DEPENDENCIES

index.html

```
  <head>
       <script type="importmap">
      {
        "imports": {
           "three": "https://cdn.skypack.dev/three@0.149.0/build/three.module",
           "STLLoader":
"https://cdn.jsdelivr.net/gh/mrdoob/three.js@dev/examples/jsm/loaders/STLLoader.js",
           "filePriceCalc":
"https://cdn.jsdelivr.net/gh/Furkan-Karakale/filePriceCalc@refs/heads/main/filePriceCalc.js"
        }
      }
    </script>

  </head>
```

ImportMap ile three.js, STLLoader ve filePriceCalc modullerinin CDN uzerinden import edilmesi.

index.html

```
  <body>
    <script src="./js/main.js" type="module"></script>
  </body>
```

Body tagının içerisine yeni javascript dosyasi eklenir.

main.js

```
  import { PRICECALCULATOR } from "./filePriceCalc.js";
```

Kutuphanemiz ve icerisinde “**PRICECALCULATOR**“ classi importlanir.

Class`in Kullanimi

main.js

```
document.getElementById("poule").
  addEventListener("change", onFileSelected);
function onFileSelected(event) {
  if (event.target.files.length > 0) {
    let stlFile = new PRICECALCULATOR(dosyaId);
    let res = stlFile.INIT(event.currentTarget.files[0]);
  }
}
```

Yeni bir model dosyası yüklendiğinde class dosyanin id`si ile cagrilir. Class içerisinde bulunan "**INIT**" fonksiyonu ile dosya class a yuklenir.

main.js

```
document.addEventListener("modelFileLoaded", (e) => {
  let modelClass = e.detail.data;
});
```

Dosyanin yüklenmesi tamamlanmasıyla "modelFileLoaded" eventi tetiklenir. Eventin tetiklenmesi ile class erisime ve fiyat hesaplamaya hazir hale gelir.

Fiyat Hesaplanmasi

Fiyat hesaplanmasi icin üretim yöntemi, hammadde, giderler ve uretim hakkindaki secenekler nesnelerine ihtiyac vardir. Nesneler tanımlandıktan sonra fiyat hesaplamalari yapilabilir.

```
document.addEventListener("modelFileLoaded", (e) => {
  let stlFile = e.detail.data;
  let cost = stlFile.calcPrice(manufacturer1, filament, expenses, selections);
  console.log(cost);
  cost = stlFile.calcPrice(manufacturer2, resin, expenses);
  console.log(cost);
});
```

Ayni STL dosyasının farklı üretim yöntemleri ve hammaddeleri ile fiyat hesaplamanmasinin örneğidir.

Class Ozellikleri ve Fonksiyonlari

- **PRICECALCULATOR.Id** : Class oluşturulduğu sirada verilen INT degerdir.
- **PRICECALCULATOR.FILE** : INIT fonksiyonu ile eklenen dosyaya ulaşabilir.
- **PRICECALCULATOR.price** : Fiyat hesaplamasi yapildiktan sonra ilk defa girilebilir. Urunun son fiyat hesaplamasini gosterir.
- **PRICECALCULATOR.boxSize** : Modelin boyutlari gosterir.
- **PRICECALCULATOR.modelWeight** : Modelin kutlesini gram cinsinden gosterir.
- **PRICECALCULATOR.printTime** : Tahmini üretim süresini saat cinsinden gösterir.
- **PRICECALCULATOR.name** : Modeling adini gosterir.
- **PRICECALCULATOR.mesh** : Three.js kütüphanesi kullnilarak dosyanın içerisinde bulunan 3d modelin mesh`idir.
- **PRICECALCULATOR.canFileReadeble** : Yüklenen modelin mevcut versiyonda okunup okunmadığını gösteren BOOLEAN değişkendir.
- **PRICECALCULATOR.modelFileLoaded** : Dosyanin yüklenmesi ile tetiklenen ozel eventtir.
- **PRICECALCULATOR.loaded** : Dosyanin yuklenme yuzdeligini gosterir.
- **PRICECALCULATOR.INIT(Dosya)** : Dosyanın yüklenmesini saglayan fonksiyondur.
- **PRICECALCULATOR.calcPrice(uretimYontemi,hammadde,giderler,secenekler)**: Modeli ve girdileri kullanarak fiyat hesaplamasi yapar. Fiyat nesnenin içerisine kayit eder. Seçenekler girilmediğinde varsayılan değerler kullanilir.

Class Fonksiyon Geri Dönüşleri ve Durum Kodlari

Tüm fonksiyonlar nesne geri döndürürler. Nesne içerisinde durum kodu ve açıklama bilgisi bulunur. Eğer ki fonksiyon basarili olursa aciklama bilgisi icerisinde fonksiyonun ciktisi bulunur.(Orn: ürün fiyat bilgisi)

- **Status Code 200** : Istek basarili
- **Status Code 400** : Geçersiz istek

```
return {
  status: `200`,
  data: `OK`,
};
```

Geçerli istek örneğidir.

```
return {
  status: `400`,
  data: `${FILE.name} Bu dosya formatini desteklemiyoruz.`,
};
```

Geçersiz istek örneğidir.

Girdi Nesneleri

Girdi nesneleri veri tabanindan veya kullanici tarafindan secilen seceneklerdir. Bu secenekler ve veriler JSON formatinda olup yontem ve hammaddeye gore degisiklik gosterebilir. Orn: FDM üretim şeklinde kullanılan “nozzleThickness” değişkeni bulunurken SLA uretim seklinde bulunmamaktadir.

Uretim Yöntemi

```
let manufacturer = {
  id: 1, //ID
  type: "FDM", //STRING
  powerConsumption: 0.125, //DOUBLE
  deteriorationPrice: 6, //INT
  preparationTime: 0.2, //DOUBLE
  laborMultiplier: 0.5, //DOUBLE
  nozzleThickness: 0.4, //DOUBLE
  minBuildSize: {
    x: 20, //INT
    y: 20, //INT
    z: 5, //INT
  },
  buildVolume: {
    x: 256, //INT
    y: 256, //INT
    z: 256, //INT
  },
};
```

Hammadde

```
 let filament = {
   mType: "FDM", //STRING
   type: "TPU", //STRING
   id: 8, //ID
   color: "BLACK", //STRING
   density: 1.24, //DOUBLE
   price: 550, //DOUBLE
   printSpeed: 6, //DOUBLE
 };
 let resin = {
   mType: "SLA", //STRING
   type: "RESIN", //STRING
   id: 10, //ID
   color: "GRAY", //STRING
```

```
    density: 1.1, //DOUBLE
    price: 1113.54, //DOUBLE
    printSpeed: 14.75, //DOUBLE
  };
```

Secenekler

```
  let selections = {
    infill: 15, //INT
    wallCount: 2  //INT
  };
```

Giderler

```
let expenses = {
  price: 50, //DOUBLE
  kWPrice: 1.5, //DOUBLE
  tax: 1.2, //DOUBLE
};
```