FACULTÉ DES SCIENCES ET TECHNIQUES
TANGER LSI-1- 2020/2021

PROJET :APPLICATION WEB **ORE Risk Dashboard**

Rapport :Stage interne

Encadrée par : Prof. Lotfi EL AACHAK

Réalisé par
ET-TOUIL Younes

Table des matières

1 Introduction **1**

2 BackEnd Django **2**

2.1 Models : 2

2.1.1 Exemples : 2

2.2 Rest Api : 3

2.2.1 serialization : 3

2.2.2 Views.py : 4

2.2.3 Urls : 5

3 FrontEnd ReactJs **6**

3.1 Axios pour gerer les APIs 6

3.2 Components : 6

3.2.1 Components Pages : 7

3.3 Les Vues : 8

3.3.1 Page Credit Rating : 8

3.3.2 Page Counterparty : 9

3.3.3 Page Limits : 10

4 Conclusion **11**

1 Introduction

Objectif :
L'objectif principal de la formation est de reconcevoir l'architecture du projet maven/java vers Python et "ReactJs" et le projet est un tableau de bord des risques un affichage graphique des principales mesures de risque d'entreprise généralement utilisées dans les rapports à la haute direction.
l'application sera composée de deux parties, une partie Front end ≪ReactJs≫, et la partie Back end ≪ Django ≫.
-Backend : Il gérera la base de données et convertire les donnee de extension.CSV vers des tableaux dans Base de donnee SQL
-Frontend : Il gérera la Partie presentation des graphes et gestion des api

2 BackEnd Django

Django est un cadre de développement web open source en Python.
Il a pour but de rendre le développement web 2.0 simple et rapide. Pour cette raison, le projet a pour slogan « Le framework pour les perfectionnistes avec des deadlines. ». Développé en 2003 pour le journal local de Lawrence (État du Kansas, aux États-Unis), Django a été publié sous licence BSD à partir de juillet 2005.

2.1 Models :

Un modèle est la source unique et définitive d'informations sur vos données.
Il contient les champs et comportements essentiels des données que vous stockez.
En règle générale, chaque modèle correspond à une seule table de base de données.
Les bases : -Chaque modèle est une classe Python qui sous-classe django.db.models.Model.
-Chaque attribut du modèle représente un champ de base de données.
-Avec tout cela, Django vous offre une API d'accès à la base de données générée automatiquement

2.1.1 Exemples :

```python
class npv_total(models.Model):
    date=models.DateField(null=True)
    Total = models.CharField(max_length=150)
    TradeType = models.CharField(max_length=150)
    Maturity = models.CharField(max_length=150)
    MaturityTime = models.CharField(max_length=150)
    NPV = models.CharField(max_length=150)
    NpvCurrency=models.CharField(max_length=150)
    NPV_Base=models.FloatField()
    BaseCurrency=models.CharField(max_length=150)
    CE_Base=models.FloatField()
    def __str__(self):
        return self.Total
```

Model NPV Total

```python
class npv_nettingset(models.Model):
    date=models.DateField(null=True)
    nettingset = models.CharField(max_length=150)
    TradeType = models.CharField(max_length=150)
    Maturity = models.CharField(max_length=150)
    MaturityTime = models.CharField(max_length=150)
    NPV = models.CharField(max_length=150)
    NpvCurrency=models.CharField(max_length=150)
    NPV_Base=models.FloatField()
    BaseCurrency=models.CharField(max_length=150)
    CE_Base=models.FloatField()

    def __str__(self):
        return self.nettingset
```

Model NPV Netting set

```python
class  Limits_CreditRating(models.Model):
        Limits = models.CharField(max_length=150)
        NPV = models.CharField(max_length=150)
        CE = models.FloatField()
        EEPE = models.FloatField()
        Total_Exposeur = models.CharField(max_length=150)
        ColVA = models.CharField(max_length=150)
        CVA = models.FloatField()
        DVA = models.FloatField()
        FCA =models.FloatField()
        FBA = models.FloatField()
        FVA =models.FloatField()
        IM = models.CharField(max_length=150)
        def __str__(self):
            return self.Limits
class  Limits_Nettingset(models.Model):
        Limits = models.CharField(max_length=150)
        NPV = models.CharField(max_length=150)
        CE = models.FloatField()
        EEPE = models.FloatField()
        Total_Exposeur = models.CharField(max_length=150)
        ColVA = models.CharField(max_length=150)
        CVA = models.FloatField()
        DVA = models.FloatField()
        FCA =models.FloatField()
        FBA = models.FloatField()
        FVA =models.FloatField()
        IM = models.CharField(max_length=150)
        def __str__(self):
            return self.Limits
```

Model Limits Credit Rating Limits netting Set

2.2 Rest Api :

Django REST Framework est une boite à outils puissante et flexible qui vous facilite la création d'application web API .

2.2.1 serialization :

Le cadre de sérialisation de Django fournit un mécanisme pour « traduire » les modèles Django dans d'autres formats. Habituellement, ces autres formats seront basés sur du texte et utilisés pour envoyer des données Django sur un câble, mais

il est possible pour un sérialiseur de gérer n'importe quel format (basé sur du texte ou non).

Exemples :

```python
from rest_framework import serializers
from .models import *

class npv_totalserializer(serializers.ModelSerializer):

    class Meta:
        model= npv_total
        fields='__all__'
```

```python
class limits_TrSeerializer(serializers.ModelSerializer):
    class Meta:
        model= Limits_Trade
        fields= '__all__'
```

exemple 2

2.2.2 Views.py :

Une vue est un appelable qui prend une requête et renvoie une réponse. Cela peut être plus qu'une simple fonction, et Django fournit un exemple de certaines classes qui peuvent être utilisées comme vues. Ceux-ci vous permettent de structurer vos vues et de réutiliser le code en exploitant l'héritage et les mixins.

Exemples :

```python
csvApp > views.py > ...
import csv, io
from django.shortcuts import render
from django.contrib import messages
from .models import *
from .serializers import *
from django.shortcuts import render
from rest_framework import viewsets

# Create your views here.

class npvView(viewsets.ModelViewSet):
    serializer_class = npv_totalserializer
    queryset = npv_total.objects.all()
```

exemple 1

```python
class limitsTradeView(viewsets.ModelViewSet):
    serializer_class = limits_TrSeerializer
    queryset = Limits_Trade.objects.all()
```

exemple 2

2.2.3 Urls :

Certains frameworks Web tels que Rails fournissent des fonctionnalités permettant de déterminer automatiquement comment les URL d'une application doivent être mappées à la logique qui traite le traitement des demandes entrantes. Le framework REST ajoute la prise en charge du routage d'URL automatique à Django et vous offre un moyen simple, rapide et cohérent de connecter notre logique de vue à un ensemble d'URL.

Exemples :

```
from django.contrib import admin
from django.urls import path,include
from csvApp import views
from rest_framework import  routers
router = routers.DefaultRouter()
router.register(r'dates',views.datesView,'date')
router.register(r'npvs',views.npvView,'npv')
router.register(r'counterpartys',views.Counterpartyview,'counterparty')
router.register(r'nettingsets',views.nettingsetView,'nettingset')
router.register(r'trades',views.tradeView,'trade')
router.register(r'creditratings',views.creditratingView,'creditrating')
router.register(r'xpvCPs',views.xpv_counterpartyView,'xpvCP')
router.register(r'xpvNss',views.xpv_NettingsetlView,'xpvNs')
router.register(r'xpvCRs',views.xpv_creditratinglView,'xpvCR')
router.register(r'xpvs',views.xpv_totalView,'xpv')
router.register(r'exposeurT',views.exposeurtotalView,'exposeur')
router.register(r'Limits_total',views.limitsTotalView,'lt')
router.register(r'limite_CR',views.limitsCRView,'CR')
router.register(r'Limits_CP',views.limitsCPView,'CP')
router.register(r'Limits_NS',views.limitsNSlView,'NS')
router.register(r'Limits_TR',views.limitsTradeView,'TR')
router.register(r'allData',views.alldataView,'alldata')
router.register(r'allData2',views.alldata2View,'alldata2')

```

APIs Utilisée

3 FrontEnd ReactJs

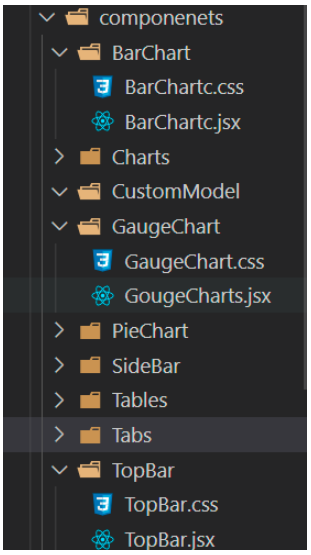
React (également connu sous le nom de React.js ou ReactJS) est une bibliothèque JavaScript frontale gratuite et open source[3] permettant de créer des interfaces utilisateur ou des composants d'interface utilisateur.
Il est maintenu par Facebook et une communauté de développeurs individuels et d'entreprises. React peut servir de base au développement d'applications mono-pages ou mobiles. Cependant, React ne concerne que la gestion de l'état et le rendu de cet état au DOM, donc la création d'applications React nécessite généralement l'utilisation de bibliothèques supplémentaires pour le routage, ainsi que certaines fonctionnalités côté client.

3.1 Axios pour gerer les APIs

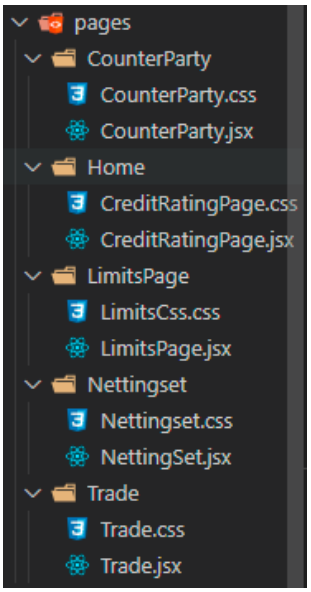
Axios est une bibliothèque JavaScript fonctionnant comme un client HTTP. Elle permet de communiquer avec des API en utilisant des requêtes.

3.2 Components :

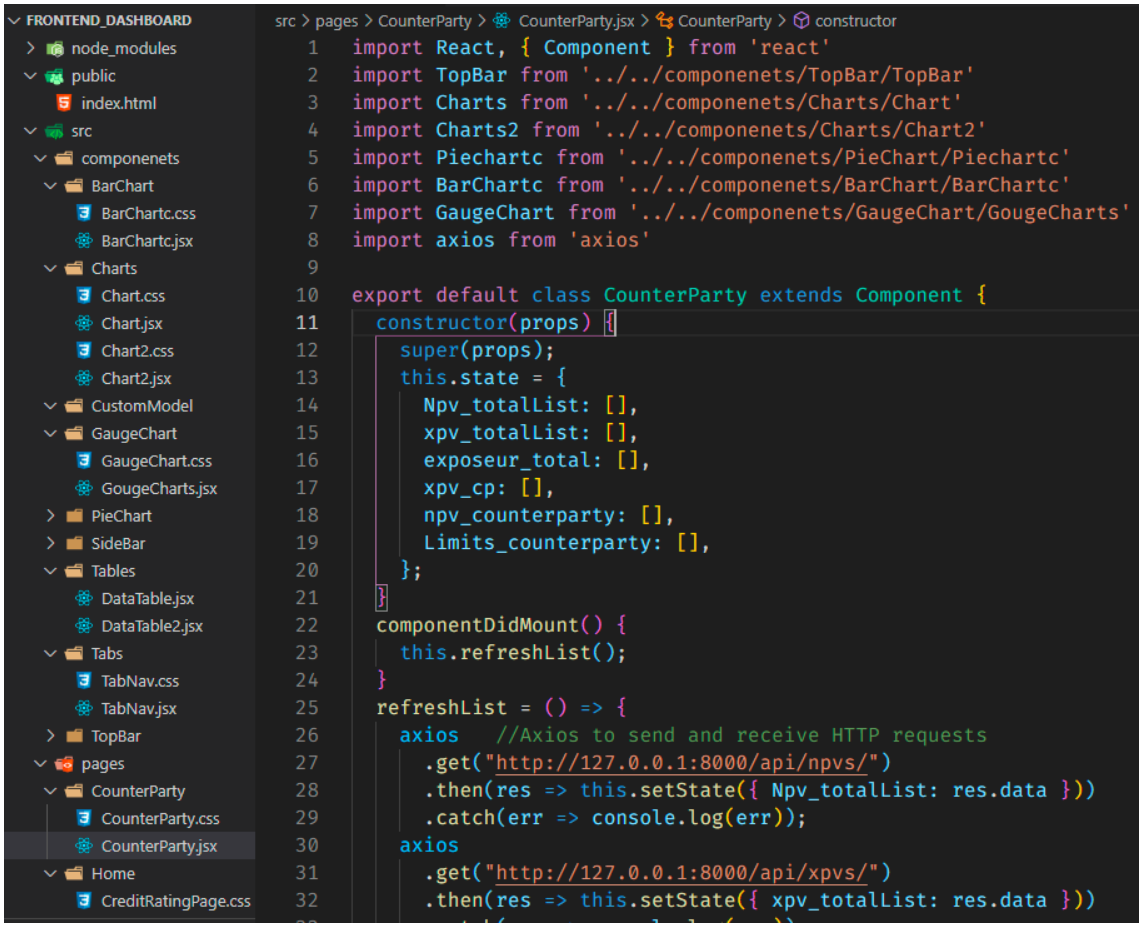
Les composants sont des morceaux de code indépendants et réutilisables. Elles ont le même objectif que les fonctions JavaScript, mais fonctionnent de manière isolée et renvoient du HTML via une fonction render(). Les composants sont de deux types, les composants de classe et les composants de fonction. Dans ce didacticiel, nous nous concentrerons sur les composants de classe.



3.2.1 Components Pages :

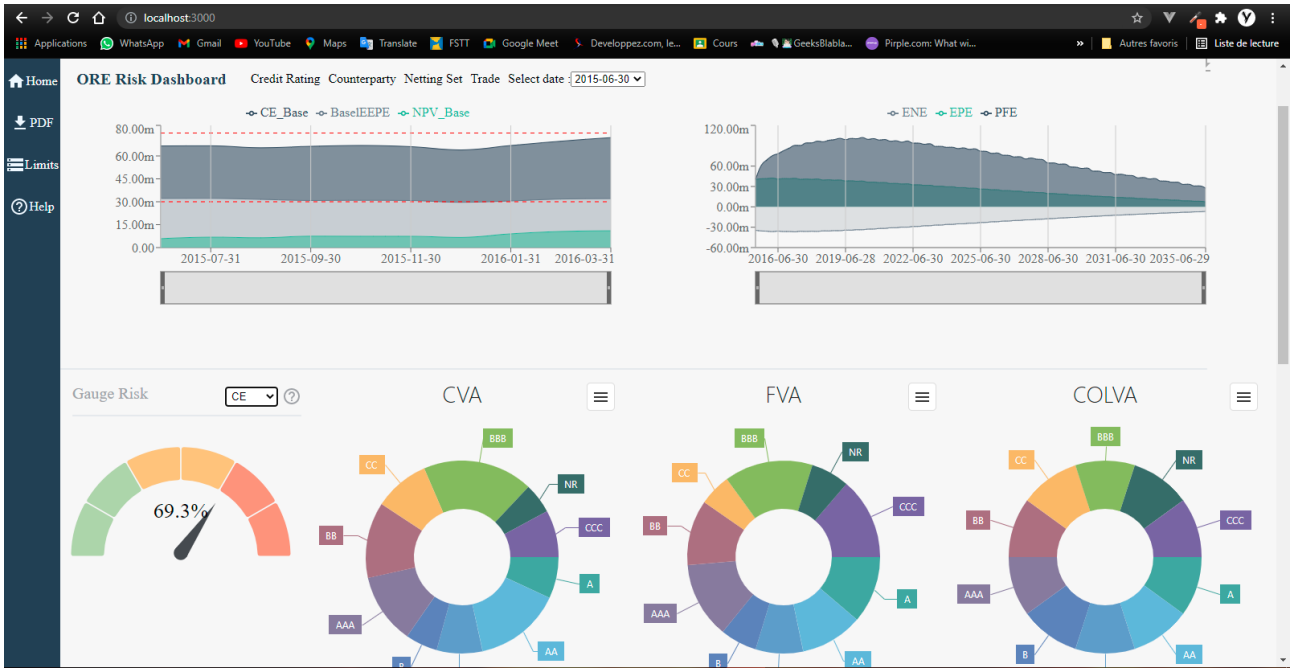


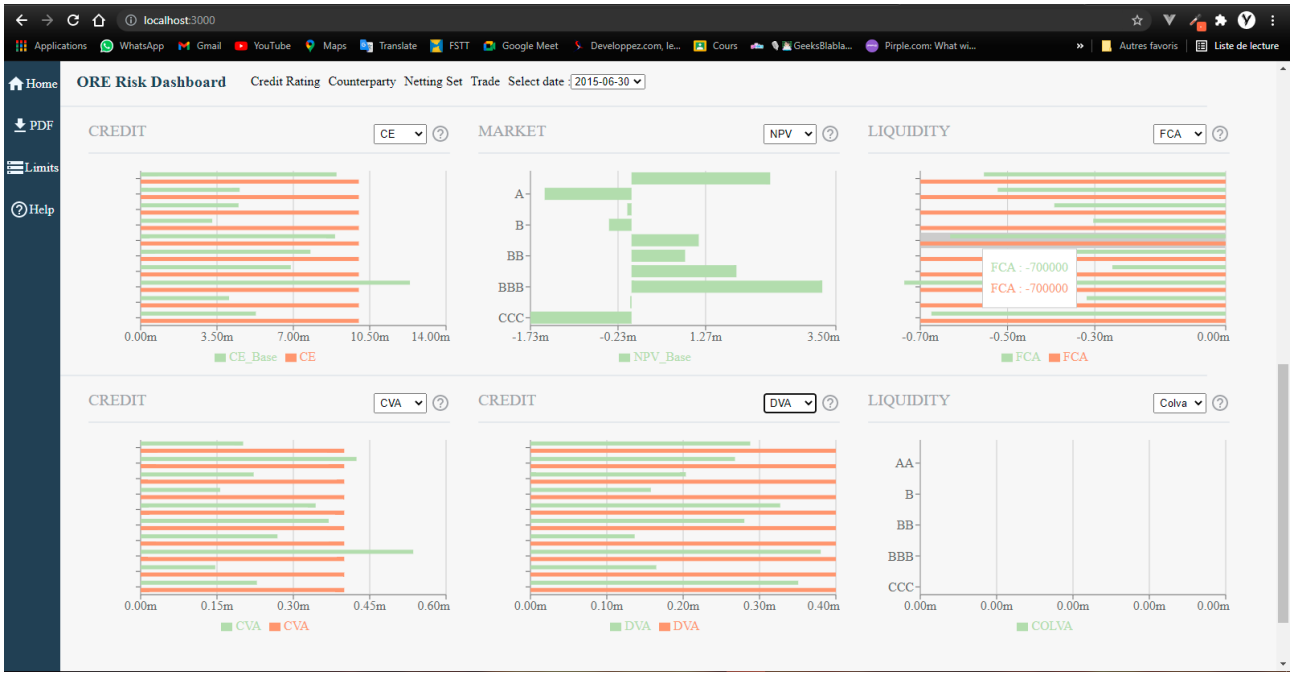
exemple Component :



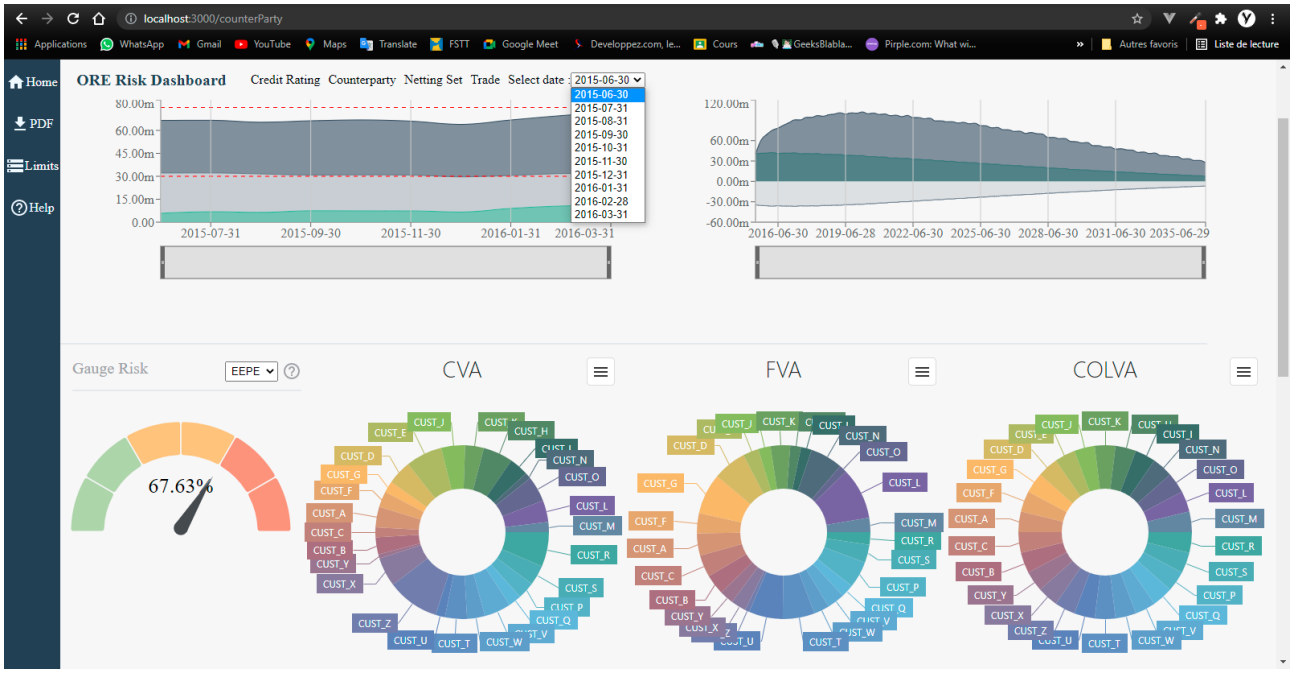
3.3 Les Vues :

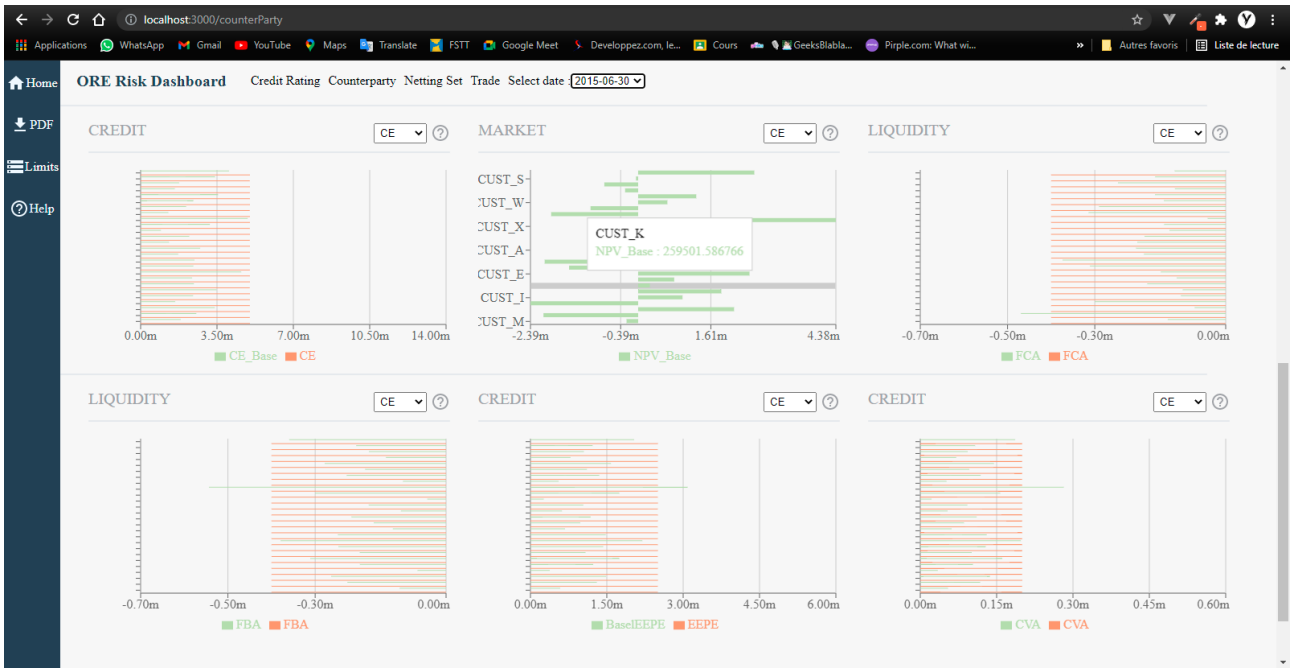
3.3.1 Page Credit Rating :



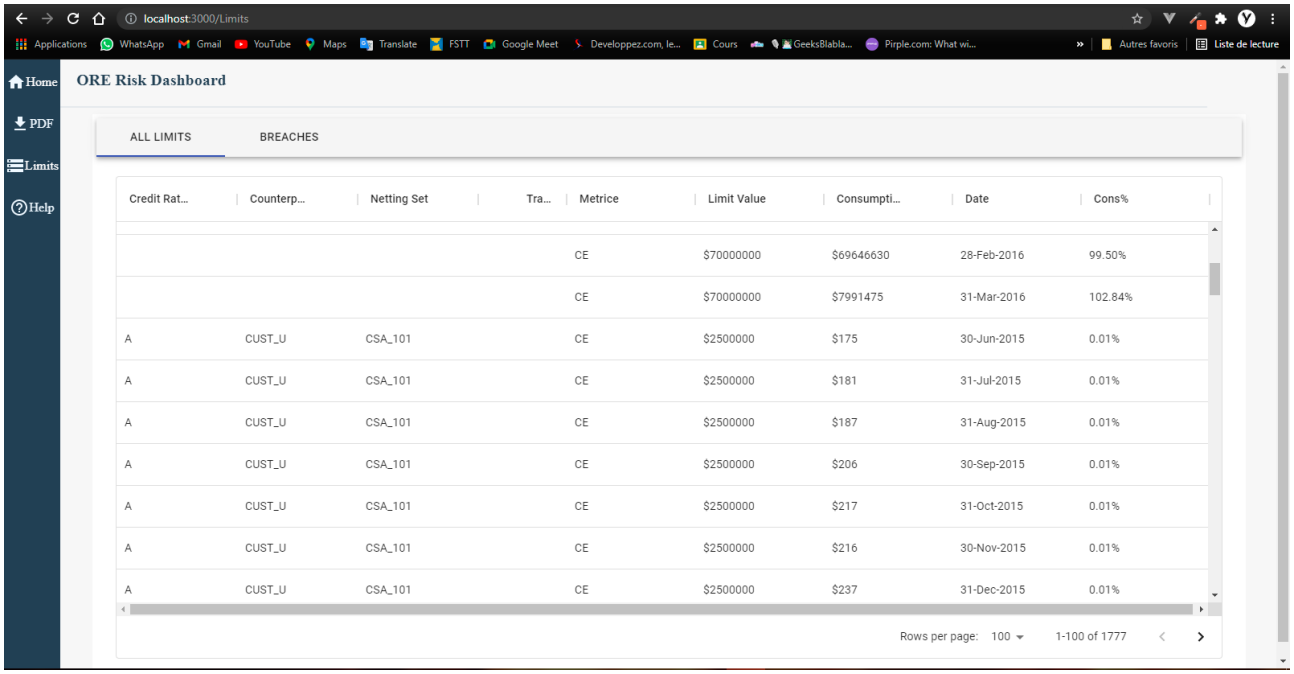


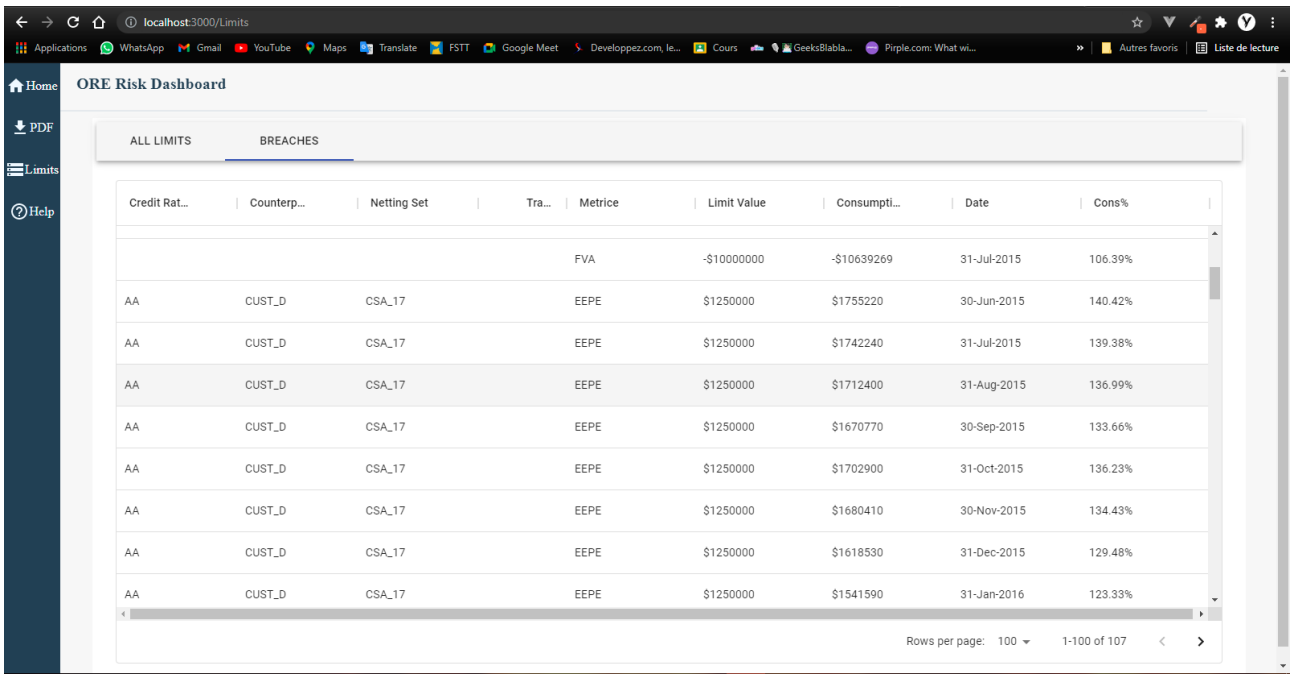
3.3.2 Page Counterparty :





3.3.3 Page Limits :





4 Conclusion

Ce projet nous a permit de bien comprendre le fonctionnement des framworks, et aussi fonctionnement de library de javaScript ReactJs .

Lien vers **Repositorie Github :**
`https://github.com/younes-ettouil/ore-risk-dashboard-code-source`