

16/12/2016

Rapport du projet de Génie Logiciel

Tests pour la sélection des astronautes de l'ESA



Laura Lille et Arthur Lemaire de Mil
BORDEAUX INP - ENSC

Table des matières

Introduction.....	2
I - Spécifications générales.....	2
➤ Analyse fonctionnelle	2
➤ Analyse des besoins pour l'interface avec l'utilisateur	3
➤ Modélisation de l'application (schéma UML et architecture).....	4
➤ Répartition des tâches dans le groupe et planning.....	6
II - Spécifications détaillées	8
➤ Formulaire composants l'application	8
➤ Choix de conception et de production du code source	15
III - Résultats et tests	18
➤ Résultats du programme	18
➤ Tests unitaires	18
➤ Tests fonctionnels.....	18
IV - Bilan et perspectives	20
➤ Points positifs et négatifs	20
➤ Evolutions possibles	20
Annexe.....	21
➤ Maquettes réalisées au début du projet.....	21
➤ Spécifications fonctionnelles détaillées	24
➤ Tableau des exigences.....	32

Introduction

Le but de ce projet était de réaliser un programme de tests similaire à celui qui est utilisé par l'agence spatiale européenne pour la sélection des astronautes.

Dans le rapport suivant, nous utiliserons certaines abréviations :

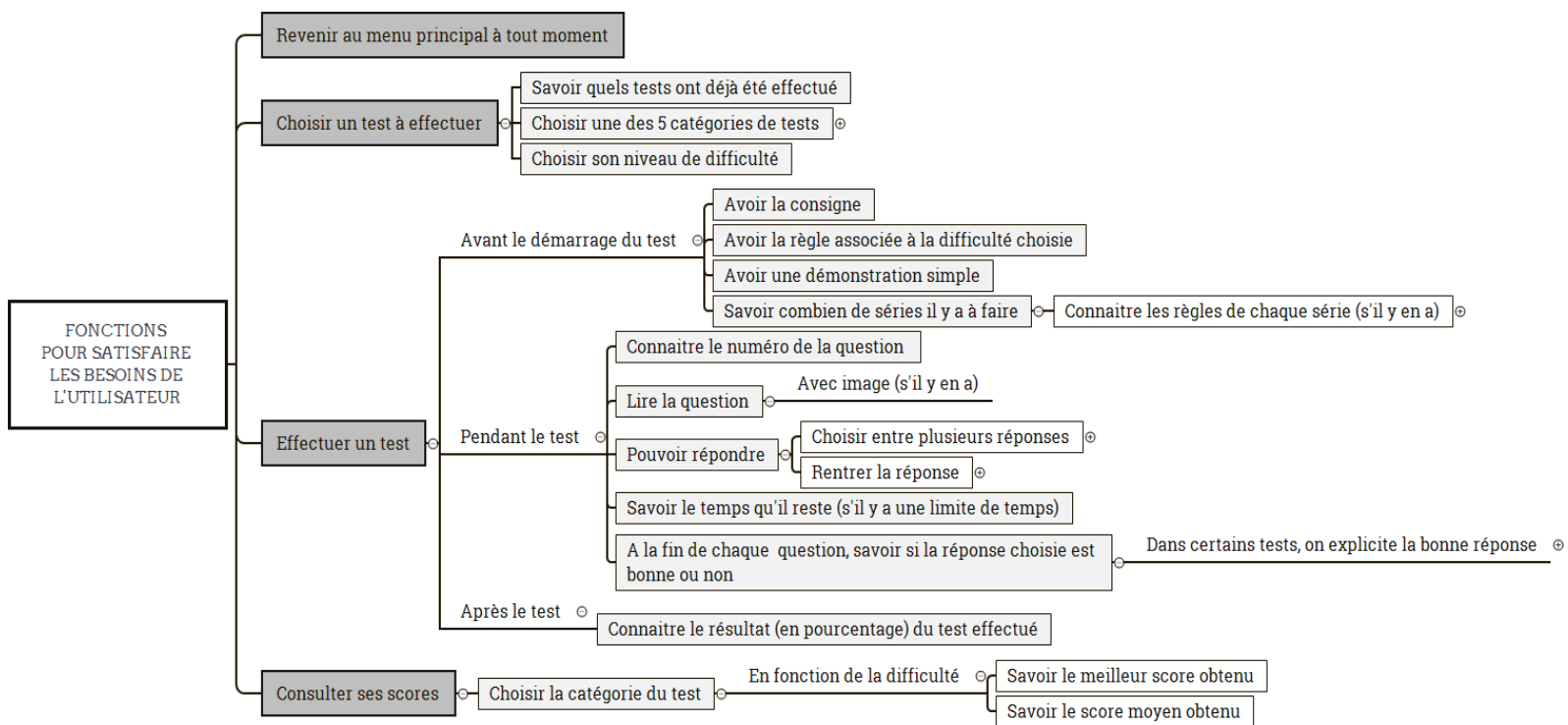
- le test « Perception et mémoire associative » sera appelé test 1
- le test « Attention et concentration » sera appelé test 2
- le test « Calcul mental » sera appelé test 3
- le test « Problèmes mathématiques » sera appelé test 4
- le test « Problèmes physique » sera appelé test 5

I - Spécifications générales

Avant de commencer, abordons la notion d'utilisateur. Chaque fois que l'application sera lancée, elle correspondra à un nouvel utilisateur. Il n'y a pas de nom associé, mais à chaque lancement, tout est réinitialisé.

➤ Analyse fonctionnelle

Ci-dessous une map (réalisée sur Xmind) contenant les fonctions offertes par l'application pour satisfaire les besoins de l'utilisateur.



Quelques précisions sur les tests associés à certaines fonctions (nous ne l'avons pas représenté sur la map pour ne pas surcharger encore plus) :

Fonction "Choisir une des 5 catégories de tests" :

- Perception et mémoire associative
- Attention et concentration
- Calcul mental
- Problèmes mathématiques
- Problèmes physiques

Fonction "Connaitre les règles de chaque série (s'il y en a)" :

- Attention et concentration

Fonction "Choisir entre plusieurs réponses" :

- Attention et concentration
- Problèmes mathématiques
- Problèmes physiques

Fonction "Rentrer la réponse":

- Perception et mémoire associative
- Calcul mental

➤ Analyse des besoins pour l'interface avec l'utilisateur

Au début du projet, nous avons imaginé des maquettes. Nous n'avons pas pensé à l'aspect design, mais plutôt fonctionnel pour l'utilisateur. Ces maquettes sont disponibles en Annexe.

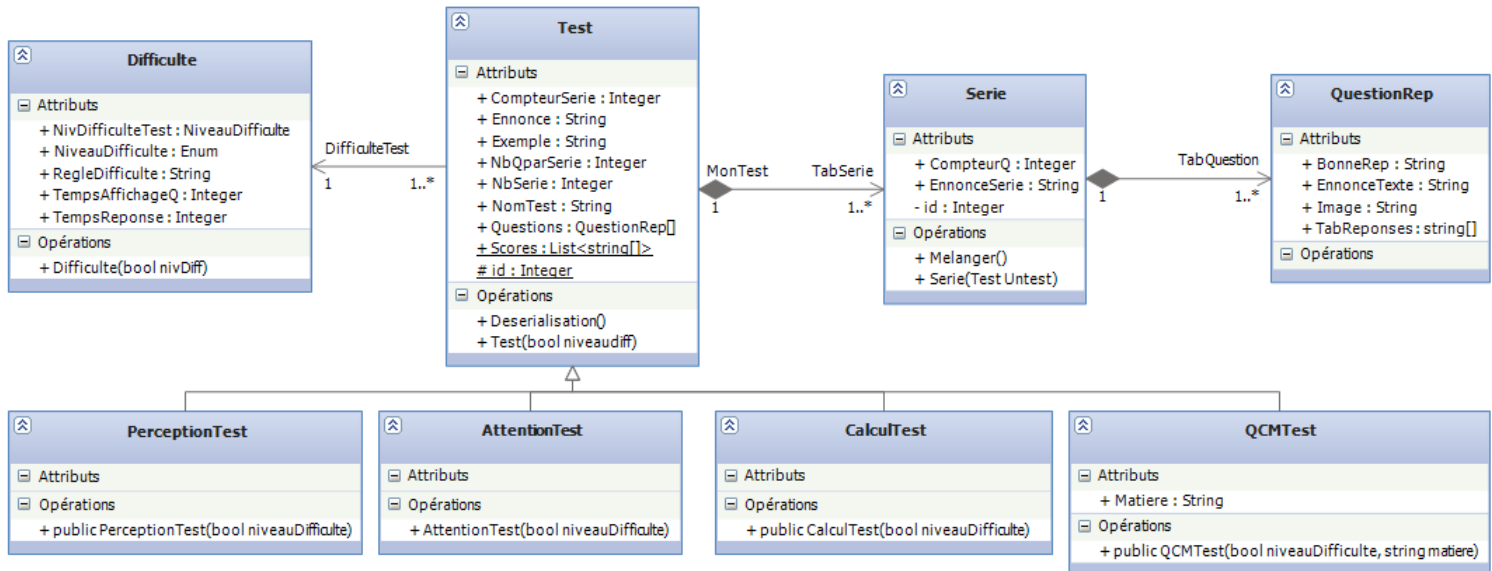
Vers la fin du projet, nous avons choisi la charte graphique et avons utilisé les critères suivants :

Critères	Applications
Facilité d'utilisation	-Les possibilités de navigation sont restreintes : 2 boutons principaux (menu principal et OK) - choix : boutons radio seulement -Structure simple de l'application : menu -> énoncé du test -> test (question/réponse)
Lisibilité	-Mise en avant des données pertinentes/prioritaires en bleu (bouton pour passer) -Regroupement des données similaires avec les groupes box - Sens/ordre de lecture occidentale - Stabilité d'un écran à l'autre Par ex : Boutons pour changer de pages au même endroit (menu principal en haut à droite, bouton OK en bas à droite) -Fond noir avec écriture claire
Standardisation	-Tous les éléments fixes ont la même taille : titre, bouton
Ergonomie	-La taille des boutons est assez grande (minimum 40*40) -La police est de taille minimum 10

La difficulté a été d'adapter un même design à différents tests non homogènes. Par exemple, sur certains tests, pour accéder à la question suivante il fallait cliquer sur OK ; dans d'autres, il fallait attendre quelques secondes.

➤ Modélisation de l'application (schéma UML et architecture)

Voici le diagramme de classes de notre application (présent dans le code source également), les méthodes présentes dans chaque classe sont détaillées dans un tableau plus bas :



Quelques précisions sur les méthodes des classes (les constructeurs ne sont pas détaillés) :

Class : Test

Appel de la méthode	Argument(s)	Explications
<code>public void Deserialisation (string filePath)</code>	filePath est le chemin du fichier ou sont stocké les fichiers XML	La méthode Deserialisation n'est utilisé que par les tests mathématiques et physiques permet de désérialiser les questions des QCM.

Class : Serie

Appel de la méthode	Argument(s)	Explications
<code>public Serie (Test Untest)</code>	filePath est le chemin du fichier ou sont stocké les fichiers XML	La méthode Deserialisation n'est utilisé que par les tests mathématiques et physiques permet de désérialiser les questions des QCM.
<code>public static void Melanger (QuestionRep[] tab)</code>	tab est un tableau de questions réponses	La méthode mélanger permet de mélanger un tableau de questions.

Justification des choix :

- **3 grandes classes : Test/Serie/QuestionRep**

La classe Test contient toutes les informations sur le test.

La classe Serie permet de réaliser plusieurs séries de questions différentes en changeant les règles facilement (cas du test 2), elle contient notamment une liste de questions.

Remarque : cette liste de questions n'est pas à confondre avec la liste de questions contenue dans Test, qui elle contient l'ensemble des questions possibles (désérialisées). La liste de questions dans Serie contient par exemple seulement 10 questions pour le test 1. Nous avons fixé que tous les tests réalisent une série de questions sauf le test 2 qui en réalise trois. Après, pour le nombre des questions par série, tous les tests ont dix questions sauf le test 3 qui en a cinq.

La classe QuestionRep sert à coder des fichiers XML (cas des tests 4 et 5). Nous ne l'avons pas lié directement à la classe QCMTTest (qui modélise les tests 4 et 5, car la seule chose qui change c'est l'importation d'une liste de questions différentes) dans la perspective d'une évolution future.

- **Classe Difficulte**

Nous n'étions pas obligés de créer une classe pour la difficulté, mais dans la perspective d'une évolution future, nous avons jugé utile de la penser comme ceci. Ainsi, nous avons pour chaque test les données de temps et les règles.

Nous avons choisi une énumération également si plus tard nous voudrions rajouter une difficulté moyenne par exemple.

En réalité, nous avons pensé dix tests séparément (avec facile et difficile).

- **Stockage des réponses dans la classe QuestionRep**

Cette classe contient en plus de l'énoncé de la question un tableau de réponses. Elle contient également la bonne réponse sous forme de chaîne de caractères pour pouvoir comparer. Tout centralisé dans une classe a été très utile pour l'utilisation dans les winforms.

- **Enregistrement des scores**

Les scores de tous les tests sont contenus dans une liste *static* dans la classe Test. Cette liste contient des éléments de type *string[]*, contenant chacun :

Nom du test	Difficulté	Score obtenu
-------------	------------	--------------

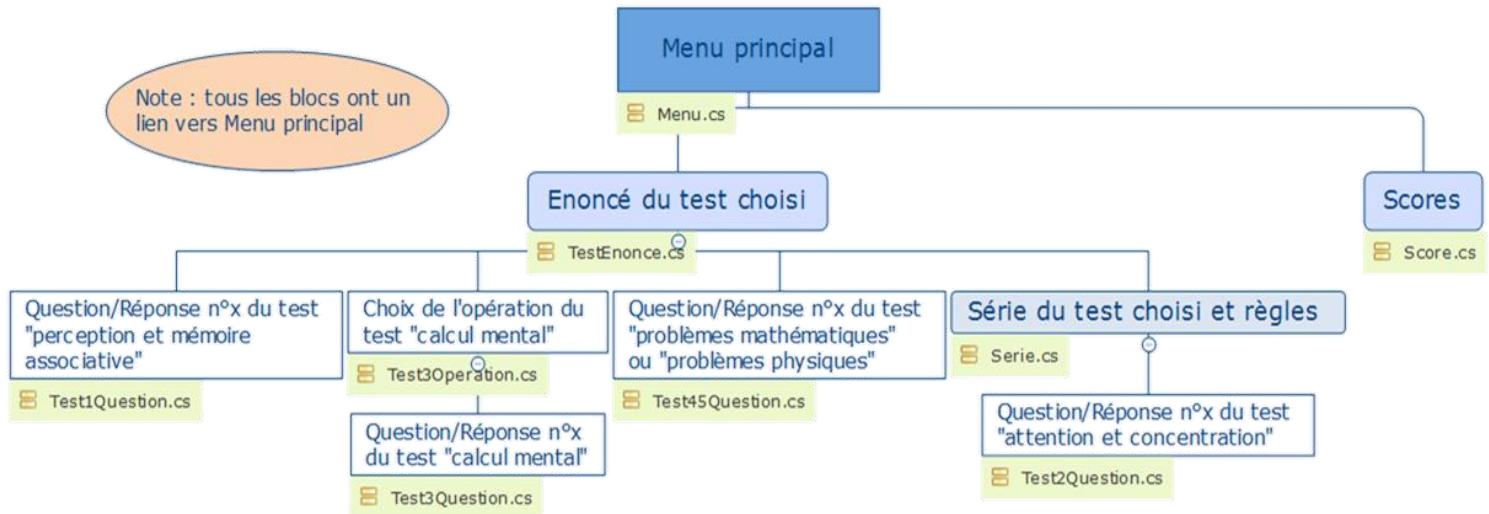
Cette liste prend les résultats dans l'ordre où les tests sont effectués.

Par exemple :

"Attention et concentration"	"Perception et mémoire associative"	"Perception et mémoire associative"	"Calcul mental"	"Perception et mémoire associative"
"Facile"	"Facile"	"Difficile"	"Difficile"	"Difficile"
"100"	"90"	"50"	"70"	"70"

L'utilisateur a effectué le test de perception et mémoire associative en niveau difficile en troisième.

Après cette phase de modélisation, nous avons construit l'architecture de notre application, la voici :



La page d'énoncé du test est commune à chaque test, mais nous avons dû réaliser des pages spécifiques en fonction des tests.

➤ Répartition des tâches dans le groupe et planning

Au début du projet, il y a une phase de réflexion (jusqu'au 21 novembre).

Nous avons décortiqué ensemble le sujet pour en sortir les fonctionnalités, et partager notre vision de l'application : nous avons dessiné les maquettes principales sur papier et esquisser l'architecture de l'application (classes).

Puis Arthur a pensé toutes les maquettes sous Publisher avec plus de détails. Laura a créé l'architecture de l'application sur Visual Studio avec un schéma UML plus complet (attributs, quelques méthodes) et créer les classes.

Un planning a été également mis au point, il a été complété au fur et à mesure du projet.

Puis à partir du 21 novembre, première séance de TD, nous avons commencé à créer l'application. Nous avons choisi de créer tout d'abord tous les winforms, avant d'implémenter la structure derrière. Ainsi, nous nous sommes séparés les pages à créer et nous les avons construites.

Du 21 novembre au 13 décembre, nous avons codé l'"intérieur" de l'application.

Nous avons commencé par les tests "Problèmes mathématiques" et "Problèmes physiques". Arthur s'est occupé de la sérialisation et de créer les XML contenant les problèmes mathématiques et quelques problèmes physiques. Laura a codé les liens entre les différentes pages et le code des tests problèmes mathématiques et physiques. Puis, Arthur s'est occupé du calcul mental et Laura a fini les XML.

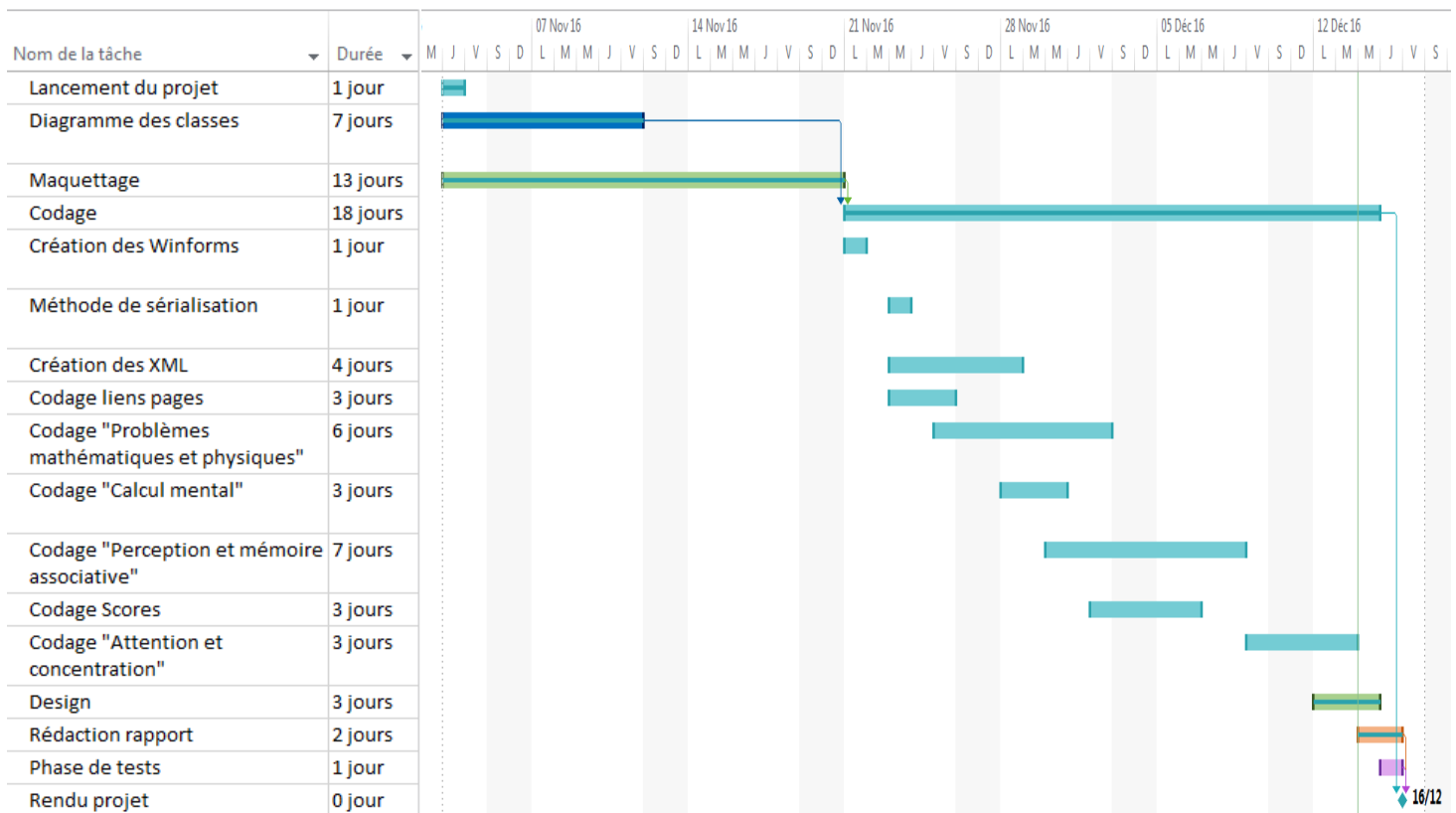
Ensuite, Arthur s'est occupé du test de "Perception et mémoire associative, pendant que Laura codait la récupération de tous les scores.

Enfin, Arthur s'est occupé du test "Attention et concentration" que Laura a complété, et Laura s'est occupé de l'aspect "design" des winforms.

Enfin, nous avons rédigé ensemble le rapport.

Ainsi, durant toute la durée de ce projet, nous n'avons pas effectué les mêmes tâches, travaillant des fois chacun de notre côté. Mais en échangeant souvent, nous nous sommes mutuellement aidé et avons abordé chacun finalement tous les aspects de ce projet.

Voici le planning final (réalisé sur Microsoft Project):



II - Spécifications détaillées

➤ Formulaire composants l'application

Voici le détail de chaque formulaire et de ce qu'ils effectuent.

❖ Menu.cs

Lorsqu'on ouvre un MenuForm, on coche tous les tests déjà réalisés (en vérifiant dans [Test.Scores](#) quels tests ont déjà un score).

On peut ensuite accéder aux scores ou lancer un test.

Menu principal

Test ESA - Menu principal

Bienvenue sur l'application de test ESA !

Choisir une catégorie de test

	Facile	Difficile
<input checked="" type="radio"/> Perception et mémoire associative	<input type="checkbox"/>	<input type="checkbox"/>
<input type="radio"/> Attention et concentration	<input type="checkbox"/>	<input type="checkbox"/>
<input type="radio"/> Calcul mental	<input type="checkbox"/>	<input type="checkbox"/>
<input type="radio"/> Problèmes mathématiques	<input type="checkbox"/>	<input type="checkbox"/>
<input type="radio"/> Problèmes physiques	<input type="checkbox"/>	<input type="checkbox"/>

Choisir une difficulté

☒ Facile ☐ Difficile

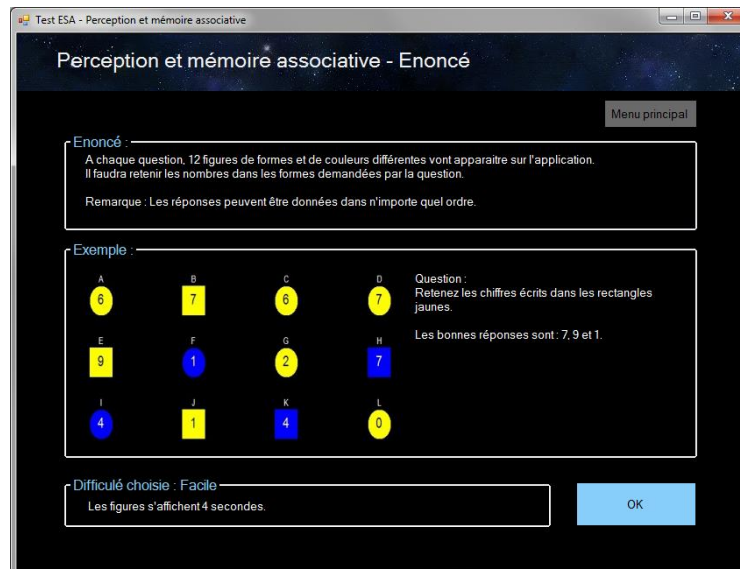
Scores Lancer le test

❖ TestEnonce.cs

Lors de l'ouverture d'un TestEnonceForm on récupère l'énoncé, l'exemple et on charge l'image relative au test en cours et on les affiche sur le form.

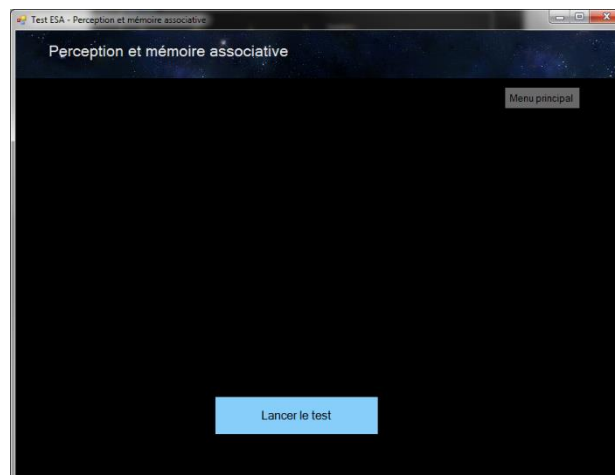
On peut ensuite lancer le test. Dans le cas des tests 1, 3, 4 et 5 on tombe directement sur la page du test. Dans le cas du test 2, on renvoie sur une page SerieForm.

Voici le screenshot correspondant au test 1, mais tous les tests ont la même disposition de page pour l'énoncé.



❖ Test1Question.cs

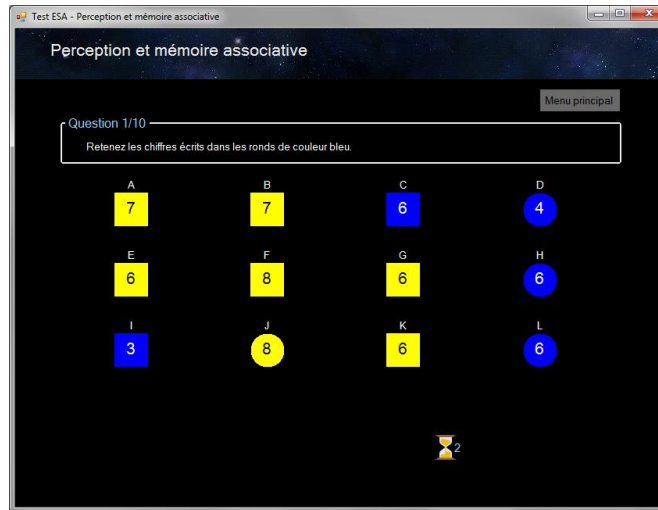
L'affichage des figures du test 1 est limité dans le temps (quel que soit la difficulté), c'est pourquoi il est nécessaire de ne pas lancer le test dès l'ouverture du form, d'où l'ajout d'un bouton « Lancer le test ».



C'est dans le constructeur du form, qu'on définit la règle pour la première question (c'est dire la forme et la couleur des figures dont il faudra retenir la valeur). La règle pour les questions suivantes sont ensuite générées lors que click sur le bouton OKBtn.

Une fois le test lancé, on génère la liste des figures à tracer. Pour cela, on génère une liste aléatoire de figures, si elle ne contient pas 3 ou 4 figures correspondant à la règle de la question, on en génère une nouvelle et cela jusqu'à obtenir une liste satisfaisante.

On trace ensuite les figures à l'écran grâce à la méthode draw. Cette méthode utilise l'espace de nom System.Drawing et les méthodes du type DrawRectangle afin de tracer les 12 figures.

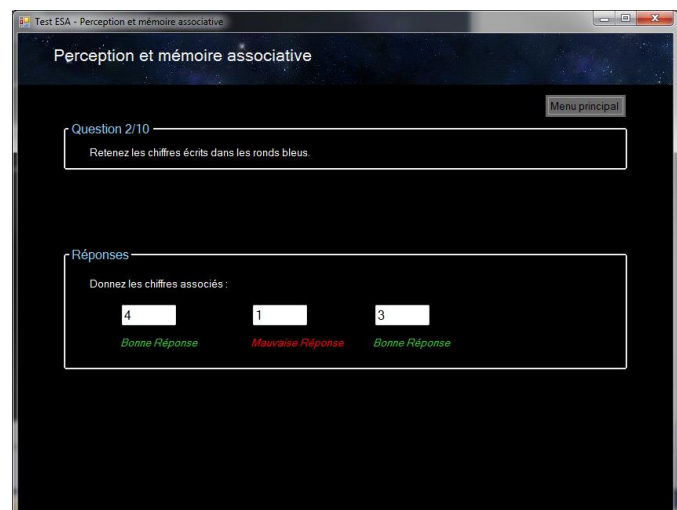
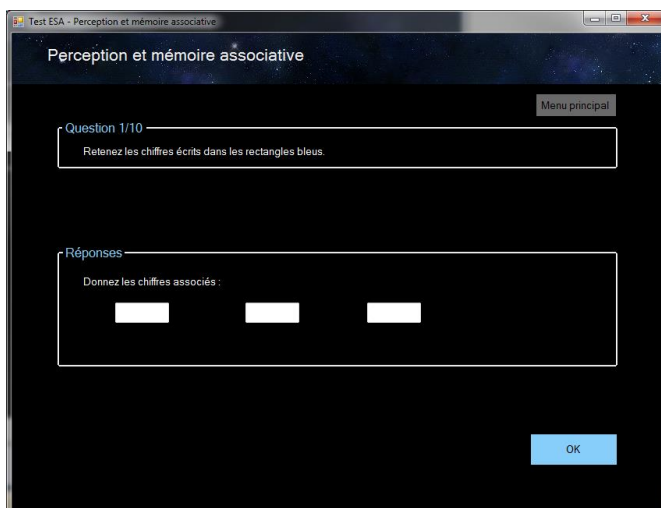


Une fois le temps d'affichage écoulé, on efface les figures grâce à la méthode Refresh et on affiche le formulaire de réponse. Lorsque l'utilisateur a entré ces valeurs et cliqué sur Ok, on enregistre ces valeurs dans une liste et on appelle la méthode Verif_rep.

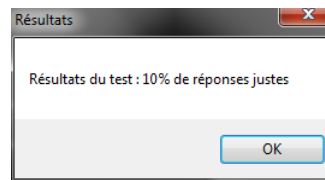
Remarque : on empêche les utilisateurs de rentrer autre chose que des entiers en réponse grâce à un système de try/catch. En effet dès que l'utilisateur modifie le contenu de la boîte réponse, si celui-ci n'est pas un entier, il est automatiquement effacé.

Cette méthode permet de vérifier, grâce à un tableau de booléens, que chaque réponse rentrée par l'utilisateur appartient à la liste des bonnes réponses. Ce qui permet à l'utilisateur de rentrer les réponses dans l'ordre qu'il souhaite.

Une fois les réponses validées et l'affichage du « bonne/mauvaise » réponse, on génère la question suivante.



A la fin de chaque test, il s'affiche une message box contenant le score en pourcentage :



❖ Serie.cs

Les fenêtres séries ne sont utiles que pour le test 2, et elle sert à commencer une série de questions. On va donc, lors du chargement de la page, établir la liste des règles (grâce à la méthode `GenereRegle`) appliqué lors de la réalisation de la série. On peut ensuite lancer le test 2.

La méthode `GenereRegle` ne va créer un tableau de règles que lorsqu'on on réalise la première série du test 2 facile ou à chaque série du test 2 difficile dans les autres cas le tableau de règle sera rempli par le constructeur de la série.

Lorsqu'on lance le test, on passe en paramètre le tableau de règles ainsi que, dans le cas où ce n'est pas la première série, le score du test en cours.



Emplacement du form :

Dans l'architecture initialement prévu, nous avons envisagé de placer le form `Serie.cs` entre l'énoncé et le début de chaque test. Il s'est toutefois révélé que c'était inutile pour la plupart des tests. C'est pourquoi nous avons modifié l'architecture afin de l'utiliser seulement pour le test 2.

Il est toutefois possible, en cas d'évolution du logiciel de rajouter simplement des séries aux tests actuels.

❖ Test2Question.cs

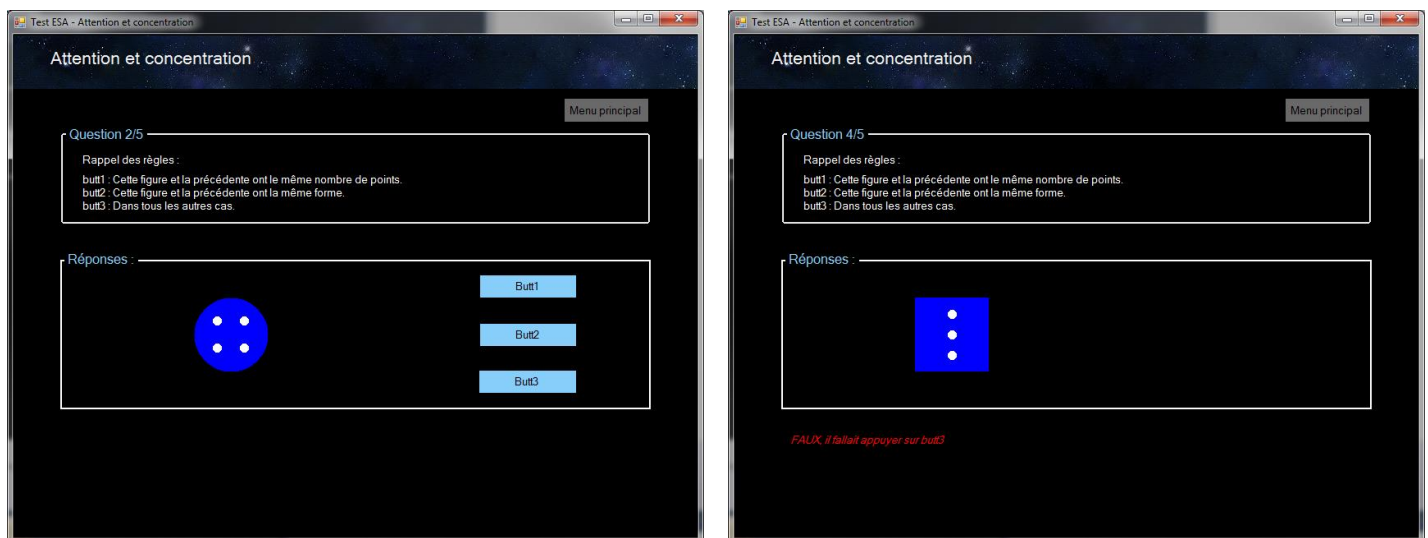
Lorsqu'on ouvre un form pour le test 2, on génère la suite de figures qui servira pour la série en cours. Pour cela, on appelle la méthode `genereFigure` qui va générer un tableau de 5 figures.

On commence par créer de manière aléatoire la première figure, puis on crée les 4 autres une par une, en vérifiant qu'elle possède au maximum un point commun avec la précédente. Si ce n'est pas le cas, on recrée une figure jusqu'à ce qu'elle corresponde au besoin.

Une fois la liste terminée, on crée un tableau de similitudes entre les figures consécutives. On compare chaque case de ce tableau aux règles établies avec un autre tableau, contenant les boutons sur lesquels il faudrait appuyer pour avoir tout bon à cette série. On utilise ensuite ce tableau pour vérifier que cette série de figures nécessite d'appuyer au moins une fois sur chacun des boutons. On recommence ce procédé tant que la liste des figures n'est pas conforme.

Lorsque la suite de figures correspond aux critères, on les trace à l'écran avec les mêmes méthodes que le test 1. Pour tracer les figures, on lance lors du chargement de la page un timer très court (`AfficheOnLoadTmr`) qui permet d'appeler la méthode `Draw` (car on ne peut pas dessiner dans la méthode `Test2QuestionForm_Load`).

Une fois la série finie, on renvoie vers un `SerieForm` à qui on transmet le score actuel. Dans le cas d'un test facile, on transmet aussi les règles.



Génération des figures :

Il était initialement prévu de créer une classe `Figures` dans notre architecture. Nous avons plus tard décidé de gérer les figures en interne pour les tests 1 et 2. En effet, chaque figure est représentée par un tableau de 3 entiers :

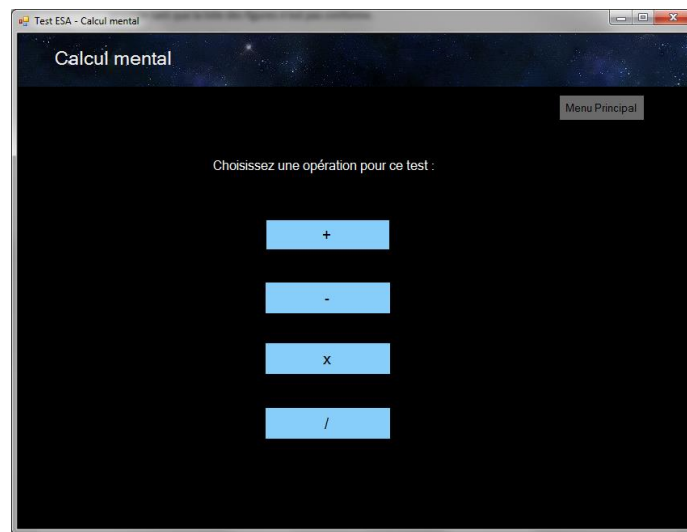
Forme	Couleur	Valeur inscrite dans la figure / Nombre de point
-------	---------	---

Cette représentation permet une gestion simple des figures à l'intérieur de chaque classe et évite la gestion d'une classe supplémentaire.

Note : le tableau des règles du test 1 est géré de la même manière : on crée un tableau de deux cases dans lequel on stock la forme et la couleur voulu.

❖ Test3Operation.cs

Le form permet de choisir quel opération on souhaite réaliser dans le test 3. Lors du clic sur un bouton, on ouvre un Test3QuestionForm en lui passant en paramètre l'opération choisie.

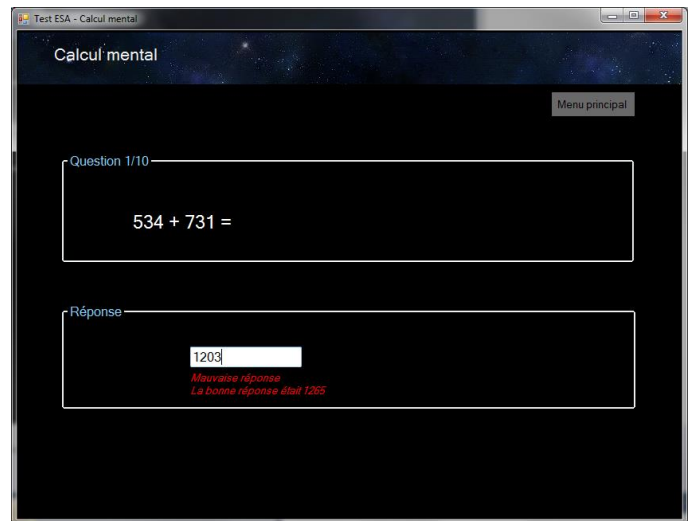
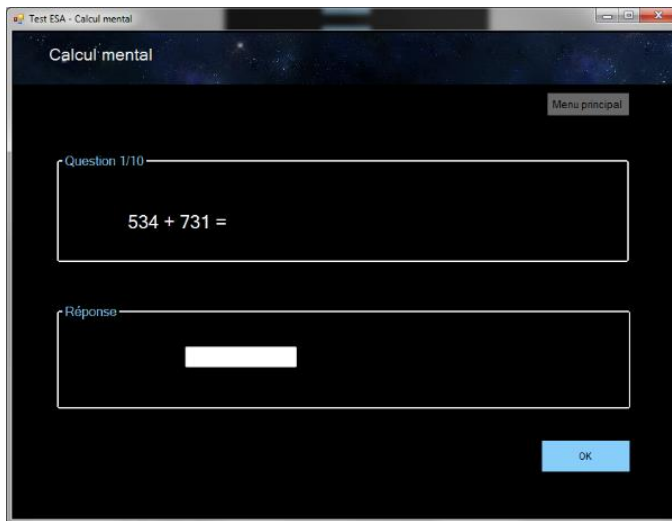


❖ Test3Question.cs

L'ouverture d'un test 3 dans le cas facile entraine la génération du calcul. Dans le cas difficile, il faut d'abord cliquer sur le bouton lancer le test, comme dans le test 1.

Cette génération passe par l'utilisation de la méthode : Generation_Calcul, qui en fonction de l'opération choisie et des règles imposées par le sujet, génère aléatoirement deux entiers auquel il faut appliquer l'opération.

On vérifie ensuite la réponse de l'utilisateur, on affiche s'il a eu bon ou pas, puis quelques secondes après, on bascule directement à la question suivante.

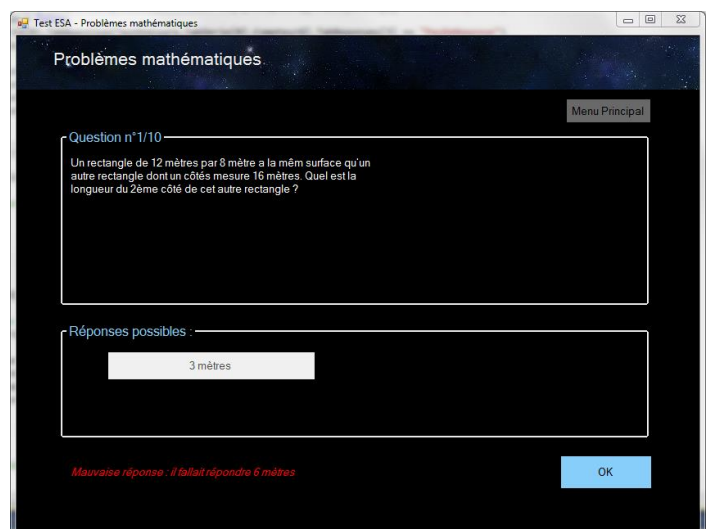
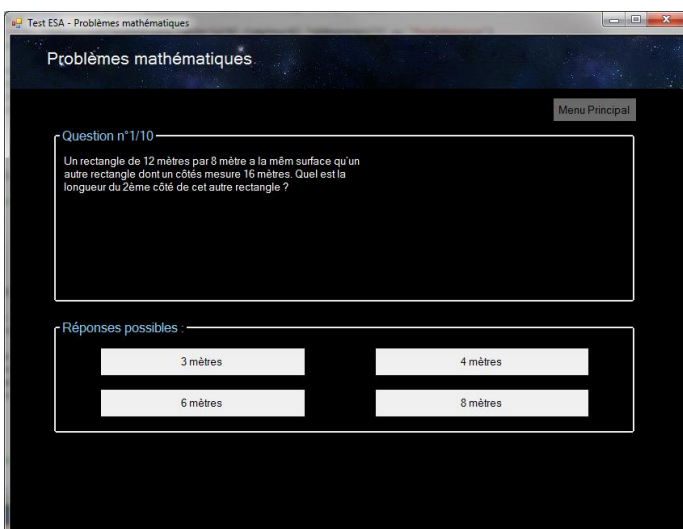


❖ Test45Question.cs

Lorsqu'on lance un test 4 ou 5, on récupère la première question qui a été désérialisée lors de la création du test `TestEnCours.TabSerie[0].TabQuestion[TestEnCours.TabSerie[0].CompteurQ].EnnonceTexte`, ainsi que les réponses associées.

Lorsque l'utilisateur clique sur une réponse, on compare sa réponse à un tableau de 4 booléens. Ce tableau contient dans chaque case le booléen : `Bouton1==bonne réponse`.

L'utilisateur doit cliquer sur OK pour passer à la question suivante.



Timer :

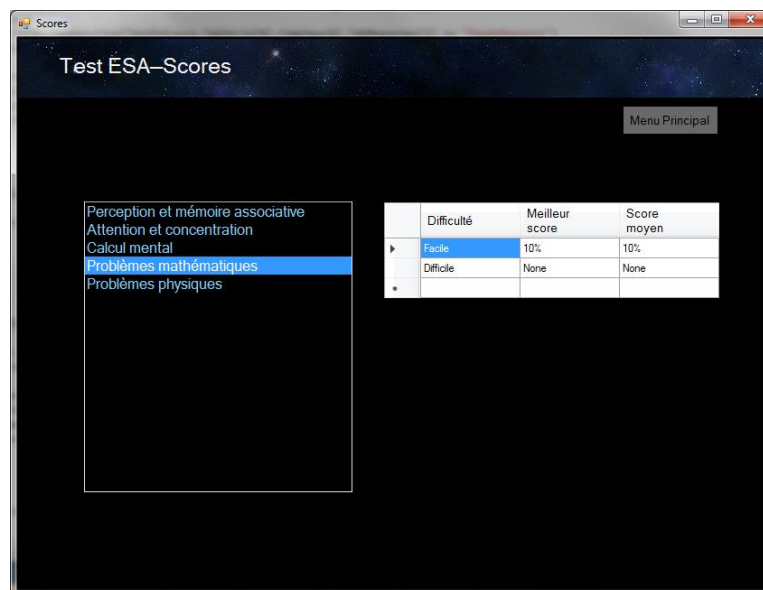
Afin de réaliser l’affichage des réponses, ou même les temps de réponse aux questions, on utilise les timer de visual studio. C’est timer permettent de gérer les événements chronométrés et donc l’enchaînement des actions.

Fin du test :

A la fin de chaque test décrit précédemment, on enregistre le score (en pourcentage) dans la classe Test et on renvoie l’utilisateur sur un MenuForm.

❖ **Score.cs**

Dans le form score, on a à droite la liste des tests. Lorsqu’on clique sur une liste, on va chercher dans Test.Scores s’il existe un/des score(s) correspondant(s) au test voulu. Lorsque c’est le cas, on calcule la moyenne de l’utilisateur sur ce test dans les deux difficultés.



➤ Choix de conception et de production du code source

• **Principes de conception**

Les principes de conception que nous avons vraiment réalisé sont mis en bleu, les autres sont des essais.

- **Séparation des responsabilités**

Nous avons tenté de séparer le plus possible l’aspect affichage de l’aspect traitement/calcul. Par exemple, nous avons désérialisé les questions pour les tests 4 et 5 dans la classe Test, et non directement dans le code du Winform.

- Réutilisation

Nous avons seulement réutilisé la méthode de désérialisation vu en cours. Nous nous sommes aussi inspirés de certaines réponses sur les forums pour comprendre comment dessiner les figures.

- Encapsulation maximale

Nous avons essayé de penser à cela en mettant des propriétés à la place des attributs, ou en les mettant en private, mais nous n'avons pas ajouté de contraintes sur le set.

- Couplage faible

Vu la taille du projet et de notre architecture, la question ne s'est pas vraiment posée.

- Cohésion forte

Nous avons pensé notre modélisation pour avoir des classes avec des responsabilités cohérentes pour éviter les classes avec des responsabilités trop variées. D'où la création des classes Difficulte, Serie et QuestionRep pour ne pas tout contenir dans la classe Test.

- DRY

Nous avons essayé de mettre dans des méthodes toutes les portions de code qui pouvaient être redondants.

- KISS

Nous n'avons pas vraiment pensé directement au plus simple, mais avons tout de même choisi des raccourcis pour éviter de compliquer la production.

- **Règles de production du code source que nous nous sommes fixés**

- Conventions de nommages

Nous avons déjà admis des conventions de nommages basiques. Pour les propriétés par exemple, nous les avons écrits avec une majuscule. Ou aussi par exemple la manière de nommer un attribut : on notait SerieEnCours et pas Serie_en_cours. Nous avons essayé de garder un nom similaire dans les différentes classes pour la cohérence du passage des variables.

Pour nommer les objets du winform, nous avons créé un tableau sur le drive que nous suivions :

Boite	Notation
form	<i>nomForm</i>
label	<i>nomLb</i>
picturebox	<i>nomImg</i>
checkbox	<i>nomCheckB</i>

button	<i>nomBtn</i>
groupbox	<i>nomGrB</i>
textbox	<i>nomTextB</i>
radioButton	<i>nomRadioBtn</i>
dataGridView	<i>nomDataGV</i>
listbox	<i>nomListB</i>

- Langue utilisée

La langue utilisée dans tout le projet était le français.

- Formatage du code

C# offre un formatage automatique du code. Nous avons veillé à avoir une lecture agréable du code avec des régions par exemple.

- Commentaire

Nous avons commenté ce qui nous semblait important, il y a peut-être un peu trop de commentaires.

- **Travail en groupe : utilisation de Git**

La réalisation du projet nécessitait un travail à deux sur le code source de l'application, c'est pourquoi il nous a été demandé de travailler avec Github. Nous nous sommes familiarisés avec l'environnement et les commandes, car nous étions tous les deux novices dans cette utilisation.

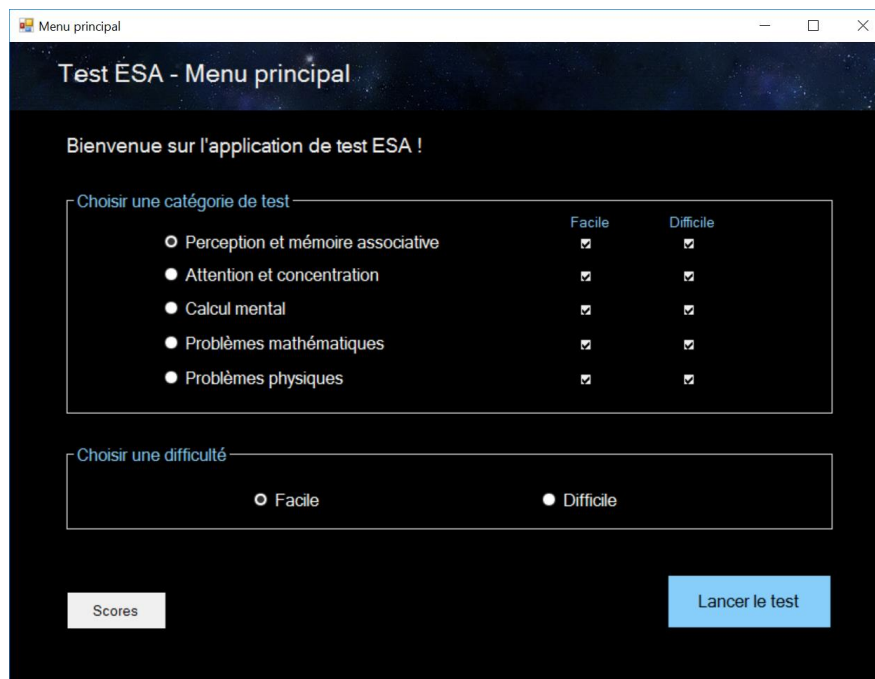
Nous avons rencontré des problèmes pour « merdger » nos fichiers. Nous avons créé des branches, cependant par exemple, nous avons réalisé de nombreux push le 12/12 en tentant de merdger nos deux fichiers, tous furent infructueux. Nous avons donc dû les mélanger à la main, ce qui ne fut toutefois pas un problème car nous avons travaillé de manière à ne pas interférer l'un avec l'autre.

III - Résultats et tests

➤ Résultats du programme

Comme nous avons mis une majorité des screenshots possibles dans la partie II, nous ne voulions pas surcharger le rapport.

Voici le screenshot du menu principal avec tous les tests effectués :



➤ Tests unitaires

Nous n'avons pas effectué de tests unitaires au cours du projet.

➤ Tests fonctionnels

Les protocoles de ces tests sont présentés en annexe dans spécifications fonctionnelles détaillées (7 pages) et le tableau des exigences (2 pages).

Voici les résultats obtenus :

Domaine fonctionnel ▼	Exigence ▼	Priorité ▼	Description ▼	Statut d'exécution ▼	Anomalie (si KO)
CU_001	CU_001_RG_001	1 - Haute		OK	
CU_001	CU_001_RG_002	1 - Haute		OK	
CU_001	CU_001_RG_003	2 - Moyenne		OK	
CU_001	CU_001_RG_004	3 - Basse		OK	
CU_001	CU_001_RG_005	2 - Moyenne		OK	
CU_001	CU_001_RG_006	1 - Haute		OK	
CU_002	CU_002_RG_001	2 - Moyenne		OK	
CU_002	CU_002_RG_002	2 - Moyenne		OK	
CU_002	CU_002_RG_003	2 - Moyenne		OK	
CU_003	CU_003_RG_001	1 - Haute		OK	
CU_003	CU_003_RG_002	2 - Moyenne		OK	
CU_003	CU_003_RG_003	3 - Basse		OK	
CU_003	CU_003_RG_004	1 - Haute		OK	
CU_004	CU_004_RG_001	1 - Haute		OK	
CU_004	CU_004_RG_002	1 - Haute		OK	
CU_004	CU_004_RG_003	1 - Haute		OK	
CU_004	CU_004_RG_004	1 - Haute		OK	
CU_004	CU_004_RG_005	2 - Moyenne		OK	
CU_004	CU_004_RG_006	2 - Moyenne		OK	
CU_004	CU_004_RG_007	2 - Moyenne		OK	
CU_004	CU_004_RG_008	1 - Haute		OK	
CU_004	CU_004_RG_009	3 - Basse		OK	
CU_004	CU_004_RG_010	2 - Moyenne		OK	
CU_004	CU_004_RG_011	2 - Moyenne		OK	
CU_004	CU_004_RG_012	1 - Haute		OK	
CU_004	CU_004_RG_013	1 - Haute		OK	
CU_004	CU_004_RG_014	2 - Moyenne		OK	
CU_004	CU_004_RG_015	3 - Basse		OK	
CU_005	CU_005_RG_001	2 - Moyenne		OK	
CU_005	CU_005_RG_002	1 - Haute		OK	
CU_005	CU_005_RG_003	2 - Moyenne		OK	
CU_005	CU_005_RG_004	1 - Haute		OK	
CU_006	CU_006_RG_001	1 - Haute		OK	
CU_006	CU_006_RG_002	3 - Basse		OK	
CU_006	CU_006_RG_003	3 - Basse		OK	
CU_006	CU_006_RG_004	2 - Moyenne		OK	
CU_006	CU_006_RG_005	2 - Moyenne		OK	
CU_006	CU_006_RG_006	2 - Moyenne		OK	
CU_006	CU_006_RG_007	1 - Haute		OK	
CU_006	CU_006_RG_008	1 - Haute		OK	
CU_006	CU_006_RG_009	3 - Basse		OK	
CU_006	CU_006_RG_010	3 - Basse		OK	
CU_006	CU_006_RG_011	2 - Moyenne		OK	
CU_006	CU_006_RG_012	3 - Basse		OK	
CU_006	CU_006_RG_013	2 - Moyenne		OK	
CU_006	CU_006_RG_014	2 - Moyenne		OK	
CU_006	CU_006_RG_015	3 - Basse		OK	
CU_007	CU_007_RG_001	3 - Basse		OK	
CU_007	CU_007_RG_002	3 - Basse		OK	
CU_007	CU_007_RG_003	1 - Haute		OK	
CU_008	CU_008_RG_001	3 - Basse		OK	

CU_008	CU_008_RG_002	3 - Basse	OK
CU_008	CU_008_RG_003	2 - Moyenne	OK
CU_008	CU_008_RG_004	1 - Haute	OK
CU_008	CU_008_RG_005	1 - Haute	OK
CU_008	CU_008_RG_006	2 - Moyenne	OK
CU_008	CU_008_RG_007	3 - Basse	OK
CU_008	CU_008_RG_008	1 - Haute	OK
CU_008	CU_008_RG_009	2 - Moyenne	OK
CU_008	CU_008_RG_010	3 - Basse	OK
CU_009	CU_009_RG_001	1 - Haute	OK
CU_009	CU_009_RG_002	3 - Basse	OK
CU_009	CU_009_RG_003	2 - Moyenne	OK
CU_009	CU_009_RG_004	3 - Basse	OK
CU_009	CU_009_RG_005	1 - Haute	OK
CU_009	CU_009_RG_006	2 - Moyenne	OK
CU_009	CU_009_RG_007	3 - Basse	OK

IV - Bilan et perspectives

➤ Points positifs et négatifs

- **Points positifs :**

- Application fonctionnelle
- Respect des consignes : contrat rempli
- Pas de problèmes détectés

Nous concernant :

- Travail à deux : mise en place de règles, évolution en équipe, partage du travail
- Mise en pratique de l'utilisation de GIT
- Réalisation d'une application de Winforms

- **Points négatifs :**

- Pas entière compréhension de GIT (problème avec les commit)
- Pas de démonstration interactive des tests (simplement un screenshot d'exemple)
- Pas de possibilité d'adapter la taille de la fenêtre
- 10 questions seulement par difficulté dans problèmes physiques

➤ Evolutions possibles

Nous pourrions ajouter une classe Utilisateur pour enregistrer ses scores facilement (dans notre application, on pourrait ajouter une méthode de sérialisation de la liste Scores mais il vaut mieux créer une classe spécifique). Nous demanderions à l'utilisateur son nom, son âge, son adresse mail par exemple.

Annexe

➤ Maquettes réalisées au début du projet

Test ESA—Menu principal

Choisir une catégorie de test

Effectué

Facile Difficile

☐ Perception et mémoire associative

☐ Attention et concentration

☐ Calcul mental

☐ Problèmes mathématiques

☐ Problèmes physiques

Choisir une difficulté


☐ Facile ☐ Difficile

Résultats OK

Test ESA—Enoncé

Enoncé :

Exemple :



Difficulté : facile donc

Menu OK

Test ESA— %Nom du test%

Série n°x

Règle pour cette série :

Menu OK

Test ESA— %Nom du test%

Résultats

X bonnes réponses

Y mauvaises réponses

50% de réussites

Menu

Fin de tous les tests

Test ESA— Perception et mémoire associative

Question 1/10 :

Enoncé

Question :

A 2	B 8	C 1	D 8
E 7	F 1	G 5	H 2
I 9	J 3	K 8	L 1

Menu

Test ESA— Perception et mémoire associative

Question 1/10 :

Enoncé

Réponse :

Rond bleu E

Rond bleu G


Rond bleu J

Menu OK

Test ESA — Attention et concentration

Bouton 1—Même nombre de point
Bouton 2—Même couleur
Bouton 3—Autre cas

?



Butt 1
Butt 2
Butt 3

Menu

Test ESA — Scores

	Facile	Difficile
Perception et mémoire associative	50%	20%
Attention et concentration		
Calcul mental		
Problèmes mathématiques		
Problème physiques		

Menu

Test ESA — Calcul mental

Vous voulez faire :

+

-

x

/

Menu

Test ESA — Calcul mental

Calculez de tête :

12+56+130

Réponse :

Menu OK

Pour le problème 3 (choix)

Test ESA — Calcul mental

Bonne réponse
Ou
Faux, la bonne réponse était ...

Menu OK

Test ESA — Mathématiques

Question : -----

Réponses :

Test ESA — Mathématiques

Question : -----

Bonne réponse : **A**

➤ Spécifications fonctionnelles détaillées

1. CU_001 – Choisir un test

1.1. Description

Sur la page du menu, on doit pouvoir choisir un test avec sa difficulté et le lancer. On doit pouvoir voir quels tests ont déjà été réalisés.

On a aussi accès au tableau des scores.

1.2. Description des champs (CU_001_RG_000)

Nom du champ	Type du champ	Remarques
Titre	Texte	
Bienvenue	Texte	
Choisir une catégorie de test	GroupBox	
Type de test	RadioButton	
Test Réalisés	CheckBox	
Choisir une difficulté	GroupBox	
Type de difficulté	RadioButton	
Scores	Bouton	
Lancer le test	Bouton	

1.3. Règles de gestion

Identifiant	Description
CU_001_RG_001	La liste des tests doit être visualisable affichée. Si un test a déjà été effectués (et dans quel difficulté), on coche la case correspondante.
CU_001_RG_002	Il n'est pas permis de sélectionner plusieurs tests.
CU_001_RG_003	Il est possible de choisir la difficulté dans laquelle on souhaite effectuer le test.
CU_001_RG_004	Au chargement de la page, les tests qui ont déjà été réalisés sont cochés.
CU_001_RG_005	Au clic sur le bouton Scores, on accède à la page Score. (Voir CU_002)
CU_001_RG_006	Au clic sur le bouton Lancer le test, on accède à la page d'énoncé du test choisi (avec la difficulté choisie). (Voir CU_003)

2. CU_002 – Visualiser les scores

2.1. Description

On doit pouvoir visualiser les moyennes des scores obtenus par test et pour chacune des difficultés.

2.2. Description des champs (CU_002_RG_000)

Nom du champ	Type du champ	Remarques
Titre	Texte	
Menu Principale	Bouton	
Liste des tests	Liste	
Tableau des scores	Tableau	Possède deux lignes (facile & difficile) et deux colonnes (score moyen et score maximum).

2.3. Règles de gestion

Identifiant	Description
CU_002_RG_001	Au clic sur le bouton menu on renvoie vers le menu (CU_001).
CU_002_RG_002	Lorsqu'on choisit un test dans la liste, le tableau des scores affiche les scores du test correspondant.
CU_002_RG_003	Le tableau affiche None si le test n'a pas été réalisé.

3. CU_003 – Lancer le test

3.1. Description

On doit pouvoir savoir en quoi va consister le test, connaître les consignes et avoir un exemple de ce qui va suivre.

3.2. Description des champs (CU_003_RG_000)

Nom du champ	Type du champ	Remarques
Titre	Texte	
Menu Principale	Bouton	
Enoncé	Texte	
Exemple	Texte + Image	
Difficulté	Texte	
Validation (OK)	Bouton	

3.3. Règles de gestion

Identifiant	Description
CU_003_RG_001	La page doit afficher les consignes du test, un exemple imagé du test ainsi que les règles s'appliquant à la difficulté choisie.
CU_003_RG_002	Au clic sur le bouton menu on renvoie vers le menu (CU_001).
CU_003_RG_003	Le titre doit afficher le nom du test sélectionné.
CU_003_RG_004	Au clic sur le bouton de validation on arrive sur la page du test (dans le cas du test 1 : CU_004, du test 3 : CU_007 et des tests 4 et 5 : CU_008), ou sur la page de la série (dans le cas du test 2 : CU_006).

4. CU_004 – Réaliser le test 1

4.1. Description

On doit pouvoir réaliser le test jusqu'au bout et connaître son score.

4.2. Description des champs (CU_004_RG_000)

Nom du champ	Type du champ	Remarques
Titre	Texte	
Menu Principale	Bouton	
Lancer le test	Bouton	
Numéro de la question	Texte	
Question	Texte	
Affichage des figures	Zone de dessin	
Réponse	GroupBox	Contient 3 ou 4 TextBox
Validation (OK)	Bouton	
Bonne/Mauvaise réponse	Texte	
Temps	Entier	Le temps modélisé par un décompte et permet à l'utilisateur de savoir combien de temps il reste avant la disparition des figures.

4.3. Règles de gestion

Identifiant	Description
CU_004_RG_001	Lors du chargement de la page, seul le titre, le bouton menu et le bouton lancer doivent être visible.
CU_004_RG_002	Le titre doit afficher le nom du test.
CU_004_RG_003	Au clic sur le bouton menu on renvoie vers le menu (CU_001).

CU_004_RG_004	Au clic sur le bouton Lancer le test, le numéro de la question et l'énoncé de la question apparaissent. L'énoncé est généré aléatoirement et demande à l'utilisateur de regarder une forme et une couleur.
CU_004_RG_005	Au clic sur le bouton Lancer le test, une série aléatoire de figures (dont 3 ou 4 correspondent à la règle générée cf. CU_004_RG_004) apparaît sur la fenêtre.
CU_004_RG_006	Au clic sur le bouton Lancer le test, une image de sablier ainsi que le temps apparaissent au bas de la fenêtre. Le temps commence à s'écouler.
CU_004_RG_007	La valeur de départ du compte à rebours dépend de la difficulté choisie (4 secs ou 2 secs).
CU_004_RG_008	Lorsque le temps est écoulé, les figures s'effacent pour laisser place aux boîtes de réponse. Le compteur et le sablier disparaissent et le bouton OK apparaît.
CU_004_RG_009	Lorsqu'on rentre une valeur qui n'est pas un entier dans une des zones de texte, celle-ci se vide automatiquement.
CU_004_RG_010	Au clic sur Ok, un message apparaît sous chaque réponse pour valider ou invalider la réponse. Ce message reste environ 3 secondes.
CU_004_RG_011	Une réponse laissée vide doit être considérée comme une mauvaise réponse.
CU_004_RG_012	Lors de la disparition des messages (cf CU_004_RG_009), la série de figure suivante apparaît la règle change, le numéro de la question est mis à jour et le compte à rebours réapparaît.
CU_004_RG_013	On répond à 10 questions.
CU_004_RG_014	Lorsqu'on a répondu à la dernière question, on vide la fenêtre et un pop-up apparaît pour donner le score (en pourcentage).
CU_004_RG_015	Lorsqu'on clique sur OK (sur le pop-up), on est renvoyé vers le menu (CU_001).

5. CU_005 – Affichage de la série

5.1. Description

On doit pouvoir savoir en quoi va consister le test, connaître les consignes et avoir un exemple de ce qui va suivre.

5.2. Description des champs (CU_005_RG_000)

Nom du champ	Type du champ	Remarques
Titre	Texte	
Menu Principale	Bouton	
Numéro de la série	Texte	
Enoncé	Texte	
Lancer le test	Bouton	

Règles de gestion

Identifiant	Description
CU_005_RG_001	Lors du chargement de la page on affiche le numéro de la série en cours et les règles relatives à la série à venir.
CU_005_RG_002	Les règles sont générées de manière aléatoire pour la première série ou de la réalisation du test en mode difficile. Les règles sont récupérées de la série précédente dans tous les autres cas.
CU_005_RG_003	Au clic sur le bouton menu on renvoie vers le menu (CU_001).
CU_005_RG_004	Au clic sur le bouton Lancer le test, on lance le test (CU_006).

6. CU_006 – Réaliser le test 2**6.1. Description**

On doit pouvoir réaliser le test jusqu'au bout et connaître son score.

6.2. Description des champs (CU_006_RG_000)

Nom du champ	Type du champ	Remarques
Titre	Texte	
Menu Principale	Bouton	
Numéro de la question	Texte	
Règles	Texte	
Affichage de la figure	Zone de dessin	
Réponse	GroupBox	Contient 3 boutons (butt1, butt2 et butt3)
Bonne/Mauvaise réponse	Texte	
Temps	Entier	Le temps modélisé par un décompte et permet à l'utilisateur de savoir combien de temps pour répondre.

6.3. Règles de gestion

Identifiant	Description
CU_006_RG_001	Lors du chargement de la page, on affiche le titre, le bouton menu, le numéro de la question, les règles, la figure et le bouton butt3.
CU_006_RG_002	Le titre doit afficher le nom du test.
CU_006_RG_003	Au clic sur le bouton menu on renvoie vers le menu (CU_001).
CU_006_RG_004	Les règles à l'écran doivent correspondre aux règles vues dans la page série (CU_005).
CU_006_RG_005	La première figure doit être générée de manière aléatoire.
CU_006_RG_006	Au clic sur le bouton butt3, il disparaît et on affiche ok pendant 3 secondes.
CU_006_RG_007	La liste des figures est générée de manière aléatoire mais elle vérifie qu'il faut appuyer au moins une fois sur chaque bouton lors de la réalisation de la série.
CU_006_RG_008	Lorsque les trois secondes sont écoulées, on trace la nouvelle figure qui possède au maximum un point commun avec la précédente, on affiche aussi les trois boutons.
CU_006_RG_009	Dans le cas du test difficile, on lance un compte à rebours.
CU_006_RG_010	Lors de la fin du compte à rebours, on cache les boutons et on affiche un message donnant la réponse sur laquelle il fallait appuyer.
CU_006_RG_011	Si l'utilisateur clic sur un des boutons (avant le fin du compte à rebours dans le cas difficile), on cache les boutons et on affiche un message lui disant si il s'est trompé ou non.
CU_006_RG_012	On affiche le message pendant 3 secondes (cf CU_006_RG_011 & CU_006_RG_010).
CU_006_RG_013	Au bout des 3 secondes, on efface le message et on recommence (à partir du CU_006_RG_008).
CU_006_RG_014	Lorsqu'on réalise la dernière question de la série, un pop-up indique le score (en pourcentage) réalisé par l'utilisateur.
CU_006_RG_015	Lorsqu'on clique sur OK (sur le pop-up), on est renvoyé vers la page série (CU_005) ou vers la page menu (CU_001) si on a fini la dernière série.

7. CU_007 – Choisir l'opération du test 3

7.1. Description

On doit pouvoir choisir l'opération avec laquelle on souhaite réaliser le test 3.

7.2. Description des champs (CU_007_RG_000)

Nom du champ	Type du champ	Remarques
Titre	Texte	
Menu Principale	Bouton	
Opérations	Boutons	Il y a 4 boutons : +, -, x, /

7.3. Règles de gestion

Identifiant	Description
CU_007_RG_001	Le titre doit afficher le nom du test.
CU_007_RG_002	Au clic sur le bouton menu on renvoie vers le menu (CU_001).
CU_007_RG_003	Au clic sur un bouton opération, on exécute renvoie vers la page du test. (CU_008)

8. CU_008 – Réaliser le test 3**8.1. Description**

On doit pouvoir réaliser le test jusqu'au bout et connaître son score.

8.2. Description des champs (CU_008_RG_000)

Nom du champ	Type du champ	Remarques
Titre	Texte	
Menu Principale	Bouton	
Numéro de la question	Texte	
Opération	Texte	
Réponse	GroupBox	
Bonne/Mauvaise réponse	Texte	
Validation (OK)	Bouton	
Temps	Entier	Le temps modélisé par un décompte et permet à l'utilisateur de savoir combien de temps pour répondre.

8.3. Règles de gestion

Identifiant	Description
CU_008_RG_001	Lors du chargement de la page, on affiche le titre et le bouton menu.
CU_008_RG_002	Au clic sur le bouton menu on renvoie vers le menu (CU_001).
CU_008_RG_003	Dans le cas d'un test facile, on affiche le numéro de la question, l'opération à réaliser, la boîte réponse ainsi que le bouton OK.
CU_008_RG_004	Dans le cas d'un test difficile, on affiche un bouton Lancer le test.
CU_008_RG_005	Lorsqu'on clique sur Lancer le test, on affiche un compte à rebours et tout ce qui était présent dans le cas facile.
CU_008_RG_006	Lorsque l'utilisateur clique sur OK ou que le temps alloué s'écoule, on affiche un message disant à l'utilisateur si il s'est trompé ou non. Le bouton OK disparaît.

CU_008_RG_007	On affiche le message pendant 3 secondes.
CU_008_RG_008	Au bout des 3 secondes, on génère un nouveau calcul, on change le numéro de la question et on réaffiche la boîte de réponse et le bouton OK.
CU_008_RG_009	Lorsqu'on réalise la dernière question de la série, un pop-up indique le score (en pourcentage) réalisé par l'utilisateur.
CU_008_RG_010	Lorsqu'on clique sur OK (sur le pop-up), vers la page menu (CU_001).

9. CU_009 – Réaliser du test 4 ou du test 5

9.1. Description

On doit pouvoir réaliser le test jusqu'au bout et connaître son score.

9.2. Description des champs (CU_009_RG_000)

Nom du champ	Type du champ	Remarques
Titre	Texte	
Menu Principale	Bouton	
Numéro de la question	Texte	
Question	Texte	
Réponses possibles	GroupBox	Contient entre 2 et 4 boutons réponse
Bonne/Mauvaise réponse	Texte	

9.3. Règles de gestion

Identifiant	Description
CU_009_RG_001	Lors du chargement de la page, on affiche le titre, le bouton menu, le numéro de la question, la question et les boutons réponse.
CU_009_RG_002	Au clic sur le bouton menu on renvoie vers le menu (CU_001).
CU_009_RG_003	Au clic sur un bouton réponse, on affiche un message disant à l'utilisateur s'il s'est trompé ou non. On fait aussi disparaître les autres boutons.
CU_009_RG_004	On affiche le message pendant 3 secondes.
CU_009_RG_005	Au bout des 3 secondes, on affiche une nouvelle question et les réponses correspondantes.
CU_009_RG_006	Lorsqu'on réalise la dernière question de la série, un pop-up indique le score (en pourcentage) réalisé par l'utilisateur.
CU_009_RG_007	Lorsqu'on clique sur OK (sur le pop-up), on est renvoyé vers la page menu (CU_001).

➤ Tableau des exigences

Exigences ▼	Risque métier ▼	Risque technique ▼	Criticité résultante ▼
CU_001_RG_001	1 - Elevé	3 - Bas	1 - Haute
CU_001_RG_002	1 - Elevé	3 - Bas	1 - Haute
CU_001_RG_003	2 - Moyen	3 - Bas	2 - Moyenne
CU_001_RG_004	3 - Bas	1 - Elevé	1 - Haute
CU_001_RG_005	3 - Bas	3 - Bas	3 - Basse
CU_001_RG_006	1 - Elevé	3 - Bas	1 - Haute
CU_002_RG_001	2 - Moyen	3 - Bas	2 - Moyenne
CU_002_RG_002	2 - Moyen	3 - Bas	2 - Moyenne
CU_002_RG_003	3 - Bas	3 - Bas	3 - Basse
CU_003_RG_001	1 - Elevé	3 - Bas	1 - Haute
CU_003_RG_002	3 - Bas	3 - Bas	3 - Basse
CU_003_RG_003	3 - Bas	3 - Bas	3 - Basse
CU_003_RG_004	1 - Elevé	3 - Bas	1 - Haute
CU_004_RG_001	1 - Elevé	3 - Bas	1 - Haute
CU_004_RG_002	3 - Bas	3 - Bas	3 - Basse
CU_004_RG_003	3 - Bas	3 - Bas	3 - Bas
CU_004_RG_004	1 - Elevé	2 - Moyen	1 - Haute
CU_004_RG_005	1 - Elevé	2 - Moyen	1 - Haute
CU_004_RG_006	3 - Bas	2 - Moyen	2 - Moyenne
CU_004_RG_007	3 - Bas	3 - Bas	3 - Basse
CU_004_RG_008	1 - Elevé	3 - Bas	1 - Haute
CU_004_RG_009	3 - Bas	3 - Bas	3 - Basse
CU_004_RG_010	3 - Bas	3 - Bas	3 - Basse
CU_004_RG_011	2 - Moyen	3 - Bas	2 - Moyenne
CU_004_RG_012	1 - Elevé	2 - Moyen	1 - Haute
CU_004_RG_013	2 - Moyen	3 - Bas	2 - Moyenne
CU_004_RG_014	3 - Bas	3 - Bas	3 - Basse
CU_004_RG_015	3 - Bas	3 - Bas	3 - Basse
CU_005_RG_001	1 - Elevé	3 - Bas	1 - Haute
CU_005_RG_002	2 - Moyen	2 - Moyen	2 - Moyenne
CU_005_RG_003	3 - Bas	3 - Bas	3 - Basse
CU_005_RG_004	1 - Elevé	3 - Bas	1 - Haute
CU_006_RG_001	1 - Elevé	2 - Moyen	1 - Haute
CU_006_RG_002	3 - Bas	3 - Bas	3 - Basse
CU_006_RG_003	3 - Bas	3 - Bas	3 - Basse
CU_006_RG_004	1 - Elevé	3 - Bas	1 - Haute
CU_006_RG_005	2 - Moyen	3 - Bas	2 - Moyenne
CU_006_RG_006	2 - Moyen	3 - Bas	2 - Moyenne

CU_006_RG_007	2 - Moyen	1 - Elevé	1 - Haute
CU_006_RG_008	2 - Moyen	2 - Moyen	2 - Moyenne
CU_006_RG_009	2 - Moyen	3 - Bas	2 - Moyenne
CU_006_RG_010	3 - Bas	3 - Bas	3 - Basse
CU_006_RG_011	2 - Moyen	3 - Bas	3 - Basse
CU_006_RG_012	3 - Bas	3 - Bas	3 - Basse
CU_006_RG_013	1 - Elevé	3 - Bas	1 - Haute
CU_006_RG_014	3 - Bas	3 - Bas	3 - Basse
CU_006_RG_015	3 - Bas	3 - Bas	3 - Basse
CU_007_RG_001	3 - Bas	3 - Bas	3 - Basse
CU_007_RG_002	3 - Bas	3 - Bas	3 - Basse
CU_007_RG_003	1 - Elevé	3 - Bas	1 - Haute
CU_008_RG_001	3 - Bas	3 - Bas	3 - Basse
CU_008_RG_002	3 - Bas	3 - Bas	3 - Basse
CU_008_RG_003	1 - Elevé	3 - Bas	3 - Basse
CU_008_RG_004	3 - Bas	3 - Bas	3 - Basse
CU_008_RG_005	1 - Elevé	3 - Bas	3 - Basse
CU_008_RG_006	3 - Bas	2 - Moyen	2 - Moyenne
CU_008_RG_007	3 - Bas	3 - Bas	3 - Basse
CU_008_RG_008	1 - Elevé	3 - Bas	3 - Basse
CU_008_RG_009	3 - Bas	3 - Bas	3 - Basse
CU_008_RG_010	3 - Bas	3 - Bas	3 - Basse
CU_009_RG_001	1 - Elevé	3 - Bas	1 - Haute
CU_009_RG_002	3 - Bas	3 - Bas	3 - Basse
CU_009_RG_003	1 - Elevé	2 - Moyen	1 - Haute
CU_009_RG_004	3 - Bas	3 - Bas	3 - Basse
CU_009_RG_005	1 - Elevé	3 - Bas	1 - Haute
CU_009_RG_006	3 - Bas	3 - Bas	3 - Basse
CU_009_RG_007	3 - Bas	3 - Bas	3 - Basse