



Ecole Polytechnique de l'Université de Tours
Département Informatique
64 avenue Jean Portalis
37200 Tours, France
Tél. +33 (0)2 47 36 14 14
polytech.univ-tours.fr

Projet Recherche & Développement 2021-2022

RFAI2

Génération automatique de tableau de bord



POLYTECH®
TOURS

Tuteur académique
Gilles VENTURINI

Étudiant
Nathanaël DAUVOIS (DI5)

3 avril 2022



Liste des intervenants

Nom	Email	Qualité
Nathanaël DAUVOIS	nathanael.dauvois@etu.univ-tours.fr	Étudiant DI5
Gilles VENTURINI	gilles.venturini@univ-tours.fr	Tuteur académique, Département Informatique



Avertissement

Ce document a été rédigé par Nathanaël DAUVOIS susnommé l'auteur.

L'Ecole Polytechnique de l'Université de Tours est représentée par Gilles VENTURINI susnommé le tuteur académique.

Par l'utilisation de ce modèle de document, l'ensemble des intervenants du projet acceptent les conditions définies ci-après.

L'auteur reconnaît assumer l'entière responsabilité du contenu du document ainsi que toutes suites judiciaires qui pourraient en découler du fait du non respect des lois ou des droits d'auteur.

L'auteur atteste que les propos du document sont sincères et assume l'entière responsabilité de la véracité des propos.

L'auteur atteste ne pas s'approprier le travail d'autrui et que le document ne contient aucun plagiat.

L'auteur atteste que le document ne contient aucun propos diffamatoire ou condamnable devant la loi.

L'auteur reconnaît qu'il ne peut diffuser ce document en partie ou en intégralité sous quelque forme que ce soit sans l'accord préalable du tuteur académique et de l'entreprise.

L'auteur autorise l'école polytechnique de l'université François Rabelais de Tours à diffuser tout ou partie de ce document, sous quelque forme que ce soit, y compris après transformation en citant la source. Cette diffusion devra se faire gracieusement et être accompagnée du présent avertissement.



Pour citer ce document

Nathanaël DAUVOIS, *RFAI2: Génération automatique de tableau de bord*, Projet Recherche & Développement, Ecole Polytechnique de l'Université de Tours, Tours, France, 2021-2022.

```
@mastersthesis{
  author={DAUVOIS, Nathanaël},
  title={RFAI2: Génération automatique de tableau de bord},
  type={Projet Recherche \& Développement},
  school={Ecole Polytechnique de l'Université de Tours},
  address={Tours, France},
  year={2021-2022}
}
```

Table des matières

Liste des intervenants	a
Avertissement	b
Pour citer ce document	c
Table des matières	i
Table des figures	v
1 Introduction	1
1 Acteurs, enjeux et contexte	1
2 Objectifs	1
3 Hypothèses	1
2 Description générale	2
1 Environnement du projet	2
2 Caractéristiques des utilisateurs	3
3 Fonctionnalités du système	3
3 État de l'art / Veille technologique	5
1 Services de visualisation de données existants.....	5
Tableau	5
PowerBI	5
Microsoft Excel	5
2 Bibliothèques JavaScript	5
Angular	5
D3.js	6

Vega-Lite	6
4 Analyse et conception	7
1 Analyse.....	7
1.1 Spécifications.....	7
2 Modélisation proposée.....	7
5 Mise en œuvre	9
1 Outils et librairies utilisés	9
2 Analyse des résultats, évaluation, qualité	9
3 Principales IHM.....	10
3.1 IHM 1 : Sélection des fichiers.....	10
3.2 IHM 2 : Affichage d'un tableau de bord	11
6 Bilan et conclusion	12
1 Bilan du semestre 9	12
2 Bilan du semestre 10.....	12
3 Bilan sur la qualité	13
4 Bilan auto-critique.....	14
Annexes	15
A Planification, gestion de projet	16
1 Evolution du projet	16
2 Description des tâches.....	16
Tâche 1 : Dessin d'une visualisation.....	16
Tâche 2 : Lecture des fichiers.....	17
Tâche 3 : Agencement des visualisations.....	17
Tâche 4 : Sélection des données	17
Tâche 5 : Interactivité	17
Tâche 6 : Test	17
B Description des interfaces	18
1 Interfaces homme/machine.....	18
2 Interfaces logiciel/logiciel	18
C Cahier de Spécifications	19
1 Spécifications Fonctionnelles	19
1.1 Définition de la fonction : affichage d'un tableau de bord	19
Description.....	19
Traitement	19

1.2	Définition de la fonction : lecture du fichier de définition de tableau de bord	20
	Description.....	20
1.3	Définition de la fonction : lecture du fichier de jeu de données.....	20
	Description.....	20
1.4	Définition de la fonction : dessin d'une visualisation.....	20
	Description.....	20
	Traitement	20
1.5	Définition de la fonction : agencement des visualisations	20
	Description.....	21
	Traitement	21
1.6	Définition de la fonction : sélection de données.....	21
	Description.....	21
	Traitement	21
	Exemples de sélection.....	21
1.7	Définition de la fonction : interactivité	21
	Description.....	21
	Traitement	22
	Exemple d'interaction.....	22
2	Spécifications non fonctionnelles	22
2.1	Contraintes de développement et conception	22
2.2	Contraintes de fonctionnement et d'exploitation.....	22
2.2.1	Performances.....	22
2.2.2	Capacités	22
2.2.3	Sécurité	22
D	Cahier du développeur	23
1	Introduction	23
2	Diagrammes architecturaux et UML	24
3	Descriptions détaillées de données exploitées.....	25
3.1	Exemple : tableau de bord du jeu de données Iris.....	26
4	Descriptions détaillées des classes, modules, réalisations.....	28
4.1	Module csv_reader.....	28
4.2	Module dashboard	28
4.2.1	Classe Dashboard	28
4.2.2	Classe Slide	28
4.2.3	Classe Visualization.....	29
4.3	Module visualisations.....	29
4.4	Module index.....	29
E	Document d'installation	31

F	Document d'utilisation	32
1	Erreurs	32
G	Cahier de test	33
1	Tests d'intégration	33
H	Bibliographie	35
I	Glossaire	36
J	Acronymes	37



Table des figures

2	Description générale	
2.1	Environnement du projet.....	2
2.2	Diagramme de séquence du système	3
4	Analyse et conception	
4.1	Le diagramme de classe représentant un tableau de bord.....	8
5	Mise en œuvre	
5.1	L'IHM finale de sélection des fichiers.....	10
5.2	L'IHM finale de sélection des fichiers lorsqu'une erreur a eu lieu.....	10
5.3	L'IHM finale d'affichage d'un tableau de bord	11
6	Bilan et conclusion	
6.1	Tableau de bord contenant un arbre à deux niveaux d'imbrication	13
A	Planification, gestion de projet	
A.1	Le diagramme de Gantt initial	16
A.2	Le diagramme de Gantt final	16
B	Description des interfaces	
B.1	Interface de chargement d'un tableau de bord	18
D	Cahier du développeur	
D.1	Diagramme de séquence du système	24

D.2	Diagramme de classe de la représentation d'un tableau de bord.....	25
D.3	Extrait du jeu de données Iris	26
D.4	Définition d'un tableau de bord pour le jeu de données Iris	26
D.5	Tableau de bord dessiné.....	27

1

Introduction

1 Acteurs, enjeux et contexte

Le projet consiste à créer un site web de visualisation de tableau de bord. La définition du tableau de bord à afficher est générée par un autre site web, développé par un doctorant, à partir d'un jeu de données.

2 Objectifs

L'objectif principal du projet est de créer un outil de visualisation de données sous la forme d'un tableau de bord web. Les types de visualisations et leurs emplacements sont définis par un fichier généré par l'application développée par le doctorant.

3 Hypothèses

Au début du développement, je devrai avoir le format final du fichier de définition d'un tableau de bord. Sinon, je travaillerai en priorité sur les tâches de dessin et d'agencement des visualisations.

2

Description générale

1 Environnement du projet

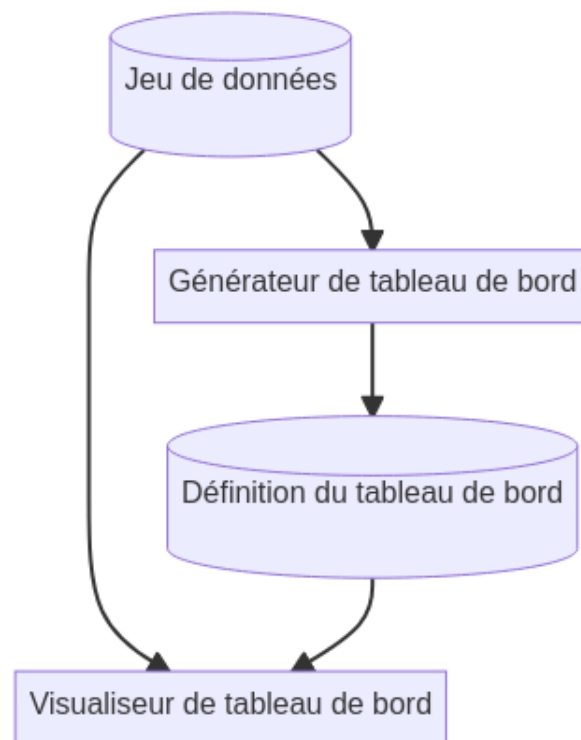


Figure 2.1 – *Environnement du projet*

Ce projet fait partie d'un plus grand projet, où une autre application (le générateur de tableau de bord) génère un fichier de description d'un tableau de bord à partir d'un jeu de données. Ce fichier est ensuite donné à l'application décrite dans ce document (le visualiseur de tableau de bord), ainsi que le jeu de données, et le tableau de bord est affiché.

2 Caractéristiques des utilisateurs

Les utilisateurs sont des scientifiques qui ne sont pas forcément à l'aise avec l'informatique.

3 Fonctionnalités du système

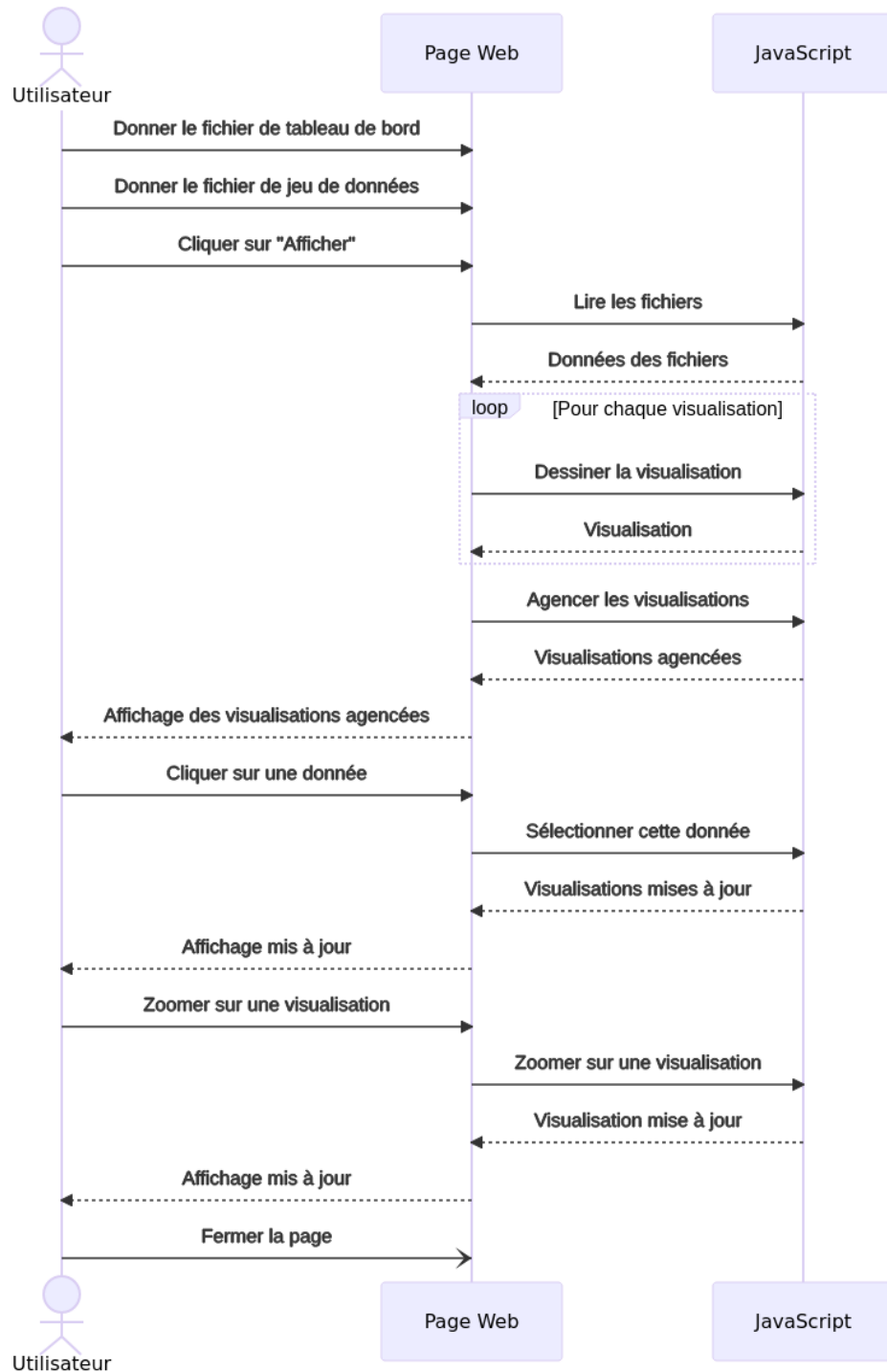


Figure 2.2 – Diagramme de séquence du système

Le système est composé de la page Web avec laquelle l'utilisateur peut interagir et du code

JavaScript qui réagit à ces interactions. La page Web relaie donc la demande de l’affichage d’un tableau de bord et la sélection de données au code JavaScript.

3

État de l'art / Veille technologique

1 Services de visualisation de données existants

Ici, je présente les services existants et leurs fonctionnalités.

Tableau

Tableau est un service de visualisation de données destiné aux entreprises. Il permet d'interroger des bases de données directement et de créer des tableaux de bord interactifs, automatiquement optimisés pour représenter les données le plus clairement possible et personnalisables graphiquement. Ces tableaux de bord peuvent ensuite être partagés en privé ou publiquement. [2]

PowerBI

PowerBI est un service de visualisation de données destiné aux entreprises, concurrent de Tableau. Lui aussi peut créer des tableaux de bords optimisés mais avec une personnalisation moindre.

Microsoft Excel

Microsoft Excel est un logiciel de tableur qui permet de créer des visualisations à partir de jeux de données. Ces visualisations sont créées manuellement par l'utilisateur, sans optimisation de la représentation pour une lecture plus claire.

2 Bibliothèques JavaScript

Ici, je présente les bibliothèques JavaScript que nous avons considérées.

Angular

Angular est une bibliothèque facilitant la création de pages web complexes en JavaScript.

Nous avons choisi de ne pas l'utiliser pour le projet car elle est principalement destinée à créer des interfaces. Notre besoin est plutôt de dessiner des diagrammes, tandis que la création de l'interface peut être faite en HTML directement.

D3.js

D3.js est une librairie bas niveau de manipulation du **Document Object Model (DOM)** en JavaScript. Elle simplifie grandement des opérations sur le DOM. Par exemple, le code suivant, qui change la couleur du texte de tous les paragraphes de la page web en bleu :

```
1 var paragraphs = document.getElementsByTagName("p");
2 for (var i = 0; i < paragraphs.length; i++) {
3   var paragraph = paragraphs.item(i);
4   paragraph.style.setProperty("color", "blue", null);
5 }
```

Devient ceci avec D3.js :

```
1 d3.selectAll("p").style("color", "blue");
```

D3.js permet également de manipuler le DOM, le dessin en SVG, les animations et les interactions utilisateurs. Cela en faisait un excellent candidat pour notre cas d'utilisation.

Vega-Lite

Vega-Lite est un format de spécification de visualisations concis et simple, qui peut être inséré dans une page Web et interprété par la librairie JavaScript. [1]

Nous avons décidé d'utiliser cette outil pour dessiner et agencer les diagrammes car cela nous a semblé plus ergonomique.

4

Analyse et conception

1 Analyse

1.1 Spécifications

Le projet est composé de six fonctionnalités principales :

- La lecture des fichiers ;
- Le dessin des visualisations ;
- L'agencement de ces visualisations en un tableau de bord ;
- L'affichage de ce tableau de bord ;
- La sélection et l'interactivité entre ces visualisations.

Les fichiers à lire sont le jeu de données et la définition du tableau de bord. Les deux sont au format **CSV**, et le fichier de définition de tableau de bord contient des colonnes spécifiques qui décrivent les visualisations.

Le projet sera implémenté sous la forme d'un site web. Les données ne doivent pas être envoyées sur un serveur par soucis de confidentialité, et donc le code doit fonctionner entièrement sur la machine de l'utilisateur.

Le langage utilisé est, par conséquent, JavaScript. L'apparence de la page web est réalisée en HTML et CSS, et le tableau de bord est dessiné avec la librairie Vega-Lite.

2 Modélisation proposée

Les fonctionnalités sont découpées en plusieurs modules :

- "index" : la construction du **DOM** et la gestion des événements sur la page web ;
- "csv_reader" : la lecture de fichiers **CSV** ;
- "dashboard" : la modélisation du tableau de bord et les fonctions d'agencement des visualisations ;
- "visualisations" : les fonctions de dessin des visualisations spécifiques au format Vega-Lite.

Chacun de ces modules contient les fonctions nécessaires pour accomplir ses tâches, et le module "dashboard" contient les classes suivantes :

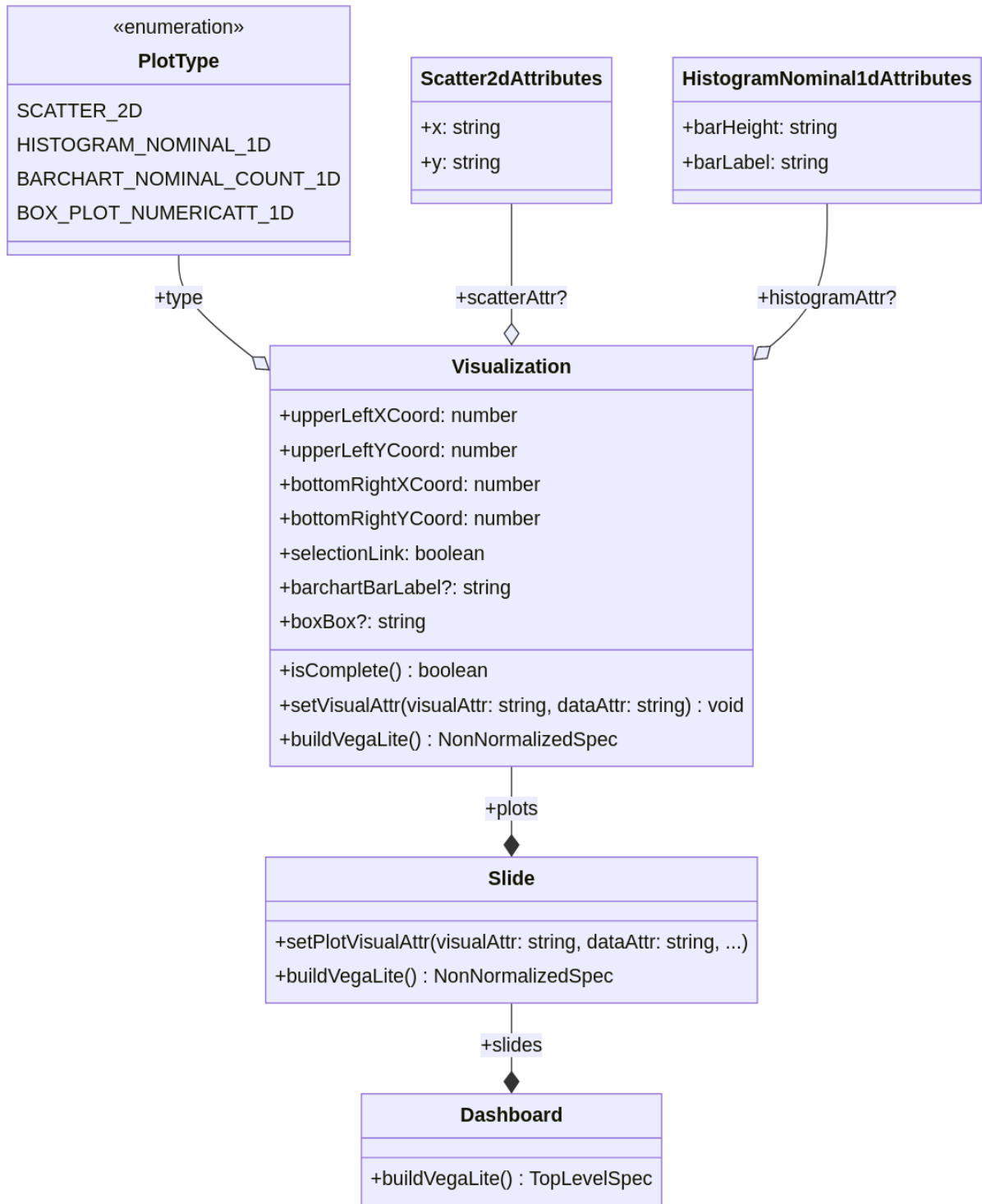


Figure 4.1 – Le diagramme de classe représentant un tableau de bord

5

Mise en œuvre

1 Outils et librairies utilisés

Durant le développement, j'ai utilisé la librairie Vega-Lite pour le dessin du tableau de bord et la librairie D3.js pour les interactions avec le **DOM**. J'ai développé le projet en **Typescript**, un langage compatible avec le Javascript qui y ajoute des annotations de type, ce qui rend le code plus clair et maintenable. Pour convertir mes sources ainsi que toutes les dépendances en un unique fichier qui est facile à inclure dans la page HTML, j'ai utilisé l'outil *webpack*, destiné à cette tâche.


2 Analyse des résultats, évaluation, qualité

Pour évaluer la qualité du code, j'ai réalisé quelques tests d'intégration qui vérifie que le site est fonctionnel. Ceux-ci sont décrits dans l'annexe *Cahier de test*.

3 Principales IHM

3.1 IHM 1 : Sélection des fichiers

Dashboard Viewer



The screenshot shows a light gray rounded rectangle containing two rows of controls. The first row is labeled 'Dashboard definition' and contains a 'Browse...' button followed by the text 'No file selected.'. The second row is labeled 'Dataset (CSV)' and contains a 'Browse...' button followed by the text 'No file selected.'. Below these two rows is a single 'Display' button.

Figure 5.1 – L’IHM finale de sélection des fichiers

Dashboard Viewer



The screenshot shows the same light gray rounded rectangle as in Figure 5.1, but now the text next to the 'Browse...' buttons is 'Example1Iris.dataset.csv'.

DashboardDefinitionError

Dashboard definition is incorrect.

Figure 5.2 – L’IHM finale de sélection des fichiers lorsqu’une erreur a eu lieu

3.2 IHM 2 : Affichage d'un tableau de bord

Dashboard Viewer

Close

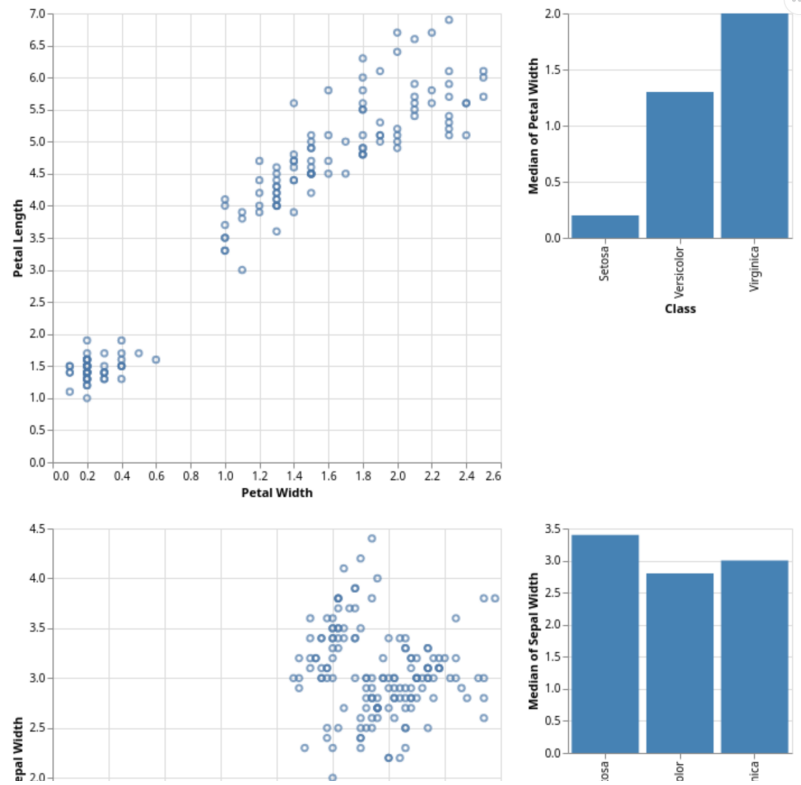


Figure 5.3 – L'IHM finale d'affichage d'un tableau de bord

6

Bilan et conclusion

1 Bilan du semestre 9

Durant le semestre 9, j'ai complété le cahier de spécification. Il me reste à faire le développement du projet.

2 Bilan du semestre 10

Parmi les fonctionnalités définies dans le cahier des charges, la lecture du fichier de définition de tableau de bord et de jeu de données et le dessin de visualisations ont été complétés, avec l'agencement des visualisations et la sélection de données partiellement complétés. La fonction d'interactivité, consistant du zoom et du défilement à l'intérieur d'une visualisation, a été abandonnée pendant la phase de développement.

Malgré que l'agencement, une fonctionnalité primordiale, n'ait pas été complétée, le projet reste fonctionnel : les visualisations sont simplement arrangées horizontalement. Il n'a pas été possible d'implémenter l'agencement tel que prévu car la définition d'un tableau de bord contient les coordonnées de chaque visualisation, or Vega-Lite ne permet pas de simplement placer les visualisations à des coordonnées précises. Des solutions possibles dans le futur sont :

- de modifier le format de la définition de tableau de bord pour donner l'emplacement de chaque visualisation dans une structure d'arbre, ce qui correspondrait mieux aux fonctions d'agencement présentes dans Vega-Lite ;
- d'étudier si la librairie Vega, une version plus avancée et complexe de Vega-Lite, permet de placer les visualisations à des coordonnées précises ;
- de calculer l'emplacement de chaque visualisation dans un arbre selon ses coordonnées. J'ai créé un algorithme simple qui réalise cette solution, cependant celui-ci ne fonctionne que sur les tableaux de bord contenant une structure d'arbre de deux niveaux d'imbrication où le premier niveau est horizontal et le second est vertical. Il est présent dans la branche *smart_layout* du dépôt Git.

Quant à la sélection de données, celle-ci est uniquement implémentée lors de la présence d'un histogramme faisant la médiane d'un attribut et d'un nuage de points sur le même tableau de bord : dans ce cas, l'utilisateur peut cliquer sur une barre de l'histogramme pour la sélectionner, faisant ainsi n'apparaître que les points présents dans cette barre sur le nuage de points. Afin de

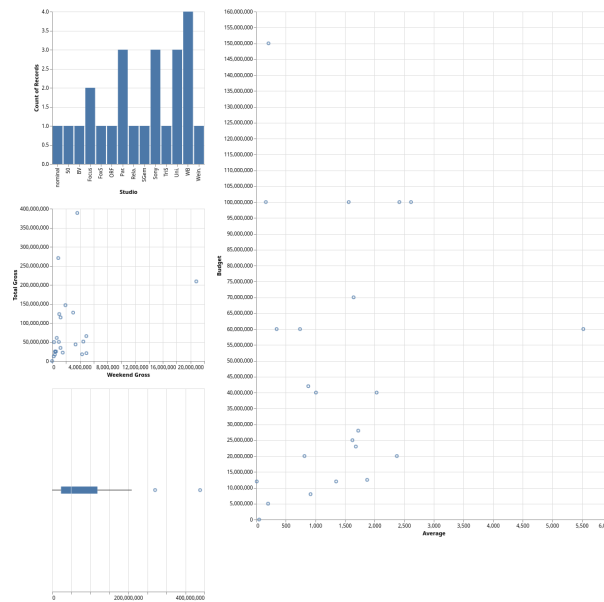


Figure 6.1 – Tableau de bord contenant un arbre à deux niveaux d’imbrication

compléter cette fonctionnalité, il faudrait donner à chaque visualisation qui peut être sélectionnée un identifiant unique qui serait utilisé dans la spécification Vega-Lite du tableau de bord à la fois pour indiquer que les données de cette visualisation peuvent être sélectionnées et que les données des autres visualisations doivent être filtrées selon ce qui a été sélectionné. Cela peut être réalisé en modifiant le format de définition du tableau de bord pour décrire plus précisément les conditions de sélection, ou en générant à la volée les identifiants uniques pour les visualisations pour lesquels la sélection est activée.

Une autre amélioration possible est l’ajout du dessin d’autres types de visualisations : à ce jour, seuls quatre types de visualisations sont implémentés (nuage de points, histogramme de la médiane d’un attribut, histogramme du compte des données et boîte à moustaches). Ceci est le cas car, plutôt que d’implémenter plus de visualisations, nous avons jugé plus judicieux de travailler sur les autres fonctions.

3 Bilan sur la qualité

Tout au long du développement, j’ai utilisé l’outil de gestion de versions Git afin de garder une trace de l’évolution du code source. Le gestionnaire de configuration `npm` permet de compiler les sources du projet en deux commandes sur un ordinateur ayant NodeJS, `npm` et `Typescript`, celles-ci étant documentées dans le document d’installation et le fichier "README.md" dans le dépôt Git.

La grande majorité des classes et fonctions sont documentées dans les commentaires au format TSDoc, et cette documentation peut-être convertie en HTML avec le paquet `TypeDoc`. Et pour le peu de code qui n’est pas documenté, les annotations de types de Typescript ont été utilisées, clarifiant sa signification. De plus, les étapes de l’ajout d’un type de visualisation, qui est une mise à jour très probable si le projet est repris, sont documentées, et plusieurs de ces étapes causeront une erreur à l’exécution si jamais elles ont été oubliées.

En prenant en compte ces éléments, je considère que le projet pourra être réutilisé et maintenu assez facilement, même s’il aurait été optimal que tout le code soit documenté.

4 Bilan auto-critique

Durant ce projet, j'aurai pu mieux g  r   mon temps. Lors des r  unions avec mon tuteur, je ne r  alisais pas de compte-rendu formels, alors que ceci aurait s  rement emp  ch   certaines incompr  hensions. Nous aurions peut-  tre pu avancer sur certains points comme la gestion de l'agencement des tableau de bords avant la fin du projet.

Annexes

A

Planification, gestion de projet

1 Evolution du projet

Le diagramme de Gantt initial pour la planification de ce projet.

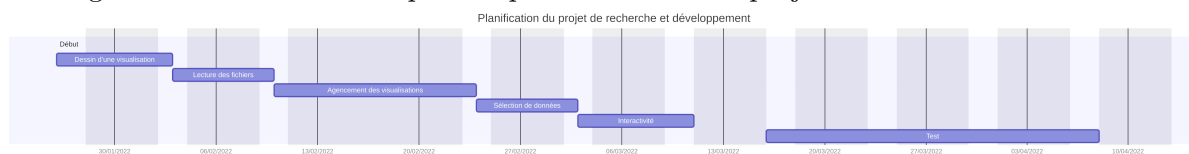


Figure A.1 – Le diagramme de Gantt initial

Le diagramme de Gantt final.



Figure A.2 – Le diagramme de Gantt final

2 Description des tâches

Tâche 1 : Dessin d'une visualisation

- Date de début : 26/01/2022
- Date de fin : 02/02/2022
- Durée : 3 jours
- Description : Créer les fonctions de dessin de visualisation à partir de la liste des types de visualisation.

Tâche 2 : Lecture des fichiers

- Date de début : 03/02/2022
- Date de fin : 09/02/2022
- Durée : 2 jours
- Description : Créer les fonctions de lecture des fichiers de description du tableau de bord et des jeux de données.

Tâche 3 : Agencement des visualisations

- Date de début : 10/02/2022
- Date de fin : 23/02/2022
- Durée : 2 jours
- Description : Créer les fonctions d'agencement des visualisations.

Tâche 4 : Sélection des données

- Date de début : 24/02/2022
- Date de fin : 02/03/2022
- Durée : 2 jours
- Description : Créer les fonctions de sélection des données sur une visualisation.

Tâche 5 : Interactivité

- Date de début : 03/03/2022
- Date de fin : 09/03/2022
- Durée : 2 jours
- Description : Créer les fonctions d'animation des visualisations lorsqu'une sélection a lieu.

Tâche 6 : Test

- Date de début : 10/03/2022
- Date de fin : 07/04/2022
- Durée : 8 jours
- Description : Tester le produit final.

B

Description des interfaces

1 Interfaces homme/machine

L'application est composée de deux interfaces : la première permet à l'utilisateur de sélectionner un fichier de définition du tableau de bord ainsi qu'un fichier de jeu de données et de lancer l'affichage du tableau de bord, et la seconde entoure le tableau de bord affiché et permet à l'utilisateur de télécharger celui-ci.

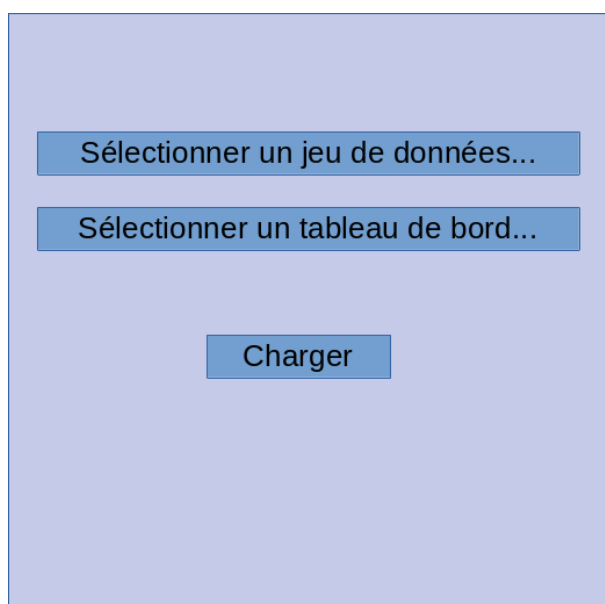


Figure B.1 – *Interface de chargement d'un tableau de bord*

2 Interfaces logiciel/logiciel

L'application de génération de tableau de bord donne son résultat sous la forme d'un fichier décrivant l'interface de ce tableau de bord au format CSV. Il doit définir des éléments d'interface (sous la forme de différentes visualisations) avec leur emplacements les colonnes utilisées, etc. Celui-ci peut ensuite être importé par l'application de visualisation de tableau de bord.

C

Cahier de Spécifications

1 Spécifications Fonctionnelles

1.1 Définition de la fonction : affichage d'un tableau de bord

Cette fonction a pour rôle d'afficher un tableau de bord. Elle est primordiale et englobe le projet entier.

Description

Afficher un tableau de bord

Entrée : Un fichier de définition de tableau de bord, un fichier de jeu de données

Sortie : Un tableau de bord interactif sous la forme d'une page Web

Préconditions :

- Un tableau de bord n'est pas déjà affiché.
- Le fichier de définition de tableau de bord et le fichier de jeu de données sont tous les deux valides.
- La définition du tableau de bord se réfère exclusivement aux données présentes dans le jeu de données.

Traitement

1. Lire le fichier de description de tableau de bord avec la fonction « lecture du fichier de définition de tableau de bord » ;
2. Lire le fichier de jeu de données avec la fonction « lecture du fichier de jeu de données » ;
3. Dessiner chacune des visualisations décrites dans la description du tableau de bord avec la fonction « dessin d'une visualisation » ;
4. Agencer les visualisations selon la description du tableau de bord avec la fonction « agencement des visualisations » ;
5. Afficher le tableau de bord, rendu interactif par les fonctions « sélection de données » et « interactivité ».

1.2 Définition de la fonction : lecture du fichier de définition de tableau de bord

Cette fonction a pour rôle de lire un fichier de description de tableau de bord et de le transformer en une structure de données utilisable pour les fonctions de dessin et d'agencement des visualisations. Elle est primordiale.

Description**Lire un fichier de définition de tableau de bord**

Entrée : Un fichier de définition de tableau de bord

Sortie : Une structure de données correspondante au contenu de ce fichier

Précondition : Le fichier de définition de tableau de bord a une syntaxe valide.

1.3 Définition de la fonction : lecture du fichier de jeu de données

Le rôle de cette fonction est de lire un fichier de jeu de données et de le transformer en une structure de données utilisable pour les fonctions de dessin des visualisations. Elle est primordiale.

Description**Lire un fichier de jeu de données**

Entrée : Un fichier de jeu de données

Sortie : Une structure de données correspondante au contenu de ce fichier

Précondition : Le fichier de jeu de données a une syntaxe valide.

1.4 Définition de la fonction : dessin d'une visualisation

Le rôle de cette fonction est de dessiner une visualisation à partir de la description de celle-ci et du jeu de données. Elle est primordiale.

Description**Dessiner une visualisation**

Entrée : La définition d'une visualisation, un jeu de données

Sortie : Le dessin de la visualisation décrite

Précondition : La description de la visualisation se réfère uniquement aux données présentes dans le jeu de données.

Traitement

1. Déterminer le type de visualisation à dessiner (tableau, diagramme circulaire, histogramme, nuage de points, etc.) ;
2. Déterminer les données à utiliser dans la visualisation ;
3. Dessiner la visualisation.

1.5 Définition de la fonction : agencement des visualisations

Cette fonction a pour rôle de placer les différentes visualisations dans le tableau de bord selon la description de celui-ci. Sa priorité est primordiale.

Description**Agencer les visualisations**

Entrée : Les visualisations dessinées pour ce tableau de bord, la définition de celui-ci

Sortie : L'interface contenant toutes les visualisations agencées comme demandé

Précondition : La définition du tableau de bord utilise uniquement les visualisations données.

Traitement

1. Pour chaque visualisation, déterminer son emplacement et la dessiner au bon endroit ;

1.6 Définition de la fonction : sélection de données

Cette fonction a pour rôle de permettre à l'utilisateur de sélectionner des données à mettre en valeur dans toutes les visualisations. Sa priorité est secondaire.

Description**Gérer la sélection de données**

Entrée : Le tableau de bord dessiné, le jeu de données

Sortie : Aucune, la fonction ne se termine pas

Précondition : Le jeu de données correspond au tableau de bord dessiné.

Traitement

1. Pour chaque visualisation, déterminer son emplacement et la dessiner au bon endroit ;
2. Attendre que l'utilisateur sélectionne (ou désélectionne) des données sur une visualisation ;
3. Redessiner toutes les autres visualisations pour ne montrer que les données sélectionnées ;
4. Retourner à l'étape 1.

Exemples de sélection

- diagramme circulaire : l'utilisateur clique sur l'une des parties et les données correspondantes à cette partie sont sélectionnées ;
- histogramme : l'utilisateur clique sur l'une des barres et les données correspondantes à cette barre sont sélectionnées ;
- tableau de données : l'utilisateur clique et glisse la souris puis relâche pour sélectionner une ou plusieurs lignes. Les données correspondantes à ces lignes sont alors sélectionnées ;
- nuage de points : l'utilisateur clique et glisse la souris puis relâche pour dessiner un rectangle qui contient tous les points à sélectionner. Les données correspondantes à ces points sont alors sélectionnées.

Une erreur possible est que l'utilisateur fasse une sélection vide. Dans ce cas, il faut l'ignorer.

1.7 Définition de la fonction : interactivité

Le rôle de cette fonction est de permettre à l'utilisateur de zoomer et faire défiler le contenu d'une visualisation. Sa priorité est facultative.

Description**Gérer les interactions**

Entrée : Le tableau de bord dessiné

Sortie : Aucune, la fonction ne se termine pas

Traitement

1. Attendre que l'utilisateur fasse une interaction de zoom ou de défilement sur une visualisation ;
2. Redessiner la visualisation pour prendre en compte ce zoom ou défilement ;
3. Retourner à l'étape 1.

Exemple d'interaction

Dans le cas d'un graphique en aires, l'utilisateur peut faire défiler sa molette de souris tout en survolant le graphique pour zoomer et dézoomer. Pour faire défiler les données, il peut faire glisser un rectangle sur une plus petite version du graphique dessiné en dessous.

2 Spécifications non fonctionnelles

2.1 Contraintes de développement et conception

Il s'agit d'une application web qui doit donc tourner dans le navigateur de l'utilisateur. Elle sera programmée en langage JavaScript avec la librairie Vega-Lite pour dessiner les diagrammes et les placer dans la page web.

2.2 Contraintes de fonctionnement et d'exploitation

2.2.1 Performances

L'application doit être réactive pendant l'utilisation du tableau de bord.

2.2.2 Capacités

Les tableaux de bord à dessiner ne devraient contenir que jusqu'à dix visualisations simultanées et les jeux de données ne doivent contenir que 2 000 données au maximum.

2.2.3 Sécurité

L'application est intégralement exécutée du côté du client et n'envoie jamais les données sur un serveur.

D

Cahier du développeur

1 Introduction

J'ai commencé le développement en créant un dépôt sur GitLab que mon tuteur pouvait visionner. Ensuite, j'ai créé un projet NPM. J'ai décidé d'utiliser le langage **Typescript**, transpilable en Javascript, pour sa clarté accrue comparé à celui-ci, ainsi que l'outil "webpack" qui permet de compiler les sources Javascript ainsi que les dépendances en un seul fichier à inclure dans la page HTML.

Comme prévu, j'ai utilisé la librairie Vega-Lite pour générer des spécifications de dessin de visualisations à partir des définitions de tableau de bord, et la librairie D3.js pour interagir plus facilement avec la page HTML.

Au cours du développement, j'ai fait des réunions hebdomadaires avec M. Venturini pour suivre la progression et réfléchir à des modifications possibles.

2 Diagrammes architecturaux et UML

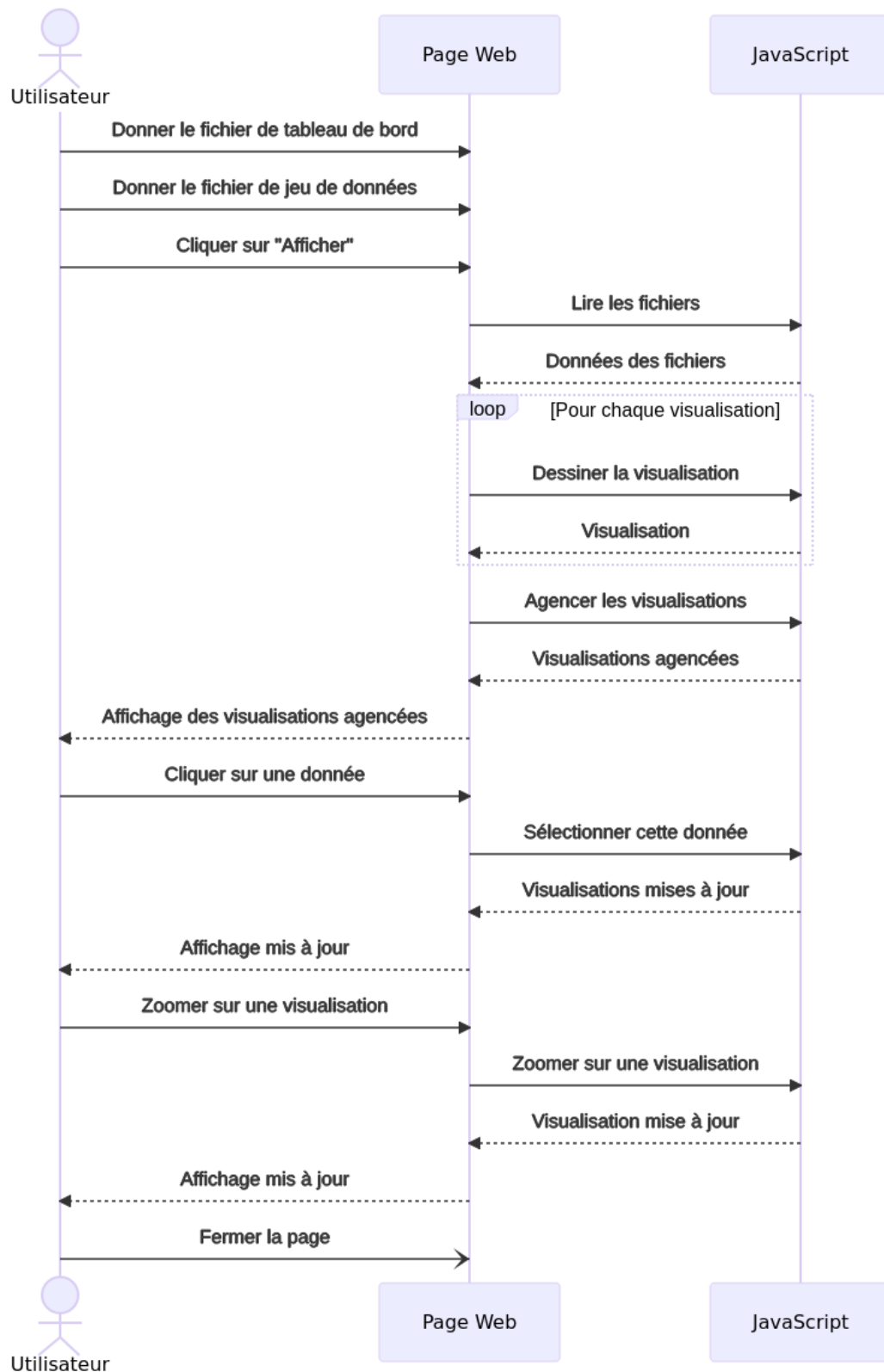


Figure D.1 – Diagramme de séquence du système

J'ai réalisé un diagramme de séquence du système du point de vue d'un utilisateur durant la phase d'analyse. Durant le développement, nous avons décidé que le zoom sur une visualisation n'était plus nécessaire.

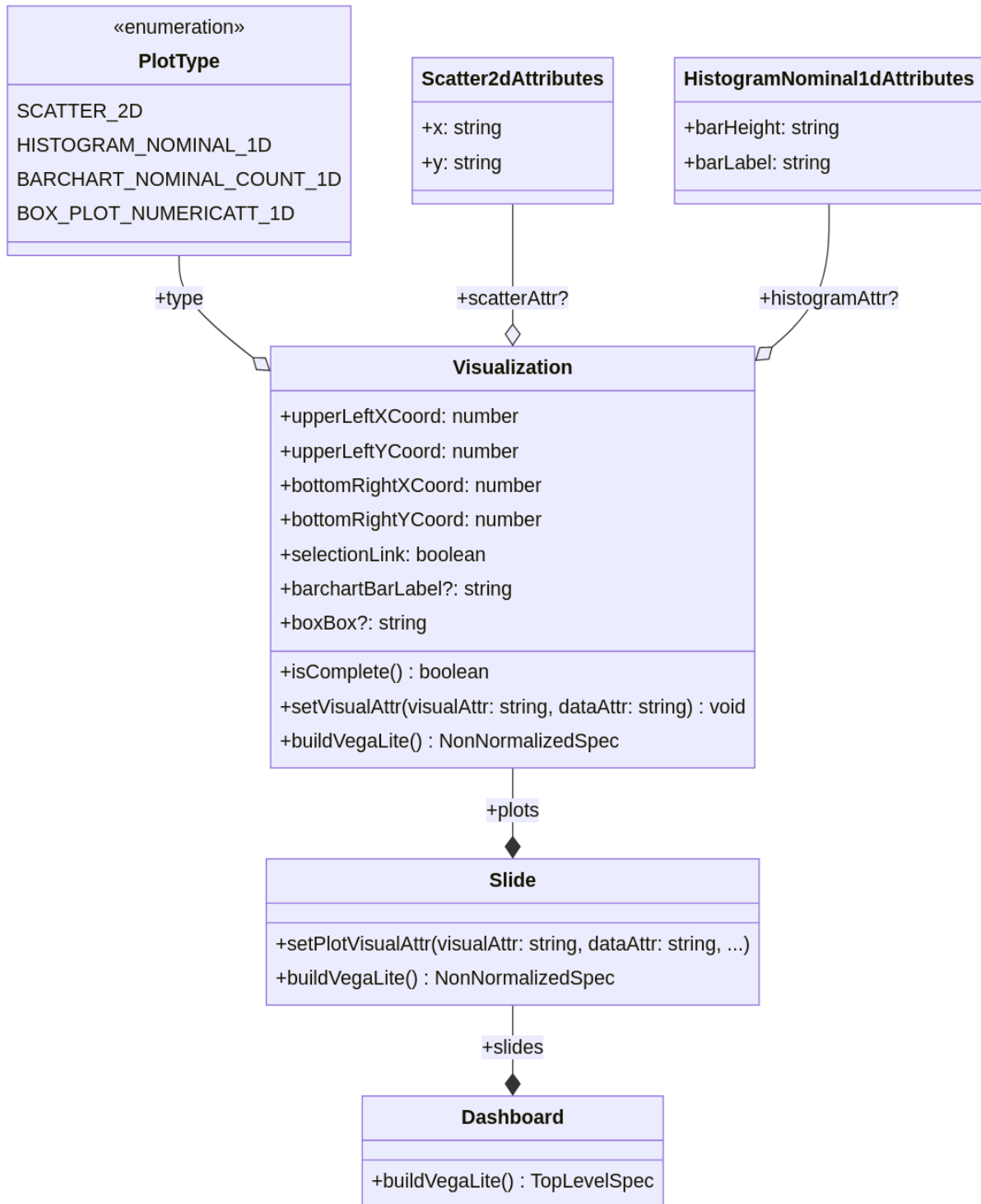


Figure D.2 – Diagramme de classe de la représentation d'un tableau de bord

À la fin du développement, j'ai créé ce diagramme de classe pour expliquer la représentation interne du tableau de bord dans mon programme.

3 Descriptions détaillées de données exploitées

Deux types de fichiers sont utilisés : les fichiers de jeux de données, simplement au format avec des noms de colonnes quelconques, et les fichiers de définition de tableaux, eux aussi au format

CSV mais avec des noms de colonne précis à respecter. Les valeurs doivent être séparées par des virgules pour les deux fichiers.

Le jeu de données est créé en dehors du cadre du projet. La définition du tableau de bord est généré par l'application du doctorant à partir du jeu de données.

3.1 Exemple : tableau de bord du jeu de données Iris

Sepal Length	Sepal Width	Petal Length	Petal Width	Class
5,1	3,5	1,4	0,2	Setosa
4,9	3	1,4	0,2	Setosa
4,7	3,2	1,3	0,2	Setosa
7	3,2	4,7	1,4	Versicolor
6,4	3,2	4,5	1,5	Versicolor
6,9	3,1	4,9	1,5	Versicolor
6,3	3,3	6	2,5	Virginica
5,8	2,7	5,1	1,9	Virginica
7,1	3	5,9	2,1	Virginica

Figure D.3 – Extrait du jeu de données Iris

Name of visualization	Name of visual attribute	Name of data attribute	Vis. Upper Left X1 coord	Vis. Upper Left Y1 coord	Vis. Bottom right X2 coord	Vis. bottom right Y2 coord	Selection link	Slide number	Other parameters
2DSCATTER	X	Petal Width	0	0	400	400	yes	1	none
2DSCATTER	Y	Petal Length	0	0	400	400	yes	1	none
1ATTRHISTOGRAMNOMINAL	BARHEIGHT	Petal Width	400	0	600	200	yes	1	none
1ATTRHISTOGRAMNOMINAL	BARLABEL	Class	400	0	600	200	yes	1	none
2DSCATTER	X	Sepal Length	0	0	400	400	yes	2	none
2DSCATTER	Y	Sepal Width	0	0	400	400	yes	2	none
1ATTRHISTOGRAMNOMINAL	BARHEIGHT	Sepal Width	400	0	600	200	yes	2	none
1ATTRHISTOGRAMNOMINAL	BARLABEL	Class	400	0	600	200	yes	2	none

Figure D.4 – Définition d'un tableau de bord pour le jeu de données Iris

La définition du tableau de bord contient les attributs suivants :

- *Name of visualization* : l'identifiant du type de visualisation à dessiner. "2DSCATTER" est un nuage de points et "1ATTRHISTOGRAMNOMINAL" est un histogramme de la médiane d'un attribut.
- *Name of visual attribute* : le nom d'un attribut visuel à dessiner dans le diagramme. Les valeurs possibles dépendent du type de visualisation : par exemple, "X" ou "Y" pour un nuage de points et "BARHEIGHT" ou "BARLABEL" pour un histogramme. Tous les attributs visuels d'une visualisation doivent être définis, donc certains types de visualisation sont définis sur plusieurs lignes.
- *Name of data attribute* : le nom de l'attribut du jeu de données à associer à l'attribut visuel donné dans la colonne précédente. Celui-ci doit être le nom d'une colonne du jeu de données, et l'attribut doit parfois être d'un certain type : par exemple, l'attribut visuel "X" doit toujours être quantitatif.
- *Vs. Upper Left X1 coord*, *Vs. Upper Left Y1 coord*, *Vs. Bottom right X2 coord* et *Vs. bottom right Y2 coord* sont les coordonnées que doivent prendre chaque visualisation lors de l'agencement.
- *Selection link* indique si les données de la visualisation peuvent être sélectionnées par l'utilisateur ("yes") ou non ("no").
- *Slide number* : le numéro de diapositive sur laquelle la visualisation doit être placée.
- *Other parameters* : d'autres paramètres potentiels des visualisations. Ceux-ci n'ont finalement pas été utilisés.

Ici, le tableau de bord suivant est défini tel que :

- Diapositive 1 :

- Nuage de points de la largeur des pétales selon la longueur des pétales ;
- Histogramme de la largeur médiane des pétales par catégorie de fleurs.
- Diapositive 2 :
 - Nuage de points de la longueur des sépales selon la largeur des sépales ;
 - Histogramme de la largeur médiane des sépales par catégorie de fleurs.

À partir de cette définition et du jeu de données, il est dessiné :

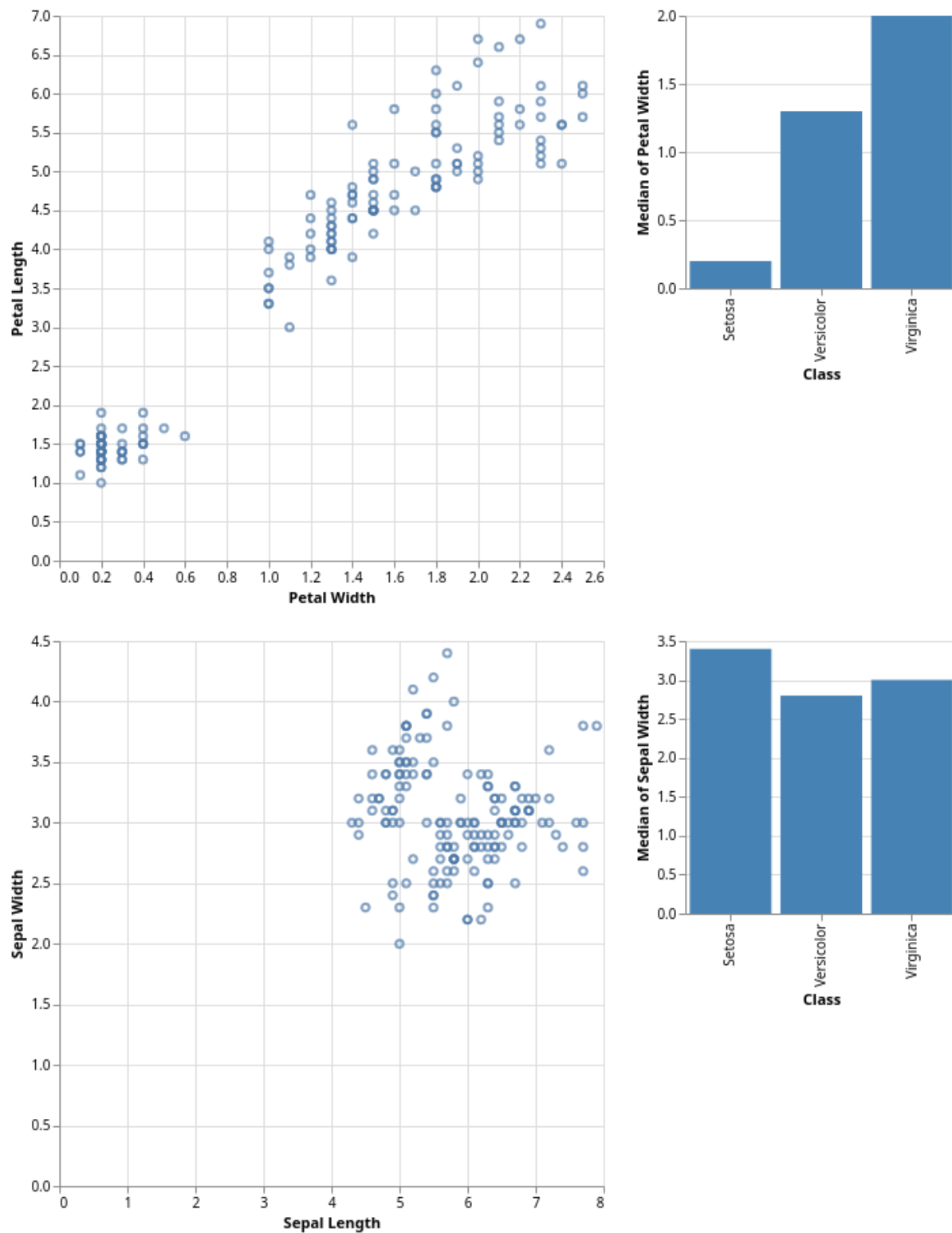


Figure D.5 – Tableau de bord dessiné

4 Descriptions détaillées des classes, modules, réalisations

Le code source est composé de plusieurs modules :

- *csv_reader* ;
- *visualisations* ;
- *dashboard* ;
- *index*.

4.1 Module *csv_reader*

Le module *csv_reader* définit les fonctions de lecture des fichiers **CSV**.

La fonction *readCsvArray* convertit un fichier CSV en une liste de liste des valeurs du tableau. Pendant la lecture, les lignes vides ou commentées par un caractère "%" sont ignorées.

La fonction *readCsvMap* convertit un fichier CSV en une liste de dictionnaires contenant les attributs, en utilisant la première ligne du tableau comme les noms des attributs. Si une valeur est un nombre réel, elle est convertie en variable de type *number*. Une erreur *CsvReadError* est signalée lorsque le nombre de colonnes diffère d'une ligne à la suivante.

Le type *Dataset*, retourné par la fonction précédente, est une liste de dictionnaires avec pour clés des chaînes de caractères et pour valeurs des chaînes de caractères ou nombres.

4.2 Module *dashboard*

Le module *dashboard* contient les classes représentant internement le tableau de bord : la classe *Dashboard* est construite à partir des données extraites du fichier de définition de tableau de bord et contient une liste de *Slide*, qui elles-mêmes contiennent les *Visualization*. Une fois le *Dashboard* construit, il est possible de construire la spécification Vega-Lite.

4.2.1 Classe *Dashboard*

La classe *Dashboard* est construite avec le *Dataset* retourné par *readCsvMap* et contient un unique attribut *slides* qui est la liste des diapositives.

Sa fonction *buildVegaLite* prend en argument le jeu de données associé au tableau de bord et retourne la spécification Vega-Lite de toutes ses diapositives concaténées verticalement. Elle peut signaler une erreur *InvalidDashboardError* si il s'avère que le tableau de bord contient des erreurs.

4.2.2 Classe *Slide*

La classe *Slide* contient un unique attribut *plots* qui est une liste de visualisations.

Elle possède une fonction *setPlotVisualAttr* qui prend en argument tous les attributs d'une ligne du fichier de définition de tableau de bord et, selon son propre contenu, ajoute une visualisation à sa liste ou définit un des attributs d'une visualisation existante.

Sa fonction *buildVegaLite* ne prend pas d'arguments et retourne la spécification Vega-Lite de ses visualisations, concaténées ensemble horizontalement.

4.2.3 Classe Visualization

La classe *Visualization* contient le type de la visualisation, ses coordonnées, si elle doit être utilisée pour les sélections, et les attributs visuels spécifiques à son type de visualisation. Elle contient plusieurs fonctions :

- *isComplete*, qui ne prend pas d'arguments et indique si tous les attributs visuels de son type ont été implémentés ;
- *setVisualAttr*, qui prend en argument le nom d'un attribut visuel et l'attribut du jeu de données à y associer ;
- *buildVegaLite*, qui construit la spécification Vega-Lite de cette visualisation.

Chacune de ces fonctions signalera une erreur *UnimplementedPlotTypeError* si jamais le type de la visualisation n'est pas géré dans le code, par soucis de facilitation d'ajout d'un type.

Les types de visualisations sont listés dans l'énumération *PlotType*. Il est possible d'ajouter un type de visualisation en suivant sept étapes documentées :

1. Ajouter le type à l'énumération ;
2. Ajouter à la classe *Visualization* les attributs nécessaires ;
3. Si nécessaire, initialiser ces attributs dans le constructeur de *Visualization* ;
4. Dans la fonction *setVisualAttr*, implémenter la définition de ces attributs ;
5. Dans la fonction *buildVegaLite*, implémenter le dessin de ce type de visualisation ;
6. Dans la fonction *isComplete*, vérifier que les attributs de ce type de visualisation sont définis ;
7. Dans la fonction *setPlotVisualAttr* de *Slide*, ajouter le nouveau type à la vérification du type de visualisation.

4.3 Module visualisations

Le module *visualisations* contient des fonctions pour aider à la construction de la spécification Vega-Lite :

- *initVegaLiteSpec* construit la spécification complète à partir d'un jeu de données et de la spécification de toutes les diapositives ;
- *buildScatter2D* construit la spécification d'un nuage de points ;
- *buildHistogramNominal1D* construit la spécification d'un histogramme à médiane d'un attribut ;
- *buildBarchartNominalCount1D* construit la spécification d'un histogramme à compte des membres d'une classe ;
- *buildBoxPlot1D* construit la spécification d'une boîte à moustaches.

4.4 Module index

Le module *index* contient les fonctions de construction du **DOM** et de gestion des évènements.

Les erreurs lancées par les fonctions des autres modules sont converties en erreurs *DashboardDefinitionError* ou *DatasetError*, selon si leur origine est le tableau de bord ou le jeu de données, puis affichées dans une section rouge en dessous de la section de sélection des fichiers.

Au chargement du script, la fonction *window.load* est redéfinie par une fonction qui attend le chargement de la page HTML et ajoute les différents éléments au **DOM** : les sections de sélection des fichiers, d'affichage du tableau de bord et des erreurs, ainsi que le bouton "Close". Initialement, seule la section de sélection des fichiers est affichée.

Ensuite, les fonctions de gestion des évènements sont définies :

- Au clic sur le bouton "Display", les fichiers sélectionnés sont chargés et le tableau de bord est généré puis affiché, tandis que la section de sélection est caché. S'il y a une erreur, le processus est annulé et celle-ci est affichée dans la section d'erreur qui devient visible.
- Au clic sur le bouton "Close", la section du tableau de bord est cachée et celle de sélection des fichiers est affichée à nouveau.

E

Document d'installation

Pour empaqueter le code source, il faut installer :

- la dernière version de l'outil npm en suivant le guide d'installation à cette adresse : <https://docs.npmjs.com/downloading-and-installing-node-js-and-npm/> ;
- la dernière version du transpileur Typescript en suivant le guide d'installation à cette adresse : <https://typescriptlang.org/download>.

Il faut ensuite installer les dépendances avec la commande `npm install` et empaqueter les sources avec la commande `npx webpack`.

Pour mettre en production le projet, il faut installer un serveur HTTP, tel que Apache HTTP Server ou Nginx, et le configurer pour servir le répertoire "html" du projet.

Une fois que le code source a été empaqueté, seul le répertoire "html" et le serveur sont nécessaires. Toutes les dépendances sont incluses dans ce répertoire et ni npm ni Typescript ne sont utilisés par le serveur.

F

Document d'utilisation

Lors de l'utilisation, il faut d'abord se fournir d'un jeu de données au format CSV et de la spécification de tableau de bord lui correspondant. Des exemples sont disponibles dans le dépôt Git final, et il est également possible de générer un tableau de bord avec l'application Web de génération.

Pour afficher le tableau de bord, il faut aller à l'adresse du site web et fournir le fichier de définition du tableau de bord dans le champ "Dashboard definition" et le jeu de données dans le champ "Dataset (CSV)". En cliquant sur le bouton "Display", le tableau de bord sera affiché.

En cliquant sur le bouton marqué d'une ellipse en haut à droite du tableau de bord, il est possible de l'exporter aux formats PNG ou SVG en cliquant sur "Save as PNG" ou "Save as SVG".

En cliquant sur le bouton "Close" en haut à gauche de la page, on peut fermer le tableau de bord et en charger un nouveau.

1 Erreurs

En cas d'invalidité d'un des fichiers d'entrée, un message d'erreur sera affiché.

Si un tableau de bord sans données s'affiche, cela signifie que le jeu de données fourni ne correspond pas à la définition du tableau de bord.



Cahier de test

L'exactitude du comportement du site Web produit a été vérifié avec des tests d'intégration, complétés à la main.

1 Tests d'intégration

IDENTIFICATION OF COMPONENT
Afficher un tableau de bord
DESCRIPTION OF THE TEST (granularity, scenario, values, actions)
Entrer un fichier de définition de tableau de bord et un fichier de jeu de données, tous les deux valides et se correspondant.
EXPECTED RESULTS
Le tableau de bord s'affiche tel que décrit et contenant les données.
OBTAINED RESULTS
Le tableau de bord s'affiche tel que décrit et contenant les données.

IDENTIFICATION OF COMPONENT
Afficher une erreur en cas d'incohérence entre les deux fichiers
DESCRIPTION OF THE TEST (granularity, scenario, values, actions)
Entrer un fichier de définition de tableau de bord et un fichier de jeu de données, tous les deux valides mais ne se correspondant pas.
EXPECTED RESULTS
Un message d'erreur spécifique et lisible par un utilisateur s'affiche.
OBTAINED RESULTS
Le tableau de bord s'affiche tel que décrit sans contenir de données.

IDENTIFICATION OF COMPONENT
Afficher une erreur en cas d'invalidité de la définition du tableau de bord
DESCRIPTION OF THE TEST (granularity, scenario, values, actions)
Entrer un fichier de définition de tableau de bord invalide et un fichier de jeu de données valide.
EXPECTED RESULTS
Un message d'erreur spécifique et lisible par un utilisateur s'affiche.
OBTAINED RESULTS
Un message d'erreur spécifique et lisible par un utilisateur s'affiche.

IDENTIFICATION OF COMPONENT
Afficher une erreur en cas d'invalidité du jeu de données
DESCRIPTION OF THE TEST (granularity, scenario, values, actions)
Entrer un fichier de définition de tableau de bord valide et un fichier de jeu de données invalide.
EXPECTED RESULTS
Un message d'erreur spécifique et lisible par un utilisateur s'affiche.
OBTAINED RESULTS
Un message d'erreur spécifique et lisible par un utilisateur s'affiche.

IDENTIFICATION OF COMPONENT
Retourner à la sélection de fichiers après ouverture d'un tableau de bord
DESCRIPTION OF THE TEST (granularity, scenario, values, actions)
Pendant qu'un tableau de bord est affiché, cliquer sur le bouton "Close" en haut à gauche.
EXPECTED RESULTS
Le tableau de bord est caché et la fenêtre de sélection est à nouveau affichée..
OBTAINED RESULTS
Le tableau de bord est caché et la fenêtre de sélection est à nouveau affichée..

H

Bibliographie

- [1] Arvind SATYANARAYAN, Dominik MORITZ, Kanit WONGSUPHASAWAT et Jeffrey HEER. « Vega-Lite : A Grammar of Interactive Graphics ». In : *IEEE Transactions on Visualization & Computer Graphics (Proc. InfoVis)* (2017). DOI : [10.1109/tvcg.2016.2599030](https://doi.org/10.1109/tvcg.2016.2599030). URL : <http://idl.cs.washington.edu/papers/vega-lite>.
- [2] Chris STOLTE, Diane TANG et Pat HANRAHAN. « Polaris : A System for Query, Analysis, and Visualization of Multidimensional Databases ». In : *Communications of the ACM* (2008). DOI : [10.1145/1400214.1400234](https://doi.org/10.1145/1400214.1400234). URL : <https://dl.acm.org/doi/10.1145/1400214.1400234>.

I

Glossaire

CSV Un format de fichier textuel contenant des données sous forme de tableau où chaque colonne correspond à un attribut et chaque ligne correspond à un élément. Les valeurs sont séparées par des virgules, d'où son nom, "Comma-separated values". [7](#), [28](#)

Typescript Langage basé sur le Javascript, ajoutant des annotations de type pour un code source plus clair. [9](#), [13](#), [23](#)

J

Acronymes

DOM Document Object Model. [6](#), [7](#), [9](#), [29](#)

Contexte

Il existe de nombreuses solutions de génération de tableaux de bord destinés aux entreprises, tel que Microsoft PowerBI et Tableau. Cependant, ce type de solution n'existe pas pour des utilisations de type scientifique, particulièrement lorsqu'il s'agit de données sensibles.

Objectifs

L'objectif est de visualiser des jeu de données scientifiques de la manière la plus optimale possible sous la forme d'un tableau de bord. L'utilisateur doit avoir la possibilité de configurer ses propres objectifs. Ce tableau de bord doit être interactif, avec la possibilité de sélectionner des données sur un graphe et voir cette sélection appliquée aux autres visualisations. Par soucis de confidentialité, les données ne doivent pas être traitées sur un serveur externe.

Mise en œuvre

Ce projet consiste à afficher un tableau de bord pré-généré et s'intègre à un projet de génération de tableau de bord.

1. Lecture de jeux de données
2. Dessin des visualisations
3. Agencement des visualisations
4. Sélection des données par l'utilisateur

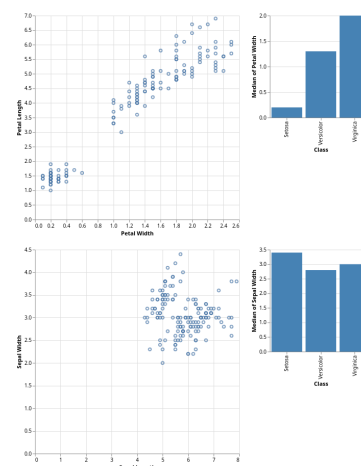
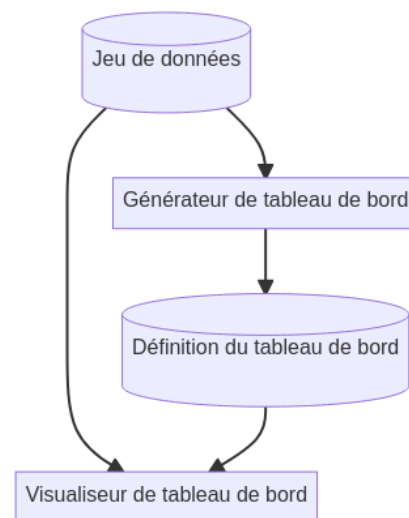
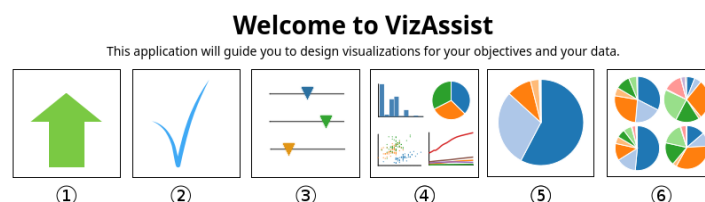


Tableau de bord affiché



Site de génération automatique de visualisations, la base de ce projet

Il existe de nombreuses solutions de génération de tableaux de bord destinées aux entreprises, tel que Microsoft PowerBI et Tableau. Cependant, ce type de solution n'existe pas pour des utilisations de type scientifique, particulièrement lorsqu'il s'agit de données sensibles.

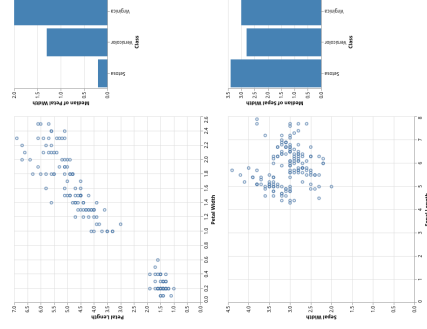
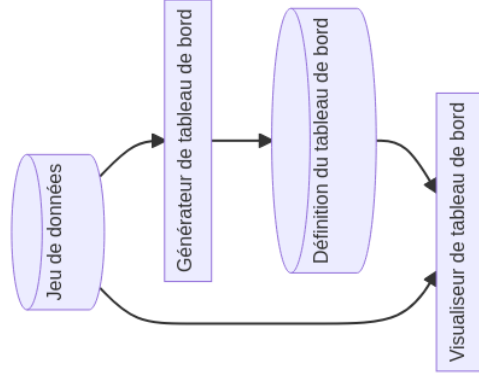
Objectifs

L'objectif est de visualiser des données scientifiques de la manière la plus optimale possible sous la forme d'un tableau de bord. L'utilisateur doit avoir la possibilité de configurer ses propres objectifs. Ce tableau de bord doit être interactif, avec la possibilité de sélectionner des données sur un graphe et voir cette sélection appliquée aux autres visualisations. Par soucis de confidentialité, les données ne doivent pas être traitées sur un serveur externe.

Mise en œuvre

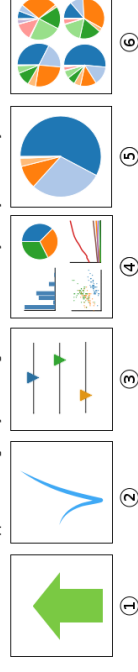
Ce projet consiste à afficher un tableau de bord pré-généré et s'intégrer à un projet de génération de tableau de bord.

1. Lecture de jeux de données
2. Dessin des visualisations
3. Agencement des visualisations
4. Sélection des données par l'utilisateur



Welcome to VizAssist

This application will guide you to design visualizations for your objectives and your data.



Site de génération automatique de visualisations, la base de ce projet

RFAI2

Génération automatique de tableau de bord

Résumé

Ce projet de recherche et développement consiste à créer une plateforme web d'affichage de tableaux de bord prégénérés, destinés à visualiser des jeux de données.

Mots-clés

Système d'information, Visualisation, Jeu de données, JavaScript, Vega-Lite

Abstract

This research and development project consists of creating a web platform for displaying pregenerated dashboards, with the goal of visualizing datasets.

Keywords

Information system, Visualization, Dataset, JavaScript, Vega-Lite