	**Démonstrateur web d'analyse syntaxique**	Université de la Méditerranée (U2)

Rapport final

Objet :

Ce document fait état du rapport final du projet Démonstrateur web d'outils d'analyse syntaxique.

Site officiel du projet :http://macaweb.lif.univ-mrs.fr .

Dépôt du projet : https://github.com/oualid13/MacaWeb .

Auteurs

Auteurs
M. Mahri Oualid M. Gahmousse Abdel
Rédigé le : 29/04/2011

Historique

Version	Auteur	Date	Commentaires
v1.0	M. Mahri Oualid	29/04/11	Version préliminaire
v2.0	M. Mahri Oualid	02/05/11	Version de travail
v3.0	M. Mahri Oualid	05/05/11	Version de travail
v4.0	M.Gahmousse Abdel	06/05/11	Version de travail
v5.0	M.Gahmousse Abdel	07/05/11	Version de travail
v6.0	M. Mahri Oualid	08/05/11	Version de travail
v7.0	M.Gahmousse Abdel	11/05/11	Version de travail
v8.0	M.Gahmousse Abdel	12/05/11	Version de travail
v9.0	M. Mahri Oualid	12/05/11	Version de travail
v10.0	M. Mahri Oualid	13/05/11	Version final

Table des matières

1.Remerciement

Avant d’entamer ce rapport, nous profitons de l’occasion pour remercier tout d’abord notre tuteur M. Benoit Favre pour ses conseils, son aide et son soutient tout au long du projet .

A l'issue des deux agréables années au sein du département informatique de l'université de la Méditerranée de Marseille nous adressons des remerciements a toute l'équipe enseignante pour la qualité de l'enseignement qui nous a été dispensé et de nous avoir inciter à travailler en mettant à notre disposition leurs expériences et leurs compétences.

2.Introduction

Certaines applications sont difficiles a installer et reposent sur plusieurs bibliothèques qui doivent être installées séparément, de plus ces applications ne sont généralement pas multiplateformes.
Cela pourra empêcher un grand nombre d'utilisateurs d'utiliser ce genre d'applications.
Une solution pour résoudre ce problème, est de créer un démonstrateur web pour publier ces applications pour qu'elles deviennent accessible au grand public.
Le but de notre projet est de réaliser un démonstrateur web pour l'application MACAON développée par l'équipe TALEP au LIF pour faciliter son utilisation et lui donner plus de portabilité pour qu'elle fonctionne quel que soit le système d'exploitation utilisé. Elle sera accessible depuis un navigateur directement sur le web sans être installée sur l'ordinateur de l'utilisateur.
Cette solution va certainement augmenter le nombre d'utilisateurs de l'application MACAON et c'est le but principal de notre projet.

3.MACAON

MACAON est une chaîne de traitement permettant d'effectuer des tâches standard de traitement automatique de la langue aussi bien sur du texte natif (produit par un être humain) que sur des hypothèses multiples issues de processus automatiques (reconnaissance de la parole, traduction automatique ...).

MACAON est composée de plusieurs modules réalisant des traitements classiques (découpage en mots, étiquetage morpho-syntaxique, lemmatisation, analyse morphologique, analyse syntaxique partielle).

Les différents modules communiquent entre eux par l'intermédiaire de fichiers XML qui permettent de représenter les différents niveaux d'analyse .
Les principaux modules :

maca_tools	outils de conversion vers et depuis le format MACAON
maca_segmenter	segmenteur en phrase
maca_tokenizer	segmenteur en atomes (tokens)
maca_lexer	segmenteur en mots
maca_tagger	étiqueteur en catégories morpho-syntaxiques
maca_anamorph	analyseur morphologique
maca_chunker	analyseur syntaxique partiel
macaviz	outil d'édition et de visualisation de fichiers MACAON

site officiel de MACAON : http://macaon.lif.univ-mrs.fr .

4.Membres du projet

Encadrant pédagogique :

Benoit Favre : Maître de conférences,
Laboratoire d'Informatique Fondamentale de Marseille(France).
E-mail : benoit.favre@lif.univ-mrs.fr

Étudiants :

Mahri Oualid : Étudiant à l'Université de la méditerranée .
E-mail : oualid.mahri@etumel.univmed.fr .

Gahmousse Abdel : Étudiant à l'Université de la méditerranée .
E-mail : typsou@hotmail.fr .

5.Présentation du sujet

L'objectif du projet est de réaliser un démonstrateur web de l'outil d'analyse syntaxique MACAON développé au LIF en utilisant de nouvelles technologies web avec une architecture client-serveur.
Pour l'instant l'outil a une interface classique, il faudrait refaire quelque chose de similaire en html5 de façon à ce que l'utilisateur puisse taper une phrase qui serait analysée côté serveur et dont le résultat s'afficherait côté client. L'idée serait de faire quelque chose de similaire à la recherche instantanée de Google où le résultat s'affiche en même temps qu'on tape la requête.

6.Architecture Client-Serveur

L'architecture client-serveur désigne l'ensemble des moyens matériel, logiciel et de communication permettant de mettre en place une application conforme au modèle client-serveur. Le modèle client-serveur est un mode de fonctionnement théorique basé sur la séparation des rôles. Il décrit le mode de fonctionnement des applications.

6.1.Serveur

Le terme serveur fait référence a tous processus qui reçoivent une demande de service venant d'un client via le réseau, traite cette demande et renvoie le résultat au demandeur, c'est-à-dire le client.

6.2.Client

Le terme client fait référence a tous processus utilisant des services offerts par un serveur et communiquant avec lui a l'aide de messages.

6.3.Fonctionnement des applications Client-Serveur

C'est l'application client qui prend l'initiative du dialogue, le programme dit « Client » demande, par l'envoi d'un message, un service extérieur à un autre programme dit « Serveur ».une fois le service rendu, le programme « Serveur » renvoie un message au programme « Client ».Toute cette communication se fait de manière transparente.

Le modèle client-serveur est un modèle de fonctionnement coopératif entre programmes.
Une architecture client-serveur fournit des services distant a des clients qui peuvent utiliser de manière transparente l'ensemble de ressources informatiques mises à leur disposition.

6.4.Les objectifs du client-serveur

Répartir les taches entre le client et le serveur :

- Décharger le serveur de l'exploitation de données.
- Décharger la station cliente de la gestion des données.
- Réduire le trafic sur le réseau.

7.Analyse des risques

Il s'agit de développer une application web qui va jouer le rôle d'un démonstrateur de l'application MACAON et qui doit reposer sur une architecture client-serveur .

il est indispensable de procéder a une étude complète et raisonner des risques potentiel plus au moins prévisibles.

Une analyse de risques rigoureuse est une bonne garantie de réussite du projet.

7.1.Risques organisationnels

Risque: Communication insuffisante

Description: Problèmes de communication interne

Impact: Critique

Solutions:

- Rester autant que possible en contact et organiser des réunions internes régulièrement .
- Travailler en binôme .

Risque: Gestion des versions inexistante

Description: Travail en parallèle sur le même logiciel

Impact: Intermédiaire

Solutions:

- Utiliser un gestionnaire de version tel que SVN ou Git pour gérer les multiples modifications et les concurrences entre celles-ci et partager le code sources entre les membres de projet.

Risque: Manque de temps pour la réalisation du projet

Description: Mauvaise estimation de la durée des différentes tâches et mauvaise répartition entrainant le projet final non terminé par manque de temps .

Impact: Intermédiaire

Solutions :

- Bonne organisation du groupe et bonne répartition des taches.
- Prévoir la fin du projet en avance pour finir quand même en cas d'imprévus.
- Diviser les taches en deux catégories obligatoires et facultatives, commencer par les taches obligatoires et laisser les taches facultatives vers la fin du projet.

7.2.Risques techniques

Risque: Manque de formation .

Description: Manque de connaissance en programmation web(en php et JavaScript en particulier).

Impact: Critique

Solutions :

- Au lieu d'apprendre deux langages différents, un langage coté client et l'autre coté serveur on utilisera un seul langage coté client et coté serveur :
  - JavaScript coté client.
  - NodeJs (JavaScript coté serveur).
- Commencer la formation à nodeJs et JavaScript le plus tôt possible et travailler en binôme pour trouver des solutions plus rapidement .

Risque: Manque de documentation sur nodeJs.

Description: nodeJs est un nouveau langage de programmation et pour le moment il n'est pas bien documenté et il n'y a aucune documentation en français.

Impact: Critique

Solutions :

- Profiter de l'expérience du professeur encadrant dans le domaine de la programmation web en général et en nodeJs en particulier .

8.Analyse des besoins

8.1.Objectif du projet

Il s'agit de développer un démonstrateur web pour l'application MACAON développé par l'équipe TALEP au laboratoire LIF.

8.2.Objectif de l'application

Le résultat de l'application et qui sera mise à jour a chaque fois que l'utilisateur saisie un caractère sera affiché comme un graphe représentant l'analyse syntaxique de la phrase saisie sous forme d'une image sans occasionner le rechargement de la page.

Le client est une page Web avec une interface simple avec un formulaire où on peut taper une phrase et une zone pour afficher le résultat sous forme d'image.
Le serveur est un programme qui reçois une requête du client, exécute l'application MACAON avec la phrase envoyée par le client comme paramètre, récupère le résultat et le renvoie vers le client . Pour améliorer les performances du démonstrateur il faudra penser a créer une mémoire cache où seront stockés les résultats les plus demandés. il faut aussi trouver une solution pour diminuer le trafic de données .
Il est très important de penser a la sécurité de l'application surtout si elle est hébergée sur le serveur de l'université.
L'application doit être multiplateformes et marche sur la majorité des navigateurs Web connus.

9.Analyse technique

9.1.Outils utilisés

Systèmes d'exploitation de développement : Linux.

Systèmes d'exploitation de tests : Linux, Mac OS et Windows.

Google Chrome : navigateur web développé par google et qui contient un module de débogage très puissant et efficace.

Planner : logiciel de planification(planning prévisionnel/planning réel).

Umbrello : logiciel de création de diagrammes techniques.

OpenOffice Writer: Logiciel de traitement de texte Open source.

Git : Logiciel de gestion de versions décentralisée .

Nos choix d'outils ont été basé sur nos expériences antérieures avec ces outils et sur leur facilité d'utilisation et nous avons suivi aussi les conseils de notre encadrant pédagogique .

9.2.Langages de programmation utilisé

Pour réaliser une architecture client-serveur il faut généralement utiliser deux langages de programmation différents, un langage coté serveur et l'autre coté client.
Pour éviter d'apprendre deux langages différents et diminuer la période d'apprentissage, vu la durée de projet (3mois) on a décidé d'utiliser le langage JavaScript coté client et coté serveur car cela est devenu possible avec le nouveau langage de programmation nodeJs (JavaScript coté serveur) écrit en javascript et C++ et qui offre la même syntaxe que JavaScript, de plus il contient un nombre important de modules système, nodeJs est le langage le plus approprié pour ce projet.

Donc les langages utilisé sont :

- Coté serveur : nodeJs, qui est un framework évènementiel très rapide et optimisé basé sur la machine virtuelle V8 développée par Google.
- Coté client :
  - Html5 : la dernière version du langage de balisage HTML avec beaucoup de nouveautés qui offrent un tas de facilités aux programmeurs et une grande interactivité avec l'utilisateur.
  - JavaScript: un langage de programmation de scripts principalement utilisé dans les pages web interactives. C'est un langage orienté objet à prototype.
  - Css : un langage informatique qui sert à décrire la présentation des documents HTML et XML.

La combinaison de ces langages nous permet d'utiliser le concept de programmation Ajax qui offre une interaction dynamique entre les programmes client et serveur et une très grande interactivité avec l'utilisateur.

L'idée est de faire communiquer la page Web avec le serveur Web sans occasionner le rechargement de la page. C'est la raison pour laquelle JavaScript est utilisé, car c'est lui qui va se charger d'établir la connexion entre la page Web et le serveur et de changer le contenu de la page Web.

Pour diminuer le trafic de données on a utilisé une bibliothèque JavaScript pour dessiner le graphe résultat coté client, avec cette solution on décharge le serveur de la génération des images et on envoie du texte au lieu d'envoyer des images, cela rend l'application beaucoup plus rapide.

Pour donner au démonstrateur plus de portabilité on a décidé d'utiliser la bibliothèque JavaScript « jQuery » qui vise a faciliter le développement d'applications Web dynamiques et la manipulation du DOM et qui offre un large support ajax.

Malheureusement La bibliothèque canviz, un composant essentiel du projet utilise la bibliothèque JavaScript « prototype » qui peut entrer en conflit avec « jQuery ».

Il existe des méthodes pour résoudre ce conflit mais on a opté pour le choix de « prototype » (qui offre presque les mêmes fonctionnalités offertes par « jQuery ») par précaution et pour respecter la contrainte de délai.

9.3.Formats de données utilisés

Pour faire communiquer les modules du projet, nous avons utilisé plusieurs formats de données :

le format XML : Les différents modules de MACAON communiquent entre eux par l'intermédiaire de fichiers XML qui permettent de représenter les différents niveaux d'analyse.
Le format XML de MACAON permet d'avoir une structure de stockage de données commune à tous les différents modules. Ainsi cette dernière permet à chaque module de comprendre et d'ajouter de l'information dans le fichier XML.

le format PNG : Dans la version 1.0.0 du projet, le serveur en utilisant MACAON, génère des fichiers image en format PNG qui vont être renvoyer vers le client comme résultat.

le format DOT : Dans la version 2.0.0 du projet, le serveur en utilisant MACAON, génère des fichiers DOT.

L'extension de fichier DOT représente un fichier de description écrit en langage DOT pour la visualisation graphique en utilisant des outils tels que graphviz ou comme dans notre cas la bibliothèque canviz.
Malheureusement la bibliothèque canviz ne prend pas en entrée des fichiers de l'extension DOT mais des fichiers GV.

le format GV : Après avoir généré le fichiers DOT on le converti en fichier GV et en le renvoie vers le client.

le format JSON : JSON est un format de données textuels, génétique, dérivé de la notation des objets du langage ECMAScript.

Un document JSON ne comprend que deux éléments structurels :

- Ensemble de paire valeur nom /valeur.
- Des listes ordonnées des valeurs.

Ces mêmes éléments représentent 3 types de données :

- Des tableaux.
- Des objets.
- Des valeurs génériques.

Le principal avantage de JSON est qu'il est simple à mettre en œuvre par un programmeur tout en étant complet.

Il sert à faire communiquer des applications dans un environnement hétérogène. Il est notamment utilisé comme langage de transport de données par AJAX et les services Web.

10.NodeJs

NodeJs est un framework Javascript évènementiel très rapide et optimisé construit au dessus de V8, le très performant moteur Javascript open source soutenu par Google et utilisé notamment par le navigateur Chrome.

Il est destiné à l'écriture de programmes de réseau évolutives telles que les serveurs web.Il a été créé par Ryan Dahl en 2009, et sa croissance est parrainé parJoyent.

Les Objectifs de Nodejs sont similaires à ceux de Twisted pour Python, Perl Objec Environment pour Perl, et eventmachine pour Ruby. Contrairement à JavaScript, il n'est pas exécuté dans un navigateur web, mais exécuté côté serveur. NodeJs met en œuvre certaines spécifications de CommonJS.

11.Canviz

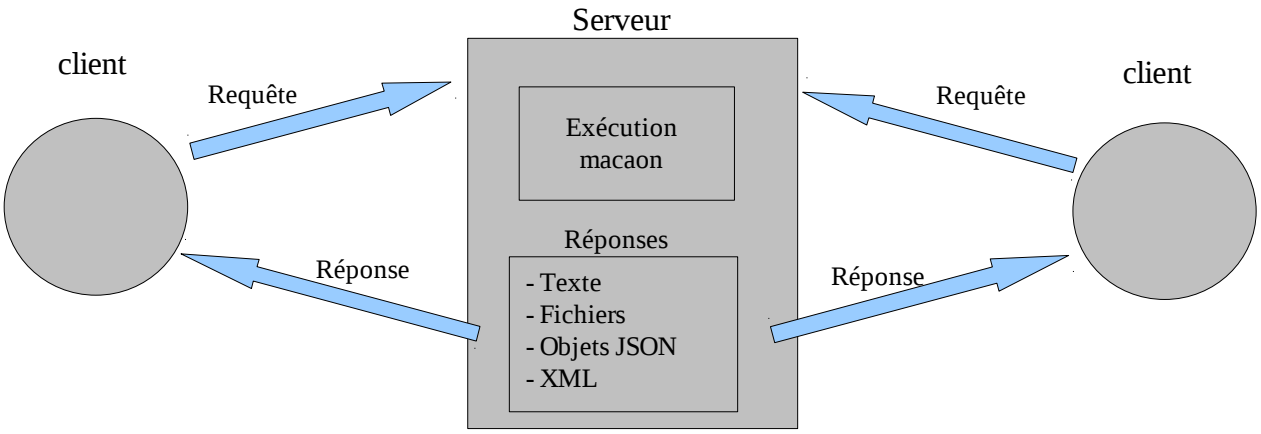
Canviz est une bibliothèque JavaScript pour dessiner des graphes dans un élément canvas d'une page web.

L'utilisation de Canviz présente des avantages pour une application web, au lieu de générer des images et les envoyer vers un navigateur :

- Le serveur n'a besoin que de générer du texte DOT, ce qui est plus rapide que la génération d'images bitmap.
- Seul le texte DOT doit être transféré vers le navigateur, ce qui est plus petit que les données d'image binaire, et si le navigateur le prend en charge le texte peut être compressé en gzip ou bzip2.
- Le navigateur Web génère l'image, et pas le serveur, ce qui réduit la charge du serveur.
- L'utilisateur peut redimensionner le graphique sans avoir besoin d'impliquer le serveur, ce qui est plus rapide que d'avoir le serveur dessiner et envoyer le graphique dans un format différent.

12.Diagrammes de séquence

Le démonstrateur repose sur une architecture client-serveur, les deux programmes communiquent entre eux a l'aide de messages sous forme du texte, objets JSON ou XML et fichiers selon le schéma suivant :

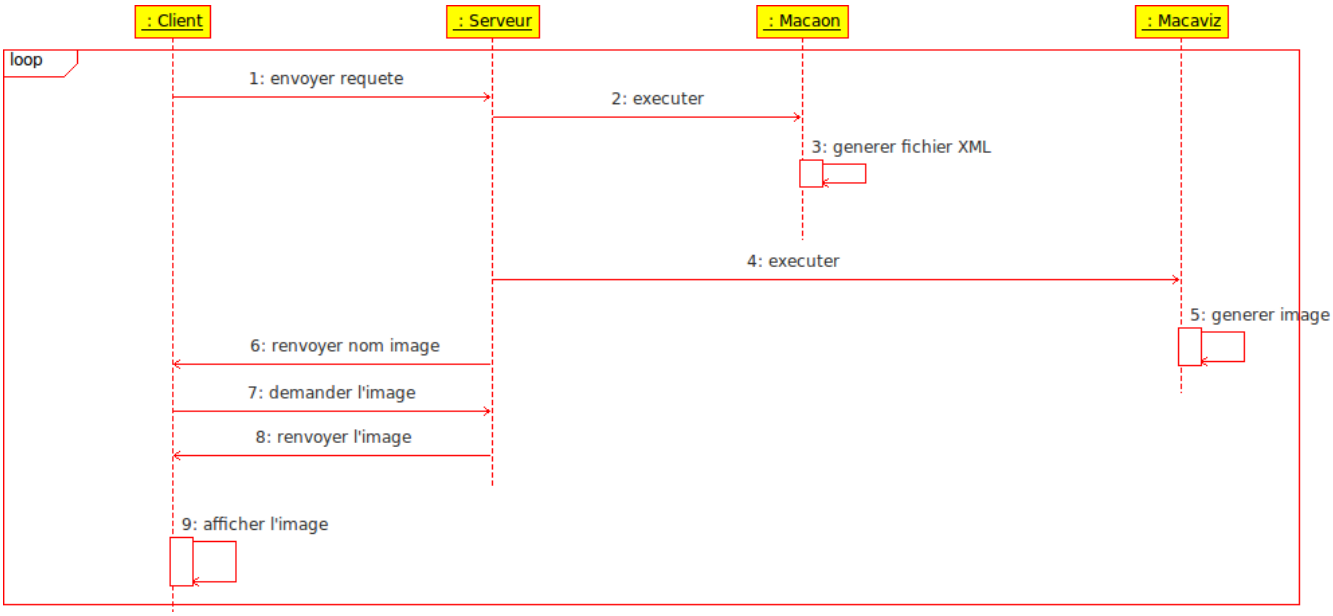


Le client prend l'initiative du dialogue en envoyant une requête au serveur, le serveur analyse la requête reçu et lui répond selon le contenue de la requête soit avec un fichier qui a été demandé soit avec un résultat en utilisant l'application MACAON.

Pour assurer un redu fonctionnel dès le début du projet on a créer plusieurs versions.
La première version a pour but de réaliser une application qui répond aux besoins de base.
Chaque version repose sur la version précédente, apporte des améliorations et répond aux nouveaux besoins.

12.1.Version-1.0.0

Dans cette version préliminaire le but était de réaliser une version fonctionnelle du démonstrateur et qui réponde aux besoins de base .



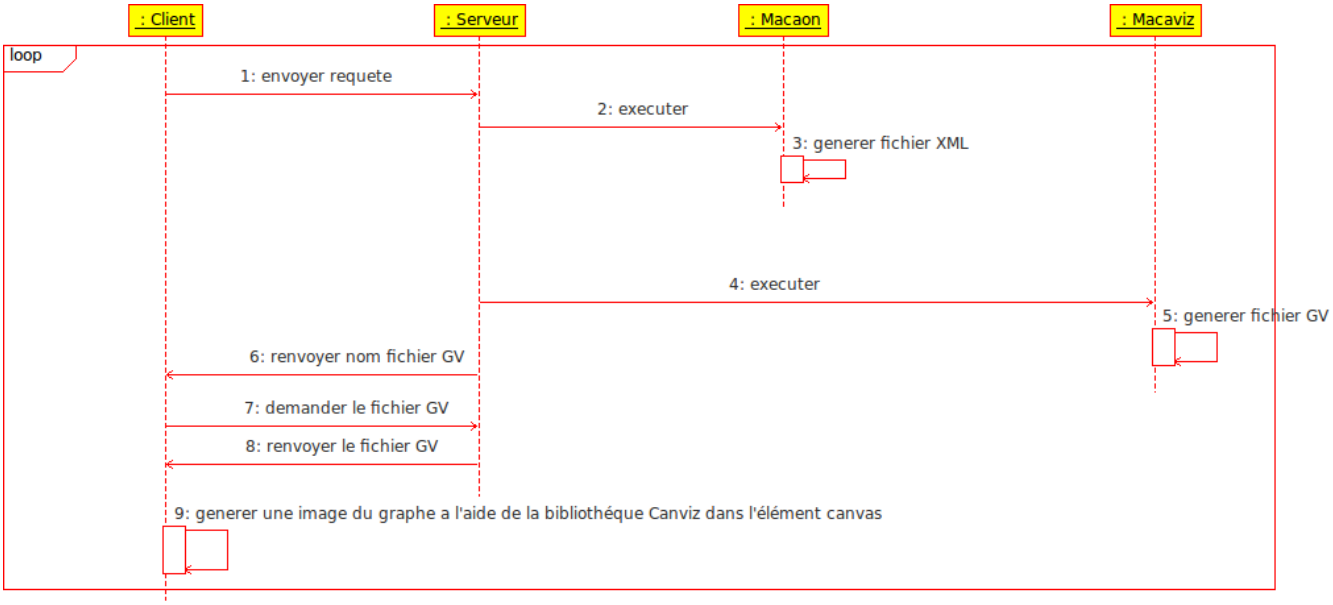
Le client récupère la phrase saisie par l'utilisateur et envoie une requête http contenant la phrase récupérée vers le serveur.

Le serveur analyse la requête reçue, récupère la phrase saisie par l'utilisateur et utilise des composants de MACAON pour analyser syntaxiquement cette phrase et ensuite il génère un graphe sous forme d'une image avec le composant macaviz de MACAON. Ensuite le serveur renvoie au client le nom du fichier généré.

À la réception du nom de fichier, le client change le contenu de la page Web et le navigateur Web va se charger de récupérer le fichier en regardant d'abord dans sa mémoire cache, si le fichier n'est pas présent le navigateur envoie une requête au serveur pour lui demander le fichier, le serveur lui répond avec le fichier demandé.

12.2.Version-2.0.0

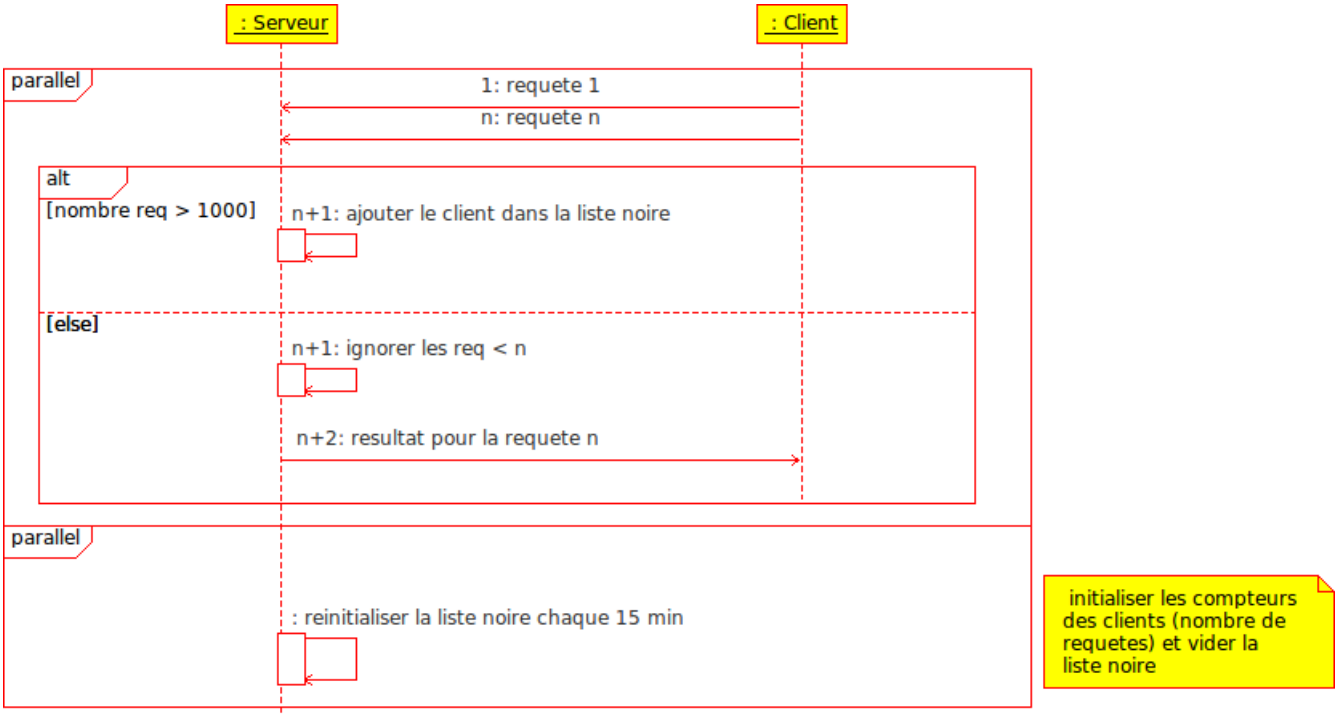
Dans cette version nous avons essayé d'améliorer les performances de l'application et de minimiser le trafic de données.



L'idée c'était de générer un fichier GV qui contienne les informations du graphe au lieu de générer une image, sachant que la taille de l'image fait a peu près 4 fois la taille du fichier GV,
donc en utilisant cette méthode on gagnera beaucoup d'espace dans la mémoire cache, de plus la réponse du serveur sera 3 fois plus rapide.
Le client, en recevant le fichier GV utilisera canviz, une bibliothèque JavaScript qui permet de dessiner un graphe dans l'élément canvas du document HTML.

12.3.Version-3.0.0

Dans cette version nous avons essayé de sécuriser l'application avec quelques améliorations de performances .



Sécurité :

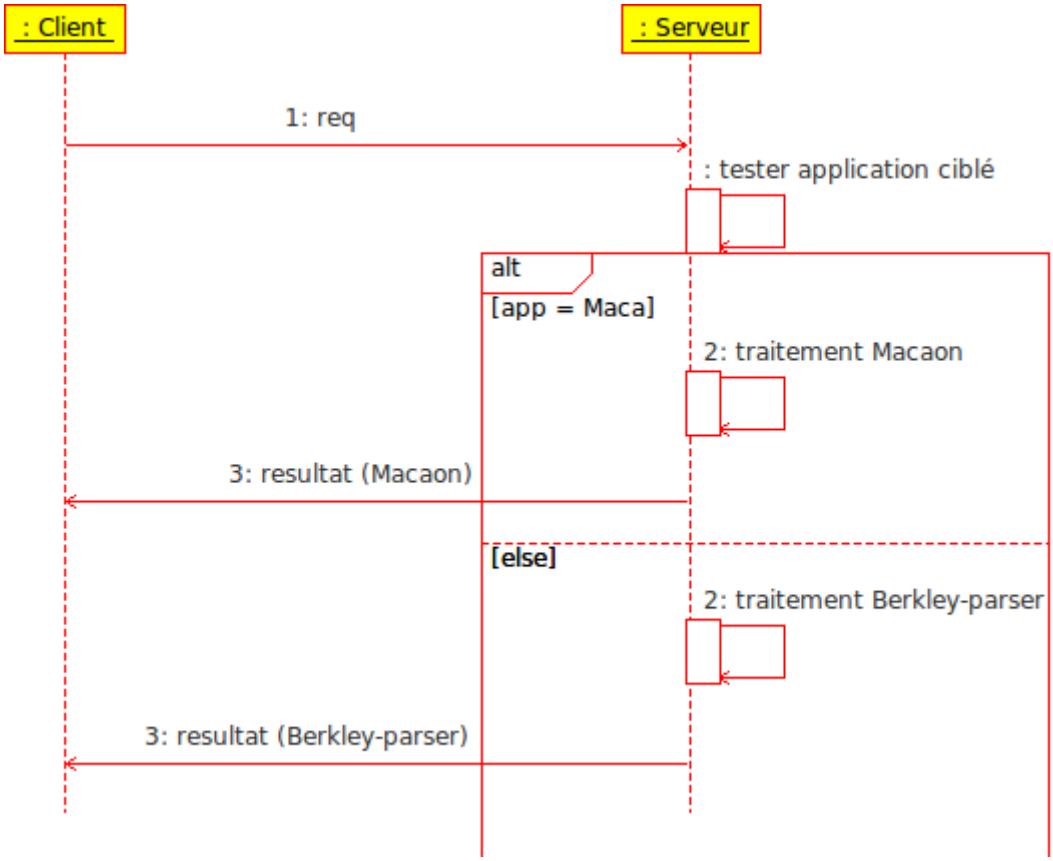
- Analyser et nettoyer chaque requête reçu .
- Limiter le nombre de requêtes possibles dans un intervalle de temps pour pénaliser les scripts qui tentent de saturer le serveur.

Performances :

- créer une file d'attente pour chaque client qui sert a stockés les requêtes qui ne sont pas encore traitées, si la file contient plusieurs requêtes le serveur ne va traiter que la dernière requête effectuée par le client.

12.4.Version-4.0.0

Dans cette version Le but est de profiter du serveur et de le modifier de tel façon qu'il puisse être démonstrateur pour plusieurs applications et non pas seulement pour MACAON .



L'idée est de créer un client pour chaque application et que chaque client spécifie l'application ciblée dans la requête envoyée vers le serveur.

Le serveur traite la requête selon l'application ciblée et renvoie le résultat au client.

13.Découpage du projet en lots

Client :

- Interface.
- Envoie de requêtes instantané.
- Modifications DOM.
- Génération du Graphe coté client.

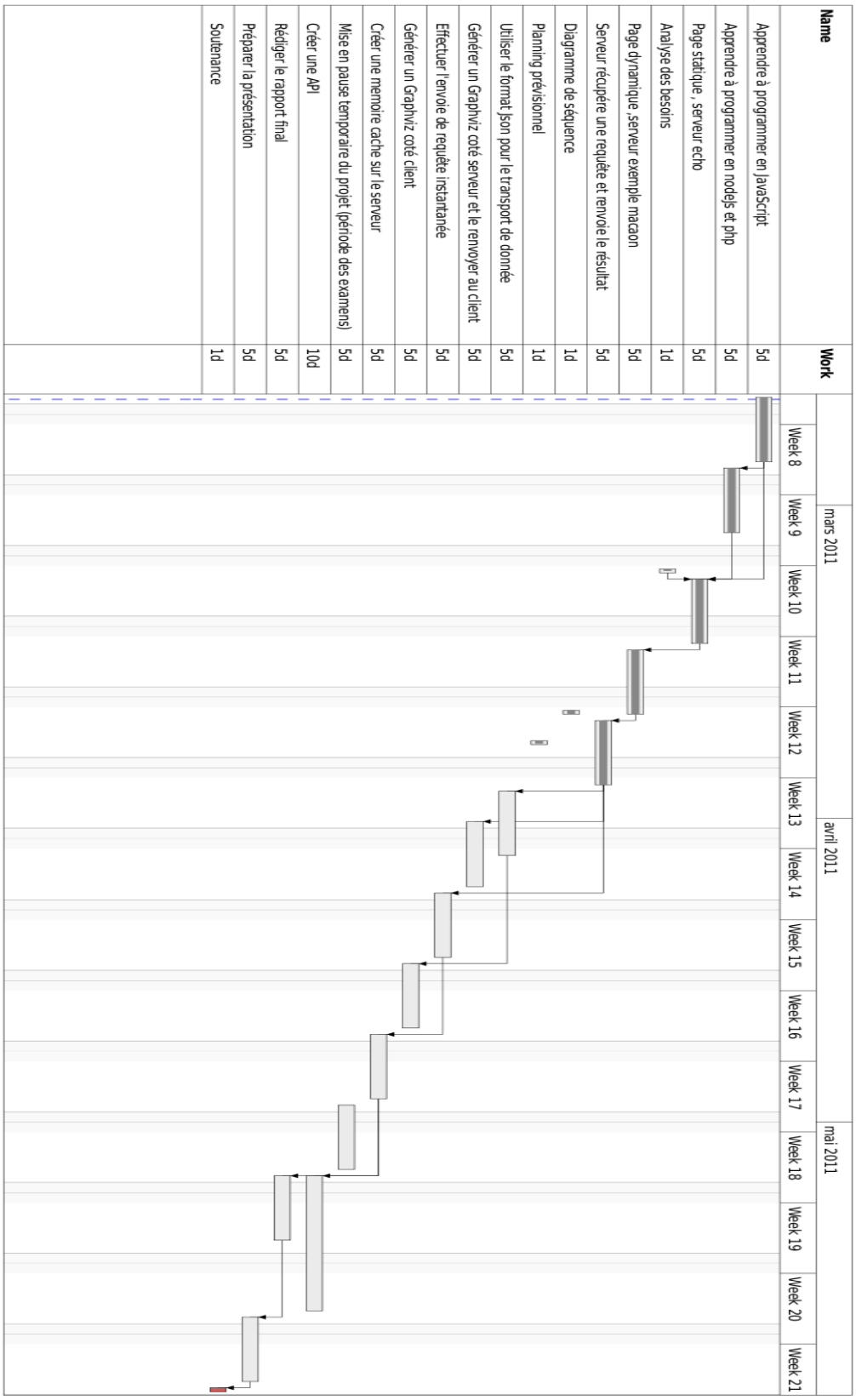
Serveur :

- Interface serveur/macaon.
- Mémoire cache.
- Gestion clients.
- Sécurité.

Communication client-serveur :

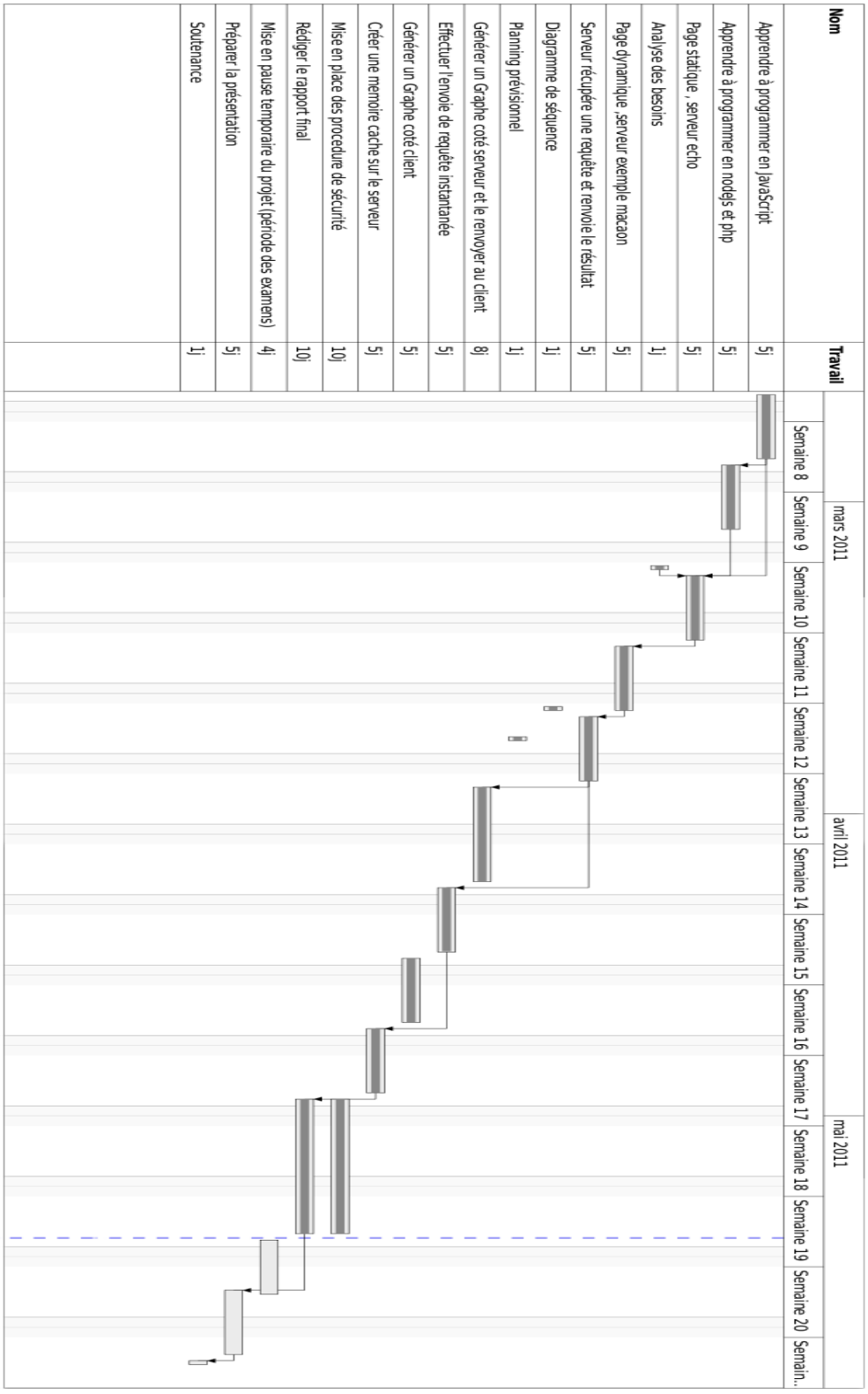
- Établir connexion client serveur.
- Protocole de communication entre le client et le serveur.

14.Planning prévisionnel



WBS	Name	Start	Finish	Work	Duration	Slack	Cost	Assigned to	% Complete
1	Apprendre à programmer en JavaScript	févr. 18	févr. 24	5d	5d	26d	0		100
2	Apprendre à programmer en nodeJs et php	févr. 25	mars 3	5d	5d	26d	0		100
3	Page statique , serveur echo	mars 8	mars 14	5d	5d	24d	0		100
4	Analyse des besoins	mars 7	mars 7	1d	1d	24d	0		50
5	Page dynamique ,serveur exemple macaon	mars 15	mars 21	5d	5d	24d	0		100
6	Serveur récupére une requête et renvoie le résultat	mars 22	mars 28	5d	5d	24d	0		100
7	Diagramme de séquence	mars 21	mars 21	1d	1d	49d	0		100
8	Planning prévisionnel	mars 24	mars 24	1d	1d	46d	0		80
9	Utiliser le format Json pour le transport de donnée	mars 29	avril 4	5d	5d	34d	0		0
10	Générer un Graphviz coté serveur et le renvoyer au client	avril 1	avril 7	5d	5d	36d	0		0
11	Effectuer l'envoie de requête instantanée	avril 8	avril 14	5d	5d	16d	0		0
12	Générer un Graphviz coté client	avril 15	avril 21	5d	5d	26d	0		0
13	Créer une memoire cache sur le serveur	avril 22	avril 28	5d	5d	10d 5h	0		0
14	Mise en pause temporaire du projet (période des examens)	avril 29	mai 5	5d	5d	16d	0		0
15	Créer une API	mai 6	mai 19	10d	10d	6d	0		0
16	Rédiger le rapport final	mai 6	mai 12	5d	5d	5d	0		0
17	Préparer la présentation	mai 20	mai 26	5d	5d		0		0
18	Soutenance	mai 27	mai 27	1d	1d		0		0

15.Planning réel



TPÉ	Nom	Démarré	Terminé	Travail	Durée	Latitude	Coût	Assigné à	État d'avancement (%)
1	Apprendre à programmer en JavaScript	févr. 18	févr. 24	5j	5j	19j	0		100
2	Apprendre à programmer en nodeJs et php	févr. 25	mars 3	5j	5j	18j 5h	0		100
3	Page statique , serveur echo	mars 8	mars 14	5j	5j	16j 5h	0		100
4	Analyse des besoins	mars 7	mars 7	1j	1j	16j 5h	0		100
5	Page dynamique ,serveur exemple macaon	mars 15	mars 21	5j	5j	16j 5h	0		100
6	Serveur récupére une requête et renvoie le résultat	mars 22	mars 28	5j	5j	16j 5h	0		100
7	Diagramme de séquence	mars 21	mars 21	1j	1j	47j	0		100
8	Planning prévisionnel	mars 24	mars 24	1j	1j	44j	0		100
9	Générer un Graphe coté serveur et le renvoyer au client	mars 29	avril 7	8j	8j	34j	0		100
10	Effectuer l'envoie de requête instantanée	avril 8	avril 14	5j	5j	8j	0		99
11	Générer un Graphe coté client	avril 15	avril 21	5j	5j	24j	0		100
12	Créer une memoire cache sur le serveur	avril 22	avril 28	5j	5j	3j	0		100
13	Mise en place des procedure de sécurité	avril 29	mai 12	10j	10j	9j	0		100
14	Rédiger le rapport final	avril 29	mai 12	10j	10j	3j	0		100
15	Mise en pause temporaire du projet (période des examens)	mai 13	mai 18	4j	4j	5j	0		0
16	Préparer la présentation	mai 18	mai 24	5j	5j		0		0
17	Soutenance	mai 25	mai 25	1j	1j		0		0

16.Hébergement de l'application

L'application est hébergé sur la machine macaweb.lif.univ-mrs.fr (IP:139.124.22.36).

Pour héberger l'application on a utilisé un accès ssh depuis notre machine personnelle vers la machine sol et depuis cette dernière vers la machine macaweb.

L'accès ssh nous offre un shell distant pour pouvoir copier les fichier sur le machine macaweb, lancer le programme serveur et installer les application utilisé par le serveur.

17.Solutions aux problèmes rencontrés

Au cours du projet, nous avons rencontré de nombreux problèmes dont nous avons trouvé la solution pour la majorité.

Parmi ces problèmes :

Problème : same origin policy .

Description : « la politique de même origine »est un élément important de sécurité pour un certain nombre de langages de programmation côté client qui empêche les pages qui appartiennent aux sites différents de communiquer.

Nous avons rencontré ce problème pendant les tests du démonstrateur sur la machine local.

Solutions : Modifier le programme serveur pour qu'il renvoie le programme client, du coup le navigateur ne va pas bloquer les message entre le client et le serveur car ils auront le même nom de domaine.

Problème : Difficulté de l'intégration de canviz dans l'application .

Description : nous avons rencontrés quelques bugs en utilisant la bibliothèque canviz.

Solutions :

- Utiliser le débogueur de google chrome qui offre des outils pour détecter les erreurs dans les programmes JavaScript et débugger le code dans le navigateur.

Problème : sécurité du serveur

Description : au cours du développement de l'application, l'hébergement de cette dernière peut causé des failles de sécurité, permettant a des pirates de pénétrer le système .

Solutions :

- Limité l'accès au serveur pendant les phases de tests et le rendre possible que depuis la machine Sol (IP:139.124.14.122).
- Comme le serveur n'est accessible que depuis la machine sol, on lance le serveur sur la machine macaweb.lif.univ-mrs.fr (IP:139.124.22.36) et on crée un tunnel ssh pour pouvoir accédé au site depuis un navigateur Web sur notre machine personnelle.
- Arrêter le programme serveur après les tests effectué.
- Pénaliser chaque client qui tente d'effectuer un très grand nombre de requêtes et de saturer le serveur.

Problème : Incompatibilité entre les données généré par macaviz (DOT) et utiliseé par canviz (GV) .

Description : Les deux formats DOT et GV sont très proches mais ils ont quelques différences importantes pour la bibliothèque canviz.

Solutions :

- Convertir les fichiers DOT en GV en utilisant la commande « dot -Txdot -o fichier.gv fichier.dot »

Problème : l'Intégration de l'application berkley-parser dans le démonstrateur.

Description : Pour analyser une phrase, l'application berkley-parser doit être exécuté avec la phrase a analysé en entrée et un fichier de grammaire en paramètre.

Donc l'application charge le fichier de grammaire qui a une taille très importante a chaque exécution. Cela ralenti le résultat a peut prés de 5 secondes.

Solutions : Modifier l'application berkley-parser afin qu'elle puisse s'exécuter, attendre sur son stdin une phrase a analyser, afficher ensuite le résultat et en fin attendre a nouveau une autre phrase a analyser et ainsi de suite.

En nodeJs on exécute le berkley-parser, on récupère son stdin et a chaque requête reçu on écrit la phrase a analyser dans le stdin récupéré cela permet une exécution sans occasionner le rechargement du fichier de grammaire.

Problème : difficultés pour installer MACAON sur la machine Macaweb.

Description : l'application MACAON repose sur plusieurs bibliothèques qui doivent être installées,

certaines sont difficiles a installer sur le serveur.

Solution :modifier MACAON et supprimer les fonctionnalités qui ne sont pas utiles pour le démonstrateur, cela supprime des dépendances de plusieurs bibliothèques.

Exemple : le composant macaviz dépend de la bibliothèque gtk qui gère l'affichage graphique.
Cette fonctionnalité est inutile pour notre démonstrateur.
En supprimant les fonctionnalités utilisant la bibliothèque gtk on aura pas besoin d'installer cette bibliothèque.

18.Réunions

Pour assurer l'avancement et l'évolution constante du projet avec un suivi du professeur encadrant on a organisé des réunions hebdomadaire avec lui .

Ensemble des réunions :

- Réunion N°1 le 11/02/2011 :
    - Commencement du projet.
    - Présentation et explication générale du sujet.
    - Discuter sur l'organisation du projet.
    - Questions divers.

- Réunion N°2 le 18/02/2011 :
    - Analyser les besoins.
    - Mettre le point sur les technologies a utiliser.
    - Éclaircissement concernant le fonctionnement de l'application.
    - Explication détaillée sur la première version a réaliser.
    - Questions divers.

- Réunion N°3 le 25/03/2011 :
    - Installation des composants MACAON.
    - Teste de l'application MACAON.
    - Présentation de la maquette.
    - Questions divers.

- Réunion N°4 le 31/03/2011 :
    - Présentation de la première version de l'application avec quelques dis-fonctionnements.
    - Résolution de quelques problèmes.
    - Éclaircissement concernant le fonctionnement de nodeJs.
    - Questions divers.

- Réunion N°5 le 04/04/2001 :
    - Présentation d'une première version fonctionnelle.
    - Explication détaillée sur la deuxième version et les améliorations a apporter.
    - Questions divers.

- Réunion N°6 le 13/04/2011 :
    - Présentation d'un prototype de la deuxième version.
    - Discuter sur les procédures de sécurités et les améliorations a apporter.
    - Questions divers.

- Réunion N°7 le 20/04/2011 :
    - Présentation d'une deuxième version fonctionnelle.
    - Explications pour l'hébergement de l'application sur la machine Macaweb.
    - Questions divers.

- Réunion N°8 le 26/04/2011 :
    - Résolution de quelques problèmes rencontrés pendant la réalisation de la troisième version.
    - Résolution des problèmes rencontrés pondant l'hébergement de l'application.

- Réunion N°9 le 10/05/2011 :
    - Présentation d'une troisième version fonctionnelle.
    - Correction du rapport et quelques propositions pour l'améliorer.

- Réunion N°10 le 12/05/2011 :
    - Mettre un dernier point sur le rapport.

19.Perspectives

Afin d'améliorer le démonstrateur, nous aurions pu développer notre propre API, qui a pour but de faciliter aux développeurs l'intégration du démonstrateur dans leur site Web.
De plus ce démonstrateur pourra participer à l'amélioration de l'application MACAON en incluant un rapport de bug de l'application.
Cependant, le temps et la complexité de ces tâches ont été les principaux facteurs qui nous ont rebuté.
Ces parties pourront peut être faire l'objet d'un autre projet.
De plus ce démonstrateur et indépendant de MACAON et il peut être utilisé avec d'autres applications.
Pour pouvoir reprendre ce projet il faudra suivre les étapes suivantes :

- Installer nodeJs : https://github.com/joyent/node/wiki/Installation .
- Installer MACAON : http://macaon.lif.univ-mrs.fr/index.php?page=installation .
- Récupérer la dernière version du projet qui se trouve sur le dépôt : https://github.com/oualid13/MacaWeb.
- Suivre les instructions pour exécuter un programme nodeJs : http://nodejs.org/ .
- Se documenter sur nodeJs : http://nodejs.org/docs/v0.4.7/api/all.html.

20.Bibliographie

NodeJs :http://nodejs.org/ .

Documentation officielle NodeJs : http://nodejs.org/docs/v0.4.5/api/all.html .

Tutoriel de Javascript : http://www.siteduzero.com/tutoriel-3-309961-dynamisez-vos-sites-web-avec-javascript.html

Canviz : http://code.google.com/p/canviz/ .

JSON : http://www.json.org/ .

Prototype : http://www.prototypejs.org/ .

Site officiel du projet :http://macaweb.lif.univ-mrs.fr .

Dépôt du projet : https://github.com/oualid13/MacaWeb .

Site officiel de MACAON : http://macaon.lif.univ-mrs.fr .

21.Conclusion

Nous avons réalisé un démonstrateur Web pour l'outil MACAON dans le cadre notre de projet de fin d'année du Master 1 en informatique. Ce démonstrateur permet de faciliter l'utilisation de cette application et de la publier au grand public .

La réalisation de ce projet nous a permis d'apprendre de nouvelles technologies Web et d'appliquer les connaissances qui nous ont été inculquées au cours de cette année de Master Informatique à l'université de la méditerranée Aix-Marseille II .

Nous avons été confrontés à de nombreux problèmes et dans la plupart des cas nous avons pu trouver une solution afin de les résoudre .

Enfin ce projet était l'occasion de découvrir et d'utiliser des outils dont nous n'avions pas la moindre idée de leurs existences.