1. Indice

2. Activities

a. MainActivity

i. Attributs

- private var introAnimationDelayTime: Long = 250

Cet attribut contient le délai en millisecondes que l'animation initiale aura avant qu'elle s'initialise.

- var sounds: Sounds = null

L'instance de la classe Sounds qui contient la méthode pour jouer des fichiers audio.

ii. Méthodes

- override fun onCreate(savedInstanceState: Bundle?)

Ici se trouve tout ce qui va se passer à la création de l'activité comme les bindings par example.

La méthode introAnimations(introAnimationDelayTime) est appelée avec la durée du délai de l'animation.

Le fragment log in est créé et affiché.

Des listeners sont passés au fragment visible (login ou register), pour pouvoir passer de l'un à l'autre.

- private fun introAnimations(introAnimationDelayTime: Long)

Ici se trouvent des instructions pour les animations initiales et le son qui les accompagne.

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

Crée le menu de l'App bar (celui du haut à droite). Dans cette activité ce menu n'a que le bouton de son qui est visible. Ce bouton contrôle l'état du son du jeu et est sauvegardé dans les shared preferences du joueur.

- fun onLanguageItemClick(item: MenuItem)

Cette méthode est déclenchée par un click sur le item « Langue » du menu de l'App bar. Dans cette activité il n'est pas actif.

- fun onHighscoresItemClick(item: MenuItem)

Cette méthode est déclenchée par un click sur le item « Highscores » du menu de l'App bar. Dans cette activité il n'est pas actif.

- fun onSoundItemClick(item: MenuItem)

Cette méthode est déclenchée par un click sur le item « Son » du menu de l'App bar.

L'état de son est changé dans les shared preferences de l'utilisateur, et l'icône est aussi changé selon cet état.

b. GameActivity

i. Attributs

- private lateinit var toggle: ActionBarDrawerToggle

Contient l'état du menu caché (celui de gauche) – visible ou caché, et donc aussi de son icône dans le coin gauche de l'App bar (en haut).

- var appBarMenu: Menu = null

Ceci contient le menu de l'App bar (en haut à droite).

- var timer10: CountDownTimer = null

Ceci est un CountDownTimer qui est utilisé quand le joueur clique une lettre. Il représente le nombre de points possibles de gagner lors de la prochaine lettre devinée correctement.

- var avatarAnimations:AvatarAnimations = null

Cet attribut contiendra une instance de la classe AvatarAnimations.

- var sounds: Sounds = null

L'instance de la classe Sounds qui contient la méthode pour jouer des fichiers audio.

- var gameRound: GameRound = null

L'instance de la classe GameRound qui contrôle plusieurs aspects du tour courant du jeu.

- private val viewModel: MainViewModel

Ceci est une instance du MainViewModel.

ii. Méthodes

- override fun onCreate(savedInstanceState: Bundle?)

Ici se trouve tout ce qui va se passer à la création de l'activité comme les bindings par example.

La méthode initialiseBindings() est appelée.

Les objets de type GameRound, AvatarAnimations, et Sounds sont instantiés.

La méthode initialiseUi() est appelée.

La liste des avatars est créé dans le viewModel.

La liste des langues est créé dans le viewModel.

Un objet d'utilisateur est créé dans le viewModel et son nom d'utilisateur est affiché sur l'App bar.

Vérifie si l'utilisateur a déjà un avatar choisi et appelle la méthode chooseAvatars(true) si un n'est pas choisi. Un Boolean est passé en argument pour indiquer si c'est la première fois que l'utilisateur choisit l'avatar. Si c'est le cas, un appel à la méthode pour choisir la langue du jeu sera effectué après le choix d'avatar. Si un avatar est déjà enregistré sur la base de données, un objet avatar est créé dans le viewModel et la méthode initViewModel() est appelée pour initialiser les observables. Les valeurs correctes sont aussi enregistrées sur avatarLastSelectedCheckbox et languageLastSelectedCheckbox dans le viewModel.

Les bindings du menu caché de gauche sont alors faits et les respectifs listeners sur tous lest items du menu.

- private fun initialiseBindings()

Méthode qui appelle toutes les méthodes d'initialisation de bindings.

Le initialiseKeyboardBinding() pour le clavier.

Le initialiseRoundBtnBinding() pour le bouton rouge (jouer / continuer).

Le initialiseHintBtnBinding() pour le bouton « Acheter de l'aide » du menu central de droite.

Le initialiseExchangeBtnBinding() pour le bouton « Échanger ressources » du menu central de droite.

Le initialiseAbandonBtnBinding() pour le bouton « Abandonner la partie » du menu central de droite.

- private fun initialiseKeyboardBinding()

Fait le binding de toutes les touches du clavier et les respectifs listeners.

- private fun initialiseRoundBtnBinding()

Fait le binding du bouton rouge (jouer / continuer). La méthode initiateNewRound() est appelée lors d'un click.

- private fun initialiseHintBtnBinding()

Fait le binding du bouton « Acheter de l'aide » du menu central de droite. La méthode showHelpMenu() est appelée lors d'un click.

- private fun initialiseExchangeBtnBinding ()

Fait le binding du bouton « Échanger ressources » du menu central de droite. La méthode showExchangeMenu() est appelée lors d'un click.

- private fun initialiseAbandonBtnBinding ()

Fait le binding du bouton « Abandonner la partie » du menu central de droite. La méthode showAbandonGameRound () est appelée lors d'un click.

- private fun initialiseUi()

Appelle la méthode hideAvatarGraphics() de l'objet avatarAnimations pour cacher tous les éléments graphiques de l'avatar situés dans les layer-list.

Appelle la méthode hideAllAnimations() pour cacher toutes les animations qui puissent être affichées. Ces animations sont celles affichées quand le joueur rate ou devine un mot.

Appelle la méthode hideHintBtn() qui cache le bouton « Acheter de l'aide » du menu central de droite.

Appelle la méthode hideExchangeBtn() qui cache le bouton « Échanger ressources » du menu central de droite.

Appelle la méthode hideAbandonBtn() qui cache le bouton « Abandonner la partie » du menu central de droite.

Utilise Glide pour afficher la petite animation d'attente pendant que les requêtes aux APIs sont faites.

- private fun initViewModel()

Initialise les observables dans le viewModel.

Quand l'objet word du viewModel change, la méthode validateRandomWord() est appellée.

Quand activeAvatarMood du viewModel change, la méthode updateAvatarMood() est appellée.

- private fun updateAvatarMood(it: AvatarMoods)

Appelle une méthode de l'objet avatarAnimations selon la valeur d'activeAvatarMood.

- private fun showHelpMenu()

Cette méthode est appelée lors d'un click sur le bouton « Acheter de l'aide » du menu central de droite.

Un AlertDialog est affiché contenant les prix des items d'aide que le joueur peut acheter avec ses ressources de jeu.

En achetant ces items, les valeurs de l'Asset bar sont actualisées par l'appel de la méthode updateAssetBar(). L'Asset bar est celle qui est en dessous de l'App bar, avec les valeurs des ressources et est visible uniquement pendant un tour de jeu.

Si une lettre est achetée, la lettre correspondante du clavier prends la couleur verte des lettres correctement devinées, la valeur de displayedWord dans l'objet word est actualisée, ainsi que les ressources de l'utilisateur dans l'objet activeUser du viewModel.

Si une définition est achetée, une des définitions existantes est choisie parmi les revealedDefinitions de l'objet word, et est passé à la méthode revealDefinition() qui va l'afficher.

Si une partie du corps est achetée, l'attribut letterMisses de l'objet gameRound est actualisée et la méthode updateAvatar de l'objet avatarAnimations est appelée pour actualiser l'avatar affiché.

- private fun revealDefinition(definition: String)

Cette méthode crée une AlertDialog qui affiche une définition du mot caché.

- private fun showExchangeMenu()

Cette méthode est appelée lors d'un click sur le bouton « Échanger ressources » du menu central de droite.

Un AlertDialog est affiché contenant les prix des échanges que le joueur peut faire avec ses ressources de jeu.

En achetant ces items, les valeurs de l'Asset bar sont actualisées par l'appel de la méthode updateAssetBar(). L'Asset bar est celle qui est en dessous de l'App bar, avec les valeurs des ressources et est visible uniquement pendant un tour de jeu.

- private fun showAbandonGameRound()

Cette méthode est appelée lors d'un click sur le bouton « Abandonner la partie » du menu central de droite.

Un AlertDialog est affiché contenant une demande de confirmation.

Si le joueur décide d'abandonner le tour de jeu, la méthode abandonGameRound() est appelée.

- private fun abandonGameRound()

Retourne les valeurs par défaut des attributs de l'objet gameRound.

Appelle la méthode resetKeyboard() pour retourner le clavier à son état de défaut.

Retourne les valeurs par défaut des attributs de l'objet activeUser du viewModel.

Annule tous les timers de l'objet avatarAnimations et celui du prix des lettres s'il en existent actifs.

Appelle la méthode updateAssetBar() pour actualiser l'Asset bar.

Appelle la méthode hideKeyboard() pour cacher le clavier.

Appelle la méthode hideAssetBar() pour cacher l'Asset bar.

Appelle la méthode hideHintBtn() pour cacher le bouton « Acheter de l'aide » du menu central de droite.

Appelle la méthode hideExchangeBtn() pour cacher le bouton « Échanger ressources » du menu central de droite.

Appelle la méthode hideAbandonBtn() pour cacher le bouton « Abandonner la partie » du menu central de droite.

Appelle la méthode hideDisplayedWord() pour cacher le mot affiché en dessous du clavier (celui qui affiche les lettres devinées et cachées par un *****.

Appelle la méthode hideAllAnimations() pour cacher toutes les animations qui puissent être affichées. Ces animations sont celles affichées quand le joueur rate ou devine un mot.

Appelle la méthode hideAvatarGraphics() de l'objet avatarAnimations pour cacher tous les éléments graphiques de l'avatar situés dans les layer-list.

Retourne le bouton rouge (jouer/ continuer) à son état par défaut et appelle la méthode showNewRoundBtn() pour le rendre visible.

- override fun onOptionsItemSelected(item: MenuItem): Boolean

Méthode qui gère l'icône du menu caché de gauche.

- private fun chooseLanguage()

Cette méthode est appelée lors d'un click sur l'item « Changer la langue » du menu caché de gauche.

Un AlertDialog est affiché contenant la liste des langues que le joueur peut choisir pour les mots du jeu.

La langue choisie sera alors actualisée dans l'objet user du viewModel, et la valeur de languageLastSelectedCheckbox du viewModel sera aussi actualisé. L'attribut activeLanguage sera aussi actualisé dans le viewModel.

Choisir la langue n'est pas possible pendant un tour de jeu actif.

- private fun chooseAvatar()

Cette méthode est appelée lors d'un click sur l'item « Changer avatar » du menu caché de gauche.

Un AlertDialog est affiché contenant la liste des avatars que le joueur peut choisir pour être affiché comme pendu dans le jeu.

L'avatar choisi sera alors actualisé dans l'objet user du viewModel, et la valeur de avatarLastSelectedCheckbox du viewModel sera aussi actualisé. L'attribut activeAvatar sera aussi actualisé dans le viewModel.

Choisir l'avatar n'est pas possible pendant un tour de jeu actif.

Si l'utilisateur choisit l'avatar pour la première fois, la méthode chooseLanguage() est aussi appelée.

- private fun chooseUsername()

Cette méthode est appelée lors d'un click sur l'item « Changer le nom d'utilisateur » du menu caché de gauche.

Un AlertDialog est affiché qui permet au joueur choisir un nouveau nom d'utilisateur.

Si le nom existe déjà dans la base de données où si le champ est vide, un message d'erreur est affiché.

Le nouveau nom d'utilisateur est enregistré dans l'objet user du viewModel et l'App bar est actualisée avec le nouveau nom.

- private fun choosePassword()

Cette méthode est appelée lors d'un click sur l'item « Changer le mot de passe » du menu caché de gauche.

Un AlertDialog est affiché qui permet au joueur choisir un nouveau mot de passe.

Le nouveau mot de passe est enregistré dans l'objet user du viewModel.

- private fun displayHelp()

Cette méthode est appelée lors d'un click sur l'item « Règles du jeu » du menu caché de gauche.

Un AlertDialog est affiché avec les règles du jeu.

- private fun deleteAccount()

Cette méthode est appelée lors d'un click sur l'item « Supprimer le compte » du menu caché de gauche.

Un AlertDialog est affiché qui demande une confirmation.

Si le joueur décide de supprimer son compte, ses informations sur la base de données sont supprimées et ses shared preferences effacées, avant de lancer l'activité MainActivity avec le fragment LoginFragment.

- private fun logout()

Cette méthode est appelée lors d'un click sur l'item « Déconnectez-vous » du menu caché de gauche.

Un AlertDialog est affiché qui demande une confirmation.

Si le joueur décide de se déconnecter, ses shared preferences sont effacées, avant de lancer l'activité MainActivity avec le fragment LoginFragment.

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

Crée le menu de l'App bar avec les icônes de son, highscore, et langue tous actifs.

L'icône de son et celui de la langue changent selon les choix de l'utilisateur.

Cliquer sur l'icône de son active où désactive le son du jeu.

Cliquer sur l'icône de langue ouvre la fenêtre pour choisir la langue du jeu.

Cliquer sur l'icône des highscores ouvre la fenêtre pour afficher les 5 top scores.

- fun onLanguageItemClick(item: MenuItem)

Gère un click sur l'icône de langue et appelle la méthode chooseLanguage().

- fun onHighscoresItemClick(item: MenuItem)

Gère un click sur l'icône des highscores et appelle la méthode showHighscores().

- fun onSoundItemClick(item: MenuItem)

Gère un click sur l'icône de son. Change l'icône et change les shared preferences.

- private fun showHighscores()

Cette méthode est appelée lors d'un click sur l'icône des highscores du menu de l'App bar.

Un AlertDialog utilisant l'adaptateur HighscoresAdapter est affiché avec une liste des 5 meilleurs scores parmi tous les utilisateurs par ordre décroissante.

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

Cette méthode est appelée lorsque le joueur clique une lettre.

Si la lettre devinée est bonne, le nombre de points gagnés est calculé. Plusieurs facteurs peuvent contribuer pour un prix plus avantageux – le nombre de lettres correctement devinées d'affilé par exemple. La méthode wordGuessed() est alors appelée.

Si la lettre n'est pas devinée, le joueur perd une vie. Si ses vies sont épuisées _activeAvatarMood est actualisé dans le viewModel. S'il en restent encore, la méthode updateAvatar() de l'objet avatarAnimations est appelée pour afficher l'avatar avec une partie de son corps ajoutée. La méthode wordFailed() est alors appelée.

Les valeurs de l'Asset bar sont actualisées par l'appel de la méthode updateAssetBar(). L'Asset bar est celle qui est en dessous de l'App bar, avec les valeurs des ressources et est visible uniquement pendant un tour de jeu.

Les attributs de l'objet gameRound et de l'objet activeUser sont aussi actualisées.

En cliquant sur une lettre, la méthode prizeFadeOutCountdown() est aussi appelée. Cette méthode est responsable par le timer du prix possible pour deviner chaque lettre.

- private fun resetKeyboard()

Ceci retourne toutes les lettres du clavier à ses états de couleur par défaut.

- private fun resetGameRound()

Ceci retourne touts les attributs de l'objet gameRound à ses valeurs par défaut.

- fun initiateNewRound()

Appelle la méthode resetGameRound() pour réinitialiser l'objet avec ses valeurs par défaut.

Appelle la méthode hideNewRoundBtn() pour cacher le bouton rouge (jouer / continuer).

Appelle la méthode hideAllAnimations() pour cacher toutes les animations qui puissent être affichées. Ces animations sont celles affichées quand le joueur rate ou devine un mot.

Appelle la méthode hideDisplayedWord() pour cacher le mot affiché en dessous du clavier (celui qui affiche les lettres devinées et cachées par un *.

Retourne _activeAvatarMood à sa valeur par défaut.

Appelle la méthode hideAvatarGraphics() de l'objet avatarAnimations pour cacher tous les éléments graphiques de l'avatar situés dans les layer-list.

Rend visible la petite animation d'attente pendant que les requêtes aux APIs sont faites.

Appelle la méthode getRandomWordFr() ou getRandomWordEn() dans le viewModel selon la langue de jeu choisie par le joueur.

- private fun validateRandomWord(it: Word)

Si aucun mot n'est retourné par les APIs, la méthode initiateNewRound() est rappelé, sinon la méthode startNewRound() est appelée.

- private fun startNewRound()

S'il s'agit d'un nouveau game round, les valeurs de l'objet gameRound sont retournées à ses valeurs par défaut et la variable activeRound est rendu true. La méthode resetKeyboard() est

aussi appelée pour retourner le clavier à son état de défaut, ainsi que la méthode prizeFadeOutCountdown(), responsable par le timer du prix possible pour deviner chaque lettre.

Rend visible la petite animation d'attente pendant que les requêtes aux APIs sont faites.

La méthode showKeyboard() est appelée pour rendre visible le clavier.

Les valeurs de l'Asset bar sont actualisées par l'appel de la méthode updateAssetBar(). L'Asset bar est celle qui est en dessous de l'App bar, avec les valeurs des ressources et est visible uniquement pendant un tour de jeu.

La méthode showAssetBar() est appelée pour rendre visible l'Asset bar.

La méthode showHintBtn() est appelée pour rendre visible le bouton « Acheter de l'aide » du menu central de droite.

La méthode showExchangeBtn() est appelée pour rendre visible le bouton « Échanger ressources » du menu central de droite.

La méthode showAbandonBtn() est appelée pour rendre visible le bouton « Abandonner la partie » du menu central de droite.

La méthode showDisplayedWord() est appelée pour rendre visible le mot affiché sur l'écran en dessous du pendu.

- private fun wordGuessed()

Annule le timer du prix des lettres.

Retourne la variable lettersGuessedConsecutively à sa valeur par défaut de 0.

Ajoute plusieurs bonus selon certaines conditions et actualise l'objet user dans le viewModel.

Les valeurs de l'Asset bar sont actualisées par l'appel de la méthode updateAssetBar(). L'Asset bar est celle qui est en dessous de l'App bar, avec les valeurs des ressources et est visible uniquement pendant un tour de jeu.

Appelle la méthode hideKeyboard() pour cacher le clavier.

Appelle la méthode resetKeyboard() pour retourner le clavier à son état de défaut.

Le texte du bouton rouge (jouer / continuer) est changé par « Continuer » au lieu de « Jouer ».

Appelle la méthode showNewRoundBtn() pour rendre visible le bouton rouge.

Appelle la méthode pickEndAnimation() en passant « win » comme argument.

Appelle la méthode showEndDefinitions() pour afficher toutes les définitions disponibles du mot caché.

- private fun wordFailed()

Annule le timer du prix des lettres.

Retourne la variable lettersGuessedConsecutively à sa valeur par défaut de 0 et enlève 1 aux vies du joueur.

Les valeurs de l'Asset bar sont actualisées par l'appel de la méthode updateAssetBar(). L'Asset bar est celle qui est en dessous de l'App bar, avec les valeurs des ressources et est visible uniquement pendant un tour de jeu.

Remplace la displayedWord par l'hiddenWord et révéle alors le mot caché.

Appelle la méthode hideKeyboard() pour cacher le clavier.

Appelle la méthode resetKeyboard() pour retourner le clavier à son état de défaut.

Le texte du bouton rouge (jouer / continuer) est changé par « Continuer » au lieu de « Jouer ».

Appelle la méthode showNewRoundBtn() pour rendre visible le bouton rouge.

Appelle la méthode pickEndAnimation() en passant « lose » comme argument.

Appelle la méthode showEndDefinitions() pour afficher toutes les définitions disponibles du mot caché.

- private fun loseGameRound()

Retourne les valeurs par défaut des attributs de l'objet gameRound.

Appelle la méthode resetKeyboard() pour retourner le clavier à son état de défaut.

Retourne les valeurs par défaut des attributs de l'objet activeUser du viewModel.

Annule tous les timers de l'objet avatarAnimations et celui du prix des lettres s'il en existent actifs.

Appelle la méthode updateAssetBar() pour actualiser l'Asset bar.

Appelle la méthode hideKeyboard() pour cacher le clavier.

Appelle la méthode hideAssetBar() pour cacher l'Asset bar.

Appelle la méthode hideHintBtn() pour cacher le bouton « Acheter de l'aide » du menu central de droite.

Appelle la méthode hideExchangeBtn() pour cacher le bouton « Échanger ressources » du menu central de droite.

Appelle la méthode hideAbandonBtn() pour cacher le bouton « Abandonner la partie » du menu central de droite.

Appelle la méthode hideDisplayedWord() pour cacher le mot affiché en dessous du clavier (celui qui affiche les lettres devinées et cachées par un *.

Appelle la méthode hideAllAnimations() pour cacher toutes les animations qui puissent être affichées. Ces animations sont celles affichées quand le joueur rate ou devine un mot.

Appelle la méthode hideAvatarGraphics() de l'objet avatarAnimations pour cacher tous les éléments graphiques de l'avatar situés dans les layer-list.

Affiche un AlertDialog avec le score final du joueur.

Si le score est supérieur au highscore personnel du joueur, il est enregistré dans la table users de la base de données et l'objet activeUser est actualisé dans le viewModel.

Si le score est assez haut il peut aussi être enregistré dans la table highscores de la base de données.

- private fun showEndDefinitions()

Un AlertDialog utilisant l'adaptateur DefinitionsAdapter est affiché avec une liste des définitions disponibles du mot caché.

Si le joueur n'a plus de vies, la méthode loseGameRound() est appelée.

Retourne le bouton rouge (jouer/ continuer) à son état par défaut et appelle la méthode showNewRoundBtn() pour le rendre visible.

- private fun pickEndAnimation(type: String)

Affiche une animation finale si le joueur a deviné le mot caché ou non.

- private fun showAssetBar()

Rend l'Asset bar visible.

- private fun hideAssetBar()

Cache l'Asset bar.

- private fun updateAssetBar()

Actualise les valeurs de l'Asset bar.

- private fun showKeyboard()

Rend le clavier visible.

- private fun hideKeyboard()

Cache le clavier.

- private fun showDisplayedWord()

Rend visible le mot à deviner en dessous du pendu.

- private fun hideDisplayedWord()

Cache le mot à deviner en dessous du pendu.

- private fun showHintBtn()

Rend visible le bouton « Acheter de l'aide » du menu central de droite.

- private fun hideHintBtn()

Cache le bouton « Échanger ressources » du menu central de droite.

- private fun showExchangeBtn()

Rend visible le bouton « Échanger ressources » du menu central de droite.

- private fun hideExchangeBtn()

Cache le bouton « Acheter de l'aide » du menu central de droite.

- private fun showAbandonBtn()

Rend visible le bouton « Abandonner la partie » du menu central de droite.

- private fun hideAbandonBtn()

Cache le bouton « Abandonner la partie » du menu central de droite.

- private fun showNewRoundBtn()

Rend visible le bouton rouge (jouer/ continuer).

- private fun hideNewRoundBtn()

Cache le bouton rouge (jouer/ continuer).

- private fun hideAllAnimations()

Cache toutes les animations qui puissent être affichées. Ces animations sont celles affichées quand le joueur rate ou devine un mot.

- private fun prizeFadeOutCountdown()

Déclenche le timer10 pour l'animation du countdown du nombre de points possibles de gagner lorsque le joueur devine correctement une lettre.

3. Fragments

a. LoginFragment

Ce fragment contient tout ce qui est relatif à la connexion d'un utilisateur à son compte. Il est situé dans l'activité MainActivity.

i. Attributs

- private val viewModel: MainViewModel

Ceci est une instance du MainViewModel.

- var pwVisibilityState: Boolean = false

L'état de visibilité du champ mot de passe. Permet à l'utilisateur de voir ou cacher le mot de passe qui est écrit.

- var sounds: Sounds = null

L'instance de la classe Sounds qui contient la méthode pour jouer des fichiers audio.

ii. Méthodes

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

Ici se trouve tout ce qui va se passer à la création du fragment comme les bindings par example.

- private fun login (view: View)

Ici se fait la validation des champs que l'utilisateur doit remplir pour la connexion à son compte.

Si tous les champs sont validés, l'id de l'utilisateur et son choix de « Se souvenir de moi » lors de la connexion sont alors sauvegardés dans les shared preferences et le jeu commence en lançant l'activité GameActivity. Si au moins un des champs n'est pas validé, un message d'erreur s'affiche.

b. RegisterFragment

Ce fragment contient tout ce qui est relatif à la création d'un nouveau compte d'utilisateur. Il est situé dans l'activité MainActivity.

i. Attributs

- private val viewModel: MainViewModel

Ceci est une instance du MainViewModel.

- var pwVisibilityState: Boolean = false

L'état de visibilité du champ mot de passe. Permet à l'utilisateur de voir ou cacher le mot de passe qui est écrit.

- var pw2VisibilityState: Boolean = false

L'état de visibilité du champ confirmation du mot de passe. Permet à l'utilisateur de voir ou cacher le mot de passe qui est écrit.

- var sounds: Sounds = null

L'instance de la classe Sounds qui contient la méthode pour jouer des fichiers audio.

ii. Méthodes

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

Ici se trouve tout ce qui va se passer à la création du fragment comme les bindings par example.

- private fun register (view: View)

Ici se fait la validation des champs que l'utilisateur doit remplir pour la création d'un compte.

Si tous les champs sont validés, un nouveau utilisateur est créé dans la table users de la base de données et ce fragment est remplacé par le fragment log in. Si au moins un des champs n'est pas validé, un message d'erreur s'affiche.

4. Classes

a. Avatar

Cette classe contient des informations concernant l'avataret est utilisée dans la création des avatars dans table avatars de la base de données.

i. Attributs

- private long id

Contient l'id de l'avatar dans la table avatars de la base de données.

- private String head_shot

Contient un String avec l'identifiant du fichier graphique de l'image utilisé dans la liste de sélection desavatars.

- private String head_src

Contient un String avec l'identifiant du fichier graphique de latête de l'avatar.

- private String torso_src

Contient un String avec l'identifiant du fichier graphique du torse de l'avatar.

- private String left_arm_src

Contient un String avec l'identifiant du fichier graphique du bras gauche de l'avatar.

- private String right_arm_src

Contient un String avec l'identifiant du fichier graphique du bras droit de l'avatar.

- private String left_leg_src

Contient un String avec l'identifiant du fichier graphique de la jambe gauche de l'avatar.

- private String right_leg_src

Contient un String avec l'identifiant du fichier graphique de la jambe droit de l'avatar.

- private int eyesId

Contient l'id des yeux utiliséspar l'avatar. Cet id fait référence à la table eyes de la base de données.

- private int mouthId

Contient l'id de la bouche utiliséepar l'avatar. Cet id fait référence à la table mouths de la base de données.

- private int eyebrowsId

Contient l'id des sourcils utiliséspar l'avatar. Cet id fait référence à la table eyebrows de la base de données.

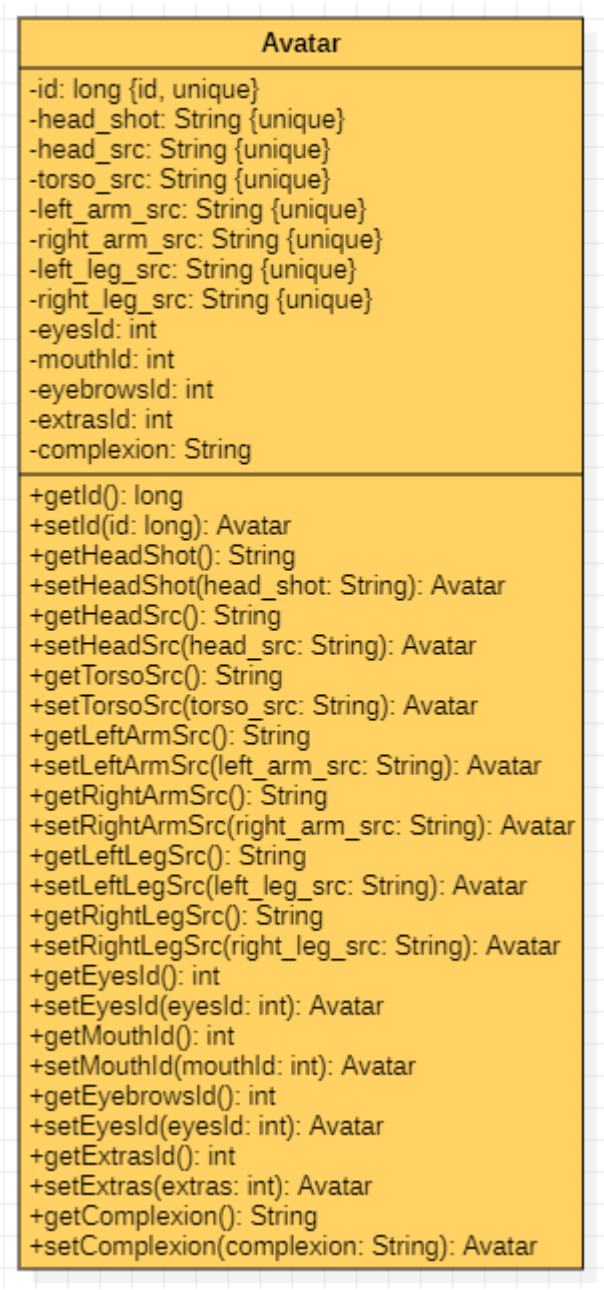
- private int extrasId

Contient l'id des extras (lunettes)utiliséspar l'avatar. Cet id fait référence à la table extras de la base de données.

- private String complexion

Contient un String avec un identifiant de teint qui déterminera la couleur de certains éléments du visage de l'avatar.

ii. Class diagram



b. Extra

Cette classe contient des informations concernant les extras (lunettes) de l'avatar et est utilisée dans la création des yeux dans table extras de la base de données.

i. Attributs

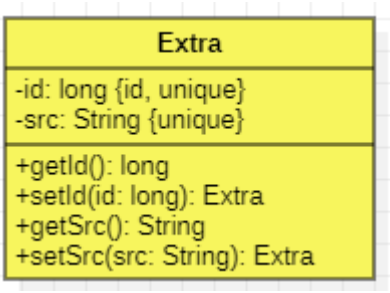
- private long id

Contient l'id des extras (lunettes) dans la table extras de la base de données.

- private String src

Contient un String avec l'identifiant du fichier graphique des extras de l'avatardans la table extras de la base de données.

ii. Class diagram



c. Eyebrows

Cette classe contient des informations concernant les sourcils de l'avatar et est utilisée dans la création des yeux dans table eyebrows de la base de données.

i. Attributs

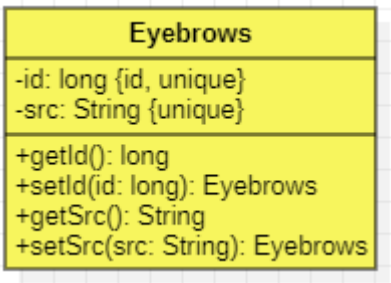
- private long id

Contient l'id des sourcils dans la table eyebrows de la base de données.

- private String src

Contient un String avec l'identifiant du fichier graphique des sourcils de l'avatar dans la table eyebrows de la base de données.

ii. Class diagram



d. Eyes

Cette classe contient des informations concernant les yeux de l'avataret est utilisée dans la création des yeux dans table eyes de la base de données.

i. Attributs

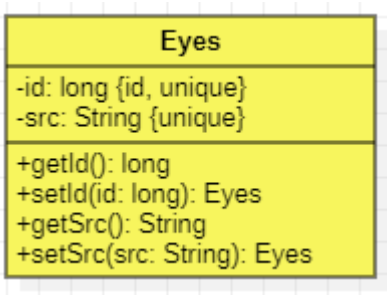
- private long id

Contient l'id des yeux dans la table eyes de la base de données.

- private String src

Contient un String avec l'identifiant du fichier graphique des yeux de l'avatardans la table avatars de la base de données.

ii. Class diagram



e. Highscore

Cette classe contient des informations concernant les 5 meilleurs scores de l'ensemble d'utilisateurs et est utilisée dans la création d'un highscore dans table highscores de la base de données.

i. Attributs

- private long id

Contient l'id du highscoredans la table highscores de la base de données.

- private int score

Contient le nombre de points de ce highscore.

- private String date

Contient un String avec la date à laquelle le highscore c'est fait.

- private int languageId

Contient l'id de la langueutilisée dans le jeu pendant lequel ce highscore c'est fait.

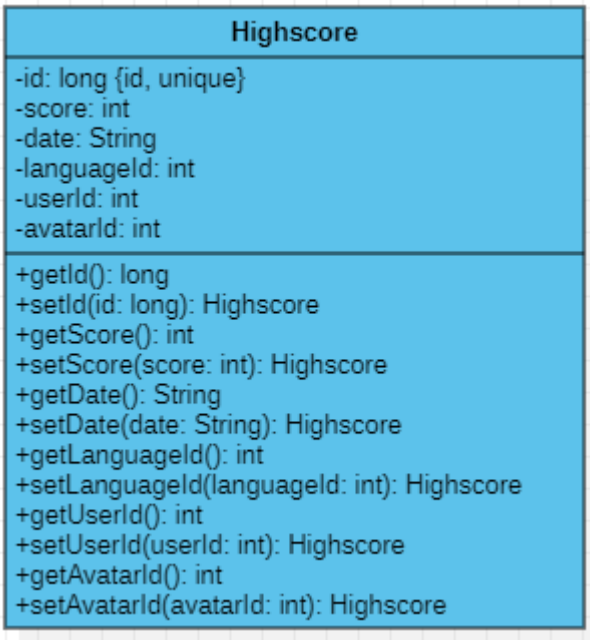
- private int userId

Contient l'id de l'utilisateur qui a fait ce highscore.

- private int avatarId

Contient l'id de l'avatar utilisé dans le jeu pendant lequel ce highscore c'est fait.

ii. Class diagram



f. Language

Cette classe contient des informations concernant la langue de jeu choisie par l'utilisateur et est utilisée dans la création d'une langue dans table languages de la base de données.

i. Attributs

- private long id

Contient l'id de la langue dans la table languages de la base de données.

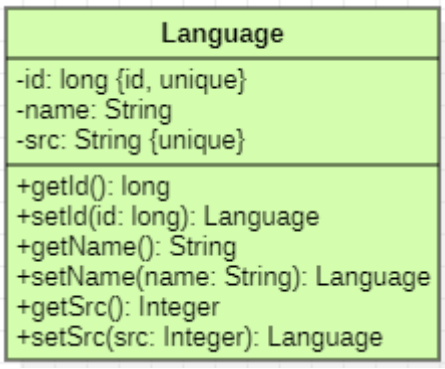
- private String name

Contient un String avec le nom de la langue de jeu choisie par l'utilisateur.

- private String src

Contient un String avec l'identifiant du fichier graphique de l'icône de la langue choisie par l'utilisateur.

ii. Class diagram



g. Mouth

Cette classe contient des informations concernant la bouche de l'avatar et est utilisée dans la création d'une bouche dans table mouths de la base de données.

i. Attributs

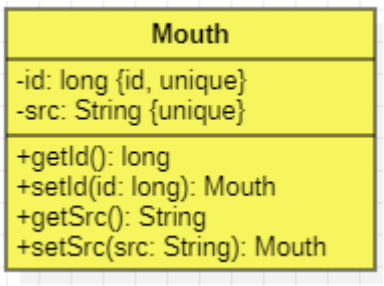
- private long id

Contient l'id de la bouche dans la table mouths de la base de données.

- private String src

Contient un String avec l'identifiant du fichier graphique de la bouche de l'avatardans la table avatars de la base de données.

ii. Class diagram



h. User

Cette classe contient des informations concernant l'utilisateur connecté (le joueur) et est utilisée dans la création d'un utilisateur dans table users de la base de données.

i. Attributs

- private final Contextcontext

Contient le contexte.

- private long id

Contient l'id de l'utilisateur dans la table users de la base de données.

- private String username

Contient le nom d'utilisateur de l'utilisateur dans la table users de la base de données.

- private String password

Contient le mot de passe de l'utilisateur dans la table users de la base de données.

- private int highscore

Contient le highscore de l'utilisateur dans la table users de la base de données.

- private int languageId

Contient l'id de la langue de l'utilisateur dans la table users de la base de données et fait référence à la table languages.

- private int avatarId

Contient l'id de l'avatar de l'utilisateur dans la table users de la base de données et fait référence à la table avatars.

- private int coins = 0

Contient le nombre de pièces de l'utilisateur dans la table users de la base de données.

- private int banknotes = 0

Contient le nombre de billets de l'utilisateur dans la table users de la base de données.

- private int diamonds = 0

Contient le nombre de diamants de l'utilisateur dans la table users de la base de données.

- private int lives = 5

Contient le nombre de vies de l'utilisateur dans la table users de la base de données.

- private int score = 0

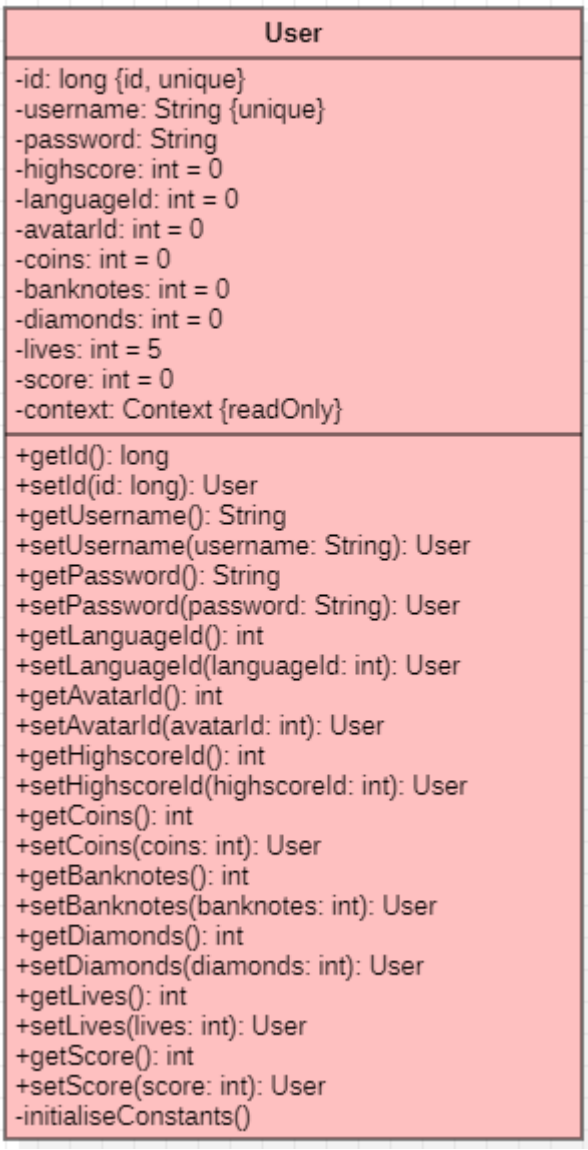
Contient le nombre de points de l'utilisateur dans la table users de la base de données.

ii. Méthodes

- private void initialiseConstants()

Initialise les attributs de la classe avec les valeurs des constantes de jeux présents dans une Array dans les ressources.

iii. Class diagram



i. AvatarAnimations

Cette classe contient surtout des méthodes qui contrôlent l'affichage des différentes parties du corps de l'avatar.

i. Attributs

- var blinkTimerInit: CountDownTimer = null

Un CountDownTimer qui contrôle le temps où les yeux de l'avatar sont ouverts.

- var blinkTimerEnd: CountDownTimer = null

Un CountDownTimer qui contrôle le temps où les yeux de l'avatar sont fermés.

ii. Méthodes

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

Retourne l'id de la ressource avec son identifiant String.

- fun hideAvatarGraphics(context: Context)

Cache toutes les images de l'avatar qui se situent dans les LayerDrawable.

- private fun hideAvatarEyesGraphics(context: Context)

Cache toutes les images des yeuxde l'avatar qui se situent dans les LayerDrawable.

- private fun hideAvatarEyebrowsGraphics(context: Context)

Cache toutes les images des sourcilsde l'avatar qui se situent dans les LayerDrawable.

- private fun hideAvatarExtraGraphics(context: Context)

Cache toutes les images desextras(lunettes) de l'avatar qui se situent dans les LayerDrawable.

- private fun hideAvatarMouthGraphics(context: Context)

Cache toutes les images des bouchesde l'avatar qui se situent dans les LayerDrawable.

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

Affiche les graphiques montrant l'avatar mort.

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

Affiche les graphiques montrant un visage de l'avatar avec des yeux, bouche, et sourcils spécifiques (visage par défaut).

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

Affiche les graphiques montrant un visage de l'avatar avec des yeux, bouche, et sourcils spécifiques (avatar qui s'ennuie et regarde en haut à gauche).

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

Affiche les graphiques montrant un visage de l'avatar avec des yeux, bouche, et sourcils spécifiques (avatar qui s'ennuie et regarde en haut à droite).

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

Affiche les graphiques montrant un visage de l'avatar avec des yeux, bouche, et sourcils spécifiques (avatar qui s'ennuie et regarde en bas à gauche).

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

Affiche les graphiques montrant un visage de l'avatar avec des yeux, bouche, et sourcils spécifiques (avatar qui s'ennuie et regarde en bas à droite).

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

Affiche les graphiques montrant les yeux fermés de l'avatar.

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

Affiche les graphiques montrant les yeux ouverts par défaut de l'avatar.

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

Affiche les graphiques montrant les parties du corps de l'avatar qui doivent être visibles selon le nombre de lettres ratées par le joueur et les types de visages choisis.

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

Affiche les graphiques montrant les yeux de l'avatar qui doivent être visibles selon les types choisis.

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

Affiche les graphiques montrant les sourcils de l'avatar qui doivent être visibles selon les types choisis.

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

Affiche les graphiques montrant les extras (lunettes) de l'avatar qui doivent être visibles selon les types choisis.

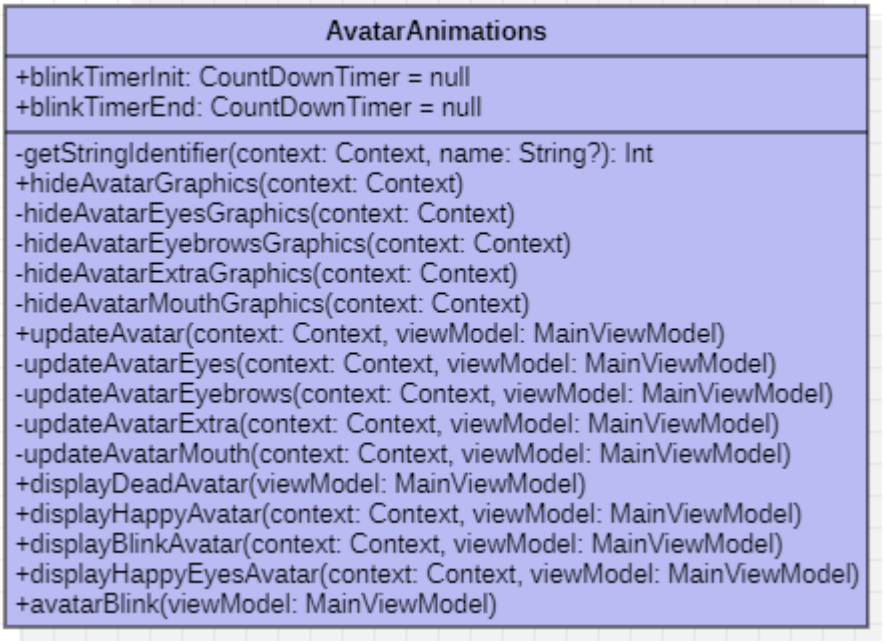
- suspend fun updateAvatarMouth(context: Context, viewModel: MainViewModel)

Affiche les graphiques montrant les bouches de l'avatar qui doivent être visibles selon les types choisis.

- fun avatarBlink(viewModel: MainViewModel)

Lance les deux CountDownTimers pour les clignements des yeux de l'avatar.

iii. Class diagram



j. AvatarMoods

Cette classe est un Enum qui contient les clefs correspondantes à des états graphiques du visage de l'avatar sur l'écran

k. GameRound

Cette classe contient plusieurs informations concernant le tour courant du jeu.

i. Attributs

- private val context: Context

Contient le contexte.

- var activeRound: Boolean = false

Un Boolean qui est vrai seulement si une partie est en cours.

- var letterMisses: Int = 0

Un Int avec le nombre de lettres sélectionnées par le joueur mais qui ne sont pas dans le mot caché (lettres ratés).

- var guessedLetters: Int = 0

Un Int avec le nombre de lettres du mot caché correctement devinées par le joueur.

- var lettersGuessedConsecutively: Int = 0

Un Int avec le nombre de lettres du mot caché correctement devinées d'affilée.

- var wordsGuessedConsecutively: Int = 0

Un Int avec le nombre de mots cachés correctement devinées d'affilée.

- var wordsGuessedConsecutivelyNoFaults: Int = 0

Un Int avec le nombre de mots cachés correctement devinées d'affilée sans avoir raté aucune lettre.

- var potentialPrize: Int = 10

Un Int avec le nombre de points possibles de gagner si le joueur devine une lettre correctement. Ce prix diminue chaque seconde jusqu'à 0.

- var exchangeValues_Banknotes_price: Int = 0

Un Int avec le prix en nombre de pièces d'un billet.

- var exchangeValues_Diamonds_price: Int = 0

Un Int avec le prix en nombre de billets d'un diamant.

- var exchangeValues_Lives_price: Int = 0

Un Int avec le prix en nombre de diamants d'unevie.

- var helpValues_Letter_price: Int = 0

Un Int avec le prix en nombre de pièces d'une lettre.

- var helpValues_Definition_price: Int = 0

Un Int avec le prix en nombre de billets d'une définition.

- var helpValues_BodyPart_price: Int = 0

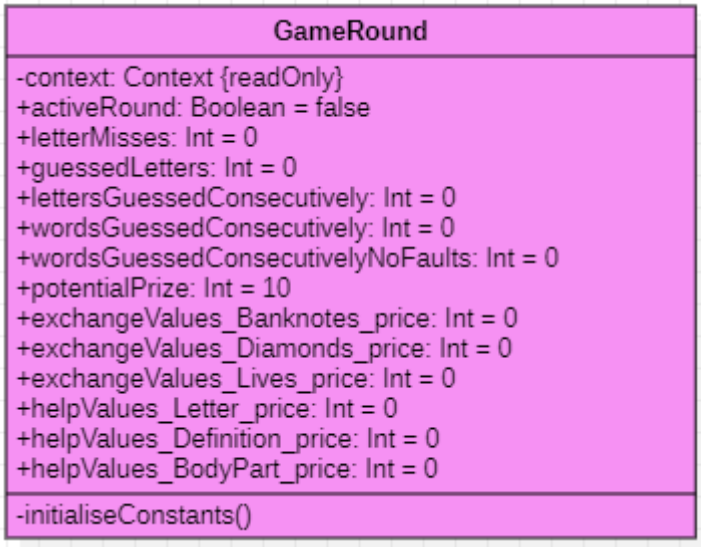
Un Int avec le prix en nombre de diamants pour enlever une partie du corps de l'avatar pendu.

ii. Méthodes

- private fun initialiseConstants()

Initialise les constantes du jeu en donnant des valeurs aux attributs concernés. Les valeurs de cettes constantes sont localisés dans une Array dans les ressources.

iii. Class diagram



l. Sounds

Cette classe contient une simple méthode pour jouer un son.

i. Attributs

- private val context: Context

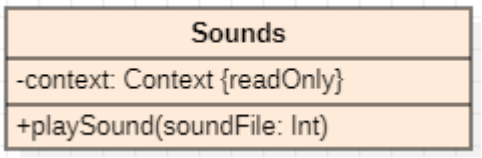
Contient le contexte.

ii. Méthodes

- fun playSound(soundFile: Int)

Lance un fichier de son à partir de son id.

iii. Class diagram



m. Word

Cette classe contient des informations sur le mot aléatoire que le joueur doit deviner.

i. Attributs

- var hiddenWord: String

Cette variable contient un String avec le mot aléatoire recherché dans une des APIs.

- var definitions: ArrayList<String>

Contient un ArrayList de String avec les définitions du mot aléatoire recherché dans une des APIs.

- private var language: String

Contient un String avec la langue du mot aléatoire recherché dans une des APIs.

- var displayedWord: String

Contient le mot affiché sur l'écran en dessous du pendu. Ce mot peut avoir des lettres cachées (affichées par un *****).

- var revealedDefinitions= ArrayList<String>

Contient un ArrayList de String avec les définitions qui n'ont pas encore étés révélés au joueur.

ii. Méthodes

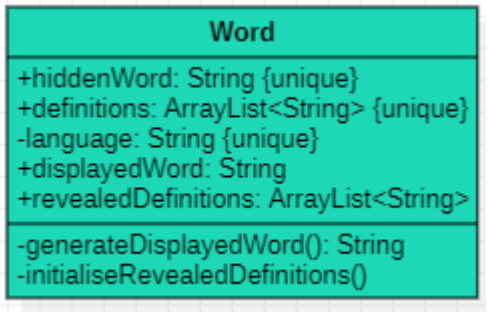
- private fun generateDisplayedWord(): String

Cette fonction génère un mot caché, représenté par des astérisques, et le sauvegarde dans la variable displayedWord.

- private fun initialiseRevealedDefinitions()

Sauvegarde les définitions dans la variable revealedDefinitions.

iii. Class diagram



5. Adapters

a. DefinitionsAdapter

Cet Adapter prépare les données relatives à l'AlertDialog avec les définitions du mot actif pour affichage.

b. HighscoresAdapter

Cet Adapter prépare les données relatives à l'AlertDialog avec les Top 5 scores pour affichage.

6. ViewModel

a. MainViewModel

i. Attributs

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

Utilisés pour sauvegarder un objet de type User actif. L'objet user contient les informations de l'utilisateur connecté.

- private var _randomWord: MutableLiveData<String> = “”

Utilisé pour sauvegarder un String avec le mot aléatoire qui a été recherché dans une des APIs.

- private var _definitions = MutableLiveData<ArrayList<String>

Utilisé pour sauvegarder un ArrayList de String avec les définitions du mot aléatoire qui a été recherché dans une des APIs.

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

Utilisés pour sauvegarder un objet de typeWord actif. L'objet word contient des informations sur le mot actif.

- var avatarList = MutableLiveData<ArrayList<Avatar>>

Utilisé pour sauvegarder un ArrayList de typeAvatar avec la liste de tous les avatars possibles. L'objet avatar contient des informations sur l'avatar choisi par l'utilisateur.

- var languageList = MutableLiveData<ArrayList<Language>>

Utilisé pour sauvegarder un ArrayList de typeLanguage avec la liste de toutes les langues possibles.

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

Utilisés pour sauvegarder un objet de typeAvatar actif.

- var _activeAvatarMood = MutableLiveData<AvatarMoods>
- val activeAvatarMood: LiveData<AvatarMoods>

Utilisés pour sauvegarder un Enum de typeAvatarMoods actif. Cet Enum contient les clefs correspondantes à des états graphiques du visage de l'avatar sur l'écran.

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

Utilisés pour sauvegarder un objet de typeLanguage actif. Ceci est la langue des mots et définitions que le jeu affichera.

- var avatarLastSelectedCheckbox:MutableLiveData<Int> = 0

Utilisé pour sauvegarder un Int avec l'indice de la dernière checkbox cochée dans le menu où l'utilisateur peut choisir un avatar.

- var languageLastSelectedCheckbox:MutableLiveData<Int> = 0

Utilisé pour sauvegarder un Int avec l'indice de la dernière checkbox cochée dans le menu où l'utilisateur peut choisir une langue.

ii. Méthodes

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

Cherche tous les éléments dans la table avatars de la base de données et retourne unArrayList d'objets de typeAvatar.

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

Cherche tous les éléments dans la table languages de la base de données et retourne un ArrayList d'objets de type Language.

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

Crée l'objet de type User et le sauvegarde dans la variable _activeUser du MainViewModel.

Affiche sur l'App bar (menu du haut) l'icône de la langue sauvegardée pour cet utilisateur dans la table users de la base de données.

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

Actualise l'objet de type User qui est sauvegardé dans la variable _activeUser du MainViewModel.

Affiche sur l'App bar (menu du haut) l'icône de la langue sauvegardée pour cet utilisateur dans la table users de la base de données.

- private fun generateNewWord(language: String)

Crée un objet de type Word et le sauvegarde dans la variable _word du MainViewModel.

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

Prends la lettre devinée par le joueur et la cherche dans le mot caché.

Pendant ce processus, les caractères du mot caché sont formatés par la méthode prepareCharacter de manière à être reconnus s'ils sont des caractères spéciaux.

Si la lettre existe dans le mot caché, l'attribut guessedLetters de l'objet gameRound et l'attribut displayedWord de l'objet _word sont actualisés.

Ça retourne un Boolean selon la lettre être existante dans le mot caché ou non.

- fun prepareCharacter(character: String): String

Prends une lettre et retourne l'équivalente en majuscule sans accents.

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

Cherche tous les éléments dans la table highscores de la base de données et retourne un ArrayList d'objets de type Highscore.

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

Insère un élément dans la table highscores de la base de données.

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

Modifie un élément dans la table highscores de la base de données.

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

Cherche la table avatars de la base de données et retourne un ArrayList de String avec les noms de fichiers des sources d'images des avatars utilisés dans le menu pour choisir l'avatar.

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

Cherche la table users de la base de données et retourne un Boolean selon le nom d'utilisateur être déjà existant ou non dans la base de données.

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

Prends un id et cherche l'avatar correspondant dans la table avatars de la base de données pour retourner une List d'objets de type Avatar avec tous les avatars dans la base de données avec cet id (un seul objet est présent dans la List).

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

Prends un id et cherche la langue correspondant dans la table languages de la base de données pour retourner une List d'objets de type Language avec toutes les langues dans la base de données avec cet id (un seul objet est présent dans la List).

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

Prends un nom d'utilisateur et un mot de passe comme arguments et cherche l'utilisateur correspondant dans la table users de la base de données pour retourner un Long avec l'id de cet utilisateur.

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

Cherche tous les éléments dans la table users de la base de données et retourne une List d'objets de type User.

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

Prends un id et cherche l'utilisateur correspondant dans la table users de la base de données pour retourner une List d'objets de type User avec tous les utilisateurs dans la base de données avec cet id (un seul objet est présent dans la List).

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

Prends un id et cherche les sourcils correspondantes dans la table eyebrows de la base de données pour retourner une List d'objets de type Eyebrows avec tous les sourcis dans la base de données avec cet id (un seul objet est présent dans la List).

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

Prends un id et cherche les yeux correspondants dans la table eyes de la base de données pour retourner une List d'objets de type Eyes avec tous les yeux dans la base de données avec cet id (un seul objet est présent dans la List).

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

Prends un id et cherche les extras (lunettes) correspondants dans la table extras de la base de données pour retourner une List d'objets de type Extra avec tous les extras dans la base de données avec cet id (un seul objet est présent dans la List).

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

Prends un id et cherche les bouches correspondantes dans la table mouths de la base de données pour retourner une List d'objets de type Mouth avec toutes les bouches dans la base de données avec cet id (un seul objet est présent dans la List).

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

Insère un élément dans la table users de la base de données.

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

Supprime un élément de la table users de la base de données.

- fun getRandomWordFr()

Appelle l'API pour recevoir un mot aléatoire en Français.

Le mot est sauvegardé dans la variable _randomWord et le passe comme argument à la méthode getFrenchDefinition.

- fun getRandomWordEn()

Appelle l'API pour recevoir un mot aléatoire en Anglais.

Le mot est sauvegardé dans la variable _randomWord et le passe comme argument à la méthode getEnglishDefinition.

- fun getFrenchDefinition(word: String)

Appelle l'API pour recevoir les définitions du mot en Français et les sauvegarde dans la variable _definitions.

- fun getEnglishDefinition(word: String)

Appelle l'API pour recevoir les définitions du mot en Anglais et les sauvegarde dans la variable _definitions.

7. APIs

a. RandomWordAPI

API qui retourne un mot aléatoire en Anglais.

Lien

b. API Dictionary French

API qui retourne un mot aléatoire et des définitions d'un mot en Français.

Lien

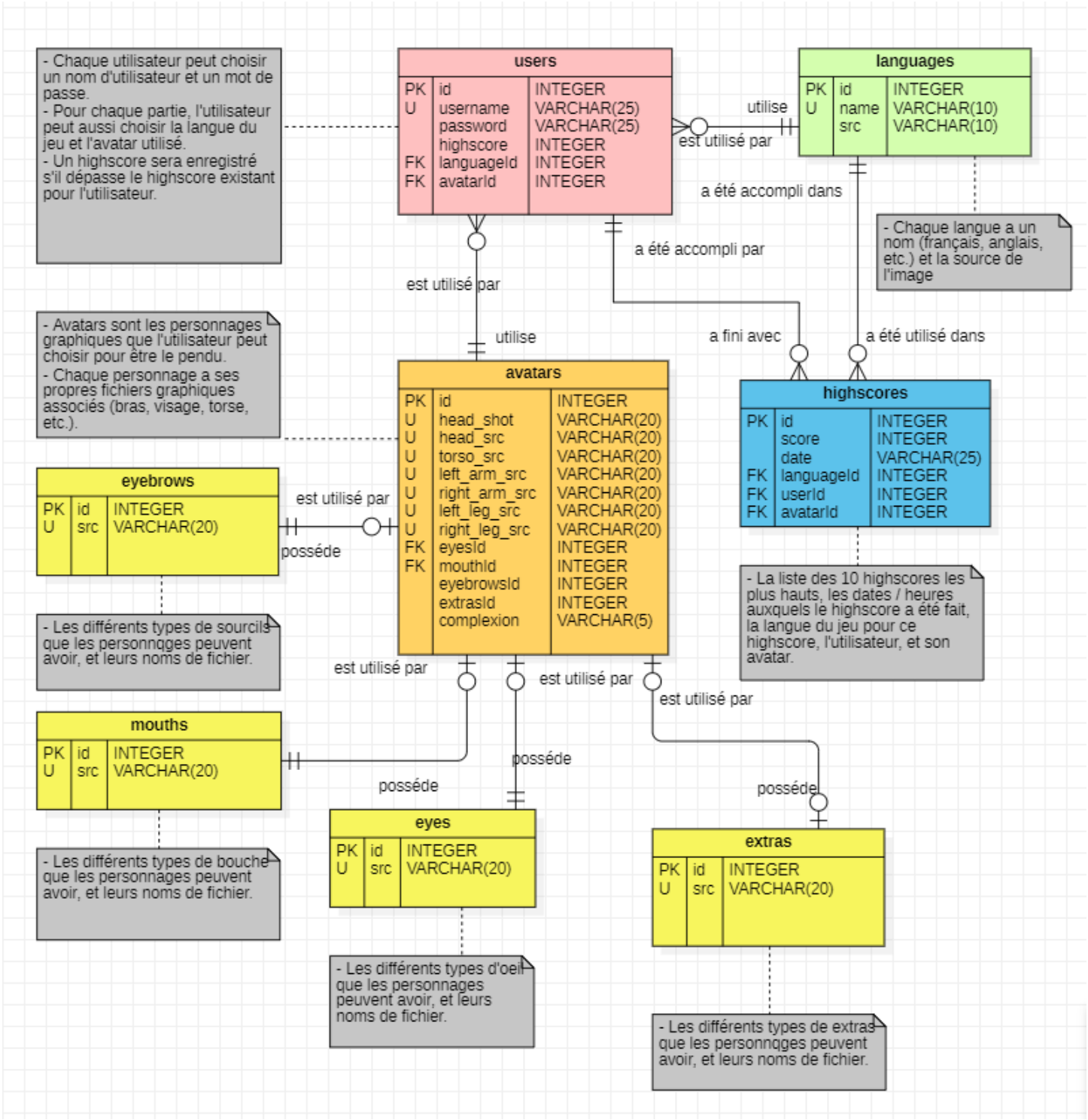
c. Free Dictionary API

API qui retourne des définitions d'un mot en Anglais.

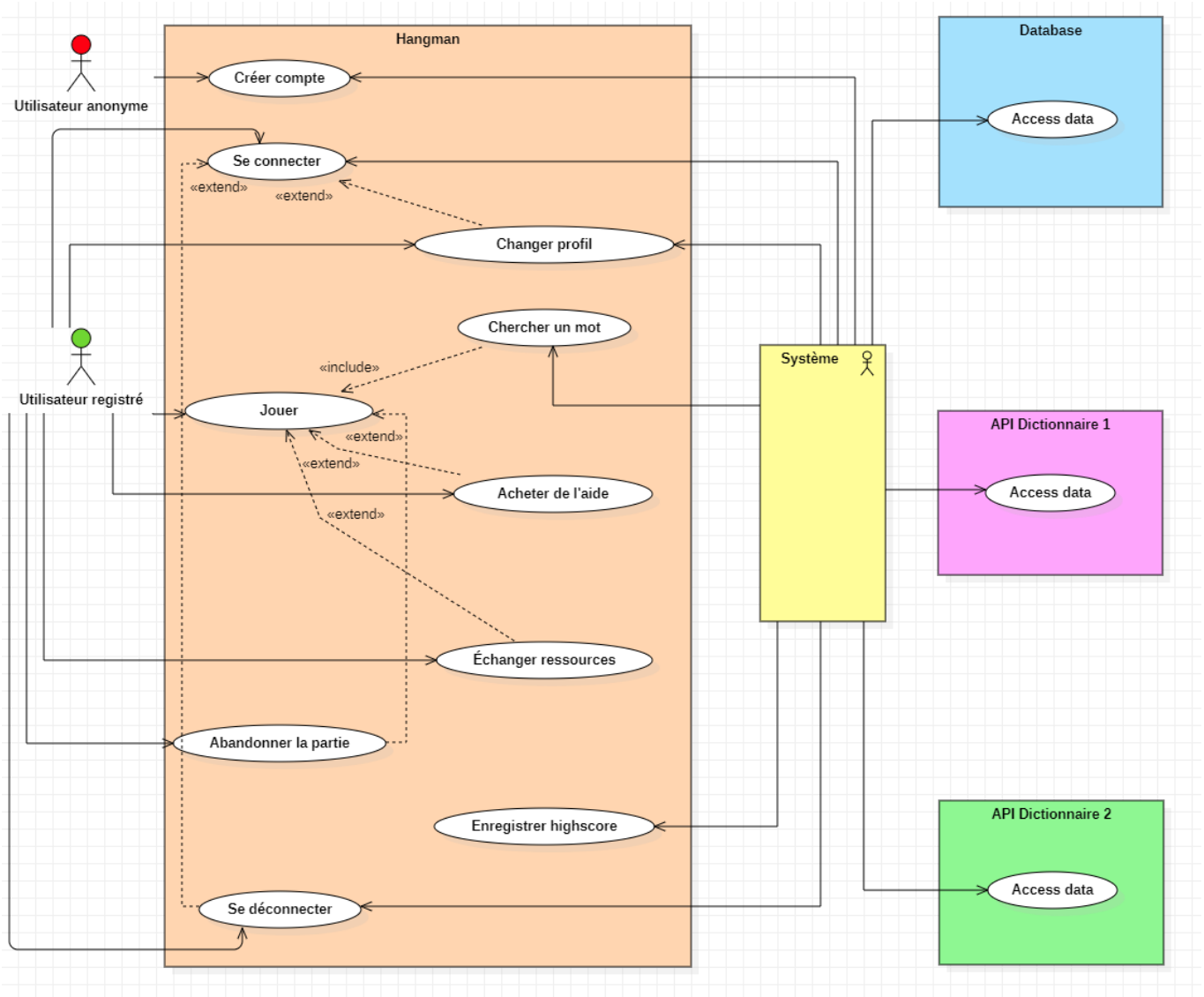
Lien

8. Database

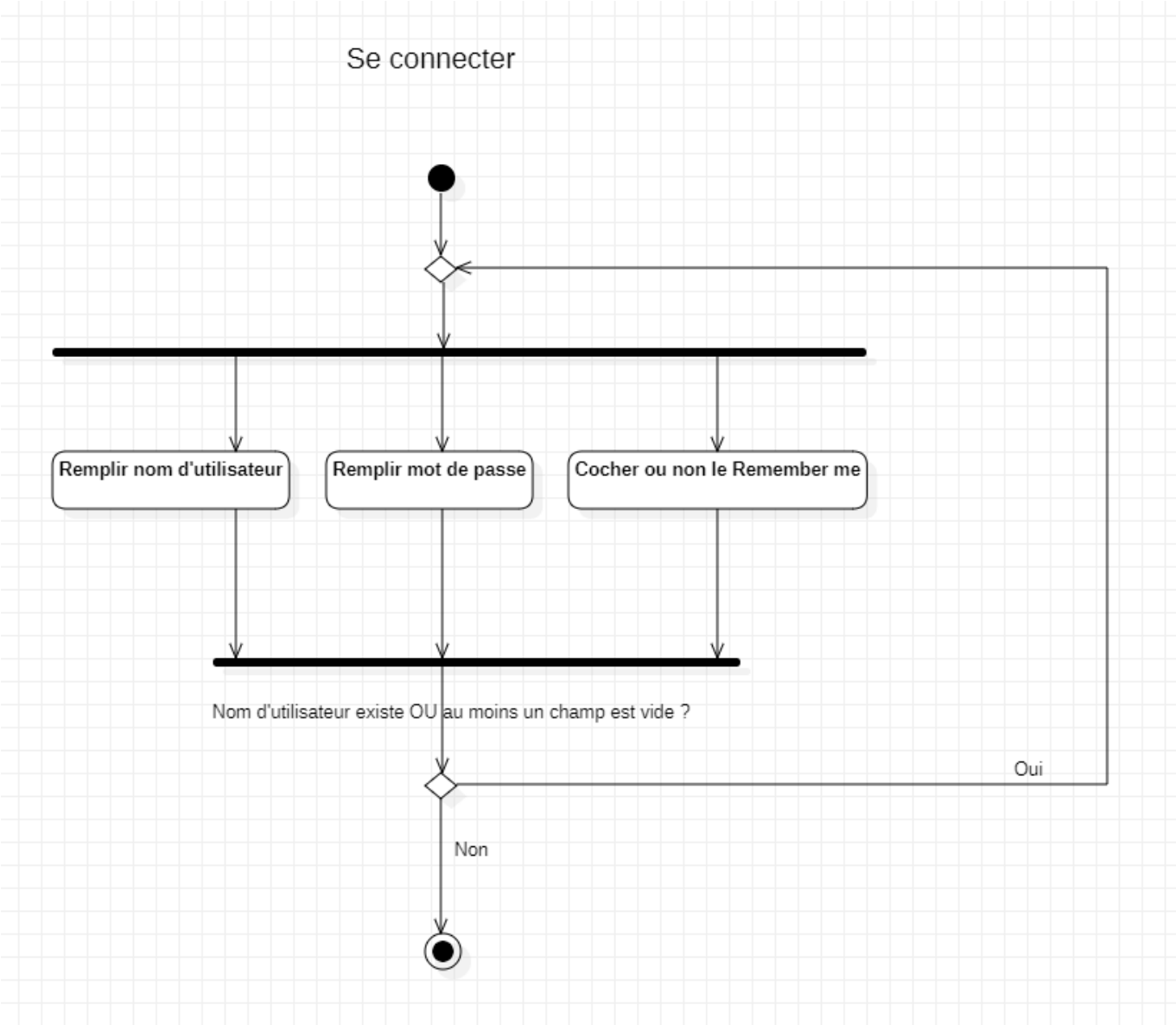
a. HangmanDb – Entity relationship diagram

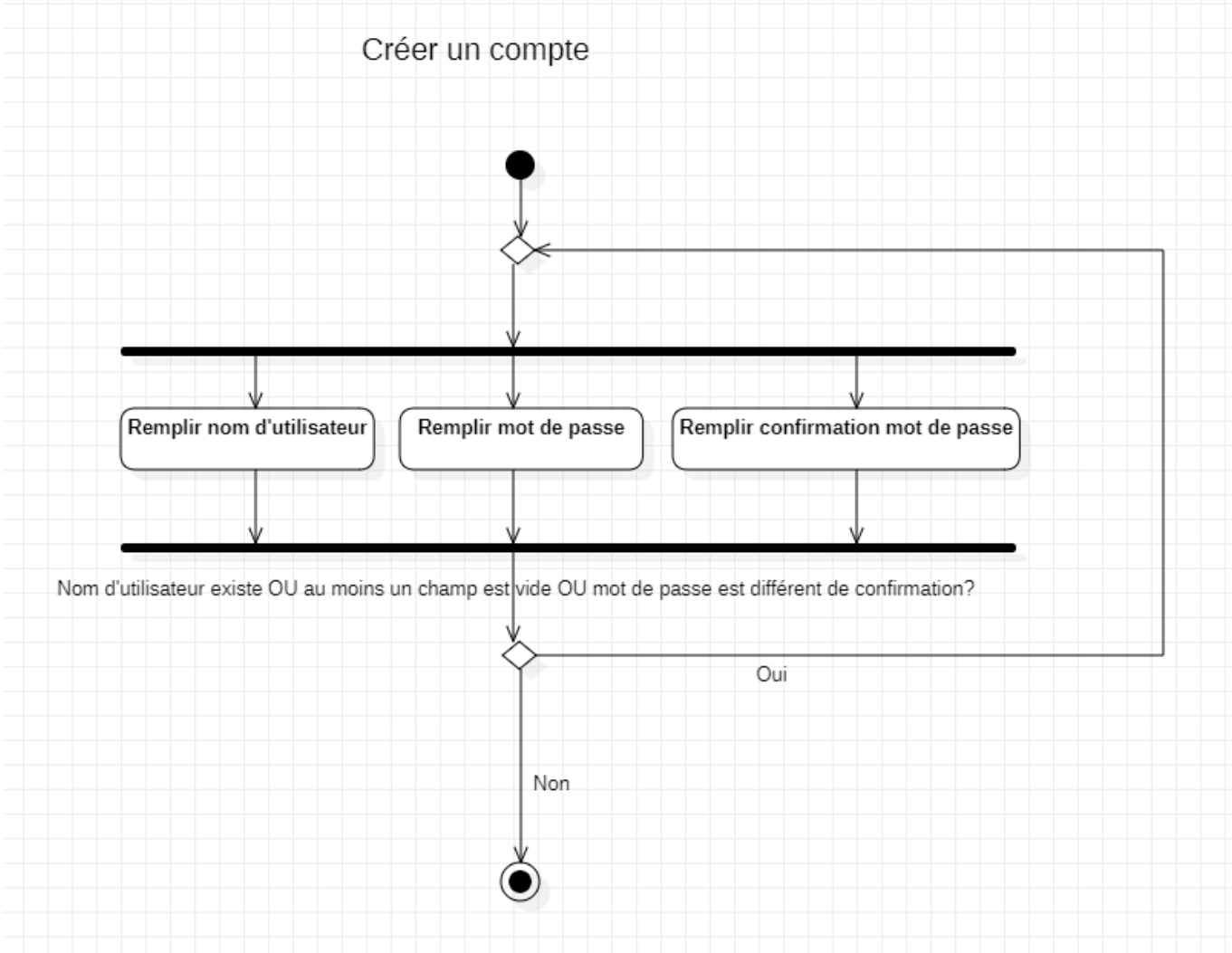


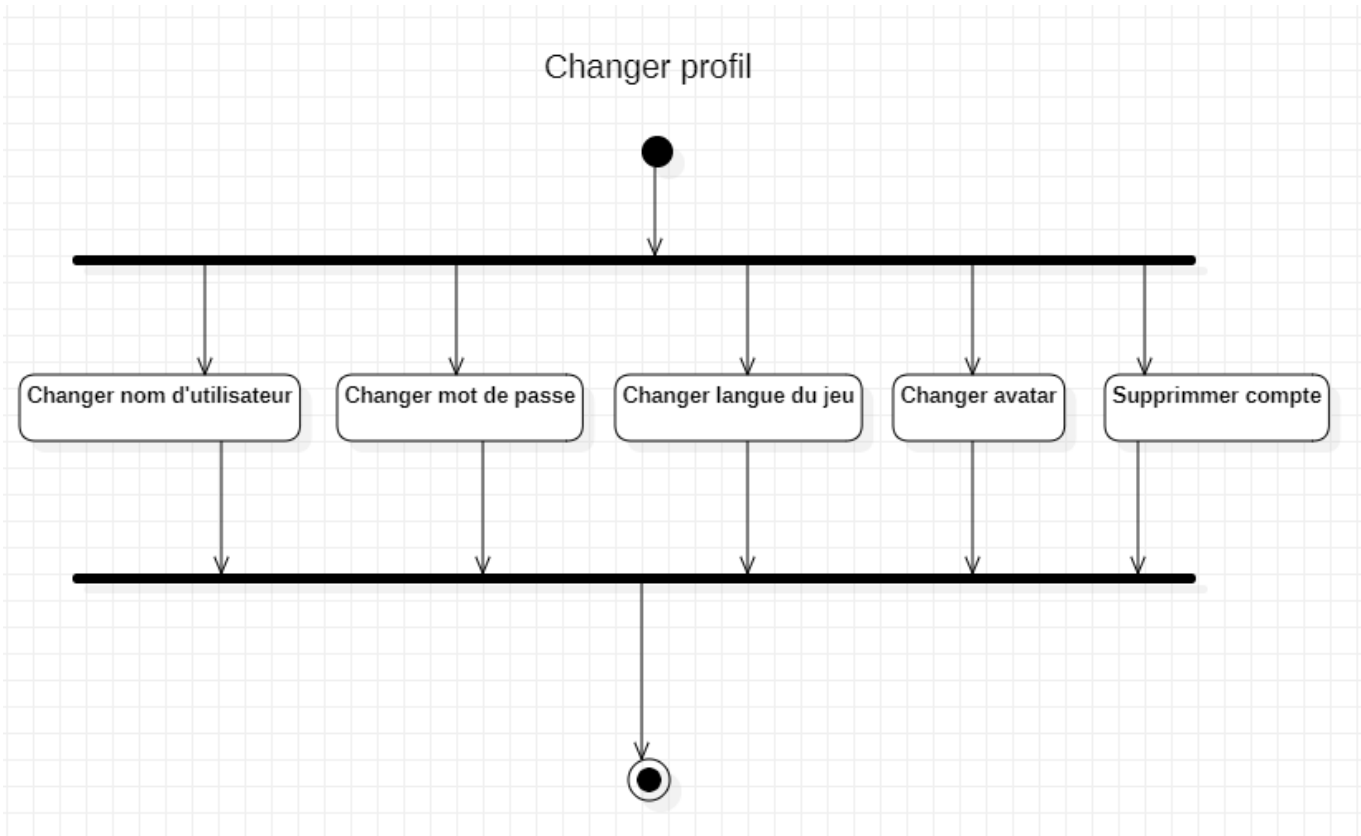
9. Use case diagram

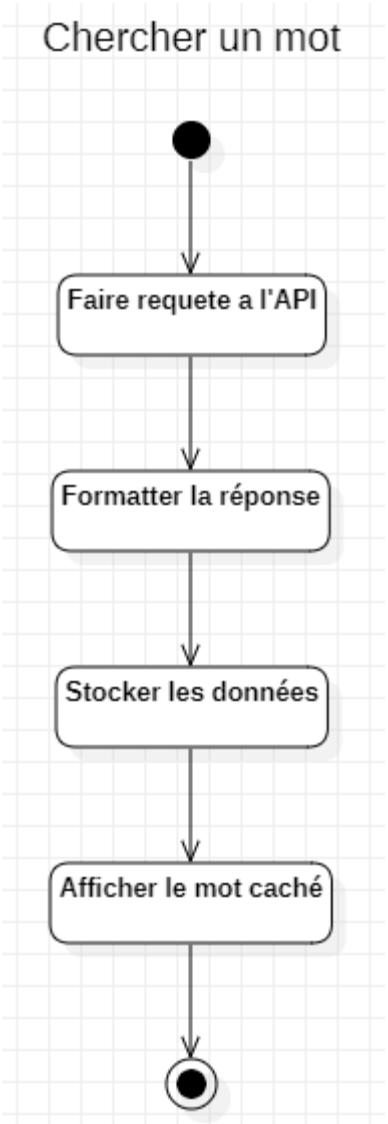


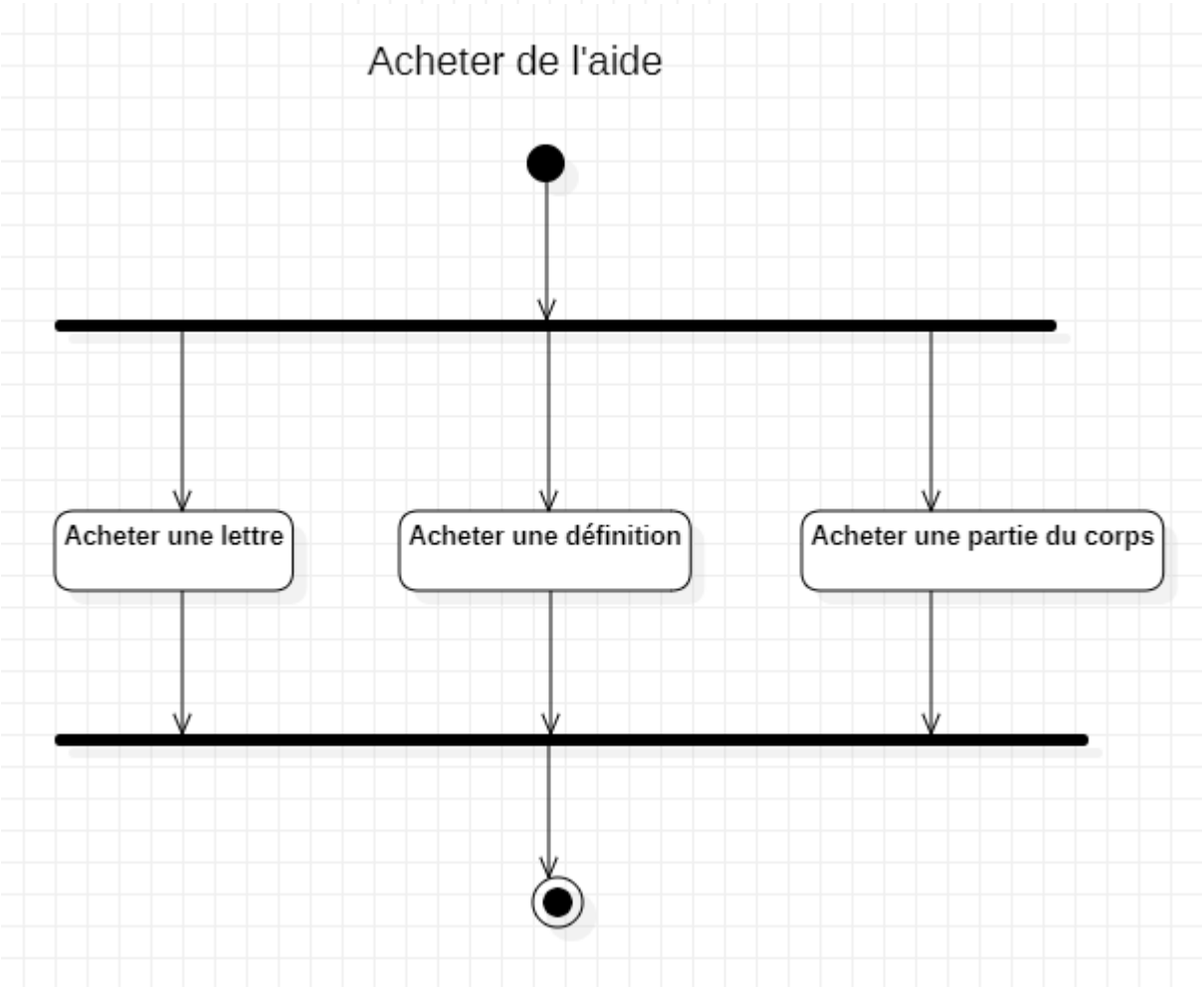
10. Activity diagrams

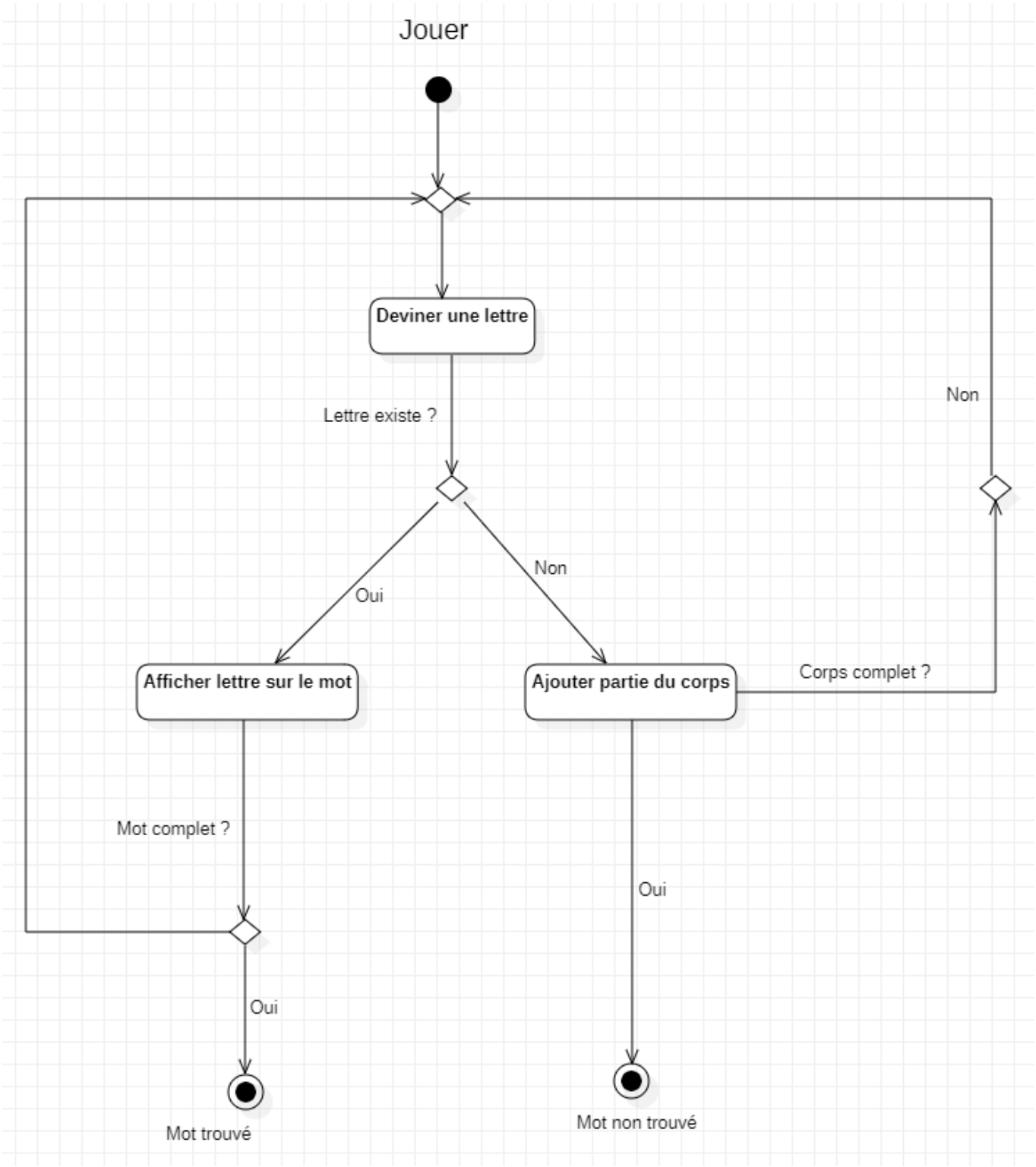


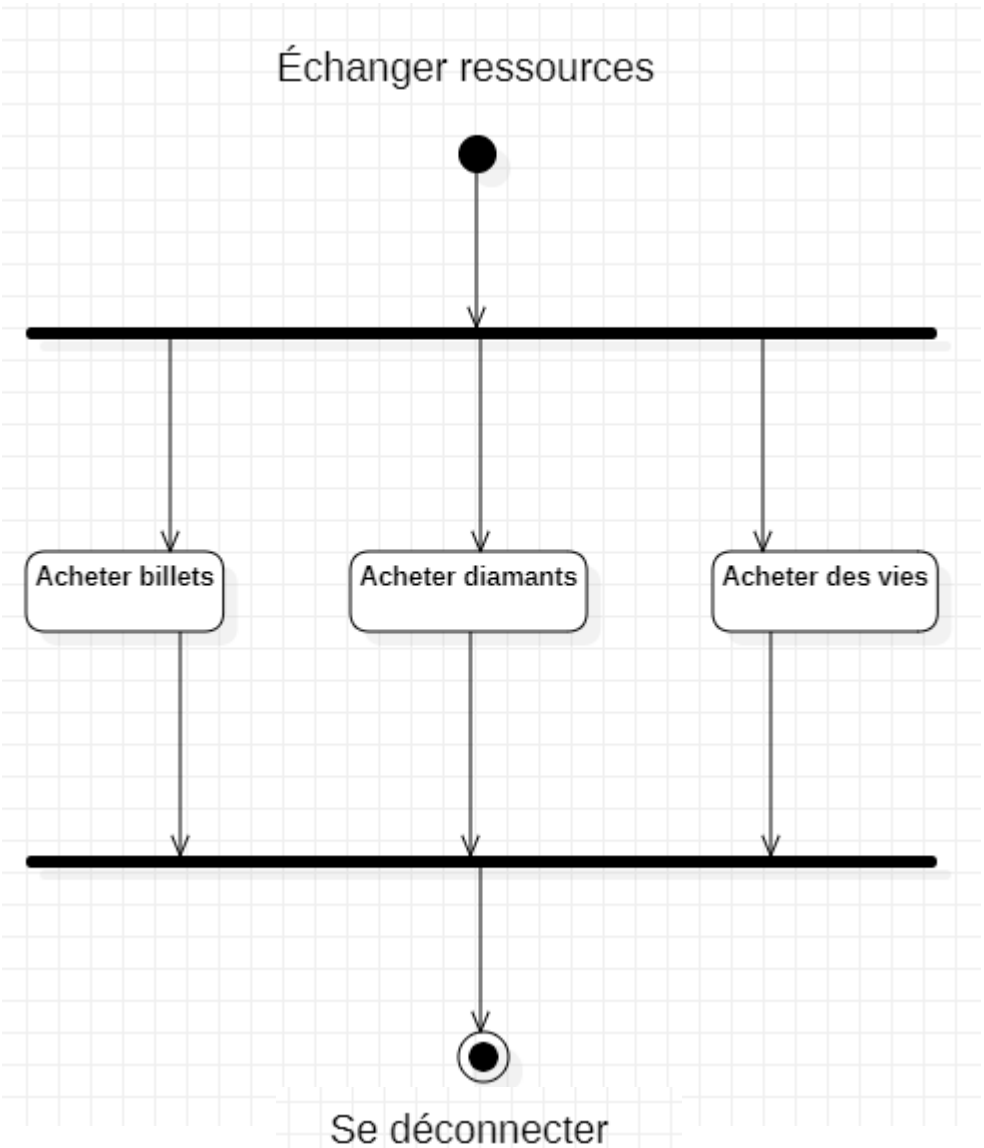

Créer un compte
Remplir nom d'utilisateur
Remplir mot de passe
Remplir confirmation mot de passe
Nom d'utilisateur existe OU au moins un champ est vide OU mot de passe est différent de confirmation?
Oui
Non

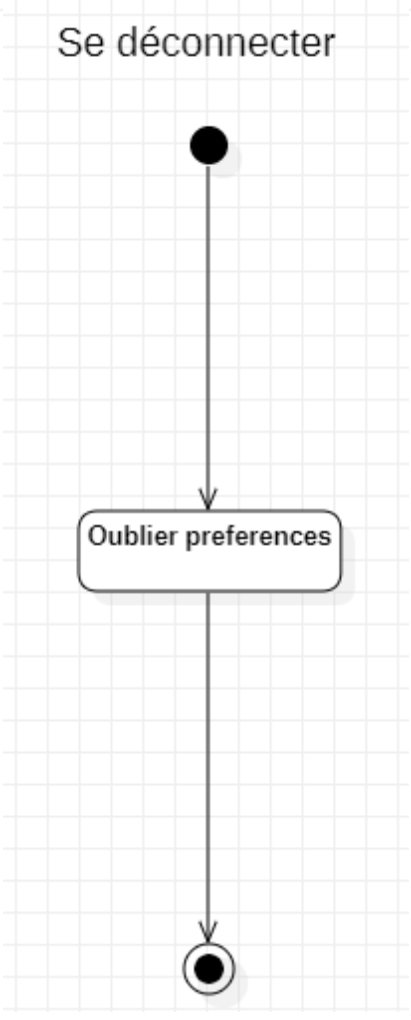

Changer profil
Changer nom d'utilisateur
Changer mot de passe
Changer langue du jeu
Changer avatar
Supprimmer compte

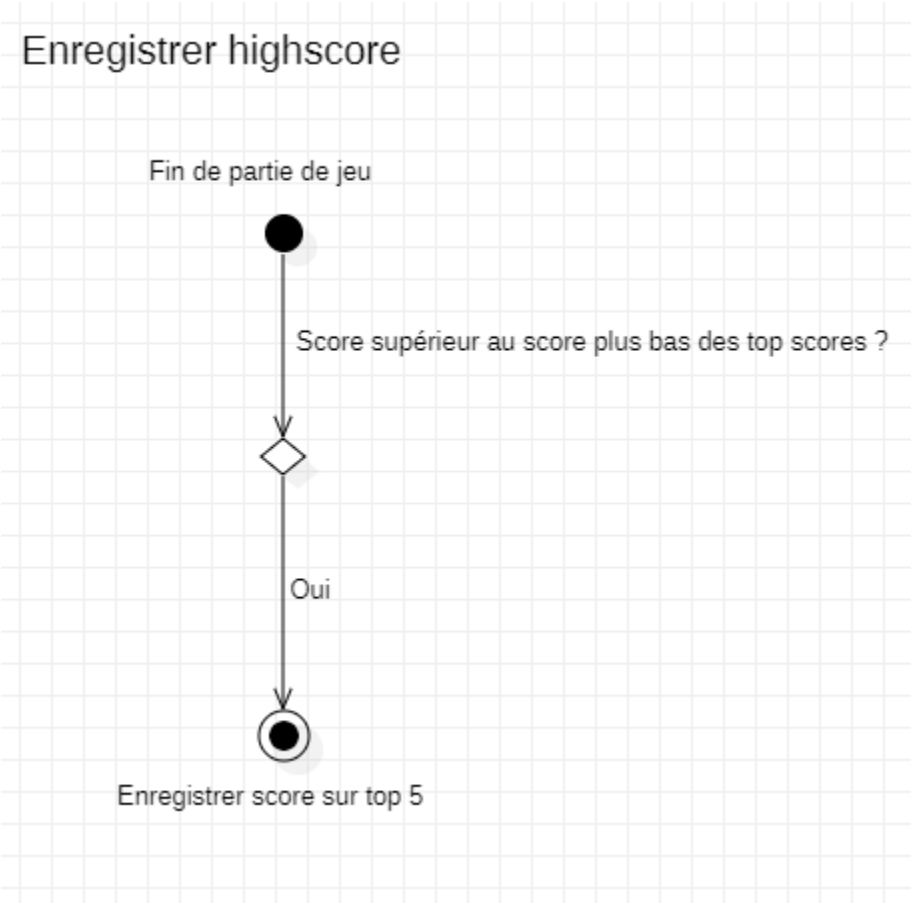

Chercher un mot
Faire requete a l'API
Formatter la réponse
Stocker les données
Afficher le mot caché

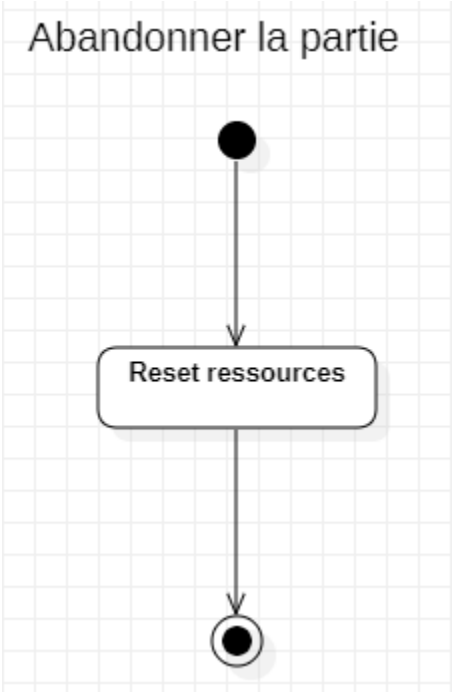

Acheter de l'aide
Acheter une lettre
Acheter une définition
Acheter une partie du corps


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