Trivial Diamond

*Rapport d’étude préliminaire*

Melissa Boucher et Charlie Laplante

27/02/2015

**Table des matières**

Introduction

Objectif du document 2

Portée du document 2

Références 2

Problématique 2

Définitions 2

Description de contexte

Diagramme des cas d’utilisation 3

Description des cas d’utilisation 4

Cas 1 4

Cas 2 5

Cas 3 6

Cas 4 7

Diagramme de classes UML 8

Diagramme de package 8

Conclusion

évaluer la faisabilité 9

Droits d’auteur et de reproduction 9

Solutions d’implémentation 10

Échéancier d’implémentation 10

Annexes

Modèle Relationnel 11

Modèle Conceptuel 12

Modèle Entité relation 13

Prototype de l’application 14

**Introduction**

## Objectif du document

Ce document servira à prouver que la demande du client a bien été comprise et ainsi démontrer l’analyse qui en a été fait. Une analyse de modélisation, des clarifications de l’énoncé et des besoins du client ainsi que des évaluations de cas et de solutions auront été fait avant l’écriture de ce document afin de permettre de comprendre le projet à qui le veux bien.

## Portée du document

Le document présent s’adresse à toute personne pouvant lire et comprendre des diagrammes de cas d’utilisations et d’UML, soit les programmeurs qui devront écrire le programme. Le client (Saliha Yacoub) pourra également y retrouver un résumé des fonctionnalités du système pour vérifier si nous avons compris l’énoncé.

## Références

* L’énoncé remis par le professeur
* Période de questions répondu par le professeur
  + Est-ce que les joueurs peuvent abandonner n’importe quand?
  + Clarifier la situation de gagner ? (Nombre de question à répondre par catégorie)
  + Est-ce qu’on doit gérer plusieurs parties ?
  + Doit-on avoir une table score ou pouvons-nous gérer le nombre de bonne réponse avec des counts ?

## Problématique

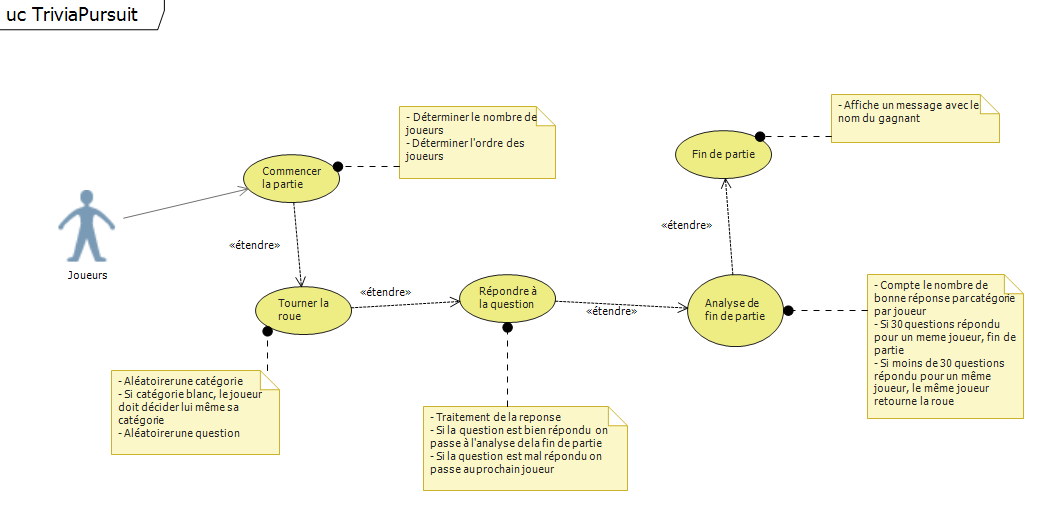
Le client désire un jeu nommé Trivial Diamond. Il s’agit d’un jeu questionnaire pouvant être joué de 2 à 4 joueurs, 4 catégories de questions sont disponible pour les divertir. Chacun leur tour, ils devront répondre à des questions. Le jeu se termine lorsqu’un joueur réussi à répondre correctement à 3 questions de chaque catégorie. Ce jeu doit être composé d’une base de données Oracle ainsi que d’un programme fait à partir du langage Ado.Net.

***Définitions***

* **ADO.NET** : Bibliothèque logicielle de Microsoft fournissant une interface d’accès aux données dans l’environnement Windows.
* **Implémentation** : Mise en place de fonctionnalité informatique.
* **SQL :** *Structured Query Language* est langage informatique normalisé servant à exploiter des bases de données

**Description de contexte**

## Diagramme de cas d’utilisation



Description des cas d’utilisation

**ID :** Case 1

**Titre :** Commencer une partie

**Brève description :** Des utilisateurs créent une partie de Trivial Diamond.

**Acteurs :** Les joueurs

**Préconditions :**

* Il y a minimum 2 joueurs présents
* Il y a maximum 4 joueurs présents
* Les joueurs doivent savoir jouer au jeu

**Post conditions :**

* Les joueurs jouent au jeu

**Enchaînement des opérations**

|  |  |
| --- | --- |
| **Acteurs** | **Système** |
| 1. Le cas d’utilisation débute lorsqu’un des joueurs démarrent le programme. | 1. Le système demande un nombre de joueurs valide (2 à 4). |
| 1. Les joueurs entrent leur nom de joueur. | 1. Le système crée ou cherche l’identité des joueurs dans la BD. |
|  | 1. Le système détermine l’ordre des joueurs. |
|  | 1. Le système démarre la partie. |

**Scénarios alternatifs :**

**A1 :** Un joueur décide de quitter ou d’abandonner pendant la création de partie.

* Une nouvelle partie doit être créée

**A2 :** Il n’y a pas assez de joueurs ou trop de joueurs.

* Les utilisateurs doivent par eux même décider quoi faire

**ID :** Case 2

**Titre :** Tourner la roulette

**Brève description :** Une catégorie aléatoire est choisie par le programme. Si la catégorie choisie est blanche l’utilisateur peut choisir sa catégorie. Ensuite le programme pose une question aléatoire au joueur.

**Acteurs :** Roulette

**Préconditions :**

* Déterminer le nombre de joueurs.
* Déterminer l’ordre des joueurs.

**Post conditions :**

* La question ainsi que sa réponse sont gardées en mémoire.

**Enchaînement des opérations**

|  |  |
| --- | --- |
| **Acteurs** | **Système** |
| 1. Le cas débute lorsque c’est au tour du joueur à jouer. |  |
| 1. La roulette se met à tourner |  |
|  | 1. Le programme calcule une valeur aléatoire |
| 1. La roulette s’arrête sur une position aléatoire déterminer par le système |  |

**Scénarios alternatifs :**

**A1 :** La roulette tombe sur la couleur blanche (Système)

- Le joueur choisit sa catégorie. (Acteur)

**ID :** Case 3

**Titre :** Vérification de fin de partie

**Brève description :** La partie sera terminée lorsqu'un joueur aura rempli les conditions pour gagner.

**Acteurs :** Un joueur

**Préconditions :**

* Une partie doit avoir été débutée
* Une réponse doit avoir été répondue correctement

**Post conditions :**

* Les joueurs arrêtent de jouer
* Les joueurs continuent de jouer

**Enchaînement des opérations**

|  |  |
| --- | --- |
| **Acteurs** | **Système** |
| 1. Un joueur répond à une question correctement. | 1. Le système met la question répondu dans la table Réussi avec l’ID du joueur et de la question. |
|  | 1. Le système compte le nombre de questions réussies. |

**Scénarios alternatifs :**

**A1 :** Il y a 30 questions réussites dans la table Réussi pour un même joueur.

* La partie se termine

**A2 :** Il n’y a pas 30 questions réussites dans la table Réussi pour un même joueur.

* La partie continue

**ID :** Case 4

**Titre :** Répondre à une question

**Brève description :** Un joueur répond à une question du quizz.

**Acteurs :** Le joueur

**Préconditions :**

* La roulette doit être tournée.
* Les joueurs doivent être encore dans la partie.
* Une catégorie doit être choisie.

**Post conditions :**

* Si bonne réponse on vérifie s’il y a un gagnant.

**Enchaînement des opérations**

|  |  |
| --- | --- |
| **Acteurs** | **Système** |
|  | 1. Le système affiche la question et les choix de réponses. |
| 1. Le joueur choisit une réponse. | 1. Le système traite la réponse. |
|  | 1. Le système affiche le résultat (Bonne ou mauvaise réponse) |

**Scénarios alternatifs :**

**A1 :** Le joueur ne choisit pas de réponse

- La réponse est comptée comme étant mauvaise.

- On passe au prochain joueur.

**A2 :** Le joueur a la bonne réponse

- C’est encore son tour, donc un nouveau tour sera joué.

Diagramme de classes UML

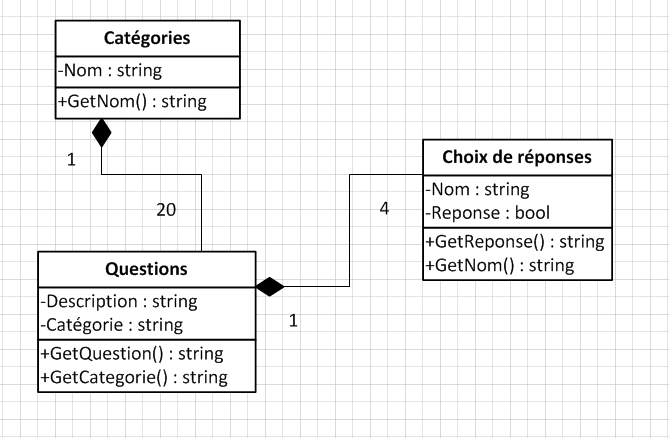
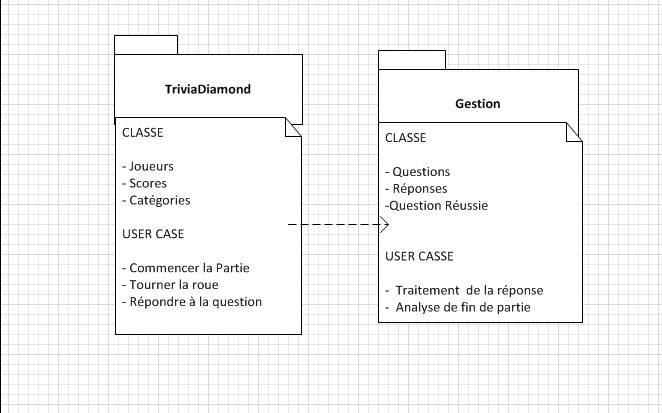
******

Diagramme de packages

**

**Conclusion**

Évaluation de la faisabilité

Le jeu Trivial Diamond est un projet réalisable pour le temps alloué puisqu’il englobe des connaissances informatiques pour lesquelles l’équipe de production (Melissa et Charlie) a les compétences requises. De plus, puisqu’il existe déjà des logiciels similaires sur le marché nous avons accès à certaines informations pour nous inspirer et nous guider dans notre démarche. (Ex. : Trivia Crack ou Trivial Pursuit)

Dans le même courant de pensée, l’équipe a accès à des outils de programmation à coût presque inexistant. Cet aspect du travail couvert, il est encore plus facile de se concentrer sur la conception du jeu questionnaire.

D’autre part, étant donné que la planification n’est pas comptée dans le temps de conception, il ne reste qu’à calculer le temps prévu pour les différentes étapes de création ce qui simplifie l’ampleur de la tâche globale.

Cependant, il faut prévoir les problèmes que pourrait rencontrer l’équipe en lien avec la connexion réseau nécessaire pour avoir accès à la base de données et à GitHub, une application de service web d’hébergement et de gestion de développement de logiciels qui nous permettra de travailler en équipe sans difficultés.

Droits d’auteur et de reproduction

Tous droits réservés. Aucun extrait de ce document ne peut être reproduit, sous quelque forme ou procédé quel qu’il soit.

Solution d’implémentation 1

Description de la solution

Pour notre première solution, nous commencerons par implémenter le jeu afin qu’il fonctionne pour un seul joueur. Cette solution nous permettra de faciliter la construction du jeu tout en permettant l’ajout de la fonctionnalité multijoueur qui sera ajouté par la suite. Puisque la plupart des options du jeu peuvent être appliquées qu’à un seul utilisateur, cette solution nous permettra entre autre d’économiser du temps et d’avoir quelque chose à remettre au client en cas de problème avec l’intégration du multijoueur. Le projet devrait être terminé aux alentours du 15 mars.

Échéancier d’implémentation

|  |  |
| --- | --- |
| **Description des étapes** | **Temps requis** |
| BD | 6 heures |
| Création des tables | 1 heure |
| Insertion des données | 2 heures |
| Création des packages, procédures et fonctions | 3 heures |
| Ado.net et C# | 20 heures |
| Fonctionnement du logiciel | 2 heures |
| Intégration des commandes sql | 18 heures |
| Réglage final des bugs | 2 heures |

Solution d’implémentation 2

Description de la solution

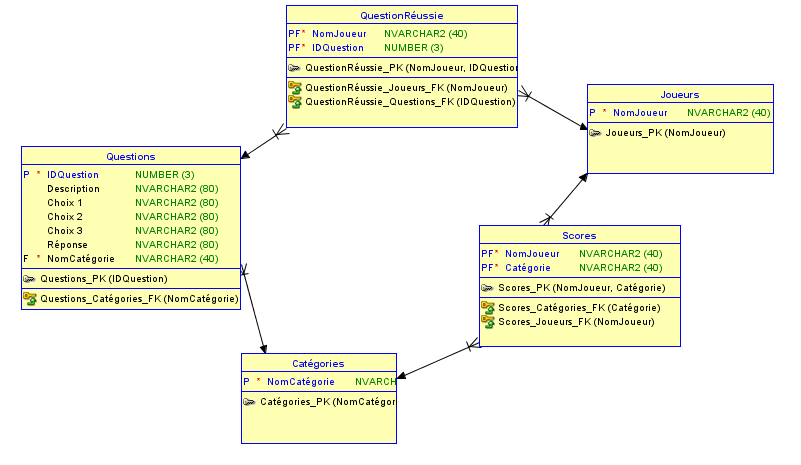
La deuxième solution est la suite de la première. Elle englobe l’ajout de la fonctionnalité multijoueur dans le jeu et le peaufinage de l’interface graphique. Cette dernière est considérée dans la planification normale comme étant de base, mais peut être abandonnée en cas de manque de temps dût à des imprévus. Le projet devrait être terminé aux alentours du 17 mars.

Échéancier d’implémentation

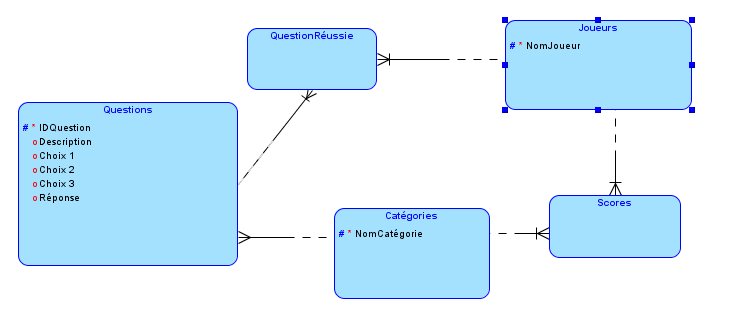
|  |  |
| --- | --- |
| **Description des étapes** | **Temps requis** |
| BD | 6 heures |
| Création des tables | 1 heure |
| Insertion des données | 2 heures |
| Création des packages, procédures et fonctions | 3 heures |
| Interface | 4 heures |
| Développement | 4 heures |
| Ado.net et C# | 20 heures |
| Fonctionnement du logiciel | 2 heures |
| Intégration du multijoueur | 5 heures |
| Intégration des commandes sql | 18 heures |
| Règlage final des bugs | 2 heures |

**Annexes**

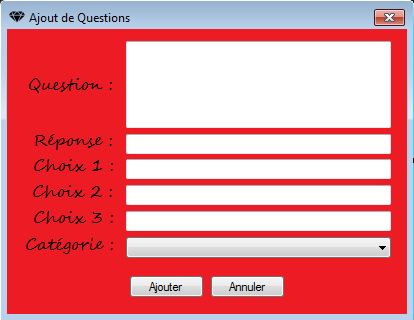
Annexe 1 – Modèle Relationnel

******

Annexe 2 – Modèle Conceptuel

******

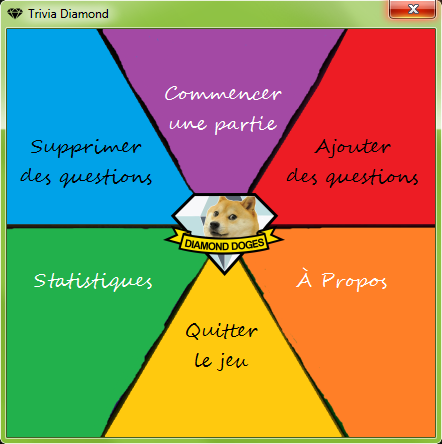
Annexe 3 – Prototype de l’application

******

Windows Form pour ajouter une question dans la base de données selon une catégorie choisie.



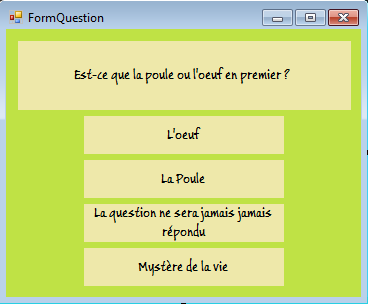
Windows Form pour afficher les informations générales de l’application. Communément appelé À Propos.



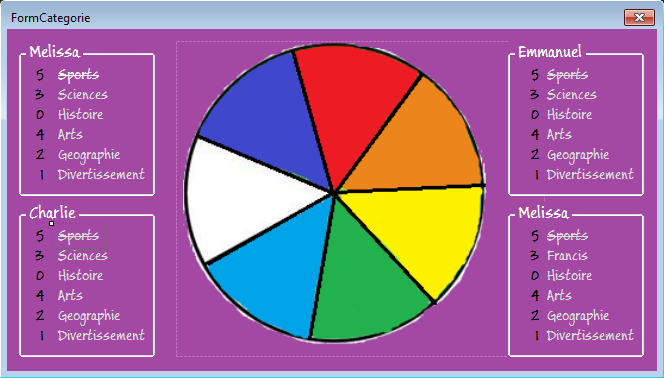
Windows Form du menu de l’application qui va être composé de 6 boutons «Flash» qui permettent d’accéder à chaque partie de l’application.

******

Windows Form pour supprimer une question dans la base de données selon une catégorie choisie.

******

Windows Form qui affiche la question à répondre à l’utilisateur. Les choix de réponses sont des boutons «Flash»

******

Windows Form du jeu avec affichage du nombre de questions répondues et des catégories terminées.