

ARTHUR AUBIN  
MAXIME DA SILVA

PROJET GENIE LOGICIEL  
**TESTS POUR LA SELECTION  
DES ASTRONAUTES DE L'ESA**

---

# PLAN

## INTRODUCTION

## 1. SPECIFICATIONS GENERALES

1.1. ANALYSE FONCTIONNELLE

1.2. BESOINS UTILISATEUR

1.3. SCHEMAS UML

1.4. ARCHITECTURE APPLICATION

1.5. PLANNING ET REPARTITION DES TACHES

## 2. SPECIFICATION DETAILLEES

2.1. FORMULAIRES

2.2. CHOIX DE CONCEPTION ET DE PRODUCTION

## 3. RESULTATS DES TESTS

3.1. VISUELS DES RESULTATS

3.2. PROTOCOLES DES TESTS

3.3. RESULTATS

## 4. BILAN ET PERSPECTIVES

4.1. POINT POSITIFS ET NEGATIFS

4.2. EVOLUTION POSSIBLES

## CONCLUSION

# INTRODUCTION

Ce projet a été réalisé dans le cadre du module de génie logiciel du premier semestre de la 2<sup>ème</sup> année du cursus ingénieur au sein de l'ENSC (Ecole Nationale Supérieure de Cognitique de Bordeaux). Il était question de familiariser avec la programmation événementielle en C# .NET via notamment les « Winforms » mais également d'appliquer la conception et la méthodologie à un projet. Ce projet a démarré le 03 Novembre et c'est terminé le 16 décembre 2016. Il a été encadré et supervisé par Mme Clermont, M Pesquet et M Salotti.

## 1. SPECIFICATIONS GENERALES

### 1.1. ANALYSE FONCTIONNELLE

La principale fonction de ce logiciel est de proposer un ensemble de tests pour la sélection des astronautes de l'agence spatiale européenne. Un ensemble de 5 tests indépendants devront être présentés à l'utilisateur, ces tests portent sur les fonctions cognitives des utilisateurs mettant en jeu la Perception, la mémoire, l'attention, la concentration, le calcul mental ainsi que des connaissances en mathématiques et physique. Ces tests possèdent deux niveaux de difficultés chacun (facile et difficile) affectant des paramètres tels que le temps de présentation des énoncés ou la variabilité des questions posées. A la fin de chaque test le résultat de l'utilisateur est affiché sous la forme d'un pourcentage de bonne réponse. La dernière performance de chaque test sera aussi affichée sur le menu principale afin que l'utilisateur puisse avoir une vue globale de ses résultats de l'ensemble des tests.

### 1.2. BESOINS UTILISATEURS

Notre interface doit répondre à certaines contraintes afin que l'utilisateur puisse interagir correctement et en tout confort avec celle-ci. En terme d'utilisabilité car l'interface doit permettre une identification immédiate des différents tests mais aussi aux autres options qui lui sont disponibles, le tout dès la première utilisation. En terme d'efficacité car il est essentiel pour notre solution de minimiser voir éviter les possibilités d'erreurs dans le parcours de l'utilisateur, ce critère d'efficacité est très important car dans le contexte de l'application une erreur pourrait fausser complètement les résultats des tests. Une dernière contrainte liée au besoin de l'utilisateur est la facilité, car l'utilisateur doit pouvoir circuler au travers de notre application sans rencontrer de difficulté, notamment lors des tests car sur certains tests le facteur temps intervient dans le déroulement du test donc la facilité de compréhension de l'interface doit permettre à l'utilisateur de s'orienter et comprendre rapidement le contenu de chaque fenêtre.

L'interface graphique avec laquelle l'utilisateur interagit doit pouvoir présenter les aspects ergonomiques classiques :

- Présentation de l'ensemble des tests sur une page afin de pouvoir choisir

- navigabilité entre les différents tests rapide et cohérente
- rapidité de lancement des tests
- Affichage clair des résultats en fin de test

Afin de bien orienter le développement de nos différentes interfaces nous avons dans un premier temps conçu les maquettes de la page d'accueil et des pages de tests :

## Le test de perception :

### Consigne :

Mémoriser les valeurs de tous les ronds bleus

A 2	B 8	C 1	D 4
E 7	F 3	G 8	H 5
I 2	J 1	K 6	L 9

Valeur de E

Valeur de G

Valeur de J

Valider

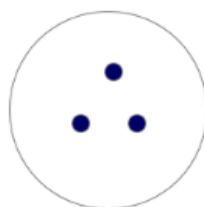
### Consigne :

Mémoriser les valeurs de tous les ronds bleus

## Le test d'attention :

### Consigne :

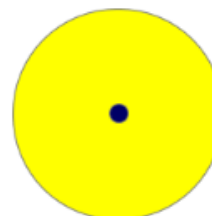
Appuyer sur le bouton 1 si deux objets consécutifs ont la même couleur  
Appuyer sur le bouton 3 si deux objets consécutifs ont le même nombre de points.  
Appuyer sur le bouton 3 dans tous les autres cas.



Bouton 3

### Consigne :

Appuyer sur le bouton 1 si deux objets consécutifs ont la même couleur  
Appuyer sur le bouton 3 si deux objets consécutifs ont le même nombre de points.  
Appuyer sur le bouton 3 dans tous les autres cas.



Bouton 1

Bouton 2

Bouton 3

## Le menu principal :

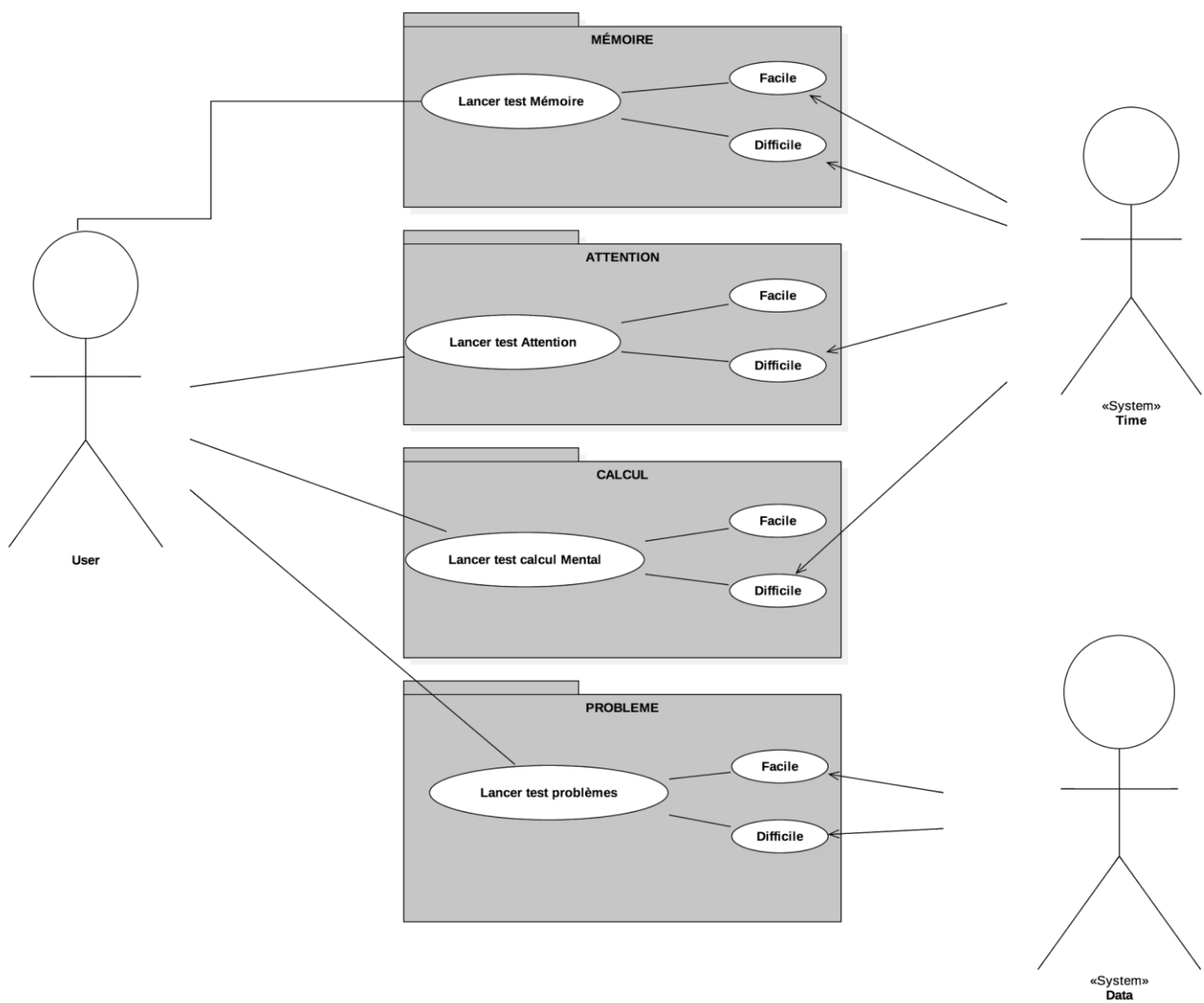


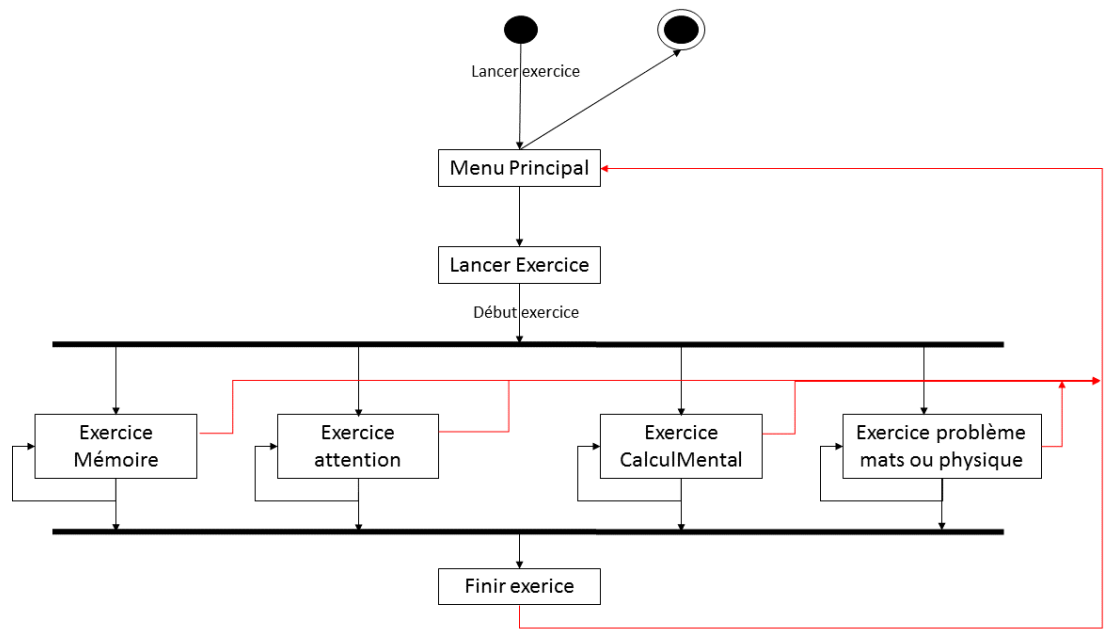
ESA SELECTION

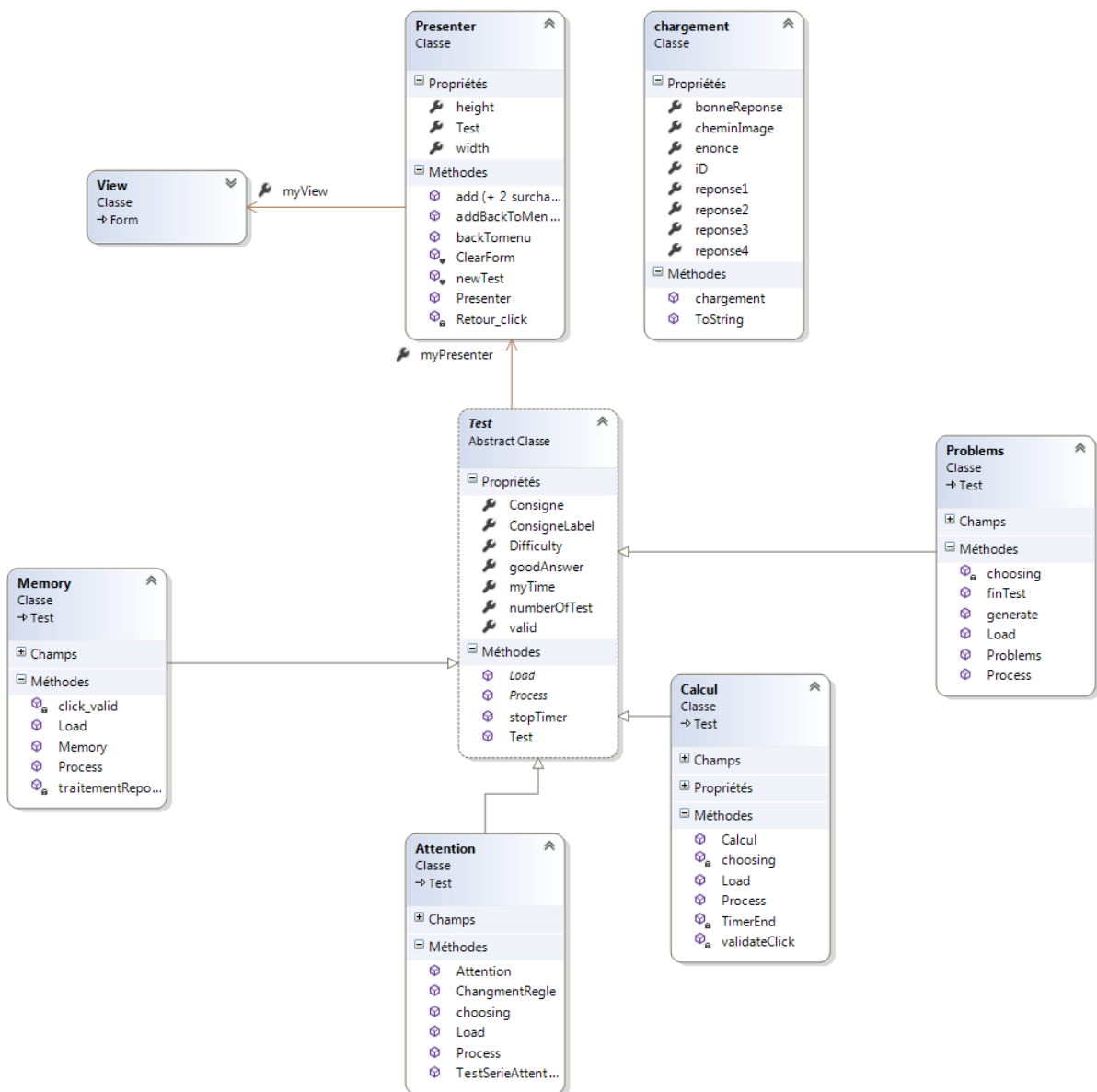
Description  
ipsum dolor sit amet, consectetur adipiscing elit. Aenean euismod bibendum laoreet. Proin gravida dolor sit amet lacus accumsan et viverra justo commodo. Proin sodales pulvinar tempor. Cum sociis natoque penatibus et magnis dis parturient montes, nascetur ridiculus mus. Nam fermentum, nulla luctus pharetra vulputate, felis tellus mollis orci, sed rhoncus sapien nunc eget.

	Perception et mémoire associative	Facile
		Compliqué
	Attention et concentration	Facile
		Compliqué

### 1.3. SCHEMAS UML



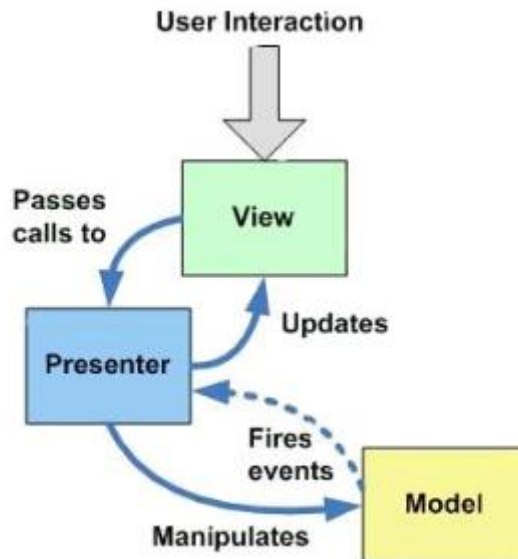




## 1.4. ARCHITECTURE DE L'APPLICATION

Pour ce projet nous avons décidé d'adopter une architecture particulièrement adaptée aux solutions Winforms, à savoir une architecture en Model-View-Presenter. Cette architecture permet une grande séparation des responsabilités (separation of concerns) qui est un principe de design essentiel pour une programmation qui ait un couplage faible (le moins de dépendance possible entre les sous parties) et une cohésion forte (les composants qui font la même "chose" vont au même endroit). Les composants de notre application peuvent donc être utilisés séparément. De plus si nous voulons exporter notre programme sur un nouveau type d'affichage, il nous suffit d'adapter la vue et le presenter, nous n'aurons pas besoin de toucher au modèle.

Notre modèle est un peu particulier, puisque n'ayant que des tests et un menu, il n'y a pas de réel mais il est "caché" dans les classes "test" et ses filles.



*Manipulation "idéale" d'une architecture MVP*

Notre application fonctionne donc comme suit:

- La vue reçoit les événements utilisateurs et gère l'affichage
- Le presenter, en fonction du modèle envoie des requêtes de modifications à la vue.
- Le modèle gère les tests, les données reçues et envoyées charge les xml si besoin et etc...

Notre modèle est donc composé des tests à effectuer. Il y a une classe abstraite tests qui est la généralisation de tous les tests que nous pouvons effectuer. Ensuite nos tests implémentent cette classe (Attention, mémoire, calcul mental et problèmes). Les problèmes mathématiques et physiques sont réunis au sein de la même classe car le traitement est strictement identique, il n'y a que les problèmes en entrée qui diffèrent entre physique et mathématique.

Nous avons également 3 classes "hors" de notre architecture. En effet nous avons une classe permettant de charger les fichiers XML pour les problèmes mathématiques et physiques et nous avons deux classes qui héritent de "Panel" et qui nous permettent d'afficher les formes dans les exercices d'attention et de mémoire.

Pour finir nous avons créé trois énumérés pour gérer la forme des éléments graphiques, la difficulté du niveau et le type de problème. Nous avons également 3 classes "hors" de notre architecture. En effet nous avons une classe permettant de charger les fichiers XML pour les problèmes mathématiques et physiques et nous avons deux classes qui héritent de "Panel" et qui nous permettent d'afficher les formes dans les exercices d'attention et de mémoire.

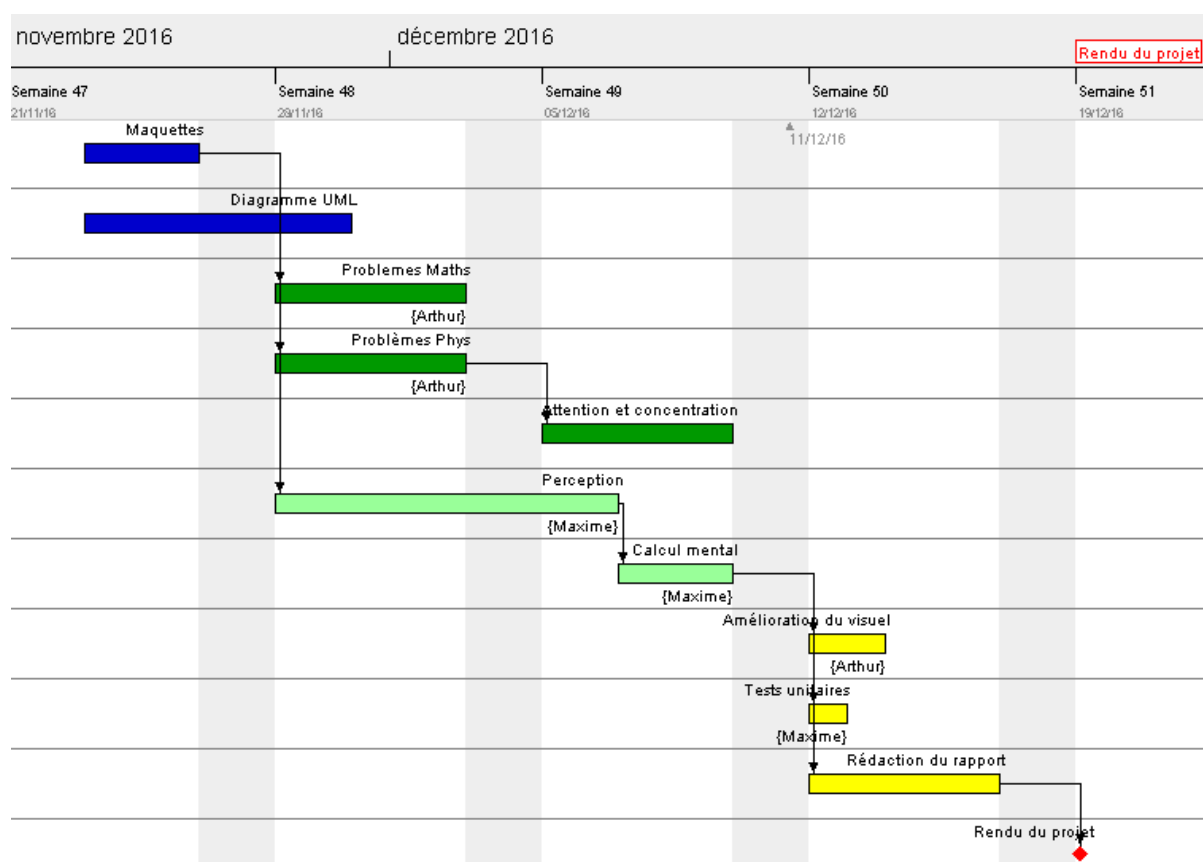
Pour finir nous avons créé trois énumérés pour gérer la forme des éléments graphiques, la difficulté du niveau et le type de problème.



## 1.5. PLANNING ET REPARTITION DES TACHES

Sur l'ensemble des tâches de ce projet nous avons fait le choix de travailler ensemble sur la phase initiale ou de lancement du projet et la phase finale afin de choisir ensemble l'architecture de notre logiciel. Dans la phase de production du code nous nous sommes réparti le travail en s'attribuant à chacun le développement des tests, chose que nous avons bien sûr modulée en fonction des difficultés rencontrées lors du développement et l'avancement du projet. Maxime était responsable du développement des tests "Perception et mémoire associative" et "Calcul mental" ; Arthur était quant à lui responsable des tests "Attention et concentration", "Problèmes de mathématiques" et "Problèmes de physique". Nous parlons de responsabilité car même si la majeure partie du code était produit par le responsable du test, la deuxième personne devait se tenir informé de son avancement et de la qualité de la solution, le début des séances de chaque TP étaient notamment dédiés au compte rendu de l'avancement de chacun sur sa partie.

Voici le planning final que nous avons suivi depuis le début du projet jusqu'à la date de rédaction de ce rapport :



## 2. SPECIFICATION DETAILLEES

### 2.1. FORMULAIRE

Notre solution ne comprend qu'un seul formulaire qui correspond à la page principal, sur laquelle on peut choisir le test à faire et le niveau de difficulté. A chaque lancement d'un test on repart un formulaire vierge à l'aide de la commande `myPresenter.ClearForm()` ce qui nous permet de réintroduire tous les composants à l'aide du code directement.

Nous avons donc décidé de tout concentrer dans ce form pour qu'il représente la vue de notre application. La génération des boutons se fait dynamiquement dans le modèle et ce grâce à l'utilisation de fonctions de notre presenter (`Add` notamment pour ajouter, selon la surcharge une forme graphique ou un bouton).

### 2.2. CHOIX DE CONCEPTION ET DE PRODUCTION

Dans notre production de code nous avons essayé de suivre au mieux certains principes de conception (DRY, KISS, YAGNI, etc...) afin de rendre notre code le plus compréhensible pour d'autres développeurs.

L'exemple le plus significatif de l'utilisation du principe DRY est la conception des tests portant sur les problèmes de mathématiques et de physique car ces deux tests étant très proches nous avons fait le choix de simplement séparer les questions propres à chaque test dans deux fichier XML différents et d'appeler un même constructeur de la classe "Problèmes". En utilisant cette méthode on évite de dupliquer l'ensemble du code permettant d'afficher les questions puis de les traiter, car la seule différence entre les deux tests est l'appelle d'un XML qui diffère selon les cas où l'utilisateur choisit le test de mathématiques ou le test de physique.

Nous avons également voulu appliquer au maximum le partage de fichier grâce au SCM (Source Code Manager) Git et à gitHub, car même si son utilisation était imposée, nous avons voulu l'utiliser un maximum, et tester ses possibilités (avec l'utilisation des branches différentes que le master par exemple)

## 3. RÉSULTATS DES TESTS

### 3.1. VISUELS DES RÉSULTATS

Ici seront présentés les différents visuels de notre application.

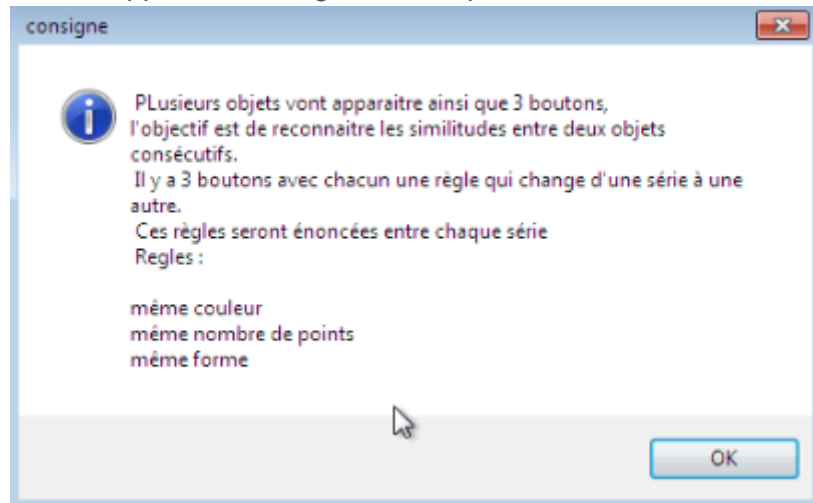
Menu principal :

**ESA SELECTION**

Bienvenu sur l'application de test des astronautes de l'agence spatiale européenne.  
Sur cette application 5 tests sont à votre disposition, chacun d'entre eux va tester une ou plusieurs de vos facultés cognitives.  
Il existe deux niveaux de difficulté pour chaque test (facile/difficile)  
A la fin de chaque exercice votre résultat sera affiché en pourcentage de bonne réponse.

	Difficulté	Dernier résultat
<b>Perception et mémoire associative</b>	<input checked="" type="radio"/> Facile	0
	<input type="radio"/> Compliqué	0
<b>Attention et concentration</b>	<input checked="" type="radio"/> Facile	0
	<input type="radio"/> Compliqué	0
<b>Calcul mental</b>	<input checked="" type="radio"/> Facile	0
	<input type="radio"/> Compliqué	0
<b>Problèmes mathématiques</b>	<input checked="" type="radio"/> Facile	0
	<input type="radio"/> Compliqué	0
<b>Problèmes physiques</b>	<input checked="" type="radio"/> Facile	0
	<input type="radio"/> Compliqué	0

Rappel des consignes à chaque début d'exercice :



Test mémoire :

Retour

Mémorisez les chiffres présents dans les Rond bleu

A 9	B 5	C 9	D 2
E 1	F 6	G 3	H 7
I 9	J 1	K 4	L 8

Quel numéro se cachait derrière la lettre A

0

Valider

Test attention :



Bouton 1  
même couleur

Bouton 2  
même nombre de points

Bouton 3  
Autre cas

Test Calcul mental :

[Retour Menu](#)

+

-

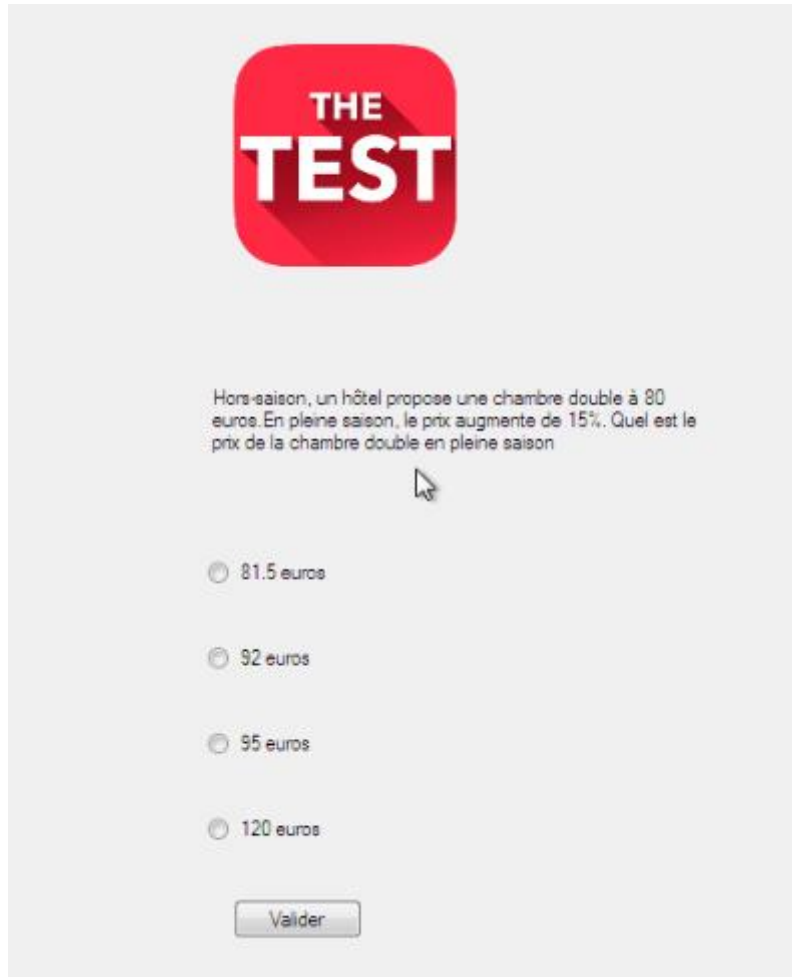
\*

/

551 - 290

Valider

Problèmes de mathématiques ou de physique :



The screenshot shows a mobile application interface with a red header containing the text "THE TEST". Below the header, a math problem is displayed: "Hors-saison, un hôtel propose une chambre double à 80 euros. En pleine saison, le prix augmente de 15%. Quel est le prix de la chambre double en pleine saison". A mouse cursor is pointing at the text. Below the question, there are four radio button options: "81.5 euros", "92 euros", "95 euros", and "120 euros". At the bottom of the form is a button labeled "Valider".

### 3.2. PROTOCOLES DES TESTS

Pour tester notre application nous avons décidé de nous concentrer sur les tests unitaires et fonctionnels. Notre application a cependant été peu sujette aux tests unitaires, puisque la plupart des méthodes sont « instable » unitairement, s'agissant d'action utilisateur. Nos tests unitaires se focalisent donc majoritairement sur les constructeurs des classes pour vérifier qu'à la création de nos instances, les attributs de la classe étaient bien affectés grâce au bon paramètre du constructeur. Cependant nous n'avons pas pu tout tester, puisque nos deux classes permettant l'affichage de formes nécessite un argument (`system.drawing.Color`) qui ne peut pas être utilisé dans un test unitaire, malheureusement.

C'est pourquoi les tests fonctionnels ont été particulièrement important puisqu'ils permettaient de vérifier le bon déroulement de notre application suivant les actions des utilisateurs. Ils nous ont permis de vérifier que le programme compte des points seulement lorsque l'utilisateur rentre une réponse correcte. Chaque page de notre application est donc régie par une vingtaine de tests fonctionnels.

### 3.3. RÉSULTATS

Nos tests unitaires ont tous été passé avec succès.

✓ AttentionTest	797 ms
✓ CalculTest	94 ms
✓ chargementTest	< 1 ms
✓ MemoryTest	184 ms
✓ ProblemsTest	215 ms
✓ ToStringTest	6 ms

Quant aux tests fonctionnels ils ont été séparés en « test ». Au lieu d'énumérer toutes les contraintes pour un aspect de l'application, nous avons préféré énumérer toutes les contraintes pour chaque test. Au final les mêmes contraintes seraient remontées, mais l'organisation est légèrement différente.

M_O1	Le bouton perception et mémoire associative lance le bon test
M_02	Le bouton Attention et concentration lance le bon tes
M_03	Le bouton Calcul mental lance le bon test
M_04	Le bouton problème mathématique lance le bon test
M_05	Le bouton problème physique lance le bon test
M_06	Les niveaux de difficulté sont cohérants avec la difficulté choisie
M_07	Après le retour au menu depuis un exercice, les évènements liés au timer de ce dernier sont stoppés
M_08	Après le retour au menu depuis un exercice, on peut relancer un test

Tests Fonctionnels du menu

P_01	La consigne est rappelée avant le début de l'exercice
P_02	Une réponse correcte est comptée comme correcte
P_03	une réponse incorrecte est compté comme incorrecte
P_04	Le temps de réponse est infini
P_05	Il y a bien dix série de test
P_06	Le pourcentage est cohérent par rapport au nombre de bonne réponses
P_07	Les formes sont toujours soit des carrés soit des cercles
P_08	Les formes sont toujours jaune ou bleu
P_09	Il y a bien entre 3 et 4 réponses à donner à chaque fois



P_10	Les lettres des réponses à donner correspondent bien à la forme sur laquelle il faut se concentrer
P_11	Les chiffres sont bien générés aléatoirement
P_12	les formes sont bien générées aléatoirement
P_13	Le niveau facile correspond à 4 secondes d'observation
P_14	le niveau difficile correspond à 2 secondes d'observation
P_15	le retour menu manuel fonctionne
P_16	A la fin du test, il y a un retour automatique au menu principal

#### Tests Fonctionnels de l'exercice perception

A_01	La consigne est rappelée avant le début de l'exercice
A_02	Une réponse correcte est comptée comme correcte
A_03	une réponse incorrecte est compté comme incorrecte
A_04	Le temps de réponse est infini pour le niveau facile
A_05	Il y a bien 5 série de test
A_06	Le pourcentage est cohérent par rapport au nombre de bonne réponses
A_07	Les formes sont tous des rectangles des carrés ou des cercles
A_08	Les formes sont toujours en jaune vert ou bleu
A_09	Il y a bien au moins une bonne réponse à chaque fois
A_10	La règle change bien toutes les 5 questions
A_11	La forme, la couleur et le nombre de points sont générées aléatoirement
A_12	Il y a 1, 2 ou 3 points à chaque test
A_13	Le niveau facile correspond à une réflexion infinie
A_14	Le niveau difficile correspond à 1s de réponse
A_15	le retour menu manuel fonctionne
A_16	A la fin du test, il y a un retour automatique au menu principal
A_17	Le clic sur un bouton fait passer au test suivant automatique

#### Tests fonctionnels du test attention

C_01	La consigne est rappelée avant le début de l'exercice
C_02	Une réponse correcte est comptée comme correcte
C_03	une réponse incorrecte est compté comme incorrecte
C_04	Le temps de réponse est infini pour le niveau facile
C_05	Il y a bien 5 secondes pour répondre au niveau difficile

C_06	Le pourcentage est cohérent par rapport au nombre de bonne réponses
C_07	Les opérations à résoudre sont bien en adéquation avec le choix utilisateur (+, -, *, /)
C_08	La valeur des opérandes sont toujours en adéquations avec les règles énoncées
C_09	le retour menu manuel fonctionne
C_10	A la fin du test, il y a un retour automatique au menu principal
C_11	Le clic sur un bouton fait passer au test suivant automatique

#### Tests fonctionnels de l'exercice Calcul mental

MP_01	La consigne est rappelée avant le début de l'exercice
MP_02	Une réponse correcte est comptée comme correcte
MP_03	une réponse incorrecte est compté comme incorrecte
MP_04	Les exercices du niveau facile sont bien appelés au niveau facile seulement
MP_05	Les exercices du niveau difficile sont bien appelés au niveau difficile seulement
MP_06	Il y a bien 10 questions par test
MP_07	Le pourcentage est cohérent par rapport au nombre de bonne réponses
MP_08	Il y a toujours 4 réponses proposées
MP_09	Les réponses proposées ne sont pas identiques
MP_10	Il y a bien au une bonne réponse et une seule à chaque fois
MP_11	Il y a un temps de réponse infini
MP_12	Les 10 questions sont prises aléatoirement dans le panel de questions
MP_13	L'exercice mathématique ne demande que des questions qui sont qualifiés comme questions mathématique
MP_14	L'exercice physiques ne demande que des questions qui sont qualifiés comme questions physiques
MP_15	le retour menu manuel fonctionne
MP_16	A la fin du test, il y a un retour automatique au menu principal
MP_17	Le clic sur un bouton fait passer au test suivant

#### Tests Fonctionnels des exercices problème

## 4. BILAN ET PERSPECTIVES

### 4.1. POINTS POSITIFS ET NEGATIFS

Sur l'ensemble du projet nous sommes plutôt satisfait de notre travail, les tests sont tous fonctionnels même si les deux tests concernant les problèmes de mathématiques et de physiques ne sont pas complet car la base de données de problèmes n'est pas complète (30 problèmes pour chacun des tests) car en effet nous avons fait le choix de concentrer notre travail à améliorer le fonctionnement des autres tests plutôt que de remplir le fichier XML en problèmes. Le fichier XML est simple à remplir, une ligne correspond à un problème et il suffit donc de remplir l'ensemble des arguments, ID, énoncé, les quatre réponses et la bonne réponse.

Ce projet nous a permis de mettre en application un ensemble de principes et méthodes de conception vus en cours de Génie logiciel, notamment avec l'utilisation de l'architecture Modèle-Vue-Présentation, très pratique dans notre cas pour construire plusieurs interfaces utilisateurs.

Cependant nous aurions souhaité disposer de plus de temps afin de pouvoir améliorer le visuel de nos interfaces graphiques.

### 4.2. EVOLUTIONS POSSIBLES

Une des évolutions possibles pour notre logiciel est d'améliorer la séparation des responsabilités afin de la rendre plus forte ce qui pourrait rendre plus simple l'ajout d'un test par un développeur extérieur au projet.

De plus il serait intéressant de rajouter une fonctionnalité de sauvegarde des résultats de l'utilisateur afin que celui-ci puisse à l'aide d'un identifiant, accéder à l'ensemble de ses résultats afin de juger de sa performance. On peut imaginer aussi une fonction qui analyse ces performances dans le temps afin de pouvoir visualiser une évolution au cours du temps de la progression de l'utilisateur.

## CONCLUSION

Ce projet a finalement été une très bonne expérience et il nous a permis de mettre en pratique toutes les notions vues en cours sur un projet assez conséquent. Même si notre architecture est loin d'être parfaite, nous avons tout tenté pour suivre les patrons de conception vus en cours.