Haute Ecole de Gestion & Tourisme
Hochschule für Wirtschaft & Tourismus

Travail de Bachelor

Information and Communication Technologies (ICT)

Création de composants BI custom dans Power BI

Auteur:

Blendar Berisha

Professeur :

Prof. Cosette Bioley

Déposé à la HES-SO Valais-Wallis à Sierre, le 9 août 2025

Résumé

Les visuels standard de Power BI ne couvrant plus l'ensemble des besoins analytiques des clients d'ECRINS SA, l'entreprise souhaite internaliser la création de **visuels personnalisés** (*custom visuals*). Ce travail de Bachelor poursuit deux finalités : *(1)* élaborer un **cadre méthodologique complet** — de l'analyse fonctionnelle jusqu'au déploiement automatisé — pour développer ces composants, et *(2)* en démontrer la faisabilité au moyen d'un **prototype de visuel pilote**.

La démarche s'appuie sur une étude critique des visuels natifs, un benchmark d'autres plateformes BI et une collaboration étroite avec la responsable produit afin de cibler un besoin prioritaire. Le développement adopte un processus itératif inspiré des méthodes Agiles : configuration de l'environnement `pbiviz`, implémentation en TypeScript / D3, tests et intégration continue via GitHub Actions, puis documentation et bonnes pratiques de gouvernance.

L'évaluation combine des tests techniques, une revue de code et une validation fonctionnelle auprès du client interne. Les livrables — code source, pipeline CI/CD opérationnel et guide d'intégration — constituent une base réutilisable pour accélérer la livraison future de visuels sur mesure tout en renforçant la gouvernance BI d'ECRINS SA.

Mots clés : Business Intelligence ; Power BI ; visuels personnalisés ; CI/CD ; gouvernance

Remerciements

Avant d'entamer la lecture de ce travail, je souhaite exprimer ma profonde gratitude à toutes les personnes qui ont contribué, de près ou de loin, à son aboutissement.

À ma directrice de Bachelor, Prof. Cosette Bioley, pour sa rigueur scientifique, ses retours toujours constructifs et la confiance accordée dès les premières discussions. Son exigence académique a largement façonné la qualité de ce mémoire.

Aux enseignantes et enseignants de la filière ICT de la HES-SO Valais, qui m'ont transmis les fondements théoriques et pratiques indispensables à la réalisation de ce projet.

À mes camarades de promotion, pour l'entraide quotidienne, les revues de code improvisées et l'indispensable bonne humeur qui ont rythmé ces mois intenses.

À ma famille et à mes proches, pour leur soutien inconditionnel, leur patience face aux longues soirées de développement et leurs encouragements constants.

Que chacune et chacun trouve ici l'expression de ma reconnaissance la plus sincère.

Contents

List of Figures

List of Tables

1 Introduction

1.1 Contexte

Depuis plusieurs années, les outils de Business Intelligence (BI) sont devenus indispensables aux organisations qui souhaitent baser leurs décisions sur des données fiables et facilement interprétables. **Power BI** de Microsoft s'est imposé comme l'une des plateformes de référence, grâce à sa prise en main intuitive, sa connectivité riche et son intégration étroite à l'écosystème Microsoft (Microsoft, 2024). Malgré une bibliothèque déjà fournie de visuels standards, certains besoins métier demeurent inassouvis : diagrammes de Gantt avancés pour la gestion de projet, variances financières multi-niveaux, ou encore respect strict de chartes graphiques internes. **ECRINS SA**, société suisse spécialisée dans le conseil BI, fait régulièrement face à ces demandes « hors catalogue » et souhaite internaliser la création de *visuels personnalisés* afin de gagner en réactivité, en qualité et en gouvernance.

1.2 Problématique

La société n'a encore jamais mis en place de processus complet pour concevoir, tester et déployer des visuels Power BI sur mesure ; chaque tentative ponctuelle a fait ressortir plusieurs points critiques :

- **Incertitude technique** : absence de référentiel sur les performances, la scalabilité et la sécurité des composants personnalisés ;
- **Manque de normes et de traçabilité** : code développé au cas par cas, difficile à maintenir et à auditer ;
- **Planning imprévisible** : effort, coût et délai impossibles à estimer sans méthode éprouvée ;
- **Risque fonctionnel** : sans cadre clair, le livrable peut ne pas répondre pleinement aux attentes métier ou se révéler difficile à faire évoluer.

La question centrale se formule ainsi :

> **Comment concevoir et valider un cadre méthodologique complet permettant à ECRINS SA d'industrialiser, de manière fiable et pérenne, la création et la mise en production de visuels Power BI personnalisés ?**

1.3 Objectifs

O1 **Élaborer un cadre méthodologique détaillé**, couvrant analyse du besoin, développement, tests, gouvernance et mise en production ;

O2 **Réaliser un visuel pilote**, sélectionné avec la Product Owner, pour démontrer la faisabilité du cadre proposé ;

O3 **Évaluer la solution**, à l'aide de critères techniques (maintenabilité, performances, sécurité) et fonctionnels (adéquation métier).

1.4 Portée et limites

Inclus :

- développement d'un visuel pilote représentatif ;
- environnements Power BI Desktop et Service internes ;
- mise en place d'une chaîne CI/CD (GitHub Actions).

Exclus :

- publication sur Microsoft AppSource ;
- support mobile/tablette étendu ;
- constitution d'une bibliothèque exhaustive de visuels.

1.5 Structure du rapport

1. **Chapitre 2 – Cadre théorique et état de l'art** : principes des visuels Power BI, limites des visuels Python/R, justification du SDK ;
2. **Chapitre 3 – Analyse interne** : audit, benchmark, recueil des besoins et sélection du visuel pilote ;
3. **Chapitre 4 – Méthodologie de projet** : organisation Agile (Scrum-but), artefacts et planification ;
4. **Chapitre 5 – Conception et développement** : architecture, implémentation (TypeScript + D3) et tests ;
5. **Chapitre 6 – Industrialisation** : pipeline CI/CD, signature numérique et normes internes ;

6. **Chapitre 7 – Évaluation** : résultats techniques et validation fonctionnelle ;
7. **Chapitre 8 – Conclusion et perspectives**.

Cette structure garantit la cohérence et la traçabilité de l'ensemble du travail, des besoins initiaux jusqu'aux recommandations finales.

2 État de l'art

2.1 Architecture des visuels Power BI

Cette section présente l'architecture technique d'un visuel personnalisé (custom visual) dans Power BI. Comprendre ces fondations est essentiel pour aborder correctement le développement, l'intégration et la mise en production de ces composants.

2.1.1 Isolation et exécution dans un environnement sandboxé

Power BI charge chaque visuel dans une `iframe` sandboxée, isolée du reste du rapport. Ce conteneur est orchestré par le « Power BI host », qui injecte le SDK JavaScript, le bundle du visuel (`.js`, `.css`) et transmet les données et options de rendu via une interface standardisée.

Ce mécanisme garantit :

- une isolation stricte, évitant les conflits de dépendances et les fuites mémoire (CORPORATION, 2016) ;
- une compatibilité multi-visuels au sein d'un même rapport ;
- un canal de communication contrôlé (filtres croisés, sélection, thème).

Depuis la version 4.6 de l'API Power BI, le fichier `capabilities.json` doit déclarer explicitement les privilèges d'accès, comme les connexions réseau ou fichiers externes (DOCS, 2024). La branche 5.x, publiée en mai 2024, étend ce modèle avec la prise en charge des Web Workers pour les traitements asynchrones (TEAM, 2025).

2.1.2 Structure d'un projet `pbiviz`

L'outil en ligne de commande `pbiviz` permet de générer la structure d'un nouveau composant via :

```
pbiviz new <nom_du_visual>
```

Il en résulte l'arborescence suivante :

- `pbiviz.json` : métadonnées globales (nom, version, `apiVersion`) ;
- `capabilities.json` : déclaration des rôles de données et des options d'interactivité ;
- `src/visual.ts` : classe principale du visuel (implémente `IVisual`) ;
- `assets/` : ressources telles que les icônes ou les fichiers de traduction ;

- fichiers de build : `package.json`, `webpack.config.js`, etc.

L'ensemble de ces fichiers constitue le cycle de vie d'un visual Power BI, depuis sa construction jusqu'à son exécution dans le client.

2.1.3 Cycle de vie du visuel : interface `IVisual`

Le SDK Power BI impose l'implémentation d'une interface nommée `IVisual`. Celle-ci repose sur quatre méthodes principales (MICROSOFT, 2023) :

`constructor(options)` : Initialise le visuel, reçoit le conteneur DOM et les paramètres de configuration.

`update(options)` : Méthode clé appelée à chaque modification du rapport (données, filtre, taille, thème).

`enumerateObjectInstances()` (optionnelle) : Permet de générer dynamiquement les options du volet « Format ».

`destroy()` (optionnelle) : Nettoie les ressources (listeners, canvas, workers) lors du déchargement du visuel.

L'objet `VisualUpdateOptions` contient notamment :

- la `dataView` (structure tabulaire de données) ;
- les dimensions du conteneur ;
- les options de thème et d'interactivité.

2.1.4 Gestion du rendu et des interactions

Le moteur Power BI invoque `update()` selon un modèle différentiel (*diff rendering*) : seul le rendu des éléments affectés par une modification est nécessaire. Cette approche nécessite un code idempotent et optimisé.

Les interactions croisées (filtres, surbrillance) sont gérées via un `Selection Manager` fourni par la bibliothèque `powerbi-visuals-utils-interactivityutils`. Un cycle d'interaction typique est le suivant :

1. L'utilisateur sélectionne un élément dans un autre visuel (ex. une barre d'un histogramme).
2. Le host applique le filtre et met à jour la `dataView` du visuel custom.

3. Le visuel adapte son affichage (ex. opacité réduite des éléments non sélectionnés).

2.1.5 Versions de l'API et implications sécuritaires

Le champ `apiVersion` de `pbiviz.json` détermine l'ensemble des fonctionnalités disponibles. Les versions 5.x, introduites en 2024, apportent plusieurs nouveautés majeures (TEAM, 2025) :

- chargement dynamique via `import('@powerbi/visuals-api')` ;
- support natif des Web Workers pour l'exécution parallèle ;
- support du `advancedEditModeSupport` (niveau 2) pour les options conditionnelles.

Le moteur sandbox interdit tout accès direct au DOM parent, au stockage local ou aux appels `fetch()` non autorisés, sauf si explicitement déclarés dans la clé `privileges`.

Résumé opérationnel

Cette architecture impose une rigueur dès les premières étapes de développement :

- tests ciblés par méthode : `constructor` vs `update` ;
- stratégie de rendu performante (DOM partiel, SVG, Canvas ou virtual DOM) ;
- déclaration précise des rôles de données dans `capabilities.json` ;
- compatibilité stricte avec les versions `apiVersion` dans le pipeline CI/CD.

Les prochaines sections examineront les autres approches disponibles (Python, R), pour justifier ensuite le choix du SDK TypeScript comme solution prioritaire.

2.2 Visuels natifs : capacités et limites

Avant d'envisager la création de visuels personnalisés via Python, R ou le SDK TypeScript, il convient de dresser un panorama complet des visuels dits *natifs* disponibles dans Power BI. Ceux-ci sont préinstallés et activement maintenus par Microsoft, couvrant la majorité des besoins courants en analyse et restitution de données. Cette section propose non seulement une typologie complète, mais aussi une lecture critique de leur rôle et de leurs limites dans un contexte professionnel.

2.2.1 Typologie fonctionnelle des visuels standards

2.2.1.1 Graphiques de comparaison

Barres et colonnes (groupées, empilées, 100 %) permettent de visualiser des agrégations comparées entre catégories. Elles acceptent le tri dynamique et le forage hiérarchique (drill-down) jusqu'au niveau de granularité le plus fin. Leur usage est optimal pour des tableaux de bord orientés suivi de performance, mais reste limité par l'absence de sous-totaux intermédiaires et la rigidité des axes (un seul axe secondaire).

2.2.1.2 Graphiques de tendance

Les courbes et aires (simples, empilées, 100 %) modélisent des séries temporelles ou continues. Idéales pour l'analyse temporelle de KPI, les courbes supportent l'affichage d'un axe secondaire, et les aires mettent en valeur les volumes cumulés. Leur capacité de drill-down et de filtrage croisé les rend adaptées à des rapports interactifs.

2.2.1.3 Graphiques combinés

Les visuels de type "combo" (barres + lignes) permettent de superposer par exemple un chiffre d'affaires mensuel et une marge en pourcentage. Ils sont recommandés lorsque deux échelles de mesure doivent coexister dans un même visuel, sans multiplier les graphiques.

2.2.1.4 Graphique de ruban

Spécifique à la représentation du *rang*, le graphique de ruban est adapté à l'analyse de parts de marché dynamiques. Il est utile pour suivre les changements de position relative de catégories dans le temps (produits, régions, marques).

2.2.1.5 Graphiques de processus

Cascade (waterfall) permet de visualiser des variations successives contribuant à un total, tandis que *entonnoir* (funnel) illustre les déperditions d'un processus (par exemple ventes ou parcours client). Le waterfall accepte les sous-totaux intermédiaires, contrairement à l'entonnoir.

2.2.1.6 Nuages de points et bulles

Le scatter plot est utile pour explorer les corrélations entre deux variables quantitatives. L'ajout d'une dimension via la taille de la bulle permet des lectures croisées. Ces visuels sont pertinents pour des analyses exploratoires mais se heurtent à une limite de volume : au-delà de 30 000 points, l'agrégation est imposée.

2.2.1.7 Graphiques circulaires

Secteurs (pie) et anneaux (donut) représentent les parts d'un total. Bien que fréquemment utilisés, ils sont à réserver à un nombre réduit de segments (idéalement < 6) sous peine de perte de lisibilité. Leur usage reste plus esthétique que fonctionnel.

2.2.1.8 Treemaps et arborescences

Les treemaps permettent d'encapsuler plusieurs niveaux hiérarchiques dans une surface limitée. Leur lecture est efficace pour les structures catégorielles denses (portefeuilles produits, répartitions géographiques). Un clic fore dans la hiérarchie, avec mise en forme conditionnelle possible.

2.2.1.9 Cartographie

- **Carte Bing** : affiche des bulles proportionnelles sur fond routier ; recommandée pour des localisations simples.
- **Carte remplie (choroplèthe)** : colore les régions selon une valeur agrégée ; utile pour les données administratives.
- **Azure Maps** : propose une cartographie vectorielle moderne avec gestion de clusters et carte thermique.
- **ArcGIS for Power BI** : permet des fonctions spatiales avancées (zones isochrones, couches socio-démographiques), avec certaines restrictions sans compte Esri.

2.2.1.10 Cartes de KPI et jauges

- **Carte (single)** : affiche une métrique unique, très lisible sur petits écrans.
- **Carte multi-lignes** : combine plusieurs KPIs dans un seul composant.
- **KPI** : présente une valeur, une cible et une tendance ; bien adapté au suivi d'objectifs.
- **Jauge** : visualisation qualitative d'une performance vs objectif ; limitée à un usage monométrique.

2.2.1.11 Tableaux et matrices

Les tableaux affichent les données ligne à ligne ; les matrices pivotent les dimensions et permettent les totaux intermédiaires. Pour des rapports très formatés, Power BI propose *le rapport paginé* (RDL), utile dans un contexte administratif.

2.2.1.12 Filtres interactifs

Segments (slicers) : composants filtrants par catégorie, plage ou date. Le segment bouton améliore l'ergonomie sur mobile et permet une navigation par clic visuel.

2.2.1.13 Visuels d'intelligence artificielle

- **Influenceurs clés** : détecte les dimensions explicatives d'une mesure cible.
- **Arborescence de décomposition** : propose des détails successifs de manière semi-automatisée.
- **Q&A** : traduit une requête en langage naturel en visuel dynamique.
- **Narratif intelligent** : génère du texte automatisé en fonction des filtres actifs.

2.2.1.14 Intégrations actionnables

- **Power Apps visual** : intègre une app canvas interactive pour actions en base (formulaires, validation).
- **Power Automate visual** : ajoute un bouton qui lance un processus automatisé (email, mise à jour CRM).

Analyse critique

Points forts

- Performance optimale : chargement rapide, compatibilité multi-plateforme.
- Maintenance assurée : mise à jour automatique et fiabilité.
- Accessibilité native : compatibilité avec les lecteurs d'écran et thèmes d'entreprise.
- Intégration simple : aucun développement requis.

Limitations clés

- Faible personnalisation : interface de configuration figée.
- Interactivité limitée à ce que prévoit Microsoft ; pas de logique conditionnelle.
- Structure tabulaire obligatoire : peu adapté aux modèles de graphes, répétitions ou réseaux.

- Pas d'accès au DOM, au code ou aux `event listeners`.

Conclusion intermédiaire

Les visuels natifs forment un socle de démarrage fiable, rapide et adapté à 80 % des besoins analytiques en entreprise. Cependant, leur manque de souplesse les rend insuffisants pour des besoins de personnalisation graphique avancée, d'interactivité dynamique ou d'intégration à des architectures complexes. C'est précisément dans ces cas que les visuels personnalisés (Python, R, SDK) prennent toute leur valeur.

2.3 Visuels Python / R : usages, atouts et limites

Power BI étend ses capacités analytiques en permettant l'intégration directe de scripts Python et R, répondant ainsi aux besoins spécifiques non couverts par ses visuels natifs. Ces langages offrent une palette de possibilités très avancées en matière d'analyse statistique, de visualisation spécialisée et de machine learning, répondant particulièrement aux attentes des utilisateurs experts et académiques.

2.3.1 Intérêt d'intégrer Python et R dans Power BI

L'utilisation des langages Python et R dans Power BI répond à plusieurs objectifs précis :

- Création de visuels avancés ou très spécifiques (heatmaps, boxplots, diagrammes radar, clustering visuel).
- Analyse statistique poussée : régressions, tests statistiques, analyses prédictives.
- Préparation et transformation complexe des données directement dans Power Query.
- Accès aux puissantes bibliothèques open source (`ggplot2`, `matplotlib`, `seaborn`, `scikit-learn`, etc.).
- Automatisation de processus analytiques au sein même du reporting.

2.3.2 Mécanisme de fonctionnement des visuels Python et R

La mise en place d'un visuel Python ou R dans Power BI suit un workflow précis :

1. L'utilisateur sélectionne les champs pertinents depuis l'interface Power BI.
2. Ces champs sont automatiquement transformés en un `DataFrame` nommé `dataset`.
3. L'utilisateur rédige ou importe son script Python ou R directement dans l'éditeur intégré à Power BI Desktop.

4. Le script est exécuté localement par le moteur installé sur l'ordinateur.
5. Le résultat de l'exécution du script (une image PNG) est affiché dans le rapport.

2.3.3 Prérequis techniques

L'intégration de Python ou R dans Power BI nécessite :

- Une installation locale du moteur Python (via Anaconda ou python.org) ou R (via CRAN).
- Une configuration adéquate dans les options de Power BI Desktop pour pointer vers l'exécutable Python ou R.
- L'installation locale des bibliothèques spécifiques utilisées par les scripts.
- Vérification préalable de la compatibilité des packages avec Power BI Service si publication en ligne (restrictions de sécurité strictes).

2.3.4 Forces majeures des visuels Python et R

Les principaux atouts de ces visuels sont :

- **Flexibilité graphique totale** : Chaque élément visuel est entièrement personnalisable grâce au code.
- **Puissance statistique intégrée** : Capacité à intégrer directement des analyses statistiques complexes et avancées.
- **Accès immédiat à la data science** : Intégration directe de modèles de prédiction, classification, clustering.
- **Réutilisation de scripts existants** : Gain de temps significatif pour les analystes utilisant déjà ces langages dans leur environnement.

2.3.5 Limitations et contraintes à considérer

Cependant, ces visuels comportent des limites notables :

- **Absence d'interactivité dynamique** : Les visuels générés sont des images statiques, ce qui empêche les interactions dynamiques typiques des visuels natifs ou SDK.
- **Performance restreinte** : Traitement local uniquement avec des contraintes de volume de données (max 150 000 lignes, mémoire maximale 250 Mo).

- **Restriction des packages** : Tous les packages disponibles en Python ou R ne sont pas supportés par Power BI Service.
- **Non-compatibilité mobile** : Ces visuels ne sont pas disponibles sur les applications mobiles Power BI.
- **Maintenance complexe** : Le code doit être réédité en local puis republié pour toute modification, augmentant la complexité en production.

2.3.6 Exemples d'usages pertinents en contexte académique et professionnel

Voici quelques scénarios où l'utilisation de Python ou R est particulièrement adaptée :

- **Analyse exploratoire approfondie** : Visualisation complexe comme des heatmaps de corrélations, boxplots détaillés ou graphes radar.
- **Analyses prédictives et ML** : Visualisation directe des résultats de modèles de clustering, régressions, séries temporelles avancées (ARIMA, Prophet).
- **Traitement du langage naturel (NLP)** : Nuages de mots interactifs et analyse textuelle qualitative (opinions clients, analyse sémantique).

2.3.7 Comparaison Python vs R

La décision de choisir entre Python et R dépend des contextes spécifiques du projet :

- **Python** : Langage polyvalent et flexible, idéal pour des analyses ML ou la mise en production rapide de modèles prédictifs grâce à sa large gamme de bibliothèques avancées.
- **R** : Plus adapté aux analyses statistiques poussées, réputé pour sa visualisation riche et claire via `ggplot2`, particulièrement pertinent dans un cadre académique.

Conclusion intermédiaire

L'intégration de visuels Python et R dans Power BI constitue une avancée majeure pour les utilisateurs cherchant à dépasser les limites des visuels standards. Malgré leurs contraintes techniques et leur manque d'interactivité directe, ces outils apportent une réelle valeur ajoutée dans les contextes de recherche, de formation ou d'analyse spécialisée. Ils demeurent toutefois moins adaptés à une utilisation quotidienne en entreprise, où les visuels SDK avec interactivité avancée sont généralement plus appropriés pour l'utilisateur final.

2.4 SDK Custom Visuals : principes, sécurité et pipeline

Le Software Development Kit (SDK) pour visuels personnalisés de Power BI permet de créer des composants graphiques entièrement sur mesure en TypeScript, répondant précisément à des besoins analytiques, interactifs ou visuels très spécifiques. Cette section détaille les principes de développement, les aspects sécuritaires à respecter, ainsi que les bonnes pratiques pour gérer un pipeline d'intégration et de déploiement continu (CI/CD).

2.4.1 Principes généraux du développement via SDK

Le développement d'un visuel personnalisé avec le SDK Power BI implique les étapes suivantes :

1. Initialisation du projet via l'outil `pbiviz` (ligne de commande).
2. Définition des capacités du visuel via le fichier `capabilities.json`.
3. Implémentation en TypeScript dans `src/visual.ts` en suivant l'interface `IVisual`.
4. Gestion du rendu graphique à l'aide de bibliothèques comme D3.js, React ou autres frameworks JavaScript.
5. Compilation et packaging du visuel en fichier `.pbiviz` prêt à être déployé.

Le SDK impose une structure modulaire claire, facilitant la maintenance et l'évolution du composant au fil des besoins et des versions du rapport.

2.4.2 Sécurité et isolation des visuels custom

Power BI exécute chaque visuel personnalisé dans une sandbox HTML sécurisée (iframe). Ce mécanisme garantit l'isolation complète du code personnalisé par rapport au reste du rapport, protégeant ainsi :

- Contre les conflits de bibliothèques JavaScript (versions incompatibles).
- Contre les failles potentielles (accès non autorisé au DOM principal).
- En limitant strictement les échanges à travers une API contrôlée (sélections, filtres croisés, thèmes).

Depuis la version 4.6 du SDK, il est obligatoire de déclarer explicitement les privilèges et autorisations nécessaires (fichiers externes, accès réseau) dans le fichier `capabilities.json`. Ces contraintes assurent une transparence complète en matière de sécurité et facilitent l'audit des visuels déployés dans un environnement de production.

2.4.3 Pipeline CI/CD pour les visuels custom

Pour industrialiser efficacement le développement et le déploiement des visuels personnalisés, la mise en place d'un pipeline d'intégration et de déploiement continu (CI/CD) est recommandée. Ce pipeline typique inclut les étapes suivantes :

- **Intégration continue (CI)** :
  - Validation automatique des changements via des tests unitaires et fonctionnels.
  - Compilation du projet TypeScript avec vérification des erreurs et des standards de codage.
  - Génération automatique du package `.pbiviz` pour chaque commit.
- **Déploiement continu (CD)** :
  - Déploiement automatique du visuel sur des environnements de test puis de production.
  - Automatisation des étapes de vérification de sécurité et d'intégration.
  - Gestion simplifiée des versions via le contrôle de source (Git, Azure DevOps, GitHub Actions).

Ce processus garantit une haute qualité logicielle, une traçabilité complète des modifications, et une réduction drastique des risques liés à des mises en production manuelles ou ponctuelles.

2.4.4 Bonnes pratiques et conseils de mise en œuvre

Pour réussir le développement de visuels personnalisés avec le SDK Power BI, il est essentiel de suivre quelques recommandations clés :

- **Structure claire du code** : Séparer distinctement la logique métier (traitement de la dataView) de la logique d'affichage (D3.js, Canvas).
- **Tests rigoureux** : Couvrir au maximum les méthodes critiques avec des tests unitaires et d'intégration (Jest, Mocha).
- **Documentation exhaustive** : Documenter clairement le fonctionnement du visuel, les options disponibles et les limites éventuelles pour faciliter l'intégration par les équipes de développement ou les utilisateurs finaux.

- **Optimisation performance** : Privilégier un rendu différentiel (update incrémental), utiliser des méthodes efficaces pour la gestion des grands jeux de données.

Conclusion intermédiaire

Le SDK Custom Visuals de Power BI offre une solution optimale pour répondre à des besoins spécifiques dépassant les capacités natives ou les limites des visuels Python et R. Grâce à son cadre structurant, ses exigences de sécurité élevées, et son potentiel d'automatisation via CI/CD, il représente la meilleure option pour un usage professionnel à grande échelle. Ce choix technologique s'impose particulièrement lorsque la performance, l'interactivité dynamique, et la personnalisation graphique poussée sont des critères essentiels du projet BI.

2.5 Test 2.4

Ceci est un test.

3 Conclusion

Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Ut purus elit, vestibulum ut, placerat ac, adipiscing vitae, felis. Curabitur dictum gravida mauris. Nam arcu libero, nonummy eget, consectetuer id, vulputate a, magna. Donec vehicula augue eu neque. Pellentesque habitant morbi tristique senectus et netus et malesuada fames ac turpis egestas. Mauris ut leo. Cras viverra metus rhoncus sem. Nulla et lectus vestibulum urna fringilla ultrices. Phasellus eu tellus sit amet tortor gravida placerat. Integer sapien est, iaculis in, pretium quis, viverra ac, nunc. Praesent eget sem vel leo ultrices bibendum. Aenean faucibus. Morbi dolor nulla, malesuada eu, pulvinar at, mollis ac, nulla. Curabitur auctor semper nulla. Donec varius orci eget risus. Duis nibh mi, congue eu, accumsan eleifend, sagittis quis, diam. Duis eget orci sit amet orci dignissim rutrum.

Nam dui ligula, fringilla a, euismod sodales, sollicitudin vel, wisi. Morbi auctor lorem non justo. Nam lacus libero, pretium at, lobortis vitae, ultricies et, tellus. Donec aliquet, tortor sed accumsan bibendum, erat ligula aliquet magna, vitae ornare odio metus a mi. Morbi ac orci et nisl hendrerit mollis. Suspendisse ut massa. Cras nec ante. Pellentesque a nulla. Cum sociis natoque penatibus et magnis dis parturient montes, nascetur ridiculus mus. Aliquam tincidunt urna. Nulla ullamcorper vestibulum turpis. Pellentesque cursus luctus mauris.

I An appendix

Fusce mauris. Vestibulum luctus nibh at lectus. Sed bibendum, nulla a faucibus semper, leo velit ultricies tellus, ac venenatis arcu wisi vel nisl. Vestibulum diam. Aliquam pellentesque, augue quis sagittis posuere, turpis lacus congue quam, in hendrerit risus eros eget felis. Maecenas eget erat in sapien mattis porttitor. Vestibulum porttitor. Nulla facilisi. Sed a turpis eu lacus commodo facilisis. Morbi fringilla, wisi in dignissim interdum, justo lectus sagittis dui, et vehicula libero dui cursus dui. Mauris tempor ligula sed lacus. Duis cursus enim ut augue. Cras ac magna. Cras nulla. Nulla egestas. Curabitur a leo. Quisque egestas wisi eget nunc. Nam feugiat lacus vel est. Curabitur consectetuer.

Suspendisse vel felis. Ut lorem lorem, interdum eu, tincidunt sit amet, laoreet vitae, arcu. Aenean faucibus pede eu ante. Praesent enim elit, rutrum at, molestie non, nonummy vel, nisl. Ut lectus eros, malesuada sit amet, fermentum eu, sodales cursus, magna. Donec eu purus. Quisque vehicula, urna sed ultricies auctor, pede lorem egestas dui, et convallis elit erat sed nulla. Donec luctus. Curabitur et nunc. Aliquam dolor odio, commodo pretium, ultricies non, pharetra in, velit. Integer arcu est, nonummy in, fermentum faucibus, egestas vel, odio.

Sed commodo posuere pede. Mauris ut est. Ut quis purus. Sed ac odio. Sed vehicula hendrerit sem. Duis non odio. Morbi ut dui. Sed accumsan risus eget odio. In hac habitasse platea dictumst. Pellentesque non elit. Fusce sed justo eu urna porta tincidunt. Mauris felis odio, sollicitudin sed, volutpat a, ornare ac, erat. Morbi quis dolor. Donec pellentesque, erat ac sagittis semper, nunc dui lobortis purus, quis congue purus metus ultricies tellus. Proin et quam. Class aptent taciti sociosqu ad litora torquent per conubia nostra, per inceptos hymenaeos. Praesent sapien turpis, fermentum vel, eleifend faucibus, vehicula eu, lacus.

Pellentesque habitant morbi tristique senectus et netus et malesuada fames ac turpis egestas. Donec odio elit, dictum in, hendrerit sit amet, egestas sed, leo. Praesent feugiat sapien aliquet odio. Integer vitae justo. Aliquam vestibulum fringilla lorem. Sed neque lectus, consectetuer at, consectetuer sed, eleifend ac, lectus. Nulla facilisi. Pellentesque eget lectus. Proin eu metus. Sed porttitor. In hac habitasse platea dictumst. Suspendisse eu lectus. Ut mi mi, lacinia sit amet, placerat et, mollis vitae, dui. Sed ante tellus, tristique ut, iaculis eu, malesuada ac, dui. Mauris nibh leo, facilisis non, adipiscing quis, ultrices a, dui.

Références

CORPORATION, M. (2016). Power BI Custom Visuals Security and Sandboxing. https://docs.microsoft.com/en-us/power-bi/developer/visuals/security. Consulté en juin 2025.

DOCS, M. (2024). capabilities.json structure (Power BI visuals). https://learn.microsoft.com/en-us/power-bi/developer/visuals/capabilities-json. Documentation officielle SDK, version 4.6, consultée en juin 2025.

MICROSOFT. (2023). Power BI Visuals API Version 4.6. https://github.com/microsoft/PowerBI-visuals-tools/wiki/API-Version-4.6. GitHub SDK Power BI, consulté en juin 2025.

TEAM, M. D. (2025). Power BI Custom Visuals API v5.0 release notes. https://github.com/microsoft/PowerBI-visuals-tools/releases/tag/v5.0. Consulté en juin 2025.

Informations sur ce travail

Informations de contact

Auteur: Blendar Berisha

HES-SO Valais-Wallis

E-mail: *blendar.berisha@students.hevs.ch*

Déclaration sur l'honneur

Je déclare, par ce document, que j'ai effectué le travail de bachelor ci-annexé seul, sans autre aide que celles dûment signalées dans les références, et que je n'ai utilisé que les sources expressément mentionnées. Je ne donnerai aucune copie de ce rapport à un tiers sans l'autorisation conjointe du RF et du professeur chargé du suivi du travail de bachelor, à l'exception des personnes qui m'ont fourni les principales informations nécessaires à la rédaction de ce travail.

Lieu, date: _____

Signature: _____