



ECOLE NATIONALE SUPÉRIEURE D'INFORMATIQUE ET D'ANALYSE DES
SYSTÈMES - RABAT

Rapport de Projet de Compilation : Réalisation d'un compilateur pour le langage Cloudian

Réalisé par :

El Mehdi OUDAUD
Fatima Ezzahra LAHNINE

Encadré par :

Pr. Youness TABII

Année Scolaire 2021/2022

Table des matières

1	Analyse du Sujet	2
1.1	Analyse du besoin	2
1.1.1	Description des SOA	2
1.2	Exemple d'architecture SOA	3
1.2.1	Problèmes d'implémentation	4
1.3	Solution	5
2	Conception	6
2.1	Introduction	6
2.2	Grammaire	7
2.2.1	Productions	7
2.2.2	Terminaux	8
3	Implémentation	13
3.1	Guide d'installation	13
3.2	Technologies	13
3.3	Bout du code	14
4	Amélioration	16
4.1	Problèmes rencontrés	16

Introduction

Pour appliquer les méthodologies et les notions enseignées durant le cours de Compilation, nous sommes invités à réaliser un projet qui nous permet d'employer nos connaissances théoriques sur le champ pratique. Le projet consiste à créer, à l'aide de YACC et de LEX, un compilateur d'un langage de programmation qui répond à un certain besoin.

Notre projet se focalisera principalement sur l'amélioration du flux de travail des développeurs intéressés par la création et la gestion d'une application en architecture orientée service.

Chapitre 1

Analyse du Sujet

Ce chapitre permet de faire une analyse théorique de notre application. En effet, cette conception est indispensable afin de réaliser une application qui satisfait la totalité des contraintes exprimées.

1.1 Analyse du besoin

Parlant premièrement des exigences minimales du projet :

- Inclusion de :
 1. Types
 2. Opérateurs
 3. Structures conditionnelles
 4. Boucles
 5. Instructions de saisi/affichage de base
- Une grammaire LL (1).
- Un Analyseur lexical.
- Un Analyseur syntaxique.

Parlant maintenant du besoin qu'on essayera de traiter à travers notre langage de programmation

1.1.1 Description des SOA

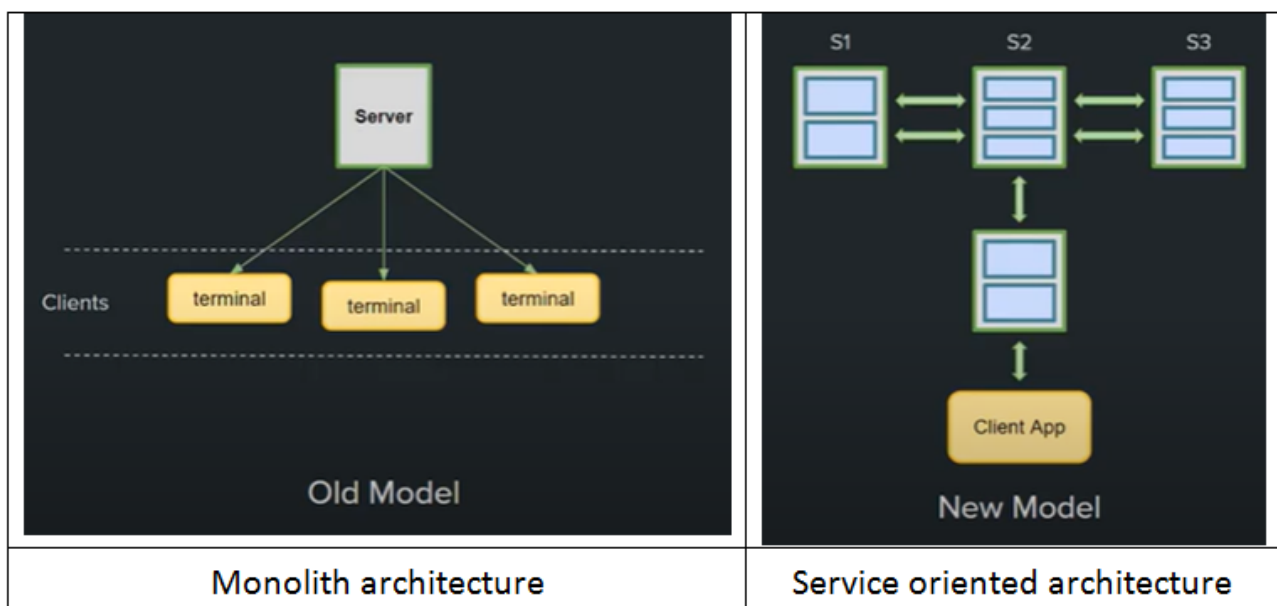


FIGURE 1.1

La définition dépend certainement du contexte. Mais, en général, la définition du concept est la suivante : Un SOA est un paradigme ou une approche pour concevoir le développement, le déploiement et la maintenance de systèmes logiciels basés sur les systèmes distribués (services, données, matériel).

Notre projet se focalisera sur les applications avec **des composantes logicielles autonomes couplé entre eux**.

1.2 Exemple d'architecture SOA

Voici une présentation d'une architecture SOA :

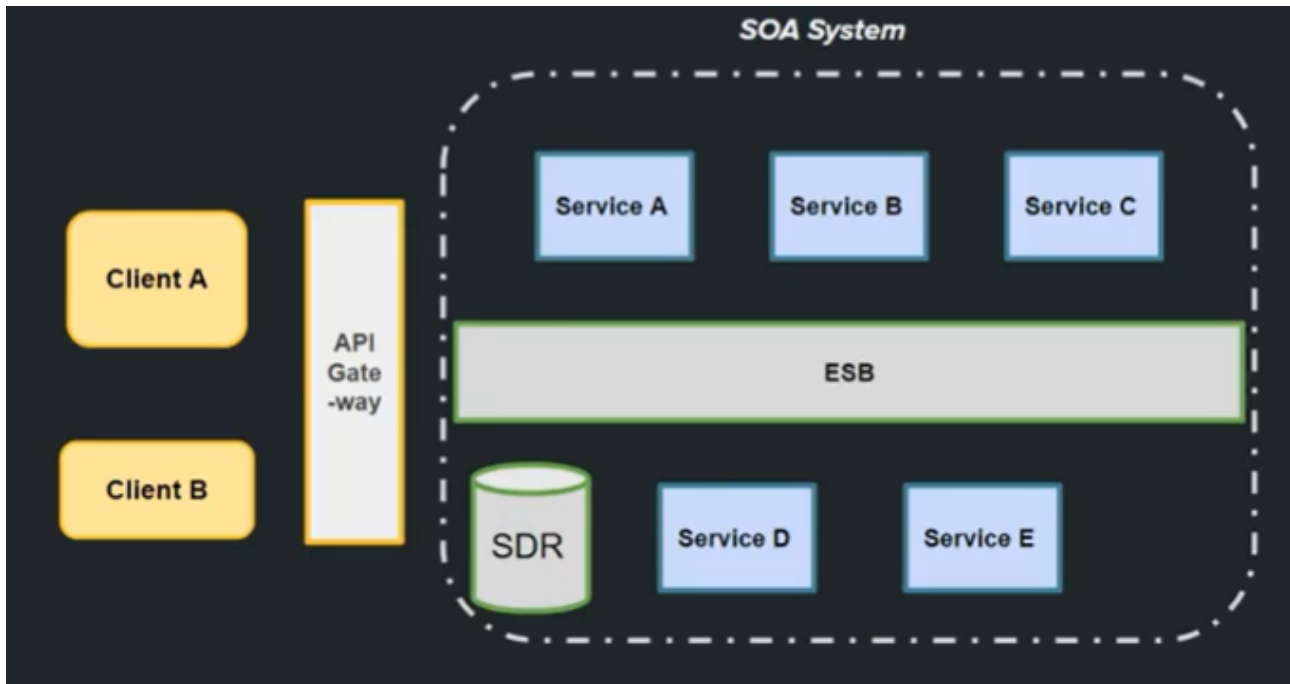


FIGURE 1.2

On note que l'architecture en monolithe est similaire à cette présentation :

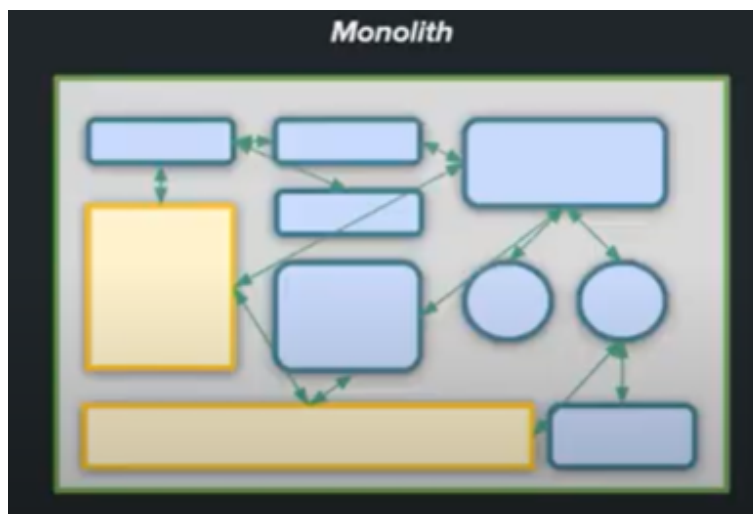
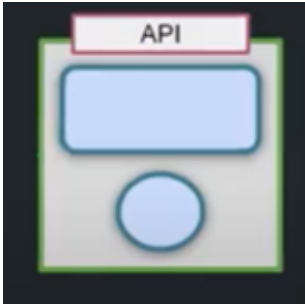

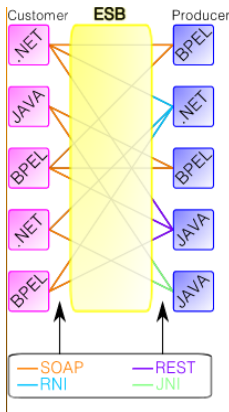
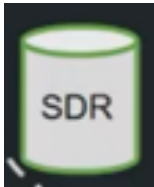
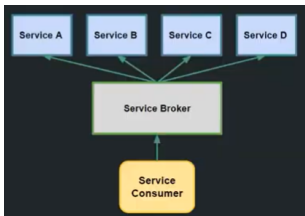


FIGURE 1.3

Par comparaison on peut déduire que le couplage entre les services est plus libre et organisé comparé à un monolithe :

<p>Service : Un service est un ensemble de micro-services (services atomique) interconnecter entre eux, et exposée à travers un API</p>	 <p>Service Composé</p>	 <p>Service Atomique</p>
<p>ESB : Enterprise service Bus : C'est une technique informatique qui a pour but d'organiser et traduire la communication entre les différents services qui possèdent des protocoles de communication distinct.</p>		<p>Un ESB est responsable de :</p> <ul style="list-style-type: none"> — Connectivité entre services — Transformation des données et conversion de messages (MOM). — Routage — Sécurité de communication interne — Monitoring et sauvegarde des logs (BAM,BPM)
<p>SDR : Service Description Registry : C'est ainsi que les consommateurs de services découvrent les services et la manière appropriée de communiquer avec eux ainsi que leur disponibilité et les automatisations utilisées par l'ESB pour communiquer efficacement avec les services.</p>	 <p>contient metadata</p>	 <p>Un Agent de message (service broker) est la méthode utilisée pour gérer les expositions des services et leurs API aux consommateurs (interne ou externe) pour des petites applications. Cependant, pour les systèmes grands et complexes, on opte pour un SDR.</p>

1.2.1 Problèmes d'implémentation

L'architecture orientée service possède plusieurs limitations comme :

- La complexité de la gestion des services.
- La sécurité, SOA est construit sur des standards comme XML, WSDL, SOAP. Et ses standards ne possèdent aucune sécurité innée. Donc sont vulnérables contre les attaques :

Home au milieu (man-in-the-middle)

DOS (Denial o service)

Aucune trace vu quil na pas de logs extensifs

Écoute clandestine sur le réseau interne

- La performance non prédictible (vu quil a plusieurs points de faille).
- Enfermement propriétaire
- Sémantique : un problème se pose lors du développement en particulier lorsque les services utilisent des structures de données différentes et sinvoque de façon intriqué.

1.3 Solution

Notre solution est duniformiser la façon avec laquelle les applications darchitecture orienté service son écrite à travers un langage de programmation qui se compile en un ensemble de :

- Scripts pour conteneur ou configuration (dockerfiles , nginx.cfg).
- Un orchestrateur de centenaire (exemples : docker, kubernetes)
- Un agent de message (exemple : Rabbit MQ, Kafka)
- Un ensemble dAPI généré à partir des choix dutilisateurs indépendamment du langage source du service.
- Un système de sécurité flexible pour crypter la communication.
- Une sémantique uniforme, maintenable et facilement si on veut changer la SOAIF (SOA implementation Framework).
- Un système performant pour sauvegarder les logs, monitorer le trafic, voir létat des services et leurs descriptions ainsi que monitorer les performances des services.
- Support de la Livraison continu

Chapitre 2

Conception

2.1 Introduction

La figure ci-dessus montre une vue générale sur un gestionnaire d'architecture qui sera décrit par notre langage.

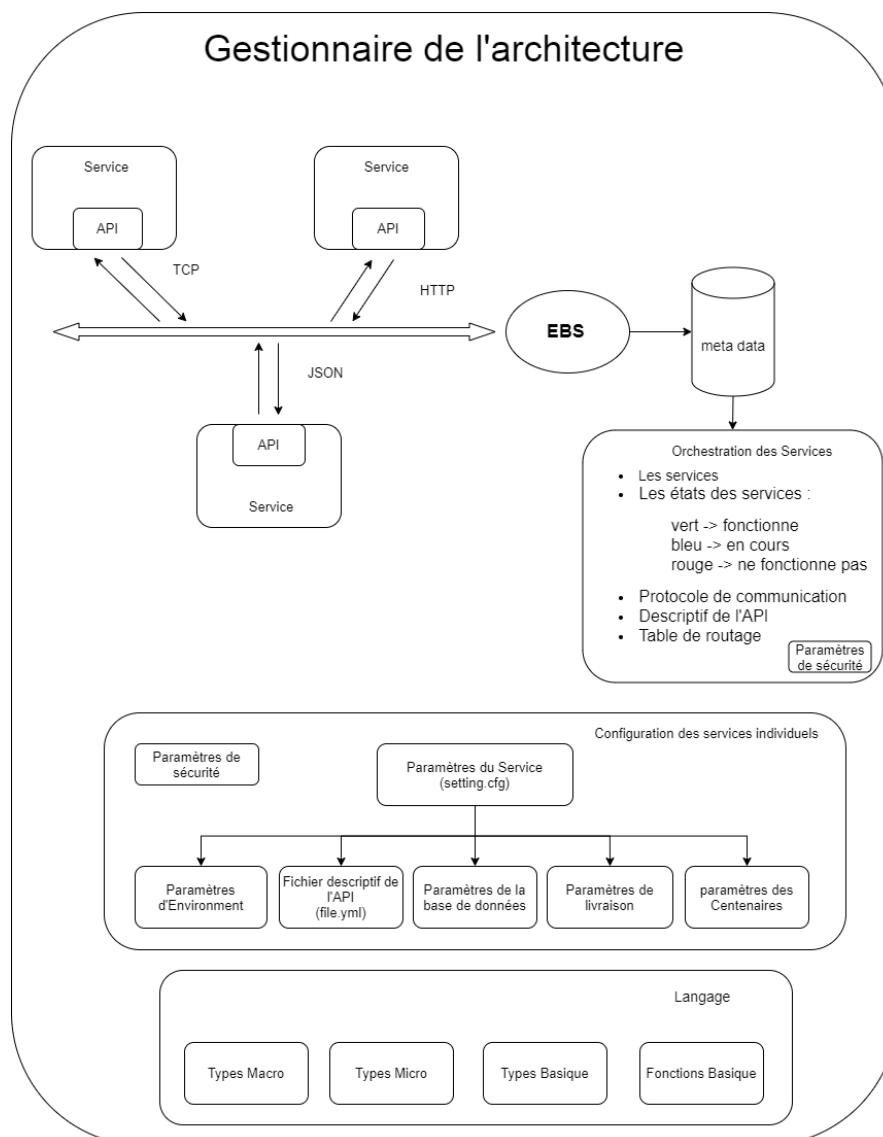


FIGURE 2.1

2.2 Grammaire

On note que cette grammaire nest quexpérimentale et quelle ne présente quune idée sur la logique avec laquelle on veut construire notre projet. Ce langage nest pas indépendant : il se compile en un ensemble de scripts, centenaire avec leurs propres technologies précisées par lutilisateur et il peut par après personnalisé les fichiers compilés (ajouter des timeouts au fichier de configuration dun serveur web, changer les paramètres par défaut). En fin de compte notre but est de faciliter la tâche au développeur et automatiser la majorité de leur travail ; Il suffit quil envisage une architecture, fournissent le code qui tourne sur un service, une description des fonctions de communication (API) sur chaque service et un ensemble de fichier configuration écrit sur notre langage.

2.2.1 Productions

```

P = {
|postfix_expression PTR_OP IDENTIFIER
primary_expression ::= IDENTIFIER | HEXA | NUMBER | STRING_LITERAL | '(' expression ')'
postfix_expression ::= primary_expression | postfix_expression '[' expression ']' | postfix_expression
'(' ')' | postfix_expression '(' argument_expression_list ')' | postfix_expression '?' IDENTIFIER | post-
fix_expression INC_OP | postfix_expression DEC_OP
argument_expression_list ::= assignment_expression | type_name | argument_expression_list ',' as-
signment_expression
unary_expression ::= postfix_expression | INC_OP unary_expression | DEC_OP unary_expression
| unary_operator cast_expression
unary_operator ::= '&' | '*' | '+' | '-' | '~' | '!'
cast_expression ::= unary_expression | '(' argument_expression_list ')' cast_expression
multiplicative_expression ::= cast_expression | multiplicative_expression '*' cast_expression | multi-
plicative_expression '/' cast_expression | multiplicative_expression '%' cast_expression
additive_expression ::= multiplicative_expression | additive_expression '+' multiplicative_expression
| additive_expression '-' multiplicative_expression
shift_expression ::= additive_expression | shift_expression LEFT_OP additive_expression | shift_expression
RIGHT_OP additive_expression
relational_expression ::= shift_expression | relational_expression '<' shift_expression | relational_expression
'>' shift_expression | relational_expression LE_OP shift_expression | relational_expression GE_OP
shift_expression
equality_expression ::= relational_expression | equality_expression EQ_OP relational_expression |
equality_expression NE_OP relational_expression
and_expression ::= equality_expression | and_expression '&' equality_expression
exclusive_or_expression ::= and_expression | exclusive_or_expression '^' and_expression
inclusive_or_expression ::= exclusive_or_expression | inclusive_or_expression '|' exclusive_or_expression
logical_and_expression ::= inclusive_or_expression | logical_and_expression AND_OP inclusive_or_expression
logical_or_expression ::= logical_and_expression | logical_or_expression OR_OP logical_and_expression
conditional_expression ::= logical_or_expression | logical_or_expression '?' expression ':' condition-
al_expression
assignment_expression ::= conditional_expression | unary_expression assignment_operator assign-
ment_expression
assignment_operator ::= '=' | MUL_ASSIGN | DIV_ASSIGN | MOD_ASSIGN | ADD_ASSIGN |
SUB_ASSIGN | LEFT_ASSIGN | RIGHT_ASSIGN | AND_ASSIGN | XOR_ASSIGN | OR_ASSIGN
expression ::= assignment_expression | expression ',' assignment_expression
constant_expression ::= conditional_expression
declaration ::= declaration_specifiers ';' | declaration_specifiers init_declarator_list ';'
declaration_specifiers ::= type_specifier | type_specifier declaration_specifiers | type_qualifier | type_qualifier
declaration_specifiers
ini_declarator_list ::= init_declarator | init_declarator_list ',' init_declarator
init_declarator ::= declarator | declarator '=' initializer
type_specifier ::= PROXY | PROTOCOL | ATOM | SERVICE | SERVICE_BUS | DELIVERY | XML
| HTML | JSON | HTTP | HTTPS | BOB | BLOB | ENCB | FRAME | STRING | DOUBLE | FLOAT
| INT | BOOLEAN | LIST | MAP | Tag_specifier | enum_specifier | TYPE_NAME
Tag_specifier ::= Tag IDENTIFIER '{' compound_statement '}' | Tag '{' compound_statement '}' |
Tag IDENTIFIER utility init_declarator_list | Tag utility init_declarator_list
utility ::= TOOL | DAEMON | LANGUAGE | GATEWAY
Tag ::= CHAIN | PLUGIN

```

```

specifier_qualifier_list ::= type_specifier specifier_qualifier_list | type_specifier | type_qualifier specifier_qualifier_list | type_qualifier
map_specifier ::= MAP '{' enumerator_list '}' | MAP IDENTIFIER '{' enumerator_list '}' | MAP IDENTIFIER
enumerator_list ::= enumerator | enumerator_list ',' enumerator
enumerator ::= IDENTIFIER ':' IDENTIFIER | IDENTIFIER ':' constant_expression | constant_expression ':' constant_expression
type_qualifier ::= CONST | ECONST | XCONST | VAR | EVAR | XVAR
declarator ::= direct_declarator
direct_declarator ::= IDENTIFIER | '(' declarator ')' | direct_declarator '[' constant_expression ']' | direct_declarator '[' ']' | direct_declarator '(' parameter_type_list ')' | direct_declarator '(' identifier_list ')' | direct_declarator '(' ')'
type_qualifier_list ::= type_qualifier | type_qualifier_list type_qualifier
parameter_type_list ::= parameter_list | parameter_list ',' ELLIPSIS
parameter_list ::= parameter_declaration | parameter_list ',' parameter_declaration
parameter_declaration ::= declaration_specifiers declarator | declaration_specifiers abstract_declarator | declaration_specifiers
identifier_list ::= IDENTIFIER | identifier_list ',' IDENTIFIER
type_name ::= specifier_qualifier_list | specifier_qualifier_list abstract_declarator
abstract_declarator ::= direct_abstract_declarator
direct_abstract_declarator ::= '(' abstract_declarator ')' | '[' ']' | '[' constant_expression ']' | direct_abstract_declarator '[' ']' | direct_abstract_declarator '[' constant_expression ']' | '(' ')' | '(' parameter_type_list ')' | direct_abstract_declarator '(' ')' | direct_abstract_declarator '(' parameter_type_list ')' | special_function
special_function ::= GET | POST | PUT | DEL | CGET | CPOST | CPUT | CDEL | LOG | READ | WRITE
initializer ::= assignment_expression | '{' initializer_list '}' | '{' initializer_list ',' '}'
initializer_list ::= initializer | initializer_list ',' initializer
statement ::= labeled_statement | compound_statement | expression_statement | selection_statement | iteration_statement | jump_statement
labeled_statement ::= IDENTIFIER ':' statement | CASE constant_expression ':' statement | DEFAULT ':' statement
compound_statement ::= '{' '}' | '{' statement_list '}' | '{' declaration_list '}' | '{' declaration_list statement_list '}'
declaration_list ::= declaration | declaration_list declaration
statement_list ::= statement | statement_list statement
expression_statement ::= ';' | expression ';'
selection_statement ::= IF '(' expression ')' statement | IF '(' expression ')' statement ELSE statement | SWITCH '(' expression ')' statement
iteration_statement ::= WHILE '(' expression ')' statement | DO statement WHILE '(' expression ')' ';' | FOR '(' expression_statement expression_statement ')' statement | FOR '(' expression_statement expression_statement expression_statement ')' statement
jump_statement ::= CONTINUE ';' | BREAK ';' | RETURN ';' | RETURN expression ';'
translation_unit ::= external_declaration | translation_unit external_declaration
external_declaration ::= function_definition | declaration
function_definition ::= declaration_specifiers declarator declaration_list compound_statement | declaration_specifiers declarator compound_statement | FUNCTION declarator declaration_list compound_statement | FUNCTION declarator compound_statement | declarator declaration_list compound_statement
chain_expression ::= postfix_expression | chain_expression PTR_OP postfix_expression | chain_expression PTR_OP compound_statement
}

```

2.2.2 Terminaux

Macro types :	
Atom	<include service> {...}
Service	api {call(), details(), domain, host, port, state, ...}
Proxy	balance([services] , layer, ...), load-from-cfg(FILE)
Protocol	TLS/SSL, TCP, HTTP , HTTPS ...
GateWay	passerelle
ServiceBus	Suivi des affaires électroniques (EBS)
DILIVERY	livraison

TABLE 2.1

Micro types :	
CHAIN	chaîne
GATEWAY	passerelle
PLUGIN	plugin
DAEMON	daemon
TRACK	suivi
ENABLE	activer
DISABLE	désactiver

TABLE 2.2

Protocoles types :	
XML, HTML, JSON, HTTP, HTTPS	Layer 7
BOB	Binary object
BLOB	Binary Large object
ENCB	Encrypted binary
FRAME	Layer 4

TABLE 2.3

Basic types :	
int	nombres entiers
float	nombre réels
double	nombre réels
long	nombre réels
String	chaîne de caractères
list	collection ordonnée d'éléments
map	collection triée de paires d'éléments, une clé et une valeur
bool	déclarations vrai/faux
TIME	new Time("nombre de seconde")

TABLE 2.4

Balises particulières :	
communicationProtocol	Protocole de communication
networkConf	Configuration du réseau
apiDescriptor	Descripteur d'API
containerConf	Configuration du conteneur
run	Exécution
serviceSecurity	Sécurité des services
communicationSecurity	Sécurité des communications
gatewaySecurity	Sécurité des passerelles
routingTable	Table de routage
databaseConf	Configuration de la base de données

TABLE 2.5

Les déclarations :	
const	constante interne
econst	constante d'environnement
xconst	constante du côté client
var	variable interne
evar	variable d'environnement
xvar	variable du côté client

TABLE 2.6

Les mots clés :	
GET	méthode d'obtention
POST	méthode de post-traitement
PUT	mise en place
DEL	suppression
CGET	méthode d'obtention du côté clien
CPOST	méthode de post-traitement du côté clien
CPUT	mise en place du côté clien
CDEL	suppression du du côté clien
ENABLE	mise en uvre
DISABLE	désamorçage
LOG	enregistrement
READ	lecture
WRITE	écriture
import	

TABLE 2.7

Les boucles et les conditions :	
switch case default	tester par rapport à une liste de valeurs
if else continue	déclarations conditionnelles
for while break	répéter une série d'instructions
function return	module de code autonome

TABLE 2.8

Les opérations :		
" ... "	ELLIPSIS	utilisé pour étaler une liste
 "+=", "-=", "*=", "/=", "%=", "&=", "≐=", " ="	ADD_ASSIGN, SUB_ASSIGN, MUL_ASSIGN, DIV_ASSIGN, MOD_ASSIGN, AND_ASSIGN, XOR_ASSIGN, OR_ASSIGN	opérateurs d'affectation
" » "	RIGHT_OP	déplacement vers la droite
" « "	LEFT_OP	déplacement vers la gauche
" ++ "	INC_OP	opérateur d'incrément
" -- "	DEC_OP	opérateur de décrémentation
" -> "	PTR_OP	utilisé pour CHAIN
" & & " , " "	AND_OP, OR_OP	opérateur logique ET/OU
" < = " , " > = " , " = = " , " ! = "	LE_OP, GE_OP, EQ_OP, NE_OP	opérateurs de comparaison
(" { " " < % ")	'{'	accolade ouvrante
(" } " " % > ")	'}'	accolade fermante
" = "	=	opérateur d'affectation
" ("	(parenthèse ouvrante
") ")	parenthèse fermante
(" [" " < : ")	[crochet ouvrante
("] " " : > ")]	crochets fermante
" + " , " - " , " * " , " / " , " ^ "	+, -, *, /, ^	opérateurs mathématiques

TABLE 2.9

Chapitre 3

Implémentation

3.1 Guide d'installation

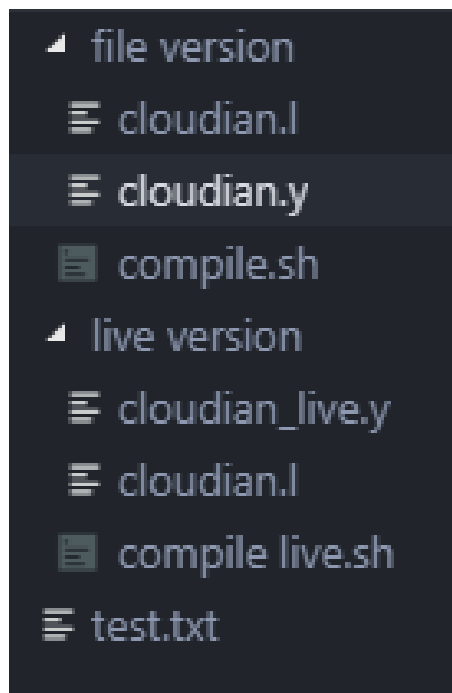


FIGURE 3.1

Il suffit d'exécuter les script shell pour compiler le projet.

Le premier dossier pour la version qui valide un fichier.

Le deuxième pour avoir un éditeur en direct.

3.2 Technologies

Pour la réalisation d'un compilateur pour le langage **Cloudian**, nous avons utilisés LEx, YACC et Bison. Ce compilateur est décomposé en deux parties :

- Lire le programme source et découvrir sa structure.
- Traiter la structure, par exemple pour générer le programme cible.

Lex et Yacc peuvent générer des fragments de programme qui résolvent la première tâche.

La tâche consistant à découvrir à nouveau la structure de la source est décomposée en sous-tâches :

- Découper le fichier source en tokens (Lex).
- Trouver la structure hiérarchique du programme (Yacc).

Pour contruire le parser, nous avons utilisé Bison. Il convertit une description de grammaire pour une grammaire sans contexte LALR(1) en un programme C pour parser cette grammaire. De plus, Bison est compatible avec Yacc : toutes les grammaires Yacc correctement écrites devraient fonctionner avec Bison sans changement.

3.3 Bout du code

```

1  import "/Encryption.cl" as Enc
2  import "/User_Data.cl" as UData
3  import "/Auth.cl" as Auth
4  simport "JENKINS==v1.0" as jenkins
5
6  ManagerSetup{
7      routingTable{
8          CHAIN c1{
9              #//login
10             Auth.api.CheckUser -> Enc.api.Decrypt -> UData.api.CheckDataBase ->{
11                 PROTOCOL p = UData.details.CommunicationProtocol;
12
13                 JSON response = (p,JSON) this.data;
14                 /*To check for fields in this.data: LOG(this.data.describe)*/
15                 if (response.has('has_user')){
16                     PROTOCOL p2 = Enc.details.CommunicationProtocol;
17                     TIME timeout = Time("50s");
18
19                     JSON session = (p2,JSON)Enc.api.NewSessionToken(timeout);
20                     #//return example {"session_token":"hjvsks4qd7q5"}
21                     #ajout de la clef de la session à la base de données
22                     PROTOCOL p3 = UData.details.CommunicationProtocol;
23                     p3 converted_session = (JSON,p3) session;
24                     p3 resp = UData.api.AddSecureSession(response.get('user_id'),converted_session);
25
26                     JSON response = session
27                 }
28                 else{
29                     JSON response = '{}';
30                 }
31                 LOG(response);
32                 return response;
33             }
34         }
35         #//check if the user has session
36         CHAIN c2{
37             Auth.api.CheckSession ->
38             Enc.api.CheckSession ->
39             #//returns session key if timeout hasn't passed (if timeout has passed delete session from db)
40             UData.api.CheckSession->
41             #//returns user_id,session
42             {
43                 PROTOCOL p = UData.details.CommunicationProtocol;
44                 JSON user = (p,JSON) this.data;
45                 if(user.has('user_id')){
46                     return Auth.api.HasUser((JSON,p)user);#//can return chain!
47                 }
48                 else{
49                     return Auth.api.NoUser();
50                 }
51             }
52         }
53         #//logout
54         CHAIN c3{
55             Auth.api.Logout ->

```

```
56     Enc.api.CheckSession ->
57     UData.api.DeleteSession;
58 }
59 GATEWAY(c1);
60 GATEWAY(c2);
61 GATEWAY(c3);
62 }
63 serviceSecurity{
64     #//services have 4 states : Disabled(gray) , Running(green) , Loding(blue) , Down(red)
65     PLUGIN s1{
66         if ((string) UData.getState() == "Down"){
67             Auth.Disable();
68         }
69     }
70     DAEMON s1;
71 }
72 communicationSecurity{
73     CHAIN c1{
74         Auth -> UData -> Enc;
75     }
76     CHAIN c2{
77         Enc -> Auth;#//Enc can't comunicate directly with Auth in order to hide it's processes
78         Auth-> UData; #// the request needs to pass by Enc first!!
79     }
80     TRACK c1;
81     DISABLE c2;
82 }
83 Delivery TOOL(jenkins);
84 }
```

Chapitre 4

Amélioration

4.1 Problèmes rencontrés

La façon avec laquelle le langage est conçu n'est pas LL(1)

On a essayé d'employer un algorithme de conversion en LL(1) cependant le résultat contenait beaucoup de conflits!

On rappelle que Bison compile que les langages LARL(1)

Pas d'analyse sémantique

```
primary_expression
: IDENTIFIER
| HEXA
| NUMBER
| STRING_LITERAL
| '(' expression ')'
;

postfix_expression
: primary_expression
| postfix_expression '[' expression ']'
| postfix_expression '(' ')'
| postfix_expression '('
argument_expression_list ')'
| postfix_expression '.' IDENTIFIER
| postfix_expression INC_OP
| postfix_expression DEC_OP
;
```

FIGURE 4.1

Conclusion

Notre projet consiste à réaliser notre propre langage et d'offrir un ensemble de fonctions permettant de compiler un programme écrit en ce langage.

Pour atteindre cet objectif, on a commencé par présenter le lexique et la grammaire. Après, on a défini l'analyseur lexical et syntaxique. Finalement, nous avons attaqué l'analyseur sémantique.

Ce projet est une expérience très enrichissante. En effet, ce fût une occasion pour nous de mettre en pratique et délargir nos connaissances acquises à IENSIAS.

Bibliographie

- [1] <<http://dinosaur.compilertools.net/lex/index.html>>, 2021. [Online ; accessed 12-Mars-2021].
- [2] <<http://dinosaur.compilertools.net/yacc/index.html>>, 2021. [Online ; accessed 13-Mars-2021].
- [3] Doug Brown John R. Levine, Tony Mason. *Lex Yacc*. Paperback - 366 pages 2nd/updated edition. O'Reilly Associates, October 1992.