VINCI ANDROID PROGRAMMER HANDBOOK

Page de service

Référence : N/A
Plan de classement : android, spritebot cybersecurity game
Niveau de confidentialité : Corporate

Mises à jour :

VERSION	DATE	AUTEUR	DESCRIPTION
1.0	03/01/2024	Thomas PRADEAU	Création du document
1.1	07/01/2024	Thomas PRADEAU	Ajout de la gestion de projet

Validations :

VERSION	DATE	NOM	RÔLE
1.1	07/01/2024	Jérôme VALENTI	Enseignant

Diffusions :

VERSION	DATE	NOM	CADRE DE LA DIFFUSION

Sommaire

Table des matières

Analyse conception

Définition des besoins

Comme plus de la moitié des entreprises françaises, le groupe Vinci fait régulièrement l'objet d'attaques informatiques :

https://www.lerevenu.com/bourse/la-chronologie-de-lattaque-sur-vinci-reconstituee

Pour sensibiliser les collaborateurs du groupe à la cybersécurité, la DRH et la DSI ont décidé de financer le développement d'une plateforme de quiz qui pourra par la suite évoluer vers un jeu en ligne : **Sprite Bot Cybersecurity Game - SBCG.**

Ce jeu est un outil d'acculturation des collaborateurs Vinci à la cybersécurité (cf. https://www.ssi.gouv.fr/).

Le jeu est jouable en deux versions :

- une version mono-joueur
- une version multi-joueur

En 2016, un simple communiqué de presse et en quelques minutes, la cotation en bourse du groupe Vinci a chuté de 19% (https://www.daf-mag.fr/Thematique/gestion-risque-1241/Breves/Vinci-victime-d-une-fraude-au-communique-de-presse-311088.htm). Depuis, le nombre de cyberattaques sur le groupe ne cessent de croitre.

Le groupe Vinci a décidé de sensibiliser et former à la cybersécurité, ses 268 000 collaborateurs dans le monde. La sensibilisation se fera par un jeu éducatif (https://en.wikipedia.org/wiki/Serious_game | https://fr.wikipedia.org/wiki/Jeu_s%C3%A9rieux).

Cas d'utilisation

La figure suivante présente les cas d'utilisation sous forme d'un diagramme UML :

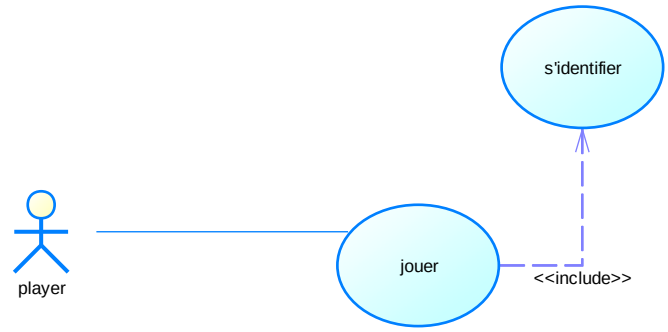


Figure 1 : Diagramme des cas d'utilisation UML.

Dans le diagramme ci-dessous, l'acteur principal, le joueur, dispose de deux cas d'utilisation définis. Celui-ci peut jouer une partie mais est dans l'obligation de s'authentifier afin de commencer la partie.

Le cas d'utilisation inclus donc le cas s'authentifier.

Présentation des cas d'utilisation

Cas s'authentifier :

L'application se lance. Le joueur est invité à taper son identifiant ainsi que son mot de passe. Si l'authentification réussis, le jouer est redirigé sur le menu principal sur lequel il peut choisir de jouer en solo ou multi-joueur.

Cas jouer une partie :

Une fois authentifié, le joueur choisis de jouer seul ou à plusieurs. S'il choisit de jouer seul, il clique sur le bouton « Solo » pour débuter une partie. Sinon, il clique sur un bouton « Multijoueur », qui le mène vers un menu où le joueur choisis d'intégrer une partie existante, ou alors de créer sa propre partie.

Diagrammes de classes

La figure ci-dessous présente le diagramme de classes métier :

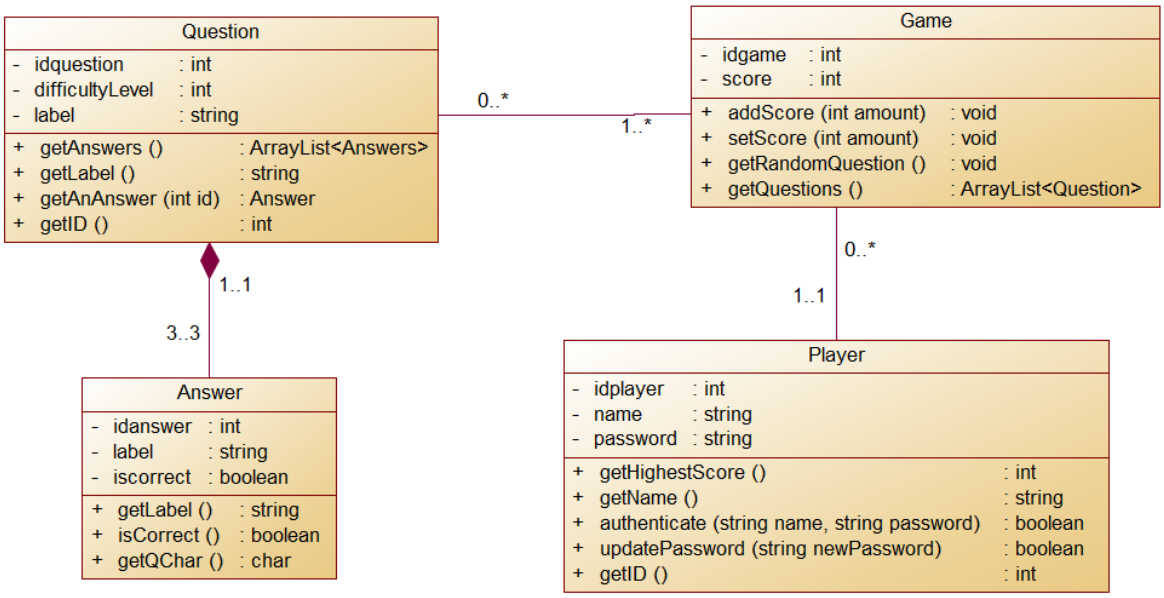


Figure 2 : Diagramme de classes UML.

Diagramme de séquence

La figure suivante montre le diagramme de séquence en boîte noire de l'application :

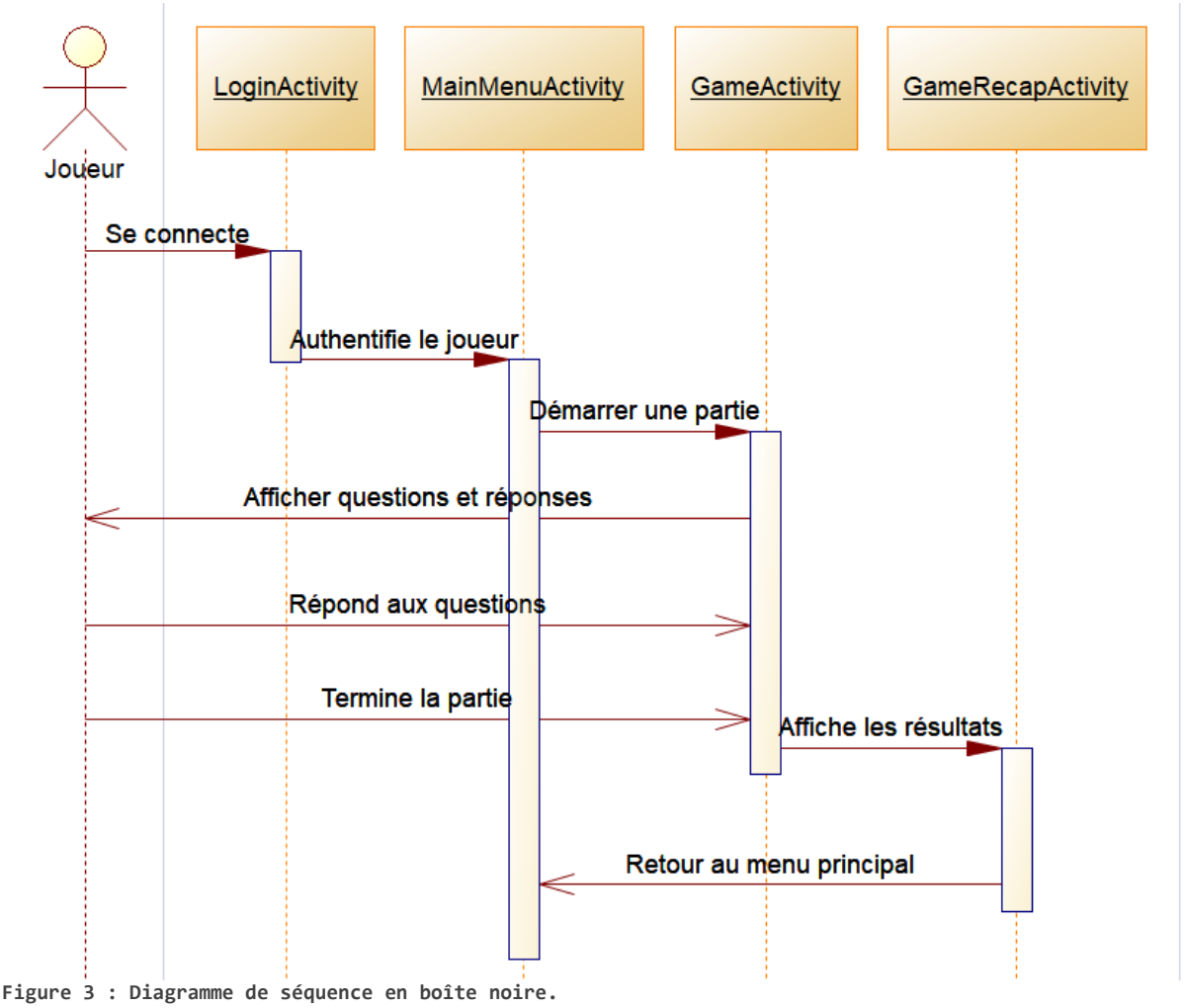


Figure 3 : Diagramme de séquence en boîte noire.

Ce diagramme représente une situation d'utilisation classique et sans échecs. Le joueur joue une partie composée de N questions, puis affiche le récapitulatif à la fin de sa partie, et enfin retourne au menu principal.

Maquettes IHM

Les quatre maquettes suivantes présentent les différents écrans de l'applications, en outre la connexion, le menu principal, la partie en cours et enfin le récapitulatif de la partie.

Authentification

La figure suivante montre la maquette de l'écran de connexion :

Figure 4 : Maquette de l'écran de connexion.

Cette maquette reprend et répond au cas d'utilisation « s'authentifier ». L'utilisateur utiliser les champs dédiés pour saisir son mot de passe et son identifiant, puis il clique sur le bouton « Login » pour s'authentifier.

Menu principal

Cet écran fait le lien entre la connexion et la partie en cours. Celui-ci contient un simple bouton permettant de démarrer une partie en solo, faisant donc le lien avec le cas d'utilisation « Jouer une partie ».

Dans de futures versions, il sera possible de démarrer une partie en multi-joueur, mais également de voir les scores des dernières parties du joueur connecté et accéder à un menu de paramètres pour modifier certains aspects de l'application, tels que la langue ou le thème de l'interface (clair ou sombre).

La figure ci-dessous montre la maquette représentant le menu d'accueil :

Figure 5 : Maquette de l'écran d'accueil.

Jouer une partie

Cet écran contient la partie en cours démarrée par un joueur, quelle soit en solo ou multijoueur. Dans cet écran, les questions sont affichées une à une avec deux boutons en bas de l'écran.

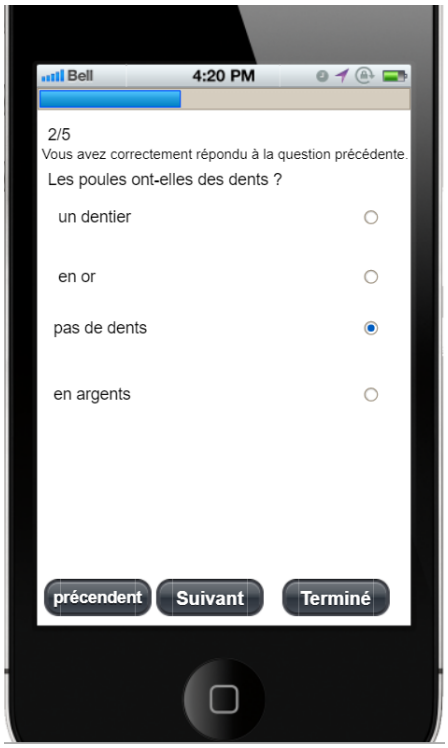


Figure 6 : Maquette du cas d'utilisation « Jouer ».

Le joueur ne peut valider la partie tant que celui-ci n'a pas répondu à toutes les questions. Si la toute dernière question est affichée et que toutes les questions ont été répondues, alors le bouton « Suivant » peut être utiliser pour valider la partie et soumettre les réponses choisies.

Une barre de progression se trouve en haut de l'écran pour montrer au joueur son avancée dans la partie, celle-ci peut être utiliser pour comparer différents joueurs dans le cas d'une épreuve de vitesse par exemple.

La figure ci-contre montre la maquette de cet écran :

Récapitulatif

Cet écran affiche le compte des points ainsi que les réponses choisies par le joueur et si celles-ci sont bonnes ou fausses.

Un bouton est présent en bas pour permettre au joueur de retourner au menu principal et de recommencer une partie ou quitter l'application.

La figure ci-dessous montre cette maquette :

Figure 7 : Maquette du récapitulatif d'une partie.

Conception

Architecture

Android dispose d'une toute autre architecture. En effet, le système d'exploitation Android fonctionne différemment comparer aux systèmes conventionnels utilisés sur des ordinateurs de bureau ou portable.

Pour faire simple, Android gère les applications sous forme de tâches, ou processus indépendants, eux-mêmes composés de blocs plus petits, eux aussi indépendants. Ces plus petits blocs sont appelés « Activités », et composent la logique ainsi que l'affichage des composants graphiques de l'application.

L'objectif premier de cette application est de, rappelons-le, de réutiliser au maximum de code déjà écrit en Java, et ce, afin d'éviter d'avoir à gérer toute une équipe uniquement dédiée à la gestion de l'application Java. La réutilisation de la codebase existant permet donc de simplifier la maintenance et la mise à jour de l'application.

La version client lourd utilise une architecture MVC (Modèle Vue Contrôleur), où le contrôleur contient la logique applicative et communique bi directionnellement ment avec les modèles et les vues.

On utilisera ici un dérivé de cette architecture, adaptée à l'architecture typique des applications Android. La figure suivante présente cette architecture :

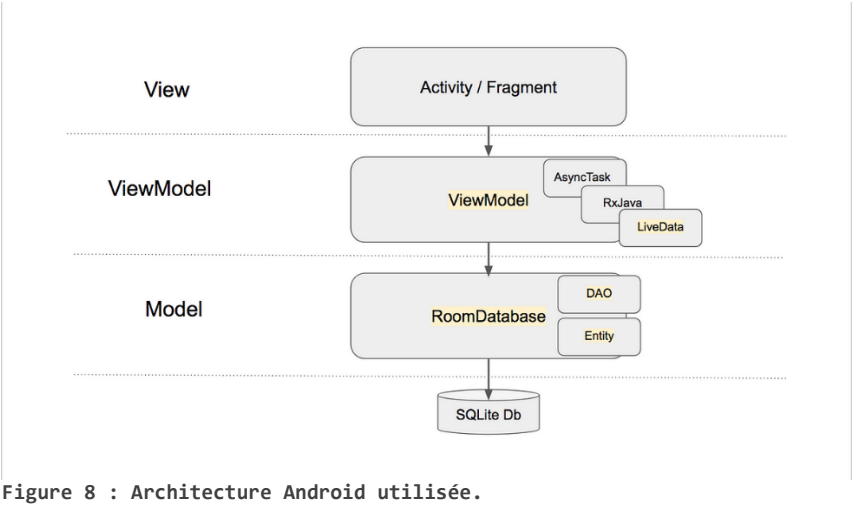


Figure 8 : Architecture Android utilisée.

Ici, l'activité va contenir la logique de l'application et gérer la vue (Activity et ViewModel). Les données utilisées l'or de l'affichage proviennent soit d'un DAO (Data Access Object), ou d'une base de données locale (SQLite), ou distante (MySQL, PostgreSQL, Oracle, …).

En résumé : Le fonctionnement et la logique ainsi que les données utilisées entre les deux clients ne changent pas. Les seuls changements concernent l'organisation du code afin de correspondre à l'architecture des applications Android.

Application de l'architecture

Comme expliqué précédemment, une application Android est segmentée en activités. Une activité représente dans la très grande majorité des cas une fenêtre de l'application, par exemple un écran de connexion, une fenêtre de jeu ou un menu de paramètres.

L'application **Vinci SpriteBot cybersecurity game** est donc découpée en quatre activités :

- Connexion utilisateur
- Menu principal (Hub)
- Ecran de jeu (Questions)
- Récapitulatif de la partie

Chaque activité sont composées des parties suivantes :

- **Classe** : Elle contient la logique de l'activité ainsi que les interactions avec la vue.

- **Layout XML :** Cette partie compose le squelette de la vue qui sera affiché sur l'écran de l'utilisateur. Elle définit au format XML les différents éléments comme les boutons, textes, menus, etc…
  Les éléments définis dans le layout peuvent être modifiés et adaptés directement depuis la classe.

Toutes les classes liées à des activités doivent hériter de la classe « AppCompatActivity[1] ». Cette classe définis toutes les méthodes pour rendre la classe utilisable avec la vue.

On trouve dans ces classes une méthode héritée nommé « onCreate[2] ». Cette méthode doit impérativement être présente dans la classe pour pouvoir être utilisé. En effet, cette méthode est appelée l'or de la création de l'activité. C'est en quelques sortes le constructeur de l'activité.
On va alors y définir les différents attributs utilisés dans l'activité.

Passage d'une activité à une autre :

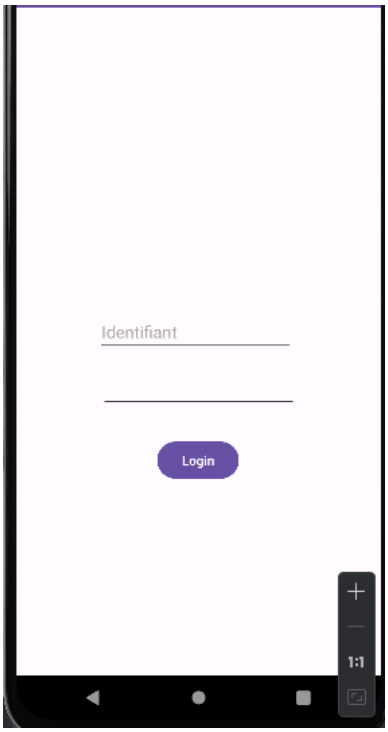
Les activités sont complètement séparées les unes des autres. Autrement dit, il n'est pas possible de faire passer un objet tel-quel d'une activité à une autre.

On peut cependant « **copier-coller** » un objet dans une autre activité. Ce procédé se nomme la « **Serialization** » ou « **Parcelation**[3] » sous Android. Cela permet de prendre un objet et tous ces attributs, de créer une nouvelle instance de la classe correspondante dans la nouvelle activité, puis de remettre les valeurs des attributs correspondants.

Bien que les valeurs des attributs soient les mêmes, il ne s'agit pas du même objet. Cela prouve bien que les activités sont complètement indépendantes les unes des autres.

[1] https://developer.android.com/reference/androidx/appcompat/app/AppCompatActivity
[2] https://stackoverflow.com/questions/19538976/what-is-a-oncreate-method-in-android
[3] https://www.sciencedirect.com/topics/engineering/parcellation

La figure suivante présente l'activité qui réponds au cas d'utilisation « s'authentifier » :



On y retrouve les deux champs correspondants à l'identifiant ainsi qu'au mot de passe, mais aussi le bouton permettant de valider l'authentification.

Cette activité fait appel au modèle Joueur pour effectuer l'authentification. Puisque le modèle se charge du lien avec la base de données, c'est lui qui va comparer les identifiants fournis par l'utilisateur avec ceux présents en base de données.

Si l'authentification réussis, alors un nouvel objet Joueur sera instancié correspondant aux identifiants renseignés, et celui-ci sera redirigé vers le menu principal.

Figure 9 : Activité d'authentification.

Remarque : Si la base de données MySQL n'est pas accessible, un message apparaîtra en haut de l'écran et le joueur ne pourra pas procéder à l'authentification.
Un bouton apparaît donnant la possibilité au joueur de retenter la connexion à la base de données, la figure ci-dessous montre cela :

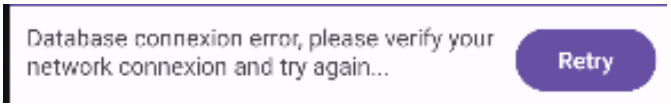


Figure 10 : Message d'erreur de connexion à la base de données.

La figure ci-dessous montre la méthode permettant de passe à l'activité suivante :

```
public void onLoginButtonClick(View v) {
    if(player.authenticate(username.getText().toString(), password.getText().toString())) {
        Intent gameIntent = new Intent( packageContext: this, MainMenuActivity.class);
        gameIntent.p
        this.startAc
    }
}
```

Intent gameIntent = new Intent(this, MainMenuActivity.class)
AndroidDevForBegginers.app.main

Figure 11 : Méthode appelé l'or de l'appui sur le bouton « Login ».

Cette méthode est appelée lorsque le joueur appui sur le bouton. Celle-ci va tout d'abord tenter l'authentification, si celle-ci se passe correctement, alors l'activité « MainMenuActivity » sera affiché.

Menu principal

La figure ci-dessous présente l'activité du menu principal :

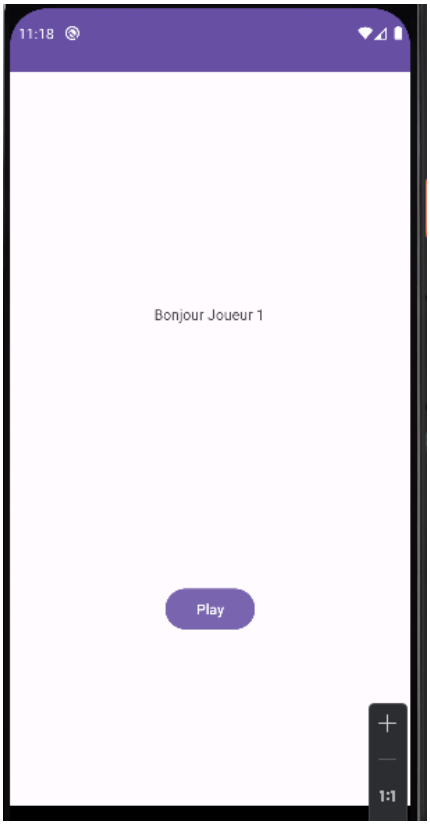


Figure 12 : Activité du menu principal.

Le menu principal permet au joueur de choisir entre les parties seul ou en multi joueur.
Celui-ci pourra également visionner les résultats de ses dernières parties et modifier les paramètres de l'application.

Remarque : Ces fonctionnalités sont prévues mais non implémentées dans la version actuelle de l'application.
Celle-ci ne constitue qu'un prototype permettant d'illustrer le fonctionnement global de l'application.

Lorsque le joueur clique sur le bouton « Play », celui-ci débute directement une partie en mono joueur en ligne. Cela signifie qu'il joue seul mais que le connexion réseau reste obligatoire pour pouvoir charger la partie en publier ses résultats.

La figure suivante montre l'activité représentant une partie en cours :

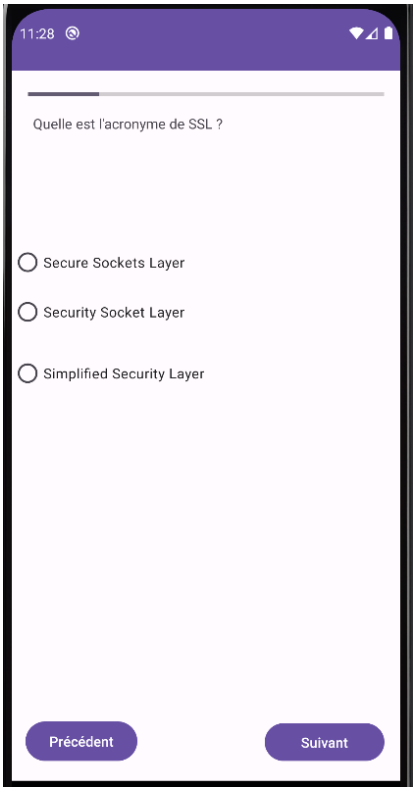


Figure 13 : Activité d'une partie en cours.

Cette activité est composée d'une barre de chargement représentant la progression du joueur dans la partie.

En dessous se trouve l'intitulé de la question, ainsi que les trois réponses proposées. Enfin, tout en bas de l'écran, se trouvent les deux boutons permettant de revenir à la question précédente ou d'aller à la question suivante.

Remarque : Les réponses ne peuvent être qu'au nombre de trois, car le layout XML ne permet pas d'afficher plus de trois réponses par question.

Cette activité contient la logique la plus complexe, puisqu'elle a la charge de la gestion complète d'une partie. On retrouve en réalité quasiment toutes les méthodes présentes dans la classe « **GameController** » du client lourd Java.

Cette logique comprend la gestion des appuis sur les boutons, l'enregistrement en mémoire vive des réponses choisies par le joueur et surtout l'avancée de la partie.
La principale problématique pour cette activité est de correctement suivre où en est le joueur et quoi faire à ce moment. Dans cette version, seule la gestion de la fin de la partie n'est gérée (lorsque le joueur à répondu à toutes les questions), mais à terme, d'autres modes de jeux pourront demander une gestion plus pointilleuse du suivi de la partie.

Récapitulatif

La figure ci-dessous présente l'activité du récapitulatif d'une partie :

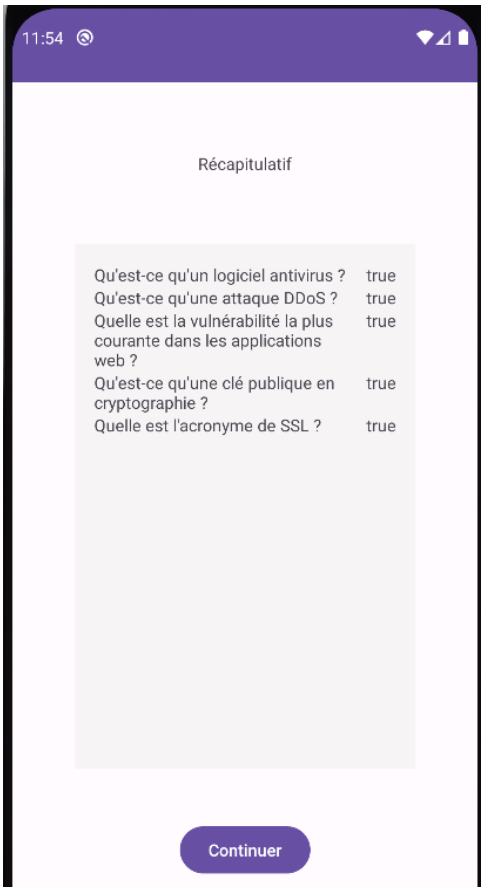


Figure 14 : Activité du récapitulatif de la partie jouée.

Cette activité affiche sous forme d'un tableau les questions posées au joueur et si celui-ci a correctement répondu ou non.

Remarque : Dans une future version, le joueur pourra, s'il le souhaite, cliquer une un question pour obtenir une des informations et sources supplémentaire sur le sujet pour comprendre son erreur et améliorer **l'aspect éducatif** de l'application.

Ici, les « true » et « false » montrent si le joueur a bien répondu ou non à la question posée dont l'intitulé est sur la ligne correspondante.

Ces valeurs seront remplacées par des icônes pour une meilleur lecture.

Le bouton « Continuer » permet de revenir au menu principal.

Persistance des données

L'application possède deux bases de données afin de faire fonctionner celle-ci dans le cas où il n'y aurait pas de connectivité réseau.

On utiliser une base de données MySQL sur un serveur distant pour que l'utilisateur puisse se connecter, jouer une partie et enregistrer ses résultats. Pour que l'application puisse être utiliser en mode hors ligne, on implémentera une base SQLite locale afin d'enregistrer les informations cruciales au bon fonctionnement de l'application.

Base MySQL

La base MySQL est la base de données principale, commune à tous les utilisateurs qui fournit l'intégralité des services de l'application, en outre l'authentification, les question et enregistrement des résultat.

Remarque : La gestion des parties multijoueur n'est pas prise en compte par ce schéma.

La figure ci-dessous montre le modèle physique de données (MPD) utilisé :

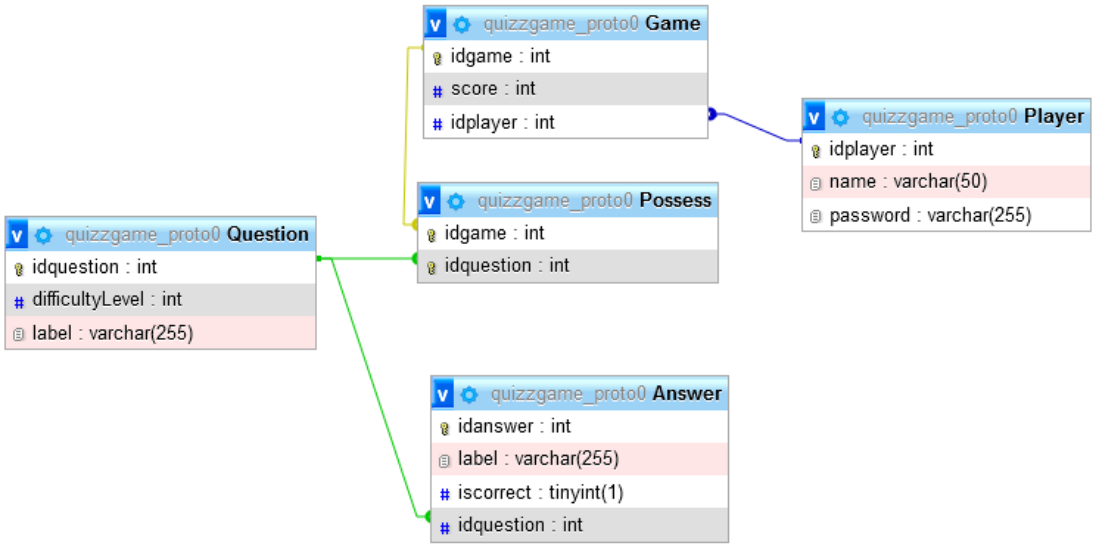


Figure 15 : Modèle physique de données de la base MySQL distante (Visualisation sous PHPMyAdmin).

Base SQLite

La base SQLite quant à elle a pour but de supplanter la base MySQL lorsque celle-ci devient inaccessible. En outre on y enregistrera les informations dont l'application a besoin pour fonctionner hors ligne.

Cette base utiliser le même schéma relationnel que la base MySQL pour des raisons de compatibilité des données. La différence entre ces deux bases se fait au niveau de l'implémentation dans l'application.

Organisation du projet

La figure suivante présente le diagramme de Gantt correspondant à l'organisation du projet convenue :

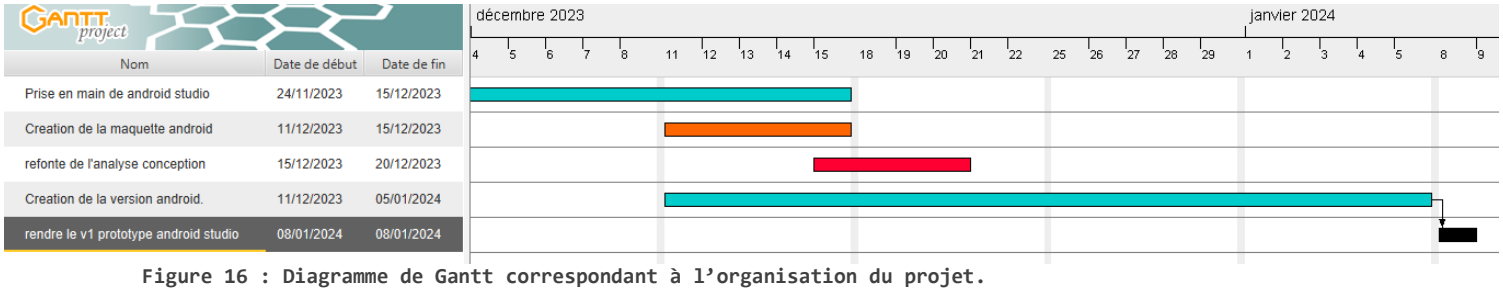


Figure 16 : Diagramme de Gantt correspondant à l'organisation du projet.

Nous avons choisi de faire un diagramme peut précis au niveau des tâches mais facilement lisible. Le projet tourne autour de ces 5 tâches principales :

- Prise en main d'Android Studio
- Création des maquettes Android
- Refonte de l'analyse conception (Adaptation à Android)
- Création de la version Android
- Livraison de la version 1 prototype.

La figure ci-dessous montre la répartition approximative entre les membres de l'équipe de développement :

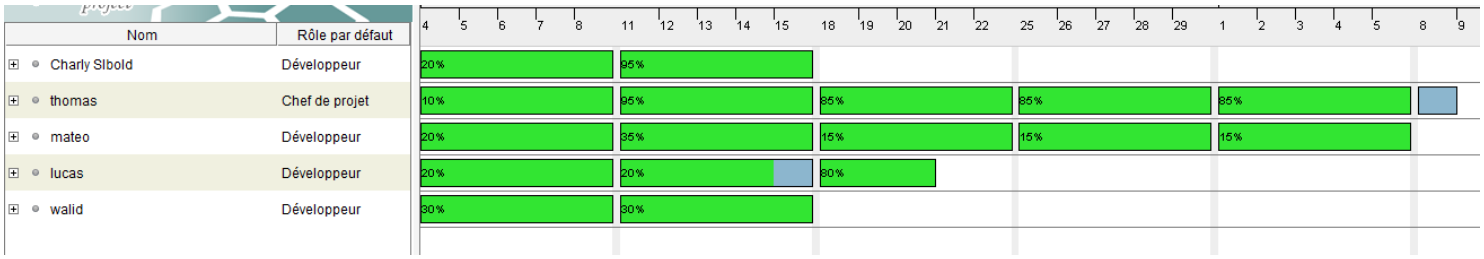


Figure 17 : Répartition des tâches entre les membres de l'équipe en pourcentage.

Les tâches ont été confié en prenant en compte les points forts et faibles de chacun, mais également le niveau global en technique et analyse.

Ainsi, **Lucas, Charly et Walid** se sont principalement concentrés sur l'analyse conception et le maquettage.
Enfin, **Matéo et Thomas** se sont occupés de la création de la version Android du client lourd.

Le chef de projet désigné est **Thomas**, et a eu également la charge de la livraison finale de la version.

Bibliographie

Vous pourrez retrouver la bibliographie consultée pour l'élaboration de ce Handbook dans le répertoire « biblio » du build, ou directement en ligne, via les URLs suivantes :

Table des illustrations