

**Département de génie logiciel et des T.I.**

Rapport de Laboratoire

|  |  |
| --- | --- |
| **Numéro du laboratoire** | 04 |
| **Nom du laboratoire** | Projet de session |
| **Étudiant(s)** | Jean-Simon Bonin  Xavier Duval  Martin Gingras |
| **Code(s) permanent(s)** | BONJ02098701  DUVX20048607  GINM10108708 |
| **Cours** | LOG 430 |
| **Session** | Hiver 2012 |
| **Groupe** | 01 |
| **Chargé(e) de laboratoire** | Samir Djeffal |
| **Date** | 16 avril 2012 |

Table des matières

[Introduction 5](#_Toc322013890)

[Développement 6](#_Toc322013891)

[Discussion 7](#_Toc322013892)

[Conclusion 8](#_Toc322013893)

**Liste des tableaux**

**No table of figures entries found.**

**Liste des figures**

**No table of figures entries found.**

# Introduction

Ce laboratoire a pour objectif de nous permettre d’effectuer la conception d’une architecture d’un système en se basant sur un document de vision, un SRS et des cas d’utilisation tout en suivant l’ensemble des méthodologies vues dans le cours d’architecture logicielle. La réalisation des exigences de ce laboratoire mettra en pratique ces diverses méthodologies et servira de récapitulatif complet pour ce cours.

La suite de ce rapport contient trois sections. D’abord, la section principale du document, le développement, contient l’ensemble de notre conception architecturale. La deuxième section est une discussion comparant les quatre laboratoires sur lesquels nous avons eu à travailler cette session et faisant état de notre point de vue sur ce quatrième laboratoire. Une conclusion vient clore le document en faisant un retour sur les objectifs du laboratoire.

# Développement

**Rappel de la nature et la mission commerciale du système**

Le système est un client de télémétrie qui communique par l’entremise du protocole JAUS et qui gère un ensemble de widgets, perspectives et alertes. Le système permet de créer des widgets qui présenteront les données acquis par les différents senseurs externes au client qui seront interrogés à l’aide de la librairie libJAUS. De plus, un ensemble de widgets dans une configuration donnée peut être enregistré sous forme de perspective. Il y a donc des perspectives qui peuvent être créé pour chaque club étudiant ou utilisateur du client. Le système permet aussi la création d’alertes sur certaines valeurs des senseurs.

La mission du système est de profiter non seulement pour le club étudiant SONIA mais aussi pour tout autre club étudiant pour autant qu’il respecte le protocole de JAUS. De plus, le système doit être multiplateforme et être développé en JAVA.

L’architecture de notre système est déterminée par certains attributs de qualité plus dominants soit la maintenabilité, l’interopérabilité, l’utilisabilité et la portabilité.

**Présentation de l’architecture et des approches architecturales**

Pour ce qui est de l’approche architecturale, nous avons utilisé certaines tactiques afin de bien soutenir nos attributs de qualités.

**Maintenabilité**

L’architecture proposée utilise le patron architecturale MVC afin de favoriser la maintenabilité du système. Il y a donc séparation de la logique d’affaire des données et des éléments lié à la vue du client. Dans l’architecture, le seul point d’interaction avec la librairie ce produit dans le module de « persistence » soit dans la classe « SensorsData ». De cette façon i sera possible de facilement remplacer la librairie puisqu’il n’y aura qu’un seul point de contact avec celle-ci à vérifier.

**Utilisabilité**

Pour l’utilisabilité, on permet à l’utilisateur de créer ses propres perspectives et de les enregistrés.

**Portabilité**

Pour s’assurer de profiter d’une grande portabilité et de rendre le système compatible pour plusieurs types de véhicules autonomes, on se sert d’une seule interface afin de communiquer avec les capteurs et périphériques externe soit la librairie JAUS.

**Interopérabilité**

Vue de type modules - décomposition:

****

|  |  |
| --- | --- |
| **Module** | LibJAUS |
| **Responsabilités** | Ce module est responsable de la communication entre les capteurs et les autres modules du système. |
| **Cas d’utilisations supporté** |  |
| **Interfaces** | Interface qui fournit des services qui permet d’obtenir les valeurs des différents capteurs et périphériques externe. |
| **Scénarios supporté** |  |
| **Contraintes** | Communication à l’aide du protocole JAUS |
| **Données produit** | Retourne la valeur du capteur selon la requête. |
| **Données requis** | Doit connaître le capteur voulu et le type de valeur demandé. |

|  |  |
| --- | --- |
| **Module** | LiveUV.Persistence |
| **Responsabilités** | Ce module est responsable de conserver les données des capteurs acquérir par l’intermédiaire de la librairie JAUS et de conserver les préférences des utilisateurs du client. |
| **Cas d’utilisations supporté** |  |
| **Interfaces** | Fournit le service pour accéder aux données |
| **Scénarios supporté** | U1, U5, I3, P2 |
| **Contraintes** | Utilisation des librairies de communication JAUS (module libJAUS), Langage de programmation JAVA. |
| **Données produit** | Ce module produit les données pour le module LiveUV.Controller |
| **Données requis** | Ce module requiert la librairie JAUS afin de communiquer avec les capteurs pour avoir les données. |

|  |  |
| --- | --- |
| **Module** | LiveUV.Controller.WidgetManager |
| **Responsabilités** | Ce module est responsable de gérer les widget du client de télémétrie. |
| **Cas d’utilisations supporté** | CU01, CU03 |
| **Interfaces** | Fournit les services pour gérer les widgets. |
| **Scénarios supporté** | M1, M2, M3, U2, I3, P1 |
| **Contraintes** | Langage de programmation JAVA. |
| **Données produit** | Ce module produit les widgets à présenter dans le module LiveUV.View |
| **Données requis** | Ce module à besoin des données du module LiveUV.Persistence. |

|  |  |
| --- | --- |
| **Module** | LiveUV.Controller.PerspectiveManager |
| **Responsabilités** | Ce module est responsable de gérer les perspectives, soit l’ensemble des widgets et leur positionnement dans l’application. |
| **Cas d’utilisations supporté** | CU02, CU05, CU06, CU07, CU08, CU09. CU12 |
| **Interfaces** | Fournit les services pour gérer les perspectives. |
| **Scénarios supporté** | I3 |
| **Contraintes** | Langage de programmation JAVA. |
| **Données produit** | Ce module produit les perspectives à présenter dans le module LiveUV.View |
| **Données requis** | Ce module a besoin des données du module LiveUV.Persistence. |

|  |  |
| --- | --- |
| **Module** | LiveUV.Controller.AlerteManager |
| **Responsabilités** | Ce module est responsable de gérer les alertes du client de télémétrie. |
| **Cas d’utilisations supporté** | CU10 |
| **Interfaces** | Fournit le service pour créer des alertes. |
| **Scénarios supporté** | I3 |
| **Contraintes** | Langage de programmation JAVA. |
| **Données produit** | Ce module produit les alertes à présenter dans le module LiveUV.View au moment de leur déclenchement. |
| **Données requis** | Ce module a besoin des données du module LiveUV.Persistence. |

**Analyse des approches architecturales**

**Arbre d’utilité**

****

**Scénarios de qualité**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Scénario #** | U1 – La perspective peut être changée en moins de 5 minutes | | | |
| **Attribut** | Utilisabilité | | | |
| **Environnement** | Run time | | | |
| **Stimulus** | Souhaite modifier la perspective actuelle des widgets dans le client | | | |
| **Réponse** | La perspective est changée en moins de 5 minutes | | | |
| **Décision architectural** | | **Risque / Non-risque** | **Sensibilité** | **Compromis** |
| Perspective sont générique | | NR1 |  |  |
| Séparation des éléments de vue de la logique d’affaire | |  |  |  |
| **Raisonnement** | | Les perspectives sont génériques pour permettre de créer des types de perspectives spécifiques à un club étudiant ou à un utilisateur. | | |
| **Diagramme architectural** | | Voir diagramme de la vue de décomposition | | |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Scénario #** | M4 – La mise à jour de l’interface libJAUS nécessite moins que 1 mois par 1 personne | | | |
| **Attribut** | Maintenabilité | | | |
| **Environnement** | Fonctionnement dégradé | | | |
| **Stimulus** | Besoin de mettre à jour la librairie JAUS | | | |
| **Réponse** | Les autres modules ne dépendant pas de la librairie demeurent inaffectés | | | |
| **Décision architectural** | | **Risque / Non-risque** | **Sensibilité** | **Compromis** |
| Architecture modulaire | |  |  |  |
| Interface unique utilisant la librairie | |  |  |  |
| **Raisonnement** | |  | | |
| **Diagramme architectural** | | Voir diagramme de la vue de décomposition | | |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Scénario #** | M1 – La nouvelle fonctionnalité est codée en moins de 4 heures | | | |
| **Attribut** | Maintenabilité | | | |
| **Environnement** | Mode d’opération normal | | | |
| **Stimulus** | Le programmeur veut ajouter une nouvelle fonctionnalité | | | |
| **Réponse** | La fonctionnalité est codée et fonctionne adéquatement | | | |
| **Décision architectural** | | **Risque / Non-risque** | **Sensibilité** | **Compromis** |
| Architecture en couche | | R1 | S1 | C1 |
| Programmation modulaire en Java | | NR1 |  |  |
| Séparation de la persistance des données et des contrôleurs | | R2 |  | C2 |
| **Raisonnement** | | Pour faciliter l’ajout de nouvelles fonctionnalités, le système doit être le plus possible extensible et modulaire. Dans l’architecture du système, la séparation des éléments logique de code est un moyen indispensable à la maintenabilité. | | |
| **Diagramme architectural** | |  | | |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Scénario #** | U2 – L’ajout d’un widget se fait en moins de 6 clics de souris | | | |
| **Attribut** | Utilisabilité | | | |
| **Environnement** | Mode d’opération normal | | | |
| **Stimulus** | Le programmeur veut ajouter une widget | | | |
| **Réponse** | Le widget est ajouté sans problème | | | |
| **Décision architectural** | | **Risque / Non-risque** | **Sensibilité** | **Compromis** |
| Conception simple de l’interface. | | R1 | S1 | C1 |
| Module de gestion des widgets. | | NR1 | S2 | C2 |
|  | |  |  |  |
| **Raisonnement** | | Pour que l’ajout d’un widget soit le plus simple et efficace possible, il faut que l’interface soit simple et que la gestion widgets dans le code soit bien implémentée. | | |
| **Diagramme architectural** | |  | | |

Tableau des risques

|  |  |  |
| --- | --- | --- |
| **Scénarios** | **Risque** | **Description** |
| M1 | R1 |  |
| R2 |  |
| M4 |  |  |
| U1 | R1 |  |
| U2 |  |  |
|  |  |  |

Tableau des non-risques

|  |  |  |
| --- | --- | --- |
| **Scénarios** | **Non-risque** | **Description** |
| M1 | NR1 |  |
| M4 |  |  |
| U1 | NR1 | Avoir les perspectives génériques permet de créer plusieurs types de perspective spécifique pour plusieurs utilisateurs et donc de sauver tu temps dans le changement des perspectives en utilisant d’autres configurations existantes. |
| U2 | NR1 |  |
|  |  |  |

Tableau des sensibilités

|  |  |  |
| --- | --- | --- |
| **Scénarios** | **Sensibilités** | **Description** |
| M1 | S1 |  |
| M4 |  |  |
| U1 |  |  |
| U2 |  |  |

Tableau des compromis

|  |  |  |
| --- | --- | --- |
| **Scénarios** | **Compromis** | **Description** |
| M1 | C1 |  |
| M4 |  |  |
| U1 |  |  |
| U2 |  |  |

# Discussion

Text

# Conclusion

Ce laboratoire avait pour objectif de