README

Usage

Requerimientos.

- flutter.
- android studio.

Clonar el proyecto.

```
git clone https://github.com/Sonya-c/mathlingo.git
```

Descargar las dependencias.

```
flutter flutter pud
```

Ejecutar el programa utilizando un emulador.

```
flutter run android
```

O conectador a un movíl fisico.

```
flutter run -d [device-id]
```

Para buscar el id de tu dispositivo ejecuta.

```
flutter devices
```

UI

Login page

Singup page

Home page

Esta página solo es visible si se esta autenticado. Estados:

- Level: int. Inicia en 0 y se actualiza cada vez que se regresa de una sección de juego

Ver controlador

Game page

Esta página solo es visible si se esta autenticado. Estados:

- Answer: string. Inicia vacia. Es la respuesta del usuario. Cambiar cuando:
    - El usuario escribe/borra en el numpad.
    - El usuario reponse una pregunta (regresa al estado inicial)
- numQuestions: int. Inicia en 1. Es el número de pregunta actual. Cambia cuando:
    - El usuario enviar una respuesta aumentando.
    - El usuario termina una sección de juego, regresando al estado original.
- correctAnswers: int. Inicia en 0 y representa el número de respuestas correctas del usuario. Cambia:
    - Cuando el usuario envía una respuesta correcta.
    - Cuando el usuario termina una sección de juego, regresando al estado original.
- levelUp: int. Indica si el usuario ha subido o bajado de nivel. Este inicia en 0 y se actualiza cada vez que termine una sección. Este valor es obtenido desde el controllador.
- totalTime: Duration. Representa el tiempo total de la sección. Inicia en 0 y se reinicia luego de cada sección de juego.

Ver controlador

Controllers

Auth controller

- login(email, password) -> void. Realiza el login. Si es posible, cambiara el valor de la variable observable `logged`. Ver login page.
- signUp(email, password) -> bool. Realizara el singup. No tiene algun cambio en estado de la aplicación, se debe realizar el login para autenticarse. Ver singup page.
- logOut() -> void. Realiza el logout. Cambiara el valor de la variable observable `logged`.

Game controller

- generateProblem() -> MathProblem. Crea un problema al azar basado en el nivel. El nivel esta representado por una ecuación de la siguiente forma:

  $$ level = # digits * \text{operation dificulty factor} $$

  En el cual, cada operación tiene un factor de dificultad. Suma = 1, resta = 2, multiplicación = 3. De acuerdo al nivel actual, se generaran las preguntas.

  MathProblem tiene el siguiente modelo: los dos operandos, el operador y la respuesta.

```
MathProblem:
- num1: int
- num2: int
- answer: int
- op: string
```

- getAnswer() -> int
- verifyAnswer(int answer, Duration timeTaken) -> bool. Data la respuesta y el tiempo de duración, regresa si la respuesta es correcta o no (bool). Además, guarda el resultado en un array de respuestas.
- getAnswers() -> List[MathAnswers]. Regresa las respuesta almacenadas. MathAnswers tiene el siguiente modelo.

```
MathAnswer:
- mathProblem: MathProblem
- duration: Duration
- isCorrect: bool
```

- levelUp() -> int. Dada el número de respuestas correctas, se indicara si se debe subir, bajar o quedarse en el mismo nivel. Los valores posibles son:
    - 1: subir de nivel; si el número de respuestas correctas es mayor que la mitad.
    - 0: quedarse en este nivel; si el número de respuestas en exactamente la mitad.
    - -1: bajar de nivel; si el número de respuestas correctas es menos de la mitad
- getLevel() -> int. Obtener el nivel actual (revisar punto anterior para conocer posibles valores).
- clearAnswers() -> void. Vacia toda la lista de respuestas. Esta función debería ser la última en llamarse luego de completar una sección pues se perderan todos los datos.