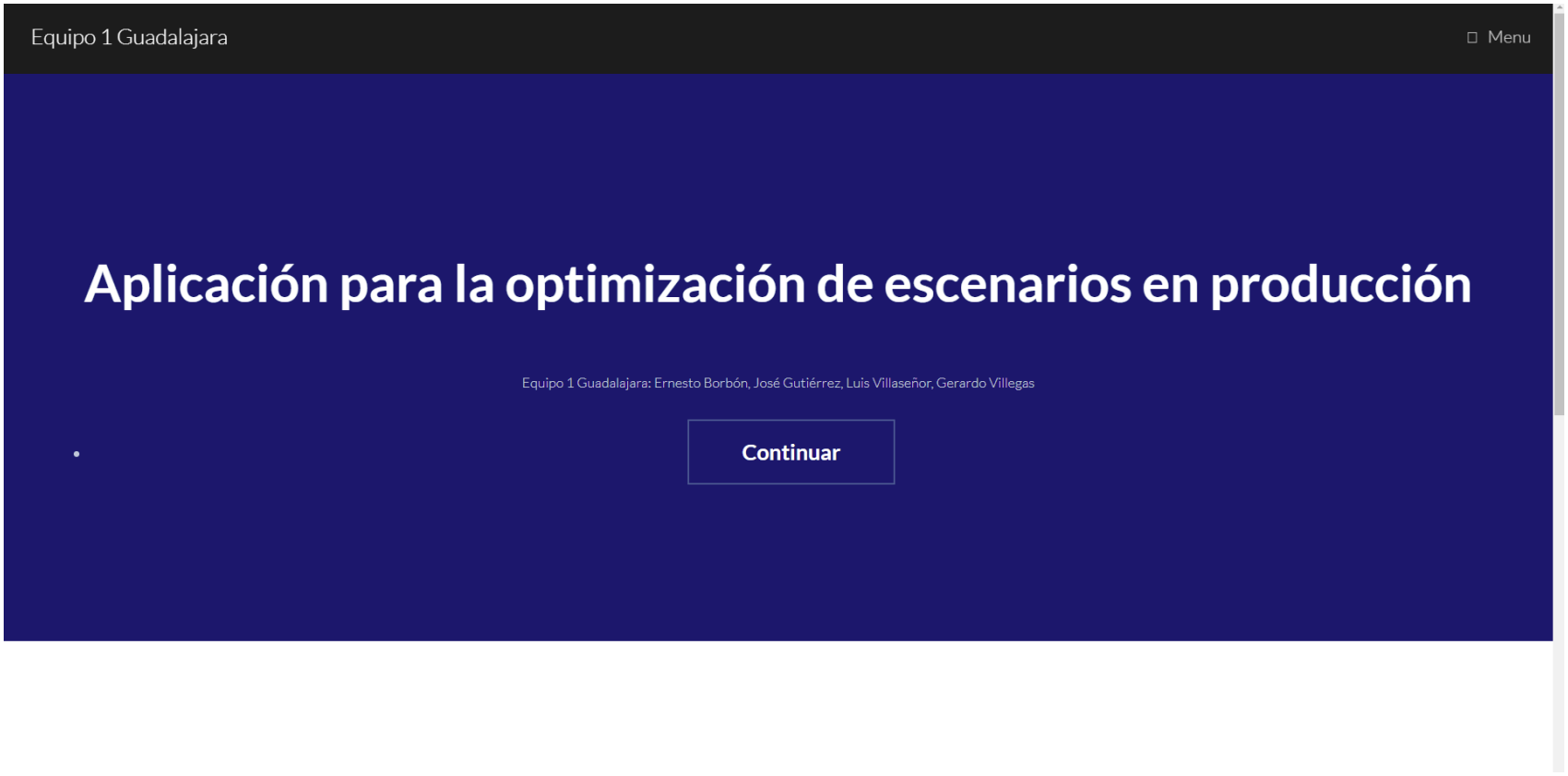
Ernesto Ignacio Bombón Martínez A01701515
José de Jesús Gutiérrez Aldrete A01637812
Luis Felipe Villaseñor Navarrete A01023976
Gerardo Villegas Contreras A00571388
10-03-2021

Aplicación Web en Flask

Liga: https://e1-cemex-app.nn.r.appspot.com

Capturas de pantalla inicial:



Modelo Predictivo

Calidad

0.5

Dureza

100

Tasa de Producción

400

Submit

Capturas de pantalla resultados predicción:

Valores Óptimos

Para una **Calidad** de: 0.5

Dureza de: 100.0

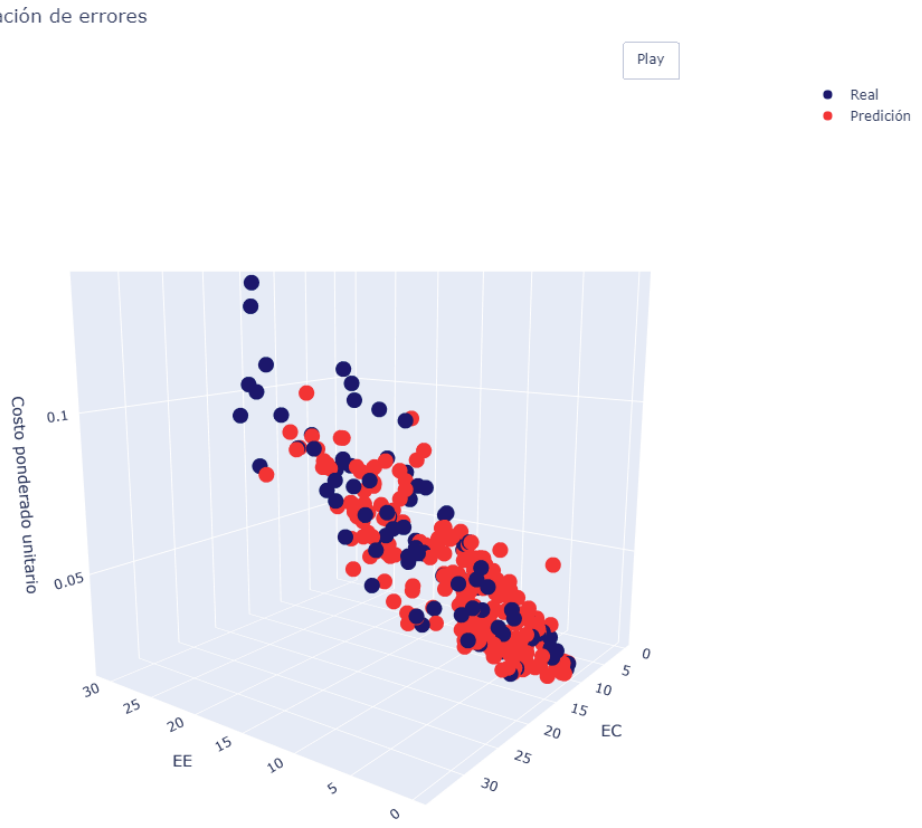
Tasa de Producción de: 400.0

Los valores óptimos son:

EE: 25.21

EC: 21.796

Costo Ponderado Unitario: 0.0807



Video Explicativo

Aquí va el video Explicativo

Home

Dashboard

Valores Óptimos

Para una **Calidad** de: 0.5

Dureza de: 100.0

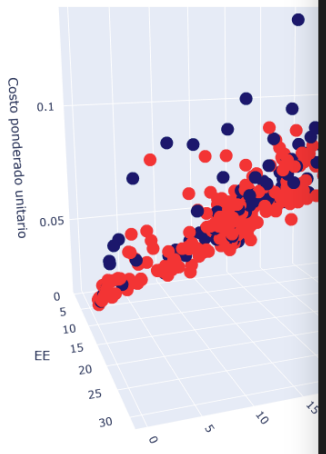
Tasa de Producción de: 400.0

Los valores óptimos son:

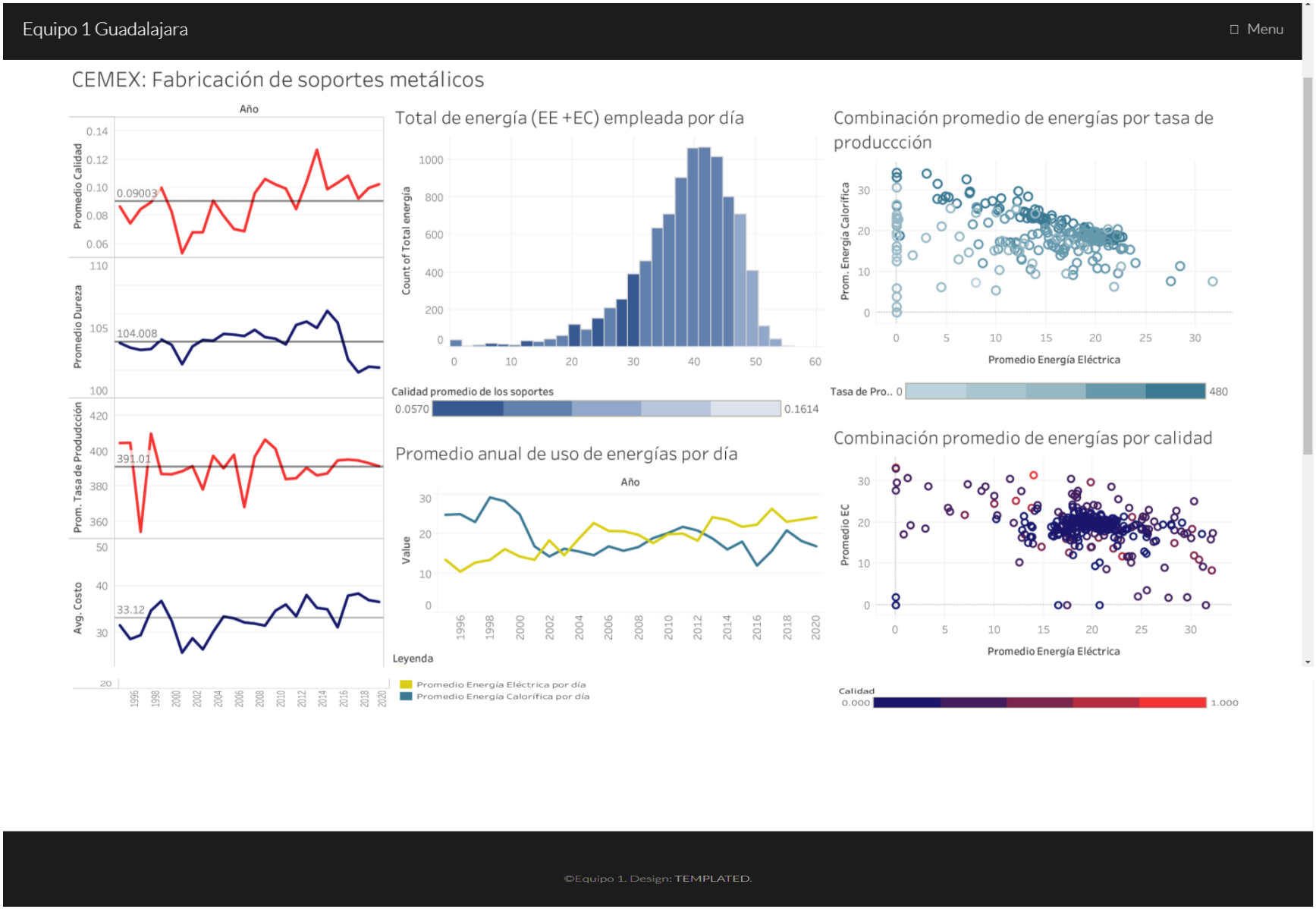
EE: 25.21

EC: 21.796

Costo Ponderado Unitario: 0.0807



Capturas de pantalla Dashboard:



Código main.py:

```
from flask import Flask
from flask import Flask, request, render_template
import pickle5 as pickle
import numpy as np
import os
from sklearn.ensemble import RandomForestRegressor
from sklearn.multioutput import MultiOutputRegressor
import RegresorClass
from RegresorClass import Regresor

#TEMPLATE = '/templates'
#STATIC = '/static'

class CustomUnpickler(pickle.Unpickler):

    def find_class(self, module, name):
        try:
            return super().find_class(__name__, name)
        except AttributeError:
            return super().find_class(module, name)

#, template_folder=TEMPLATE, static_folder=STATIC
app = Flask(__name__)
ROOT=os.path.dirname(os.path.abspath("ObjectFile.picl"))

MODEL_PATH = os.path.join(ROOT, "ObjectFile.picl")
# set path to the model
#model = pickle.load(open(MODEL_PATH, 'rb'))
```

```
model = CustomUnpickler(open("ObjectFile.picl", 'rb')).load()
# load the pickled model

@app.route("/", methods=['GET', 'POST'])
def index():
     return render_template('index.html')

@app.route("/dashboard", methods=['GET', 'POST'])
def dashboard():
     return render_template('dashboard.html')

@app.route('/submit', methods=['GET', 'POST'])
def make_prediction():
  features = [float(x) for x in request.form.values()]
  final_features = [features]
  prediction = model.calculaOptimo(final_features)
  pee=round(prediction[0][0],4)
  pec=round(prediction[0][1],4)
  pcpu=round(prediction[0][2],4)
  return render_template('predictionpro.html',
pee=pee,pec=pec,pcpu=pcpu,c=final_features[0][0],d=final_features[0][1],tp=final_features[0][2])

if __name__ == '__main__':
     app.run(host="127.0.0.1",port=8080)
```

Código RegresorClass.py:

```
import numpy as np
class Regresor:
    def __init__(self, regresor):
        self.regressor = regresor

    def calculaOptimo(self, datos_entrada):
        datos = np.array(datos_entrada)
        datos = datos.reshape(1, -1)
        datos_salida = self.regressor.predict(datos)
        return datos_salida

    def imprimeOptimo(self, datos_entrada):
        datos = np.array(datos_entrada)
        datos = datos.reshape(1, -1)
        EE = self.regressor.predict(datos)[0][0]
        EC = self.regressor.predict(datos)[0][1]
        cost = self.regressor.predict(datos)[0][2]
        print('EE: %s\nEC: %s\nCosto Ponderado unitario: %s'% (EE,EC,cost))
```