

Informatique pour les sciences humaines – Projet de développement logiciel – Rapport

Table des matières :

• Contexte	p.1
• Idée du projet	pp.1-2
• Concurrence	p.2
• Contraintes et risques	pp.2-3
• Avantages	p.3
• Objectifs	p.3
• Organisation	pp.3-4
• Diagramme de Gantt	p.5
• Ressources	p.5
• Maquettes	pp.5-9
• Réalisation du projet et structure du code	pp.9-10

Contexte :

Dans le cadre du cours *Projet de développement logiciel*, nous avons été amenées à réfléchir à une idée de projet informatique à réaliser durant un semestre (de février 2023 à mai 2023). Tout au long de leurs études, les étudiants doivent apprendre de nouvelles langues. Pour ce faire, de nombreux jeunes adoptent la méthode des *flashcards*, c'est-à-dire des cartes, créées manuellement, présentant au recto un mot dans la langue d'origine de l'apprenant et au verso sa traduction dans la langue à apprendre. Cette méthode, bien qu'efficace dans l'apprentissage d'une langue étrangère, comporte quelques désavantages liés au fait que les cartes soient faites à la main. En effet, l'étudiant ne peut pas réellement juger de sa performance ni voir totalement sa progression étant donné qu'il n'y a aucun moyen de comptabiliser le nombre de réponses justes ou fausses données. Les cartes n'étant pas toujours bien stockées, ni sauvegardées, il est fréquent d'en perdre ou d'en abîmer obligeant alors l'apprenant à en refaire de nouvelles. C'est à partir de ces observations que nous avons pensé à un logiciel informatisé d'apprentissage de langues basé sur le concept de *flashcards* que nous avons nommé *Intelligent FlashCards (IFC)*. Le rendu du projet est fixé au 28 mai 2023.

Idée du projet :

Intelligent FlashCards est un logiciel permettant d'apprendre de nouvelles langues facilement tout en observant sa progression. En se connectant à son compte, l'utilisateur aura accès à différentes options :

- Mes cartes : permet à l'utilisateur de voir les cartes déjà créées et enregistrées telles qu'il les a regroupées par paquet. L'utilisateur a alors la possibilité de se faire interroger sur ces mots ou de modifier des cartes (ajouter, modifier ou supprimer des mots et leur traduction).
- Nouvelles cartes : permet à l'utilisateur d'ajouter de nouvelles cartes et de nouveaux paquets de cartes.
- Mes résultats : permet à l'utilisateur de visualiser ses différents résultats obtenus lors de ses interrogations sur le logiciel. Ceci lui permet de repérer quels sont les mots ou groupes de mots où se trouvent le plus de réponses incorrectes et qui lui posent ainsi le plus de difficultés. Cela permet également de voir une progression, élément motivant lors de l'apprentissage d'une langue.

Tous ces éléments proposent à l'utilisateur une nouvelle manière ludique de se familiariser avec des langues inconnues.

➔ Attention ! Ce logiciel n'est pas un programme d'apprentissage d'une langue totalement autonome. La participation de l'utilisateur est primordiale.

Concurrence :

L'apprentissage de nouvelles langues est déjà l'objet de nombreuses autres applications et logiciel. Notons, parmi les principaux acteurs de ce domaine, l'application et site internet *Duolingo* qui propose divers exercices à l'utilisateur pour apprendre une langue, ou encore *Babel* qui offre à l'usager différents cours de langues. Néanmoins, ces deux exemples ne se centrent pas autour de la méthode des *flashcards*, méthode pourtant très pratique pour les étudiants étant donné qu'ils n'ont qu'un certain vocabulaire précis à apprendre et que le seul fait d'inscrire les mots et leurs traductions permet déjà à l'élève de se familiariser avec son vocabulaire. Notre principal concurrent serait donc le site internet *Quizlet* qui se base également sur le principe des *flashcards* créées par l'utilisateur. Néanmoins, bien que l'utilisateur puisse y créer ses propres cartes tel qu'il le désire et se faire questionner dessus, le système de résultats lors de l'interrogation par cartes n'est pas complètement similaire au notre qui offre plus de liberté à l'apprenant.

Contraintes et risques :

Comme tout autre projet, celui-ci comporte plusieurs contraintes, que ce soit pour l'utilisateur ou pour nous développeurs, et quelques risques apparaissent à sa réalisation :

Contraintes du logiciel pour l'utilisateur et risques liés :

- **Compte** : pour l'utilisation du logiciel, l'apprenant doit créer un compte en donnant son nom d'utilisateur, ainsi que son adresse mail et un mot de passe qu'il invente.
- **Support** : le projet ne sera disponible que sur ordinateur
- ➔ Le risque serait que l'utilisateur n'accepte pas ces conditions et ne veuille pas utiliser notre logiciel.

Contraintes du projet pour le développeur et risques liés :

- Temps : le projet doit être terminé dans un délai imparti d'environ trois mois.
- ➔ Le risque serait que nous ne parvenions pas à finaliser le projet dans les temps.
- Autres contraintes inattendues qui risqueraient de mettre en péril le projet.

Avantages :

Intelligent FlashCards possède de nombreux avantages :

- Utilisation intuitive du logiciel.
- Logiciel s'adaptant à tous les niveaux d'apprentissages de langues.
- Enregistrement et stockage sûr des mots et leurs traductions dans la base de données.
- Onglet *mes résultats* permettant à l'utilisateur de visualiser sa progression.
- Bien que le public cible soit prioritairement les écoliers et les étudiants, le logiciel est accessible à toutes les personnes souhaitant apprendre de nouvelles langues.

Objectifs :

Le projet *Intelligent FlashCards* comporte de nombreux objectifs visés dans différents domaines. Ces objectifs sont listés ci-dessous :

Objectifs fonctionnels :

- Apprendre une langue grâce à des *flashcards* informatisées.
- Evaluer son apprentissage dans une nouvelle langue.

Objectifs d'affaire :

- Projet terminé et fonctionnel pour le 28.05.23.

Objectifs techniques :

- Créer une base de données fonctionnelle (*backend*).
- Créer une interface intuitive et fonctionnelle (*frontend*).
- Relier ce *backend* au *frontend*.

Objectifs de qualité :

- Produit facile d'utilisation.
- Destiné à un public cible, les élèves et les étudiants, mais accessible à tous.

Organisation :

Nous avons organisé la réalisation du projet en cinq étapes majeures dont la planification sera exposée au chapitre suivant :

- I. Conception et mise en route du projet
 - i. Description du projet

- ii. Etude de marché
 - iii. Répartition des tâches
- II. Définition et planification du projet
 - i. Plan du projet
 - ii. Calendrier du projet / Diagramme de Gantt
 - iii. Maquettes
- III. Lancement et exécution du projet
 - i. *Backend*
 - ii. *Frontend*
 - iii. Lien entre le *backend* et le *frontend*
- IV. Performances et contrôle du projet
 - i. Tests
- V. Clôture du projet
 - i. Rapports

Le rôle de directeur de projet, ainsi que celui de directeur technique n'est pas prédéfini et chacun se voit attribué, au commencement du projet, une tâche dont il est responsable. Ainsi :

- Laure Margot est responsable à 100% de la partie *frontend* (créer une interface intuitive et fonctionnelle) et de la gestion des comptes utilisateurs en association avec Leonie Nussbaum (partie *view*) et de la réalisation de la version avant-projet du *Project plan*.
- Leonie Nussbaum est responsable à 100% de la partie *frontend* (créer une interface intuitive et fonctionnelle) et de la gestion des comptes utilisateurs en association avec Laure Margot (partie *view*) et de la réalisation du diagramme de Gantt.
- Olivia Verbrugge est responsable à 100% de la partie *backend* (créer une base de données fonctionnelle) (partie *model*) et de la réalisation des maquettes, ainsi que de la rédaction de ce rapport.
- Sinem Kilic est responsable à 100% de relier le *backend* au *frontend* (partie *controller*) et de la réalisation de certaines maquettes.

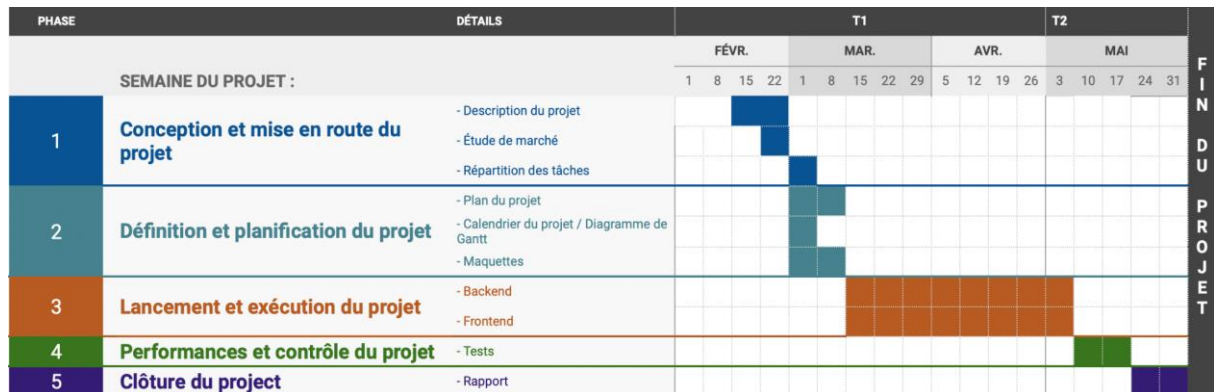
De plus, chacun est responsable de commenter ses propres parties du code et d'alimenter le *readme* du projet.

- ➔ Cette organisation a été revue en cours de réalisation du projet. Ainsi, Leonie Nussbaum, en plus des tâches qui lui étaient assignées ci-dessus, a pris en charge la partie relier le *backend* au *frontend* (partie *controller*) et Sinem Kilic s'est occupée de commenter toutes les parties du code et de rédiger le *readme*.

Pour organiser au mieux notre travail et afin de rendre celui-ci dans le délai imparti, nous avons décidé de toutes nous réunir au minimum une fois par semaine en présentiel, à l'université de Lausanne, les mercredis de 16h15 à 17h45 dans le but d'avancer sur le projet. Si besoin, des réunions par *zoom* ou en présentiel sont envisageables pendant la semaine pour parler du projet et avancer sur celui-ci avec les quatre membres de l'équipe ou une partie d'entre eux. La présentation finale du projet se déroulera le mercredi 31 mai 2023 à l'université de Lausanne.

Diagramme de Gantt :

Ci-dessous notre diagramme de Gantt exposant le temps planifié pour chaque tâche :



Ressources :

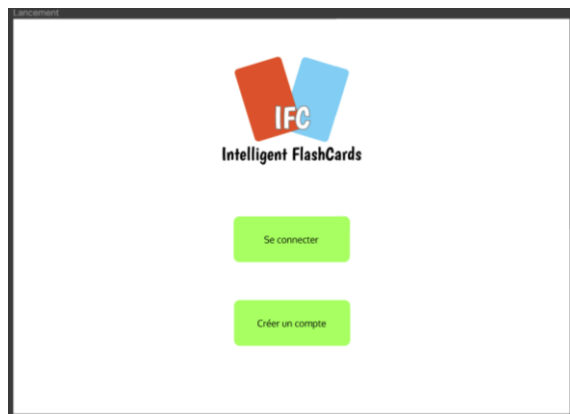
Pour nous aider dans la réalisation de ce projet, nous avons à notre disposition divers outils. Tout d'abord, nos maquettes sont réalisées avec l'éditeur en ligne *Figma* qui permet l'élaboration de prototypes. Nous nous servons de l'environnement de développement proposé par *PyCharm* pour coder. En effet, *PyCharm* propose de nombreuses fonctionnalités utiles et offre la possibilité d'un lien direct avec *GitHub* que nous utilisons pour déposer, partager et stocker nos différents codes. Finalement, l'application *Streamlit* est également employée pour l'élaboration du *frontend*.

L'entièreté de nos codes (parties *view*, *model* et *controller*) sont codées en *python*.

Maquettes :

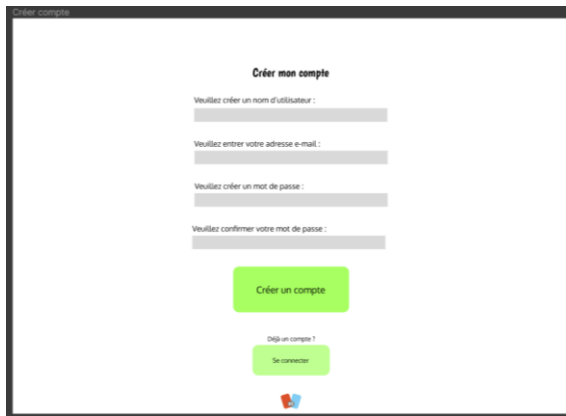
Ci-dessous les maquettes de nos principaux écrans réalisées avec l'éditeur en ligne *Figma*. Précisons, néanmoins, que celles-ci ne sont pas des images figées de ce à quoi devrait ressembler notre logiciel, mais des modèles malléables représentant une idée générale des éléments qui doivent figurer sur dans notre interface.

Ecran de lancement :

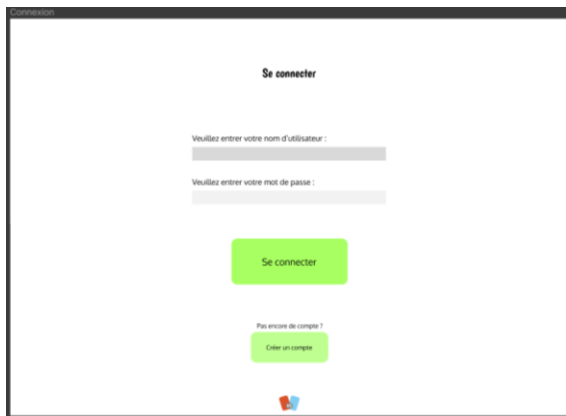


Ecran apparaissant lors de l'ouverture du logiciel.

Création ou connexion à un compte :



Ecran de création d'un compte utilisateur.



Ecran de connexion à un compte utilisateur

Menu principal :



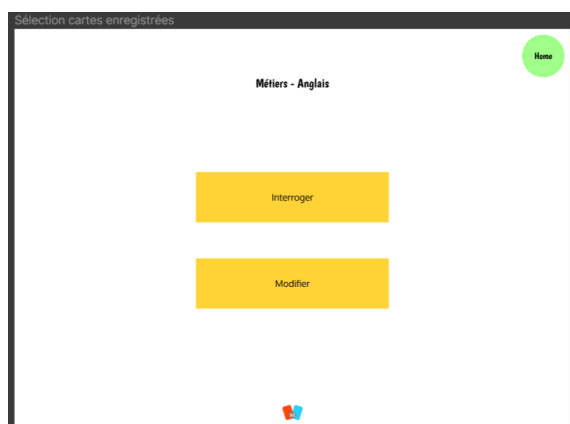
Ecran du menu principal, une fois l'utilisateur connecté.

Mes cartes :

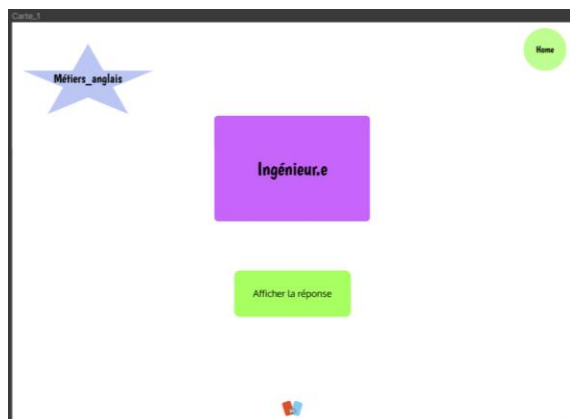


l'utilisateur.

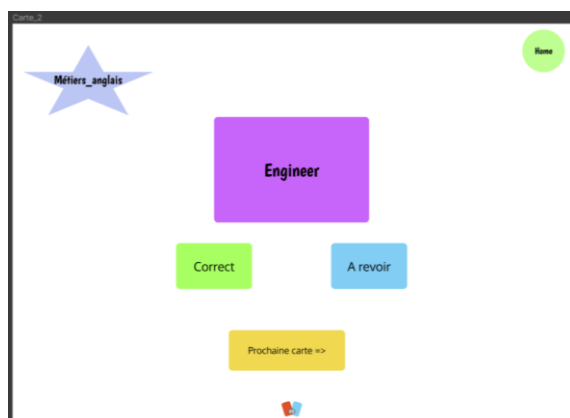
Ecran affichant les paquets de cartes enregistrés de



Ecran pour un paquet de cartes enregistrées sélectionné.



Ecran de question lors d'une interrogation de cartes.



Ecran de réponse lors d'une interrogation de cartes.

Création nouvelles cartes manuellement

Home

Modifier des cartes

Nom du paquet de cartes : métiers

Votre liste de mots à traduire en : anglais

- Ingénieur.e
- ...
- ...
- ...

Voulez-vous ajouter d'autres mots à la liste ?

Ajouter d'autres mots

Si votre liste est complète, finaliser sa sauvegarde :

Sauvegarder la liste

Ecran de modification de cartes.

Nouvelles cartes :

Création nouvelles cartes

Home

Créer de nouvelles cartes

Veuillez inscrire le titre de votre nouveau paquet :

Veuillez inscrire un nouveau mot :

Veuillez inscrire sa traduction :

Ajouter d'autres mots

Sauvegarder la liste

Ecran de création de nouvelles cartes.

Mes résultats :

Résultats sélection

Home

Mes résultats

Filtrer par paquet :

Veuillez sélectionner un paquet de cartes :

Métiers
Anglais

Animaux
Néerlandais

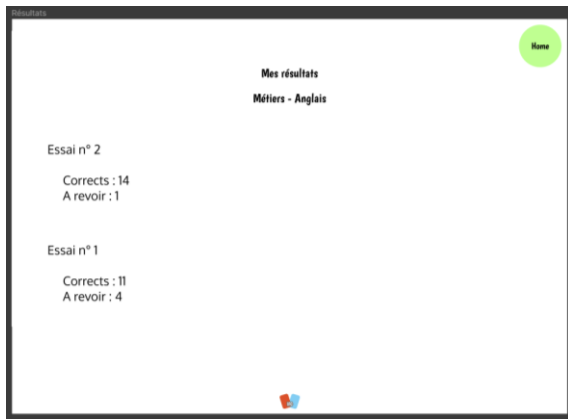
Ecole
Allemand

Sports
Néerlandais

Ecole
Anglais

Bâtiments
Anglais

Ecran affichant les paquets de cartes enregistrés de l'utilisateur.



Ecran de résultats pour un paquet de cartes enregistrées sélectionné.

Réalisation du projet et structure du code :

Le code exécutable du projet se trouve sur la branche *main* du dossier *dpicca/IFC* sur *GitHub*.

Nous avons structuré notre code selon le modèle *mvc* (*model – view - controller*), le *model* stockant les différentes données, le *view* créant l'interface graphique et le *controller* gérant la logique des actions exécutables par l'utilisateur et faisant ainsi le lien entre le *model* et le *view*. Notre dossier principal *IFC* contient ainsi trois dossiers distincts : *controler*, *model* et *view*. Le dossier *controler* contient un fichier de code *python* nommée *controler.py*. Comme expliqué plus haut, celle-ci fait le lien entre les parties *model* et *view* en extrayant les différentes méthodes utiles du *model* pour les passer ensuite au *view*. Le dossier *model* contient deux fichiers : *ifc.db* et *sqlite_flashcard.py*. Le premier (*ifc.db*) est la base de données où sont stockées les différents mots, leurs traductions, les utilisateurs et leurs résultats, etc. (voir schéma ci-dessous). Le second fichier (*sqlite_flashcard.py*) contient le code *python* donnant toutes les instructions nécessaires à l'interaction avec la base de données. Cinq classes, représentant les cinq tables du schéma ci-dessous, sont présentes dans ce code avec, pour chacune, différentes méthodes applicables selon la demande. Finalement, la partie *view* contient une image de notre logo sous *logoIFC.png*, un fichier *main.py* codant en *python* la création de compte ou la connexion à son compte d'un utilisateur et un dossier *pages* contenant différents fichiers de code *python* : *MenuIFC.py*, *mes_cartes.py*, *nouveau_paquet.py* et *resultats.py*. Chacun de ces fichiers correspond à l'écran qui lui est associé. Ainsi, *MenuIfc.py* code l'interface et les interactions possibles du menu principal de l'utilisateur connecté, *mes_cartes.py* celui des paquets et cartes, *nouveau_paquet.py* celui de la création de nouveaux paquets de cartes et *resultats.py* celui des résultats.

Maquette de la base de données :

