



UNIVERSITÉ MOHAMMED VI
POLYTECHNIQUE

EMINES - UM6P
SCHOOL OF INDUSTRIAL MANAGEMENT
PROJET INFORMATIQUE

HPC interface de soumission de travaux slurm

ANALYSE DE PROJET

Réalisé par:

Badr LAAJAJ

Nisrine HAMMOUT

Encadré par:

M. El Hassane AIT

ZEMZAMI

13 Octobre 2017

Contents

1	Phase 1 : Analyse Fonctionnelle	3
1.1	Introduction	3
1.1.1	Objet du document	3
1.1.2	Structure du document	4
1.2	Description de la solution	4
1.2.1	Caractéristiques de la solution	4
1.2.2	Processus utilisateurs impactés	4
1.2.3	Applications connexes	4
1.3	Description fonctionnelles détaillées	5
1.3.1	Module/Fonctionnalités	5
1.3.2	Acteurs :	6
1.3.3	Matrices acteurs / Fonctionnalités	7
1.3.4	Cas d'utilisation :	7
1.4	Conclusion	9
2	Phase 2 : Analyse technique	10
2.1	Introduction	10
2.2	Architecture globale	10
2.3	Fonctions de service (IHM) et Maquette	11
2.4	Description des données et des traitements	18
2.4.1	Modélisation des données	18
2.4.2	Description des traitements	18
2.5	Conclusion	20
3	Conclusion générale	21

Liste des figures

1	Gestion des accès	7
2	Diagramme de cas d'utilisation de la solution	9
3	Architecture globale de notre application web	10
4	Accueil de l'application web	11
5	Formulaire pour générer un script	12
6	Sign In et demande d'inscription	12
7	Fonction de soumettre un job	13
8	Historique des Jobs	14
9	Erreur de Sign In	15
10	Monitoring	16
11	Erreur admin	17
12	Diagramme de classe de la solution	18
13	Diagramme de séquence du visiteur	19
14	Diagramme de séquence du visiteur	20

1 Phase 1 : Analyse Fonctionnelle

1.1 Introduction

Notre projet "HPC interface de soumission des travaux SLURM" cherche à répondre au besoin du laboratoire de Modélisation et de Simulation (SIMLAB). Ce laboratoire représente l'une des unités de recherche de l'Université Mohammed VI Polytechnique-UM6P qui cherche à développer des concepts, méthodes et méthodologies qui permettent de mieux comprendre les phénomènes mis en jeu dans les procédés industriels, afin de mieux les modéliser pour les améliorer. Dans cette perspective, l'unité compte utiliser des grappes serveurs afin d'effectuer du calcul haute performance (connu sous le sigle **HPC**-High Performance Computing) et d'exécuter plusieurs tâches simultanément (le parallélisme).

En effet, l'utilisation des superordinateurs devient de plus en plus indispensable pour résoudre des problèmes complexes de modélisation dans plusieurs disciplines telles que : la modélisation climatique, l'analyse cryptographique, la géophysique, la mécanique des fluides, la biologie moléculaire, la dynamique moléculaire... Toutefois, ces machines sont sous des systèmes d'exploitation libres, le plus répandu est Linux¹, qui a pour objectif, d'une part, de réduire les coûts (s'exonérer des licences) et, en d'autres part, pouvoir adapter la puissance de calcul au besoin en développant ces propres outils "maison" d'exploitation comme les scripts d'administration système.

D'un point de vue utilisateur, la plupart des chercheurs ne sont pas familiarisés avec l'environnement Linux ni avec la technologie utilisant le HPC. Pour résoudre ce problème, on propose notre application web qui permettra à ces utilisateurs du serveur de calcul de générer des scripts qui leur permettent de soumettre des travaux de calcul d'une manière claire et facile.

1.1.1 Objet du document

Cette première phase représente l'analyse fonctionnelle de notre produit « HPC interface de soumission des travaux SLURM » qui a pour but de caractériser et d'ordonner les différentes fonctions précisées par le client. Outre cela, elle va nous permettre de définir l'environnement du produit en d'autres termes les types d'utilisateur concerné ainsi que les interfaces connectées à notre interface.

¹système d'exploitation libre sous la licence GNU-GPL, se basant sur la ligne de commande pour exécuter des tâches.

1.1.2 Structure du document

Afin de bien traiter cette partie, on va examiner les points suivants :

- **Description de la solution**

A ce stade, on doit tout d'abord décrire d'une manière détaillée l'ensemble des caractéristiques de notre solution. Il faut préciser ensuite non seulement les processus utilisateurs qui seront impactés par notre application mais aussi les autres applications qui interagissent avec notre solution et définir cette relation.

- **Description fonctionnelles détaillée :**

On peut répartir cette section en trois grands axes :

1. Modules/ Fonctionnalités : on regroupe les fonctionnalités de notre application sous forme de package puis on décrit le service qu'elle rend à l'utilisateur ainsi que ses règles de gestion.
2. Acteurs : On décrit les profils d'utilisateurs qui vont interagir avec l'interface.
3. Matrices acteurs/ Fonctionnalités : on lie les fonctionnalités avec les acteurs concernés grâce à une matrice de gestion de droit.

1.2 Description de la solution

1.2.1 Caractéristiques de la solution

Notre solution est sous forme d'une application web qui va permettre aux utilisateurs de générer des scripts et ensuite préciser la configuration de leur job qui va être soumis aux grappes serveurs. Notre application permet aussi aux administrateurs d'examiner l'état des machines et de gérer le job.

1.2.2 Processus utilisateurs impactés

Les utilisateurs de notre application sont en général les concernés par le calcul, ils peuvent être alors des chercheurs, ingénieurs, postdoctoraux ou des doctorants et en particulier ceux de l'Université Mohammed VI Polytechnique.

1.2.3 Applications connexes

Notre application web peut avoir une connexion avec le site du laboratoire SIMLAB . En effet, le laboratoire peut ajouter une rubrique sur leur site réservée au domaine HPC où notre application peut figurer.

1.3 Description fonctionnelles détaillées

1.3.1 Module/Fonctionnalités

On peut décomposer notre interface web en 4 modules principaux regroupant l'ensemble des fonctionnalités comme suit :

- **Authentification des utilisateurs** : ce module contient plusieurs fonctionnalités autres que l'identification des utilisateurs qui est basée sur l'utilisation du LDAP ². On cite parmi eux la possibilité de connaître l'historique de leurs jobs, la consommation des ressources ainsi l'état des jobs (terminés, en cours ou en échec...). Il s'ajoute que grâce à ce module on peut différencier les types d'utilisateur ce qui nous permet d'autoriser aux administrateurs de gérer les droits d'utilisation de certains logiciels à licence, compilateur ou bibliothèque scientifique par les utilisateurs normaux.
- **Génération des scripts batch** : ce package représente l'axe principale du projet. En effet, grâce à six zones multi-choix l'utilisateur peut déterminer la configuration de son travail, principalement le nombre de nœuds à utiliser, leurs noms ainsi que le nombre de cœurs, le nombre de threads, la durée du job et son nom. Il peut aussi préciser les entrées/sorties utiles pour son job et même télécharger une version texte du script.
- **Monitoring: Administration du serveur de calcul**, qui regroupe les fonctionnalités suivantes :
 - Permettre aux administrateurs de contrôler les machines (marche/arrêt) et de connaître leurs états ;
 - Générer des statistiques d'utilisation sous forme de graphe ;
 - Gestion des travaux soumis.
- **Soumission des travaux de calcul**, cette fonctionnalité consiste à soumettre le job constitué du script et des commandes SLURM³.

²(Lightweight Directory Access Protocol): un protocole qui permet de lire uniquement d'une base de données et par conséquent partager l'utilisation des informations d'une façon légère avec les applications web

³(Simple Linux Utility for Resource Management): Système de gestion de tâche (ordonnanceur) qui gère l'utilisation de ressources des machines

1.3.2 Acteurs :

Dans notre cas, les acteurs sont les utilisateurs de notre application web. Cependant il existe deux types de profil d'utilisateur qui ont accès à des fonctionnalités différentes, on distingue entre :

- **Les administrateurs:** ceux qui gèrent et veillent au bon fonctionnement du serveur de calcul, c'est-à-dire configurer, installer et contrôler l'accès à ces machines, outre les avantages des utilisateurs normaux;
- **Les utilisateurs normaux** , par contre eux ils peuvent uniquement soumettre des jobs, suivre l'état de leurs jobs et accéder aux informations de leurs travaux(en attente, courants, rejetés). On peut aussi distinguer parmi eux les **utilisateurs inscrits** qui ont l'autorisation d'utiliser les machines, en d'autres termes ils possèdent des comptes validés, et les **utilisateurs non-inscrits** qui peuvent être des simples visiteurs de l'interface.

1.3.3 Matrices acteurs / Fonctionnalités

Acteurs	Administrateur	Utilisateurs Inscris	Utilisateur non-inscrit
Fonctionnalités			
Génération des scripts batch			
Remplir les zones multi-choix	X	X	X
Préciser les entrées/sorties du job	X	X	
Télécharger une version texte du script	X	X	X
Authentification des utilisateurs			
Se connecter dans l'application	X	X	
Consulter l'historique des jobs et leurs états	X	X	
Connaître la consommation des ressources	X		
Gérer les droits d'utilisation des logiciels à licence, bibliothèques ou des compilateurs.	X		
Monitoring: Administration du serveur de calcul			
Contrôler les machines (marche/arrêt)	X		
Connaître l'état des grappes serveurs	X		
Consulter les statistiques d'utilisation des machines	X		
Gérer les travaux soumis	X		
Soumission des travaux de calcul			
Exécuter les scripts générés	X	X	

Figure 1: Gestion des accès

1.3.4 Cas d'utilisation :

Le diagramme 'Cas d'utilisation' ou 'Use Case' définit les liaisons fonctionnelles entre les acteurs et le système à l'étude, il est composé des éléments suivant :

- **Acteur** : rôle joué par un utilisateur humain ou un autre système qui interagit directement avec le système.

- **Cas d'utilisations** : ensemble de séquence d'actions réalisées par le système produisant un résultat observable intéressant pour un acteur particulier.
- **Association** : utilisé pour relier les acteurs et les cas d'utilisation par une relation qui signifie simplement « participe à ».
- **Inclusion** : le cas d'utilisation de bas incorpore implicitement un autre, de façon obligatoire.
- **Extension** : le cas d'utilisation de bas incorpore implicitement un autre, de façon optionnelle.
- **Généralisation** : les cas d'utilisations descendants héritent de la description de leur parent commun.

En ce qui concerne le « use case » de notre projet, il permet de mieux représenter les interactions entre les différents utilisateurs et les fonctionnalités proposées par notre application web. En fait, on distingue entre trois acteurs : les administrateurs, les utilisateurs inscrits et les non-inscrits. Quant aux fonctionnalités, il diffère d'un acteur à un autre mais grâce au concept généralisation on aperçoit les fonctionnalités communes entre eux, ce qui rend ce diagramme très utile pour effectuer la gestion des droits/accès. Voici donc ci-dessous notre diagramme Use Case :

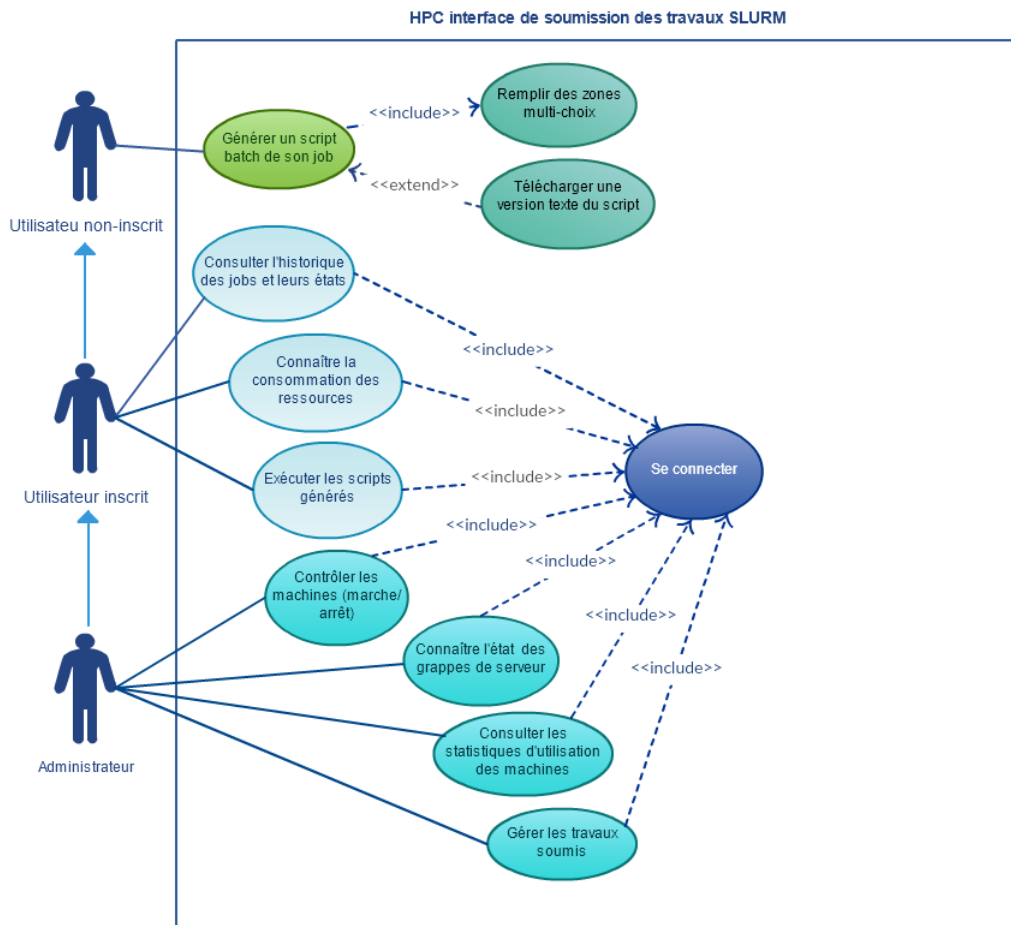


Figure 2: Diagramme de cas d'utilisation de la solution

1.4 Conclusion

Après avoir établi la phase d'analyse où on cherche à comprendre et à décrire d'une façon bien détaillée les besoins des clients et les fonctionnalités des différents utilisateurs en utilisant la matrices Acteurs/Fonctionnalités et le diagramme de cas du langage visuel de modélisation l'UML⁴. On passe à l'étape de conception où on cherche à clarifier les aspects techniques de notre application en se servant d'une maquette explicitant notre application web, et aussi une architecture globale et des diagrammes tels diagramme de classe et diagramme de séquence .

⁴Unified Modeling Language

2 Phase 2 : Analyse technique

2.1 Introduction

L'analyse technique consiste à choisir ou concevoir des solutions techniques qui vont remplir les différentes fonctions élaborés dans la phase précédente de l'analyse fonctionnelle. Les étapes de cette phase sont :

1. Élaboration de l'architecture globale de l'interface.
2. Précisant des fonctions de service qui décrivent les fonctionnalité, les données en entrée et les données en sortie. Ainsi, une maquette pour donner une idée ce quoi ressemblé notre produit final.
3. Représentation de la structure de l'application, du point de vue des données, et définit également les dépendances ou relations entre ces différentes objets en utilisant un diagramme de classe.
4. Élaboration du diagramme de séquence pour traiter l'aspect dynamique de notre interface.

2.2 Architecture globale

Elle s'agit de structurer notre application web en précisant les processus et les intervenants internes et externes et les composants de notre système:

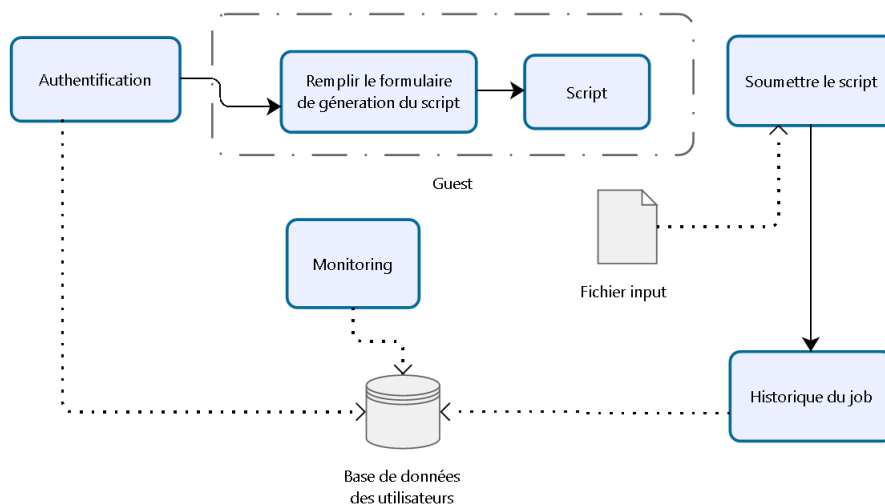


Figure 3: Architecture globale de notre application web

2.3 Fonctions de service (IHM) et Maquette

Cette partie concerne la description détaillée des étapes à accomplir pour répondre aux fonctionnalités que notre application web propose.

Notre application va contenir en premier lieu une page d'accueil qui sert à introduire aux visiteurs le but de création d'un tel service.

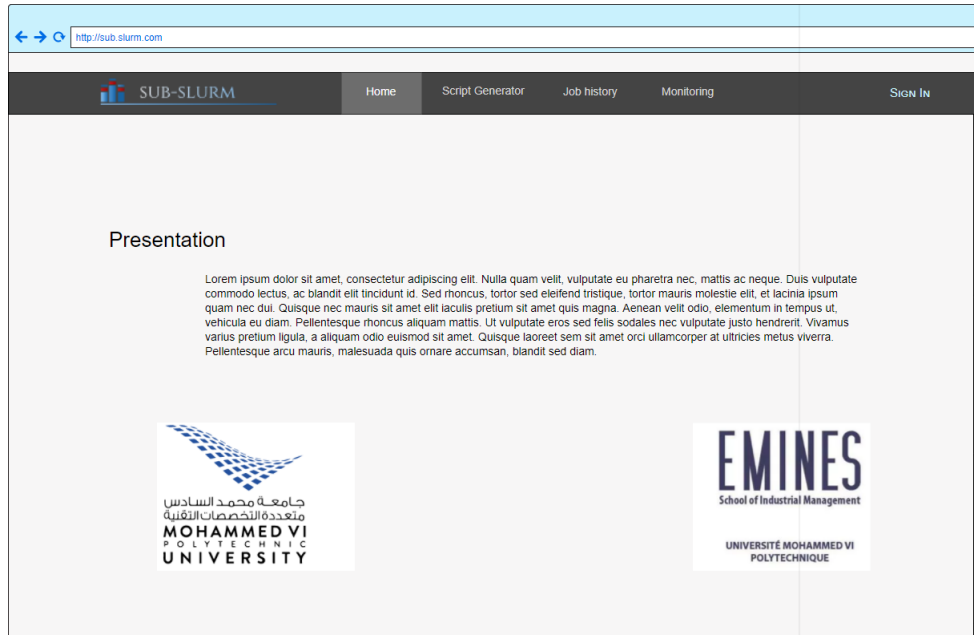
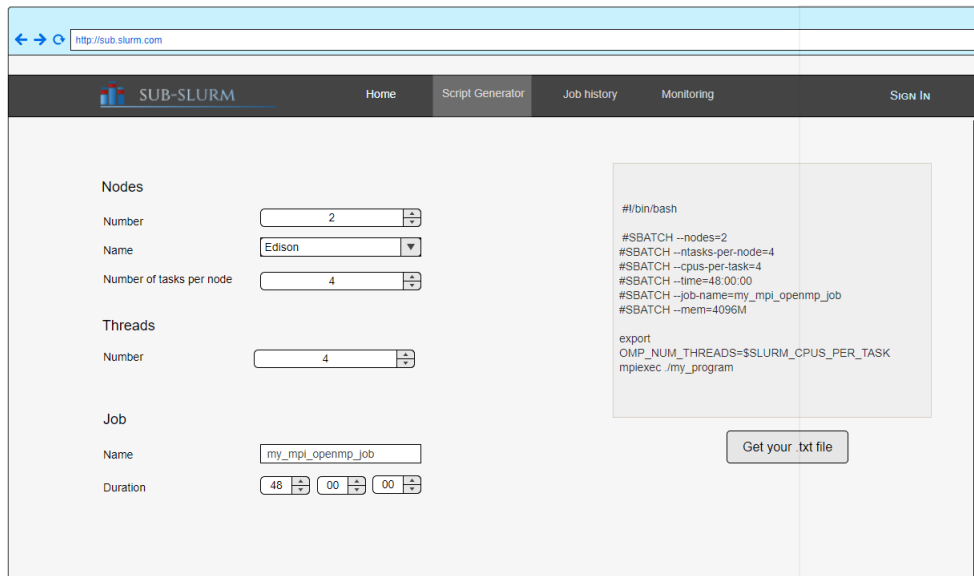


Figure 4: Accueil de l'application web

Un utilisateur non-inscrit, inscrit ou bien administrateur voulant résoudre des problèmes complexes de modélisation en utilisant la technologie des superordinateurs. Il a besoin tout d'abord d'un script batch généré par notre interface qui lui permettra après de soumettre son job. La génération se fait en choisissant :

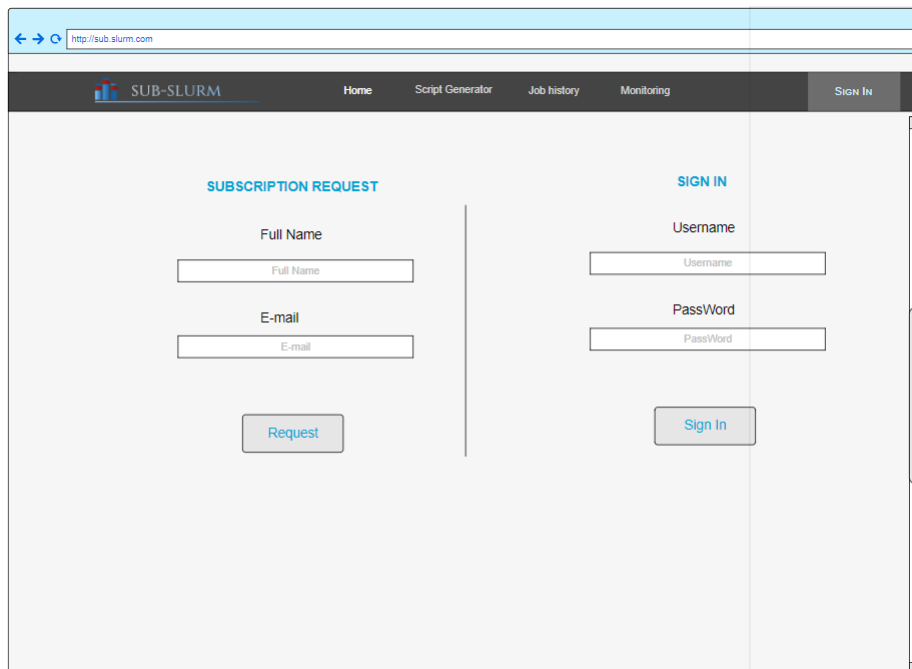
- le nombre de nœuds à utiliser;
- le nom de noeud ;
- le nombre de tasks par noeud;
- le nombre de threads ;
- le nom du job ;
- le durée du job



The screenshot shows the 'Script Generator' page of the SUB-SLURM interface. The browser address bar shows 'http://sub.slurm.com'. The navigation menu includes 'Home', 'Script Generator', 'Job history', 'Monitoring', and 'Sign In'. The main content area is divided into two columns. The left column contains form fields for job configuration: 'Nodes' (Number: 2, Name: Edison, Number of tasks per node: 4), 'Threads' (Number: 4), and 'Job' (Name: my_mpi_openmp_job, Duration: 48:00:00). The right column displays a generated script snippet in a text area, starting with '#!/bin/bash' and including Slurm directives like '#SBATCH --nodes=2' and '#SBATCH --ntasks-per-node=4'. Below the script is a 'Get your .txt file' button.

Figure 5: Formulaire pour générer un script

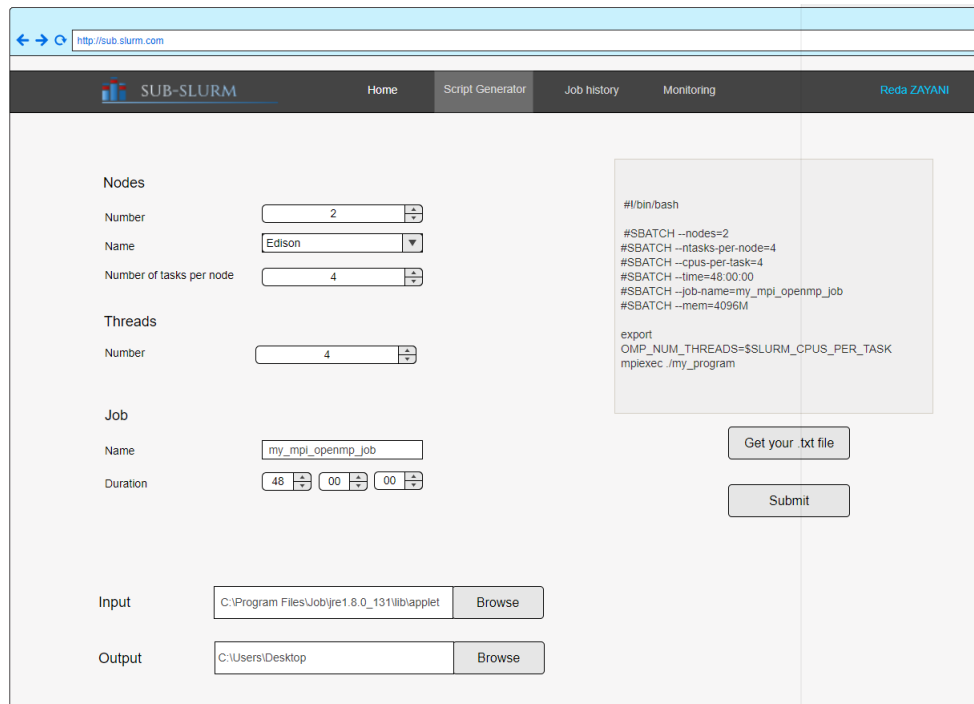
Pour préciser les inputs et les outputs d'un job, l'utilisateur a besoin d'authentifier en fournissant un username et un mot de passe propre à lui, si l'utilisateur n'a pas un compte il doit alors remplir un formulaire de demande d'inscription qui va être envoyé à l'administrateur.



The screenshot shows the 'Sign In' and 'Subscription Request' page of the SUB-SLURM interface. The browser address bar shows 'http://sub.slurm.com'. The navigation menu includes 'Home', 'Script Generator', 'Job history', 'Monitoring', and 'Sign In'. The main content area is divided into two columns. The left column is titled 'SUBSCRIPTION REQUEST' and contains form fields for 'Full Name' and 'E-mail', with a 'Request' button below. The right column is titled 'SIGN IN' and contains form fields for 'Username' and 'PassWord', with a 'Sign In' button below.

Figure 6: Sign In et demande d'inscription

Une fois l'utilisateur est authentifié, il a le droit non seulement de préciser les entrées et les sorties de son job, mais aussi de soumettre un job après avoir rempli le formulaire de génération de script:



The screenshot shows the SUB-SLURM web interface for job submission. The browser address bar shows `http://sub.slurm.com`. The navigation menu includes Home, Script Generator, Job history, and Monitoring. The user name Reda ZAYANI is displayed in the top right corner.

The main form is divided into several sections:

- Nodes:** Number (2), Name (Edison), Number of tasks per node (4).
- Threads:** Number (4).
- Job:** Name (my_mpi_openmp_job), Duration (48:00:00).
- Input:** C:\Program Files\Job\jre1.8.0_131\lib\applet Browse
- Output:** C:\Users\Desktop Browse

On the right side, a preview of the generated Slurm script is shown:

```
#!/bin/bash
#SBATCH --nodes=2
#SBATCH --ntasks-per-node=4
#SBATCH --cpus-per-task=4
#SBATCH --time=48:00:00
#SBATCH --job-name=my_mpi_openmp_job
#SBATCH --mem=4096M
export
OMP_NUM_THREADS=$SLURM_CPUS_PER_TASK
mpirun /my_program
```

Buttons for "Get your .txt file" and "Submit" are located below the script preview.

Figure 7: Fonction de soumettre un job

L'utilisateur inscrit a le droit aussi de consulter son propre historique de ses jobs et leurs états:

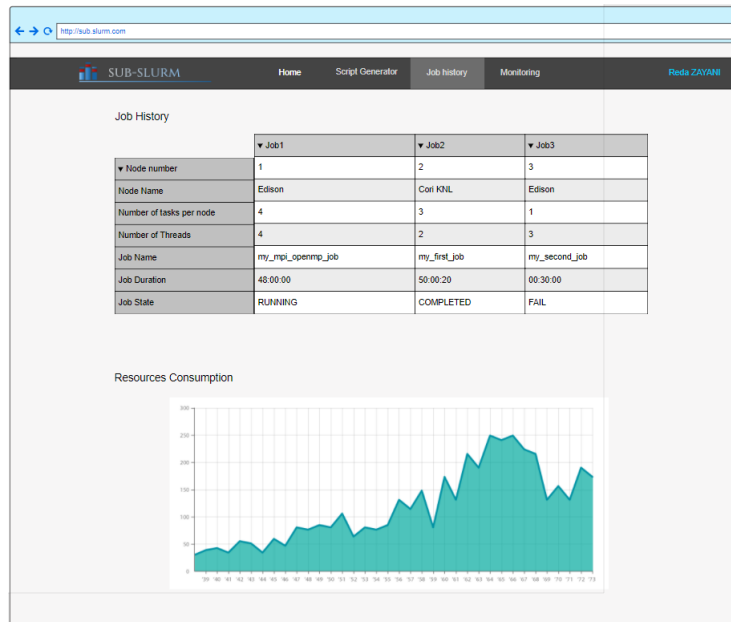


Figure 8: Historique des Jobs

Cependant, si un visiteur non-inscrit essaie de voir la partie de "History Job" il va avoir la page d'erreur suivante:

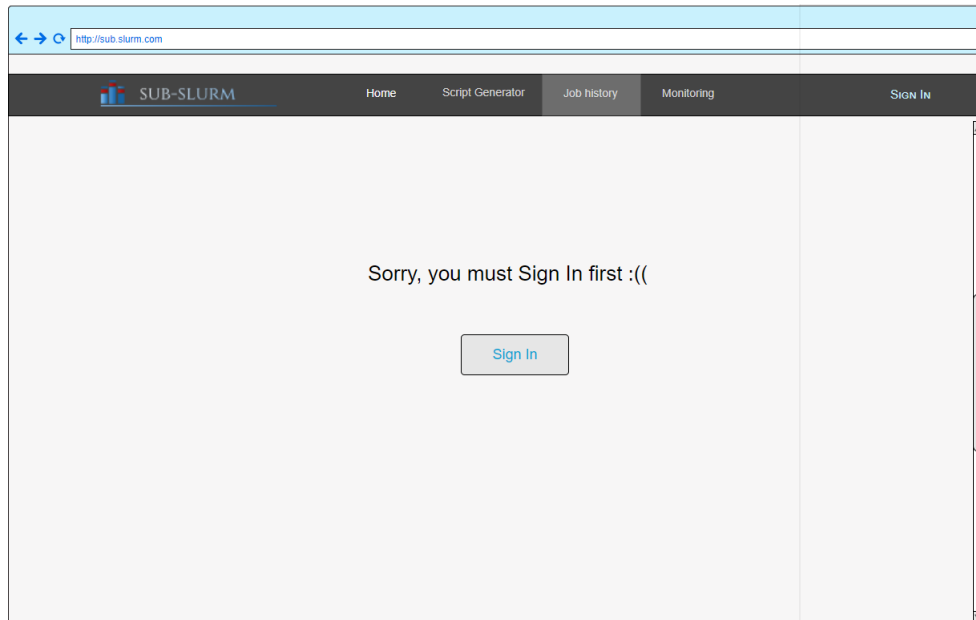


Figure 9: Erreur de Sign In

En ce qui concerne l'administrateur, il a droit de toutes les fonctionnalités d'un utilisateur inscrit à côté de la partie monitoring qui permet de contrôler la machine (marche/arrêt) et de gérer les jobs:

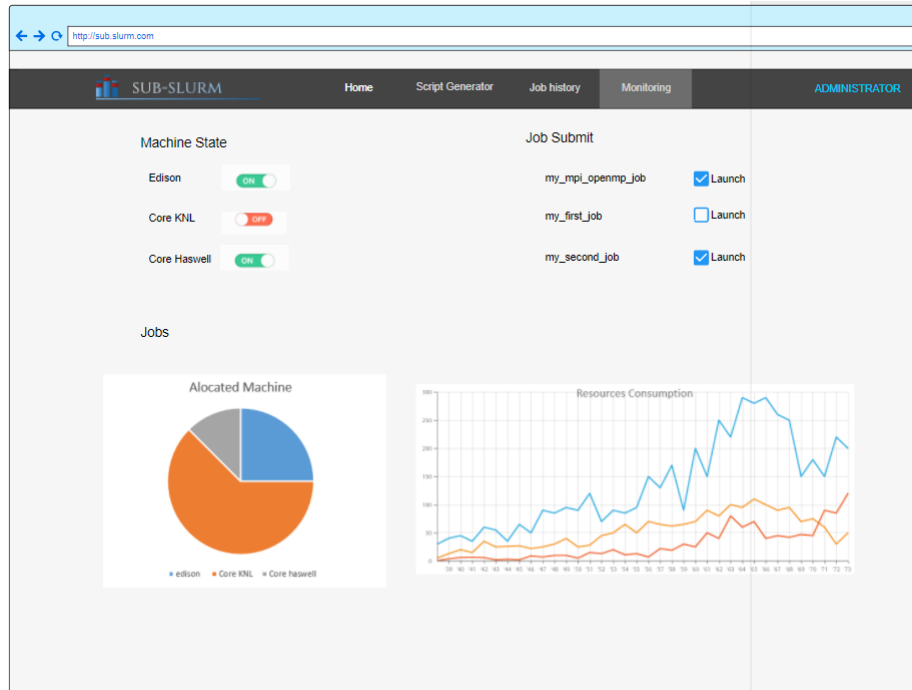


Figure 10: Monitoring

Et si un utilisateur inscrit veut accéder au monitoring, il aura l'erreur suivante:

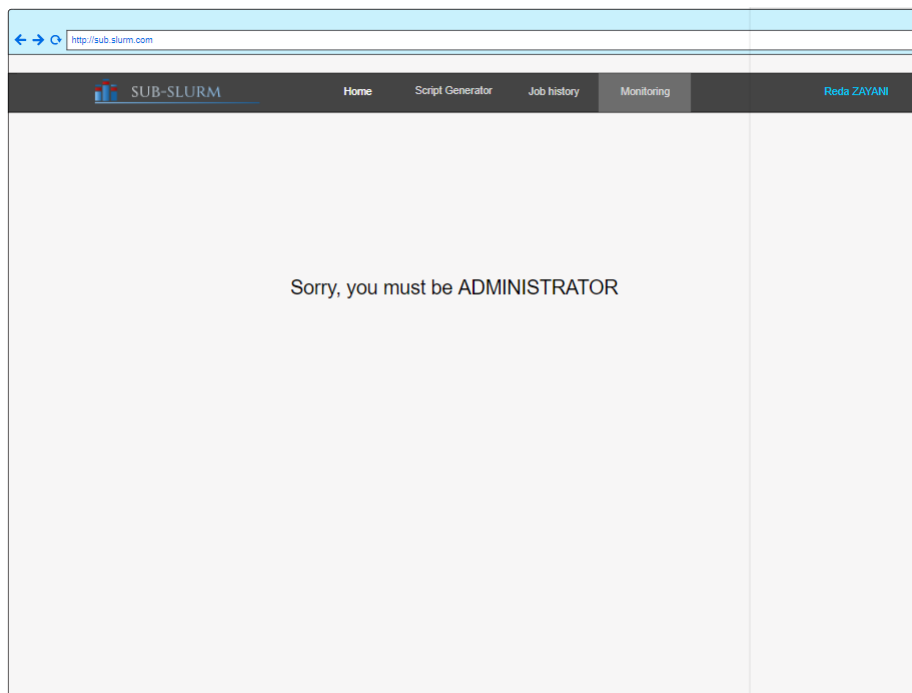


Figure 11: Erreur admin

2.4 Description des données et des traitements

2.4.1 Modélisation des données

Dans cette partie, on s'intéresse à la structure du système d'information de notre solution d'un côté statique ce qui nous permet de modéliser les relations et les dépendances entre les différentes classes. Alors en analysant les données liées à notre interface on obtient le diagramme de classe suivant :

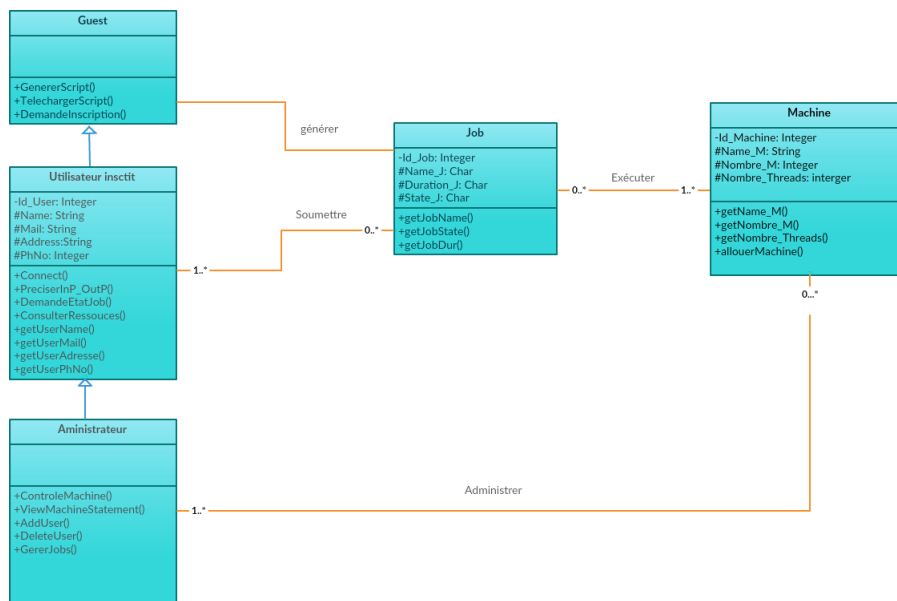


Figure 12: Diagramme de classe de la solution

2.4.2 Description des traitements

Afin de décrire les traitements, nous avons eu recours au diagramme de séquence. Le diagramme de séquence représente une modélisation dynamique du fonctionnement de notre solution. En effet, il s'intéresse aux interactions entre les différents objets et précise comment ils communiquent entre eux afin d'accomplir une action. Afin de bien présenter notre diagramme de séquence, on l'a divisé en deux parties: la première concerne le visiteur (utilisateur non-inscrit) et la deuxième les utilisateurs inscrits (utilisateur normal et administrateur) vu qu'ils partagent plusieurs fonctionnalités principales. Voici donc nos diagrammes de séquences :

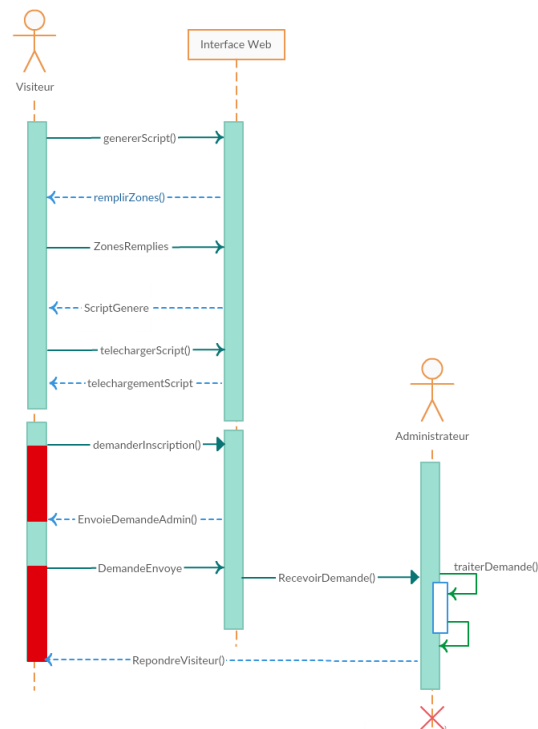


Figure 13: Diagramme de séquence du visiteur

Ce premier diagramme montre les fonctionnalités disponibles aux visiteurs ainsi qu'une démarche détaillée de la génération des scripts qui est commune avec les fonctionnalités des utilisateurs inscrits qui ont le diagramme de séquence suivant :

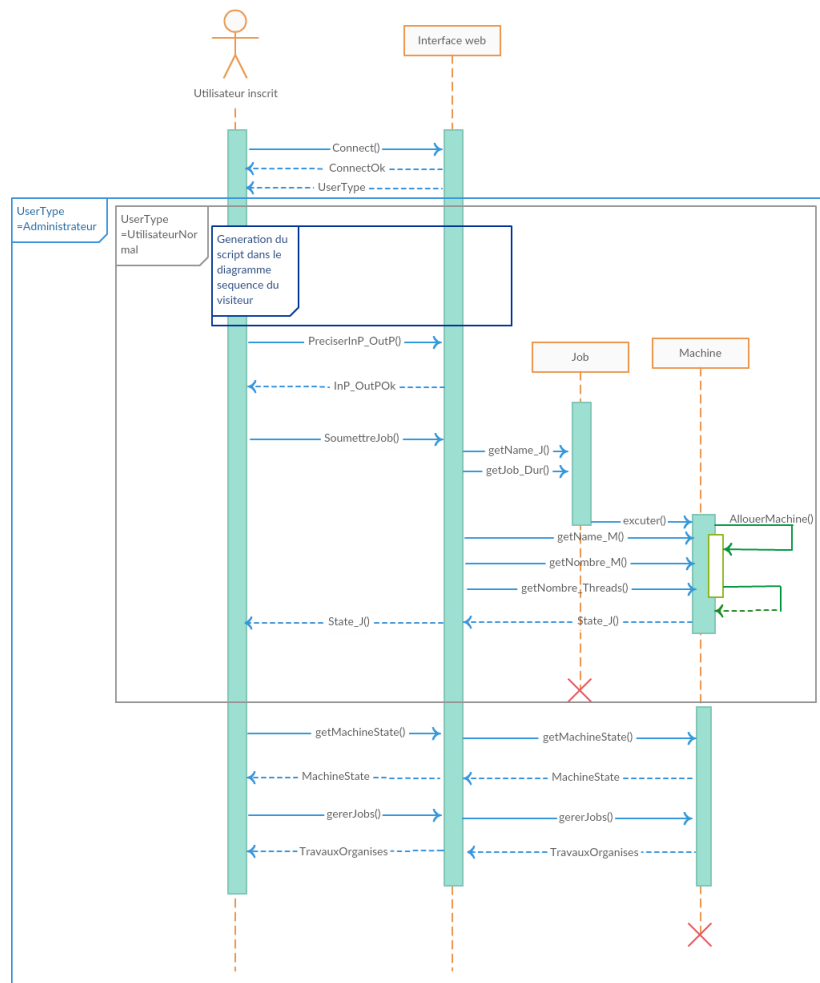


Figure 14: Diagramme de séquence du visiteur

2.5 Conclusion

Dans cette partie d'analyse technique, on a décrit d'un point de vue dynamique l'interaction entre les objets de notre application et les différents actions à exécuter pour remplir les fonctionnalités décrites dans la phase analyse fonctionnelle grâce à aux diagrammes de classe et de séquence ainsi qu'une maquette qui nous donne une idée sur le design de l'interface web en correspondance avec le scénario suivi.

3 Conclusion générale

En guise de conclusion, cette analyse de projet a porté sur trois aspects essentiels à savoir l'aspect fonctionnel, qui nous a permis de décrire en détail notre solution et de définir les interactions entre ses acteurs et fonctionnalités, l'aspect statique qui se résume dans le diagramme de classe qui décrit les relations entre les objets et l'aspect dynamique qui détermine les différents scénarios possibles que notre application peut rencontrer. Finalement, cette analyse nous sera très utile dans l'élaboration de notre application vu qu'elle nous a donnée des solutions techniques à développer dans la phase de réalisation .