SOUTENANCE POEI
INGÉNIEUR LOGICIELS C++ :

Projet de GPS
TraceMyPlane

IDENTIFIANTS DU DOCUMENT	
Auteur	Aurélien PLAZZOTTA
Date de création	2023-01-12
Dernière mise à jour	2022-01-16
Status	Travail en cours
Version	0.0.1
License	ISC

SOMMAIRE

I. ANALYSE DU PROJET

1. FEUILLE DE ROUTE

VISION TECHNIQUE

roadmap

2. SPÉCIFICATION FONCTIONNELLES

1. L'utilisateur s'inscrit via une page d'inscription et fournit des renseignements personnels pour obtenir des identifiants. Un email de confirmation est envoyé à l'adresse électronique indiquée pour vérifier sa validité.

2. L'utilisateur enregistré s'authentifie via un module d'identification avec courrier électronique et mot de passe pour accéder aux contenus de l'application.

3. L'utilisateur accède à une page d'accueil contenant une liste d'aéroports, des avions en vol, une carte et une fenêtre de supervision des ressources matérielles sollicitées par l'application.
Il peut effectuer une recherche par aéroport pour afficher tous les avions en vol ayant décollé de cet aéroport ou ayant pour destination celui-ci.

4. Une carte géographique est affichée sur la page d'accueil et permet de suivre en temps réel l'affichage des avions ainsi que la position des aéroports.

- Une **version 2.0** de l'application permettra d'afficher des données supplémentaires comme l'altitude, la vélocité ou la direction de l'appareil.

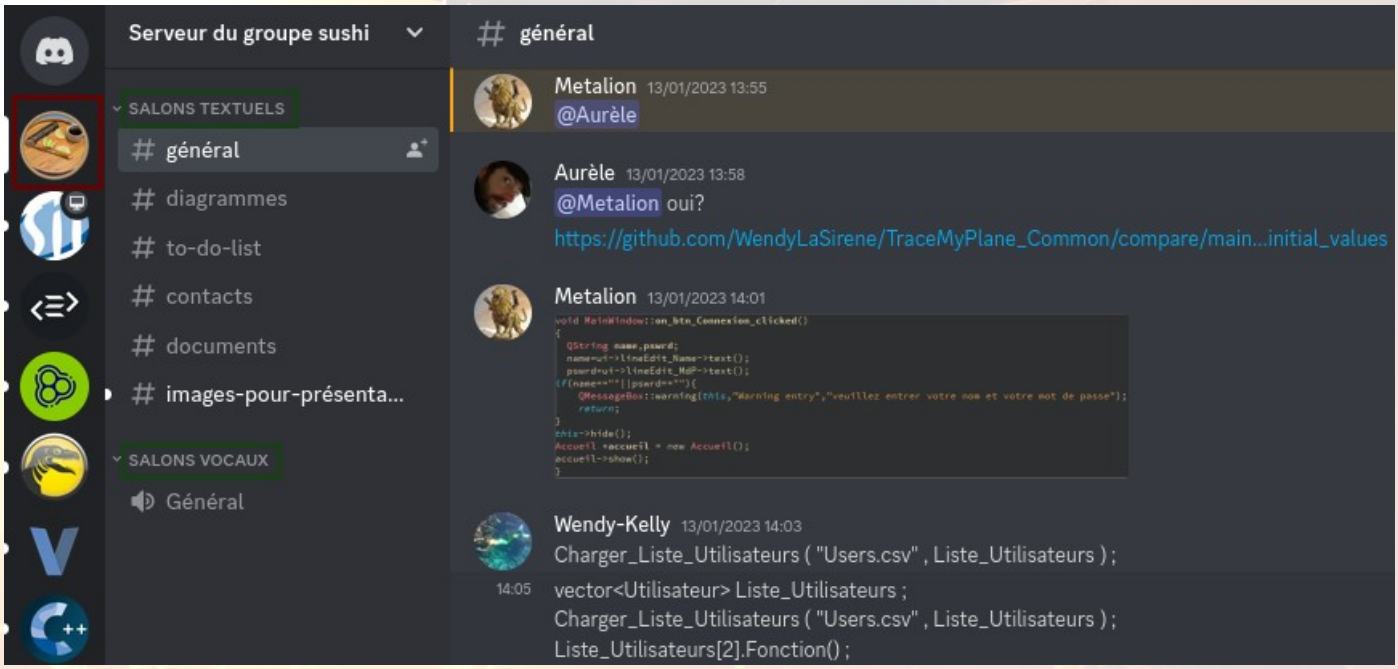
3. VISION TECHNIQUE

3.1 COMMUNICATION INTERNE

Il fut dès le démarrage du projet décidé d'employer un outil simple à prendre en main pour échanger des messages textuels, audio et partager des fichiers dans un canal de discussion privé,
Microsoft Teams n'offrant pas une granularité de contrôle assez fine et s'avérant assez instable.

1. **discord.com**: un serveur créé et administré par François Lecointe pour assurer 24-7 la communication à la fois textuelle et audio entre tous les membres de l'équipe.

Le site web, accessible sur ordinateurs, tablettes et téléphones portables permet également le partage d'écran, facilitant ainsi la prise de décision commune ou la démonstration spontanée d'une avancée majeure ou d'un point de blocage.



2. **Notion.so**: un concurrent direct du nom de Wondershare EdrawMax fut d'abord préssenti à ce rôle mais son manque d'ergonomie décida l'équipe à opter pour Notion.so. Limité en taille d'éléments (jusqu'à 1000, ce qui peut être assez rapidement atteint). Il propose un outil de liste à tâches assez robuste et expressif.

Un deuxième outil permettant la mise à jour commune d'un document partagé et facile d'accès devait permettre aux membres de l'équipe d'établir simplement et de manière intuitive une liste de tâches, ce que discord ne permet pas dû à la modification mono-utilisateur d'un fichier.

To-do-list

All | Board | Mine | Upcoming +

	Task name	Assign	Due	Tags	Description
☑	Road map	Aurélien Plazzotta	January 10, 2023		Recherche documentaire dans d'anciens projets persos
☐	Cadrage du projet	francois.lecointe18@gmail.com wendy.szyskowski@orange.fr Aurélien Plazzotta	January 10, 2023		Priorité à définir
☑	Modélisation MCD	francois.lecointe18@gmail.com	January 10, 2023	JMerise	
☑	Modélisation: cas d'utilisation	Aurélien Plazzotta	January 10, 2023	Umbrello diagrams.net	
☑	Dev: création des classes	wendy.szyskowski@orange.fr	January 11, 2023		
☑	Dev: Lecture des csv	wendy.szyskowski@orange.fr			
☐	Dev: fenêtre Login				
☐	Dev: écran principal				
☐	Dev: affichage de la liste "Aéroport"				
☐	Dev: affichage de la liste "Vols"				
☐	Dev: sélection d'un aéroport-> affichage des vols au départ et à l'arrivée				
☐	Dev: sélection d'un vol-> affichage des aéroport départ/checkpoint/arrivé				
☐	Test: unitaires				Biblitothèque externe?
☐	Test: cas d'utilisation				
☐	Documentation OPEN	francois.lecointe18@gmail.com wendy.szyskowski@orange.fr Aurélien Plazzotta	January 27, 2023		
☑	Déploiement sur github	Aurélien Plazzotta			
☐	GUI pour système d'authentification			Qt SDL2 openGL	
☑	Maquette	francois.lecointe18@gmail.com			
☑	Interface graphique Login	francois.lecointe18@gmail.com			
☐	Interface graphique Utilisation	francois.lecointe18@gmail.com			
☑	Normalisation compilation/déploiement	Aurélien Plazzotta	January 13, 2023	CMake	
☐	Diagrammes SysML	Aurélien Plazzotta		gaphor	Séquence, définition de block, état-machine

Le "no man's land" au milieu de la capture d'écran met en exergue un problème de communication entre les membres. Comment assigner une tâche qui requiert plusieurs sous-tâches?

La formulation de cette question a permis de comprendre qu'il était nécessaire de **sub-diviser le travail** en objectifs au périmètre fonctionnel et à la complexité moindre.

II. DÉPLOIEMENT VIA GIT ET GITHUB

1. ABSTRACT

Tout projet professionnel de génie logiciel implique plus d'un intervenant. Différents acteurs aux coeurs de métier différents interviennent selon des périmètres variés au sein du projet dépendemment de leur rôle, grade et attentes en termes de valeur ajoutée.

Qu'il s'agisse de produire un système, soit un logiciel qui s'adresse à d'autres logiciels ou un applicatif, dont l'utilisateur final est un opérateur humain, la complexité est telle qu'il est nécessaire de faire appel à des outils de travail collaboratif afin d'absorber cette complexité technique et humaine et garantir les chances de succès du sus-dit projet.

Le déploiement en environnement de productif n'est qu'un symptôme; le criterion du succès est la valeur utile rendue accessible aux professionnels métier l'exploitant dans le cadre de leur opérations quotidiennes afin de remplir leur rôle en société.

Le cycle de vie normal du projet requiert l'élaboration de phases en amont du projet avant sa mise en production, savoir:

- Les spécifications techniques
- Analyse
- Conception
- Implementation
- Tests
- Documentation
- Déploiement

Après son déploiement en environnement de production livré chez client, ou rendu accessible à distance pour lui et ses collaborateurs, le projet logiciel entre alors en phase de maintenance évolutive.
Les besoins métiers du client peuvent et changeront sans doute, et le logiciel subit régulièrement des nouvelles itérations logicielles incluant une nouvelle fois les étapes susmentionnées.

Un logiciel est donc un projet vivant qui regroupe de nombreux protagonistes aux jeux de compétences, objectifs et besoins très variés. C'est pourquoi il est nécessaire de rendre le code source du logiciel disponible auprès de différentes équipes travaillant en parallèle, qu'il s'agisse d'un même site physique ou d'équipes délocalisées sur plusieurs continents et oeuvrant sur des fuseaux horaires variés.

2. RATIONALE

A ce motif, des outils de gestion de projet collaboratif et de contrôle de version ont vu le jour au début du XXIe siècle afin de palier aux besoins des programmeurs pour opérater concomitamment sur un projet en constante évolution.
Le logiciel est un flux interrompu de mises à jour et de corrections qui impose un cadre strict de gestion des accès et de politique de modification et d'historisation, afin que chacun puisse contrôler les choix d'implémentation technique ainsi que la direction générale vers laquelle s'oriente le projet.

Après des logiciels comme BitBucket (toujours actif sur le marché), Mercurial et Subversion (qui sont tous des deux des outils de gestion de version centralisés, avec les problèmes que cela implique; un homme du nom de Linus Torvalds, auteur du noyau GNU/linux, lui même basé sur Minix, fruit des efforts de Andrew S. Tanenbaum, publie en ligne une première version à source ouverte et gratuite d'un nouvel entrant sur le marché en 2005: git.

Git est un système de version de contrôle distribué permettant d'assurer l'intégrité d'échanges de données et flux de travaux non-linéraires.

Il est accessible en ligne de commande et s'avère très utile pour normaliser le développement du projet logiciel, y compris pour des projets qui seraient menés par une seule personne.

3. ADOPTION DE GIT

Git est adopté par l'équipe en raison de son immense popularité, tant à la fois pour les projets open source individuels que professionnels, et au sein de l'industrie civile et de la sphère académique.

Ses barrières à l'entrée sont faibles:

- ses documentations gratuites et payantes sont nombreuses,
- il existe une très grande communité active de programmeurs du monde entier, qui s'avère disponible pour renseigner et aider 24-7 sur les canaux IRC et discord.com.
- son utilisation est gratuite (financièrement).

Git est un outil qui incarne le socle fondamental de nombreux acteurs sur le marché de la gestion de projets logiciels communautaires.

3.1 NOTATION BACKUS-NAUR

Un metalanguage a été défini par le mathématicien nord-américan du XXe siècle Backus et le mathématicien perse du XIe siècle Naur. Il permet la communication d'expressions logico-mathématiques couramment usitées et est adoptée dans le présent document afin de faciliter la transmission d'idées riches avec une densité d'informations plus élevée.

En voici les cinq éléments fondamentaux:

Symbol	Definition
::=	Défini comme suit
|	OU logique
< >	concept (*i.d. nom d'objet, valeur*)
[ ]	possible
{ }	*item* à définir

3.2 CONFIGURATION

Il existe trois options de configuration impactant le périmètre de git sur la machine:

1. `--system` : modifie la configuration pour tous les utilisateurs ayant un compte sur la machine physique
2. `--global` : le paramètre est modifié pour le compte utilisateur actuel
3. `--local` : seul le dépôt actuel bénéfie de cette modification

Les priorités sont définies du plus restreint au plus général (*e.g.* `--local` supplante --system).

Nota Bene: les examples de commandes qui suivent utilisent l'option "`--global`" à titre purement indicatif, libre à vous d'utiliser `--system` ou `--local` selon vos besoins.
Les 3 options pouvant être cumulées sur une même machine, au travers de multiples dépôts.

Dans le but de préparer l'espace de travail à la connexion et l'alimentation d'un dépôt distant (*i.e.* "remote") et ainsi partager les nouvelles itérations du code source à toutes les personnes impliquées dans le développement.
Il est requis de renseigner au minimum un nom et une adresse électronique pour identifier son compte utilisateur.

1.

Saisir et valider avec la touche [Return]:

```
git config --global user.name <user_name>
git config --global user.email <email>
```

```
→ git config --global user.name aurèle

~
→ git config --global user.email aurelien.plazzotta@tutanota.com
```

Nota bene:

D'autres options peuvent être personnalisées:

- un éditeur de texte: par défaut, lancé automatiquement lorsque vous omettez le paramètre `-m <message>` lors de l'exécution d'une commande **`commit`**.
- Un paginateur peut être défini pour remplacer les commandes **`less`** et **`more`**.
- Un nom de branche par défaut pour vos futurs créations de dépôts.
- Des alias pour faciliter la saisie de commandes couramment employées
- Une signature électronique (pour utiliser implicitement l'option `-S` de la commande **`commit`**

2.

3.1.3 Pour afficher la liste de vos paramètres personnalisés, saisir:

```
git config --list
```

```
➜ git config --list
user.name=aurele
user.email=aurelien.plazzotta@tutanota.com
user.signingkey=ssh-ed25519 AAAAC3NzaC1lZDI1NTE5AA
init.defaultbranch=master
alias.last=log -1 HEAD
alias.st=status -s
core.editor=nvim
core.pager=nvimpager
color.pager=true
commit.gpgsign=false
gpg.format=ssh
pull.ff=only
push.autosetupremote=true
push.default=simple
```

Nota bene 2:

Pour créer un alias, et faciliter l'emploi d'une commande régulièrement usitée:

```
git config –global alias.<alias_name> <command>
e.g. git config –global alias.st ”status -s”
```

Nota bene 2:

L'option **`--show-origin`** vous permet d'afficher en sus, le chemin d'accès de votre fichier de configuration globale **`git`**.

```
➜ git config --list --show-origin
```

4. CRÉATION D'UN DÉPÔT

Un espace de travail local (sur la machine physique du programmeur), s'appelle un dépôt. Chaque dépôt correspond donc à un système à developper et maintenir, ou même à un sous-système selon la complexité du projet.

1.

Création du dossier de travail en local

```
~/dev/SII
→ mkdir TraceMyPlane; cd TraceMyplane
```

2.

Initialisation du dépôt git pour démarrer le pistage des fichiers. Cette commande crée un sous-dossier .git contenant les fichiers de fonctionnement interne à `git` lui-même.

`git init`

```
dev/SII/TraceMyPlane
→ git init
Initialized empty Git repository in /home/aurele/dev/SII/TraceMyPlane/.git/

TraceMyPlane on  master
→
```

3.

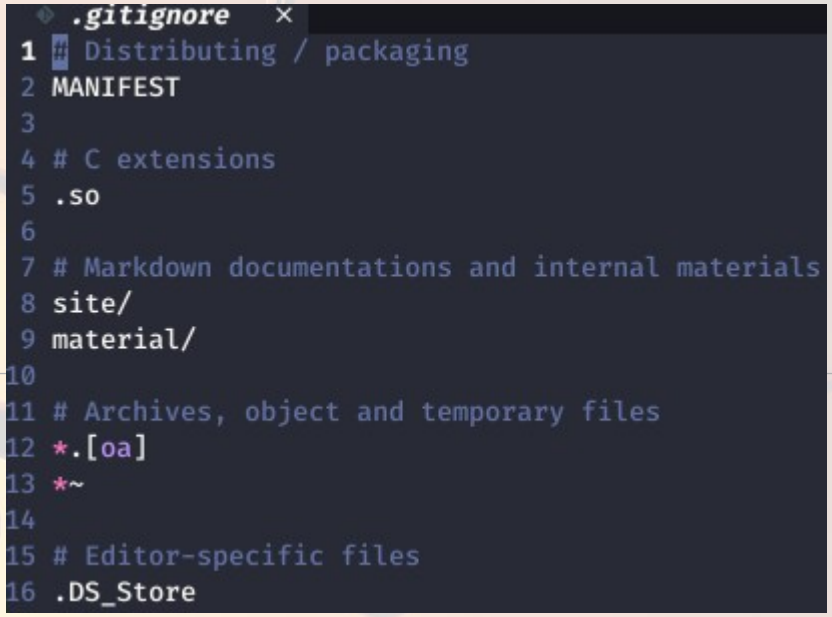
3. Préparation du projet avec création des fichiers administratifs et logistiques:

- `LICENSE`: la license retenue est **ISC**. Originellement utilisée par l'Université de Berkeley en California, elle fût depuis adoptée par le projet de système d'exploitation OpenBSD. Elle est approuvée par la fondation Open Source Iniative, consortium qui supervise, promeut et règlemente les licences open sources à l'échelle internationale.

  La license ISC est recommandée par les mainteneurs du système d'exploitation FreeBSD pour les nouveaux projets et protège la propriété intellectuelle des entrepreneurs de la vampirisation de la base de code face aux très grandes entreprises et conglomérats qui voudraient absorber la substance du projet pour l'intégrer à ses solutions payantes et propriétaires.

- `README.md` : le fichier "Lisez-moi" au format Markdown. Il présente le projet et récapitule son contenu, son rôle et ses aspirations. Devenu au fil du temps plus publicitaire que technique, il permet de mettre en avant les avantages d'un projet et de susciter l'intérêt dans l'esprit des visiteurs désireux d'exploiter une nouvelle solution logicielle qui répond mieux à leurs besoins actuels, ou même de nouveaux contributeurs en quête de défis techniques cherchant améliorer leurs compétences en programmation et/ou gestion de projet (selon les flux de travail autorisés par le créateur).

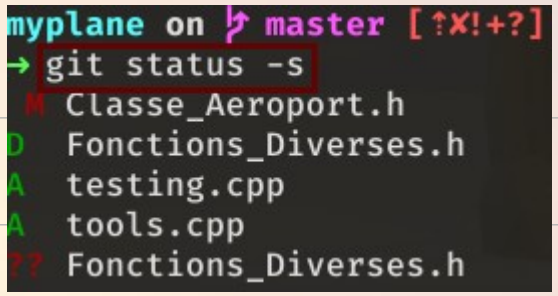
- `.gitignore`: le fichier caché ".gitignore" permet de lister des fichiers, répertoires et ou une nomenclature des deux précédents devant être ignorés par la fonctionalité de pistage de git. Cela permet d'éviter de téléverser au dépôt distant (hébergé en-ligne) des fichiers inutiles aux utilisateurs finaux du projet (*e.g.* dossiers de tests internes, fichiers temporaires, recherches documentaires utiles seulement aux contributeurs dans le cadre du développement du projet ou de bibliothèques partagées.



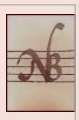
4.

Contrôler l'état actuel du dossier de travail:

`git status [-s]`



Nota bene:
L'option `--s` (pour "short") permet d'augmenter la densité de l'information afin de réduire la pollution informationnelle.

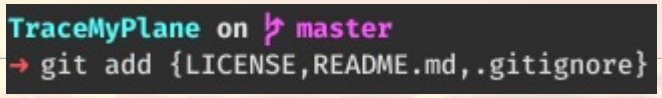


Lexikon:
- **Rouge**: action utilisateur requise
- **Vert** : nouvel état valide
- **??** : non pisté
- A : ajouté
- M : modifié

5.

Ajouter des fichiers à pister par `git`:

`git add <file(s)>`



6. Soumettre du code dans le dépôt local:

```
TraceMyPlane on  master [+]
➜ git commit -m "Adding LICENSE, README.md and .gitignore files"
[master (root-commit) d4df0b6] Adding LICENSE, README.md and .gitignore files
 3 files changed, 52 insertions(+)
 create mode 100644 .gitignore
 create mode 100644 LICENSE
 create mode 100644 README.md
```

`git commit -m <message>`

Nota bene:

L'option `--a` (pour "all") permet d'outrepasser la zone tampon (étape inderméditaire "stage").
Tous les nouveaux fichiers sont automatiquement ajoutés puis soumis ("commit").
La perte d'un filet de sûreté s'accompagne d'un gain de temps et évite les omissions.

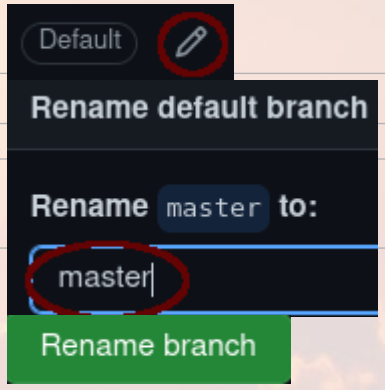
[RENOMMER UNE BRANCHE PAR DÉFAUT]

1. Rendez-vous sur la page suivante: (pensez à modifier le nom d'utilisateur), dans l'onglet "Overview".

https://github.com/kenaryn/TraceMyPlane/branches Overview

2. Cliquez sur l'icône en forme de stylo.

3. Saisir le nouveau nom de branche par défaut puis cliquez sur le bouton "Rename branch" pour confirmer l'action.



4. Actualisez la nouvelle branche du dépôt distant vers votre espace de travail local:

Commande	Rôle
`git branch -M <branch>`	Créer une nouvelle branche, écraser le nom de l'actuelle et basculer dessus.
`git fetch origin`	Configure le flux pour rappatrier les données localement

	depuis le dépôt distant "origin"
`git branch -u <upstream>/<remote_branch> <local_branch>`	Mettre à jour le pistage de référence de la branche distante master du dépôt "origin" depuis la branche locale locale "master"
`git remote set-head origin -a`	Actualiser le pointage du curseur vers le nouveau nom

```
➜ git branch -M master
➜ git fetch origin
➜ git branch -u origin/master master
➜ git remote set-head-origin -a
```

5. CHOISIR SA SOLUTION POUR PARTAGER SES TRAVAUX VERS UN DÉPÔT DISTANT

Partager ses travaux vers un dépôt public ou privé pour rendre disponible l'instant des fichiers d'en-tête, les fichiers source, les images et les documentations liées à la compréhension du projet requiert l'hébergement de ces données vers un serveur web disponible en permance.
La location des serveurs leur administration peut être à la fois coûteux en temps et en argent. C'est pourquoi une solution gratuite d'hébergement dédiée à la gestion décentralisée des flux de travaux collaboratifs s'est imposée à l'équipe.

De plus, les différents acteurs du marché intègrent tous une interface graphique utilisateur pour aider à l'usage des solutions d'hébergement pour les utilisateurs les moins avertis, et ainsi faciliter son adoption auprès du plus grand nombre, renforçant par la même occasion la continuité d'activité du service, et *in fine*, la base d'utilisateurs actifs à même d'en assurer la promotion et pourquoi des documentations liées à la plate-forme.

Le marché présente de nombreuses solutions directement concurrentes:

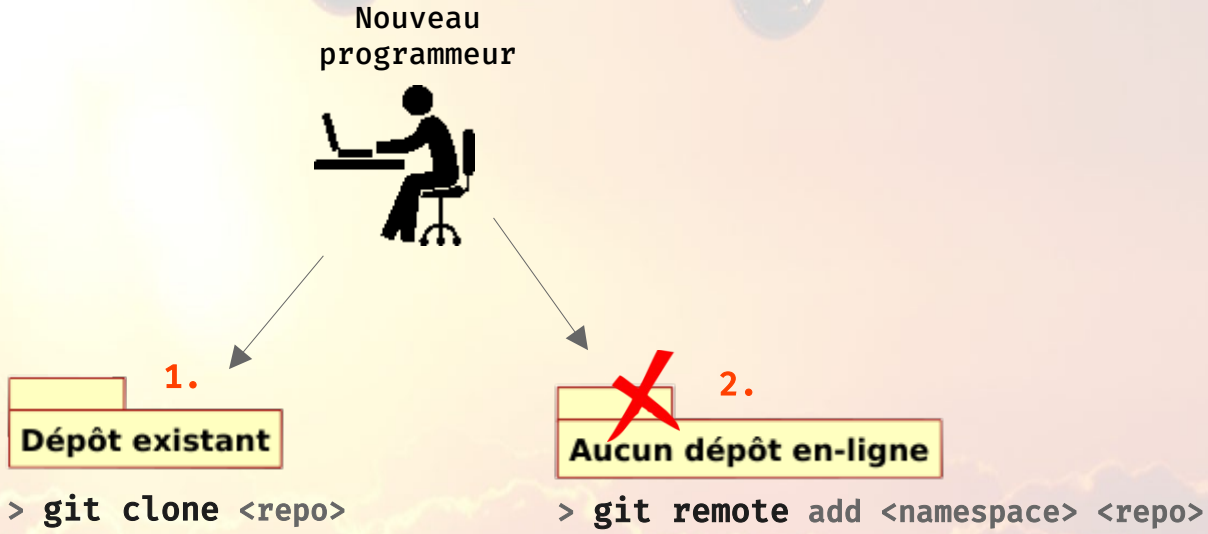
Solution	Site web officiel	Particularités
Gitea	gitea.io/en-us	Léger, facile à prendre en main
Fossil	fossil-scm.org/	Executable portable, stocke ses fichiers sous SQLite, préserve le véritable historique
Gogs	gogs.io	S'execute sur tout environnement supportant le Go, faible consommation de resources matérielles
* **Github**	github.com	Gigantesque base d'utilisateurs

Gitlab	gitlab.com	Intègre nativement des outils de développement/integration continus, métriques avancés
Bitbucket	bitbucket.org	Premier acteur historique, nombreuses offres pro, intègre Jira
Sourcehut	sr.ht	Granularité fine des accès utilisateurs et des dépôts
Gitly	gitly.org	Ecrit en V, expérimental, extrêmement léger, fonctionne sans JS, rapide

Le choix de l'équipe s'est porté sur **github** en raison de la connaissance préalable du service de certains membres de l'équipe.

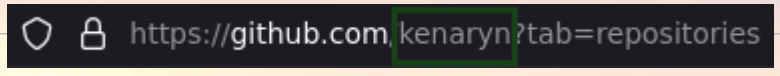
6. TÉLÉVERSEMENT DES TRAVAUX VERS UN DÉPÔT DISTANT

Deux scénarii se présentent à chaque nouveau membre d'une équipe de développement:



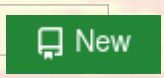
1. A

Le dépôt n'existe pas. Créer d'un nouveau dépôt vierge distant ("remote", sur un réseau en-ligne) *via* l'interface graphique de l'hébergeur sur la page des dépôts du compte **github.com**



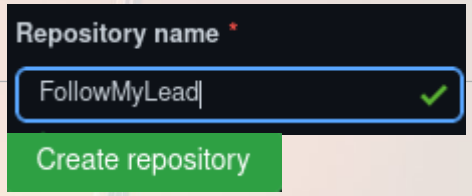
2.

Cliquez sur le bouton vert "New"



3.

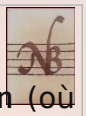
1.3 Définir un nom de dépôt et valider



1. B

Le dépôt existe déjà. Télécharger son contenu sur sa machine locale *via* la commande git clone:
git clone git@github.com:<user>/<repo_name> [--depth=n] [<custom_repository_name>]

```
~/dev/SII
➜ git clone git@github.com:WendyLaSirene/TraceMyPlane_Common.git sii_project
Cloning into 'sii_project'...
Enter passphrase for key '/home/aurele/.ssh/id_ed25519':
remote: Enumerating objects: 51, done.
remote: Counting objects: 100% (51/51), done.
remote: Compressing objects: 100% (31/31), done.
remote: Total 51 (delta 19), reused 51 (delta 19), pack-reused 0
Receiving objects: 100% (51/51), 72.39 KiB | 692.00 KiB/s, done.
Resolving deltas: 100% (19/19), done.
```

Nota bene:

l'option **--depth** permet de cloner le dépôt en tronquant tous les commit à l'exception des "n" derniers commit définis en argument de la commande de la forme `--depth=n` (où "n" est un nombre naturel).

5.

1.4 Enregistrer la nouvelle adresse distante dans votre espace de travail pour pouvoir synchroniser localement:

git remote add git@github.com:<user>/<repository.git>

```
TraceMyPlane on  master [⇕?]
➜ git remote add origin git@github.com:kenaryn/TraceMyPlane.git
```

6.

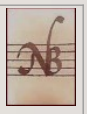
Si au coeur du cycle de vie applicatif, le dépôt distant vient à changer de nom et/ou d'adresse, votre espace de travail doit pouvoir interroger la nouvelle adresse pour y accéder:

git remote set-url --add git@github.com:<user>/<new_url>

```
TraceMyPlane on  master [⇕?]
➜ git remote set-url --add git@github.com:wendy/TraceMyPlane.git
```

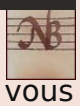
7.

Téléverser la dernière itération des travaux locaux vers le dépôt github.com



Nota bene:
`git` compute le *delta* (différence entre l'existant en local et la version des travaux en-ligne) et téléverse ("upload") les fichiers objets mis à jour.
Un nouveau commit avec son identification par hashage est généré et ajouté à la branche pointée par le curseur.

```
TraceMyPlane on  master [⇡]
➜ git push -u origin master
Enter passphrase for key '/home/aurele/.ssh/id_ed25519':
Enumerating objects: 7, done.
Counting objects: 100% (7/7), done.
Delta compression using up to 4 threads
Compressing objects: 100% (4/4), done.
Writing objects: 100% (4/4), 270.43 KiB | 2.73 MiB/s, done.
Total 4 (delta 1), reused 0 (delta 0), pack-reused 0
remote: Resolving deltas: 100% (1/1), completed with 1 local object.
To github.com:kenaryn/TraceMyPlane.git
   7c5d777..30ff4f3  master -> master
branch 'master' set up to track 'origin/master'.
```



Nota bene:
L'identification via une phrase de passe (plus complexe à déchiffrer qu'un mot de passe vous êtes demandé si vous avez configuré une clé publiqué associée à une clé privée).

Plus d'informations ici: *https://docs.github.com/en/authentication/connecting-to-github-with-ssh/working-with-ssh-key-passphrases*

6.1 METTRE À JOUR LA BRANCHE DE DEVELOPPEMENT PRINCIPALE

Un membre de l'équipe a intégré dans son espace de travail une nouvelle itération du code intégrant l'ajout ou la modification d'une fonctionnalité.
Ce nouveau code est révisé soit lors d'une session de programmation en binôme, soit leur d'une révision technique par un pair. Une fois validé, le gestionnaire du dépôt doit fusionner la branche dédiée à cette fonctionnalité ("feature") avec les travaux principaux (la branche "master").

1.

S'assurer de rappatrier en local la dernière version du dépôt distant:

```
git fetch
origin
```

```
TraceMyPlane on  master [⇕] took 7s
➜ git fetch origin
Enter passphrase for key '/home/aurele/.ssh/id_ed25519':
```

2.

Fusionner les modifications de la branche <feature> pour mettre à jour la branche maîtresse

```
git merge <feature_branch>
```

```
myplane on  master
➜ git merge deployment
Updating fb72003..ceb49b8
Fast-forward
 CMakeLists.txt | 17 ++++++++++++++++++
 1 file changed, 17 insertions(+)
 create mode 100644 CMakeLists.txt
```

Nota bene:
Une autre commande fusionne les modifications à l'instar de **merge:**
`git rebase <base_branch> <topic_branch>`

Toutefois, un unique parent est ajouté au journal de commits, comme si la mise à jour est effectuée de manière séquentielle au lieu de présenter 2 branches traitées de manière parallèle.

6.2 PROBLÈMES DE FUSION

[RESYNCHRONISER SON ESPACE DE TRAVAIL AVEC LE DÉPÔT]

1. Lorsque l'espace de travail local est désynchronisé avec le dépôt distant, toute fusion devient impossible. **git** tente ainsi d'éviter les pertes irrécouvrables de données:
Tenter de fusionner les modifications échoue.

```
TraceMyPlane on  master [⇕]
→ git push -u origin master
Enter passphrase for key '/home/aurele/.ssh/id_ed25519':
To github.com:kenaryn/TraceMyPlane.git
 ! [rejected]        master -> master (non-fast-forward)
error: failed to push some refs to 'github.com:kenaryn/TraceMyPlane.git'
hint: Updates were rejected because the tip of your current branch is behind
hint: its remote counterpart. Integrate the remote changes (e.g.
hint: 'git pull ...') before pushing again.
```

2. Afficher le statut présente un diagnostic de la situation suivi d'un moyen de résoudre le conflit.

```
TraceMyPlane on  (git)-[master|merge]- [⇕=]
→ git status
On branch master
Your branch and 'origin/master' have diverged,
and have 1 and 1 different commits each, respectively.
  (use "git pull" to merge the remote branch into yours)

You have unmerged paths.
  (fix conflicts and run "git commit")
  (use "git merge --abort" to abort the merge)

Unmerged paths:
  (use "git add <file>..." to mark resolution)
        both modified:   design/github_deployment.pdf
```

3. Pour résoudre le conflit, taper:
git add <file>
git commit -m <message>

6.3 GESTION DES BRANCHES

Afficher la liste des dernières soumissions de code selon des critères de date
`git log --oneline --since 1months --until 3days –graph`

```
TraceMyPlane on  master [⇡?]
→ git log --oneline --since 5days --until now --graph
* 45199f0 (HEAD -> master) Creating Classe_Aeroport.cpp and Coeur_Vol.cpp's firt versions
* 595c56b (origin/deployment, deployment) Adding CMakeLists.txt
*   26f5b25 (origin/master, origin/HEAD) Merge remote-tracking branch 'refs/remotes/origin/master'
|\
| * 30ff4f3 Upgrading git/github deployment design document to v0.0.2
* | 1a15188 Upgrading git/github deployment design document to v0.0.2
|/
* 7c5d777 Adding 'git deployment's design document first draft
* 9473851 Adding deployment via git's design document first draft.
* 472ff25 Removing deployment via git.odt
* 8793dc3 Adding deployment via git's design document first draft.
* 4f1dd2d Upgrading use case nominal scenarii to v0.0.2
* b5e29d6 Altering primary use case in the Nominal Use case diagram
```

[CRÉER UNE BRANCHE DEPUIS UNE ITÉRATION SPÉCIFIQUE]

1. Il est souvent utile de créer une branche depuis un commit particulier, notamment lorsqu'une ou plusieurs soumissions de code ont été téléversées dans une branche qui requierait une sub-division pour plus de flexibilité et ou s'assurer d'une non-régression.
La commande fait usage d'un identifiant de hashage, c'est pourquoi vous devez utiliser `git log` au préalable.

`git {branch|checkout –b} <new_branch> <hash_commit>`

```
TraceMyPlane on  master
→ git log --oneline --graph --since 4days
* e6ae0a3 (HEAD -> master, origin/master, origin/hotfix, origin/
* 45199f0 Creating Classe_Aeroport.cpp and Coeur_Vol.cpp's firt
* 595c56b (origin/deployment, deployment) Adding CMakeLists.txt
*   26f5b25 Merge remote-tracking branch 'refs/remotes/origin/ma
```

```
TraceMyPlane on  master
→ git checkout -b tooling 595c56b
Switched to a new branch 'tooling'

TraceMyPlane on  tooling
→ git log --oneline --graph
* 595c56b (HEAD -> tooling, origin/deployment, deployment) Adding CMakeL
```

[SUPPRIMER UNE BRANCHE]

1. Lorsqu'une branche de développement a accompli son rôle, il est judicieux de la supprimer afin de réduire au maximum la pollution informationelle d'une part et les erreurs d'inattention en téléversant dans la mauvaise branche d'autre part.

`git branch -d <existing_branch>`

```
TraceMyPlane on  master
➜ git branch -d tooling
Deleted branch tooling (was 595c56b).
```

Nota bene:

Le notificateur ("flag") -D permet de forcer la suppression d'une branche, même lorsque celle-ci présente des commit non fusionnés avec la branche maîtresse.
`git branch -D <existing_branch>`

III. CMAKE OU NORMALISATION DU DÉPLOIEMENT

1. ABSTRACT

Comme nous l'avons vu précédemment, tout projet de génie logiciel implique de nombreux intervenants, et parmis les programmeurs impliqués dans son développement, il est courant qu'une multitude d'environnements foisonne.
Au sein d'une même équipe, différents aspects techniques impactant directement ou indirectement la pile technologique de l'mainteneurs du projet opère sous différents aspects

Au sein d'une même équipe, la pile technologique des mainteneurs du projet varie selon plusieurs aspects:

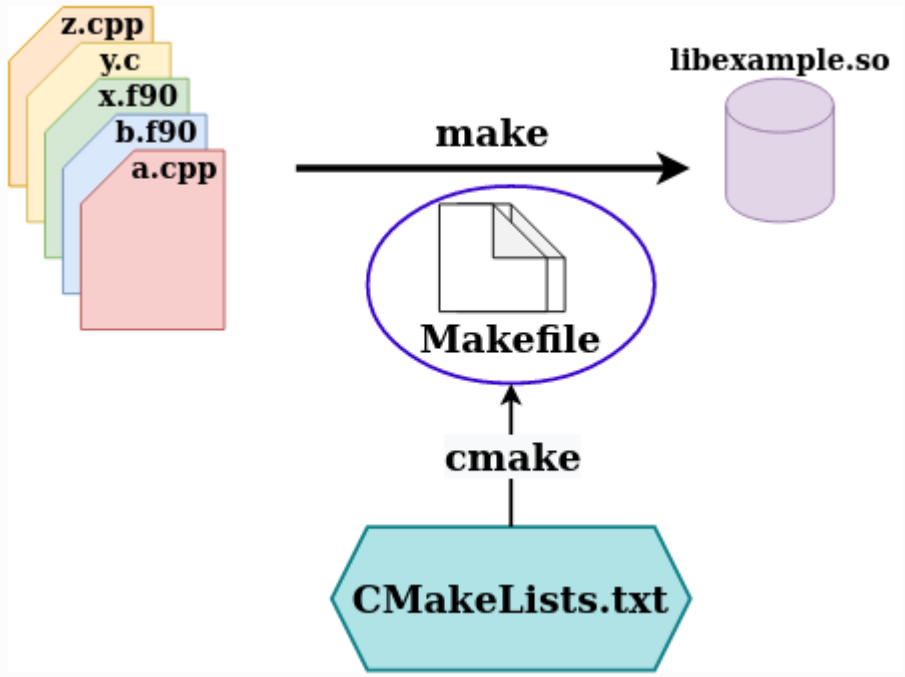
- Le système d'exploitation
- L'environnement de développement intégré / éditeur de texte
- Le compilateur
  - La version
  - Le jeu d'options
- Le générateur de fichiers

Tous ses facettes de l'environnement technique impactent directement le code compilé pour chaque postal de travail. Il est donc nécessaire d'homogénéiser la transformation du code source en binaire exécutable afin d'assurer la collaboration de chacun, et *in fine*, la continuité d'activité du projet.

2. RATIONALE

Comment déterminer quel devra être le processus de compilation général afin de s'assurer que chacun génère le même code? Sans une normalisation de ce procédé, comment savoir qui a produit correctement le binaire executable au sein de l'équipe?
Le code généré par l'environnement hôte du programmeur dépend donc des différents aspects cités plus haut. La conséquence est que chaque programmeur va produire un exécutable binaire différents selon ses propres paramètres et outils, quand bien même la base de code partagée entre les divers fichiers source est strictement identique!

Afin d'éviter les asymétries et les frictions dans le code, et l'incompréhension entre les personnes impliquées directement dans le développement, **manager le processus** de construction (*i.e.* "build", transformation des fichiers sources en fichiers objects, puis reliés statiquement aux bibliothèques externes) **indépendante du compilateur** est essentielle au bon déroulement des phases d'itérations du projet.



Source: *coderefinery.github.io/cmake-workshop/motivation*

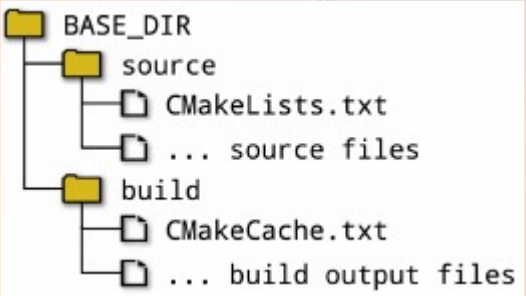
La pile technologique du projet TraceMyPlane:

Membre	Système d'exploitation	IDE/ éditeur	Générateur de fichiers	Compilateur	Options du compilateur
Aurélien	VoidLinux	Neovim	Unix Makefiles	gcc-12.2.0	17 options
François	Windows 10	Visual studio	Visual Studio 2022	MSVC	-std=c++20
Wendy	Windows 7	Embarcado Dev-C++	Visual Studio 2019	gcc-9.2.2	Aucune

Un bref aperçu nous indique qu'aucun membre ne compilera un exécutable ou une bibliothèque partagée identique aux 2 autres, ce qui complique les tests fonctionnels et le déploiement en environnement de production.
Il est donc judicieux de normaliser nos process afin de collaborer en bonne intelligence *via* un générateur de d'outil de construction.

3. ARBORESCENCE DU PROJET

Une bonne pratique dans la définition de l'arborescende du projet permet d'éviter des écueils comme le mélange des fichiers générés par Cmake pour son fonctionnement interne et les fichiers sources directement responsables qui seront transformés en exécutable après compilation.



Source: *Professional CMake de Craig Scott*

4. FICHIER CMAKELISTS.TXT

Le DSL (*i.e.* Domain Specific Language, language spécifique à un métier) CMake requiert un fichier nommé `CMakeLists.txt` dans le sous-dossier `source` pour fonctionner.

Voici le contenu de la version actuelle:

<SCREEN SHOT DE CMAKELISTS.TXT>

5. CONFIGURATION ET GÉNÉRATION DU PROJET

Configurer et générer le projet:

```
cmake -G "Unix Makefiles" ../source -DCMAKE_EXPORT_COMPILE_COMMANDS=1 –fresh
```

```
TraceMyPlane wendy on  main [?]
➜ cmake -G "Unix Makefiles" . -DCMAKE_EXPORT_COMPILE_COMMANDS=1 --fresh
-- The CXX compiler identification is GNU 12.2.0
-- Detecting CXX compiler ABI info
-- Detecting CXX compiler ABI info - done
-- Check for working CXX compiler: /bin/g++ - skipped
-- Detecting CXX compile features
-- Detecting CXX compile features - done
-- Performing Test CMAKE_HAVE_LIBC_PTHREAD
-- Performing Test CMAKE_HAVE_LIBC_PTHREAD - Success
-- Found Threads: TRUE
-- Performing Test HAVE_STDATOMIC
-- Performing Test HAVE_STDATOMIC - Success
-- Found WrapAtomic: TRUE
-- Found OpenGL: /usr/lib/libOpenGL.so
-- Found WrapOpenGL: TRUE
-- Found XKB: /usr/lib64/libxkbcommon.so (found suitable version "1.4.1"
-- Found WrapVulkanHeaders: /usr/include
-- Configuring done
-- Generating done
-- Build files have been written to: /home/aurele/dev/SII_course/TraceMyP
```

Nota bene:

- `-G` : Cmake utilise le générateur de fichiers par défaut d'Unix
- puis configure le projet en recherchant le `CmakeLists.txt` dans le noeud jumeau `source`,
- génère ensuite une représentation complète du projet avec un notificateur supplémentaire permettant de bénéficier du Language de Spécification Server pour tous les fichiers du projet. Le but est l'accès au gestionnaire d'erreur, autocomplétion et documentation multi-fichiers.
- `--fresh`: ensures that the project can be rebuild properly by removing `CMakeCache.txt` and `CmakeFiles` directory.

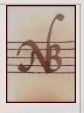
6. COMPILATION

Construire les fichiers objets depuis les fichiers sources et relie les bibliothèques logicielles afin de **compiler** le projet (build + link = compilation).

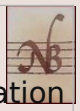
```
cmake --build . --target tracemyplane
```

<SCREEN SHOT DE CMAKE --BUILD .>

Nota bene:

`--target`: notificateur optionnel qui détermine le fichier exécutable à générer en sortie. 

Nota bene:

La commande `--build` permet d'outrepasser l'appel à l'outil `make` et facilite l'automatisation du process en l'intégrant au sein d'un script `[z]shell`. 

6. AVERTISSEMENTS

Le résultat de la compilation produit un retour d'informations précieux permettant l'amélioration de la base de code dans le debut de produire un logiciel plus robuste, plus sûr et/ou plus portable.
Toute la batterie de notificateurs additionnels renforce les exigences du compilateur et ses garanties, et transmet en retour des avertissements et notes qui eurent été ignorés en l'absence de paramètrage du compilateur.

SPECIFICATIONS TECHNIQUES

AJOUTER DIAGRAMMES SysML:
- cas d'utilisation
- définition de bloc
- séquence

```
[ 25%] Building CXX object CMakeFiles/tracemyplane.dir/Coeur_Vol.cpp.o
In file included from /home/aurele/tracemyplane/Coeur_Vol.cpp:7:
/home/aurele/tracemyplane/Fonctions_Diverses.h: In function 'int Ajouter_Sans_Doublon(T, std::vector<T>&)':
/home/aurele/tracemyplane/Fonctions_Diverses.h:6:5: warning: this 'for' clause does not guard... [-Wmisleading-indentation]
    6 |     for ( i = 0 ; i < Taille ; i ++ )
      |     ^~~
/home/aurele/tracemyplane/Fonctions_Diverses.h:9:9: note: ...this statement, but the latter is misleadingly indented as if it were guarded by the 'for'
    9 |         if ( i == Taille ) // Fin du tableau atteint : n'existe pas encore
      |         ^~
/home/aurele/tracemyplane/Fonctions_Diverses.h: In function 'int Ajouter_Sans_Doublon_String(std::string, std::vector<std::__cxx11::basic_string<char> >
/home/aurele/tracemyplane/Fonctions_Diverses.h:21:24: warning: conversion from 'std::vector<std::__cxx11::basic_string<char> >::size_type' {aka 'long un
   21 |     int Taille = V.size() ;
      |                  ~~~~~~^~
/home/aurele/tracemyplane/Fonctions_Diverses.h:23:16: warning: conversion to 'std::vector<std::__cxx11::basic_string<char> >::size_type' {aka 'long unsi
   23 |         if ( V[i] == element ) // Existe deja
      |              ^
/home/aurele/tracemyplane/Fonctions_Diverses.h:22:5: warning: this 'for' clause does not guard... [-Wmisleading-indentation]
   22 |     for ( i = 0 ; i < Taille ; i ++ )
      |     ^~~
/home/aurele/tracemyplane/Fonctions_Diverses.h:25:9: note: ...this statement, but the latter is misleadingly indented as if it were guarded by the 'for'
   25 |         if ( i == Taille ) // Fin du tableau atteint : n'existe pas encore
      |         ^~
In file included from /home/aurele/tracemyplane/Coeur_Vol.cpp:9:
/home/aurele/tracemyplane/Classe_Moment.h: In constructor 'Moment::Moment()':
/home/aurele/tracemyplane/Classe_Moment.h:12:17: warning: 'Moment::Heure' should be initialized in the member initialization list [-Weffc++]
   12 |                 Moment () { } ;
      |                 ^~~~~
/home/aurele/tracemyplane/Classe_Moment.h:12:17: warning: 'Moment::Minute' should be initialized in the member initialization list [-Weffc++]
/home/aurele/tracemyplane/Classe_Moment.h: In constructor 'Moment::Moment(int, int)':
/home/aurele/tracemyplane/Classe_Moment.h:13:17: warning: 'Moment::Heure' should be initialized in the member initialization list [-Weffc++]
   13 |                 Moment ( int h , int m ) { Heure = h ; Minute = m ; }
      |                 ^~~~~~
/home/aurele/tracemyplane/Classe_Moment.h:13:17: warning: 'Moment::Minute' should be initialized in the member initialization list [-Weffc++]
In file included from /home/aurele/tracemyplane/Coeur_Vol.cpp:10:
/home/aurele/tracemyplane/Classe_Position.h: In constructor 'Position::Position()':
/home/aurele/tracemyplane/Classe_Position.h:13:17: warning: 'Position::Latitude' should be initialized in the member initialization list [-Weffc++]
   13 |                 Position () { } ;
      |                 ^~~~~~~~
/home/aurele/tracemyplane/Classe_Position.h:13:17: warning: 'Position::Longitude' should be initialized in the member initialization list [-Weffc++]
/home/aurele/tracemyplane/Classe_Position.h:13:17: warning: 'Position::Altitude' should be initialized in the member initialization list [-Weffc++]
/home/aurele/tracemyplane/Classe_Position.h: In constructor 'Position::Position(double, double, double)':
/home/aurele/tracemyplane/Classe_Position.h:14:17: warning: 'Position::Latitude' should be initialized in the member initialization list [-Weffc++]
   14 |                 Position ( double Latit , double Longit , double Altit ) { Latitude = Latit ; Longitude = Longit ; Altitude = Altit ; }
      |                 ^~~~~~~~
/home/aurele/tracemyplane/Classe_Position.h:14:17: warning: 'Position::Longitude' should be initialized in the member initialization list [-Weffc++]
/home/aurele/tracemyplane/Classe_Position.h:14:17: warning: 'Position::Altitude' should be initialized in the member initialization list [-Weffc++]
/home/aurele/tracemyplane/Classe_Position.h:14:87: warning: conversion from 'double' to 'float' may change value [-Wfloat-conversion]
   14 |                 Position ( double Latit , double Longit , double Altit ) { Latitude = Latit ; Longitude = Longit ; Altitude = Altit ; }
      |                                                                                       ^~~~~
/home/aurele/tracemyplane/Classe_Position.h:14:107: warning: conversion from 'double' to 'float' may change value [-Wfloat-conversion]
   14 |                 Position ( double Latit , double Longit , double Altit ) { Latitude = Latit ; Longitude = Longit ; Altitude = Altit ; }
      |                                                                                                     ^~~~~~
/home/aurele/tracemyplane/Classe_Position.h:14:127: warning: conversion from 'double' to 'float' may change value [-Wfloat-conversion]
   14 |   Position ( double Latit , double Longit , double Altit ) { Latitude = Latit ; Longitude = Longit ; Altitude = Altit ; }
      |                                                                                                               ^~~~~

/home/aurele/tracemyplane/Classe_Position.h: In member function 'void Position::Set_Latitude(double)':
/home/aurele/tracemyplane/Classe_Position.h:15:61: warning: conversion from 'double' to 'float' may change value [-Wfloat-conversion]
   15 |                 void Set_Latitude ( double l ) { Latitude = l ; }
```

```
/home/aurele/tracemyplane/Classe_Vol.h:53:50: warning: conversion to 'std::vector<Aeroport>::size_type' {aka 'long unsigned int'} from 'int' may change the
   53 |                             Aeroport_Arrivee = A[i] ;
      |                                                  ^
/home/aurele/tracemyplane/Classe_Vol.h:56:28: warning: comparison of integer expressions of different signedness: 'int' and 'std::vector<Aeroport>::size_typ
   56 |                     if ( i == A.size() )
      |                          ~~^~~~~~~~~~
/home/aurele/tracemyplane/Classe_Vol.h: In member function 'void Vol::Afficher_Etape(int)':
/home/aurele/tracemyplane/Classe_Vol.h:64:36: warning: conversion to 'std::vector<Moment>::size_type' {aka 'long unsigned int'} from 'int' may change the si
   64 |                     Moments_Etapes[i].Afficher_Moment() ;
      |                                    ^
/home/aurele/tracemyplane/Classe_Vol.h:66:38: warning: conversion to 'std::vector<Position>::size_type' {aka 'long unsigned int'} from 'int' may change the
   66 |                     Positions_Etapes[i].Afficher_Position() ;
      |                                      ^
/home/aurele/tracemyplane/Classe_Vol.h: In function 'int Charger_Liste_Vols(std::string, std::vector<Vol>&, std::vector<Aeroport>&)':
/home/aurele/tracemyplane/Classe_Vol.h:92:10: warning: empty parentheses were disambiguated as a function declaration [-Wvexing-parse]
   92 |     Vol v() ;
      |          ^~
/home/aurele/tracemyplane/Classe_Vol.h:92:10: note: remove parentheses to default-initialize a variable
   92 |     Vol v() ;
      |          ^~
      |          --
/home/aurele/tracemyplane/Classe_Vol.h:92:10: note: or replace parentheses with braces to value-initialize a variable
/home/aurele/tracemyplane/Classe_Vol.h:109:23: warning: declaration of 'ss' shadows a previous local [-Wshadow]
  109 |         istringstream ss ( Ligne ) ;
      |                       ^~
/home/aurele/tracemyplane/Classe_Vol.h:97:18: note: shadowed declaration is here
   97 |     stringstream ss ;
      |                  ^~
/home/aurele/tracemyplane/Classe_Vol.h:121:16: warning: comparison of integer expressions of different signedness: 'int' and 'std::vector<Vol>::size_type' {
  121 |         if ( n >= V.size() ) // Le vol a un nom non encore trouve : cree un nouveau
      |              ~~^~~~~~~~~~
/home/aurele/tracemyplane/Classe_Vol.h:124:23: warning: conversion to 'std::vector<Vol>::size_type' {aka 'long unsigned int'} from 'int' may change the sign
  124 |                     V[n].Add_Moment_Position ( m , p ) ;
      |                       ^
/home/aurele/tracemyplane/Classe_Vol.h:134:25: warning: comparison of integer expressions of different signedness: 'int' and 'std::vector<Vol>::size_type' {
  134 |         for ( n = 0 ; n < V.size() ; n ++ )
      |                         ~~^~~~~~~~~
/home/aurele/tracemyplane/Classe_Vol.h:135:15: warning: conversion to 'std::vector<Vol>::size_type' {aka 'long unsigned int'} from 'int' may change the sign
  135 |             V[n].Rechercher_Depart_Arrivee ( A ) ;
      |               ^
/home/aurele/tracemyplane/Classe_Vol.h:95:13: warning: unused variable 'j' [-Wunused-variable]
   95 |         int j ;
      |             ^
In file included from /home/aurele/tracemyplane/Coeur_Vol.cpp:14:
/home/aurele/tracemyplane/Classe_Utilisateur.h: In constructor 'Utilisateur::Utilisateur(int, QString, QString)':
/home/aurele/tracemyplane/Classe_Utilisateur.h:20:9: warning: 'Utilisateur::ID' should be initialized in the member initialization list [-Weffc++]
   20 |         Utilisateur ( int i , QString Name , QString MDP )
      |         ^~~~~~~~~~
/home/aurele/tracemyplane/Classe_Utilisateur.h:20:9: warning: 'Utilisateur::Nom' should be initialized in the member initialization list [-Weffc++]
/home/aurele/tracemyplane/Classe_Utilisateur.h:20:9: warning: 'Utilisateur::Mot_De_Passe' should be initialized in the member initialization list [-Weffc++]
/home/aurele/tracemyplane/Coeur_Vol.cpp: In function 'int main()':
/home/aurele/tracemyplane/Coeur_Vol.cpp:21:13: warning: unused variable 'i' [-Wunused-variable]
   21 |         int i ;
      |             ^
```