

CARDIONET 2.0

Juan Pablo Sierra Rios
Jesus David Gamboa Badel

CardioNet 2.0 es una versión avanzada de **CardioNet 1.0**, que presenta una serie de mejoras y nuevas funcionalidades, junto con una interfaz de usuario optimizada. Esta actualización mantiene el objetivo central de la aplicación: proporcionar una plataforma web robusta para desplegar un modelo de aprendizaje automático que predice enfermedades cardíacas basado en exámenes y datos médicos del paciente.

Características Principales:

- **Predicción en el Navegador:** La nueva versión de CardioNet permite realizar predicciones directamente en el navegador mediante un modelo de red neuronal entrenado con TensorFlow. Esto elimina la necesidad de servidores externos, ofreciendo una experiencia más ágil y rápida para los usuarios.
- **Interfaz Mejorada:** La interfaz de usuario ha sido rediseñada para ser más intuitiva y amigable, facilitando la navegación y la interacción con la aplicación.
- **Registro y Gestión de Usuarios:** A diferencia de la versión anterior, CardioNet 2.0 incluye un sistema completo de registro y gestión de usuarios, con protección de contraseñas para garantizar la seguridad de la información.
- **Generación de Códigos QR:** Los pacientes pueden recibir códigos QR únicos que facilitan la identificación y recuperación de sus datos médicos en la plataforma.
- **Acceso a Datos y Predicciones:** Los usuarios pueden ingresar datos médicos para recibir predicciones sobre posibles enfermedades cardíacas, así como consultar resultados históricos y tendencias.
- **Integración con Base de Datos:** La aplicación se integra con una base de datos para almacenar y gestionar la información de los pacientes, lo que permite un acceso eficiente y seguro a los datos necesarios para las predicciones.

Beneficios:

- **Accesibilidad y Rapidez:** Al realizar las predicciones en el navegador, los usuarios experimentan tiempos de respuesta más rápidos y no dependen de servidores externos.
- **Seguridad Mejorada:** El sistema de autenticación y la protección de contraseñas aseguran que solo los usuarios autorizados puedan acceder a la información sensible.

- **Interfaz Amigable:** La nueva interfaz facilita la navegación y el uso de la aplicación, haciendo el proceso de gestión de datos y obtención de predicciones más sencillo y eficiente.

Tecnologías Utilizadas:

- **TensorFlow:** Utilizado para entrenar el modelo de red neuronal que realiza las predicciones de enfermedades cardíacas.
- **Flask:** El microframework en Python que sirve como la base para la aplicación web, gestionando las rutas y la integración con la base de datos.
- **HTML/CSS:** Tecnologías empleadas para el diseño y la estructura de la interfaz de usuario.
- **JavaScript:** Para las funcionalidades interactivas y la ejecución de scripts en el navegador.

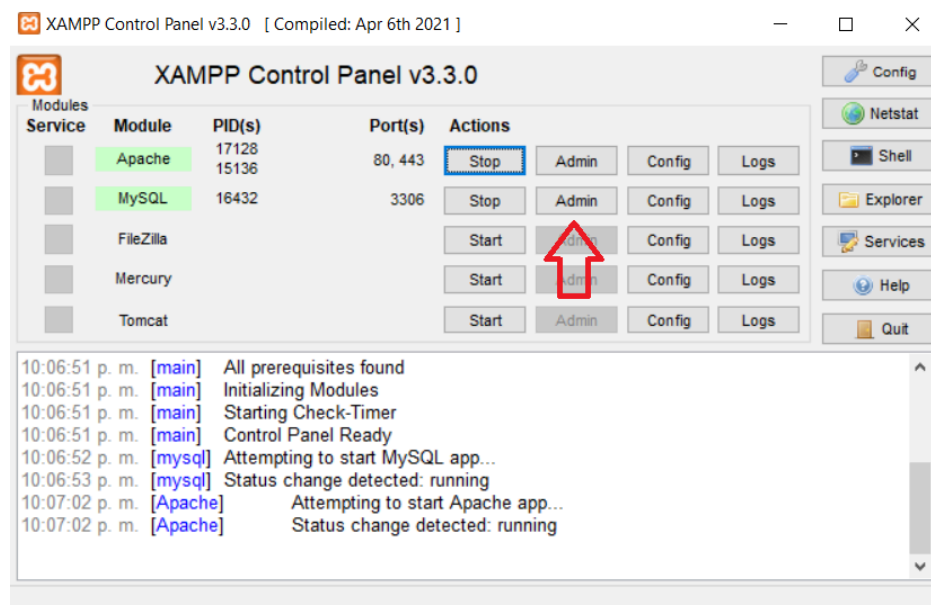
CardioNet 2.0 no solo mejora la funcionalidad y la interfaz de la aplicación, sino que también asegura una experiencia de usuario más fluida y segura, apoyada por las últimas tecnologías en aprendizaje automático y desarrollo web.

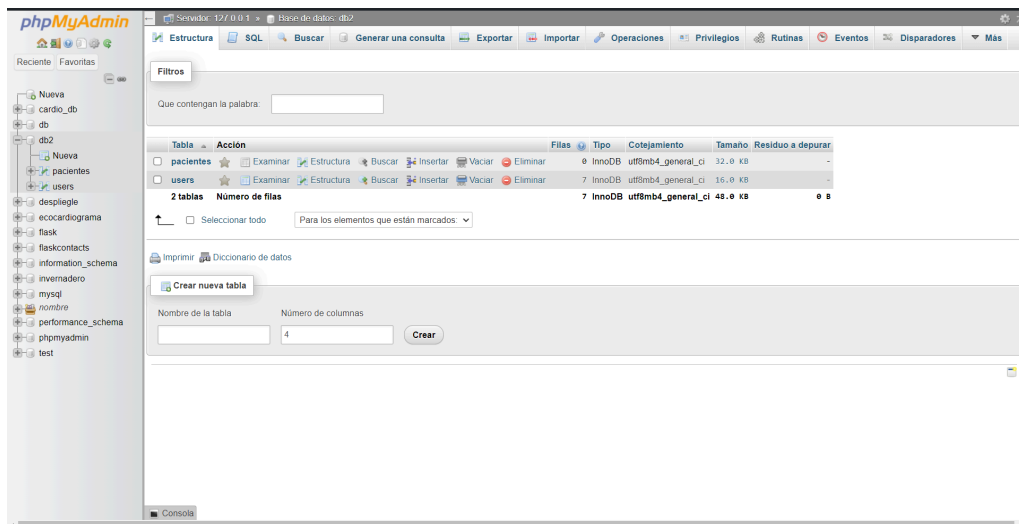
GUIA PARA EL DESPLIEGUE DE MODELOS DE MACHINE LEARNING EN PÁGINA WEB

1. Creación de base de datos en MySQL

Para la creación de base de datos debemos previamente tener instalado xampp con sus servicios, entre estos MySQL.

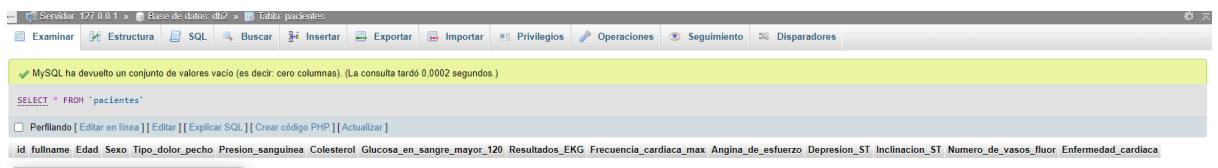
Lo que haremos será ejecutar el panel de control de xampp y activaremos el servicio apache y mysql, luego justo al lado del botón de iniciar servicio se encuentra uno llamado admin, es ahí donde presionaremos y seremos redirigidos al gesto de base de datos.





Crearemos una nueva base de datos, para efectos prácticos será nombrada como “db2”, luego le daremos aceptar.

Una vez creada la base de datos creamos las tablas para pacientes.



Esta tabla debe concordar con el tipo de variables y el orden con las que se entrenó el modelo

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 270 entries, 0 to 269
Data columns (total 13 columns):
#   Column                                     Non-Null Count  Dtype
---  -
0   Edad                                     270 non-null    int64
1   Sexo                                    270 non-null    int64
2   Tipo_dolor_pecho                        270 non-null    int64
3   Presion_sanguinea                       270 non-null    int64
4   Colesterol                              270 non-null    int64
5   Glucosa_en_sangre_mayor_120             270 non-null    int64
6   Resultados_EKG                           270 non-null    int64
7   Frecuencia_cardiaca_max                  270 non-null    int64
8   Angina_de_esfuerzo                       270 non-null    int64
9   Depresion_ST                             270 non-null    float64
10  Inclinacion_ST                           270 non-null    int64
11  Numero_de_vasos_fluor                    270 non-null    int64
12  Enfermedad_cardiaca                       270 non-null    int64
dtypes: float64(1), int64(12)
memory usage: 27.5 KB
```

Creamos las columnas para usuarios quienes serán personal autorizado de usar la plataforma.

☐ Mostrar todo | Número de filas: 25 | Filtrar filas: Buscar en esta tabla | Ordenar según la clave: Ninguna

Opciones extra

← T →

ID nombres apellidos edad correo contra

despues de crear nuestras tablas, nos dirigimos a la barra superio, en privilegios y crearemos un nuevo usuario

Examinar Estructura SQL Buscar Insertar Exportar Importar Privilegios Operaciones Seguimiento Disparadores

Usuarios con acceso a "db2.users"

	Nombre de usuario	Nombre del servidor	Tipo	Privilegios	Conceder	Acción
<input type="checkbox"/>	admin	localhost	global	ALL PRIVILEGES	Sí	
<input type="checkbox"/>	admin3	localhost	global	ALL PRIVILEGES	Sí	
<input type="checkbox"/>	root	127.0.0.1	global	ALL PRIVILEGES	Sí	
<input type="checkbox"/>	root	:::1	global	ALL PRIVILEGES	Sí	
<input type="checkbox"/>	root	localhost	global	ALL PRIVILEGES	Sí	

↑ ☐ Seleccionar todo Para los elementos que están marcados:

Nuevo

llenaremos los campos tal y como aparecen en la imagen.

Bases de datos SQL Estado actual Cuentas de usuarios Exportar Importar Configuración Replicación Variables

Agregar cuenta de usuario

Información de la cuenta

Nombre de usuario: db3

Nombre de Host: Local

Contraseña: Fuerza:

Debe volver a escribir:

plugin de autenticación: Autenticación de MySQL nativo

Generar contraseña:

Base de datos para la cuenta de usuario

☐ Crear base de datos con el mismo nombre y otorgar todos los privilegios.

☐ Otorgar todos los privilegios al nombre que contiene comodín (username)_%.

☒ Otorgar todos los privilegios para la base de datos db2.

Privilegios globales ☒ Seleccionar todo

Nota: Los nombres de los privilegios de MySQL están expresados en inglés.

☒ Datos

- ☒ SELECT
- ☒ INSERT
- ☒ UPDATE
- ☒ DELETE
- ☒ FILE

☒ Estructura

- ☒ CREATE
- ☒ ALTER
- ☒ INDEX
- ☒ DROP
- ☒ CREATE TEMPORARY TABLES
- ☒ SHOW VIEW
- ☒ CREATE ROUTINE
- ☒ ALTER ROUTINE

☒ Administración

- ☒ GRANT
- ☒ SUPER
- ☒ PROCESS
- ☒ RELOAD
- ☒ SHUTDOWN
- ☒ SHOW DATABASES
- ☒ LOCK TABLES
- ☒ REFERENCES

☒ Limites de recursos

Nota: si cambia los parámetros de estas opciones a 0 (cero), remueve el límite.

MAX QUERIES PER HOUR

MAX UPDATES PER HOUR

MAX CONNECTIONS PER HOUR

MAX USER_CONNECTIONS

Consola

2. Conexión base de datos y creación de rutas con Flask

Para conexión a base de datos usaremos el microframework Flask para python y demás dependencias. Para ello creamos la siguiente estructura:

/Nombre del proyecto

|-- app.py

|-- modelo.py

|-- modelo_clas_rf.pkl

|-- static |

| |-- estilo de las plantillas |

|-- templates |

 |-- formularios en html |

en app.py sera donde implementaremos la conexión a la base de datos y demás funciones backend.

```
1  from flask import Flask, render_template, request, redirect, url_for, session, flash
2  from modelo import Random_forest
3  import pandas as pd
4  import mysql.connector
5  import bcrypt
6  import pickle
7  import qrcode
8  import os
9
```

Creamos una función para hacer la conexión a la base de datos y las tablas.

```
app = Flask(__name__)
app.secret_key = 'jkSMKnZkaSPn7nrF' # Cambia esto a una clave secreta segura

# Configuración de La base de datos
db_config = {
    'user': 'admin3', # Reemplaza con tu usuario de MySQL
    'password': '4nxus0A9@.M1R6F(', # Reemplaza con tu contraseña de MySQL
    'host': 'localhost',
    'database': 'DB2' # El nombre de la base de datos que creaste
}

MASTER_KEY = 'CardioNet'
```

Como vemos, usamos el usuario que creamos en MySQL con su respectiva contraseña, especificamos el tipo de base de datos y el host, todo similar a como creamos la base de datos.

```

25 <section class="form">
26 <form action="/for_pacientes" method="post">
27 <label for="Fullname">Nombre Completo:</label>
28 <input class="controls" type="text" name="Fullname" id="Fullname" placeholder="Ingrese su nombre completo" required><br><br>
29
30 <label for="Edad">Edad:</label>
31 <input class="controls" type="number" id="Edad" name="Edad" min="0"><br><br>
32
33 <label for="masculino">Masculino:</label>
34 <input class="controls" type="radio" id="masculino" name="Sexo" value="1" required>
35
36 <label for="femenino">Femenino:</label>
37 <input class="controls" type="radio" id="femenino" name="Sexo" value="0" required><br><br>
38
39 <label for="Tipo_dolor_pecho">Tipo de angina:</label>
40 <select id="Tipo_dolor_pecho" name="Tipo_dolor_pecho" required>
41 <option value="1">Clase I</option>
42 <option value="2">Clase II</option>
43 <option value="3">Clase III</option>
44 <option value="4">Clase IV</option>
45 </select><br><br>
46
47 <label for="Presion_sanguinea">Presión Sanguínea:</label>
48 <input class="controls" type="number" id="Presion_sanguinea" name="Presion_sanguinea" min="0" placeholder="mmHg" required><br><br>
49
50 <label for="Colesterol">Colesterol:</label>
51 <input class="controls" type="number" id="Colesterol" name="Colesterol" min="0" placeholder="mg/dL" required><br><br>
52
53 <label for="Glucosa_en_sangre_mayor_120">Glucosa en Sangre > 120 mg/dL:</label>
54 <select id="Glucosa_en_sangre_mayor_120" name="Glucosa_en_sangre_mayor_120" required>
55 <option value="1">Si</option>

```

```

114 @app.route('/for_pacientes', methods=['GET', 'POST'])
115 def for_pacientes():
116     if 'loggedin' in session:
117         if request.method == 'POST':
118             Fullname = request.form['Fullname']
119             Edad = int(request.form['Edad'])
120             Sexo = int(request.form['Sexo'])
121             Tipo_dolor_pecho = int(request.form['Tipo_dolor_pecho'])
122             Presion_sanguinea = int(request.form['Presion_sanguinea'])
123             Colesterol = int(request.form['Colesterol'])
124             Glucosa_en_sangre_mayor_120 = int(request.form['Glucosa_en_sangre_mayor_120'])
125             Resultados_EKG = int(request.form['Resultados_EKG'])
126             Frecuencia_cardiaca_max = int(request.form['Frecuencia_cardiaca_max'])
127             Angina_de_esfuerzo = int(request.form['Angina_de_esfuerzo'])
128             Depresion_ST = float(request.form['Depresion_ST'])
129             Inclination_ST = int(request.form['Inclinacion_ST'])
130             Numero_de_vasos_fluor = int(request.form['Numero_de_vasos_fluor'])
131
132     # Realizar predicciones

```

En el formulario de html que hemos creado por cada casilla anotamos las mismas tablas que en MySQL, posterior a esto, dentro de la ruta creada en app.py ingresamos las variables que entrarán a las columnas de la tabla de pacientes.

```

131 # Realizar predicciones
132 Enfermedad_cardiaca = Random_forest(Edad, Sexo, Tipo_dolor_pecho, Presion_sanguinea, Colesterol, Glucosa_en_sangre_mayor_120, Resultados_EKG, Frecuencia_cardiaca_max, Angina_de_esfuerzo)
133 Enfermedad_cardiaca = int(Enfermedad_cardiaca)
134
135 # Conexión a la base de datos
136 conn = mysql.connector.connect(**db_config)
137 cursor = conn.cursor()
138
139 # Insertar datos del paciente en la base de datos
140 query = "INSERT INTO pacientes (Fullname, Edad, Sexo, Tipo_dolor_pecho, Presion_sanguinea, Colesterol, Glucosa_en_sangre_mayor_120, Resultados_EKG, Frecuencia_cardiaca_max, Angina_de_esfuerzo, Depresion_ST, Inclination_ST, Numero_de_vasos_fluor) VALUES (%s, %s, %s, %s, %s, %s, %s, %s, %s, %s, %s, %s, %s)"
141 cursor.execute(query, (Fullname, Edad, Sexo, Tipo_dolor_pecho, Presion_sanguinea, Colesterol, Glucosa_en_sangre_mayor_120, Resultados_EKG, Frecuencia_cardiaca_max, Angina_de_esfuerzo, Depresion_ST, Inclination_ST, Numero_de_vasos_fluor))
142
143 # Obtener el ID del paciente insertado
144 paciente_id = cursor.lastrowid
145
146 conn.commit()
147 cursor.close()
148 conn.close()

```

Como vemos, esas variables entran a la función del modelo proveniente del archivo modelo.py y nos retornaran la predicción, dicha predicción se hace antes de insertar los datos del paciente en la base de datos para asegurar el envío correcto de los datos.

3. Implementación del modelo de machine learning

en el archivo modelo.py creamos una función donde entrarán las variables de los formularios, el modelo hará la predicción y la retornará.

```
1 import pandas as pd
2 import pickle
3
4 def Random_forest(Edad, Sexo, Tipo_dolor_pecho, Presion_sanguinea, Colesterol, Glucosa_en_sangre_mayor_120, Resultados_EKG, Frecuencia_cardiaca_max, Angina_de_esfuerzo, Depresion_ST,
5                  Numero_de_vasos_fluor):
6     # Crear el diccionario con nombres de columnas
7     datos = {
8         'Edad': [Edad],
9         'Sexo': [Sexo],
10        'Tipo_dolor_pecho': [Tipo_dolor_pecho],
11        'Presion_sanguinea': [Presion_sanguinea],
12        'Colesterol': [Colesterol],
13        'Glucosa_en_sangre_mayor_120': [Glucosa_en_sangre_mayor_120],
14        'Resultados_EKG': [Resultados_EKG],
15        'Frecuencia_cardiaca_max': [Frecuencia_cardiaca_max],
16        'Angina_de_esfuerzo': [Angina_de_esfuerzo],
17        'Depresion_ST': [Depresion_ST],
18        'Inclinacion_ST': [Inclinacion_ST],
19        'Numero_de_vasos_fluor': [Numero_de_vasos_fluor]
20    }
21
22    # Crear DataFrame a partir del diccionario
23    df = pd.DataFrame(datos)
24
25    # Cargar el modelo y objetos necesarios
26    filename = 'modelo_clas_rf.pkl'
27    modelTree, Labelencoder, variables = pickle.load(open(filename, 'rb'))
28
29    # Realizar predicciones
30    Y_fut = modelTree.predict(df[variables])
31
32    #print("Resultados de predicción:")
33    #print(Y_fut)
34    return Y_fut[0]
```

El orden debe estar en el mismo con el que se entrenó como método de prevención de errores.

El crear una función en otro archivo nos asegura una legibilidad más sencilla del código en cuestión.

Para más detalle del código es necesario visitar el siguiente enlace a github donde se encontrará disponible para su clonación.