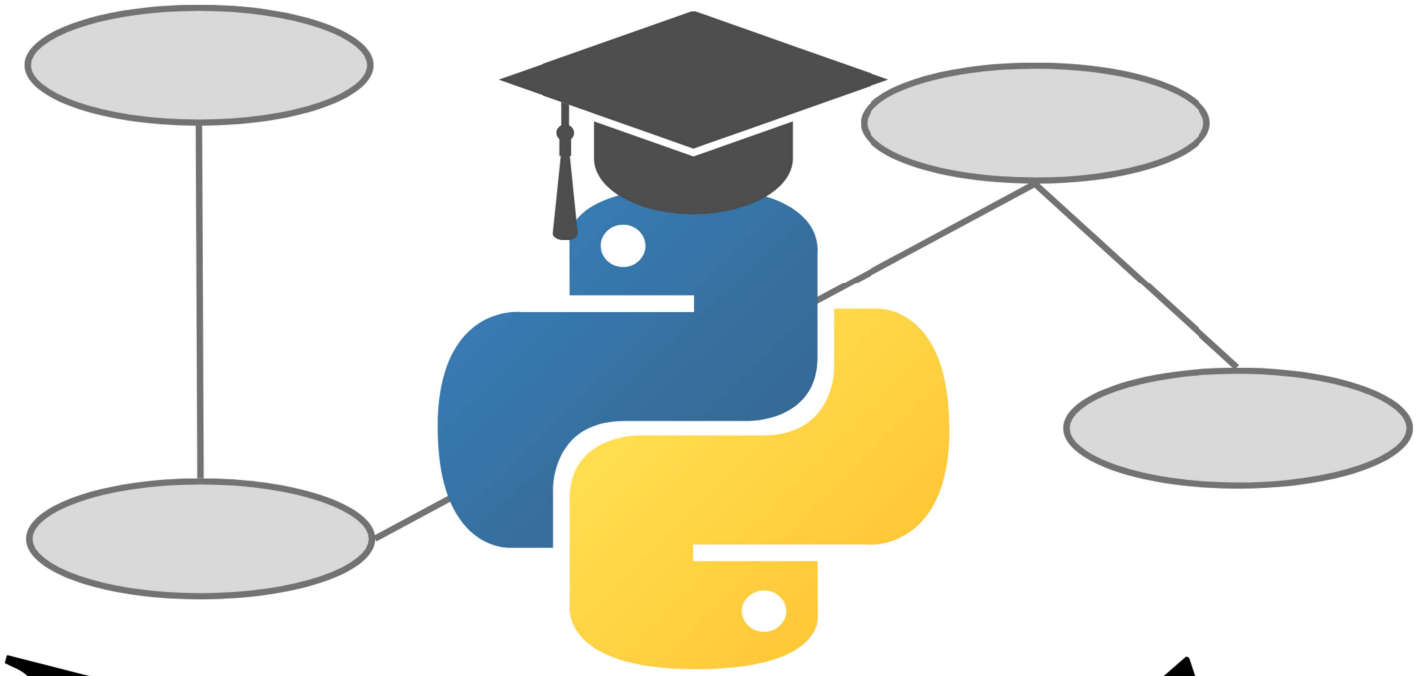
Édition 2023

DOSSIER DE CANDIDATURE
PRÉSENTATION DU PROJET



WikiPath

Ce document est l'un des livrables à fournir lors du dépôt de votre projet : 4 pages maximum (hors documentation).

Pour accéder à la liste complète des éléments à fournir, consultez la page **Préparer votre participation**.

Vous avez des questions sur le concours ? Vous souhaitez des informations complémentaires pour déposer un projet ? Contactez-nous à **info@trophees-nsi.fr**.

NOM DU PROJET : WikiPath

> PRÉSENTATION GÉNÉRALE :

Wikipath, et tu trouveras ton chemin à travers Wikipédia. (D'où le nom: Wiki = wikipédia, Path = chemin)

Wikipath est une application qui permet aux utilisateurs de naviguer facilement entre les articles de Wikipédia en visualisant les liens sous forme de graphe. Les utilisateurs peuvent ainsi explorer les sujets qui les intéressent de manière intuitive et découvrir de nouveaux articles en quelques clics.

Lors de la recherche de l'idée, nous avons rapidement réalisé quelque chose: il serait encore plus amusant et excitant de créer quelque chose d'**unique**, et si possible de répondre à une demande. Nous nous sommes ainsi lancés dans une petite étude de marché, pour trouver une idée qui pourrait être utile et novatrice. Elle s'est rapidement orientée sur l' encyclopédie collaborative co-créée par Jimmy Wales et Larry Sanger qu'est **Wikipédia**. Et là, bingo ! Qui n'a jamais été confronté à l'angoisse de la barre de recherche du site ? Une envie de découverte, d'apprentissage, mais aucune inspiration quant à la page que l'on devrait rechercher. C'est à ce moment que nous avons eu notre idée. Wikipédia est une source intarissable de contenu (plus ou moins fiable), mais qui manque cruellement de **visuel** et d'une manière plus tangible de représenter les relations entre les savoirs, au-delà des liens hypertextes. Du texte, du texte, une image et encore du texte …
Nous nous sommes donc questionnés: "**Comment pourrions nous rendre la navigation sur Wikipédia plus fluide et instinctive ?**"
Pour rendre ces données plus attractives, nous avons eu l'intuition que les afficher sous la forme d'un **graphe** serait la solution. Il est en effet possible de visualiser Wikipédia comme un unique graphe géant.

Notre **équipe** qui s'est soudée au gré du déroulement du projet est composée de 3 Wikipédiens: **Esteban**, **Lise**, **Arnaud**.

Le célèbre leitmotiv: "Seul on va plus vite, ensemble on va plus loin", et nous avons décidé d'aller très loin dans l'exploration de la forêt de données appelée Wikipédia.

> ORGANISATION DU TRAVAIL :

Ce projet a fait grandir la place que nous attribuons à la **planification des tâches**, car nous nous sommes vite rendu compte que se lancer tête baissée dans le code était rarement la bonne idée. D'autre part, c'était une première pour nous trois de travailler sur un projet si conséquent en équipe. Cela a donc mené à la création d'outils nous permettant de communiquer et programmer tous ensemble, comme le **Github** pour se partager le code et faciliter la mise en commun de nos parties, le **Discord** pour communiquer et le **Notion** (google docs mais en mieux) du projet pour s'organiser et noter les informations importantes. Nous avons tout de même mis du temps avant de parfaitement maîtriser notre espace de travail, et nombre de fonctions se sont fait écraser par les push "fous" de Esteban sur le Github.

Une autre chose que nous avons dû apprendre à faire pour le bon déroulement du travail d'équipe est de rendre le **code propre**. Fini les fonctions non commentées, non optimisées, les "break" et les "global", fini aussi l'unique fichier interminable avec tout le code.

Nous avons été lancé sur le projet début Avril, nous avions donc un mois pour réaliser un projet. Au début, cette **deadline** nous a semblé impossible à respecter, mais nous étions déterminés à faire naître notre idée. C'est ainsi qu'en plus des 6h que nous avons eu en cours, nous avons dû prendre une cinquantaine d'heures de travail chacun sur notre temps libre (qui se sont parfois prolongées tard le soir), afin de rendre un travail dont nous soyons fier, et qui soit à la hauteur des objectifs que nous nous sommes donné. Nous sommes contents du résultat, même si nous pensons qu'avec un peu plus de temps il aurait été possible de pousser encore un peu plus loin ce projet.

Lise: Interface graphique, directions, idées et répartition des tâche

Esteban: Algorithmie, corp du programme
Esteban a une capacité à chercher la solution à un bug ou à un problème sans jamais abandonner. Il a donc beaucoup débuggé car les autres membres du projet étaient plus vite lassé de chercher, tel une aiguille dans une botte de foin, un pauvre caractère mal placé au milieu du code.

Arnaud: réflexion ergonomie de l'interface, communication (README, description et vidéo), BDD

LES ÉTAPES DU PROJET :

Notre idée originelle était un site qui consiste en une **carte mentale** auto-générée à l'aide de **chat-GPT** et peut-être d'un scraping de wikipédia (car nous avions déjà des connaissances en la matière), et qui stockerait les données dans une base de donnée.

Nous nous sommes ensuite rendu compte que l'API de ChatGPT était payante (encore envisageable car les coûts sont dérisoires). Nous n'avons donc pas retenu cette piste car nous avons pensé que cela apporterait de l'aléa et diminuerait la pertinence des informations.
Nous avons de ce fait décidé d'utiliser uniquement les données de Wikipédia, mais nous avons pris beaucoup de temps à trouver la solution la plus efficace. En effet, nous avons en premier eu l'idée de scrapper les données du site, puis d'utiliser Wikidata, la base de données de Wikipédia, pour enfin nous tourner vers l'**API** de ce même site (qui va en fait scrapper les données) car c'était le plus simple à comprendre et mettre en place.
Nous avons donc appris à nous en servir, ce qui nous a mené à une première grosse étape.
Cette première version du code consistait en un **algorithme** qui allait chercher toutes les **pages « voisines »**, c'est à dire toutes les pages qui sont référencées (au moyens d'un lien hypertexte) d'une page donnée et qui les affichait dans la console Python.
Nous avons commencé par coder cet algorithme de sorte à ce que nous puissions continuer de façon récursive l'opération sur les pages voisines. Nous avions donc une fonction qui prenait un niveau "n" en paramètre, et qui allait chercher les pages sur n niveaux (exemple: la page voisine de la page voisine

de la page de départ pour n = 2). Nous avons vite pris conscience de la **complexité** exponentielle de l'algorithme, en effet, avec seulement un niveau on avait en moyenne une cinquantaine de voisins, tandis qu'avec trois niveaux, on pouvait atteindre (si l'on compte en moyenne 50 liens par pages) 125000 pages, soit environ cinq fois la totalité de Wikipédia en français. Et qui dit une grande complexité, dit **temps de chargement important.**
Nous avons ainsi décidé de n'afficher qu'un niveau de graphe, c'est-à-dire les 50 voisins (ou moins si la page n'en a pas autant) de ladite page afin de limiter le temps de chargement.

Après la réalisation de notre algorithme, nous nous sommes attelés à la **modélisation de la base de données**. Nous avons ainsi conçu deux tables pour stocker les informations sur les utilisateurs et leur pages enregistrées (schéma relationnel en annexe).

Parallèlement, nous avons travaillé sur la conception et la réalisation de **l'interface graphique**. Pour cela, nous avons choisi d'utiliser la bibliothèque **Tkinter**, qui permet de créer facilement des interfaces en Python. Cependant, pour personnaliser davantage notre application, nous avons décidé d'ajouter des éléments de "custom tkinter" afin de rendre l'interface plus moderne et attrayante.

Une fois la modélisation de la base de données et l'interface graphique achevées, nous nous sommes concentrés sur la **finalisation et l'amélioration du code**. Nous avons pris soin de vérifier que notre application fonctionnait correctement, d'optimiser les performances et de débugger. En parallèle, nous avons rédigé la **documentation détaillée** pour faciliter l'utilisation de notre outil et assurer une bonne compréhension de son fonctionnement.

Enfin, pour présenter notre projet de manière claire et engageante, nous avons réalisé une **vidéo**. Nous avons beaucoup peiné à faire rentrer tout ce que nous voulions dire en seulement **deux minutes**, si cela ne tenait qu'à nous on en aurait fait un documentaire de plusieurs heures.

(diagramme de Gantt en annexe)

> FONCTIONNEMENT ET OPÉRATIONNALITÉ :

Les utilisateurs commencent par entrer un mot-clé ou sélectionner un article de Wikipédia, puis Wikipath génère un graphe montrant les articles liés. Les utilisateurs peuvent ensuite cliquer sur les nœuds du graphe pour accéder aux articles correspondants et explorer de nouveaux sujets.

Au cours de l'écriture du code du projet, nous avons maintes fois faillit nous égarer. En effet, il y avait de nombreuses situations où nous voulions utiliser des **langages hors du programme** de terminal. Tout d'abord, nous voulions pour la partie graphique utiliser un site web comme support, avec donc du HTML, CSS, JS, et peut être PHP. Nous avons donc basculé sur une interface graphique **Tkinter**.

Ensuite, lors de la création de la base de données, il nous a semblé pertinent de partir sur une base de données de graphe comme neo4j, mais elle requiert l'utilisation du langage Cypher. Nous nous sommes donc orienté vers une **base de données SQL**, néanmoins sûrement un peu moins optimisée pour stocker des graphes.
Nous avons bâti ce projet techniquement ambitieux en majorité sur les notions étudiées au fur et à mesure de notre année, avec notamment la notion de graphe, l'utilisation de la **programmation orientée objet** pour modéliser les noeud de celui-ci, ou encore le **SQL** et les **bases de données relationnelles** pour stocker, organiser et manipuler les données.

> OUVERTURE :

- afficher les plus de niveaux pour faire une vrai carte mentale
- faire une version Mac OS et Linux
- Ajouter la possibilité de parcourir Wikipédia en Anglais
- Augmenter le côté communautaire en ajoutant de l'interaction entre les utilisateurs (messagerie interne, vision des nœuds visités par les autres).
- Pouvoir faire des dossiers dans ses favoris

DOCUMENTATION

Installation :

.exe :
Nous avons mis à disposition un fichier zip contenant un .exe du programme python. Il est donc entièrement utilisable sans installer aucun module ni même avoir installé python sur son ordinateur, les fichiers ont été convertis avec le module auto-py-to-exe qui est très facile d'utilisation (lien vers la doc : https://pypi.org/project/auto-py-to-exe/). ! Il se peut que l'antivirus de l'ordinateur bloque l'utilisation du fichier, si cela arrive, nous vous conseillons de le désactiver le temps du test ou simplement de passer par le lancement par python.
python :
Pour exécuter le fichier depuis python il faut avoir installé tout les modules dont le programme a besoin pour fonctionner. Voici la liste des modules pour pouvoir lancer le projet :
- customtkinter: https://pypi.org/project/customtkinter/0.3/
-wikipediaapi: https://pypi.org/project/Wikipedia-API/
-PyQt6: https://pypi.org/project/PyQt6/
-Pillow: https://pypi.org/project/Pillow/
-requests: https://pypi.org/project/requests/
-beautifulsoup: https://pypi.org/project/beautifulsoup4/

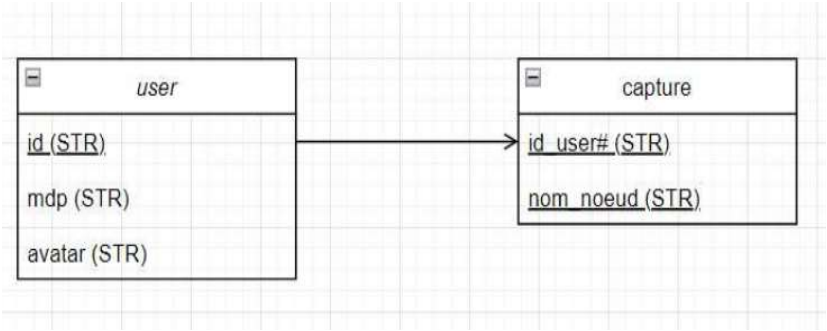
Les autres modules sont normalement installés automatiquement avec l'installation de python, l'installation des modules ci dessus devrait suffire a faire fonctionner le programme. Une fois les modules installé lancer le fichier main.py et le programme devrait s'exécuter correctement.

Utilisation :

Le programme fonctionne comme une application avec une interface utilisateur.
dans le graphe :
-double clic :Pour changer de nœud et afficher le nouveau nœud sur lequel l'utilisateur viens de cliquer -clic sur un nœud : Pour le déplacer dans le canvas
-clic sur un espace vide : Pour déplacer l'ensemble des éléments
- dans l'application : navigation à l'aide des boutons : La navigation hors du canvas est assez intuitive, suivez simplement ce qu'il y à écrit sur les boutons.

Spécifications technique :
Langages utilisés :
- SQL : pour la base de donnée
- python : pour l'algorithmie
- tkinter et customtkinter pour l'interface
- wikipediaapi pour récupérer les données de Wikipédia
- PIL pour la gestion d'image
- requests et beautifulsoup pour récupérer les images
- sqlite3 pour interagir avec la base de donnée

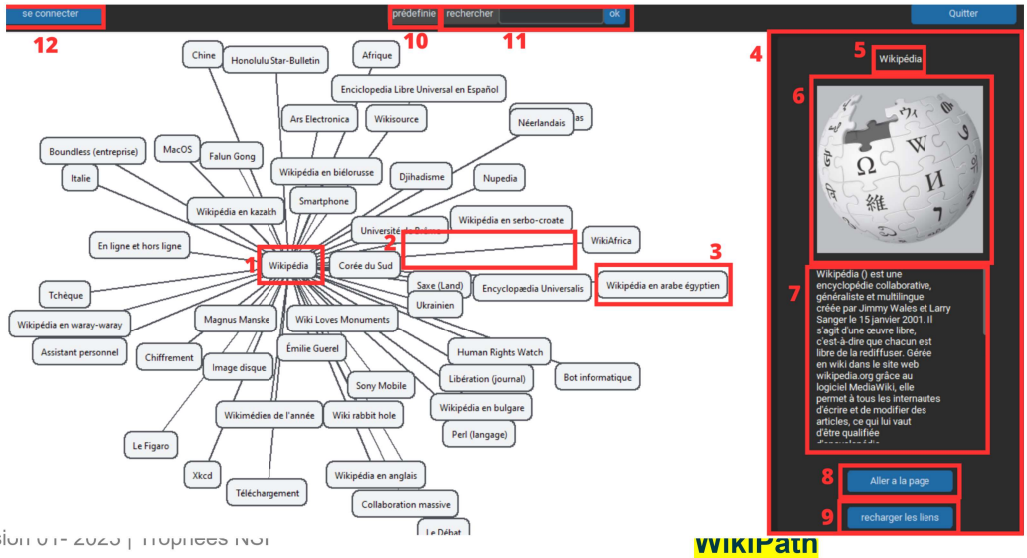
Stockage des données :
Les données sont stockées dans une base de donnée relationnelle, la base de donnée n'est utilisé que pour stocker les données utilisateur.



ANNEXE

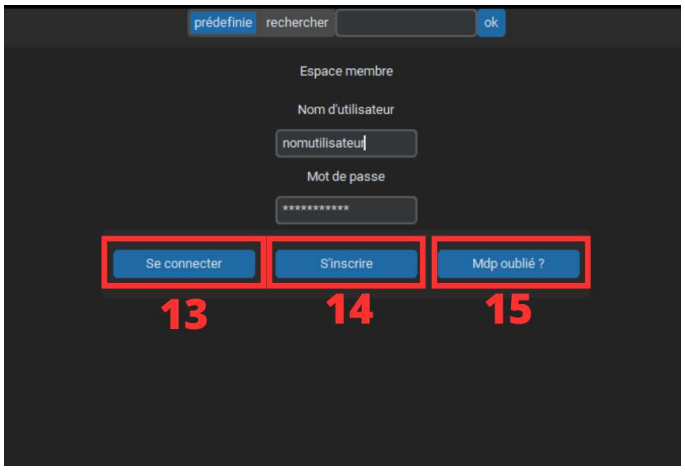
Présentation détaillé de l'application :

Page d'accueil :



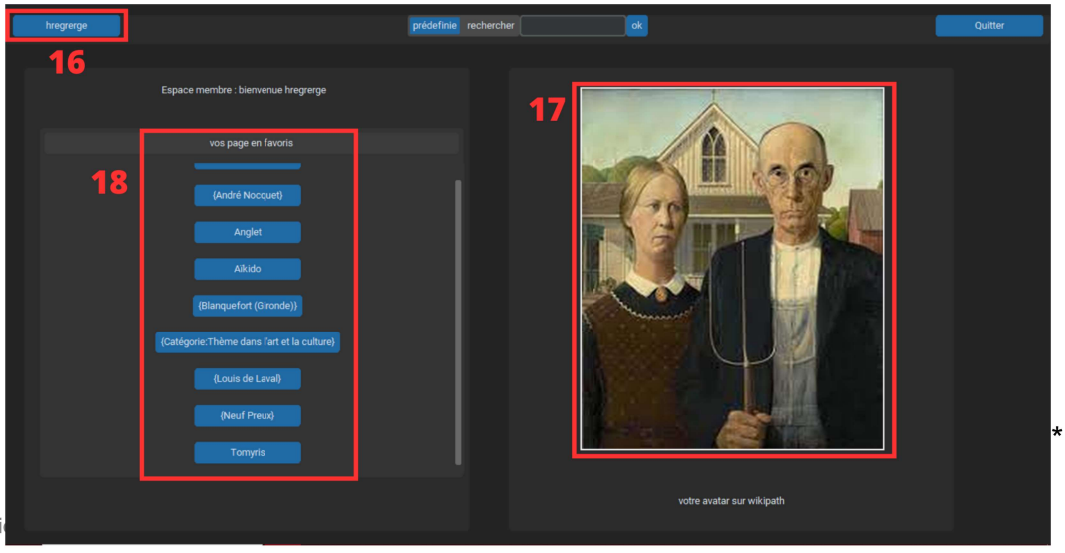
1) nœud central, correspond a la page qui est visité
2) liens entre les nœuds, utilisé pour montré le lien entre deux éléments
3) nœud voisin, permet d'accéder a une nouvelle page de Wikipédia
4) informations relatives au nœud central
5) titre du nœud central
6) image, si disponible sur le wiki, de présentation de la page
7) résumé de la page
8) bouton permettant d'aller sur la page wiki du nœud central
9) bouton permettant de générer un nouveau graphe a partir du même nœud central, choisis aléatoirement de nouveau voisins
10) bouton permettant d'accéder a la page du graphe prédéfini par nous même, une sélection de catégorie qui permet d'explorer de nombreux sujets
11) bouton permettant d'accéder a la page de recherche, permet de partir de n'importe quelle page du wiki en recherchant son nom.
12) bouton permettant d'accéder a la page de connexion.

Page de connexion :



13) si l'utilisateur possède déjà un compte
14) si l'utilisateur ne possède pas de compte
15) permet de modifier son mot de passe pour un compte existant

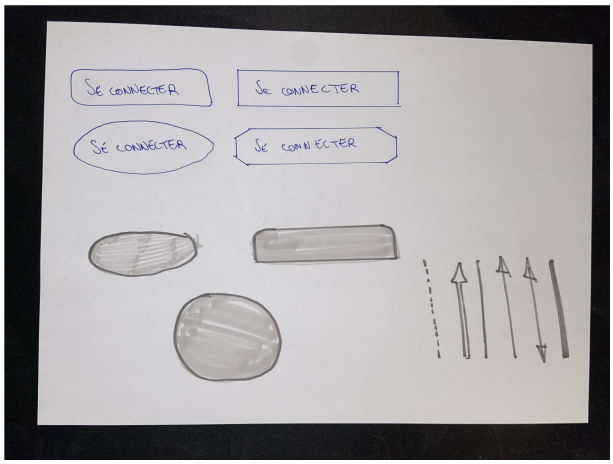
Page de gestion de compte :



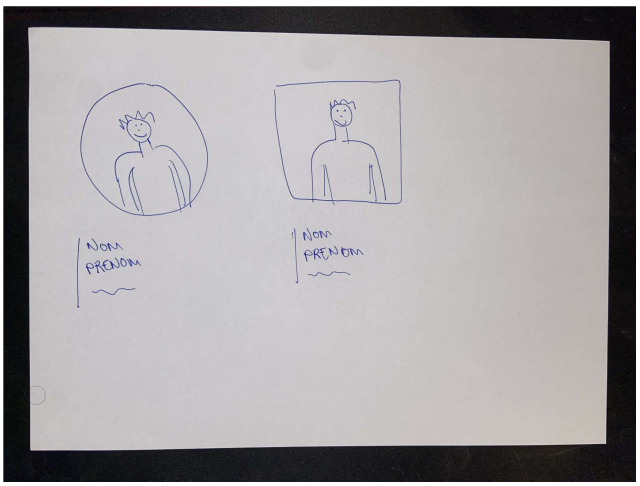
*

16) nom d'utilisateur une fois connecté, bouton pour accéder a la page de gestion de compte
17) avatar lié au compte
18) nœuds enregistrées lors de la navigation, permet de retourné sur une page déjà visitée.

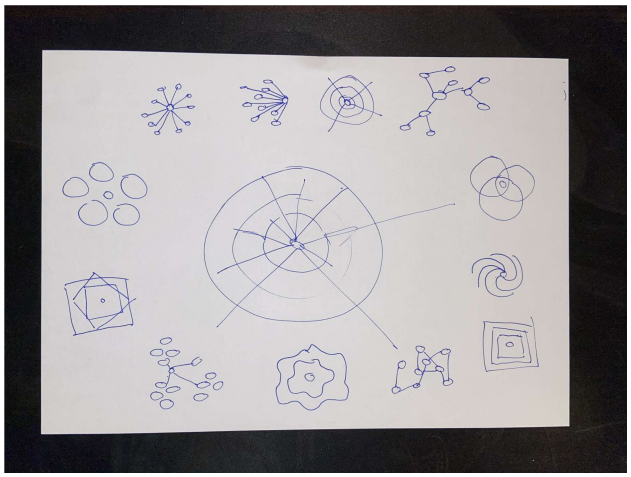
Quelques trace de nos réflexions :



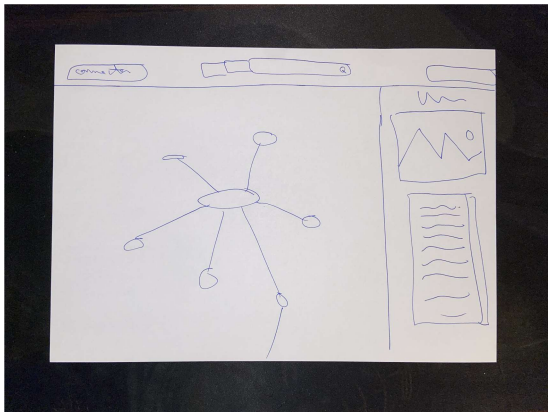
apparence des nœuds et liens entre eux, apparence des boutons



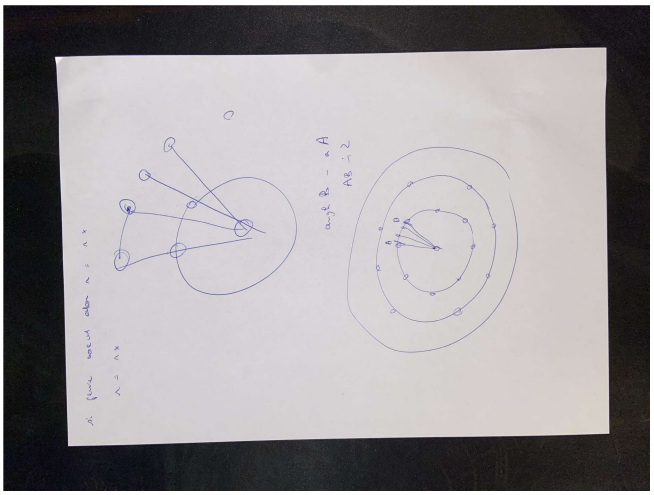
apparence de la page de gestion de compte



apparence du graphe / questionnement sur la génération du graphe



aperçu de la page d'accueil



génération des éléments du graphe

Répartition des taches dans le temps représenté avec un diagrame de gantt: