

# **R**apport de stage

**Passage de la visualisation des paramètres  
de vol en interface WEB**

**Étude de faisabilité et réalisation d'un  
prototype**

**Stage du 12/04/2021 au 02/07/2021**

**à l'Aéroport du Bourget (93 Le Bourget)**

**Pierre HERVELIN**

**Élève en DUT informatique à l'Université Sorbonne  
Paris Nord**



# Remerciements

Avant toute chose, je souhaite remercier chaleureusement mon maître de stage Henri DENIS pour sa grande disponibilité et son aide précieuse tout au long de mon stage.

Un grand merci également à Florent DURU qui m'a accueilli et aidé au début de mon stage, en l'absence non prévue d'Henri.

Je remercie de plus Nicolas HOLVOET et Angélique LEFEVRE pour m'avoir aidé et présenté LEA du côté utilisateur. Grâce à eux j'ai pu participer à la lecture d'enregistreurs de vols et de calculateurs au sein du laboratoire.

Sans oublier l'équipe du pôle Informatique qui m'a mis à disposition tout le matériel nécessaire et qui m'a accompagné dans l'installation de l'environnement logiciel.

Je remercie enfin toute l'équipe du pôle Enregistreurs et Systèmes avioniques du département Technique du BEA pour m'avoir accueilli.



# Introduction

Dans le cadre de ma deuxième année de D.U.T informatique, j'ai eu l'opportunité de réaliser mon stage de fin d'étude au sein du BEA : Bureau d'Enquêtes et d'Analyses pour la sécurité de l'aviation civile, situé sur la plateforme aéroportuaire de Paris-Le Bourget.

Ce stage de fin d'étude, d'une durée de 11 semaines, s'est déroulé du 12 avril au 2 juillet 2021. Il m'a permis de mettre en application les différentes notions techniques acquises durant mes deux années d'IUT en réalisant un projet technique utile au BEA.

Ce projet s'est articulé autour de trois grandes parties, une phase d'observation et d'analyse, une phase de réflexion et de recherche et pour finir une partie pratique et très concrète en codant un prototype. Ce travail m'a permis de concevoir une interface web à l'image de LEA, le logiciel utilisé par le département Technique du BEA pour exploiter et analyser les paramètres de vol.

Cette interface prototype, qui n'est pas un équivalent de LEA mais une interface de test, n'avait pas vocation pour le BEA à devenir pérenne. Il s'agissait d'évaluer la faisabilité du passage à l'IHM web pour la représentation des paramètres. J'ai pris conscience au fur et à mesure que le stage avançait que mon prototype allait en quelque sorte devenir « jetable ».

La gestion du projet a suivi le principe de la méthode agile. Si cette méthode a de nombreux atouts, mon ressenti, du moins au début, était un sentiment d'avancer un peu en « aveugle ». Cependant, la flexibilité offerte par cette méthode a permis de s'adapter rapidement, tout au long du projet, aux avancées du développement. Et le produit fourni, finalement, est devenu un produit moins « jetable » qu'un simple prototype de faisabilité.

Après avoir présenté le BEA et ses missions puis le logiciel LEA, je décrirai la méthode de développement choisie ainsi que le produit réalisé. Je terminerai enfin par un retour d'expérience sur la manière dont j'ai appréhendé ce travail en entreprise.

# Table des matières

|  |    |
|--|----|
| REMERCIEMENTS .....                              | 3  |
| INTRODUCTION .....                               | 5  |
| TABLE DES MATIÈRES .....                         | 6  |
| 1. Contexte du stage .....                       | 8  |
| 1.1 Présentation du BEA et de ses missions ..... | 8  |
| 1.2 Ma place au sein du BEA .....                | 8  |
| 2. Le logiciel LEA.....                          | 9  |
| 2.1 Présentation .....                           | 9  |
| 2.2 Refonte du logiciel.....                     | 12 |
| 2.3 Ma mission .....                             | 13 |
| 3. L'interface web .....                         | 14 |
| 3.1 Choix de la librairie.....                   | 14 |
| 3.2 Choix du mode de développement .....         | 16 |
| 3.3 Présentation du produit .....                | 17 |
| 3.4 Utilisation par les enquêteurs .....         | 30 |
| 4. Expérience acquise.....                       | 32 |
| CONCLUSION.....                                  | 33 |
| ANNEXES .....                                    | 34 |



# 1. Contexte du stage

## 1.1 Présentation du BEA et de ses missions

Le Bureau d'Enquêtes et d'Analyses pour la sécurité de l'aviation civile (BEA) est l'autorité responsable des enquêtes de sécurité dans l'aviation civile. Le siège du BEA se situe dans la zone aéroportuaire de l'aéroport Paris-Le Bourget et dispose aussi de cinq antennes locales (Toulouse, Aix en Provence, Rennes, Lyon et Bordeaux). Son effectif est composé d'un peu moins de cent de personnes dont 50 enquêteurs de sécurité.

Le BEA est un organisme directement rattaché au ministère des transports qui est appelé à intervenir lors d'accident ou d'incident d'avion ou hélicoptère de transport public ou d'aviation générale. Le BEA est l'autorité à compétence nationale et internationale dès lors que la France est le pays de construction, d'exploitation ou d'immatriculation de l'appareil accidenté. Il est observateur lorsque l'accident a fait des victimes françaises.

Il est de fait en charge des enquêtes lorsque l'évènement survient sur le territoire national français, cela inclut les territoires d'Outre-mer. Par ailleurs, le BEA offre une assistance technique quand une autorité étrangère fait appel à ses compétences, le plus souvent en matière de lecture des enregistreurs de vol. Ces dispositions sont décrites dans l'annexe 13 de la convention relative à l'organisation de l'aviation civile internationale (OACI).

Les enquêtes du BEA sont des enquêtes techniques, menées en toute indépendance de l'enquête judiciaire. Le seul objectif est d'améliorer et de renforcer la sécurité des aéronefs et de leurs usagers.

Les missions principales du BEA sont :

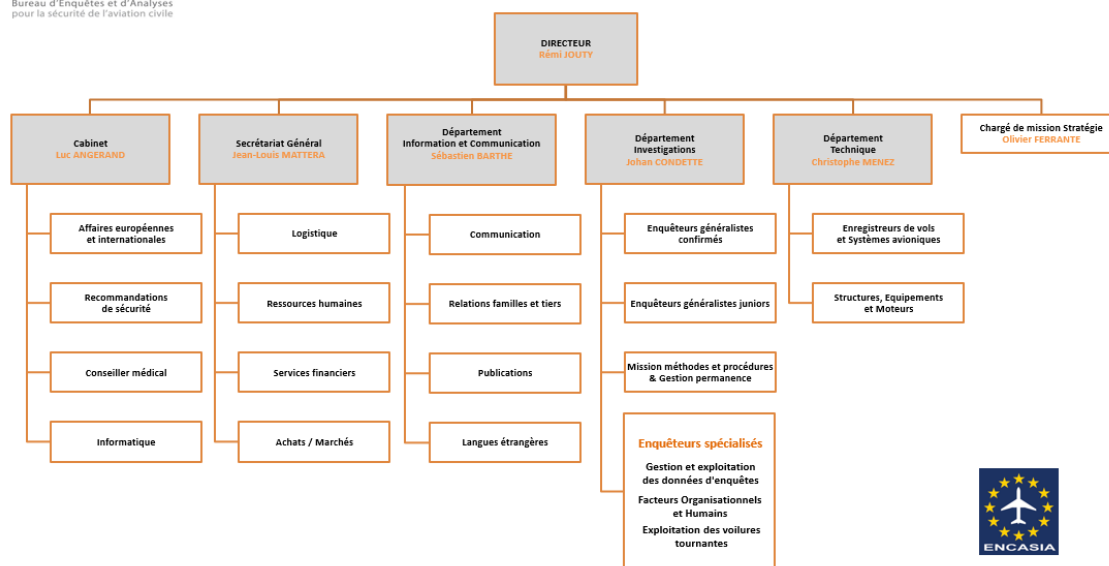
- ☐ Continuer à améliorer la sécurité aérienne et maintenir la confiance du public envers le transport aérien au travers de ses enquêtes et études de sécurité conduites en toute indépendance avec efficacité et impartialité.
- ☐ Contribuer à la qualité et l'objectivité des enquêtes à l'étranger auxquelles il participe, a minima pour ce qui concerne les organismes français impliqués.
- ☐ Exploiter et mettre en valeur les données et enseignements de sécurité acquis par le BEA pour la prévention de futurs accidents en aviation civile.

Enfin le BEA possède des moyens propres d'examen et d'analyses d'éléments prélevés sur des sites d'accidents grâce à plusieurs laboratoires situés au Bourget.

## 1.2 Ma place au sein du BEA

Le BEA est assez original du fait que beaucoup d'outils utilisés dans le cadre de ses enquêtes ont été développés en interne par les enquêteurs du département Technique. C'est dans ce service que j'ai pu effectuer mon stage.





Le département Technique est composée de deux pôles : Enregistreurs de vol & Systèmes avioniques (PESA) et Structure Équipement & Moteur (PSEM).

Le BEA dispose de trois laboratoires techniques qui permettent un travail dans les meilleures conditions : les laboratoires « Enregistreurs de vol », « Avionique » et « Examens des matériaux ».

Le pôle PESA est chargé, entre autres, après réception des enregistreurs de vol (traditionnellement nommés « boîtes noires » par le grand public) de réaliser le processus d'exploitation des enregistreurs de vol, qui consiste à lire l'intégralité des données qu'ils contiennent et de sauvegarder ces données. Ces données sont ensuite décodées et analysées dans le cadre de l'enquête de sécurité.

Le logiciel LEA (Logiciel d'Étude et d'Analyse) est l'outil central qui permet l'exploitation des données de vol. Ce logiciel a été développé en interne par le département Technique. Henri DENIS, mon maître de stage et chef de ce projet, a été un des contributeurs de LEA au cours des dernières années.

## 2. Le logiciel LEA

### 2.1 Présentation

Les enregistreurs de vol équipent les avions de transport et enregistrent pendant le vol des données destinées à être utilisées après un accident ou un incident.

Les avions de transport sont équipés de deux enregistreurs de vol :

- Le CVR (Cockpit Voice Recorder-enregistreur phonique) enregistre des données audio : les échanges entre les pilotes et avec les contrôleurs ainsi que l'environnement acoustique du poste de pilotage (conversations, bruits, alarmes sonores).
- Le FDR (Flight Data Recorder-enregistreur de paramètres) enregistre les valeurs des paramètres de l'avion (vitesse, altitude, régime des moteurs, engagement et désengagement d'automatisme, position des gouvernes, des commandes de vol...). Selon l'âge et le type de l'aéronef le nombre de

paramètres enregistrés varie de 16 paramètres pour les enregistreurs les plus anciens à environ 3 500 paramètres pour des avions récents comme l'Airbus A350.

La durée totale réglementaire d'enregistrement est d'au moins 25h pour le FDR et d'au moins 2h pour le CVR. Pendant le vol, les nouvelles données enregistrées viennent de façon continue remplacer les données les plus anciennes. Le principe utilisé ici est un enregistrement cyclique des données, avec un pointeur d'écriture qui parcourt la mémoire de manière « circulaire ».

### La lecture du contenu des enregistreurs

La première étape du processus d'exploitation des enregistreurs de vol consiste à lire l'intégralité des données qu'ils contiennent et de sauvegarder ces données. C'est une étape particulière, qui dépend du modèle de l'enregistreur et qui requiert des logiciels spécialisés.

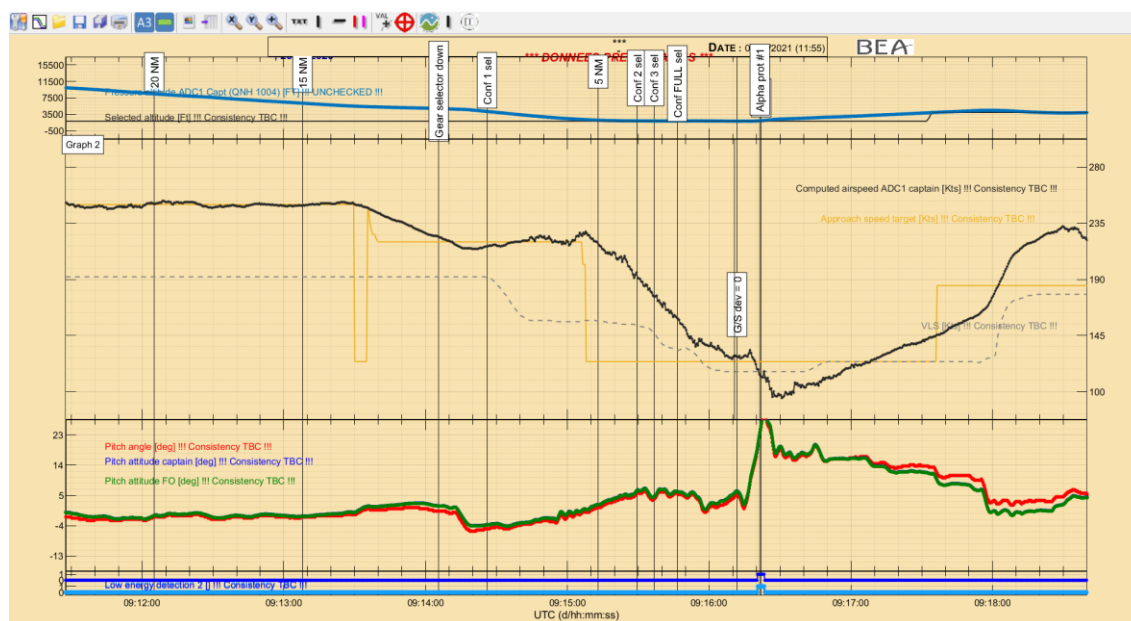
### Analyse des données

Une fois les données lues et sauvegardées, elles sont analysées par les enquêteurs. Le fichier extrait de l'enregistreur est un fichier binaire appelé « fichier brut ». Le décodage de ce fichier brut est réalisé à partir d'un document de décodage propre à chaque avion. Dans un premier temps, il faut récupérer les informations binaires codant le paramètre en leur associant un temps, puis il faut transformer cette information binaire en valeur ingénierie pour l'exploiter.

Les valeurs des paramètres et leurs évolutions au cours du temps peuvent être ensuite représentées sous forme de courbes ou de tableaux et être utilisées dans des calculs pour modéliser le comportement de l'avion (calcul de paramètres dit « dérivés »).

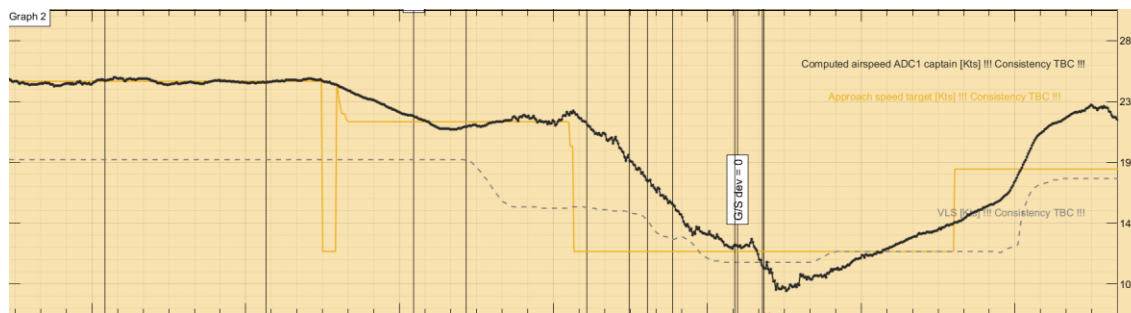
La représentation des paramètres est effectuée grâce au logiciel LEA. Je vais maintenant procéder à une présentation de ce logiciel. Je ne traiterai, étant donné l'objectif de mon stage, que les fonctionnalités qui touchent à cette représentation.

Voici comment se présente une planche LEA.



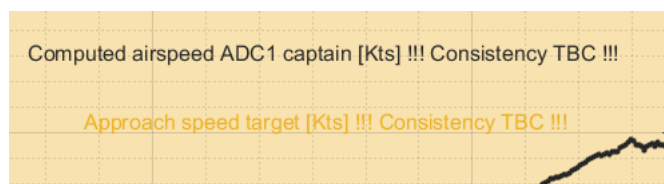
Une planche est composée d'un ou plusieurs graphes, eux même composés d'un ou plusieurs paramètres. Ici la planche est composée de 4 graphes. On remarque également que les graphes peuvent avoir des tailles différentes, ils ont chacun un

domaine (ou hauteur) différent. Ce domaine est modifiable dynamiquement grâce à un clic glissé de la souris.



*Un graphe*

Ce graphe est composé de trois paramètres. On peut retrouver le nom de ces paramètres grâce aux légendes.



*Légendes*

Ces légendes sont déplaçables dynamiquement avec la souris. On trouve dans ces légendes des « consistency TBC ». Comme vu précédemment dans l'analyse des données, le décodage des fichiers brut est réalisé à partir d'un document de décodage propre à chaque avion. Cependant, ce document de décodage peut parfois ne pas être utilisable, et nécessite des corrections manuelles de la part des enquêteurs.

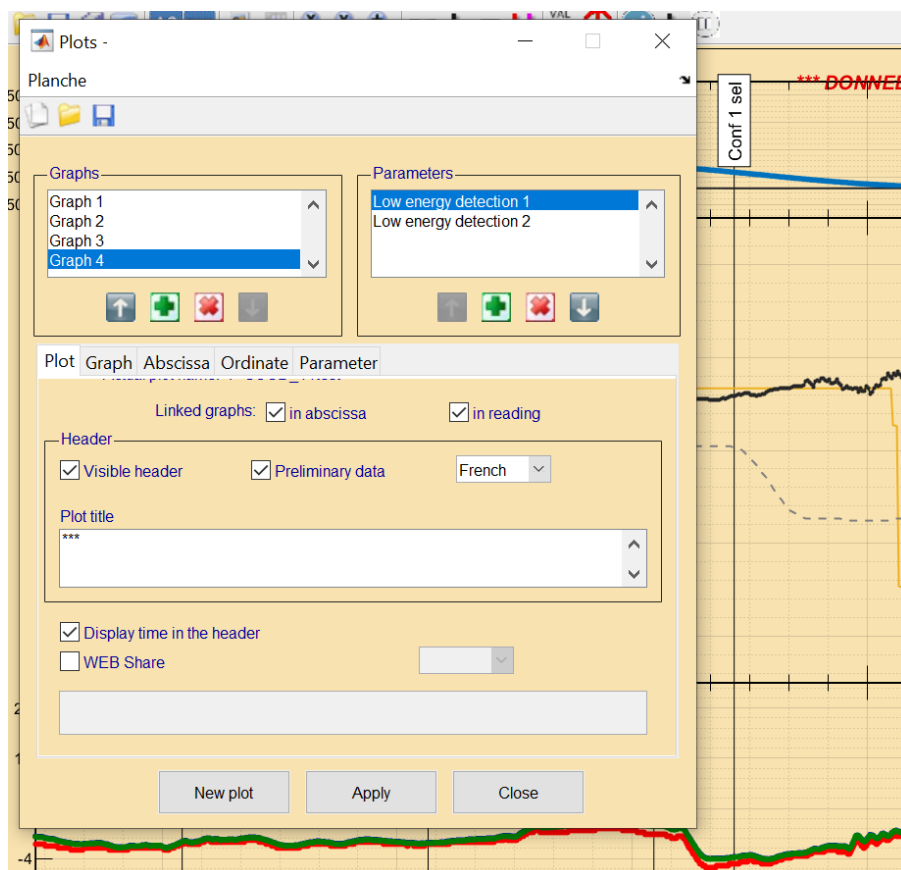
Ainsi, 5 valeurs ont été établies :

- **CHECKED** : la fonction de décodage du paramètre a été vérifié par rapport à la documentation constructeur
- **UNCHECKED** : la fonction de décodage du paramètre n'a pas été vérifié par rapport à la documentation constructeur

Une fois la fonction de décodage vérifiée, les valeurs ingénieurs obtenues sont qualifiées :

- **Consistent** : les valeurs ingénieurs obtenues sont cohérentes par rapport à ce qui est attendu
- **Consistency TBC** : La cohérence des valeurs ingénieurs n'a pas été vérifiée.
- **Not consistent** : Les valeurs ingénieurs ne sont pas cohérentes (par exemple : vitesse de l'avion non nulle, alors qu'il ne bouge pas).

Toutes modifications de la planche peuvent être faites à partir de l'interface « plots ».



On y retrouve la structure de la planche, avec les différents graphes et paramètres. Ainsi que les onglets « Plot », « Graph », « Abscissa », « Ordinate », « Parameter » (voir annexe 2).

## 2.2 Refonte du logiciel

### 2.2.1 Points d'améliorations

Le développement du logiciel LEA, codé sous Matlab, a démarré dans les années 2000. Ce développement a été itératif et s'est appuyé sur de nombreux stages, qui ont permis de générer un logiciel certes très complet, mais aussi très compliqué à faire évoluer et à maintenir. Le code commence à s'empiler et à « vieillir », posant des problèmes de fluidité et de bugs.

Certaines fonctionnalités ne sont pas complètement optimisées – ou ne sont plus optimisées - et ne peuvent l'être facilement car l'hétérogénéité du code de LEA ne le permet pas. C'est en particulier le cas pour l'affichage des paramètres et les fonctionnalités associées.

Certaines fonctionnalités ne sont pas ou plus utilisées. En observant les utilisateurs, j'ai pu assister à des bugs non bloquants ou facilement contournables qui étaient bien connus des utilisateurs. C'est pourquoi la question d'une refonte de l'affichage des paramètres par le logiciel LEA se posait.

### 2.2.2 Choix offerts

Pour envisager la refonte d'une partie d'un tel logiciel, une étude sérieuse doit d'abord être conduite afin notamment de sélectionner les fonctionnalités à garder, celles à optimiser ou à créer puis le langage ou le type de plateforme à choisir, etc.

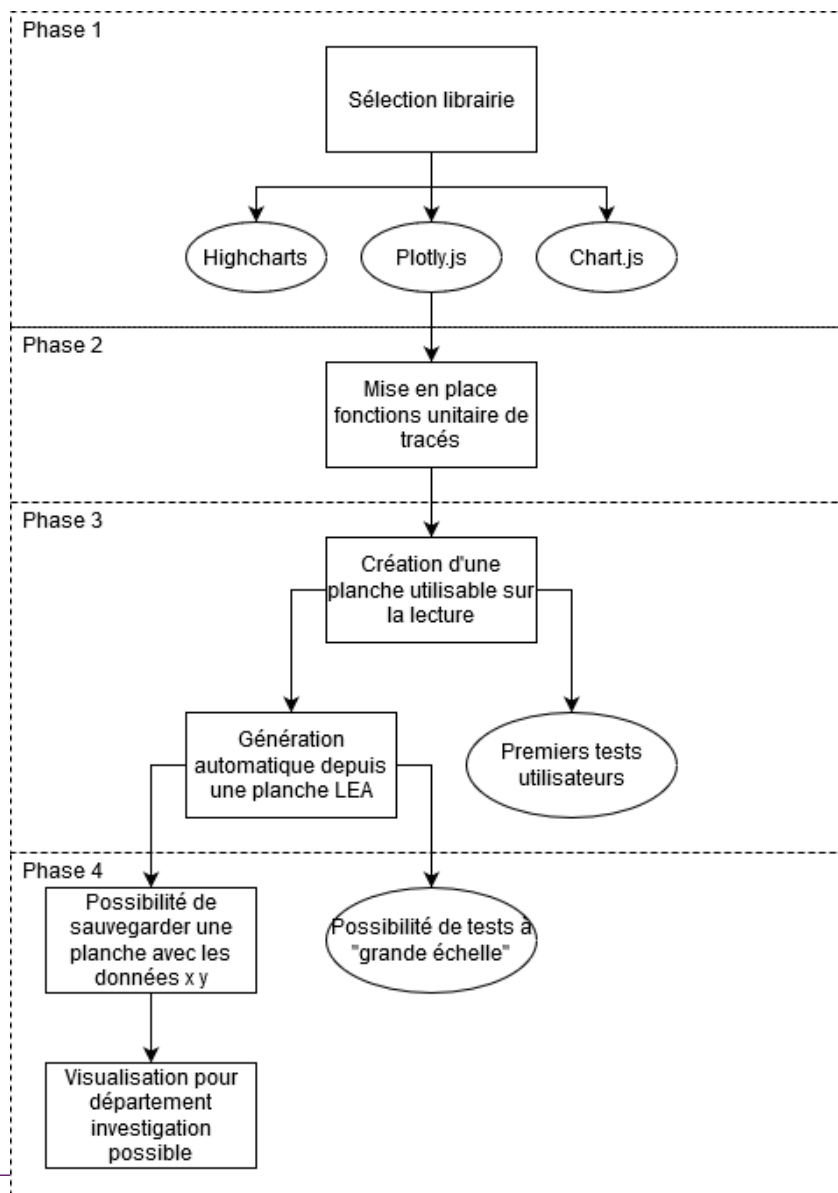
Pour l'instant, deux possibilités ont été pensées, une refonte de zéro sous Matlab ou une refonte en web. Le web est l'inconnu dans ce choix, tout le but de mon stage était de réaliser une étude de faisabilité pour évaluer la pertinence de ce choix logiciel.

## 2.3 Ma mission

L'objectif de mon stage était la réalisation pas à pas d'un prototype de test, faisant aussi office d'étude de faisabilité. Ce prototype devait permettre aux enquêteurs spécialisés du département Technique d'utiliser une visualisation web des paramètres de vol produits par LEA, pour décider si une refonte web était un choix viable ou pas.

Le graphe suivant résume les quatre phases réalisées durant mon stage. Dans la troisième partie de ce rapport ces quatre phases seront décrites en détail.

### Synopsis du déroulement de ma mission



## 3. L'interface web

### 3.1 Choix de la librairie

#### 3.1.1 Besoins utilisateur

La première phase (voir synopsis du déroulement de ma mission) a consisté à cerner les besoins utilisateurs. C'était une phase d'observation et d'analyse. J'ai beaucoup observé les utilisateurs en situation et j'ai profité de leurs retours en temps réel. J'ai aussi participé à des lectures de paramètres ainsi qu'à la construction d'un historique de vol, deux des utilisations du logiciel LEA.

Dans une refonte d'un logiciel existant, le point le plus important est l'avis de l'utilisateur. S'il n'apprécie pas le nouveau produit, même s'il est bien meilleur que le précédent, alors il ne sera pas adopté, d'où l'importance de bien identifier les fonctionnalités utiles pour l'utilisateur de celles dont il peut se passer. J'ai donc pu lister les fonctionnalités de LEA que j'ai jugé importantes, car très utilisées, afin de les intégrer au prototype d'interface web qu'on me demande de développer.

Par ailleurs, j'ai eu accès à un projet LEA pour me familiariser avec le logiciel, son ergonomie et ses fonctionnalités. Sur cette base j'ai dressé un tableau de toutes les fonctionnalités relatives à l'affichage et aux modifications des planches de paramètres (voir Annexe 1).

#### *Extrait de la liste des fonctionnalités :*

##### Modifs dynamiques sur les graphes :

- Réglage de la taille d'un graph
- Zoom sur un axe avec molette
- Déplacement sur un axe
- Déplacement d'un label
- Changement de couleurs d'une courbe
- Mettre au premier/second plan une courbe (en cas de superposition de courbe)
- Ajout d'un tag (faire naviguer un point sur la courbe pour avoir ses coordonnées)
- Mettre les coordonnées d'une courbe à droite ou à gauche
- Aligner les légendes à celle sélectionnée
- Éditer le paramètre (modifier le .gri)
- Exporter les paramètres en .mat
- Ligne de paramètres
- Ajout/retrait d'un paramètre
- Ajout/retrait d'événements
- Modification du titre de la planche

Bien évidemment je n'ai pas eu à coder chaque fonctionnalité listée car il aurait fallu bien plus que 3 mois de stage pour un tel projet. De plus, le choix des fonctionnalités que j'allais devoir développer ne m'était pas initialement connu.

Le début du travail a consisté à travailler à partir de fonctionnalités essentielles à l'utilisateur et au bon fonctionnement du prototype. Sur la base de ce travail j'ai pu commencer à me documenter sur les différentes librairies existantes pour y retrouver les fonctionnalités recherchées.

### 3.1.2 Comparatif entre les fonctionnalités existantes dans les librairies et le besoin en développements

Le choix de l'utilisation d'une librairie m'a été imposé dès le départ. Le meilleur moyen de parvenir à ce choix était de dresser un tableau comparatif. Ce travail m'a demandé plus de temps que prévu, j'y suis parvenu en trois semaines alors que c'était prévu en une.

Plutôt qu'un choix précipité j'ai en effet préféré, en accord avec mon chef de projet, consolider ma sélection par de la documentation et des tests. Certaines fonctionnalités me demandaient des recherches plus poussées pour savoir si elles étaient réalisables ou pas avec certaines librairies. Par exemple, j'ai examiné des critères tels que le nombre de points maximum sur un graphe avant d'observer un trop fort ralentissement de la page web ou même un dépassement de la capacité logiciel de la librairie, notamment en apprenant à utiliser chaque librairie. Certaines librairies ont été écartées rapidement grâce au travail comparatif (exemple : Googlechart car elle n'est pas open source).

#### Comparatif :

| FONCTIONNALITES                   | Highcharts | Plotly.js | Chart.js | Dygraphs | Chartist.js | C3.js | Googlechart |
|-----------------------------------|------------|-----------|----------|----------|-------------|-------|-------------|
| Free : non-commercial             |            |           |          |          |             |       |             |
| Free : commercial use             |            |           |          |          |             |       |             |
| Open-source                       |            |           |          |          |             |       |             |
| <b>Fonctionnalités techniques</b> |            |           |          |          |             |       |             |
| Subplot                           |            |           |          |          |             |       |             |
| Subplot avec abscisses partagés   |            |           |          |          |             |       |             |
| Zoom                              |            |           |          |          |             |       |             |
| Zoom molette                      |            |           |          |          |             |       |             |
| Zoom molette sur axe              |            |           |          |          |             |       |             |
| Valeurs des points                |            |           |          |          |             |       |             |
| Valeurs désynchronisées           |            |           |          |          |             |       |             |
| Responsif                         |            |           |          |          |             |       |             |
| Pan abscisses                     |            |           |          |          |             |       |             |
| Pan ordonnées                     |            |           |          |          |             |       |             |
| Plusieurs axes Y                  |            |           |          |          |             |       |             |
| Insérer txt/ligne/events          |            |           |          |          |             |       |             |
| Export png,svg...                 |            |           |          |          |             |       |             |
| Canvas                            |            |           |          |          |             |       |             |
| SVG                               |            |           |          |          |             |       |             |
| <b>Nombre de points</b>           |            |           |          |          |             |       |             |
| 1K                                |            |           |          |          |             |       |             |
| 10K                               |            |           |          |          |             |       |             |
| 100K                              |            |           |          |          |             |       |             |
| 500K                              |            |           |          |          |             |       |             |
| 1M                                |            |           |          |          |             |       |             |
| 10M                               |            |           |          |          |             |       |             |
| 100M                              |            |           |          |          |             |       |             |



Finalement, après deux semaines de tests, 3 librairies sont sorties du lot, Plotly.js, Chart.js et Highcharts. De mon côté, techniquement, j'estimais que Plotly.js était la plus adaptée.

N'ayant pas suffisamment réussi à défendre ce choix, mon chef de projet m'a demandé de commencer le premier sprint avec les 3 librairies. Au bout d'une semaine de travail supplémentaire, à la fin du premier sprint, j'ai convaincu mon chef de sélectionner Plotly.js, notamment au vue des difficultés pour coder les fonctionnalités attendues avec les 2 autres librairies.

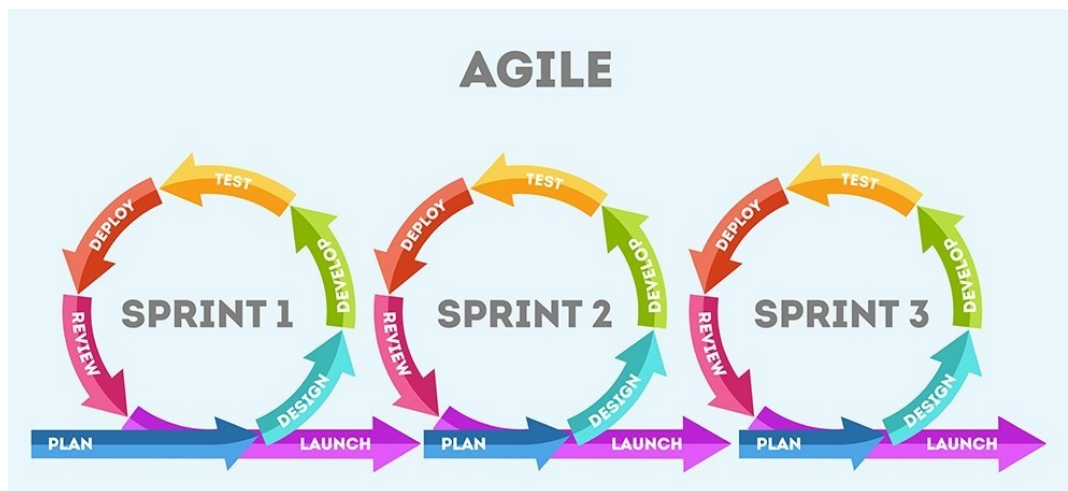
## 3.2 Choix du mode de développement

### 3.2.1 Méthode agile

La gestion de ce projet s'est organisé en sprints hebdomadaires, grâce à la méthode agile qui m'a été imposée par le BEA.

Cette méthode de développement permet de s'adapter très rapidement en fonction de l'avancement d'un projet et des besoins de l'utilisateur qui peuvent apparaître. Comme un projet ne se déroule jamais sur un rail, cette méthode permet de changer de chemin, ou d'y ajouter une bifurcation qui n'était pas pensée au début. Elle permet de réorienter le projet lorsqu'il le faut.

Pour mettre en pratique cette méthode, il faut que le chef de projet maîtrise parfaitement son sujet, et surtout, qu'il prévoit une grande disponibilité. Or comme c'était le cas, c'était une méthode possible.



Ce schéma représente bien la définition connue de la méthode agile qui n'est pas exactement ce que j'ai vécu. Au vu de mon projet, les phases de « lancement » et de « déploiement » n'étaient pas nécessaires. Les tests réalisés n'étaient pas extrêmement poussés, ils se passaient lors de mon point hebdomadaire en fin de semaine, où je présentais les fonctionnalités du sprint et étais guidés par moi-même. De plus, comme mon chef de projet était également un utilisateur, je pouvais avoir un retour. Finalement, les phases de « test », « review », « design » se passaient au même moment. Le prochain sprint était directement défini après que le précédent était validé.

L'exemple parfait de cette « adaptation » permise par la méthode agile est la bifurcation qui a eu lieu à la 9<sup>ème</sup> semaine de mon stage, d'un prototype initialement destiné à être jetable vers un outil de test plus pérenne. En effet, mon chef de projet



m'a présenté une nouvelle manière de voir le prototype, qui allait plus loin que la finalité qu'il avait envisagée initialement, au vu de la bonne avancée et de la qualité de mon travail. Il m'a alors demandé de consolider mon prototype avec de nouvelles fonctionnalités.

À la fin de cette dernière tâche, il s'est avéré que mon prototype pouvait devenir un outil de test qui serait d'une aide appréciable pour les enquêteurs généralistes du département Investigations, conformément à leur demande.

### 3.2.2 Les sprints

Les sprints hebdomadaires étaient réfléchis pour qu'il y ait assez à faire pour 1 semaine. Chaque point de fin de semaine permettait de réfléchir aux prochaines fonctionnalités à développer. Ainsi je participais également à la définition des fonctionnalités du sprint suivant.

Pour présenter les sprints, je vais m'appuyer sur un exemple, la fonctionnalité de sauvegarde. Cette fonctionnalité a posé problème et a été réalisée en plus d'une semaine. Le but était de pouvoir sauvegarder une planche, avec tous les graphes et paramètres, en gardant en mémoire l'emplacement des légendes, le titre et également le zoom. Cependant, dû à un problème de communication, la manière dont j'ai procédé n'était pas celle pensée par mon chef de projet. Dans un premier temps, je sauvegardais grâce à un serveur local et un fichier .php les données de la planche dans un .json directement dans le package du projet. Cette solution fonctionnait, mais ne convenait pas. Ce qui rappelle un point essentiel de la méthode agile (mais aussi de tout projet) : la communication et la bonne compréhension des directives est un impératif.

Ce problème s'est répété lors du développement de l'autosave. Il fallait gérer le cas d'un crash de l'interface ou du pc, une autosave se fait donc toutes les 5 minutes. Lorsqu'elle se fait, elle écrase la précédente, et si l'on ferme la fenêtre correctement, elle est supprimée. Lors de l'ouverture d'une fenêtre, on vérifie s'il y a une autosave, si c'est le cas ça signifie qu'un problème a eu lieu lors de la précédente fermeture. On demande alors à l'utilisateur s'il veut l'ouvrir à nouveau. Cependant, comme pour la sauvegarde, son emplacement était dans le package du projet. Ainsi, après un point hebdomadaire, la fonctionnalité s'est affinée, et je me suis rendu compte qu'il était préférable de passer par Matlab pour laisser le choix de l'emplacement à l'utilisateur, et de supprimer l'utilisation d'un serveur local.

## 3.3 Présentation du produit

### 3.3.1 Mise en place des fonctions unitaires de tracés

Comme expliqué plus haut, il a été décidé de réaliser le prototype en établissant une connexion entre LEA et le navigateur. On m'a imposé le protocole websocket, car il avait déjà été utilisé pour un autre projet nommé Cesium. Cesium permet la visualisation 3D d'un vol grâce à la librairie du même nom, à travers une interface Web. Le fonctionnement de WebSocket était ainsi connu et sûr. Au début, j'ai utilisé le serveur ouvert sur le port utilisé par Cesium. Par la suite j'ai utilisé un port dédié à mon prototype (voir le code de la classe WebPlotServer et le fichier websocket.js en annexe 4 et 5).

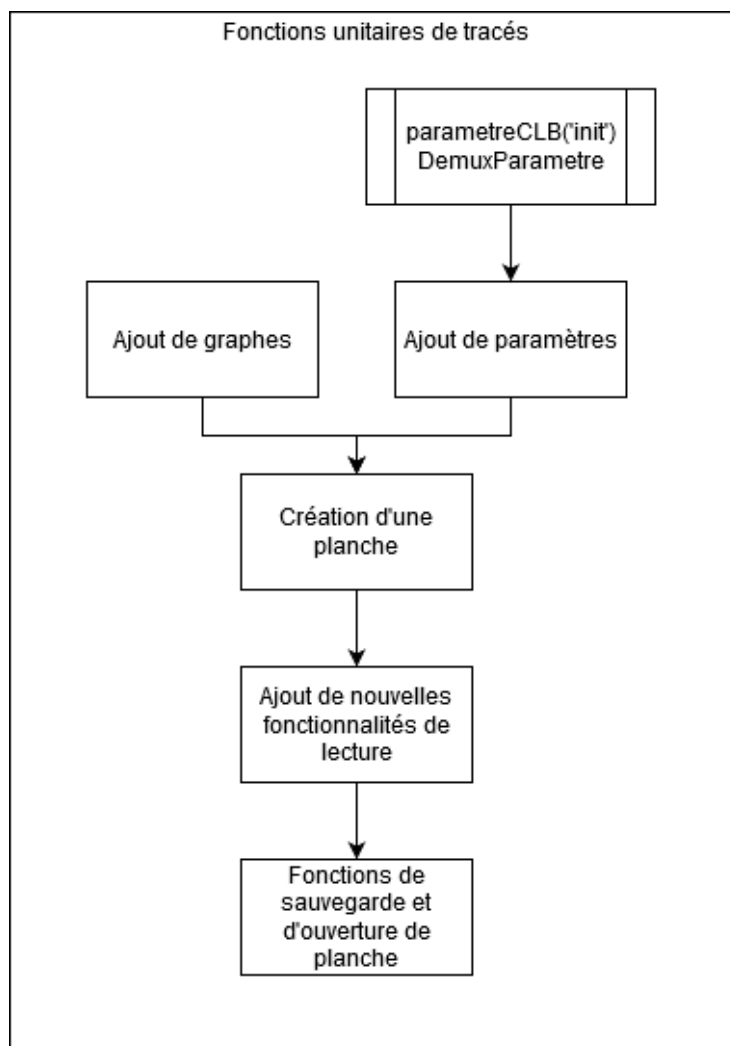
L'envoi et la réception de données entre LEA et le navigateur se fait sous le format TLV (Type, Length, Value), c'est-à-dire une chaîne de caractères contenant en entête

le type (sur deux caractères), suivie de la longueur de la valeur (sur 15 caractères), et enfin la valeur.

*Exemple de messages au format TLV (reçus sur la console) :*

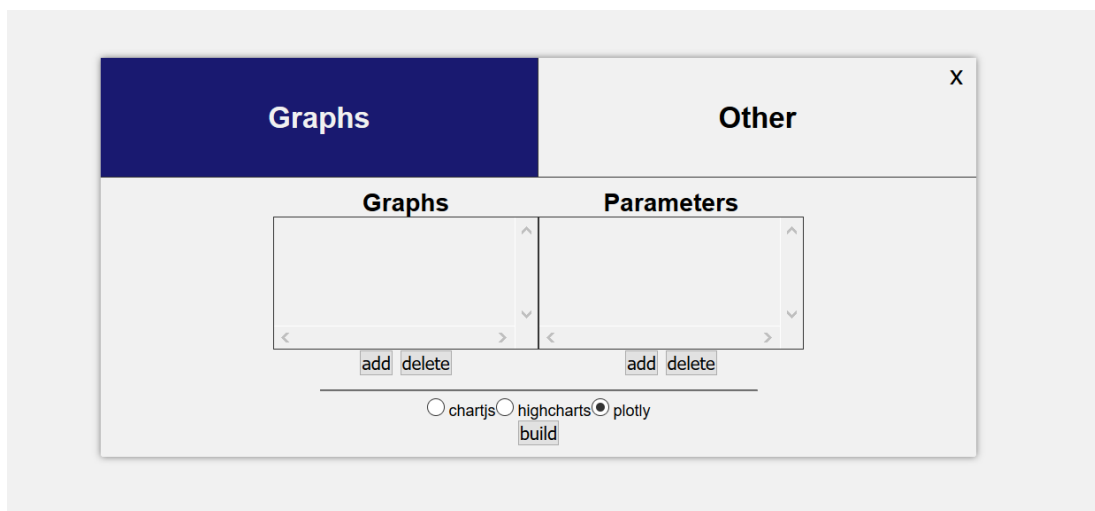
|                       |                    |
|-----------------------|--------------------|
| message envoyé        | websocket.js:22:13 |
| test                  | websocket.js:13:13 |
| 03000000000000004test | websocket.js:31:11 |
| 01000000000000004test | websocket.js:25:13 |
| c'est un type 1       | websocket.js:26:13 |

Une fois la connexion établie, j'ai pu commencer la deuxième phase (voir synopsis), et développer les fonctions unitaires de tracés en javascript (voir le schéma suivant).



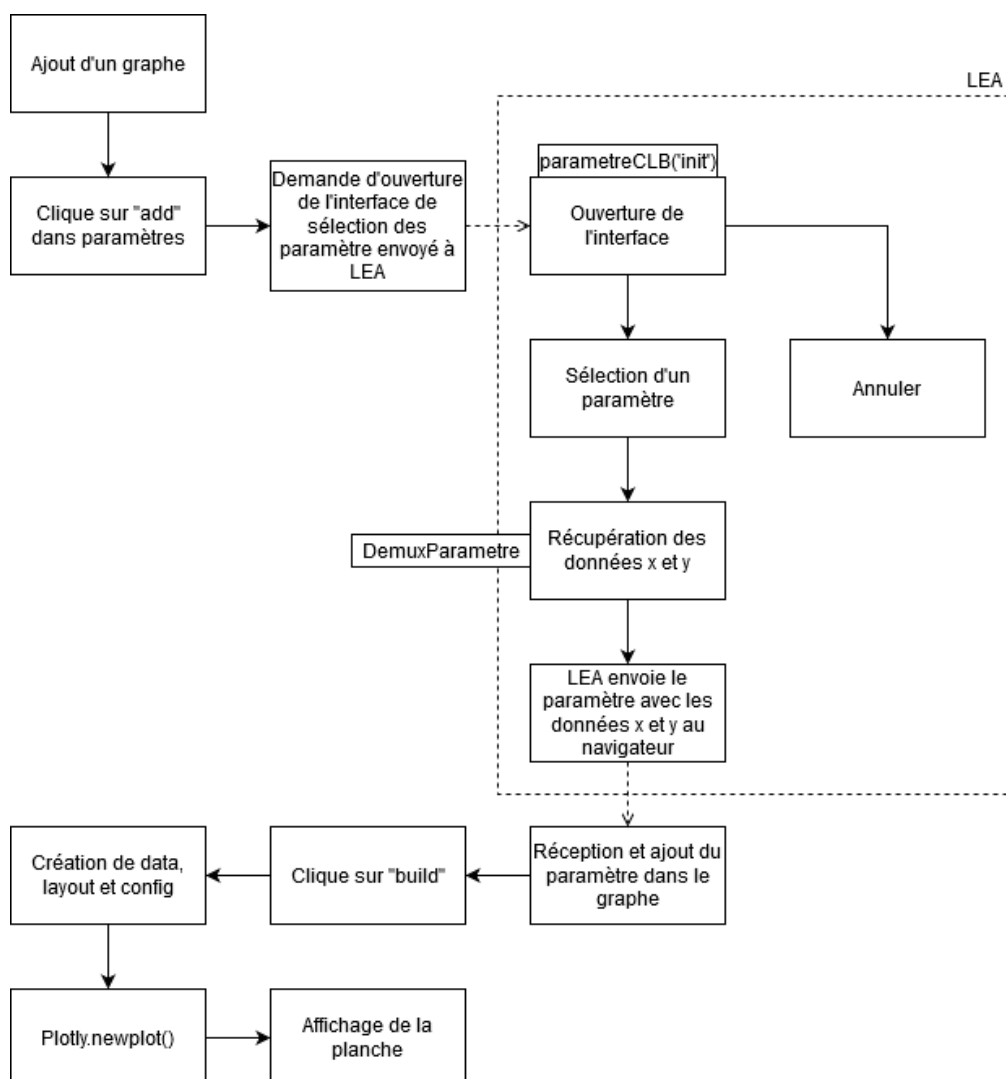
Dans un premier temps, j'ai développé une interface qui me permettait de tester les premières fonctions unitaires de tracés, comme par exemple l'affichage d'un paramètre dans un graphe.

Cette interface ressemblait dans la manière de fonctionner à celle du prototype final.

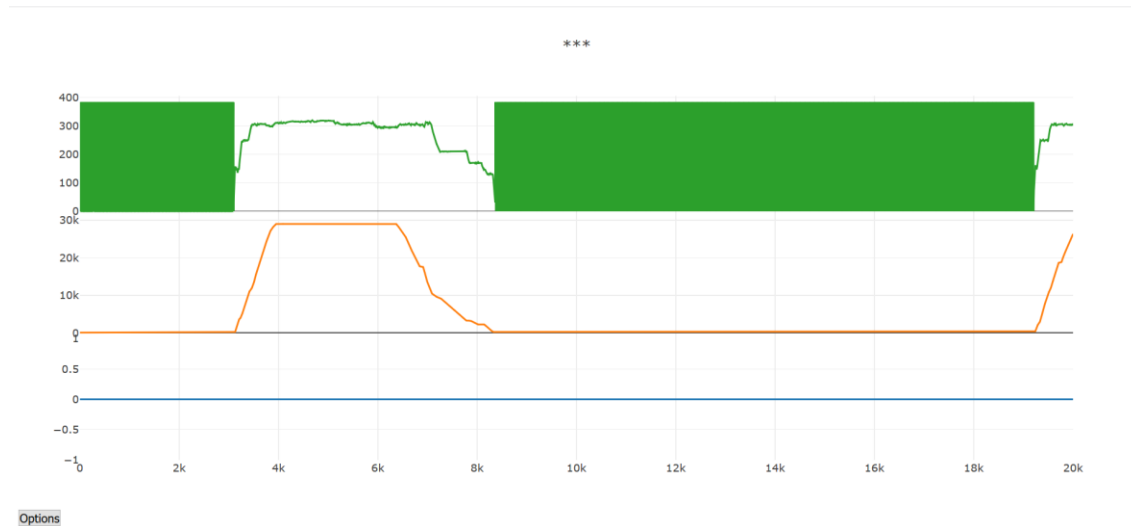


L'interface ci-dessus a été réalisée lors du premier sprint et permettait l'affichage de paramètres, via LEA, dans des graphes associés à un même axe des abscisses. La librairie n'ayant été choisie à ce stade, il y avait encore la possibilité de tracer la planche en utilisant les trois librairies chart.js, highcharts et plotly.js.

Le tracé des paramètres dans une planche, réalisé à ce moment du stage, est présenté dans le schéma ci-dessous.



Comme vu sur le schéma ci-dessus, l'ajout de paramètre se fait grâce à LEA, il existe une commande « parametreCLB('init') » qui ouvre l'interface de sélection de paramètre et qui renvoie la structure du paramètre sélectionné. Ensuite, Les données sont récupérées grâce à la commande « DemuxParametre » qui prend en paramètre la structure précédemment sélectionnée et renvoie ses données x et y. Le tout est envoyé au navigateur et est ensuite affiché grâce au « build ».



### *Une des premières planches web affichées*

Remarque : Cette manière de procéder a quelque peu évolué lors du projet. Au départ, lors de l'ajout d'un paramètre, LEA envoyait de suite les informations de données et de temps. En effet, la visualisation WEB ne gérait pas au départ les différents temps possibles et leur format (normal ou hh :mm :ss). Ensuite, la gestion du choix des temps pour la représentation a été intégrée dans la planche WEB. Il était alors plus simple de laisser LEA gérer ces transformations des valeurs de temps et récupérer les données uniquement au moment du tracé de la planche WEB.

La méthode permettant l'affichage de la planche est « Plotly.newplot() » et prend en paramètre un div, les data, un layout et une config.

- Le div permet de sélectionner une zone d'affichage dans la page web
- Les data sont une structure contenant les propriétés ainsi que les tableaux de données x et y des paramètres

```
data
(31) [...]
  0: {...}
    line: Object { color: "#0000ff" }
    name: "ALTITUDE ELAB Bits 14-27+29"
    type: "scatter"
    x: Array(19637) [ 1592106518257.8125, 1592106519257.8125, 1592106520257.8125, ... ]
    y: Array(19637) [ 36996, 36996, 37000, ... ]
    yaxis: "y"
    <prototype>: Object { ... }
  1: Object { name: "COMPUTED AIRSPEED bits16-27", yaxis: "y2", type: "scatter", ... }
  2: Object { name: "LONGITUDINAL ACCELERATION bits19-27+29", yaxis: "y3", type: "scatter", ... }
  3: Object { name: "LATERAL ACCELERATION bits19-27+29", yaxis: "y4", type: "scatter", ... }
  4: Object { name: "NORMAL ACCELERATION bits18-28+29", yaxis: "y5", type: "scatter", ... }
  5: Object { name: "PITCH ATTITUDE bits20-27+29", yaxis: "y6", type: "scatter", ... }
  6: Object { name: "ROLL ATTITUDE bits20-28+29", yaxis: "y7", type: "scatter", ... }
  7: Object { name: "HEADING", yaxis: "y8", type: "scatter", ... }
  8: Object { name: "DAY OF DATE bits 24-29", yaxis: "y9", type: "scatter", ... }
  9: Object { name: "MONTH OF DATE bits 19-23", yaxis: "y9", type: "scatter", ... }
  10: Object { name: "AUTOBRAKE FAULT", yaxis: "y10", type: "scatter", ... }
```

- Le layout est la structure de l'affichage de la planche, c'est-à-dire qu'elle contient toutes les légendes, titres, graphes... Et également des propriétés de style.

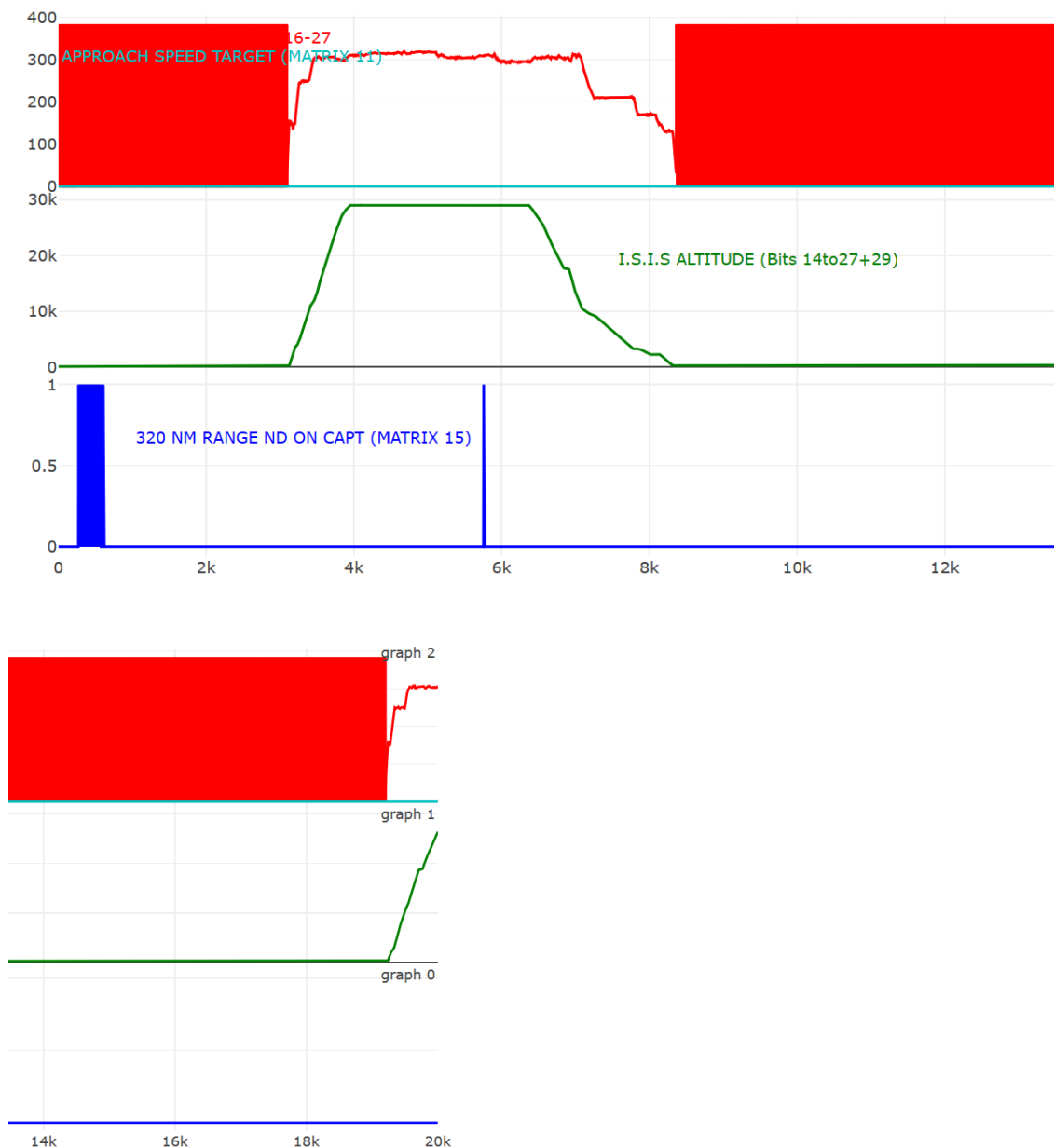
```
layout
{...}
  annotations: Array(45) [ {...}, {...}, {...}, ... ]
  autosize: true
  margin: Object { l: 30, r: 30, b: 80 }
  showlegend: false
  title: Object { text: "****" }
  xaxis: Object { type: "date", range: (2) [...], autorange: true }
  yaxis: Object { side: "left", type: "linear", autorange: true, ... }
  yaxis10: Object { side: "left", type: "linear", autorange: true, ... }
  yaxis11: Object { side: "left", type: "linear", autorange: true, ... }
  yaxis12: Object { side: "left", type: "linear", autorange: true, ... }
  yaxis13: Object { side: "left", type: "linear", autorange: true, ... }
  yaxis14: Object { side: "left", type: "linear", autorange: true, ... }
  yaxis2: Object { side: "left", type: "linear", autorange: true, ... }
  yaxis3: Object { side: "left", type: "linear", autorange: true, ... }
  yaxis4: Object { side: "left", type: "linear", autorange: true, ... }
  yaxis5: Object { side: "left", type: "linear", autorange: true, ... }
  yaxis6: Object { side: "left", type: "linear", autorange: true, ... }
  yaxis7: Object { side: "left", type: "linear", autorange: true, ... }
  yaxis8: Object { side: "left", type: "linear", autorange: true, ... }
  yaxis9: Object { side: "left", type: "linear", autorange: true, ... }
  <prototype>: Object { ... }
```

Concernant le protocole Websocket, la crainte des enquêteurs du département Technique qui avaient réfléchi le projet était que l'envoi et la réception de données

prennent trop de temps et que le buffer de mémoire émission/réception soit dépassé. Cependant, après des tests (Ex : affichage de planches avec plus de 25h de vols), il est constaté que les messages sont envoyés/réceptionnés presque instantanément.

Une fois que l'affichage des planches était solide, certaines fonctionnalités de lecture ont été ajoutées :

- L'affichage des légendes à la manière de LEA
- L'affichage des titres des graphes
- La modification du titre de la planche



Avec toutes ces fonctionnalités, j'ai développé la fonctionnalité de sauvegarde comme expliqué au paragraphe 3.2.2 et puis la fonctionnalité d'ouverture des planches web.

À ce stade mon interface se superposait sous forme de fenêtre (fausse fenêtre car page web) à la planche ce qui n'était pas très pratique. J'ai donc procédé à une refonte complète de l'interface à la demande de mon chef de projet afin que les planches et l'interface soit affichées ensemble.

### 3.3.2 Présentation du produit

Suite à l'amélioration de l'interface et à l'ajout de certaines fonctionnalités, voici comment se présente mon produit de lecture de planche pour les enquêteurs spécialisés du département technique.



Du côté de la planche, on retrouve les fonctionnalités suivantes vues sur LEA :

- L'affichage des légendes déplaçables dynamiquement sur les graphes,
- les « consistency TBC » « unchecked » et « not consistant » dans celles-ci,
- le zoom et le pan,
- la possibilité de modifier le titre et l'affichage des titres des graphes.

De plus, certaines fonctionnalités ont été améliorées, comme l'affichage des légendes et des titres de graphes, qui ont un fond blanc semi opaque, permettant de les lire par-dessus les courbes (ce qui n'est pas le cas pour les planches tracées par LEA).


Afin de modifier la planche, on peut ouvrir le menu « options », qui s'ouvre de manière à garder l'affichage de la planche (voir ci-dessous).



Il est divisé en quatre sous menu « planche », « graph », « abs. » et « param ».



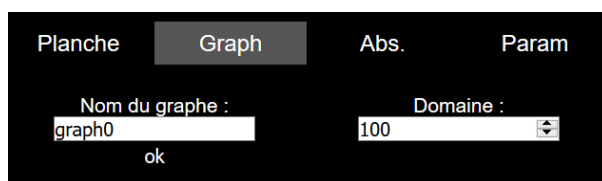
Dans ce menu, les deux zones « graphes » et « paramètres » permettent toutes deux de voir les graphes et paramètres de la planche.

On peut en ajouter en cliquant sur . En ajoutant un nouveau graphe, par défaut il sera nommé « graph » + le nombre de graphes sur la planche.

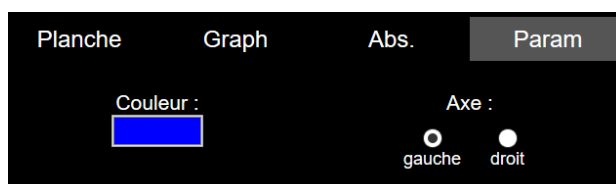
En sélectionnant un graphe ou un paramètre, on a accès à la suppression avec .

On peut ensuite ajouter un paramètre dans le graphe comme expliqué précédemment.

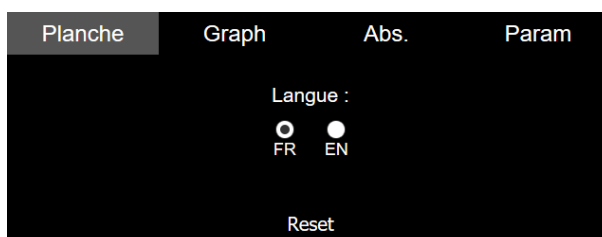
La sélection d'un paramètre ou d'un graphe débloque respectivement les sous menus « param » et « graph ».



Le menu « graph » permet la modification de son nom et de son domaine. Le domaine est la hauteur que prend le graphe sélectionné. Ici, il n'y a qu'un graphe, donc c'est 100 %. Le minimum est de 5 %, et le maximum de 100.

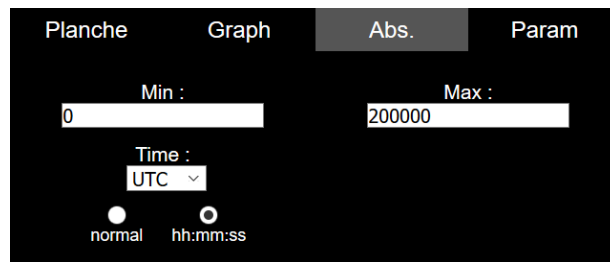


Le menu « param » permet la modification de la couleur, et de l'axe des ordonnées (droite ou gauche) du paramètre sélectionné.



Le menu « planche » permet de changer la langue. Elle concerne le nom des paramètres, qui ont une traduction anglaise et française. Il est également possible de réinitialiser la planche.



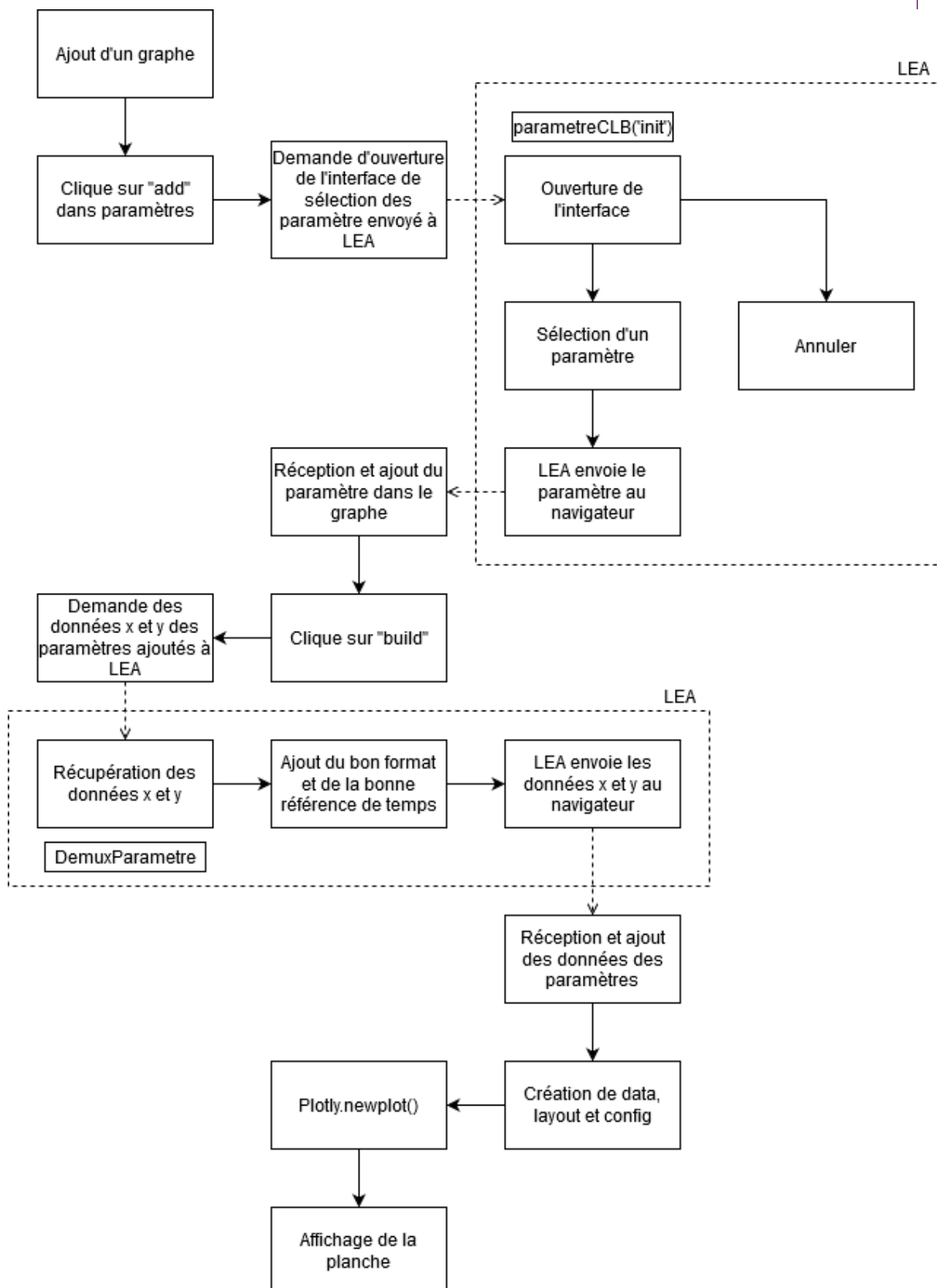


Le menu « abs » permet la modification de l'intervalle de temps (en secondes), avec les deux input min et max. Il permet aussi de choisir parmi les temps de référence, qui sont récupéré via LEA. Ces différents temps sont utilisés pour se retrouver par rapport à l'instant 00:00:00, c'est-à-dire l'accident. Pour finir on peut sélectionner le format d'affichage, en hh:mm:ss ou en seconde (normal).



Le menu déroulant « Fichier » permet d'ouvrir et de sauvegarder une planche, ainsi que de faire une sauvegarde spéciale pour le département Investigations.

En résumé, avec l'interface finale, le déroulement de l'affichage d'une planche est décrit dans le schéma ci-dessous :



### 3.3.3 Faciliter les tests utilisateurs

Le problème de ce prototype était qu'il fallait construire une planche de zéro. Les utilisateurs de LEA auraient été réticents à l'idée de devoir, à chaque projet, créer une planche de zéro, alors qu'elle est déjà faite dans LEA. La solution à ce problème est présente dans cette dernière phrase, la planche existe déjà. Or mon prototype prenait en compte assez de paramètres (légendes, couleur, axe Y gauche ou droite, domaine des graphes...) pour pouvoir convertir une planche de LEA en planche web.

Ainsi, j'ai ajouté un bouton dans LEA à la barre d'outils existante, dans l'affichage de la planche, à la demande de mon chef de projet, suite à un sprint dédié.

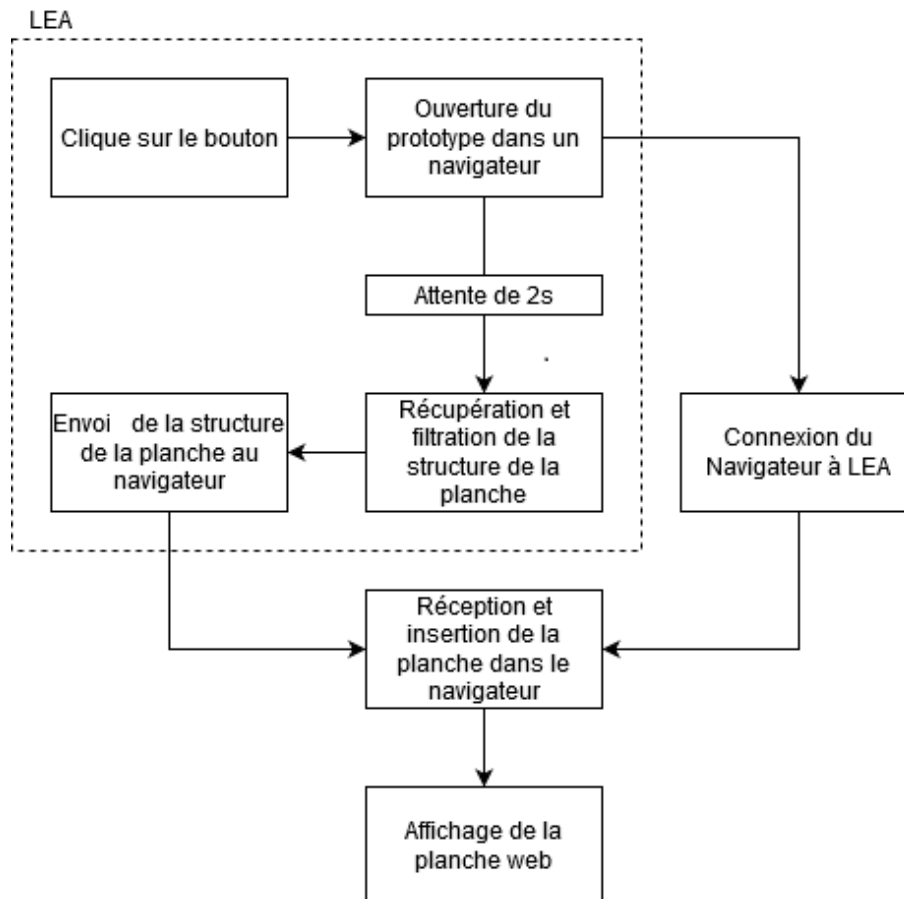
*Barre d'outils existante :*



Le bouton ajouté est le suivant (il n'est pas très beau et pourra faire l'objet d'une amélioration évidente !) :



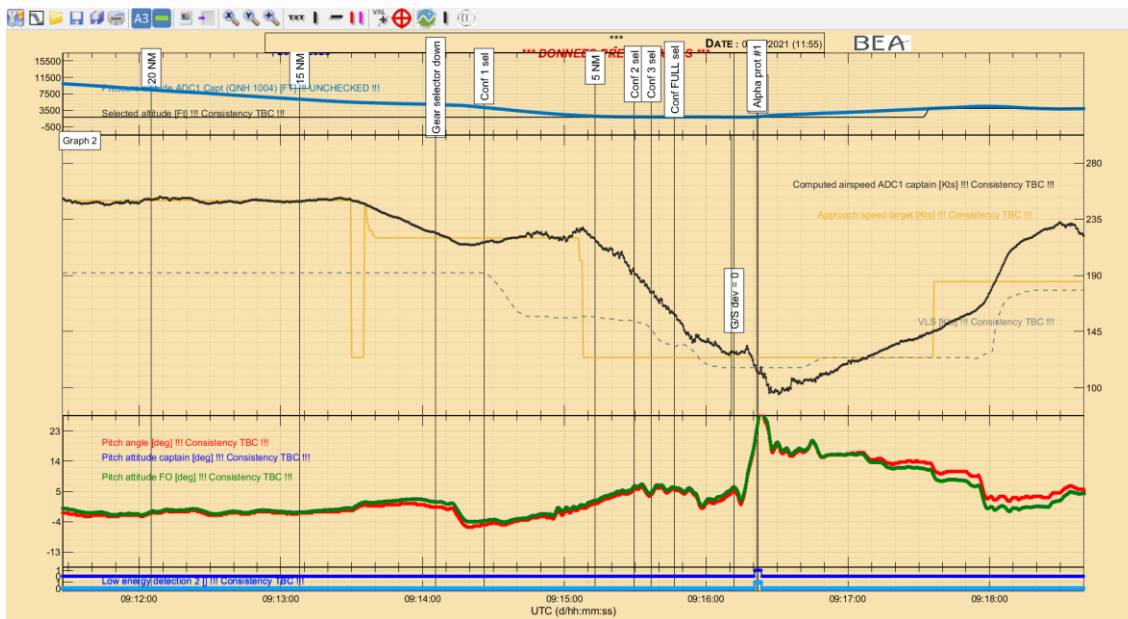
Sa fonction est d'ouvrir la planche LEA en cours de modification dans mon prototype web. Son fonctionnement est décrit dans le schéma ci-dessous.



Finalement je n'ai eu à développer que la fonction du bouton car mon prototype savait déjà créer une planche. Cependant, la structure de la planche web (voir annexe 3) n'étant pas exactement la même que celle de LEA, il fallait procéder à une transformation pour rendre la planche lisible par le prototype. La planche LEA est donc triée avant d'être envoyée au navigateur. Une fois la structure de la planche réceptionnée, la planche web est affichée.

La planche étant envoyée sans les données x et y, elles sont récupérées de la même manière que précédemment entre la réception et l'affichage de la planche web.

### Planche LEA :



### Planche Web :



Remarque : L'ordre des graphes sur le navigateur est inversé, il construit les a planche en partant du bas, contrairement à LEA.

Ce bouton de génération automatique permet ainsi d'avoir un prototype de test accessible très facilement par les utilisateurs et donc, d'obtenir un plus grand nombre de testeurs potentiels.

### 3.3.4 De l'intérêt de la méthode AGILE

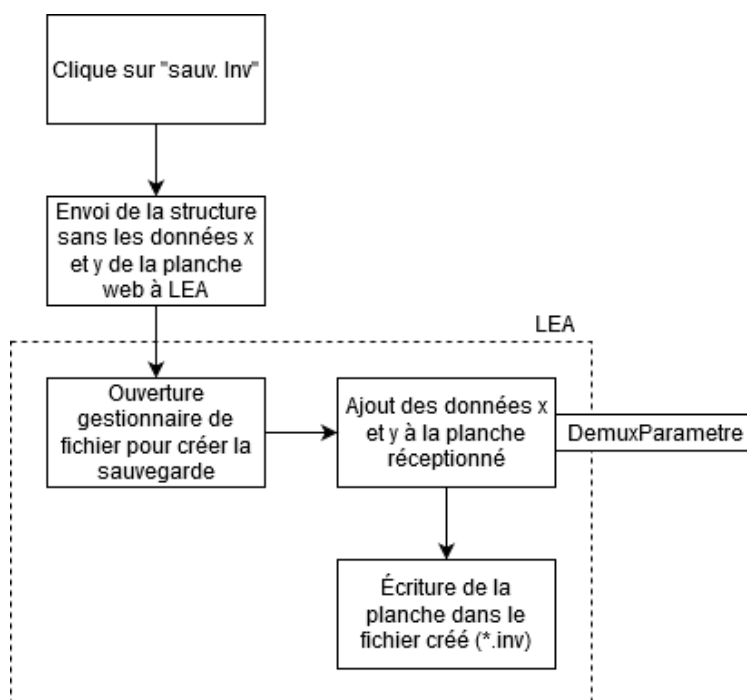
Les enquêteurs généralistes du département Investigations ont besoin de pouvoir lire des paramètres de manière dynamique. C'est-à-dire d'avoir accès à une planche avec toutes les fonctionnalités de lecture actuelles.

Ce besoin n'était pas réalisable jusqu'à ce que je crée ce bouton de génération automatique. Cependant, la réalisation de ce besoin nécessitait des fonctionnalités qui avaient été initialement écartées, comme l'écriture des données x et y dans le fichier de sauvegarde.

Précédemment, j'ai présenté le menu déroulant « fichier » ci-dessous.



J'ai présenté l'option « sauv. Inv » sans détailler. Voici sa présentation détaillée :

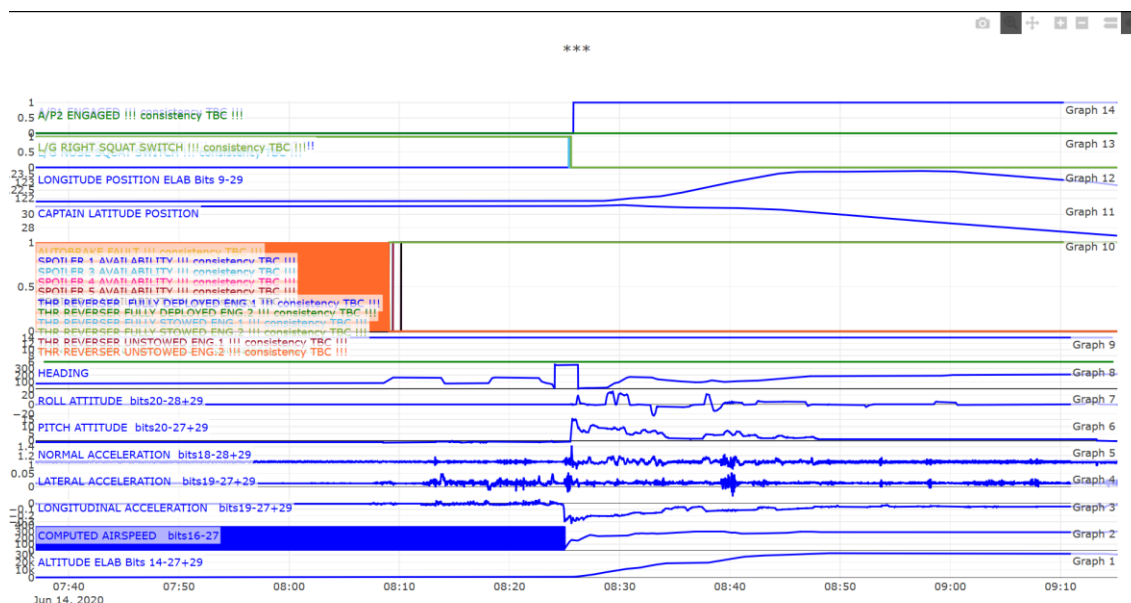


Les données x et y ne sont pas envoyées par le navigateur car l'envoi serait trop important. L'écriture dans le fichier est un point qui pourrait poser problème, n'étant pas optimisé (manque de temps), il y a un délai d'attente entre le moment du clic sur « sauv. inv » et l'écriture complète dans le fichier, de l'ordre de 5 à 10 secondes sur les tests réalisés.

Le fichier .inv créé peut ensuite être envoyé aux enquêteurs Investigations. De leur côté, ils ouvrent un script (voir annexe 6) situé dans le package du prototype qui leur permet de sélectionner un fichier de sauvegarde. Ce script a été ajouté car le prototype était uniquement côté client. Ainsi, j'ai créé un data.js dans le package, ouvert dans le fichier html du prototype, ce data.js est remplacé par le fichier sélectionné grâce au script. Ce script permet donc de contourner l'utilisation d'un serveur web.

Lorsqu'un utilisateur ouvre le prototype sans LEA, le mode « Investigations » est déclenché, bloquant l'utilisation des menus. Les enquêteurs ouvrent donc leur planche en mode lecture seulement.

*Planche ouverte en tant qu'enquêteur généraliste du département Investigations :*



Remarque : la sécurité n'a du tout pas été gérée, si l'utilisateur veut « pirater » (par exemple en ouvrant la console et en tapant une commande, il peut réafficher les menus), c'est possible mais dans ce cas l'interface plantera car il n'a pas accès à LEA.

L'ajout de cette nouvelle fonctionnalité de visualisation pour les enquêteurs généralistes du département Investigations va changer la méthode de communication entre les enquêteurs généralistes du département Investigations et les enquêteurs spécialisés du département Technique.

### 3.4 Utilisation par les enquêteurs

Les enquêteurs du département Technique peuvent utiliser mon prototype en installant le package sur leur machine (voir ci-dessous son arborescence).

```
<PlanchePierreHervelin> 13 639,41 Ko
├── 0001 index.html
├── <bin> 18/06/2021 10:46:24
│   ├── 0001 start.bat
│   └── 0002 master.css
├── <css> 26/05/2021 14:01:42
│   ├── 0001 main.css
│   ├── 0002 master.css
│   └── 0003 UI.css
├── <data> 18/06/2021 15:01:36
│   ├── 0001 data.js
│   └── 0002 webSocket.js
├── <js> 05/07/2021 07:22:16
│   ├── 0001 package-lock.json
│   ├── 0002 plotly.min.js
│   ├── 0003 plotlyChart.js
│   ├── 0004 save.js
│   ├── 0005 UI.js
│   └── 0006 webSocket.js
└── 01/07/2021 11:07:28 8,69 Ko
    ├── 0,63 Ko
    ├── 18/06/2021 15:01:30 0,63 Ko
    ├── 10,77 Ko
    ├── 27/05/2021 12:13:22 0,33 Ko
    ├── 20/05/2021 13:06:46 3,94 Ko
    ├── 29/06/2021 10:18:24 6,50 Ko
    ├── 10 155,91 Ko
    ├── 18/06/2021 19:25:16 10 155,91 Ko
    ├── 3 463,41 Ko
    ├── 05/05/2021 11:22:48 2,77 Ko
    ├── 05/05/2021 16:03:30 3 420,15 Ko
    ├── 01/07/2021 14:37:18 18,69 Ko
    ├── 30/06/2021 16:38:44 3,17 Ko
    ├── 01/07/2021 09:29:20 12,13 Ko
    └── 01/07/2021 10:46:22 6,50 Ko
```

Voici comment se présente le dossier js/ de mon package :

- plotlyChart.js contient les fonctions de lecture et de création de la planche (742 lignes)
- save.js contient les fonctions de sauvegarde d'une planche (167 lignes)
- UI.js contient les fonctions liées à l'affichage de l'interface (484 lignes)
- websocket.js permet la connexion à LEA et contient les fonctions d'envoi et de réception de messages (267 lignes)

Une fois installé, mon fichier « WebPlotServer.m » doit être ajouté à LEA. Ce fichier gère la réception des messages du navigateur envoyés sur le port 8828. Ensuite, il leur suffit de lancer LEA et de cliquer sur le bouton de génération automatique. Ils peuvent de plus lancer le .html dans le package pour construire une planche de zéro.

L'usage pour les enquêteurs du département Investigations diffère. Ils peuvent utiliser le prototype en exécutant start.bat, le script évoqué précédemment, situé dans package/bin/. Une fois exécuté, ils ouvrent le fichier d'extension .inv fournit par un enquêteur spécialisé du département Technique.

Remarque : *plusieurs fichiers peuvent être ouverts en même temps si l'utilisateur n'actualise pas les pages.*

## 4. Expérience acquise

Ce stage au sein du pôle PESA du BEA, m'a apporté énormément. D'un point de vue professionnel, j'ai appris la vie en entreprise. N'ayant jamais eu de grosse expérience en entreprise, j'ai pu poser mes repères tout au long du stage.

D'un point de vue technique, j'ai également appris énormément. Pour commencer, j'ai pu vivre la méthode agile en situation réelle. Cette méthode est assez déstabilisante au début car on avance à l'« aveugle ». Avec du recul, mon ressenti sur cette méthode est qu'elle est rassurante. J'avancais avec un nouvel objectif chaque semaine, ce dernier était toujours réfléchi pour être réalisable. Ainsi je n'étais jamais dépassé par le travail à réaliser.

J'ai de plus pu apprendre beaucoup sur les langages javascript, Matlab et css. J'ai également pris en main pour la première fois des librairies. J'ai pris conscience que l'utilisation de librairies permet un gain de temps considérable plutôt que de partir d'une feuille blanche.

Par ailleurs, c'était la première fois que je développais avec un langage de calcul matriciel tel que Matlab. Matlab est très utile pour l'analyse de données.

Enfin, j'ai eu la chance de travailler sur un projet en phase de réflexion et d'évaluation. Cette expérience a été très enrichissante. C'était pour moi une nouvelle manière d'aborder un produit.



# Conclusion

Mon stage de fin d'études s'est déroulé sur 11 semaines dans les bureaux du BEA (Bureau d'Enquêtes et d'Analyse pour la sécurité de l'aviation civile).

À mon arrivée j'ai été affecté au sein du pôle enregistreurs et systèmes avioniques (PESA) et le BEA m'a demandé de coder une interface web traçant des paramètres de vol. Celle-ci devait permettre de tester des fonctionnalités essentielles de l'IHM existante LEA. La gestion de ce projet s'est faite sous la méthode agile.

L'observation et l'analyse de LEA et de ses utilisateurs m'ont permis dans un premier de capturer les besoins utilisateur « haut niveau » pour sélectionner la librairie web la mieux adaptée. Suite à cette sélection la librairie plotly.js a été sélectionnée.

J'ai ensuite développé un prototype sous un environnement web afin de tester la faisabilité de codage des fonctionnalités principales de création et de lecture de planches de LEA.

Ce prototype n'avait pas initialement vocation à être conservé. Finalement, cette interface sera mise en en pré production au BEA à l'automne pour être testée par les utilisateurs. En particulier les enquêteurs généralistes du département Investigations, pourront avoir accès aux planches fournies par les enquêteurs spécialisés du département Technique, conformément à leur demande. Cette utilisation de mon travail n'était pas prévue et j'en suis très fier.

Ce stage a constitué une expérience très enrichissante car il m'a permis de mettre en application mes connaissances mais aussi de les développer. Développer ce type de projet m'a beaucoup plu et me permet de me projeter avec optimisme dans une carrière professionnelle dans ce domaine.

# ***Annexes***

# Annexe 1

## Liste des fonctionnalités de LEA

Entête :

- Settings window (voir fenêtre plot)
- Redraw a plot (actualiser le plot pour résoudre les bugs)
- Open a plot
- Save/save as
- Print :
  - Fromat papier
  - Orientation
  - Renderer
  - Config avancée
- Format A3/A4
- Orientation de la page
- Export jpg,tif,eps,bmp
- Listing export csv
- Zoom avec loupe (en X ou Y)
- Insérer du texte
- Insérer une ligne verticale
- Insérer une ligne horizontale
- Cursor (avec min/max, variance, écart type... de la zone sélectionnée)
- Display tag-values (min/max des courbes du graph)
- Pointeur (désactiver une valeur sur la courbe)
- cesium

Modifications dynamiques sur les graphes :

- Réglage de la taille d'un graph
- Zoom sur un axe avec molette
- Déplacement sur un axe
- Déplacement d'un label
- Changement de couleurs d'une courbe
- Mettre au premier/second plan une courbe (en cas de superposition de courbe)
- Ajout d'un tag (faire naviguer un point sur la courbe pour avoir ses coordonnées)
- Mettre les coordonnées d'une courbe à droite ou à gauche
- Aligner les légendes à celle sélectionnée
- Éditer le paramètre (modifier le .gri)
- Exporter les paramètres en .mat
- Ligne de paramètres
- Ajout/retrait d'un paramètre
- Ajout/retrait d'events
- Modification du titre de la planche

Fenêtre plots :

- Ajout/suppression d'un graphe
- Ajout/suppression d'une courbe

HEADER :

- Modifs de l'entête (titre, visibilité)
- Changer le langage

GRAPH :

- Changer le nom
- Rendre visible ou non le nom
- Margin left/right
- Height
- Image de fond

ABSCISSE :

- Changer l'unité (TTC/UTC)
- Changer le Format (d/hh:mm:ss)
- Changer le label
- Min/max
- Changer le pas
- Changer la police et la couleur
- Invertir l'axe

ORDONNEES :

- Ordonnées à droite et à gauche
- Min/max
- Changer le pas
- Changer la police et la couleur

PARAMETRE :

- Modifier le nom
- Choix de le placer à droite ou à gauche
- Modification de la largeur de la courbe
- Modification de la largeur des marqueurs
- Modification du style de courbe (pointillés...)
- Modification du style de marqueurs (+, \*, ,,o)
- Changer la police de la légende
- Changer la couleur de la légende
- Rendre visible ou non la légende

# Annexe 2

## Sous menu de « plots »

Plot Graph Abscissa Ordinate Parameter

Linked graphs: ☒ in abscissa ☒ in reading

Header

☒ Visible header ☒ Preliminary data French

Plot title

\*\*\*

☒ Display time in the header

☐ WEB Share

### Sous menu « plot »

Plot Graph Abscissa Ordinate Parameter

Mise en forme

Actual graph name: Graph 1 ☐ Visible

Actual graph height: 10.3708 %  Ticks number: 10.370

Left margin: 5 % Right: 5 %

Image en fond

☐ Active picture Picture name:

Xmin picture: 0 Xmax picture: 1

Ymin picture: 0 Ymax picture: 1

### Sous menu « graph »

Plot Graph **Abscissa** Ordinate Parameter

**Readout**

☐ Auto Time: TCC Format: normal TBegin: -2000000 TEnd: 2000000

**Abscissa axis**

Time: UTC Format: d/hh:mm:ss

Label: UTC (d/hh:mm:ss) ☐ auto

Min scale: 09:11:27.68 Max: 09:18:40.04 Phase: ☐

Min Grad: 09:12:00 Pas: 60 Autoscale

Police: Arial 8

☐ Inverted abscissa axis scale

New plot Apply Close

### Sous menu « abscissa »

Plot Graph **Abscissa** Ordinate Parameter

**Left Axis**

Min scale: -2500 Max: 17500

Graduation Min: -500 Step: 4000 Autoscale

Police: Arial 8

**Right Axis**

Min scale: -1000 Max: 9000

Graduation Min: 0 Step: 2000 Autoscale

Police: Arial 8

☐ Inverted Left ordinate axis scale ☐ Inverted Right ordinate axis scale

New plot Apply Close

### Sous menu « ordinate »

| Plot  | Graph | Abscissa | Ordinate | Parameter |
|---|-------|----------|----------|-----------|
| <div>Parameters</div> <div> <div>Displayed name: <input type="text" value="Selected altitude"/> <input type="button" value="French"/></div> <div> <div>left</div> <div><input checked="" type="checkbox"/> Name &amp; units</div> <div><input type="checkbox"/> Name &amp; source</div> <div><input type="button" value="Edit param"/></div> </div> <div> <div>Tag format: <input type="text" value="%.0f"/></div> <div><input type="checkbox"/> Raw value</div> <div><input type="button" value="Update param"/></div> </div> <div><input type="checkbox"/> Plot with another parameter in abscissa</div> </div> |       |          |          |           |
| <div>Plots</div> <div> <div>Line</div> <div>Markers</div> <div>Color</div> </div> <div> <div>-</div> <div>1</div> <div>.</div> <div>2</div> <div></div> <div><input type="checkbox"/> Param clipping</div> </div>   |       |          |          |           |
| <div>Legend</div> <div> <div><input checked="" type="checkbox"/> Visible</div> <div>Rot (°): <input type="text" value="0"/></div> <div>Police: <input type="text" value="Arial 8"/></div> </div>  |       |          |          |           |
| <div> <input type="button" value="New plot"/> <input type="button" value="Apply"/> <input type="button" value="Close"/> </div>  |       |          |          |           |

Sous menu « parameter »

## Annexe 3

### Structure de la planche web

```
{...}
  date: "20/12/2020"
  dateGetTime: 1608418800000
  firstSave: true
  formatTemps: "hh:mm:ss"
  graph: (4) [...]
    0: {...}
      domain: Array [ 0, 0.10370795316898748 ]
      nom: "Graph 1"
      parametre: (2) [...]
        0: {...}
          axe: "left"
          check: true
          consistant: 0
          couleur: "#262626"
          data: Object { x: (12216) [...], y: (12216) [...] }
          format: "%.0f"
          nom: "Selected altitude"
          nomAffichage: Object { anglais: "Selected altitude", francais: "Selected altitude" }
          source: "q1024facf022 (G)"
          unite: "Ft"
          <prototype>: Object { ... }
          1: Object { unite: "FT", axe: "left", format: "%5.0f", ... }
            length: 2
          <prototype>: Array []
          pas: 0.10370795316898748
          <prototype>: Object { ... }
        1: Object { pas: 0.5698820591731257, nom: "Graph 2", parametre: (3) [...], ... }
        2: Object { pas: 0.2821116132236319, nom: "Graph 3", parametre: (3) [...], ... }
        3: Object { pas: 0.04429837443425492, nom: "Graph 4", parametre: (2) [...], ... }
          length: 4
          <prototype>: Array []
      langue: "français"
      nbParam: 10
      temps: "UTC"
      timeRange: Object { tDebut: -2000000, tFin: 2000000 }
      titre: "****"
      <prototype>: Object { ... }
```



# Annexe 4

## Code du fichier « webPlotServer.m »

```
classdef WebPlotServer < WebSocketServer
    %WEBPLOTSERVER Summary of this class goes here
    % Detailed explanation goes here

    properties
        savePath='';
        saveFile='';
    end

    methods
        function obj = WebPlotServer(varargin)
            %Constructor
            obj@WebSocketServer(varargin{:});
        end
    end
end
methods (Access = protected)
    function onOpen(obj,conn,message)
        global GLOBALS;

        CesiumCLB(obj,'open',message);

        preference=GetPreference('parametre');
        date=GetProjet;
        time=GetTemps;

        content={};
        content.preference=preference;
        content.date=date.date;
        content.time=time;

        json=jsonencode(content);
        lengthContent=num2str(length(json));
        for i=length(lengthContent):1:14
            lengthContent=strcat('0',lengthContent);
        end
        TLVmsg=strcat('05',lengthContent,json);
        conn.send(TLVmsg);

        GLOBALS.network.WebPlot={};
        GLOBALS.network.WebPlot.cnx=conn;
    end
    function onTextMessage(obj,conn,message)
        fprintf(message);
        TLV_type = message(1:2);
        TLV_length = str2double(message(3:17));
        if isnan(TLV_length)
            return
        end
    end
end
```

```

TLV_value = message(18:18+TLV_length-1);

%      Différents types:
%      ENVOI DE MESSAGES AU NAVIGATEUR :
%      00:envoi d'une info à afficher dans la console
%      01:envoi d'un paramètre
%      02:envoi des données des paramètres
%      05:envoi des préférences, temps de références et de la
date
%      06:envoi d'une planche LEA
%      98:envoi d'une sauvegarde
%
%      RECEPTION DE MESSAGES :
%      03:demande d'un parametreCLB('init')
%      04:demande des données d'un paramètre
%      10:demande du max sur un intervalle
%      (ce type n'est plus codé dans LEA,
%      tout envois ne sera traité)
%      96:demande de sauvegarde INV
%      97:demande de suppression de l'autosave OU
%      demande s'il existe une autosave
%      (grâce à la structure suivante : {type:'delete'} OU
%      {type:'isAutoSave'})
%      98:demande d'ouverture de sauvegarde
%      99:demande de sauvegarde

switch (TLV_type)
    case '03'
        % ouverture de la fenêtre de sélection de
paramètre
        % envoi du paramètre sélectionné
        parametre={};

        searchParametre = ParametreCLB('init');

        parametre.unite=searchParametre.unite;
        parametre.nom=searchParametre.nom;
        parametre.couleur=searchParametre.couleur;
        parametre.source=searchParametre.source;

parametre.nomAffichage=searchParametre.nomAffichage;
        parametre.axe=searchParametre.axe;
        parametre.couleur=searchParametre.couleur;

        if searchParametre.verifie
            parametre.check=true;
        else
            parametre.check=false;
        end
        parametre.consistent=searchParametre.valide;

        json=jsonencode(parametre);
        lengthContent=num2str(length(json));
        for i=length(lengthContent):1:14
            % on remplit avec des 0 jusqu'à avoir 15
caractères
            lengthContent=strcat('0',lengthContent);
        end

        TLVmsg=strcat('01',lengthContent,json);

```

```

        conn.send(TLVmsg);
    case '04'
        % reception : une structure paramètre incomplète
        % Objectif : envoi des données x et y du paramètre
        % format de la structure reçu :
        % param={
        %     nom:string
        %     source:string
        %     lastParam:bool
        %     temps:string
        %     formatTemps:string
        %     date:int
        %     indice:array[],length=2
        % }
        % les requêtes de données sont envoyés à la suite

au
        % moment du build, elles sont envoyés une par une,
il
        % faut donc savoir laquelle est la dernière

        % lastParam==true si c'est la dernier paramètre
        % sinon false

        % date est un entier contenant le nombre de
        % millisecondes passé depuis le 1er janvier 1970
        % jusqu'à la date du projet

        % indice est un tableau de 2 valeurs permettant de
        % définir la position dans la planche du paramètre
        % indice(1)==indice du graphe
        % indice(2)==indice du paramètre dans le graphe

        identite=jsondecode(TLV_value);
        parametre={};

        % on récupère le paramètre complet
        [errCode,index]=listeParamManagement(...
            'getIndexFromParam',identite.param);
        [errCode,paramStruct]=listeParamManagement(...
            'getParamFromIndex',index.source,index.param);

        data=DemuxParametre(...

paramStruct,identite.tDebut,identite.tFin,'TCC');

        parametre.nom=identite.param.nom;
        parametre.indice=identite.param.indice;

        parametre.data={};

        temps=GetTemps;
        planche=GetPlanche;

        % on cherche s'il y a un offset avec le temps de
        % référence sélectionné
        for i=1:length(temps)
            if strcmp(identite.param.temps,temps(i).nom)
                if ~isempty(temps(i).offset)
                    offset=temps(i).offset.tOffset;
                else
                    offset=0;
                end
            end
        end
    end
end

```

```

        end
    end
    if strcmp(identite.param.formatTemps, 'hh:mm:ss')
        produit=1000; % On multiplie par 1000 pour
                       % avoir des millisecondes
    else
        produit=1; % format normal donc pas de produit
    end

    parametre.data.x=(...
        (data.temps*produit)+identite.param.date)+...
        (offset*produit);

    parametre.abscisse={};
    parametre.abscisse.min=...
        (...

(planche.graphe(1).abscisse.echelleMin*produit)+...
        identite.param.date)+(offset*produit);
    parametre.abscisse.max=...
        (...

(planche.graphe(1).abscisse.echelleMax*produit)+...
        identite.param.date)+(offset*produit);

    parametre.data.y=data.data;

    json=jsonencode(parametre);
    lengthContent=num2str(length(json));
    for i=length(lengthContent):1:14
        lengthContent=strcat('0',lengthContent);
    end
    TLVmsg=strcat('02',lengthContent,json);
    conn.send(TLVmsg);

    if identite.param.lastParam
        lengthData='4';
        for i=1:1:14
            lengthData=strcat('0',lengthData);
        end
        conn.send(strcat('00',lengthData,'load'));
    end

case '97'
    % suppression ou vérification de l'autosave
    % si content.type=='delete' -> suppression de
    % l'autosave
    % si content.type=='isAutoSave' -> renvoi de
    % 'isAutoSave' ou 'noAutoSave'
    dir=fullfile(fileparts(...
        GetProjet('nomProjet')), 'Plots');
    content=jsondecode(TLV_value);

    switch(content.type)
        case 'delete'
            if content.isAutoSave
                delete(fullfile(dir,'auto-
save.phvn'));

            end
            lengthData='7';
            for i=1:1:14
                lengthData=strcat('0',lengthData);

```

```

end

conn.send(strcat('00',lengthData,'deleted'));
case 'isAutoSave'
[ID]=fopen(fullfile(dir,'auto-
save.phvn'));
lengthData='10';
for i=2:1:14
lengthData=strcat('0',lengthData);
end
if ID== -1
conn.send(strcat(...
'00',lengthData,'noAutoSave'));
else
conn.send(strcat(...
'00',lengthData,'isAutoSave'));
end
end
case '96'
% sauvegarde INV
% les données x et y des paramètres de la
sauvegarde ne
% sont pas réceptionné
% matlab s'occupe de les récupérer puis
d'enregistrer
% la sauvegarde dans le fichier créé (.inv)
% la sauvegarde est une chaine de caractère

dir=fullfile(...
fileparts(GetProjet('nomProjet')), 'Plots');
save=jsondecode(TLV_value);

[file,path]=uiputfile(strcat(dir,'/*.inv'),'Save
as');

for i=1:length(save.planche.graph)
for
j=1:length(save.planche.graph(i).parametre)
param={};
param.source=...

save.planche.graph(i).parametre(j).source;
param.nom=...

save.planche.graph(i).parametre(j).nom;

[errCode,index]=listeParamManagement(...
'getIndexFromParam',param);

[errCode,paramStruct]=listeParamManagement(...
'getParamFromIndex',...
index.source,index.param);

data=DemuxParametre(...
paramStruct,...
save.planche.timeRange.tDebut,...
save.planche.timeRange.tFin,'TCC');

temps=GetTemps;

```

```

        % même système que vu pour le type 04
        for k=1:length(temps)
            if
                strcmp(save.planche.temps,temps(k).nom)
                    if ~isempty(temps(k).offset)

                        offset=temps(k).offset.tOffset;

                    else
                        offset=0;
                    end
                end
            end

            if
                strcmp(save.planche.formatTemps,'hh:mm:ss')
                    produit=1000;
                else
                    produit=1;
                end

            save.planche.graph(i).parametre(j).data.x=(...
                (data.temps*produit)+...

            save.planche.dateGetTime)+(offset*produit);

            save.planche.graph(i).parametre(j).data.y=...
                data.data;

            end
        end
        content=strcat('var content=',jsonencode(save));
        c=cellstr(content);

        f=fopen(fullfile(path,file),'wt');
        fprintf(f,'%s',c{:});
        fclose(f);

        lengthData='5';
        for i=1:1:14
            lengthData=strcat('0',lengthData);
        end

        conn.send(strcat('00',lengthData,'saved'));

    case '98'
        % ouvrir une sauvegarde
        % la reception est soit 'open' soit 'openAutoSave'
        dir=fullfile(fileparts(GetProjet('nomProjet')),...
            'Plots');
        switch (TLV_value)
            case 'open'

                [file,path]=uigetfile(strcat(dir,'/*.phvn'),...
                    'Open plot');
                obj.saveFile=file;
                obj.savePath=path;
            case 'openAutoSave'
                file='auto-save.phvn';
                path=dir;

        end
    end
end

```

```

f=fopen(fullfile(path,file),'r');
save=fgetl(f);

lengthData=num2str(length(save));
for i=length(lengthData):1:14
    lengthData=strcat('0',lengthData);
end
TLVmsg=strcat('98',lengthData,save);
conn.send(TLVmsg);

lengthData='4';
for i=length(lengthData):1:14
    lengthData=strcat('0',lengthData);
end
TLVmsg=strcat('00',lengthData,'open');
conn.send(TLVmsg);

fclose(f);

case '99'
    % sauvegarde d'une planche navigateur
    % si save.planche.firstSave==true on ouvre le
    % gestionnaire de fichier
    % sinon si save.isAutoSave==true, c'est une
autoSave
    % donc on connais le nom du fichier,
onl'enregistre

    % dans le projet/plots
    % sinon on connais déjà le chemin du fichier
    % (obj.saveFile et obj.savePath)

    dir=fullfile(fileparts(GetProjet(...
        'nomProjet')), 'Plots');
    save=jsondecode(TLV_value);
    if save.planche.firstSave
        [file,path]=uinputfile(...
            strcat(dir,'/*.phvn'),'Save as');

        obj.saveFile=file;
        obj.savePath=path;
    else
        if save.isAutoSave
            file='auto-save.phvn';
            path=dir;
        else
            file=obj.saveFile;
            path=obj.savePath;
        end
    end
end

f=fopen(fullfile(path,file),'w');
fprintf(f,TLV_value);
fclose(f);

lengthData='5';
for i=1:1:14
    lengthData=strcat('0',lengthData);
end

conn.send(strcat('00',lengthData,'saved'));

end

```

```

end

function onBinaryMessage(obj,conn,bytearray)
    % This function sends an echo back to the client
    disp('Receipt binary text');
    %conn.send(bytearray); % Echo
end

function onError(obj,conn,message)
    fprintf('%s\n',message)
end

function onClose(obj,conn,message)
    CesiumCLB(obj, 'closed',message);
    %
    %     global GLOBALS;
    %     for i=1:length(GLOBALS.network.WebPlot.cnx)
    %         disp(conn);
    %         if
GLOBALS.network.WebPlot.cnx(i).HashCode==conn.HashCode
    %             GLOBALS.network.WebPlot.cnx(i)=[];
    %         end
    %     end
end
end
end
end

```



# Annexe 5

## Code de « websocket.js »

```
1  window.onload=function(){
2      try {
3          ws=new WebSocket('ws://localhost:8828');
4          loading();
5      } catch (e) {
6          console.error(e);
7      }
8      ws.onerror=function(error){
9          console.error(error);
10         $('#TEC').hide();
11         stopLoading();
12         loadINV();
13     }
14     ws.onopen=function(){
15         console.log('Connexion établie');
16         $('#TEC').show();
17         stopLoading();
18     }
19     ws.onclose=function(){
20         console.log('Connexion perdue');
21     }
22     ws.onmessage=function(message){
23         if (isTLV(message.data)){
24             TLVconvert(message.data);
25         }else {
26             try {
27                 console.log(JSON.parse(message.data));
28             } catch (e) {
29                 console.log(message.data);
30             }
31         }
32     }
```

```

33 }
34
35 function dataRequest(param=null){
36     //demande de d'un parametreCLB('init') OU des données d'un paramètre
37     //SI param==null alors on demande parametreCLB('init')
38     //ATTENTION param doit contenir qu'UN SEUL paramètre (voir plotlyChart.js ligne 631)
39     var msgTLV='03',
40         content=new Object();
41
42     if(param){
43         content.param=param;
44         msgTLV='04';
45     }
46     content.tDebut=planche.timeRange.tDebut;
47     content.tFin=planche.timeRange.tFin;
48
49     var json=JSON.stringify(content);
50     msgTLV+=TLVlength(json.length)+json;
51
52     ws.send(msgTLV);
53 }
54 function ifData(input){
55     if (input.val()!=''){
56         if (parseInt(input.val())!=NaN){
57             return true;
58         }
59     }
60     return false;
61 }
62 function maxRequest(rangeStart,rangeEnd){
63     //fonction envoyant une demande du maximum sur l'interval donné à LEA
64     //note : cette fonction n'est pas utilisée

```

```

65     var msgTLV='10',
66         content={};
67
68     content.rangeStart=rangeStart;
69     content.rangeEnd=rangeEnd;
70
71     var json=JSON.stringify(content);
72     msgTLV+=TLVlength(json.length)+json;
73
74     ws.send(msgTLV);
75 }
76
77 function addParametre(parametre){
78     var graph=getSelectedGraph();
79     var graphIndice=parseInt(graph.getAttribute('data-index')),
80         graphName=graph.innerText,
81         parametreIndice=document.querySelectorAll('#paramName').length;
82
83     removeParam();
84     planche.graph[graphIndice].parametre.push({
85         nom:parametre.nom,
86         source:parametre.source,
87         couleur:convertRGB(parametre),
88         axe:parametre.axe,
89         consistent:parametre.consistent,
90         check:parametre.check,
91         nomAffichage:parametre.nomAffichage,
92         data:{
93             x:[],
94             y:[],
95         },
96         indice:[

```

```

97     graphIndice,
98     parametreIndice
99     ]
100 });
101 planche.nbParam+=1;
102
103 affichParam(graphIndice);
104 }
105
106 function TLVlength(length){
107     var lengthTLV=length.toString();
108     for(i=(length.toString()).length;i<15;i++){
109         lengthTLV='0'+lengthTLV;
110     }
111     return lengthTLV;
112 }
113
114 function isTLV(message){
115     if(message.length<17){
116         return false;
117     }else {
118         var length=message.substr(2,15);
119         if ((message.substr(16,parseInt(length))).length!=length) {
120             return false;
121         }else {
122             return true;
123         }
124         switch (message.substr(0,2)){
125             case '01':
126                 return true;
127                 break;
128             case '02':

```

```

129         return true;
130         break;
131     case '06':
132         return true;
133         break;
134     default:
135         return false;
136     }
137 }
138 }
139 /*
140 Différents types:
141 RECEPTION DE MESSAGES LEA :
142 00:reception d'une info à afficher dans la console
143 01:reception d'un paramètre
144 02:reception des données des paramètres
145 05:reception des préférences, temps de références et de la date
146 06:reception d'une planche LEA
147 98:reception d'une sauvegarde
148
149 ENVOIS DE MESSAGES A LEA:
150 03:demande d'un parametreCLB('init')
151 04:demande des données d'un paramètre
152 10:demande du max sur un intervalle (ce type n'est plus codé dans LEA, tout envois ne sera traité)
153 96:demande de sauvegarde INV
154 97:demande de suppression de l'autosave OU demande s'il existe une autosave
155     (grâce à la structure suivante : {type:'delete'} OU {type:'isAutoSave'})
156 98:demande d'ouverture de sauvegarde
157 99:demande de sauvegarde
158 */
159 function TLVconvert(message){
160     var length=parseInt(message.substr(2,15));

```

```

161     switch (message.substr(0,2)){
162         case '00':
163             type0(message.substr(17,length));
164             break;
165         case '01':
166             type1(message.substr(17,length));
167             break;
168         case '02':
169             type2(message.substr(17,length));
170             break;
171         case '05':
172             type5(message.substr(17,length));
173             break;
174         case '06':
175             type6(message.substr(17,length));
176             break;
177         case '98':
178             type98(message.substr(17,length));
179         case '99':
180             type99(message.substr(17,length));
181             break;
182         default:
183             console.log(message);
184             console.error(new Error('Pas de la norme TLV'));
185             return;
186     }
187 }
188
189 function type1(parameter){
190     //reception d'un paramètre
191     parameter=JSON.parse(parameter);
192     addParametre(parameter);

```

```

193     unlockContent('param');
194
195     $('#paramOptions').click();
196 }
197 function type2(data){
198     //reception des données d'un paramètre
199     data=JSON.parse(data);
200
201     planche.graph[data.indice[0]].parametre[data.indice[1]].data.x=data.data.x;
202     planche.graph[data.indice[0]].parametre[data.indice[1]].data.y=data.data.y;
203 }
204 function type5(content){
205     //reception des préférences, temps de références et de la date
206     content=JSON.parse(content);
207     planche.preference=content.preference;
208     planche.date=content.date;
209     time=content.time;
210
211     getOptionTime();
212 }
213 function type6(content) {
214     //reception d'une planche LEA
215     content=JSON.parse(content);
216
217     console.log(content);
218     content=plancheVerif(content);
219
220     planche=content;
221     buildDomains('build');
222
223     refresh();
224     loadPlanche();

```

```

225 }
226 function type98(data){
227     //reception d'une sauvegarde
228     isSave=true;
229     data=JSON.parse(data);
230
231     planche=data.planche;
232     layout=data.layout;
233     refresh();
234
235     loadPlanche();
236 }
237 function type0(info){
238     //reception d'une info à afficher dans la console
239     switch (info){
240         case 'saved':
241             stopLoading();
242             console.log('[LEA]:'+info);
243             $('#saveMessage').show();// affichage d'un message précisant lorsque la sauvegarde s'est effectué
244             $('#saveMessage').fadeIn(1000,function(){
245                 $('#saveMessage').fadeOut(1000,function(){
246                     $('#saveMessage').hide();
247                 })
248             })
249             break;
250         case 'deleted':
251             console.log('[LEA]:'+info);
252             break;
253         case 'isAutoSave':
254             openAutoSave();
255             console.log('[LEA]:'+info);
256             break;

```

```

257         case 'load': // une fois que les données x et y sont chargées
258             console.log('[LEA]:'+info);
259             stopLoading();
260             openPlotOptions(close=true);
261             refresh();
262             plotlyBuild();
263             break;
264         default:
265             console.log('[LEA]:'+info);
266     }
267 }

```



# Annexe 6

## Code du script start.bat

```
1 k# : start.bat
2 :: launches a File... Open sort of file chooser and outputs choice(s) to the console
3
4 @echo off
5 setlocal
6
7 for /f "delims=" %%I in ('powershell -nopprofile "iex (${$?} | out-string)"') do (
8     echo Opening %%I ...
9     copy %%I %~f0\...\data\data.js
10    start firefox file:///~f0\...\index.html
11 )
12 goto:EOF
13
14 : end Batch portion / begin Power Shell hybrid chimera #>
15
16 Add-Type -AssemblyName System.Windows.Forms
17
18 $f = New-Object System.Windows.Forms.OpenFileDialog -Property @{
19     InitialDirectory = [Environment]::GetFolderPath('Desktop')
20     Filter = 'Planche (*.inv)|*.inv'
21 }
22 $null = $f.ShowDialog()
23 $f.FileName
24
```



Bureau d'Enquêtes et d'Analyses  
pour la sécurité de l'aviation civile

Zone Sud - 200 rue de Paris  
Aéroport du Bourget  
93352 Le Bourget Cedex - France  
T : +33 1 49 92 72 00 - F : +33 1 49 92 72 03  
[www.bea.aero](http://www.bea.aero)