



FULLSTACK

CM #4 du 10/11/2022

Antonin ROSA-MARTIN - IDU7

Vidéo du cours:

https://drive.google.com/file/d/13sxql9zO8hciTR9knxYBW6XDWeDMPLh/view?usp=share_link

Contenu du cours

01

Questions par rapport au TPs ?

02

Utilisation de Trello

03

Utilisation de Git

04

Présentation des projets

Questions par rapport au TP2-1 ?

Créer un projet NodeJS avec `npm init`

Ajouter des dépendances avec `npm install <package>`

Installer les dépendances d'un fichier `package.json` avec `npm install`

Utilisation de NodeJS, Express, MongoDB et Redis (et SocketIO) pour le backend

Utilisation de NextJS pour le frontend

- `npx create-next-app@latest`

Utilisation de Docker et docker-compose pour le déploiement

- `docker build -t info734/tp2-1/api .`
- `docker build --build-arg BACKEND_URL=api-tp2-1:80 -t info734/tp2-1/website .`
- `docker-compose up -d`

Utilisation de Docker pour créer des containers de BDD au lieu de les installer

- `docker run -d --name mongo-bdd -p 27017:27017 mongo`
- `docker run -d --name redis-bdd -p 6379:6379 redis`

Gestion de projet avec Trello

Template:

<https://trello.com/invite/b/WkMaDETZ/ATTId78b9102cdac0ac2c0fda20b15e371914C49E85A/agile-board-template-info732>

Partage des tâches

Vous pouvez et devez partager les tâches entre les différents membres du projet, ces tâches seront des stories

Gestion du code

Votre code se structurera en stories, qui seront une itération du projet et qui dans le code seront des branches. Retour en arrière et meilleure compréhension du code !

Les stories avec Trello

Agile Board Template | INFO732 ☆ Workspace visible Board

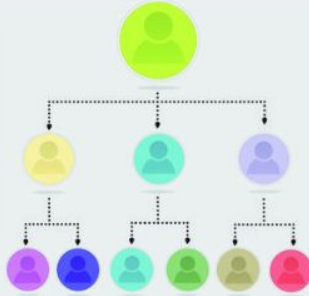
Done

- Demand Marketing
 - [IDU-1] Review Tech partner pages
- [IDU-2] Make sure sponsors are indicated for Tech Talk
- Planning
 - [IDU-3] Top 10 Trends list - Forbes
- [IDU-4] TBC Webinar: Ship Now, Not Later
- Happiness
 - [IDU-5] 1:1 Nancy
 - [IDU-6] Lead Gen Mandrill stats

+ Add a card

Current Sprint

- Government
 - [IDU-7] Going live with server deployment
- Planning
 - [IDU-8] Google Adwords list of referrers
 - [IDU-9] Q3 Webinar Content Planning



[IDU-17] IT Solutions page
1

+ Add a card

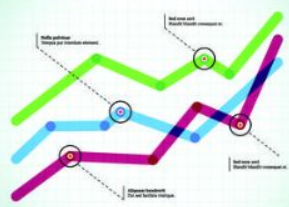
In Progress

- Remarket
 - [IDU-10] Android App new landing page
- [IDU-11] Analytics
- Remarket Partners
 - [IDU-12] Branding guidelines

+ Add a card

On Hold

- Partners
 - [IDU-13] CSS Rules
- Happiness
 - [IDU-14] Retail order
 - [IDU-15] Mobile UI reboot



[IDU-16] Google Analytics data - Q1
1

+ Add a card

Les stories avec Jira (Gitlab)

The screenshot displays a Jira interface for managing user stories. It features a sidebar with a tree view under the heading 'User Stories'. The tree is expanded to show two main categories: 'Must have' and 'Nice to have'. Under 'Must have', there are three stories, each with a detailed view card. The first story, 'As a shopper, I want to be able to add a book to my virtual shopping cart', is associated with 'TEST-3048' and has a status of 'In Progress'. The second story, 'As a shopper, I want to be able to see the total value of my current shopping cart', is associated with 'TEST-3049' and has a status of 'To Do'. The third story, 'As a shopper I want to be able to remove a book from my shopping cart', is associated with 'TEST-3050' and has a status of 'To Do'. Under 'Nice to have', there is one story, 'As a shopper, I would like to save items in my shopping cart for the future', associated with 'TEST-3051' and having a status of 'To Do'. Each story card includes a priority indicator (1 or 2) and a menu icon (three dots).

User Stories

- Must have**
 - > As a shopper, I want to be able to add a book to my virtual shopping cart
 - ▶ **TEST-3048** **In Progress** As a shopper, I want to be able to add a book to my virtual shopping cart 1 ...
 - > As a shopper, I want to be able to see the total value of my current shopping cart
 - ▶ **TEST-3049** **To Do** As a shopper, I want to be able to see the total value of my current shopping cart 1 ...
 - > As a shopper I want to be able to remove a book from my shopping cart
 - ▶ **TEST-3050** **To Do** As a shopper I want to be able to remove a book from my shopping cart 2 ...
- Nice to have**
 - > As a shopper, I would like to save items in my shopping cart for the future
 - ▶ **TEST-3051** **To Do** As a shopper, I would like to save items in my shopping cart for the future ...



Utilisation de GIT

git add <fichiers/dossier>

Ajoute le fichier ou le dossier dans les éléments à suivre

git commit -m <message>

Permet de commit une snapshot du projet

git branch <branche>

Permet de créer une branche

git checkout <branche>

Permet de mettre le “curseur” sur une branche et de faire des actions depuis cette dernière

git merge <branche>

Permet de merge la branch actuelle avec la branch cible et d’avoir les modifications des deux branches

git rebase <branche>

Permet de prendre toutes les modifications qui ont été validées sur la branche et les rejouer sur la branche cible



Fusionner les branches avec Git

La fusion de branches

Quand vous travaillez à plusieurs vous serez amenés à avoir plusieurs branches en parallèle. Comme vous n'êtes pas des sauvages vous allez faire vos modifications sur des branches séparées et il arrivera forcément le moment fatidique où vous devrez fusionner vos modifications...

git merge <branche>

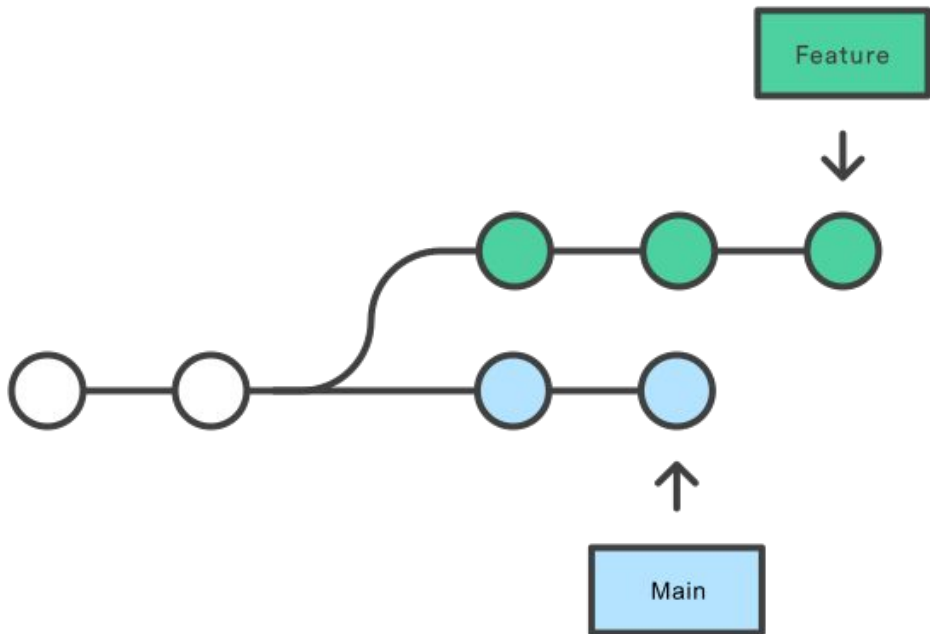
Intègre les modifications des commits de la branche cible dans la branche actuelle. Cette commande est utilisée par git pull pour incorporer les modifications d'un autre dépôt et peut être utilisée à la main pour fusionner les modifications d'une branche dans une autre.

git rebase <branche>

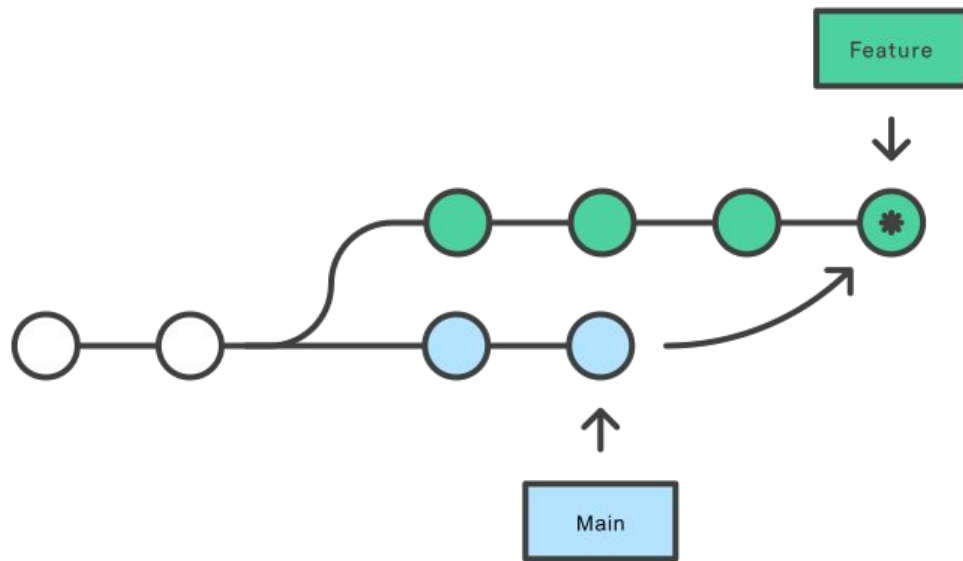
Cherche l'ancêtre (le commit) commun le plus récent des deux branches (celle sur laquelle vous vous trouvez et la branche cible), en récupérant toutes les différences introduites par chaque commit de la branche courante en appliquant chaque modification dans le même ordre.

git merge Main (depuis Feature)

A forked commit history



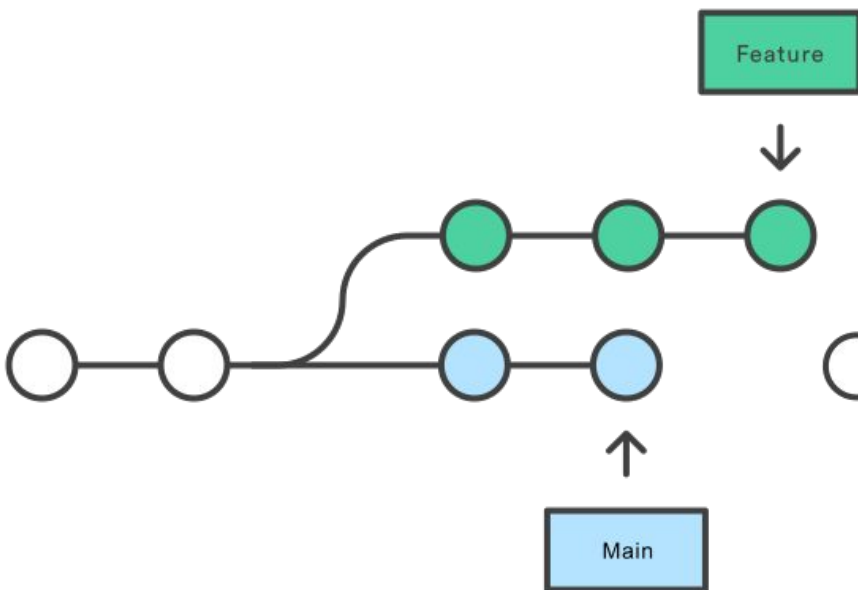
Merging main into the feature branch



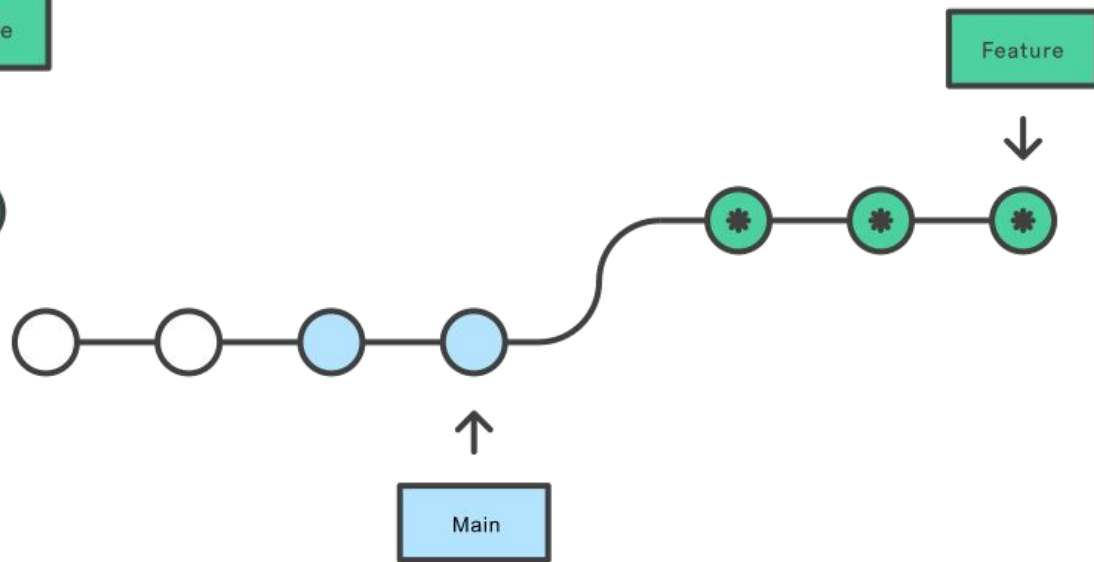
* Merge Commit

git rebase Main (depuis Feature)

A forked commit history



Rebasing the feature branch onto main



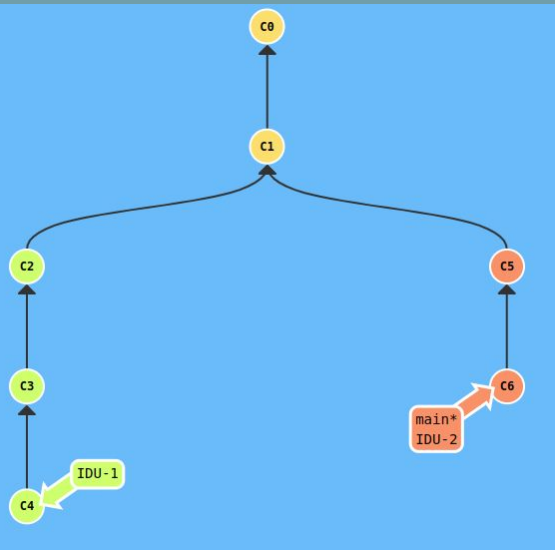
* Brand New Commit

Simulation d'un projet avec 2 branches

```
Apprenez Git Branching
Cacher les cibles  ↗ Level Introduction aux commits avec Git  Instructions

$ level intro1
$ hint
Il suffit de saisir 'git commit' deux fois pour réussir !

$ delay 2000
$ show goal
$ git branch IDU-1
$ git branch IDU-2
$ git checkout IDU-1
$ git commit
$ git commit
$ git commit
$ git checkout IDU-2
$ git commit
$ git commit
$ git checkout main
$ git merge IDU-2
En avance rapide...
```



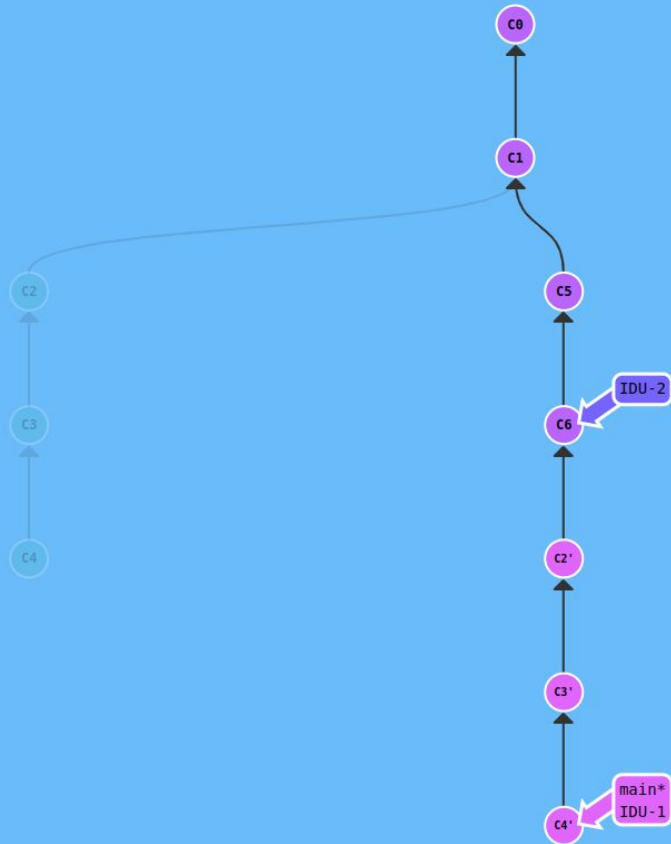
Création de deux branches IDU-1 et IDU-2. Commits sur IDU-1 et IDU-2 puis merge de IDU-2 sur main.

```
Apprenez Git Branching
Cacher les cibles  Level Introduction aux commits avec Git Instructions

$ level intro1
$ hint
Il suffit de saisir 'git commit' deux fois pour réussir !

$ delay 2000
$ show goal
$ git branch IDU-1
$ git branch IDU-2
$ git checkout IDU-1
$ git commit
$ git commit
$ git commit
$ git checkout IDU-2
$ git commit
$ git commit
$ git checkout main
$ git merge IDU-2
En avance rapide...

$ git checkout IDU-1
$ git rebase main
$ git checkout IDU-2
$ git checkout main
$ git merge IDU-1
En avance rapide...
```



On récupère le nouveau main sur IDU-1 puis on merge IDU-1 sur main.

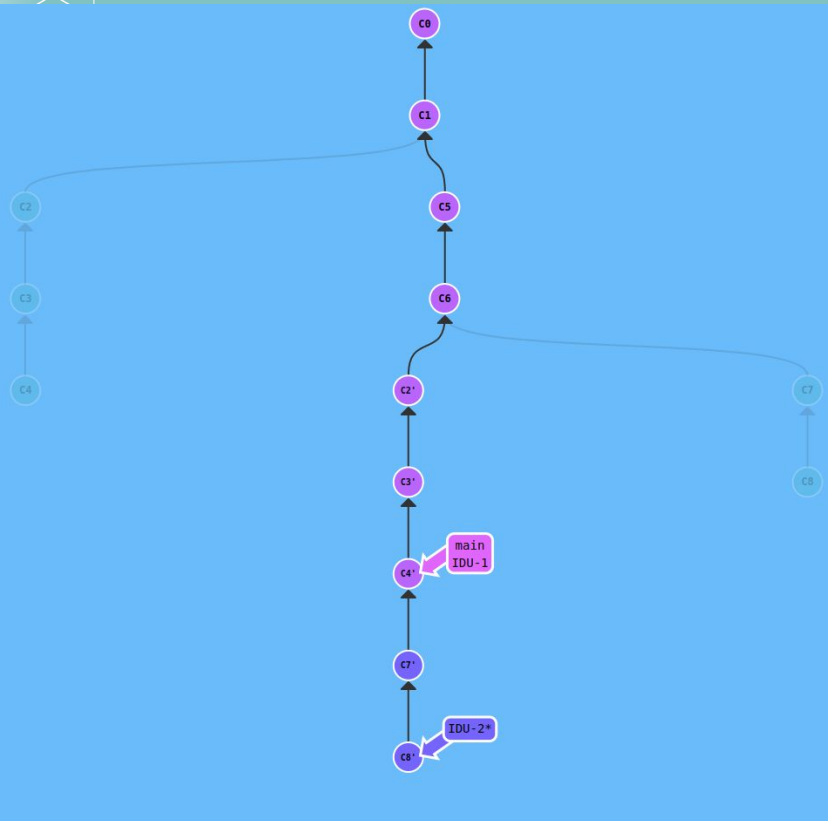
```
Apprenez Git Branching
Cacher les cibles  ↑ Level Introduction aux commits avec Git  Instructions

$ level intro1
$ hint
Il suffit de saisir 'git commit' deux fois pour réussir !

$ delay 2000
$ show goal
$ git branch IDU-1
$ git branch IDU-2
$ git checkout IDU-1
$ git commit
$ git commit
$ git commit
$ git checkout IDU-2
$ git commit
$ git commit
$ git checkout main
$ git merge IDU-2
En avance rapide...

$ git checkout IDU-1
$ git rebase main
$ git checkout IDU-2
$ git checkout main
$ git merge IDU-1
En avance rapide...

$ git checkout IDU-2
$ git commit
$ git commit
$ git rebase main
$
```



**On se remet sur IDU-2
qui est basé sur un
ancien commit du
main... On fait quelques
commits puis on rebase
le main sur IDU-2 pour
récupérer les commits
créés par IDU-1.**

The background features a teal-to-purple gradient with abstract geometric patterns of hexagons and lines. Some hexagons are solid, while others are outlined with small teal dots at their vertices. The title 'Manipulation des commandes git' is centered in a bold, white, sans-serif font.

Manipulation des commandes git

https://learngitbranching.js.org/?locale=fr_FR



Les projets d'info 734

Les projets sont disponibles ici:

<https://github.com/MrWormsy/CoursPolytech/blob/main/INFO734/Projets/PROJETS.MD>

Pour le moment il y a une description sommaire mais je vais mettre ça à jour...



Récapitulatif du projet

Projet seul, à deux ou à trois

Gestion de projet et travail en équipe.

Git / Gestion de projet / Communication

3 TP's (12h) et 2 CM's (3h)

Environ 13h (minimum) de travail et 2h de restitution.

Développement / Gestion de projet / Communication

6+ projets

6 projets plus ou moins compliqués et qui demanderont plus ou moins de compétences.

Gestion de projet / Communication

Avec les technologies qui vous conviennent

Frontend : EJS, pure HTML, React, Next.js, Flutter, Angular...

Backend: NodeJS, Python, Go, Java, Elixir...

Développement

Après le CM 4

M'envoyer un mail (antonin.martin.rosa@gmail.com) avec votre projet

Si vous choisissez un projet de la liste n'envoyez que votre équipe avec le nom du projet et un lien vers un répo git (publique) et un gestionnaire de projet. Si projet "perso", m'envoyer ces choses avec une description du projet (si il n'est pas recevable je vous donnerai une réponse pour essayer de le "corser" un peu).

Pour le CM 5 (distanciel)

<https://meet.google.com/usg-ruco-toz>

Faire une mini présentation de l'avancement du projet

Les choses que vous avez faites: capture d'écran de votre gestionnaire de projet et de l'architecture de votre projet avec énonciation des technologies utilisées.

Pour le CM 6

Faire une mini présentation du projet

Les choses que vous avez faites, les obstacles rencontrés, capture d'écran du gestionnaire de projet, mini démonstration du projet devant la classe.

[#5] – facile – Créer un site pour commander des Pizzas et historique de commandes

Création d'une application Fullstack pour commander des Pizzas (ou autre) en étant connecté à un compte, il y aura un formulaire complet pour créer son compte et commander des pizzas. Il y aura un compte admin qui permettra de voir les commandes de pizzas pour les réaliser (fictivement). L'admin pourra ainsi accepter la commande ou la refuser et l'utilisateur pourra suivre l'état de sa commande depuis son interface.

- Gestion de compte utilisateur avec mot de passe et avec session
- Avoir des permissions sur les utilisateurs ou groupes pour pouvoir gérer les utilisateurs si besoin
- Faire une interface pour commander UNE ou PLUSIEURS pizza(s)
- Les pizzas commandées sont envoyées à un administrateur pour qu'il valide ou non la commande
- L'utilisateur pourra suivre ses commandes depuis son interface et voir son historique de commande
- Optionnel : Vous pouvez faire des stats sur la pizza la plus achetée ou faire un système de recommandation

[#4] – facile – Créer un site de statistiques sur des chaines Youtube

Application pour récupérer des informations sur des chaines Youtube (que les vidéos dans un premier temps) grâce à l'API de Youtube et montrer ces informations dans l'interface sous la forme qui vous conviendra le mieux (dashboard, tableau,

ect). Les données récupérées doivent être persistées dans une base de données pour ne pas devoir refaire des requêtes. Les requêtes et les traitements associés doivent tourner si possible en arrière-plan quand on demande une requête. Vous pouvez créer un système de status pour savoir quelles sont les chaînes en cours de traitement.

- Créer une interface pour mettre le lien d'une chaine youtube pour commencer le traitement de cette chaîne
- Créer une interface pour lister toutes les chaines Youtube qui ont été traitées
- Créer une interface pour voir les informations recueillies d'une chaine Youtube
- Créer un système de mises à jour des données déjà présentes avec possibilité de faire du versioning, par exemple, on récupère les données aujourd'hui et demain et on veut voir combien de vues supplémentaires sont présentes depuis la dernière fois

[#2] – Intermédiaire – Création d'un système de message, style messenger avec gestion de compte utilisateur

Application pour créer un système de messagerie où un ou plusieurs utilisateurs peuvent discuter dans des salons (utiliser le principe de rooms pour SocketIO). Faire un replica de Facebook Messenger...

- Gestion de compte utilisateur (avec ou sans mot de passe) avec session
- Il sera possible de récupérer une conversation entre un OU plusieurs utilisateurs à l'aide d'une base de données "froide" pour persister les données
- Il y aura une interface avec plusieurs pages, une pour la "connexion", une page d'accueil avec ce que fait le projet et une page où se trouve les conversations
- Il faut que les messages puissent être reçus en temps réel (avec SocketIO par exemple)

[#3] – Intermédiaire – Créer un replica de stackoverflow (ou reddit ou twitter) qui utilise un système de threads avec des commentaires

Application pour créer un forum qui est composé d'un sujet ou thread et où des utilisateurs peuvent y répondre avec des commentaires, ils peuvent aussi les modifier ou les supprimer.

- Gestion de compte utilisateur avec mot de passe et avec session
- Avoir des permissions sur les utilisateurs ou groupes pour pouvoir gérer les utilisateurs si besoin
- Faire en sorte de pouvoir créer des threads avec un titre et une description
- Des utilisateurs pourront commenter sous ces threads et réagir
- Si possible utiliser SocketIO pour notifier l'utilisateur que quelqu'un a répondu à son thread ou à son commentaire

[#6] – Complexe – Faire un système de ticket de support avec notifications

Création d'une application pour créer des tickets de support qui auront des états (ouverts, fermés, en cours, ect) et qu'un admin pourra actualiser et mettre à jour. Dès que le status d'un ticket est mis à jour alors une notification est envoyé à l'utilisateur.

- Gestion de compte utilisateur avec mot de passe et avec session
- Avoir des permissions sur les utilisateurs ou groupes pour pouvoir gérer les utilisateurs si besoin
- Créer une interface pour créer un ticket avec au minimum les champs suivants : titre, description, status, priorité, type et dateDeCreation
- Créer une interface pour visualiser SEULEMENT ses tickets ou tous les tickets si on est un admin
- Créer une interface pour mettre à jour l'état d'un ticket
- Optionnel : Vous pouvez utiliser SocketIO pour envoyer des notifications à l'utilisateur qui a créé le ticket si son état a changé et des notifications à l'administrateur pour chaque mise à jour de ticket et lors de la création de ticket

[#1] – Complexe – Visualisation sous forme de dashboard d'une capture Stream Twitter

Création d'un projet qui permet de récupérer des tweets via l'API de Twitter en mode streaming sur un sujet, il y aura un engine pour la collecte de tweet qui opérera sur le backend ou sur une autre application. Ces tweets seront stockés dans une base de données et vous pourrez les retrouver via une interface.

Puis en rapport avec le projet de DATA732 vous allez créer un dashboard qui montrera les données de Twitter sous forme de diagrammes, de nuages de mots, de courbes et pourquoi pas de graphs en temps réel. Si vous avez des questions je peux vous aiguiller.

- Capture Twitter depuis son compte sur un mot clef
- Création d'un dashboard pertinent pour montrer ces données
- Utilisation d'une base de données "froide" pour persister les données
- Utilisation d'une websocket (genre SocketIO) pour envoyer les données récupérées lors d'une stream vers le dashboard