la visualización de vidas
    } else {
        showCompletionMessage();
    }
}
```

Actividad de Vocabulario - Laranit App

Descripción General

La **actividad de vocabulario** en la aplicación **LearnIt** está diseñada para ayudar a los usuarios a aprender y practicar vocabulario en inglés categorizado. En esta actividad, los usuarios deben identificar a qué categoría pertenece una palabra mostrada en la pantalla. La actividad incluye categorías como **Animales, Frutas, Comida, Ropa y Transporte**, y el progreso se mide mediante un **ProgressBar** que avanza con cada respuesta correcta.

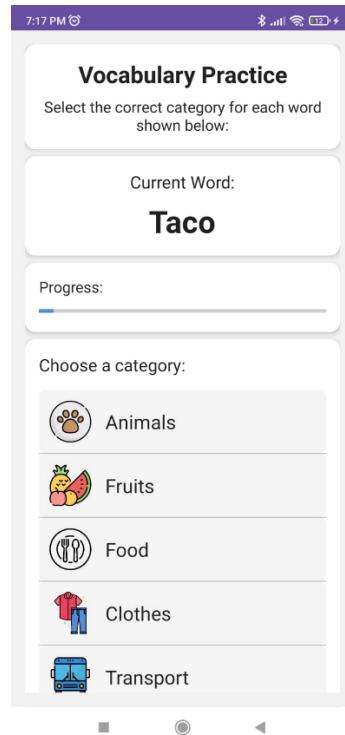
Características Principales

- **Categorías de vocabulario:** 5 categorías predefinidas (Animales, Frutas, Comida, Ropa, Transporte).
- **Lista de palabras aleatorias:** Cada vez que se muestra una nueva palabra, se selecciona aleatoriamente de una lista de palabras.
- **Interfaz visual interactiva:** Los usuarios deben seleccionar la categoría correcta para la palabra mostrada.
- **Progreso visual:** Un ProgressBar muestra el progreso del jugador durante la actividad.
- **Cálculo de respuestas correctas:** Se incrementa el contador de respuestas correctas a medida que el usuario clasifica correctamente las palabras.

Interfaz de Usuario

La interfaz está compuesta por los siguientes elementos:

1. **Texto de la palabra:** Un TextView que muestra la palabra que debe clasificarse.
2. **Layouts de categorías:** Cinco LinearLayouts para cada categoría (Animales, Frutas, Comida, etc.), donde el jugador arrastra la palabra.
3. **Progreso visual:** Un ProgressBar que se actualiza conforme el jugador avanza en el ejercicio.



Lógica de Implementación

1. Inicialización de Categorías y Palabras:

- Se crean un arreglo de categorías (categories[]) que corresponden a las categorías de vocabulario. Cada categoría tiene un layout en el XML.
- Se inicializa una lista de palabras que el jugador deberá clasificar en las categorías correctas.

2. Selección Aleatoria de Palabra:

- Cada vez que se muestra una palabra, se selecciona aleatoriamente de la lista.

3. Categorización de Palabras:

- Cuando se muestra una palabra, el jugador debe arrastrarla a la categoría correcta.

4. Progreso:

- Se lleva un registro de las respuestas correctas y se actualiza el ProgressBar conforme el jugador avanza.

```
// Orden de categorias exactamente igual al layout XML
private String[] categories = { 3 usages
    "Animals", // categoryLayout1
    "Fruits", // categoryLayout2
    "Food", // categoryLayout3
    "Clothes", // categoryLayout4
    "Transport" // categoryLayout5
};
```

```
private void initializeWordCategories() { 1 usage
    wordToCategory = new HashMap<>();

    // Animals (categoryLayout1)
    addWordsToCategory( category: "Animals", Arrays.asList("Dog", "Cat"));

    // Fruits (categoryLayout2)
    addWordsToCategory( category: "Fruits", Arrays.asList("Apple", "Banana", "Strawberry"));

    // Food (categoryLayout3)
    addWordsToCategory( category: "Food", Arrays.asList("Pizza", "Taco", "Salad", "Soup", "Bread", "Hamburger"));

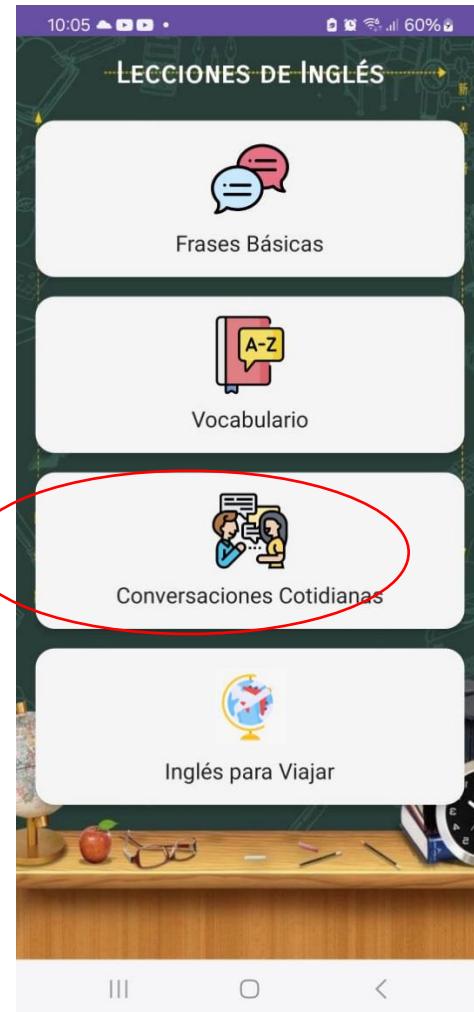
    // Clothes (categoryLayout4)
    addWordsToCategory( category: "Clothes", Arrays.asList("Shirt", "Pants", "Shoes"));

    // Transport (categoryLayout5)
    addWordsToCategory( category: "Transport", Arrays.asList("Car", "Bicycle", "Airplane"));
}
```

ReadingGameActivity.java

Descripción General

La clase ReadingGameActivity es la actividad principal del juego. Permite a los usuarios leer historias, responder preguntas y visualizar el progreso mediante un sistema de puntuación y un contador de vidas representado por corazones.



Componentes Principales

1. TextViews:

- `storyText`: Muestra el texto de la historia actual.
- `scoreText`: Indica la puntuación acumulada del jugador.

2. Botones de Respuesta:

- Cuatro botones (`responseButtons`) que presentan las posibles respuestas a las preguntas.

3. Sistema de Vidas:

- life1, life2, life3: Tres ImageView que representan las vidas del jugador como corazones. Desaparecen cuando el jugador pierde una vida.

4. Controladores de Juego:

- StoryViewModel: Clase que organiza las historias y respuestas correctas.
- Variables internas como currentStoryIndex, totalScore y lifeCount controlan el estado del juego.

Lógica Principal

1. Inicialización de Componentes

```
protected void onCreate(Bundle savedInstanceState) {...}

private void loadStory() { 3 usages
    if (currentStoryIndex < storyViewModel.getStories().size()) {
        Story currentStory = storyViewModel.getStories().get(currentStoryIndex);

        storyText.setText(currentStory.getStory());
        String[] responses = currentStory.getPossibleResponses();
        for (int i = 0; i < responseButtons.length; i++) {
            responseButtons[i].setText(responses[i]);
        }

        // Actualizar puntuación
        scoreText.setText("Score: " + totalScore);
    }
}

private void checkAnswer(int selectedResponseIndex) { 1 usage
    Story currentStory = storyViewModel.getStories().get(currentStoryIndex);

    if (selectedResponseIndex == currentStory.getCorrectResponseIndex()) {
        totalScore += 10;
    } else {
        totalScore -= 5; // Restar puntos si la respuesta es incorrecta
        reduceLife(); // Reducir una vida si la respuesta es incorrecta
    }

    currentStoryIndex++;

    if (currentStoryIndex < storyViewModel.getStories().size()) {
        loadStory();
    } else {
        showFinalScore();
    }
}
```

- **Función:** Inicializa los componentes visuales y la lógica del juego. Configura los botones para responder preguntas.
- **Relevancia:** Permite que el usuario interactúe con las historias y vea su progreso.

2. Mostrar Historias

```
private void loadStory() { 3 usages
    if (currentStoryIndex < storyViewModel.getStories().size()) {
        Story currentStory = storyViewModel.getStories().get(currentStoryIndex);

        storyText.setText(currentStory.getStory());
        String[] responses = currentStory.getPossibleResponses();
        for (int i = 0; i < responseButtons.length; i++) {
            responseButtons[i].setText(responses[i]);
        }

        // Actualizar puntuación
        scoreText.setText("Score: " + totalScore);
    }
}
```

- **Función:** Muestra el texto de la historia actual y las opciones de respuesta en los botones.
- **Relevancia:** Proporciona el contenido principal del juego.

3. Verificar Respuestas

```
private void checkAnswer(int selectedResponseIndex) { 1 usage
    Story currentStory = storyViewModel.getStories().get(currentStoryIndex);

    if (selectedResponseIndex == currentStory.getCorrectResponseIndex()) {
        totalScore += 10;
    } else {
        totalScore -= 5; // Restar puntos si la respuesta es incorrecta
        reduceLife(); // Reducir una vida si la respuesta es incorrecta
    }

    currentStoryIndex++;

    if (currentStoryIndex < storyViewModel.getStories().size()) {
        loadStory();
    } else {
        showFinalScore();
    }
}
```

- **Función:** Evalúa si la respuesta seleccionada es correcta. Actualiza la puntuación y el estado de las vidas.
- **Relevancia:** Determina el progreso del juego y si el jugador puede continuar.

4. Reducir Vidas

```

private void reduceLife() { 1 usage
    if (lifeCount > 0) {
        lifeCount--;
        // Ocultar los corazones (vidas) conforme se pierden
        if (lifeCount == 2) {
            life3.setVisibility(View.INVISIBLE); // Eliminar el tercer corazón
        } else if (lifeCount == 1) {
            life2.setVisibility(View.INVISIBLE); // Eliminar el segundo corazón
        } else if (lifeCount == 0) {
            life1.setVisibility(View.INVISIBLE); // Eliminar el primer corazón
        }
    }

    if (lifeCount == 0) {
        showFinalScore();
    }
}

```

- **Función:** Gestiona la visualización de las vidas restantes. Cada vez que el jugador falla, se oculta un corazón.
- **Relevancia:** Proporciona retroalimentación visual inmediata al jugador sobre su estado.

5. Mostrar Resultados Finales

```

private void showFinalScore() { 2 usages
    // Mostrar el puntaje final cuando se acaban las vidas
    new android.app.AlertDialog.Builder( context: this)
        .setTitle("Game Finished!")
        .setMessage("Your total score: " + totalScore + " points\n" +
                    "Great job practicing your reading skills!")
        .setPositiveButton( text: "Play Again", (dialog, which) -> {
            currentStoryIndex = 0;
            totalScore = 0;
            lifeCount = 3; // Restaurar vidas
            life1.setVisibility(View.VISIBLE);
            life2.setVisibility(View.VISIBLE);
            life3.setVisibility(View.VISIBLE);
            loadStory();
        })
        .setNegativeButton( text: "Exit", (dialog, which) -> finish())
        .show();
}

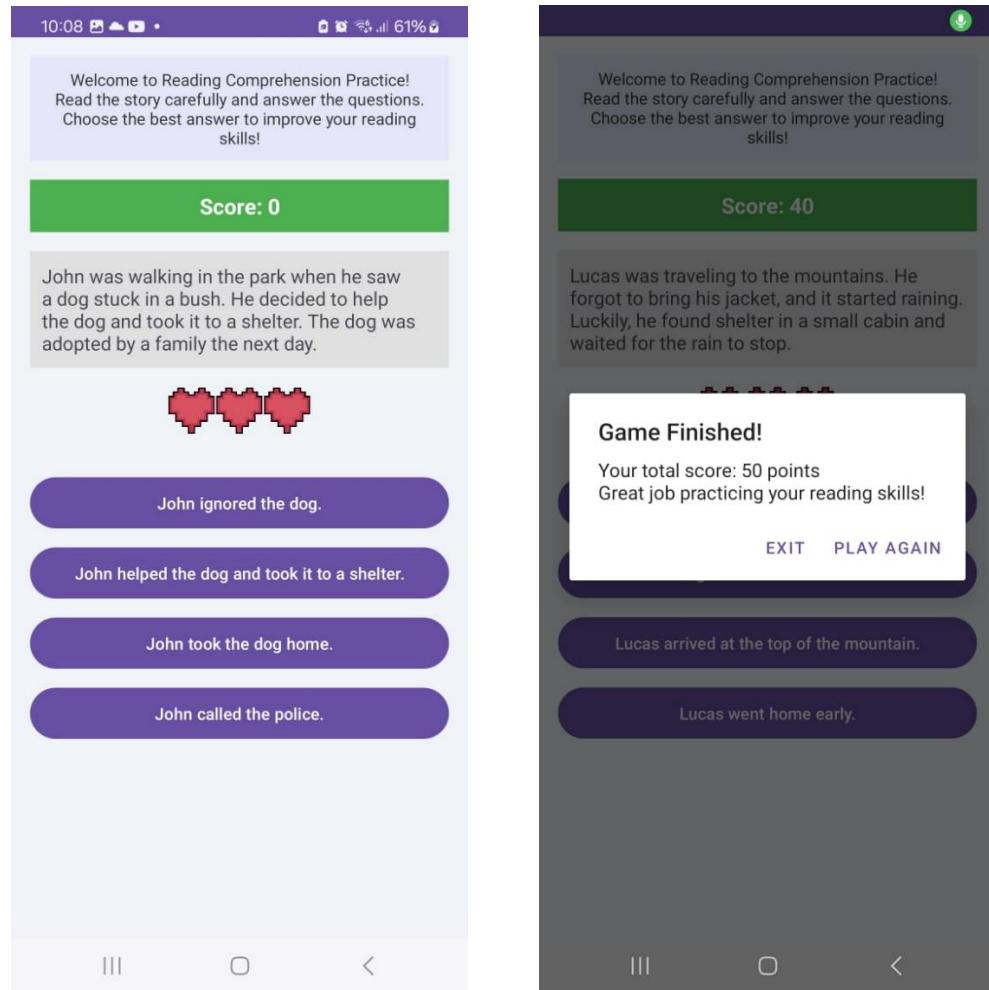
```

- **Función:** Muestra un cuadro de diálogo cuando el jugador pierde todas las vidas o completa el juego.
- **Relevancia:** Cierra la experiencia de juego con una retroalimentación clara.

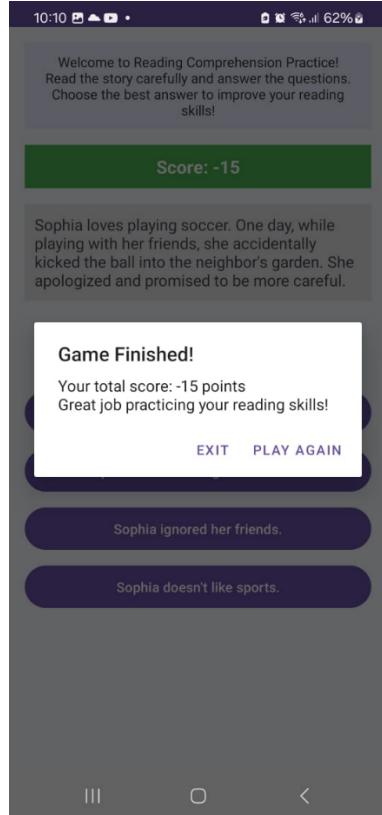
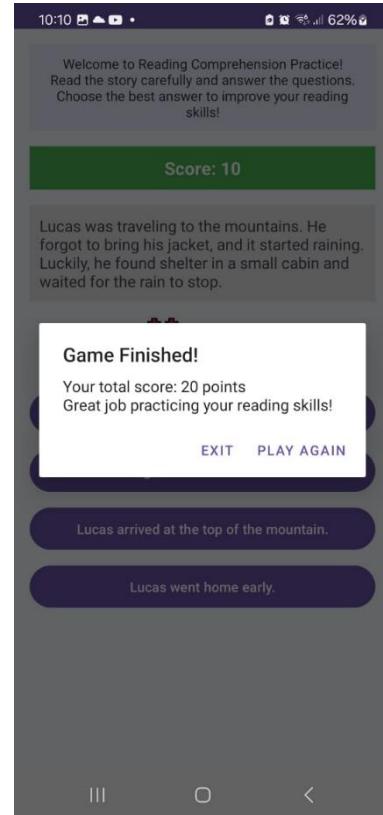
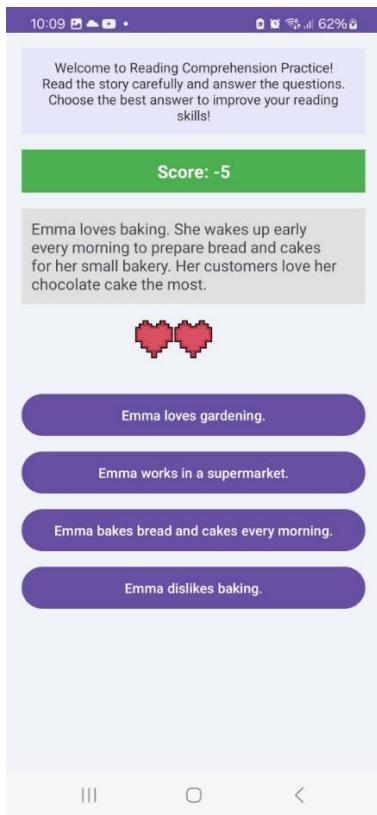
A continuación, se mostrará el funcionamiento del juego:

Tenemos principalmente la vista de nuestro juego y enseñaremos como es que se llega al final de este resolviendo todo de la manera correcta, que sucede si te equivocas en algunas y que pasa si respondes todo de manera errónea. Esta actividad es de comprensión lectora, contamos breves historias y luego se hacen una serie de preguntas para saber si el usuario las comprendió.

Así es como se ve la pantalla al introducirnos en el juego y al completarlo satisfactoriamente.



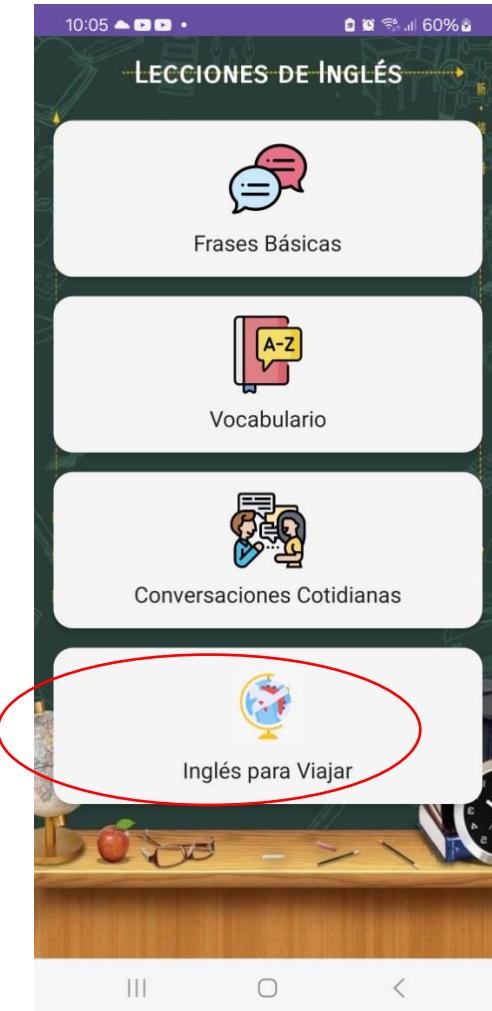
Después mostramos como es que al contestar correctamente te aumenta 10 puntos, pero si se equivoca se le restan -5 puntos y como al final solo logro obtener 20 puntos, en cambio en la última opción obtuvo muy pocas respuestas correctas, ya que el puntaje resultó negativo.



PhraseGuessActivity.java

Descripción General

La actividad PhraseGuessActivity es la base del juego de coincidencia de palabras. Los jugadores deben emparejar palabras en español con sus traducciones al inglés. Incluye un sistema de puntuación, un temporizador y una barra de progreso para rastrear el avance.



Componentes Principales

1. TextViews:

- gameIntroText: Muestra el mensaje introductorio del juego.
- scoreText: Indica la puntuación acumulada.
- timerText: Muestra el tiempo restante.

2. Containers:

- spanishContainer: Contiene botones con palabras en español.
- englishContainer: Contiene botones con palabras en inglés.

3. ProgressView:

- Una barra de progreso personalizada que se incrementa a medida que el jugador empareja correctamente.

4. Controladores:

- phrasesList: Lista de pares de palabras (español-inglés) a emparejar.
- Métodos como initializePhrases() y loadQuestion() gestionan el contenido.

Lógica Principal

1. Inicialización

```
@Override  
protected void onCreate(Bundle savedInstanceState) {  
    super.onCreate(savedInstanceState);  
    setContentView(R.layout.activity_phrase_guess);  
  
    initializeViews();  
    initializePhrases();  
    startGame();  
}
```

- **Función:** Configura la actividad inicializando vistas, palabras y llamando a startGame() para iniciar el juego.
- **Relevancia:** Configura el juego y asegura que todo esté listo para la interacción del usuario.

2. Configuración de Vistas

```
private void initializeViews() {  
    gameIntroText = findViewById(R.id.gameIntroText);  
    scoreText = findViewById(R.id.scoreText);  
    timerText = findViewById(R.id.timerText);  
    spanishContainer = findViewById(R.id.spanishContainer);  
    englishContainer = findViewById(R.id.englishContainer);  
    progressView = findViewById(R.id.progressView);  
}
```

- **Función:** Encuentra y asigna vistas desde el layout XML a variables de la actividad.
- **Relevancia:** Permite la manipulación de los elementos visuales.

3. Inicialización de Palabras

```
private void initializePhrases() { 1 usage
    phrasesList = new ArrayList<>();
    phrasesList.add(new Phrase( spanish: "aeropuerto", english: "airport"));
    phrasesList.add(new Phrase( spanish: "pasaporte", english: "passport"));
    phrasesList.add(new Phrase( spanish: "hotel", english: "hotel"));
    phrasesList.add(new Phrase( spanish: "restaurante", english: "restaurant"));
    phrasesList.add(new Phrase( spanish: "equipaje", english: "luggage"));
    phrasesList.add(new Phrase( spanish: "taxi", english: "taxi"));
    phrasesList.add(new Phrase( spanish: "tren", english: "train"));
    phrasesList.add(new Phrase( spanish: "boleto", english: "ticket"));
    phrasesList.add(new Phrase( spanish: "mapa", english: "map"));
    phrasesList.add(new Phrase( spanish: "playa", english: "beach"));
}
```

- **Función:** Define las palabras en español e inglés que se usarán en el juego.
- **Relevancia:** Proporciona el contenido necesario para jugar.

4. Lógica del Juego

4.1. Inicio del Juego

```
private void startGame() { 2 usages
    score = 0;
    scoreText.setText("Score: 0");
    progressView.setProgress(0);
    progressView.setEnabled(true);
    startGameTimer();
    loadQuestion();
}
```

- **Función:** Resetea los valores iniciales, inicia el temporizador y carga las preguntas.
- **Relevancia:** Permite que el jugador comience desde el inicio con condiciones limpias.

4.2. Cargar Preguntas

```

private void loadQuestion() { 1 usage
    spanishContainer.removeAllViews();
    englishContainer.removeAllViews();

    Collections.shuffle(phrasesList);

    List<Button> spanishButtons = new ArrayList<>();
    List<Button> englishButtons = new ArrayList<>();

    for (Phrase phrase : phrasesList) {
        spanishButtons.add(createButton(phrase.getSpanish(), isSpanish: true));
        englishButtons.add(createButton(phrase.getEnglish(), isSpanish: false));
    }

    Collections.shuffle(spanishButtons);
    Collections.shuffle(englishButtons);

    for (Button spanishButton : spanishButtons) {
        spanishContainer.addView(spanishButton);
    }
    for (Button englishButton : englishButtons) {
        englishContainer.addView(englishButton);
    }
}

```

- **Función:** Genera botones con palabras y los mezcla aleatoriamente en los contenedores.
- **Relevancia:** Mantiene el juego dinámico y desafiante.

4.3. Crear Botones

```

private Button createButton(String text, boolean isSpanish) { 2 usages
    Button button = new Button(context: this);
    button.setText(text);
    button.setLayoutParams(new LinearLayout.LayoutParams(
        LinearLayout.LayoutParams.MATCH_PARENT,
        LinearLayout.LayoutParams.WRAP_CONTENT
    ));

    button.setOnClickListener(v -> handleButtonClick((Button) v, isSpanish));
    return button;
}

```

- **Función:** Crea botones para cada palabra con texto y eventos de clic.
- **Relevancia:** Permite al jugador interactuar con las palabras.

4.4. Manejo de Clicks

```
private void handleButtonClick(Button button, boolean isSpanish) { 1 usage
    if (isSpanish) {
        if (selectedSpanishButton != null) {
            selectedSpanishButton.setBackgroundColor(0);
        }
        selectedSpanishButton = button;
    } else {
        if (selectedEnglishButton != null) {
            selectedEnglishButton.setBackgroundColor(0);
        }
        selectedEnglishButton = button;
    }

    button.setBackgroundColor(0xFFFFFFFF);

    if (selectedSpanishButton != null && selectedEnglishButton != null) {
        checkMatch();
    }
}
```

- **Función:** Cambia el estado de los botones seleccionados y verifica si las palabras coinciden.
- **Relevancia:** Es la base de la interacción del jugador.

4.5. Verificar Coincidencias

```
private void checkMatch() { 1 usage
    boolean isCorrect = phrasesList.stream()
        .anyMatch(p -> p.getSpanish().equals(selectedSpanishButton.getText()) &&
                  p.getEnglish().equals(selectedEnglishButton.getText()));

    if (isCorrect) {
        selectedSpanishButton.setVisibility(View.INVISIBLE);
        selectedEnglishButton.setVisibility(View.INVISIBLE);
        score += 10;
        scoreText.setText("Score: " + score);
        progressView.incrementProgress( value: 10);

        if (checkIfGameCompleted()) {
            endGame("Complete all the words! Great job!");
        }
    } else {
        selectedSpanishButton.setBackgroundColor(0xFFFFCCCC);
        selectedEnglishButton.setBackgroundColor(0xFFFFCCCC);
    }

    resetSelections();
}
```

- **Función:** Evalúa si las palabras seleccionadas coinciden.
- **Relevancia:** Determina el avance en el juego

5. Temporizador

```

private void startGameTimer() { 1 usage
    gameTimer = new CountDownTimer( millisInFuture: 60000, countDownInterval: 1000 ) {
        @Override no usages
        public void onTick(long millisUntilFinished) {
            timerText.setText("Time: " + millisUntilFinished / 1000);
        }

        @Override no usages
        public void onFinish() { endGame("Time's up! Try again."); }
    }.start();
}

```

- **Función:** Disminuye el tiempo restante y termina el juego si llega a cero.
- **Relevancia:** Introduce un elemento de presión y desafío.

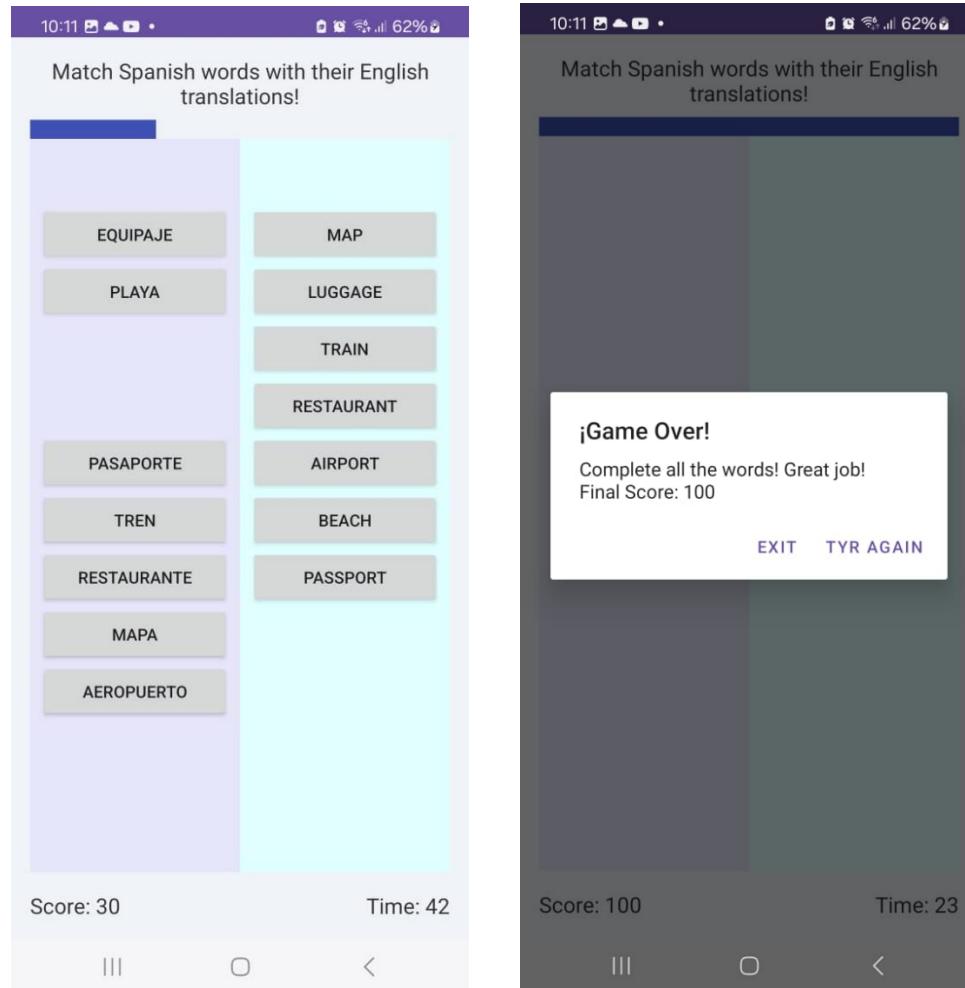
Ahora mostraremos el funcionamiento del juego:

Aquí tenemos principalmente la vista del juego el cual trata de dos columnas que serán para relacionar palabras en español y su traducción al inglés basándose en el contexto de viaje y palabras simples. En esta se tiene un temporizador de 1 minuto al igual que una barra de progreso que avanza conforme se relacionen correctamente las palabras.

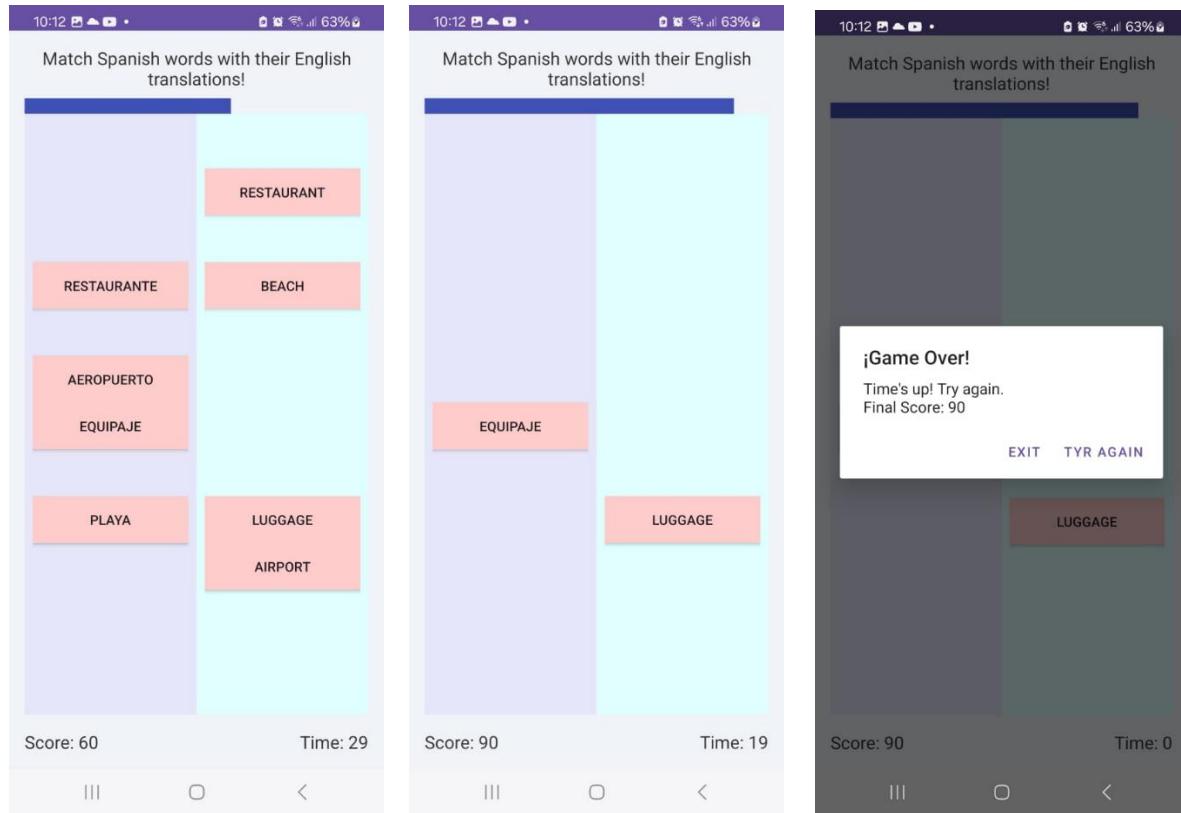
Así se ve nuestro juego al ingresar.



Ahora podemos ver la barra de progreso como avanza y también como las palabras se esfuman al relacionarlas correctamente y como se logra completar.



Despues podemos observar como se tiñen de rojo al equivocarnos y como podemos corregirlas si nos sobra tiempo, por ultimo se muestra como es que se nos termina el tiempo y ya no se puede responder.



Menu de Juegos - Lernit App

Descripción General

La **actividad de juegos** en la aplicación **LearnIt** es la pantalla principal desde donde los usuarios pueden acceder a diferentes juegos educativos que ayudan a mejorar su vocabulario y comprensión del inglés. La pantalla está compuesta por cuatro juegos disponibles, cada uno representado por un CardView que al hacer clic redirige a la actividad correspondiente.

Características Principales

- **Acceso a múltiples juegos educativos:** Cuatro juegos disponibles en esta actividad, cada uno representado por un CardView.
- **Interactividad:** Cada CardView es un botón que al hacer clic lanza una nueva actividad, permitiendo al usuario elegir el juego que desea jugar.
- **Navegación sencilla:** Uso de Intents para navegar entre actividades.

Interfaz de Usuario

La interfaz de la actividad está compuesta por los siguientes elementos principales:

1. **CardViews de Juegos:** Cuatro CardView representando los diferentes juegos:

- **Juego de Palabras**
- **Juego de Pronunciación**
- **Juego de Memoria**
- **Crucigrama**

Cada uno de estos CardView se utiliza para navegar a la actividad específica del juego al hacer clic sobre ellos.

2. **Estilo Visual de los CardViews:** Cada CardView tiene un estilo visual atractivo que incluye una imagen o ícono representativo del juego.



Lógica de la Actividad

- **Navegación entre actividades:** Cuando el usuario hace clic en un CardView, se crea un Intent que lanza la actividad correspondiente.

```
24     // Configuración de intents para cada CardView
25     wordGameCard.setOnClickListener(new View.OnClickListener() {
26         @Override
27         public void onClick(View v) {
28             // Llamar a WordsActivity cuando se hace clic en el CardView de Word Game
29             startActivity(new Intent(getApplicationContext(), WordsActivity.class));
30         }
31     );
32
33     quizGameCard.setOnClickListener(new View.OnClickListener() {
34         @Override
35         public void onClick(View v) {
36
37             startActivity(new Intent(getApplicationContext(), PronunciationGameActivity.class));
38         }
39     );
39
40     memoryGameCard.setOnClickListener(new View.OnClickListener() {
41         @Override
42         public void onClick(View v) {
43             startActivity(new Intent(getApplicationContext(), MemoryGameActivity.class));
44         }
45     );
46
47     crosswordGameCard.setOnClickListener(new View.OnClickListener() {
48         @Override
49         public void onClick(View v) {
50             startActivity(new Intent(getApplicationContext(), CrosswordGameActivity.class));
51         }
52     );
53
54 }
```

WordsActivity.java

Descripción General

El juego de palabras es un componente interactivo y educativo de la aplicación LearnIt. Diseñado para reforzar el vocabulario en inglés, reta a los usuarios a formar palabras correctamente mediante la mecánica de arrastrar y soltar letras en los espacios correspondientes.

Características Principales

- Niveles incrementales con palabras de dificultad progresiva.
- Temporizador en tiempo real que limita el tiempo disponible para completar cada nivel.
- Sistema de vidas para agregar un elemento de desafío.
- Retroalimentación visual:
 - Letras correctas resaltadas en verde.
 - Mensajes de éxito o fallo al finalizar un nivel.
- Interfaz moderna y dinámica con Material Design.

Interfaz de Usuario

El diseño de la interfaz está compuesto por los siguientes elementos principales:

1. Cabecera:

- Indicador de nivel.
- Puntuación acumulada.
- Contador de vidas restantes.

2. Temporizador:

- Muestra el tiempo restante en segundos para completar el nivel.

3. Contenedor de la palabra:

- Espacios vacíos que el usuario debe llenar arrastrando las letras correctas.

4. Contenedor de letras:

- Letras disponibles, incluyendo letras distractoras.

5. Mensajes de retroalimentación:

- Indicaciones como "Correcto", "Intenta de nuevo" o "Juego terminado!".

6. Diálogos emergentes:

- Ventanas para "Game Over" o victoria al completar todos los niveles.

Estructura del Código

Clases Principales

1. **WordsActivity**: Controla toda la lógica del juego, incluyendo la inicialización de niveles, el manejo del temporizador y las interacciones de arrastrar y soltar.

```
public class WordsActivity extends AppCompatActivity {
    private LinearLayout wordContainer; 8 usages
    private LinearLayout letterContainer; 3 usages
    private TextView scoreTextView; 2 usages
    private TextView levelTextView; 2 usages
    private TextView timerTextView; 2 usages
    private TextView livesTextView; 2 usages
    private int currentLevel = 0; 10 usages
    private int score = 0; 6 usages
    private int lives = 3; 6 usages
    private StringBuilder currentWord = new StringBuilder();
    private CountDownTimer timer; 9 usages
    private static final long TIMER_DURATION = 60000; // 60

    private final String[] words = { 6 usages
        "APPLE",
        "HOUSE",
        "BEACH",
        "BIRTHDAY",
        "SNAKE"
    };
}
```

```
@Override
protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_words);

    wordContainer = findViewById(R.id.wordContainer);
    letterContainer = findViewById(R.id.letterContainer);
    scoreTextView = findViewById(R.id.scoreTextView);
    levelTextView = findViewById(R.id.levelTextView);
    timerTextView = findViewById(R.id.timerTextView);
    livesTextView = findViewById(R.id.livesTextView);

    setupNewLevel();
}

private void startTimer() { 1 usage
    if (timer != null) {
        timer.cancel();
    }
}
```

```
private void checkWord() { 1 usage
    if (currentWord.toString().equals(words[currentLevel])) {
        // Resaltar la palabra correcta
        for (int i = 0; i < wordContainer.getChildCount(); i++) {
            TextView letterView = (TextView) wordContainer.getChildAt(i);
            letterView.setBackgroundColor(Color.GREEN);
            letterView.setTextColor(Color.WHITE);
        }

        score += 100;
        Toast.makeText(context: this, text: "Correct! +100 points", Toast.LENGTH_SHORT).show();
        currentLevel++;
    }

    wordContainer.postDelayed(this::setupNewLevel, delayMillis: 1000);
} else {
    loseLife();
}
}
```

2. Diseño del Layout (activity_words.xml): Proporciona la estructura visual del juego.

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="vertical"
    android:padding="16dp"
    android:background="#F0F4F8">

    <!-- Header con nivel, puntuación, vidas y temporizador -->
    <LinearLayout
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:orientation="horizontal"
        android:padding="8dp">

        <TextView
            android:id="@+id/levelTextView"
            android:layout_width="0dp"
            android:layout_height="wrap_content"
            android:layout_weight="1"
            android:textSize="20sp"
            android:textStyle="bold"
            android:textColor="#2D3748"/>

        <TextView
            android:id="@+id/livesTextView"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:textSize="20sp"
            android:textStyle="bold"
            android:textColor="#E53E3E"/>

    <!-- Timer -->
    <TextView
        android:id="@+id/timerTextView"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:textSize="24sp"
        android:textStyle="bold"
        android:textColor="#4A5568"
        android:textAlignment="center"
        android:padding="8dp"/>

    <!-- Instructions -->
    <TextView
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:text="Press and hold a letter, then drag it to the correct space"
        android:textSize="18sp"
        android:textStyle="center"
        android:padding="16dp"
        android:textColor="#2D3748"/>
```

```
<!-- Word container -->
<LinearLayout
    android:id="@+id/wordContainer"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:orientation="horizontal"
    android:gravity="center"
    android:background="@drawable/container_background"
    android:padding="16dp"
    android:layout_margin="16dp"
    android:elevation="4dp"/>

<!-- Letter container -->
<LinearLayout
    android:id="@+id/letterContainer"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:orientation="horizontal"
    android:gravity="center"
    android:background="@drawable/container_background"
    android:padding="16dp"
    android:layout_margin="16dp"
    android:elevation="4dp"/>

</LinearLayout>
```

Mecánica del Juego

Inicialización del Nivel

Cada nivel incluye:

- La palabra objetivo que el usuario debe formar.
- Letras mezcladas, incluyendo algunas distractoras.
- Temporizador configurado a 60 segundos por nivel.

Sistema de Arrastrar y Soltar

- Las letras se pueden arrastrar desde el contenedor de letras hacia los espacios vacíos.
- Cada vez que el usuario suelta una letra, se verifica si coincide con la posición correcta:
 - **Correcto:** La letra se fija en su lugar.
 - **Incorrecto:** No se fija y el usuario puede intentarlo nuevamente.

Sistema de Retroalimentación

- **Visual:**

- Letras correctas resaltadas en verde.
- Letras incorrectas permanecen grises.
- **Auditiva:**
 - Mensajes de retroalimentación (éxito o fallo).
- **Diálogos emergentes:**
 - "¡Felicitaciones!" al completar todos los niveles.
 - "¡Juego terminado!" si se acaban las vidas.

Temporizador

- **Duración:** 60 segundos por nivel.
- **Actualización:** Cada segundo se reduce el contador en la pantalla.
- **Acciones:**
 - Al llegar a 0, se pierde una vida.
 - Si se completan los niveles dentro del tiempo, se acumulan puntos adicionales.

Palabras y Dificultad

El juego incluye las siguientes palabras (ordenadas por nivel de dificultad):

1. **Fácil:** APPLE, HOUSE.
2. **Intermedio:** BEACH, SNAKE.
3. **Difícil:** BIRTHDAY.

Sistema de Puntuación

1. **Base:**
 - +100 puntos por palabra completada correctamente.
2. **Penalizaciones:**

- o Pérdida de una vida por no completar a tiempo.

3. Bonificaciones:

- o +50 puntos si se completa con más de 10 segundos restantes.

Diseño del Layout

Cabecera

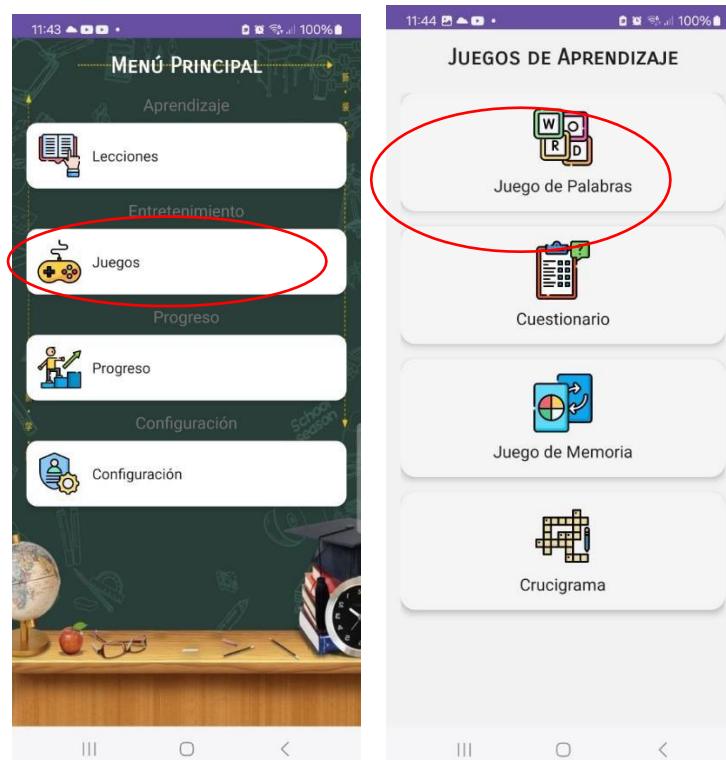
- ② Indicador de nivel: Muestra el nivel actual.
- ② Puntuación: Incrementa con cada palabra completada correctamente.
- ② Vidas: Se actualizan visualmente al perder una vida.

Contenedor de Palabra

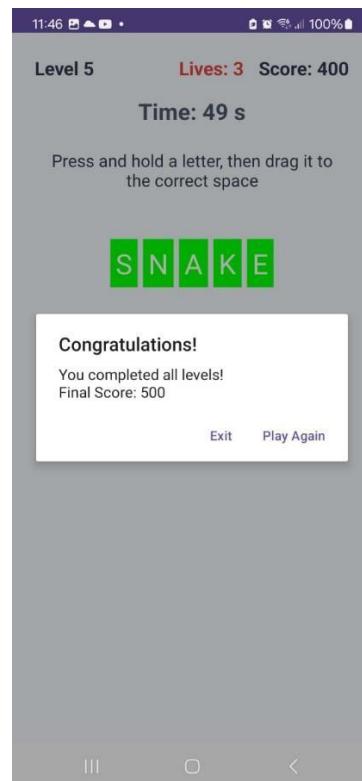
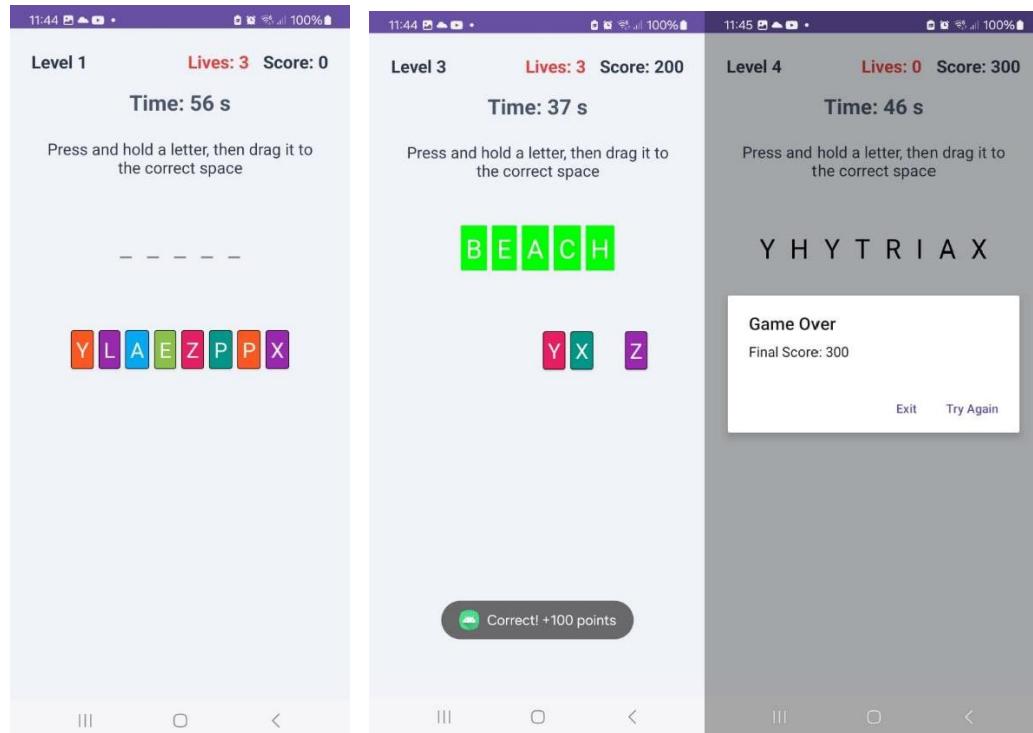
- ② Espacios vacíos que indican el número de letras en la palabra objetivo.
- ② Letras resaltadas cuando se completan correctamente.

Contenedor de Letras

- ② Letras organizadas horizontalmente, listas para ser arrastradas.



El juego inicia con 3 vidas y cuenta con un timer, conforme contestes correctamente se te suman puntos y si te equivocas pierdes vidas y no te deja avanzar hasta que las contestes correctamente.



PronunciationGameActivity.java

Descripción General

El juego de pronunciación es un componente interactivo de la aplicación LearnIt, diseñado para mejorar la capacidad de los usuarios en la correcta pronunciación de palabras en inglés. Utiliza tecnologías de reconocimiento de voz y síntesis de texto a voz para ofrecer una experiencia inmersiva y educativa.

Características Principales

1. Reconocimiento de Voz:

- Utiliza la API de SpeechRecognizer de Android para detectar la pronunciación de las palabras.
- Proporciona retroalimentación basada en la precisión del usuario.

2. Síntesis de Texto a Voz:

- El TextToSpeech permite que el usuario escuche la pronunciación correcta de la palabra objetivo antes de intentar reproducirla.

3. Progresión del Juego:

- Sistema de niveles y puntuación.
- Palabras aleatorias en cada nivel con sus significados asociados.

4. Interfaz Amigable:

- Botones interactivos para reproducir la palabra, iniciar grabación y avanzar al siguiente nivel.
- Retroalimentación visual y auditiva.

Interfaz de Usuario

Elementos Principales

1. Cabecera:

- **Nivel actual:** Indica el nivel que el jugador está jugando.
- **Puntuación:** Muestra los puntos acumulados.

2. Palabra y Significado:

- La palabra objetivo aparece destacada junto con su significado.

3. Botones:

- **Reproducir Pronunciación:** Escucha cómo se pronuncia la palabra.
- **Hablar:** Intenta pronunciar la palabra y recibe retroalimentación.
- **Siguiente:** Avanza al próximo nivel.

4. Retroalimentación Auditiva:

- Sonidos de éxito y fallo.

Mecánica del Juego

Flujo de la Actividad

1. Aliniciar la actividad:

- Se selecciona una palabra aleatoria del diccionario interno.
- Se muestra en pantalla con su significado.
- Se inicializan los botones y controles.

2. Pronunciación de la Palabra:

- El usuario presiona "Press to Speak" para comenzar a hablar.
- La aplicación evalúa la pronunciación usando SpeechRecognizer.

3. Validación:

- Si la pronunciación es correcta:
 - El usuario gana puntos (+10).
 - Se muestra una animación de éxito.
- Si es incorrecta:
 - La palabra hablada se muestra como retroalimentación.
 - El usuario puede intentar nuevamente.

4. Progresión:

- Cada 30 puntos acumulados, el nivel aumenta.
- El juego finaliza cuando el usuario completa todas las palabras.

```
public class PronunciationGameActivity extends AppCompatActivity {  
    private static final int PERMISSION_REQUEST_CODE = 123; 2 usages  
    private SpeechRecognizer speechRecognizer; 5 usages  
    private TextToSpeech textToSpeech; 6 usages  
    private TextView wordTextView; 6 usages  
    private TextView scoreTextView; 3 usages  
    private TextView levelTextView; 3 usages  
    private TextView meaningTextView; 3 usages  
    private ImageButton speakerButton; 3 usages  
    private Button nextButton; 3 usages  
    private Button startRecordingButton; 9 usages  
    private int currentScore = 0; 6 usages  
    private int currentLevel = 1; 4 usages  
    private MediaPlayer correctSound; 5 usages  
    private MediaPlayer incorrectSound; 5 usages  
    private boolean isListening = false; 3 usages  
  
    // HashMap para almacenar palabras y sus significados  
    private HashMap<String, String> words = new HashMap<String, String>() {{ 2 usages  
        put("Apple", "A round fruit with red or green skin and white flesh");  
        put("Beautiful", "Pleasing to the senses or mind aesthetically");  
        put("Computer", "An electronic device for storing and processing data");  
        put("Dancing", "Moving rhythmically to music");  
        put("Elephant", "A very large plant-eating mammal with a trunk");  
        put("Family", "A group of people who are related to each other");  
        put("Garden", "A piece of ground used to grow plants, fruits, or vegetables");  
        put("Hospital", "A place where sick or injured people receive medical care");  
        put("Internet", "A global computer network providing information and communication");  
        put("Journey", "An act of traveling from one place to another");  
    }  
}
```

```
private void initializeSoundEffects() { 1 usage  
    correctSound = MediaPlayer.create( context: this, R.raw.correct_sound);  
    incorrectSound = MediaPlayer.create( context: this, R.raw.incorrect_sound);  
}
```

```
private void resetRecordingButton() { 4 usages
    isListening = false;
    startRecordingButton.setText("Presiona para hablar");
    startRecordingButton.setEnabled(true);
    startRecordingButton.setBackgroundResource(R.drawable.rounded_button);
}

private void handleCorrectPronunciation() { 1 usage
    currentScore += 10;
    playCorrectSound();
    Toast.makeText(context: this, text: "Excelente pronunciación! +10 puntos", Toast.LENGTH_SHORT).show();
    scoreTextView.setText("Score: " + currentScore);

    // Animación de éxito
    wordTextView.setTextColor(getResources().getColor(android.R.color.holo_green_dark));
    wordTextView.postDelayed(() -> {
        wordTextView.setTextColor(getResources().getColor(R.color.text_primary));
        wordList.remove(currentWord);
        checkLevelProgress();
        showNextWord();
    }, delayMillis: 1000);
}

private void handleIncorrectPronunciation(String spokenText) { 1 usage
    playIncorrectSound();
    Toast.makeText(context: this, text: "Intenta de nuevo. Dijiste: " + spokenText, Toast.LENGTH_LONG).show();
}

@Override 16 usages
public void onRequestPermissionsResult(int requestCode, @NonNull String[] permissions, @NonNull int[] grantResults) {
    super.onRequestPermissionsResult(requestCode, permissions, grantResults);
    if (requestCode == PERMISSION_REQUEST_CODE) {
        if (grantResults.length > 0 && grantResults[0] == PackageManager.PERMISSION_GRANTED) {
            Toast.makeText(context: this, text: "Permisos concedidos", Toast.LENGTH_SHORT).show();
        } else {
            Toast.makeText(context: this, text: "Se necesitan permisos para el micrófono",
            Toast.LENGTH_SHORT).show();
            finish();
        }
    }
}
```

```
<?xml version="1.0" encoding="utf-8"?>
<androidx.constraintlayout.widget.ConstraintLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:padding="16dp">

    <TextView
        android:id="@+id/levelTextView"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_marginTop="16dp"
        android:text="Level: 1"
        android:textSize="18sp"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toTopOf="parent" />

    <TextView
        android:id="@+id/scoreTextView"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_marginTop="16dp"
        android:text="Score: 0"
        android:textSize="18sp"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintTop_toTopOf="parent" />
```

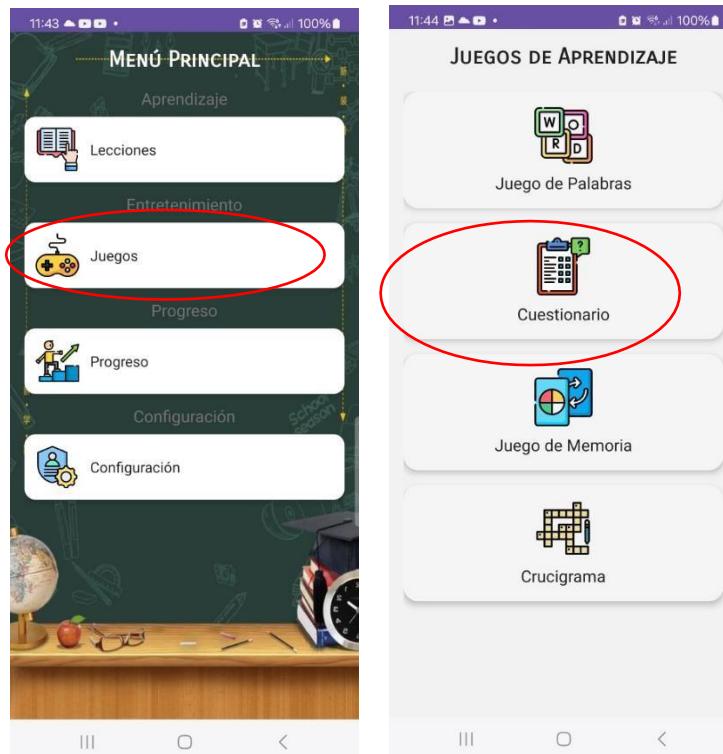
```
<TextView
    android:id="@+id/wordTextView"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_marginTop="48dp"
    android:textSize="32sp"
    android:textStyle="bold"
    android:textColor="@color/text_primary"
    app:layout_constraintEnd_toEndOf="parent"
    app:layout_constraintStart_toStartOf="parent"
    app:layout_constraintTop_toBottomOf="@+id/levelTextView" />

    <TextView
        android:id="@+id/meaningTextView"
        android:layout_width="0dp"
        android:layout_height="wrap_content"
        android:layout_marginTop="16dp"
        android:gravity="center"
        android:textSize="16sp"
        android:padding="16dp"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toBottomOf="@+id/wordTextView" />
```

```
<Button  
    android:id="@+id/startRecordingButton"  
    android:layout_width="wrap_content"  
    android:layout_height="wrap_content"  
    android:layout_marginTop="32dp"  
    android:text="Press to Speak"  
    android:textColor="@color/black"  
    android:padding="16dp"  
    android:background="@drawable/rounded_button"  
    app:layout_constraintEnd_toEndOf="parent"  
    app:layout_constraintStart_toStartOf="parent"  
    app:layout_constraintTop_toBottomOf="@+id/speakerButton" />
```

El juego te da una pequeña descripción sobre la palabra que se pronunciara, despúeshay un texto que dice Press to Speak y al presionarlo el micrófono se activa

escuchara tu pronunciación, de igual manera si presionas la pantalla por encima del Press to Speak te dice la pronunciación correcta, por ende, si lo dices bien o mal suenaun sonido de correcto o incorrecto y te muestra un mensaje, si lo dices bien te pasa ala siguiente, aumenta el Score y subes de nivel, sino no te deja avanzar a menos que ledes en Next.



Three screenshots of a mobile application interface, likely a language learning game, showing three different levels of a task.

Level 1: Score: 0. Task: Identify an "Elephant". Description: "A very large plant-eating mammal with a trunk". Buttons: "Press to Speak" and "Next". Status: "Escuchando..." (Listening).

Level 2: Score: 10. Task: Identify an "Elephant". Description: "A very large plant-eating mammal with a trunk". Buttons: "Presiona para hablar" (Press to speak) and "Next". Status: "Escuchando..." (Listening).

Level 3: Score: 40. Task: Identify a "Beautiful". Description: "Pleasing to the senses or mind aesthetically". Buttons: "Presiona para hablar" (Press to speak) and "Next". Status: "Escuchando..." (Listening).

Two screenshots of a mobile application interface, likely a language learning game, showing two different levels of a task.

Level 4: Score: 100. Task: Complete the game. Message: "¡Felicitaciones! Has completado el juego Puntaje final: 100". Buttons: "Presiona para hablar" and "Next". Status: "Escuchando..." (Listening).

Level 5: Score: 0. Task: Identify an "Apple". Description: "A round fruit with red or green skin and white flesh". Buttons: "Presiona para hablar" and "Next". Status: "Intenta de nuevo. Dijiste: House" (Try again. You said: House).

Juego de Crucigrama - LearnIt App

Descripción General

El juego de crucigrama es un componente educativo de la aplicación LearnIt, diseñado para ayudar a los usuarios a aprender vocabulario en inglés relacionado con la naturaleza. El juego presenta un crucigrama interactivo de 10x10 con palabras en inglés y pistas en forma de definiciones.

Características Principales

- Tablero de crucigrama interactivo de 10x10
- Sistema de puntuación dinámico
- Temporizador en tiempo real
- Verificación de respuestas con retroalimentación visual
- Pistas organizadas por dirección (horizontal y vertical)
- Interfaz de usuario intuitiva y responsive

Interfaz de Usuario

La interfaz está compuesta por varios elementos principales:

- Barra superior con título "Nature Crossword"
- Panel de información con temporizador y puntuación
- Cuadrícula del crucigrama
- Sección de pistas
- Botón de verificación de respuestas

Estructura del Código

- Clases Principales

```
> public class CrosswordGameActivity extends AppCompatActivity {  
    private EditText[][] cellGrid; 7 usages  
    private final int GRID_SIZE = 10; 9 usages  
    private int currentScore = 0; 4 usages  
    private int attempts = 0; 3 usages  
    private TextView scoreText; 2 usages  
    private TextView timerText; 2 usages  
    private long startTime; 3 usages  
    private Handler timerHandler = new Handler(); 5 usages  
    private boolean gameCompleted = false; 4 usages
```

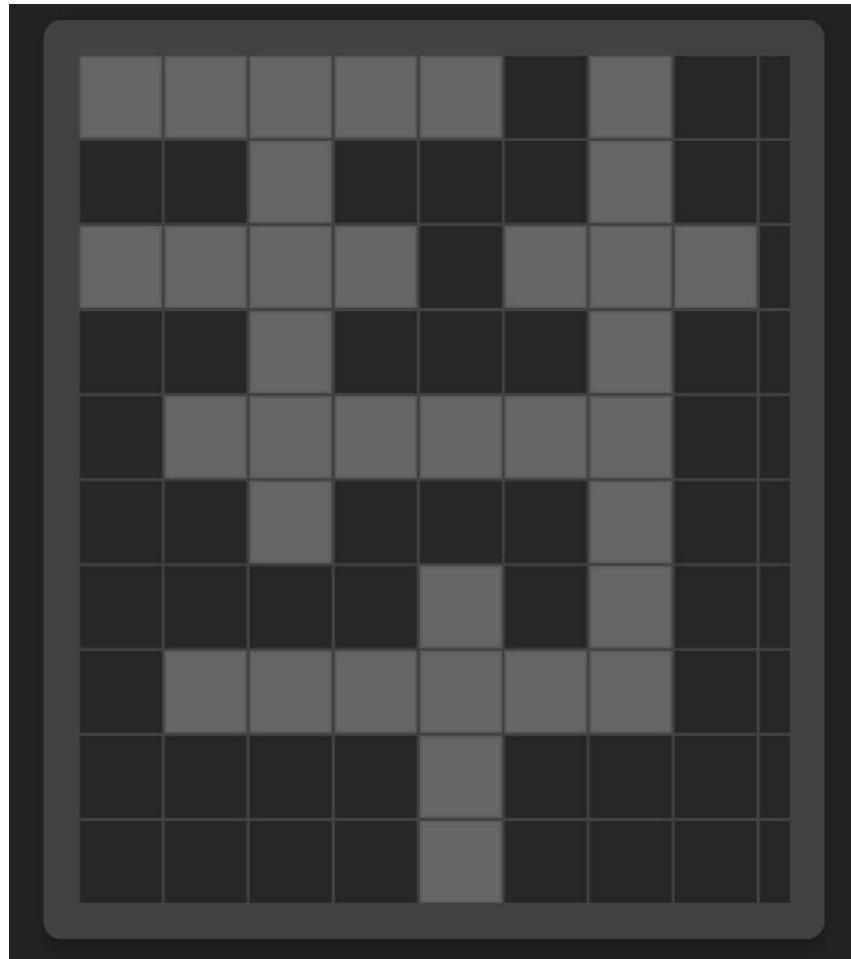
Constantes del Juego

```
private static final int MAX_SCORE = 100; 1 usage
private static final int TIME_BONUS_MAX = 50; 1 usage
private static final int ATTEMPT_PENALTY = 10; 1 usage
private static final int INCORRECT_CELL_PENALTY = 5; 1 usage
```

Implementación

- Inicialización del Tablero

```
private void initializeGrid() { 1 usage
    GridLayout gridLayout = findViewById(R.id.gridLayout);
    gridLayout.setColumnCount(GRID_SIZE);
    gridLayout.setRowCount(GRID_SIZE);
    cellGrid = new EditText[GRID_SIZE][GRID_SIZE];
```



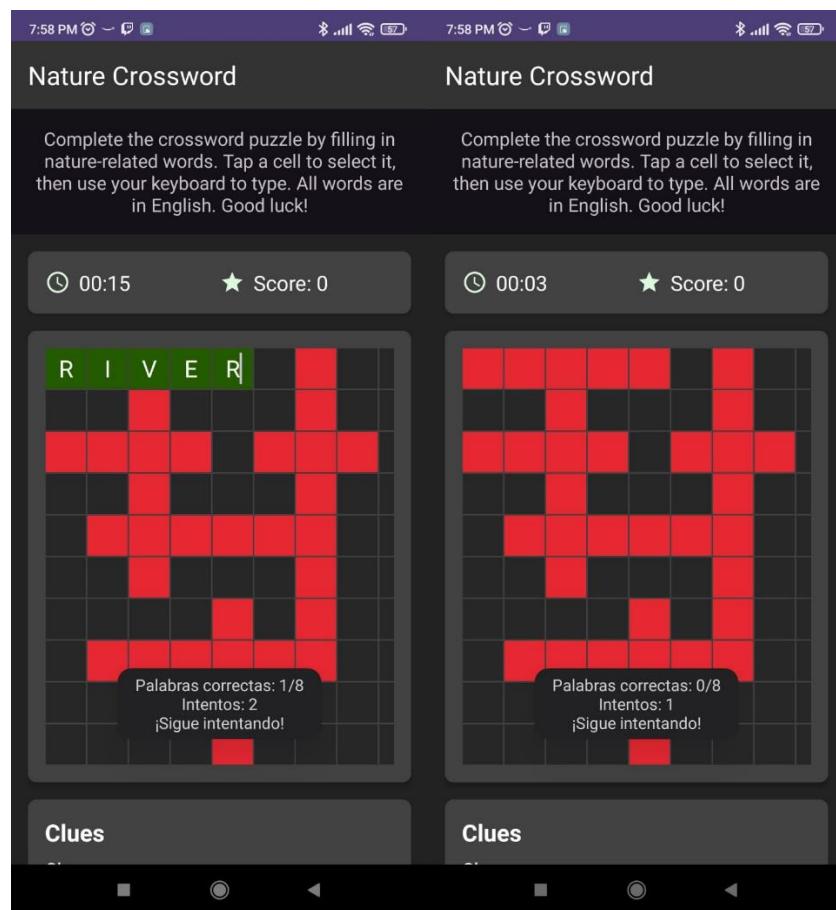
Palabras y Posiciones

El juego incluye 8 palabras relacionadas con la naturaleza:

- Horizontales: RIVER, DESERT, HILL, SUN, LAGOON
- Verticales: MOUNTAIN, VALLEY, POND

Sistema de Verificación

```
public void checkAnswers(View view) { 1 usage
    if (gameCompleted) {
        Toast.makeText(context, text: "¡Juego ya completado!", Toast.LENGTH_SHORT).show();
        return;
    }
```



Sistema de Puntuación

Cálculo de Puntuación

La puntuación se calcula considerando varios factores:

1. **Puntuación Base:** Máximo 100 puntos
2. **Bonus por Tiempo:** Hasta 50 puntos adicionales
3. **Penalizaciones:**
 - -10 puntos por cada intento
 - -5 puntos por cada celda incorrecta



Documentación del Juego de Memoria - LearnIt App

Descripción General

El juego de memoria (Memory Game) es un componente educativo de la aplicación LearnIt diseñado para ayudar a los estudiantes a aprender y reforzar los verbos irregulares en inglés, específicamente la relación entre el presente simple y el pasado simple. El juego implementa la mecánica clásica de memorama con elementos interactivos y retroalimentación multimedia.

Características Principales

- Tablero de 4x4 con 6 pares de verbos
- Temporizador de 90 segundos
- Efectos de sonido y animaciones
- Retroalimentación visual y auditiva
- Sistema de parejas de verbos presente-pasado
- Interfaz moderna con Material Design

Interfaz de Usuario

Componentes Principales

1. Título del Juego

- CardView con título "Memory Game: Simple Past Verbs"
- Diseño elevado y esquinas redondeadas

2. Temporizador

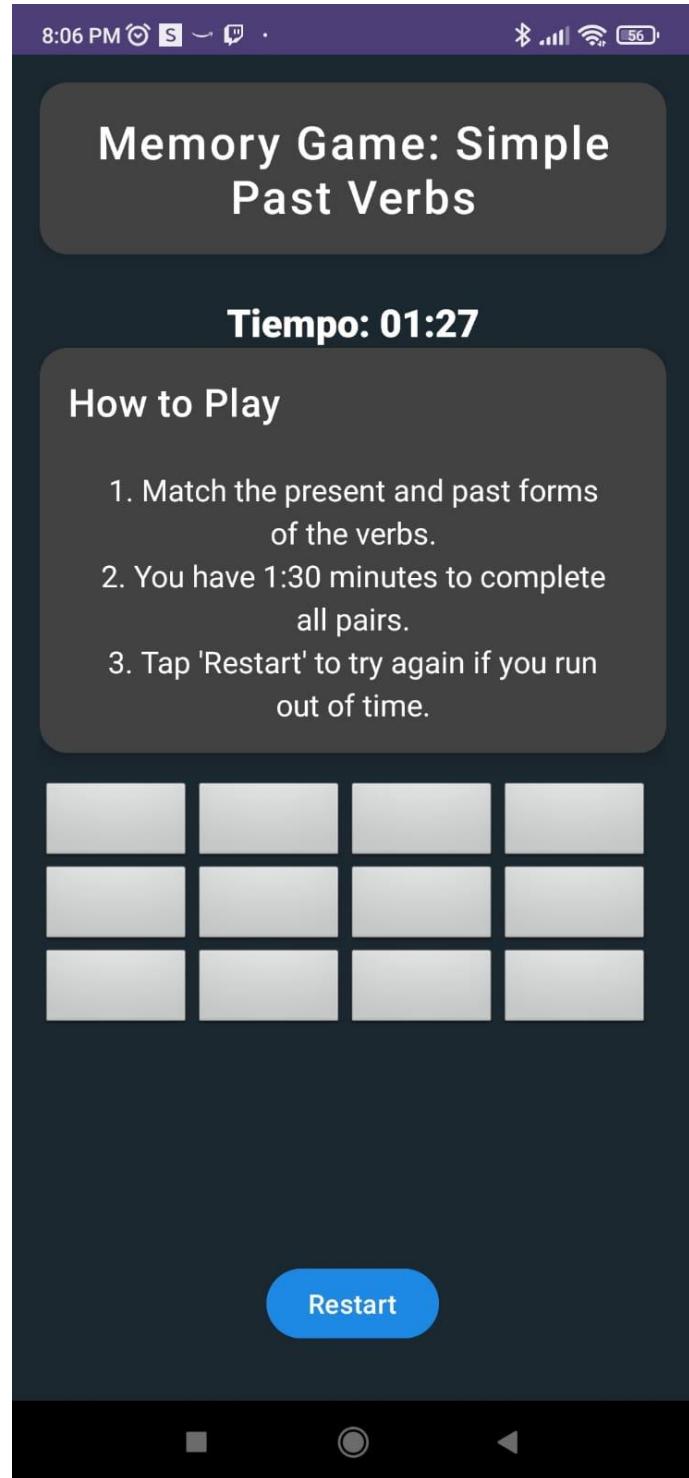
- Formato MM:SS
- Efectos de sonido tick-tack
- Ubicación central prominente

3. Instrucciones

- CardView con reglas básicas del juego
- Formato paso a paso

4. Tablero de Juego

- Grid de 4x4
- Tarjetas interactivas con animación
- Diseño responsivo



Estructura del Código

Clases Principales

```
public class MemoryGameActivity extends AppCompatActivity {  
  
    private GridLayout gridMemoryGame; 5 usages  
    private Button btnRestart; 2 usages  
    private TextView tvTitle, tvTimer; 1 usage  
    private ArrayList<Button> buttons;| 2 usages  
    private ArrayList<String> verbs; 16 usages  
    private Button firstSelected, secondSelected; 9 usages  
    private boolean isClickable = true; 5 usages  
    private int matches = 0; 3 usages  
    private CountDownTimer countDownTimer; 7 usages  
    private MediaPlayer tickTackPlayer; 8 usages
```

Pares de Verbos Implementados

```
private void loadMemoryGame() { 2 usages  
    verbs = new ArrayList<>();  
    verbs.add("run");  
    verbs.add("ran");  
    verbs.add("go");|  
    verbs.add("went");  
    verbs.add("eat");  
    verbs.add("ate");  
    verbs.add("write");  
    verbs.add("wrote");  
    verbs.add("read");  
    verbs.add("read");  
    verbs.add("fly");  
    verbs.add("flew");
```

Mecánica del Juego

Inicialización

```
private void loadMemoryGame() { 2 usages
```

Sistema de Juego

1. Selección de Tarjetas

- Primera selección: revela el verbo
- Segunda selección: revela y verifica el par
- Animación de volteo en cada selección

2. Verificación de Pares

- *Verifica si los verbos forman un par válido*
- *Actualiza el estado del juego*
- *Proporciona retroalimentación*

```
private void checkMatch() { 1 usage
```



Sistema de Retroalimentación

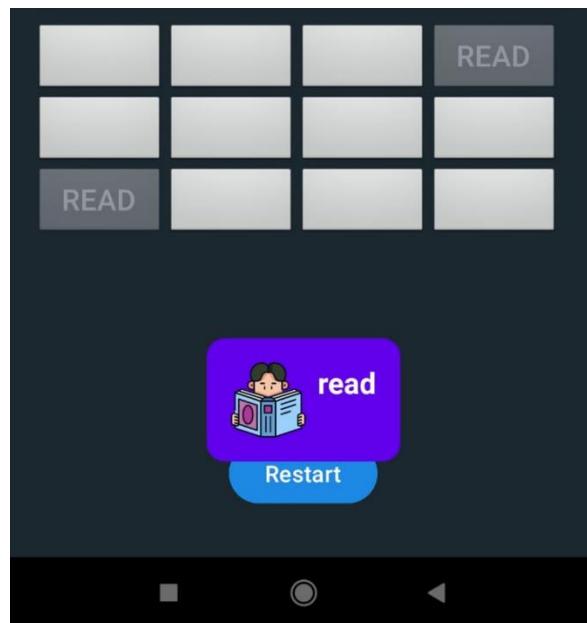
Visual

- Animaciones de volteo de tarjetas
- Desaparición de pares correctos
- Toast personalizado con imagen del verbo

Auditiva

- Sonido de tick-tack durante el juego
- Efectos de sonido para pares correctos
- Pronunciación del verbo en pasado

```
private void showCustomToast(String pastVerb) { 1 usage
    LayoutInflater inflater = getLayoutInflater();
    View layout = inflater.inflate(R.layout.custom_toast, root: null);
```



Sistema de Tiempo

Temporizador

- Duración: 90 segundos
- Actualización: cada segundo
- Formato de visualización: MM:SS
- Efectos de sonido sincronizados

```
private void startTimer() { 1 usage
    if (countDownTimer != null) {
        countDownTimer.cancel();
    }
```

Actividad de Configuración - Leranit App

Descripción General

La actividad de configuración en la aplicación LearnIt permite a los usuarios gestionar diversas opciones relacionadas con su cuenta, preferencias y ajustes de la aplicación. Desde esta sección, los usuarios pueden personalizar aspectos como el

idioma de la aplicación, el nivel de dificultad de los ejercicios, activar o desactivar las notificaciones, y ver información sobre su perfil, como el nombre de usuario y el nivel de progreso. Además, incluye una barra de tiempo de estudio (StudyTimeBar) que permite a los usuarios llevar un registro de su progreso diario.

Características Principales

- Gestión de usuario: Visualización y actualización de la información del usuario (nombre, nivel).
- Idioma: Opción para cambiar el idioma de la aplicación (Español/Inglés).
- Nivel de dificultad: Ajustes para personalizar la dificultad de los ejercicios.
- Notificaciones: Opción para activar o desactivar las notificaciones de la aplicación.
- Barra de tiempo de estudio (StudyTimeBar): Herramienta para medir el tiempo de estudio y progreso.
- Cerrar sesión: Opción para cerrar la sesión actual del usuario.

Interfaz de Usuario

La interfaz se compone de los siguientes elementos:

1. Información del usuario:

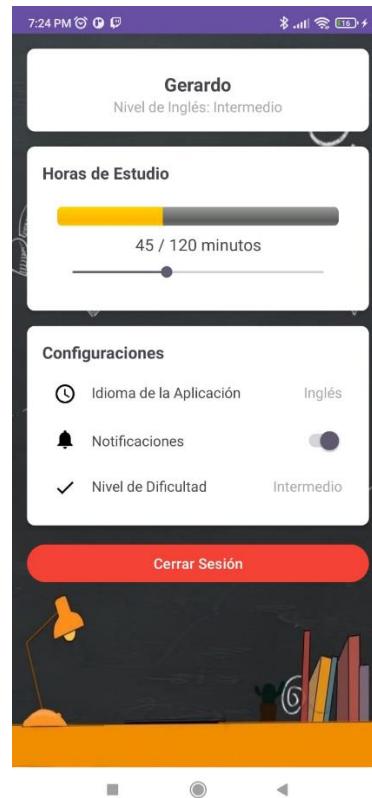
- Nombre de usuario: TextView que muestra el nombre del usuario actual.
- Nivel del usuario: TextView que muestra el nivel de progreso o aprendizaje del usuario.

2. Configuración de preferencias:

- Idioma: LinearLayout con opción para cambiar el idioma.
- Dificultad: LinearLayout con opción para cambiar el nivel de dificultad (principiante, intermedio, avanzado).
- Notificaciones: Switch que permite activar o desactivar las notificaciones de la aplicación.

3. Barra de progreso de estudio:

- StudyTimeBar: Un componente visual que muestra el tiempo de estudio registrado por el usuario.
4. Cerrar sesión:
- Botón de cerrar sesión: Button que permite al usuario cerrar sesión y volver a la pantalla de inicio.



Lógica de la Actividad

- Inicialización de vistas:
 - Se inicializan todos los elementos de la interfaz como TextViews, Switch, y Buttons en el método initializeViews().
- Carga de datos del usuario:
 - Se obtiene la información del usuario desde la base de datos y SharedPreferences para mostrarla en los TextViews.
- Cerrar sesión:
 - El botón de cerrar sesión elimina **los datos de sesión y redirige a la pantalla de inicio**.

Base de Datos - Lernit App

Descripción General

La base de datos en la aplicación LearnIt se utiliza para gestionar la información del usuario, específicamente los datos de inicio de sesión (nombre de usuario y contraseña). Está implementada utilizando SQLite a través de la clase SQLiteOpenHelper, que facilita la creación, actualización y manejo de la base de datos.

Estructura de la Base de Datos

La base de datos **LearnItDatabase** contiene una sola tabla, llamada **usuarios**, que almacena la información de los usuarios registrados. Los campos relevantes en esta tabla son:

- **id**: Identificador único del usuario (clave primaria, autoincremental).
- **username**: Nombre de usuario único (debe ser único en la base de datos).
- **password**: Contraseña del usuario.

```
public class DatabaseHelper extends SQLiteOpenHelper { 4 usages
    private static final String DATABASE_NAME = "LearnItDatabase"; 1 usage
    private static final int DATABASE_VERSION = 1; 1 usage

    // Tabla de usuarios
    private static final String TABLE_USUARIOS = "usuarios"; 5 usages
    private static final String COLUMN_ID = "id"; 2 usages
    private static final String COLUMN_USERNAME = "username"; 5 usages
    private static final String COLUMN_PASSWORD = "password"; 3 usages

    // Script de creación de tabla
    private static final String CREATE_TABLE_USUARIOS = 1 usage
        "CREATE TABLE " + TABLE_USUARIOS + "("
            + COLUMN_ID + " INTEGER PRIMARY KEY AUTOINCREMENT, "
            + COLUMN_USERNAME + " TEXT UNIQUE, "
            + COLUMN_PASSWORD + " TEXT)";

    public DatabaseHelper(Context context) { 2 usages
        super(context, DATABASE_NAME, null, DATABASE_VERSION);
    }
}
```

Clases y Métodos Principales

DatabaseHelper

La clase DatabaseHelper extiende SQLiteOpenHelper y se utiliza para crear y actualizar la base de datos. Es responsable de la inicialización y gestión de la base de datos en la aplicación.

- Método onCreate(SQLiteDatabase db): Este método se ejecuta cuando se crea la base de datos por primera vez. Crea la tabla usuarios mediante el script SQL definido en CREATE_TABLE_USUARIOS.
- Método onUpgrade(SQLiteDatabase db, int oldVersion, int newVersion): Este método se ejecuta cuando hay una actualización en la versión de la base de datos. En este caso, elimina la tabla usuarios y la recrea.

La clase DatabaseHelper proporciona varios métodos para realizar operaciones básicas sobre la base de datos.

insertUser(String username, String password)

Este método permite insertar un nuevo usuario en la tabla usuarios.

Funcionamiento:

- Parámetros:
 - username: Nombre de usuario que se insertará en la base de datos.
 - password: Contraseña asociada al nombre de usuario.
- Proceso: Crea un objeto ContentValues y agrega los valores username y password. Luego, inserta estos valores en la tabla usuarios.

checkUser(String username, String password)

Este método permite verificar si un usuario con el nombre de usuario y contraseña proporcionados existe en la base de datos.

```

// Método para insertar usuario
public boolean insertUser(String username, String password) { 1 usage
    SQLiteDatabase db = this.getWritableDatabase();
    ContentValues contentValues = new ContentValues();
    contentValues.put(COLUMN_USERNAME, username);
    contentValues.put(COLUMN_PASSWORD, password);

    try {
        long result = db.insertOrThrow(TABLE_USUARIOS, nullColumnHack: null, contentValues);
        return result != -1;
    } catch (Exception e) {
        Log.e(tag: "DatabaseHelper", msg: "Error inserting user", e);
        return false;
    } finally {
        db.close();
    }
}

// Método para verificar usuario
public boolean checkUser(String username, String password) { 1 usage
    SQLiteDatabase db = this.getReadableDatabase();
    String[] columns = {COLUMN_ID};
    String selection = COLUMN_USERNAME + " = ? AND " + COLUMN_PASSWORD + " = ?";
    String[] selectionArgs = {username, password};

    try {
        Cursor cursor = db.query(TABLE_USUARIOS, columns, selection, selectionArgs, groupBy: null, having: null, orderBy: null);
        int cursorCount = cursor.getCount();
        cursor.close();
        return cursorCount > 0;
    } catch (Exception e) {
        Log.e(tag: "DatabaseHelper", msg: "Error checking user", e);
        return false;
    } finally {
        db.close();
    }
}

```

Uso de la Base de Datos en la Aplicación

1. Inserción de Usuarios:

- En la pantalla de registro o al inicio de sesión, cuando un usuario se registra, se usa el método insertUser para agregar un nuevo usuario a la base de datos.

2. Verificación de Usuario:

- En el proceso de inicio de sesión, el método checkUser se usa para verificar que el nombre de usuario y la contraseña coincidan con un registro en la base de datos.

3. Recuperación de Información de Usuario:

- Al cargar los datos del usuario (como el nombre de usuario), se usa el método getUsername para obtener el nombre almacenado.

CONCLUSIÓN

Este proyecto, **LearnIt**, representa uno de nuestros primeros acercamientos al desarrollo de aplicaciones móviles, específicamente utilizando Android Studio. A lo largo del semestre, hemos aprendido los fundamentos del desarrollo en Android, incluyendo la creación de interfaces intuitivas, la implementación de clases y métodos, y la integración de elementos interactivos como botones, barras de progreso y temporizadores.

Este proyecto nos permitió aplicar de manera práctica todos los conocimientos adquiridos en la materia de **Dispositivos Móviles**, superando los retos iniciales de trabajar con un entorno de desarrollo nuevo y enfrentándonos a problemas reales de diseño y programación.

El proceso de desarrollo no solo fortaleció nuestras habilidades técnicas, sino que también nos permitió comprender mejor cómo las aplicaciones educativas pueden impactar positivamente el aprendizaje de los usuarios. Al implementar juegos como crucigramas y ejercicios de vocabulario, vimos cómo la gamificación puede hacer que el aprendizaje sea más atractivo y efectivo.

En conclusión, **LearnIt** no solo es un reflejo de nuestro crecimiento técnico a lo largo del semestre, sino también de nuestra capacidad para diseñar soluciones creativas y funcionales. Este proyecto nos motiva a seguir explorando el potencial de la tecnología para resolver problemas del mundo real.

REFERENCIAS:

<https://developer.android.com/training/data-storage/sqlite?hl=es-419>

<https://developer.android.com/reference/androidx/cardview/widget/CardView>

<https://developer.android.com/reference/androidx/appcompat/app/AppCompatActivity>

<https://developer.android.com/reference/android/widget/LinearLayout>

<https://developer.android.com/design?hl=es-419>