République Algérienne Démocratique et Populaire
Université Abou Bakr Belkaid–Tlemcen
Faculté des Sciences

Département d'Informatique

Mémoire de fin d'études Pour l'obtention du diplôme de Master en Informatique

Option : Système d'information et de connaissance (SIC)

Thème

Réalisation d'un site web pour la vérification automatique des solutions du TP de compilation et un analyseur lexicale syntaxique des requêtes SQL

Réalisé par :

- MEDJAHED Manel
- BOUTI Nesrine

Présenté le 18 septembre 2023 devant le jury composé de :

EL YEBDRI Zeyneb	*Présidente*
KHITRI Souad	*Examinatrice*
SETTOUTI Ahmed Khalid Yassine	*Encadrant*

Année universitaire : 2022-2023

Remerciements

Aujourd'hui, nous tenons à exprimer notre profonde gratitude à chacun d'entre vous alors que je prends un moment pour réfléchir sur le chemin parcouru pour atteindre notre mémoire. Cette étape aurait été impossible sans le soutien, les conseils et l'encouragement constants que nous avons reçus tout au long de ce voyage.

Tout d'abord, Nous voudrions adresser nos sincères remerciements à notre encadrant Mr Settouti Ahmed , qui a investi son temps, son expertise et sa passion pour nous guider tout au long de la rédaction de notre mémoire.

L'expression de nos remerciements les plus sincères vont aux membres de notre jury Mme El Yabdri Zeineb et Mme Khitri Souad pour avoir accepté de lire notre mémoire et d'évaluer notre travail, et surtout pour leurs accompagnement et efforts pendant tous notre cursus.

Merci à tous les enseignants du département informatique qui ont fournis des efforts pour nous donner leurs savoir durant tout notre cursus universitaire.

Enfin, merci à nos parents, nos amis et à tous les membres de nos familles respectives de nous avoir apporté leur soutien tout au long de cette démarche.

Dédicaces

À ma chère famille, mes parents aimants, mes amis précieux et mes professeurs inspirants. En ce moment de célébration, je tiens à adresser une dédicace profonde à chacun d'entre vous. Votre soutien, vos encouragements et votre influence ont été les piliers qui ont rendu possible l'achèvement de ce voyage mémorable.

À mes chers parents , vous avez été mon roc, ma source inépuisable de force et de réconfort. Votre confiance en moi et vos encouragements constants ont été les motivations derrière chaque pas que j'ai fait sur ce chemin. Vos sacrifices et votre amour inconditionnel ont été les fondations sur lesquelles j'ai construit cette réalisation.

À mes très chers frères Marwane et hicham pour leur aides et leur encouragements.

À tous les membres de ma famille qui se reconnaîtront dans ce message. Mes chers cousine Abir et Chaimaa

A mes amies : Khadidja, Houda, Ghizlane, Kawthar,Rihem, Asma et Ines, votre présence a illuminé les moments sombres et a doublé la joie des victoires. Vos conversations stimulantes, vos rires contagieux et votre soutien indéfectible ont rendu chaque étape de ce parcours mémorable. Merci d'avoir été mes alliés dans cette quête. Ainsi qu'à tous les gens que j'ai côtoyé durant mon cursus scolaire.

A mes chers professeurs, vous avez été mes guides vers l'excellence. Votre engagement envers l'apprentissage, vos enseignements enrichissants et vos encouragements à repousser mes limites ont façonné mon développement académique et personnel. Votre influence perdurera bien au-delà de ce moment, dans chacune de mes réalisations futures.

Avec tout mon amour et ma reconnaissance

Medjahed Manel

Dédicaces

Tout d'abord, merci à Dieu de m'avoir aidé dans la réalisation de ce modeste travail, que je vais dédier :

A mes très chers parents, aucune dédicace, aucun mot ne pourrait exprimer à leur juste valeur la gratitude et l'amour que je vous porte. Votre soutien et votre encouragement m'ont toujours donné de la force pour persévérer et pour prospérer dans la vie.

A mon chèr mari et mon fils Zakaria.

A ma chère sœur source de force et motivation Et son mari.

A mes frères Faudel et Younes.

A tout ma famille et ma belle famille .

A mes amies et mes proches.

Bouti nesrine

Table des matières

Table des figures

Liste des tableaux

Introduction Générale

L'apprentissage des langages de programmation occupe une place centrale dans le cursus des étudiants en informatique. Parmi ces langages, le langage C joue un rôle essentiel en raison de sa popularité et de son utilisation répandue dans de nombreux domaines. Cependant, l'évaluation des codes écrits en langage C peut représenter un défi pour les enseignants, car elle nécessite une analyse précise de la syntaxe et de la logique du code.

Dans ce contexte, la validation automatique des codes écrits en C offre une perspective prometteuse pour faciliter et améliorer l'apprentissage des étudiants en informatique.

Notre projet de fin d'étude se situe dans le domaine de l'apprentissage de la programmation en langage C, plus spécifiquement dans le contexte des travaux pratiques (TP). Actuellement, la correction manuelle des codes source par les enseignants est la méthode prédominante. Cependant, cette approche présente plusieurs inconvénients, notamment la lourde charge de travail pour les enseignants, la baisse d'efficacité dans l'évaluation des étudiants, et l'incapacité à corriger l'ensemble des erreurs commises par tous les étudiants.

Les problèmes découlant de la méthode de correction manuelle sont multiples. Tout d'abord, la charge de travail excessive pour les enseignants, en raison de grands groupes d'étudiants, entraîne des retards dans les consultations et une baisse de l'efficacité de l'évaluation. De plus, l'attention se concentre souvent sur la correction du code, sans tenir compte du raisonnement sous-jacent suivi par l'étudiant pour aboutir à son code. En outre, les erreurs commises par les étudiants ne peuvent pas toutes être corrigées par les assistants.

La problématique que nous cherchons à résoudre est la suivante : comment développer une solution efficace pour la validation automatique des codes écrits en langage C afin d'améliorer l'apprentissage des étudiants en deuxième année de licence informatique dans le domaine de la théorie des langages ?

Pour répondre à cette problématique, nous envisageons de développer une solution informatique qui repose sur un analyseur lexical et syntaxique des requêtes SQL. Cette solution offrira une plateforme aux étudiants pour améliorer leurs compétences dans le domaine de la théorie des langages. L'objectif principal est de faciliter la correction des TP pour les enseignants, tout en favorisant un apprentissage plus efficace pour les étudiants.

L'objectif de ce mémoire est d'explorer les différentes étapes nécessaires à la réalisation d'un site web vérifiant les solutions des étudiants déposées par le biais de requêtes SQL. En mettant l'accent sur le développement de l'analyseur lexical et syntaxique.Notre système baptisé QueryInspector répend a un besoin réel que recontrent les enseignant durant leurs correction aux solution des étudiants.Pour cela, notre système constitue une aide précieuse et utile pour les parties prenantes.

Mis à part cette introduction, le mémoire est organisé en trois chapitres :

Le premier chapitre commence par expliquer les bases de la théorie des langages et de la compilation. Ceci permet de mieux comprendre le domaine et de repérer les points pertinents pour notre projet. Ensuite, une étude d'ingénierie est présentée pour définir le contexte de notre travail. Par la suite, nous effectuons une analyse approfondie de l'état actuel des connaissances. Cela nous aide à établir un cahier des charges complet et à identifier tous les besoins essentiels pour notre projet.

Dans le deuxième chapitre, intitulé "Conception du QueryInspector", nous décrivons la conception de notre système en utilisant le langage de modélisation UML. Nous incluons des diagrammes de classe, des diagrammes de séquence et des diagrammes de cas d'utilisation. De plus, nous expliquons la méthode que nous avons adoptée pour concevoir notre système, à savoir la méthode UP.

Le troisième chapitre, appelé "Mise en œuvre de QueryInspector", expose l'architecture globale de notre système. Nous détaillons les outils et les langages de programmation que nous avons utilisés pour sa réalisation. Nous présentons également les principales interfaces de notre système et nous illustrons son fonctionnement à l'aide de captures d'écran, mettant en avant ses fonctionnalités clés et les interactions possibles avec le système.

Enfin, ce mémoire se conclut par une synthèse générale de nos travaux.

Chapitre 1

Fondements Théoriques

1.1 Introduction

Ce chapitre des fondements théoriques vise à présenter les principes et les concepts clés qui sous-tendent le développement de notre site Web de correction automatique. Nous avons exploré les bases du theorie de langage et de compilation , les règles d'analyse lexicale et syntaxique, ainsi que les techniques et outils utilisés pour implémenter des analyseurs.

1.2 Théorie des langages

En linguistique et en informatique, nous utilisons la théorie du langage pour décrire les langages formels. Plus généralement, la théorie des langages concerne tout langage fini ou infini qui peut être spécifié par des méthodes ou mécanismes finis et non ambigus permettant de le produire ou de l'analyser. Un langage se définit notamment grâce à la grammaire, et les mots d'une langue sont analysés par des automates.[1]

1.2.1 Définition

La théorie des langages est un domaine qui se consacre à l'étude des langages formels, qui sont des ensembles de mots construits selon des règles précises. Elle vise à développer des modèles mathématiques et des méthodes d'analyse pour comprendre et caractériser les langages, ainsi que pour concevoir des outils et des algorithmes pour leur traitement.
La théorie des langages englobe des concepts tels que les grammaires formelles, les automates, les expressions régulières, les langages réguliers, les langages contextuels, les langages algébriques et d'autres classes de langages formels. Elle trouve des applications dans des domaines variés tels que la compilation, la vérification formelle, la traduction automatique, la bio-informatique, les bases de données et d'autres domaines où la manipulation et l'analyse de langages sont nécessaires.[2]

1.2.2 Mots [1][2]

On se donne un ensemble A, appelé alphabet dont les éléments sont appelés des lettres.

— Un mot de longueur k est une suite de k lettres.
— Le mot vide, de longueur 0, est noté : .
— On note A+ l'ensemble des mots de longueur supérieure ou égale à 1 que l'on peut construire à partir de l'alphabet A.
— On note A^* l'ensemble des mots que l'on peut construire à partir de A, y compris le mot vide .

1.2.3 Langages

un langage L sur l'alphabet A est un ensemble de mots sur A. En d'autres termes, il s'agit d' un sous-ensemble de A* .[2]

1.2.4 Grammaires

Une grammaire est un quadruple G = (T, N, S, R) tel que

— T est le vocabulaire terminal, c'est-à-dire l'alphabet sur lequel est défini le langage.

— N est le vocabulaire non terminal, c'est-à-dire l'ensemble des symboles qui n'apparaissent pas dans les mots générés, Un symbole non terminal désigne une "catégorie syntaxique".

Les non-terminaux représentent une "catégorie grammaticale".

— R est un ensemble de règles dites de réécriture ou de production : $u1 \rightarrow u2$, avec $u1 \in (N \cup T)^+$ et $u2 \in (N \cup T)$.

— $S \in N$ est le symbole de départ ou axiome. C'est à partir de ce symbole non terminal que l'on commencera la génération de mots au moyen des règles de la grammaire. [2]

1.2.5 Les langages réguliers et les automates finis

a) Les automates finis déterministes

M = (Q, S, d, q0 , F) est un AFD

— Q ensemble fini d'états

— S alphabet fini q0 état initial

— F ensemble des états terminaux

— d fonction de transition Q x S -> Q.[4]

b) Les automates finis non déterministes

A priori plus général. Utilité théorique, mais aussi plus facile à construire. Mais plus coûteux à utiliser.

$$M = (Q, S, d, q_0, F)$$

est un AFN Seule différence : $d : QS \rightarrow 2^Q$.[4]

c) Langages réguliers

Un langage L est régulier s'il existe un automate déterministe fini M qui le reconnaît, i.e. si L = L (M).[5] [6]

1.3 Compilation

La compilation permet de convertir un programme écrit dans un langage source en séquence de mots équivalent écrit dans un langage cible. Typiquement, on compile un langage de programmation comme C en assembleur (langage machine), et on utilise un langage intermédiaire comme C. L'analyse lexicale et l'analyse syntaxique sont les premières étapes de la compilation. Dans les compilateurs actuels, elles se font en étroite collaboration : l'analyse lexicale résulte d'une séquence de mots que l'analyse syntaxique peut utiliser. Ils permettent de convertir du texte en un arbre de syntaxe abstraite, qui sera une structure que nous pourrons utiliser pour compiler. Les deux analyses s'appuient sur des outils théoriques simples dont l'expressivité permet d'obtenir des calculs efficaces : expressions rationnelles et syntaxe algébrique. [7]

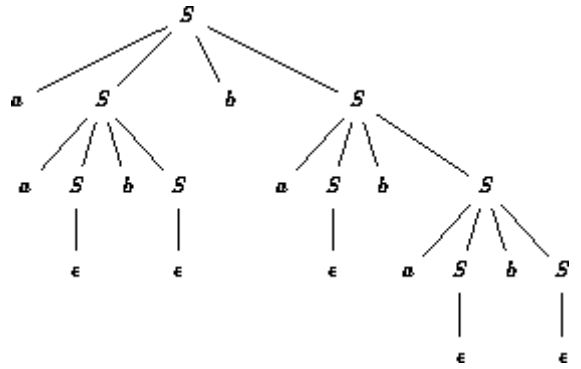


FIGURE 1.1 – arbre de syntaxe abstraite
.[8]

1.3.1 Définition

Le processus de compilation implique d'accéder au code source écrit par le programmeur et d'effectuer des vérifications pour s'assurer que l'ordinateur n'a pas d'ambiguïté. Chaque langage de programmation possède un ensemble de mots-clés qui lancent certaines fonctions. La séquence dans laquelle les mots clés sont utilisés pour former des instructions significatives qu'un ordinateur peut comprendre s'appelle la syntaxe. Le processus de compilation garantit également que le code source est syntaxiquement correct. Le code source est ensuite converti pas à pas en langage machine. [9]

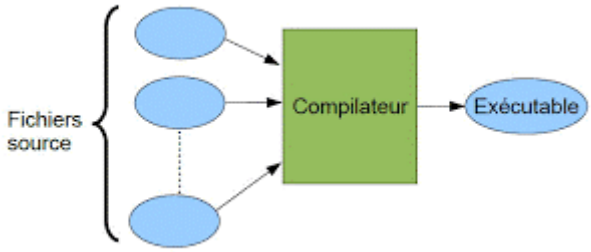


FIGURE 1.2 – Schéma général d'un compilateur
.[10]

1.3.2 Phase de compilation [3]

a) Analyse lexicale

Au cours de cette phase, les caractères isolés qui composent le texte source sont combinés pour former des unités lexicales, les mots de la langue.

b) Analyse syntaxique

L'analyse lexicale identifie les mots dans une langue tandis que l'analyse syntaxique identifie les phrases. Le rôle principal de cette étape est de juger si le texte source appartient à la langue considérée, c'est-à-dire s'il est correct du point de vue de la grammaire de cette dernière.

c) Analyse sémantique

La structure du texte source est correcte, le problème est ici de vérifier certaines propriétés sémantiques. Génération de code intermédiaire Après la phase d'analyse, certains compilateurs ne génèrent pas directement le code de sortie attendu, mais génèrent une représentation intermédiaire, un code pour une machine abstraite. Cela permet de concevoir indépendamment la première étape du compilateur (qui constitue ce que l'on appelle le frontal), qui ne dépend que du langage source Ne dépend que des étapes compilées et finales de la langue cible (formant son verso) ; idéalement il y a plusieurs fronts et plusieurs versos qui peuvent être librement assemblés.

d) Optimisation du code

Cela implique généralement de transformer le code afin que le programme résultant s'exécute plus rapidement.

e) Génération du code final

est la plus impressionnante pour les novices, mais ce n'est pas forcément la plus difficile à réaliser. Elle nécessite la connaissance de la machine cible (réelle, virtuelle ou abstraite), notamment ses possibilités en termes de registres, piles, etc.

La compilation d'un programme se déroule en deux parties successives :

1.3.3 l'analyse et la synthèse

a) L'analyse

Elle permet de trouver la structure du programme à partir du texte du programme, c'est-à-dire d'une suite de caractères. L'analyse est uniquement dépendante du langage à traduire (indépendante du langage cible) . [11]

b) La synthèse

Après analyse du programme source, c'est la partie synthèse. Il permet de traduire la structure d'un programme en tenant compte des possibilités du langage cible (optimisation). Cela dépend de la définition du langage cible (choix de l'architecture de la machine, des ressources disponibles, etc.) et du langage compilé (c'est-à-dire de la signification liée à la structure du programme : la sémantique du langage) . [11]

c) Table des symboles

Information concernant le type, la portée, l'emplacement-mémoire des identificateurs. Initialisée par l'analyseur lexical, complétée et utilisée par les autres phases .

d) Gestion des erreurs

Erreur lexicale (caractère interdit). Erreurs syntaxiques (non-respect des règles structurelles). Erreur sémantique (fonctionnement incohérent : contrôle statique).

Selon l'objectif du compilateur, diverses récupérations sont possibles : arrêt à la première erreur ; resynchronisation avec la construction correcte suivante ; correction des tentatives.[12]

1.3.4 Outils pour la construction automatique d'un compilateur

a) Générateur d'analyseur lexicale

FLex est un outil pour construire automatiquement des analyseurs lexicaux. Il reçoit des fichiers de spécification pour différents jetons du langage en utilisant la notation des expressions régulières et génère des programmes (c source) qui permettent l'analyse lexicale des fichiers d'entrée .[13]

Flex est une version libre de l'analyseur lexical Lex. Nous présentons une figure qui définit le processus de création d'un analyseur lexical avec Flex :

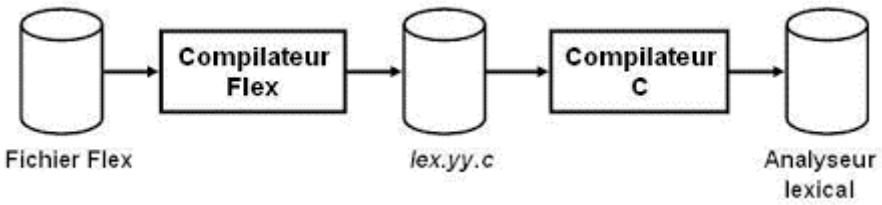


FIGURE 1.3 – Processus de création d'un analyseur lexical avec Flex
.[14]

b) Constructeur d'analyseur syntaxique(GNU Bison)

bison est un compilateur de compilateur, version gnu de la célèbre commande yacc acronyme de « yet another compiler of compilers ». Il construit un compilateur d'un langage décrit par un ensemble de règles et actions d'une grammaire LARL .[15]

1.4 Recherche d'ingénierie

Flex/bison	Lex/yacc
Bison et Flex sont des alternatives gratuites, et open-source.[16]	Yacc (Yet Another Compiler Compiler) et lex, deux générateurs d'analyseurs syntaxique et lexical pour les systèmes Unix.[17] [18]
Bison est d'analyseur LALR, il est déterministe, il a un temps de calcul linéaire[19]	Lex est capable de traiter des langages de type 3 (réguliers) [20]
les fichiers Bison et Flex de départ sont relativement simples à lire et écrire.	Yacc fournit le code nécessaire à l'analyse de langages de type 2 (non-contextuels).[20]
Flex joue le rôle d'analyseur. Bison joue le rôle du parseur.[21]	Yacc utilise des règles de grammaire qui lui permettre d'analyser les jetons de lex et de créer un arbre de syntaxe.[22]
La génération d'un parseur très rapide.[16]	Les grammaires pour yacc sont décrites en utilisant une variante de la forme BNF.(Une grammaire BNF peut être utilisée pour exprimer des langages sans contexte .[22]
Un code avec moins de bogues dû à ses nombreuses années d'expériences.[21]	lex accepte une spécification de haut niveau axée sur les problèmes pour la correspondance des chaînes de caractères.[23]

TABLE 1.1 – Comparaison des outils similaires

La recherche des outils similaires à la contribution est une partie indispensable. Bien que Flex/GNU Bison et Lex/Yacc partagent des fonctionnalités similaires pour l'analyse lexicale et syntaxique, Flex et Bison ont beaucoup

de possibilités supplémentaires. Ils sont plus récents et performants que Lex et Yacc qui sont les versions originales. Ils sont disponibles pour un très grand nombre de plateformes. L'une des raisons courantes de choisir Flex/GNU Bison plutôt que Lex/Yacc est liée à leur implémentation plus moderne.

1.5 Analyseur lexical syntaxique des requêtes SQL

Un analyseur lexical prend une chaîne de caractères en entrée et la découpe en "jetons" ou "tokens", ces unités lexicales sont des noms de table , des noms de colonne , des opérateurs , des valeurs , etc. L'analyseur utilise ces jetons pour vérifier la validité de la syntaxe de la requête.

Forts de cette comprehension theorieque approfondie, nous sommes maintenant en mesure d'aborder la phase de conception avec une perspective éclairée. Les concepts et modèles abordés dans ce chapitre nous fournissent une boîte à outils solide pour formuler des solutions créatives et efficaces, en tenant compte des contraintes et des objectifs du projet.

1.6 Conclusion

En conclusion, ce premier chapitre consacré à la théorie du langage, à la compilation, ainsi qu'aux outils similaires, a permis d'établir une solide base de connaissances.

Dans le chapitre suivant, nous entamerons l'état de l'art, une phase consacré pour la comparaison entre des système similaire a notre système déjà existants.

Chapitre 2

Etat de L’art

2.0.1 Introduction

L'état de l'art représente une étape cruciale dans la recherche, permettant d'explorer en profondeur les travaux existants dans un domaine spécifique. C'est une démarche qui va au-delà de la simple découverte des réalisations antérieures, car elle nous offre la possibilité de signaler les limites des projets similaires. En analysant les réussites et les échecs de ces projets précédents, nous pouvons identifier les lacunes et les défis non résolus. De plus, l'état de l'art nous permet de mettre en évidence les avantages de notre projet proposé par rapport aux projets similaires déjà existants. En nous appuyant sur les enseignements tirés de ces travaux antérieurs, nous sommes en mesure de développer des méthodes innovantes pour résoudre ces limites et relever ces défis. En fin de compte, cette démarche nous aide à démontrer clairement la valeur ajoutée de notre projet par rapport à d'autres initiatives similaires, tout en contribuant à l'avancement des connaissances et à l'innovation dans le domaine.

2.0.2 Comparatifs des projet similaires

Codacy : est un logiciel qui permet de tester et de réviser un code automatiquement et surveille sa qualité . Analyse statique, couverture de code et métriques pour Ruby, JavaScript, PHP, Scala, Java, Python, CoffeeScript et CSS. Il envoie les résultats sous forme de commentaires dans vos pull requests ou sous forme de notifications dans les canaux Slack ou Hipchat. Codacy joue également bien avec vos outils d'intégration continue et sert de complément idéal à vos tests unitaires. [24]

SQL Fiddle : est un interpréteur SQL en ligne, disponible sur le Web, qui permet la création et l'exécution de code SQL sans installer de SGBD.

OnlineGBD : C'est un outil de compilation et de débogage en ligne pour le langage C/C++. C'est le premier IDE en ligne au monde à fournir des capacités de débogage à l'aide du débogueur gdb intégré. Il s'agit d'une application Web très pratique pour les codeurs qui aiment coder dans un IDE en ligne mais qui sont confrontés à des plantages inattendus et à des erreurs délicates dans leur code. OnlineGDB fournit des fonctionnalités de débogage pour ces utilisateurs afin de les aider. Dans l'ensemble, c'est un excellent IDE en ligne alimenté par un éditeur de code, un compilateur et un débogueur. [25]

Le tableau présente un comparatif des projet précedents : Codacy , OnlineGBD , SQL Fiddle , suite à notre recherche et analyse des références suivantes : [26] [27] [28] [29].

	CODACY	ONLINEGBD	SQL FIDDLE
UTILISATEURS	Les développeurs	Les entreprises et Les professionelles	Pour tous le monde
ELIMINATION DE CHARGE DE TRAVAIL (POUR LA CORRECTION MANUELLE)	oui	oui	oui
CORRECTION DES ERREURS QUI SE REPETE	oui	oui	oui
CORRECTION DU CODE (PAS DU RAISONNEMENT)	non	oui	oui
EVALUATION FACILE	oui	non	non
ASSISTANCE FIABLE	oui	oui	oui
ANALYSE DU CODE	oui	oui	oui
ABONNEMENT MENSUELLE	oui	non	non

TABLE 2.1 – Comparaison des projet similaires

Le tableau précédent montre une comparaison entre les principaux sites qui font l'analyse du code et/ou la correction, cette comparaison est faite suite à des critères :

On remarque que CODACY est le seul qui fait une évaluation par rapport au code ainsi qu'il fait la correction du code et du raisonnement le contraire de SQL FIDDLE et ONLINEGBD qui corrigent uniquement le code et non pas le raisonnement.

De plus CODACY est un site payant , ONLINEGBD a une version gratuite qui peut devenir payante si le besoin des outils ou de fonctionnalité augmente, par contre SQL FIDDLE est gratuit .

Par rapport a l'analyse de code, l'assistance, l'élimination du charge de travail et correction des erreurs persistes on les trouve sur tout les sites présentés.

Il nous reste les utilisateurs qui change pour chaque site par exemple pour CODACY on trouve les développeurs , ONLINEGBD est réservé pour les entreprise et les professionnels , et SQL FIDDLE est dédié pour tout type d'utilisateurs .

Donc on constate des petits problemes après notre analyse :

— ONLINEGBD et CODACY les deux sites ne sont pas d'édier aux etudiants qui ont aussi besoin de tels service.

— La version gratuite de ONLINEGBD ne contient pas toute les fontionnalité dont l'utilisateur besoin.

— SQL FIDDLE et ONLINEGBD ne contient pas la possibilité d'évaluer le travail .

— Pour CODACY , les frais d'abonnement sont un peut dure pour un simple etudiant qui n'as aucun revunu.

2.0.3 Suggestion

Nous proposons un site web spécialement conçu pour automatiser la vérification du code d'un analyseur lexical et syntaxique des requêtes SQL. Dans ce contexte, le site évalue l'étudiant en se basant sur les cas de test réussis plutôt que l'enseignant. Cette approche vise à alléger considérablement la charge de travail du professeur, qui peut être très importante.

2.1 Conclusion

En parcourant l'état de l'art, nous avons pu prendre connaissance des avancées récentes, des tendances et des outils innovants .

Dans le chapitre suivant, nous entamerons la phase cruciale de la conception de notre système.

Chapitre 3

Conception du QueryInspector

3.1 Introduction

Dans le but d'assurer une gestion de projet optimale et de structurer efficacement notre travail, nous avons opté pour une phase de conception , basée sur l'utilisation de diagrammes. En effet, cette approche nous a permis de mieux comprendre et visualiser les différentes interactions entre les différentes parties de notre système. Grâce à ces diagrammes, nous avons pu schématiser notre système de manière claire et concise, facilitant ainsi le développement et assurant un travail plus efficace.

En somme, la phase de conception basée sur l'utilisation de diagrammes s'est avérée être une étape cruciale dans la réussite de notre projet, nous permettant d'assurer une gestion optimale et une réalisation efficace de notre système

3.2 Le langage de modélisation UML

UML (Unified Modeling Language, qui veut dire langage de modélisation unifié en français) est une notation permettant de modéliser un problème de façon standard. Ce langage est né de la fusion de plusieurs méthodes existant auparavant, et est devenu désormais la référence en terme de modélisation objet.

3.3 La méthode UP

Unified Process(que l'on peut traduire par Processus Unifié) est un processus de développement logiciel conçu par les développeurs d'UML Jacobson, Booch, Rumbaugh en 1999, regroupe les activités à mener pour transformer les besoins d'un utilisateur en système logiciel.
L'objectif d'un processus unifié est de maîtriser la complexité des projets informatiques en diminuant les risques. Il permet de répondre aux quatre préoccupations suivantes :

- QUI participe au projet ?
- QUOI,qu'est-ce qui est produit durant le projet ?
- COMMENT doit-il être réalisé ?
- QUAND est réalisé chaque livrable ?

Le processus de développement UP, associé à UML, met en œuvre plusieurs principes, parmi eux l'utilisation des diagrammes de cas d'utilisation, de séquence et de classe que l'on va essayer de réaliser pour notre projet.[30]

3.4 Analyse et conception du systeme de correction automatique

Dans cette partie, nous allons explorer la manière dont nous avons conçu notre site web, ainsi que notre analyseur lexical. Nous avons utilisé les différents diagrammes UML tels que le diagramme de cas d'utilisation, le diagramme de classes et le diagramme de séquence pour modéliser notre système de manière claire et précise. Nous avons également effectué une analyse approfondie des besoins de nos utilisateurs, ce qui nous a permis de concevoir un système adapté et efficace.

Dans notre projet l'analyseur lexical syntaxique est la partie essentielle , c'est a partir de cette dernière qu'on peut comparer les résultat des codes solutions des étudiants pour les évaluer.

3.4.1 Conception du site web

Dans notre système, on aura trois différents acteurs et chacun d'entre eux représente une entité externe qui interagit directement avec le système, ces acteurs sont :

— L'administrateur.

— L'enseignant.

— les étudiants.

On a choisi d'utiliser 3 diagrammes principaux d'UML qui nous ont semblé être les plus adéquats pour notre étude, à savoir :

— Le diagramme de cas d'utilisation qui sert à modéliser les besoins des utilisateurs.

— Le diagramme de séquence système pour détailler tous les scénarios possible des acteurs avec le système.

— le diagramme de classe pour la modélisation, la communication et la conception de systèmes orientés objet. Il facilite la compréhension de la structure du système, l'analyse des fonctionnalités requises et la mise en œuvre efficace du logiciel.

Enfin, pour la création des différents diagrammes qui vont suivre, on a utilisé :

— Modelio qui est un outil open source qui prend en charge la norme UML2.

3.4.2 Diagrammes de Cas d'Utilisation

Le diagramme de cas d'utilisation permet d'identifier les différents utilisateurs d'un système ainsi que les actions qu'ils sont censés effectuer. Chaque action effectuée par un utilisateur est représentée par un cas d'utilisation distinct. Dans le contexte de notre application, trois acteurs principaux sont à distinguer : l'enseignant et l'etudiant et L'administrateur.

Nous allons présenter ci-dessous les différents diagrammes qu'on a réalisé.

a) Cas d'utilisation enseignant et L'administrateur

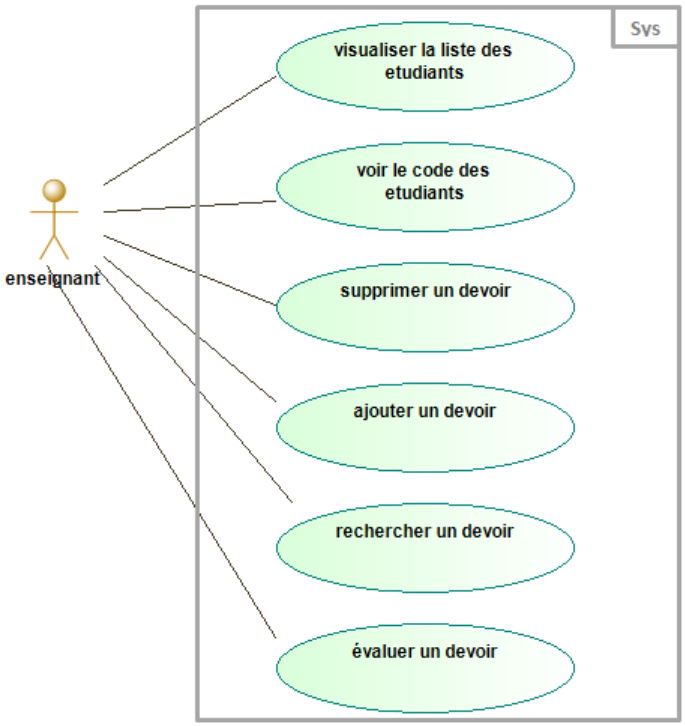


FIGURE 3.1 – Diagramme de Cas d'Utilisation Enseignant

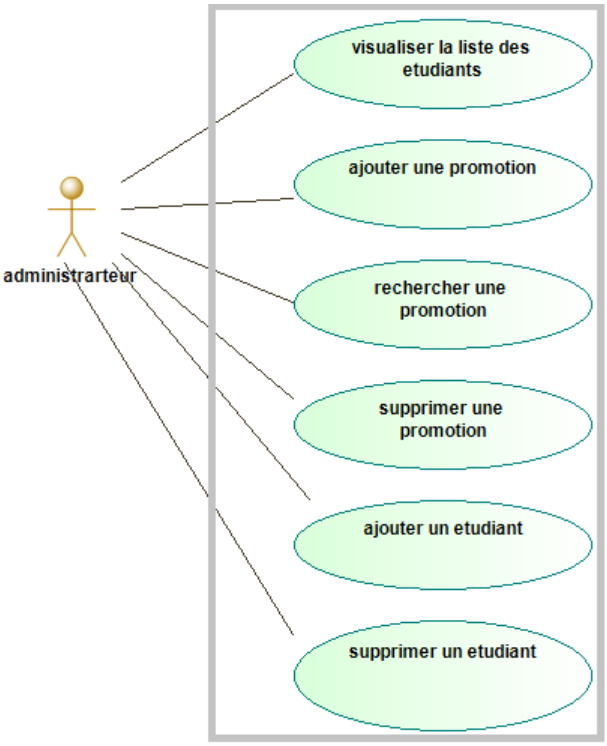


FIGURE 3.2 – Diagramme de Cas d'Utilisation administrateur

Scenario :

— L'enseignant se connecte à son compte à l'aide de ses identifiants de connexion.

- Après être authentifié, l'enseignant accède à son espace sur la liste des promotions et peut choisir d'effectuer une seule action :
  - Rechercher une promotion : l'enseignant saisit les critères de recherche tels que le libellé, la date de début, etc., dans la barre de recherche prévue à cet effet, puis lance la recherche.
  - L'administrateur, quant à lui, peut effectuer toutes les actions mentionnées ci-dessous :
  - Ajouter une promotion : l'administrateur saisit les informations de la promotion, telles que la date de début, la date de fin et le nombre de groupes, dans le formulaire prévu à cet effet, puis enregistre les données.
  - Supprimer une promotion : l'administrateur sélectionne la promotion à supprimer dans une liste de promotions existantes.
  - Rechercher une promotion : l'administrateur saisit les critères de recherche tels que le libellé, la date de début, etc., dans la barre de recherche prévue à cet effet, puis lance la recherche.
    - Ainsi, l'enseignant a uniquement la possibilité de rechercher des promotions, tandis que l'administrateur peut effectuer toutes les actions mentionnées dans le scénario initial.
    - Le système affiche la liste des étudiants inscrits dans cette promotion, avec leurs informations personnelles telles que leur nom, prénom, adresse e-mail, etc.
    - L'administrateur a également la possibilité d'ajouter un nouvel étudiant à cette promotion en cliquant sur le bouton "Ajouter un étudiant" et en remplissant le formulaire avec les informations requises.
    - Lorsque l'administrateur souhaite supprimer un étudiant, il sélectionne l'étudiant à supprimer dans une liste des étudiants existants.
    - Si l'enseignant et l'administrateur choisissent de rechercher un étudiant, ils saisissent les critères de recherche (nom, prénom, etc.) dans la barre de recherche, puis lancent la recherche.
- Si l'enseignant choisit d'ajouter un devoir, il saisit les informations du devoir (numéro du TP, date de début, date de fin, fiche de TP, etc.) dans le formulaire prévu à cet effet, puis sélectionne la promotion à laquelle il souhaite attribuer le devoir dans une liste de promotions existantes (par défaut il peut choisir que la promotion de l'année actuelle).
- Une fois le devoir attribué aux étudiants, l'enseignant peut consulter la liste des étudiants inscrits dans cette promotion ainsi leurs devoirs attribués.
- Le système affiche la liste des étudiants à qui le devoir a été attribué, ainsi que leur avancement (envoyé ou non envoyé). L'enseignant peut sélectionner un étudiant pour voir son devoir.
- Lorsque l'étudiant a soumis son devoir, l'enseignant peut consulter le code et attribuer une note.
- le système enregistre les données et peut passer à l'étudiant suivant.
- Si l'enseignant choisit de supprimer un devoir, il sélectionne le devoir à supprimer dans une liste de devoirs existants.
- Si l'enseignant choisit de rechercher un devoir, il saisit les critères de recherche (numéro de TP, date de fin, etc.) dans le formulaire prévu à cet effet, puis lance la recherche.
- Après avoir effectué une ou plusieurs actions, l'enseignant peut se déconnecter en cliquant sur le bouton de déconnecter.

b) Cas d'utilisation étudiant

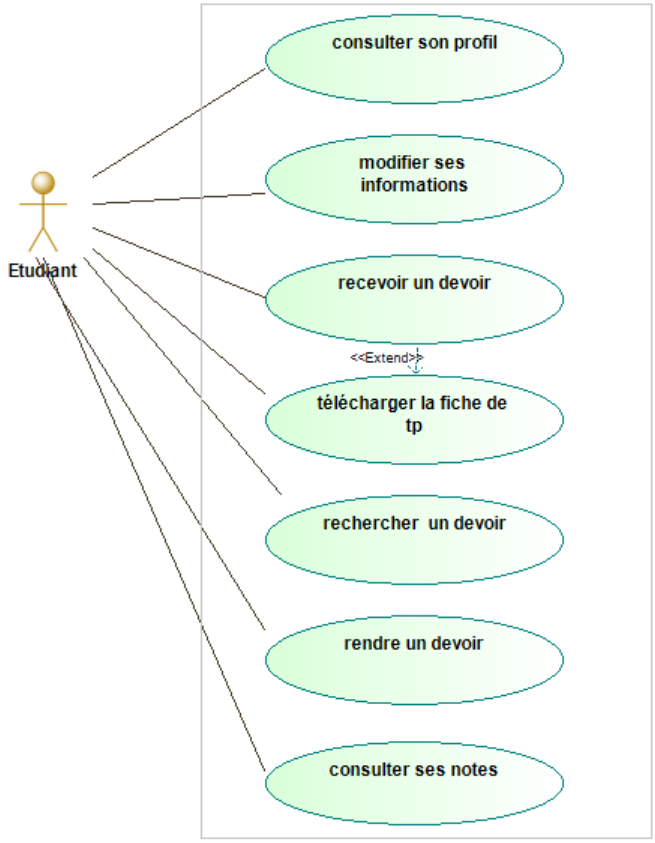


FIGURE 3.3 – Diagramme de Cas d'Utilisation Etudiant

Scenario :

— L'étudiant accède à la page d'inscription et saisit ses informations personnelles dans le formulaire prévu à cet effet. Il valide ensuite l'inscription.

— L'étudiant se connecte à son compte en saisissant ses identifiants de connexion (adresse e-mail et mot de passe).

— Après s'être authentifié, l'étudiant est redirigé vers son profil, où il peut consulter ses informations personnelles. S'il le souhaite, il peut modifier ces informations en cliquant sur le bouton "Modifier le profil".

— L'étudiant consulte la liste des devoirs envoyés par l'enseignant en sélectionnant « Devoirs » dans le menu principal. Il peut rechercher un devoir en saisissant les critères de recherche (numéro du TP, date de fin, etc.) dans la barre de recherche.

— L'étudiant sélectionne le devoir qu'il souhaite rendre et soumet son travail avant la date limite.

— Si le délai n'est pas encore dépassé, l'étudiant peut re-rendre son devoir en cas d'erreur ou de modification de dernière minute.

— L'étudiant peut consulter ses notes pour chaque TP. Si le système a attribué une note, elle sera affichée. Sinon, la note sera indiquée comme 0 par défaut.

— Après avoir effectué les actions souhaitées, l'étudiant peut se déconnecter en cliquant sur le bouton de déconnexion.

— l'étudiant doit être authentifié pour accéder à toutes les fonctionnalités, sauf l'inscription.

3.4.3 Diagrammes de Séquence Système

Le diagramme de séquence est un type de diagramme qui permet de représenter les interactions entre les acteurs et le système selon un ordre chronologique. Dans ce contexte.
Nous allons présenter les différents diagrammes qu'on a réalisé dans les lignes qui suivent.

a) L'ajout d'une promotion et l'inscription des étudiants

Le diagramme présenté ci-dessous illustre la fonctionnalité permettant à un administrateur d'ajouter une nouvelle promotion et d'y inscrire des étudiants. Lorsqu'un étudiant souhaite s'inscrire, un formulaire s'affiche et doit être rempli. Une fois le formulaire complété, le système effectue automatiquement des tests de vérification. Si aucune erreur n'est détectée, l'étudiant est ajouté à la base de données et est redirigé vers son profil.

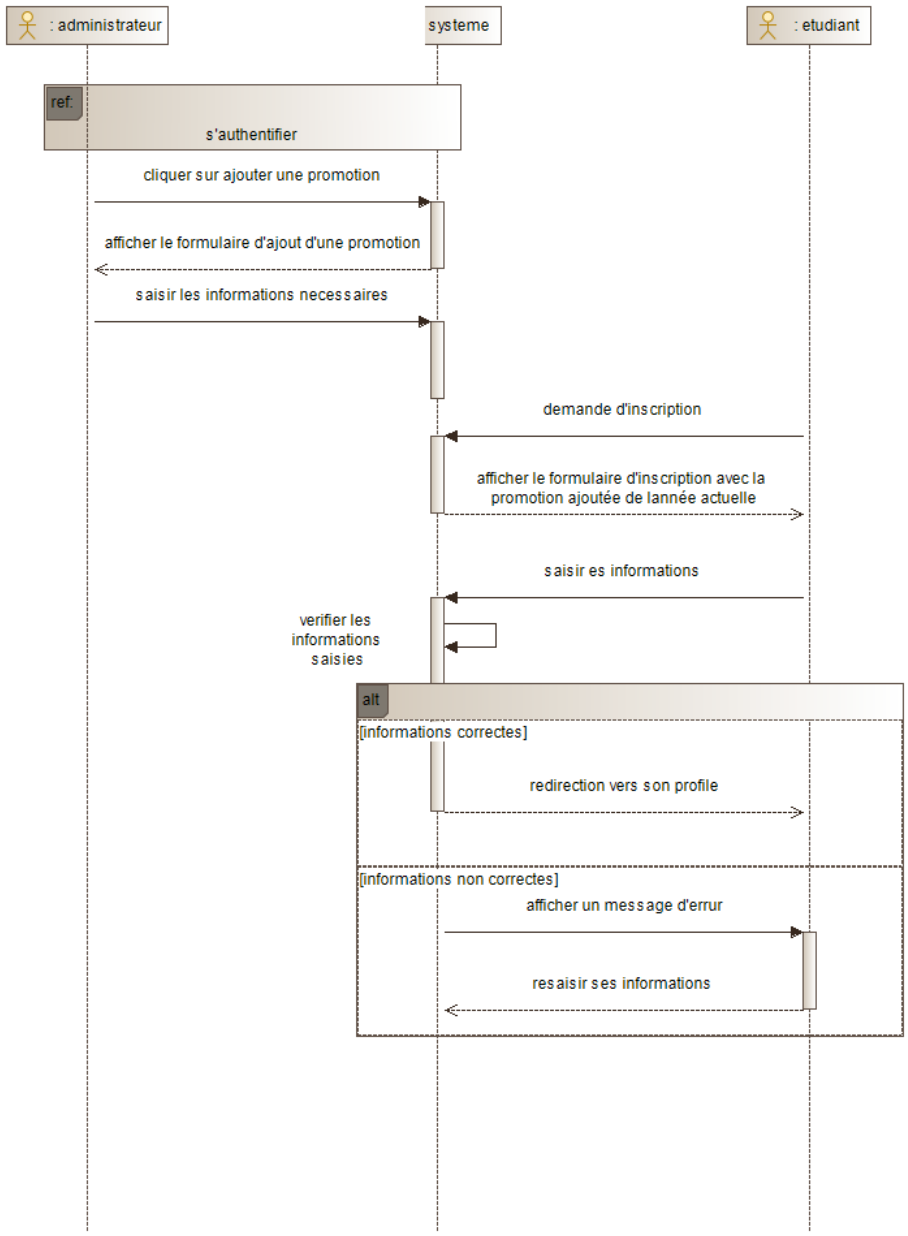


FIGURE 3.4 – Diagramme de Séquence ajout promotion et inscription etudiants

b) L'ajout et la soumission d'un devoir

Le diagramme ci-dessous présente la fonctionnalité permettant à un enseignant d'ajouter un nouveau devoir. Il remplit le formulaire et choisit la promotion à laquelle il souhaite envoyer ce devoir. Les étudiants de la promotion sélectionnée reçoivent le devoir et peuvent le compléter jusqu'à ce que la date limite soit atteinte. Pendant cette période, ils peuvent soumettre ou re-soumettre leurs devoirs. Passé le délai, ils ne peuvent plus rien faire.

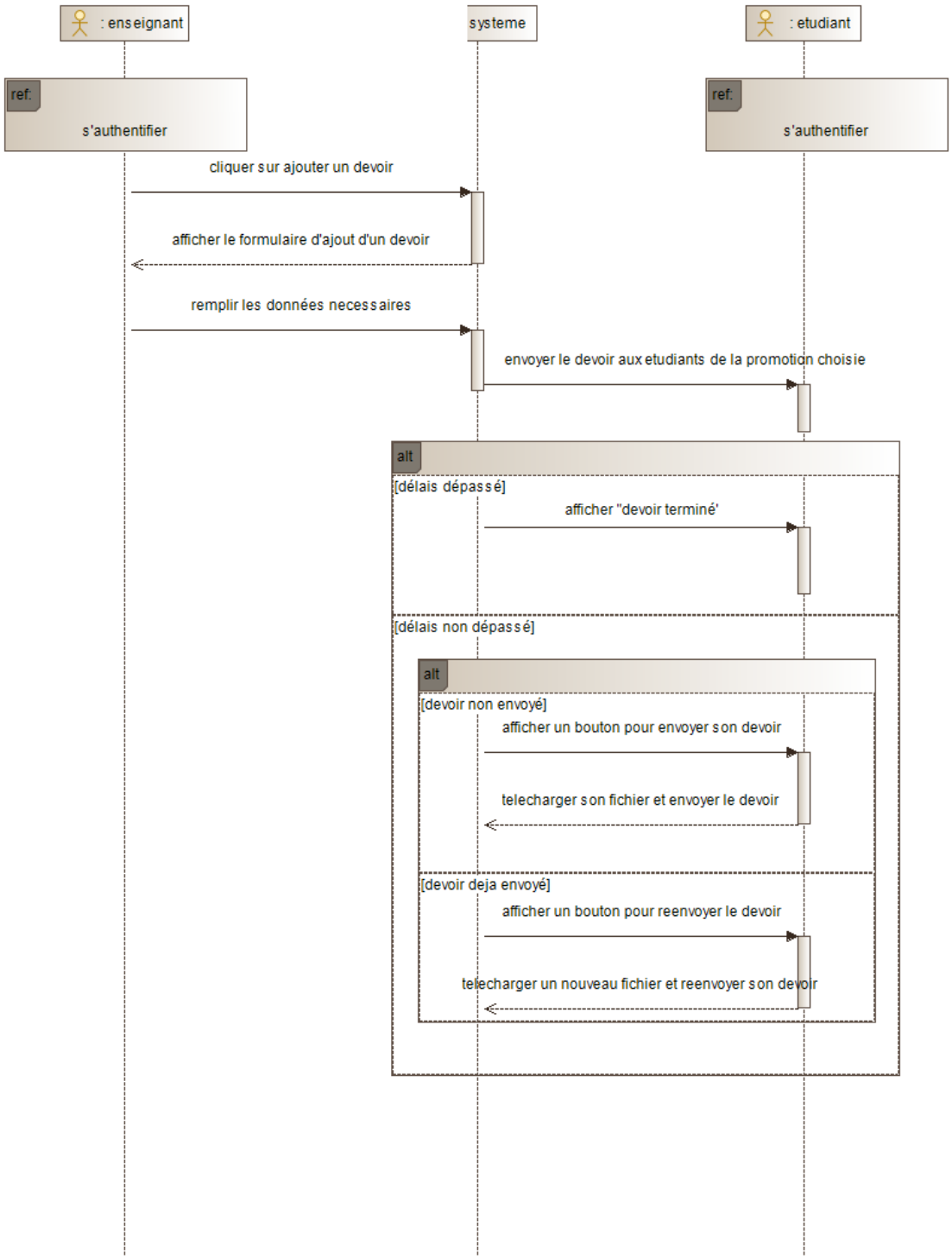


FIGURE 3.5 – Diagramme de Séquence ajout et soumission des devoirs

c) Evaluation des devoirs

Le diagramme ci-dessous présente la fonctionnalité permettant à un enseignant de noter et de visualiser les devoirs des étudiants. Pour cela, il clique sur le devoir qu'il souhaite consulter et la liste des étudiants de la promotion qui ont reçu ce devoir s'affiche. Si un étudiant a soumis son devoir, deux boutons apparaissent : "voir le code", qui permet de visualiser le devoir soumis, et "analyser le code", qui permet de donner une note pour le devoir soumis à l'aide de l'analyseur lexical. Si aucun devoir n'a été soumis, la note sera automatiquement 0 pour ce devoir.

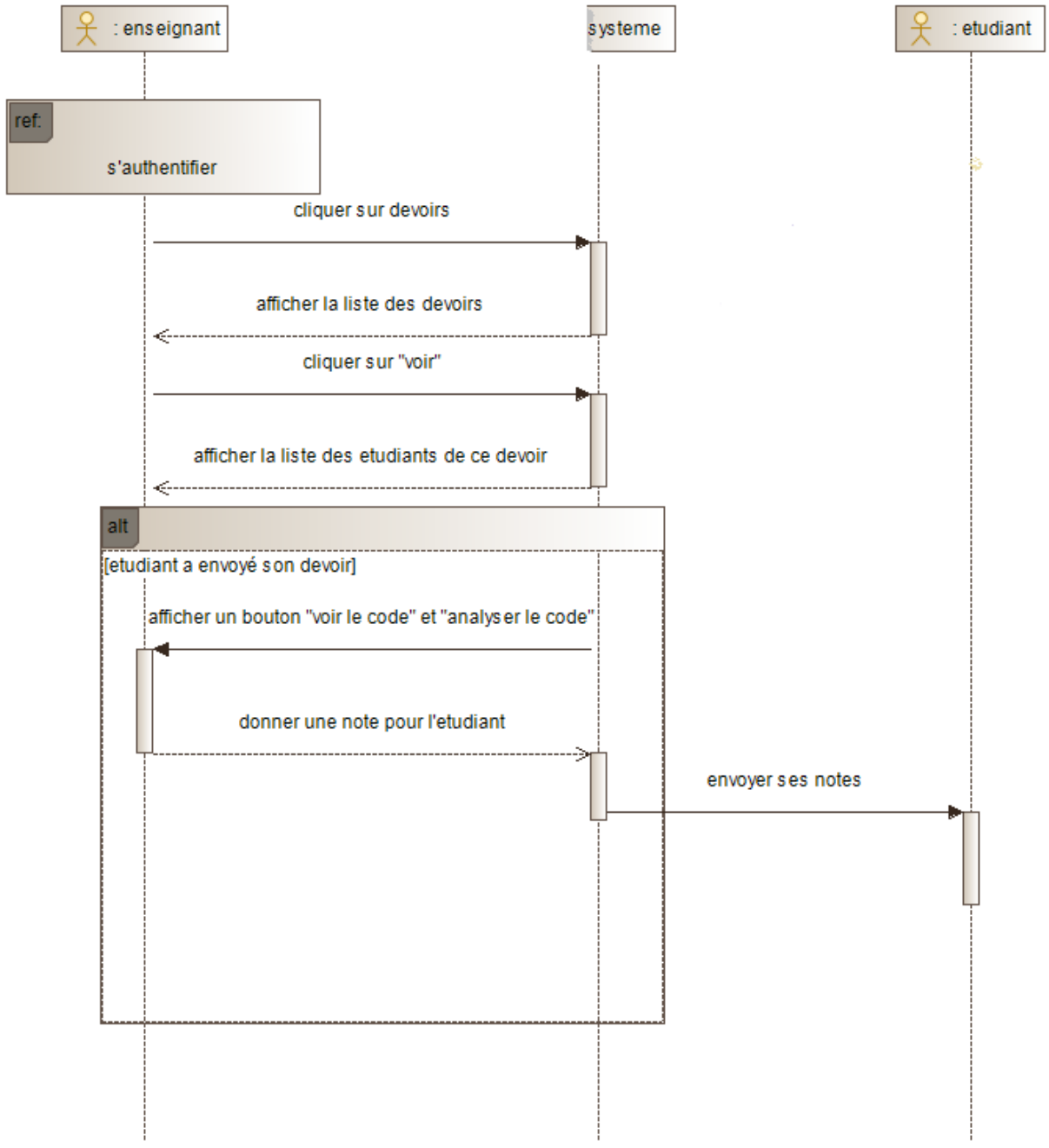


FIGURE 3.6 – Diagramme de Séquence Evaluation des devoirs

d) Diagramme de classe

Le diagramme de classe présente les connexions entre les différentes tables qui sont nécessaires pour assurer le bon fonctionnement de notre système.

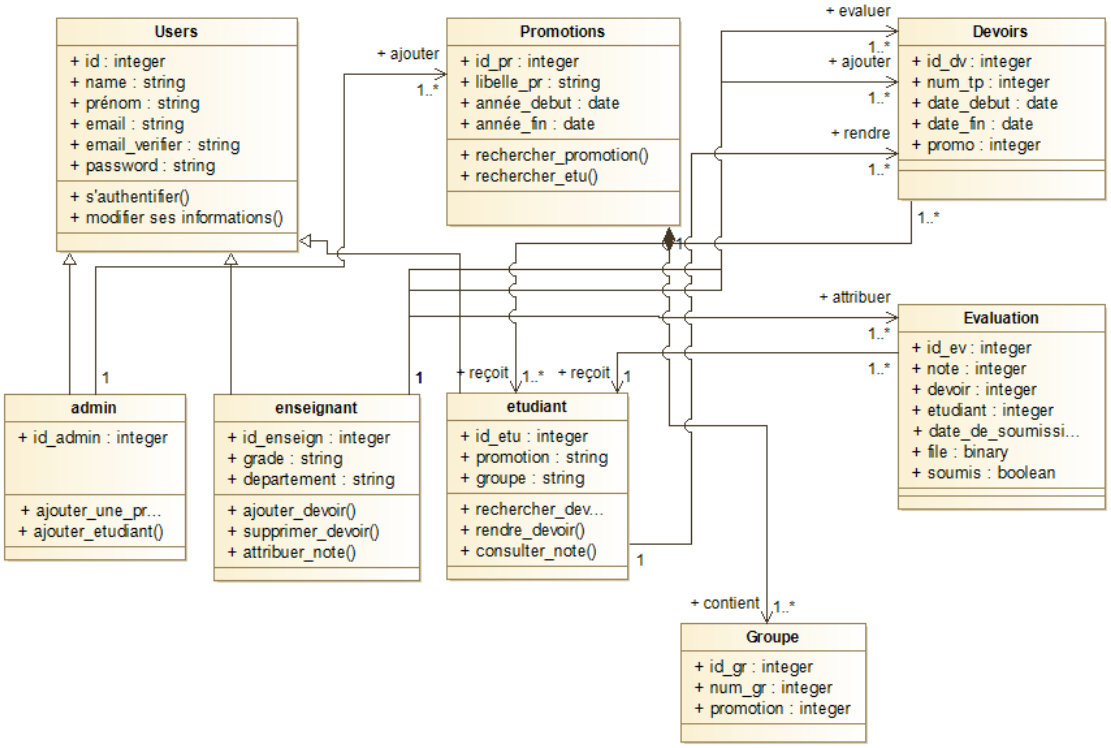


FIGURE 3.7 – Diagramme de Classe

3.4.4 Conception de l'analyseur lexical syntaxique

L'analyseur lexical et syntaxique (ou "parser") est une partie essentielle de notre système qui permet de vérifier la validité de la syntaxe des requêtes SQL soumises par les utilisateurs .Pour générer l'analyseur nous avons suivi les étapes suivante :

1. Définition de la grammaire : Commencez par définir la grammaire formelle de la requête SQL que nous souhaitons analyser. Cela comprend les règles syntaxiques et la structure attendue de la requête.
2. Écriture du fichier Flex (analyseur lexical) : Nous avons utiliser Flex pour écrire les règles de correspondance entre les motifs lexicaux (tokens) et les actions associées. Nous avons Défini les expressions régulières pour reconnaître les différents types de tokens dans la requête SQL, tels que les mots-clés, les identificateurs, les opérateurs, les parenthèses, etc.l'analyseur doit être écrit comme un automate globale permettant de reconnaitre toutes ces dernières. Une unité lexicale est reconnue si l'automate est dans l'état final de celle-ci et un caractère séparateur est lu. Ainsi, celui-ci ajoute l'unité lexicale reconnue et transite à l'état initiale.

   Ci-dessus, nous exposons un automate qui prend en compte une requête SQL simple dans le cadre du TP de compilation :

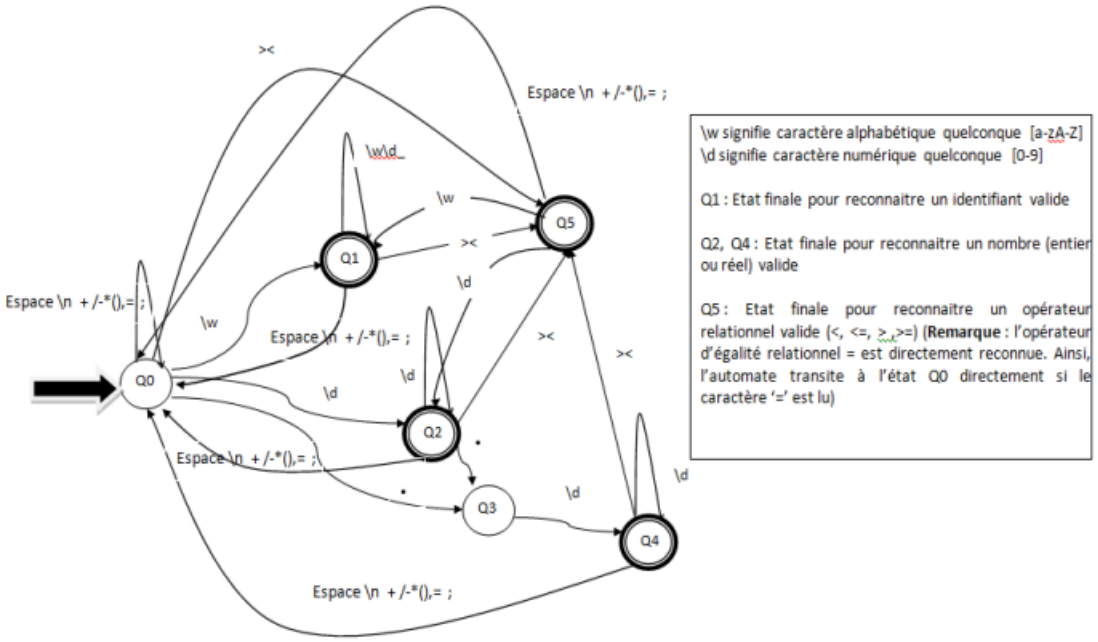


FIGURE 3.8 – Automate d'une requete SQL

3. Écriture du fichier Bison (analyseur syntaxique) : Nous avons employé Bison pour établir les règles de syntaxe ainsi que les actions correspondantes. Nous avons défini les règles pour diverses composantes de la requête SQL, notamment les clauses SELECT, FROM, WHERE, JOIN, etc. Pour chaque règle, nous avons précisé les actions à entreprendre lors de la reconnaissance de ladite règle, telles que la validation syntaxique et la création d'une structure de données qui représente la requête. Afin d'illustrer notre démarche de manière plus explicite, nous présentons une grammaire d'une requête simple. :

```
Requête : SELECT colone_list FROM table_nom WHERE colone ou valeur
// ici dans requête on définit la syntaxe de la requête sql (comment
doit être écrite)
Colone_list : colone_nom | colone_list and colone_nom
// la définition de la grammaire d'une colonne qui doit être un nom
d'une colonne ou une liste de nom de colonne
Colone_nom : IDENTIFIANT   //le nom de la colonne est une chaine de
caractère
Table_nom :  IDENTIFIANT   // le nom de la table est une chaine de
caractère
// le where est suivi soit par un nom d'une colonne oui une valeur
précise
Colone  : IDENTIFIANT    // colonne est une chaine de caractère
Valeur : NUMBER   // valeur est un numéro
```

4. Intégration du code Flex et Bison : Le code Flex doit générer des jetons que le code Bison peut reconnaître et utiliser dans ses règles de syntaxe.

3.5 Conclusion

Ce chapitre fournit un aperçu de notre application "QueryInspector". Nous avons commencé par la conception de notre analyseur lexical, puis nous avons présenté différents diagrammes UML tels que :

Le diagramme de cas d'utilisation, qui décrit les acteurs impliqués dans notre système ainsi que leurs rôles respectifs.

Le diagramme de séquence, qui présente les interactions chronologiques entre les acteurs et les différents objets de notre système.

Le diagramme de classe, qui identifie les différentes classes impliquées dans notre application.

Dans le chapitre suivant, nous allons aborder la dernière partie de notre travail qui représente la réalisation de notre application web.

Chapitre 4

Mise en œuvre de QueryInspector

4.1 Introduction

La réalisation est une étape très importante car c'est grâce à elle que notre application web va voir le jour. Sa réussite dépend principalement du choix des outils choisis pour sa mise en place, afin de garantir une réponse optimale aux différents besoins des utilisateurs. Ce chapitre est donc dédié à cette étape où l'on va d'abord présenter tous les logiciels, outils et langages de programmation qu'on a utilisé pour ensuite enchaîner avec les interfaces de l'application web et les fonctionnalités qu'elles englobent.

4.2 Présentation des logiciels et outils de travail

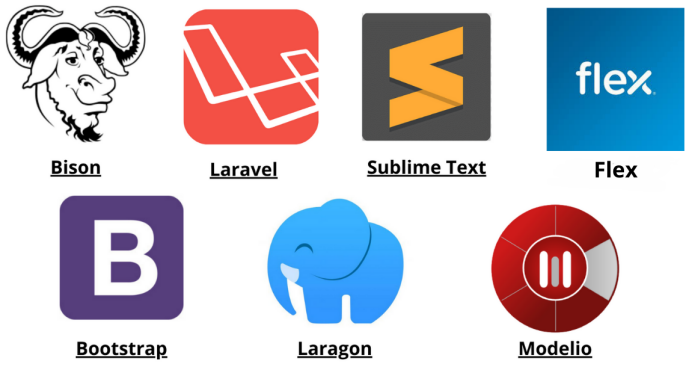


FIGURE 4.1 – Liste des outils et technologies utilisés pour la réalisation

4.2.1 GNU Bison

Bison est un générateur de parseur polyvalent qui transforme une grammaire contextuelle annotée en un parseur LR déterministe ou un parseur LR généralisé (GLR) en utilisant des tables de parseur LALR(1). Bison peut également générer des tables de parseur IELR(1) ou LR(1) canonique.[31]

4.2.2 Flex(Fast Lexical Analyzer Generator)

est un outil ou un programme informatique permettant de générer des analyseurs lexicaux (scanneurs ou lexers. Il est utilisé conjointement avec le générateur de parseur Berkeley Yacc ou le générateur de parseur GNU Bison. Flex et Bison sont tous les deux plus flexibles que Lex et Yacc et produisent un code plus rapide.[32]

4.2.3 Modelio

est un outil de modélisation et de conception logicielle utilisé pour créer des modèles et des diagrammes visuels faits par UML, représentant différents aspects d'un système logiciel.[33]

4.2.4 Laravel

est un framework PHP open-source utilisé pour le développement d'applications web.[34]

4.2.5 Sublime Text

est un éditeur de texte avancé et polyvalent utilisé pour écrire du code source dans différents langages de programmation.[35]

4.2.6 Bootstrap

un framework front-end populaire et open-source qui facilite la création de sites web réactifs et esthétiquement attrayants. Il fournit une collection de composants prédéfinis, de styles CSS et de scripts JavaScript pour accélérer le développement web.Il permet de générer une interface web adaptable pour les PCs,les tablettes et les smartphones en utilisant un code.[36]

4.2.7 Laragon

est un environnement de développement local tout-en-un pour les projets web basés sur PHP. Il offre une installation simple et rapide de PHP, Apache, MySQL et d'autres outils essentiels, permettant aux développeurs de configurer facilement des environnements de développement isolés sur leur propre machine.[37]

4.3 Langages de programmation utilisés

4.3.1 Le langage de script PHP

est un langage de programmation open-source largement utilisé pour le développement web. Il est principalement utilisé pour créer des sites web dynamiques et des applications web. [38]

4.3.2 Le langage de requête SQL

(Structured Query Language) est un langage de programmation spécialement conçu pour la gestion et l'interrogation des bases de données relationnelles. Ce langage informatique est notamment très utilisé par les développeurs web pour communiquer avec les données d'un site web Il permet aux utilisateurs de manipuler et de récupérer des données, de créer, de modifier et de supprimer des tables, ainsi que d'exécuter des opérations avancées telles que les jointures, les agrégations et les transactions.[39]

4.3.3 Javascript

est un langage de programmation polyvalent utilisé principalement pour le développement web. Il permet d'ajouter des fonctionnalités interactives et dynamiques aux sites web, de manipuler le contenu de la page, d'effectuer des animations et d'interagir avec les utilisateurs.[40]

4.3.4 HTML(HyperText Markup Language)

est un langage de balisage utilisé pour structurer et présenter le contenu des pages web. Il permet de définir la structure logique d'une page en utilisant des balises pour décrire les éléments tels que les titres, les paragraphes, les liens, les images, etc.[41]

4.3.5 CSS (Cascading Style Sheets)

est un langage de feuille de style utilisé pour définir l'apparence visuelle et le style des éléments HTML d'une page web. Il permet de contrôler les couleurs, les polices, les marges, les tailles, les mises en page et bien plus encore.[42]

4.3.6 Langage C

Pour programmer avec Flex et Bison, On a utilisé généralement le langage de programmation C. Flex est un générateur de scanners lexical (analyseurs lexicaux) qui est souvent utilisé conjointement avec Bison, un générateur d'analyseurs syntaxiques. Flex génère un code en C qui effectue l'analyse lexicale, c'est-à-dire la reconnaissance des motifs lexicaux dans un flux de caractères. Bison, quant à lui, génère un code en C qui effectue l'analyse syntaxique en utilisant les règles de grammaire spécifiées.

4.4 Architecture du système QueryInspector

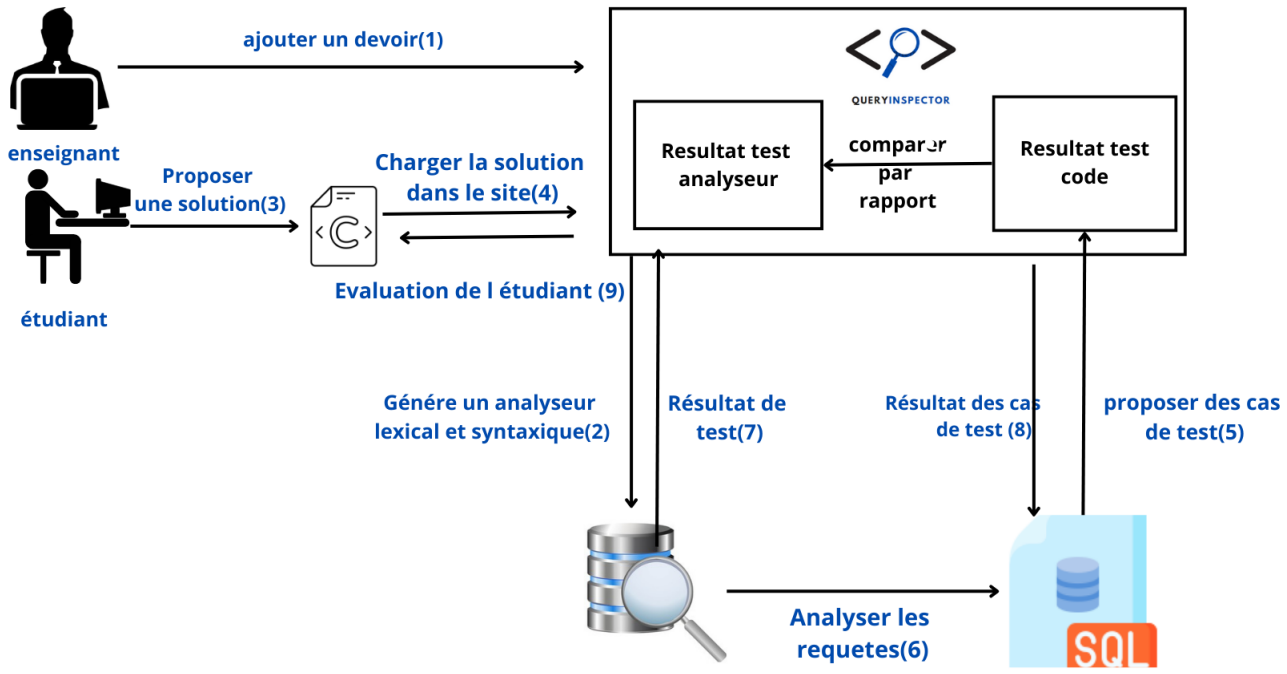


FIGURE 4.2 – Architecture du systeme

Nous avons développé un analyseur lexical et syntaxique en utilisant les outils Flex et Bison. L'objectif de cet analyseur était de vérifier la validité des fichiers en langage C soumis par les étudiants.

En parallèle, Nous avons créé un site web en utilisant le framework Laravel. Ce site permet aux étudiants de soumettre leurs devoirs en chargeant leurs fichiers en langage C.

D'un autre coté on a un analyseur lexical syntaxique de requêtes SQL générer par Flex et Bison qui nous permet de faire la notation du code de l'étudiant une fois que le fichier et soumis en comparant les résultats des deux selon le nombre de tests juste effectués , pour plus de précision supposons que après le teste de l'analyseur on a eu 40 requête juste sur 50 si le code de l'étudiant aura le même résultat sur le meme test que l'analyseur l'étudiant aura une note complète sinon il obtiendra une note selon le pourcentage de test juste sur son code .

Cette combinaison de l'analyseur lexical et syntaxique développé avec Flex et Bison et du site Laravel a permis de créer un environnement complet pour la soumission et l'évaluation automatique des devoirs en langage C.

4.5 Avantages et bénéfices

Gain de temps : La correction automatique permet de réduire le temps nécessaire pour détecter et corriger les erreurs dans le code . Amélioration de la qualité du code : Les suggestions de correction et les conseils aident les utilisateurs à améliorer la qualité de leurs travail. Facilité d'utilisation : L'interface conviviale du site web rend l'utilisation du système intuitive pour les utilisateurs.

4.6 Réalisation d'analyseur

En utilisant Flex et GNU-Bison on peut génerer 2 analyseurs (l'analyseur lexical et l'analyseur syntaxique), en suivant les étapes suivantes :

1. Créez un fichier lexer.l avec un éditeur de texte puis construire notre code d'analyseur lexical (code Flex) .

   Le code est le suivant :

```
%{
#include<stdio.h>
#include "string.h"
#include "stdlib.h"
#include "syn.tab.h"
extern YYSTYPE yylval;
%}
%%
SELECT     { return SELECT; }
FROM       { return FROM; }
WHERE      { return WHERE; }
AND        { return AND; }
OR         { return OR; }
INSERT    { return INSERT; }
INTO      { return INTO; }
VALUES    { return VALUES; }
JOIN      { return JOIN; }
ON        { return ON; }
GROUP     { return GROUP; }
BY        { return BY; }
"*"   { return etoi; }
"("   { return  prO;}
")"   { return  prf;}
"]"   {return brf;}
"["   {return bro;}
">"     { return SUPP;}
"<"     { return INF;}
"=="    { return EGALE;}
"!="    { return NONEGALE;}
">="    { return SUPEGALE;}
"<="    { return INFEGALE;}
","     { return ver;}
";"     { return pv;}
"\"[^\"]+\"" { yylval.string_value = strdup(yytext); return
IDENTIFIER; }
[a-zA-Z_][a-zA-Z0-9_]* { yylval.string_value = strdup(yytext);
return IDENTIFIER;}
[0-9]+   { yylval.int_value = atoi(yytext); return INTEGER; }
[0-9]+\.[0-9]+   { yylval.double_value = atof(yytext);
return DECIMAL; }
'[^']*'   { yylval.string_value = strdup(yytext); return STRING; }
[ \t\n]+  {}
.     {printf("Erreur lexical entite %s \n", yytext);}
```

2. Créez un fichier sql.y avec un éditeur de texte puis construire notre code d'analyseur syntaxique (code Bison) .

```
%{
#include <stdio.h>
#include <stdbool.h>
#include <string.h>
extern FILE* yyin;
%}
%union {
    char*  string_value; // Pour les chaînes de caractères
    int    int_value;      // Pour les entiers
    double double_value; // Pour les décimaux
}
%token SELECT FROM WHERE AND OR INSERT INTO VALUES JOIN ON GROUP BY
prO prf brf bro etoi
%token IDENTIFIER INTEGER DECIMAL STRING ver EGALE SUPP INF NONEGALE
SUPEGALE INFEGALE pv '\n'
%%
sql_stmt : select_stmt
         | insert_stmt
         ;
select_stmt : SELECT select_list FROM table_list join_clause
where_clause group_by_clause pv
              {
      printf("prog syntaxiquement correct");
      YYACCEPT;
              }
              |SELECT select_list FROM table_list join_clause
              where_clause group_by_clause error{
     fprintf(stderr,"Point vergule n'existe pas \n");
                yyerrok;
              }
              |SELECT select_list error{
     fprintf(stderr,"Le mot cle 'FROM' n'existe pas \n");
                yyerrok;
              }
               |SELECT error{
     fprintf(stderr,"'Nom de la table (ou (*)) ' n'existe pas \n");
                yyerrok;
              }
            ;
```

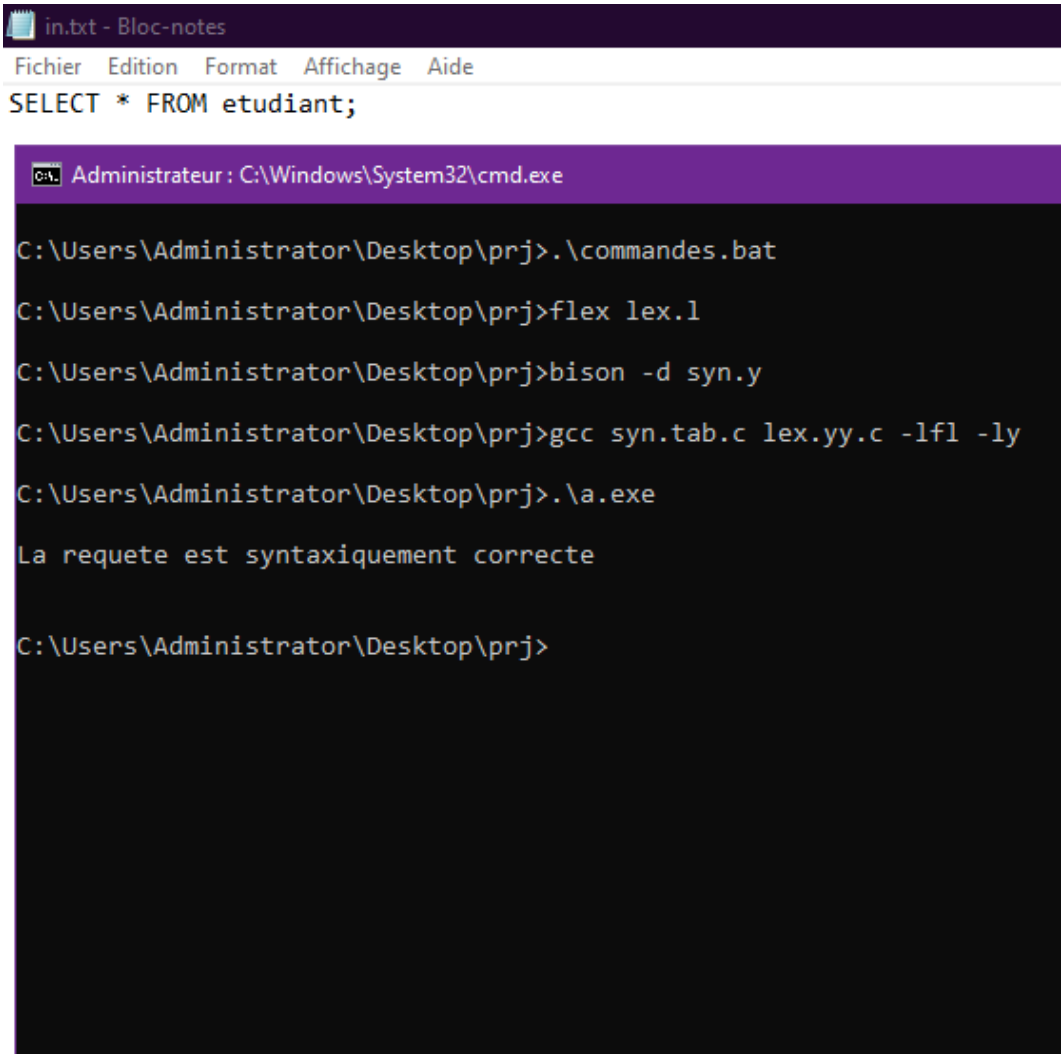
```
insert_stmt : INSERT INTO IDENTIFIER prO column_list prf VALUES prO
value_list prf pv
    { printf("prog syntaxiquement correct");
       YYACCEPT;}
        |INSERT error{
        fprintf(stderr,"Le mot cle 'INTO' n'existe pas ");
        yyerrok;}
        | error INTO{
        fprintf(stderr,"Le mot cle 'INSERT' n'existe pas ");
        yyerrok;};
where_clause :WHERE condition
             | /* empty */
             |condition error{
        fprintf(stderr,"Le mot cle 'WHERE' n'existe pas \n");
        yyerrok;}
             |WHERE error{
    fprintf(stderr,"'La Condition de WHERE' n'existe pas \n");
    yyerrok;};
group_by_clause : GROUP BY column_list
                | /* empty */ ;
join_clause : JOIN table_list ON expr
            | /* empty */ ;
select_list : column_list
            | etoi  /* for selecting all columns */ ;
table_list : IDENTIFIER ver table_list
            | IDENTIFIER ;
column_list : IDENTIFIER ver column_list
            | IDENTIFIER ;
value_list : value ver value_list
           | value ;
value : IDENTIFIER
      | INTEGER
      | DECIMAL
      | STRING
      | expr ;
expr : IDENTIFIER EGALE IDENTIFIER ;
IDF_NBB: IDENTIFIER
         |INTEGER
         |DECIMAL ;
```

```
condition: IDENTIFIER logique IDF_NBB ;
logique:   SUPP
      |INFEGALE
           |SUPEGALE
           |INF
           |EGALE
           |NONEGALE ;
%%
int main ()
{  int yylex();
yyin = fopen("in.txt", "r");
yyparse ();
fclose (yyin);
return 0;  }
int yyerror(const char *s){
        fprintf(stderr,"parser error: %s\n", s);
        return 0; }
```

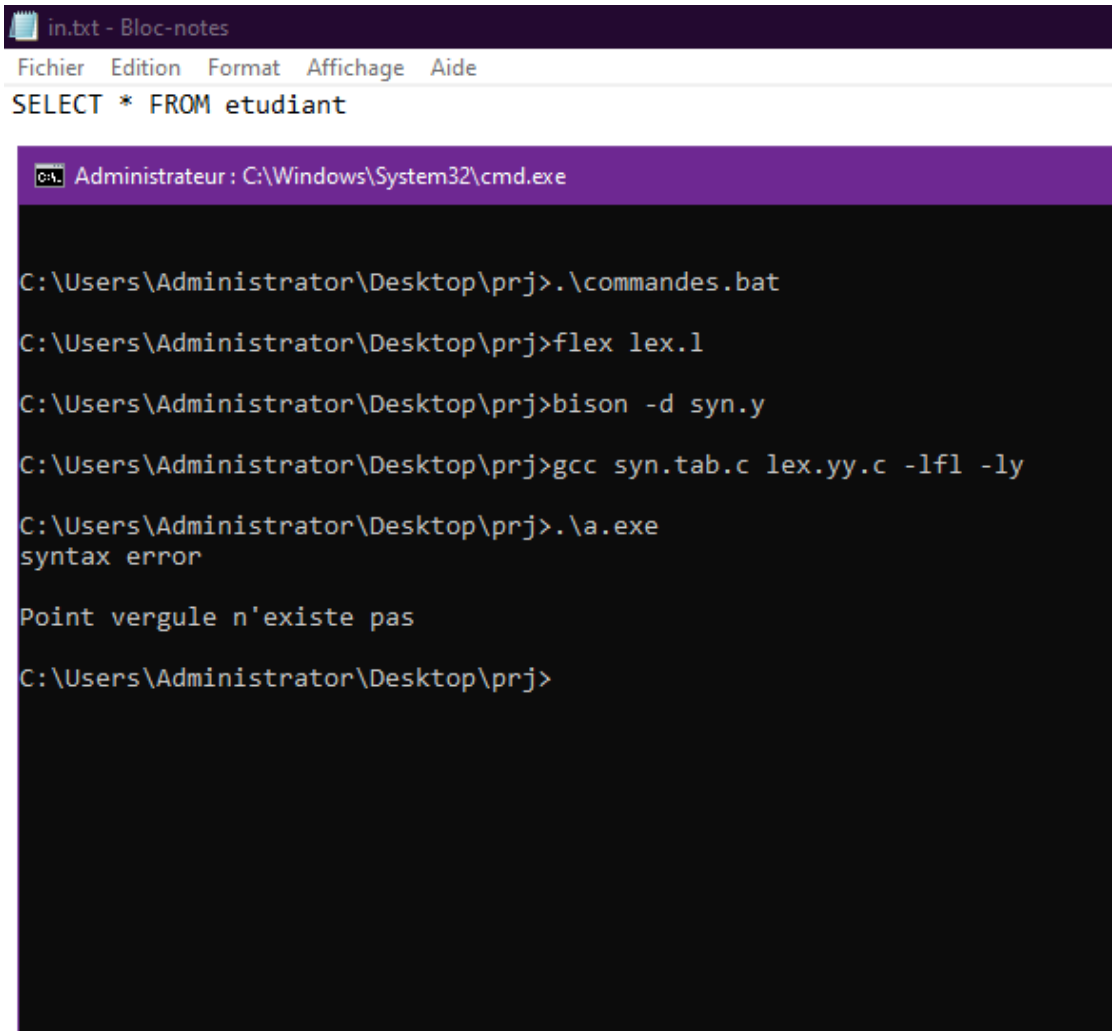
3. Assurez que les deux fichiers (sql.y et lexer.l) sont dans le même répertoire.
4. Ouvrez une fenêtre de terminal et naviguez jusqu'au répertoire contenant les fichiers sql.y et lexer.l.
5. Générez le fichier lex.yy.c en exécutant la commande suivante : flex lexer.l Cela générera le fichier lex.yy.c qui contient le code source de l'analyseur lexical généré par Flex.
6. Générez le fichier y.tab.c et y.tab.h en exécutant la commande suivante : bison -d sql.y Cela générera les fichiers y.tab.c et y.tab.h qui contiennent respectivement le code source et les déclarations générés par Bison pour l'analyseur syntaxique.
7. Compilez les fichiers sources générés (lex.yy.c, y.tab.c) en un exécutable en utilisant la commande suivante : gcc lex.yy.c requetes.tab.c -o analyseur Cela compilera les fichiers sources en utilisant le compilateur GCC et produira un exécutable nommé analyseur.
8. Exécutez l'analyseur en lançant la commande suivante : ./analyseur L'analyseur sera exécuté et attendra des entrées de l'utilisateur pour analyser les requêtes SQL.

4.6.1 Test sur l'analyseur :

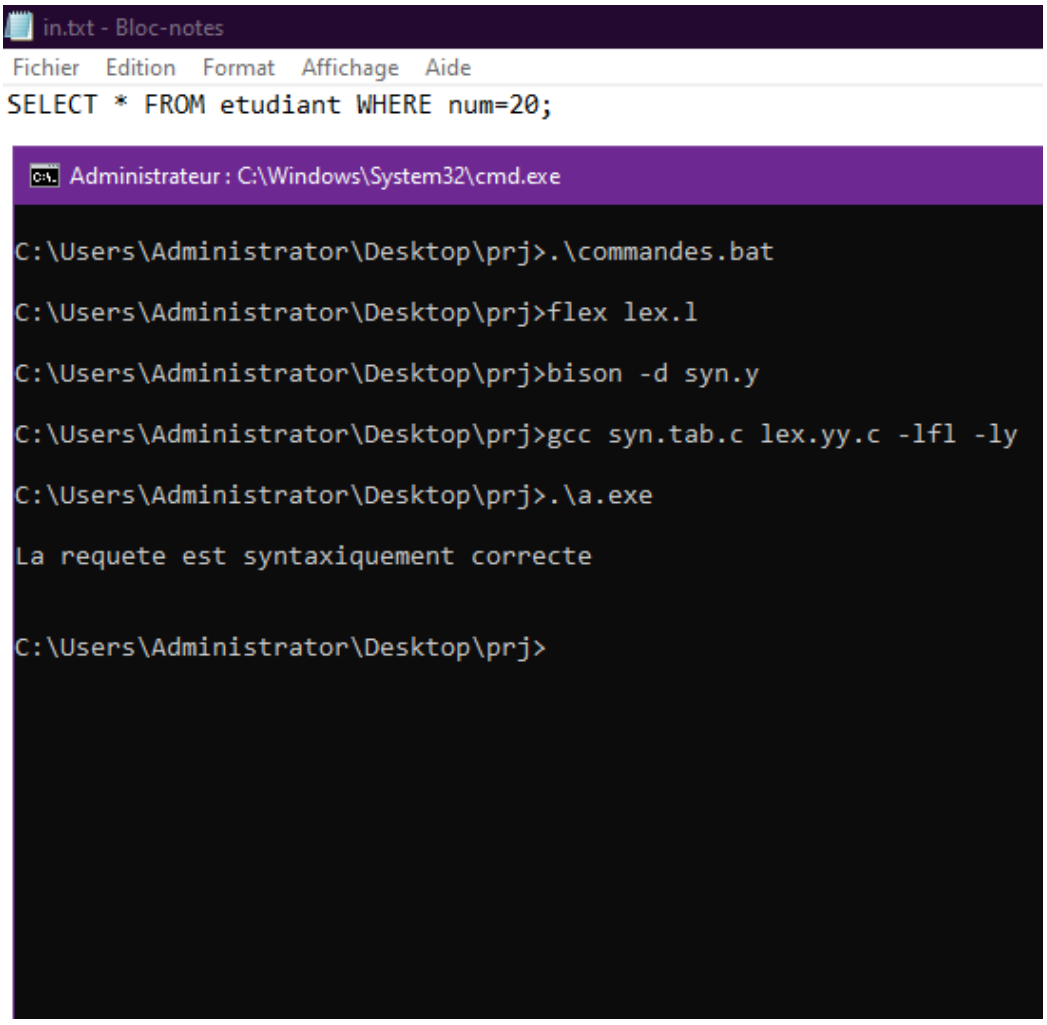
1. requete select correcte : Nous faisons un test sur une simple requete select from qui est ecrite sur le blic-note et le résultat indiqué sur le cmd (la requete est syntaxiquement correct).



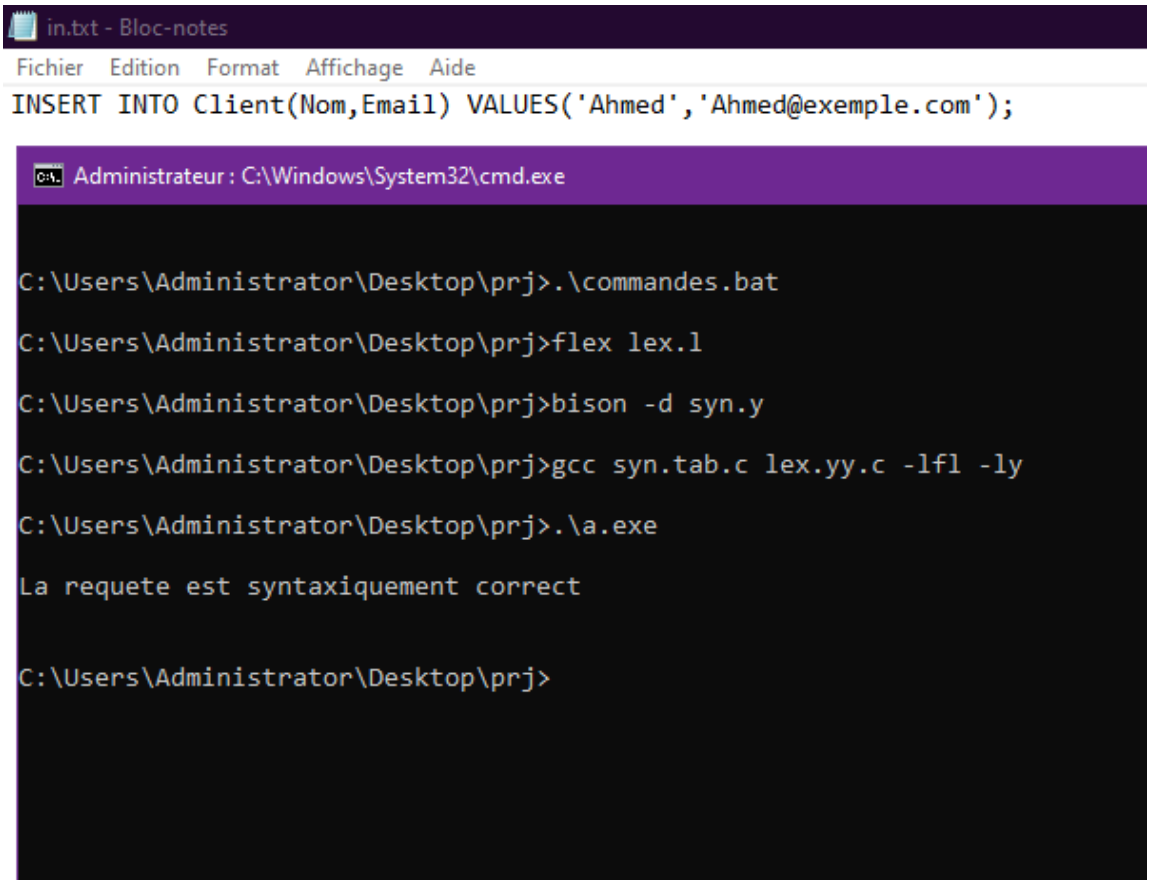
2. requete select incorrecte : Nous faisons un test sur une simple requete select from qui manque le point virgule et le résultat indiqué sur le cmd (syntax error plus le type d'erreur).



3. requete Where correcte : test sur une requete SELECT FROM WHERE correct syntaxiquement.



4. requete insert correcte : test sur une requete INSERT correct syntaxiquement.



5. requete insert incorrecte : test sur une INSERT incorrect qui manque le INTO et le résultat indiqué sur le cmd (syntax error plus le type d'erreur).

in.txt - Bloc-notes

Fichier Edition Format Affichage Aide

```
INSERT Client(Nom,Email) VALUES('Ahmed','Ahmed@exemple.com');
```

Administrateur : C:\Windows\System32\cmd.exe

```
C:\Users\Administrator\Desktop\prj>.\commandes.bat

C:\Users\Administrator\Desktop\prj>flex lex.l

C:\Users\Administrator\Desktop\prj>bison -d syn.y

C:\Users\Administrator\Desktop\prj>gcc syn.tab.c lex.yy.c -lfl -ly

C:\Users\Administrator\Desktop\prj>.\a.exe
syntax error

Le mot cle 'INTO' n'existe pas syntax error
```

4.7 Presentation du site

4.7.1 Coté étudiant

a) Inscrir

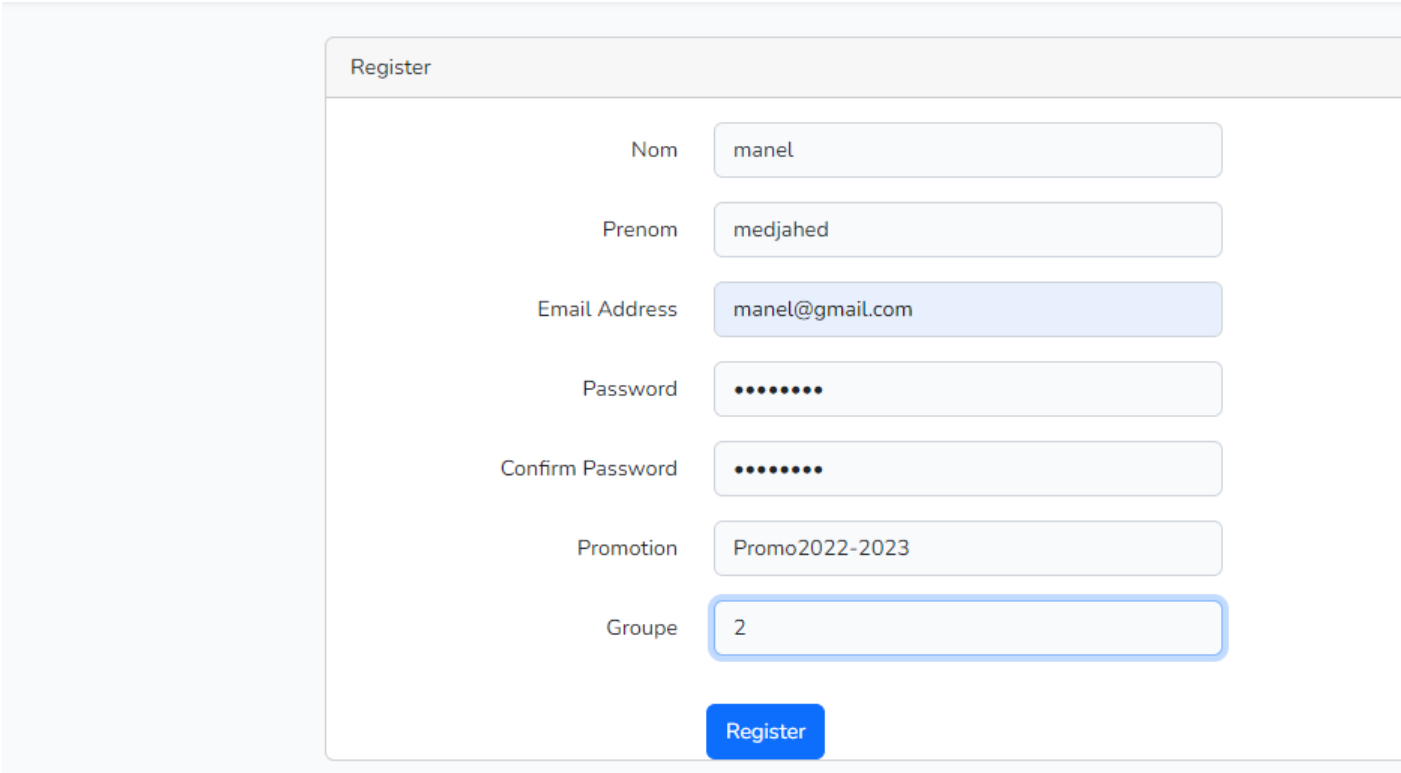


FIGURE 4.3 – Capture d'inscription

b) Se connecté et modifier

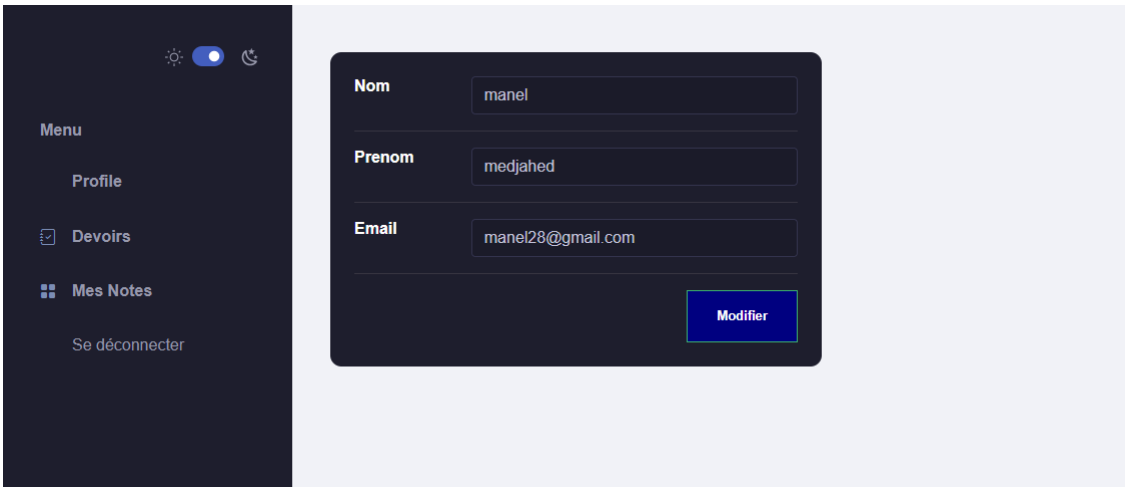


FIGURE 4.4 – Capture connexion et modification

c) Envoyer le devoir

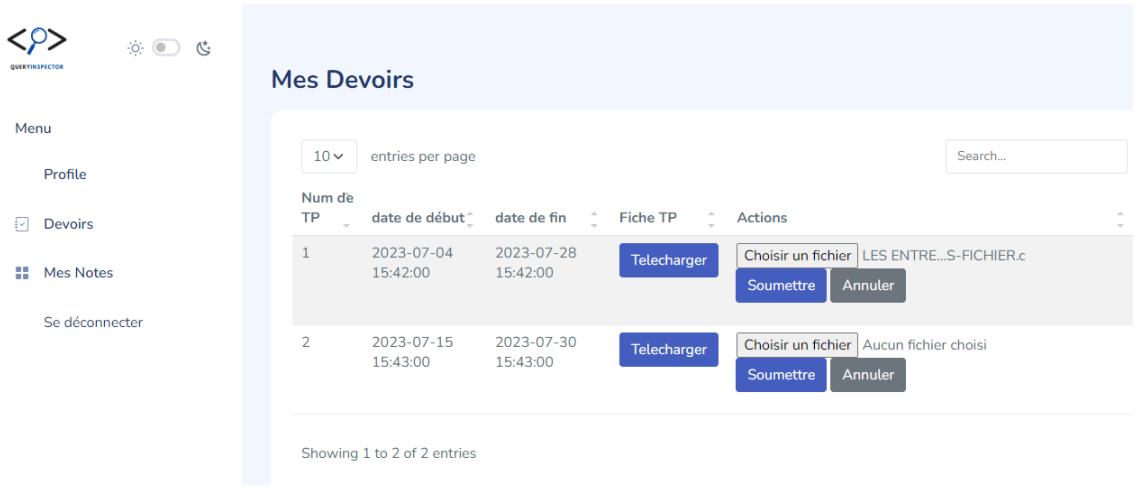


FIGURE 4.5 – Capture envoie du devoir

d) consulter note

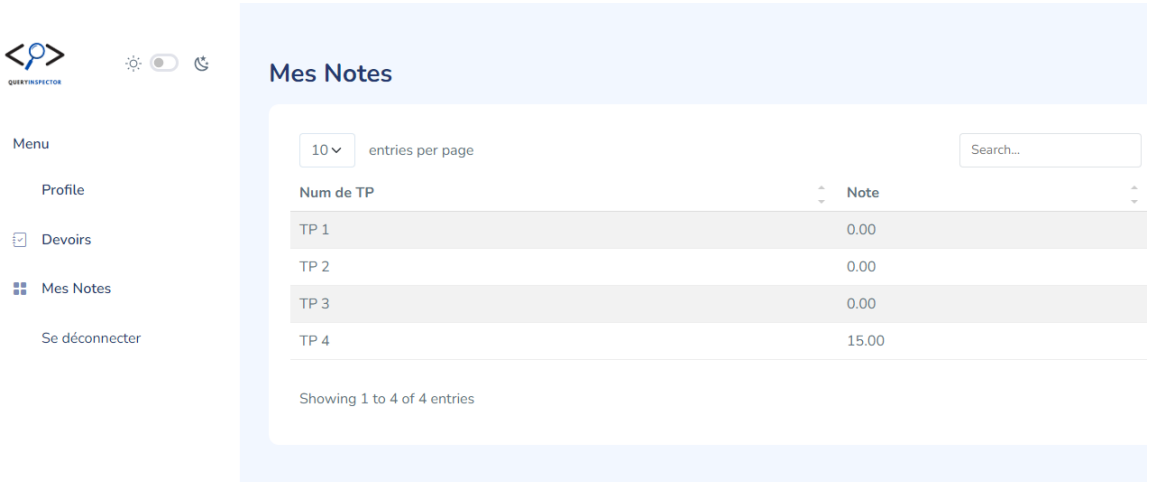


FIGURE 4.6 – capture consultation note

e) Consulter et télécharger le devoir

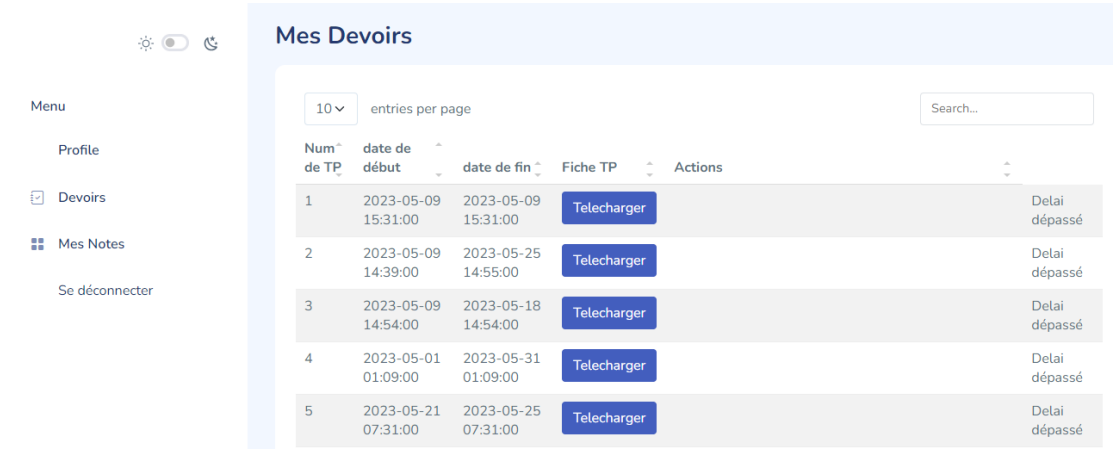


FIGURE 4.7 – Capture consultation et téléchargement du devoir

4.7.2 Coté administrateur

a) Se connecter

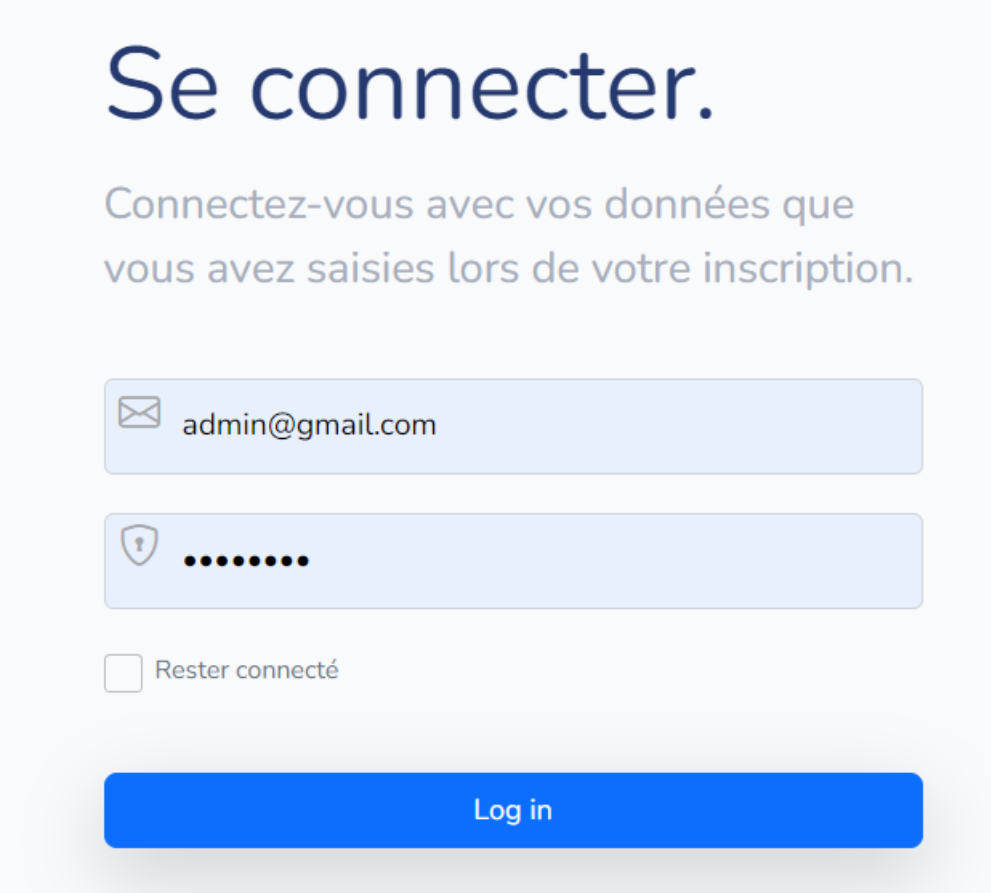


FIGURE 4.8 – Capture connexion

b) Ajouter promotion

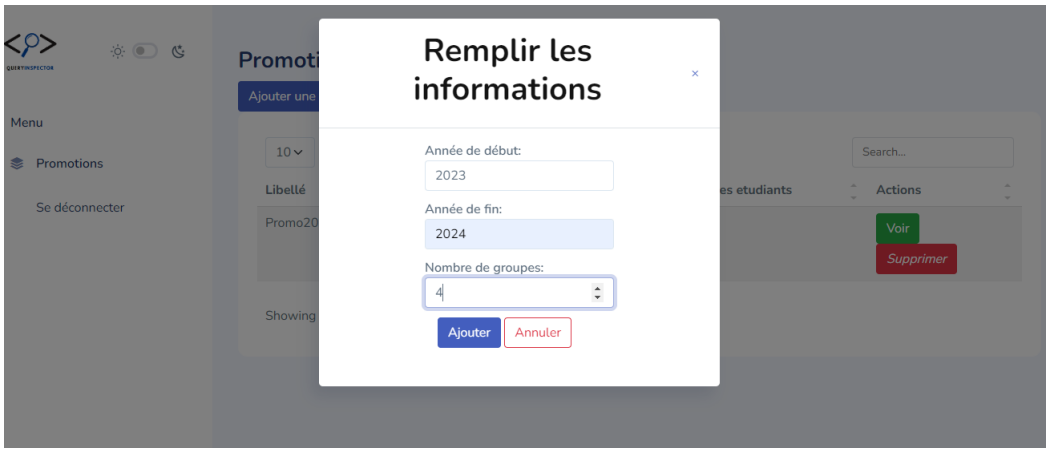


FIGURE 4.9 – Capture ajout promotion

c) supprimer promotion

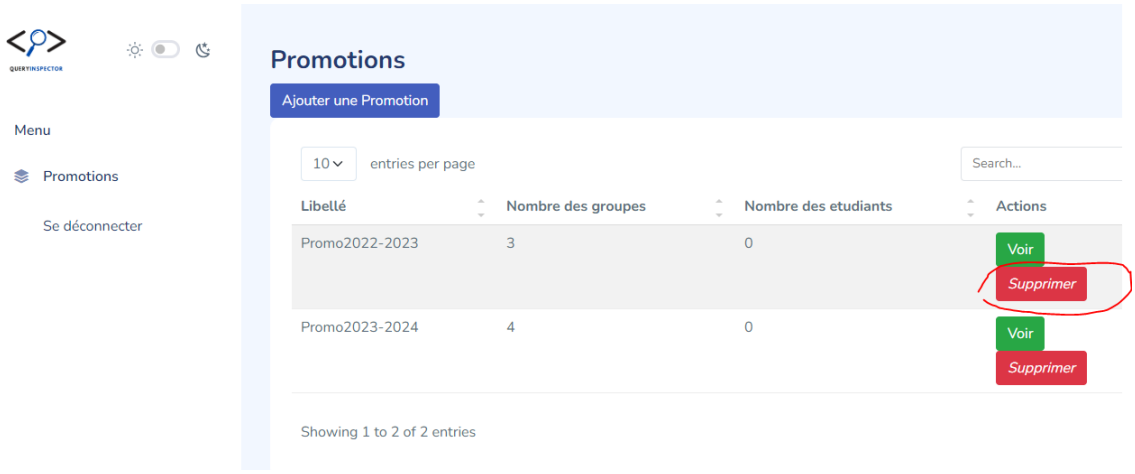


FIGURE 4.10 – capture de suppression de promotion

d) ajouter étudiant

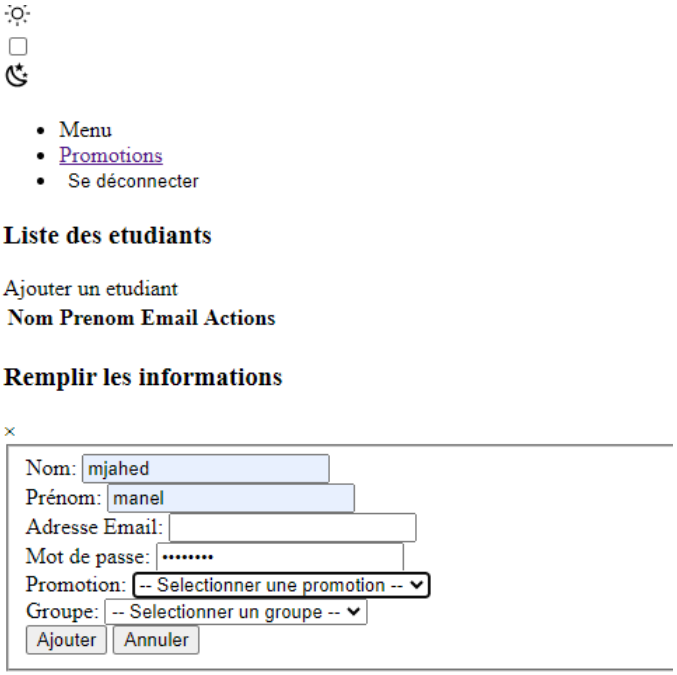


FIGURE 4.11 – Capture d'ajout d'un étudiant

e) supprimer étudiant

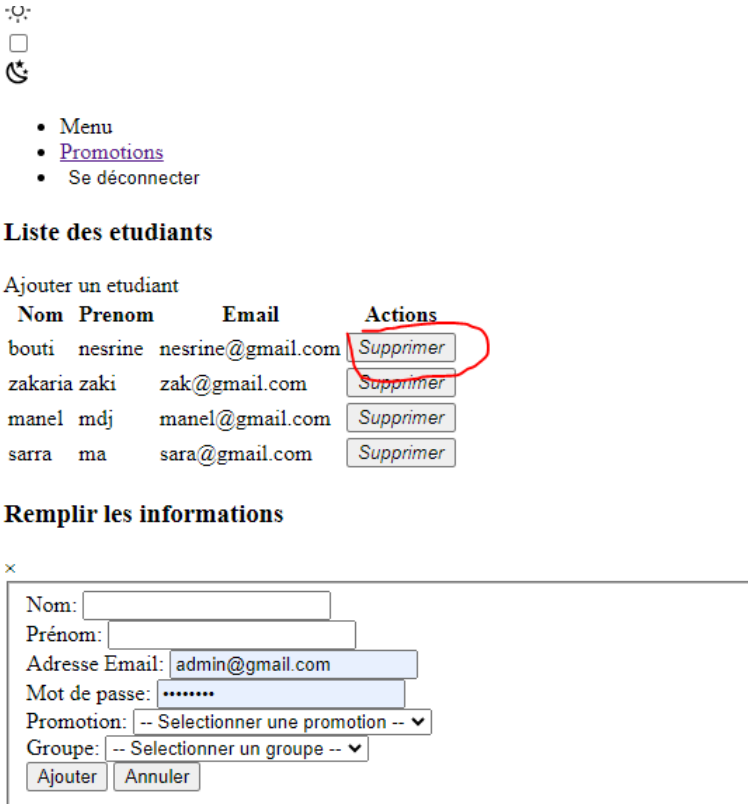


FIGURE 4.12 – Capture de suppression d'un étudiant

4.7.3 Coté enseignant

a) Se connecter

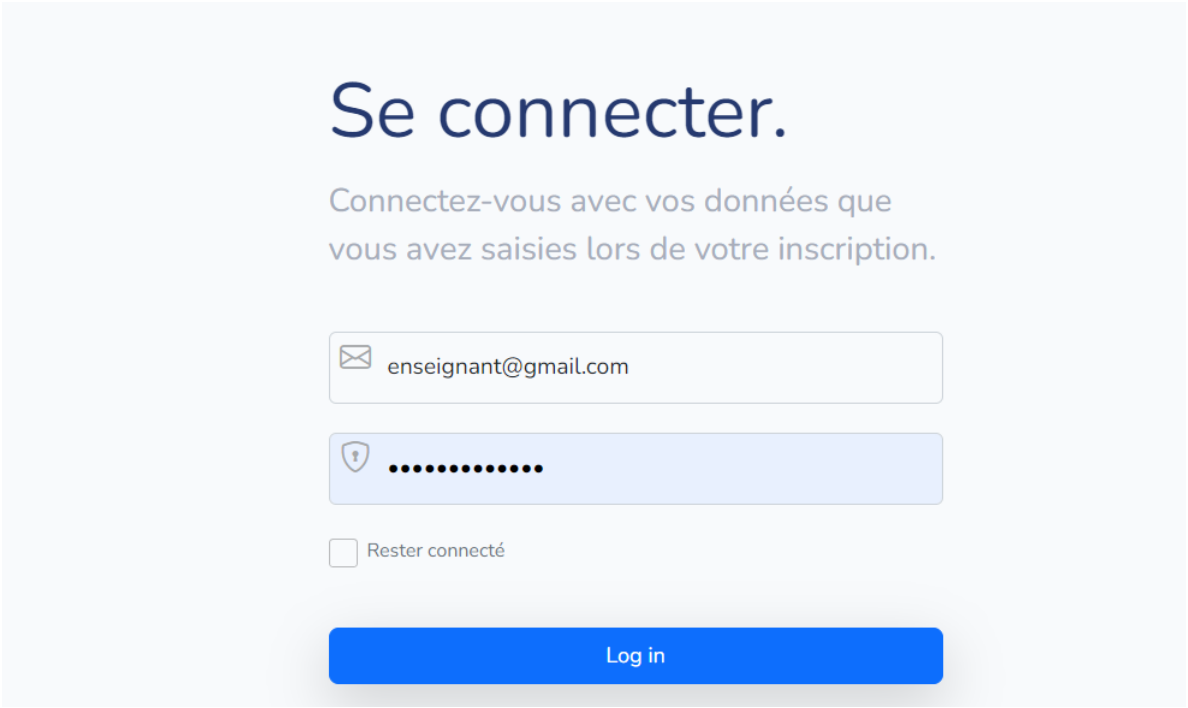


FIGURE 4.13 – Capture connexion

b) Promotion

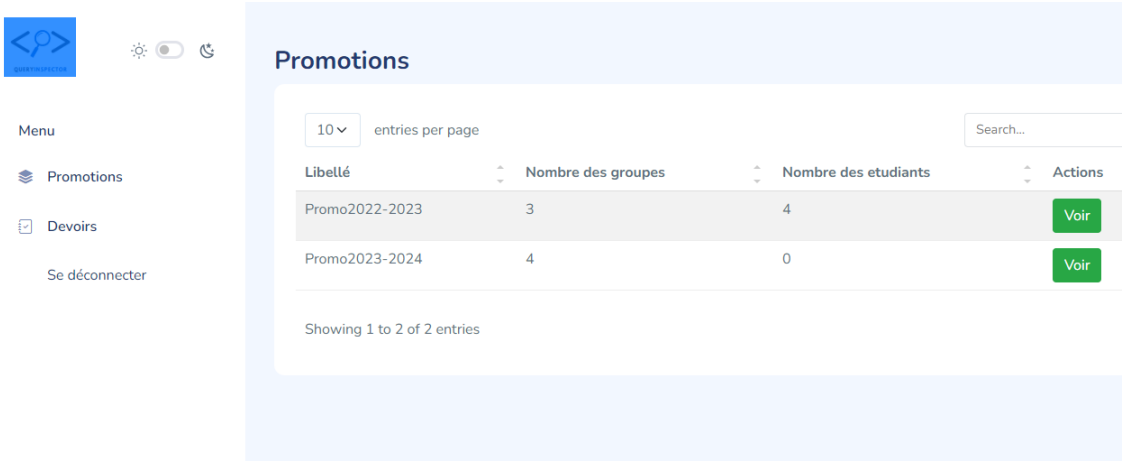


FIGURE 4.14 – Capture promotion

c) Recherché une promotion

FIGURE 4.15 – Capture recherche promotion

d) Consulter liste étudiant

FIGURE 4.16 – Capture consultation liste étudiant

e) Menu devoir

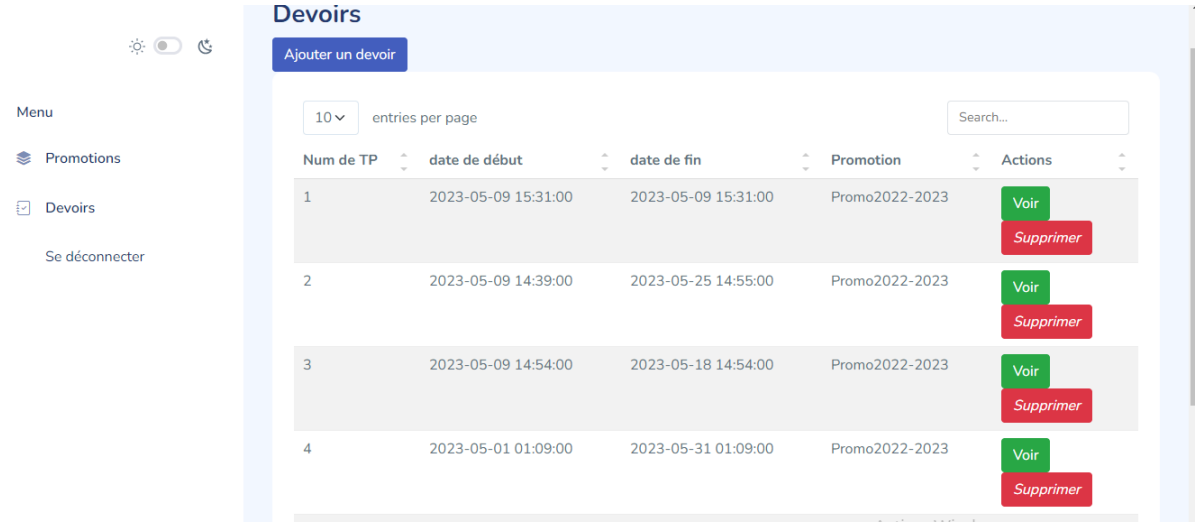


FIGURE 4.17 – Capture menu devoir

f) Ajouter devoir

FIGURE 4.18 – Capture ajout devoir

g) Consulter devoir

- Menu
- Promotions
- Devoirs
- Se déconnecter

Liste des etudiants

ΓP numero : 11

Nom	Prénom	Devoir soumis	Groupe	Note	Actions
zakaria	zaki	non	1	0.00	
nesrine	nesrine	non	2	0.00	
sidou	sidou123	oui	3	0.00	Analyser Voir code
bouti	nesrine	non	2	0.00	
mjahed	manel	non	1	0.00	
bouti	wissam	non	1	0.00	
sarra	boutigff	non	2	0.00	
manel	medjahed	non	1	0.00	

FIGURE 4.19 – Capture consultation devoir

h) Consulter code

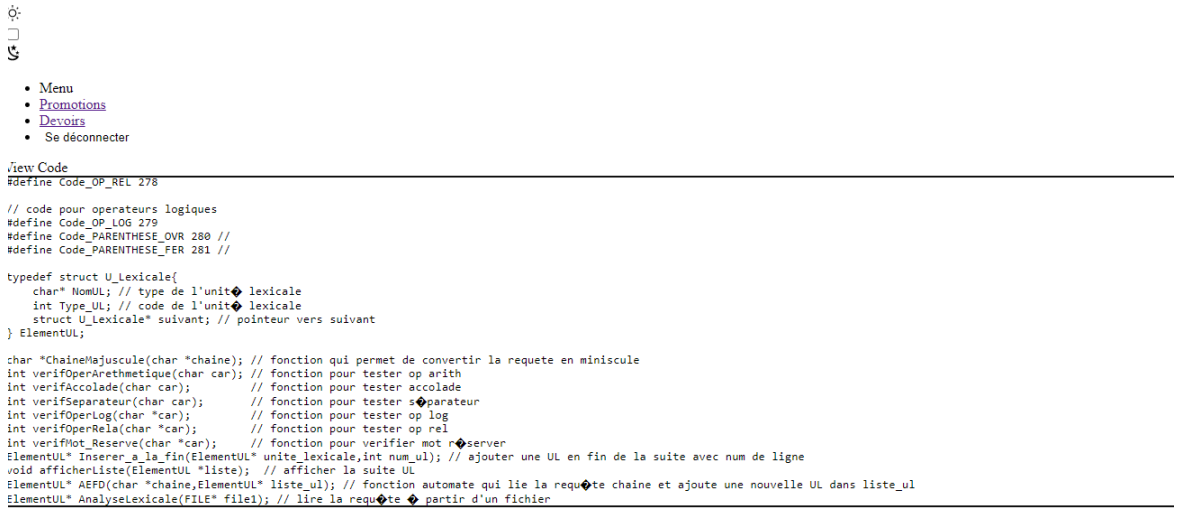


FIGURE 4.20 – Capture consultation code

4.8 Conclusion

Dans ce chapitre, nous avons exposé en détail la phase de réalisation de notre projet, en mettant en avant les diverses technologies utilisées ainsi que l'environnement logiciel et de développement qui ont contribué à sa concrétisation. De plus, nous avons présenté une architecture globale de notre systèmeEnfin, on a exposé toutes les interfaces de notre application web avec leurs fonctionnalités respectives. Pour finir, on va faire une conclusion générale qui englobe toutes les étapes du déroulement de notre travail.

Conclusion Générale

Tout au long de notre mémoire, nous avons mis en avant l'aspect conception et réalisation du projet mais pour y parvenir, nous avons commencé par faire une étude de l'existant dans notre environnement qui est un site de correction automatique.

Afin de satisfaire les besoins , nous avons commencé par utiliser UML2 ainsi que la méthode UP pour la conception des diagrammes de cas d'utilisation, de classe et de séquence système.

Puis, nous avons créé les différentes tables de notre base de données SQL pour pouvoir émettre les requêtes qui vont nous aider à mettre en relation la bdd avec l'application.

Cette dernière baptisée QueryInspector est une application web pour la correction automatique des codes d'analyseur lexicale syntaxique de requêtes SQL des étudiants. Elle a été réalisée grâce aux différents outils cités plus haut et particulièrement le framework PHP open-source avec son environnement Laravel et le générateur d'analyseur lexical Flex et le générateur d'analyseur syntaxique GNU Bison.

Cependant, au cours du développement de notre système, nous avons rencontré un obstacle : l'intégration de notre analyseur dans le site web était problématique en raison des limitations du langage PHP. Parallèlement, nous avons déjà créé un site web qui simplifie la correction des travaux pratiques, même en utilisant l'analyseur de manière indépendante. De plus, cet analyseur présente l'avantage supplémentaire de vérifier la validité des requêtes SQL et peut éventuellement être utilisé dans le module de gestion de base de données.

Pour conclure, ce projet nous a été très bénéfique et fructueux car il nous a permis d'avoir une meilleure maîtrise dans le domaine de la programmation , travailler avec de nouveaux outils et de nouveaux langages ,et de nous initier au domaine du travail. Cependant, il existe des possibilités d'amélioration pour notre travail par exemple de trouver des solution pour l'integration d'analyseur, prendre en charge d'autres langages de programmation populaires pour offrir à nos utilisateurs une plate-forme de développement complète et surtout au niveau de la correction du code d'analyseurs sémantique.

Bibliographie

[1] C. Mohamed, “Evaluateur d’expression de types : arithmétiques, logiques et relationnelles,” 2010,2011.

[2] christine.solnon, “Théoriedeslangages.” https://perso.liris.cnrs.fr/christine.solnon/langages.pdf [Consulté : 30-mars-2023].

[3] H. Garreta, “Techniques et outils pour la compilation,” 2018. Consulté : 02-février-2023.

[4] K. BESSAOUD, “Théoriedeslangages.” http://e-biblio.univmosta.dz/bitstream/handle/123456789/18775/PINF02.pdf?sequence=1 [Consulté : 31-mars-2023].

[5] “Ch.1 automates finis.” https://www-igm.univ-mlv.fr/~desar/Cours/automates/ch1.pdf [Consulté : 31-mars-2023].

[6] “Automates à nombre fini d’états et langages réguliers.” https://perso.univ-rennes1.fr/dimitri.petritis/enseignement/lcm1/lcm1-chap3.pdf year = "2014" [Consulté : 31-mars-2023].

[7] J. Parreaux, “Leçon 923 : Analyse lexicale et analyse syntaxique. applications,” 2018. Consulté : 31-mars-2023.

[8] “arbre de syntaxe abstraite.” https://www.google.com/url?sa=i&url=http%3A%2F%2Fpauillac.inria.fr%2F~levy%2Fx%2Ftc%2Fpolycopie-1.6%2Fmain010.html&psig=AOvVaw2bd5WJiTTZ4Q5rBjWcYHWI&ust=1695652747237000&source=images&cd=vfe&opi=89978449&ved=0CBAQjRxqFwoTCJidyfG8w4EDFQAAAAAdAAAAABAE [Consulté : 24-septembre-2023].

[9] “Qu’est-ce qu’une compilation ? - définition de techopedia - développement 2023. icy science,” 1970. Consulté : 02-février-2023.

[10] “Schéma général d’un compilateur.” https://www.google.com/url?sa=i&url=http%3A%2F%2Fcours.thirion.free.fr%2FCours%2FCommon-Premieres-Notions%2FLa-Programmation.html&psig=AOvVaw1sr85bxu20hI-siPwZALG0&ust=1695653136273000&source=images&cd=vfe&opi=89978449&ved=0CBAQjRxqFwoTCMi3gau-w4EDFQAAAAAdAAAAABAE [Consulté : 24-septembre-2023].

[11] “Applications des langages formels et des automates finis..” http://www.desmontils.net/emiage/Module209EMiage/c7/CCh7.htmhttps://fr.theastrologypage.com/compile year = "2018" [Consulté : 02-février-2023].

[12] “Analyse lexicale..” http://www2.ift.ulaval.ca/~dadub100/cours/A19/3101/slides3.pdf [Consulté : 02-février-2023].

[13] A. Dargham, "Chapitre 6 : Outil d'analyse lexicale : Flex.," Septembre, 2012. Consulté : 05-février-2023.

[14] "Processus de création d'un analyseur lexical avec flex." https://www.google.com/imgres?imgurl=x-raw-image%3A%2F%2F%2F29af18ed139f3542016011ed77cdc2e0286301fa63d573631f988bc55da47b70&tbnid=QjWfE-0YrymeFM&vet=12ahUKEwiBnIuev8OBAxWQmicCHYswAAIQMygCegQIARBT..i&imgrefurl=https%3A%2F%2Fv-assets.cdnsw.com%2Ffs%2Froot%2F98y1s-outil_flex.pdf&docid=whN3ZJuNWoN1eM&w=626&h=157&q=processus%20de%20cr%C3%A9ation%20d%E2%80%99un%20analyseur%20lexical%20avec%20Flex&ved=2ahUKEwiBnIuev8OBAxWQmicCHYswAAIQMygCegQIARBT [Consulté : 24-septembre-2023].

[15] P. Langevin, "Introduction à la compilation via les commandes flex et bison." https://langevin.univ-tln.fr/CDE/LEXYACC/Lex-Yacc1.html year = " 2002, January " [Consulté : 05-février-2023].

[16] M. Meynard, *Interprétation et compilation (HLIN604) - LIRMM*. 14 janvier 2016.

[17] Gridaine, "Présentation flex/bison." Consulté : 09-février-2023.

[18] F. Harrouet, "G en erer un analyseur avec flex bison - enib." Consulté : 11-février-2023.

[19] "Bison - gnu project - free software foundation." https://www.gnu.org/software/bison/ [Consulté : 02-juillet -2023].

[20] "Lex et yacc : Définition et explications. techno.."

[21] "What are flex and bison ?.." http://aquamentus.com/flex_bison.html [Consulté : 02-juillet -2023].

[22] F. Harrouet, *A GUIDE TO LEX YACC*. Consulté : 02-juillet -2023.

[23] E. Schmidt, "Lex a lexical analyzer generator - mbzz.," 2014. Consulté : 02-juillet -2023.

[24] "Logiciels.pro." https://www.logiciels.pro/logiciel-saas/codacy/ [Consulté : 20-février-2023].

[25] "Débogueur en ligne gdb." https://www.onlinegdb.com/about [Consulté : 20-février-2023].

[26] Y.Poiron, "Codacy : Un outil de revue de code automatisé en bêta privée. blognt : le blog des nouvelles technologies.." https://www.blog-nouvelles-technologies.fr/39317/codacy-un-outil-de-revue-de-code-automatise-en-beta-privee/ [Consulté : 20-février-2023].

[27] C. (n.d.), "Codacy pricing, alternatives more 2023." https://www.capterra.com/p/144689/Codacy/ [Consulté : 20-février-2023].

[28] "Onlinegdb avis prix alternatives : Comparateur logiciels.pro. logiciels.pro. (2021, may 31)." https://librecours.net/module/bdd0/intro-sgbd/pres/co/db-fiddle.html?mode=html%5D [Consulté : 20-février-2023].

[29] "Logiciels.pro." https://www.logiciels.pro/logiciel-saas/onlinegdb/ [Consulté : 20-février-2023].

[30] "Processus unifié." https://www.academia.edu/39598956/Processus_Unifi%C3%A9_UP_and_2TUP.

[31] “Bison - gnu project - free software foundation..” https://www.gnu.org/software/bison/, [Consulté : 01-juin-2023].

[32] “Fast lexical analyzer generator.” https://www.geeksforgeeks.org/flex-fast-lexical-analyzer-generator/, year = "2023, April 10", [Consulté : 01-juin-2023].

[33] “Modelio. modelio open source - uml and bpmn free modeling tool.” https://www.modelio.org/, [Consulté : 03-juin-2023].

[34] “The php framework for web artisans. laravel..” https://laravel.com/, [Consulté : 03-juin-2023].

[35] “The sophisticated text editor for code, markup and prose. sublime text..” https://www.sublimetext.com/, [Consulté : 03-juin-2023].

[36] J. T. Mark Otto, “Bootstrap. bootstrap • the most popular html, css, and js library in the world..” https://getbootstrap.com/, [Consulté : 03-juin-2023].

[37] Leokhoa., “Download. laragon..” https://laragon.org/download/index.html, year="2021, May 25", [Consulté : 03-juin-2023].

[38] “Hypertext preprocessor. php..” https://www.php.net/, [Consulté : 05-juin-2023].

[39] “Cours et tutoriels sur le langage sql. sql..” https://sql.sh/, [Consulté : 05-juin-2023].

[40] “Javascript. mdn..” https://developer.mozilla.org/fr/docs/Web/JavaScript, [Consulté : 05-juin-2023].

[41] “Html tutorial. tutorials..” https://www.w3schools.com/html/, [Consulté : 05-juin-2023].

[42] “Css tutorial. tutorials..” https://www.w3schools.com/css/, [Consulté : 05-juin-2023].

Résumé

Notre projet émerge du besoin pressant de simplifier et d'optimiser la correction des codes en langage C, principalement utilisés dans les travaux pratiques du module de compilation. Simultanément, nous avons entrepris l'intégration d'un analyseur lexical et syntaxique pour évaluer les requêtes SQL. Cette initiative a pour but de créer un environnement d'apprentissage plus fluide et adapté à l'évolution constante des besoins pédagogiques. L'objectif central est de faciliter l'interaction entre les enseignants et les étudiants, en automatisant les aspects les plus techniques de l'évaluation, ce qui permet aux éducateurs de se concentrer davantage sur l'accompagnement et le soutien individualisé des apprenants.

Cependant, notre projet découle d'une problématique majeure : la charge de travail considérable qui pèse sur les enseignants. Face à des effectifs d'étudiants toujours plus importants, les assistants pédagogiques ne sont plus en mesure de corriger manuellement toutes les erreurs commises par l'ensemble des étudiants. Pour répondre à ce défi, nous avons entrepris l'automatisation de cette tâche à l'aide d'un site web qui réalise des corrections en utilisant un analyseur de requêtes SQL. Notre solution repose sur des technologies de pointe telles que Laravel, Flex et Bison.

Mot clés : correction automatique des travaux pratiques., analyseur lexical, analyseur syntaxique, compilation des requêtes SQL, Analyse automatique du code source, Flex ,Bison

Abstract

Our project emerges from the pressing need to simplify and optimize the correction of codes in C language, mainly used in the practical work of the compilation module. Simultaneously, we undertook the integration of a lexical and syntactic analyzer to evaluate SQL queries. This initiative aims to create a more fluid learning environment adapted to constantly evolving educational needs. The central objective is to facilitate interaction between teachers and students, by automating the more technical aspects of assessment, allowing educators to focus more on individualized guidance and support for learners.

However, our project arises from a major challenge: the significant workload burdening the teachers. Faced with increasingly large numbers of students, teaching assistants are no longer able to manually correct all errors made by all students. To address this challenge, we embarked on the automation of this task using a website that performs corrections using an SQL query analyzer. Our solution relies on cutting-edge technologies such as Laravel, Flex, and Bison

Keywords: automatic correction of practical work, lexical analyzer, syntactic analyzer, compilation of SQL queries, automatic analysis of source code, Flex, Bison.

ملخص

ينشأ مشروعنا من الحاجة الملحة لتبسيط وتحسين تصحيح البرامج في لغةC ، والتي تستخدم بشكل رئيسي في العمل التطبيقي لوحدة التجميع. وفي الوقت نفسه، قمنا بدمج محلل معجمي ونحوي لتقييم استعمالاتSQ L . تهدف هذه المبادرة إلى خلق بيئة تعليمية أكثر مرونة تتكيف مع الاحتياجات التعليمية المتطورة باستمرار. الهدف الرئيسي هو تسهيل التفاعل بين الاساتذة و الطلاب ، من خلال أتمتة الجوانب الاكثر تقنية للتقييم، مما يسمح الاساتذة بالتركيز بشكل أكبر على التوجيه الفردي والدعم للمتعلمين .ومع ذلك، فإن مشروعنا ينشأ من مشكلة كبيرة: عبء العمل الكبير مما يثقل كاهل الاساتذة. في مواجهة الأعداد المتزايدة باستمرار من ا الطلاب والمساعدين لم يعد المتخصصون التربويون قادرين على تصحيح جميع األلخطاء التي ارتكبها يدوًيا كل طالب. ولمواجهة هذا التحدي، قمنا بأتمتة هذا المهمة باستخدام موقع ويب يقوم بإجراء التصحيحات باستخدام محلل استعالم .SQL يعتمد حلنا على أحدث التقنيات مثل Laravel و Flexو Bison

الكلمات المفتاحية: التصحيح التلقائي للعمل العملي، المحلل المعجمي، المحلل النحوي، تجميع استعالماتSQL ، التحليل التلقائي للكود Flex،.Bison