



Antoine de ROQUEMAUREL

Stage effectué chez Memobox – 2G technologies
Du 10 Avril 2012 au 22 Juin 2012

Pour M. Denis MALLET

Pour M. Patrick MAGNAUD

Rapport de stage

Refonte du système multilingue pour une application web

— Annexes —



Memobox – 2G Technologies
12 Bv de l'Europe
31850 – MONTRABÉ

Antoine de ROQUEMAUREL

Stage effectué chez Memobox – 2G technologies
Du 10 Avril 2012 au 22 Juin 2012

Pour M. Denis MALLET
Pour M. Patrick MAGNAUD

Rapport de stage

Refonte du système multilingue pour une application web

— Annexes —

Memobox – 2G Technologies
12 Bv de l'Europe
31850 – MONTRABÉ



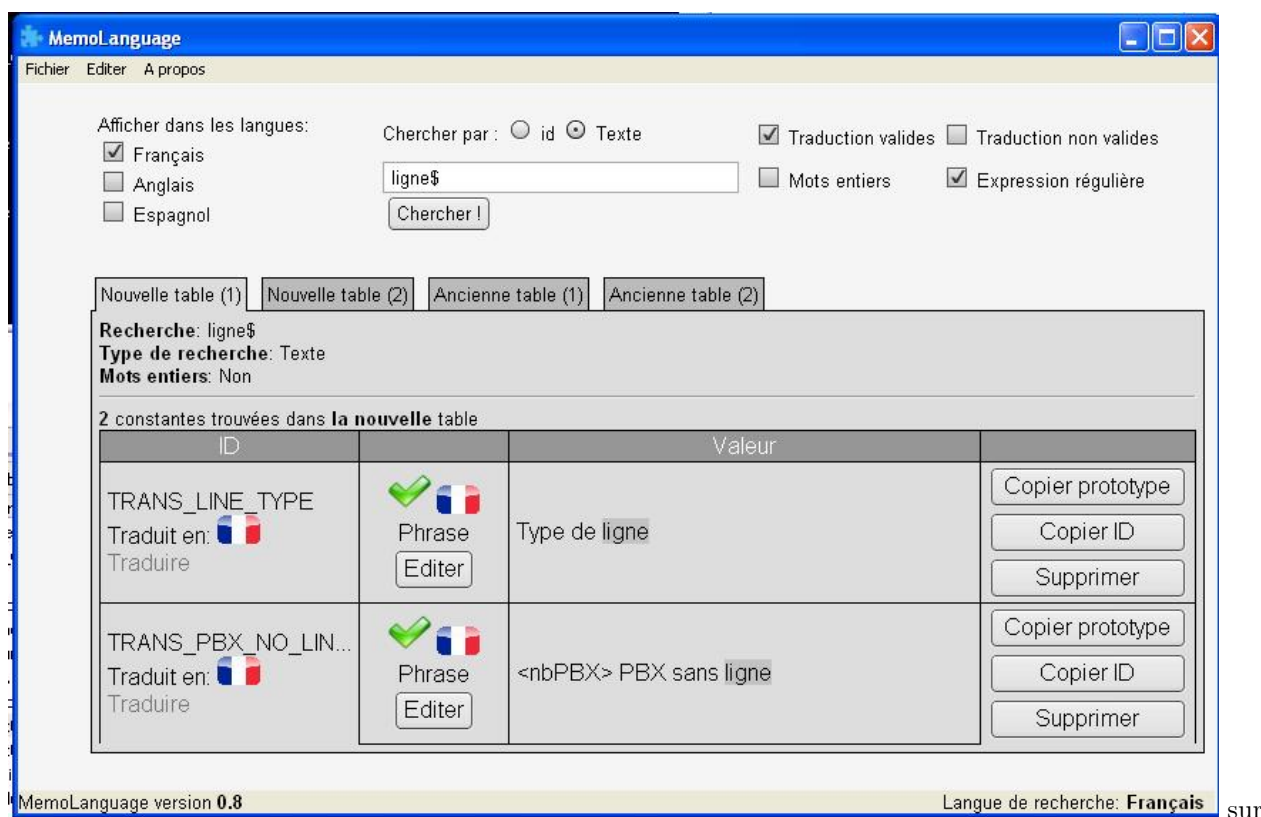
Table des annexes

A	Glossaire	7
B	Liste des figures	9
C	Liste des tableaux	11
D	Liste des codes sources	13
E	Code source	15
E.1	Moteur de traduction	15
E.2	Memolanguage	21

Résultats

A.1 MemoLanguage

A.1.1 Recherche



sur cette capture, on peut voir les différentes case à cocher permettant d'affiner la recherche. Également, 4 onglets sont présents, suivant sur l'onglet où on se trouve, cela cherchera dans la nouvelle table ou dans l'ancienne table, comme indiqué par l'onglet.

La présence de plusieurs onglets, permet de ne pas effacer une recherche qui est intéressante.

FIGURE A.1. Recherche avec expression régulière

A.1.2 Édition

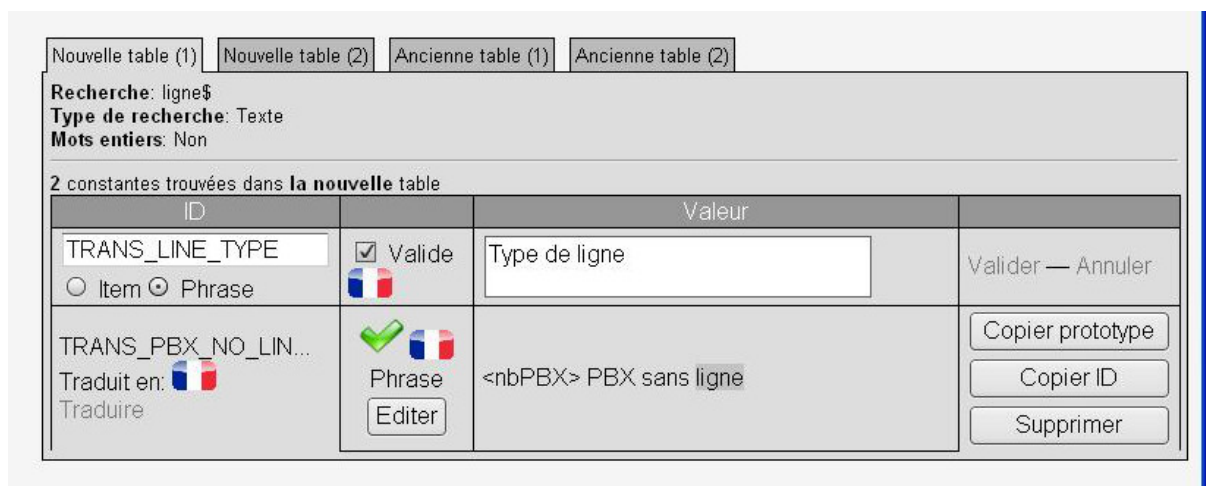
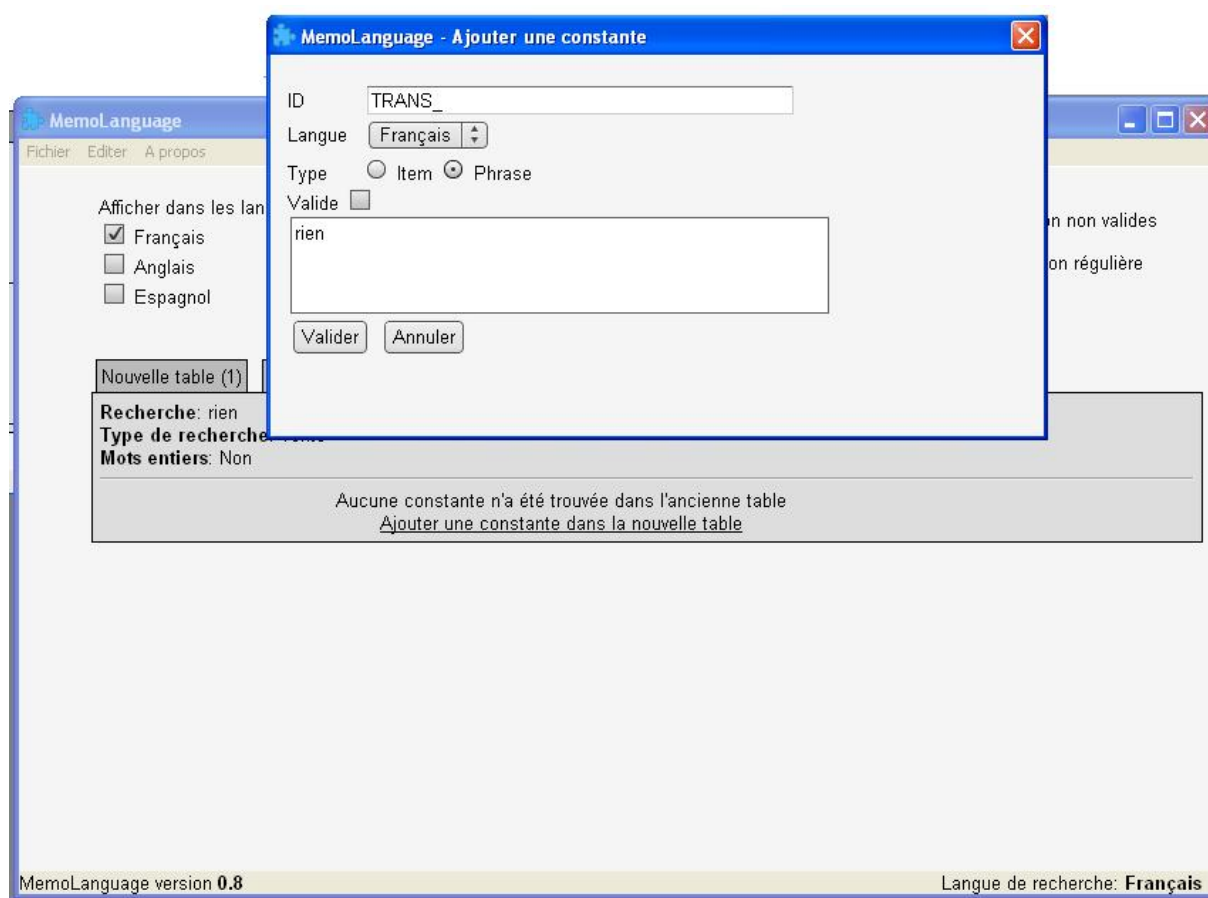


FIGURE A.2. Édition d'une constante

A.1.3 Ajout de constante



Sur cette capture d'écran, l'utilisateur a effectué la recherche "rien" et n'a trouvé aucune correspondance, ainsi il a cliqué sur le lien "Ajouter une constante dans la nouvelle table" ce qui lui a ouvert la fenêtre avec le champ de valeur pré-rempli avec "rien". Le nom d'ID est également pré-rempli avec TRANS_ afin d'inciter les développeurs à utiliser ce préfixe.

FIGURE A.3. Ajouter d'une constante après une recherche infructueuse

A.1.4 Suppression



FIGURE A.4. Suppression d'une constante

A.1.5 Copier vers nouvelle table

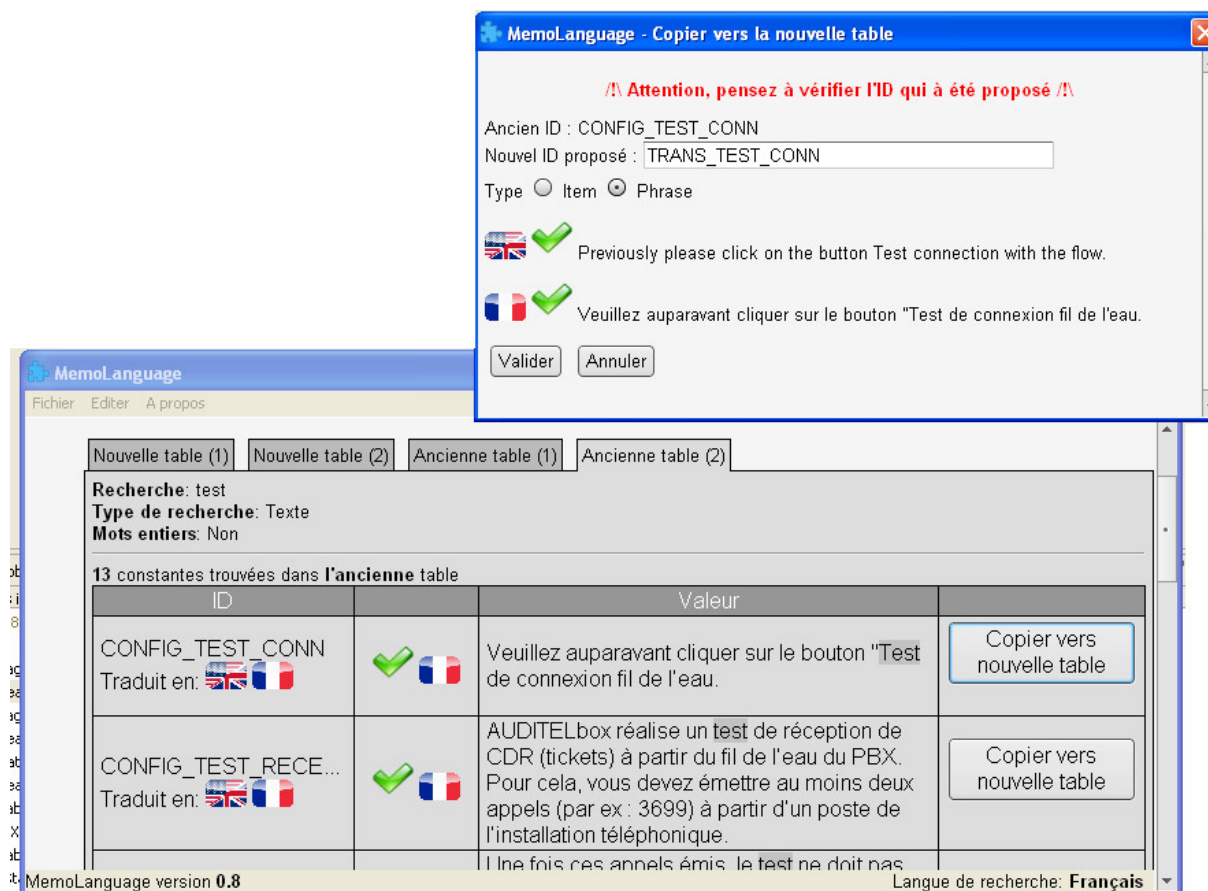


FIGURE A.5. Basculer une constante de l'ancienne base vers la nouvelle

Dans cette capture d'écran, on remarque qu'automatiquement un nouvel ID est proposé afin de garder la cohérence avec les préfixe, cependant une invitation à vérifier et indiquer au développeur.

A.1.6 Traduction

Nouvelle table (1) Nouvelle table (2) Ancienne table (1) Ancienne table (2)

Recherche: ligne\$
Type de recherche: Texte
Mots entiers: Non

2 constantes trouvées dans la nouvelle table

ID		Valeur	
TRANS_LINE_TYPE ○ Item ○ Phrase	<input checked="" type="checkbox"/> Valide 🇫🇷	Type de ligne	Valider — Annuler
TRANS_PBX_NO_LIN... Traduit en: 🇫🇷 🇫🇷 Traduire	<input checked="" type="checkbox"/> Phrase 🇫🇷 Editer	<nbPBX> PBX sans ligne	Copier prototype Copier ID Supprimer

FIGURE A.6. Traduction d'une constante

A.1.7 Champs TranslatedText_short et TranslatedText_long

MemoLanguage - Ajouter une constante

ID: TRANS_

Langue: Français

Type: ☐ Item ☒ Phrase

Valide: ☐

Champ normal:

Champ court:

Ajouter un champ long

Valider Annuler

FIGURE A.7. Insertion d'une constante

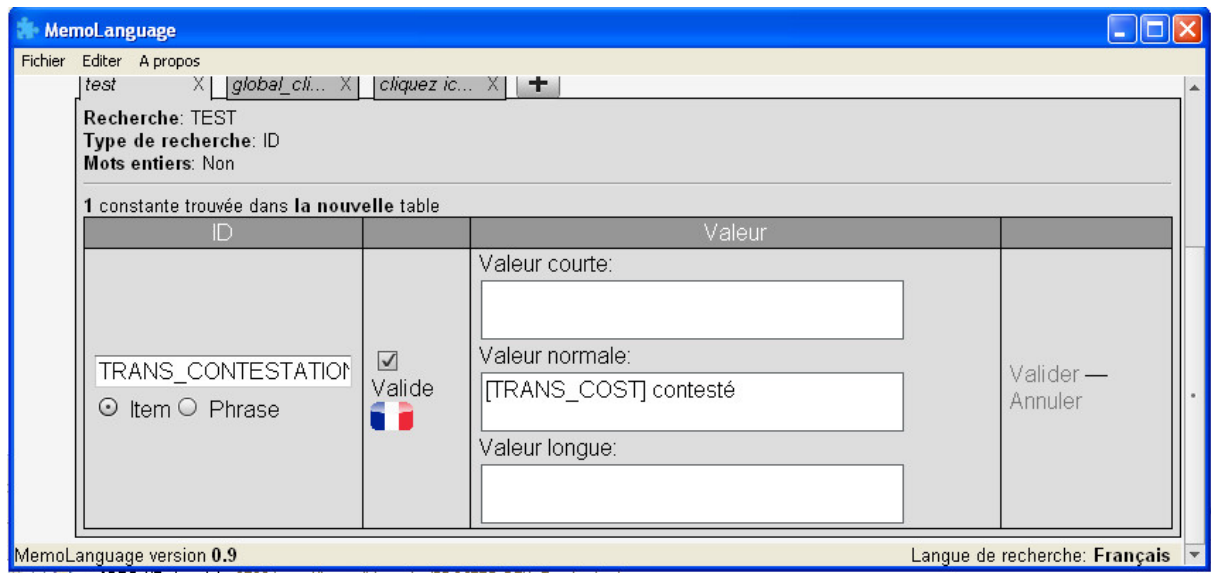


FIGURE A.8. Édition d'une constante

A.2 AUDITELcom multilingue

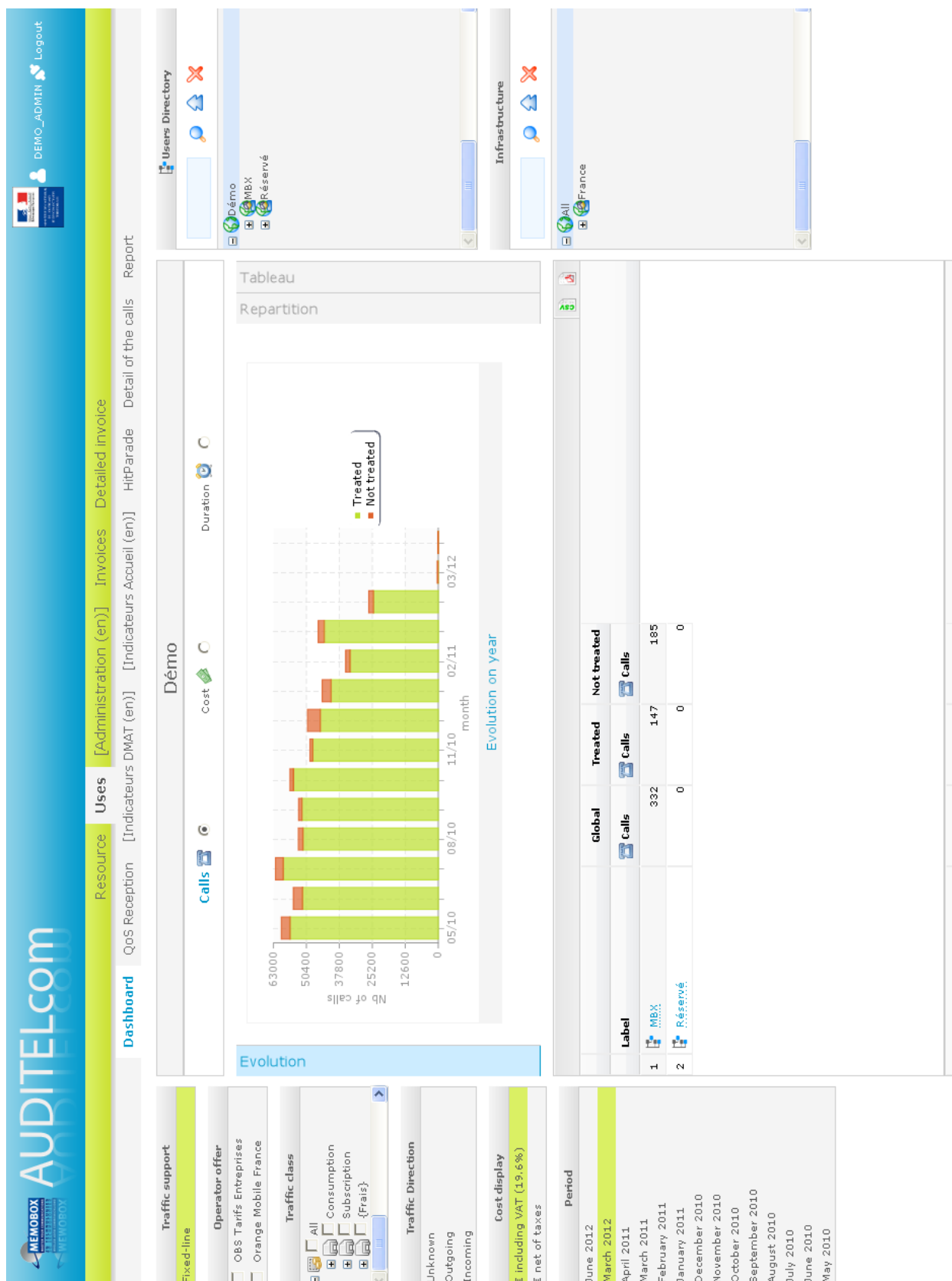


FIGURE A.9. AUDITELcom en anglais

Comme vous pouvez le voir, l'intégration n'est pas finit, certains mots s'affichent en français, cependant une grande partie du travail à été fait.

Glossaire

Adobe AIR Adobe Integrated Runtime, anciennement nommé Apollo, est une machine virtuelle multiplateforme, multilangage, multi interface qui s'exécute sur le système d'exploitation, qui permet de développer une application native avec des technologies Web.

AJAX Asynchronous Javascript and XML, c'est un ensemble de technologies destinées à réaliser de rapides mises à jour du contenu d'une page Web, sans qu'elles nécessitent le moindre rechargement visible par l'utilisateur de la page Web. Les technologies employées sont diverses et dépendent du type de requêtes que l'on souhaite utiliser, mais d'une manière générale le Javascript est constamment présent.

CSS Cascading Style Sheets, un langage permettant de décrire la présentation des documents HTML.

Expressions régulières Chaîne de caractères qui décrit un ensemble de chaînes de caractères possibles selon une syntaxe précise. Elles sont aujourd'hui utilisées par les informaticiens dans l'édition et le contrôle de texte ainsi que dans la manipulation des langues.

GFT Gestion Financière des Télécoms(TEM en anglais). Ensemble des outils et ressources qui ont pour objectif d'optimiser et de rationaliser la gestion des dépenses télécoms (Mobile / Fixe, Voix / data, VoIP) des entreprises.

JavaScript Langage de programmation de scripts principalement utilisé dans les pages web interactives côté client.

HTML Hypertext Markup Language, format de données conçu pour représenter les pages web.

Méthodes Agiles Groupes de pratiques pouvant s'appliquer aux projets de développement en informatique. Les méthodes agiles impliquent au maximum le client et permettent une grande réactivité à ses demandes. Elles visent la satisfaction réelle du besoin du client en priorité aux termes d'un contrat de développement.

MySQL Système de gestion de base de données libres, principalement couplé à des serveurs web.

PHP Hypertext Preprocessor, langage de scripts libre principalement utilisé pour produire des pages web dynamiques via un serveur HTTP.

SaaS Software as a Service, technologie consistant à fournir des services ou des logiciels informatiques par le biais du Web et non plus dans le cadre d'une application de bureau ou client-serveur.

SQL Structured Query Language, pseudo langage informatique (de type requête) standard et normalisé, destiné à interroger ou à manipuler une base de données relationnelle.

Liste des figures

1.1	Organigramme du service R&D à Toulouse	13
2.1	AUDITELcom V7	15
2.2	Table RES_Translations	16
2.3	Schéma des constantes	18
2.4	Le problème qui sera évité avec les items et les phrases.	19
2.5	Table RES_dicoLanguage	19
2.6	Architecture client – serveur	22
2.7	Diagramme de classe du serveur	23
2.8	Table RES_dicoLanguage	27

Liste des tableaux

2.1	Avantages et inconvénients du système	17
2.2	Exemple de constante	20
2.3	Exemple de variables	20
2.4	Exemple avec du pluriel/singulier	21

Liste des codes sources

2.1	Exemple d'appels de getTranslation	17
2.2	Exemple de constante – Code	20
2.3	Exemple de variables – Code	20
2.4	Exemple avec du pluriel/singulier – Code	21
2.5	Exemple de variables courtes ou longues	27
E.1	TLanguage.class.php	15
E.2	database.class.php	21
E.3	databaseSearch.class.php	22
E.4	index.php	25
E.5	page.class.php	26
E.6	pageselect.class.php	26
E.7	search.js	28
E.8	actionTranslate.js	33
E.9	Page HTML principale homeSearch.html	34
E.10	Descripteur de fichier MemoLanguage-app.xml	36

Code source

F.1 Moteur de traduction

```

1 <?php
2 class TLanguage {
3     private static $instance;
4     private $dico = array();
5     private $sTranslate = '';
6
7     public static function singleton() {
8         if (!isset (self :: $instance)) {
9             $c = __CLASS__;
10            self :: $instance = new $c ();
11            return (self :: $instance);
12        }
13        return (self :: $instance);
14    }
15
16    public static function TraducExist($xsLangue) {
17        return (($xsLangue == 'fr') || ($xsLangue == 'en') || ($xsLangue == 'es'))
18            ;
19    }
20
21    //Necessaire pour l'API gOwlReport
22    public function getLang($xsValue,$xbDebugMode=false){
23        $bEtatDebugMode = DEBUG_MODE;
24        if($xbDebugMode)
25            $this->_xbDebugMode = TRUE;
26        $sTraduction = $this->getTranslation($xsValue);
27        $this->_xbDebugMode = $bEtatDebugMode;
28        return $sTraduction;
29    }
30
31    private function pushInDico($idRes, $xaValue) {
32        if(isset($xaValue['Language'])) {
33            $this->dico[$idRes] = $xaValue['Language'];
34            $this->dico[$idRes]['TYPE_RES'] = $xaValue['Type_RES'];
35        }
36    }
37
38    /**
39     * Retourne la langue de traduction
40     */
41    public function getCurrentLanguage() {
42        $sLangReturn = '';
43        $access_manager = TDataAccessManager :: singleton();

```

```

44     if ($access_manager->sessionLoaded()) { // on recupere la langue de la
        session
45         $sLangReturn = $access_manager->getContextManager(CONTEXT_LANGUE,
            ALL_MODULE);
46         if (!$sLangReturn) {
47             try {
48                 $G_Langue = $access_manager->getWebObject('G_Langue');
49                 $sLangReturn = $G_Langue->getText();
50             } catch (exception $e) {
51                 $sLangReturn = null;
52             }
53
54         }
55     } else { // on repere la langue du navigateur
56         $sLangReturn = ($this->returnLanguage());
57         if (!self::TraducExist($sLangReturn))
58             $sLangReturn = LANG_DEFAULT;
59     }
60
61     return ($sLangReturn);
62 }
63 /**
64  * Retourne un array traduit contenant toutes les traductions
65  * @param $xaId Array contenant des constantes de traduction.
66  * @param $aMapping
67  * @return L'array contenant toutes les traductions
68  */
69 public function getArrayTranslate($xaId, $aMapping = NULL){
70     $aResult = array();
71     if(is_array($xaId)){
72         if(TArray::isArraySetAndSupTo0($xaId)){
73             foreach($xaId as $ind => $keyVal){
74                 $keyVal = (($aMapping != NULL) && array_key_exists($keyVal,$aMapping
75                     ))? $aMapping[$keyVal] : $keyVal;
76                 $aResult[$ind] = $this->getTranslation($keyVal);
77             }
78         }else{
79             $keyVal = (($aMapping != NULL) && array_key_exists($xaId,$aMapping))?
80                 $aMapping[$xaId] : $xaId;
81             $aResult = $this->getTranslation($keyVal);
82         }
83     }
84     return $aResult;
85 }
86 /**
87  * Retourne une chaine traduite, on peut faire passer les variables
88  * a integrer dans la traduction
89  * soit en parametree de methode getTranslation('TRANS_MONID', var1, var2,
90  * varn) ;
91  * soit integrees dans l'ID getTranslation('TRANS_MONID'{$var1, var2, varn
92  * }')');
93  * @param $xIdToTranslate
94  * @return Chaine traduite
95  */
96 public function getTranslation($xIdToTranslate) {
97     $sRetour = '';
98     $i = 1;
99     $typeWordToTranslate = '';

```



```

97 $variables = func_get_args();
98 // Si on a fait passer les variables dans un tableau
99 if(isset($variables[1]) && is_array($variables[1])) {
100     $aVarBuff = $variables[1];
101     foreach($aVarBuff as $var) {
102         $variables[$i] = $var;
103         ++$i;
104     }
105 }
106
107 // on repere le type de mot a afficher (court long ou normal)
108 $typeWordToTranslate = $this->getTypeWord($xIdToTranslate);
109
110 //Si les variables sont passe en ID grace a {var1, var2, var3}}
111 if(strpos($xIdToTranslate, '{')!==false){
112     $aVariables = array();
113     $aVariablesBuff = array();
114     preg_match_all('#{(.+)}#isU', $xIdToTranslate, $aVariablesBuff);
115     $xIdToTranslate = preg_replace('#{(.+)}#isU', '', $xIdToTranslate);
116
117     $sVariablesToSplit = $aVariablesBuff[1][0];
118     if(strpos($sVariablesToSplit, ',')!==false){
119         $aVariables = explode(',', $sVariablesToSplit);
120     } else {
121         $aVariables = $sVariablesToSplit;
122     }
123
124     return ($this->getTranslation($xIdToTranslate, $aVariables));
125 }
126
127 // si on concatene plusieurs constante avec CONST1++ ++CONST2
128 if(strpos($xIdToTranslate, '++')!==false){
129     $aId_res = explode('++', $xIdToTranslate );
130     $forbidden='\"\'\\?()*:./@|<>-';
131     foreach($aId_res as $id_res_temp){
132         $sRetour .= (strcspn(trim($id_res_temp), $forbidden) !== 0)? $this->
            getTranslation($id_res_temp) : $id_res_temp;
133     }
134     return $sRetour;
135 }
136
137 //Si la cle existe en Constante elle est utilisee en priorite
138 if (defined($xIdToTranslate)) {
139     if (defined($xIdToTranslate.'_').$this->getCurrentLanguage())
140         $xIdToTranslate = $xIdToTranslate.'_'. $this->getCurrentLanguage();
141     return constant($xIdToTranslate);
142 }
143
144 // si la clef n'est pas dans le dico, on la cherche et on l'ajoute au dico
145 if(!isset($this->dico[$xIdToTranslate])) {
146     $access_manager = TDataAccessManager :: singleton();
147     $datalink = $access_manager->getDataLink();
148     $value = $datalink->RES_getDicoTranslation($xIdToTranslate, $this->
        getCurrentLanguage());
149     $this->pushInDico($xIdToTranslate, $value);
150 }
151
152 // si la clef est trouve dans le dico, on peut l'afficher
153 if(isset($this->dico[$xIdToTranslate][$this->getCurrentLanguage()])) {

```

```

154     $$Retour = $this->getValueKeyExist($xIdToTranslate, $typeWordToTranslate
155         );
156 } else {
157     $$Retour = $this->getValueKeyNotExistInCurrentLang($xIdToTranslate,
158         $typeWordToTranslate);
159 }
160
161 // on parse toutes les variables et constantes
162 $$Retour = $this->parserVariables($$Retour, $variables);
163 $constants = $this->getConstants($$Retour);
164 if(isset($this->dico[$xIdToTranslate][$this->getCurrentLanguage()]))
165     $$Retour = $this->parserConstantes($$Retour, $constants);
166 else
167     $$Retour = $this->parserConstantes($$Retour, $constants, 'fr');
168
169 return $$Retour;
170 }
171 /**
172  * Retourne la valeur d'une clef existante dans le dico en fonction du type
173  * (long short ou normal)
174  * @param String $xIdToTranslate La clef
175  * @param String $xsTypeWordToTranslate Le type
176  */
177 private function getValueKeyExist($xIdToTranslate, $xsTypeWordToTranslate,
178     $xsLang='') {
179     if($xsLang == '') {
180         $xsLang = $this->getCurrentLanguage();
181     }
182
183     $$Retour = '';
184     if($xsTypeWordToTranslate == '#SHORT#' &&
185         $this->dico[$xIdToTranslate]['short'][$xsLang] != '') {
186         $$Retour = $this->dico[$xIdToTranslate]['short'][$xsLang];
187     } else if($xsTypeWordToTranslate == '#LONG#' &&
188         $this->dico[$xIdToTranslate]['long'][$this->getCurrentLanguage()] != '')
189     {
190         $$Retour = $this->dico[$xIdToTranslate]['long'][$xsLang];
191     } else {
192         $$Retour = $this->dico[$xIdToTranslate][$xsLang];
193     }
194
195     return $$Retour;
196 }
197 /**
198  * Retourne l'affichage d'une clef qui n'existe pas dans la langue courante
199  * @param String $xIdToTranslate La clef
200  * @param String $xsTypeWordToTranslate Le type
201  */
202 private function getValueKeyNotExistInCurrentLang($xIdToTranslate,
203     $xsTypeWordToTranslate) {
204     $$Retour = '';
205     // la clef n'existe pas pour la langue courante, on cherche si elle existe
206     // en francais
207     if(isset($this->dico[$xIdToTranslate]['fr'])) {
208         if(DEBUG_MODE)
209             $$Retour = '{' . $this->getValueKeyExist($xIdToTranslate,
210                 $xsTypeWordToTranslate, 'fr') . '}';
211         else
212             $$Retour = $this->getValueKeyExist($xIdToTranslate,
213                 $xsTypeWordToTranslate, 'fr');
214     }

```

```

205 //la clef est inexistante.
206 } else {
207     TUtil::DebugToFile('Constante '.$xIdToTranslate.' non trouve', '',
208         DEBUG_PATH.'DebugTranslation.txt', true);
209     if(DEBUG_MODE)
210         $sRetour = '='.$xIdToTranslate.'=';
211     else
212         $sRetour = $xIdToTranslate;
213 }
214
215 return $sRetour;
216 }
217 /**
218  * Retourne le type de mot d'un ID (SHORT, LONG ou NORMAL)
219  * @param String $xIdToTranslate L'id
220  */
221 private function getTypeWord($xIdToTranslate) {
222     $sRetour = 'NORMAL';
223     if(strpos($xIdToTranslate, '#')!==false){
224         $aTypeWordBuff = array();
225         if(preg_match_all('#\#(.+)\#isU', $xIdToTranslate, $aTypeWordBuff) ==
226             1) {
227             $sRetour = $aTypeWordBuff[0][0];
228         }
229     }
230 }
231
232 return $sRetour;
233 }
234
235 /**
236  * Parse une chaine de caractere en remplaçant les constantes par leurs
237  * valeurs
238  * @param String $xstring Chaine a parser
239  * @param Array $xaconstants Constantes
240  */
241 private function parserConstantes($xstring, $xaconstants, $xslanguage='') {
242     if($xslanguage == '') {
243         $xslanguage = $this->getCurrentLanguage();
244     }
245     $sRetour = $xstring;
246     $access_manager = TDataAccessManager :: singleton();
247     $datalink = $access_manager->getDataLink();
248
249     for($i=0; $i < count($xaconstants) ; ++$i) {
250         if(!isset($this->dico[$xaconstants[$i]])) {
251             $value = $datalink->RES_getDicoTranslation($xaconstants[$i],
252                 $xslanguage);
253             $this->pushInDico($xaconstants[$i], $value);
254         }
255
256         // On donne la valeur aux constantes
257         if(isset($xaconstants[$i])) {
258             if(isset($this->dico[$xaconstants[$i]][$xslanguage]) &&
259                 $this->dico[$xaconstants[$i]]['TYPE_RES'] == 'ITEM') {
260                 $sRetour = preg_replace('#\#[(.+)\#isU',
261                     $this->dico[$xaconstants[$i]][$xslanguage], $sRetour, 1);
262             } else if(isset($this->dico[$xaconstants[$i]]['fr'])) {
263                 $sRetour = preg_replace('#\#[(.+)\#isU',
264                     $this->dico[$xaconstants[$i]]['fr'], $sRetour, 1);
265             } else {

```

```

261     $$Retour = preg_replace('#\[.(+)\]#iSU', '$1', $$Retour, 1);
262 }
263 }
264 }
265
266 return $$Retour;
267 }
268
269 /**
270  * Parse une chaine de caractere en remplaçant les variables par leurs
271  * valeurs
272  * @param String $xstring Chaine a parser
273  * @param Array $xavariabes Variables
274  */
275 private function parserVariables($xstring, $xavariabes) {
276     $$Retour = $this->parserPlurialOrSingular($xstring, $xavariabes);
277     for($i=1; $i < count($xavariabes) ; ++$i) {
278         // On donne la valeur aux variables
279         if(isset($xavariabes[$i])) {
280             $$Retour = preg_replace('#<(.+)>#iSU', $xavariabes[$i], $$Retour, 1);
281         }
282     }
283
284     return $$Retour;
285 }
286
287 /**
288  * Met au pluriel ou au singulier une chaine en fonction des variables qu'
289  * elle contient.
290  * @param String $xstring Chaine a accorder contenant des <variable> et {
291  *   singulier/pluriel}
292  * @param Array $xaVariables Tableau de variables
293  * @return La chaine accordee
294  */
295 private function parserPlurialOrSingular($xstring, $xaVariables) {
296     $aStringSplitVariables = preg_split('#<[a-z]{1,}>#i', $xstring);
297     // Met au pluriel ou au singulier
298     for($i=1; $i < count($xaVariables) ; ++$i) {
299         if(isset($aStringSplitVariables[$i])) {
300             if($xaVariables[$i] > 1) { // Pluriel
301                 $aStringSplitVariables[$i] = preg_replace('#\(((+)\)|(.+)\)#iSU', '$2
302                 ',
303                     $aStringSplitVariables[$i]);
304             } else {
305                 $aStringSplitVariables[$i] = preg_replace('#\(((+)\)|(.+)\)#iSU', '$1
306                 ',
307                     $aStringSplitVariables[$i]);
308             }
309         }
310     }
311
312     //On Reconcatene la string en remettant les <value> servants de
313     //separateurs precedemment
314     $$Retour = '';
315     $i=0;
316     foreach ($aStringSplitVariables as $astring) {
317         $$Retour .= $astring;
318         if($i < count($aStringSplitVariables)-1 ){
319             $$Retour .= '<value>';
320         }
321     }

```

```

315     ++$i;
316 }
317 return $sRetour;
318 }
319
320 /**
321  * Obtient un tableau de constantes a partir d'une chaine de caractere
322   * contenant des [constante]
323  * @param string $xsValueToParse Chaine a parser
324  */
325 private function getConstants($xsValueToParse) {
326     $aConstants = array();
327     $aConstantesBuff = array();
328     preg_match_all('#\[([.+)\\]#isU', $xsValueToParse, $aConstantesBuff);
329     $aConstants = $aConstantesBuff[1];
330
331     return $aConstants;
332 }
333
334 /**
335  * Retourne la langue du navigateur
336  */
337 private function returnLanguage() {
338     $retRL = LANG_DEFAULT; // Langue par default
339     if (isset($_SERVER)) {
340         if (isset($_SERVER['HTTP_ACCEPT_LANGUAGE'])) {
341             $retRL = explode(',', $_SERVER['HTTP_ACCEPT_LANGUAGE']);
342             $retRL = substr($retRL[0], 0, 2);
343         }
344     }
345     if (!$this->traducExist($retRL))
346         $retRL = LANG_DEFAULT;
347
348     return strtolower($retRL);
349 }
350 }
351 ?>

```

Listing F.1. TLanguage.class.php

F.2 Memolanguage

F.2.1 Le serveur

```

1 <?php
2 class DataBase {
3     public function __construct($xsDbName, $xsDbHost, $xsDbUser, $xsDbPassword)
4     {
5         $this->connect($xsDbName, $xsDbHost, $xsDbUser, $xsDbPassword);
6     }
7
8     public function connect($xsDbName, $xsDbHost, $xsDbUser, $xsDbPassword) {
9         $this->db = new PDO('mysql:host='.$xsDbHost.';dbname='.$xsDbName.'',
10             $xsDbUser, $xsDbPassword);
11         $this->db->setAttribute(PDO::ATTR_ERRMODE, PDO::ERRMODE_EXCEPTION);
12     }
13 }

```

Listing F.2. database.class.php

```

1 <?php
2 class DatabaseSearch extends Database {
3     public function __construct() {
4         parent::__construct(DBNAME, DBHOST, DBUSER, DBPASS);
5     }
6
7     public function getValueConstantsFr($xaConstants, $xsLanguage, $xsTable =
8         DBTABLENAMENOUVELLE) {
9         $aRetour = $xaConstants;
10        $i = 0;
11
12        if(isset($query)) {
13            while($data = $query->fetch()) {
14                $aRetour[$i] = $data;
15                ++$i;
16            }
17        }
18
19        return $aRetour;
20    }
21
22    public function getTranslateSearchText($xaKeyword, $xaLangResult,
23        $xaLangSearch, $xbValide, $xsTable = DBTABLENAMENOUVELLE) {
24        $aRetour = array();
25        $i = 0;
26        $fctWholeWord = util::testGet('whole');
27        $fctRegex = util::testGet('regex');
28
29        $txtQuery = "SELECT ID_RES, TranslatedText, Language, Valide ";
30        if($xsTable == DBTABLENAMENOUVELLE)
31            $txtQuery .= ", Type_RES, TranslatedText_short, TranslatedText_long ";
32
33        $txtQuery .= "FROM ".$xsTable." WHERE ID_RES IN (SELECT ID_RES FROM ".
34            $xsTable."
35            WHERE " ;
36
37        foreach($xaKeyword as $k) {
38            $sKBuff = trim($k);
39            if($fctRegex == 1) { // // $fctWholeWord == 1) {
40                $txtQuery .= "lower(TranslatedText) REGEXP '$k' ";
41            } else if($fctWholeWord == 1){
42                $txtQuery .= 'lower(TranslatedText) REGEXP \'^\'.'.$sKBuff.$sCarForbidden
43                    .'|\'.'.$sCarForbidden.$sKBuff.
44                    '$|\'.'.$sCarForbidden.$sKBuff.$sCarForbidden.'|^\'.'.$sKBuff.'$\' ';
45            } else {
46                $txtQuery .= "lower(TranslatedText) LIKE '%$k%' ";
47            }
48            if($i < count($xaKeyword) - 1)
49                $txtQuery .= ' AND ';
50
51            ++$i;
52        }
53        if($xbValide != 2)
54            $txtQuery .= "AND Valide='$xbValide'";

```

```

54     $txtQuery .= "AND Language='$xaLangSearch'");
55
56
57 for($i=0 ; $i < count($xaLangResult) ; ++$i) {
58     if($i == 0)
59         $txtQuery .= " AND (";
60     else
61         $txtQuery .= " OR ";
62
63     $txtQuery .= "Language='$xaLangResult[$i]'";
64
65     if($i == count($xaLangResult) - 1)
66         $txtQuery .= ")";
67 }
68 $txtQuery .= " ORDER BY ID_RES LIMIT 50";
69
70 $query = $this->db->prepare($txtQuery);
71
72 $query->execute();
73
74
75 while($data = $query->fetch()) {
76     $data['TranslatedText'] = Util::parserConstantes($data['TranslatedText'],
77     $data['Language'], $this);
78     if($xsTable == DBTABLENAMENOUVELLE) {
79         $data['TranslatedText_short'] = Util::parserConstantes($data['TranslatedText_short'],
80         $data['Language'], $this);
81         $data['TranslatedText_long'] = Util::parserConstantes($data['TranslatedText_long'],
82         $data['Language'], $this);
83     }
84     $aRetour[$i] = $data;
85     ++$i;
86 }
87 $query->closeCursor();
88
89 return $aRetour;
90 }
91
92 public function getTraduceEffectue($id, $xsTable = DBTABLENAMENOUVELLE) {
93     $i = 0;
94     $aRetour = array();
95     $query = $this->db->prepare("SELECT * FROM ".$xsTable."
96     WHERE lower(ID_RES) = :keyword ORDER BY Language LIMIT 50");
97     $query->execute(array('keyword' => $id));
98     while($data = $query->fetch()) {
99         $aRetour[$i] = $data;
100         ++$i;
101     }
102     $query->closeCursor();
103
104     return $aRetour;
105 }
106
107 public function getTranslateSearchId($xsId, $xaLangResult, $xaLangSearch,
108     $xbValide, $xsTable = DBTABLENAMENOUVELLE,
109     $xbStrictID=0) {
110
111     $aRetour = array();
112     $i = 0;
113
114     $txtQuery = "SELECT ID_RES, TranslatedText, Language, Valide";
115     if($xsTable == DBTABLENAMENOUVELLE)

```

```

110     $txtQuery .= ", Type_RES, TranslatedText_short, TranslatedText_long ";
111
112     $txtQuery .= " FROM ".$xsTable."";
113     if($xbStrictID == 1) {
114         $txtQuery .= " WHERE lower(ID_RES) = :id ";
115     } else {
116         $txtQuery .= " WHERE lower(ID_RES) LIKE :id ";
117         $xsId = '%'.$xsId.'%';
118     }
119     // si $xbValide = 2 c'est qu'on demande les valides et les non valides
120     if($xbValide != 2) {
121         $txtQuery .= "AND Valide=:valide";
122     } else {
123         $txtQuery .= ":valide";
124         $xbValide = "";
125     }
126
127     for($i=0 ; $i < count($xaLangResult) ; ++$i) {
128         if($i == 0)
129             $txtQuery .= " AND (";
130         else
131             $txtQuery .= " OR ";
132
133         $txtQuery .= "Language='".$xaLangResult[$i]'" ";
134
135         if($i == count($xaLangResult) - 1)
136             $txtQuery .= ")";
137     }
138     $txtQuery .= 'ORDER BY ID_RES LIMIT 50';
139     $query = $this->db->prepare($txtQuery);
140     $query->execute(array('id' => $xsId,
141                          'valide' => $xbValide));
142
143     while($data = $query->fetch()) {
144         $aRetour[$i] = $data;
145         ++$i;
146     }
147     $query->closeCursor();
148
149     return $aRetour;
150 }
151
152 public function getLanguagesId($xsId, $xsTable = DBTABLENAMENOUVELLE) {
153     $aRetour = array();
154     $i = 0;
155
156     $query = $this->db->prepare("SELECT ID_RES, TranslatedText, Language
157                                FROM ".$xsTable."
158                                WHERE ID_RES = :id LIMIT 50");
159     $query->execute(array('id' => $xsId));
160
161     while($data = $query->fetch()) {
162         $aRetour[$i] = $data;
163         ++$i;
164     }
165     $query->closeCursor();
166
167     return $aRetour;
168 }

```



```

168 public function getRemainingTranslation($xsId, $xsTable =
    DBTABLENAMENOUVELLE) {
169     $aLanguageBool = array('fr' => false, 'en' => false, 'es' => false);
170     $aRetour = array();
171     $i = 0;
172
173     $query = $this->db->prepare("SELECT Language FROM $xsTable
174     WHERE ID_RES = :id");
175     $query->execute(array('id' => $xsId));
176
177     while($data = $query->fetch()) {
178         $aLanguageBool[$data['Language']] = true;
179     }
180     if(!$aLanguageBool['fr']) {
181         $aRetour[$i] = 'fr';
182         ++$i;
183     }
184     if(!$aLanguageBool['en']) {
185         $aRetour[$i] = 'en';
186         ++$i;
187     }
188     if(!$aLanguageBool['es']) {
189         $aRetour[$i] = 'es';
190         ++$i;
191     }
192
193     $query->closeCursor();
194
195     return $aRetour;
196 }
197
198 }

```

Listing F.3. databaseSearch.class.php

```

1 <?php
2 require_once('class/util.class.php');
3 Util::CreateDefine();
4
5 require_once('class/database.class.php');
6 require_once('class/databaseSchemaTable.class.php');
7 require_once('class/databaseChanges.class.php');
8 require_once('class/databaseSearch.class.php');
9
10 require_once('class/page.class.php');
11 require_once('class/pageselect.class.php');
12 require_once('class/pageinsert.class.php');
13 require_once('class/pageupdate.class.php');
14 require_once('class/pagedelete.class.php');
15
16 $titrePage = Util::testGet('p');
17 switch($titrePage) {
18     case 'select':
19         $page = new PageSelect();
20         break;
21     case 'insert':
22         $page = new Pageinsert();
23         break;
24     case 'delete':
25         $page = new PageDelete();

```

```

26     break;
27     case 'update':
28         $page = new PageUpdate();
29         break;
30     default:
31         exit();
32 }
33
34 $page->display();

```

Listing F.4. index.php

```

1 <?php
2
3 abstract class Page {
4     protected $sText;
5     protected $db;
6     private $dbSchemaTable;
7
8     public function __construct() {
9     }
10
11     public function display() {
12         echo $this->sText;
13     }
14
15     abstract protected function initText() ;
16
17 }

```

Listing F.5. page.class.php

```

1 <?php
2
3 class PageSelect extends Page {
4     private $sKeyword;
5     private $sTypeSearch;
6     private $aLangResult;
7     private $aLangSearch;
8     private $bValide;
9     private $sTable;
10    private $bSearchStrictId;
11    private $dbSchemaTable;
12
13    public function __construct() {
14        parent::__construct();
15        $this->dbSchemaTable = new DatabaseSchemaTable();
16        $this->db = new DatabaseSearch();
17
18        $this->sKeyword = addslashes(Util::testGet('k'));
19        $this->sTypeSearch = addslashes(Util::testGet('type'));
20        $this->aLangResult = explode('|', addslashes(Util::testGet('langResult')));
21        $this->aLangSearch = addslashes(Util::testGet('langSearch'));
22        $this->bValide = addslashes(Util::testGet('valide'));
23        $this->sTable = addslashes(Util::testGet('table'));
24        $this->bSearchStrictId = addslashes(Util::testGet('searchStrictId'));
25        if($this->bSearchStrictId == '') {
26            $this->bSearchStrictId = 0;
27        }
28    }
29 }

```

```

28     if($this->sTable == '' && !($this->sTypeSearch == "structTableMaxLength"))
29     {
30         $this->sTable = DBTABLENAMENOUVELLE;
31     }
32     if($this->sKeyword == '' && !($this->sTypeSearch == "structTableMaxLength")
33     )
34     {
35         exit();
36     }
37
38     $this->initText();
39
40     protected function initText() {
41         switch($this->sTypeSearch) {
42             case 'id':
43                 $this->setTextSearchId();
44                 break;
45             case 'text':
46                 $this->setTextSearchText();
47                 break;
48             case 'remainingTranslation':
49                 $this->setTextRemainingTranslation();
50                 break;
51             case 'structTableMaxLength' :
52                 $this->setTextMaxLength();
53                 break;
54         }
55     }
56
57     private function setTextSearchText() {
58         $aKeywords = explode('-', $this->sKeyword);
59         $aTranslate = $this->db->getTranslateSearchText($aKeywords, $this->
60             aLangResult, $this->aLangSearch,
61             $this->bValide, $this->sTable);
62         $this->textDisplaySearch($aTranslate);
63     }
64
65     private function setTextSearchId() {
66         $aTranslate = $this->db->getTranslateSearchId($this->sKeyword, $this->
67             aLangResult,
68             $this->aLangSearch, $this->bValide, $this->sTable, $this->
69             bSearchStrictId);
70         $this->textDisplaySearch($aTranslate);
71     }
72
73     private function textDisplaySearch($aTranslate) {
74         foreach($aTranslate as $aTranslate) {
75             $this->sText .= $aTranslate['ID_RES'];
76             $aLanguageTranslate = $this->db->getTraduceEffectue($aTranslate['ID_RES'],
77                 $this->sTable);
78             $this->sText .= SEPARATEURCOLONNE;
79             $this->sText .= $aLanguageTranslate['Language'];
80
81             $this->sText .= SEPARATEURLANGUE;
82             foreach($aLanguageTranslate as $language) {
83                 $this->sText .= $language['Language'];
84                 $this->sText .= " ";
85             }
86             $this->sText .= SEPARATEURLANGUE;
87             $this->sText .= SEPARATEURCOLONNE;

```

```

82     $this->sText .= $aTranslate['Valide'];
83     $this->sText .= SEPARATEURCOLONNE;
84
85     /* Si c'est la nouvelle table, c'est que le champ Type_RES existe */
86     if($this->sTable == DBTABLENAMENOUVELLE) {
87         $this->sText .= $aTranslate['Type_RES'];
88     }
89     $this->sText .= SEPARATEURCOLONNE;
90
91     $this->sText .= utf8_encode(Util::parserConstantes($aTranslate['
92         TranslatedText'],
93             $aTranslate['Language'], $this->db));
94
95     if($this->sTable == DBTABLENAMENOUVELLE) {
96         $this->sText .= SEPARATEURCOLONNE;
97         $this->sText .= $aTranslate['TranslatedText_short'];
98         $this->sText .= SEPARATEURCOLONNE;
99         $this->sText .= $aTranslate['TranslatedText_long'];
100     }
101     $this->sText .= SEPARATEURLIGNE;
102 }
103 if($xaTranslate == array()) {
104     $this->sText = "NO_FOUND";
105 }
106
107 private function setTextRemainingTranslation() {
108     $aTranslate = $this->db->getRemainingTranslation($this->sKeyword, $this->
109         sTable);
110     foreach($aTranslate as $aTranslate) {
111         $this->sText .= $aTranslate;
112         $this->sText .= SEPARATEURLIGNE;
113     }
114 }
115
116 private function setTextMaxLength() {
117     $aMaxLength = $this->dbSchemaTable->getMaxLength();
118
119     foreach($aMaxLength as $maxlength) {
120         if($maxlength['COLUMN_NAME'] == 'TranslatedText_short' ||
121             $maxlength['COLUMN_NAME'] == 'TranslatedText' ||
122             $maxlength['COLUMN_NAME'] == 'TranslatedText_long') {
123             $this->sText .= $maxlength['CHARACTER_MAXIMUM_LENGTH'];
124             $this->sText .= ' ';
125         }
126     }
127 }

```

Listing F.6. pageselect.class.php

F.2.2 Le client

F.2.2.1 lib

```

1 var previousRequest;
2 var previousValue;

```

```

3 var tableau = document.getElementById("tableResults"+onglet);
4 var iterateurLanguages = 0, nbLangAffichage = 0;
5
6 var dataTraduceLang, dataCurrentLang, dataValide, dataType, dataValue;
7
8 var elementInfoTable = document.getElementById('infoTable'+onglet);
9 var elementinfos = document.getElementById('info'+onglet);
10
11 /**
12  * Effectue une requete et recupere les resultats
13  */
14 function getResult(keyword, xSearchInOldTable) {
15     clicTabForbidden();
16     elementInfos = document.getElementById('info'+onglet);
17     tableau = document.getElementById("tableResults"+onglet);
18     xTable = tableDb;
19     elementInfoTable = document.getElementById('infoTable'+onglet);
20     elementInfoTable.innerHTML = '';
21     tableau.innerHTML = '';
22     chargerConf();
23     nbResults = 0;
24     var xhr = new XMLHttpRequest();
25     keyword = keyword.replace(new RegExp(' ', 'g'), '-');
26     xhr.open('GET', adresseServeur+'?p=select&type='+typeSearch+'&k='+
27         convertAccentsForGet(keyword)+
28         '&whole='+wholeWord+'&regex='+regex+'&langSearch='+
29         langSearch+'&langResult='+langResults+'&valide='+valide+'&table='+
30         xTable);
31
32     if(xTable == 'RES_dicoLanguage') {
33         elementInfoTable.innerHTML = '<p> '+
34             'Recherche en cours dans la nouvelle table</p>';
35     } else {
36         elementInfoTable.innerHTML = '<p> '+
37             'Recherche en cours dans l\'ancienne table</p>';
38     }
39
40     if(keyword == '') {
41         elementInfoTable.innerHTML = '';
42         elementInfos.innerHTML = '';
43         clicTabAllow();
44         return ;
45     }
46
47     xhr.onreadystatechange = function() {
48         if (xhr.readyState == 4 && xhr.status == 200) {
49             displayResultsSearch(xhr.responseText, keyword, xTable, xSearchInOldTable
50                 );
51         }
52     };
53
54     xhr.send(null);
55
56     return xhr;
57 }
58 /**
59  * Informe l'utilisateur que des constantes ont ete trouvees
60  */
61 function displayInfoCstNotFound(xkeyword) {
62     var textInfo = '';

```

```

60 textInfo = getInfoSearch(xkeyword);
61
62 if(tableDb == 'RES_dicoLanguage') {
63     tableau.innerHTML = '<div id="ajoutcstNonTrouve">Aucune constante n\'a &
        eacute;t&eacute; trouv&eacute;e ni '+
64     'dans la nouvelle table, ni dans l\'ancienne table<br />';
65 } else {
66     textInfo += '<div id="ajoutcstNonTrouve">Aucune constante n\'a &eacute;t&
        eacute; trouv&eacute;e dans l\'ancienne table<br />';
67 }
68
69 textInfo += '<center><a href="#" id="addCstAfterSearch">'+
70     'Ajouter une constante dans la nouvelle table</a></center></div>';
71
72 elementInfoTable.innerHTML = textInfo;
73
74 }
75
76 /**
77  * Informe l'utilisateur qu'aucune constante n'a ete trouvee dans les deux
       tables
78  * @param xTable
79  */
80 function displayInfoCstFound(xTable, xkeyword) {
81     var textInfo = '';
82     if(xTable != tableDb) {
83         textInfo = 'Aucune constante n\'a &eacute;t&eacute; trouv&eacute;e dans la
            nouvelle table. <br/>';
84     }
85
86     textInfo = getInfoSearch(xkeyword);
87     if(nbResults > 1) {
88         textInfo += '<strong>'+(nbResults)+'</strong> constantes trouv&eacute;es
            dans '+
89         '<strong>'+nomTableToNewOrOld(xTable)+'</strong> table';
90     } else {
91         textInfo += '<strong>'+(nbResults)+'</strong> constante trouv&eacute;e
            dans '+
92         '<strong>'+nomTableToNewOrOld(xTable)+'</strong> table';
93     }
94
95     elementInfos.innerHTML = textInfo;
96 }
97
98 function getInfoSearch(xkeyword) {
99     var textInfo = '';
100     xkeyword = xkeyword.replace(new RegExp('-', 'g'), ' ');
101     textInfo += '<strong>Recherche</strong>: <span id="keywordTab'+onglet+'">'+
        xkeyword+'</span><br />';
102     textInfo += '<strong>Type de recherche</strong>: '+charTypeToString(
        typeSearch)+'<br />';
103     textInfo += '<strong>Mots entiers</strong>: '+boolWholeToString(wholeWord)+'
        <br /><hr/>';
104     document.getElementById('searchTab'+onglet).innerHTML = '<em>'+cutString(
        xkeyword.toLowerCase(),0,10)+'</em>';
105     return textInfo
106 }
107 /**
108  * Informe l'utilisateur qu'aucune constante n'a ete trouvee pour la nouvelle
       table

```

```

109  */
110  function displayInfoStartSearchOldTable(xkeyword) {
111      var textInfo = '';
112      textInfo = getInfoSearch(xkeyword);
113      textInfo += '<div id="cstNonTrouveNouvelle"><p>Aucune constante trouvée
114                  ;e dans la nouvelle table</p></div>';
115      elementInfoTable.innerHTML = textInfo;
116  }
117  /**
118   * Affiche les resultats d'une recherche
119   */
120  function displayResultsSearch(response, keyword, xTable, xsearchInOldTab) {
121      var titreCol, ligne;
122      var dataSearch = new Object();
123
124      tableau.innerHTML = "";
125      elementInfos.innerHTML = '';
126      if(keyword == '' ) {
127          elementInfos.innerHTML = '';
128      }
129      if (response.length) { // On ne modifie les resultats que si on en a obtenu
130          if(response == "NO_FOUND") { // aucune constante trouvee
131              if(xTable == 'RES_dicoLanguage'){
132                  displayInfoStartSearchOldTable(keyword);
133                  if(xsearchInOldTab) {
134                      clicTabAllow();
135                      addTab('Ancienne');
136                      changeTab('Ancienne', nbTabOld);
137                      getResults(keyword, false); //on lance la recherche dans l'ancienne
138                      table
139                  }
140              } else {
141                  displayInfoCstNotFound(keyword);
142                  document.getElementById('addCstAfterSearch').onclick = function(){
143                      menuActionAddCst(null, keyword)};
144              }
145          } else {
146              elementInfoTable.innerHTML = '';
147
148              //zou, on remplit le tableau de toutes les constantes trouvees!
149              response = response.split('+++'); //chaque ligne
150              var responseLen = response.length;
151              var elementBouton;
152              ligne = tableau.insertRow(-1);
153
154              titreCol = ligne.insertCell(-1);
155              titreCol.innerHTML += "ID";
156              titreCol.className = "titreColonne";
157
158              titreCol = ligne.insertCell(-1);
159              titreCol.innerHTML += "";
160              titreCol.className = "titreColonne";
161
162              titreCol = ligne.insertCell(-1);
163              titreCol.innerHTML += "Valeur";
164              titreCol.className = "titreColonne";
165
166              titreCol = ligne.insertCell(-1);
167              titreCol.innerHTML += "";

```

```

166     titreCol.className = "titreColonne";
167
168     for (var i = 0 ; i < responseLen - 1 ; ++i) {
169         dataSearch = getDataSearch(i, response); //on recupere toutes les
170             donnees
171
172         nbLangAffichage = getNbLangToDisplay(dataSearch['TraduceLang']);
173         ++iterateurLanguages;
174
175         ligne = tableau.insertRow(-1); //on ajoute une ligne
176
177         /* on remplit le tableau avec toutes les cellules */
178         addCellsConstanteAndTraduce(i, ligne, responseLen, dataSearch, xTable)
179         ;
180         addCellsValide(i, ligne, dataSearch, xTable);
181         addCellsTraduceValue(ligne, dataSearch, i, keyword);
182         addCellsBtn(i, ligne, dataSearch, xTable);
183
184         if(iterateurLanguages >= nbLangAffichage) {
185             iterateurLanguages = 0;
186         }
187     }
188     displayInfoCstFound(xTable, keyword);
189 }
190 clicTabAllow();
191 }
192
193 /**
194  * Retourne le nombre de langues qui devront etre affiche
195  * @param Les langues dans lesquels la constante est traduite
196  * @returns Le nombre de langue
197  */
198 function getNbLangToDisplay(xsLangTraduce, xiNbLangDemande) {
199     if(!xiNbLangDemande){
200         xiNbLangDemande = nbLangDemande;
201     }
202
203     return ((xsLangTraduce.length > xiNbLangDemande) ? xiNbLangDemande :
204         xsLangTraduce.length);
205 }
206
207 /**
208  * Retourne toutes les donnees d'une recherche dans un tableau associatif
209  * @param xiNumberLine Le numero de la ligne
210  * @param xresponse La requete
211  * @returns {Object} Les donnees
212  */
213 function getDataSearch(xiNumberLine, xresponse) {
214     var data = new Object();
215     var values = new Object();
216
217     var allDdatas = xresponse[xiNumberLine].split('---'); //chaque colonnes
218     data['Id'] = trim(allDdatas[0]);
219     dataLang = allDdatas[1];
220
221     /* on s'occupe des langues: Langues d'affichage et langue traduite existante
222          dans la base */
223     data['TraduceLang'] = allDdatas[1].split('###');

```



```

222 data['CurrentLang'] = trim((data['TraduceLang'])[0]);
223 data['TraduceLang'].shift();
224 data['TraduceLang'] = data['TraduceLang'][0].split(' ');
225 data['TraduceLang'].pop();
226
227 /* Les autres champs */
228 data['Valide'] = trim(allDatas[2]);
229 data['Type'] = trim(allDatas[3]);
230
231 values['short'] = (allDatas[5]);
232 values['normal'] = (allDatas[4]);
233 values['long'] = (allDatas[6]);
234 data['value'] = values;
235
236 return data;
237 }
238
239 /**
240  * Colorie les mots trouves dans le texte
241  * @param xsKeyword Mot a chercher
242  * @param xsSentence Phrase dans laquelle on doit chercher
243  * @returns La chane coloree
244  */
245 function colorWordFound(xsKeyword, xsSentence) {
246     var sReturn = xsSentence;
247     var aKeyword = xsKeyword.split('-');
248     var regexHighlighting;
249
250     for(var i = 0 ; i < aKeyword.length ; ++i) {
251         if(aKeyword[i].length > 2 && typeSearch == 'text') {
252             regexHighlighting = new RegExp('(' + aKeyword[i] + ')', "gi");
253             regexHighlighting.exec(sReturn);
254             sReturn = sReturn.replace(RegExp.$1, "<span class=\"keyword\">" + RegExp.
                $1 + "</span>");
255         }
256     }
257
258     return sReturn;
259 }

```

Listing F.7. search.js

```

1 function actionAnnulerTraduction(numberLine, xinbreLineTotal) {
2     tableau = document.getElementById("tableResults"+onglet);
3     tableau.deleteRow(numberLine+2);
4     enCoursDeTraduction = false;
5
6     affichageLiensTraduction(xinbreLineTotal, true);
7 }
8
9 function actionValiderTraduction(numberLine, xsConstante, xinbreLineTotal) {
10     tableau = document.getElementById("tableResults"+onglet);
11     insertionNouvelleTraduction(xsConstante,
12     document.getElementById('language').options[document.getElementById('
        language').selectedIndex].value,
13     document.getElementById('valueTraduce').value, 'PHRA',
14     (document.getElementById('valideTraduce').checked) ? 1 : 0,
15     document.getElementById('shortvalueTraduce').value, document.getElementById(
        'longvalueTraduce').value);
16 }

```

```

17  tableau.deleteRow(numberLine+2);
18  enCoursDeTraduction = false;
19
20  affichageLiensTraduction(xinbreLineTotal, true);
21
22  rafraichir();
23 }
24
25 function actionVerifSizeString(xsType, element) {
26   if(!valueNotToLong(xsType, element.value)) {
27     document.getElementById('champ'+xsType+'TooLong').style.display = 'inline-
      block';
28   } else {
29     document.getElementById('champ'+xsType+'TooLong').style.display = 'none';
30   }
31 }

```

Listing F.8. actionTranslate.js

```

1  <html>
2  <head>
3    <title>MemoLanguage</title>
4    <link href="styles/global.css" rel="stylesheet" type="text/css"/>
5    <link href="styles/homeSearch.css" rel="stylesheet" type="text/css"/>
6
7    <meta http-equiv="content-type" content="text/html; charset=utf-8" />
8
9    <script type="text/javascript" src="lib/AIRAliases.js"></script>
10   <script type="text/javascript" src="lib/systray.js"></script>
11   <script type="text/javascript" src="lib/menu.js"></script>
12   <script type="text/javascript" src="lib/variablesGlobales.js"></script>
13   <script type="text/javascript" src="lib/util.js"></script>
14   <script type="text/javascript" src="lib/actions/actionCopy.js"></script>
15   <script type="text/javascript" src="lib/actions/actionSearch.js"></script>
16   <script type="text/javascript" src="lib/actions/actionTabs.js"></script>
17   <script type="text/javascript" src="lib/actions/actionTranslate.js"></
      script>
18   <script type="text/javascript" src="lib/actions/actionUpdates.js"></script>
19
20
21   <script type="text/javascript" src="lib/addCells.js"></script>
22   <script type="text/javascript" src="lib/configuration.js"></script>
23   <script type="text/javascript" src="lib/maxLength.js"></script>
24 </head>
25 <body>
26 <!--
27   Recherche
28 -->
29 <div id="divSearch">
30   <div id="choixLangue">
31     Afficher dans les langues: <br />
32     <input type="checkbox" checked=checked onclick="
      actionChangeLanguagesResults()" id="chkSearch1"/>
33     <label for="chkSearch1">Fran ais </label><br />
34     <input type="checkbox" onclick="actionChangeLanguagesResults()" id="
      chkSearch2"/>
35     <label for="chkSearch2">Anglais </label><br />
36     <input type="checkbox" onclick="actionChangeLanguagesResults()" id="
      chkSearch3"/>
37     <label for="chkSearch3">Espagnol</label><br />

```

```

38 </div>
39
40 <p>Chercher par :
41 <input class="typeSearch" type="radio" name="type" value="id" id="id"
42     onclick="actionChangeTypeSearch('id',true);"/>
43     <label for="id">id</label>
44
45 <input class="typeSearch" type="radio" name="type" value="text" checked="
46     checked" id="text"
47     onclick="actionChangeTypeSearch('text',true);" />
48     <label for="text">Texte</label>
49 </p>
50 <div id="optionsValides">
51     <input type="checkbox" id="estValide" onclick="actionValideSearch( );"
52     checked="checked"/>
53     <label for="valide">Traduction validates</label>
54     <input type="checkbox" id="estNonValide" onclick="actionValideSearch( );"
55     />
56     <label for="nonValide">Traduction non validates</label>
57 </div>
58
59 <div id="optionsText">
60     <input type="checkbox" id="wholeWord" onclick="actionChangeWholeWord( );"
61     />
62     <label for="wholeWord">Mots entiers</label>
63     <input type="checkbox" id="regex" onclick="actionChangeRegex( );"/>
64     <label for="regex">Expression reguliere</label>
65 </div>
66 <input id="search" type="text" onkeyup="actionSearch(true, false)"
67     onKeyPress="if (event.keyCode == 13) actionSearch(true, true)"/><br />
68 <input id="btnSearch" type="button"
69     onclick="actionSearch(true, true)" value="Chercher !"/>
70 </div><p>
71     <span id="onglet">
72         </span>
73
74         <a href="#" class="info" onclick="actionDisplayTypeTab()"><span class="textinfo">
76             Ajouter un onglet</span></a>
77         <select style="display:none" name="typeTab" id="typeTab">
78             <option value="Ancienne">Ancienne table</option>
79             <option value="Nouvelle">Nouvelle table</option>
80         </select>
81         <input id="btnValideAddTab" type="button" style="display:none" onclick
82             ="actionAddTab()" value="Ajouter" />
83     </p>
84 <div id="displayOnglets">
85 </div>
86
87 <script type="text/javascript" src="lib/traduction.js"></script>
88 <script type="text/javascript" src="lib/search.js"></script>
89 <script type="text/javascript" src="lib/tabs.js"></script>
90 <div id="footer"><span id="version">MemoLanguage version <strong><span id="
91     numVersionML"></span></strong></span>
92 <span id="langueRecherche">Langue de recherche: <strong><span id="
93     langSearch"></span></strong></span>
94 </div>
95 <script type="text/javascript" src="lib/init.js"></script>
96 <script type="text/javascript">
97     addTab("Nouvelle");

```

```

87     addTab("Ancienne");
88
89     initTabs();
90     changeTab('Nouvelle', 1);
91     </script>
92 </body>
93 </html>

```

Listing F.9. Page HTML principale homeSearch.html

F.2.2.2 bin

```

1  <?xml version="1.0" encoding="UTF-8"?>
2  <application xmlns="http://ns.adobe.com/air/application/2.6">
3      <id>Outils.Translation</id>
4      <versionNumber>0.9</versionNumber>
5      <filename>MemoLanguage</filename>
6      <initialWindow>
7          <content>homeSearch.html</content>
8          <visible>true</visible>
9          <width>850</width>
10         <height>400</height>
11     <transparent>false</transparent>
12     <systemChrome>standard</systemChrome>
13     <x>400</x>
14     <y>500</y>
15 </initialWindow>
16 <icon>
17     <image16x16>img/logos/logo16.png</image16x16>
18     <image32x32>img/logos/logo32.png</image32x32>
19     <image48x48>img/logos/logo48.png</image48x48>
20     <image128x128>img/logos/logo128.png</image128x128>
21 </icon>
22 </application>

```

Listing F.10. Descripteur de fichier MemoLanguage-app.xml