CodYngame

Borreill Rémy
Chen Léon
Rebour Tom
Nehad Younes
Manresa Matteo

Sommaire

Introduction

Le projet Codyngame était un des choix possibles pour le projet de fin d'année. Ce projet a pour but de renforcer nos compétences en développement logiciel, en travail d'équipe et en gestion de projet. L'objectif principal de ce projet est de concevoir une application graphique permettant aux utilisateurs de résoudre des exercices de programmation en plusieurs langages.

Cette application aura une interface complète comprenant le choix d'exercices et de ses modalités, un éditeur de code avec de la coloration syntaxique, le choix de langages ainsi qu'une console permettant d'afficher les résultats de l'exécution du programme. Le projet fera des appels système vers différents compilateurs et interpréteurs, et se fondera l'utilisation d'une base de données contenant les exercices.

Ce rapport présentera comment nous avons conçu cette application, les problèmes rencontrés, les choix effectués, dans le but d'avoir une vision globale sur notre travail sur le projet CodYngame.

Présentation du Projet

Dans le cadre du projet CodYngame, nous avons développé une application graphique dédiée à la résolution d'exercices de programmation dans plusieurs langages. Cette application permet à l'utilisateur de consulter l'énoncé d'un exercice, de choisir un langage de programmation, d'écrire son code dans un éditeur avec coloration syntaxique et gestion des indentations, puis d'exécuter ce code pour observer le résultat dans une console intégrée et aussi rajouter des exercices. Le projet s'appuie sur la gestion des appels système vers divers compilateurs et interprètes, ainsi que sur une base de données centralisant les exercices proposés. Quant à notre équipe nous sommes cinq, Borreill Rémy, Chen Léon, Rebour Tom, Nehad Younes, Manresa Mattéo tous dans la même classe, en GI2.

Notre choix s'est porté sur le projet CodYngame car il se démarquait par son côté ludique et par la nécessité de comprendre un minimum la structure des autres langages. Par ailleurs, M. Grignon nous a lancé un défi, réussir ce projet, car ce projet est proposé depuis deux ans et personne n'a su atteindre les exigences du projet.

Choix techniques

Pour développer le projet CodYngame notre équipe a codé exclusivement sur linux par sa capacité à faire des appels systèmes (interpréteur et compilateur) de manière simple, et aussi pour la gestion de base de données (MySQL). Le Java nous était imposé par les normes du projet, quant à la partie graphique, JavaFx était la seule solution à notre niveau. Pour la base de données SQL, nous avons utilisé l'application MySQL pouvant héberger un serveur local grâce à un identifiant que l'utilisateur doit créer au préalable et le mettre dans le fichier de configuration. Quant à la partie de l'IDE notre choix s'est porté sur VSCode et Github Desktop, ce dernier nous a permis de lier plus facilement Github à VSCode.

Déroulement

Avant de commencer à coder, la première tâche a été de planifier le diagramme de classe et d'utilisation puis de commencer le projet par l'installation de l'environnement sur Windows. Cependant nous nous sommes rendu compte que Linux était une meilleure solution pour plusieurs raisons. Premièrement, la gestion des appels systèmes. En effet, pour compiler du C on utilise gcc qui est bien plus simple d'accès sur Linux. Deuxièmement, nous avons essayé d'installer MySQL sur Windows mais nous n'avons pas réussi à lancer un serveur MySQL afin de gérer la base de données. Sous recommandation de M.Grignon, nous avons essayé Intellij, mais sans succès. C'est pourquoi nous nous sommes tous dirigés sur VSCode.

Le début du projet est marqué par la modélisation d'un éditeur, cependant, les partiels sont arrivés ce qui nous a contraint de mettre en semi-pause notre avancée sur le projet pour se concentrer sur nos révisions. De retour des partiels, nous nous sommes motivés à reprendre le projet. Cependant, nos connaissances en JavaFX étaient fragiles, ce qui nous a obligé de consolider notre apprentissage sur ce domaine. Ensuite, deux équipes se sont formées pour partager le travail , une qui devait relier la base de données au code avec JDBC et une autre qui devait commencer l'interface graphique en JavaFX. De plus, Maven fait en sorte de lancer sans difficulté le notre projet.

Par ailleurs, un fichier SQL avec des tables a été créé afin de stocker les exercices en parallèle de l'implémentation des compilateurs et interpréteurs pour les différents langages. Ces derniers utilisent des fichiers temporaires qui étaient une mauvaise idée car l'ouverture, la lecture et la suppression n'était pas optimal. C'est ainsi que nous avons décidé de changer cette méthode par un flux de processeur.

Au fur et à mesure de l'avancement du projet, de nouvelles fonctionnalités ont vu le jour avec par exemple l'ajout d'exercice, les différents langages, l'exécution et la vérification du code dans l'éditeur. L'idée d'implémenter un bouton pour supprimer des exercices a été abandonnée car non seulement elle n'était pas dans le cahier des charges, mais surtout car cela nous posait des problèmes pratiques qui ralentissaient la progression du projet.

Durant une des réunions avec M.Grignon, il nous a fait comprendre que le design était à revoir. C'est pourquoi nous nous sommes dirigés vers un style centré sur l'aspect jeux vidéo et, malgré de nombreux conflits au sein de notre groupe concernant la police d'écriture, le résultat final plaisait à tout le monde.

Au cours d'une réunion, notre tuteur nous a fait remarquer ensemble que notre méthode de vérification de réponse pour stdin/stdout n'était pas bon, nous vérifions si une réponse renvoyée par l'utilisateur était la même que la nôtre, donc impossible de faire des questions du style, "faire un code qui renvoie un nombre premier", c'est pour cela que le système de correction a été changé pour un système de fonction qui permet de voir si, avec une valeur donnée pour les deux codes, correction et code utilisateur, la réponse était la même.

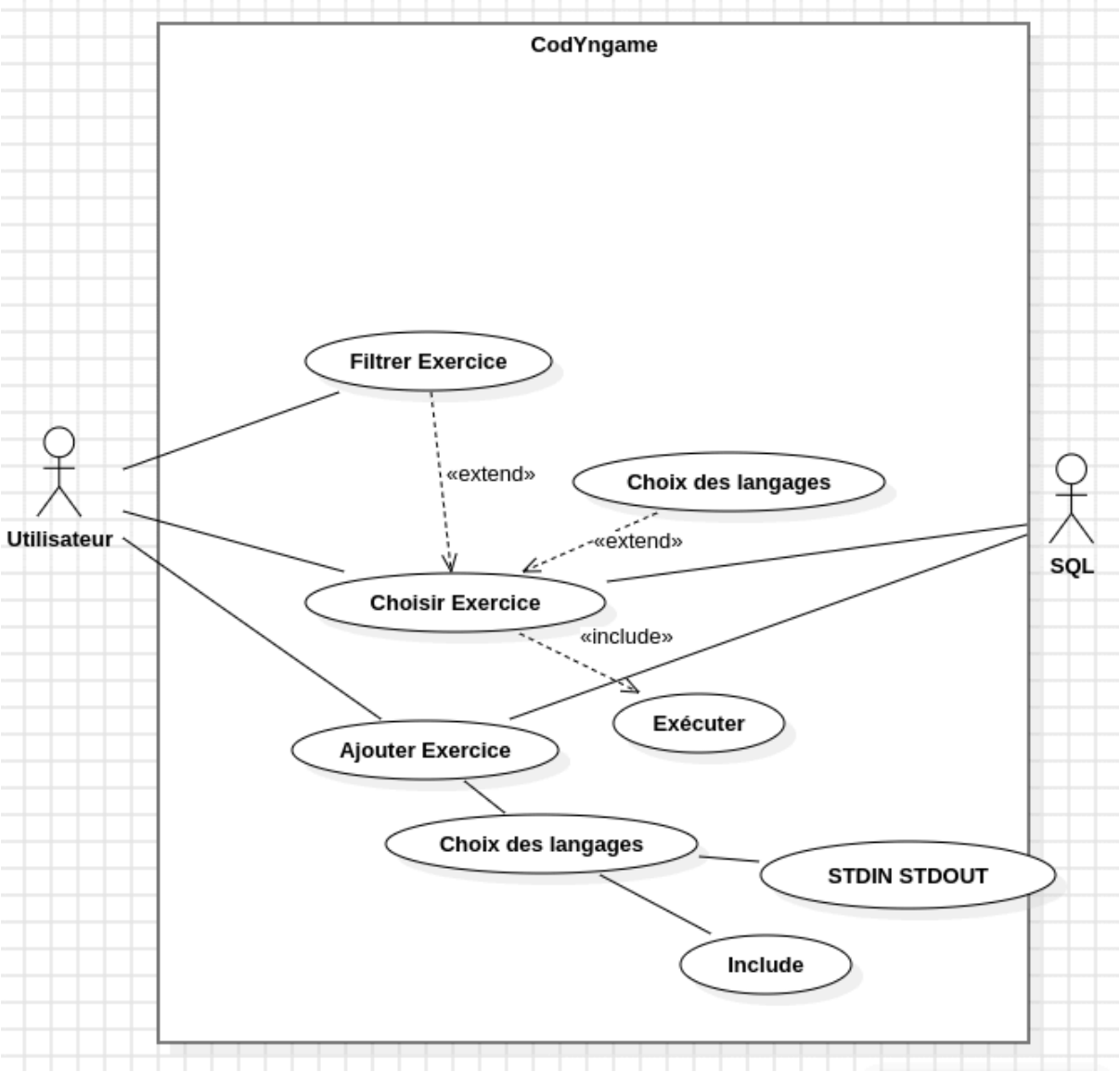
Après avoir fait toutes les fonctionnalités de Stdin/Stdout nous nous sommes concentré sur la partie Include, le plus grand problème a été de comprendre, pour les langages compilés, comment compiler deux fichiers et comment faire en sorte d'utiliser une fonction d'un fichier dans l'autre.

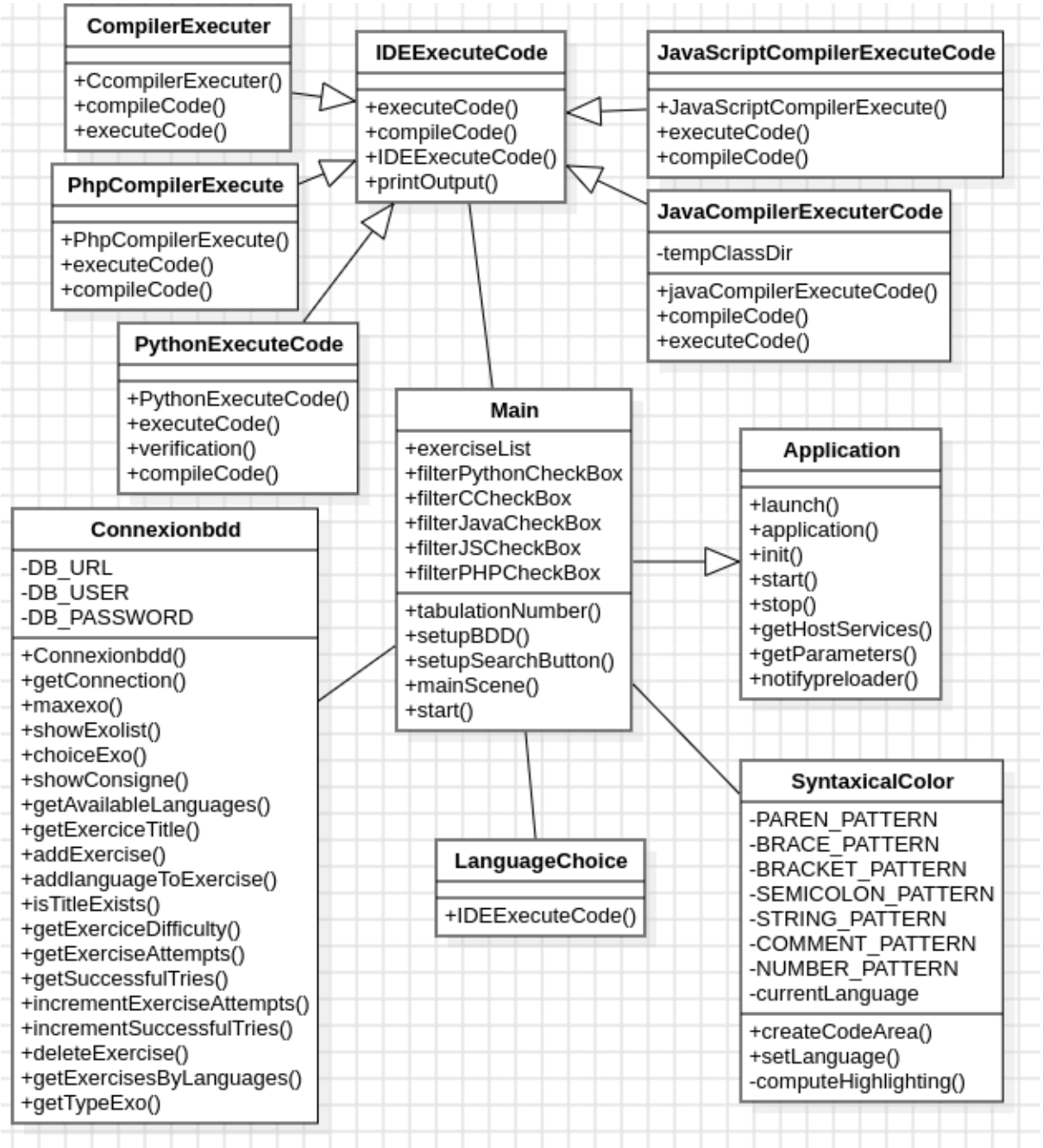
Enfin, nous avons complété le diagramme d'utilisation et de classe grâce à l'application Star UML.

Limitations techniques

Malgré les nombreuses fonctionnalités que nous avons intégrées à l’application, certaines fonctionnalités n'ont pas pu être implémentées dans le temps imparti. Nous n’avons pas réussi à implémenter la fonctionnalité permettant d’indenter plusieurs lignes simultanément via la tabulation dans l’éditeur. De plus, notre Main.java est assez désorganisé. Faute de temps nous n’avons pas pu le structurer correctement ou répartir son contenu dans d’autre classes, ce qui nuit à lisibilité du code

Diagrammes


CodYngame
Filtrer Exercice
«extend»
Choix des langages
«extend»
Utilisateur
SQL
Choisir Exercice
«include»
Exécuter
Ajouter Exercice
Choix des langages
STDIN STDOUT
Include


CompilerExecuter
+CcompilerExecuter()
+compileCode()
+executeCode()
IDEExecuteCode
+executeCode()
+compileCode()
+IDEExecuteCode()
+printOutput()
JavaScriptCompilerExecuteCode
+JavaScriptCompilerExecute()
+executeCode()
+compileCode()
PhpCompilerExecute
+PhpCompilerExecute()
+executeCode()
+compileCode()
JavaCompilerExecuterCode
-tempClassDir
+javaCompilerExecuteCode()
+compileCode()
+executeCode()
PythonExecuteCode
+PythonExecuteCode()
+executeCode()
+verification()
+compileCode()
Main
+exerciseList
+filterPythonCheckBox
+filterCCheckBox
+filterJavaCheckBox
+filterJSCheckBox
+filterPHPCheckBox
+tabulationNumber()
+setupBDD()
+setupSearchButton()
+mainScene()
+start()
Application
+launch()
+application()
+init()
+start()
+stop()
+getHostServices()
+getParameters()
+notifypreloader()
Connexionbdd
-DB_URL
-DB_USER
-DB_PASSWORD
+Connexionbdd()
+getConnection()
+maxexo()
+showExolist()
+choiceExo()
+showConsigne()
+getAvailableLanguages()
+getExerciceTitle()
+addExercise()
+addlanguageToExercise()
+isTitleExists()
+getExerciceDifficulty()
+getExerciseAttempts()
+getSuccessfulTries()
+incrementExerciseAttempts()
+incrementSuccessfulTries()
+deleteExercise()
+getExercisesByLanguages()
+getTypeExo()
LanguageChoice
+IDEExecuteCode()
SyntaxicalColor
-PAREN_PATTERN
-BRACE_PATTERN
-BRACKET_PATTERN
-SEMICOLON_PATTERN
-STRING_PATTERN
-COMMENT_PATTERN
-NUMBER_PATTERN
-currentLanguage
+createCodeArea()
+setLanguage()
-computeHighlighting()

Conclusion

Ce projet nous a permis de développer une application fonctionnelle répondant aux attentes du cahier des charges. Malgré des défis techniques nous avons appris à utiliser JavaFX et exécuter du code avec des appels système Linux.

Le projet CodYngame nous à permis d'énormément progresser en Java et en résolution de problèmes. Nous sommes fières du résultat obtenu et du chemin parcouru pour en arriver ici.