[My Personal Match](http://www.mypersonalmatch.somee.com/)

Colegio Santísima Trinidad de Salamanca

Débora R. Martins de Jesus

Proyecto Integrado (PI)

Tutor: Rafael Pérez Corro

Salamanca, 13 Noviembre de 2023

**Índice**

[**1.** **Introducción** 2](#_Toc150096599)

[**1.1** **Objetivo** 2](#_Toc150096600)

[**1.2** **Justificación** 2](#_Toc150096601)

[**1.3** **Metodología** 3](#_Toc150096602)

[**2.** **Modelado y Planificación** 4](#_Toc150096603)

[**2.1** **Arquitectura BackEnd** 4](#_Toc150096604)

[**2.2** **Machine Learning** 8](#_Toc150096605)

[**2.3 Base de datos** 12](#_Toc150096606)

[**2.4 Arquitectura FrontEnd** 14](#_Toc150096607)

# **Introducción**

## **Objetivo**

El objetivo principal de este proyecto es crear una herramienta basada en Machine Learning, que ayude a personas comunes a utilizar una herramienta con alta tecnología para predecir, con base en un historial, si una persona que esta conociendo es su pareja ideal o no. Es importante resaltar que la idea de la aplicación no es sustituir el criterio humano, sino ofrecer un apoyo a la hora de tomar decisiones informadas.

## **Justificación**

En la era digital en que vivimos, las relaciones son cada vez más superficiales, principalmente debido a las aplicaciones de cita online, que se han convertido en una forma común de conocer personas y potenciales parejas. Sin embargo, los usuarios tienen cada vez menos tiempo y las abundancias de opciones de aplicaciones genera un desafío significativo para aquellos que buscan relaciones significativas y sin perdidas de tiempo. En este contexto, se justifica la creación de una herramienta basada en Machine Learning para predecir la compatibilidad personal y ayudar las personas a tomar decisiones informadas a la hora de elegir que personas se conectar, también es importante entender que la IA es un excelente instrumento para apoyar los humanos diariamente.

**Pilares importantes para este proyecto:**

* **Necesidad de Personalización:** Debido a los avances tecnológicos, existe una mayor diversidad de intereses, preferencias y valores en la sociedad actual, principalmente entre los jóvenes, que a cada día descubren nuevas tendencias y modismos a que seguir. La herramienta que se propone aborda esta necesidad al analizar patrones de datos previamente dados por el usuario, lo que permite ofrecer recomendaciones precisas y relevantes.
* **Aplicación del Machine Learning:** La aplicación de Machine Learning a la tarea de predecir si una persona es ideal o no para los gustos personales de un determinado individuo, es una oportunidad de intersección de la tecnología a relaciones personales. Este enfoque, permite el análisis de datos complejo y la identificación de patrones que no son tan evidente para un sistema convencional.
* **Impacto Social:** Esta herramienta tiene un potencial impacto social significativo al ayudar a las personas a encontrar parejas que compartan los mismos gustos, intereses y valores. Esto puede contribuir a relaciones más saludables y duraderas, así como reducir las frustraciones a largo plazo en los términos de citas online.
* **Contribución al Conocimiento:** Más allá de las relaciones personales, este proyecto busca desmitificar el uso de la IA en la rutina de los seres humanos, además la investigación ayudará a comprender mejor la eficacia y desafíos con este tipo de tecnología.

## **Metodología**

Diagrama

Descripción generada automáticamente

Ilustración 1 - Metodología

# **Modelado y Planificación**

Este proyecto cuenta con un frontend, backend, sistema de control de versiones, una base de datos, librerías especificas, dos hostings de web, entre otros recursos softwares.

## **Arquitectura BackEnd**

Parte del backend de este proyecto esta codificado en el lenguaje C# utilizando Entity Framework .Net Core 7. Abajo se explica un poco como esta estructurado el código:

* **Patrón Repositorio & Unidad de Trabajo:**

Un repositorio no es más que una clase definida para una entidad, con todas las operaciones posibles sobre esa entidad específica.

Microsoft (2010) describió:

Utilice un repositorio para separar la lógica que recupera los datos y los asigna al modelo de entidad de la lógica empresarial que actúa sobre el modelo. La lógica empresarial debe ser independiente del tipo de datos que componen la capa de origen de datos. Por ejemplo, la capa de origen de datos puede ser una base de datos, una lista de SharePoint o un servicio web.

La separación entre los niveles de datos y de negocio tiene tres beneficios:

* Centraliza la lógica de datos o lógica de acceso a servicios web.
* Proporciona un punto de sustitución para las pruebas unitarias.
* Proporciona una arquitectura flexible que se puede adaptar a medida que evoluciona el diseño general de la aplicación. (párr. 3)

El patrón de repositorio es la abstracción de la idea de almacenamiento persistente, el patrón de Unidad de Trabajo (Unit of Work) es la abstracción de la idea de operaciones atómicas. Se hace referencia a UoW como una transacción única que involucra múltiples operaciones de inserción, actualización, eliminación, etc.

El patrón UoW gestiona los estados de la base de datos. Una vez que se completan todas las actualizaciones de las entidades en un alcance, los cambios rastreados se replican en la base de datos en una transacción para que la base de datos refleje los cambios deseados.

Las transacciones le permiten procesar varias operaciones de bases de datos de forma atómica. Si la transacción se confirma, todas las operaciones se aplican correctamente a la base de datos. Si la transacción se revierte, ninguna de las operaciones se aplica a la base de datos.

De forma predeterminada, si el proveedor de la base de datos admite transacciones, todos los cambios realizados en una llamada a *SaveChanges* se aplican a una transacción. Si alguno de los cambios falla, la transacción se revertirá y ninguno de los cambios se aplicará a la base de datos. Esto significa que se garantiza que *SaveChanges* se completará correctamente o dejará la base de datos sin cambios si se produce un error.

Captura de pantalla de computadora

Descripción generada automáticamente

Ilustración - Unit Of Work: Clase e Interface

* Entity Framework Repository, que permite trabajar con una base de datos por medio de objetos de tipo DBContext.

Texto

Descripción generada automáticamente

Ilustración 3 - EF Repositorio

La clase DbContext es una parte integral de Entity Framework. Una instancia de DbContext representa una sesión con la base de datos que se puede utilizar para consultar y guardar instancias de sus entidades en una base de datos. DbContext es una combinación de los patrones Unidad de trabajo y Repositorio. DbContext en Entity Framework Core nos permite realizar las siguientes tareas:

* Administrar la conexión de la base de datos
* Configurar modelo y relación.
* Consultando base de datos.
* Guardar datos en la base de datos.
* Interface del repositorio genérica, que define que métodos tendrán los repositorios de entidades especificas:

Texto

Descripción generada automáticamente

Ilustración - Interface Repositorio Genérico

## **Machine Learning**

El Machine Learning es un subcampo de la IA que se centra en desarrollar algoritmos y técnicas que permiten a las computadoras aprender y mejorar su rendimiento en tareas específicas a través de la experiencia y la observación de datos. En lugar de ser programados explícitamente para realizar una tarea, los sistemas de Machine Learning utilizan datos para aprender patrones y tomar decisiones autónomas. Existen diferentes tipos de aprendizaje automático, incluyendo el aprendizaje supervisado, en el que el modelo se entrena con ejemplos etiquetados, y el aprendizaje no supervisado, en el que el modelo encuentra patrones en los datos sin etiquetas. En este proyecto, se trabaja con el aprendizaje automático supervisado, ya que los datos fueron dados por el usuario previamente y los parámetros etiquetados.

* **NuGet ML.NET**

Según Microsoft (2022)

Un paquete NuGet es un único archivo ZIP con la extensión .nupkg que contiene código compilado (DLL), otros archivos relacionados con ese código y un manifiesto descriptivo que incluye información como el número de versión del paquete. Los desarrolladores con código para compartir crean paquetes y los publican en un host público o privado. Los consumidores de paquetes obtienen esos paquetes de hosts adecuados, los agregan a sus proyectos y luego llaman a la funcionalidad de un paquete en el código de su proyecto. Luego, el propio NuGet maneja todos los detalles intermedios. (párr. 3)

El NuGet ML.NET es una biblioteca de código abierto desarrollada por Microsoft que permite agregar capacidades de aprendizaje automático (machine learning) a aplicaciones desarrolladas en el entorno .NET.

El proceso de trabajo con ML.NET implica los siguientes pasos:

* **Importación o creación de datos de entrenamiento:** Debes proporcionar datos de entrenamiento en un formato adecuado. En el caso de este proyecto, se utiliza información sobre gustos personales del usuario.
* **Preparación de datos y entrenamiento:** Se crean clases que se aplicarán a los datos antes de entrenar el modelo. Esto incluye la definición de columnas, la selección de algoritmos de entrenamiento y otros procesamientos.
* **Entrenamiento del modelo:** Se entrena el modelo utilizando los datos de entrenamiento. El modelo se ajusta a los datos para aprender los parámetros necesarios.
* **Realización de predicciones:** Una vez que el modelo está entrenado, se puede usar para realizar predicciones en nuevos datos.
* **Clase Predictor**

La clase Predictor, es donde implementamos el ML.NET y es la más importante del proyecto, se encarga de entrenar, cargar y utilizar un modelo de aprendizaje automático para predecir las clasificaciones de pareja ideal del usuario.

* **Atributos:** La clase tiene varios campos privados, incluyendo un objeto MLContext llamado \_context, que proporciona el contexto de Machine Learning de ML.NET, y otros campos para mantener el modelo, el esquema de datos y un motor de predicción.
* **Constructor:** Se utiliza para crear una instancia de la clase Predictor.
* **Método TrainModel:** Este método se utiliza para entrenar un modelo de clasificación multiclase utilizando datos de entrenamiento. Los datos de entrenamiento se leen desde un flujo CSV y se cargan en una vista de datos. Luego, se dividen en conjuntos de entrenamiento y prueba. El método configura un experimento de clasificación multiclase y lo ejecuta para entrenar un modelo. El mejor modelo resultante se almacena en \_model, y se devuelve información sobre el mejor modelo y sus métricas de rendimiento.

Texto

Descripción generada automáticamente

Ilustración - Método TrainModel

* **Método EnsurePredictorExists:** Este método se utiliza para garantizar que un motor de predicción esté disponible para hacer predicciones. Si el modelo no ha sido entrenado o cargado previamente, se lanza una excepción.
* **Método ClassifyData:** Este método se utiliza para realizar predicciones de clasificación. Asegura que el motor de predicción exista y luego utiliza ese motor para hacer la predicción.

Interfaz de usuario gráfica, Texto

Descripción generada automáticamente

Ilustración - Método ClassifyData

* **Método SaveModel:** Este método permite guardar el modelo entrenado en un flujo (stream) de salida. Antes de llamar a este método, se verifica que el modelo exista.
* **Método LoadModel:** Este método permite cargar un modelo previamente guardado desde un flujo de entrada. También se asegura de crear un motor de predicción basado en el modelo cargado.
* **Método ReadCsvFromStream:** Este método se utiliza para leer datos desde un flujo CSV y convertirlos en una colección de objetos Info. Utiliza la biblioteca CsvHelper para analizar el CSV. Los datos CSV son los utilizados para el entrenamiento del modelo.

**Texto

Descripción generada automáticamente**

Ilustración 7 - Método ReadCSVFromStream

En resumen, la clase Predictor encapsula la funcionalidad para entrenar, cargar, y utilizar un modelo de clasificación multiclase con ML.NET. Puede realizar predicciones basadas en datos de entrada y se pueden guardar o cargar modelos entrenados para su uso posterior. La clase también maneja la lectura de datos desde un flujo CSV para el entrenamiento del modelo.

## **Base de datos**

La base de datos de este proyecto es sencilla. Esta hecha por código utilizando SQL Server y Entity Framework Core.

* **Clase UserDataConfiguration**

La clase UserDataConfiguration es una configuración de entidad utilizada en Entity Framework Core para definir la estructura y el mapeo de la entidad UserData a una tabla de base de datos. Aquí se explica qué hace esta clase:

**Método Configure**: El método Configure, se definen varias configuraciones para la entidad UserData, que se reflejarán en la base de datos cuando se cree el esquema.

* **builder.ToTable("UserData"):** Establece el nombre de la tabla de base de datos como "UserData".
* **builder.HasKey(e => e.UserId):** Define la clave principal de la tabla como la propiedad UserId de la entidad UserData.
* **builder.Property(e => e.UserId):** Configura las propiedades de la entidad UserData que se mapearán a las columnas de la tabla de base de datos. Se establece el nombre de la columna y, en algunos casos, se indica que la propiedad es requerida (IsRequired).
* **builder.Property(e => e.Email).HasColumnName("email"):** Aquí, se configura la propiedad Email de la entidad UserData. Indica que la propiedad Email se mapeará a la columna de la base de datos con el nombre "email". Esto significa que cuando se almacenen datos en la base de datos o se recuperen datos de ella, la propiedad Email se mapeará a la columna "email" en la tabla correspondiente.
* **builder.Property(e => e.Password).HasColumnName("pwd")**: En esta línea, se configura la propiedad Password de la entidad UserData. Indica que la propiedad Password se mapeará a la columna de la base de datos con el nombre "pwd". Esto asegura que los datos de la propiedad Password se almacenen y recuperen de la columna "pwd" en la tabla de la base de datos.
* **builder.Property(e => e.StreamModel).HasColumnName("streamModel"):** En esta última línea, se configura la propiedad StreamModel de la entidad UserData. Indica que la propiedad StreamModel se mapeará a la columna de la base de datos con el nombre "streamModel". Esto garantiza que los datos de la propiedad StreamModel se almacenen y recuperen de la columna "streamModel" en la tabla de la base de datos.

En resumen, la clase UserDataConfiguration se utiliza para definir cómo se debe mapear la entidad UserData a una tabla de base de datos en Entity Framework Core. Define el nombre de la tabla, la clave principal y las columnas de la tabla, así como sus nombres y restricciones. Esto es fundamental para asegurar que los datos se almacenen y se recuperen correctamente en la base de datos.

Con los siguientes comandos se puede crear las migraciones con la configuración que hemos establecido y generar la base de datos en SQL Server, respectivamente:

Interfaz de usuario gráfica, Texto, Aplicación

Descripción generada automáticamente

Ilustración 8 - Comandos EF

* **Diagrama de Clase**

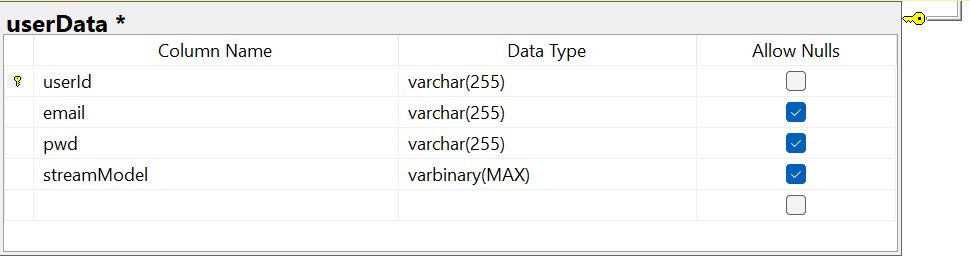
****

Ilustración 9 - Diagrama de Clase userData

## 

## **2.4 Arquitectura FrontEnd**

Bibliografía

* <https://learn.microsoft.com/es-es/previous-versions/msp-n-p/ff649690(v=pandp.10)?redirectedfrom=MSDN> 27/04/2010
* <https://learn.microsoft.com/es-es/nuget/what-is-nuget> 10/08/2023