



Trabajo de Fin De Máster

# RecoFood

Silvia Donaire Serrano  
Elena Racero González  
Manuel Fajardo Jiménez

Máster de Inteligencia Artificial y Big Data  
C.P.I.F.P - Alan Turing

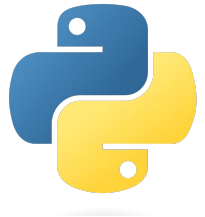
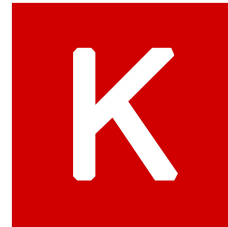
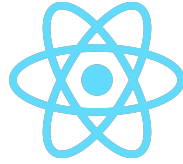


# ¿Qué es RecoFood?

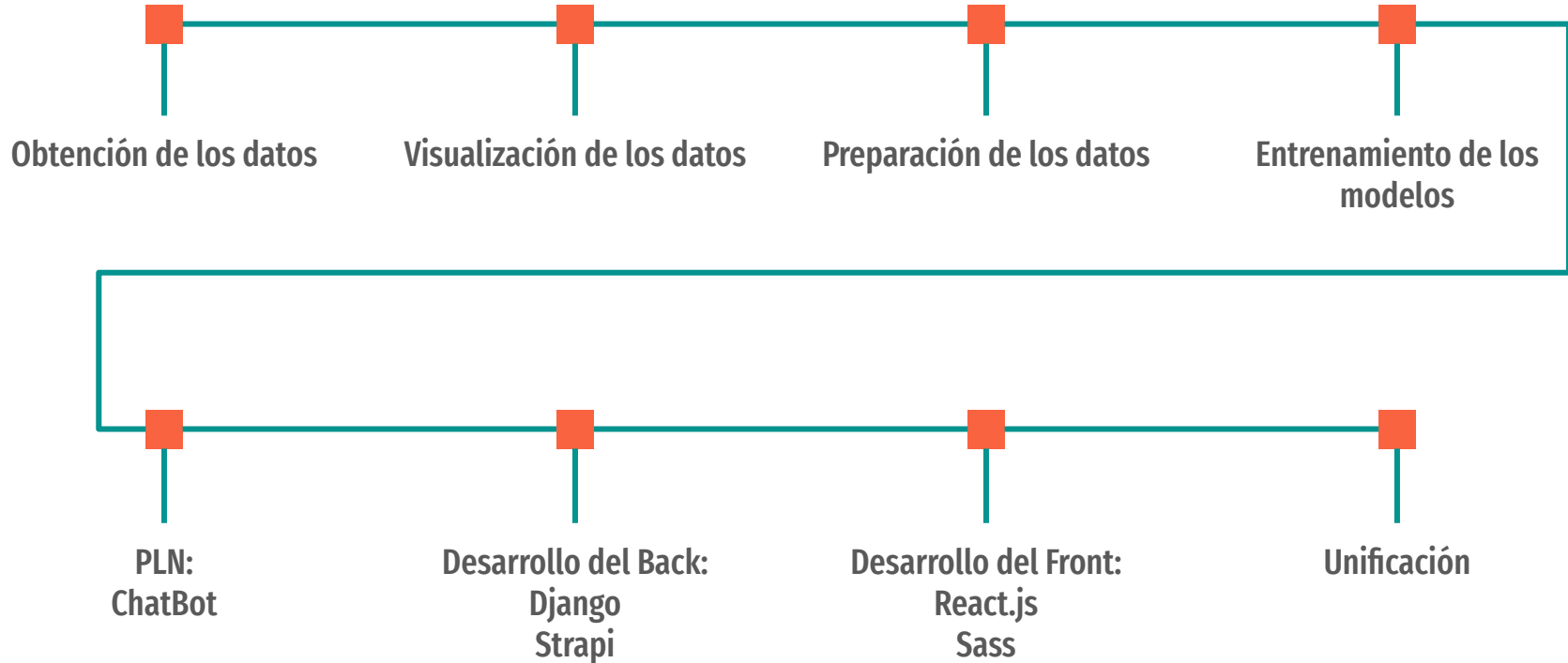
Es una genialidad creativa, una app desarrollada por nosotros para reconocer platos culinarios mediante imagen, así como obtención de recetas y aporte calórico para el usuario.



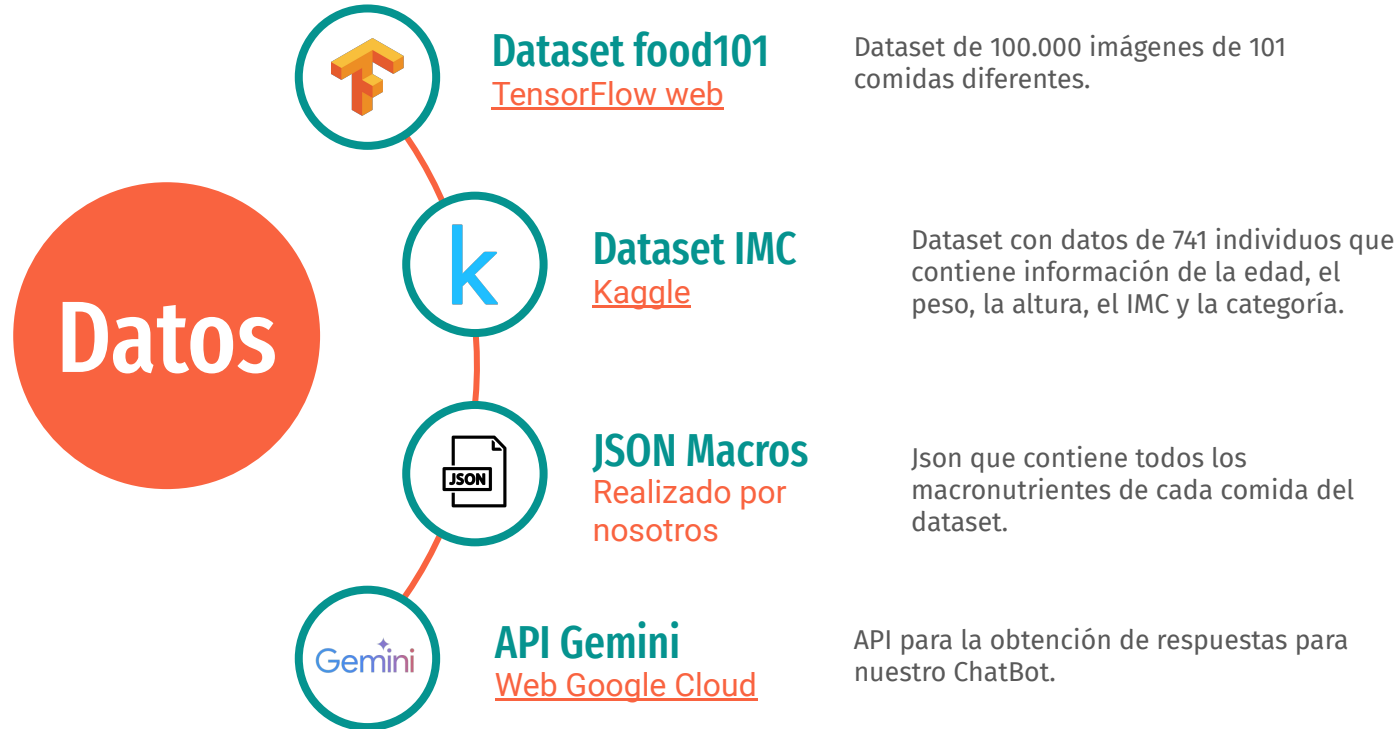
# TECNOLOGÍAS



# Desarrollo del proyecto



# Obtención de datos



# JSON

Carbohidratos

Por cada 100 gr

Calorias

Por cada 100 gr

Proteinas

Por cada 100 gr

Grasas

Por cada 100 gr

## MACRONUTRIENTES



# Scrapping de Imagenes

<https://www.recetasgratis.net/receta-de-pizza-de-pepperoni-74152.html>

<https://www.recetasgratis.net/receta-de-tacos-rancheros-76180.html>

<https://www.recetasgratis.net/receta-de-churros-mexicanos-75726.html>

BeautifulSoup

Urllib

Exclusión de  
Imágenes  
determinadas



# Modelo de clasificación de imágenes





# Visualización de los datos

Imagen aleatoria de cada comida del dataset.



# Preparación de los datos

Generalización de los datos.



```
1  train_datagen = ImageDataGenerator(  
2      rescale=1./255,  
3      shear_range=0.2,  
4      zoom_range=0.2,  
5      horizontal_flip=True,  
6      rotation_range=40,  
7      brightness_range=[0.5, 1.5],  
8      width_shift_range=0.2,  
9      height_shift_range=0.2,  
10     validation_split=0.25)  
11  
12  test_datagen = ImageDataGenerator(rescale=1./255, validation_split=0.25)
```



# Preparación de los datos

Creación de los conjuntos de entrenamiento y validación con la generalización.



```
1 train_generator = train_datagen.flow_from_directory(  
2     base_dir,  
3     target_size=(img_height, img_width),  
4     batch_size=batch_size,  
5     class_mode='categorical',  
6     subset = "training")  
7  
8 validation_generator = test_datagen.flow_from_directory(  
9     base_dir,  
10    target_size=(img_height, img_width),  
11    batch_size=batch_size,  
12    class_mode='categorical',  
13    subset = "validation")
```



# Creación del modelo

Arquitectura preentrenada InceptionV3.

```
1 inception = InceptionV3(  
2     weights='imagenet',  
3     include_top=False,  
4     input_shape=(img_height, img_width, 3))  
5  
6 x_inception = inception.output  
7 x_inception = GlobalAveragePooling2D()(x_inception)  
8  
9 predictions_inception = Dense(  
10     len(class_names),  
11     kernel_regularizer=l2(0.005),  
12     activation='softmax')(x_inception)  
13  
14 model_inception = Model(inputs=inception.input, outputs=predictions_inception)
```



# Compilación del modelo

Optimizador SGD y función de pérdida “categorical\_crossentropy”.



```
1 model_inception.compile(  
2     optimizer=SGD(learning_rate=0.0001, momentum=0.9),  
3     loss='categorical_crossentropy',  
4     metrics=['accuracy'])
```



# Creación de Callbacks

EarlyStopping para detener el modelo si no hay mejoras en la métrica durante el entrenamiento y ModelCheckpoint para guardar el modelo en cada iteración.



```
1 early_stopper_inception = EarlyStopping(  
2     monitor='val_loss',  
3     patience=10,  
4     restore_best_weights=True)  
5 checkpointer_inception = ModelCheckpoint(  
6     filepath='Modelos entrenados/best_model_trained.h5',  
7     verbose=1,  
8     save_best_only=True)
```



# Entrenamiento del modelo

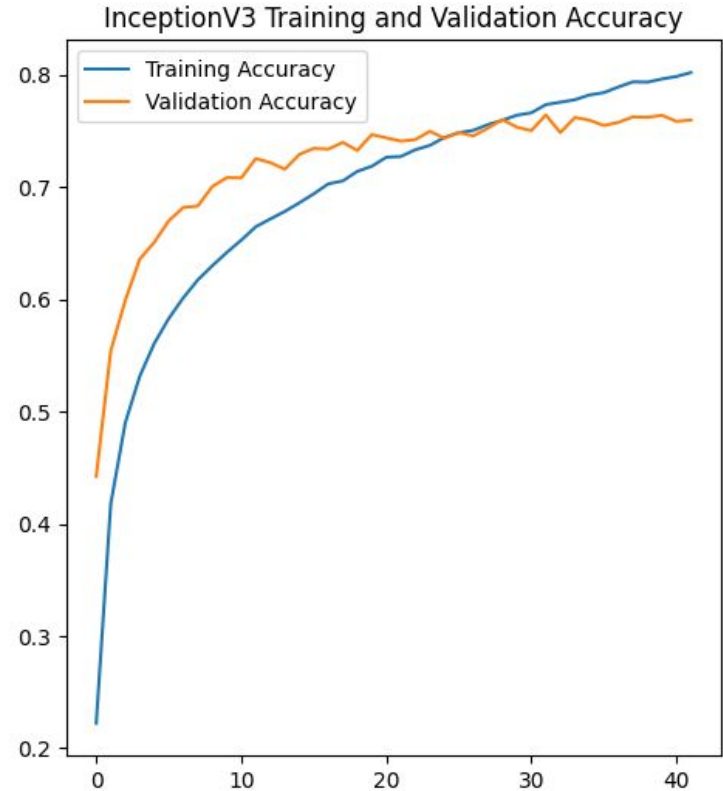
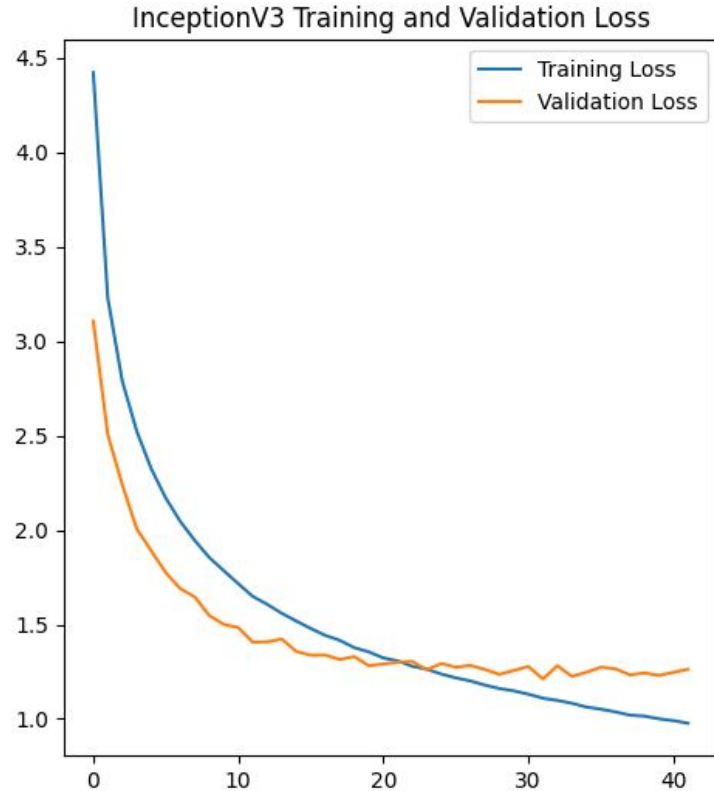
Utilización del conjunto de datos de entrenamiento y validación para entrenar el modelo.



```
1 history = model_inception.fit(train_generator,  
2                               steps_per_epoch = len(train_generator),  
3                               validation_data = validation_generator,  
4                               validation_steps = len(validation_generator),  
5                               epochs = epoch,  
6                               verbose = 1,  
7                               callbacks = [early_stopper_inception, checkpointer_inception])
```



# Métricas del rendimiento





# Funcionamiento del modelo de clasificación de imágenes

Proceso de análisis



# Modelo predictivo del IMC



# Preparación de los datos



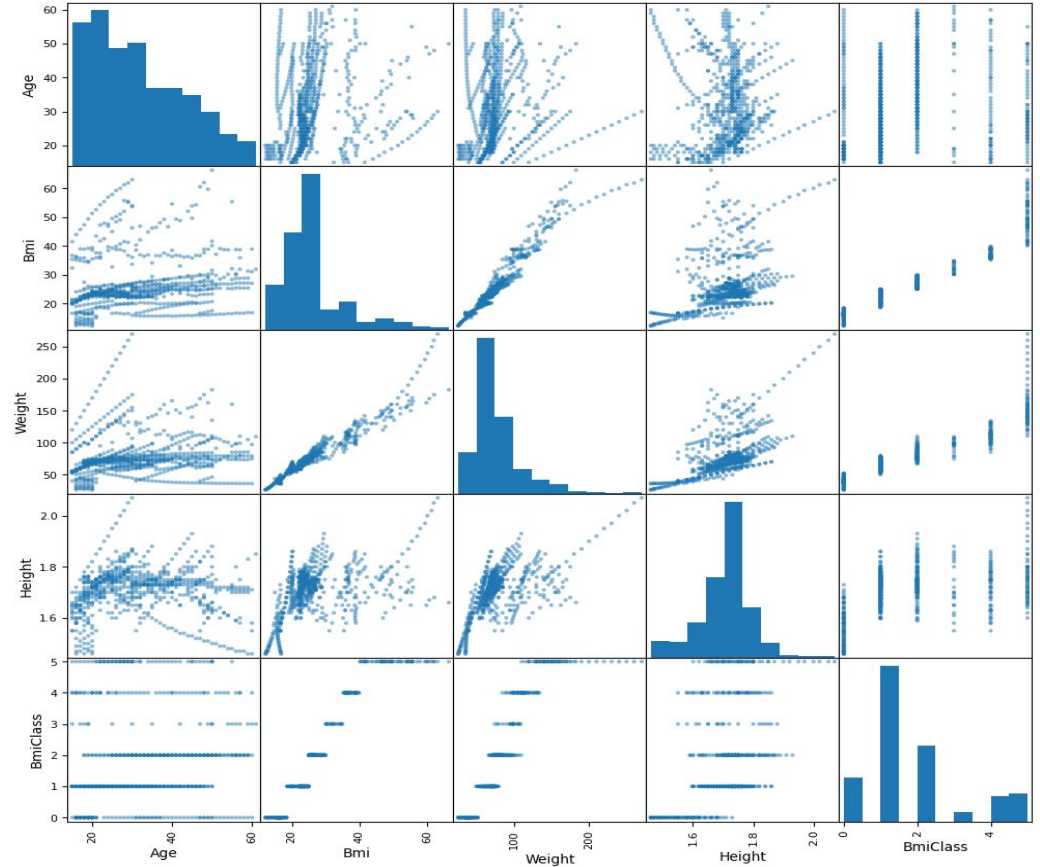
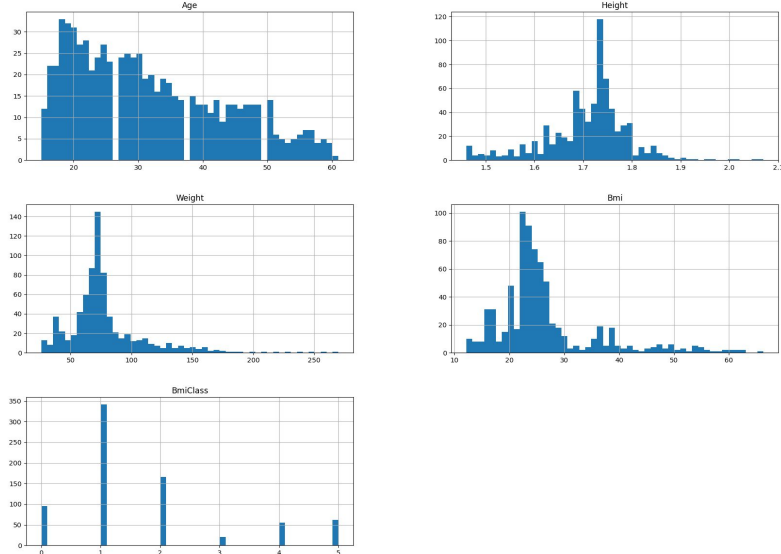
```
1 import numpy as np
2 import pandas as pd
3 import matplotlib.pyplot as plt
4 import seaborn as sns
```



```
1 file_path = '/content/drive/MyD
  rive/Dataset/bmi.csv'
2 bmidata = pd.read_csv(file_pat
  h)
```



# Gráficos y Correlaciones



# Entrenamiento y Random Forest



```
1 from sklearn.model_selection
  import train_test_split
2
3 X_train, X_test, y_train, y_
  test = train_test_split(X,
    y, test_size=0.20)
```



```
1 from sklearn.ensemble import RandomForestRegressor
2
3 bmidata_model_v3 = RandomForestRegressor()
4
5 # Entrenamiento del modelo
6 bmidata_model_v3.fit(X_train, y_train)
7
8 # Predicción
9 y_pred = bmidata_model_v3.predict(X_test)
10
11 y_test_list = y_test.tolist()
12
13 print("Prec. real    Prec. estimado    Error absoluto    Porcent
    aje de acierto")
```



# Predicción Obesidad (BMI)

**Underweight**

Random Forest

**Normal Weight**

Random Forest

**Overweight**

Random Forest

**Obese Class(1, 2, 3)**

Random Forest

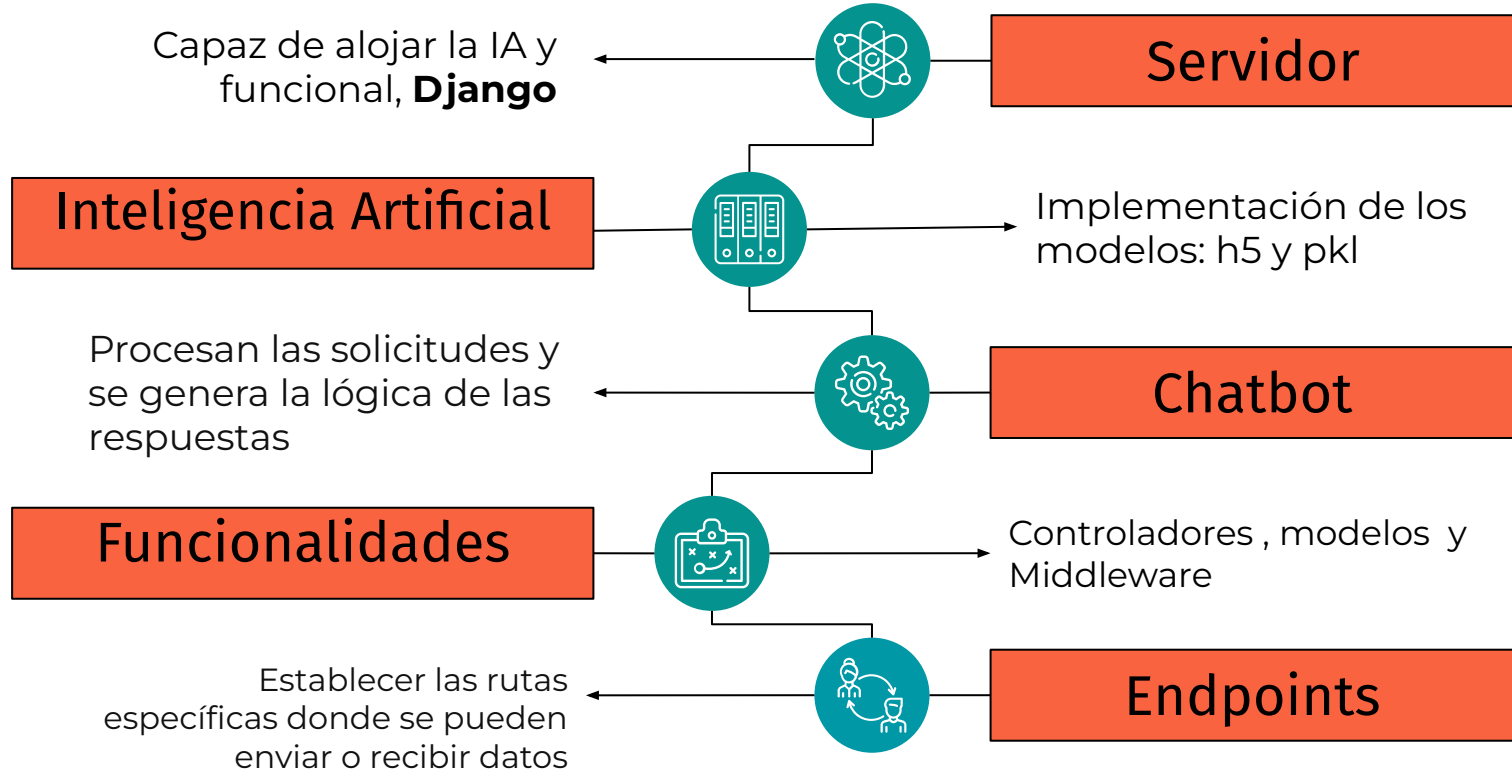


# Funcionamiento del modelo predictivo del IMC

## Proceso de análisis

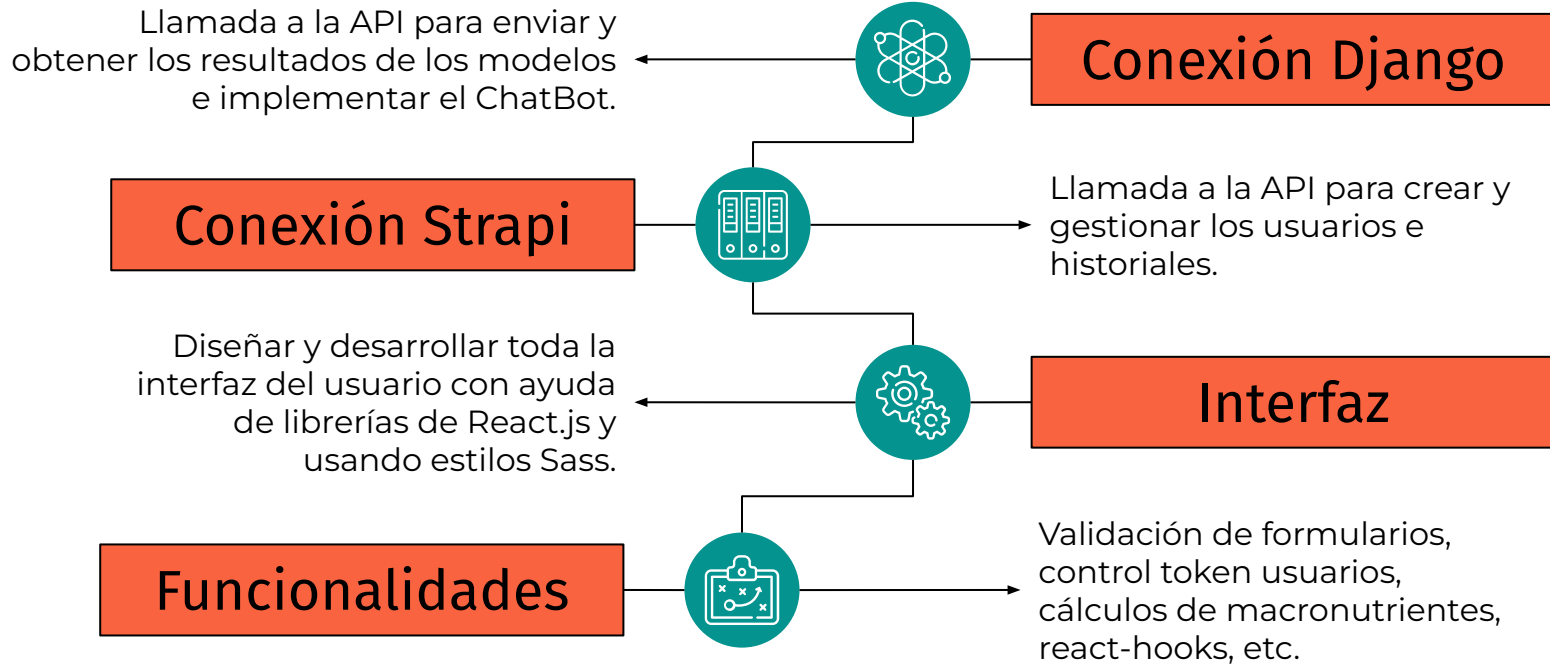


# Backend





# Frontend en React.js





**Realizado por**

**Silvia Donaire Serrano**  
**Elena Racero González**  
**Manuel Fajardo Jiménez**

Máster de Inteligencia Artificial y Big Data  
C.P.I.F.P - Alan Turing



2023/2024