ÉCOLE DE TECHNOLOGIE SUPÉRIEURE
UNIVERSITÉ DU QUÉBEC

RAPPORT DE PROJET PRÉSENTÉ À
L'ÉCOLE DE TECHNOLOGIE SUPÉRIEURE

COMME EXIGENCE PARTIELLE
À L'OBTENTION DE LA
MAITRISE EN
GÉNIE LOGICIEL

PAR
Elhadj Oumar BARRY

CONCEPTION D'UN LOGICIEL WEB DE CONTRÔLE À DISTANCE D'UN ROBOT

MONTRÉAL, LE 17 FÉVRIER 2011

PRÉSENTATION DU JURY

CE RAPPORT DE PROJET A ÉTÉ ÉVALUÉ

PAR UN JURY COMPOSÉ DE :

Dr. Alain APRIL, directeur de projet

Professeur de Génie Logiciel et membre du Laboratoire de Recherche en Génie Logiciel à l'École de Technologie Supérieure (ÉTS).
M. APRIL est co-éditeur des chapitres de la qualité et de la maintenance du guide *'Software Engineering Body of Knowledge*'. Il est aussi activement impliqué dans la rédaction de normes internationales ISO.

M. Claude LAPORTE, président du jury

Professeur de Génie Logiciel à l'École de Technologie Supérieure (ÉTS).
M. LAPORTE est co-auteur, avec le professeur Alain April, de deux ouvrages sur l'assurance qualité logicielle et l'éditeur du groupe de travail 24 d'un comité de l'ISO.

ELHADJ OUMAR BARRY A FAIT L'OBJET
D'UNE SOUTENANCE DEVANT JURY ET PUBLIC

LE 18 DÉCEMBRE 2011

À L'ÉCOLE DE TECHNOLOGIE SUPÉRIEURE

AVANT-PROPOS

Ce document utilise le gabarit de l'École de technologie supérieure et est conforme aux exigences de l'École.

Il existe plusieurs ouvrages sur les systèmes robotiques et la plupart traite de la conception matérielle des systèmes robots. Il existe peu d'ouvrages en ce qui a trait à la conception logicielle des systèmes robots et encore moins d'ouvrages traitant du contrôle à distance.

Ce document est composé de quatre chapitres qui présentent respectivement les outils et les technologies utilisés dans la conception du logiciel et les raisons pour lesquelles il est nécessaire de motiver ce choix; ensuite nous entreprendrons une revue littéraire qui fera un survol de ce qui se fait actuellement dans ce domaine. De même, nous mettrons en évidence les concepts réutilisables ainsi qu'une proposition de solution pour réaliser enfin la conception d'un prototype.

La lecture de ce document ne nécessite pas de connaissances approfondies des systèmes robotiques; il vise de ce fait un grand public qui a un intérêt pour la conception des systèmes robotiques mobiles ainsi que leur contrôle à distance.

REMERCIEMENTS

Mes sincères remerciements au docteur Alain April pour m'avoir confié la réalisation de ce projet plein de défis, mais aussi je lui témoigne ma reconnaissance pour avoir supervisé mon travail.

Ce travail m'a permis d'appliquer les bonnes pratiques de développement en Génie logiciel et m'a offert la possibilité d'améliorer mes connaissances en ce qui a trait au développement temps réel en robotique, les normes qui l'encadrent ainsi que de l'API java concernant les communications sérielles.

Je remercie également, Monsieur Christian Larose, professeur en conception de systèmes informatiques en temps réel pour son support et sa supervision de l'équipe ayant travaillé dans le « back-end ».

CONCEPTION D'UN LOGICIEL WEB DE CONTRÔLE À DISTANCE D'UN ROBOT

Elhadj Oumar BARRY

RÉSUMÉ

Le projet d'application a deux objectifs : le premier objectif consiste à mettre en pratique les techniques des systèmes embarqués et temps réel, le deuxième objectif consiste à concevoir une application web de contrôle du microcontrôleur.

La méthodologie du cadre de Basili sera utilisée. Premièrement, une revue littéraire sera effectuée pour identifier les avenues existantes afin de réaliser la prise de contrôle d'un équipement à distance par le web pour le contrôleur ATMega32. Par la suite, les exigences seront documentées et un prototype sera développé en utilisant le matériel de microcontrôleur AVR ATMega32 ainsi que la technologie Java/J2EE 5. Il se pourrait que nous utilisions des librairies spécifiques; ces librairies seront identifiées dans les phases d'analyse et de conception. Finalement, des tests et améliorations du prototype seront réalisés. À la fin du projet, l'ensemble des livrables et des codes sources sera remis à une autre équipe qui continuera à faire évoluer ce prototype.

TABLE DES MATIÈRES

LISTE DES TABLEAUX

LISTE DES FIGURES

LISTE DES ABRÉVIATIONS, SIGLES ET ACRONYMES

JAUS	Joint Architecture for Unmanned Systems
JWG	JAUS Working Group
JIRA	Japanese Industrial Robot Association
JavaComm	API Java permettant l'accès aux ports sériels de l'ordinateur
TDD	Test Driven Development
RUR	Rossum's Universal Robots
DoD	Department of Defense
CRUD	Create Read Update Delete
DAO	Data Access Object

INTRODUCTION

Aujourd'hui, les robots jouent un rôle essentiel dans notre quotidien en effectuant des tâches diverses, autrefois accomplies par nous les humains (assemblage dans les usines d'automobiles, montage spécialisé, etc.) ou des tâches impossibles à réaliser par des humains (mesure du niveau de radiation atomique).

À la fin du X^{è} siècle, Nikola Tesla construit des robots contrôlés par radio sans fil. Ceux-ci sont considérés comme les premiers robots mobiles [1].

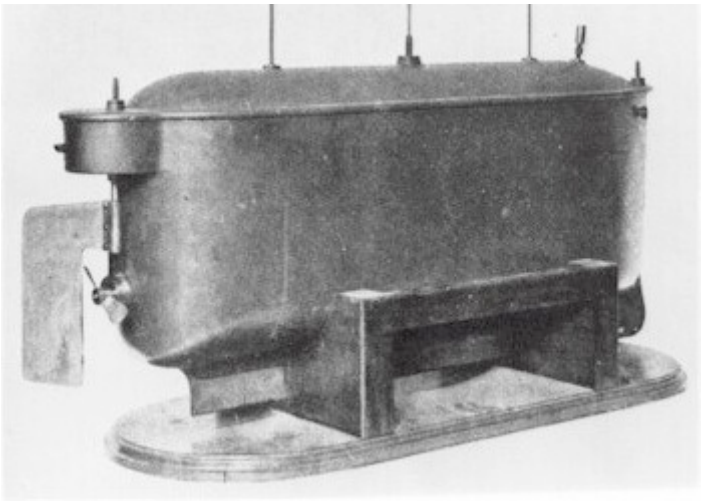


Figure 1 - Le robot de Tesla [1]

La robotique mobile est en quelque sorte un domaine multidisciplinaire qui inclut l'ingénierie et l'informatique. Le mot robot a été introduit par Karel Capek en 1923 au RUR et dérive de deux mots « robota » et « robotnik » respectivement d'origine tchèque et polonais.

Dix-neuf ans plus tard, le Russe Isaac Asimov a introduit les trois lois de la robotique ci-dessous. L'introduction de ces lois lui a valu d'être considéré comme le père de la robotique [1].

Ces trois lois sont :

- « Un robot ne peut porter atteinte à un être humain, ni, restant passif, permettre qu'un être humain ne soit exposé au danger »;
- « Un robot doit obéir aux ordres que lui donne un être humain, sauf si de tels ordres entrent en conflit avec la première loi »;
- « Un robot doit protéger son existence tant que cette protection n'entre pas en conflit avec la première ou la deuxième loi ».

Ce projet vise à concevoir un robot de Paintball; il ne traitera pas de l'aspect matériel; mais de l'aspect logiciel. Plusieurs équipes ont été mises à contribution pour atteindre cet objectif. Premièrement, une équipe constituée de trois personnes qui travaillent dans le « back-end » (celle-ci traite les commandes standards du robot). Deuxièmement, une équipe de visualisation qui travaille sur la visualisation de l'environnement dans lequel évolue le robot. Enfin moi-même, qui dois travailler sur le « front-end » : ce travail consiste à offrir à l'utilisateur une application graphique conviviale qui permet de contrôler le robot à distance. Dans ce projet quelques-uns des défis seront les suivants : l'intégration des modules des différentes équipes, l'établissement d'un protocole de communication simple qui respecte les normes de la robotique ainsi que l'exécution des commandes en temps réel.

Ce document est divisé en trois parties :

Une première partie d'analyse identifiera et documentera les besoins grâce à une exploration et à une analyse des avenues existantes pour la réalisation de la prise de contrôle d'un équipement à distance par le web pour le microcontrôleur ATMega32.

Une deuxième partie de conception explorera les normes architecturales qui encadreront la conception des robots.

Une troisième partie sera réalisée en implémentant un prototype respectant les normes et les standards adaptés au projet de Paintball.

Un site Internet offre aux internautes la possibilité de contrôler gratuitement un robot à distance : http://www.liverobotcontrol.com/live.php. Celui-ci présente une certaine similitude si on le compare à notre projet.

1.1 L'historique

Avant de démarrer le projet, quelques rencontres informelles ont été organisées avec les différentes équipes impliquées. Deux rencontres ont eu lieu avec l'équipe du « back-end » et une rencontre s'est déroulée avec l'équipe de visualisation.

1.1.1 L'équipe du « Back-end » :

La première rencontre a eu pour objectif de nous présenter l'équipement (le robot de Paintball). Une personne responsable a distribué des microcontrôleurs AVR ATMega32 servant à contrôler le robot.
La deuxième rencontre visait à définir un protocole de communication simple entre l'application web et le microcontrôleur embarqué sur le robot. Ce protocole représente les commandes standards du Paintball.

1.1.2 L'équipe de visualisation :

Cette dernière rencontre a permis de comprendre l'approche de visualisation et les technologies utilisées. Elle a aussi offert la possibilité d'étudier la compatibilité des technologies utilisées dans l'application web et celles qui sont utilisées par la visualisation.

1.2 Les choix technologiques et les outils

Les outils et les technologies ont été choisis en fonction de certaines contraintes. Ces contraintes sont : l'évolution future de l'application web, les critères de qualités comme la simplicité, la performance, la flexibilité et le coût [2]. Dans ce projet, les technologies et les outils libres occupent donc une place prépondérante.

1.2.1 Les technologies

Java/J2EE 5: est la technologie retenue dans la cadre de ce projet. La technologie J2EE est l'une des trois éditions de Java : elle définit les standards de développement des applications d'entreprise. Le choix de cette technologie est motivé par sa maturité, son efficacité, sa portabilité ainsi que l'existence de plusieurs « Framework » libres qui s'intègrent à elle et facilitent le développement d'applications robustes.

Dans le contexte de ce projet, il faut mentionner qu'il existe une API java, intitulé « JavaComm » qui offre la possibilité d'établir la communication en utilisant les ports sériels d'un PC. Cette API sera utilisée pour la communication entre l'application web et le microcontrôleur sur le port sériel.

Mentionnons que la technologie Java/J2EE a été introduite en 1995 par la société Sun Microsystems, cette dernière a été rachetée en 2009 par la société Oracle.

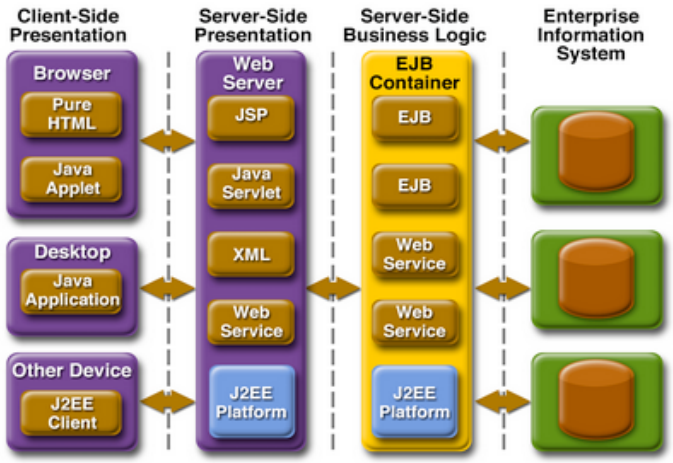


Figure 2 - Modèle d'une application d'entreprise J2EE

Struts 1.3: est un Framework de source libre conçu par la fondation Apache. Ce Framework permet de concevoir des applications web java ayant pour socle, le modèle MVC.

Ce modèle prône la séparation en trois couches d'une application web : la couche présentation, la couche d'affaires et la couche contrôleur. Struts implémente son propre contrôleur; ce Framework est flexible puisque nous pouvons utiliser les technologies de nos choix pour les couches de présentation et d'affaires.

Le choix du Framework Struts est motivé par le souci de construire une application évolutive, flexible et maintenable en utilisant des couches bien séparées : il fournira diverses fonctionnalités utiles dans ce projet. Par exemple, dans ce projet nous pouvons citer, l'internationalisation.

Le Framework Struts est téléchargeable sur le site de la fondation Apache.

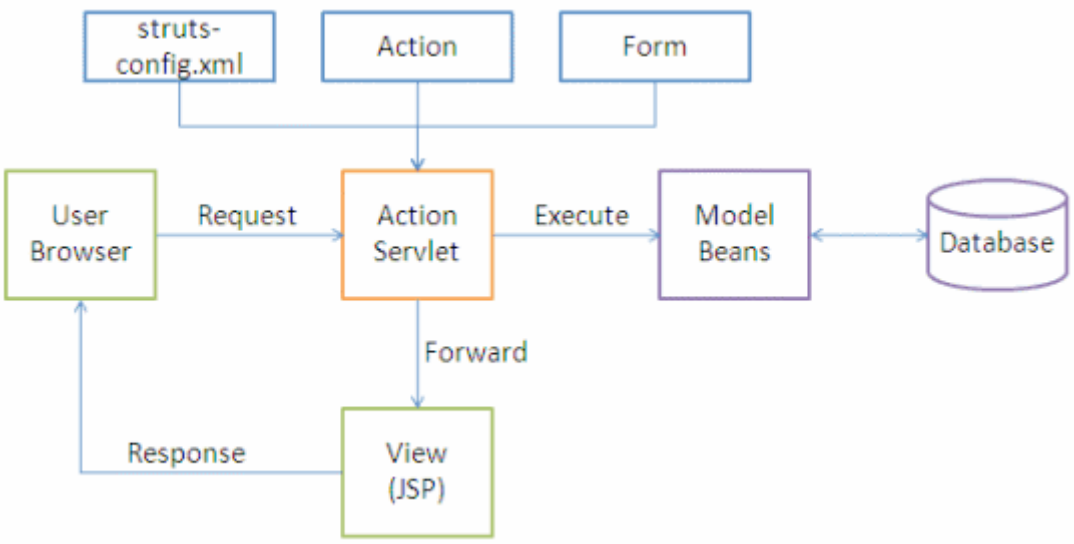


Figure 3 - Modèle MVC de Struts

Hibernate 3.2: est un Framework de source libre de persistance de données conçu par la société JBOSS en 2001 en vue de simplifier la complexité et d'améliorer la performance de la persistance des données.

Le choix du Framework Hibernate est motivé par sa performance, sa fiabilité, la transparence de la persistance. Il offre la possibilité de manipuler les données sous la forme d'objets java.

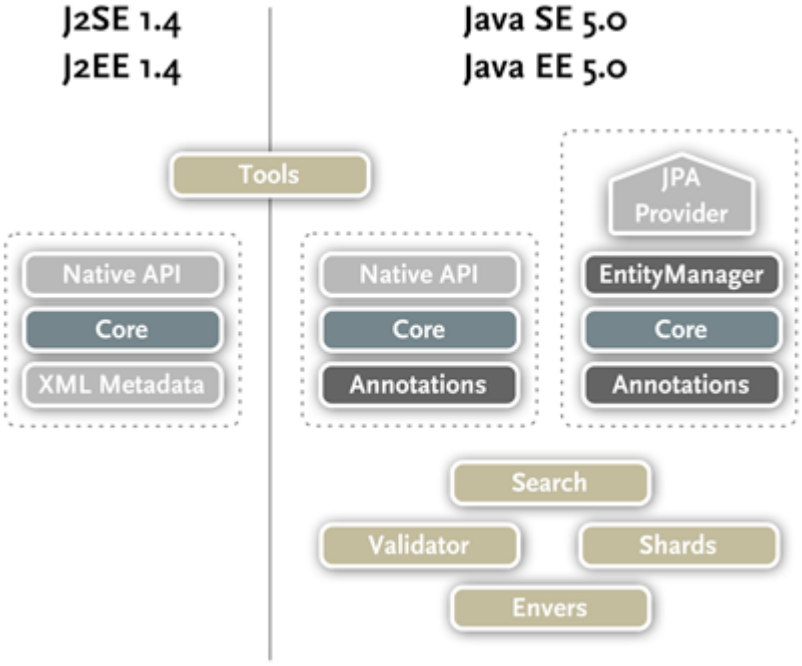


Figure 4 - Framework Hibernate

Spring 3.0: est un Framework de source libre introduit en 2003. Ce Framework offre la possibilité de développer plus facilement des applications java. Il repose sur trois concepts fondamentaux; principalement l'injection de dépendances et la programmation orientée aspect.

Le choix du Framework Spring est motivé par la volonté de développer une application en utilisant moins de couplage, donc facile à maintenir et facile à tester. Cette problématique est résolue par l'un des trois concepts de Spring : l'injection de dépendances.

JavaMail 1.4 : est une API java favorisant l'intégration de la gestion du courrier électronique dans une application. Cette API sera utilisée pour permettre aux utilisateurs de communiquer avec l'administrateur de l'application Paintball.

Velocity : à l'instar de Struts, Velocity est un projet de source libre crée par la fondation Apache. Ce projet contient plusieurs sous-projets destinés à simplifier la création de rapports sous divers formats (CSV, TXT, HTML…).

Le projet VelocityTools est celui qui nous intéresse, plus particulièrement la partie VelocityStruts du projet puisque celle-ci s'intègre facilement au Framework Strus.

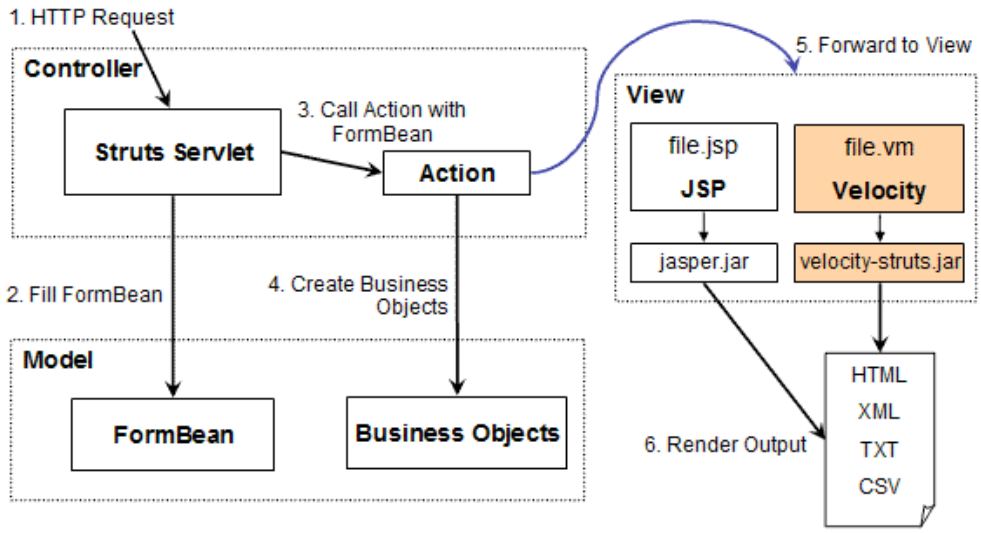


Figure 5 - Intégration de Velocity à Struts

JUnit 4.9: est un Framework de source libre de tests unitaires permettant entre autres de faire du développement orienté par les tests (TDD). Ce Framework sera utilisé pour rédiger des tests unitaires pour valider les exigences du projet.

Log4j 1.2: est un Framework de source libre de journalisation simple et flexible. Ce Framework sera utilisé pour tracer les activités de l'application web; cette approche permettra d'identifier rapidement et efficacement les anomalies et de les corriger notamment dans la phase de maintenance.

Technologies	**Description**
Java / J2EE5	Technologie de la compagnie Oracle/Sun
Struts 1.3	Framework MVC de source libre
Hibernate 3.2	Framework de source libre de persistance de données
Spring 3.0	Framework de source libre de développement simplifié d'applications.
JavaMail 1.4	API java de gestion du courrier électronique

Velocity	Framework de source libre de génération de rapports
JUnit 4.9	Framework de source libre de tests unitaires
Log4j 1.2	Framework de source libre de journalisation

Tableau 1 : Récapitulatif des technologies

1.2.2 Les outils

Eclipse : est un environnement de développement intégré. Cet outil est gratuit et offre la possibilité d'installer diverses extensions.

PowerDesign 12: est un outil de modélisation des données. Il est commercialisé par Sybase et cette société offre une version d'évaluation de 15 jours. Cette version peut être téléchargée sur leur site internet.
PowerDesign propose des outils de génération de script, de génération de données test, de « reverse engineering » réduisant considérablement le temps de modélisation.

EMS MySQL Manager 2007 : ce produit est commercialisé par la société EMS. La société se spécialise dans les solutions de gestion de bases de données. Il est possible de télécharger une version d'évaluation sur leur site internet. L'avantage de cet outil est qu'il est nettement plus convivial que l'interface qui est fourni avec le serveur MySQL.

Tomcat 5.5 : est un conteneur web libre pour les composants j2ee (servlet/jsp). Il est aussi conçu par la fondation apache et il est téléchargeable sur leur site web.

MySQL 5.0.45: est un serveur de base de données libre. Il est aussi performant que les autres serveurs payants sur le marché. Il est téléchargeable sur le site de MySQL.

Technologies	**Description**
Eclipse	Environnement de développement intégré
PowerDesign 12	Outil de modélisation des données

EMS MySQL Manager 2007	Solution de gestion de base de données MySQL
Tomcat 5.5	Serveur web libre
MySQL 5.0.45	Serveur de base de données libre

Tableau 2 : Récapitulatif des outils

1.3 Conclusion

Les technologies et les outils choisis sont des logiciels libres et utilisés par un grand nombre au sein de la communauté de développeurs. Ils nous permettront de réduire considérablement la durée et le coût de développement, mais aussi de limiter plus tard le coût de la maintenabilité et d'évolution de l'application.

Le chapitre suivant présentera un survol littéraire des systèmes robotiques en corrélation avec les technologies et les outils choisis. Ce survol littéraire nous offrira la possibilité d'avoir une vue globale sur ce qui se fait présentement dans le domaine de la robotique mobile; ce survol nous permettra de proposer une solution à la conception d'une application web de contrôle à distance du robot de Paintball.

CHAPITRE 2

REVUE LITTÉRAIRE

2.1 Introduction

L'objectif de cette revue littéraire est d'identifier ce qui se fait dans le domaine des systèmes robotiques mobiles. Il vise également à identifier les concepts réutilisables dans le contexte de ce projet ce qui nous évitera de réinventer la roue et de proposer une solution viable.

Selon la JIRA, les robots peuvent être classifiés comme suit [3]:

- Classe 1 - *dispositif de manutention manuelle* : Tout robot ayant plusieurs degrés de liberté commandés par un opérateur;
- Classe 2 - *robot à séquence fixe* : Tout robot effectuant des tâches rigides;
- Classe 3 - *robot à séquence variable* : identique à la classe précédente si ce n'est que les étapes peuvent être modifiées facilement;
- Classe 4 - *robot à lecture* : Tout robot ayant une capacité de répétition des tâches manuelles exécutées par un opérateur;
- Classe 5 - *robots à commande numérique* : tout robot incapable d'apprentissage et contrôlé par un opérateur. Ce type de robot reçoit des commandes et les exécute;
- Classe 6 - *robot intelligent* : Tout robot ayant la capacité de comprendre son environnement.

Le robot de Paintball est contrôlé à travers une interface web par un internaute; de ce fait, il appartient à la classe 5 des robots à commande numérique.

La mécanique du Paintball est composée d'une capsule noire reposant sur un support qui lui assure plusieurs degrés de liberté : une rotation horizontale de 180 degrés et une rotation verticale de 90 degrés.

Cette capsule contient les balles de Paintball. Le support constitue le cœur de la mécanique du robot, car il héberge le microcontrôleur. Finalement, trois pieds amovibles supportent le corps du robot. Le robot utilisé est représenté par la figure [2] ci-dessous.

Figure 6: Le robot de Paintball

Un robot est destiné à accomplir des tâches dans un environnement donné. Pour atteindre cet objectif, le robot est équipé d'un microcontrôleur.

« Un microcontrôleur est un microprocesseur muni d'interfaces spécialisées conçues pour connecter des capteurs ou des actionneurs » [4]. Ces actionneurs accomplissent des tâches tandis que les capteurs permettent de comprendre l'environnement, par conséquent la prise de décision. Il existe plusieurs familles de microcontrôleurs sur le marché; le choix du microcontrôleur est déterminé par les caractéristiques suivantes : la taille, la consommation, la facilité de programmation, le type et le nombre d'interfaces.

Le microcontrôleur choisi dans le cadre du projet Paintball est un AVR 32 EVK1100 appartenant à la série AT32UC3A des microcontrôleurs Amtel. Ce microcontrôleur, en plus d'être fourni, possède un environnement de développement complet et offre des fonctionnalités assez riches. Ces principales caractéristiques sont :

- des capteurs (température, lumière, potentiomètre);
- de nombreux connecteurs (JTAG, Nexus, USART, USB 2.0, TWI, SPI);
- un lecteur de cartes SD et MMC;
- un écran LCD bleu 4*20.

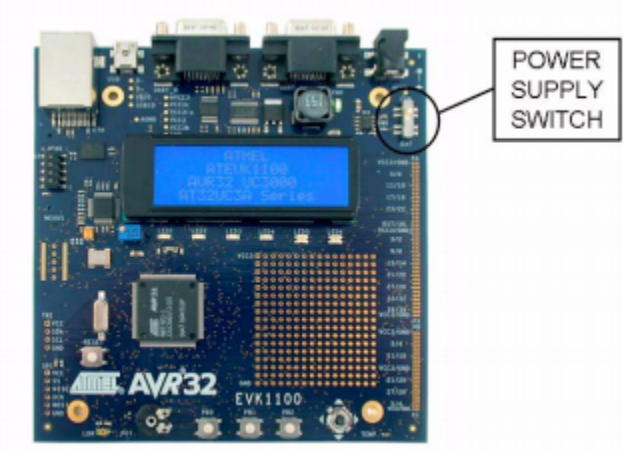


Figure 7: Microcontrôleur Amtel EVK1100 [5]

2.2 L'architecture

En 2005, le département de la défense américaine en quête de standardisation de ses systèmes robotiques a adopté une norme destinée à encadrer le développement de ses systèmes. Cette norme connue sous le nom de JAUS est divisée en deux parties : le domaine du modèle et l'architecture de référence. La deuxième partie est celle qui nous intéresse, car elle définit l'architecture et la communication du système robotique.

L'architecture définie par la norme découpe un système en sous-systèmes sous la forme d'une hiérarchie. Ainsi, un système est une composition de sous-systèmes. Un sous-système est composé de nœuds alors qu'un nœud est composé d'un ou de plusieurs composants. Ce découpage a l'avantage de favoriser l'interopérabilité du système.

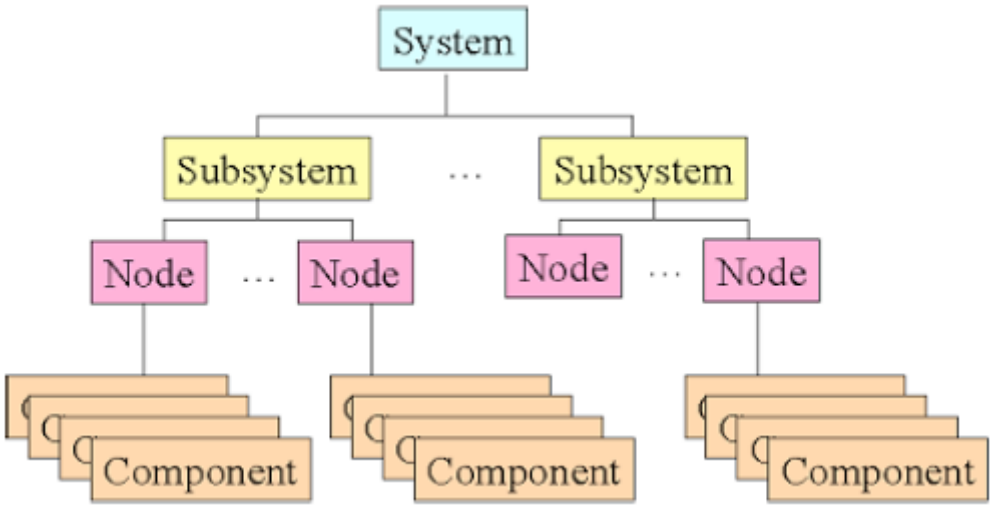


Figure 8: Architecture d'un système JAUS [6]

La communication définie par la norme est assurée par des communicateurs; il ne doit y avoir qu'un communicateur par sous-système qui se charge de transmettre les messages.

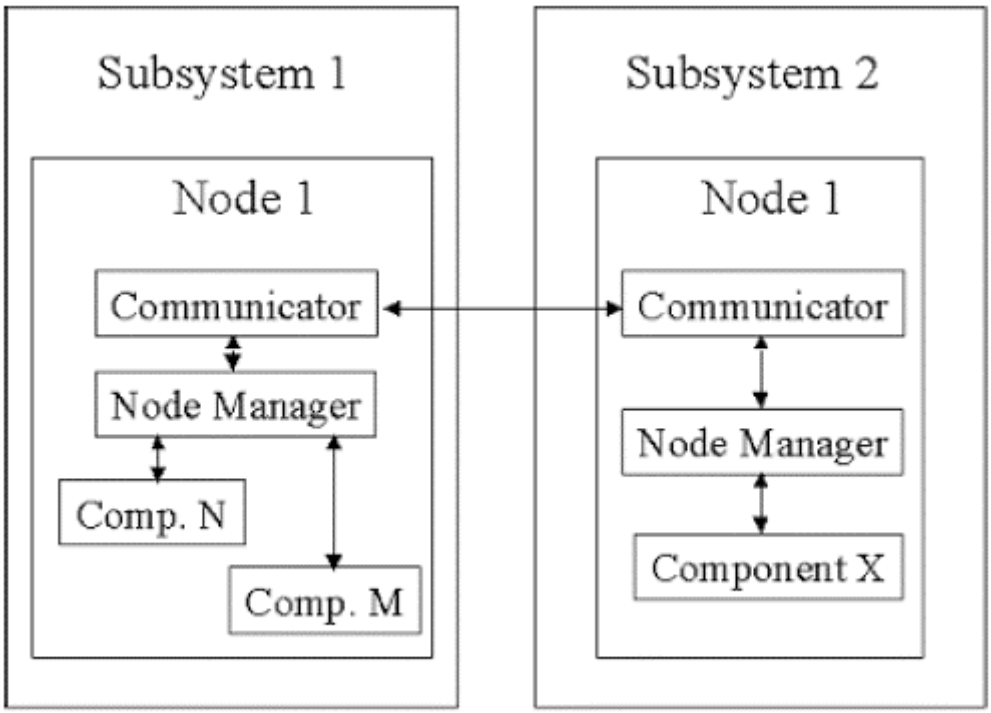


Figure 9: Communication dans un système JAUS [6]

JAUS spécifie également le format des messages pour assurer une conformité à la norme. Celle-ci définit également un ensemble de messages utilisés pour communiquer entre les composants.

La norme JAUS est encadrée par un groupe de travail nommé JWG et il existe trois niveaux de conformité de la norme. JAUS, en découpant un système en sous-systèmes indépendants et en standardisant les messages, répond ainsi à l'un des critères de qualité des systèmes robotiques, à savoir l'interopérabilité.

Ce modèle d'architecture sera utilisé dans le contexte de ce projet. Le nombre de sous-systèmes identifiés ainsi que le niveau de conformité sera développé dans la section conception. [6]

La norme JAUS, comme toutes les normes, dit quoi faire, mais pas comment le faire. Après avoir répondu au quoi avec la norme JAUS, nous allons nous intéresser à présent.

2.3 Conception

Dans un article publié en 2000 [7], Dareil et Keith répondent au comment. Ils expliquent en détail comment contrôler un robot à travers le réseau Internet en utilisant la technologie Java/J2EE.

Le système présenté comprend les composants suivants :

- Un formulaire web destiné à la saisie des commandes envoyées au robot;
- Un servlet Java qui traite les commandes du formulaire en les acheminant au robot. Pour cela, il utilise une API Java de communication de port sériel (JavaComm). Cette API offre la possibilité d'accéder aux ports sériels et d'établir une communication sérielle.

Dans ce système, nous supposons que le robot est connecté au port sériel du PC qui exécute le servlet.

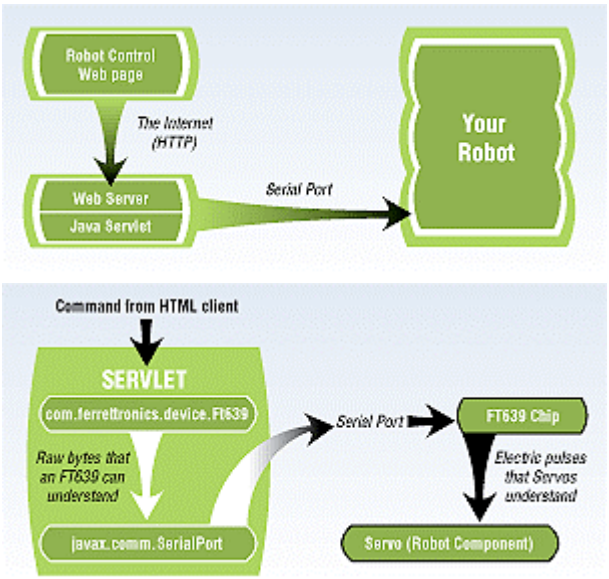


Figure 10: Conception d'un contrôleur de robot par internet [7]

Cette approche s'adapte à notre contexte puisqu'elle utilise la technologie web de Java pour contrôler un robot par internet. Elle sera couplée à la norme JAUS et servira de références lors de l'implémentation.

Le modèle précédent suppose que le PC soit connecté au microcontrôleur par le port sériel. Il se trouve que les dernières générations de PC ne sont pas équipées de port sériel, mais de port USB. Pour combler cette lacune, Ferat et Pushkin dans leur livre [8], nous suggère de recourir à un convertisseur USB / Série pour la communication entre le PC et le microcontrôleur. Un pilote est fourni avec ce convertisseur que l'on doit installer sur le PC : cette installation permet au PC d'utiliser le standard RS232.

Figure 11: Convertisseur USB / Série

Dans le développement d'une application de contrôle des systèmes robotiques, la principale contrainte demeure la réponse en temps réel. Dans leur livre [2], Lewis, Darren et Chaouki utilisent le modèle ci-dessous de communication concurrente qui exécute plusieurs « threads » sur un seul processeur. Dans ce modèle à temps réel, les tâches sont exécutées avec différentes priorités et les tâches critiques ont une haute priorité.

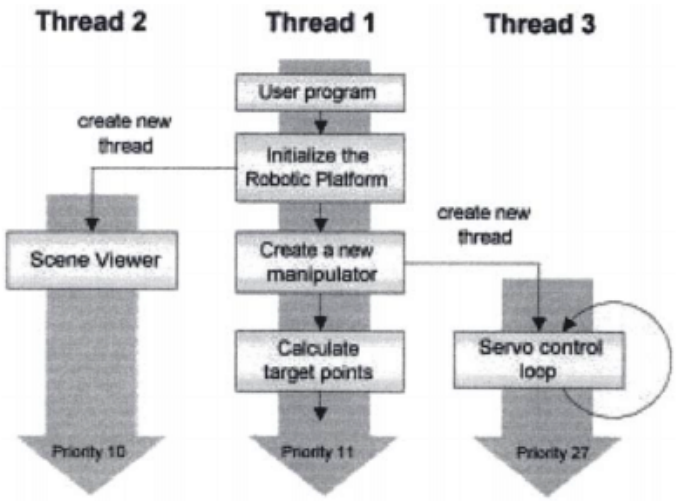


Figure 12: Création des threads [2]

Cette approche pourrait être intégrée au modèle de Dareil et Keith pour rendre celui-ci concurrent c'est-à-dire un modèle d'exécution en temps réel.

Pour l'interface utilisateur de contrôle d'un robot mobile, Thomas Braunl présente une interface simple réalisée dans le contexte du projet EyeCon [9]. Cette interface offre à l'utilisateur la possibilité de modifier la configuration comme le taux de transfert et le port sériel sur lequel est connecté le robot. Cette approche pourrait être intégrée au projet de Paintball afin d'assurer une efficacité et une flexibilité à l'application : deux critères de qualité en Génie logiciel.

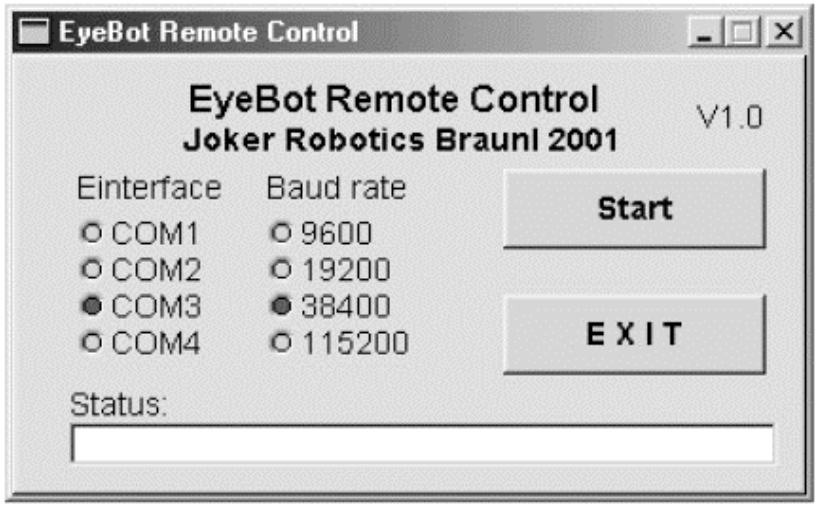


Figure 13: Interface de contrôle [9]

CHAPITRE 3

PROPOSITION DE SOLUTION

Le projet évoluera possiblement en réalisant une seconde phase. Le modèle incrémental est ainsi mieux adapté à ce projet.

Dans la mesure où il n'y a qu'une seule personne qui est impliqué dans le développement du « front-end », il n'est pas utile de ce fait de disposer d'un outil de suivi du processus de développement.

3.1 Les cas d'utilisation

Les cas d'utilisation visent à identifier les acteurs principaux et secondaires. Les acteurs principaux seront amenés à interagir en utilisant le système externe, en l'occurrence le Paintball. Les acteurs secondaires sont en général des systèmes externes. Ces cas d'utilisation visent aussi à identifier les fonctions qui sont accomplies par ces différents acteurs.

Deux acteurs principaux et un acteur secondaire sont ici identifiés.

Les deux principaux acteurs sont :

- L'internaute : qui désigne toute personne visitant l'application web de Paintball;
- L'utilisateur : qui désigne un internaute autorisé à utiliser le robot de Paintball par l'application web.

L'acteur secondaire :

- Le robot : qui reçoit les commandes de l'application web et qui les exécutent en temps réel.

Les acteurs ayant été identifiés, il est nécessaire maintenant d'identifier les rôles ou les fonctionnalités de chacun des acteurs et de les représenter à l'aide d'un diagramme de cas d'utilisation :

Les principales fonctionnalités identifiées pour chacun des acteurs :

- L'internaute :
  - s'inscrit en vue de devenir un utilisateur

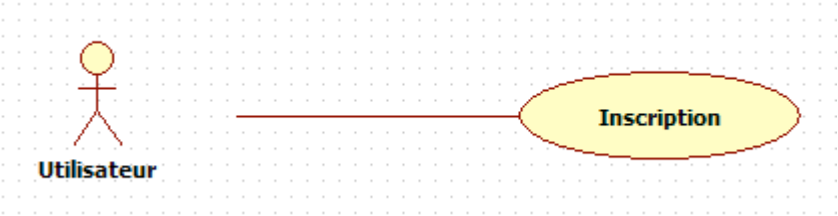


Figure 14: Cas d'utilisation de l'internaute

- L'utilisateur :
  - est authentifié avant d'accéder aux ressources;
  - visualise l'environnement du robot (caméra embarquée sur le robot);
  - détecte la disponibilité du robot;
  - prend le contrôle du robot;
  - contrôle le robot (avec des déplacements gauche-droite, bas-haut);
  - déclenche un tir.

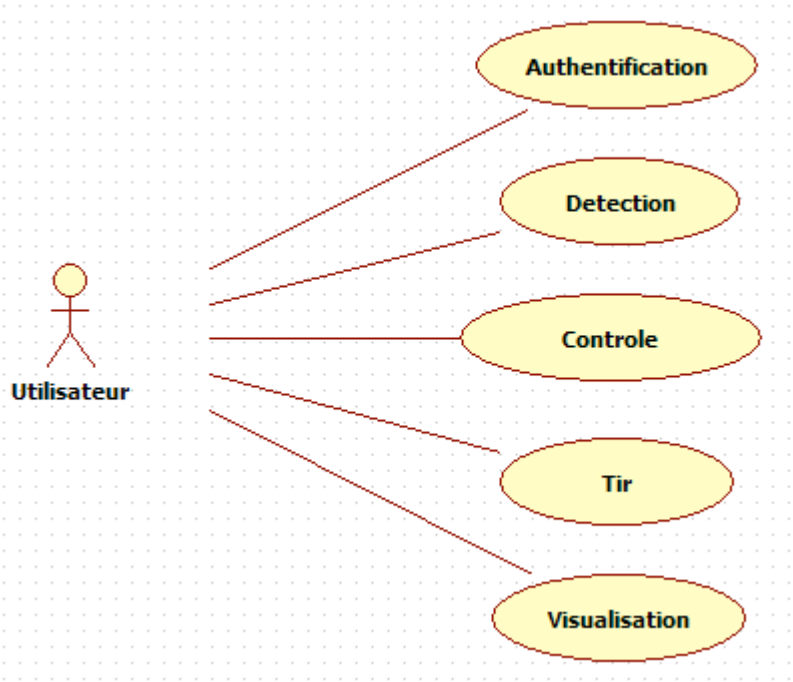


Figure 15: Cas d'utilisation de l'utilisateur

- Le robot: La conception et l'implémentation de ses fonctionnalités sont prises en charge par une autre équipe dans le contexte d'un projet distinct, à savoir l'équipe du « back-end ». Ces fonctionnalités sont mentionnées à titre informel :
    - réinitialisation;
    - positionnement en fonction des coordonnées;
    - tire une balle;
    - tire de trois balles.

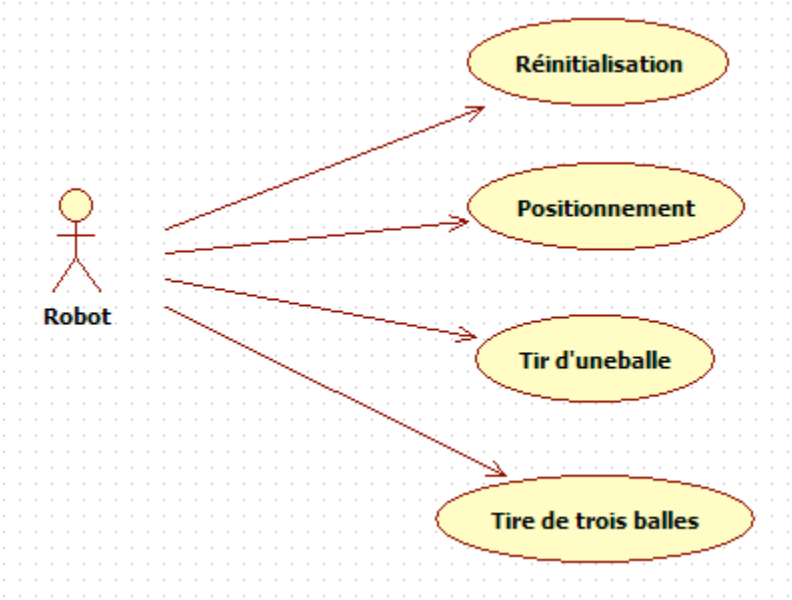


Figure 16: Cas d'utilisation du robot

L'internationalisation sera prise en charge par l'application web. L'utilisateur aura la possibilité de naviguer dans les langues officielles : le français et l'anglais.

3.2 Protocole de communication

La deuxième rencontre avec l'équipe du back-end a permis de s'entendre sur un protocole de communication simple entre le microcontrôleur et l'application web.

Il a été décidé que l'application web de contrôle du robot allait envoyer des commandes sous la forme de trame de 3 octets.

Après chaque transmission de 3 octets, des confirmations/code d'erreur sous la forme de trame de 3 octets seront retournées à l'application web. Ces trames seront échangées par UART ou par Ethernet.

La position du robot est définie par des coordonnées (x, y).Les valeurs de x et y vont de 0 a 100%. Les coordonnées (0,0) et (100,100) seront respectivement de bas à gauche et de haut à droite.

Liste des commandes que le PC peut envoyer

0x41 0x410x41 ------- Vérifie si le robot est opérationnel
0x42 0x420x42 ------- Lance une balle de Paintball
0x43 0x430x43 ------- Lance une rafale de 3 balles Paintball à vitesse maximale
0x44 0xX 0xY ------- Positionne le robot aux coordonnées X et Y

Liste des codes que le PC recevra après chaque commande envoyée.

0x61 0x610x61 ------- Robot opérationnel
0x62 0x620x62 ------- Balle tirée avec succès
0x63 0x630x63 ------- Rafale de 3 balles tirée avec succès
0x64 0x640x64 ------- Robot positionné avec succès
0x65 0x650x65 ------- Robot a un problème
0x66 0x660x66 ------- Trame reçue incohérente, demande un ré-transfert

Après l'authentification de l'internaute sur l'application web de contrôle du robot, une commande est tout de suite envoyée au robot pour vérifier si le robot est opérationnel.

3.3 Conception

L'architecture recommandée dans le développement des applications web est : l'architecture MVC. Cette architecture permet de diviser l'application web en trois différentes couches (Modèle, Vue, Contrôleur). Le modèle représente la couche de persistance et la couche d'affaires. Cette séparation a pour effet d'accroître la maintenabilité de l'application. Nous pouvons alors changer la couche présentation sans nuire à la couche d'affaires et vice-versa.

Le Framework Struts implémente l'architecture MVC : il sera intégré à l'application.

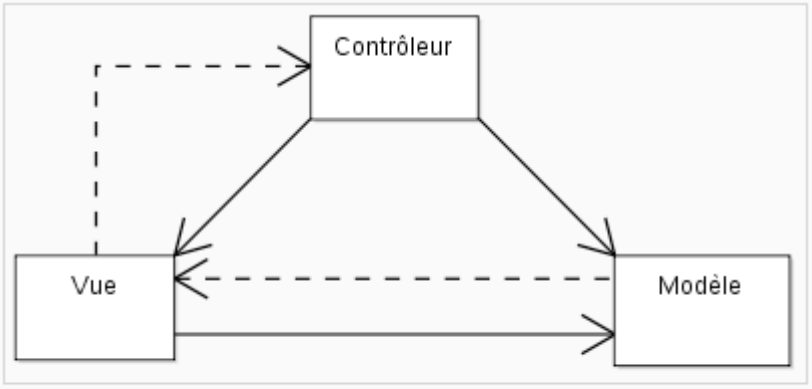


Figure 17: Architecture MVC [10]

La couche affaires :

JAUS est le modèle architectural de référence dans la conception des robots mobiles. Ce modèle est une référence au DoD. Puisque le robot de Paintball appartient à la catégorie des robots mobiles, ce modèle servira de référence dans le cadre de ce projet. Le modèle qui sera implémenté aura une conformité de niveau 1.

La machine qui hébergera l'application web et le microcontrôleur représente alors des sous-systèmes. Chaque sous-système disposera d'un communicateur unique permettant de communiquer l'un avec l'autre. Conceptuellement, une classe « Thread » assurera le rôle de communicateur, ce « thread » démarrera lors de mise en service de l'application web.

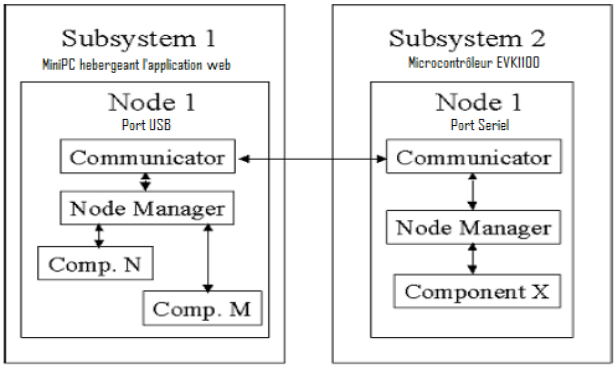


Figure 18: JAUS appliqué au Paintball

Deux patrons de conception seront utilisés pour assurer l'interaction entre l'application et le communicateur. Ce sont le patron « State » et le patron « Command ».

Le patron « State » :

Le Paintball sera en tout temps dans l'un des deux états, soit en cours d'exécution (ExecutingState), soit en attente d'une commande (ReadyState). Le patron de conception « State » offrira de matérialiser ces deux états. Quand le Paintball sera en cours d'exécution, aucune réception de commande ne sera permise.

Le patron « Command » :

Quatre commandes seront disponibles dans l'application web. Le patron de conception « Command » permet de séparer l'action de l'invocateur. Chaque commande du Paintball sera conçue en invocateur; nous disposerons alors de quatre (4) classes invocatrices. Les actions seront réalisées par une classe.

Pour s'assurer de la configurabilité de l'application et aussi de la testabilité, nous nous servirons de l'injection de dépendance de Spring. Ainsi, nous disposerons d'un fichier XML qui servira de fichier de configuration à l'application.

Les informations comme le port sériel sur lequel le Paintball est connecté, le serveur smtp utilisé pour l'envoi de courriel et le courriel de l'administrateur de l'application seront contenus dans ce fichier XML.

La couche persistance :

Nous nous servirons du Framework Hibernate pour concevoir cette couche. Dans ce framework, chaque table de la base de données sera représentée par une classe dite entité et chaque entité disposera de sa propre classe d'accès aux données permettant des opérations CRUD.

Les quatre (4) opérations CRUD étant communes à toutes les entités, ces opérations seront réalisées par une classe générique d'accès aux données. Toutes les autres classes d'accès aux données hériteront de cette classe générique.

La couche présentation :

La page web sera subdivisée en quatre zones, deux (2) zones statiques et deux (2) zones dynamiques. Voici la description de ces zones :

- L'entête de page : qui est une page statique qui sera incluse dans chaque page;
- Le pied de page : à l'instar de l'entête est une page statique qui sera incluse dans chaque page;
- Le menu de gauche : qui est une page quasi statique;
- Le contenu principal : qui représente le contenu de chaque page.

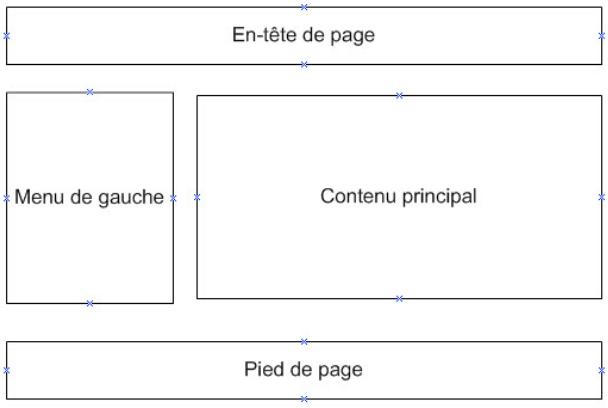


Figure 19 - Exemple d'une page

Cette subdivision permet d'assurer une bonne maintenabilité de la vue. Pour modifier l'entête des pages nous n'aurons qu'une seule page à modifier et les changements se répercuteront sur toutes les pages.

3.4 L'implémentation

La méthodologie TDD et l'approche de développement « Bottom Up » sont les approches utilisées lors de l'implémentation. La méthodologie TDD recommande l'implémentation à priori des tests unitaires, ce sont les tests qui dirigent le développement.

Ainsi, les tests unitaires ont été les premiers à être développés. Ces tests unitaires permettent de vérifier le bon fonctionnent de la couche affaires et des opérations CRUD sur les entités.

Les trois classes ci-dessous sont des tests unitaires des classes d'accès aux données des entités UserDetails, UserDao et HistoryLogDao de la couche de persistance.

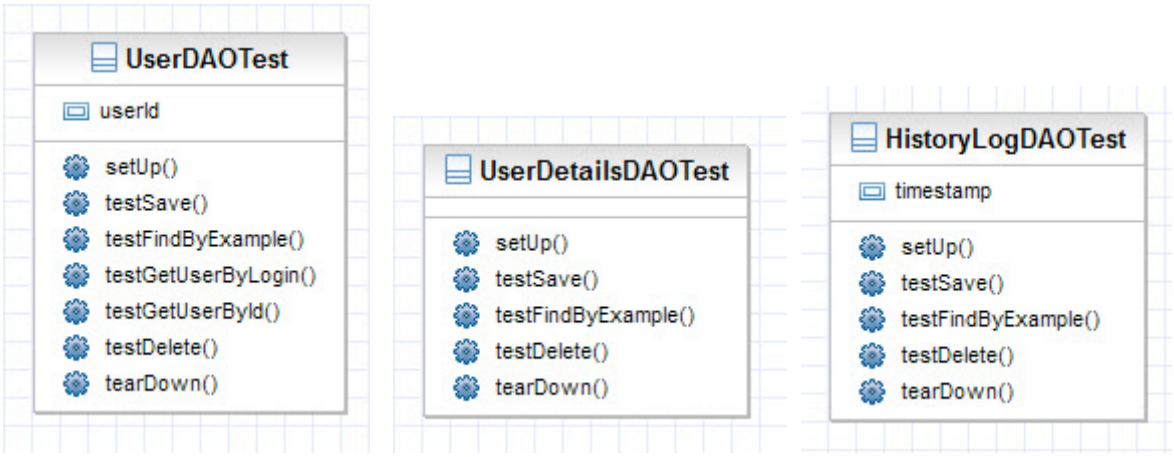


Figure 20: Tests unitaires des DAO de la couche de persistance

Les classes ci-dessous représentent les tests unitaires de la couche affaires de communication et de configuration. La classe PaintballConfigurationTest vérifie que l'application est bien configurée, la classe MicrocontrollerTest vérifie le bon fonctionnement de la classe de

communication enfin la classe SendMailTest vérifie le bon fonctionnement de la classe d'envoi de courriel.

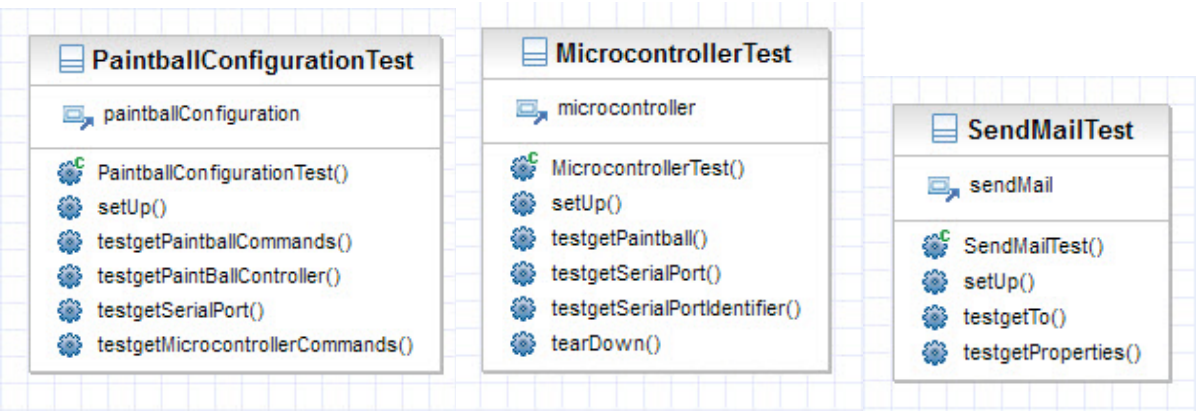


Figure 21: Tests unitaires de la couche de communication

L'approche TDD assure l'adéquation entre l'implémentation et les spécifications, ce qui réduit considérablement les erreurs, donc l'effort de développement. La figure ci-dessous montre le résultat d'exécution des vingt (20) tests unitaires.

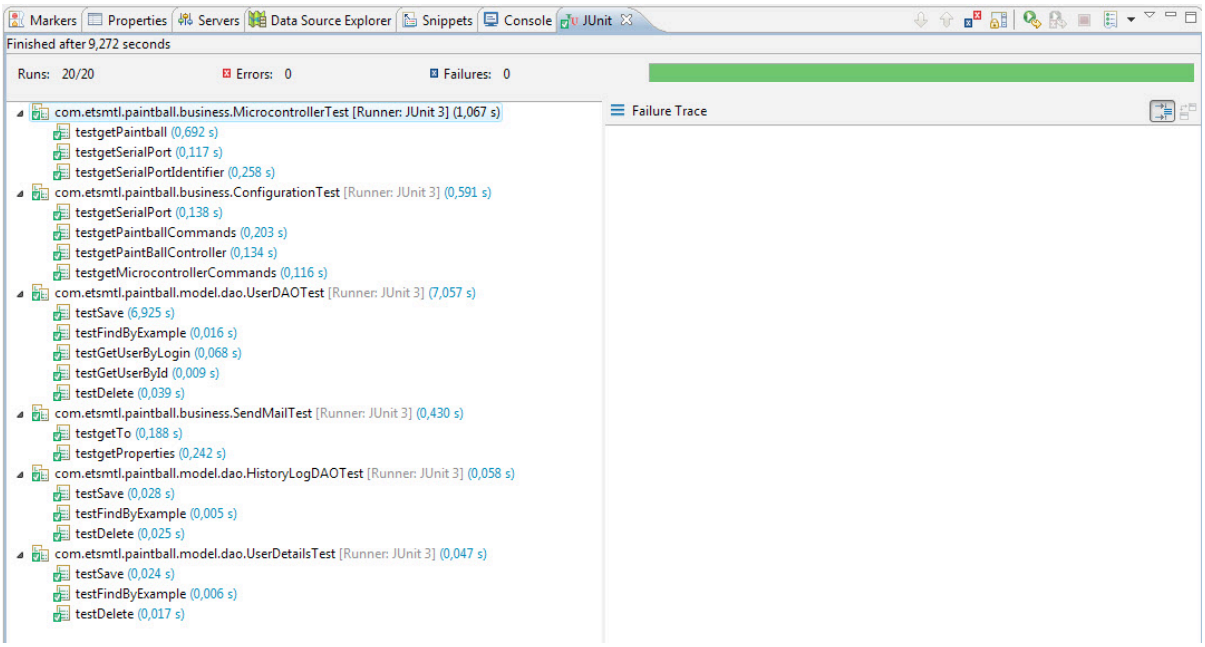


Figure 22 - Résultats d'exécution des vingt (20) tests unitaires

Dans l'approche « Bottom-Up », le développement se fait du bas vers le haut. On développe les couches de bas niveau (couches de persistance et de communication) en allant vers le sommet (la couche de présentation). Cette approche comparée à l'approche Top-Down est plus adaptée aux systèmes robotiques. Les tests unitaires présentés précédemment doivent passer au vert à la suite de l'implémentation des couches de bas de niveau.

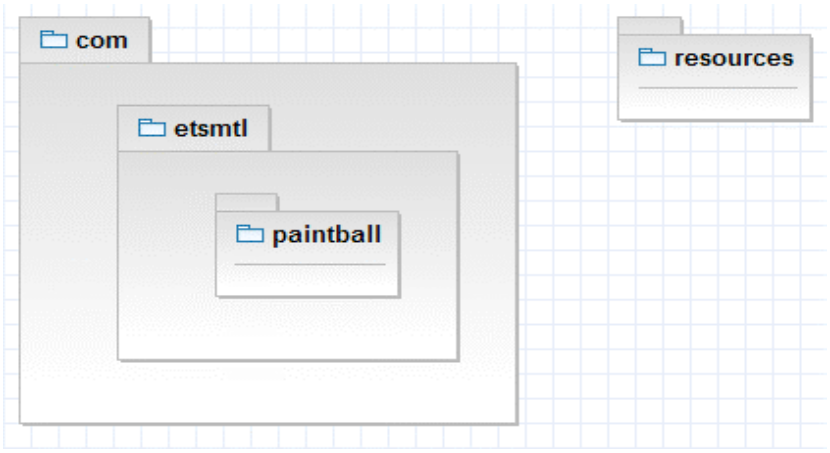


Figure 23: Diagramme de paquetage du projet

La classe GenericDao est la classe générique qui implémente les opérations standards de persistance (CRUD). Toutes les classes d'opérations concernant les entités héritent de cette classe générique. À l'image de GenericDao, la classe BasicEntity représente une entité générique; toutes les entités spécifiques héritent d'elle.

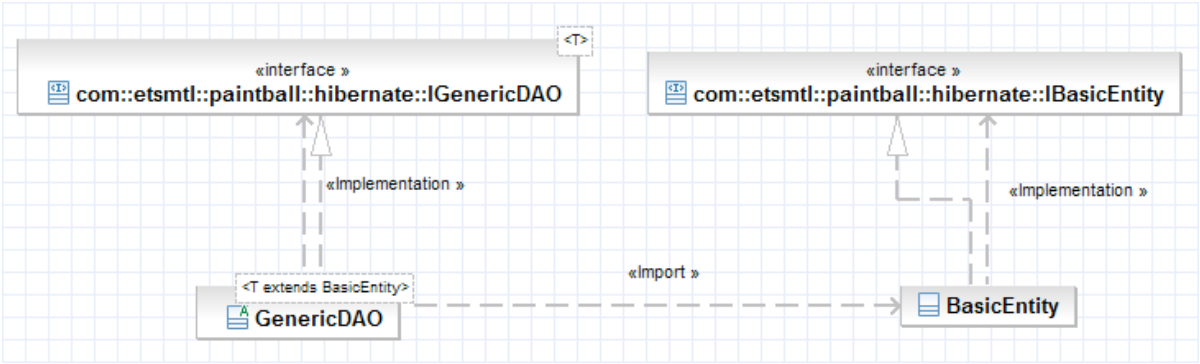


Figure 24: Diagramme de classe de la couche de persistance générique

Les classes xxxDao implémentent les opérations standards de persistance (CRUD) spécifiques des différentes entités.

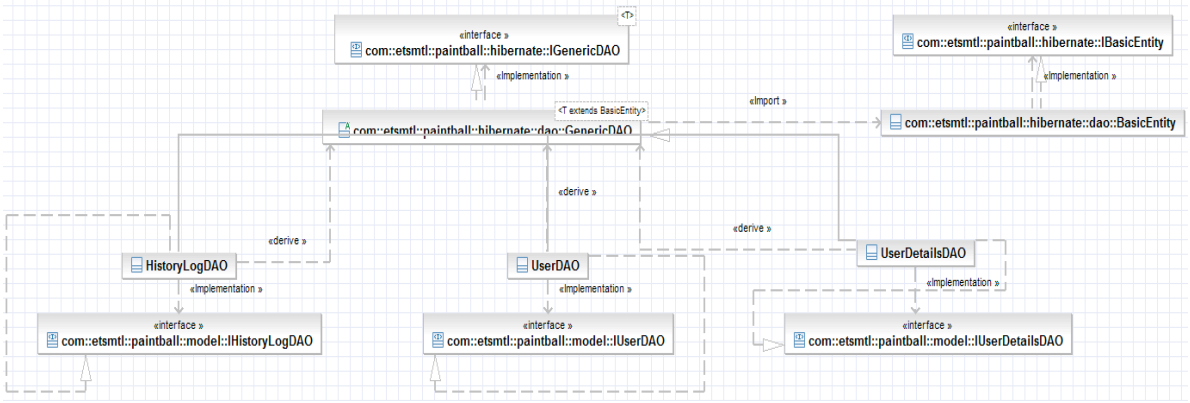


Figure 25: Diagramme de classe de la couche de persistance spécifique

La classe PaintballConfiguration sert à la configuration de l'application (les commandes du Paintball et le port sériel utilisé). Elle implémente le patron de conception Singleton. Nous nous assurons alors de l'unicité de l'instance de la classe.

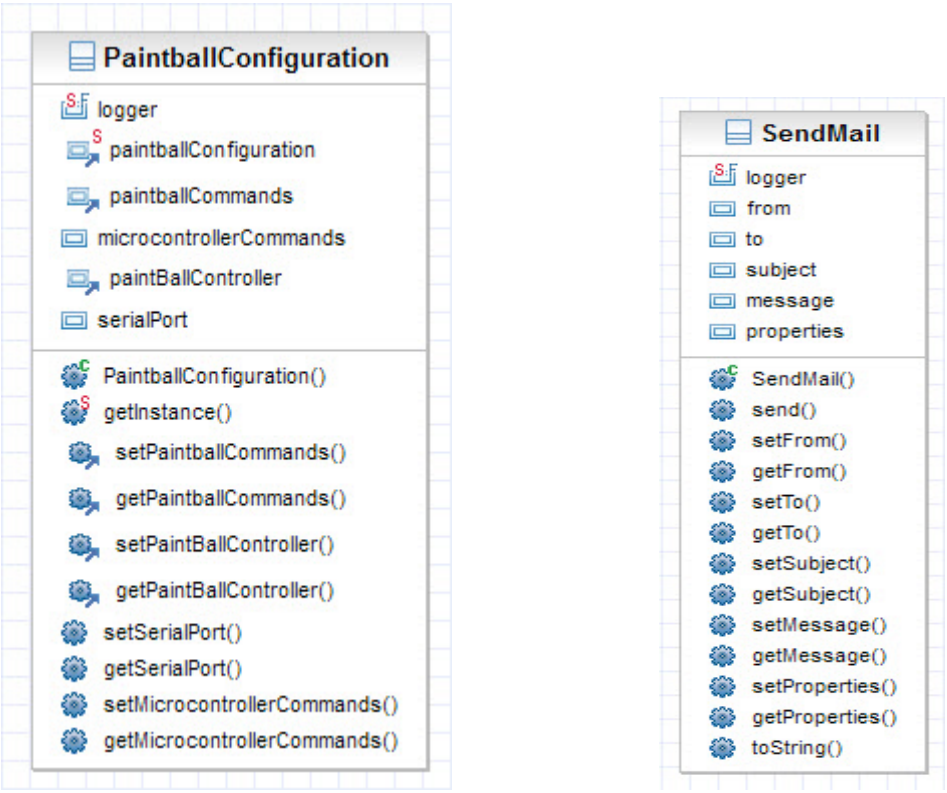


Figure 26 - Les classes PaintballConfiguration et SendMail

La classe SendMail, quant à elle, sert à l'envoi de courrier électronique. Cette fonctionnalité est intégrée à l'application web de Paintball offrant ainsi aux internautes la possibilité de contacter, par courriel, l'administrateur de l'application.

La classe Microcontroller, ci-dessous, implémente les interfaces Runnable et SerialPortEventListener ce qui permet de l'exécuter parallèlement à l'application et d'être notifié pour traiter les réponses du microcontrôleur. Dans le modèle architectural de JAUS, cette classe joue le rôle de communicateur. Elle s'occupe de la communication avec le système externe (le microcontrôleur). Elle se sert de la classe SerialPortConnection pour établir la connexion et envoyer des commandes au microcontrôleur.

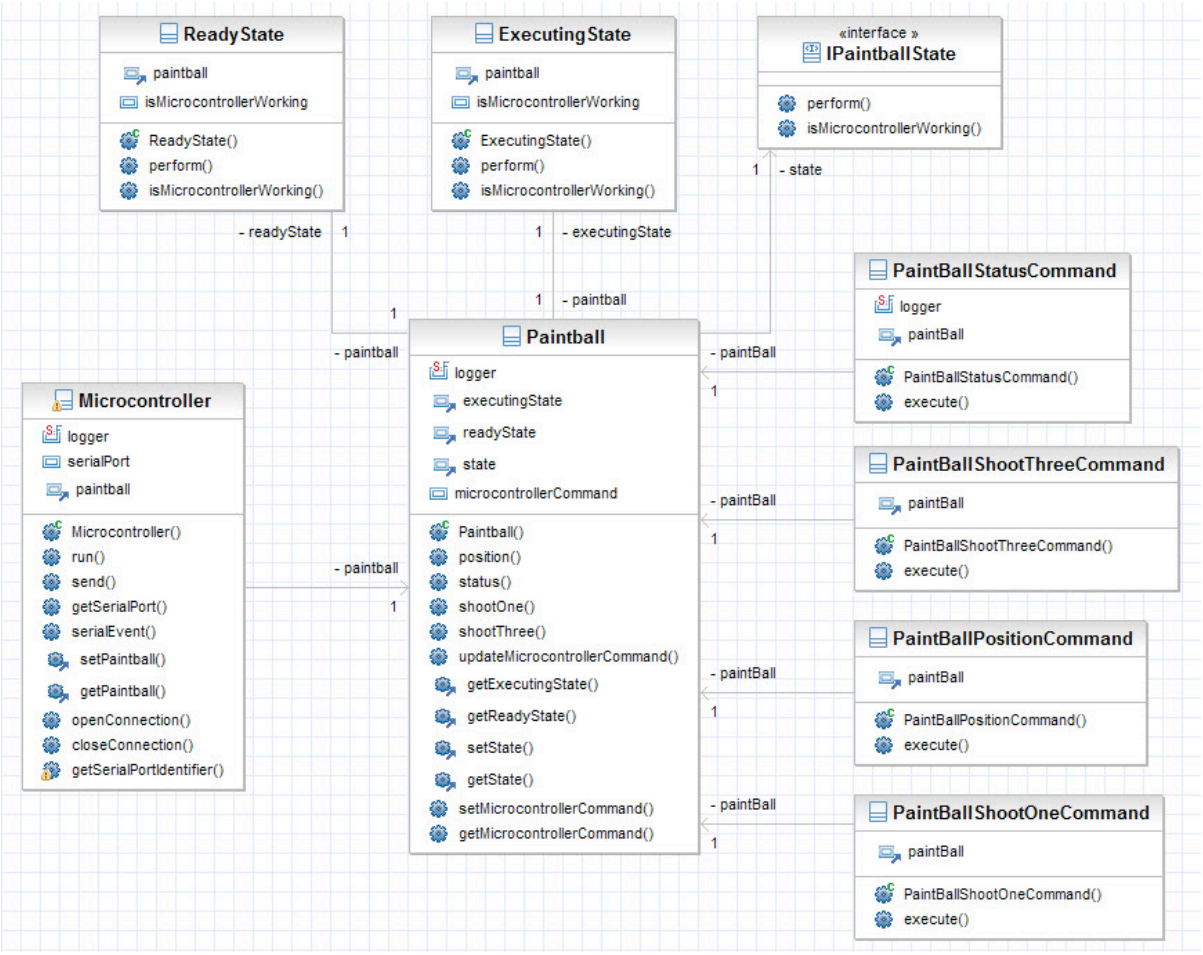
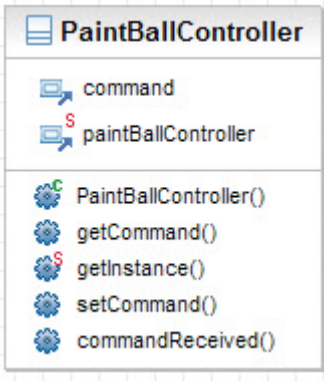


Figure 27: Diagramme de classe de la couche de communication

Les classes ReadyState et ExecutingState représentent respectivement les deux états du Paintball en attente d'une commande et en cours d'exécution d'une commande. Elles implémentent l'interface IPaintballSate. Le patron de conception « State » a été utilisé au cours de cette implémentation.

Les quatre (4) classes PaintballXxxCommand, la classe PaintballController et la classe Paintball représentent les commandes de contrôle de l'application de Paintball.
Le patron de conception « Command » a été utilisé au cours de cette implémentation; de ce fait les classes PaintballXxxCommand représentent les commandes, la classe Paintball représente le récepteur en implémentant les quatre (4) actions et la classe PaintballController représente l'invocateur.



3.5 Conclusion

Dans ce chapitre, la proposition de solution repose sur les extrants des deux chapitres précédents, sur les choix technologiques et la revue littéraire.

La solution représente en soi un condensé du cycle de développement du logiciel. Une analyse a été réalisée à partir des cas d'utilisation, permettant ainsi d'identifier les besoins des utilisateurs. Ensuite, une conception a été présentée en s'inspirant de modèles. Enfin une implémentation à été réalisée à partir de diagrammes des classes. L'approche de développement et la méthodologie de développement utilisées ont également été présentées.

Le chapitre suivant présentera le résultat de l'expérimentation du prototype conçu.

CHAPITRE 4

EXPERIMENTATION ET CONCLUSION

4.1 Introduction

L'expérimentation du prototype a pour objectif de valider les cas d'utilisation établis au chapitre 3 en ce qui a trait à la proposition de solutions qui seront détaillés en annexe. La mécanique du robot n'étant pas encore au point, l'expérience a été réalisée sur le microcontrôleur qui sera embarqué par la suite sur le robot.

Figure 28 - Page d'accueil de l'application web

4.2 L'internaute

Le contrôle du robot ne devrait être accessible qu'aux utilisateurs à la suite d'une authentification. Il fallait permettre aux utilisateurs de s'inscrire à travers une page web. Lors de l'inscription, les internautes choisissent un nom d'utilisateur et un mot de passe utilisés à l'authentification.

La figure ci-dessous montre une capture d'écran de la page d'inscription, les informations saisies par l'utilisateur sont validées avant la création de l'internaute dans la base de données. On valide l'adresse courriel de l'utilisateur, le nom d'utilisateur choisi, ainsi que tous les autres champs qui ont été remplis.

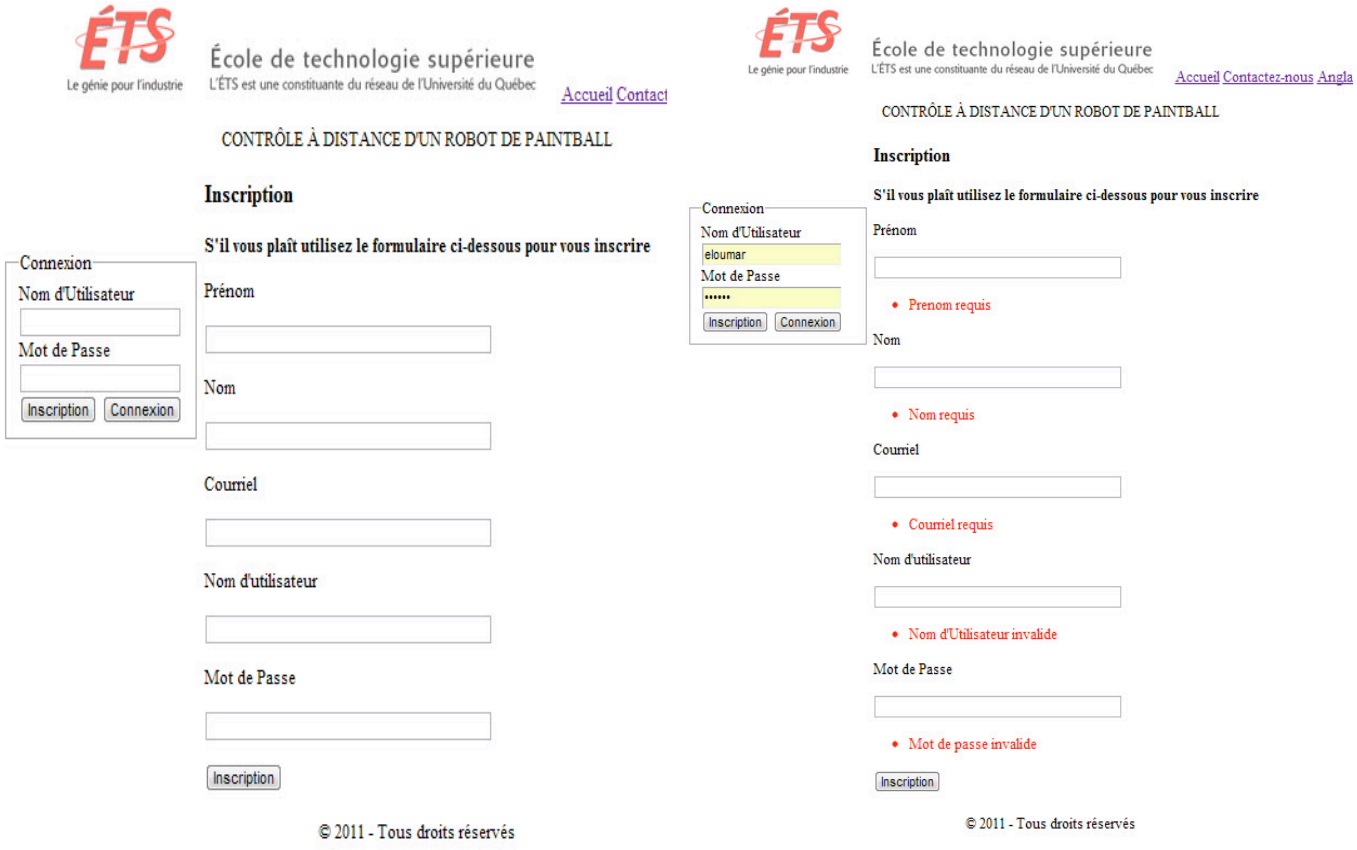


Figure 29 - Page d'inscription et exemple de validation des données

4.3 L'utilisateur

Un utilisateur de l'application est un internaute possédant les droits d'accès à l'application. Pour son authentification, l'utilisateur saisit son nom d'utilisateur et son mot de passe précédemment créé et clique ensuite sur le bouton « connexion » disposé en haut et à gauche.

Figure 30 - Exemple type de validation à l'authentification

Lorsque l'authentification de l'utilisateur est un succès, l'application envoie une commande de validation du statut du robot. Si le robot est accessible et contrôlable, l'application affiche, en bas à gauche, une couleur indiquant le statut du robot (rouge pour indisponible – vert pour disponible). L'utilisateur peut se déconnecter à tout moment en cliquant sur le bouton « déconnexion ».

À la suite d'une authentification réussie, l'application met à la disposition de l'utilisateur les différentes commandes de contrôle du robot. Ainsi, l'utilisateur peut cliquer sur les boutons : « tirer une balle » pour tirer une balle, « tirer trois balles » pour tirer une rafale de trois balles, « repositionner » en entrant les coordonnées pour repositionner le robot. À tout moment, l'utilisateur peut cliquer sur le bouton « vérifier le statut » pour mettre à jour le statut du robot.

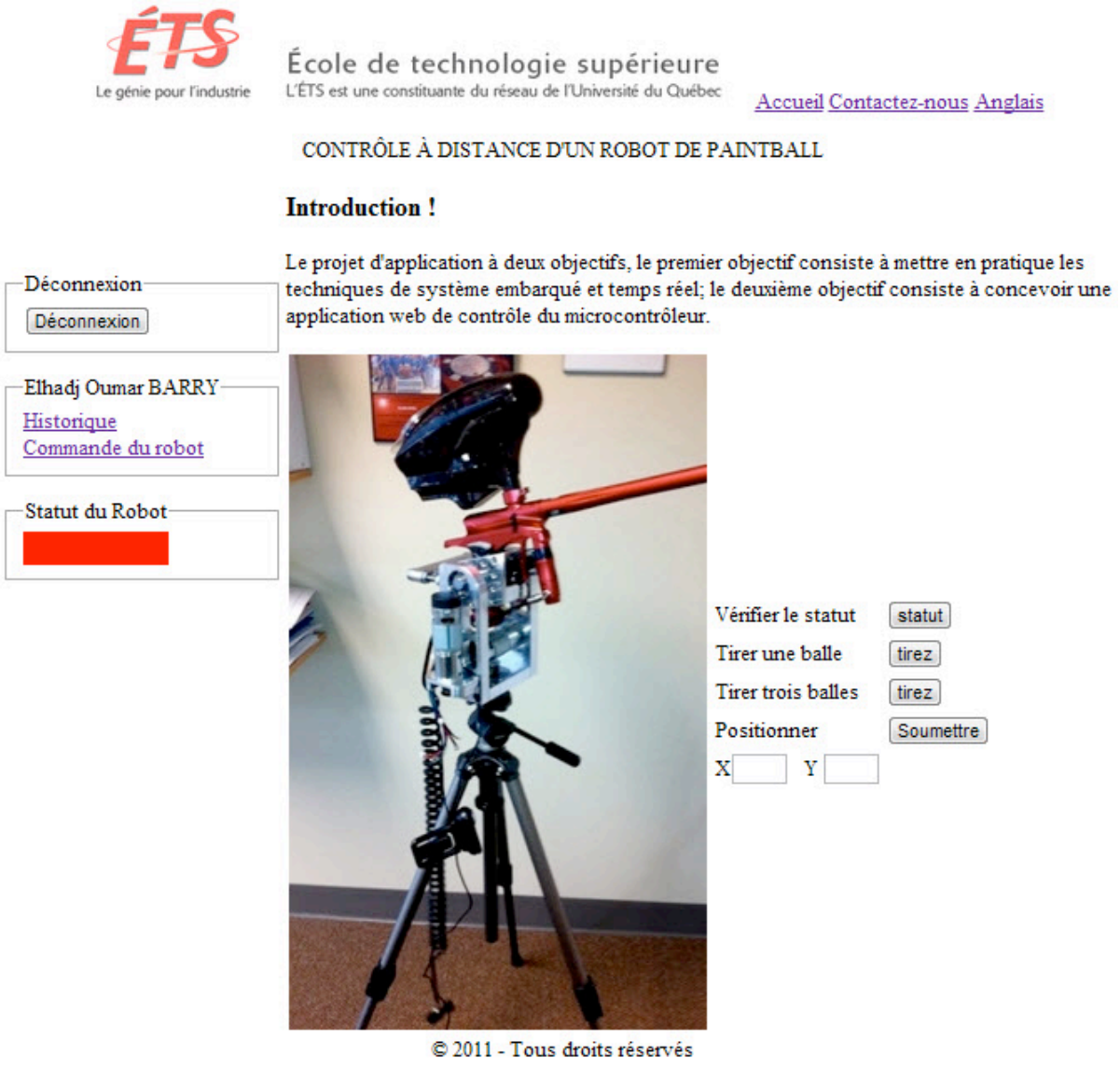


Figure 31 – Contrôle du robot

L'internationalisation est prise en charge par l'application. Un internaute et un utilisateur peuvent alors à tout moment changer la présentation dans l'une des deux langues : le français ou l'anglais.

En outre, ils ont la possibilité de contacter, par courriel, l'administrateur de l'application par un formulaire « contactez-nous ».

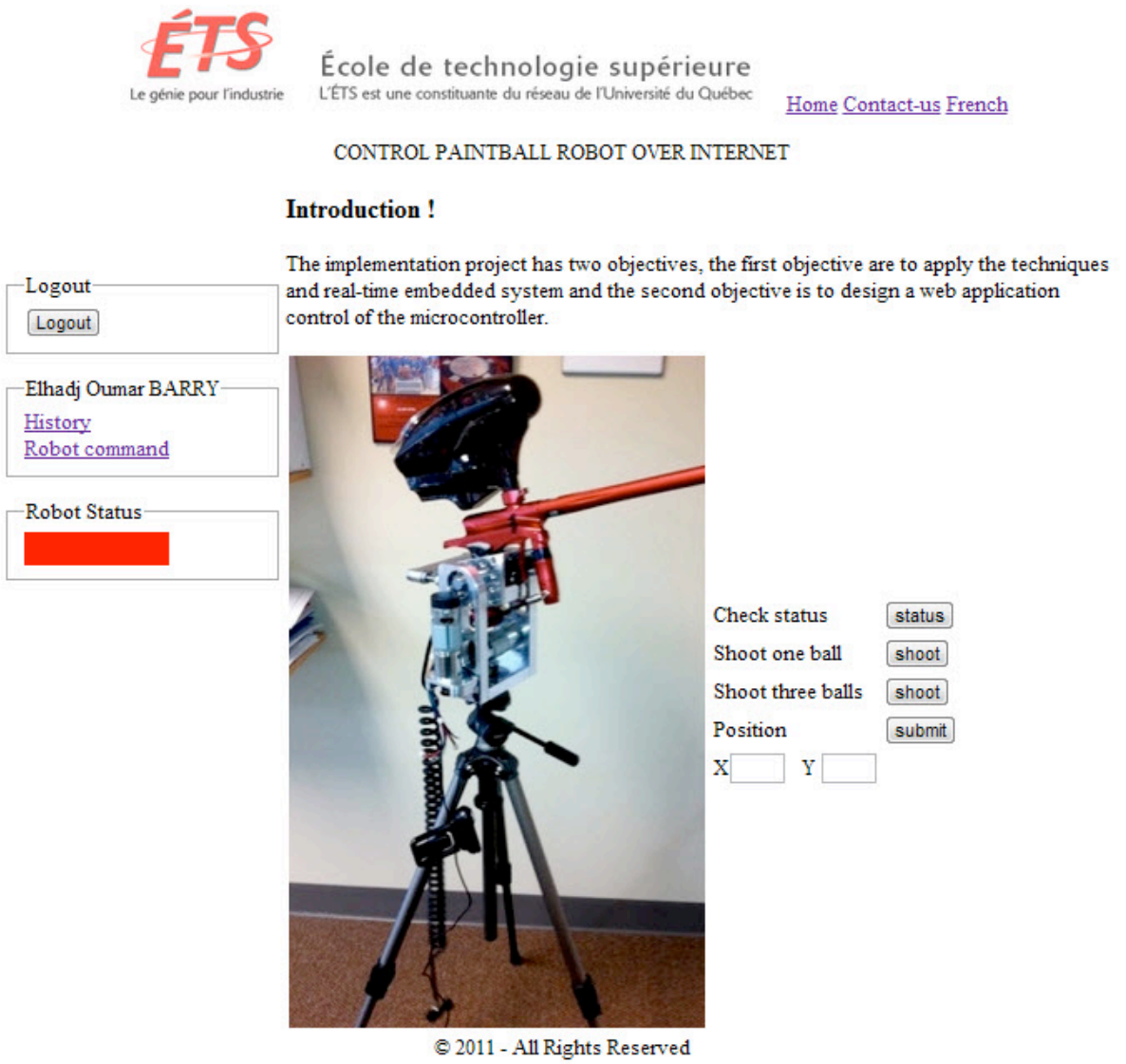


Figure 32 – Internationalisation

4.4 Conclusion

Cette application est un prototype qui couvre les cas d'utilisation du contrôle à distance d'un robot de Paintball. Dans le but d'améliorer la convivialité et la simplicité d'utilisation, elle a été soumise à quelques utilisateurs; les données recueillies ont permis de réorganiser la présentation.

Dans une deuxième phase de développement, on pourrait intégrer la visualisation à l'application à la place de l'image statique du robot. On pourrait aussi améliorer le processus d'inscription des internautes en intégrant un processus d'acceptation de l'inscription par l'administrateur de l'application. Le bouton de vérification du statut du robot pourrait être remplacé par un processus autogéré à travers duquel l'application envoie une commande de vérification à toutes les t secondes (t serait définie dans le fichier de configuration).

ANNEXE I

CAS D'UTILISATION DÉTAILLÉE

Cas d'utilisation 1 : s'inscrire en vue de devenir un utilisateur

Nom du cas d'utilisation	s'inscrire en vue de devenir un utilisateur
Code	Cas_1
Acteur principal	L'internaute
Pré-condition	L'application web de Paintball est déjà installée
Scénario principal	• **Description générale** : L'acteur principal en l'occurrence l'internaute s'inscrit pour devenir un utilisateur. • **Description détaillée** : 1. Lancer un navigateur; 2. Saisir l'URL de l'application web dans la barre d'adresse du navigateur; 3. Sélectionner le lien inscription; 4. Afficher la page d'inscription de l'application; 5. Saisir tous les champs d'information affichés comme le nom d'utilisateur, le mot de passe, le nom, le prénom; 6. Appuyer sur le bouton « inscription ». 7. Notifier l'internaute de son inscription.
Scénario alternatif	6a. Échec de l'inscription à cause d'une indisponibilité de la base de données. L'application web demande alors à l'internaute de réessayer plus tard.

Cas d'utilisation 2 : authentification de l'utilisateur

Nom du cas d'utilisation	authentification de l'utilisateur
Code	Cas_2
Acteur principal	L'utilisateur
Pré-condition	L'application web de Paintball est déjà installée
Scénario principal	• **Description générale** : L'acteur principal s'authentifie avant d'accéder aux ressources de l'application web à savoir le contrôle du robot. • **Description détaillée** : 1. Lancer un navigateur; 2. Saisir l'URL de l'application web dans la barre d'adresse du navigateur; 3. Sélectionner le lien authentification; 4. Afficher la page d'authentification de l'application; 5. Saisir le nom d'utilisateur et le mot de passe; 6. Appuyer sur le bouton « authentifier »; 7. Authentifier l'utilisateur par l'application; 8. Avoir accès à la page de contrôle du robot.
Scénario alternatif	7a. Échec de l'authentification, l'application notifie l'utilisateur et le redirige sur la page d'authentification pour un nouvel essai.

Cas d'utilisation 3 : La visualisation de l'environnement du robot

Nom du cas d'utilisation	La visualisation de l'environnement du robot
Code	Cas_3
Acteur principal	L'utilisateur
Pré-condition	L'utilisateur s'est authentifié

Scénario principal	• **Description générale** : L'acteur principal visualise l'environnement du robot à distance grâce à une caméra embarquée. • **Description détaillée** : 1. Appuyer sur le bouton « caméra ». 2. Mettre à jour la page avec l'image continue de l'environnement du robot.
Scénario alternatif	2a. Indisponibilité de la caméra, notification de l'utilisateur.

Cas d'utilisation 4 : La détection de la disponibilité du robot

Nom du cas d'utilisation	La détection de la disponibilité du robot
Code	Cas_4
Acteur principal	L'utilisateur
Pré-condition	Succès du cas_2 : l'authentification de l'utilisateur
Scénario principal	• **Description générale** : L'acteur principal a la possibilité de s'informer de la disponibilité du robot avant d'en prendre le contrôle. • **Description détaillée** : 1. Appuyer sur le bouton « vérifier la disponibilité »; 2. L'application envoie une commande au robot; 3. Le robot exécute la commande et envoie une réponse à l'application; 4. L'application reçoit la réponse et la traite; 5. L'application met à jour la page avec le statut du robot : DISPONIBLE OU NON DISPONIBLE.
Scénario alternatif	3a. Aucune réponse reçue au bout de quelques secondes. On revient à l'étape 2 pour une dernière tentative.

Cas d utilisation 5 : Le contrôle du robot

Nom du cas d'utilisation	Le contrôle du robot
Code	Cas_5
Acteur principal	L'utilisateur
Pré-condition	Succès du cas_2 : l'authentification de l'utilisateur
Scénario principal	• **Description générale** : L'acteur principal prend le contrôle du robot avec des commandes de déplacement. • **Description détaillée** : 1. Appuyer sur l'un des quatre boutons de déplacement : « Avancer / Reculer, Tourner à gauche / droite »; 2. L'application envoie la commande de déplacement au robot; 3. Le robot exécute la commande et envoie une réponse à l'application; 4. L'application reçoit la réponse et la traite; 5. L'application rafraichit la page en informant l'internaute du succès ou de l'échec.
Scénario alternatif	3a. Aucune réponse reçue au bout de quelques secondes. On revient à l'étape 2 pour une dernière tentative.

Cas d utilisation 6 : Le déclenchement de tir

Nom du cas d'utilisation	Le déclenchement de tir
Code	Cas_6
Acteur principal	L'utilisateur
Pré-condition	Succès du cas_2 : l'authentification de l'utilisateur
Scénario principal	• **Description générale** : L'acteur principal commande le robot de tirer des balles. • **Description détaillée** : 6. Appuyer sur l'un des deux boutons de tir : « Tirer une balle » et « Tirer trois balles »; 7. L'application envoie la commande de tir au robot; 8. Le robot exécute la commande et envoie une réponse à l'application; 9. L'application reçoit la réponse et la traite; 10. L'application met à jour la page en informant l'internaute du succès ou de l'échec du tir.
Scénario alternatif	3a. Aucune réponse reçue au bout de quelques secondes. On revient à l'étape 2 pour une dernière tentative.

ANNEXE II

INSTALLATION

SERVEUR WEB : Apache Tomcat

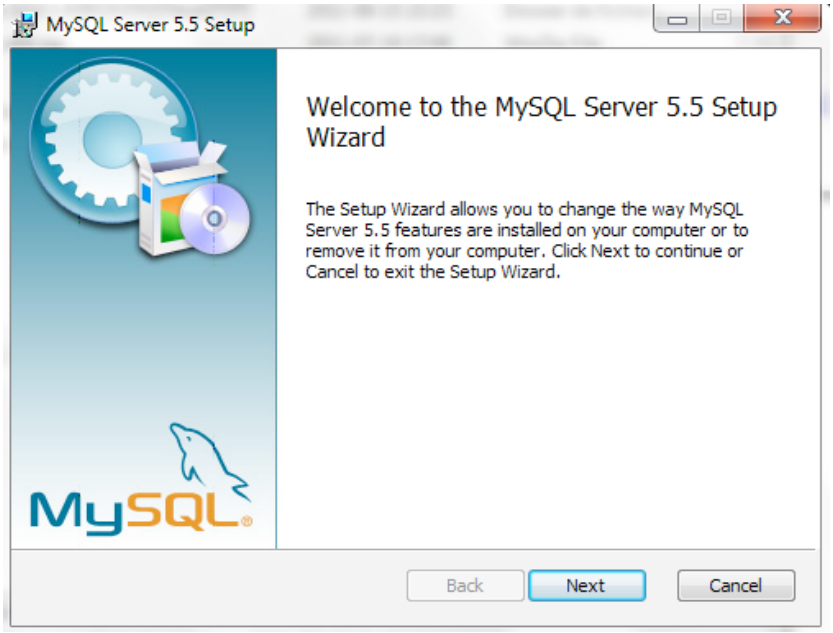
On peut se procurer l'installeur depuis cette URL http://tomcat.apache.org/download-55.cgi. Une fois l'installateur téléchargé, il ne reste plus qu'à décompresser le fichier dans le répertoire de notre choix (idéalement sur le C:).

Enfin, il faut déployer l'application web en déposant le fichier d'extension war dans le répertoire des applications de Tomcat, à savoir son dossier webapps.

Remarque: Il est indispensable que la librairie de communication du port sériel (la dll win32com.dll) soit copiée dans windows/system32.

SERVEUR DE BASE DE DONNÉES: MySQL

Le serveur MySQL est téléchargeable depuis l'URL : http://dev.mysql.com/downloads/mysql/. Une fois le serveur téléchargé, il faut « double cliquer » sur l'exécutable, la fenêtre ci-dessous devrait apparaître. Il suffit ensuite de cliquer sur le bouton « Next ». À noter que la version 5.5.14 de MySQL a été utilisée dans le cadre de ce projet.



Fig

Tout de suite après l'installation du serveur, il faut créer un utilisateur (Paintball) en exécutant la requête SQL ci-dessous:

```
CREATE USER 'PaintBall'@'localhost' IDENTIFIED BY 'PaintBall';
GRANT ALL PRIVILEGES ON *.* TO 'PaintBall'@'localhost';
```

Ensuite, il faut créer une base de données (Paintball) en exécutant la requête ci-dessous :

```
DROP DATABASE IF EXISTS `PaintBall`;
CREATE DATABASE `PaintBall`;
```

Tout de suite après l'installation du serveur, il faut créer la base de données et les tables. Pour ce faire, il suffit d'exécuter les requêtes SQL ci-dessous :

```
/*==============================================================*/
/* Nom de SGBD :  MySQL 5.0                                     */
/* Date de création :  2011-06-04 23:21:58                      */
/*==============================================================*/

/*==============================================================*/
/* Table : PB_USERHISTORYLOG                                    */
/*==============================================================*/
drop table if exists PB_USERHISTORYLOG;

create table PB_USERHISTORYLOG
(
   ID                   int not null auto_increment,
   USERID               int not null,
   LOGGEDAT             datetime not null,
   primary key (ID)
);

alter table PB_USERHISTORYLOG add constraint FK_HISTORY foreign key (USERID)
      references PB_USER (ID) on delete restrict on update restrict;


/*==============================================================*/
/* Table : PB_USER                                              */
/*==============================================================*/
drop table if exists PB_USER;

create table PB_USER
(
   ID                   int not null auto_increment,
   USERNAME             text not null,
   PASSWORD             text not null,
   primary key (ID)
```

```
);
/*==============================================================*/
/* Table : PB_USERDETAILS                                       */
/*==============================================================*/
drop table if exists PB_USERDETAILS;

create table PB_USERDETAILS
(
   ID                   int not null auto_increment,
   USERID               int not null,
   FIRSTNAME            text not null,
   LASTNAME             text not null,
   primary key (ID)
);

alter table PB_USERDETAILS add constraint FK_RENSEIGNEMENTS foreign key (USERID)
      references PB_USER (ID) on delete restrict on update restrict;
```

ANNEXE III

CODE SOURCE

CONFIGURATION : Les fichiers XML de configuration de l'application

`hibernate.cfg.xml` : Le fichier hibernate configurant l'accès à la base de données

```
<?xml version='1.0' encoding='utf-8'?>
<!DOCTYPE hibernate-configuration PUBLIC
        "-//Hibernate/Hibernate Configuration DTD//EN"
        "http://hibernate.sourceforge.net/hibernate-configuration-3.0.dtd">
<hibernate-configuration>
    <session-factory>
        <property name="connection.url">
                jdbc:mysql://localhost:3306/paintball
        </property>
        <property name="connection.driver_class">
                com.mysql.jdbc.Driver
        </property>
        <property name="connection.username">paintball</property>
        <property name="connection.password">paintball</property>
        <mapping class="com.etsmtl.paintball.model.User"/>
        <mapping class="com.etsmtl.paintball.model.UserDetails"/>
        <mapping class="com.etsmtl.paintball.model.HistoryLog"/>
        <!-- DB schema will be updated if needed -->
        <!-- <property name="hbm2ddl.auto">update</property> -->
    </session-factory>
</hibernate-configuration>
```

`spring-paintball.xml` : Le fichier de configuration de Spring. Il constitue le fichier de configuration principal de l'application (port serial utilisé par le Paintball, le courriel de l'administrateur de l'application, les informations smtp pour le courriel).

```
<?xml version="1.0" encoding="UTF-8"?>
<beans xmlns="http://www.springframework.org/schema/beans"
      xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
      xsi:schemaLocation="http://www.springframework.org/schema/beans
      http://www.springframework.org/schema/beans/spring-beans-3.0.xsd">

      <bean id="configuration"
            class="com.etsmtl.paintball.business.PaintballConfiguration"
            factory-method="getInstance">
            <property name="paintballCommands">
                  <map>
```

```
                        <entry key="position"
                              value-ref="paintBallPositionCommand" />
                        <entry key="status"
                              value-ref="paintBallStatusCommand" />
                        <entry key="shootOne"
                              value-ref="paintBallShootOneCommand" />
                        <entry key="shootThree" value-
ref="paintBallShootThreeCommand" />
                  </map>
            </property>
            <property name="microcontrollerCommands">
                  <map>
                        <entry key="position" value="0x440x%s0x%s" />
                        <entry key="status" value="0x410x410x41" />
                        <entry key="shootOne" value="0x420x420x42" />
                        <entry key="shootThree" value="0x430x430x43" />
                  </map>
            </property>
            <property name="paintBallController"
ref="paintBallControllerBean" />
            <property name="serialPort" value="COM7" />
      </bean>

      <bean id="paintBallPositionCommand"
class="com.etsmtl.paintball.business.PaintBallPositionCommand">
            <constructor-arg ref="paintballBean" />
      </bean>
      <bean id="paintBallStatusCommand"
class="com.etsmtl.paintball.business.PaintBallStatusCommand">
            <constructor-arg ref="paintballBean" />
      </bean>
      <bean id="paintBallShootOneCommand"
class="com.etsmtl.paintball.business.PaintBallShootOneCommand">
            <constructor-arg ref="paintballBean" />
      </bean>
      <bean id="paintBallShootThreeCommand"

      class="com.etsmtl.paintball.business.PaintBallShootThreeCommand">
            <constructor-arg ref="paintballBean" />
      </bean>

      <bean id="paintballBean"
class="com.etsmtl.paintball.business.Paintball"
            scope="singleton" />

      <bean id="paintBallControllerBean"
class="com.etsmtl.paintball.business.PaintBallController"
            factory-method="getInstance" />

      <bean id="microcontroller"
class="com.etsmtl.paintball.business.Microcontroller"
            scope="singleton">
            <property name="paintball" ref="paintballBean" />
      </bean>
```

```
      <bean id="sendMail" class="com.etsmtl.paintball.business.SendMail"
scope="singleton">
            <property name="properties">
                  <props>
                        <prop key="mail.smtp.host">localhost</prop>
                        <prop key="mail.smtp.port">25</prop>
                  </props>
            </property>
            <property name="to" value="alain.april@etsmtl.ca" />
      </bean>

</beans>
```

`log4j.properties` : Le fichier log4j configurant la journalisation

```
log4j.rootLogger=INFO, CA

log4j.appender.CA=org.apache.log4j.ConsoleAppender

log4j.appender.CA.layout=org.apache.log4j.PatternLayout
log4j.appender.CA.layout.ConversionPattern=%-4r [%t] %-5p %c %x - %m%n
```

INTERNATIONALISATION : Fichiers ressources pour supporter le Français et l'Anglais

`MessageResources.properties` : Le fichier ressource français

```
# -- standard errors --
errors.header=<UL>
errors.prefix=<LI>
errors.suffix=</LI>
errors.footer=</UL>

# -- validator --
errors.invalid={0} est invalide.
errors.maxlength={0} ne peut depasser {1} characters.
errors.minlength={0} ne peut contenir moins de {1} characters.
errors.range={0} n'est pas dans l'intervalle {1} à {2}.
errors.required={0} est requis.
errors.byte={0} doit être un byte.
errors.date={0} n'est pas une date.
errors.double={0} devrait être un double.
errors.float={0} devrait être un float.
errors.integer={0} devrait être un integer.
errors.long={0} devrait être un long.
errors.short={0} devrait être un short.
errors.creditcard={0} n'est pas une carte de crédit valide.
errors.email={0} n'est pas une adresse courriel valide.

# -- other --
errors.cancel=Operation annulée.
errors.detail={0}
errors.general=Le processus ne s'est pas terminée. Les détails devraient
suivre.
errors.token=Demande n'a pas pu être terminée. L'opération n'est pas dans
l'ordre.

#--login---
errors.username = Nom d'Utilisateur invalide
errors.password = Mot de passe invalide
errors.username.password = Nom d'Utilisateur ou Mot de passe invalide
login.username = Nom d'Utilisateur
login.password = Mot de Passe
login.register = Inscription
login.submit = Connexion

#--register--
register.description = S'il vous plaît utilisez le formulaire ci-dessous
pour vous inscrire
errors.firstname =  Prenom requis
errors.lastname =  Nom requis
errors.email =  Courriel requis

#--contact--
errors.fullname = Le Nom est requis
errors.subject = L'objet est requis
errors.message = Le message est requis
```

```
# -- welcome --
welcome.heading=Introduction !
welcome.message=Le projet d'application à deux objectifs, le premier
objectif consiste à mettre en pratique les techniques de système embarqué
et temps réel; le deuxième objectif consiste à concevoir une application
web de contrôle du microcontrôleur. <br><br>Les technologies et les outils
choisis sont des logiciels libres et « populaires » au sein de la
communauté de développeurs. Ce choix a permis de réduire considérablement
la durée et le coût de développement, mais aussi de réduire plus tard le
coût de la maintenabilité et d'évolution de l'application.<br><br>Le
microcontrôleur utilisé dans le cadre de ce projet Paintball est un AVR 32
EVK1100 appartenant à la série AT32UC3A des microcontrôleurs Amtel. Ce
microcontrôleur, en plus d'être fourni avec un environnement de
développement complet, offre des fonctionnalités assez riches.
welcome.message.private=Le projet d'application à deux objectifs, le
premier objectif consiste à mettre en pratique les techniques de système
embarqué et temps réel; le deuxième objectif consiste à concevoir une
application web de contrôle du microcontrôleur.

# -- commande --
command.operational.command = status
command.operational.label = Vérifier le statut
command.shootOneBall.command = shootOne
command.shootOneBall.label = Tirer une balle
command.shootThreeBalls.command = shootThree
command.shootThreeBalls.label = Tirer trois balles
command.setPosition.command = position
command.setPosition.label = Positionner
command.shoot = tirez
command.status = statut
command.submit = Soumettre

# -- entete --
header.home = Accueil
header.contact = Contactez-nous
header.language.english = Anglais
header.language.french = Francais

# -- title --
home.title=CONTRÔLE À DISTANCE D'UN ROBOT DE PAINTBALL
welcome.title=CONTRÔLE À DISTANCE D'UN ROBOT DE PAINTBALL

# -- contact --
contact.description = S'il vous plaît utilisez le formulaire ci-dessous
contact.name = Nom
contact.email = Courriel
contact.subject = Objet
contact.title = Contactez-nous
contact.heading = Contactez-nous
contact.message = Message
contact.send = Envoyer

# -- header --
footer.reserved = © 2011 - Tous droits réservés
```

```
# -- register --
registerForm.firstname = Prénom
registerForm.lastname = Nom
registerForm.email = Courriel
registerForm.username = Nom d'utilisateur
registerForm.password = Mot de Passe
register.title = Inscription
register.heading = Inscription

#-- history--
history.heading = Historique
history.title = Historique
history.description = Historique de connexion
history.time = Date & Heure
# -- logout --
logout.title = Déconnexion
logout.robot.status = Statut du Robot

# -- commande du robot --
robot.commandTitle = Commande du robot
```

`MessageResources_en.properties` : Le fichier ressource anglais

```
# -- standard errors --
errors.header=<UL>
errors.prefix=<LI>
errors.suffix=</LI>
errors.footer=</UL>
# -- validator --
errors.invalid={0} is invalid.
errors.maxlength={0} can not be greater than {1} characters.
errors.minlength={0} can not be less than {1} characters.
errors.range={0} is not in the range {1} through {2}.
errors.required={0} is required.
errors.byte={0} must be an byte.
errors.date={0} is not a date.
errors.double={0} must be an double.
errors.float={0} must be an float.
errors.integer={0} must be an integer.
errors.long={0} must be an long.
errors.short={0} must be an short.
errors.creditcard={0} is not a valid credit card number.
errors.email={0} is an invalid e-mail address.
# -- other --
errors.cancel=Operation cancelled.
errors.detail={0}
errors.general=The process did not complete. Details should follow.
errors.token=Request could not be completed. Operation is not in sequence.
#--login---
errors.username = Username not valid
errors.password = Password not valid
errors.username.password = Username or Password not valid
login.username = UserName
login.password = Password
login.register = Register
login.submit = Sign in
```

```
#--register--
errors.firstname = Firstname required
errors.lastname = Lastname required
errors.email = Valid email required

#--contact--
errors.fullname = Name is required
errors.subject = Subject is required
errors.message = Message is required

# -- welcome --
welcome.heading=Introduction !
welcome.message=The implementation project has two objectives, the first
objective are to apply the techniques and real-time embedded system and the
second objective is to design a web application control of the
microcontroller.<br><br> technologies and the tools chosen are free
software and popular among the developer community. This choice has
significantly reduced the duration and cost of development, but also by
reducing the cost of maintainability and evolution of the
application.<br><br>The microcontroller used in this project is a Paintball
AVR 32 EVK1100 belonging to the series AT32UC3A Amtel microcontroller. The
microcontroller, in addition to being provided with a complete development
environment, offers features rich enough.
welcome.message.private=The implementation project has two objectives, the
first objective are to apply the techniques and real-time embedded system
and the second objective is to design a web application control of the
microcontroller.
# -- home --

# -- commande --
command.operational.command = status
command.operational.label = Check status
command.shootOneBall.command = shootOne
command.shootOneBall.label = Shoot one ball
command.shootThreeBalls.command = shootThree
command.shootThreeBalls.label = Shoot three balls
command.setPosition.command = position
command.setPosition.label = Position
command.shoot = shoot
command.status = status
command.submit = submit

# -- header --
header.home = Home
header.contact = Contact-us
header.language.english = English
header.language.french = French
# -- title --
home.title=CONTROL PAINTBALL ROBOT OVER INTERNET
welcome.title=CONTROL PAINTBALL ROBOT OVER INTERNET

# -- contact --
contact.description = Please send me a message via the form below
contact.name = Name
contact.email = Email
```

```
contact.subject = Subject
contact.title = Contact-us
contact.heading = Contact-us
contact.message = Message
contact.send = Send

# -- footer --
footer.reserved = © 2011 - All Rights Reserved

# -- register --
register.description = Please register via the form below
register.heading = Register
register.title = Register
registerForm.firstname = FistName
registerForm.lastname = LastName
registerForm.email = Email
registerForm.username = UserName
registerForm.password = Password

#-- history--
history.heading = History
history.title = History
history.description = Here's your login history
history.time = Date & Time

# -- logout --
logout.title = Logout
logout.robot.status = Robot Status

# -- robot command --
robot.commandTitle = Robot command
```

MODÈLE MVC :

CONTROLEUR :

web.xml : Le fichier de configuration de Tomcat

```
<?xml version="1.0" encoding="UTF-8"?>

  <!DOCTYPE web-app PUBLIC
      "-//Sun Microsystems, Inc.//DTD Web Application 2.3//EN"
      "http://java.sun.com/j2ee/dtds/web-app_2_3.dtd">

<web-app>
      <display-name>PaintBall Application</display-name>

      <listener>
            <listener-class>
            com.etsmtl.paintball.utils.PaintballServletContextListener
            </listener-class>
      </listener>

      <listener>
            <listener-class>
            com.etsmtl.paintball.utils.PaintballServletContextListener
            </listener-class>
      </listener>

      <!-- Standard Action Servlet Configuration -->
      <servlet>
            <servlet-name>action</servlet-name>
            <servlet-class>
                        org.apache.struts.action.ActionServlet
            </servlet-class>
            <init-param>
                  <param-name>config</param-name>
                  <param-value>/WEB-INF/struts-config.xml</param-value>
            </init-param>
            <load-on-startup>2</load-on-startup>
      </servlet>
      <!-- Standard Action velocity Configuration -->
      <servlet>
            <servlet-name>velocity</servlet-name>
            <servlet-class>
                  org.apache.velocity.tools.view.VelocityViewServlet
            </servlet-class>
      </servlet>

      <!-- Standard Action Servlet Mapping -->
      <servlet-mapping>
            <servlet-name>action</servlet-name>
            <url-pattern>*.do</url-pattern>
      </servlet-mapping>
```

```
	<servlet-mapping>
		<servlet-name>velocity</servlet-name>
		<url-pattern>*.vm</url-pattern>
	</servlet-mapping>

	<!-- Standard Session Configuration -->
	<session-config>
		<session-timeout>30</session-timeout>
	</session-config>

	<!-- The Usual Welcome File List -->
	<welcome-file-list>
		<welcome-file>/pages/home.jsp</welcome-file>
	</welcome-file-list>


	<!-- Struts Tag  Library  Descriptors -->
	<taglib>
		<taglib-uri>/tags/struts-bean</taglib-uri>
		<taglib-location>
			/WEB-INF/tlds/struts-bean.tld
		</taglib-location>
	</taglib>
	<taglib>
		<taglib-uri>/tags/struts-html</taglib-uri>
		<taglib-location>
			/WEB-INF/tlds/struts-html.tld
		</taglib-location>
	</taglib>
	<taglib>
		<taglib-uri>/tags/struts-logic</taglib-uri>
		<taglib-location>
			/WEB-INF/tlds/struts-logic.tld
		</taglib-location>
	</taglib>
	<taglib>
		<taglib-uri>/tags/struts-nested</taglib-uri>
		<taglib-location>
			/WEB-INF/tlds/struts-nested.tld
		</taglib-location>
	</taglib>
	<taglib>
		<taglib-uri>/tags/struts-template</taglib-uri>
		<taglib-location>
			/WEB-INF/tlds/struts-template.tld
		</taglib-location>
	</taglib>
	<taglib>
		<taglib-uri>/tags/struts-tiles</taglib-uri>
		<taglib-location>
			/WEB-INF/tlds/struts-tiles.tld
		</taglib-location>
	</taglib>

</web-app>
```

Struts-config.xml : Le fichier de configuration de Struts qui joue le rôle de contrôleur

```
<?xml version="1.0" encoding="ISO-8859-1" ?>

<!DOCTYPE struts-config PUBLIC
  "-//Apache Software Foundation//DTD Struts Configuration 1.1//EN"
  "http://jakarta.apache.org/struts/dtds/struts-config_1_1.dtd">

<struts-config>
      <!-- Form Bean Definitions -->
      <form-beans>

            <!-- Login form bean -->
            <form-bean name="loginForm"
                  type="com.etsmtl.paintball.business.forms.LoginForm" />
            <!-- Logout form bean -->
            <form-bean name="logoutForm"
                  type="com.etsmtl.paintball.business.forms.LogoutForm" />
            <form-bean name="contactForm"
      type="com.etsmtl.paintball.business.forms.ContactDynaActionForm">
                  <form-property name="fullname" type="java.lang.String" />
                  <form-property name="email" type="java.lang.String" />
                  <form-property name="subject" type="java.lang.String" />
                  <form-property name="message" type="java.lang.String" />
            </form-bean>
            <form-bean name="registerForm"
      type="com.etsmtl.paintball.business.forms.RegisterDynaActionForm">
                  <form-property name="firstname"
                              type="java.lang.String" />
                  <form-property name="lastname" type="java.lang.String" />
                  <form-property name="email" type="java.lang.String" />
                  <form-property name="registerUsername"
                              type="java.lang.String" />
                  <form-property name="registerPassword"
                              type="java.lang.String" />
            </form-bean>
            <form-bean name="commandForm"
      type="com.etsmtl.paintball.business.forms.CommandDynaActionForm">
                  <form-property name="command" type="java.lang.String" />
                  <form-property name="x" type="java.lang.String" />
                  <form-property name="y" type="java.lang.String" />
            </form-bean>
      </form-beans>
      <!-- Global Forward Definitions -->
      <global-forwards>
            <!-- Home forward -->
            <forward name="home" path="/home.do" />
            <!-- Login forward -->
            <forward name="login" path="/login.do" />
            <!-- Logoff forward -->
            <forward name="logout" path="/logout.do" />
            <!-- failure forward -->
            <forward name="failure" path="/error.do" />
      </global-forwards>
```

```
	<!-- Action Mapping Definitions -->
	<action-mappings>
		<!-- Forwards to home.jsp -->
		<action path="/home" forward="/pages/home.jsp" />

		<action path="/languageSelection"
	type="com.etsmtl.paintball.business.actions.LanguageSelectionAction"
			parameter="language" />

		<!--Login Action Mapping Definitions -->
		<!-- Process a user to login -->
		<action path="/login"
			type="com.etsmtl.paintball.business.actions.LoginAction"
			name="loginForm"
			scope="session" input="/pages/home.jsp" validate="true">
		</action>

		<!-- Process a user to logout -->
		<action path="/logout"
			type="com.etsmtl.paintball.business.actions.LogoutAction"
			name="logoutForm"
			scope="session" input="/pages/home.jsp" validate="false">
		</action>

		<action path="/contact"
		type="com.etsmtl.paintball.business.actions.ContactAction"
			name="contactForm"
			scope="session" input="/pages/contact.jsp"
			validate="true">
		</action>

		<action path="/register"
		type="com.etsmtl.paintball.business.actions.RegisterAction"
			name="registerForm"
			scope="session" input="/pages/register.jsp"
			validate="true">
		</action>

		<action path="/command"
		type="com.etsmtl.paintball.business.actions.CommandAction"
			name="commandForm"
			scope="session" input="/pages/home.jsp" validate="false">
		</action>
	</action-mappings>
	<!-- Message Resources Definitions -->
	<message-resources parameter="resources/MessageResources" />
	<!-- Validator plugin -->
	<plug-in className="org.apache.struts.validator.ValidatorPlugIn">
		<set-property property="pathnames"
			value="/WEB-INF/validator-rules.xml,
		  /WEB-INF/validation.xml" />
	</plug-in>

</struts-config>
```

COUCHE BUSINESS : Communication avec le robot sur le port sériel

`Microcontroller.java` : Cette classe s'occupe de l'envoie des commandes au Microcontrôleur. Elle assure la communication entre l'application et le microcontrôleur. Elle est lancée au démarrage de l'application et s'exécute parallèlement à l'application.

```
package com.etsmtl.paintball.business;

import java.io.IOException;
import java.io.InputStream;
import java.io.OutputStream;
import java.util.Enumeration;
import java.util.TooManyListenersException;

import javax.comm.CommPortIdentifier;
import javax.comm.PortInUseException;
import javax.comm.SerialPort;
import javax.comm.SerialPortEvent;
import javax.comm.SerialPortEventListener;
import javax.comm.UnsupportedCommOperationException;

import org.apache.log4j.Logger;

import com.sun.comm.Win32Driver;

public class Microcontroller implements Runnable, SerialPortEventListener {

	static final Logger logger = Logger.getLogger(Microcontroller.class);

	private SerialPort serialPort;
	private Paintball paintball;

	public Microcontroller() {
		Thread thread = new Thread(this);
		thread.start();
	}

	@Override
	public void run() {
		try {
			while (true) {
				if (paintball == null) {
					continue;
				}
				If(paintball.getState().isMicrocontrollerWorking())
				{
					send(paintball.getMicrocontrollerCommand());
					Thread.sleep(500);
					paintball.setState(
							paintball.getReadyState());
```

```
                        logger.info("paintball status changed to ready");
                }
        }
        } catch (InterruptedException e) {
                logger.error(e.getMessage());
        } finally {
                closeConnection();
        }
}

public void send(String command) {

        if (getSerialPort() == null) {
                return;
        }
        try {
                OutputStream outputStream =
                                getSerialPort().getOutputStream();
                outputStream.write(command.getBytes());
                logger.info("Command sent : " + command);
        } catch (IOException e) {
                logger.error(e.getMessage());
        }
}

public SerialPort getSerialPort() {
        if (serialPort == null) {
                openConnection();
        }
        return serialPort;
}

@Override
public void serialEvent(SerialPortEvent serialPortEvent) {
        switch (serialPortEvent.getEventType()) {
                case SerialPortEvent.BI:
                case SerialPortEvent.OE:
                case SerialPortEvent.FE:
                case SerialPortEvent.PE:
                case SerialPortEvent.CD:
                case SerialPortEvent.CTS:
                case SerialPortEvent.DSR:
                case SerialPortEvent.RI:
                case SerialPortEvent.OUTPUT_BUFFER_EMPTY:
                        break;
                case SerialPortEvent.DATA_AVAILABLE:
                        byte[] readBuffer = new byte[20];
                        try {
                                InputStream inputStream =
                                                getSerialPort().getInputStream();
                                while (inputStream.available() > 0) {
                                        inputStream.read(readBuffer);
                                }
```

```
                    logger.info(new String(readBuffer));
                } catch (IOException e) {
                    logger.error(e.getMessage());
                }
                break;
            }
        }

        public void setPaintball(Paintball paintball) {
            this.paintball = paintball;
        }

        public Paintball getPaintball() {
            return paintball;
        }

        public void openConnection() {
            CommPortIdentifier portId = getSerialPortIdentifier();
            if (portId != null) {
                try {
                    serialPort = (SerialPort) portId.open("Paintball", 2000);
                    serialPort.setSerialPortParams(9600,
                                    SerialPort.DATABITS_8,
                                    SerialPort.STOPBITS_1,
                                    SerialPort.PARITY_NONE);
                    serialPort.addEventListener(this);
                    serialPort.notifyOnDataAvailable(true);
                } catch (PortInUseException portInUseException) {
                    logger.error(portInUseException.getMessage());
                } catch (TooManyListenersException e) {
                    logger.error(e.getMessage());
                } catch (UnsupportedCommOperationException e) {
                    logger.error(e.getMessage());
                }
            }
        }

        public void closeConnection() {
            if (serialPort != null) {
                serialPort.close();
            }
        }

        public CommPortIdentifier getSerialPortIdentifier() {
            // driver initialization
            Win32Driver w32Driver = new Win32Driver();
            w32Driver.initialize();

            CommPortIdentifier portId = null;
            String portName =
                        PaintballConfiguration.getInstance().getSerialPort();
            Enumeration portList = CommPortIdentifier.getPortIdentifiers();
            while (portList.hasMoreElements()) {
                portId = (CommPortIdentifier) portList.nextElement();
```

```
                if (portId.getPortType() ==
                                        CommPortIdentifier.PORT_SERIAL) {
                      if (portId.getName().equals(portName)) {
                            break;
                      }
                }
          }
          return portId;
    }
}
```

SendMail.java : Cette classe s'occupe de l'envoi de courriel.

```
package com.etsmtl.paintball.business;

import java.util.Properties;
import javax.mail.MessagingException;
import javax.mail.Session;
import javax.mail.Transport;
import javax.mail.internet.InternetAddress;
import javax.mail.internet.MimeMessage;
import javax.mail.internet.MimeMessage.RecipientType;

import org.apache.log4j.Logger;

public class SendMail {

      static final Logger logger = Logger.getLogger(SendMail.class);
      private String from;
      private String to;
      private String subject;
      private String message;
      private Properties properties;

      public SendMail() {}

      public void send() {

            Session session = Session.getDefaultInstance(properties, null);
            MimeMessage mimeMessage = new MimeMessage(session);
            try {
                  mimeMessage.setFrom(new InternetAddress(from));
                  mimeMessage.setRecipient(RecipientType.TO,
                                          new InternetAddress(to));
                  mimeMessage.setSubject(subject);
                  mimeMessage.setText(message);
                  Transport.send(mimeMessage);
                  logger.info(this);
            } catch (MessagingException e) {
                  logger.error(e.getMessage());
            }

      }
```

```
	public void setFrom(String from) {
		this.from = from;
	}

	public String getFrom() {
		return from;
	}

	public void setTo(String to) {
		this.to = to;
	}

	public String getTo() {
		return to;
	}

	public void setSubject(String subject) {
		this.subject = subject;
	}

	public String getSubject() {
		return subject;
	}

	public void setMessage(String message) {
		this.message = message;
	}

	public String getMessage() {
		return message;
	}

	public void setProperties(Properties properties) {
		this.properties = properties;
	}

	public Properties getProperties() {
		return properties;
	}

	@Override
	public String toString() {
		StringBuilder stringBuilder = new StringBuilder().append("\n");
		stringBuilder.append("From: ").append(from).append("\n");
		stringBuilder.append("To: ").append(to).append("\n");
		stringBuilder.append("Subject: ").append(subject).append("\n");
		stringBuilder.append("Message: ").append(message);
		return stringBuilder.toString();
	}

}
```

`IPaintball.java` : Cette interface définit les différentes commandes du Paintball.

```
package com.etsmtl.paintball.business;

public interface IPaintball {

	public void shootOne();

	public void shootThree();

	public void status();

	public void position();
}
```

`Paintball.java` : Cette classe gère l'état du Paintball et implémente les différentes commandes du Paintball.

```
package com.etsmtl.paintball.business;

import org.apache.log4j.Logger;

public class Paintball implements IPaintball {

	static final Logger logger = Logger.getLogger(Paintball.class);

	private ExecutingState executingState = new ExecutingState(this);
	private ReadyState readyState = new ReadyState(this);

	private IPaintballState state = readyState;
	private String microcontrollerCommand;

	public Paintball() {
	}

	@Override
	public void position() {
		updateMicrocontrollerCommand("position");
		state.perform();
	}

	@Override
	public void status() {
		updateMicrocontrollerCommand("status");
		state.perform();
	}

	@Override
	public void shootOne() {
		updateMicrocontrollerCommand("shootOne");
		state.perform();
	}
```

```
	@Override
	public void shootThree() {
		updateMicrocontrollerCommand("shootThree");
		state.perform();
	}

	private void updateMicrocontrollerCommand(String key) {
		PaintballConfiguration paintballConfiguration =
					PaintballConfiguration.getInstance();
		String microcontrollerCmd =
				paintballConfiguration.getMicrocontrollerCommands()
							.get(key);
		setMicrocontrollerCommand(microcontrollerCmd);
	}

	public ExecutingState getExecutingState() {
		return executingState;
	}

	public ReadyState getReadyState() {
		return readyState;
	}

	public void setState(IPaintballState state) {
		this.state = state;
	}

	public IPaintballState getState() {
		return state;
	}

	public void setMicrocontrollerCommand(String microcontrollerCommand)
{
		this.microcontrollerCommand = microcontrollerCommand;
	}

	public String getMicrocontrollerCommand() {
		return microcontrollerCommand;
	}
}
```

`IPaintballCommand.java` : Cette interface définit la méthode des différentes commandes du Paintball.

```
package com.etsmtl.paintball.business;

public interface IPaintballCommand {

	public void execute();
}
```

PaintballPositionCommand.java : Cette classe définit la commande de positionnement.

```
package com.etsmtl.paintball.business;

public class PaintBallPositionCommand implements IPaintballCommand {

	private Paintball paintBall;

	public PaintBallPositionCommand(Paintball paintBall) {
		this.paintBall = paintBall;
	}

	@Override
	public void execute() {
		paintBall.position();
	}
}
```

PaintballStatusCommand.java : Cette classe définit la commande de vérification de status.

```
package com.etsmtl.paintball.business;

import org.apache.log4j.Logger;

public class PaintBallStatusCommand implements IPaintballCommand {

	static final Logger logger = Logger.getLogger(Microcontroller.class);

	private Paintball paintBall;

	public PaintBallStatusCommand(Paintball paintBall) {
		this.paintBall = paintBall;
	}

	@Override
	public void execute() {
		paintBall.status();
	}
}
```

PaintballShootOneCommand.java : Cette classe définit la commande de tire d'une balle.

```
package com.etsmtl.paintball.business;

public class PaintBallShootOneCommand implements IPaintballCommand {

	private Paintball paintBall;

	public PaintBallShootOneCommand(Paintball paintBall) {
		this.paintBall = paintBall;
	}
```

```
	@Override
	public void execute() {
		paintBall.shootOne();
	}
}
```

PaintballShootThreeCommand.java : Cette classe définit la commande de tire de trois balles.

```
package com.etsmtl.paintball.business;

public class PaintBallShootThreeCommand implements IPaintballCommand {

	private Paintball paintBall;

	public PaintBallShootThreeCommand(Paintball paintBall) {
		this.paintBall = paintBall;
	}

	@Override
	public void execute() {
		paintBall.shootThree();
	}
}
```

IPaintballState.java : Cette interface définit l'état du Paintball.

```
package com.etsmtl.paintball.business;

public interface IPaintballState {

	public void perform();

	public boolean isMicrocontrollerWorking();
}
```

ReadyState.java : Cette classe définit l'état « Prêt » du Paintball.

```
package com.etsmtl.paintball.business;

public class ReadyState implements IPaintballState {

	private Paintball paintball;
	private boolean isMicrocontrollerWorking = false;

	public ReadyState(Paintball paintBall) {
		this.paintball = paintBall;
	}
```

```
	@Override
	public void perform() {
		IPaintballState newState = paintball.getExecutingState();
		paintball.setState(newState);
		newState.perform();
	}

	@Override
	public boolean isMicrocontrollerWorking() {
		return isMicrocontrollerWorking;
	}
}
```

ExecutingState.java : Cette classe définit l'état « En cours d'exécution » du Paintball.

```
package com.etsmtl.paintball.business;

public class ExecutingState implements IPaintballState {

	private Paintball paintball;
	private boolean isMicrocontrollerWorking = true;

	public ExecutingState(Paintball paintball) {
		this.paintball = paintball;
	}

	@Override
	public void perform() {
		while (paintball.getState().isMicrocontrollerWorking()) {

		}
		paintball.setState(paintball.getReadyState());
	}

	@Override
	public boolean isMicrocontrollerWorking() {
		return isMicrocontrollerWorking;
	}
}
```

PaintballConfiguration.java : Cette classe est un singleton, elle gère la configuration du Paintball.

```
package com.etsmtl.paintball.business;

import java.util.Map;

import org.apache.log4j.Logger;
```

```
public class PaintballConfiguration {

	static final Logger logger =
				Logger.getLogger(PaintballConfiguration.class);

	private static PaintballConfiguration paintballConfiguration;
	private Map<String, IPaintballCommand> paintballCommands;
	private Map<String, String> microcontrollerCommands;
	private PaintBallController paintBallController;
	private String serialPort;

	private PaintballConfiguration() {}

	public static PaintballConfiguration getInstance() {
		if (paintballConfiguration == null) {
			paintballConfiguration = new PaintballConfiguration();
		}
		return paintballConfiguration;
	}

	public void setPaintballCommands(
			Map<String, IPaintballCommand> paintballCommands) {
		this.paintballCommands = paintballCommands;
	}

	public Map<String, IPaintballCommand> getPaintballCommands() {
		return paintballCommands;
	}

	public void setPaintBallController(
			PaintBallController paintBallController) {
		this.paintBallController = paintBallController;
	}

	public PaintBallController getPaintBallController() {
		return paintBallController;
	}

	public void setSerialPort(String serialPort) {
		this.serialPort = serialPort;
	}

	public String getSerialPort() {
		return serialPort;
	}

	public void setMicrocontrollerCommands(
			Map<String, String> microcontrollerCommands) {
		this.microcontrollerCommands = microcontrollerCommands;
	}

	public Map<String, String> getMicrocontrollerCommands() {
		return microcontrollerCommands;
	}
}
```

COUCHE PERSISTENCE: Opérations de CRUD sur la base de données

HibernateUtil.java : Cette classe s'occupe de créer une session pour la persistance des données.

```
package com.etsmtl.PaintBall.hibernate;

import org.hibernate.*;
import org.hibernate.cfg.*;

public class HibernateUtil {

   private static final SessionFactory sessionFactory;

    static {
        try {
            sessionFactory = new AnnotationConfiguration().
                    configure("hibernate.cfg.xml").
                    buildSessionFactory();
        }
        catch (Throwable ex) {
            throw new ExceptionInInitializerError(ex);
        }
    }

    public static Session getSession() throws HibernateException {
        return sessionFactory.openSession();
    }
}
```

IBasicEntiry.java : Cette interface représente chacune des entités.

```
package com.etsmtl.PaintBall.hibernate;

public interface IBasicEntity {

}
```

BasicEntiry.java : Cette classe définit une entité générique.

```
package com.etsmtl.paintball.hibernate.dao;

import com.etsmtl.paintball.hibernate.IBasicEntity;

public class BasicEntity implements IBasicEntity{

}
```

`User.java` : Cette classe définit une entité User.

```
package com.etsmtl.paintball.model;

import java.util.List;
import javax.persistence.Basic;
import javax.persistence.CascadeType;
import javax.persistence.Column;
import javax.persistence.Entity;
import javax.persistence.FetchType;
import javax.persistence.GeneratedValue;
import javax.persistence.Id;
import javax.persistence.OneToMany;
import javax.persistence.OneToOne;
import javax.persistence.Table;
import org.hibernate.annotations.OrderBy;

import com.etsmtl.paintball.hibernate.dao.BasicEntity;

@Table(name = "pb_user", catalog = "paintBall")
@Entity
public class User extends BasicEntity {
	private int id;

	@Column(name = "ID")
	@Id
	@GeneratedValue
	public int getId() {
		return id;
	}

	public void setId(int id) {
		this.id = id;
	}

	private String username;

	@Column(name = "USERNAME")
	@Basic
	public String getUsername() {
		return username;
	}

	public void setUsername(String username) {
		this.username = username;
	}

	private String password;

	@Column(name = "PASSWORD")
	@Basic
	public String getPassword() {
		return password;
	}
```

```
	public void setPassword(String password) {
		this.password = password;
	}

	private UserDetails userDetails;

	@OneToOne(mappedBy = "user", cascade = CascadeType.ALL, fetch =
FetchType.EAGER)
	public UserDetails getUserDetails() {
		return userDetails;
	}

	public void setUserDetails(UserDetails userDetails) {
		this.userDetails = userDetails;
	}

	private List<HistoryLog> historyLog;

	@OneToMany(mappedBy = "user", targetEntity = HistoryLog.class,
cascade = CascadeType.ALL, fetch = FetchType.LAZY)
	@OrderBy(clause = "loggedat DESC")
	public List<HistoryLog> getHistoryLog() {
		return historyLog;
	}

	public void setHistoryLog(List<HistoryLog> historyLog) {
		this.historyLog = historyLog;
	}

	@Override
	public boolean equals(Object o) {
		if (this == o)
			return true;
		if (o == null || getClass() != o.getClass())
			return false;

		User user = (User) o;

		if (id != user.id)
			return false;
		if (password != null ? !password.equals(user.password)
				: user.password != null)
			return false;
		if (username != null ? !username.equals(user.username)
				: user.username != null)
			return false;

		return true;
	}

	@Override
	public int hashCode() {
		int result = id;
		result = 31 * result + (username != null ? username.hashCode()
: 0);
```

```
            result = 31 * result + (password != null ? password.hashCode()
: 0);
            return result;
      }

      @Override
      public String toString() {
            return "username = " + username + " && password = " + password;
      }

}
```

`UserDetails.java` : Cette classe définit une entité UserDetails.

```
package com.etsmtl.paintball.model;

import javax.persistence.*;
import com.etsmtl.paintball.hibernate.dao.BasicEntity;

@Table(name = "pb_userdetails", catalog = "paintBall")
@Entity
public class UserDetails extends BasicEntity {
      private int id;

      @Column(name = "ID")
      @Id
      @GeneratedValue
      public int getId() {
            return id;
      }

      public void setId(int id) {
            this.id = id;
      }

      private String firstname;

      @Column(name = "FIRSTNAME")
      @Basic
      public String getFirstname() {
            return firstname;
      }

      public void setFirstname(String firstname) {
            this.firstname = firstname;
      }

      private String lastname;

      @Column(name = "LASTNAME")
      @Basic
      public String getLastname() {
            return lastname;
      }
```

```
	public void setLastname(String lastname) {
		this.lastname = lastname;
	}

	private String email;
	@Column(name = "EMAIL")
	@Basic
	public String getEmail() {
		return email;
	}

	public void setEmail(String email) {
		this.email = email;
	}

	private User user;
	@OneToOne(cascade = CascadeType.ALL, fetch = FetchType.LAZY)
	@JoinColumn(name = "USERID")
	public User getUser() {
		return user;
	}

	public void setUser(User user) {
		this.user = user;
	}

	@Override
	public boolean equals(Object o) {
		if (this == o)
			return true;
		if (o == null || getClass() != o.getClass())
			return false;
		UserDetails that = (UserDetails) o;
		if (id != that.id)
			return false;
		if (firstname != null ? !firstname.equals(that.firstname)
				: that.firstname != null)
			return false;
		if (lastname != null ? !lastname.equals(that.lastname)
				: that.lastname != null)
			return false;
		return true;
	}

	@Override
	public int hashCode() {
		int result = id;
		result = 31 * result + (firstname != null ?
                                        firstname.hashCode() : 0);

		result = 31 * result + (lastname != null ?
                                        lastname.hashCode() : 0);
		return result;
	}
}
```

`HistoryLog.java` : Cette classe définit une entité HistoryLog.

```
package com.etsmtl.paintball.model;

import java.sql.Timestamp;
import javax.persistence.Basic;
import javax.persistence.Column;
import javax.persistence.Entity;
import javax.persistence.GeneratedValue;
import javax.persistence.Id;
import javax.persistence.JoinColumn;
import javax.persistence.ManyToOne;
import com.etsmtl.paintball.hibernate.dao.BasicEntity;

@javax.persistence.Table(name = "pb_userhistorylog", catalog = "paintBall")
@Entity
public class HistoryLog extends BasicEntity {
	private int id;

	@Column(name = "ID")
	@Id
	@GeneratedValue
	public int getId() {
		return id;
	}

	public void setId(int id) {
		this.id = id;
	}

	private User user;

	@ManyToOne
	@JoinColumn(name = "USERID")
	public User getUser() {
		return user;
	}

	public void setUser(User user) {
		this.user = user;
	}

	private Timestamp loggedat;

	@Column(name = "LOGGEDAT")
	@Basic
	public Timestamp getLoggedat() {
		return loggedat;
	}

	public void setLoggedat(Timestamp loggedat) {
		this.loggedat = loggedat;
	}
```

```
	@Override
	public boolean equals(Object o) {
		if (this == o)
			return true;
		if (o == null || getClass() != o.getClass())
			return false;

		HistoryLog that = (HistoryLog) o;

		if (id != that.id)
			return false;
		if (loggedat != null ? !loggedat.equals(that.loggedat)
				: that.loggedat != null)
			return false;

		return true;
	}

	@Override
	public int hashCode() {
		int result = id;
		result = 31 * result + (loggedat != null ?
					loggedat.hashCode() : 0);
		return result;
	}
}
```

IGenericDAO.java : Cette interface définit le contrat sur les opérations génériques possibles sur chaque entité.

```
package com.etsmtl.PaintBall.hibernate;

import java.util.List;

import com.etsmtl.PaintBall.hibernate.dao.BasicEntity;

public interface  IGenericDAO<T> {

	public void save(T entity);

	public void delete(T entity);

	public List<T> findAll(Class entityClass);

	public T findById(int id, Class entityClass);

	public List<T> findByExample(T entity, Class entityClass);

}
```

GenericDAO.java : Cette classe implémente les opérations génériques possibles sur chaque entité.

```
package com.etsmtl.PaintBall.hibernate.dao;

import java.util.List;
import org.hibernate.Criteria;
import org.hibernate.Session;
import org.hibernate.criterion.Example;

import com.etsmtl.PaintBall.hibernate.HibernateUtil;
import com.etsmtl.PaintBall.hibernate.IGenericDAO;


public abstract class GenericDAO<T extends BasicEntity> implements
		IGenericDAO<T> {

	protected Session session = null;

	public GenericDAO() {
		session = HibernateUtil.getSession();
	}

	public void save(T entity) {
		session.beginTransaction();
		session.saveOrUpdate(entity);
		session.getTransaction().commit();
	}

	public void delete(T entity) {
		session.beginTransaction();
		session.delete(entity);
		session.getTransaction().commit();
	}

	public List<T> findAll(Class entityClass) {
		session.beginTransaction();
		Criteria criteria = session.createCriteria(entityClass);
		List<T> allResults = criteria.list();
		session.getTransaction().commit();
		return allResults;
	}

	public T findById(int id, Class entityClass) {
		session.beginTransaction();
		T result = (T) session.load(entityClass, id);
		session.getTransaction().commit();
		return result;
	}

	public List<T> findByExample(T entity, Class entityClass) {
		session.beginTransaction();
```

```
            Criteria criteria = session.createCriteria(entityClass);
            criteria.add(Example.create(entity));
            List<T> allResults = criteria.list();
            session.getTransaction().commit();
            return allResults;
      }
}
```

UserDAO.java : Cette classe implémente quelques opérations spécifiques à l'entité User.

```
package com.etsmtl.paintball.model.dao;

import java.util.List;

import com.etsmtl.paintball.hibernate.dao.GenericDAO;
import com.etsmtl.paintball.model.IUserDAO;
import com.etsmtl.paintball.model.User;

public class UserDAO extends GenericDAO<User> implements IUserDAO {

      public User getUserByLogin(String username, String password) {
            User user = new User();
            user.setUsername(username);
            user.setPassword(password);
            List<User> users = findByExample(user, User.class);
            User authenticatedUser =
                        (users.size() == 0) ? null : users.get(0);
            return authenticatedUser;
      }

      public User getUserById(int id) {
            User user = new User();
            user.setId(id);
            List<User> users = findByExample(user, User.class);
            User authenticatedUser =
                        (users.size() == 0) ? null : users.get(0);
            return authenticatedUser;
      }

      @Override
      public void delete(User entity) {
            List<User> users = findByExample(entity, User.class);
            for (User user : users) {
                  super.delete(user);
            }
      }
}
```

UserDetailsDAO.java : Cette classe implémente quelques opérations spécifiques à l'entité UserDetails.

```
package com.etsmtl.paintball.model.dao;

import java.util.List;

import com.etsmtl.paintball.hibernate.dao.GenericDAO;
import com.etsmtl.paintball.model.IUserDetailsDAO;
import com.etsmtl.paintball.model.UserDetails;

public class UserDetailsDAO extends GenericDAO<UserDetails> implements
            IUserDetailsDAO {

      @Override
      public void delete(UserDetails entity) {
            List<UserDetails> userDetailsList = findByExample(entity,
                        UserDetails.class);
            for (UserDetails userDetails : userDetailsList) {
                  super.delete(userDetails);
            }
      }
}
```

HistoryLogDAO.java : Cette classe implémente quelques opérations spécifiques à l'entité HistoryLog.

```
package com.etsmtl.paintball.model.dao;

import java.util.List;
import com.etsmtl.paintball.hibernate.dao.GenericDAO;
import com.etsmtl.paintball.model.HistoryLog;
import com.etsmtl.paintball.model.IHistoryLogDAO;
public class HistoryLogDAO extends GenericDAO<HistoryLog> implements
            IHistoryLogDAO {

      @Override
      public void delete(HistoryLog entity) {
            List<HistoryLog> historyLogs = findByExample(
                                    entity, HistoryLog.class);
            for (HistoryLog historyLog : historyLogs) {
                  super.delete(historyLog);
            }
      }
}
```

COUCHE PRESENTATION: Répresente la vue de l'application de Paintball

`index.jsp` : page d'accueil

```
<%@ taglib uri="/WEB-INF/tlds/struts-logic.tld" prefix="logic"%>

<logic:present name="user">
      <%@include file="home_private.jsp"%>
</logic:present>
<logic:notPresent name="user">
      <%@include file="home_public.jsp"%>
</logic:notPresent>
```

`header.jsp` : en-tête de chaque page

```
<%@ taglib uri="/WEB-INF/tlds/struts-bean.tld" prefix="bean"%>
<%@ taglib uri="/WEB-INF/tlds/struts-html.tld" prefix="html"%>
<%@ taglib uri="/WEB-INF/tlds/struts-logic.tld" prefix="logic"%>


<logic:notPresent name="org.apache.struts.action.MESSAGE"
      scope="application">
      <font> ERROR: Application resources not loaded -- check servlet
      container logs for error messages. </font>
</logic:notPresent>

<div id="header" align="center"><img
      src="<%=request.getContextPath()%>/images/logo_ets.jpg"
      title="logo ets" /> <html:link action="/home">
      <bean:message key="header.home" />
</html:link> <html:link page="/pages/contact.jsp">
      <bean:message key="header.contact" />
</html:link> <logic:empty name="language">
      <html:link action="languageSelection.do?language=english">
            <bean:message key="header.language.english" />
      </html:link>
</logic:empty> <logic:equal name="language" value="en">
      <html:link action="languageSelection.do?language=french">
            <bean:message key="header.language.french" />
      </html:link>
</logic:equal> <logic:equal name="language" value="fr">
      <html:link action="languageSelection.do?language=english">
            <bean:message key="header.language.english" />
      </html:link>
</logic:equal>

<p class="content_title"><bean:message key="home.title" /></p>
</div>
```

footer.jsp : pied de page

```
<%@ taglib uri="/WEB-INF/tlds/struts-bean.tld" prefix="bean"%>
<%@ taglib uri="/WEB-INF/tlds/struts-html.tld" prefix="html"%>
<%@ taglib uri="/WEB-INF/tlds/struts-logic.tld" prefix="logic"%>

<div id="footer" align="center"><bean:message
                        key="footer.reserved" /></div>
```

aside_content.jsp : menu à gauche de chaque page

```
<%@ taglib uri="/WEB-INF/tlds/struts-bean.tld" prefix="bean"%>
<%@ taglib uri="/WEB-INF/tlds/struts-html.tld" prefix="html"%>
<%@ taglib uri="/WEB-INF/tlds/struts-logic.tld" prefix="logic"%>

<div id="aside_content"><br>
<br>
<br>
<div id="aside_login"><logic:present name="user">
      <%@include file="../logout.jsp"%>
</logic:present> <logic:notPresent name="user">
      <%@include file="../login.jsp"%>
</logic:notPresent></div>
</div>
```

register.jsp : page d'inscription

```
<%@ taglib uri="/WEB-INF/tlds/struts-bean.tld" prefix="bean"%>
<%@ taglib uri="/WEB-INF/tlds/struts-html.tld" prefix="html"%>
<%@ taglib uri="/WEB-INF/tlds/struts-logic.tld" prefix="logic"%>
<%@ taglib uri="/WEB-INF/tlds/struts-tiles.tld" prefix="tiles"%>

<html:html>
<head>
<title><bean:message key="register.title" /></title>
<link rel="stylesheet" type="text/css" href="css/index.css" />
</head>
<body>

<%@include file="common/header.jsp"%>

<!-- content -->
<table align="center" border="0">
      <tr>
            <td valign="top"><%@include file="common/aside_content.jsp"%>
            </td>
            <td>
            <div id="primary_content">
            <table id="register_form" width="600px">
                  <tr>
```

```
                        <td>
                        <h3><bean:message key="register.heading" /></h3>
                        <p><span style="color: #000000;">
                                        <strong><bean:message
                              key="register.description"
                                        /></strong></span></p>
                        <html:form action="/register.do" method="post">
                              <p><bean:message
                                    key="registerForm.firstname" /></p>
                              <html:text property="firstname"
                                    size="40"></html:text>
                              <div style="color: red"><html:errors
                                    property="register.firstname" /></div>
                              <p><bean:message
                                    key="registerForm.lastname" /></p>
                              <html:text property="lastname"
                                    size="40"></html:text>
                              <div style="color: red"><html:errors
                                    property="register.lastname" /></div>
                              <p><bean:message
                                    key="registerForm.email" /></p>
                              <html:text property="email"
                                    size="40"></html:text>
                              <div style="color: red"><html:errors
                                    property="register.email" /></div>
                              <p><bean:message
                                    key="registerForm.username" /></p>
                              <html:text property="registerUsername"
                                          size="40"></html:text>
                              <div style="color: red"><html:errors
                                    property="register.username" /></div>
                              <p><bean:message
                                    key="registerForm.password" /></p>
                              <html:password property="registerPassword"
                                          size="40"></html:password>
                              <div style="color: red"><html:errors
                                    property="register.password" /></div>
                              <p><input type="submit"
                                    value="<bean:message
                                    key="register.title" />" />
                        </html:form>
                        </td>
                  </tr>
            </table>
            </div>
            </td>
      </tr>
</table>

<%@include file="common/footer.jsp"%>
</body>
</html:html>
```

`contact.jsp` : page de contact

```
<%@ taglib uri="/WEB-INF/tlds/struts-bean.tld" prefix="bean"%>
<%@ taglib uri="/WEB-INF/tlds/struts-html.tld" prefix="html"%>
<%@ taglib uri="/WEB-INF/tlds/struts-logic.tld" prefix="logic"%>
<%@ taglib uri="/WEB-INF/tlds/struts-tiles.tld" prefix="tiles"%>

<html:html>
<head>
<title><bean:message key="contact.title" /></title>
<link rel="stylesheet" type="text/css" href="css/index.css" />
</head>
<body>

<%@include file="common/header.jsp"%>

<!-- content -->
<table align="center" border="0">
      <tr>
            <td valign="top"><%@include file="common/aside_content.jsp"%>
            </td>
            <td>
            <div id="primary_content">
            <table id="contact_form" width="600px">
                  <tr>
                        <td>
                        <h3><bean:message key="contact.heading" /></h3>
                        <p><span style="color: #000000;">
                        <strong><bean:message key="contact.description"
                              /></strong></span></p>
                        <html:form action="/contact.do" method="post">
                              <p><bean:message key="contact.name" /></p>
                              <html:text property="fullname"
                                    size="40"></html:text>
                              <div style="color: red"><html:errors
                                    property="contact.fullname" /></div>
                              <p><bean:message key="contact.email" /></p>
                              <html:text property="email"
                                    size="40"></html:text>
                              <div style="color: red">
                        <html:errors property="contact.email" /></div>
                              <p><bean:message key="contact.subject" /></p>
                              <html:text property="subject"
                                          size="40"></html:text>
                              <div style="color: red"><html:errors
                                    property="contact.subject" /></div>

                              <p><bean:message key="contact.message" /></p>
                              <html:textarea property="message" cols="40"
                                          rows="10"></html:textarea>
                              <div style="color: red"><html:errors
                                    property="contact.message" /></div>
                              <p><input type="submit"
                                    value="<bean:message
                                    key="contact.send" />" />
                        </html:form> </td>
                  </tr>
```

```
            </table>
            </div>
            </td>
      </tr>
</table>

<%@include file="common/footer.jsp"%>
</body>
</html:html>
```

command.jsp : page d'inclusion pour le contrôle du Paintball

```
<%@ taglib uri="/WEB-INF/tlds/struts-bean.tld" prefix="bean"%>
<%@ taglib uri="/WEB-INF/tlds/struts-html.tld" prefix="html"%>
<%@ taglib uri="/WEB-INF/tlds/struts-logic.tld" prefix="logic"%>
<%@ taglib uri="/WEB-INF/tlds/struts-tiles.tld" prefix="tiles"%>
<table style="border-width: 1">
      <form action="<%=request.getContextPath()%>/command.do"
            method="post"">
      <input type="hidden" id="command" name="command"
            value="<bean:message key="command.operational.command"/>" />
      <tr>
            <td><bean:message key="command.operational.label" /></td>
            <td><input type="submit"
                  value="<bean:message key="command.status"/>" /></td>
      </tr>
      </form>
      <form action="<%=request.getContextPath()%>/command.do"
            method="post"">
      <input type="hidden" id="command" name="command"
            value="<bean:message key="command.shootOneBall.command"/>" />
      <tr>
            <td><bean:message key="command.shootOneBall.label" /></td>
            <td><input type="submit"
                  value="<bean:message key="command.shoot"/>" /></td>
      </tr>
      </form>
      <form action="<%=request.getContextPath()%>/command.do"
            method="post"">
      <input type="hidden" id="command" name="command"
          value="<bean:message key="command.shootThreeBalls.command"/>" />
      <tr>
            <td><bean:message key="command.shootThreeBalls.label" /></td>
            <td><input type="submit"
                  value="<bean:message key="command.shoot"/>" /></td>
      </tr>
      </form>
      <form action="<%=request.getContextPath()%>/command.do"
            method="post"">
      <input type="hidden" id="command" name="command"
            value="<bean:message key="command.setPosition.command"/>" />
      <tr>
            <td><bean:message key="command.setPosition.label" />
            <td><input type="submit"
```

```
                  value="<bean:message key="command.submit"/>" /></td>
      </tr>
      <tr>
            <td>X<input name="x" type="text" value="" size="1" />
        Y <input name="y" type="text" value="" size="1" /></td>
      </tr>
      </form>
</table>
```

`history.jsp` : rapport sur l'historique de connexion

```
<%@ taglib uri="/WEB-INF/tlds/struts-bean.tld" prefix="bean"%>
<%@ taglib uri="/WEB-INF/tlds/struts-html.tld" prefix="html"%>
<%@ taglib uri="/WEB-INF/tlds/struts-logic.tld" prefix="logic"%>
<%@ taglib uri="/WEB-INF/tlds/struts-tiles.tld" prefix="tiles"%>
<%@taglib prefix="velocity"
      uri="http://velocity.apache.org/velocity-view"%>
<html:html>
<head>
<title><bean:message key="history.title" /></title>
<link rel="stylesheet" type="text/css" href="css/index.css" />
</head>
<body>

<%@include file="common/header.jsp"%>

<!-- content -->
<table align="center" border="0">
      <tr>
            <td valign="top"><%@include file="common/aside_content.jsp"%>
            </td>
            <td>
            <div id="primary_content">
            <table id="register_form" width="600px">
                  <tr>
                        <td>
                        <h3><bean:message key="history.heading" /></h3>
                        <p><span style="color: #000000;">
                        <strong><bean:message key="history.description"
                              /></strong></span></p>


                        <table align="center">
                              <tr>
                                    <th><bean:message
                                          key="history.time" /></th>
                              </tr>
                              <velocity:view>
                                    #foreach ($history in $user.historyLog)
                              <tr>
                                    <td>$history.loggedat</td>
                              </tr>
                                    #end
                              </velocity:view>
```

```
                        </table>
                        </td>
                    </tr>
            </table>
            </div>
            </td>
      </tr>
</table>

<%@include file="common/footer.jsp"%>
</body>
</html:html>
```

`login.jsp` : page d'authentification

```
<%@ taglib uri="/WEB-INF/tlds/struts-bean.tld" prefix="bean"%>
<%@ taglib uri="/WEB-INF/tlds/struts-html.tld" prefix="html"%>
<%@ taglib uri="/WEB-INF/tlds/struts-logic.tld" prefix="logic"%>
<%@ taglib uri="/WEB-INF/tlds/struts-tiles.tld" prefix="tiles"%>

<form name="loginForm" action="<%=request.getContextPath()%>/login.do"
      method="post">
<fieldset><legend> <bean:message key="login.submit" /></legend>

<div><label for="username"><bean:message
            key="login.username" /></label></div>
<div><input type="text" id="username" name="username" /></div>

<div><label for="password"><bean:message
            key="login.password" /></label></div>
<div><input type="password" id="password" name="password" /></div>

<input type="button" id="register"
      value="<bean:message key="login.register"/>"
      onclick="window.location='<%=request.getContextPath()%>
                              /pages/register.jsp' " />
<input type="submit" id="login"
      value="<bean:message key="login.submit"/>" />

<div style="color: red"><html:errors property="login.username" /></div>
<div style="color: red"><html:errors property="login.password" /></div>
<div style="color: red"><html:errors
      property="login.usernamePassword" /></div>
</fieldset>
</form>
```

`logout.jsp` : page de deconnexion

```
<%@ page isELIgnored="false"%>
<%@ taglib uri="/WEB-INF/tlds/struts-bean.tld" prefix="bean"%>
<%@ taglib uri="/WEB-INF/tlds/struts-logic.tld" prefix="logic"%>
```

```
<logic:present name="user">
	<form name="loginForm" method="post"
		action="${pageContext.request.contextPath}/logout.do" >
	<fieldset><legend><bean:message key="logout.title" /></legend>
		<input type="submit" id="login"
			value="<bean:message key="logout.title" />" /></fieldset>
	</form>
	<form name="history" action="#">
	<fieldset><legend>${sessionScope.user.userDetails.firstname}
	${sessionScope.user.userDetails.lastname}</legend> <a
		href="${pageContext.request.contextPath}/pages/history.jsp">
		<bean:message key="history.title" /></a><br>
	<a href="${pageContext.request.contextPath}/home.do">
	<bean:message key="robot.commandTitle" /></a></fieldset>
	</form>
	<fieldset><legend><bean:message key="logout.robot.status" /></legend>
	<table bgcolor="red">
		<tr>
			<td width="100px"> </td>
		</tr>
	</table>
	</fieldset>
</logic:present>
```

`home_public.jsp` : page d'inclusion servant de page d'accueil pour les non authentifiés

```
<%@ taglib uri="/WEB-INF/tlds/struts-bean.tld" prefix="bean"%>
<%@ taglib uri="/WEB-INF/tlds/struts-html.tld" prefix="html"%>
<%@ taglib uri="/WEB-INF/tlds/struts-logic.tld" prefix="logic"%>
<%@ taglib uri="/WEB-INF/tlds/struts-tiles.tld" prefix="tiles"%>

<html:html>
<head>
<title><bean:message key="welcome.title" /></title>
<link rel="stylesheet" type="text/css" href="css/index.css" />
</head>
<body>

<%@include file="common/header.jsp"%>

<!-- content -->
<table align="center" border="0">
	<tr>
		<td valign="top"><%@include file="common/aside_content.jsp"%>
		</td>
		<td>
		<div id="primary_content">
		<h3><bean:message key="welcome.heading" /></h3>
		<p><bean:message key="welcome.message" /></p>

		<img src="images/avr32_atmega.png" title="" width="300px"
			width="400px" align="top" /></div>
		</td>
```

```
      </tr>
</table>

<%@include file="common/footer.jsp"%>

</body>
</html:html>
```

home_private.jsp : page d'inclusion servant de page d'accueil pour les authentifiés

```
<%@ taglib uri="/WEB-INF/tlds/struts-bean.tld" prefix="bean"%>
<%@ taglib uri="/WEB-INF/tlds/struts-html.tld" prefix="html"%>
<%@ taglib uri="/WEB-INF/tlds/struts-logic.tld" prefix="logic"%>
<%@ taglib uri="/WEB-INF/tlds/struts-tiles.tld" prefix="tiles"%>

<logic:present name="user">
      <html:html>
      <head>
      <title><bean:message key="welcome.title" /></title>
      <link rel="stylesheet" type="text/css" href="css/index.css" />
      </head>
      <body>
      <%@include file="common/header.jsp"%>
      <!-- content -->
      <table align="center" border="0">
            <tr>
                  <td valign="top">
                        <%@include file="common/aside_content.jsp"%>
                  </td>
                  <td>
                  <div id="primary_content">
                  <h3><bean:message key="welcome.heading" /></h3>
                  <p><bean:message key="welcome.message.private" /></p>
                  <table>
                        <tr>
                              <td><img src="images/IMG_0074.JPG"
                                    title="" width="300px"
                                    width="400px" align="top" /></td>
                              <td><logic:present name="user">
                                    <%@include file="command.jsp"%>
                              </logic:present></td>
                        </tr>
                  </table>
                  </div>
                  </td>
            </tr>
      </table>
      <%@include file="common/footer.jsp"%>
      </body>
      </html:html>
</logic:present>
```

TESTS UNITAIRES : Ces tests garantissent le bon de l'application avant le déploiement

Couche d'affaires d'accès au Paintball et d'envoi de courriel

`PaintballConfigurationTest.java` : Teste la bonne configuration du paintball

```
package com.etsmtl.paintball.business;

import java.util.Map;

import junit.framework.TestCase;

import org.junit.Test;
import org.springframework.context.support.ClassPathXmlApplicationContext;

public class PaintballConfigurationTest extends TestCase {

	PaintballConfiguration paintballConfiguration;

	public PaintballConfigurationTest() {

	}

	@Override
	protected void setUp() throws Exception {
		super.setUp();
		ClassPathXmlApplicationContext context = new
ClassPathXmlApplicationContext(
				"spring-paintball.xml");
		paintballConfiguration = (PaintballConfiguration) context
				.getBean("configuration");
	}

	@Test
	public void testgetPaintballCommands() {
		Map<String, PaintballCommand> actual = paintballConfiguration
				.getPaintballCommands();
		assertTrue(actual.get("position") instanceof
PaintBallPositionCommand);
		assertTrue(actual.get("status") instanceof
PaintBallStatusCommand);
		assertTrue(actual.get("shootOne") instanceof
PaintBallShootOneCommand);
		assertTrue(actual.get("shootThree") instanceof
PaintBallShootThreeCommand);
	}
```

```
	@Test
	public void testgetPaintBallController() {
		PaintBallController actual = paintballConfiguration
				.getPaintBallController();
		assertNotNull(actual);
	}

	@Test
	public void testgetSerialPort() {
		String actual = paintballConfiguration.getSerialPort();
		assertNotNull(actual);
	}

	@Test
	public void testgetMicrocontrollerCommands() {
		Map<String, String> actual = paintballConfiguration
				.getMicrocontrollerCommands();
		assertNotNull(actual.get("position"));
		assertNotNull(actual.get("status"));
		assertNotNull(actual.get("shootOne"));
		assertNotNull(actual.get("shootThree"));
	}
}
```

`MicrocontrollerTest.java` : Teste le bon fonctionnement du microcontrôleur

```
package com.etsmtl.paintball.business;

import javax.comm.CommPortIdentifier;
import javax.comm.SerialPort;

import junit.framework.TestCase;

import org.junit.Test;
import org.springframework.context.support.ClassPathXmlApplicationContext;

public class MicrocontrollerTest extends TestCase {

	Microcontroller microcontroller;

	public MicrocontrollerTest() {

	}

	@Override
	protected void setUp() throws Exception {
		super.setUp();
		ClassPathXmlApplicationContext context = new
ClassPathXmlApplicationContext(
				"spring-paintball.xml");
		microcontroller = (Microcontroller)
context.getBean("microcontroller");
	}
```

```
	@Test
	public void testgetPaintball() {
		Paintball actual = microcontroller.getPaintball();
		assertNotNull(actual);
	}

	@Test
	public void testgetSerialPort() {
		SerialPort actual = microcontroller.getSerialPort();
		assertNotNull(actual);
	}

	@Test
	public void testgetSerialPortIdentifier() {
		CommPortIdentifier actual =
microcontroller.getSerialPortIdentifier();
		assertNotNull(actual);
	}

	@Override
	protected void tearDown() throws Exception {
		microcontroller.closeConnection();
	}
}
```

SendMailTest.java : Teste le bon fonctionnement de la classe utilisée pour envoyer des courriels

```
package com.etsmtl.paintball.business;

import java.util.Properties;

import org.junit.Test;

import junit.framework.TestCase;

import org.junit.Test;
import org.springframework.context.support.ClassPathXmlApplicationContext;

public class SendMailTest extends TestCase {

	SendMail sendMail;

	public SendMailTest() {
	}

	@Override
	protected void setUp() throws Exception {
		super.setUp();
		ClassPathXmlApplicationContext context = new
ClassPathXmlApplicationContext(
				"spring-paintball.xml");
```

```
		sendMail = (SendMail) context.getBean("sendMail");
	}

	@Test
	public void testgetTo() {
		String actual = sendMail.getTo();
		assertNotNull(actual);
	}

	@Test
	public void testgetProperties() {
		Properties actual = sendMail.getProperties();
		assertNotNull(actual.getProperty("mail.smtp.host"));
		assertNotNull(actual.getProperty("mail.smtp.port"));
	}
}
```

Couche d'affaires d'accès aux données

`UserDAOTest.java` : Teste le bon fonctionnement de la persistance (CRUD) d'un utilisateur

```
package com.etsmtl.paintball.model.dao;

import java.util.List;

import junit.framework.TestCase;

import org.junit.Test;

import com.etsmtl.paintball.hibernate.HibernateUtil;
import com.etsmtl.paintball.model.User;

public class UserDAOTest extends TestCase {

	int userId;

	@Override
	protected void setUp() throws Exception {
		super.setUp();
		HibernateUtil.getSession();
	}

	@Test
	public void testSave() {
		User user = new User();
		user.setUsername("username");
		user.setPassword("password");

		UserDAO userDAO = new UserDAO();
		userDAO.save(user);
```

```
		List<User> list = userDAO.findByExample(user, User.class);
		assertEquals(1, list.size());
	}

	@Test
	public void testFindByExample() {
		User user = new User();
		user.setUsername("username");
		user.setPassword("password");
		UserDAO userDAO = new UserDAO();

		List<User> list = userDAO.findByExample(user, User.class);
		assertEquals(1, list.size());
	}


	@Test
	public void testGetUserByLogin() {
		UserDAO userDAO = new UserDAO();
		User user = userDAO.getUserByLogin("username", "password");
		userId = user.getId();
		assertNotNull(user);
	}


	@Test
	public void testGetUserById() {
		UserDAO userDAO = new UserDAO();
		User user = userDAO.getUserById(userId);
		assertNotNull(user);
	}


	@Test
	public void testDelete() {
		User user = new User();
		user.setUsername("username");
		user.setPassword("password");
		UserDAO userDAO = new UserDAO();
		userDAO.delete(user);

		List<User> list = userDAO.findByExample(user, User.class);
		assertEquals(0, list.size());
	}

	@Override
	protected void tearDown() throws Exception {
		super.tearDown();
	}
}
```

`UserDetailsDAOTest.java` : Teste le bon fonctionnement de la persistance (CRUD) des informations d'un utilisateur

```
package com.etsmtl.paintball.model.dao;

import java.util.List;

import junit.framework.TestCase;

import org.junit.Test;

import com.etsmtl.paintball.hibernate.HibernateUtil;
import com.etsmtl.paintball.model.User;
import com.etsmtl.paintball.model.UserDetails;


public class UserDetailsDAOTest extends TestCase {

	@Override
	protected void setUp() throws Exception {
		super.setUp();
		HibernateUtil.getSession();
	}


	@Test
	public void testSave() {
		User user = new User();
		user.setUsername("username");
		user.setPassword("password");

		UserDAO userDAO = new UserDAO();
		userDAO.save(user);

		UserDetails userDetails = new UserDetails();
		userDetails.setUser(user);
		userDetails.setFirstname("firstname");
		userDetails.setLastname("lastname");
		userDetails.setEmail("email@email.com");

		UserDetailsDAO userDetailsDAO = new UserDetailsDAO();
		userDetailsDAO.save(userDetails);

		List<UserDetails> list =
userDetailsDAO.findByExample(userDetails,
				UserDetails.class);
		assertEquals(1, list.size());
	}
```

```
	@Test
	public void testFindByExample() {
		UserDetails userDetails = new UserDetails();
		userDetails.setFirstname("firstname");
		userDetails.setLastname("lastname");
		userDetails.setEmail("email@email.com");

		UserDetailsDAO userDetailsDAO = new UserDetailsDAO();

		List<UserDetails> list =
userDetailsDAO.findByExample(userDetails,
				UserDetails.class);
		assertEquals(1, list.size());
	}


	@Test
	public void testDelete() {
		UserDetails userDetails = new UserDetails();
		userDetails.setFirstname("firstname");
		userDetails.setLastname("lastname");
		userDetails.setEmail("email@email.com");
		UserDetailsDAO userDetailsDAO = new UserDetailsDAO();
		userDetailsDAO.delete(userDetails);


		List<UserDetails> list =
userDetailsDAO.findByExample(userDetails,
				UserDetails.class);
		assertEquals(0, list.size());

		User user = new User();
		user.setUsername("username");
		user.setPassword("password");
		UserDAO userDAO = new UserDAO();
		userDAO.delete(user);

		List<User> users = userDAO.findByExample(user, User.class);
		assertEquals(0, users.size());
	}

	@Override
	protected void tearDown() throws Exception {
		super.tearDown();
	}

}
```

HistoryLogDAOTest.java : Teste le bon fonctionnement de la persistance (CRUD) des historiques

```
package com.etsmtl.paintball.model.dao;

import java.sql.Timestamp;
import java.util.Date;
import java.util.List;

import junit.framework.TestCase;

import org.junit.Test;

import com.etsmtl.paintball.hibernate.HibernateUtil;
import com.etsmtl.paintball.model.HistoryLog;
import com.etsmtl.paintball.model.User;

public class HistoryLogDAOTest extends TestCase {
	Timestamp timestamp = new Timestamp((new Date()).getTime());

	@Override
	protected void setUp() throws Exception {
		super.setUp();
		HibernateUtil.getSession();
	}

	@Test
	public void testSave() {
		User user = new User();
		user.setUsername("username");
		user.setPassword("password");

		UserDAO userDAO = new UserDAO();
		userDAO.save(user);

		HistoryLog historyLog = new HistoryLog();
		historyLog.setUser(user);
		historyLog.setLoggedat(timestamp);

		HistoryLogDAO historyLogDAO = new HistoryLogDAO();
		historyLogDAO.save(historyLog);

		List<HistoryLog> list = historyLogDAO.findByExample(historyLog,
				HistoryLog.class);
		assertEquals(1, list.size());
	}

	@Test
	public void testFindByExample() {
		HistoryLog historyLog = new HistoryLog();
		historyLog.setLoggedat(timestamp);

		HistoryLogDAO historyLogDAO = new HistoryLogDAO();
```

```
		List<HistoryLog> list = historyLogDAO.findByExample(historyLog,
				HistoryLog.class);
		assertEquals(1, list.size());
	}


	@Test
	public void testDelete() {
		HistoryLog historyLog = new HistoryLog();
		historyLog.setLoggedat(timestamp);

		HistoryLogDAO historyLogDAO = new HistoryLogDAO();
		historyLogDAO.delete(historyLog);

		List<HistoryLog> list = historyLogDAO.findByExample(historyLog,
				HistoryLog.class);
		assertEquals(0, list.size());

		User user = new User();
		user.setUsername("username");
		user.setPassword("password");
		UserDAO userDAO = new UserDAO();
		userDAO.delete(user);

		List<User> users = userDAO.findByExample(user, User.class);
		assertEquals(0, users.size());
	}

	@Override
	protected void tearDown() throws Exception {
		super.tearDown();
	}
}
```

BUIILD : Génération du « war » à copier dans le répertoire web-apps du serveur web.

`build.xml` : fichier de build

```
<?xml version="1.0" ?>
<project name="paintball" default="war">

	<path id="compile.classpath">
		<fileset dir="WebContent/WEB-INF/lib">
			<include name="*.jar"/>
		</fileset>
	</path>

	<target name="init">
		<mkdir dir="build/classes"/>
		<mkdir dir="dist" />
	</target>

	<target name="compile" depends="init" >
		<javac destdir="build/classes" debug="true" srcdir="src">
			<classpath refid="compile.classpath"/>
		</javac>
	</target>

	<target name="war" depends="compile">
		<war destfile="dist/paintball.war"
			webxml="WebContent/WEB-INF/web.xml">
			<fileset dir="WebContent"/>
			<lib dir="WebContent/WEB-INF/lib"/>
			<classes dir="build/classes"/>
		</war>
	</target>

	<target name="clean">
		<delete dir="dist"/>
		<delete dir="build" />
	</target>
</projet>
```

BIBLIOGRAPHIE

[1] Gregory Dudek, Michael Jenkin (2, 5, 7). 2000. Computational principles of mobile robotics, Cambridge University Press. 280 p.

[2] Frank L. Lewis, Darren M. Dawson, Chaouki T. Abdallah (520). 2004. Robot Manipulator Control Theory and Practice, Marcel Dekker, Inc., 607 p.

[3] Nehmzow, Ulrich (7). 2003. Mobile Robotics: A Pratical Introduction 2nd Edition, Springer. 279 p.

[4] Duval, Thomas (4). 2003. Robotique mobile, 68HC11 et OS dédié, Éditions techniques et scientifiques françaises. 215 p.

[5] «EVK1100». 2011. <http://www.atmel.com/dyn/products/tools_card.asp?tool_id=4114>.

[6] Steve Rowe et Christopher R. Wagner (2005). An Introduction to the Joint Architecture for Unmanned Systems (JAUS). 2005 Titre du journal. http://www.openskies.net/papers/07F-SIW-089%20Introduction%20to%20JAUS.pdf

[7] Darrel Riekhof, Keith Fligg (2000). How to Control A Robot Over the Internet. 2000. http://java.sys-con.com/node/36409

[8] Ferat Sahin, Pushkin KachRoo (137). 2007. Pratical and Experimental Robotics, CRC Press. 439 p.

[9] Thomas Braunl (138). 2003. Embedded Robotics. 434 p.

[10] « Modèle-Vue-Contrôleur ». <http://fr.wikipedia.org/wiki/Modèle-Vue-Contrôleur#En_Java >.