Rapport technique - Portail Résilience

ICEDD
INSTITUT DE CONSEIL ET D'ÉTUDES
EN DÉVELOPPEMENT DURABLE

Mai 2023

Bouteille Thomas

Table des matières

1 Introduction **3**

2 Données **4**

2.1 Source 4

2.1.1 ICEED 4

2.1.2 WalStat 5

2.1.3 StatBel 5

2.2 Récupération 6

2.3 Transformation 6

2.3.1 calcul des valeurs 7

3 Détail d'implémetation **15**

3.1 Architecture du projet 16

4 Modification **18**

4.1 Modifier les couleurs 19

4.2 Modifier le texte 19

4.3 Affichage des données dans le texte 23

4.4 Ajouter un nouveau chapitre 24

A Annexe **26**

A.1 Structure des données 27

Chapitre 1

Introduction

Le portail résilience présente à travers 7 sections les types de risques environnementaux majeurs en Wallonie et les secteurs fortement impactés lié aux changements climatiques. Son objectif est de mieux comprendre les impacts des changements climatiques grâce à des techniques de visualisation. Ce document présente l'implémentation de ce portail et des techniques de visualisation utilisées, il contient trois parties distinctes permettant de mieux comprendre le développement du portail. Premièrement, les données utilisées sur le portail sont présentées et leurs sources et transformations nécessaires documentées. Deuxièmement, l'implémentation du portail est décrite, mettant l'accent sur les technologies utilisées et les choix d'implémentation concernant les graphiques. Troisièmement, une partie servant de guide à toute personne souhaitant modifier le contenu du portail, cela peut être la modification de texte ou l'ajout de section.

Chapitre 2

Données

Les données affichées sur le portail proviennent de plusieurs sources. Certaines sont internes à l'ICEDD, d'autres sont récupérées via des sources extérieures. Les données récupérées subissent une transformation telle qu'elles peuvent être utilisées par le portail.

2.1 Source

Cette sous-section reprend en détail les sources de données utilisées, donne leur provenance et explique brièvement leur utilisation sur le portail.

2.1.1 ICEED

Pour les données internes à l'ICEDD. Deux sources sont utilisées :

1. **ECL** - Reprend les données relatives aux cartographies permettant de mieux appréhender certaines vulnérabilités.
2. **Donnees_Portail** - Reprend des informations plus génériques que l'on retrouve sur le portail original.

2.1.2 WalStat

Lien : WalStat.

Il s'agit d'une source externe qui permet de récupérer des données concernant la population wallonnes sur certains aspects. Les données sont structurées sont forme d'identificateur.

1. **200300** - Reprend les données démographiques par communes.
2. **201001** - Reprend la superficie par communes.
3. **215700** - Reprend les superficies associées à l'artificialisation du territoire par communes.
4. **201001** - Reprend les données concernant la taille et le nombre de ménage par communes.
5. **811302** - Reprend la consommation d'eau en m^3 par communes.

Ces données peuvent être récupérées via des fichiers .xlsx disponible en téléchargement sur leur site internet, ou via leur api gratuite. Toutes les informations concernant cette api peuvent être trouvées dans leur documentation. Pour ce projet, la récupération des données n'a utilisé que les fichiers .xlsx.

2.1.3 StatBel

Lien : Agriculture et Production/Consommation

Source externe qui permet de récupérer des données concernant l'agriculture par communes.

1. **DBREF-L05-2022-TAB-B-2-FR** - Reprend les superficies en are associées à l'exploitation agricole et horticoles par communes.

2. **Billans d'approvisionnement viande.** - Reprend les données concernant la production et consommation de viande en Belgique
3. **Billans d'approvisionnement huiles et graisses** - Reprend les données concernant la production et consommation d'huile et de graisses en Belgique
4. **Billans d'approvisionnement lait** - Reprend les données concernant la production et consommation de lait en Belgique
5. **Billans d'approvisionnement fruits et légumes** - Reprend les données concernant la production et consommation de fruit et légume en Belgique.
6. **Billans d'approvisionnement pommes de terre** - Reprend les superficies en are associées à l'exploitation agricole et horticoles par communes.

2.2 Récupération

Les données sont regroupées grâce à un script Python. Un .pdf servant de documentation est présent à la racine du projet et explique en détail comment le script fonctionne. À noter que ce script n'est pas optimal et doit être revu s'il veut être utilisé pour un autre projet. Le fichier sources.yaml associé au projet peut être trouvé dans ./script du repository.

2.3 Transformation

Afin de pouvoir utiliser les données sur le portail, celles-ci doivent être transformées et structurées adéquatement. Pour pouvoir communiquer le plus

facilement avec le portail, l'extension .json a été choisie comme format de fichier. La structure de ce fichier suit le schéma qui peut être trouvé en Annexe A.1.

2.3.1 calcul des valeurs

Cette section reprend tous les calcul pour chaque champ utilisé dans le fichier de données.

Municipality	
ins	-
name	-
superficy	-
Biodiversity	
carbonStorage	95 / (superficie Wallonie en km^2) * (superficie commune en km^2)
airFiltrationLeisureHealth	120 / (superficie Wallonie en km^2) * (superficie commune en km^2)
woodProduction	120 / (superficie de production de bois en Wallonie en km^2) * (supeficie de la commune en km^2 * part des forêt pour la commune / 100)
agriculturalArea	15 / (superficie agricole en Wallonie en km^2) * (superficie de la commune en km^2 * (part des terre arable pour la commune + part des terre) / 100)
surfaceWaterArea	90 / (superficie d'eau en Wallonie en km^2) * (superficie de la commune en km^2 * (part des zones humide pour la commune + part des surface d'eau pour la commune) / 100)

Water	
rationCost	nombre de ménage de la commune * consommation d'eau par ménage en m^3 * 10,8125
Heat	
nbHospitalAtRisk	-
nbNurseryAtRisk	-
nbNursingHomeAtRisk	-
nbPeopleAtRisk	part résidentiel dans zone à fort coefficient d'imperméabilisation * population commune
pctPeopleAtRisk	part résidentiel dans zone à fort coefficient d'imperméabilisation * 100
excessDeathCount	300 / population Wallonne * population commune
displayedDeathCount	excessDeathCount arrondi à la valeur supérieure
excessCost	30000000 / population Wallonne * population commune
displayedExcessCost	excessCost arrondi à la dizaine de millier

Health	
nbDeathByAirPollutants	11000 / population Belgique * population commune
Flood	
nbRetailOfficesServicesAtRisk	-
nbIndustryCraftsAtRisk	-
nbResidencesAtRisk	-
pctRetailOfficesServicesAtRisk	part des commerces, bureaux et services en zone d'aléa d'inondation * 100
pctIndustryCraftsAtRisk	part des secteurs d'industries et artisanats en zone d'aléa d'inondation * 100
pctResidencesAtRisk	part des zones résidentielle en zone d'aléa d'inondation * 100
nbNursingHomeAtRisk	-
nbHospitalAtRisk	-
nbNurseryAtRisk	-
nbPeopleAtRisk	(pctResidencesAtRisk / 100 * population commune) arrondi à la valeur supérieure

Artificialisation	
territory	part superficie artificialisé en 2002 et 2020 pour la commune * superficie commune
residential	part superficie residentielle en 2002 et 2020 pour la commune * superficie commune
displayedTerritory2002	part superficie artificialisé en 2002 * 100
displayedTerritory2020	part superficie artificialisé en 2020 * 100
artificialisationIncrease	(part superficie artificialisé en 2020 - part superficie artificialisé en 2002) * 100
residentialIncrease	(part superficie résidentielle en 2020 - part superficie résidentielle en 2002) * 100
pctArtificialisationIncrease	artificialisationIncrease / part superficie artificialisé en 2020
pctR..S..A..	minimun entre (1 et residentialIncrease / artificialisationIncrease * 100)
residentialShare	part de la superficie du résidentiel pour la commune / 100
commercialShare	part de la superficie des commerces, bureaux et services pour la commune / 100
industrialShare	part de la superficie du secteurs industriel et artisanal pour la commune / 100
population2002	-
population2020	-
populationIncrease	1 - population2002 / population2020 * 100

Agriculture	
usedAgriculturalArea	-
pctUsedAgriculturalArea	usedAgriculturalArea / superficie commune en are
permanentMeadows	-
winterWheat	-
feedCorn	-
potato	-
sugarBeet	-
s..H..C..Oilseed	46206 / population Wallonie * population commune
s..H..C..FruitAndVegetable	528070 / population Wallonie * population commune
s..H..C..Legume	79210 / population Wallonie * population commune
s..H..C..Potato	99013 / population Wallonie * population commune
s..H..C..Sugar	26403 / population Wallonie * population commune
s..H..C..Meat	56768 / population Wallonie * population commune
s..H..C..Milk	198026 / population Wallonie * population commune

s..H..C..Cereal	422456 / population Wallonie * population commune
productionOilseed	523203 / superficie colza/navette Belgique * superficie colza/navette commune
productionFruitAndVegetable	2107174 / superficie fruits et légumes Belgique * superficie fruits et légumes commune
productionLegume	superficie légumineuse commune / 100 * 4.5
productionPotato	2811546 / superficie pommes de terre Belgique * superficie pommes de terre commune
productionSugar	superficie bettraves commune / 100 * 13,7
productionMeat	226119 / animaux Belgique * animaux commune
productionMilk	1275496 / vaches laitières Belgique * vaches laitières commune
productionCereal	superficie froment commune / 100 * 8,3 + superficie orge commune / 100 * 8,1 + superficie maïs commune / 100 * 11,7 + superficie triticale / 100 * 6,3
consumptionF..AndV...	consommation fruit en légume en kg/hab en Belgique / 1000 * population commune
consumptionLegume	6,601 / population Wallonie * population commune
consumptionPotato	consommation pommes de terres en kg/hab en Belgique / 1000 * population commune
consumptionSugar	45 * population commune
consumptionMeat	consommation viande en kg/hab en Belgique * population commune

consumptionCereal	80 * population commune
consumptionMilk	consommation lait en kg/hab en Belgique * population commune

Chapitre 3

Détail d'implémetation

FIGURE 3.1 – Logo de Svelte

Le portail est implémenté en svelte, il s'agit d'un framework javascript qui permet de créer des applications web facilement à l'aide de composants. À la différence d'autres framework connus comme React ou Vue, Svelte n'utilise pas de DOM virtuel, mais à la place agit directement sur le DOM, ce qui le rend plus rapide que d'autre framework. Svelte s'accorde bien avec des librairies qui permettent de construire facilement des graphiques sous forme de composant, qui peuvent être réutilisés pour d'autres projets. De plus, le framework offre des outils permettant de mieux implémenter l'aspect "storytelling" mis en place sur le portail. Svelte est également populaire après des développeurs, et beaucoup de ressources sont disponible sur Internet. C'est pour toutes ces

raisons que ce framework a été choisi pour l'implémentation du portail.

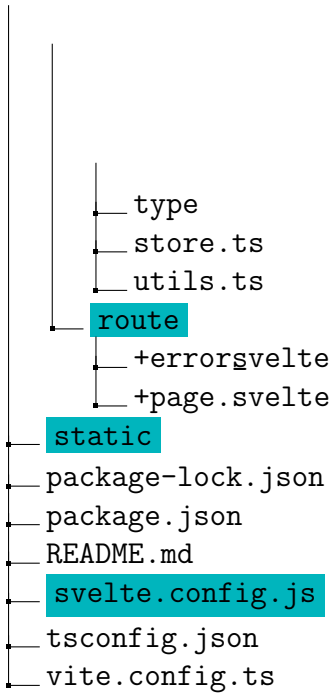
FIGURE 3.2 – Logo de D3.js

Pour l'implémentation des graphiques, la librairie d3.js a été utilisée. Cette librairie permet de créer des graphiques facilement sous forme de canvas ou svg. Accompagnée avec Svelte, elle offre un outil très puissant dans la création de graphiques interactif et narratif. Toutes les fonctionnalités de cette librairie n'étaient pas intéressantes pour ce projet, seule les fonction de permettant de situer des éléments svg dans un cadre ont été utilisés. Pour avoir plus d'information sur le fonctionnement, je vous invite à regarder le code du répertoire components du projet. Il contient tout le code relatif à la création de graphique.

3.1 Architecture du projet

Le projet est structuré de la manière suivante.

```
portail-awac-frontend
└── src
    └── lib
        ├── assets
        ├── components
        └── sections
```



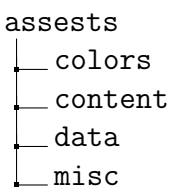
Les répertoires/fichiers en couleur représentent les parties les plus importantes à connaître pour le projet.

- assets contient toutes les données du portail (couleur, texte, données affichés, ...).
- section content tout le code relatif à la construction des différentes sections du portail.
- components contient tout le code relatif aux infographies et graphiques.
- route contient les pages du portail.
- static contient les images.
- svelte.config.js est le fichier qui permet de build l'application.

Chapitre 4

Modification

La modification du site passe par le répertoire assets du projet. Dans celui-ci, on retrouve quatre éléments :



colors : Répertoire contenant un seul fichier, colors.js, qui regroupe toutes les couleurs portail.

content : Répertoire contenant une série de fichiers représentant chaque section du portail. Ces fichiers contiennent le code HTML à afficher pour chaque carte de la section.

data : Répertoire contenant les données nécessaires pour le portail.

misc : Répertoire contenant des données pour construire des svg.

4.1 Modifier les couleurs

Le fichier ./assests/colors/colors.js reprend les couleurs utilisées dans le portail, vous pouvez les changer à cet endroit. Chaque champ est décrit de manière à indiquer à quoi la couleur correspond. Il y a deux parties dans ce fichier, une partie "css" qui correspond aux couleurs générales du portail. Et, une partie "graph", qui correspond aux couleurs attribuées aux graphiques. Cette distinction est faite pour deux raisons :

1. Cela permet de rapidement savoir si une couleur risque d'impacter les graphiques ou l'entièreté du site.
2. Cela permet de ne pas charger toutes les couleurs dans la partie css du portail.

4.2 Modifier le texte

Le répertoire ./assests/content reprend tous les textes pour chaque carte de chaque section. Chaque fichier possède une structure comme la suivante :

```
export const content = {
    nom_section : [
        'HTML pour la carte 1',
        'HTML pour la carte 2',
        ...
    ],
    nom_section_2 : [
        'HTML pour la carte 1',
        ...
    ],
    ...
}
```

Il y a trois éléments à noter avec cette structure.

1. Certaine section doivent avoir un nom bien défini, car ils sont associés à des graphiques, comme on le peut le voir sur la figure suivante. L'encadré rouge correspond à la carte associé au graphique de l'encadré bleu.

On se base sur le nom de la section dans laquelle sont placé les cartes pour avoir cette disposition. Voici un tableau reprenant le nom des sections qui permettent d'avoir cette disposition pour chaque problématique.

flood.ts	
floodMap	Les cartes seront placé par-dessus une "map" de la Wallonie
filler	Une icône remplie à un certain seuil sera placée sur la droite dans la carte, à côté du texte
people	Certain chiffre seront mis en évidence sur le dessous de la carte.
heat.ts	
exposition	Les cartes seront placées à gauche du graphique.
impact_2	Certain chiffre seront mis en évidence sur le dessous de la carte.
artificialisation.ts	
artificialisation	Les cartes seront placées à gauche du graphique.
comparison	Les cartes seront placées à gauche du graphique.
population	Les cartes seront placées à gauche du graphique.

2. Dans certaine section, il existe des entrées au tableau de chaîne de

caractères qui ne représente pas du code HTML, exemple :

```
export const content = {
    nom_section : [
        'HTML pour la carte 1',
        ...,
        [PAS DU CODE HTML],
        ...
    ]
}
```

Cela représente un "tag" utilisé pour construire une infographie dans le code du portail. Le contenu de ce "tag" doit rester le même si l'on souhaite que l'infographie soit affichée. Cependant, vous pouvez déplacer le "tag" à un autre endroit si nécessaire.

3. Certaine section se termine par une carte offrant un menu pour naviguer entre des adaptation possible pour la problématique présenté. Pour modifier ces menus, il faut aller dans la section qui porte le nom "mesure". Elle est un peu différente des autres, car elle comporte trois champs obligatoires :

   intro : Une chaîne de caractères contenant le code HTML de l'introduction de la carte.

   menu : Un tableau de chaînes de caractères contenant le titre des menus à présenter. Ce tableau est parallèle à texts.

   texts : Un tableau de chaînes de caractères correspondant aux codes HTML du menu sélectionné. Ce tableau et parallèle à menu.

   Exemple :

```
export const content = {
    ...
```

```
    mesure : {
        intro : 'HTML pour l'introduction de la carte',
        menu : [
            'menu 1',
            'menu 2',
            ...
            'menu k'
        ],
        texts: [
            'HTML pour le contenu associe a menu 1',
            'HTML pour le contenu associe a menu 2',
            ...
            'HTML pour le contenu associe a menu k'
        ]
    }
}
```

4.3 Affichage des données dans le texte

Pour afficher une données dans le texte de la carte, il faut placer dans la chaîne de caractère correspondant au code HTML de la carte, à l'endroit où l'on désire afficher la donnée, une balise "¡span¿¡/span¿" dont l'attribut "class" sera le nom du champ attribué à la donnée dans le fichier source.

Imaginons que je désire afficher une donnée qui a le nom "ma_super_data" dans mon fichier de données sources. Pour l'afficher, il faut donc écrire :

```
export const content = {
    nom_section : [
        'HTML pour la carte 1 avec ma nouvelle data <span
           class="ma_super_data"></span>',
    ]
```

```
}
```

L'insertion de la donnée sera faite automatiquement. Cela signifie que vous pouvez ajouter des nouvelles données par commune si nécessaire, et les affichées à l'endroit que vous souhaitez.

4.4 Ajouter un nouveau chapitre

Cette partie nécessite d'aller modifier des fichiers un peu plus profondément dans le code. Un chapitre est défini par un nom, ses liens avec les autres chapitres, une icône et son contenu. Voici la marche à suivre pour ajouter un nouveau chapitre :

1. Dans le fichier ./lib/assests/content/chapter.ts se trouve une variable intitulé "chaptersData". Il s'agit d'un tableau d'objet. Ajoutez une nouvelle entrée représentant le nouveau chapitre tel que l'entrée ait la forme :

```
{
    displayedName: "nomChapitre",
    linkName: "newChapterID"
}
```

2. Dans le fichier ./lib/components/Icon/iconData.ts se trouve une variable intitulé icons. Il s'agit d'un tableau d'objet. Ajouter une nouvelle entrée représentant l'icone du nouveau chapitre tel que l'entrée ait la forme :

```
{
    name: 'nomChapitre',
    fill: ["#FF0000", "#FFFFFF"],
    d: ["attribut 'd' d'un 'path' du svg"]
```

```
}
```

Cette partie n'est pas idéal, malheureusement, je n'ai pas eu le temps de la peaufiner pour la rendre pour accessible à la modification.

3. Dans le répertoire ./sections, ajoutez le code correspondant à la section. Le fichier 'Energy.svelte' offre un bon template.
4. Dans ./routes/+page.svelte ajouter un composant Chapter suivit de la section que vous avez créé au point précédent.

Annexe A

Annexe

A.1 Structure des données