UNIVERSITÉ CÔTE D'AZUR

IUT NICE CÔTE D'AZUR

Méthodologie orientée métier

HemaTraining

Membres de l'équipe

AMADI **Hachim**

ANDRE--AUGUSTINE **Llona**

MARIN **Ryan**

PASQUIER **Winona**

TEROL **Eloi**

VIANO **Anthony**

–

Encadrants

REY **Gaëtan** & DARTIGUES-PALLEZ **Christel**

HemaTraining

Bilan des fonctionnalités

À l'heure actuelle, notre logiciel peut lire un fichier de type « classique » ou « chorégraphie » qui contient des mouvements ou des actions. De plus, il peut vérifier si le fichier comporte des erreurs. Si le fichier est correct, l'utilisateur pourra rentrer les différents paramètres de son exercice.

Notre instructeur va lire les noms des différents mouvements ou actions avec un certain intervalle de temps durant la réalisation de l'exercice. L'utilisateur pourra mettre en pause, relancer ou quitter l'exercice. Mais contrairement à notre dernière version où l'utilisateur devait cliquer sur les différentes icônes pour effectuer les actions précédentes, il pourra désormais aussi le faire à voix haute.

Notre logiciel possède également une page pour les paramètres où il est possible de modifier le son et la langue du logiciel.

Licences

Pour les licences sur les bibliothèques, la majorité d'entre elles dispose d'une licence BSD (Berkeley Software Distribution) qui nous permet de réutiliser l'entièreté du logiciel, ainsi que la GNU Library General Public License qui garantit que nous pouvons le partager.
Concernant les outils utilisés, Eclipse utilise la licence Eclipse Public License, et GitLab utilise The GitLab Enterprise Edition (EE) licence.

Bibliothèques et outils

Pour le développement du logiciel, nous avons utilisé l'outil d'Oracle, l'IDE Eclipse. Nous utilisons la version 11 de Java et par conséquent le JRE 11. Notre projet a été développé sur un projet Maven, ce qui nous a permis de séparer les tests unitaires des ressources, mais aussi de gérer les dépendances des bibliothèques. L'outil EGit (installé sur Eclipse via l'espace « Marketplace ») nous a permis d'utiliser les fonctionnalités de git avec Eclipse, et ainsi de sauvegarder les versions de notre logiciel et de pouvoir mettre en communs nos différents codes.

Concernant les bibliothèques que nous utilisons sur le projet, JavaFX nous a permis de réaliser l'interface graphique du logiciel en Java.
De plus, nous avons dû ajouter des bibliothèques pour que notre instructeur soit capable de parler et de reconnaître une dictée vocale d'un utilisateur. Nous en avons sélectionné 5 et nous les avons comparés selon leurs caractéristiques.

Premièrement, certaines bibliothèques nécessitaient d'être connectées à un réseau wifi, comme c'est le cas avec l'API Google ou l'API IBM. Cela aurait été contraignant pour l'utilisateur car s'il n'est pas connecté à Internet, il n'aurait pas pu utiliser notre instructeur sur le logiciel.

Pour les bibliothèques pour la synthèse vocale et la reconnaissance vocale de l'instructeur, il y en a 3 qui nous avaient principalement intéressées :

1. FreeTTS, qui nous permettait de lire un texte en anglais.
2. Espeak, qui est un lecteur de voix disponible dans plusieurs langues
3. SI_VOX, qui nous servait uniquement pour la synthèse vocale en français.

Nous avons fini par remplacer ces bibliothèques par Azure de Microsoft, car sur les anciennes bibliothèques, il était plutôt compliqué d'effectuer des modifications sur les fichiers qui définissent la voix de l'instructeur, son intonation, et sa voix était moins compréhensible pour les utilisateurs. De plus, nous disposons de crédits sur Microsoft Azure et d'accès étudiant.

Aussi, les voix fouri permettra une meilleure compréhension de l'instructeur pour les utilisateurs de notre logiciel et un plus large panel de choix pour les langues.

Pour la reconnaissance vocale, nous utilisions dans notre version de projet précédente Sphinx 4 qui permettait de reconnaître les mots prononcés par l'utilisateur. Nous avons décidé de remplacer cette bibliothèque par la bibliothèque Azure de Microsoft. En effet, avec notre précédente bibliothèque, il était nécessaire de créer un dictionnaire pour reconnaître convenablement les mots prononcés par l'instructeur. Il était également obligatoire d'enregistrer les utilisateurs à reconnaître pendant au moins une heure pour une reconnaissance optimale. Avec cette nouvelle librairie, il est beaucoup plus simple de détecter ce que dit l'utilisateur. En effet, la reconnaissance se fait immédiatement et elle est plutôt précise.

Pour la vérification de notre fichier nous avons utilisé la bibliothèque JavaCC qui permet d'analyser un fichier de grammaire et de le convertir en programme java qui va regarder si les règle sont respectées.

Modèle d'écriture de code

Langue employée

Le code est développé en anglais hormis les commentaires et la Javadoc.

Indentation

Les accolades des instructions suivront le modèle de la variante OTBS du style K&R.
L'indentation sera de 4 espaces (Une tabulation)

Casse et nommage

Les classes, méthode, variable et constante suivront la convention de nommage de Java

Variables, méthodes	camelCase
Classes	PascalCase
Constantes, variables d'énumération	SCREAMING_SNAKE_CASE

Réduction du nombre de lignes

Les méthodes ne comportant qu'une ligne (relativement courte) comme les getters et setters pourront s'écrire sur une seule est même ligne.

Conventions de nommage JavaFX

Les composants JavaFX devront tous être nommés avec un préfixe de deux ou trois caractères correspondant à son type. Voici la liste des préfixes conventionnels de l'équipe :

Nœuds						Conteneurs	
acc	Accordion	btn	Button	chk	CheckBox	vb	VBox
cho	ChoiceBox	clr	ColorPicker	cmb	ComboBox	hb	HBox
dat	DatePicker	ico	DesktopIcon	edt	EditorPane	tp	TilePane
fch	FileChooser	ifr	InternalFrame	img	Image	fp	FlowPane
imv	ImageView	lbl	Label	lyp	LayeredPane	gp	Group
lst	ListView	mnu	Menu	mnb	MenuBar	sp	StackPane
opt	OptionBar	psw	PasswordField	pnl	Panel	bp	BorderPane
pmn	PopupMenu	prg	ProgressBar	rad	RadioButton	ap	AnchorPane
rot	RootPane	scb	ScrollPane	spr	Separator	rg	Region
sld	Slider	spn	Spinner	smb	SplitMenuButton	pn	Pane
spl	SplitPane	tab	TabPane	tbl	TablePane	tf	TextFlow
tbh	TableHeader	txt	Text	txa	TextArea		
txf	TextField	tlp	TittledPane	tgl	ToggleButton		
tgg	ToggleGroup	tlb	ToolBar	trt	TreeTableView		
trv	TreeView	vpr	ViewPort				

Organisation et fonctionnement

Nom	Rôle pour troisième étape
Llona	En tant que chef de projet je me suis occupé de l'organisation, j'ai planifié les étapes et les tâches de la phase 3. Je me suis occupé des sauvegardes de l'application. Lors de la semaine consacrée aux projets j'ai assisté mes camarades, fixé les bugs et réalisée le design de l'application.
Hachim	Je me suis occupé de la recherche sur la solution à utiliser pour la gestion de la vérification de la grammaire du fichier et j'ai appris à utiliser la notation EBNF(extended BNF) pour ensuite utiliser la librairie JavaCC avec laquelle j'ai créé les 2 fichiers de grammaire qui s'assureront que le fichier rentrer par l'utilisateur étais bien conforme.
Winona	Pendant cette étape, je me suis occupée de la reconnaissance vocale de l'utilisateur à l'aide de la bibliothèque Microsoft Azure. J'ai aussi aidé en ce qui concerne la synthèse vocale.
Eloi	Durant cette étape, je me suis occupé de la synthèse vocale de l'instructeur avec Microsoft Azure, et aidé aussi pour la reconnaissance vocale.
Ryan	Je me suis chargé de la rédaction des aides, de la programmation des interfaces d'aide et de paramètres ainsi que de la revue des fichiers d'internationalisation. J'ai également réalisé un logo pour l'application.
Anthony	Durant cette étape, j'ai continué d'avancer sur la méthode de construction des exercices. Je me suis occupé des méthodes de la classe Session. Je me suis également occupé de la méthode pour créer des exercices de chorégraphie. Et enfin, je me suis occupé des tests unitaires de la partie "core"

Pour cette étape, nous avons défini le rôle de chacun en effectuant une réunion. Chaque membre du groupe a continué le travail commencé à l'étape précédente. Nous tenions également des réunions chaque semaine dans lesquels de petits objectifs nous étaient fixés, mais aussi pour savoir si chaque membre du groupe avait réussie à tenir la deadline qui lui était fixé.
Quand un membre du groupe rencontrait une difficulté particulière, la personne devait s'adresser au chef de projet ou aux autres membres, et si la réponse n'était toujours pas trouvée on décidait de créer un message pour le forum.

Nous avons également effectué des tests unitaires pour la partie du corps fonctionnel de notre projet et nous avons travaillé dans différentes branches pour les différentes parties de notre projet.

Stratégie d'intégration

Tout au long du projet, nous avons travaillé avec Git.

Dans un premier temps, nous avons séparé le projet Git en 3 branches afin de travailler séparément 3 grandes parties du logiciel :

- La programmation du cœur fonctionnel
- Le développement de l'IHM (sans lien avec le cœur fonctionnel, seulement au niveau de l'affichage et de quelques évènements)
- L'implantation de la synthèse vocale

Nous avons également séparé l'équipe de projet en 3 binômes afin de leur affecter les travaux sur les différentes branches.

Sur chacune des branches, les binômes ont veillé à toujours travailler sur les dernières versions des codes et à pousser vers git toutes les modifications qu'ils y apportaient. Nous avons également pris soin de régler les conflits dès qu'ils apparaissaient.

Une fois que les binômes avaient réalisé leurs travaux respectifs sur les différentes branches, nous avons réuni l'ensemble des codes en une seule branche. Les binômes ont alors travaillé sur la même version de git, leur permettant de relier l'IHM, le cœur fonctionnel et l'utilisation de la synthèse vocale.

Stratégie de documentation du code

Nous avons utilisé l'outil Javadoc pour l'intégralité de la documentation de nos codes.

Ainsi, nous avons commenté tous les éléments importants (opérations, classes, attributs, packages) en suivant le format de commentaires Javadoc.

Les commentaires du code sont tous en français.

Plan de tests

JUnit est utilisé pour les tests unitaires. Conforme à la structure Maven, les tests sont séparés dans un package test au même niveau que le package sources.

Bilan sur le travail

Eloi : Durant l'entièreté du projet j'ai été chargé de la synthèse vocale de l'instructeur, tout d'abord avec les bibliothèques SI_VOX et FreeTTS puis ensuite nous comptions utiliser Espeak mais suite à la découverte d'une nouvelle bibliothèque, nous avons donc supprimer tout l'ancien code existant pour passer sur AZURE. J'ai également participé à la reconnaissance vocale.

Hachim : Je me suis occupé de la création de la syntaxe du fichier à rentrer dans le logiciel et de la création des 3 fichiers d'internationalisation qui permette le changement de langue. J'ai créé le bandeau et la page des paramètres d'un exercice. Pour la troisième phase, j'ai créé le fichier de grammaire avec la notation EBNF et en utilisant la librairie JavaCC pour la vérification du fichier.Et j'ai grandement participer au rendu final que ce soit la rédaction en grand partie du word ou de la réalisation du manuelle et de la vidéo.

Winona : Durant le projet, je me suis chargée des traitements vocaux, plus particulièrement la partie sur la reconnaissance vocale de l'utilisateur. Mais également la synthèse vocale de l'instructeur à l'aide de la bibliothèque Azure de Microsoft. Nous l'avons à la employés suite d'autres bibliothèques utilisées comme FreeTTS ou Sphinx4, qui n'ont pas été très concluante en ce qui concerne la simplicité d'utilisation et la compréhension de la voix de l'instructeur.

Ryan : Au cours du projet, j'ai principalement travaillé sur l'interface graphique, en allant du maquettage jusqu'à la programmation. Je n'ai néanmoins pas vraiment travaillé sur la couche design. Les travaux qui m'ont pris le plus de temps ont été le maquettage de tous les écrans de l'application et la programmation des écrans

Llona : Avant le début du projet j'ai pris connaissant du sujet afin de planifier une organisation. J'ai invité les membres de l'équipe à lire toutes les informations du Moodle. J'ai planifié chaque semaine une réunion pour parler du projet, de son avancement et des tache prévu pour la semaine. Pour la phase 1 j'ai demandé de travailler sur la conception de l'application, (Maquette, grammaire du fichier, recherche de licence et bibliothèque, UML). Pour la phase 2 j'ai attribué des fonctionnalités à développer à chaque membre de l'équipe (Grammaire, reconnaissance vocal, synthèse vocal, programmation du corps de l'application, IMH). J'ai programmé la lecture des fichiers de configuration et leurs vérifications. J'ai tout au long du projet assisté les membres de l'équipe Durant la dernière phase je me suis occupé de la fixation des bugs de l'application et de la sauvegarde des données du logiciel.

Anthony : Au cours du projet, j'ai travaillé sur la conception UML du projet, sur la retranscription du diagramme UML en code, sur la programmation des méthodes du code retranscrit. J'ai principalement travaillé sur les méthodes de constructions d'exercices et de chorégraphies qui doivent être créées en fonction des contraintes imposées par l'utilisateur. J'avais commencé à coder les Sessions qui devaient être finies mais par manque de temps, nous n'avons pas pu les finir. Enfin, j'ai travaillé sur la Javadoc et les tests unitaires.

Liens

https://git-iutinfo.unice.fr/rey/s4t-pt-2022-hematraining/-/tree/Version_final