C1.2 : Exploiter des référentiels, normes et standards adoptés par le prestataire informatique

J'ai effectué mon stage d'étude dans une entreprise, Fybolia, qui est en pleine refonte de son PGI (Progiciel de Gestion Intégré). Ils ont choisi de développer leur propre ERP sous React JS pour le front-end et Node JS pour le back-end. Durant la phase de conception de mon application (Calculateur de côtes à destination du public, mais aussi des salariés), j'ai dû suivre les normes et les standards que le service avez mis en place.

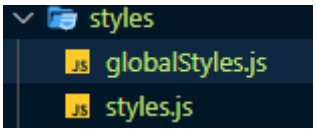
Par exemple, je devais respecter au maximum **leur chartre graphique**. L'entreprise avait définit 5 thèmes pour les utilisateurs et je devais piocher mes couleurs dedans. C'était aussi le cas pour l'aspect général de mon application. Elle devait tenir, par exemple, sur une page (100vh).

```
const Theme = createTheme({
    palette: {
        primary: {
            main: ThemeConfig.primary.main,
            contrastText: ThemeConfig.primary.text
        },
        secondary: {
            main: ThemeConfig.secondary.main,
            contrastText: ThemeConfig.secondary.text
        },
        background: {
            default: '#B2A0FF',
            main: ThemeConfig.secondary.main,
            paper: ThemeConfig.secondary.main,
            contrastText: ThemeConfig.secondary.text,
            secondary: ThemeConfig.secondary.variant1,
            light: ThemeConfig.secondary.variant2,
        },
        text: {
            primary: ThemeConfig.secondary.text,
            secondary: ThemeConfig.secondary.text,
            disabled: ThemeConfig.secondary.text,
        }
    },
```

Récupération des couleurs des thèmes prédéfinis

Je devais aussi respect **l'architecture générale du code,** adopté par l'ensemble des développeurs**.** Pour ce faire, j'ai dû apprendre à utiliser un **Framework** destiné à React JS, **Material UI**.

Dans un premier temps, j'ai dû reprendre leur fichier de styles afin de personnaliser mes composants, dans la lignée de ce que l'entreprise faisait :



globalStyles.js est le fichiers style « par défaut » de l'ERP

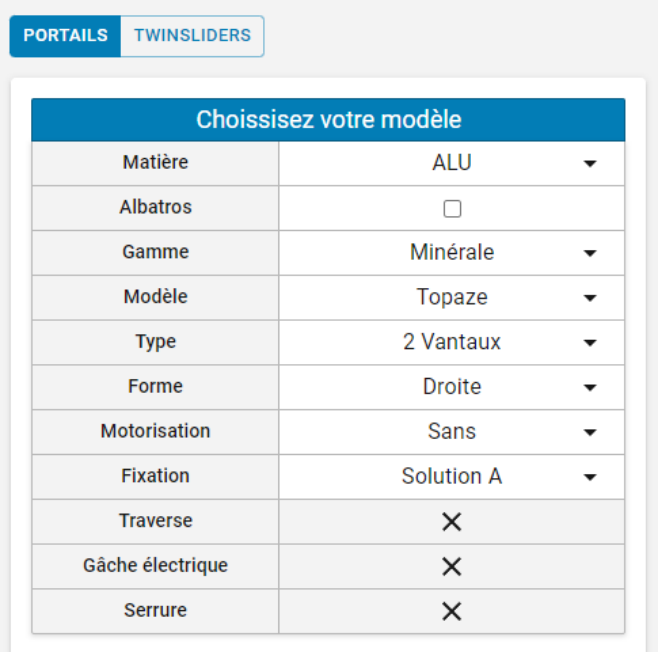
J'ai dû aussi m'adapter au fait que l'équipe utilisait le JSS à la place du CSS (Le JSS est du CSS compilé).

```jss
import { makeStyles } from "@material-ui/core/styles";

const globalStyle = makeStyles((theme) => ({
    LayoutRoot: {
        backgroundColor: theme.palette.background.main,
        display: 'flex',
        height: '100%',
        overflow: 'hidden',
        width: '100%'
    },
    cardContent: {
        backgroundColor: theme.palette.background.main,
        color: theme.palette.background.contrastText,
        padding: "15px!important",
        position: 'relative'
    },
    LayoutWrapper: {
        display: 'flex',
        flex: '1 1 auto',
        overflow: 'hidden',
        paddingTop: 64,
        zIndex: 0,
        backgroundColor: theme.palette.background.light,
        [theme.breakpoints.up('lg')]: {
            paddingLeft: 75
        }
    },
```

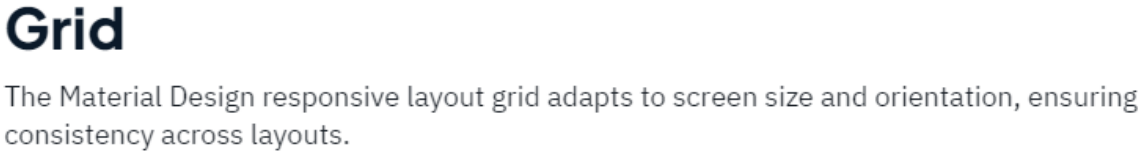
Extrait des styles des composants de base de Material UI, écrit en JSS

J'ai dû construire mon application sur les composants que propose le framework Material UI,



Aperçu du rendus avec les composants Material UI

Mais aussi grâce au « layout » que ce Framework met à disposition, à savoir un système de « Grid » :



Extrait de la documentation de Material UI - https://mui.com/material-ui/react-grid/#main-content

```
<Grid
                item
                xs={5}
                style={{ marginRight: 10 }}
            >
                <Grid container direction="column" className={[css.containerResultat, css.shadowB].join(' ')} >
                    <Grid item xs={12} className={css.labelResultat}>
                        <Typography variant='subtitle2'>Hauteur poignée <span style={{ fontSize: 10 }}>(mm)</span></Typography>
                    </Grid>
                    <Grid item xs={12} className={css.centerResultat}>
                        {Utils.objUndef(configPortail) && configPortail.type === 'Portillon' && configPortail.gache === true ? <FormControl className={css.menuIndividuel}>
                            <Controller
                                control={control}
                                name="poignee"
                                render={({ field }) =>
                                    <TextField
                                        onChange={(e) => {
                                            field.onChange(e);
                                        }}
                                        InputProps={{
                                            disableUnderline: true,
                                            inputProps: {
                                                style: { textAlign: "center" }
                                            }
                                        }}
```

Un exemple de mon code utilisant l'architecture du Framework Material UI avec les props de React JS