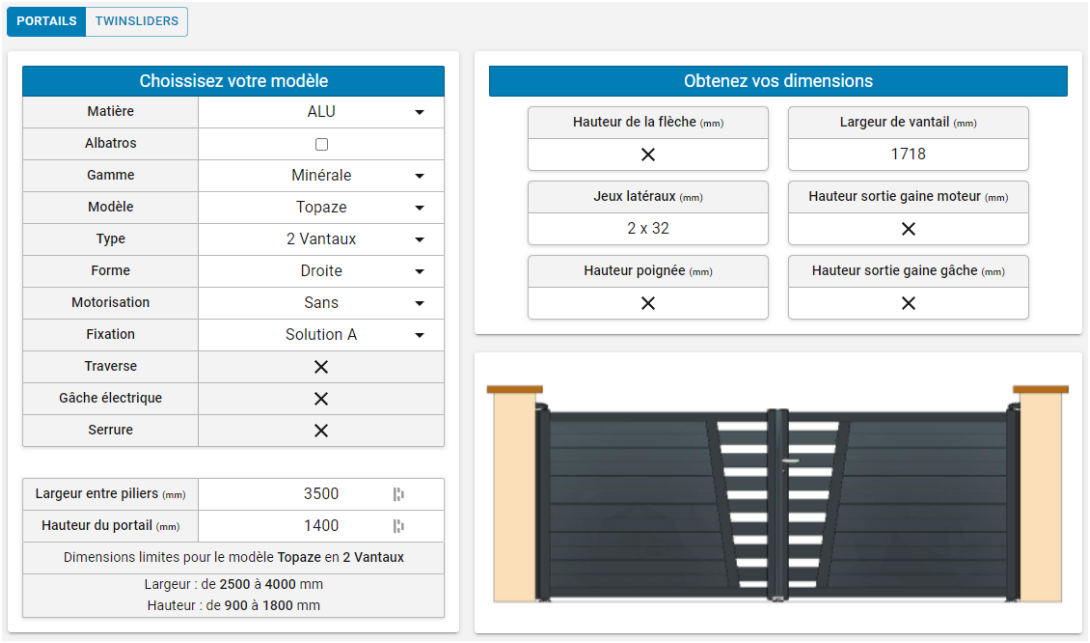
C2.3 : Traiter des demandes concernant les applications

Lors de ma période de stage, j'ai réalisé un calculateur de côtes pour la partie portail de l'entreprise Fybolia. J'ai suivi le cahier des charges demandé par l'entreprise et me basant sur l'ancienne application développé sous Windev.

Cette application devait, avec l'ensemble des gammes de portails disponibles, en fonction de la largeur disponible entre piliers et la hauteur du portail, restituer un panel de valeurs utilisées par la clientèle pour avoir un « rendu virtuel » de ce que donnerait leur portail, mais aussi des dimensions utiles aux poseurs de portails.

Ces valeurs comprenaient : la hauteur de la flèche, la largeur de vantail, les jeux latéraux, la hauteur de sortie de la gaine moteur, de la hauteur de la poignée et la hauteur de la sortie de la gaine gâche. J'ai donc réalisé ces demandes pour l'application :



Un aperçu de l'application développée lors de mon stage dans l'entreprise Fybolia, disponible sur https://hilarious-sprinkles-b1e969.netlify.app/

Lors de la période de test de l'application, j'ai dû répondre aux sollicitations de changements de la part du bureau « R&D » de l'entreprise, afin de corriger des valeurs incorrects ou d'ajouter des « features » pour rendre l'application plus intuitive au niveau de l'expérience utilisateur. J'ai aussi dû adapter mon modèle de donnée pour le rendre « plus simple à modifier par une personne non-informaticienne ». Pendant tous le développement de l'application, j'étais en contact permanent avec l'ensemble des services de l'entreprise concernés.

Jean-Claude PICARD <j-cpicard@fybolia.com>
À moi

Bonjour Sacha.
Voilà ce que j'ai pu observer ce matin :
Modèle 2Vtx Astéra, forme bombée ; Lg 3500 ; 3800 ou 4500 → les flèches données sont toutes à 100mm (en réalité c'est 100 ; 150 et 200)
Modèle 2 Vtx Astéra, forme CDG et CDGi ; Lg 3500 ; 3800 ou 4500 → les flèches données sont toutes à 100mm (en réalité c'est 100 ; 150 et 200)
Ça semble être le même problème pour le Tilleul
Idem quand on choisit coulissant → les flèches données sont toujours de 100 (en réalité c'est 100 ; 150 et 200)
Pour le modèle PIN vertical (également du groupe 1) les flèches semblent être correctes.

Autres remarques :
Sur l'ancien calculateur, si on rentrait une largeur de 5500 pour un ASTERA par ex → la cellule restait bloquée à 4900 (Largeur Maxi autorisée sur la grille de prix)
Aujourd'hui on peut rentrer des valeurs (jusqu'à aberration) supérieures aux valeurs autorisées par les grilles de prix

Détail sur la saisie clavier des valeurs : sur l'ancien calculateur, le curseur dans la cellule provoquait la mise à zéro automatique.
Aujourd'hui, il faut supprimer manuellement l'ancienne valeur avant de rentrer la nouvelle.

Une des remontées du bureau R&D portail concernant mon application

J'ai donc effectué les corrections demandées pour le calcul de mes valeurs. Puis, dans ce cas présent, j'ai créé une fonction qui prenait en argument la valeur rentrée dans l'input « Hauteur du portail » et je regardais si elle était en dessous de la valeur maximum de la hauteur présente dans mon modèle de données.

```
const isToHigh = (e) => {
        let value = undefined
        if (getLimitHauteurMax() < e.target.value) {
            value = getLimitHauteurMax()
        } else {
            value = e.target.value
        }
        return value
    }
```

Un exemple d'une fonction répondant à une demande du service R&D

Concernant la deuxième demande du courriel (la remise à zéro automatique), j'ai opté pour une solution un peu plus intuitive (validée ensuite par le service). J'ai créé une fonction basique JS « On click » qui, lors du click utilisateur, sélectionnez l'ensemble du champ. C'est-à-dire que si l'utilisateur décider de remplacer l'ensemble de la valeur, il n'aurait pas besoin de la supprimer, il aurait juste besoin d'entrer la valeur voulue.

```
onClick={(e) => e.target.select()}
```

La manière choisit pour répondre à la « remise à zéro » d'un champ lors d'un input utilisateur

```
<FormControl className={css.menuIndividuel}>
                            <Controller
                                control={control}
                                name="hauteur"
                                render={({ field }) =>
                                    <TextField
                                        InputProps={{
                                            disableUnderline: true,
                                            inputProps: {
                                                maxLength: "4",
                                                style: { textAlign: "center" }
                                            }
                                        }}
                                        id='hauteur'
                                        placeholder="Hauteur (mm)"
                                        onChange={(e) => { field.onChange(isToHigh(e)); handleHauteurChange(e) }}
                                        value={field.value}
                                        onClick={(e) => e.target.select()}
                                    />
                                }
                            />
                        </FormControl>
```

Les deux fonctions implémentées dans le champ « Hauteur du portail »