Barou Romain
Chanson Chloé

Dossier Android – Compte rendu
Application « Reing rise »

29/03/2020

Contexte :

Notre application a pour nom « Reing rise ». Il s'agit d'un jeu dans lequel le joueur se retrouve à la place d'un Dieu pour lequel il choisit un nom et dont le paradis est géolocalisé, il doit répondre de manière binaire (par oui ou non) à des actions que ses sujets vont subir.

Ces actions vont ensuite faire augmenter la barre de richesse, de pouvoir, ou de révolte du joueur. Si la barre de richesse ou de pouvoir arrive à 100%, il a gagné, dans le cas où c'est la barre de révolte qui atteint 100% il a perdu.

Le jeu ne se termine qu'à la condition que l'une de ces trois barres soit à 100% (Le joueur ne peut pas perdre en ayant plus de cartes dans sa pile, car cette dernière se remplir de manière automatique lorsque le nombre de cartes est insuffisant).

Pour indiquer son choix le joueur doit faire glisser sa carte dans le sens où il souhaite répondre (pour répondre il doit faire un swipe à la manière de l'application Tinder par exemple), il doit glisser vers la droite pour donner son accord, vers la gauche pour exprimer son refus. Ainsi le joueur pour gagner doit satisfaire au mieux sa population.

Ce jeu se joue seul et se base sur la réflexion du joueur.

Diagramme de cas d'utilisation :

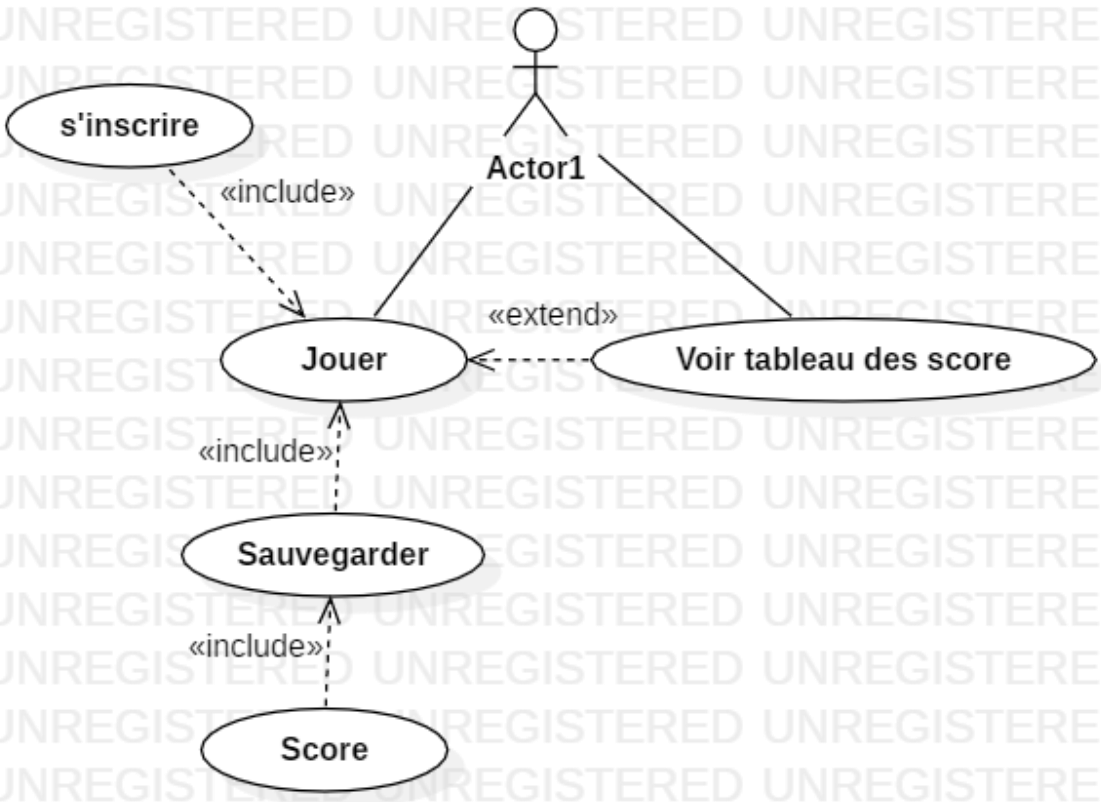
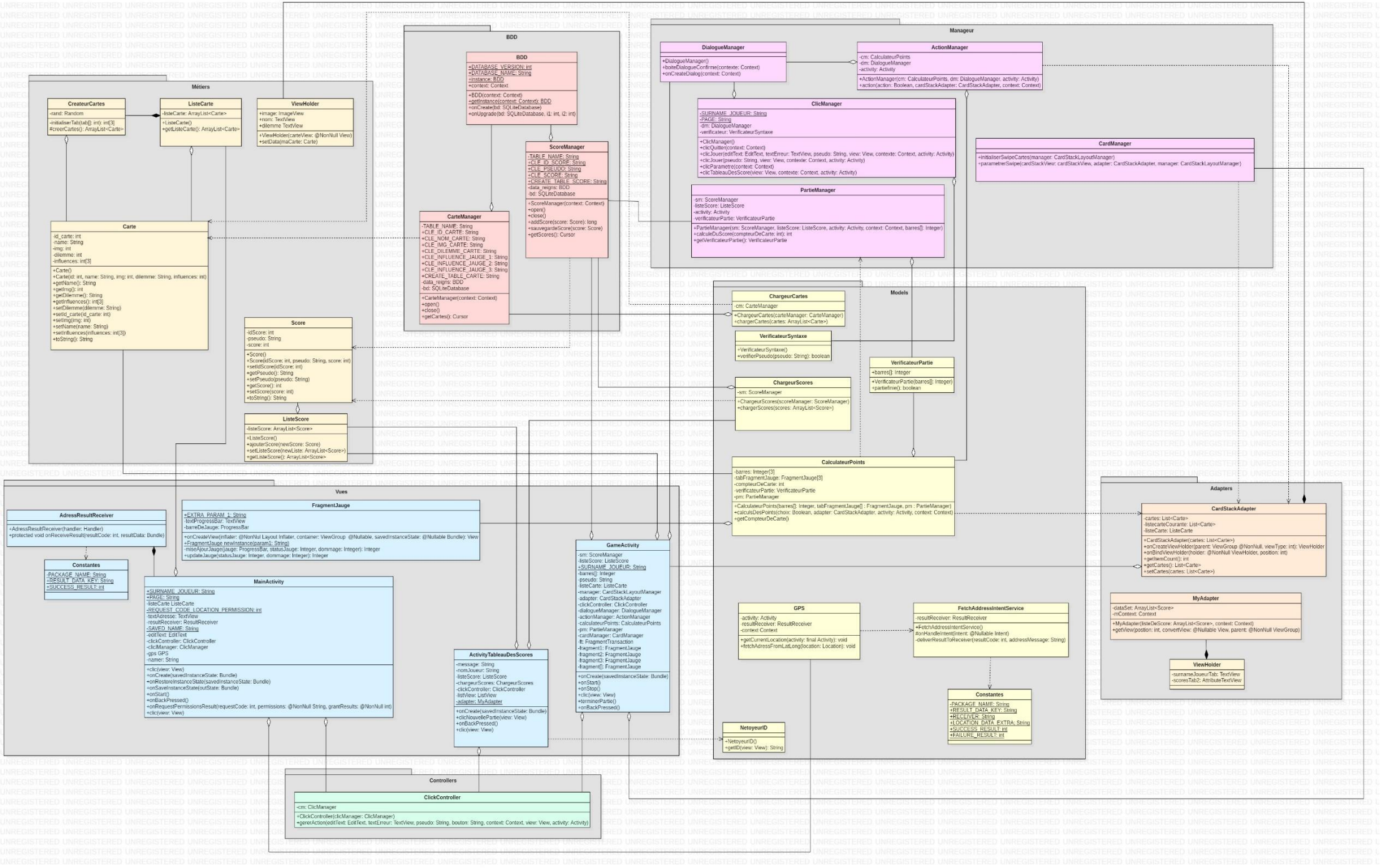


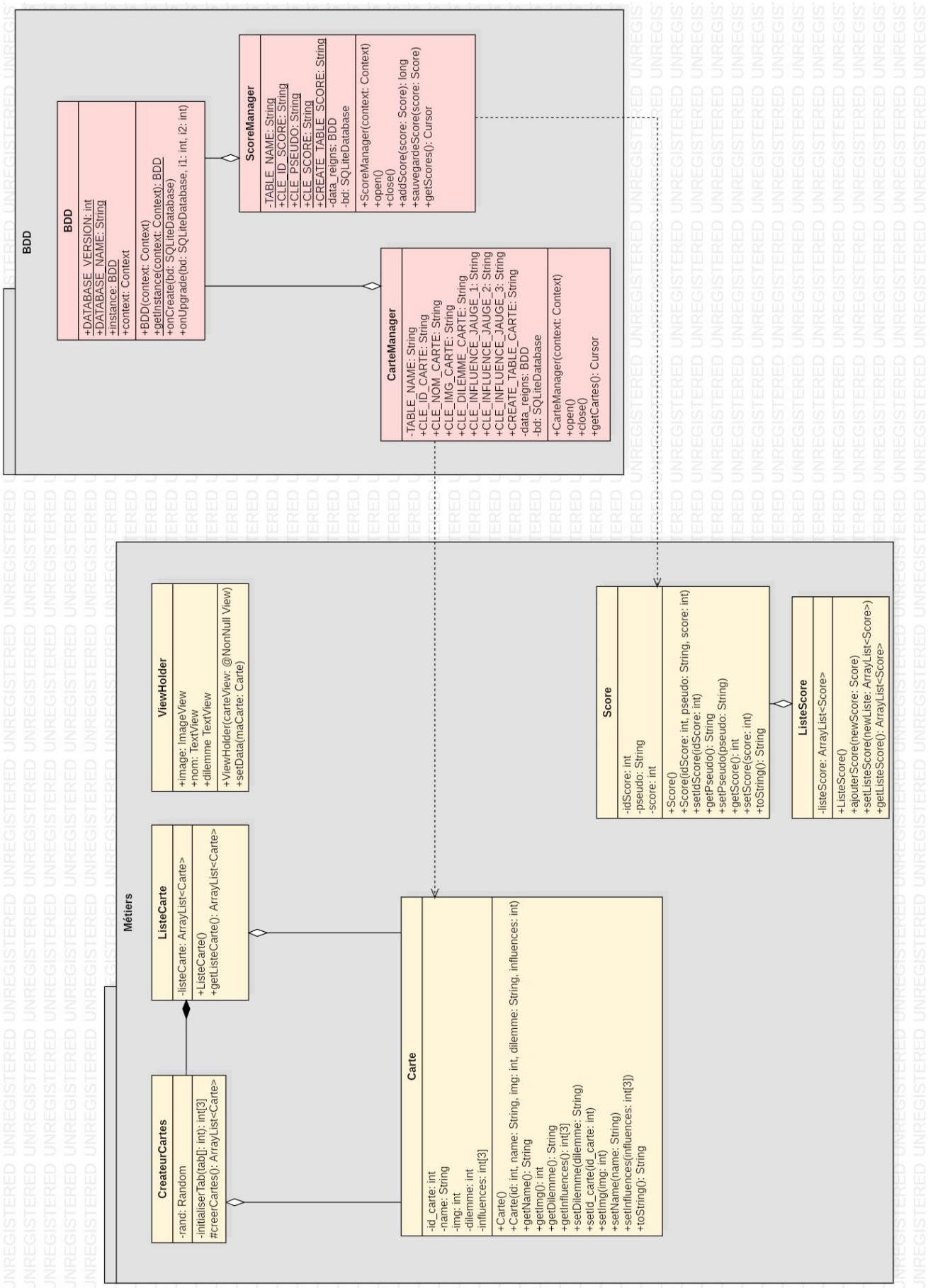
Diagramme UML (vue générale) :


Métiers
CreateurCartes
-rand: Random
-initialiserTab(tab[]: int): int[3]
+creerCartes(): ArrayList<Carte>
ListeCarte
-listeCarte: ArrayList<Carte>
+ListeCarte()
+getListeCarte(): ArrayList<Carte>
ViewHolder
+image: ImageView
+nom: TextView
+dilemme TextView
+ViewHolder(carteView: @NonNull View)
+setData(maCarte: Carte)
Carte
-id_carte: int
-name: String
-img: int
-dilemme: int
-influences: int[3]
+Carte()
+Carte(id: int, name: String, img: int, dilemme: String, influences: int)
+getName(): String
+getImg(): int
+getDilemme(): String
+getInfluences(): int[3]
+setDilemme(dilemme: String)
+setId_carte(id_carte: int)
+setImg(img: int)
+setName(name: String)
+setInfluences(influences: int[3])
+toString(): String
Score
-idScore: int
-pseudo: String
-score: int
+Score()
+Score(idScore: int, pseudo: String, score: int)
+setIdScore(idScore: int)
+getPseudo(): String
+setPseudo(pseudo: String)
+getScore(): int
+setScore(score: int)
+toString(): String
ListeScore
-listeScore: ArrayList<Score>
+ListeScore()
+ajouterScore(newScore: Score)
+setListeScore(newListe: ArrayList<Score>)
+getListeScore(): ArrayList<Score>
BDD
BDD
+DATABASE_VERSION: int
+DATABASE_NAME: String
+instance: BDD
+context: Context
+BDD(context: Context)
+getInstance(context: Context): BDD
+onCreate(bd: SQLiteDatabase)
+onUpgrade(bd: SQLiteDatabase, i1: int, i2: int)
ScoreManager
-TABLE_NAME: String
+CLE_ID_SCORE: String
+CLE_PSEUDO: String
+CLE_SCORE: String
+CREATE_TABLE_SCORE: String
-data_reigns: BDD
-bd: SQLiteDatabase
+ScoreManager(context: Context)
+open()
+close()
+addScore(score: Score): long
+sauvegardeScore(score: Score)
+getScores(): Cursor
CarteManager
-TABLE_NAME: String
+CLE_ID_CARTE: String
+CLE_NOM_CARTE: String
+CLE_IMG_CARTE: String
+CLE_DILEMME_CARTE: String
+CLE_INFLUENCE_JAUGE_1: String
+CLE_INFLUENCE_JAUGE_2: String
+CLE_INFLUENCE_JAUGE_3: String
+CREATE_TABLE_CARTE: String
-data_reigns: BDD
-bd: SQLiteDatabase
+CarteManager(context: Context)
+open()
+close()
+getCartes(): Cursor
Manageur
DialogueManager
+DialogueManager()
+boiteDialogueConfirme(contexte: Context)
+onCreateDialog(context: Context)
ActionManager
-cm: CalculateurPoints
-dm: DialogueManager
-activity: Activity
+ActionManager(cm: CalculateurPoints, dm: DialogueManager, activity: Activity)
+action(action: Boolean, cardStackAdapter: CardStackAdapter, context: Context)
ClicManager
-SURNAME_JOUEUR: String
-PAGE: String
-dm: DialogueManager
-verificateur: VerificateurSyntaxe
+ClicManager()
+clicQuitter(context: Context)
+clicJouer(editText: EditText, textErreur: TextView, pseudo: String, view: View, contexte: Context, activity: Activity)
+clicJouer(pseudo: String, view: View, contexte: Context, activity: Activity)
+clicParametre(context: Context)
+clicTableauDesScore(view: View, contexte: Context, activity: Activity)
CardManager
+initialiserSwipeCartes(manager: CardStackLayoutManager)
+parametrerSwipe(cardStackView: cardStackView, adapter: CardStackAdapter, manager: CardStackLayoutManager)
PartieManager
-sm: ScoreManager
-listeScore: ListeScore
-activity: Activity
-verificateurPartie: VerificateurPartie
+PartieManager(sm: ScoreManager, listeScore: ListeScore, activity: Activity, context: Context, barres[]: Integer)
+calculeDuScore(compteurDeCarte: int): int
+getVerificateurPartie(): VerificateurPartie
Models
ChargeurCartes
-cm: CarteManager
+ChargeurCartes(carteManager: CarteManager)
+chargerCartes(cartes: ArrayList<Carte>)
VerificateurSyntaxe
+VerificateurSyntaxe()
+verifierPseudo(pseudo: String): boolean
VerificateurPartie
+barres[]: Integer
+VerificateurPartie(barres[]: Integer)
+partieFinie(): boolean
ChargeurScores
-sm: ScoreManager
+ChargeurScores(scoreManager: ScoreManager)
+chargerScores(scores: ArrayList<Score>)
CalculateurPoints
-barres: Integer[3]
-tabFragmentJauge: FragmentJauge[3]
-compteurDeCarte: int
-verificateurPartie: VerificateurPartie
-pm: PartieManager
+CalculateurPoints(barres[]: Integer, tabFragmentJauge[] : FragmentJauge, pm : PartieManager)
+calculsDesPoints(choix: Boolean, adapter: CardStackAdapter, activity: Activity, context: Context)
+getCompteurDeCarte()
Vues
AdressResultReceiver
+AdressResultReceiver(handler: Handler)
+protected void onReceiveResult(resultCode: int, resultData: Bundle)
Constantes
-PACKAGE_NAME: String
+RESULT_DATA_KEY: String
+SUCCESS_RESULT: int
FragmentJauge
+EXTRA_PARAM_1: String
-textProgressBar: TextView
-barreDeJauge: ProgressBar
+onCreateView(inflater: @NonNul Layout Inflater, container: ViewGroup @Nullable, savedInstanceState: @Nullable Bundle): View
+FragmentJauge newInstance(param1: String)
-miseAjourJauge(jauge: ProgressBar, statusJauge: Integer, dommage: Integer): Integer
+updateJauge(statusJauge: Integer, dommage: Integer): Integer
MainActivity
+SURNAME_JOUEUR: String
+PAGE: String
-listeCarte ListeCarte
-REQUEST_CODE_LOCATION_PERMISSION: int
-textAdresse: TextView
-resultReceiver: ResultReceiver
-SAVED_NAME: String
-editText: EditText
-clickController: ClickController
-clicManager: ClicManager
-gps GPS
-namer: String
+clic(view: View)
+onCreate(savedInstanceState: Bundle)
+onRestoreInstanceState(savedInstanceState: Bundle)
+onSaveInstanceState(outState: Bundle)
+onStart()
+onBackPressed()
+onRequestPermissionsResult(requestCode: int, permissions: @NonNull String, grantResults: @NonNull int)
+clic(view: View)
ActivityTableauDesScores
-message: String
-nomJoueur: String
-listeScore: ListeScore
-chargeurScores: ChargeurScores
-clickController: ClickController
-listView: ListView
-adapter: MyAdapter
+onCreate(savedInstanceState: Bundle)
+clicNouvellePartie(view: View)
+onBackPressed()
+clic(view: View)
GameActivity
-sm: ScoreManager
-listeScore: ListeScore
+SURNAME_JOUEUR: String
-barres[]: Integer
-pseudo: String
-listeCarte: ListeCarte
-manager: CardStackLayoutManager
-adapter: CardStackAdapter
-clickController: ClickController
-dialogueManager: DialogueManager
-actionManager: ActionManager
-calculateurPoints: CalculateurPoints
-pm: PartieManager
-cardManager: CardManager
-ft: FragmentTransaction
-fragment1: FragmentJauge
-fragment2: FragmentJauge
-fragment3: FragmentJauge
-fragment[]: FragmentJauge
+onCreate(savedInstanceState: Bundle)
+onStart()
+onStop()
+clic(view: View)
+terminerPartie()
+onBackPressed()
GPS
-activity: Activity
-resultReceiver: ResultReceiver
-context Context
+getCurrentLocation(activity: final Activity): void
+fetchAdressFromLatLong(location: Location): void
FetchAddressIntentService
-resultReceiver: ResultReceiver
+FetchAddressIntentService()
#onHandleIntent(intent: @Nullable Intent)
-deliverResultToReceiver(resultCode: int, addressMessage: String)
Constantes
-PACKAGE_NAME: String
+RESULT_DATA_KEY: String
+RECEIVER: String
+LOCATION_DATA_EXTRA: String
+SUCCESS_RESULT: int
+FAILURE_RESULT: int
NetoyeurID
+NetoyeurID()
+getID(view: View): String
Adapters
CardStackAdapter
-cartes: List<Carte>
-listecarteCourante: List<Carte>
-listeCarte: ListeCarte
+CardStackAdapter(cartes: List<Carte>)
+onCreateViewHolder(parent: ViewGroup @NonNull, viewType: int): ViewHolder
+onBindViewHolder(holder: @NonNull ViewHolder, position: int)
+getItemCount(): int
+getCartes(): List<Carte>
+setCartes(cartes: List<Carte>)
MyAdapter
-dataSet: ArrayList<Score>
-mContext: Context
+MyAdapter(listeDeScore: ArrayList<Score>, context: Context)
+getView(position: int, convertView: @Nullable View, parent: @NonNull ViewGroup)
ViewHolder
-surnameJoueurTab: TextView
-scoresTab2: AttributeTextView
Controllers
ClickController
-cm: ClicManager
+ClickController(clicManager: ClicManager)
+gererAction(editText: EditText, textErreur: TextView, pseudo: String, bouton: String, context: Context, view: View, activity: Activity)

Barou Romain
Chanson Chloé

Dossier Android – Compte rendu
Application « Reing rise »

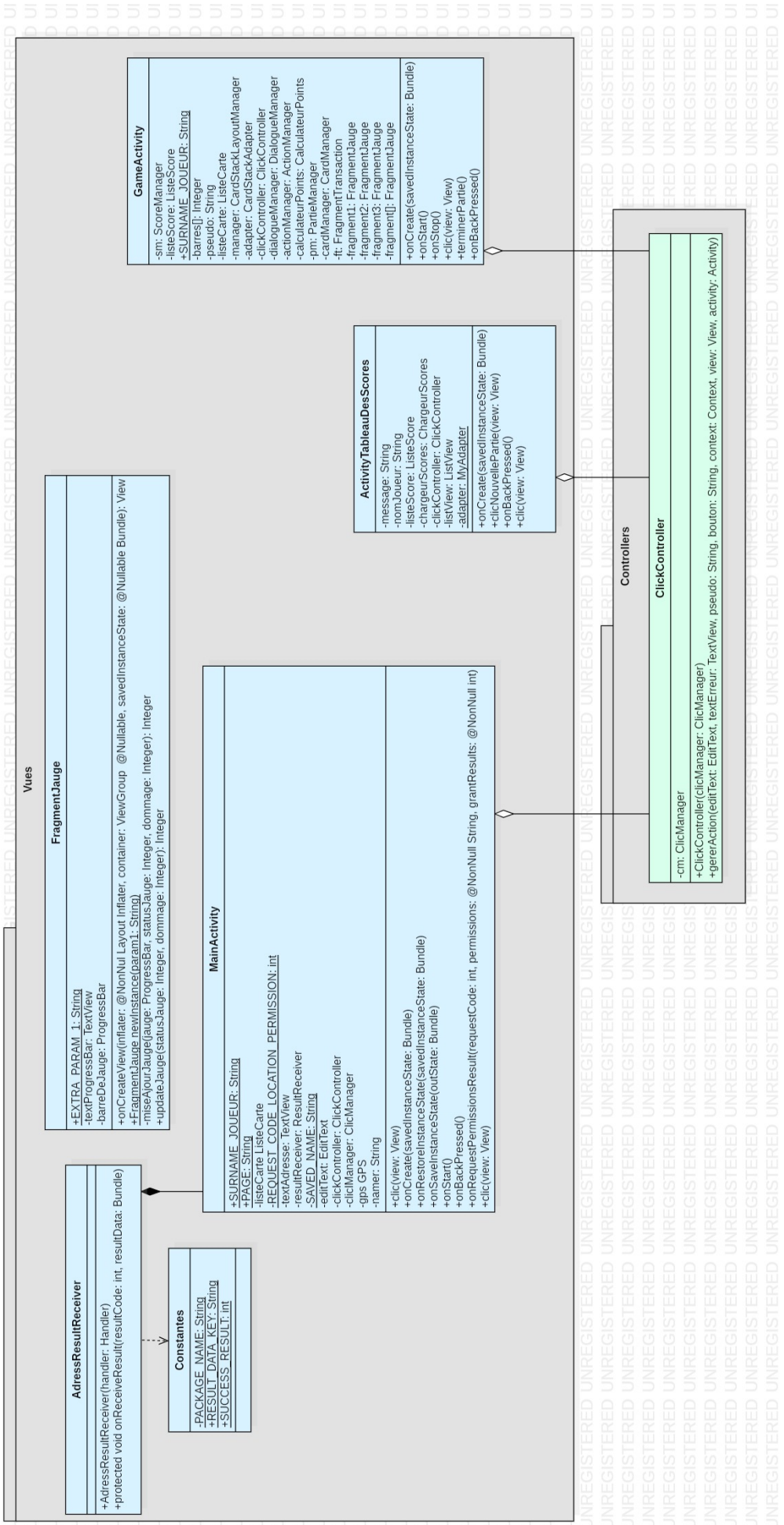
29/03/2020



Barou Romain
Chanson Chloé

Dossier Android – Compte rendu
Application « Reing rise »

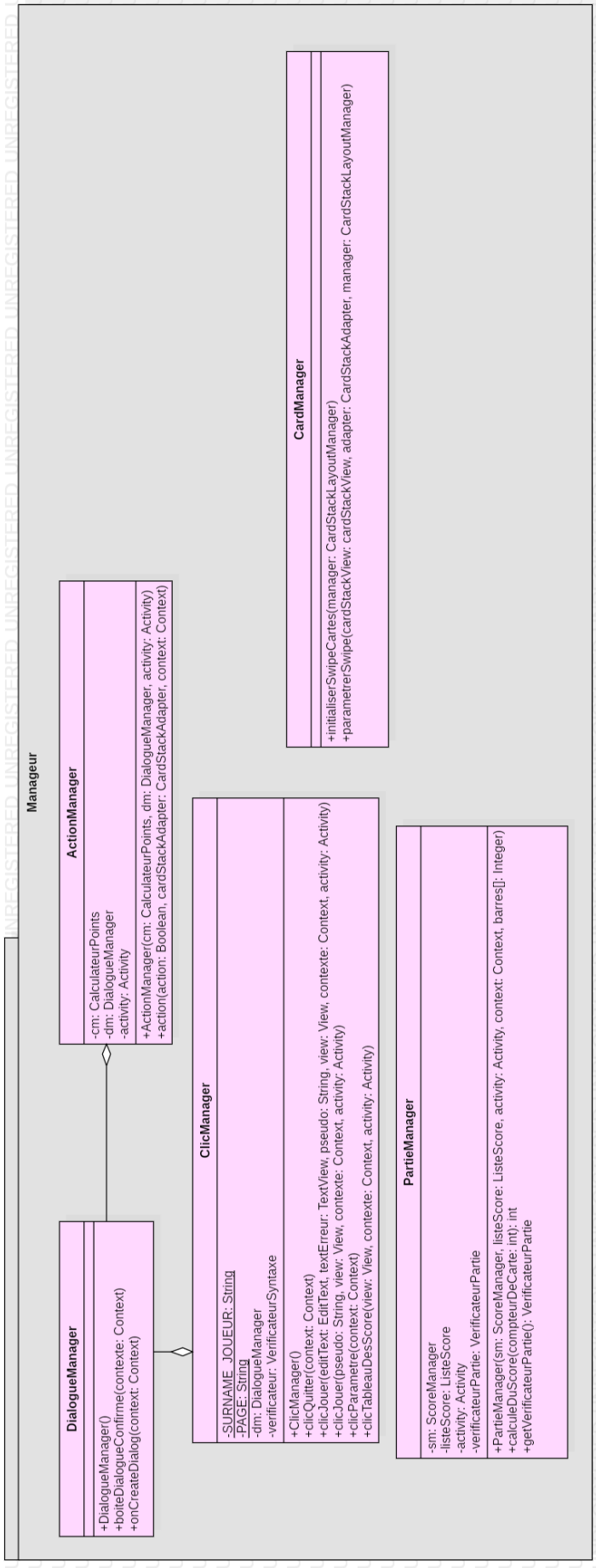
29/03/2020



Barou Romain
Chanson Chloé

Dossier Android – Compte rendu
Application « Reing rise »

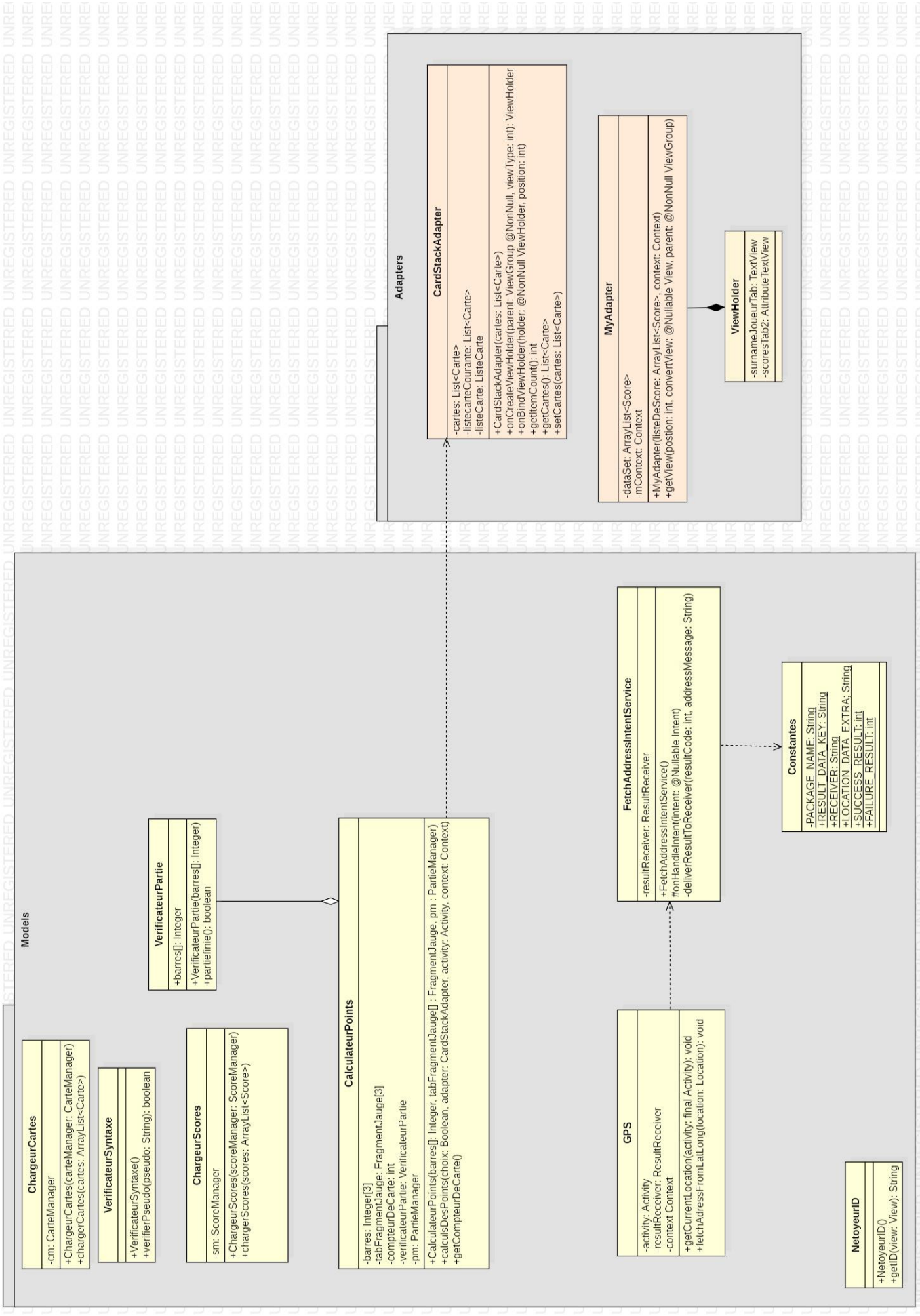
29/03/2020



Barou Romain
Chanson Chloé

Dossier Android – Compte rendu
Application « Reing rise »

29/03/2020



Barou Romain
Chanson Chloé

Dossier Android – Compte rendu
Application « Reing rise »

29/03/2020

Description du diagramme UML (vue générale) :

Afin de réaliser notre diagramme de classe, nous avons essayé de séparer au maximum les classes sous divers packages afin de repartir au mieux les responsivités entre chaque package et chaque classe. Ainsi nous avons dans notre projet 7 packages distincts représenté par 7 couleurs différentes dans le diagramme UML afin de mieux pouvoir les reconnaitre.

Tout d'abord, le package « Adapters », contient toutes les classes d'adapter qui servent à terme à afficher les données : affichage des cartes et affichage des scores.

Le package « Metier » contient toutes les classes servant à contenir de la donnée : Carte, ListeDeCarte, Score, ListeDeScore ...

Le package « BDD » est relatif à la gestion de la persistance profonde. Pour cette dernière, nous avons choisi d'utiliser une base de données SQLite, la classe BDD permet de générer la base SQLite, et les 2 managers présents permettent de faire le lien entre les classes métiers scores et classes et leur stockage en base.

Le package « Vues » représente l'assemble de nos classes qui correspondent au code behind de nos layouts.

Le package « Controller » permet de gérer les différentes actions de l'utilisateur, qui appelle les fonctions du package manageur.

Le package « Manageur » ce package regroupe les fonctions qui permet les différentes actions de l'utilisateur.

Et enfin, le package « Modele » contient l'ensemble des classes dont ce servent les autres packages.

Ces différentes séparations nous a permis que les actions d'utilisateurs passent par un clic controller. Par la suite les actions sont déléguées à un des managers correspondants. L'action tel que la partie, les clics, les dialogues (messages d'erreurs), l'initialisation de l'affichage des cartes est alors traitée par ce manageur.

Nous vous présentons ci-dessous un exemple d'interaction entre les différentes classes du diagramme lors d'une partie :

Lorsque l'utilisateur démarre l'application, il arrive sur la mainActivity. Lorsqu'il clic sur un des boutons (écran des scores, paramètres ou clique pour jouer), son action est transmise et nettoyé (pour ne conserver que le nom de l'action) via à la méthode Clic() au clicController qui va indiquer au clicManager la bonne méthode à appliquer. Dans le cas où il clique pour jouer, son action va passer par un vérificateur de syntaxe pour bien vérifié que le pseudo soit valide (un pseudo valide doit contenir au moins un caractère visible c'est-as-dire que nous n'acceptons pas les pseudos composer que d'espace par exemple, cela oblige aussi l'utilisateur à entrer un pseudo). Par la suite le joueur arrive sur la gameActivity. A ce stade, les cartes ont été soit créés, chargées via un chargeur de carte ou grâce au constructeur de carte. Le joueur peut alors, commencé à faire ses choix qui vont déterminer l'évolution des barres grâce à l'utilisation de l'attribut influence de la classe Carte. Ces évolutions de barre vont être remarquées dans les barres en haut de l'écran qui sont des fragments (classe fragments barres). A chaque choix les points sont recalculés, le vérificateur de partie est également appelé pour vérifier si la partie doit se terminer ou non (une partie est terminé si une des barres est au-dessus de 100 points). Si c'est le cas, le partieManager est appelé et se charge de calculer le score de fin de partie, l'utilisateur est redirigé dans l'activité tableau des scores, les scores présents en base de données sont chargés via le chargeurScore, et notre nouveau score est sauvegardé en base de données en passant par le scoremanager. A partir d'ici, le joueur peut observer les scores des anciennes parties réalisées sur son portable et peut s'il le souhaite relancer une partie en cliquant sur le boutton « nouvelle partie ». Ce clique appellera de nouveau le clicController qui indiquera au ClicManager de réalisé la bonne action .