1. Index

1. Index
2. Activities
   a. MainActivity
      i. *Attributes*
      ii. *Methods*
   b. GameActivity
      i. *Attributes*
      ii. *Methods*
3. Fragments
   a. LoginFragment
      i. *Attributes*
      ii. *Methods*
   b. RegisterFragment
      i. *Attributes*
      ii. *Methods*
4. Classes
   a. Avatar
      i. *Attributes*
      ii. *Class diagram*
   b. Extra
      i. *Attributes*
      ii. *Class diagram*
   c. Eyebrows
      i. *Attributes*
      ii. *Class diagram*
   d. Eyes
      i. *Attributes*
      ii. *Class diagram*
   e. Highscore
      i. *Attributes*
      ii. *Class diagram*
   f. Language
      i. *Attributes*
      ii. *Class diagram*
   g. Mouth
      i. *Attributes*
      ii. *Class diagram*
   h. Wear out
      i. *Attributes*
      ii. *Methods*
      iii. *Class diagram*
   i. AvatarAnimations
      i. *Attributes*
      ii. *Methods*
      iii. *Class diagram*
   j. AvatarMoods
   k. GameRound

i. *Attributes*
ii. *Methods*
iii. *Class diagram*

l. Sounds
i. *Attributes*
ii. *Methods*
iii. *Class diagram*

m. Word
i. *Attributes*
ii. *Methods*
iii. *Class diagram*

5. Adapters
a. DefinitionsAdapter
b. HighscoresAdapter

6. ViewModel
a. MainViewModel
i. *Attributes*
ii. *Methods*

7. Apis
a. RandomWordAPI
b. API Dictionary English
c. Free Dictionary API

8. Database
a. HangmanDb – Entity relationship diagram

9. Use case diagram

10. Activity Diagrams

2. Activities

a. MainActivity

i. Attributes

- private var introAnimationDelayTime: Long=250

This attribute contains the delay in milliseconds that the initial animation will have before it initializes.

- var sounds: Sounds = null

The instance of the Sounds class that contains the method for playing audio files.

ii. Methods

- override fun onCreate(savedInstanceState: Bundle?)

Here is everything that will happen to the creation of the activity such as bindings for example.

The introAnimations(introAnimationDelayTime) method is called with the duration of the animation delay.

The log in fragment is created and displayed.

Listeners are passed to the visible fragment (login or register), to be able to pass from one to the other.

- private fun introAnimationDelayTime: Long

Here are instructions for the initial animations and the sound that accompanies them.

- override fun onCreateOptionsMenu(menu: Menu?): Boolean

Creates the menu of the App bar (the one at the top right). In this activity this menu has only the sound button that is visible. This button controls the sound status of the game and is saved in the player's shared preferences.

- fun onLanguageItemClick(item: MenuItem)

This method is triggered by clicking on the "Language" item in the App bar menu. In this activity it is not active.

- fun onHighscoresItemClick(item: MenuItem)

This method is triggered by clicking on the "Highscores" item in the App bar menu. In this activity it is not active.

- fun onSoundItemClick(item: MenuItem)

This method is triggered by clicking on the "Sound" item in the App bar menu.

The sound state is changed in the user's shared preferences, and the icon is also changed according to this state.

b. GameActivity

i. Attributes

- private lateinit var toggle: ActionBarDrawerToggle

Contains the status of the hidden menu (the one on the left) – visible or hidden, and therefore also of its icon in the left corner of the App bar (top).

- var appBarMenu: Menu =null

This contains the menu of the App bar (top right).

- var timer10: CountDownTimer = null

This is a CountDownTimer that is used when the player clicks a letter. It represents the number of points possible to earn on the next letter guessed correctly.

- var avatarAnimations:AvatarAnimations = null

This attribute will contain an instance of the AvatarAnimations class.

- var sounds: Sounds = null

The instance of the Sounds class that contains the method for playing audio files.

- var gameRound: GameRound = null

The instance of the GameRound class that controls several aspects of the current turn of the game.

- private val viewModel: MainViewModel

This is an instance of the MainViewModel.

ii. Methods

- override fun onCreate(savedInstanceState: Bundle?)

Here is everything that will happen to the creation of the activity such as bindings for example.

The initialiseBindings() method is called.

Objects such as GameRound, AvatarAnimations, and Sounds are instantiated.

The initialiseUi() method is called.

The list of avatars is created in the viewModel.

The list of languages is created in the viewModel.

A user object is created in the viewModel and its user name is displayed on the App bar.

Checks if the user already has a chosen avatar and calls the chooseAvatar(true) method if one is not chosen. A Boolean is passed as an argument to indicate if this is the first time that the user chooses the avatar. If this is the case, a call to the method to choose the language of the game will be made after the choice of avatar. If an avatar is already registered to the database, an avatar object is created in the viewModel and the initViewModel() method is called to initialize the observables. The correct values are also saved to avatarLastSelectedCheckbox and languageLastSelectedCheckbox in the viewModel.

The bindings of the hidden menu on the left are then made and the respective listeners on all items of the menu.

- private fun initialiseBindings()

Method that calls all binding initialization methods.

The initialiseKeyboardBinding() for the keyboard.

The initializes RoundBtnBinding() for the red button (play /continue).

The HintBtnBinding() initializes for the "Buy Help" button in the right-hand center menu.

The ExchangeBtnBinding() initializes for the "Exchange Resources" button in the right-hand center menu.

The initializesAbandonBtnBinding() for the "Abandon Game" button in the right center menu.

- private fun initialiseKeyboardBinding()

Makes the binding of all the keys of the keyboard and the respective listeners.

- private fun initialiseRoundBtnBinding()

Makes the binding of the red button (play / continue). The initiateNewRound() method is called during a click.

- private fun initialiseHintBtnBinding()

Binds the "Buy Help" button in the right-hand center menu. The showHelpMenu() method is called during a click.

- private fun initialiseExchangeBtnBinding ()

Binds the "Exchange Resources" button in the right-hand center menu. The showExchangeMenu() method is called during a click.

- private fun initialiseAbandonBtnBinding ()

Binds the "Abandon Game" button in the right center menu. The showAbandonGameRound () method is called when clicked.

- private fun initialiseUi()

Calls the hideAvatarGraphics() method of the avatarAnimations object to hide all avatar graphic elements in the layer-list.

Calls the hideAllAnimations() method to hide any animations that can be displayed. These animations are those displayed when the player misses or guesses a word.

Calls the hideHintBtn() method that hides the "Buy Help" button from the right center menu.

Calls the hideExchangeBtn() method that hides the "Exchange Resources" button from the right center menu.

Calls the hideAbandonBtn() method that hides the "Abandon Part " button in the right center menu.

Uses Glide to display the small standby animation while requests to APIs are made.

- private fun initViewModel()

Initializes observables in the viewModel.

When the word object of the viewModel changes, the validateRandomWord() method is called.

When activeAvatarMood of the viewModel changes, the updateAvatarMood() method is called.

- private fun updateAvatarMood(it: AvatarMoods)

Calls a method of the avatarAnimations object based on the value of activeAvatarMood.

- private fun showHelpMenu()

This method is called when clicking on the "Buy Help" button in the right-hand center menu.

An AlertDialog is displayed containing the prices of the help items that the player can purchase with his game resources.

By purchasing these items, the asset bar values are refreshed by calling the updateAssetBar() method. The Asset bar is the one that is below the App bar, with the values of the resources and is visible only during a game turn.

If a letter is purchased, the corresponding letter on the keyboard takes the green color of the correctly guessed letters, the value of displayedWord in the word object is refreshed, and the user's resources in the activeUser object of the viewModel.

If a definition is purchased, one of the existing definitions is chosen from the revealedDefinitions of the word object, and is passed to the revealDefinition() method that will display it.

If a body part is purchased, the letterMisses attribute of the gameRound object is refreshed and the updateAvatar method of the avatarAnimations object is called to refresh the displayed avatar.

- private fun revealDefinition(definition: String)

This method creates an AlertDialog that displays a definition of the hidden word.

- private fun showExchangeMenu()

This method is called when clicking on the "Exchange resources" button in the right-hand center menu.

An AlertDialog is displayed containing the prices of the exchanges that the player can make with his game resources.

By purchasing these items, the asset bar values are refreshed by calling the updateAssetBar() method. The Asset bar is the one that is below the App bar, with the values of the resources and is visible only during a game turn.

- private fun showAbandonGameRound()

This method is called when clicking on the "Abandon Game" button in the right center menu.

An AlertDialog is displayed containing a confirmation request.

If the player decides to abort the game turn, the abandonGameRound() method is called.

- private fun abandonGameRound()

Returns the default values for the attributes of the gameRound object.

Calls the resetKeyboard() method to return the keyboard to its default state.

Returns the default values for the attributes of the viewModel's activeUser object.

Cancels all timers of the avatarAnimations object and the timers of the price of letters object if there are active ones.

Calls the updateAssetBar() method to refresh the Asset bar.

Calls the hideKeyboard() method to hide the keyboard.

Calls the hideAssetBar() method to hide the Asset bar.

Calls the hideHintBtn() method to hide the "Buy Help" button from the right center menu.

Calls the hideExchangeBtn() method to hide the "Exchange Resources" button from the right center menu.

Calls the hideAbandonBtn() method to hide the "Abandon Part" button in the right center menu.

Calls the hideDisplayedWord() method to hide the word displayed below the keyboard (the one that displays the letters guessed and hidden by a *.

Calls the hideAllAnimations() method to hide any animations that can be displayed. These animations are those displayed when the player misses or guesses a word.

Calls the hideAvatarGraphics() method of the avatarAnimations object to hide all avatar graphic elements in the layer-list.

Returns the red button (play/continue) to its default state and calls the showNewRoundBtn() method to make it visible.

- override fun onOptionsItemSelected(item: MenuItem): Boolean

Method that handles the left hidden menu icon.

- private fun chooseLanguage()

This method is called when clicking on the "Change language" item in the hidden menu on the left.

An AlertDialog is displayed containing the list of languages that the player can choose for the words of the game.

The chosen language will then be refreshed in the user object of the viewModel, and the languageLastSelectedCheckbox value of the viewModel will also be refreshed. The activeLanguage attribute will also be updated in the viewModel.

Choosing the language is not possible during an active game round.

- private fun chooseAvatar(initialCheck: Boolean)

This method is called when clicking on the "Change avatar" item in the hidden menu on the left.

An AlertDialog is displayed containing the list of avatars that the player can choose to be displayed as hanged in the game.

The chosen avatar will then be refreshed in the user object of the viewModel, and the avatarLastSelectedCheckbox value of the viewModel will also be refreshed. The activeAvatar attribute will also be updated in the viewModel.

Choosing the avatar is not possible during an active game turn.

If the user chooses the avatar for the first time, the chooseLanguage() method is also called.

- private fun chooseUsername()

This method is called when clicking on the item "Changer the username" of the hidden menu on the left.

An AlertDialog is displayed that allows the player to choose a new username.

If the name already exists in the database or if the field is empty, an error message is displayed.

The new user name is saved in the user object of the viewModel and the App bar is refreshed with the new name.

- private fun choosePassword()

This method is called when clicking on the item "Changer the password" in the hidden menu on the left.

An AlertDialog is displayed that allows the player to choose a new password.

The new password is saved in the user object of the viewModel.

- private fun displayHelp()

This method is called when clicking on the "Game Rules" item in the hidden menu on the left.

An AlertDialog is displayed with the rules of the game.

- private fun deleteAccount()

This method is called when clicking on the item "Delete account" in the hidden menu on the left.

An AlertDialog is displayed that asks for confirmation.

If the player decides to delete their account, their database information is deleted and their shared preferences deleted, before starting the MainActivity activity with the LoginFragment fragment.

- private fun logout()

This method is called when clicking on the item "Log out" from the hidden menu on the left.

An AlertDialog is displayed that asks for confirmation.

If the player decides to log out, their shared preferences are cleared, before launching the MainActivity activity with the LoginFragment fragment.

- override fun onCreateOptionsMenu(menu: Menu?): Boolean

Creates the App bar menu with sound, highscore, and language icons all active.

The sound icon and the language icon change according to the user's choices.

Click on the active sound icon where deactivates the sound of the game.

Clicking on the language icon opens the window to choose the language of the game.

Clicking on the highscores icon opens the window to display the 5 top scores.

- fun onLanguageItemClick(item: MenuItem)

Manages a click on the language icon and calls the chooseLanguage() method.

- fun onHighscoresItemClick(item: MenuItem)

Manages a click on the highscores icon and calls the showHighscores() method.

- fun onSoundItemClick(item: MenuItem)

Manages a click on the sound icon. Changes the icon and changes the shared preferences.

- private fun showHighscores()

This method is called when clicking on the highscores icon of the App bar menu.

An AlertDialog using the HighscoresAdapter adapter is displayed with a list of the top 5 scores among all users in descending order.

- private fun keyboardPressed(pressed: String, buttonPressed: Button)

This method is called when the player clicks a letter.

If the guessed letter is correct, the number of points earned is calculated. Several factors can contribute to a cheaper price – the number of letters correctly guessed in a row for example. The wordGuessed() method is then called.

If the letter is not guessed, the player loses a life. If his lives are exhausted _activeAvatarMood is updated in the viewModel. If there are still any left, the updateAvatar() method of the avatarAnimations object is called to display the avatar with an added body part. The wordFailed() method is then called.

Asset bar values are refreshed by calling the updateAssetBar() method. The Asset bar is the one that is below the App bar, with the values of the resources and is visible only during a game turn.

The attributes of the gameRound object and the activeUser object are also refreshed.

By clicking on a letter, the prizeFadeOutCountdown() method is also called. This method is responsible by the timer of the possible price to guess each letter.

- private fun resetKeyboard()

This returns all letters on the keyboard to its default color states.

- private fun resetGameRound()

This returns all attributes of the gameRound object to its default values.

- fun initiateNewRound()

Calls the resetGameRound() method to reset the object to its default values.

Calls the hideNewRoundBtn() method to hide the red button (play /continue).

Calls the hideAllAnimations() method to hide any animations that can be displayed. These animations are those displayed when the player misses or guesses a word.

Calls the hideDisplayedWord() method to hide the word displayed below the keyboard (the one that displays the letters guessed and hidden by a *.

Returns _activeAvatarMood to its default value.

Calls the hideAvatarGraphics() method of the avatarAnimations object to hide all avatar graphic elements in the layer-list.

Makes the small standby animation visible while requests to APIs are made.

Calls the getRandomWordFr() or getRandomWordEn() method in the viewModel depending on the game language chosen by the player.

- private fun validateRandomWord(it: Word)

If no words are returned by the APIs, the initiateNewRound() method is recalled, otherwise the startNewRound() method is called.

- private fun startNewRound()

If this is a new game round, the values of the gameRound object are returned to its default values and the activeRound variable is returned true. The resetKeyboard() method is also called to return the

keyboard to its default state, as well as the prizeFadeOutCountdown() method, responsible by the timer for the possible price to guess each letter.

Makes the small standby animation visible while requests to APIs are made.

The showKeyboard() method is called to make the keyboard visible.

Asset bar values are refreshed by calling the updateAssetBar() method. The Asset bar is the one that is below the App bar, with the values of the resources and is visible only during a game turn.

The showAssetBar() method is called to make the asset bar visible.

The showHintBtn() method is called to make the "Buy Help" button in the right center menu visible.

The showExchangeBtn() method is called to make the "Exchange Resources" button visible in the right center menu.

The showAbandonBtn() method is called to make visible the "Abandon Part" button in the right center menu.

The showDisplayedWord() method is called to make visible the word displayed on the screen below the hangman.

- private fun wordGuessed()

Cancels the timer of the price of letters.

Returns the lettersGuessedConsecutively variable to its default value of 0.

Adds multiple bonuses under certain conditions and refreshes the user object in the viewModel.

Asset bar values are refreshed by calling the updateAssetBar() method. The Asset bar is the one that is below the App bar, with the values of the resources and is visible only during a game turn.

Calls the hideKeyboard() method to hide the keyboard.

Calls the resetKeyboard() method to return the keyboard to its default state.

The text of the red button (play/continue) is changed to "Continue" instead of "Play".

Calls the showNewRoundBtn() method to make the red button visible.

Calls the pickEndAnimation() method with "win" as the argument.

Calls the showEndDefinitions() method to display all available definitions of the hidden word.

- private fun wordFailed()

Cancels the timer of the price of letters.

Returns the lettersGuessedConsecutively variable to its default value of 0 and removes 1 from the player's lives.

Asset bar values are refreshed by calling the updateAssetBar() method. The Asset bar is the one that is below the App bar, with the values of the resources and is visible only during a game turn.

Replaces the displayedWord with the hiddenWord and then reveals the hidden word.

Calls the hideKeyboard() method to hide the keyboard.

Calls the resetKeyboard() method to return the keyboard to its default state.

The text of the red button (play/continue) is changed to "Continue" instead of "Play".

Calls the showNewRoundBtn() method to make the red button visible.

Calls the pickEndAnimation() method with "lose" as the argument.

Calls the showEndDefinitions() method to display all available definitions of the hidden word.

- private fun loseGameRound()

Returns the default values for the attributes of the gameRound object.

Calls the resetKeyboard() method to return the keyboard to its default state.

Returns the default values for the attributes of the viewModel's activeUser object.

Cancels all timers of the avatarAnimations object and the timers of the price of letters object if there are active ones.

Calls the updateAssetBar() method to refresh the Asset bar.

Calls the hideKeyboard() method to hide the keyboard.

Calls the hideAssetBar() method to hide the Asset bar.

Calls the hideHintBtn() method to hide the "Buy Help" button from the right center menu.

Calls the hideExchangeBtn() method to hide the "Exchange Resources" button from the right center menu.

Calls the hideAbandonBtn() method to hide the "Abandon Part" button in the right center menu.

Calls the hideDisplayedWord() method to hide the word displayed below the keyboard (the one that displays the letters guessed and hidden by a *.

Calls the hideAllAnimations() method to hide any animations that can be displayed. These animations are those displayed when the player misses or guesses a word.

Calls the hideAvatarGraphics() method of the avatarAnimations object to hide all avatar graphic elements in the layer-list.

Displays an AlertDialog with the player's final score.

If the score is higher than the player's personal highscore, it is saved in the table users of the database and the activeUser object is refreshed in the viewModel.

If the score is high enough it can also be saved in the table highscores of the database.

- private fun showEndDefinitions()

An AlertDialog using the DefinitionsAdapter adapter is displayed with a list of available definitions of the hidden word.

If the player has run out of lives, the loseGameRound() method is called.

Returns the red button (play/continue) to its default state and calls the showNewRoundBtn() method to make it visible.

- private fun pickEndAnimation(type: String)

Displays a final animation whether the player has guessed the hidden word or not.

- private fun showAssetBar()

Makes the asset bar visible.

- private fun hideAssetBar()

Hides the asset bar.

- private fun updateAssetBar()

Refreshes the asset bar values.

- private fun showKeyboard()

Makes the keyboard visible.

- private fun hideKeyboard()

Hides the keyboard.

- private fun showDisplayedWord()

Makes visible the word to be guessed below the hanged.

- private fun hideDisplayedWord()

Hides the word to guess below the hangman.

- private fun showHintBtn()

Makes visible the "Buy Help" button in the right-hand center menu.

- private fun hideHintBtn()

Hides the "Exchange Resources" button from the right-hand center menu.

- private fun showExchangeBtn()

Makes visible the "Exchange resources" button in the right-hand center menu.

- private fun hideExchangeBtn()

Hides the "Buy Help" button from the right-hand center menu.

- private fun showAbandonBtn()

Makes visible the "Abandon Part" button in the right-hand center menu.

- private fun hideAbandonBtn()

Hides the "Abandon Game" button from the right center menu.

- private fun showNewRoundBtn()

Makes the red button visible (play/continue).

- private fun hideNewRoundBtn()

Hides the red button (play/continue).

- private fun hideAllAnimations()

Hides all animations that can be displayed. These animations are those displayed when the player misses or guesses a word.

- private fun prizeFadeOutCountdown()

Triggers the timer10 for the countdown animation of the number of points possible to win when the player correctly guesses a letter.

3. Fragments

a. LoginFragment

This fragment contains everything related to a user's login to their account. It is located in the MainActivity activity.

i. Attributes

- private val viewModel: MainViewModel

This is an instance of the MainViewModel.

- var pwVisibilityState: Boolean = false

The visibility status of the password field. Allows the user to see or hide the password that is written.

- var sounds: Sounds = null

The instance of the Sounds class that contains the method for playing audio files.

ii. Methods

- override fun onCreateView(inflater: LayoutInflater, container: ViewGroup?, savedInstanceState: Bundle?): View

Here is everything thatwill happen to the creation of the fragment such as bindings for example.

- private fun login (view: View)

Here is the validation of the fields that the user must fill in to connect to his account.

If all fields are validated, the user's id and choice of "Remember me" during login are then saved in the shared preferences and the game starts by launching the GameActivity activity. If at least one of the fields is not validated, an error message is displayed.

b. RegisterFragment

This fragment contains everything related to the creation of a new user account. It is located in the MainActivity activity.

i. Attributes

- private val viewModel: MainViewModel

This is an instance of the MainViewModel.

- var pwVisibilityState: Boolean = false

The visibility status of the password field. Allows the user to see or hide the password that is written.

- var pw2VisibilityState: Boolean = false

The visibility status of the password confirmation field. Allows the user to see or hide the password that is written.

- var sounds: Sounds = null

The instance of the Sounds class that contains the method for playing audio files.

ii. Methods

- override fun onCreateView(inflater: LayoutInflater, container: ViewGroup?, savedInstanceState: Bundle?): View

Here is everything that will happen to the creation of the fragment such as bindings for example.

- private fun register (view: View)

Here is the validation of the fields that the user must fill in for the creation of an account.

If all fields are validated, a new user is created in the table users of the database and this fragment is replaced by the log in fragment. If at least one of the fields is not validated, an error message is displayed.

4. Classes

a. Avatar

This class contains information about the avatar and is used in the creation of avatars in the database table avatars.

i. Attributes

- private long id

Contains the id of the avatar in the database table avatars .

- private String head_shot

Contains a String with the identifier of the graphic file of the image used in the avatars selection list.

- private String head_src

Contains a String with the identifier of the avatar header graphic file.

- private String torso_src

Contains a String with the identifier of the graphic file of the torso of the avatar.

- private String left_arm_src

Contains a String with the graphic file identifier of the left arm of the avatar.

- private String right_arm_src

Contains a String with the identifier of the graphic file of the right arm of the avatar.

- private String left_leg_src

Contains a String with the identifier of the graphic file of the left leg of the avatar.

- private String right_leg_src

Contains a String with the identifier of the graphic file of the right leg of the avatar.

- private int eyesId

Contains the id of the eyes used by the avatar. This id refers to the table eyes in the database.

- private int mouthId

Contains the id of the mouth use dby the avatar. This id refers to the table mouths in the database.

- private int eyebrowsId

Contains the id of the eyebrows used by the avatar. This id refers to the table eyebrows in the database.

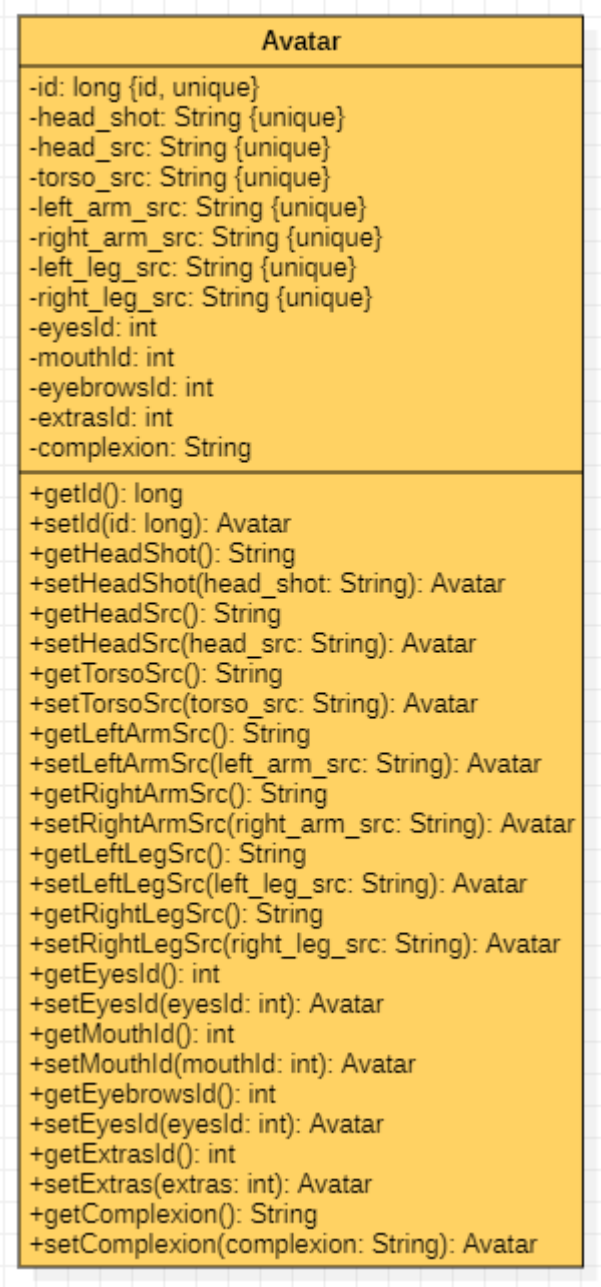
- private int extrasId

Contains the id of the extras (glasses) used by the avatar. This id refers to the table extras in the database.

- private String complexion

Contains a String with a complexion identifier that will determine the color of certain facial elements of the avatar.

ii. Class diagram



b. Extra

This class contains information about the extras (glasses) of the avatar and is used in the creation of the extras in the table extras of the database.

i. Attributes

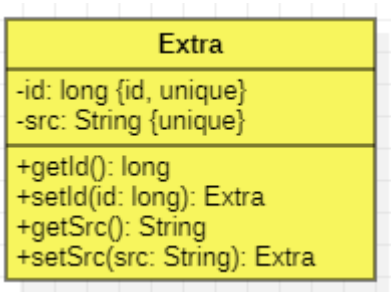
- private long id

Contains the id of the extras (glasses) in the table extras in the database.

- private String src

Contains a String with the graphic file identifier of the avatar extras in the table extras of the database.

ii. Class diagram



c. Eyebrows

This class contains information about the eyebrows of the avatar and is used in the creation of the eyebrows in the database's table eyebrows .

i. Attributes

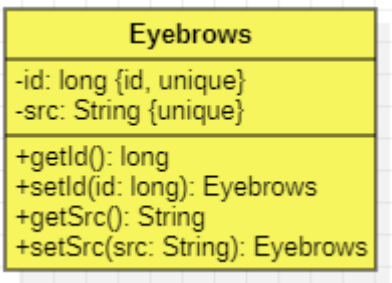
- private long id

Contains the id of the eyebrows in the table eyebrows in the database.

- private String src

Contains a String with the identifier of the graphic file of the eyebrows of the avatar in the table eyebrows of the database.

ii. Class diagram



d. Eyes

This class contains information about the avatar's eyes and is used in the creation of eyes in the database table eyes.

i. Attributes

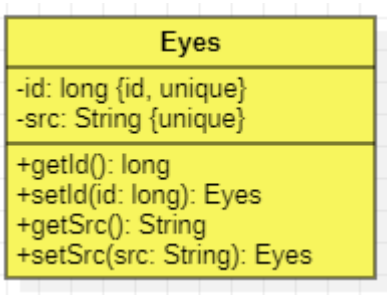
- private long id

Contains the id of the eyes in the table eyes in the database.

- private String src

Contains a String with the identifier of the graphic file of the avatar's eyes in the database table avatars .

ii. Class diagram



e. Highscore

This class contains information about the top 5 scores of the set of users and is used in creating a highscore in the database's table highscores .

i. Attributes

- private long id

Contains the id du highscorein the table highscores in the database.

- private int score

Contains the number of points of this highscore.

- private String date

Contains a String with the date on which the highscore is made.

- private int languageId

Contains the id of the language used in the game during which this highscore is made.

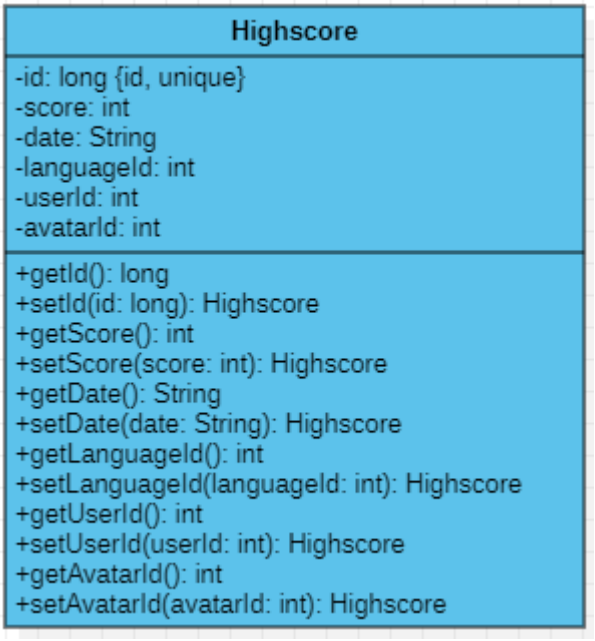
- private int userId

Contains the id of the user who made this highscore.

- private int avatarId

Contains the id of the avatar used in the game during which this highscore is made.

ii. Class diagram



f. Language

This class contains information about the game language chosen by the user and is used in creating a language in the database's table languages.

i. Attributes

- private long id

Contains the language id in the table languages in the database.

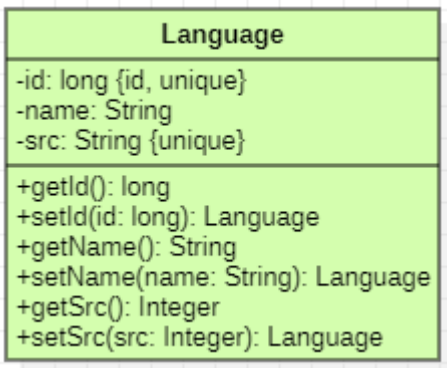
- private String name

Contains a String with the name of the game language chosen by the user.

- private String src

Contains a String with the identifier of the graphic file and the icon of the language chosen by the user.

ii. Class diagram



g. Mouth

This class contains information about the avatar mouth and is used in creating a mouth in the database table mouths.

i. Attributes

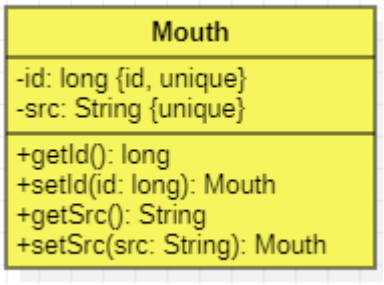
- private long id

Contains the mouth id in the table mouths in the database.

- private String src

Contains a String with the graphic file identifier of the avatar mouth in the database table avatars .

ii. Class diagram



h. Wear out

This class contains information about the logged-on user (the player) and is used in creating a user in the database's table users.

i. Attributes

- private final Context context

Contains the context.

- private long id

Contains the user id in the table users of the database.

- private String username

Contains the user name of the user in the table users of the database.

- private String password

Contains the user's password in the table users of the database.

- private int highscore

Contains the user's highscore in the table users of the database.

- private int languageId

Contains the user's language id in the table users of the database and refers to the languages table.

- private int avatarId

Contains the user's avatar id in the table users of the database and refers to the avatars table.

- private int coins = 0

Contains the number of pieces of the user in the table users of the database.

- private int banknotes = 0

Contains the number of posts of the user in the table users of the database.

- private int diamonds = 0

Contains the number of diamonds of the user in the database table users.

- private int lives = 5

Contains the number of user lives in the table users of the database.

- private int score = 0

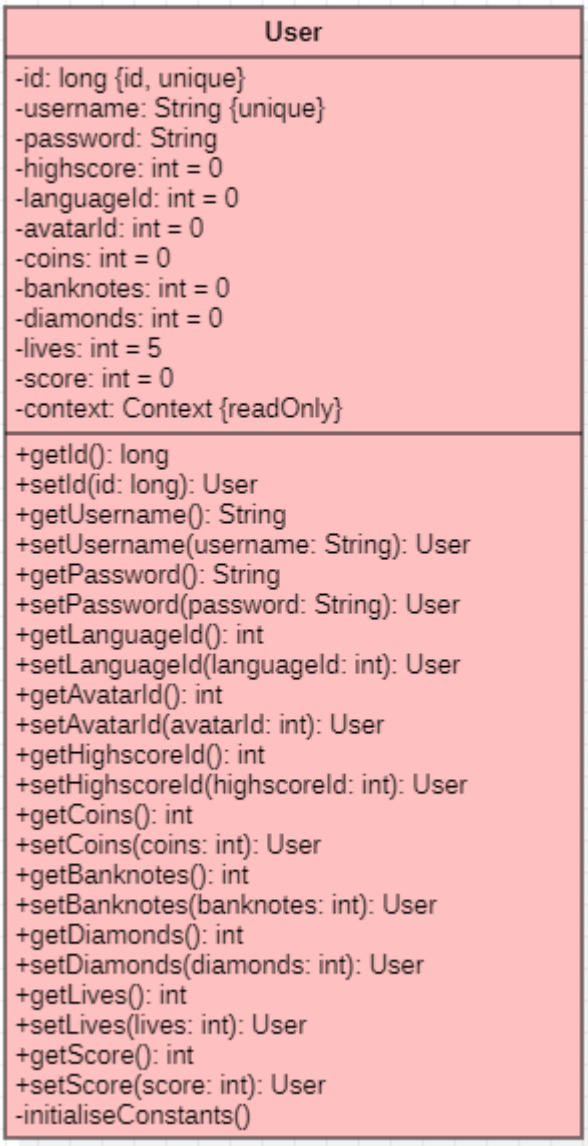
Contains the number of points of the user in the table users of the database.

ii. Methods

- private void initialiseConstants()

Initializes the attributes of the class with the values of the game constants present in an Array in the resources.

iii. Class diagram



i. AvatarAnimations

This class mainly contains methods that control the display of different parts of the avatar's body.

i. Attributes

- var blinkTimerInit: CountDownTimer = null

A CountDownTimer that controls the time when the avatar's eyes are opened.

- var blinkTimerEnd: CountDownTimer = null

A CountDownTimer that controls the time when the avatar's eyes are closed.

ii. Methods

- private fun getStringIdentifier(context: Context, name: String): Int

Returns the id of the ressource with its String ID.

- fun hideAvatarGraphics(context: Context)

Hides all avatar images that are in layerDrawable.

- private fun hideAvatarEyesGraphics(context: Context)

Hides all images of the avatar's eyes that are in the LayerDrawable.

- private fun hideAvatarEyebrowsGraphics(context: Context)

Hides all images of the avatar's eyebrows that are located in the LayerDrawable.

- private fun hideAvatarExtraGraphics(context: Context)

Hides all images of the avatar's extras (glasses) that are located in the LayerDrawable.

- private fun hideAvatarMouthGraphics(context: Context)

Hides all images of avatar plugs that are located in layerdrawable.

- suspend fun displayDeadAvatar(context: Context, viewModel: MainViewModel)

Displays graphics showing the dead avatar .

- suspend fun displayHappyFaceEyesForwardAvatar(context: Context, viewModel: MainViewModel)

Displays graphics showing an avatar face with specific eyes, mouth, and eyebrows (face by default).

- suspend fun displayHappyFaceBoredEyesUpLeftAvatar(context: Context, viewModel: MainViewModel)

Displays graphics showing an avatar face with specific eyes, mouth, and eyebrows (avatar who gets bored and looks at the top left)."

- suspend fun displayHappyFaceBoredEyesUpRightAvatar(context: Context, viewModel: MainViewModel)

Displays graphics showing an avatar face with specific eyes, mouth, and eyebrows (avatar who gets bored and looks at the top right).

- suspend fun displayHappyFaceBoredEyesDownLeftAvatar(context: Context, viewModel: MainViewModel)

Displays graphics showing an avatar face with specific eyes, mouth, and eyebrows (avatar who gets bored and looks down left).

- suspend fun displayHappyFaceBoredEyesDownRightAvatar(context: Context, viewModel: MainViewModel)

Displays graphics showing a face of the avatar with specific eyes, mouth, and eyebrows (avatar who gets bored and looks down to the right).

- suspend fun displayBlinkAvatar(context: Context, viewModel: MainViewModel)

Displays graphics showing the avatar's closed eyes.

- suspend fun displayHappyEyesAvatar(context: Context, viewModel: MainViewModel)

Displays graphics showing the avatar's default open eyes.

- suspend fun updateAvatar(context: Context, viewModel: MainViewModel)

Displays graphics showing which parts of the avatar body should be visible based on the number of letters missed by the player and the types of faces chosen.

- suspend fun updateAvatarEyes(context: Context, viewModel: MainViewModel)

Displays graphics showing the avatar's eyes that should be visible according to the chosen types.

- suspend fun updateAvatarEyebrows(context: Context, viewModel: MainViewModel)

Displays graphics showing the eyebrows of the avatar that should be visible according to the chosen types.

- suspend fun updateAvatarExtra(context: Context, viewModel: MainViewModel)

Displays graphics showing the extras (glasses) of the avatar that should be visible according to the types chosen.

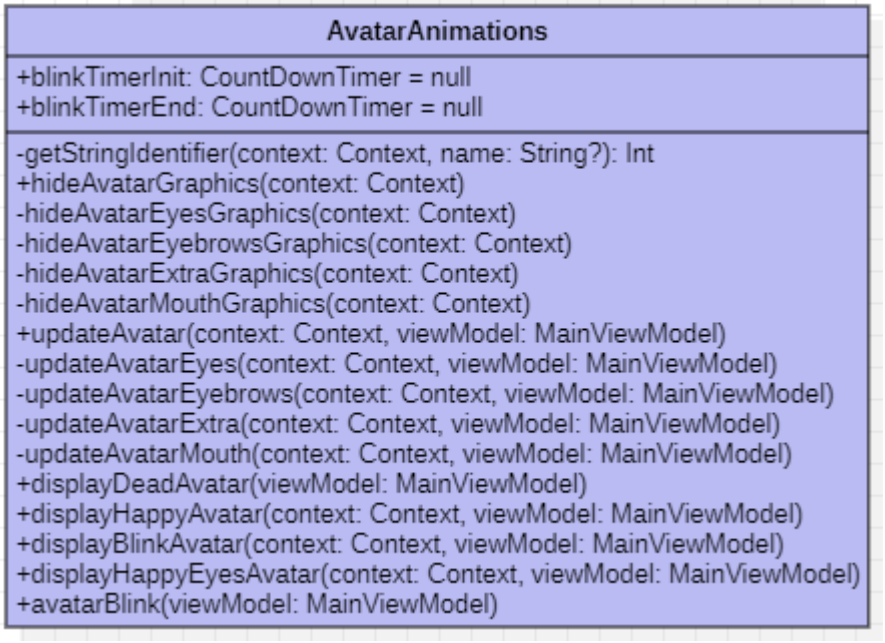
- suspend fun updateAvatarMouth(context: Context, viewModel: MainViewModel)

Displays graphics showing the mouths of the avatar that should be visible according to the chosen types.

- fun avatarBlink(viewModel: MainViewModel)

Launches the two CountDownTimers for the avatar's blinks.

iii. Class diagram



j. AvatarMoods

This class is an Enum that contains the keys corresponding to graphic states of the avatar's face on the screen

k. GameRound

This class contains several pieces of information about the current turn of the game.

i. Attributes

- private val context: Context

Contains the context.

- var activeRound: Boolean = false

A Boolean that is true only if a game is in progress.

- var letterMisses: Int = 0

An Int with the number of letters selected by the player but not in the hidden word (failed letters).

- var guessedLetters: Int = 0

An Int with the number of letters of the hidden word correctly guessed by the player.

- var lettersGuessedConsecutively: Int = 0

An Int with the number of letters of the word hidden correctly guessed in a row.

- var wordsGuessedConsecutively: Int = 0

An Int with the number of hidden words correctly guessed in a row.

- var wordsGuessedConsecutivelyNoFaults: Int = 0

An Int with the number of hidden words correctly guessed in a row without having missed any letters.

- var potentialPrize: Int = 10

An Int with the number of points possible to win if the player guesses a letter correctly. This price decreases every second to 0.

- var exchangeValues_Banknotes_price: Int = 0

An Int with the price in number of coins of a banknote.

- var exchangeValues_Diamonds_price: Int = 0

An Int with the price in number of tickets of a diamond.

- var exchangeValues_Lives_price: Int = 0

An Int with the price in number of diamonds of a lifetime.

- var helpValues_Letter_price: Int = 0

An Int with the price in number of pieces of a letter.

- var helpValues_Definition_price: Int = 0

An Int with the price in number of tickets of a definition.

- var helpValues_BodyPart_price: Int = 0

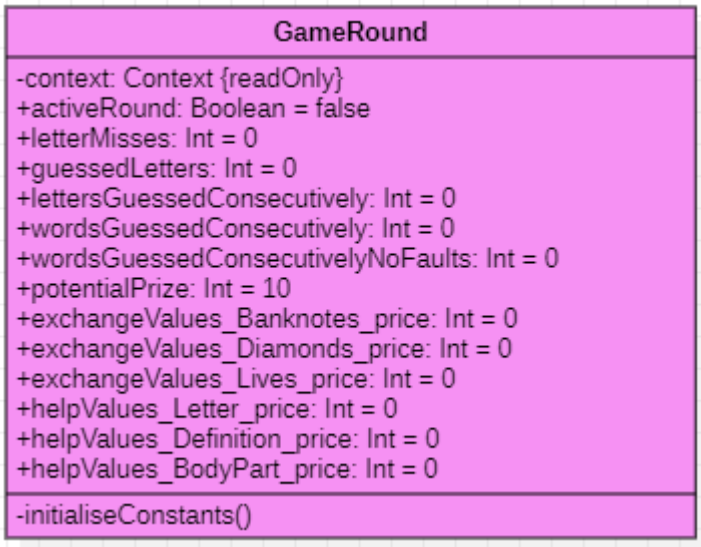
An Int with the price in number of diamonds to remove part of the body of the hanged avatar.

ii. Methods

- private fun initialiseConstants()

Initializes the constants of the game by giving values to the attributes concerned. The values of these constants are located in an Array in the resources.

iii. Class diagram



l. Sounds

This class contains a simple method for playing a sound.

i. Attributes

- private val context: Context

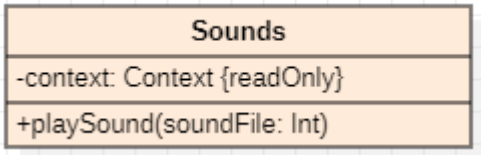
Contains the context.

ii. Methods

- fun playSound(soundFile: Int)

Launches a sound file from its id.

iii. Class diagram



m. Word

This class contains information about the random word that the player must guess.

i. Attributes

- var hiddenWord: String

This variable contains a String with the random word searched in one of the APIs.

- var definitions: ArrayList<String>

Contains a String ArrayList with definitions of the random word searched for in one of the APIs.

- private var language: String

Contains a String with the language of the random word searched for in one of the APIs.

- var displayedWord: String

Contains the word displayed on the screen below the hangman. This word can have hidden letters (displayed by a *).

- var revealedDefinitions= ArrayList<String>

Contains a String ArrayList with definitions that have not yet been revealed to the player.

ii. Methods

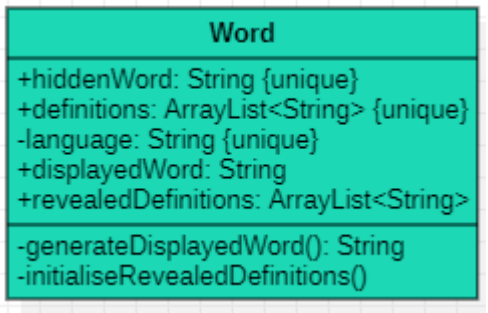
- private fun generateDisplayedWord(): String

This function generates a hidden word, represented by asterisks, and saves it in the displayedWord variable.

- private fun initialiseRevealedDefinitions()

Saves the definitions in the revealedDefinitions variable.

iii. Class diagram



5. Adapters

a. DefinitionsAdapter

This Adapter prepares the AlertDialog data with the definitions of the active word for display.

b. HighscoresAdapter

This Adapter prepares the AlertDialog data with the Top 5 scores for display.

6. ViewModel

a. MainViewModel

i. Attributes

- private var _activeUser: MutableLiveData<User> = null
- val activeUser: LiveData<User>

Used to back up an object of type Active User. The user object contains the information of the logged-on user.

- private var _randomWord: MutableLiveData<String> =""

Used to save a String with the random word that was searched in one of the APIs.

- private var_definitions = MutableLiveData<ArrayList<String>

Used to save a String ArrayList with the definitions of the random word that was searched in one of the APIs.

- var _word= MutableLiveData<Word>
- val word: LiveData<Word>

Used to back up an object of activeWord type . The word object contains information about the active word.

- var avatarList = MutableLiveData<ArrayList<Avatar>>

Used to save anAvatar ArrayList with the list of all possible avatars. The avatar object contains information about the avatar chosen by the user.

- var languageList = MutableLiveData<ArrayList<Language>>

Used to save an ArrayList of typeLanguage with the list of all possible languages.

- var _activeAvatar: MutableLiveData<Avatar> = null
- val activeAvatar: LiveData<Avatar>

Used to save an object of type ActiveAvatar .

- var_activeAvatarMood = MutableLiveData<AvatarMoods>
- val activeAvatarMood: LiveData<AvatarMoods>

Used to save an activeAvatarMoods Enum . This Enum contains the keys corresponding to graphic states of the avatar's face on the screen.

- var _activeLanguage: MutableLiveData<Language> = null
- val activeLanguage: LiveData<Language>

Used to back up an object of type ActiveLanguage . This is the language of the words and definitions that the game will display.

- var avatarLastSelectedCheckbox:MutableLiveData<Int> = 0

Used to save an Int with the index of the last checkbox checked in the menu where the user can choose an avatar.

- var languageLastSelectedCheckbox:MutableLiveData<Int> = 0

Used to save an Int with the index of the last checkbox checked in the menu where the user can choose a language.

ii. Methods

- fun findAllAvatars(context: Context): ArrayList<Avatar>

Looks for all items in the database table avatars and returns an Avatar Object Snatchlist.

- fun findAllLanguages(context: Context): ArrayList<Language>

Looks for all items in the table languages in the database and returns an ArrayList of objects of type Language.

- fun createUser(context: Context, id: Long, appBarMenu: Menu)

Creates the User object and saves it to the _activeUser variable of the MainViewModel.

Displays on the App bar (top menu) the icon of the language saved for this user in the table users of the database.

- fun updateUser(context: Context, id: Long, user: User)

Refreshes the User object that is saved in the _activeUser variable of the MainViewModel.

Displays on the App bar (top menu) the icon of the language saved for this user in the table users of the database.

- private fun generateNewWord(language: String)

Creates an object of type Word and saves it to the _word variable of the MainViewModel.

- fun updateDisplayedWord(guessedLetter: String, gameRound: GameRound): Boolean

Take the letter guessed by the player and look for it in the hidden word.

During this process, the characters in the hidden word are formatted by the prepareCharacter method so that they are recognized if they are special characters.

If the letter exists in the hidden word, the guessedLetters attribute of the gameRound object and the displayedWord attribute of the _word object are refreshed.

It returns a Boolean according to the letter be existing in the word hidden or not.

- fun prepareCharacter(character: String): String

Take a letter and return the capitalized equivalent without accents.

- fun findAllHighscores(context: Context): ArrayList<Highscore>

Looks for all items in the table highscores in the database and returns an ArrayList of objects of type Highscore.

- fun insertHighscore(highscore: Int, context: Context)

Inserts an item into the table highscores in the database.

- fun updateHighscore(context: Context, id: Long, highscore: Highscore)

Modifies an item in the table highscores in the database.

- suspend fun getAvatarsHeadshots(context: Context): ArrayList<String>

Searches for the database table avatars and returns a String ArrayList with the file names of the avatar image sources used in the menu to choose the avatar.

- suspend fun usernameExists(context: Context, username: String): Boolean

Searches for the table users in the database and returns a Boolean based on the user name that is already existing or not in the database.

- suspend fun findAvatarById(context: Context, id: Long): List<Avatar>

Take an id and look for the corresponding avatar in the database table avatars to return an Avatar Object List with all avatars in the database with that id (only one object is present in the List).

- suspend fun findLanguageById(context: Context, id: Long): List<Language>

Take an id and look for the corresponding language in the table languages of the database to return a List of objects of type Language with all the languages in the database with this id (only one object is present in the List).

- suspend fun findUserId(context: Context, username: String, password: String): Long

Take a user name and password as arguments and look for the corresponding user in the database table users to return a Long with that user's id.

- suspend fun findAllUsers(context: Context): List<User>

Searches for all items in the table users in the database and returns an Object List of type User.

- suspend fun findUserById(context: Context, id: Long): List<User>

Take an id and look for the corresponding user in the table users of the database to return a List of objects of type User with all users in the database with this id (only one object is present in the List).

- suspend fun findEyebrowsById(context: Context, id: Long): List<Eyebrows>

Take an id and look for the corresponding eyebrows in the table eyebrows of the database to return a List of objects of type Eyebrows with all the eyebrows in the database with this id (only one object is present in the List).

- suspend fun findEyesById(context: Context, id: Long): List<Eyes>

Take an id and look for the corresponding eyes in the table eyes of the database to return a List of objects of type Eyes with all eyes in the database with this id (only one object is present in the List).

- suspend fun findExtraById(context: Context, id: Long): List<Extra>

Take an id and look for the corresponding extras (glasses) in the table extras of the database to return an Extra Object List with all the extras in the database with this id (only one object is present in the List).

- suspend fun findMouthById(context: Context, id: Long): List<Mouth>

Take an id and look for the corresponding mouths in the table mouths of the database to return a List of objects of type Mouth with all the mouths in the database with this id (only one object is present in the List).

- fun insertUser(context: Context, user: User)

Inserts an item into the table users in the database.

- fun deleteUser(context: Context, id: Long)

Removes an item from the table users from the database.

- fun getRandomWordFr()

Calls the API to receive a random word in English.

The word is saved in the _randomWord variable and passes it as an argument to the getFrenchDefinition method.

- fun getRandomWordEn()

Calls the API to receive a random word in English.

The word is saved in the _randomWord variable and passes it as an argument to the getEnglishDefinition method.

- fun getFrenchDefinition(word: String)

Calls the API to receive the definitions of the word in English and saves them in the _definitions variable.

- fun getEnglishDefinition(word: String)

Calls the API to receive the English word definitions and saves them in the _definitions variable.

7. Apis

a. RandomWordAPI

API that returns a random word in English.

Link

b. API Dictionary English

API that returns a random word and definitions of a word in English.

Link

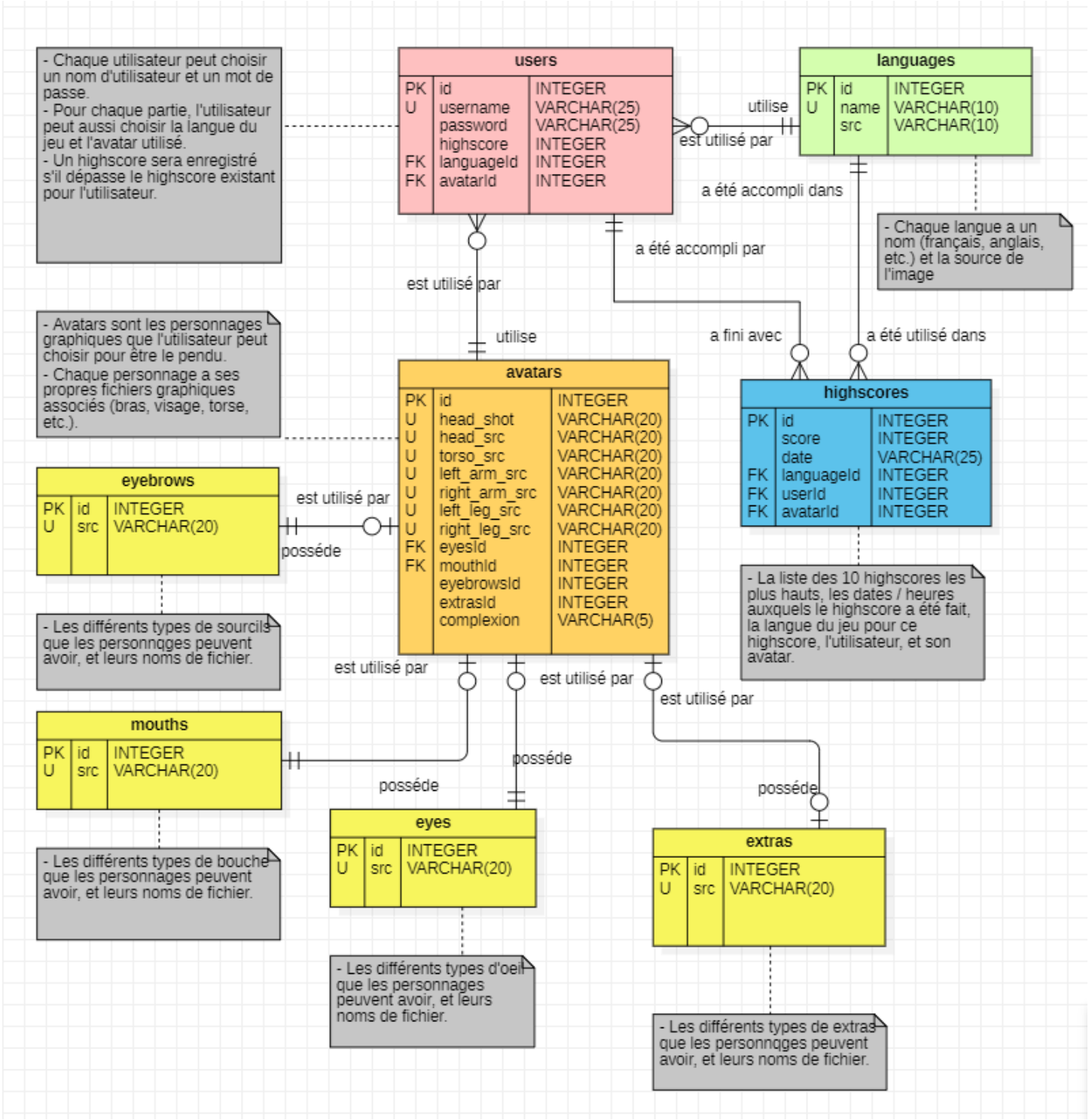
c. Free Dictionary API

API that returns definitions of a word in English.

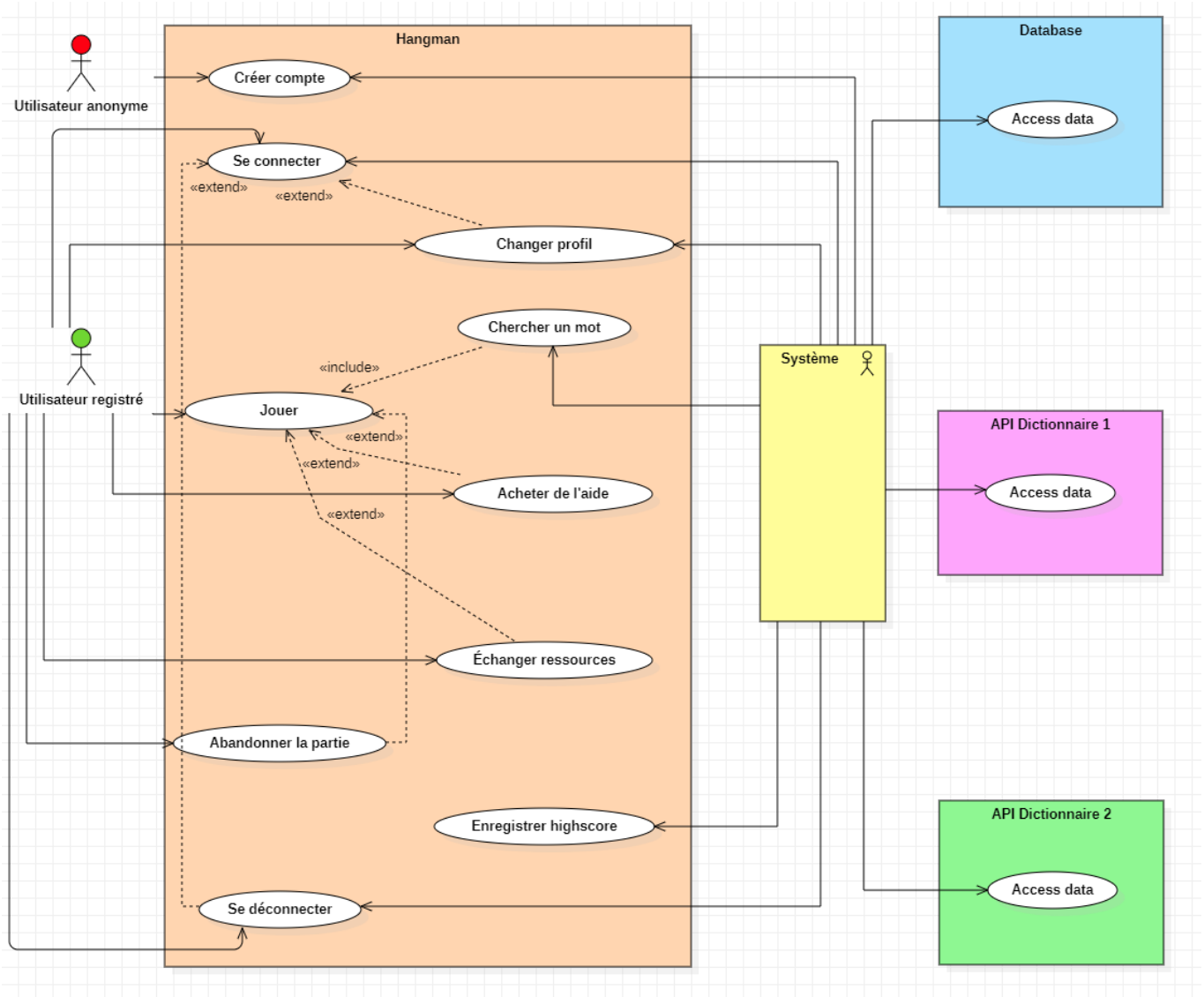
Link

8. Database

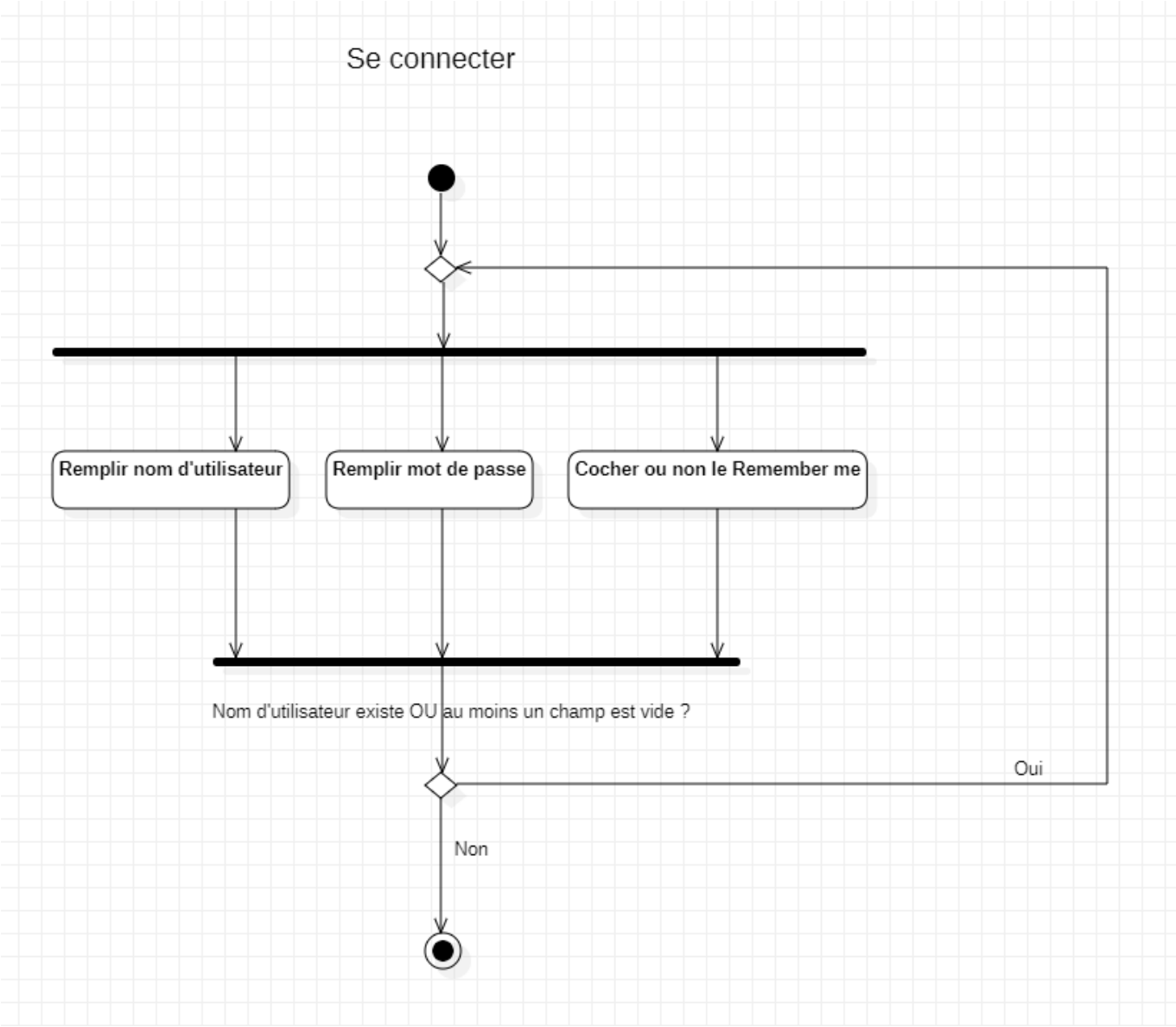
a. HangmanDb – Entity relationship diagram

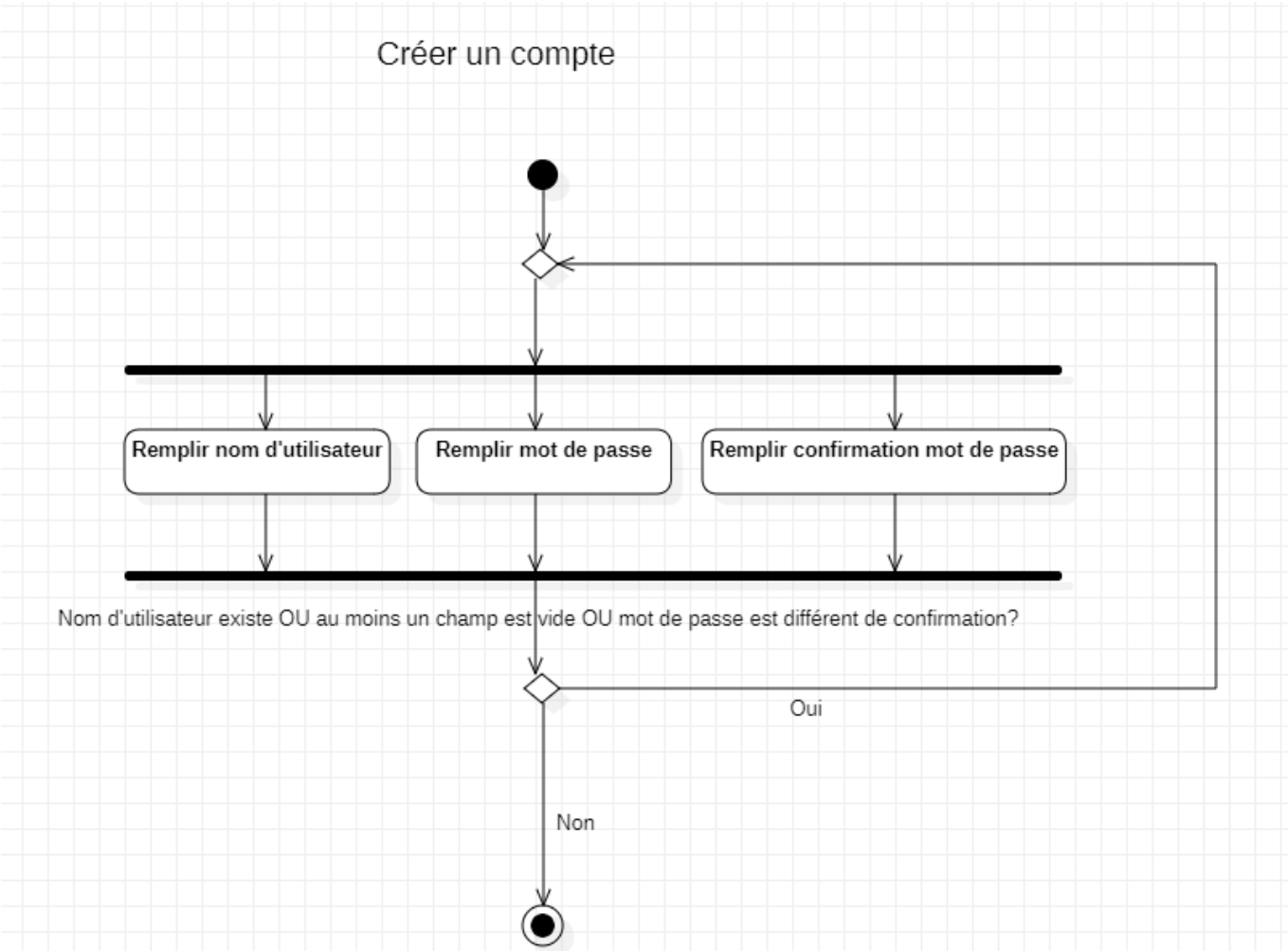


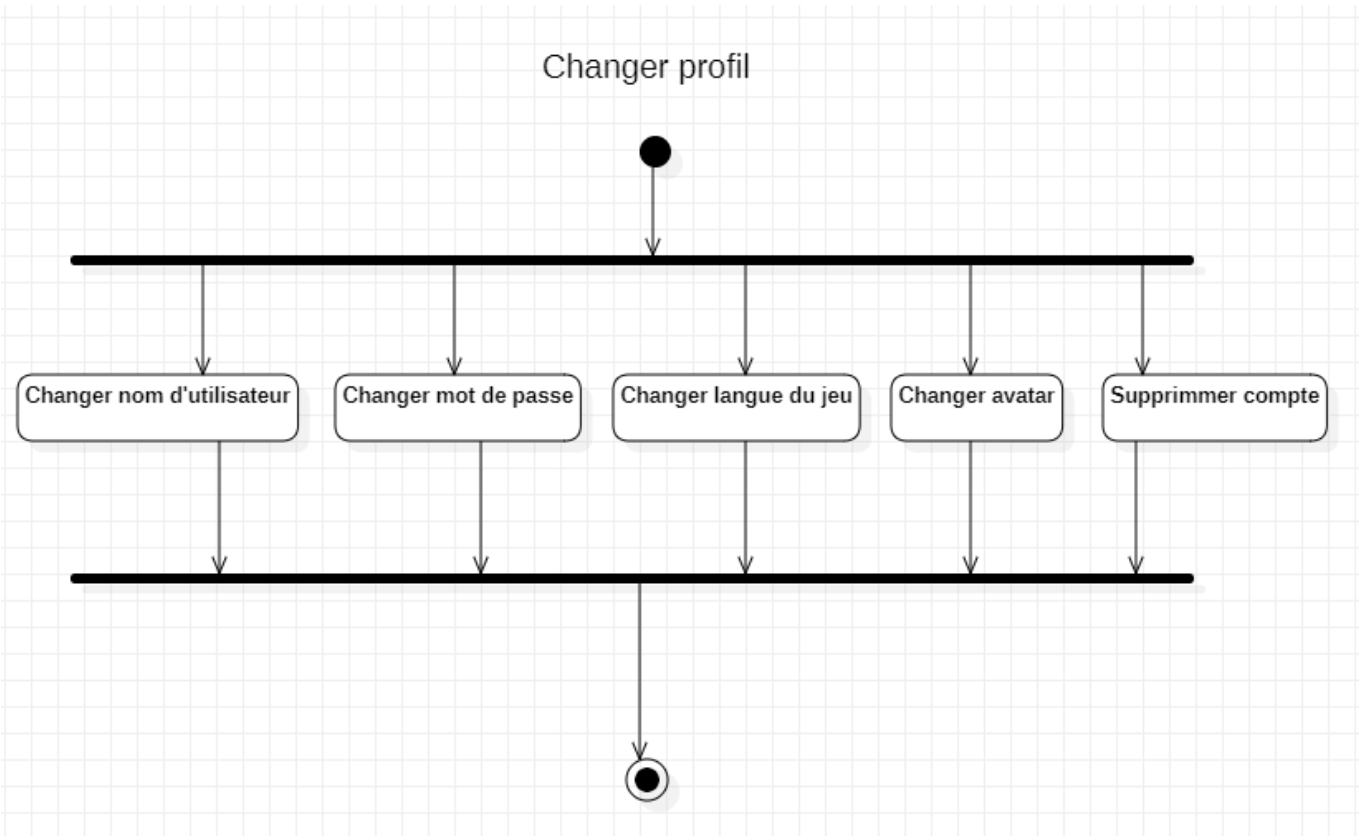
9. Use case diagram

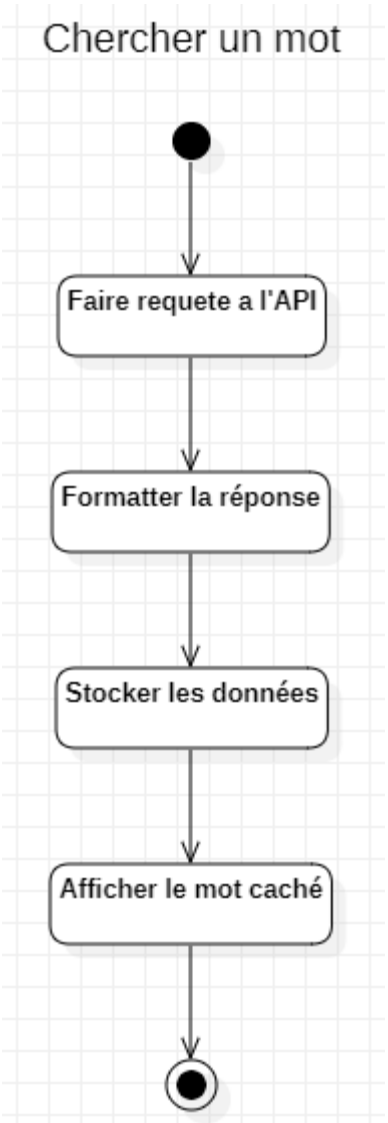


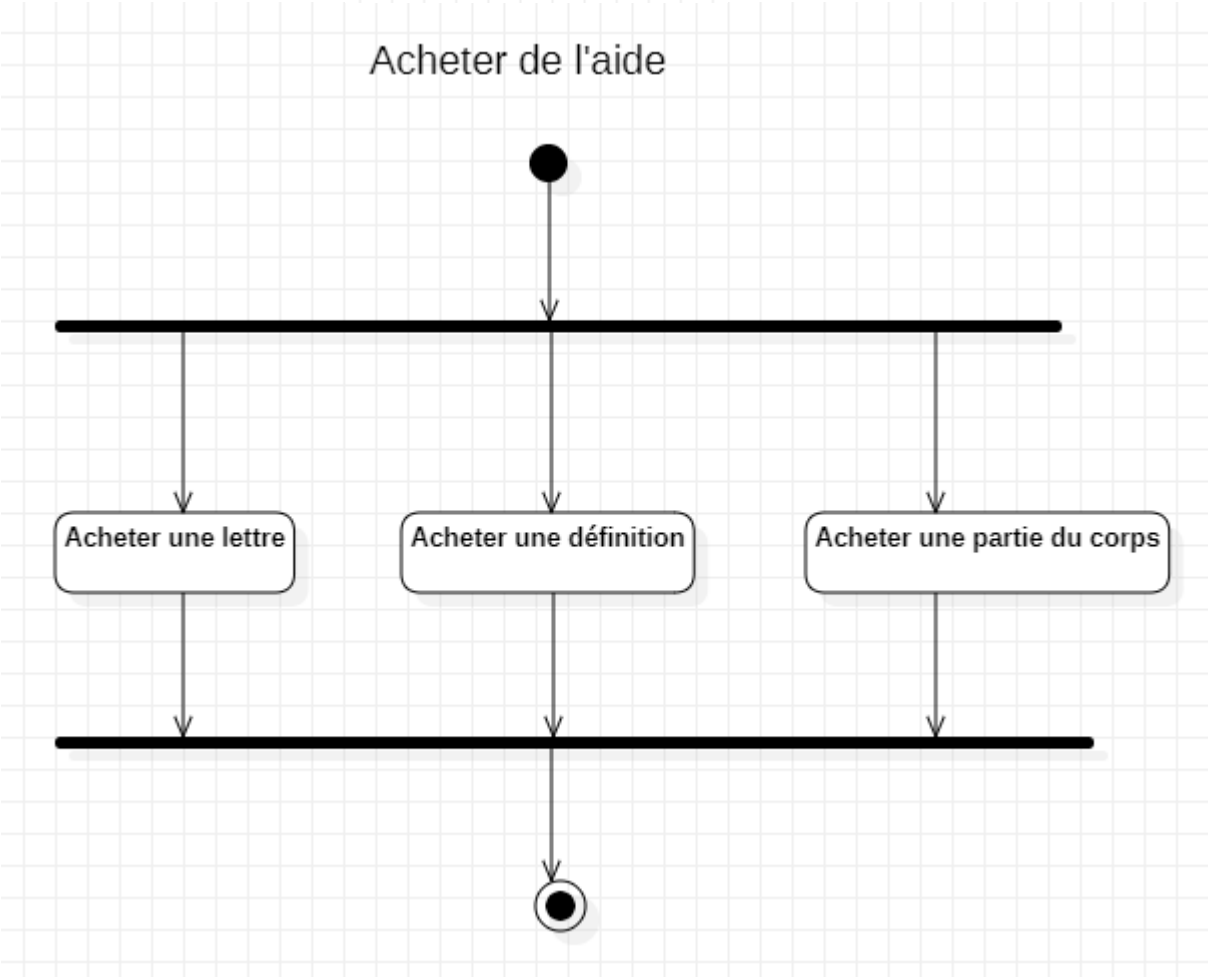
10. Activity diagrams

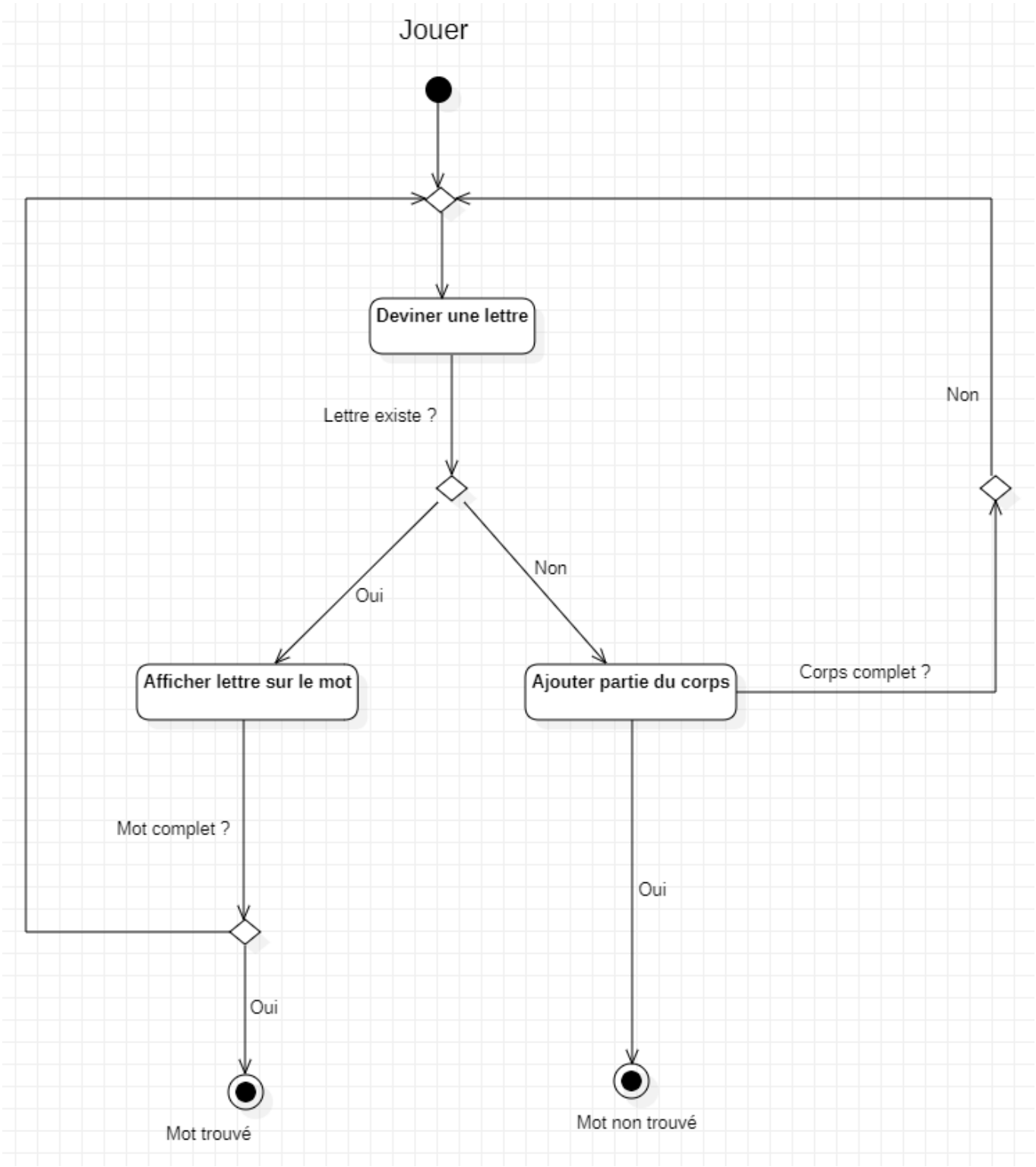


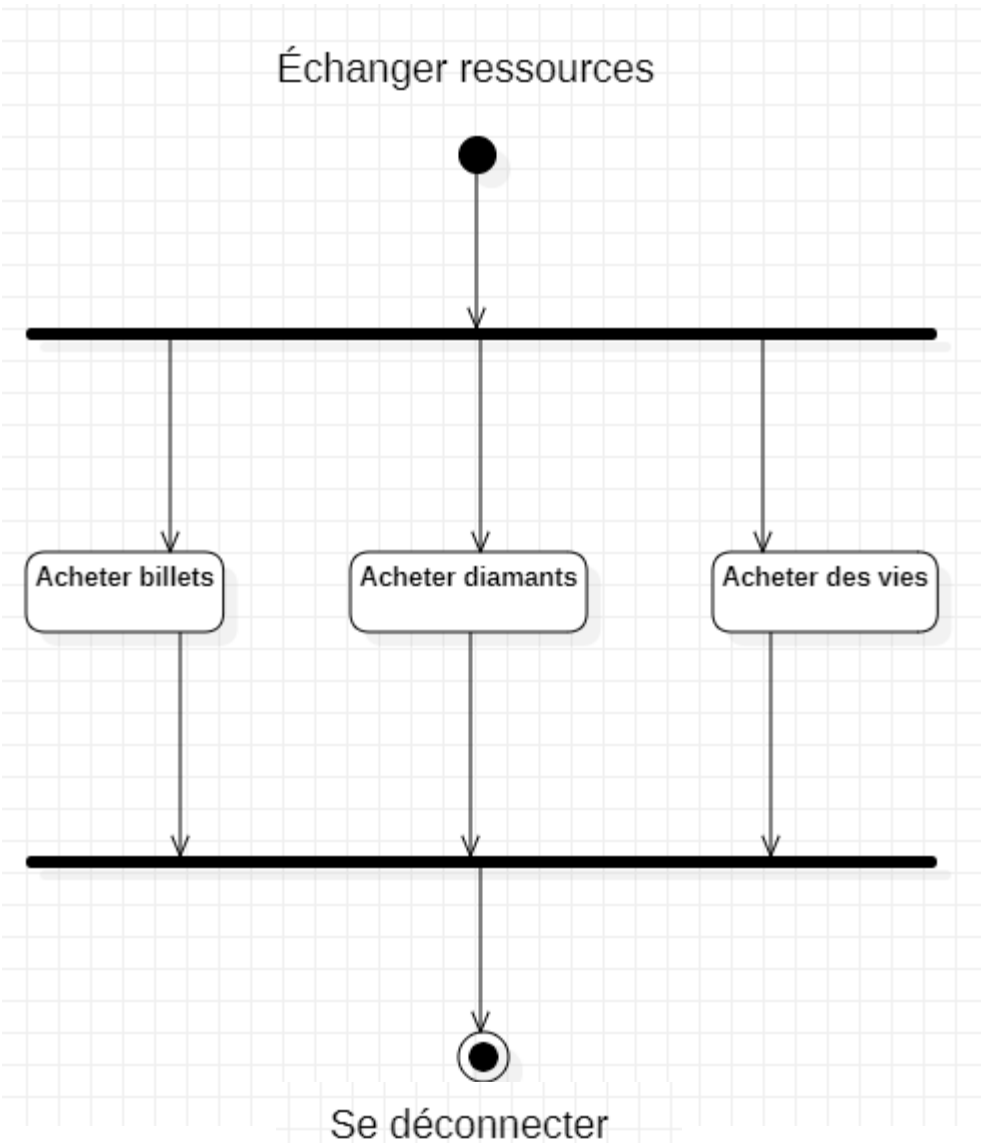

Créer un compte
Remplir nom d'utilisateur
Remplir mot de passe
Remplir confirmation mot de passe
Nom d'utilisateur existe OU au moins un champ est vide OU mot de passe est différent de confirmation?
Oui
Non

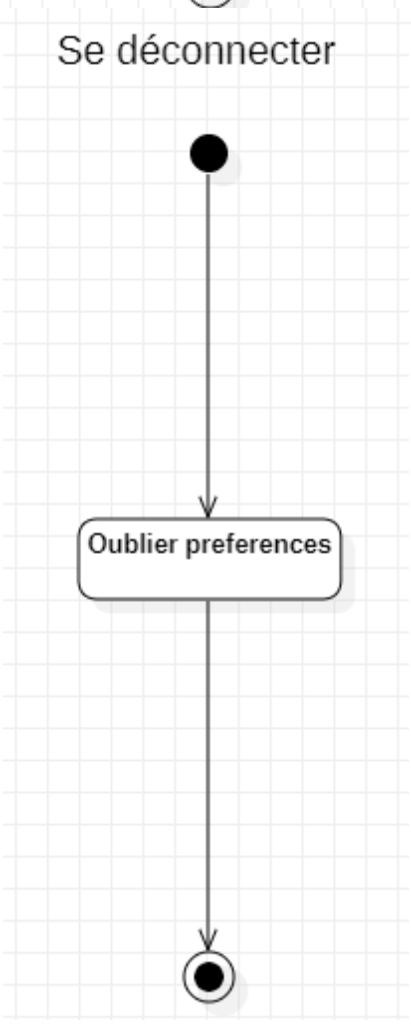

Changer profil
Changer nom d'utilisateur
Changer mot de passe
Changer langue du jeu
Changer avatar
Supprimmer compte

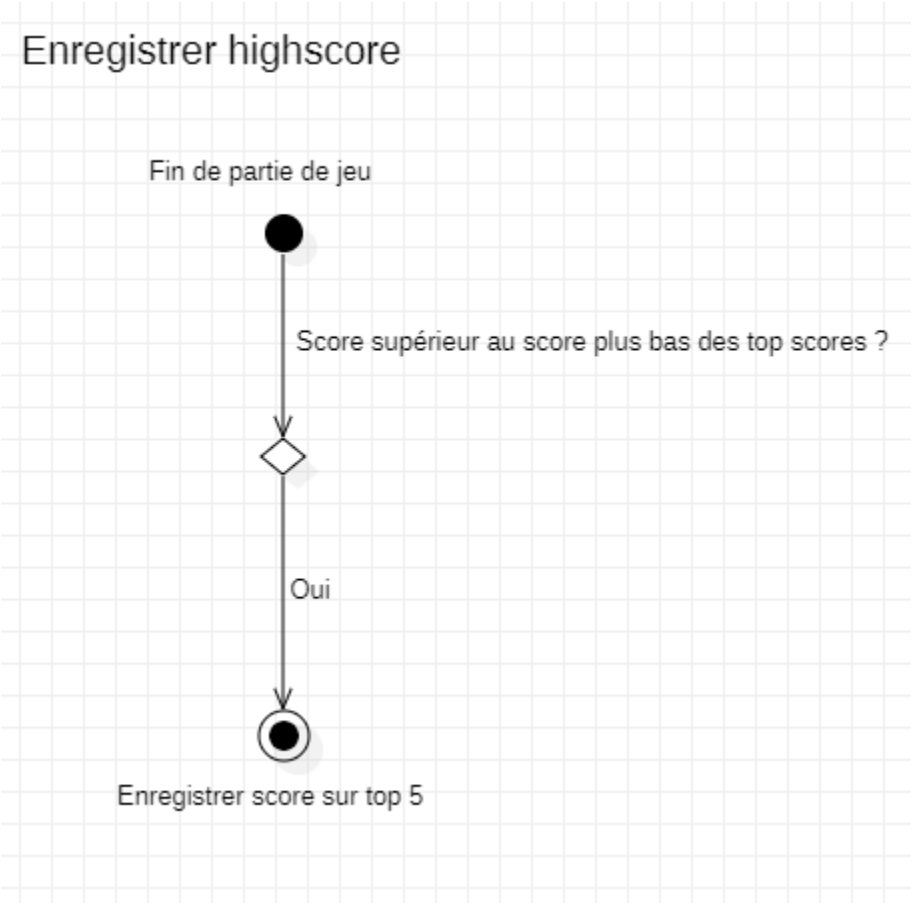

Chercher un mot
Faire requete a l'API
Formatter la réponse
Stocker les données
Afficher le mot caché

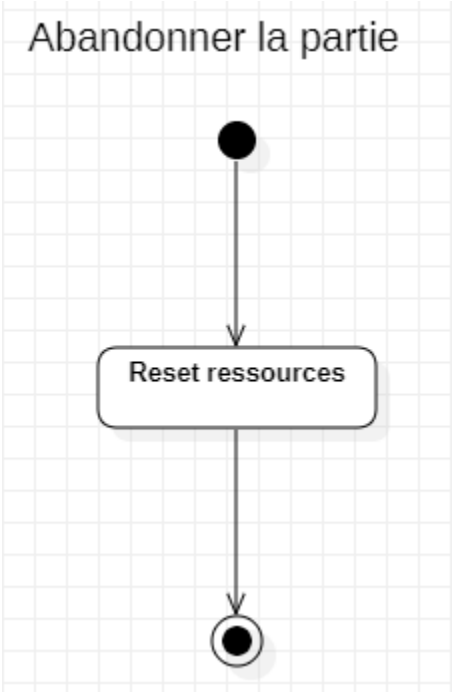

Acheter de l'aide
Acheter une lettre
Acheter une définition
Acheter une partie du corps


Jouer
Deviner une lettre
Lettre existe ?
Non
Oui
Non
Afficher lettre sur le mot
Ajouter partie du corps
Corps complet ?
Mot complet ?
Oui
Oui
Mot trouvé
Mot non trouvé


Échanger ressources
Acheter billets
Acheter diamants
Acheter des vies


Se déconnecter
Oublier preferences


Enregistrer highscore
Fin de partie de jeu
Score supérieur au score plus bas des top scores ?
Oui
Enregistrer score sur top 5


Abandonner la partie
Reset ressources