
**Plataforma de medición instantánea para la prevención
de conductas autolesivas en adolescentes**
**Platform for the prevention of self-injurious behavior in
young people**



Trabajo de Fin de Máster
Curso 2023–2024

Autor
Aldair Fredy Maldonado Honores

Director
Gonzalo Méndez
Pablo Gervás Gómez-Navarro

Plataforma de medición instantánea para la prevención de conductas autolesivas en adolescentes

Platform for the prevention of self-injurious behavior in young people

Trabajo de Fin de Máster en Ingeniería Informática
Departamento de Ingeniería del Software e Inteligencia Artificial

Autor
Aldair Fredy Maldonado Honores

Director
Gonzalo Méndez
Pablo Gervás Gómez-Navarro

Dirigida por el Doctor
Gonzalo Méndez
Pablo Gervás Gómez-Navarro

Máster en Ingeniería Informática
Facultad de Informática
Universidad Complutense de Madrid

16 de agosto de 2024

Dedicatoria

A compañeros del máster y especialmente a mis padres, quienes me han brindado su total apoyo en todo momento y me han animado a seguir con el proyecto y el máster durante estos años.

Resumen

Plataforma de medición instantánea para la prevención de conductas autolesivas en adolescentes

El suicidio se ha convertido en una de las principales preocupaciones de salud a nivel global debido a que ha ganado mayor conciencia en los últimos años. Es por eso que su prevención se ha vuelto un reto entre los profesionales de la salud.

Bajo esta premisa, este Trabajo Fin de Máster busca diseñar y crear una aplicación para el SIVARIA, un proyecto de investigación cuyo objetivo es la prevención de conductas autolesivas suicidas y no suicidas en jóvenes adolescentes de entre 12 y 21 años. Para ello, la plataforma estará compuesta por una serie de cuestionarios que proceden del sitio web de SIVARIA que los usuarios pueden llenar. Estos cuestionarios variarán dependiendo de si el usuario es un joven, un familiar o un profesional quien los llena. Estos cuestionarios serán enviados a un Sistema Experto que tratará de predecir conductas autolesivas suicidas y no suicidas, tales como la ideación del suicidio, en la población joven. Para la realización de esta función, se diseñarán un conjunto de modelos utilizando clasificadores, entrenados con datasets generados a través de scripts artificiales y a través de un LLM (*Large Language Model*), y aplicando técnicas de ajuste de parámetros e hiperparámetros.

Gracias a estas predicciones, se pueden aplicar con anticipación medidas de monitorización y seguimiento a los pacientes, reduciendo los riesgos de posibles conductas suicidas.

Palabras clave

Inteligencia Artificial, Sistema Experto, Bayes, React Native, API, Django, Scikit-learn, Clasificador, Aplicación multiplataforma

Abstract

Platform for the prevention of self-injurious behavior in young people

Suicide has become a major global health concern as it has gained greater awareness in recent years. For this reason, its prevention has become a challenge among health professionals.

Under this premise, this Master's Thesis aims to design and create an application for SIVARIA, a project focused on the prevention of suicidal and non-suicidal self-injurious behaviors. For this purpose, the platform will be composed of a set of questionnaires that come from the SIVARIA website that users can fill out. These questionnaires will vary their content depending on whether the user is a young person, a family member or a professional. These questionnaires will then be sent to an Expert System which will try to predict suicidal and non-suicidal self-injurious behaviors, such as suicidal ideation, in young population. For the realization of this task, a set of models will be designed using classifiers, trained with datasets generated through artificial scripts and later, through a Large Language Model and applying parameter and hyperparameter fitting techniques.

Thanks to these predictions, monitoring and follow-up measures can be applied to patients in advance, reducing the risks of self-injurious behavior. patients in advance, reducing the risks of possible suicidal behavior.

Keywords

Artificial Intelligence, Expert System, Bayes, React Native, API, Django, Scikit-learn, Classifier, Multiplatform application

Índice

1. Introducción	1
1.1. Motivación	1
1.2. Objetivos	1
1.3. Plan de trabajo	2
2. Estado del Arte	5
2.1. Salud mental juvenil	5
2.2. Aplicaciones	5
2.2.1. CalmHarm	5
2.2.2. Prevensuic	6
2.2.3. Suicide Safety Plan	6
2.2.4. ISNISS	6
2.2.5. PAPAGENO	7
2.3. Tabla comparativa de las aplicaciones	7
2.4. Análisis comparativo de las aplicaciones	8
2.4.1. Conclusión	9
3. Descripción del Trabajo	11
3.1. Sistema Experto	11
3.2. Redes de Bayes	12
3.2.1. Probabilidad conjunta	13
3.2.2. Probabilidad condicional	14
3.2.3. Distribuciones de probabilidad condicional	15
3.2.4. Inferencias	16
3.3. Diseño del modelo	16
3.3.1. Modelo del Autoinforme	16
3.3.2. Modelo de las familias	19
3.3.3. Modelo de los profesionales	20
3.4. Desarrollo de los modelos	20
3.4.1. Datasets	22

3.4.2. Procesos de entrenamiento y testeo del modelo	22
3.5. Versiones del entrenamiento	26
3.5.1. Primera versión: datasets artificiales y aleatorios	26
3.5.2. Segunda versión: nuevos datasets con LLM	38
3.5.3. Tercera versión: cambio en el diseño del modelo	42
3.5.4. Cuarta versión: ajuste de hiperparámetros	53
3.5.5. Conclusiones del entrenamiento	58
3.6. Aplicación	59
3.6.1. Backend	59
3.6.2. Frontend	69
3.6.3. Conexión entre el frontend y el backend	70
3.6.4. Casos de uso	70
3.6.5. Funcionamiento de la aplicación	79
3.6.6. Envío de notificaciones push	83
3.6.7. Conexión con el Sistema Experto	84
3.6.8. Seguridad de la plataforma	87
4. Herramientas de desarrollo y tecnologías	89
4.1. Herramientas	89
4.1.1. Visual Studio Code	89
4.1.2. Github	89
4.1.3. Latex	90
4.1.4. Programas de creación de diagramas	90
4.1.5. Anaconda	90
4.1.6. Postman	91
4.1.7. Expo Go	91
4.1.8. Gmail	91
4.2. Tecnologías y lenguajes	92
4.2.1. Python	92
4.2.2. Django framework	92
4.2.3. React Native	92
4.2.4. Expo framework	93
5. Conclusiones y Trabajo Futuro	95
5.1. Conclusiones	95
5.2. Trabajo a futuro	95
6. Introduction	97
6.1. Motivation	97
6.2. Goals	97
6.3. Work Plan	98

7. Conclusions and Future Work	101
7.1. Conclusions	101
7.2. Future work	101
Bibliografía	103
A. Guía de instalación	105
A.1. Contenido del repositorio	105
A.1.1. Scripts	105
A.2. Instrucciones de ejecución	106
A.2.1. Entornos de Python con Anaconda	107
A.2.2. Conexión del entorno virtual con Visual Studio Code	107
A.2.3. Instalación de los paquetes	108
B. Imágenes adicionales	113

Índice de figuras

2.1. Interfaces de Prevensuic	6
2.2. Interfaces de Suicide Safety Plan.	7
2.3. Interfaz web de ISNISS.	7
2.4. Interfaz web de PAPAGENO.	8
3.1. Estructura del Sistema Experto	12
3.2. Página inicial del sitio web de SIVARIA	13
3.3. Encuesta a jóvenes de más de 16 años de SIVARIA	13
3.4. Ejemplo de un Diagrama de Bayes. Fuente, Hassan Khosravi, Diapositiva 14	14
3.5. Red de Bayes del modelo del Autoinforme (I)	19
3.6. Red de Bayes del modelo del Autoinforme (II)	19
3.7. Red de Bayes del modelo del Autoinforme (III)	19
3.8. Red de Bayes del modelo de las familias	20
3.9. Red de Bayes del modelo de los profesionales	20
3.10. Ejemplo de la creación de una tabla de frecuencia (segunda tabla) y tabla de probabilidad (tercera tabla) dado un dataset (primera tabla). Fuente del ejemplo: Sitio web de Javatpoint	21
3.11. Ejemplo del directorio de archivos de guardado con el modelo del Autoinforme	24
3.12. Matriz de confusión del Autoinforme al final del primer ciclo	28
3.13. Matriz de confusión de las familias al final del primer ciclo	28
3.14. Matriz de confusión de los profesionales al final del primer ciclo	29
3.15. Matriz de confusión del Autoinforme al final del tercer ciclo	31
3.16. Matriz de confusión de las familias al final del tercer ciclo	32
3.17. Matriz de confusión de los profesionales al final del tercer ciclo	32
3.18. Matriz de confusión del Autoinforme al final del quinto ciclo	34
3.19. Matriz de confusión de las familias al final del quinto ciclo	35
3.20. Matriz de confusión de los profesionales al final del quinto ciclo	35
3.21. Matrices de confusión. Autoinforme. Iteración 7	40
3.22. Matrices de confusión. Autoinforme. Iteración 8	41
3.23. Matrices de confusión. Autoinforme. Iteración 9	42
3.24. Matrices de confusión. Autoinforme. Iteración 10	43

3.25. Matrices de confusión. Autoinforme. Iteración 11	45
3.26. Matrices de confusión. Autoinforme. Iteración 12	46
3.27. Matrices de confusión. Autoinforme. Iteración 14	47
3.28. Matrices de confusión. Autoinforme. Iteración 15	48
3.29. Matrices de confusión. Familia. Iteración 16	49
3.30. Matriz de confusión. Familia. Iteración 17	50
3.31. Matriz de confusión. Familia. Iteración 18	51
3.32. Matrices de confusión. Profesionales. Iteración 19	52
3.33. Matrices de confusión. Profesionales. Iteración 20	53
3.34. Matrices de confusión del Autoinforme(I), familias (II) y profesionales (III) del clasificador gausiano. Iteración 21	56
3.35. Matrices de confusión del Autoinforme(I), familias (II) y profesionales (III) del clasificador multinomial. Iteración 21	56
3.36. Matrices de confusión del Autoinforme(I), familias (II) y profesionales (III) del clasificador categórico. Iteración 21	57
3.37. Diagrama de las relaciones de las entidades	68
3.38. Arquitectura de Ngrok. Fuente: Página oficial de Ngrok	70
3.39. Diagrama de casos de uso.	71
3.40. Pantalla de registro	79
3.41. Pantalla de inicio de sesión	80
3.42. Pantalla de recuperación de contraseña	81
3.43. Pantallas de cuestionarios de jóvenes (I), familiares (II) y profesionales (III)	82
3.44. Pantalla de registro de cuestionarios	83
3.45. Arquitectura del envío de notificaciones de Expo. Fuente: Página oficial de Expo	84
3.46. Estructura del Controlador con el Sistema Experto	85
4.1. Programas de Anaconda	91
A.1. Ventana de creación de nuevo entorno de Python en Anaconda	107
A.2. Panel de selección de entorno de Python en Visual Studio Code	107
A.3. Consola del nuevo entorno de Python	107
A.4. Consola del servidor frontend	109
A.5. Interfaz de la aplicación de Expo	110
A.6. Login endpoint en Postman	111
A.7. Cabecera de la petición para los endpoints con autorización	111
B.1. Estructura del dataframe en la segunda versión de entrenamiento	114
B.2. Estructura del dataframe del modelo del autoinforme en la tercera versión	114
B.3. Estructura del dataframe del modelo de las familias en la tercera versión	115
B.4. Estructura del dataframe del modelo de los profesionales en la tercera versión	115
B.5. Diagrama de secuencia del comando para establecer el tipo de modelo	116
B.6. Diagrama de secuencia del comando para listar los archivos de guardado	117

B.7. Diagrama de secuencia del comando para mostrar la información del archivo de guardado	118
B.8. Diagrama de secuencia del comando para eliminar un archivo de guardado	119
B.9. Diagrama de secuencia del comando para entrenar al modelo	120
B.10. Diagrama de secuencia del comando para realizar una predicción	121

Índice de tablas

2.1. Tabla comparativa de aplicaciones de prevención contra el suicidio. Última actualización: mayo de 2024	8
2.2. Continuación de la Tabla comparativa 2.1	9
3.1. CPD de la variable <i>Alarm</i>	16
3.2. Comparativa de las métricas del modelo. Iteración 1	27
3.3. Comparativa de las métricas del modelo. Iteración 2	30
3.4. Comparativa de las métricas del modelo. Iteración 3	31
3.5. Comparativa de las métricas del modelo. Iteración 4	33
3.6. Comparativa de las métricas del modelo. Iteración 5	34
3.7. Comparativa de las métricas del modelo. Iteración 6	36
3.8. Tabla comparativa de las métricas. Primera Versión	37
3.9. Resultados de las métricas de la iteración 7	39
3.10. Resultados de las métricas de la iteración 8	40
3.11. Resultados de las métricas de la iteración 9	41
3.12. Resultados de las métricas de la iteración 10	43
3.13. Resultados de las métricas de la iteración 11	44
3.14. Resultados de las métricas de la iteración 12	45
3.15. Resultados de las métricas de la iteración 13	46
3.16. Resultados de las métricas de la iteración 14	47
3.17. Resultados de las métricas de la iteración 15	48
3.18. Resultados de las métricas de la iteración 16	49
3.19. Resultados de las métricas de la iteración 17	50
3.20. Resultados de las métricas de la iteración 18	51
3.21. Resultados de las métricas de la iteración 19	51
3.22. Resultados de las métricas de la iteración 20	53
3.23. Métricas del modelo de Gauss. Iteración 21	55
3.24. Métricas del modelo multinomial. Iteración 21	55
3.25. Métricas del modelo categórico. Iteración 21	56
3.26. Tabla comparativa de las métricas del modelo de Gauss	58
3.27. Tabla comparativa de las métricas del modelo multinomial	58

3.28. Tabla comparativa de las métricas del modelo categórico	58
3.29. Tabla comparativa de las métricas. Versión final.	59
3.30. Caso de uso 1: Iniciar sesión	72
3.31. Caso de uso 2: Crear cuenta	73
3.32. Caso de uso 3: Cerrar sesión	74
3.33. Caso de uso 4: Recuperar contraseña	74
3.34. Caso de uso 5: Modificar datos	75
3.35. Caso de uso 6: Hacer cuestionario	76
3.36. Caso de uso 7: Predecir desenlace	76
3.37. Caso de uso 8: Notificar desenlace	77
3.38. Caso de uso 9: Ver respuestas del cuestionario	78

Capítulo 1

Introducción

1.1. Motivación

En estos últimos años, el suicidio se ha convertido en un problema de salud a nivel global, siendo una de las principales causas de muerte no natural entre jóvenes y adultos. Es por ello que su prevención y mitigación ha presentado un reto para los profesionales de la salud, ya que dicha problemática afecta tanto a los propios individuos, como también a los familiares y amigos cercanos.

Es por ello que en los últimos años, se están centrando los esfuerzos y los recursos económicos en la creación de programas de prevención del suicidio. En estos planes, es fundamental realizar diagnósticos lo más pronto posible para detectar a tiempo síntomas mentales en la persona, y por lo tanto, poder ofrecerle un tratamiento. El problema es que muchas veces, las personas con riesgo de adoptar estas conductas suicidas y autolesivas no son conscientes de que los padecen o que pueden padecerlos, por lo que no acuden a los planes de prevención. Esto trae, como consecuencia, el agravamiento del problema, hasta acabar en un desenlace fatal. Es importante el poder informar a la persona a tiempo sobre su estado de salud mental de forma automática, rápida y eficaz, para que así sea consciente del riesgo al que está sometido, y por lo tanto, pueda pedir ayuda con anticipación.

Ante este panorama, se busca desarrollar herramientas para prevenir este tipo de conductas en la juventud. Es aquí donde comienza a jugar un papel fundamental la inteligencia artificial. De acuerdo a al artículo de Fonseka et al. (2019), la inteligencia artificial se está utilizando cada vez más en el campo de la salud mental, siendo de gran utilidad para la prevención del suicidio y dar soporte a los profesionales para conseguir una evaluación exacta del riesgo de suicidio. De esta manera, se pueden tratar a los potenciales pacientes que posean estas conductas autolesivas o no autolesivas en etapas muy tempranas del desarrollo de estos comportamientos.

El motivo principal de la elección de este tema es por la relevancia que ido adquiriendo este tema a lo largo de los años. Además, es un proyecto que abarcaba el apartado de desarrollo de aplicaciones web y móviles, con el que estoy más familiarizado.

1.2. Objetivos

El objetivo principal del proyecto es el poder desarrollar una plataforma que, a través de una serie de cuestionarios, pueda realizar predicciones mediante consultas a un Sistema

Experto. Para lograr este objetivo principal, se establecieron los siguientes objetivos de proyecto a lo largo del desarrollo del TFM.

- Diseñar y validar un Sistema Experto para la valoración del riesgo de conductas autolesivas suicidas y no suicidas en población adolescente (12-21 años).
- El Sistema Experto debe ser capaz de pronosticar aquellos adolescentes y jóvenes de alto riesgo.
- Implementar una forma de prevenir conductas autolesivas futuras mediante la monitorización y supervisión de profesionales.
- Diseñar y desarrollar una aplicación web y móvil que pueda albergar las predicciones del Sistema Experto e implementar los objetivos anteriores.

1.3. Plan de trabajo

En base a los objetivos marcados en el apartado anterior, se estableció el siguiente plan de trabajo.

- Selección del tipo de Sistema Experto. Se realizará una búsqueda de los distintos tipos de Sistemas Experto que hay y se seleccionará el más indicado para este caso.
- Entendimiento del proyecto de SIVARIA. Acceder a la página del proyecto de SIVARIA e investigar y entender los cuestionarios de la página, que serán de utilidad para diseñar los modelos del Sistema Experto.
- Selección de herramientas y librerías para implementar los modelos para el Sistema Experto seleccionado.
- Diseño e implementación de los modelos mediante las herramientas y librerías seleccionadas.
- Entrenar y validar los modelos creados a través de varios ciclos o iteraciones. Se interpretarán los resultados en las iteraciones y se mejorarán los parámetros y datos de entrada hasta conseguir un rendimiento aceptable en los modelos del Sistema Experto.
- Estudio de herramientas y tecnologías para la creación de la aplicación móvil.
- Creación de los casos de uso para el desarrollo de la aplicación.
- Desarrollo: se lleva a cabo el desarrollo de la aplicación en base a los casos de uso especificados mediante la utilización de las herramientas estudiadas.

Todo el código realizado durante el desarrollo del proyecto se encuentra en el siguiente repositorio de Github.

<https://github.com/NILGroup/TFM-2324-SIVARIA>

Cuenta con licencia MIT, y el repositorio es público. Dentro del repositorio se encuentran 4 carpetas:

- **DiagramBayesSketches**: se encuentran los bocetos de los diagramas de Bayes realizados.
- **Memoria**: carpeta con la memoria del proyecto.
- **app**: aquí se ubican los dos proyectos de la aplicación multiplataforma, tanto el proyecto frontend como el backend.
- **scripts**: se encuentran todos los scripts utilizados para el entrenamiento del modelo. Cuenta con una serie de cuadernos Jupyter con los pasos de los entrenamientos realizados, divididas en las versiones conforme aparecen a lo largo de la memoria. Además, cuenta con un prototipo del Sistema Experto que se encuentra dentro del servidor de backend en la carpeta *app*. También se puede utilizar esos scripts de Python para realizar las mismas pruebas que en el cuaderno, aunque en este caso el funcionamiento se basa en comandos.

Capítulo 2

Estado del Arte

En este capítulo se el estado actual de la salud mental en la población juvenil y la situación de las aplicaciones centradas en abordar estos temas.

2.1. Salud mental juvenil

La salud mental es un estado en el que una persona se encuentra en un bienestar emocional, social y psicológico. Este bienestar influye directamente en nuestra vida diaria, permitiéndonos o no el poder realizar actividades cotidianas, interactuar con otras demás, entre otros. Es por ello que, la falta de salud mental puede acarrear problemas y trastornos mentales a las personas, y por consiguiente, en la adopción de conductas autolesivas suicidas y no suicidas.

Las conductas autolesivas son comportamientos que adoptan las personas para infligirse autolesiones. Dentro de los comportamientos autolesivos, están los comportamientos no suicidas, que son conductas autolesivas que no buscan causar la muerte, sino sólo el inflingir daño a sí mismos. Estas conductas puede incluir cortarse, quemarse o golpearse. Por otro lado están los comportamientos suicidas, que son más graves ya que en estos casos sí que se busca poner fin a la vida, por lo que representan una emergencia de salud mental.

2.2. Aplicaciones

Es por ello, que se han tratado de desarrollar aplicaciones web y móviles que buscan tratar el tema de la salud mental en las personas. A continuación, mostraremos algunos ejemplos y se realizará un análisis comparativo entre todas ellas.

2.2.1. CalmHarm

CalmHarm¹ es una aplicación móvil desarrollada por la Dra. Nihara Krause, psicólogo clínica, para la organización benéfica Stem4 en Reino Unido. Se centra en ofrecer una serie de actividades generales y personalizadas basadas en la Terapia Dialéctica Conductual (DBT) (Salsman y Linehan (2006)) para regular las emociones fuertes que empujan a la

¹<https://calmharm.stem4.org.uk/>

autolesión y impulsando que se adopten conductas más sanas. Menciona que se puede usar para jóvenes de 13 años en adelante.

2.2.2. Prevensuic

Prevensuic² es un programa creado por la *Fundación Española para la Prevención del Suicidio*, destinado a la prevención, divulgación y la formación acerca del suicidio. Dicho programa cuenta con un sitio web y una aplicación móvil (disponible tanto en Android como en iOS) para ser accesible a un mayor número de personas, tanto adultos como jóvenes, como se puede observar en la Figura 2.1.



Figura 2.1: Interfaces de Prevensuic

2.2.3. Suicide Safety Plan

Suicide Safety Plan³ es una aplicación móvil desarrollada por Inquiry Health LLC y está disponible tanto en Android como en iOS (interfaces de la app en la Figura 2.2). La aplicación funciona como un recordatorio para que el usuario pueda establecer qué cosas pueden afectar a su salud mental, e incluye información para ayudar a la persona a evaluar su propio estado mental. Además, proporciona una serie de números de teléfono para contactar con profesionales en caso de necesitar ayuda urgente.

2.2.4. ISNISS

En este caso, ISNISS⁴ es un sitio web, como se puede observar en su interfaz de la Figura 2.3, que se encarga de dar información sobre cómo realizar una evaluación correcta del riesgo de suicidio en un paciente, aunque no la realiza el propio sitio web automáticamente. Aparte incluye otra serie de servicios, como terapias psicológicas.

²<https://www.prevensuic.org/>

³https://play.google.com/store/apps/details?id=com.moodtools.crisis.app&hl=en_US

⁴<https://www.isniss.es/>

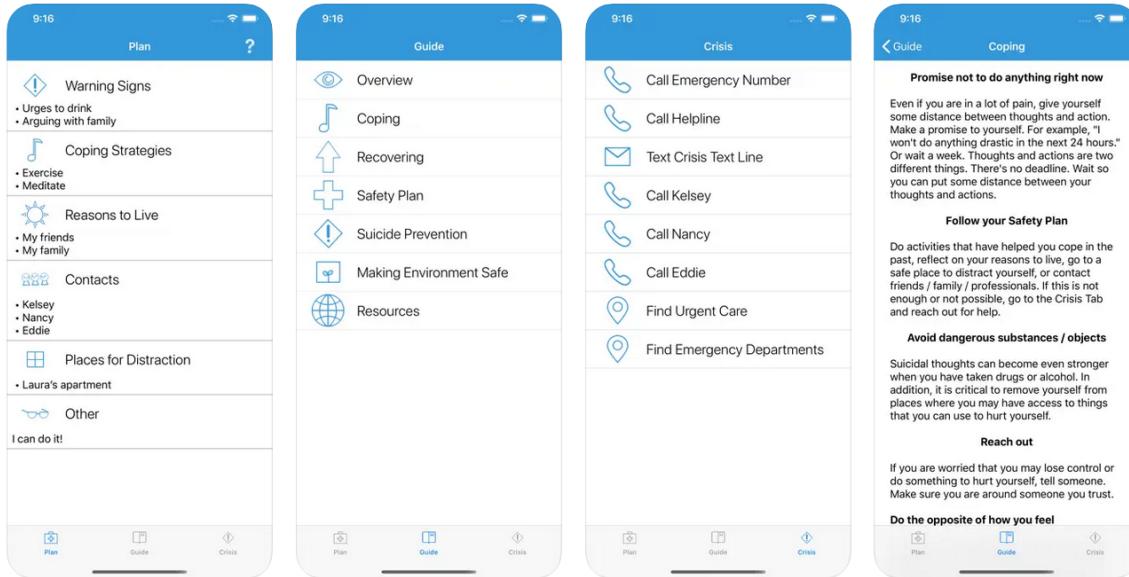


Figura 2.2: Interfaces de Suicide Safety Plan.

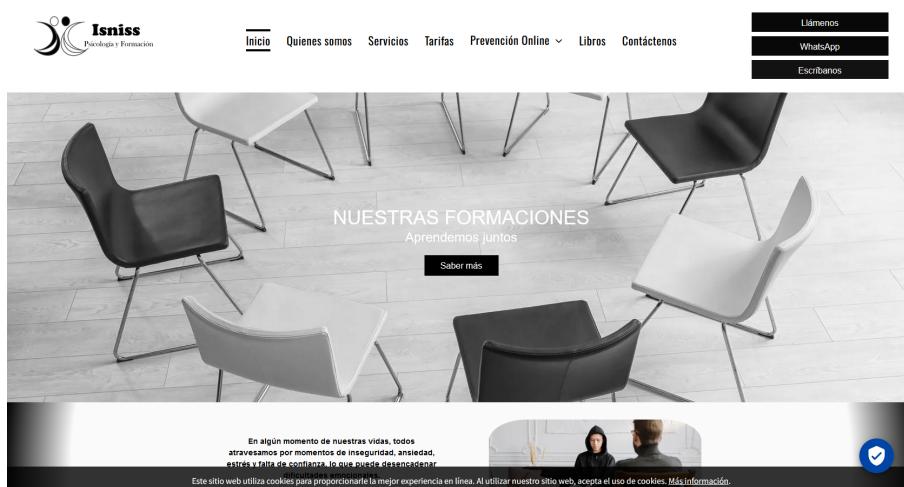


Figura 2.3: Interfaz web de ISNISS.

2.2.5. PAPAGENO

PAPAGENO⁵ es un sitio web que ofrece información, documentos científicos y materiales de autoayuda para la prevención, investigación, tratamiento y postvención de las conductas suicidas. Destaca por proporcionar enlaces y números de teléfono de urgencias contra el suicidio, guías de autoayuda, información sobre asociaciones enfocadas en esta temática, etc. Se puede observar la interfaz principal de la web en la Figura 2.4

2.3. Tabla comparativa de las aplicaciones

Se crearon dos tablas comparativas (Tabla 2.1 y Tabla 2.2), en donde se evaluaban las aplicaciones mediante una serie de criterios, como:

⁵<https://papageno.es/about/quienes-somos>



Figura 2.4: Interfaz web de PAPAGENO.

- Tipo de aplicación, que puede ser web y/o móvil.
- Interacción directa: indica si la aplicación interactúa con los usuarios, en el sentido de incluir actividades personalizadas e interactivas, seguimiento de cada caso y retroalimentación en tiempo real.
- Enfoque educativo: indica si la aplicación proporciona información educativa y materiales de formación para los usuarios.
- Plan de seguridad: se refiere a si la aplicación tiene la posibilidad de crear o gestionar un plan de emergencia en caso de crisis.
- Recursos de emergencia: se refiere a si la aplicación proporciona acceso rápido a contactos de emergencia, como números de teléfono de emergencia y ayuda.

En base a estos criterios, se elaboraron las tablas comparativas 2.1 y 2.2.

	Prevensuic	Suicide Safety Plan	CalmHarm
Tipo de aplicación	Web y móvil	Móvil	Móvil
Interacción directa	✓	✓	✓
Enfoque educativo	✓	✗	✓
Plan de seguridad	✓	✓	✗
Recursos de emergencia	✓	✓	✗

Tabla 2.1: Tabla comparativa de aplicaciones de prevención contra el suicidio. Última actualización: mayo de 2024

2.4. Análisis comparativo de las aplicaciones

Al analizar las aplicaciones, pude observar que son aplicaciones diversas, en el que cada una tiene puntos fuertes y se enfocan en aspectos concretos. Desde el punto de vista de los usuarios adolescentes que buscan aplicaciones interactivas, CalmHarm es una opción destacada. Por otro lado, Prevensuic y Suicide Safety Plan son útiles para ofrecer planes de

	ISNISS	PAPAGENO
Tipo de aplicación	Web	Web
Interacción directa	X	X
Enfoque educativo	✓	✓
Plan de seguridad	✓	X
Recursos de emergencia	✓	✓

Tabla 2.2: Continuación de la Tabla comparativa 2.1

seguridad y accesos rápidos a recursos de emergencia. Finalmente, ISNISS y PAPAGENO se especializan en proporcionar información educativa para el seguimiento, evaluación y prevención de conductas autolesivas, que pueden ser de ayuda para profesionales y padres interesados.

Sin embargo, una cosa en común entre todos ellos es que no ofrecen ningún algoritmo ni ningún sistema para poder detectar y pronosticar en tiempo real las posibles conductas autolesivas en las personas, sino que ofrecen herramientas para poder prevenirlo, tales como las actividades de CalmHarm o los planes de seguridad de Suicide Safety Plan. Además, cabe desataracar que Prevensuic, es considerada una aplicación o programa muy completo y psicoeducativo, es decir, que es capaz de brindar información a las personas que están sufriendo de un trastorno psicológico.

Sin embargo, Prevensuic presenta un gran inconveniente, y es que no está actualizado para las nuevas versiones de sistemas operativos (Android 14, por ejemplo). Su última actualización data del 11 de abril de 2019, lo que supone una disminución considerable de la accesibilidad de la aplicación y, por lo tanto, del programa en general.

2.4.1. Conclusión

En base a las ventajas y desventajas de las aplicaciones analizadas, se buscará diseñar Sistema Experto y una aplicación multiplataforma que pueda ser accesible a las versiones más recientes de Android y iOS, que además tenga versión web, y que pueda proporcionar una evaluación automática del riesgo de suicidio en base a las respuestas de los cuestionarios de SIVARIA. Dicha aplicación estará dirigida principalmente a jóvenes, pero también pueden acceder profesionales y los familiares y conocidos del paciente.

Capítulo 3

Descripción del Trabajo

Teniendo en cuenta los objetivos del proyecto, el primer paso fue diseñar la infraestructura del Sistema Experto. Para lograr dicho propósito, era necesario realizar una investigación para aprender los fundamentos de un Sistema Experto, estudiar qué tipos de sistemas pueden haber y cuál sería el tipo elegido para los modelos, que se verá a continuación.

3.1. Sistema Experto

Un sistema experto (Wikipedia (2024b)) es un programa informático diseñado para simular el conocimiento y las habilidades analíticas de un especialista en un campo concreto. Su función es simular la forma en la que un especialista toma una decisión respecto a un problema.

Dentro del concepto de sistema experto, existen los siguientes tipos:

- **Reglas previamente establecidas o RBR (*Rule Based Reasoning*)**: en base a la definición proporcionada por Grosan y Abraham (2011), son sistemas en el que el conocimiento adquirido es representado mediante reglas IF. Por ejemplo, dado una condición A y las conclusiones B y C , la estructura sería:

if A then B else C

Un gran inconveniente de este tipo de sistema requiere de una coincidencia exacta entre el problema planteado y las precondiciones establecidas para poder predecir las conclusiones, por lo que puede llegar a ser un factor limitante en grandes proyectos.

- **Basados en casos o CBR (*Case Based Reasoning*)**: de acuerdo a la definición del artículo Kolodner (1992), un sistema experto basado en CBR aprende de experiencias anteriores, en lugar de reglas, como sucedía con los sistemas RBR. De esta manera, cuando se le presenta un problema al modelo, este empleará las experiencias con las que se le ha entrenado para elaborar una solución nueva.
- **Basados en redes bayesianas**: es un sistema cuyas decisiones estarán basadas en un modelo probabilístico conformado por variables y relaciones de dependencia entre ellas. De esta manera, cuando se le presenta un problema, el sistema realizará su predicción en base a las probabilidades de las variables que se habrán calculado previamente en base a los entrenamientos que el sistema haya recibido.

En nuestro caso, se ha decidido implementar el Sistema Experto mediante el uso de redes bayesianas, ya que presentan mayor flexibilidad que los otros tipos. En el caso de los modelos RBR y CBR presentan demasiada rigidez y es necesario especificar las reglas y los casos explícitamente para que realice las predicciones correctamente.

Durante esta fase de investigación, se realizó un proceso de aprendizaje sobre los sistemas expertos y las redes bayesianas. Se leyó documentación para aprender y entender el concepto de redes bayesianas y sus componentes.

Con respecto al proyecto, el objetivo de este sistema es:

- Identificar y, por lo tanto, predecir, aquellos jóvenes que cumplen con las condiciones para ser considerados de alto riesgo (pronóstico).
- Monitorizar sus conductas de riesgo mediante evaluación momentánea (Ecological Momentary Assessment) (monitorización).
- Prevenir que se produzcan conductas autolesivas futuras (prevención), mediante la provisión de indicaciones y actuaciones personalizadas y ajustadas a los niveles de riesgo bajo la supervisión de profesionales.

Por otro lado, se ha propuesto que la estructura del sistema sea de tal forma que reciba una serie de datos de entrada, como se muestra en la Figura 3.1.

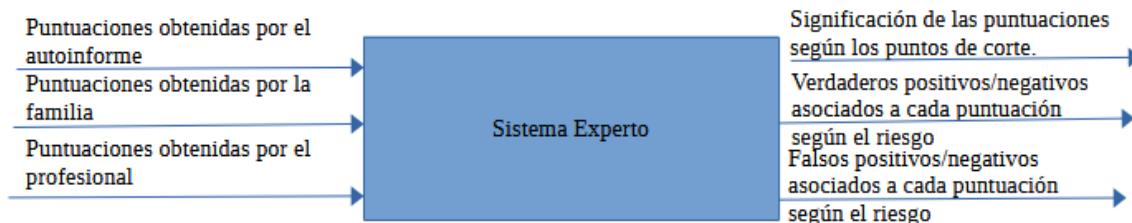


Figura 3.1: Estructura del Sistema Experto

El sistema recibirá una serie de puntuaciones obtenidas de las encuestas realizadas a los usuarios por el autoinforme, a las familias de los usuarios y a los profesionales de la salud (médicos, psicólogos, etc.).

Dichas encuestas se recogen en el sitio web de SIVARIA (Figuras 3.2 y 3.3).

En base a estos datos y al entrenamiento realizado, el sistema los procesará y devolverá el número de verdaderos y falsos positivos y negativos. Esto será interpretado por la aplicación para determinar si existen conductas autolesivas en el usuario o los potenciales comportamientos que podría llegar a tener.

3.2. Redes de Bayes

Como se ha explicado anteriormente, se ha optado por implementar el Sistema Experto mediante una red de Bayes (Sucar y Tonantzintla (2006)).

Las redes bayesianas, propuesto inicialmente por Judea Pearl en 1988, son un modelo probabilístico en el que se representan las relaciones probabilísticas entre las variables (nodos) y las dependencias condicionales que hay entre ellas, formando de esta manera un grafo acíclico dirigido (DAG, *Directed Acyclic Graph*).



Figura 3.2: Página inicial del sitio web de SIVARIA

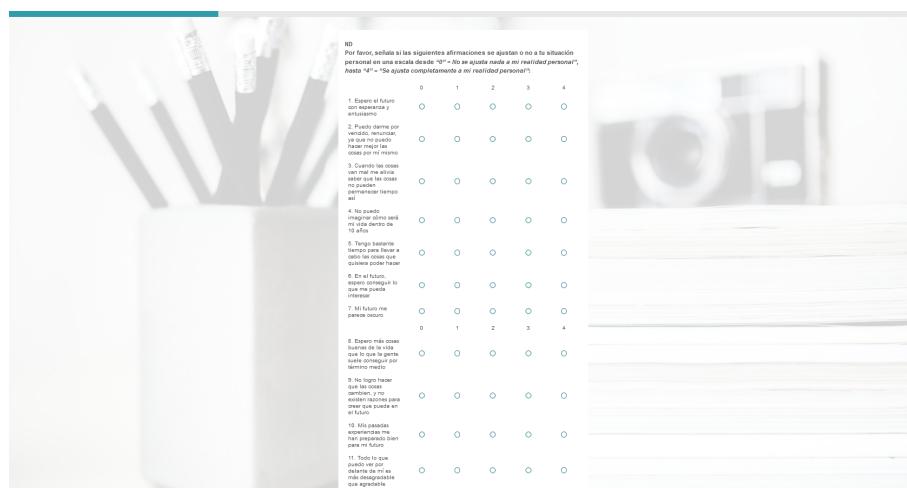


Figura 3.3: Encuesta a jóvenes de más de 16 años de SIVARIA

Para que un grafo pueda ser considerado un DAG, es necesario que todos los arcos (*edges*) del grafo sean dirigidos, es decir, que indiquen una y sólo una dirección. Además, el grafo no debe tener ciclos, es decir, que dado un vértice v , no hay un camino directo que empiece y termine en v .

Dentro de una red bayesiana, es necesario entender los conceptos de probabilidad conjunta y probabilidad condicional.

3.2.1. Probabilidad conjunta

La probabilidad conjunta es la probabilidad de que dos o más eventos ocurran al mismo tiempo. En una red de Bayes, la probabilidad conjunta de n variables se puede representar como:

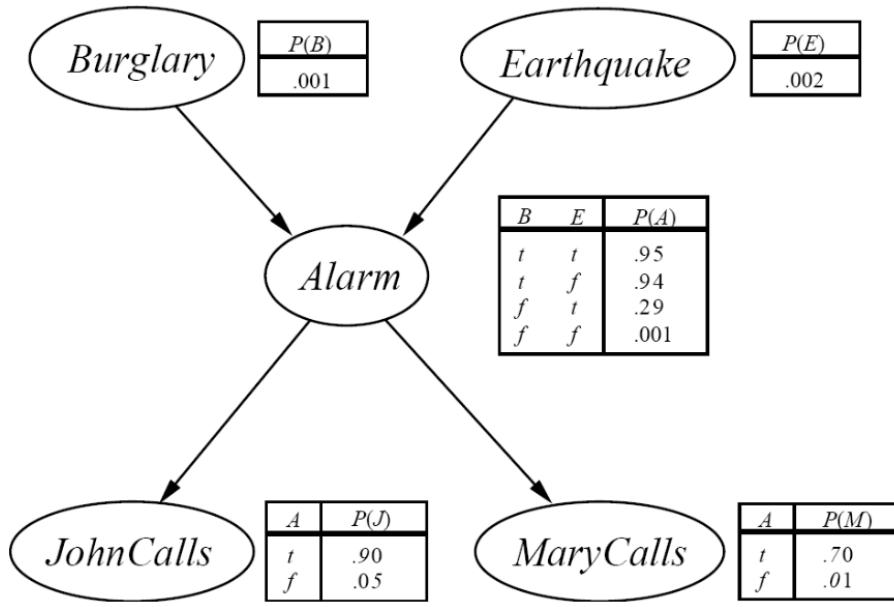


Figura 3.4: Ejemplo de un Diagrama de Bayes. Fuente, Hassan Khosravi, Diapositiva 14

$$P(X_1, X_2, \dots, X_n) = \prod_{i=1}^n P(X_i | X_{p(i)}) \quad (3.1)$$

Donde $X_{p(i)}$ es el padre de X_i .

Por ejemplo, dado:

- J = *John Calls*.
- M = *Mary Calls*.
- A = *Alarm*.
- B = *Burglary*.
- E = *Earthquake*.

La probabilidad conjunta de la Figura 3.4 sería:

$$\begin{aligned} P(J, M, A, B, E) &= P(J|A)P(M|A)P(A|B,E)P(B,E) \\ &= P(J|A)P(M|A)P(A,B,E) \\ &= P(J|A)P(M|A)P(A|B,E)P(B,E) \\ &= P(J|A)P(M|A)P(A|B,E)P(B)P(E) \end{aligned} \quad (3.2)$$

3.2.2. Probabilidad condicional

La probabilidad condicional es la probabilidad de que un evento A ocurra dado otra evento B, $P(A|B)$.

Para calcular cualquier probabilidad condicional, se utiliza el Teorema de Bayes:

$$P(A_i|B) = \frac{P(A_i \cap B)}{P(B)} = \frac{P(A_i) \cdot P(B|A_i)}{P(B)} \quad (3.3)$$

Donde $P(A_i)$ son las probabilidades a priori del espacio muestral, es decir, las probabilidades iniciales que ya sabemos su valor, y $P(A_i|B)$ son las probabilidades a posteriori, es decir, posibilidad de que suceda un resultado B dado un conjunto de eventos $A = \{A_1, A_2, \dots, A_i\}$.

Por ejemplo, observando el diagrama de la Figura 3.4, se puede saber que la probabilidad de, habiendo sonado la alarma, John haya llamado sería 0.90.

$$P(J|A) = \frac{P(J \cap A)}{P(A)} = \frac{P(J) \cdot P(A|J)}{P(A)} = 0,90$$

Otro ejemplo más complejo sería calcular la probabilidad de que suene la alarma después de que ocurra un terremoto.

$$\begin{aligned} P(A|E) &= \frac{P(A \cap E)}{P(E)} \\ &= \frac{P(A|E)P(E)}{P(E)} \\ &= \frac{P(A|E, B)P(B \cap E) + P(A|E, \neg B)P(\neg B \cap E)}{P(E)} \\ &= \frac{P(A|E, B)P(B)P(E) + P(A|E, \neg B)P(\neg B)P(E)}{P(E)} \\ &= \frac{P(E)(P(A|E, B)P(B) + P(A|E, \neg B)P(\neg B))}{P(E)} \\ &= P(A|E, B)P(B) + P(A|E, \neg B)P(\neg B) = (0,95 \cdot 0,001) + (0,29 \cdot 0,999) = 0,29066 \end{aligned} \quad (3.4)$$

Gracias a estos dos conceptos, se puede explicar uno de los elementos imprescindibles de las redes bayesianas: las tablas de distribución de probabilidad condicional.

3.2.3. Distribuciones de probabilidad condicional

Como se puede observar en la red de la Figura 3.4, una de las características fundamentales de una red de Bayes son las tablas de distribución de probabilidad condicional o CPD (*Conditional Probability Distribution*).

Una CPD es una tabla donde se especifica la probabilidad de cada posible valor de una variable dada una combinación específica de los valores de sus variables padres. Tomando como ejemplo la Figura 3.4, la ecuación de la probabilidad condicional de que la alarma suene (*Alarm*) sería la siguiente:

$$P(Alarm|MaryCalls, JohnCalls, Burglary, Earthquake) = P(Alarm|Burglary, Earthquake) \quad (3.5)$$

La variable *Alarm* es dependiente de *Burglary* y *Earthquake*, por lo que su probabilidad queda condicionada por la probabilidad conjunta de estas dos variables.

Burglary	Earthquake	P(Alarm=T)	P(Alarm=F)
T	T	0.94	0.06
T	F	0.95	0.04
F	T	0.69	0.69
F	F	0.999	0.999

Tabla 3.1: CPD de la variable *Alarm*

3.2.4. Inferencias

Una inferencia es un razonamiento probabilístico que consiste en asignar valores a ciertas variables (evidencia), y se obtiene la probabilidad posterior de las demás variables dadas las variables conocidas.

3.3. Diseño del modelo

En esta sección se explicará el proceso que se siguió para diseñar los diagramas de Bayes. En un principio, tenía pensado diseñar un sólo diagrama común para los usuarios de los jóvenes, familias y profesionales. Sin embargo, el problema era que en la página de SIVARIA habían cuestionarios que eran únicos para cada tipo de usuario. Por lo tanto, las variables que necesitaban cada tipo de usuario eran distintos. Esto provocaba que en un modelo en común hubieran variables que no estuvieran presentes en unos y en otros sí, lo que impedía el entrenamiento de dicho modelo.

Ante esta dificultad, opté por la creación de 3 modelos separados, uno por cada tipo de usuario. De esta manera, las variables estarían bien separadas y diferenciadas de cada una. Así, el entrenamiento sería correcto para todos los modelos. A continuación, se van a mostrar los diagramas resultantes, así como las variables y valores de cada uno. Cabe resaltar que todas las variables y sus valores de los modelos se han basado en la documentación inicial y básica proporcionada por el director del proyecto. A sí mismo, los valores de las variables, además de la documentación, se han escogido teniendo en cuenta las preguntas que de los cuestionarios de los jóvenes, familias y profesionales del sitio web de SIVARIA:

```
https:  
//blogs.uned.es/investigacioninfantoyjuvenil/sivaria-gestion-del-suicidio/
```

A continuación se van a mostrar el diseño inicial de los modelos para los jóvenes (Autoinforme), el de las familias y el de los profesionales. De todas formas, estos modelos están sujetas a cambios, como se podrá observar posteriormente en el entrenamiento de los modelos.

3.3.1. Modelo del Autoinforme

El modelo va a constar de los siguientes grupos de variables:

- Variables de identificación:
 - Curso: (PRIMARIA, ESO, BACHILLERATO, UNIVERSIDAD, FORMACION PROFESIONAL).
- Variables sociodemográficas:

- Sexo asignado: (HOMBRE, MUJER, OTRO).
 - Transgénero: (SÍ, NO, NO ESTOY SEGURO DE SER TRANS, NO ESTOY SEGURO DE LO QUE SE PREGUNTA).
 - Edad: (MENOR DE 12, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21, MAYOR DE 21).
 - Situación laboral padre: (NO TRABAJA, TRABAJA, PENSIONADO).
 - Situación laboral madre: (NO TRABAJA, TRABAJA, PENSIONADO).
 - Nivel profesional padre: (BAJO, MEDIO, ALTO).
 - Nivel profesional madre: (BAJO, MEDIO, ALTO).
 - Nivel promedio del rendimiento académico: (INSUFICIENTE, SUFICIENTE, NOTABLE, SOBRESALIENTE, EXTRAORDINARIO).
 - Nivel de autopercepción masculina: (0, 1, 2, 3, 4, 5, 6).
 - Nivel de autopercepción femenina: (0, 1, 2, 3, 4, 5, 6).
 - Nivel de heteropercepción masculina: (0, 1, 2, 3, 4, 5, 6).
 - Nivel de heteropercepción femenina: (0, 1, 2, 3, 4, 5, 6).
- Variables de salud:
- Altura (cm): (MENOS DE 149, 150-159, 160-169, 170-179, 180-189, MAS DE 190).
 - Peso (kg): (MENOS DE 49, 50-59, 60-69, 70-79, 80-89, MAS DE 90).
 - Tratamiento psiquiátrico previo: (SÍ, NO).
 - Presenta enfermedad crónica: (SÍ, NO).
- Variables de bullying: cuestionario EBIPQ/ECIPQ (*European Bullying Intervention Project Questionnaire*) y ECIPQ (*European Cyberbullying Intervention Project Questionnaire*). Ortega-Ruiz et al. (2016).
- Víctima de bullying: (SÍ, NO).
 - Perpetrador de bullying: (SÍ, NO).
 - Víctima de Cyberbullying: (SÍ, NO).
 - Perpetrador de Cyberbullying: (SÍ, NO).
- Variables de adicción o abuso: cuestionario MULTICAGE CAD-4. Pérez et al. (2007).
- Adicción o abuso alcohol: (SÍ, NO).
 - Adicción o abuso sustancias: (SÍ, NO).
 - Adicción o abuso Internet: (SÍ, NO).
- Psicopatología: cuestionario SENA (*Sistema de Evaluación de Niños y Adolescentes*). Fernández-Pinto et al. (2015)
- Problemas interiorizados: (SÍ, NO).
 - Problemas exteriorizados: (SÍ, NO).
 - Problemas de contexto: (SÍ, NO).
 - Problemas de recursos psicológicos: (SÍ, NO).
- Variables del constructo del comportamiento suicida:

- Percepcion de discriminación: (SÍ, NO).
- Fuente de discriminación: (EDAD, RAZA, DISCAPACIDAD, GENERO, ORIENTACION SEXUAL).
- Variables de regulación y resistencia: cuestionario CERQS (*Cognitive Emotion Regulation Questionnaire*). Garnefski y Kraaij (2006)
 - Nivel de resistencia o resiliencia: (1, 2, 3, 4, 5): cuestionario ER (*Escala de Resiliencia*). Sánchez et al. (2016)
 - Nivel de regulación positiva: (1, 2, 3, 4, 5)
 - Nivel de regulación negativa: (1, 2, 3, 4, 5)
- Variables volitivo-motivacionales para el suicidio
 - Atrapamiento interno: (SÍ, NO): escala ATI. Gilbert y Allan (1998).
 - Atrapamiento externo: (SÍ, NO): escala ATE. Gilbert y Allan (1998).
 - Nivel percibido de derrota o fracaso: (BAJO, MEDIO, ALTO): escala ED (*Escala de Derrota*).
 - Sentimiento de pertenencia frustada: (SÍ, NO): cuestionario INQ-15 (*Interpersonal Needs Questionnaire - 15*). Hill et al. (2015)
 - Percepción de ser una carga: (SÍ, NO): cuestionario INQ-15 (*Interpersonal Needs Questionnaire - 15*). Hill et al. (2015)
 - Autoeficiencia para el suicidio: (SÍ, NO). Basado en el cuestionario FASM.
- Perfil de riesgo psicosocial
 - Madre adolescente: (SÍ, NO)
 - Padre adolescente: (SÍ, NO)
 - Padres divorciados: (SÍ, NO)
 - Familia monoparental: (SÍ, NO)
 - Tratamiento psicológico de padre o madre: (SÍ, NO)
 - Adicción de padre o madre: (SÍ, NO)
 - Relaciones conflictivas del hijo con padre o madre: (SÍ, NO)
 - Familia reconstruida: (SÍ, NO)
- Tecnologías y RRSS
 - Búsqueda información autolesión: (SÍ, NO)
 - Petición de ayuda en Internet: (SÍ, NO)
 - Compartir en RRSS pensamientos sobre autolesión: (SÍ, NO)
 - Realización de autolesión después de ver un contenido en Internet: (SÍ, NO)
 - Conocidos que comparten autolesión en Internet: (SÍ, NO)
 - Contacto con información sobre autolesión: (SÍ, NO)
 - Denuncia a otra persona cometiendo autolesión en Internet: (SÍ, NO)
- Desenlace: (NINGUNO, AUTOLESIÓN, COMUNICACIÓN DE SUICIDIO, DESEO DE SUICIDIO, IDEACIÓN DE SUICIDIO, PLANIFICACIÓN DE SUICIDIO, INTENCIÓN DE SUICIDIO)

Debido al elevado número de variables que hay en el diagrama del Autoinforme, se ha dividido en dos partes para hacer fácilmente visible en el documento. De todas formas, la imagen completa de este diagrama, al igual que los otros dos diagramas se encuentran en el repositorio de Github, dentro de la carpeta *DiagramBayesSketchesv1*.

Los diagramas se encuentran en las Figuras 3.5, 3.6 y 3.7.

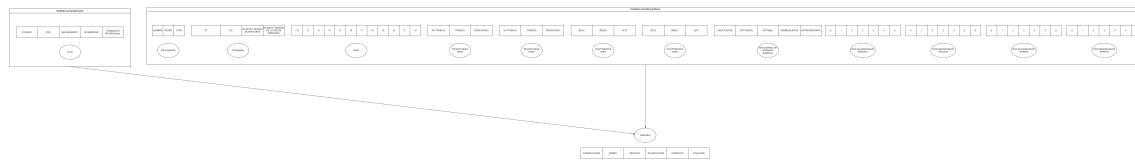


Figura 3.5: Red de Bayes del modelo del Autoinforme (I)

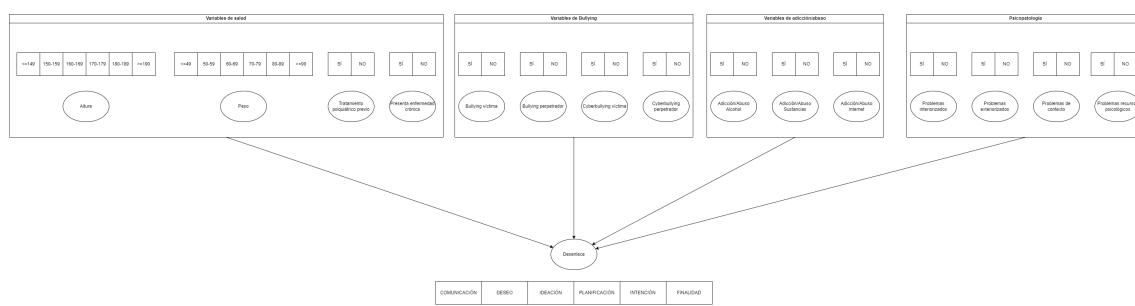


Figura 3.6: Red de Bayes del modelo del Autoinforme (II)

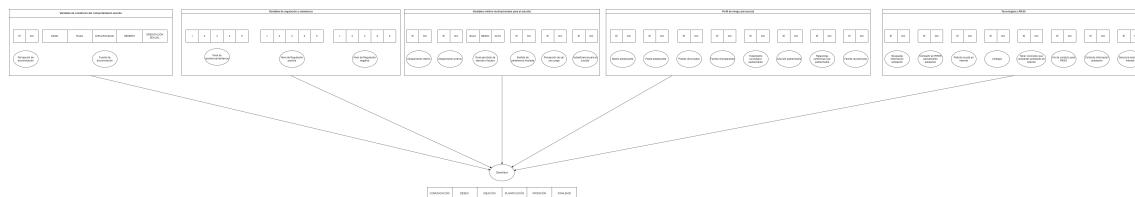


Figura 3.7: Red de Bayes del modelo del Autoinforme (III)

3.3.2. Modelo de las familias

El modelo va a constar de los siguientes grupos de variables con sus posibles valores:

- Perfil de riesgo psicosocial
- Psicopatología
- Variables de las familias: cuestionario PARQ-S (*Child-Parental Acceptance-Rejection Questionnaire - Short*). Del Barrio et al. (2014).
 - Aceptación/Rechazo parental: (ACEPTACIÓN, RECHAZO)
 - Control parental: (SÍ , NO).
- Desenlace

El diagrama se encuentra representado en la Figura 3.8.

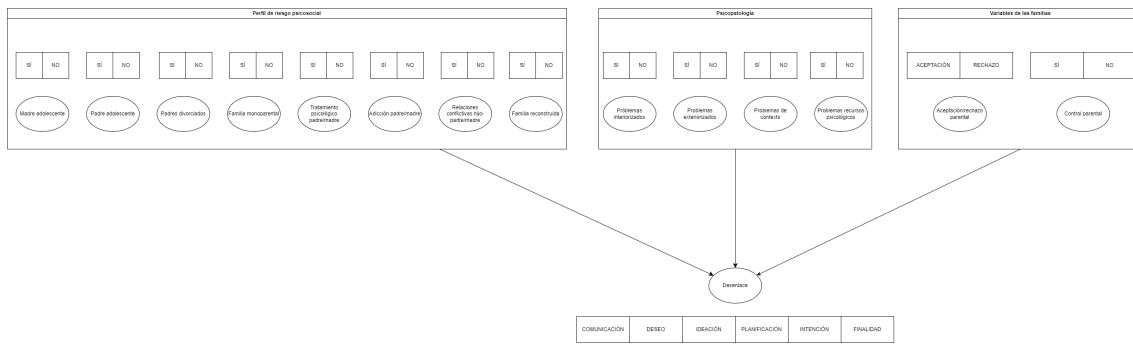


Figura 3.8: Red de Bayes del modelo de las familias

3.3.3. Modelo de los profesionales

El modelo va a constar de los siguientes grupos de variables

- Perfil de riesgo psicosocial
 - Situación económica familiar precaria: (SÍ, NO)
 - Estudios de la madre: (SÍ, NO)
 - Estudios del padre: (SÍ, NO)
 - Supervisión parental insuficiente: (SÍ, NO)
 - Maltrato al adolescente: (SÍ, NO)
 - Duelo: (SÍ, NO)
 - Ingreso familiar mensual: (<=499, 500-999, 1000-1499, 1500-1999, 2000-2499, >=2500)
- Desenlace

El diagrama se encuentra representado en la Figura 3.9.

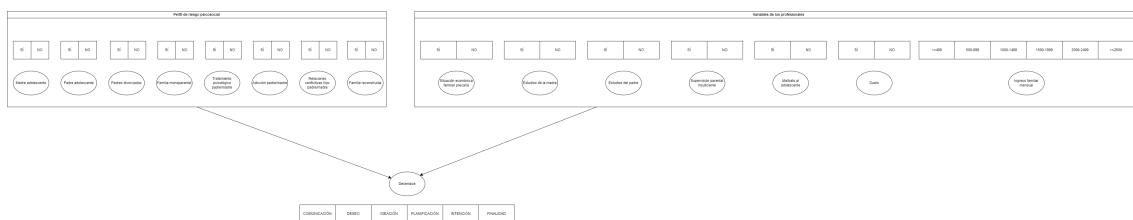


Figura 3.9: Red de Bayes del modelo de los profesionales

3.4. Desarrollo de los modelos

Para el desarrollo de los modelos, se estudiaron qué herramientas y tecnologías serían útiles para la implementación y entrenamiento de estos modelos. Hubieron varias librerías, como pybnesian o pomegranate, pero los descarté por su complejidad a la hora de crear y entrenar una red de Bayes. Finalmente, me decanté por implementar los modelos en

un Python, utilizando una librería llamada pgmpy (Ankan y Panda (2015)), ya que es una librería dedicada específicamente a los diagramas de Bayes. Además, permitía poder especificar las relaciones entre las variables y, por lo tanto, crear redes de Bayes de todo tipo. Sin embargo, lo tuve que descartar debido a que me daba errores de memoria a la hora de crear el diagrama, y las pocas veces que lograba crearse, luego tardaba demasiado tiempo en entrenarlo y guardar el modelo, a menudo llegando incluso a congelarse la ejecución. Esto se debía al tamaño del diagrama, que ocupaba demasiado espacio de memoria. Es por ello, que tuve que buscar otras alternativas. Finalmente, decidí implementarlo con scikit learn (Pedregosa et al. (2011)), ya que, a pesar de que no es una librería específica para crear diagramas de Bayes, sí que contiene una versión de los modelos de Bayes, que es el modelo Naive Bayes.

Naive-Bayes es un modelo de clasificación de aprendizaje automático que asume la independencia de las características para predecir a qué categoría pertenece un conjunto de características o *features*. Este algoritmo utiliza el Teorema de Bayes (3.3) para calcular la clase más probable dado un conjunto de eventos, o más bien, características dadas. Consideré usar esta variante porque asumí que las variables no tienen correlación entre ellas.

De esta manera, el modelo Naive-Bayes de scikit-learn realiza las predicciones a través de una tabla de frecuencia y, posteriormente, una tabla de probabilidad, que se generan durante el entrenamiento. Dichas tablas contiene un conteo de las veces que ha sucedido un desenlace final, así como el conteo de las variables que han provocado ese desenlace.

The diagram illustrates the process of creating a frequency table and a probability table from a dataset. On the left, there is a dataset table with columns 'Outlook' and 'Play'. The rows are numbered 0 to 13. The data shows various weather conditions (Overcast, Rainy, Sunny) and whether people played (Yes or No). In the middle, there is a frequency table for 'Weather' (Overcast, Rainy, Sunny) with counts for 'Yes' and 'No'. The total count for each weather type is also provided. On the right, there is a probability table where the counts from the frequency table are converted into proportions. The total probability for each weather type is also shown.

	Outlook	Play
0	Rainy	Yes
1	Sunny	Yes
2	Overcast	Yes
3	Overcast	Yes
4	Sunny	No
5	Rainy	Yes
6	Sunny	Yes
7	Overcast	Yes
8	Rainy	No
9	Sunny	No
10	Sunny	Yes
11	Rainy	No
12	Overcast	Yes
13	Overcast	Yes

Weather	Yes	No
Overcast	5	0
Rainy	2	2
Sunny	3	2
Total	10	5

Weather	No	Yes	
Overcast	0	5	5/14= 0.35
Rainy	2	2	4/14=0.29
Sunny	2	3	5/14=0.35
All	4/14=0.29	10/14=0.71	

Figura 3.10: Ejemplo de la creación de una tabla de frecuencia (segunda tabla) y tabla de probabilidad (tercera tabla) dado un dataset (primera tabla). Fuente del ejemplo: Sitio web de Javatpoint

Una vez creada la tabla, para realizar las predicciones, el modelo calcula la probabilidad posterior de cada categoría mediante el Teorema de Bayes, asumiendo la independencia de todas las características. Aquella categoría del modelo que obtenga mayor probabilidad de todas, es la que se devuelve finalmente.

3.4.1. Datasets

Antes de comenzar con el desarrollo de los scripts, hubo un inconveniente con respecto a los datasets que se tenían que usar para el entrenamiento. En un principio se iba a contar con una serie de datasets proporcionados por los profesionales de la salud mental que estaban involucrados en este proyecto. Sin embargo, dichos datasets finalmente no fueron proporcionados. Por lo tanto, ante esta falta importante de datos para tomarlos de referencia, tuve que buscar otras alternativas que se podrán ver a lo largo de las iteraciones de los entrenamientos. En un primer lugar, opté por generar yo mismo mis propios datasets de forma artificial y aleatoria, creándome un script de Python para ello. Sin embargo, debido a esta aleatoriedad de los datos, estos en su mayoría eran poco realistas y con potenciales incoherencias. Como consecuencia, el entrenamiento se veía muy perjudicado, y por consiguiente, las predicciones y los resultados obtenidos de la evaluación del modelo no serían del todo buenas.

Es por ello, que a partir de la segunda iteración, opté por buscar una alternativa. Finalmente, me decanté por tratar de generar estos datasets de entrenamiento mediante el uso de un LLM (*Large Language Model*). Utilicé ChatGPT para generar los datos, más concretamente, el modelo GPT-4o, ya que era la versión más moderna y veloz que podía ofrecer OpenAI a día de hoy.

Cabe resaltar que tanto, los primeros datasets generados por el script como los siguientes, generados por el LLM, también están a disposición pública en el repositorio de Github. Se ubicarán dentro de la carpeta *scripts/datasets*. Estarán distribuidos dependiendo de cada versión del entrenamiento y del tipo de modelo que se pretende entrenar. Del mismo modo, todas las versiones de los entrenamientos realizados se ubican en cuadernos de Jupyter, también dentro de la carpeta *scripts*.

De esta manera, por ejemplo, si se quiere llevar a cabo el entrenamiento del dataset de los familiares de la primera versión, habría que acceder al siguiente cuaderno de Jupyter: *scripts/first_version_model_training_scikit_learn.ipynb*.

Y si se quiere acceder al contenido de dicho dataset de las familias para saber que dataset se pueden utilizar, se puede acceder al siguiente path: *scripts/datasets/familia/dataset1.csv*.

3.4.2. Procesos de entrenamiento y testeo del modelo

El entrenamiento se dividirá en versiones, en el que cada una contará con una serie de iteraciones. Esta división de versiones se hizo cada vez que había un cambio en el diseño del modelo o en la forma de creación de los datasets (generados por scripts o por el LLM). Su proceso de entrenamiento en todas las versiones será de la siguiente manera:

1. Los datos procedentes del dataset se dividirán en dos subconjuntos: los datos de entrenamiento y los datos de prueba. El porcentaje de los datos que le corresponderán a cada uno, es decir, el balance, se irá variando a lo largo de las iteraciones.
2. Se crea el modelo y se le entrena con los datos de entrenamiento. Al principio, el modelo se creará con los hiperparámetros por defecto, aunque es recomendable que posteriormente se haga un ajuste de hiperparámetros para encontrar los valores más adecuados para obtener un mejor entrenamiento.
3. Se realiza el testeo del modelo, introduciendo los datos de prueba para obtener una predicción.

4. Se genera la matriz de confusión normalizada, comparando los resultados obtenidos de la predicción con los resultados reales.
5. Se calcula el número de verdaderos y falsos positivos y negativos, para evaluar de forma numérica el número de fallos y aciertos.
6. Se calculan las métricas *accuracy*, *precision*, *recall* y *f1_score* para obtener la puntuación de rendimiento del modelo.
7. De manera adicional, se valida el modelo calculando el error medio de los datos del dataset mediante la validación cruzada.
8. Se guarda el modelo en un fichero .SAV que se ubicará en la carpeta correspondiente del *scripts configFiles*. Dentro habrá una carpeta por cada tipo de modelo: autoinforme, familia y profesional. El formato del nombre del fichero de guardado será el siguiente (Figura 3.11):

model_<TIPO_DE_MODELO>_<VERSION>_<FECHA_Y_HORA_DE_CREACION>.sav.

De esta manera, se consigue que haya un histórico de todos los entrenamientos del modelo, permitiendo así un control de versiones y pudiendo borrar la versión del modelo que se considere necesario.

3.4.2.1. Clasificadores utilizados

A lo largo de las versiones del entrenamiento, se han utilizado distintos tipos de clasificadores Naive-Bayes de la librería *scikit-learn*, dependiendo de la naturaleza de los datos y su distribución en los datasets. Estos son:

- **Clasificador gausiano (*GaussianNB*):** en este clasificador, se asume que los datos siguen una distribución normal y que son características continuas, es decir, variables que no tienen un conjunto finito de posibles valores. Por ejemplo, una variable de llamado *Víctima de bullying* = {Sí, No} es una variable discreta, porque su conjunto de posibles valores es finito, mientras que una variable *Peso* = {50, 50.5, 51, 51.5, 52, 53, ...} es una variable continua, porque la variable puede tener infinitos valores. Por lo tanto, la probabilidad en este clasificador, se calcula internamente de la siguiente manera:

$$P(x_i|y) = \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{(x-\mu)^2}{2\sigma^2}} \quad (3.6)$$

Donde:

- x_i es la categoría posterior.
- y es el conjunto de condiciones para que se dé la categoría.
- μ la media de la sumatoria de todos los valores w_i de esa columna.

$$\mu = \frac{1}{n} \sum_{i=1}^n w_i \quad (3.7)$$

- σ es la desviación estándar, cuyo cálculo es el siguiente:

$$\sigma = \sqrt{\frac{1}{n-1} \sum_{i=1}^n (w_i - \mu)^2} \quad (3.8)$$

- e es el constante de Euler.

- **Clasificador multinomial** (*MultinomialNB*): el clasificador asume que los datos siguen una distribución multinomial, es decir, que cada celda del fichero representa la frecuencia con la que se repite el valor de esa columna para cada fila del dataset. Por lo tanto, sus datos son numéricos. Puede ser de utilidad con datasets que tengan características discretas.
- **Clasificador categórico** (*CategoricalNB*): en este caso, se asume que los datos siguen una distribución categórica, en donde todas las características son variables discretas con valores fijos. En un dataset, cada fila representa un valor en concreto, a diferencia del clasificador multinomial, que cada columna de una fila guarda un conteo numérico. que provocan un desenlace. Cabe destacar que existe una variante de este clasificador, que es el Bernoulli (*BernoulliNB*). Sin embargo, es necesario que todos los datos tengan que ser binarios, es decir, que cada columna del dataset sólo puede tener dos valores: (Sí, No), (0,1), etc. Como en nuestro diagrama hay algunas variables que tienen más de dos valores, entonces no se puede utilizar.

3.4.2.2. Aprendizaje supervisado

El aprendizaje supervisado o *machine learning supervisado* es una subcategoría de la Inteligencia Artificial. Se caracteriza por el uso de conjuntos de datos etiquetados para que el modelo pueda detectar patrones dentro del entrenamiento, y así realizar predicciones con alta precisión. Se ha decidido por este tipo de aprendizaje dentro de scikit learn debido a la naturaleza de los datasets, ya que los datos que contienen ya vienen etiquetados, tanto los de entrada como los de salida. Dentro de la librería scikit learn se ofrecen diversos algoritmos de clasificación que permiten predecir un resultado. Aún así, tiene la desventaja de que requiere realizar modificaciones manuales dentro del dataset para categorizar los datos o corregir datos que no estén categorizados.

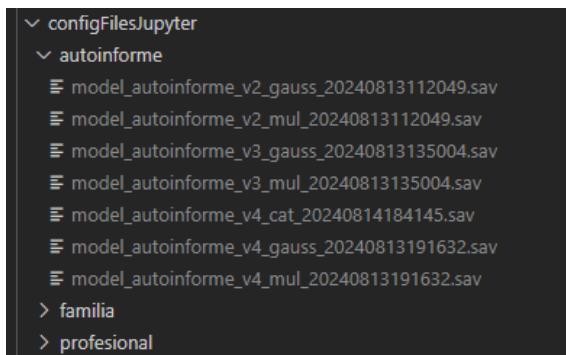


Figura 3.11: Ejemplo del directorio de archivos de guardado con el modelo del Autoinforme

3.4.2.3. Testeo del entrenamiento del modelo

Para medir el rendimiento de nuestro modelo, se debe comparar las respuestas predichas con las respuestas reales del conjunto de testeo.

Esto nos permitirá obtener la cantidad de verdaderos y falsos positivos y verdaderos y falsos negativos que ha habido, que se usarán para calcular una puntuación que indicará el ratio de acierto de las predicciones del modelo, comprendido entre 0 y 1.

Para calcular esta puntuación, existen cuatro tipos de métricas:

- **Exactitud (Accuracy)**: mide la proporción de predicciones correctas (tanto verdaderos positivos como verdaderos negativos) entre el total de predicciones realizadas.

$$\text{Exactitud} = \frac{TP + TN}{TP + TN + FP + FN} \quad (3.9)$$

Donde:

- TP: Verdaderos Positivos (*True Positives*)
- TN: Verdaderos Negativos (*True Negatives*)
- FP: Falsos Positivos (*False Positives*)
- FN: Falsos Negativos (*False Negatives*)

- **Precisión**: mide la proporción de verdaderos positivos entre todas las predicciones positivas hechas por el modelo. Es decir, de todas las veces que el modelo predijo que una instancia es positiva, cuántas veces acertó.

$$\text{Precision} = \frac{TP}{TP + FP} \quad (3.10)$$

- **Sensibilidad (Recall)**: mide la proporción de verdaderos positivos entre todas las instancias que son realmente positivas. Es decir, de todas las veces que una instancia es realmente positiva, cuántas veces el modelo la detectó correctamente.

$$\text{Sensibilidad} = \frac{TP}{TP + FN} \quad (3.11)$$

- **F1 Score**: es la media armónica entre la precisión y la sensibilidad. Es bastante útil en contextos donde se necesita un equilibrio entre la precisión y la sensibilidad, o cuando hay una distribución desequilibrada de clases.

$$F1 = 2 \cdot \frac{\text{Precision} \cdot \text{Sensibilidad}}{\text{Precision} + \text{Sensibilidad}} = \frac{2 \cdot TP}{2 \cdot TP + FP + FN} \quad (3.12)$$

El cálculo de todas estas métricas ya vienen incluido dentro del módulo scikit-learn (*sklearn.metrics*), así que no hace falta calcularlos todos de forma manual. Dentro de la función de cálculo de las métricas existen predominan dos formas de calcular las métricas:

- *micro*: con este enfoque, se suman todos los verdaderos y falsos positivos y negativos a nivel global, sin ponderar el la cantidad que haya en cada clase. Esto puede provocar que los valores de todas las métricas coincidan.

$$\text{Precision}_{\text{micro}} = \frac{\sum TP}{\sum(TP + FP)} \quad (3.13)$$

$$Recall_{micro} = \frac{\sum TP}{\sum(TP + FN)} \quad (3.14)$$

$$F1_{micro} = 2 \cdot \frac{Precision_{micro} \cdot Recall_{micro}}{Precision_{micro} + Recall_{micro}} \quad (3.15)$$

- *weighted*: en este caso, las métricas se calculan para cada clase individualmente, y luego se ponderan con respecto a un soporte (número de verdaderos ejemplos en cada clase). Esto significa que las métricas de clases más grandes tienen más influencia en el promedio final.

$$Precision_{weighted} = \sum \left(\frac{Soporte_i}{Soportetotal} \cdot Precision_i \right) \quad (3.16)$$

$$Recall_{weighted} = \sum \left(\frac{Soporte_i}{Soportetotal} \cdot Recall_i \right) \quad (3.17)$$

$$F1_{weighted} = \sum \left(\frac{Soporte_i}{Soportetotal} \cdot F1_i \right) \quad (3.18)$$

Ambos valores se establecen en el parámetro *average* de cada métrica.

3.4.2.4. Validación cruzada

Una forma de poder validar el entrenamiento de un modelo es usando la validaciónn cruzada de la librería scikit learn. La validación cruzada es una técnica para asegurar que el modelo no solo se ajusta a los datos de entrenamiento, sino también a los datos no vistos, evitando así una situación de **overfitting**, es decir, la posibilidad de que el modelo sólo puede predecir datos que ya ha visto. Consiste en dividir el conjunto de datos en varios subconjuntos o *folds*, entrenar al modelo con alguno de estos subconjuntos y evaluarlo con los otros restantes. Este proceso se repite varias veces para obtener un ratio con mayor robustez en el rendimiento del modelo.

Este método también se utilizará para calcular el el margen de error de los entrenamientos.

3.5. Versiones del entrenamiento

En esta sección se llevará un registro de los entrenamientos realizados en las distintas versiones. Figuran 4 versiones de entrenamiento.

3.5.1. Primera versión: datasets artificiales y aleatorios

En esta primera versión empecé con un entrenamiento simple, utilizando los datasets generados de forma artificial a través de un script de Python. Se generaron 5 datasets, con 10000 filas cada una, por cada tipo de modelo.

El código del entrenamiento se encuentra en el cuaderno de Jupyter:

scripts/first_version_model_training_scikit_learn.ipynb

Durante esta primera versión, se empezó utilizando el clasificador gausiano para realizar las predicciones, debido a que es el principal tipo de clasificación del modelo Naive-Bayes

de *scikit-learn*, además de que dentro del dataset se encontraban datos que consideré continuos, como la edad, el peso o la altura. Sin embargo, como había otras variables que eran discretas, entonces se tuvo que estandarizar todo el dataset. Es por ello que, internamente, se codifican los datos recibidos para que todos sean de tipo numérico, y así se pueda aplicar el modelo gausiano.

3.5.1.1. Iteración 1

En esta primera iteración se realizó con los siguientes parámetros:

- **Clasificador:** GaussianNB.
- **Dataset:** *scripts/datasets*.
- **Balance del dataset:** 75 % - 25 %.
- **average:** micro.

En base a estos parámetros, se obtuvieron las siguientes métricas para cada tipo de modelo:

Modelo	Métrica	D1	D2	D3	D4	D5	Media
Autoinforme	Accuracy	0.182	0.160	0.166	0.178	0.174	0.172
	Precision	0.182	0.160	0.166	0.178	0.174	0.172
	Recall	0.182	0.160	0.166	0.178	0.174	0.172
	F1 Score	0.182	0.160	0.166	0.178	0.174	0.172
Familia	Accuracy	0.174	0.168	0.175	0.176	0.182	0.175
	Precision	0.174	0.168	0.175	0.176	0.182	0.175
	Recall	0.174	0.168	0.175	0.176	0.182	0.175
	F1 Score	0.174	0.168	0.175	0.176	0.182	0.175
Profesional	Accuracy	0.166	0.176	0.159	0.153	0.171	0.165
	Precision	0.166	0.176	0.159	0.153	0.171	0.165
	Recall	0.166	0.176	0.159	0.153	0.171	0.165
	F1 Score	0.166	0.176	0.159	0.153	0.171	0.165

Tabla 3.2: Comparativa de las métricas del modelo. Iteración 1

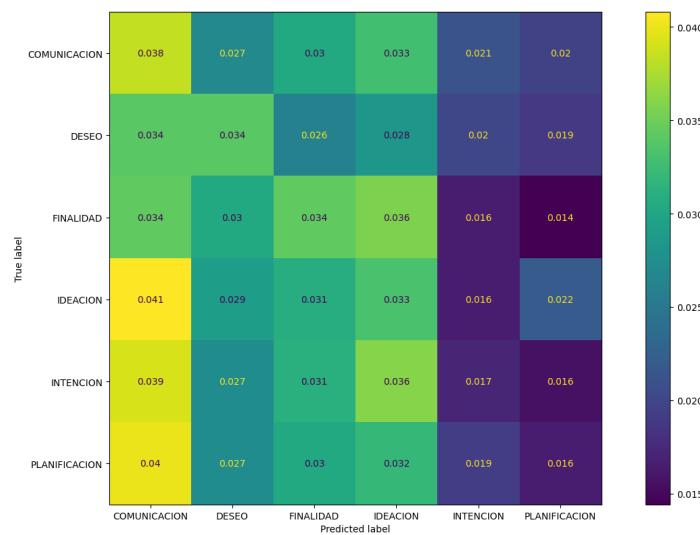


Figura 3.12: Matriz de confusión del Autoinforme al final del primer ciclo



Figura 3.13: Matriz de confusión de las familias al final del primer ciclo

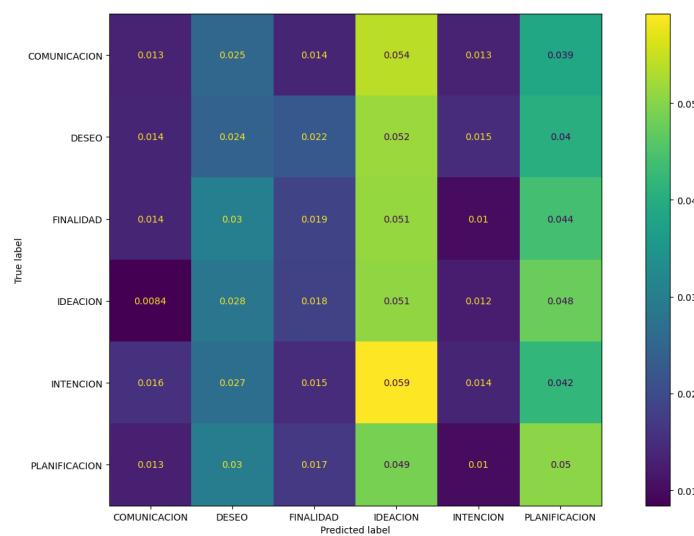


Figura 3.14: Matriz de confusión de los profesionales al final del primer ciclo

3.5.1.2. Iteración 2

Viendo que los resultados fueron bajos en la primera iteración, se consideró que se podía deber a un cambio en el cálculo de las métricas. Es por eso que en esta segunda iteración de entrenamiento se realizó un cambio en el cálculo del average de las métricas. En vez de usar la opción *micro*, se va a probar con la opción *weighted*⁶, para darle más peso a las clases con mayor número de aciertos.

- **Clasificador:** GaussianNB.
- **Dataset:** *scripts/datasets*.
- **Balance del dataset:** 75 % - 25 %.
- **average:** weighted.

Los resultados fueron los siguientes:

Modelo	Métrica	D1	D2	D3	D4	D5	Media
Autoinforme	Accuracy	0.182	0.160	0.166	0.178	0.74	0.172
	Precision	0.182	0.160	0.163	0.178	0.171	0.171
	Recall	0.182	0.160	0.166	0.178	0.174	0.172
	F1 Score	0.182	0.157	0.160	0.174	0.170	0.169
Familia	Accuracy	0.174	0.168	0.175	0.176	0.182	0.175
	Precision	0.175	0.172	0.173	0.175	0.181	0.175
	Recall	0.174	0.168	0.175	0.176	0.182	0.175
	F1 Score	0.173	0.159	0.168	0.172	0.179	0.170
Profesional	Accuracy	0.166	0.176	0.159	0.153	0.171	0.165
	Precision	0.153	0.179	0.161	0.159	0.173	0.165
	Recall	0.166	0.176	0.159	0.153	0.171	0.165
	F1 Score	0.130	0.163	0.147	0.148	0.159	0.149

Tabla 3.3: Comparativa de las métricas del modelo. Iteración 2

3.5.1.3. Iteración 3

En este tercer ciclo de entrenamiento, se ha decidido modificar el porcentaje de división del dataset. En este escenario, el 70 % del dataset será utilizado en el entrenamiento, y el 30 % restante para testeo. Esto se ha hecho con el objetivo de darle más datos para poder testear, y así poder obtener resultados mejores. Como el balance del dataset ha cambiado, los resultados del entrenamiento van a variar, por lo que las matrices de confusión también serán diferentes. Por otro lado, el valor del parámetro *average* para el cálculo de las métricas vuelve a ser *micro*.

- **Clasificador:** GaussianNB.
- **Dataset:** *scripts/datasets*.

⁶Cabe resaltar que este parámetro influye sobre las métricas durante el testeo, pero no sobre el entrenamiento. Esto implica que los resultados del entrenamiento serán los mismos, y por lo tanto, las matrices de confusión no varían.

- **Balance del dataset:** 70 % - 30 %.
- ***average:*** micro.

Los resultados fueron los siguientes:

Modelo	Métrica	D1	D2	D3	D4	D5	Media
Autoinforme	Accuracy	0.183	0.168	0.160	0.178	0.169	0.172
	Precision	0.183	0.168	0.160	0.178	0.169	0.172
	Recall	0.183	0.168	0.160	0.178	0.169	0.172
	F1 Score	0.183	0.168	0.160	0.178	0.169	0.172
Familia	Accuracy	0.172	0.165	0.166	0.175	0.183	0.172
	Precision	0.172	0.165	0.166	0.175	0.183	0.172
	Recall	0.172	0.165	0.166	0.175	0.183	0.172
	F1 Score	0.172	0.165	0.166	0.175	0.183	0.172
Profesional	Accuracy	0.164	0.169	0.169	0.159	0.174	0.167
	Precision	0.164	0.169	0.169	0.159	0.174	0.167
	Recall	0.164	0.169	0.169	0.159	0.174	0.167
	F1 Score	0.164	0.169	0.169	0.159	0.174	0.167

Tabla 3.4: Comparativa de las métricas del modelo. Iteración 3



Figura 3.15: Matriz de confusión del Autoinforme al final del tercer ciclo

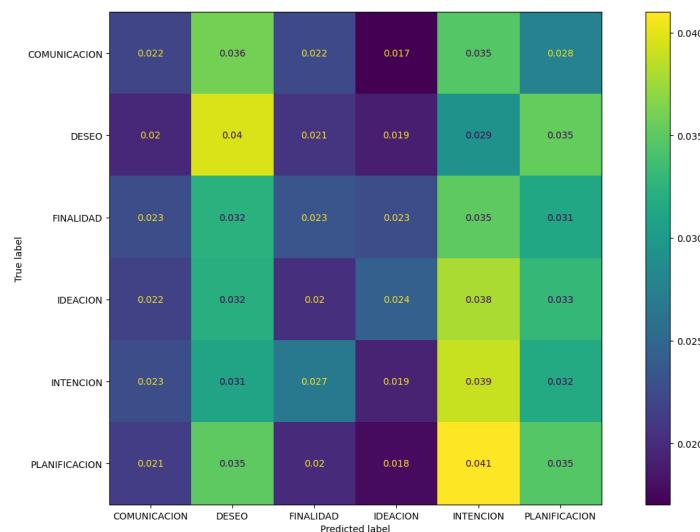


Figura 3.16: Matriz de confusión de las familias al final del tercer ciclo

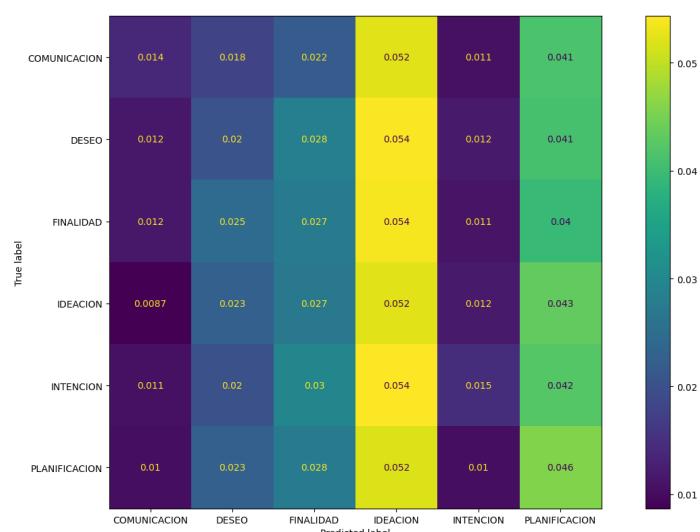


Figura 3.17: Matriz de confusión de los profesionales al final del tercer ciclo

3.5.1.4. Iteración 4

- **Clasificador:** GaussianNB.
- **Dataset:** *scripts/datasets*.
- **Balance del dataset:** 70 % - 30 %.
- ***average:*** weighted.

Los resultados fueron los siguientes:

Modelo	Métrica	D1	D2	D3	D4	D5	Media
Autoinforme	Accuracy	0.183	0.168	0.160	0.178	0.169	0.172
	Precision	0.181	0.167	0.156	0.180	0.166	0.170
	Recall	0.183	0.168	0.160	0.178	0.169	0.172
	F1 Score	0.181	0.164	0.154	0.176	0.166	0.168
Familia	Accuracy	0.172	0.165	0.166	0.175	0.183	0.172
	Precision	0.172	0.164	0.164	0.176	0.183	0.172
	Recall	0.172	0.165	0.166	0.175	0.183	0.172
	F1 Score	0.171	0.155	0.159	0.171	0.181	0.167
Profesional	Accuracy	0.164	0.166	0.169	0.159	0.174	0.166
	Precision	0.144	0.176	0.172	0.156	0.181	0.166
	Recall	0.164	0.166	0.169	0.159	0.174	0.166
	F1 Score	0.132	0.155	0.159	0.154	0.164	0.153

Tabla 3.5: Comparativa de las métricas del modelo. Iteración 4

3.5.1.5. Iteración 5

En este ciclo, se prueba una diferente división del dataset. Esta vez la división será un 80 % para entrenamiento y un 20 % para testeo. Esto implica un cambio en el rendimiento del entrenamiento.

- **Clasificador:** GaussianNB.
- **Dataset:** *scripts/datasets*.
- **Balance del dataset:** 80 % - 20 %.
- ***average:*** micro.

Los resultados fueron los siguientes:

Modelo	Métrica	D1	D2	D3	D4	D5	Media
Autoinforme	Accuracy	0.182	0.164	0.168	0.180	0.177	0.174
	Precision	0.182	0.164	0.168	0.180	0.177	0.174
	Recall	0.182	0.164	0.168	0.180	0.177	0.174
	F1 Score	0.182	0.164	0.168	0.180	0.177	0.174
Familia	Accuracy	0.161	0.168	0.171	0.172	0.188	0.172
	Precision	0.161	0.168	0.171	0.172	0.188	0.172
	Recall	0.161	0.168	0.171	0.172	0.188	0.172
	F1 Score	0.161	0.168	0.171	0.172	0.188	0.172
Profesional	Accuracy	0.168	0.186	0.163	0.159	0.174	0.170
	Precision	0.168	0.186	0.163	0.159	0.174	0.170
	Recall	0.168	0.186	0.163	0.159	0.174	0.170
	F1 Score	0.168	0.186	0.163	0.159	0.174	0.170

Tabla 3.6: Comparativa de las métricas del modelo. Iteración 5

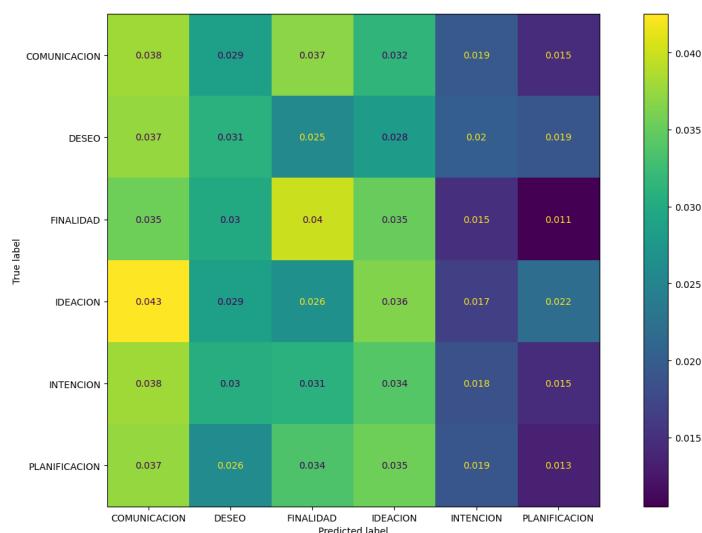


Figura 3.18: Matriz de confusión del Autoinforme al final del quinto ciclo

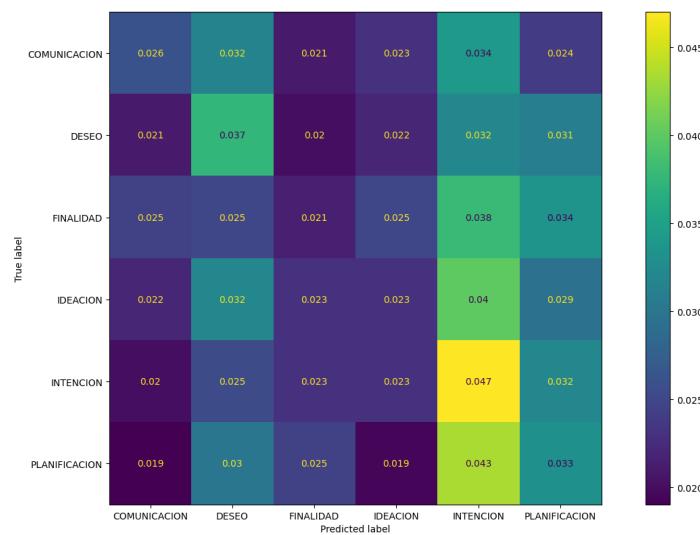


Figura 3.19: Matriz de confusión de las familias al final del quinto ciclo

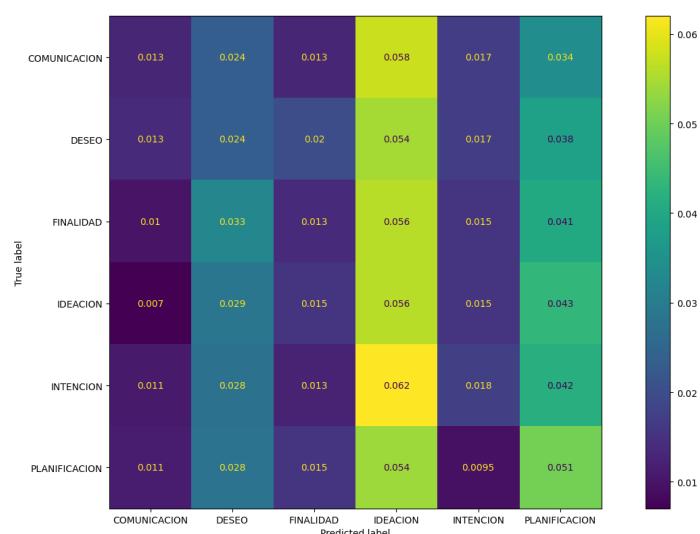


Figura 3.20: Matriz de confusión de los profesionales al final del quinto ciclo

3.5.1.6. Iteración 6

Finalmente, en esta iteración el balance se mantiene con respecto al ciclo anterior, con la diferencia de que el parámetro de cálculo de las métricas se vuelve a modificar de *micro* a *weighted*.

- **Clasificador:** GaussianNB.
- **Dataset:** *scripts/datasets*.
- **Balance del dataset:** 80 % - 20 %.
- ***average*:** weighted.

Los resultados fueron los siguientes:

Modelo	Métrica	D1	D2	D3	D4	D5	Media
Autoinforme	Accuracy	0.182	0.164	0.168	0.180	0.177	0.174
	Precision	0.180	0.165	0.168	0.179	0.174	0.173
	Recall	0.182	0.164	0.168	0.180	0.177	0.174
	F1 Score	0.179	0.162	0.162	0.177	0.172	0.170
Familia	Accuracy	0.159	0.168	0.175	0.175	0.185	0.172
	Precision	0.157	0.171	0.178	0.175	0.179	0.172
	Recall	0.159	0.168	0.175	0.175	0.185	0.172
	F1 Score	0.155	0.159	0.168	0.170	0.177	0.166
Profesional	Accuracy	0.168	0.186	0.163	0.159	0.174	0.170
	Precision	0.152	0.185	0.168	0.160	0.185	0.168
	Recall	0.168	0.186	0.163	0.159	0.174	0.170
	F1 Score	0.148	0.174	0.151	0.150	0.160	0.157

Tabla 3.7: Comparativa de las métricas del modelo. Iteración 6

3.5.1.7. Análisis inicial de los resultados

En esta sección, analizamos los resultados obtenidos de todas las iteraciones de entrenamiento, y se decide qué tipo de métrica se va a utilizar para el cálculo de la fiabilidad del Sistema Experto, el tipo de parámetro *average* (*micro* o *weighted*) y el porcentaje en la división del dataset en el futuro. Para llevar a cabo esta tarea, realicé una tabla comparativa de las medias aritméticas de las métricas de cada iteración para cada tipo de modelo (Autoinforme, familia y profesional). Se puede ver en la siguiente Tabla 3.8.

Analizando la tabla, concluí que la iteración que generalmente ofrece mejores resultados en los modelos es la quinta. El motivo principal es que en líneas generales, es el que devuelve un ratio más balanceado para cada uno de los modelos. El resto de iteraciones obtienen un ratio ligeramente alto para un modelo, pero luego flaquean en el resto.

- En el modelo del Autoinforme, la mejor iteración es la quinta, ya que las métricas *accuracy* y *recall* ofrecen un mejora de un 0.002 con respecto al resto de iteraciones. En la *precision*, la diferencia está entre un 0.002 y un 0.004. Finalmente, con la métrica *F1 Score*, la mejora es de entre un 0.002 y 0.008 con respecto al resto.

Iteraciones	Modelo	Accuracy	Precision	Recall	F1 Score
Iteración 1	Autoinforme	0.172	0.172	0.172	0.172
	Familia	0.175	0.175	0.175	0.175
	Profesional	0.165	0.165	0.165	0.165
Iteración 2	Autoinforme	0.172	0.171	0.172	0.169
	Familia	0.175	0.175	0.175	0.170
	Profesional	0.165	0.165	0.165	0.149
Iteración 3	Autoinforme	0.172	0.172	0.172	0.172
	Familia	0.172	0.172	0.172	0.172
	Profesional	0.167	0.167	0.167	0.167
Iteración 4	Autoinforme	0.172	0.170	0.172	0.168
	Familia	0.172	0.172	0.172	0.167
	Profesional	0.166	0.166	0.166	0.153
Iteración 5	Autoinforme	0.174	0.174	0.174	0.174
	Familia	0.172	0.172	0.172	0.172
	Profesional	0.170	0.170	0.170	0.170
Iteración 6	Autoinforme	0.172	0.172	0.172	0.166
	Familia	0.172	0.172	0.172	0.166
	Profesional	0.170	0.168	0.170	0.157

Tabla 3.8: Tabla comparativa de las métricas. Primera Versión

- En el modelo de las familias, la iteración 1 es la que ofrece el mejor rendimiento, ya que ofrece un ratio de 0.175 en todas las métricas, con una diferencia del 0.003 con respecto al resto de iteraciones. También logra una diferencia del 0.005 con respecto a la iteración 2 en el *F1 Score*, haciendo que sea el mejor escenario para los modelos de las familias. En la iteración 1, las características fue una división del dataset del 75 (Entrenamiento) - 25 (Testeo), y el uso del valor *micro* en el parámetro *average* en el cálculo de las métricas.
- En el modelo de los profesionales, la tabla muestra que la iteración que ofrece mejor rendimiento en todas las métricas es la quinta, ya que ofrece un ratio de 0.170 en todas las métricas, con una diferencia de entre 0.002 y 0.005 con respecto al resto de iteraciones. Además, si se compara con la iteración 6, que es la segunda mejor iteración, vemos que logra una diferencia del 0.002 en la métrica *precision* y un 0.13 con el *F1 Score*.

3.5.1.8. Elección de la métrica y los parámetros

Con respecto a la elección de la métrica, en general, la que ofrece mejores resultados y más consistentes, en todas las iteraciones es el *accuracy*, a diferencia de las otras, que varían dependiendo de la división del dataset o por la forma en la se calcula el promedio de los verdaderos/falsos positivos y verdaderos/falsos negativos. Por consiguiente, el *accuracy* será la métrica utilizada para determinar la fiabilidad del Sistema Experto.

Por otro lado, se puede observar que la iteración 5 es el que ofrece, en general, el *accuracy* más elevado en las tablas del Autoinforme y el de los profesionales. En el caso del Autoinforme, la mejora de la métrica con respecto al resto es del 0.002 de manera uniforme. Mientras que en el caso de los profesionales, la mejora suele ser de entre un 0.003 y 0.005 en comparación con el resto de iteraciones. Considero que estas diferencias

compensan la ligera pérdida de rendimiento en el modelo de la familia, ya que en este escenario, la iteración 1 da mejores resultados que la iteración 5.

Recordando las características de la iteración 5, se utiliza el valor *micro* en el parámetro *average* y se utiliza una división del 80 % (Entrenamiento) - 20 % (Testeo).

3.5.1.9. Conclusión final de la primera versión del Sistema Experto

Teniendo en cuenta los resultados obtenidos y el análisis posterior, se puede concluir que se ha obtenido una fiabilidad del 0.174 (17.4 %) en las predicciones en el modelo del autoinforme, 0.172 (17.2 %) en el modelo de las familias, y un 0.172 (17.2 %) en el modelo de los profesionales.

Realizamos una media aritmética de las tres mediciones de fiabilidad F :

$$F_{SistemaExperto} = \frac{F_{autoinforme} + F_{familias} + F_{profesionales}}{NúmeroModelos} = \frac{0,174 + 0,172 + 0,172}{3} \approx 0,1733$$

Por ende, podemos afirmar que hemos logrado crear un Sistema Experto con una fiabilidad general del 0.173 (17.3 %).

Se ha podido observar que el resultado no ha sido demasiado elevado. Esto puede deberse principalmente a dos motivos:

1. **La naturaleza de los datasets.** Cómo bien se explicó en la sección 3.4.1, los datasets utilizados para el entrenamiento de los modelos se generaron de manera aleatoria y artificial, debido a que los originales no fueron proporcionados por los profesionales involucrados. Esto afectó negativamente a los procesos de entrenamiento, predicción y evaluación.
2. **La estructura de los modelos bayesianos.** Los modelos se pueden refinar más, aumentando o reduciendo el número de variables, especificando mejor las existentes.

3.5.2. Segunda versión: nuevos datasets con LLM

Viendo los resultados presentados en la primera versión, traté de buscar otra forma de generar datasets más coherentes y realistas. Es por ello, que me decanté por utilizar el modelo GPT-4o del ChatGPT para generarlos. Cabe resaltar que también se intentó utilizar Gemini, la IA de Google, pero presentaba varias limitaciones. Las respuestas que generaban eran muy limitadas, los datos generados carecían de variedad y coherencia, y no permitía generar suficientes filas de datos como para formar un dataset.

En esta versión, decidí cambiar el enfoque del entrenamiento. A diferencia de la anterior versión, donde en cada iteración se entrenaban simultáneamente los tres modelos con 5 datasets distintos para cada uno, a partir de esta nueva versión, se entrenaría cada modelo de manera individual en cada iteración hasta conseguir un resultado lo suficientemente bueno como para pasar al siguiente modelo. Mediante esta estrategia, se puede enfocar con mayor facilidad el entrenamiento del modelo en concreto, y así poder identificar más errores. Por otro lado, también se hizo ciertas modificaciones en el diseño de los jóvenes para especificar ciertos valores de las variables. El nuevo contenido del dataset se encuentra en la Figura B.1.

Además, se ha añadido un nuevo clasificador además del gausiano, que es el modelo multinomial. El clasificador multinomial es otro tipo de modelo Naive-Bayes, también presente en *scikit-learn*, que es utilizado en contextos donde los datos sean discretos, en

lugar de continuos, en los que encajaría mejor el clasificador GaussianNB. En este nuevo clasificador, los datos deben seguir una distribución multinomial, es decir, que los datos representan la frecuencia con la que se repiten esa característica.

El script del entrenamiento corresponde con el cuaderno de Jupyter:

```
scripts/second_version_model_training_scikit_learn.ipynb
```

3.5.2.1. Iteración 7

Comenzamos esta segunda versión del entrenamiento con el modelo de los jóvenes, es decir, el del Autoinforme. En esta iteración se va a utilizar un primer dataset de prueba de 40 filas con datos de los jóvenes. Además, se ha añadido un nuevo tipo de clasificador para las pruebas del modelo de los jóvenes, que es el clasificador multinomial. Este nuevo tipo de clasificación, cuyo nombre en la librería *scikit learn* es *MultinomialNB*, realiza las predicciones asumiendo que los datos de entrenamiento siguen una distribución multinomial.

Se utilizarán los siguientes parámetros para iniciar esta nueva versión del entrenamiento.

- **Modelo:** Autoinforme.
- **Clasificador:** GaussianNB, MultinomialNB.
- **Dataset:** *scripts/datasets/autoinforme/v2/dataset_v2_1.csv* - 40 filas.
- **Balance del dataset:** 70 % - 30 %.
- **average:** micro.

En este caso se devolvió los siguientes resultados con la siguiente matriz de confusión (Figura 3.21):

	GaussianNB	MultinomialNB
Accuracy	0.583	0.417
Precision	0.583	0.417
Recall	0.583	0.417
F1	0.583	0.417
Error medio	0.122	0.0

Tabla 3.9: Resultados de las métricas de la iteración 7

Se puede observar que se ha aumentado el rendimiento del entrenamiento con este primer dataset generado. Sin embargo, el error medio en el modelo gausiano es relativamente alto, y en el modelo multinómico es directamente 0. Esto puede deberse a los pocos datos que hay en el dataset, por lo que las predicciones son más dispares y erróneas. De hecho, gran parte del aumento de la métrica se debe a los aciertos en el desenlace de NINGUNO en ambas matrices.

3.5.2.2. Iteración 8

Ante los problemas de la iteración anterior, se optó por incrementar el tamaño del dataset previo, pidiendo al LLM que genere más filas para el dataset, se le pidió que generé

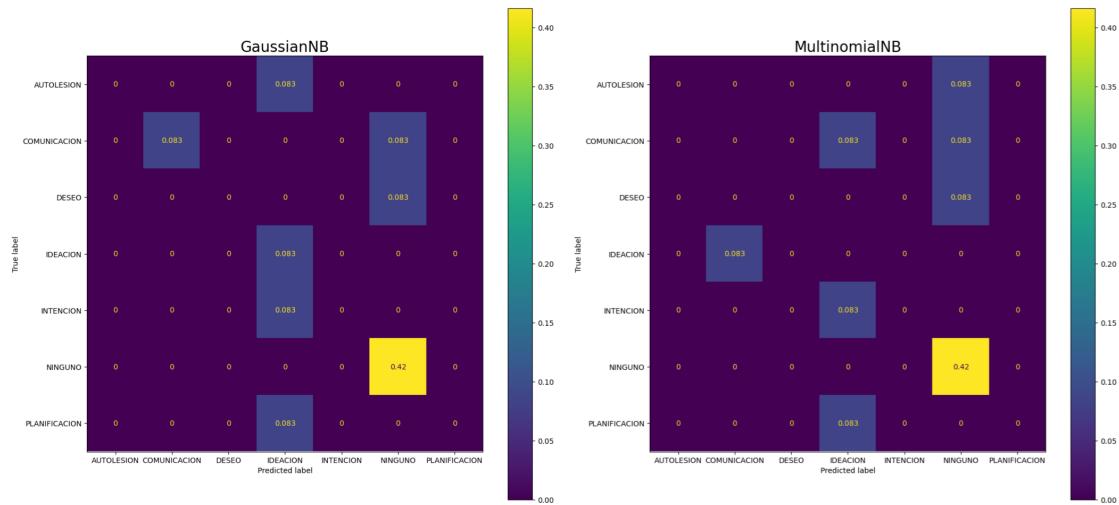


Figura 3.21: Matrices de confusión. Autoinforme. Iteración 7

datos más diversos, para que la gran mayoría de filas no sean con el desenlace NINGUNO. El resultado final fue un nuevo dataset con 68 filas más aparte de las 40 ya existentes, con variedad de desenlaces dependiendo de las variables del dataset.

En esta iteración, no se ha modificado ninguno de los parámetros establecidos. Esto es debido a que se quiso dar más enfoque a la generación del dataset, ya que es la principal forma de mejorar el rendimiento de un modelo.

- **Modelo:** Autoinforme.
- **Clasificador:** GaussianNB, MultinomialNB.
- **Dataset:** *scripts/datasets/autoinforme/v2/dataset_v2_2.csv* - 108 filas.
- **Balance del dataset:** 70 % - 30 %.
- ***average*:** micro.

Se volvió a entrenar el modelo de los jóvenes y se obtuvieron los siguientes resultados:

	GaussianNB	MultinomialNB
Accuracy	0.727	0.636
Precision	0.727	0.636
Recall	0.727	0.636
F1	0.727	0.636
Error medio	0.149	0.0774

Tabla 3.10: Resultados de las métricas de la iteración 8

Se percibe una mejora en las métricas y en las matrices de confusión, especialmente en el gausiano, porque los datos del dataset se ajustan más a una distribución guasiana en lugar de una multinomial. Se puede observar que comienza a acertar ligeramente las predicciones de AUTOLESION y algunas de COMUNICACION, además de NINGUNO. No obstante, siguen sin ser suficientes para el resto de desenlaces, por lo que es necesario una modificación del dataset, ya sea creando uno nuevo o ampliando el que ya existe.

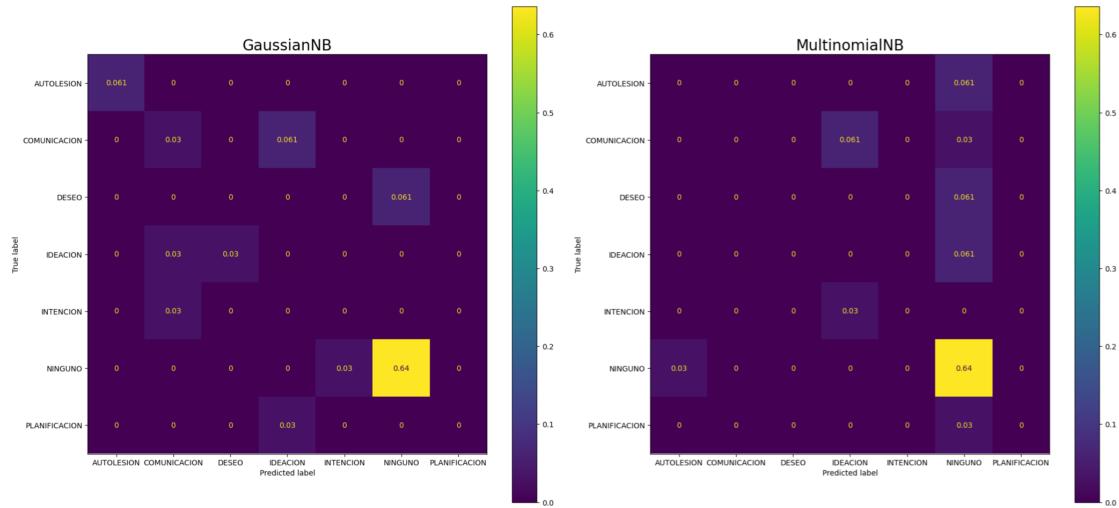


Figura 3.22: Matrices de confusión. Autoinforme. Iteración 8

3.5.2.3. Iteración 9

Se optó por seguir ampliando el dataset, generando 4 filas más de ejemplo de COMUNICACION, DESEO, etc., resultando en un dataset con un total de 112 filas de entrenamiento. Al igual que en el resto de iteraciones, de momento no se han ajustado los parámetros de entrenamiento, con el fin de enfocarse en el desarrollo del dataset.

- **Modelo:** Autoinforme.
- **Clasificador:** GaussianNB, MultinomialNB.
- **Dataset:** *scripts/datasets/autoinforme/v2/dataset_v2_3.csv* - 112 filas.
- **Balance del dataset:** 70 % - 30 %.
- ***average*:** micro.

	GaussianNB	MultinomialNB
Accuracy	0.588	0.647
Precision	0.588	0.647
Recall	0.588	0.647
F1	0.588	0.647
Error medio	0.185	0.124

Tabla 3.11: Resultados de las métricas de la iteración 9

Como resultado, se ha obtenido un peor rendimiento con respecto a la octava iteración en el modelo gausiano, aunque hay una ligera mejora en el categórico, en donde acierta en algunas predicciones con el desenlace COMUNICACION y PLANIFICACION. Además, observando la matriz de confusión, las predicciones han empeorado en el modelo gausiano. Ahora falla en el desenlace de AUTOLESION y COMUNICACION, valores que anteriormente acertaba. Esto puede ser debido a que las filas añadidas no siguen los patrones

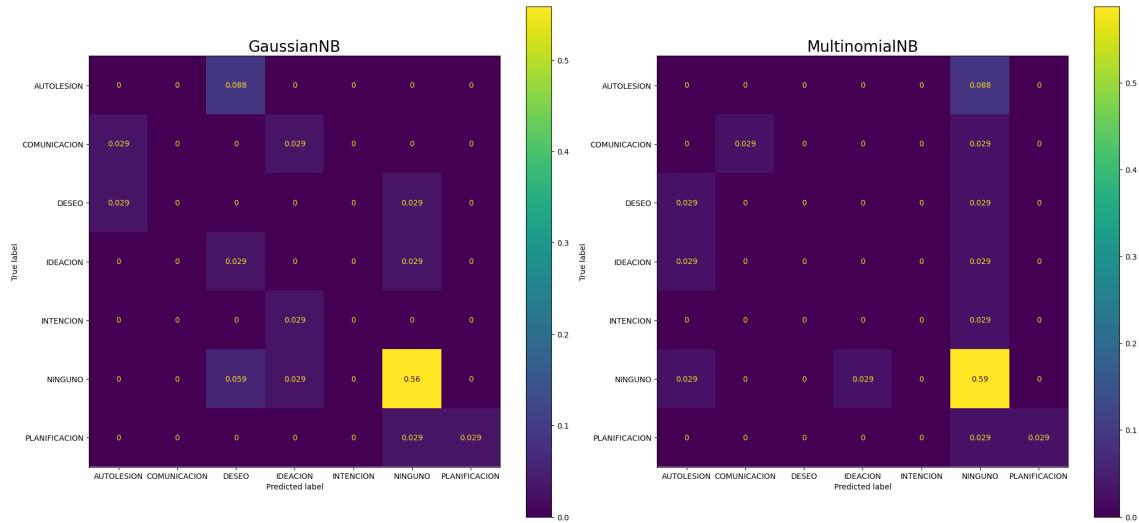


Figura 3.23: Matrices de confusión. Autoinforme. Iteración 9

previamente entrenados, lo que deriva en una mayor cantidad de fallos en las predicciones. Por otro lado, en el modelo multinomial, ha comenzado a acertar en el desenlace de COMUNICACION y en el de PLANIFICACION, pero sigue errando considerablemente en el resto.

3.5.2.4. Conclusiones de la segunda versión de entrenamiento

Después de 3 iteraciones de entrenamiento, se ha visto una mejora en las métricas con respecto a la primera versión de entrenamiento, pero analizando las matrices de confusión, se ha podido observar que el aumento de las puntuaciones de las métricas se deben a una mayor tasa de acierto en las predicciones de NINGUNO. Sin embargo, no hay mejora con el resto de predicciones. Esto se produce porque el modelo no percibe diferencias claras entre las distintas categorías de desenlace. Por ejemplo, no hay un patrón que diferencie una fila con desenlace de PLANIFICACION o IDEACION. Esto provoca un peor rendimiento en el entrenamiento y posteriormente en el testeo del modelo. Además, los márgenes de error calculados en estas 3 iteraciones los consideré demasiado dispares como para continuar con este enfoque.

3.5.3. Tercera versión: cambio en el diseño del modelo

Los resultados de la segunda versión habían mejorado con respecto a la primera, pero seguían siendo relativamente bajos. Además, sus matrices de confusión demostraban que las predicciones realizadas por el modelo del Autoinforme no eran acertadas en la mayoría de las ocasiones.

Es por ello que se decidió rediseñar el modelo de los jóvenes, modificando algunas variables y sus valores. Esto también implicaba realizar cambios en el prompt utilizado en ChatGPT para generar los datasets de entrenamiento, para que se ajusten al nuevo tipo de modelo. Las nuevas estructuras de los datasets están disponibles en las Figuras B.2, B.3 y B.4.

En este caso, el script del entrenamiento corresponde con el cuaderno de Jupyter:

scripts/third_version_model_training_scikit_learn.ipynb

3.5.3.1. Iteración 10

Se le pidió al LLM generar un nuevo dataset con un nuevo formato diferente a la de la anterior versión. Este nuevo dataset estaría constituido por 140 filas. De nuevo, se ha intentado que los datos sean los más diversos y realistas posibles.

- **Clasificador:** GaussianNB, MultinomialNB.
- **Dataset:** *scripts/datasets/autoinforme/v3/dataset_v3_1.csv* - 140 filas.
- **Balance del dataset:** 70 % - 30 %.
- **average:** micro.

En base a este primer dataset y a los parámetros establecidos, se obtuvo los siguientes resultados:

	GaussianNB	MultinomialNB
Accuracy	0.595	0.857
Precision	0.595	0.857
Recall	0.595	0.857
F1	0.595	0.857
Error medio	0.0698	0.0544

Tabla 3.12: Resultados de las métricas de la iteración 10

Con este dataset ha habido una reducción del margen de error medio en comparación con la iteración 9, convirtiéndose en algo residual que no influye de forma drástica en la variación las métricas, como ocurría en la versión anterior.

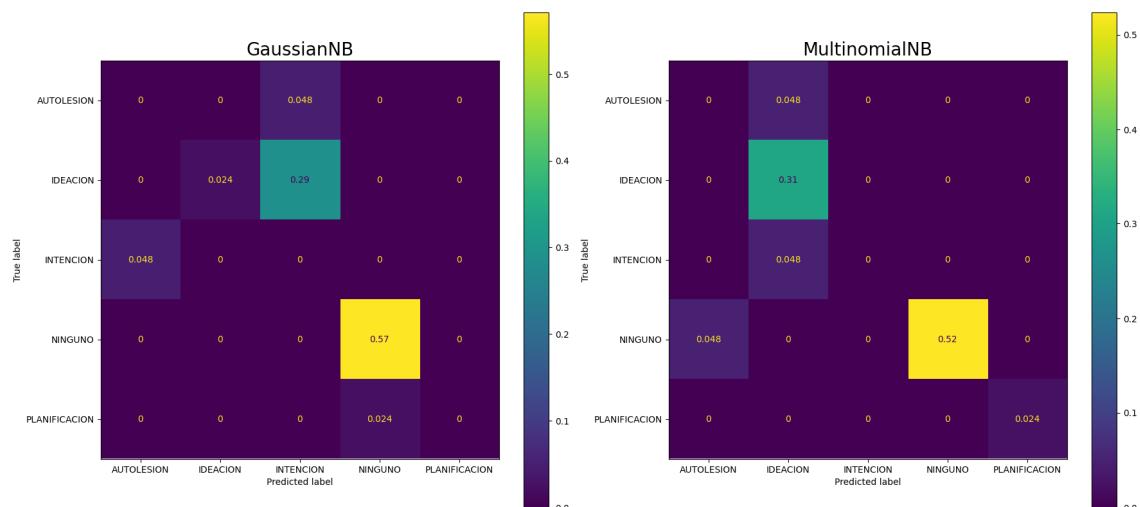


Figura 3.24: Matrices de confusión. Autoinforme. Iteración 10

Con la reducción de número de desenlaces ha permitido un aumento en el rendimiento, como se puede ver en la matriz de confusión del modelo gausiano y, especialmente, del

modelo multinomial, en donde ha acertado correctamente los desenlaces de NINGUNO, IDEACION y PLANIFICACION.

3.5.3.2. Iteración 11

Se decidió ampliar el dataset de la iteración 10, añadiendo más ejemplos de AUTO-LESION, ya que en la anterior no había suficientes como para conseguir un rendimiento óptimo para esa clase, como se pudo observar en la matriz de confusión. El número total resultante del dataset son de 258 filas.

Un punto a destacar es que a partir de esta iteración 11, se ha añadido un nuevo clasificador, que es el categórico (CategoricalNB). Decidí incluir este nuevo tipo porque consideré que se ajusta más al tipo de datos con los que se estaban trabajando. Esto es debido a que, analizando los datasets, se ha comprobado que los datos siguen una distribución categórica, en donde cada dato es una categoría, es decir, un valor de una variable, en lugar de la distribución normal o multinomial.

De todas formas, opté por mantener el entrenamiento de todos los clasificadores, para no omitir posibles resultados que puedan ser mejores que el categórico. Este clasificador se intentó incluir en la iteración anterior. Sin embargo, no se pudo ya que en el dataset generado no estaban presentes todos los valores de las variables del diagrama, lo que generaba un error en durante la fase de testeo.

- **Clasificador:** GaussianNB, MultinomialNB, CategoricalNB.
- **Dataset:** *scripts/datasets/autoinforme/v3/dataset_v3_2.csv* - 258 filas.
- **Balance del dataset:** 70 % - 30 %.
- ***average*:** micro.

Se obtuvieron los siguientes resultados:

	GaussianNB	MultinomialNB	CategoricalNB
Accuracy	0.910	0.885	0.910
Precision	0.910	0.885	0.910
Recall	0.910	0.885	0.910
F1	0.910	0.885	0.910
Error medio	0.0555	0.0693	-

Tabla 3.13: Resultados de las métricas de la iteración 11

En este caso se han conseguido resultados similares en todos los clasificadores. Hay una mejora positiva de las predicciones de AUTOLESION, IDEACION y PLANIFICACION. Sin embargo, todavía comete ciertos fallos con INTENCION. Por lo tanto, se realizarán una ampliación del dataset y repetir el entrenamiento para probar si se puede mejorar ese aspecto. El rendimiento del modelo multinomial sube ligeramente con respecto a la décima iteración, y el modelo categórico presenta unas métricas relativamente altas, similares al del modelo gausiano. Sin embargo, ha habido un problema de que el error medio del clasificador categórico no se ha podido calcular. Esto se debe a que durante la validación cruzada, en alguna de las posibles combinaciones se habrá utilizado un conjunto que no tenga todos los posibles desenlaces (AUTOLESION, IDEACION, PLANIFICACION, INTENCION o

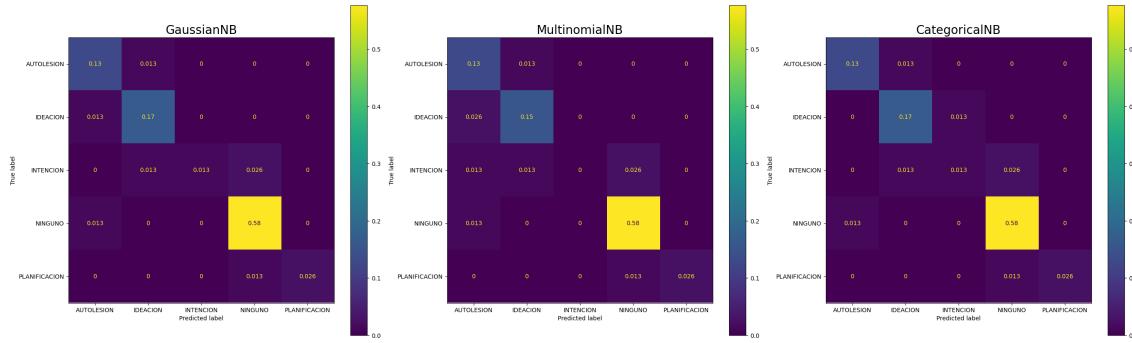


Figura 3.25: Matrices de confusión. Autoinforme. Iteración 11

NINGUNO), por lo que da error en la validación. Para solucionarlo, habría que generar un nuevo dataset o aumentar el existente para añadir mayor variedad en los datos y una mejor distribución.

3.5.3.3. Iteración 12

En este caso, se ha vuelto a ampliar el dataset generando más filas y llegando hasta las 295 filas en total. Se han añadido más filas con desenlace de PLANIFICACION e INTENCION, con el objetivo de mejorar el rendimiento de las predicciones en esas clases. Los parámetros se mantienen intactos.

- **Clasificador:** GaussianNB, MultinomialNB, CategoricalNB.
- **Dataset:** *scripts/datasets/autoinforme/v3/dataset_v3_3.csv* - 295 filas.
- **Balance del dataset:** 70 % - 30 %.
- **average:** micro.

Se obtuvieron los siguientes resultados:

	GaussianNB	MultinomialNB	CategoricalNB
Accuracy	0.854	0.820	0.831
Precision	0.854	0.820	0.831
Recall	0.854	0.820	0.831
F1	0.854	0.820	0.831
Error medio	0.0573	0.084	0.0723

Tabla 3.14: Resultados de las métricas de la iteración 12

En este caso, el modelo categórico sí que devuelve un cálculo del error medio, debido al aumento de datos en el dataset, lo que implica mayor variedad y distribución de los datos. Se puede percibir un ligero descenso del rendimiento en todos los clasificadores, pero teniendo en cuenta las matrices de confusión, las predicciones se siguen manteniendo similares en líneas generales. En este caso, la el desenlace INTENCION ha empezado a ofrecer uns resultados ligeramente mejores, en donde la mitad de las filas de INTENCION del dataset se están prediciendo correctamente, en el caso de los clasificadores gausiano y categórico. En esta iteración, se observa que el que ha ofrecido mejor rendimiento ha sido el gausiano, pero las diferencias entre los tres clasificadores es mínima.

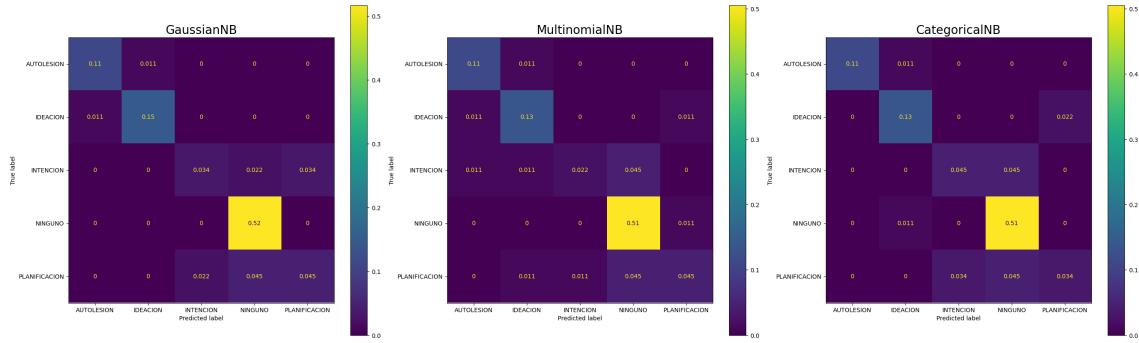


Figura 3.26: Matrices de confusión. Autoinforme. Iteración 12

3.5.3.4. Iteración 13

Decidí cambiar la forma de calcular las métricas para observar si puede haber un aumento en la evaluación del modelo. Como este parámetro sólo modifica esta parte del proceso, la matriz de confusión no varía.

- **Clasificador:** GaussianNB, MultinomialNB.
- **Dataset:** *scripts/datasets/autoinforme/v2/dataset_v2_3.csv* - 295 filas.
- **Balance del dataset:** 70 % - 30 %.
- **average:** weighted.

Se obtuvieron los siguientes resultados:

	GaussianNB	MultinomialNB	CategoricalNB
Accuracy	0.854	0.820	0.831
Precision	0.834	0.803	0.816
Recall	0.854	0.820	0.831
F1	0.838	0.796	0.815
Error medio	0.0573	0.084	0.0723

Tabla 3.15: Resultados de las métricas de la iteración 13

El cambio de métrica nos permite obtener resultados más variados, en lugar de recibir el mismo resultado en todas las métricas. Es por ello, que a partir de ahora se calcularán las métricas teniendo en cuenta el soporte o la importancia de cada clase dentro del dataset, en lugar de calcularlos de forma global.

3.5.3.5. Iteración 14

En esta iteración, consideré que el rendimiento podría mejorar si se modifica el balance del dataset, ajustándolo a un nuevo del 80 % - 20 %, en lugar del 70 % - 30 %.

- **Clasificador:** GaussianNB.
- **Dataset:** *scripts/datasets/autoinforme/v2/dataset_v2_3.csv* - 295 filas.

- **Balance del dataset:** 80 % - 20 %.

- **average:** weighted.

Se obtuvieron los siguientes resultados:

	GaussianNB	MultinomialNB	CategoricalNB
Accuracy	0.881	0.847	0.898
Precision	0.861	0.785	0.893
Recall	0.881	0.847	0.898
F1	0.866	0.813	0.893
Error medio	0.0573	0.084	0.0723

Tabla 3.16: Resultados de las métricas de la iteración 14

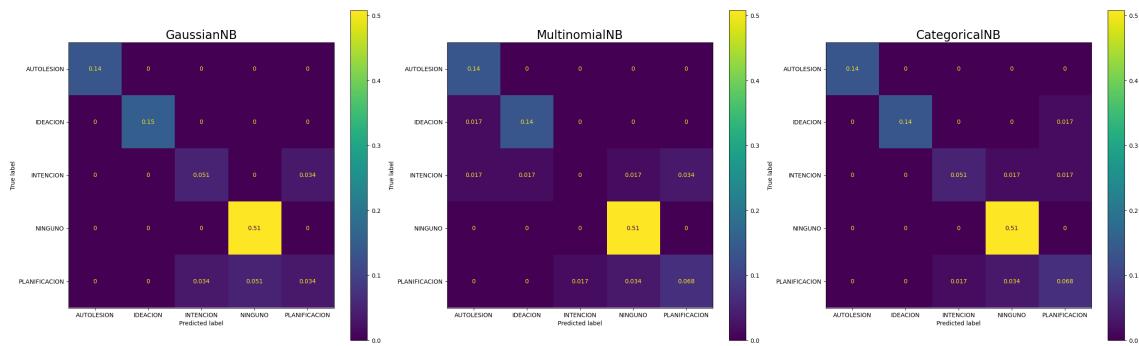


Figura 3.27: Matrices de confusión. Autoinforme. Iteración 14

Vemos que el modelo ha mejorado ligeramente, y el error medio se mantiene el mismo, por lo que no habría ninguna caída de rendimiento ni de puntuación en las métricas. Sin embargo, hay una ligero empeoramiento en el rendimiento de PLANIFICACION en el clasificador de Gauss. En este caso, las filas de esta clase se están prediciendo por lo general, como NINGUNO. Por otro lado, en el modelo multinomial se mantiene relativamente igual, aunque ha empezado a acertar las predicciones de INTENCION. Finalmente, el modelo categórico es el que se ha visto más beneficiado con esta modificación, ya que su puntuación es la más alta de los tres clasificadores.

3.5.3.6. Iteración 15

A lo largo de todas estas iteraciones, se ha ido ampliando el dataset utilizado, agregando cada vez más filas y redistribuyéndolas a lo largo del dataset para obtener mejores predicciones. Es por ello, que en esta última iteración del modelo del Autoinforme, se quiso probar el entrenamiento con un dataset completamente diferente al de los anteriores. Este dataset consta de 123 filas totalmente diferentes a las del dataset anterior, para probar si el rendimiento se replica si recibe un dataset distinto. Es por ello que seleccioné el entrenamiento de la Iteración 13 como la indicada para utilizarlo en la aplicación.

- **Clasificador:** GaussianNB, MultinomialNB, CategoricalNB.
- **Dataset:** *scripts/datasets/autoinforme/v2/dataset_v3_4.csv* - 123 filas.

- **Balance del dataset:** 70 % - 30 %.

- ***average:*** weighted.

Se obtuvieron los siguientes resultados:

	GaussianNB	MultinomialNB	CategoricalNB
Accuracy	0.811	0.595	0.622
Precision	0.873	0.637	0.650
Recall	0.811	0.595	0.622
F1	0.794	0.603	0.626
Error medio	0.03908	0.0578	0.0623

Tabla 3.17: Resultados de las métricas de la iteración 15

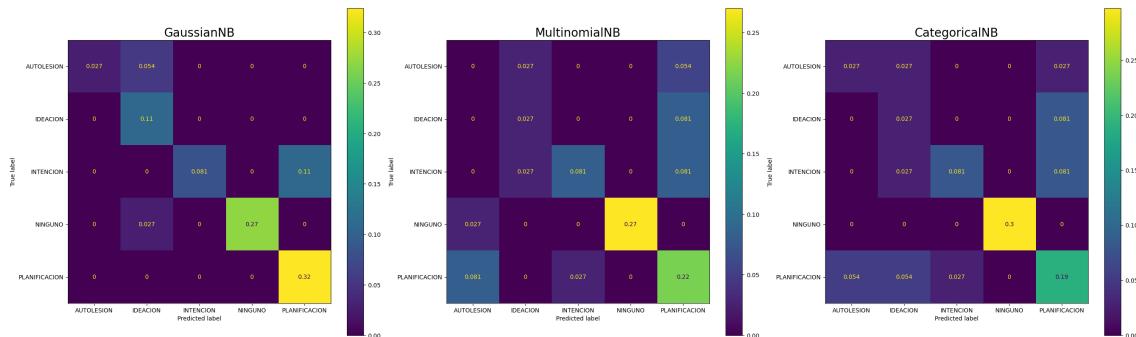


Figura 3.28: Matrices de confusión. Autoinforme. Iteración 15

El resultado final ofrece una matriz de confusión en donde se puede observar que realizan las predicciones correctamente, pero que ofrece peores resultados en comparación al entrenamiento de la iteración anterior. Es por ello, y además por la necesidad de entrenar a los otros 2 modelos restantes, por lo que decidí concluir el entrenamiento del modelo de los jóvenes.

3.5.3.7. Iteración 16

Esta es la primera iteración del modelo de las familias. Empecé con una primera versión de un dataset generado por ChatGPT. Consta de 100 filas utilizando la estructura de la Figura B.3. Los parámetros establecidos fueron los siguientes.

- **Clasificador:** GaussianNB, MultinomialNB, CategoricalNB.
- **Dataset:** *scripts/datasets/familia/v3/dataset_v3_1.csv* - 100 filas.
- **Balance del dataset:** 70 % - 30 %.
- ***average:*** weighted.

En este caso, escogí este balance y este valor del *average* porque eran los parámetros seleccionados de la Iteración 13, que es el entrenamiento elegido para el modelo de los jóvenes. Esto lo hice con el objetivo de mantener los mismos parámetros en todos los modelos, en lugar de tener unos parámetros para un modelo y otros distintos para el resto.

	GaussianNB	MultinomialNB	CategoricalNB
Accuracy	0.767	0.800	0.833
Precision	0.892	0.812	0.909
Recall	0.767	0.800	0.833
F1	0.735	0.763	0.811
Error medio	0.0399	0.1319	0.1029

Tabla 3.18: Resultados de las métricas de la iteración 16

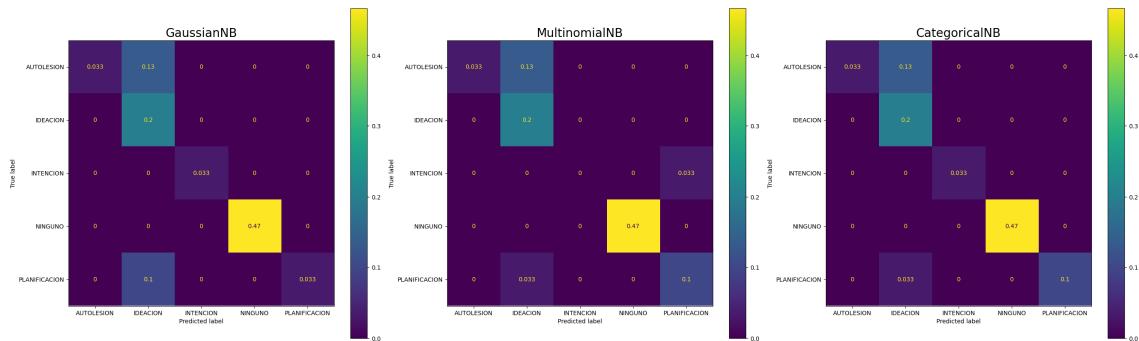


Figura 3.29: Matrices de confusión. Familia. Iteración 16

Este primer entrenamiento ha devuelto unos resultados medianamente correctos.

Por un lado, el modelo gausiano acierta con las categorías de IDEACION, INTENCION y NINGUNO, pero falla en la mayoría de las predicciones con AUTOLESION y PLANIFICACION.

Por otro lado, el modelo multinomial acierta con IDEACION, NINGUNO y PLANIFICACION, pero falla en AUTOLESION e INTENCION.

Finalmente, el modelo categórico, que es el ha obtenido mejor rendimiento en este caso, acierta con todas las categorías, excepto con AUTOLESION.

En líneas generales, la categoría que fallan todas las clasificaciones es la categoría AUTOLESION, por lo que hay que intentar corregir este fallo. Por consiguiente, es necesario mejorar el dataset en las próximas iteraciones, añadiendo más filas de las clases AUTOLESION, además de las categorías de INTENCION y PLANIFICACION.

3.5.3.8. Iteración 17

Se ha ampliado el dataset de la anterior iteración a través del modelo GPT-4o, dando un total de 150 filas, añadiendo filas de AUTOLESION, INTENCION y PLANIFICACION. Los parámetros son los siguientes:

- **Clasificador:** GaussianNB, MultinomialNB, CategoricalNB.
- **Dataset:** *scripts/datasets/familia/v3/dataset_v3_2.csv* - 150 filas.
- **Balance del dataset:** 70 % - 30 %.
- **average:** weighted.

Se obtuvieron las siguientes puntuaciones:

	GaussianNB	MultinomialNB	CategoricalNB
Accuracy	0.822	0.711	0.756
Precision	0.886	0.688	0.768
Recall	0.822	0.711	0.756
F1	0.805	0.691	0.742
Error medio	0.1445	0.1218	0.1103

Tabla 3.19: Resultados de las métricas de la iteración 17

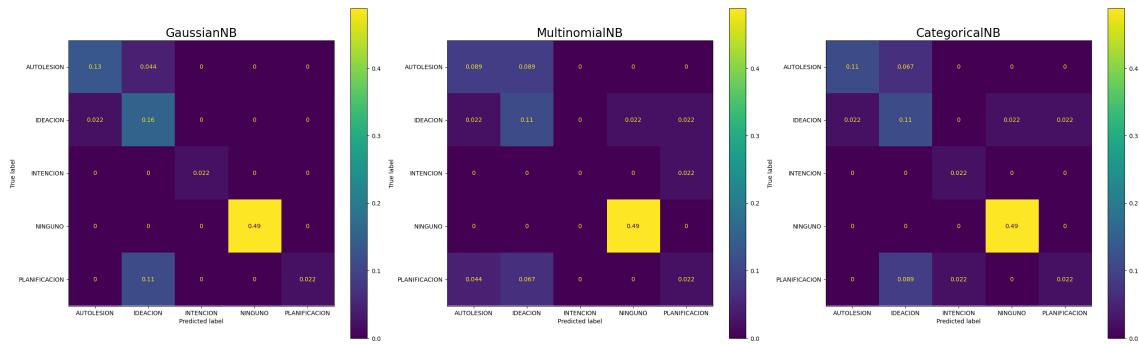


Figura 3.30: Matriz de confusión. Familia. Iteración 17

El rendimiento ha mejorado en el caso de los clasificadores gausiano y categórico, pero disminuye en el multinomial. Sin embargo, el punto más destacado de este entrenamiento es el aumento del error medio en todos ellos, siendo un ratio demasiado elevado como para detener el entrenamiento en este punto. Además, siguen habiendo fallos en las predicciones en distintas categorías en cada uno de los clasificadores.

3.5.3.9. Iteración 18

El aumento del error medio y los fallos en las predicciones en la Iteración 17 hicieron que se tuviese que generar un nuevo dataset. Este dataset consta de 107 filas.

- **Clasificador:** GaussianNB, MultinomialNB, CategoricalNB.
- **Dataset:** *scripts/datasets/familia/v3/dataset_v3_3.csv* - 107 filas.
- **Balance del dataset:** 70 % - 30 %.
- **average:** weighted.

Después de realizar el entrenamiento para los 3 clasificadores, se obtuvieron los siguientes resultados:

Se ha conseguido reducir el error medio en todos los clasificadores y la matriz de confusión muestra mejores predicciones para todas las clases en el modelo gausiano, aunque el rendimiento del multinomial y categórico han bajado. Por este motivo, como se ha conseguido que al menos uno de los modelos ha logrado una predicción correcta de los datos en general, se tomó la decisión de finalizar el entrenamiento de los modelos de la familia.

	GaussianNB	MultinomialNB	CategoricalNB
Accuracy	0.848	0.727	0.788
Precision	0.864	0.668	0.792
Recall	0.848	0.727	0.788
F1	0.852	0.678	0.773
Error medio	0.0682	0.0997	0.0818

Tabla 3.20: Resultados de las métricas de la iteración 18

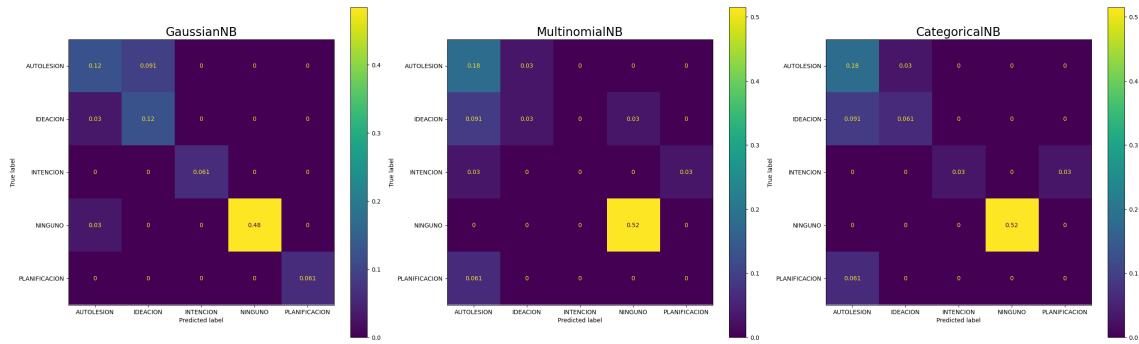


Figura 3.31: Matriz de confusión. Familia. Iteración 18

3.5.3.10. Iteración 19

A partir de esta iteración, el entrenamiento se enfocó al modelo de los profesionales. De nuevo, los datasets generados fueron a través de GPT-4o.

Al ser menos columnas que en el resto, para el LLM le resulta más fácil y rápido generar las filas necesarias para hacer el entrenamiento. Se empezó con un primer dataset de 200 filas y con los siguientes parámetros:

- **Clasificador:** GaussianNB, MultinomialNB, CategoricalNB.
- **Dataset:** *scripts/datasets/profesional/v3/dataset_v3_1.csv* - 200 filas.
- **Balance del dataset:** 70 % - 30 %.
- **average:** weighted.

De esta iteración se obtuvieron los siguientes resultados:

	GaussianNB	MultinomialNB	CategoricalNB
Accuracy	0.283	0.317	0.217
Precision	0.278	0.321	0.227
Recall	0.283	0.317	0.217
F1	0.278	0.314	0.218
Error medio	0.0556	0.0578	0.0748

Tabla 3.21: Resultados de las métricas de la iteración 19

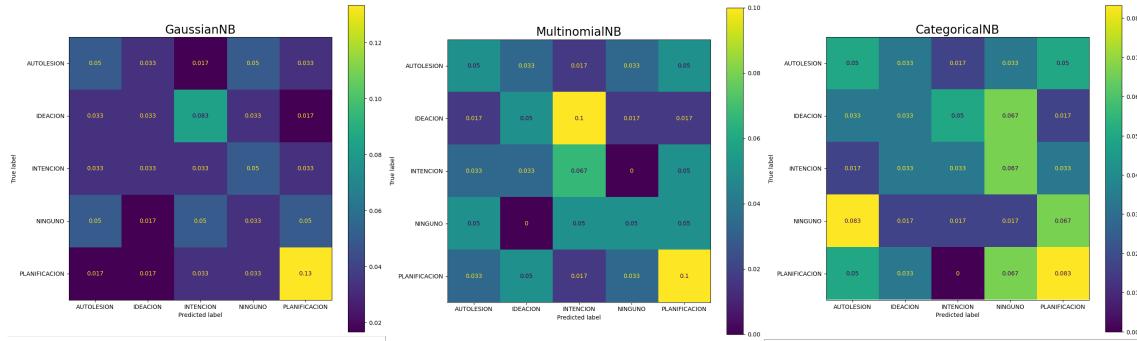


Figura 3.32: Matrices de confusión. Profesionales. Iteración 19

En este primer entrenamiento del modelo de los profesionales, se puede observar una gran disparidad en la matriz de confusión, además de unas métricas considerablemente bajas. Esto puede deberse a la falta de patrones claros dentro del dataset de los profesionales, por lo que las predicciones fallan con demasiada frecuencia.

3.5.3.11. Iteración 20

En esta iteración, se iba a crear un dataset nuevo, que iba a estar constituido de 220 filas, este dataset era el que se encontraba en el directorio *scripts/datasets/profesional/v3/dataset_v3_2.csv*.

Sin embargo, durante la preparación de la actual iteración, me percaté de un fallo que había en el contenido de este dataset y también en el de la iteración 19. En ellas, existen 4 columnas que no corresponden al modelo de los profesionales. Estos son los siguientes:

- Problemas interiorizados.
- Problemas exteriorizados.
- Problemas de recursos.
- Problemas de recursos psicológicos.

Estos campos no corresponden a este modelo porque son variables que se miden en base al cuestionario SENA de la página de SIVARIA. Sin embargo, se comprobó que no estaba presente en el cuestionario de los profesionales, por lo que no habría manera alguna de obtener esos valores si se conectasen a la aplicación. Es por ello que tuve que rehacer el dataset, eliminando estas 4 columnas y volver a entrenar el modelo.

- **Clasificador:** GaussianNB, MultinomialNB, CategoricalNB.
- **Dataset:** *scripts/datasets/profesional/v3/dataset_v3_3.csv* - 220 filas.
- **Balance del dataset:** 70 % - 30 %.
- **average:** weighted.

Después del entrenamiento, se obtuvieron los resultados representados en la Tabla 3.22 y las matrices de la Figura 3.33.

	GaussianNB	MultinomialNB	CategoricalNB
Accuracy	0.439	0.530	0.621
Precision	0.611	0.396	0.609
Recall	0.439	0.530	0.621
F1	0.447	0.412	0.605
Error medio	0.01398	0.0265	0.0334

Tabla 3.22: Resultados de las métricas de la iteración 20

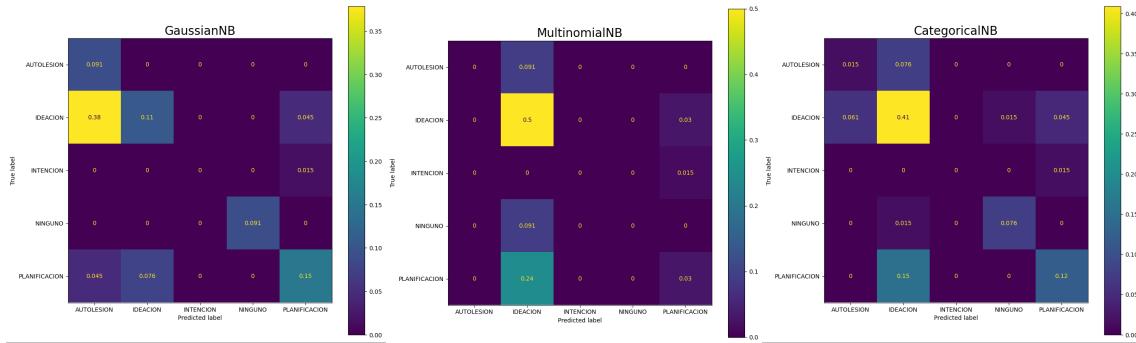


Figura 3.33: Matrices de confusión. Profesionales. Iteración 20

Se puede observar que el rendimiento ha disminuido. Esto se debe a la eliminación de las 4 filas, que representaban factores importantes para diferenciar entre los distintos desenlaces. No obstante, consideré que se podría aumentar el rendimiento si se intentaba ajustar los hiperparámetros del modelo.

3.5.4. Cuarta versión: ajuste de hiperparámetros

A pesar de la considerable mejora en los resultados de la tercera versión, no se había realizado ningún ajuste de hiperparámetros y poder encontrar el mejor modelo con la mejor combinación de hiperparámetros posible. Para ello, se aplicó el algoritmo *GridSearchCV* de la librería *Scikit-learn*. El *GridSearch* es un algoritmo que recibe el tipo de modelo que queremos combinar y una lista de valores de cada uno de los hiperparámetros. Los hiperparámetros que se especifiquen determinarán el número de combinaciones que debe hacer el algoritmo posteriormente. A continuación, internamente realiza una validación cruzada mediante la división de los datos en *K-folds* o K subconjuntos.

En este caso, se ha establecido que sean 5 folds, ya que es el valor por defecto. Por lo tanto, el algoritmo dividirá los datos en 5 subconjuntos, de los cuales k - 1 se usarán para entrenar al clasificador, mientras que el uno restante será para el proceso de evaluación. El algoritmo combina los 5 folds dentro de cada clasificador, por lo tanto, por cada clasificador habrá 5 combinaciones posibles. De esta manera, se puede obtener el clasificador que mejor rendimiento proporcione.

El script del entrenamiento corresponde con el cuaderno de Jupyter:

scripts/forth_version_model_training_scikit_learn.ipynb

3.5.4.1. Iteración 21

En este caso, se decidió entrenar a los 3 modelos, usando los datasets que se seleccionaron en su momento que dieron mejores resultados dieron en sus entrenamientos.

- Autoinforme: *scripts/datasets/autoinforme/v3/dataset_v3_3.csv* (Iteración 13)
- Familia: *scripts/datasets/familia/v3/dataset_v3_3.csv* (Iteración 18)
- Profesional: *scripts/datasets/profesional/v3/dataset_v3_3.csv* (Iteración 20)

Los parámetros seleccionados fueron los siguientes:

- **Clasificador:** GaussianNB, MultinomialNB, CategoricalNB.
- **Balance del dataset:** 70 % - 30 %.
- **average:** weighted.

Y los hiperparámetros a los que se han ajustado fueron los siguientes:

- **GaussianNB:**

- **var_smoothing** (suavizado de la variable): es un parámetro de estabilidad para suavizar la curva, y, por lo tanto, tener en cuenta más muestras de distribución. En este caso, *numpy.logspace(0, -9, n=100)*. Devuelve un array de 100 números separados uniformemente en una escala logarítmica de base 10. Por lo que los valores estarán comprendidos entre 10^0 y 10^{-9} .

```

1     param_grid = {
2         'var_smoothing': np.logspace(0, -9, num=100) # devuelve un array
3             de 100 elementos.
4 }
```

Listing 3.1: Posibles valores de los hiperparámetros del modelo gausiano

Teniendo en cuenta el tamaño del array, se puede saber que habrá 100 modelos que se van a probar, uno por cada valor del hiperparámetro.

- **MultinomialNB y CategoricalNB:**

- **alpha:** hiperparámetro similar al *var_smoothing* del modelo gausiano. Es un parámetro de suavizado cuyo valor por defecto es 1.0, y que se encarga de añadir internamente a los recuentos de datos para eliminar posibles errores de divisiones por 0. Esto es debido a que el modelo calcula las frecuencias de las características en base al recuento de datos, y es probable que si algunas características no están presentes en el dataset, el denominador de la frecuencia arrojaría un error de división por 0.
- **force_alpha:** sirve para establecer un valor alpha fijo. Si su valor es True, el valor de alpha no se modifica. Sin embargo, si es False y alpha tiene un valor inferior a $1 \cdot 10^{-10}$, entonces alpha tendrá este valor establecido.
- **fit_prior:** si el valor es True, entonces las probabilidades *a priori* se calcularán en base a los datos recibidos del dataset. Pero si es False, estas probabilidades tendrán un valor igual fijo.

```

1     param_grid_mul_cat = {
2         'alpha': [0.00001, 0.0001, 0.001, 0.01, 0.1, 1, 10, 100,1000],
3         'force_alpha': [True, False],
4         'fit_prior': [True, False]
5     }
6

```

Listing 3.2: Posibles valores de los hiperparámetros del modelo multinomial

En este caso, el número total de combinaciones sería $|alpha| \cdot |force_alpha| \cdot |fit_prior| = 9 \cdot 2 \cdot 2 = 36$

Se ha configurado el *GridSearchCV* para que entrene al modelo utilizando 5 folds, utilizando la métrica de *accuracy* para la evaluación del modelo con el fold restante. Mediante este método, a cada candidato se le aplican 5 posibles combinaciones de datos. Esto da como resultado un total de $n_candidatos \cdot n_folds = 100 \cdot 5 = 500$ ajustes en el caso del modelo GaussianNB, y $36 \cdot 5 = 180$ posibles ajustes en el modelo MultinomialNB y CategoricalNB.

Finalmente, los rendimientos obtenidos fueron las siguientes:

	Autoinforme	Familia	Profesionales
Accuracy	0.854	0.848	0.712
Precision	0.834	0.864	0.741
Recall	0.854	0.848	0.712
F1	0.838	0.852	0.711
Error medio	0.0573	0.0576	0.0231
Hiperparámetros	var_smoothing= $2,848 \cdot 10^{-9}$	var_smoothing= $1,232 \cdot 10^{-6}$	var_smoothing= 0,02848

Tabla 3.23: Métricas del modelo de Gauss. Iteración 21

	Autoinforme	Familia	Profesionales
Accuracy	0.798	0.758	0.530
Precision	0.720	0.753	0.396
Recall	0.798	0.758	0.530
F1	0.751	0.750	0.412
Error medio	0.0783	0.0743	0.0445
Hiperparámetros	alpha=10 force_alpha=True fit_prior=True	alpha= $1 \cdot 10^{-5}$ force_alpha=True fit_prior=False	alpha=1.0 force_alpha=True fit_prior=True

Tabla 3.24: Métricas del modelo multinomial. Iteración 21

	Autoinforme	Familia	Profesionales
Accuracy	0.798	0.758	0.621
Precision	0.766	0.753	0.617
Recall	0.798	0.758	0.621
F1	0.753	0.749	0.610
Error medio	0.04497	0.07760	0.04453
Hiperparámetros	alpha=10 force_alpha=True fit_prior=False	alpha=1.0 force_alpha=True fit_prior=False	alpha=0.1 force_alpha=True fit_prior=True

Tabla 3.25: Métricas del modelo categórico. Iteración 21

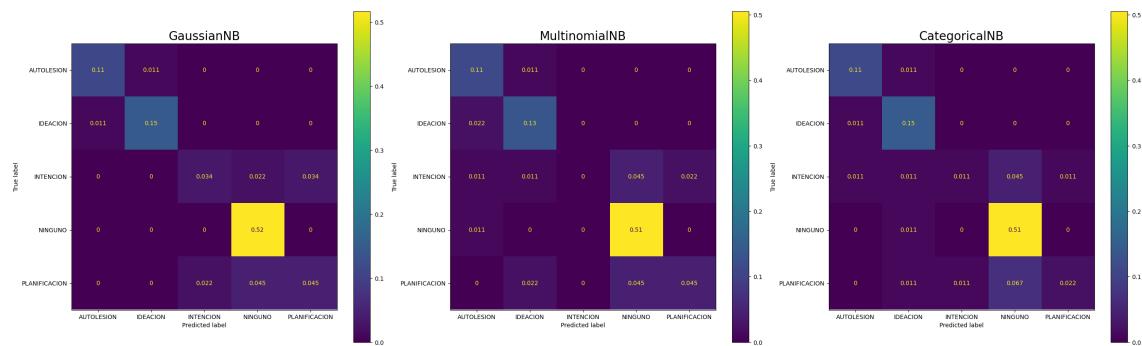


Figura 3.34: Matrices de confusión del Autoinforme(I), familias (II) y profesionales (III) del clasificador gausiano. Iteración 21

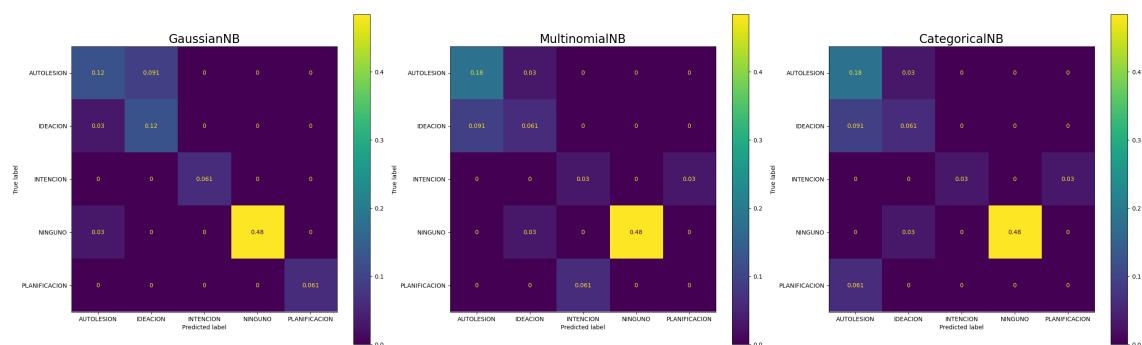


Figura 3.35: Matrices de confusión del Autoinforme(I), familias (II) y profesionales (III) del clasificador multinomial. Iteración 21

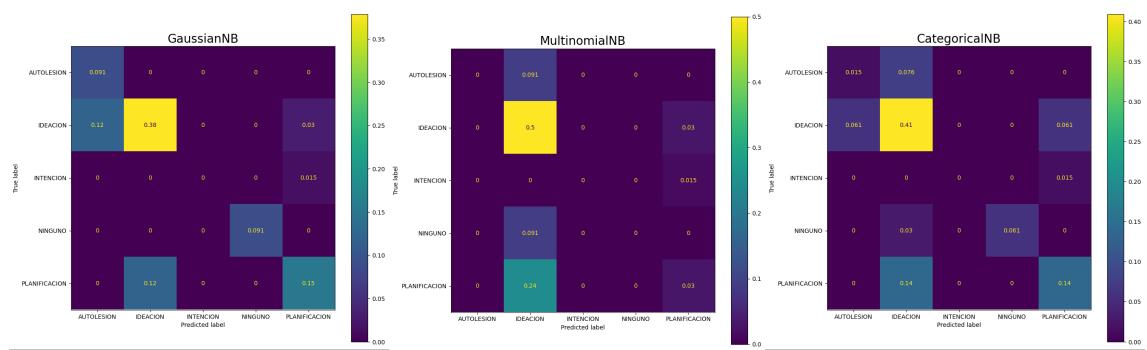


Figura 3.36: Matrices de confusión del Autoinforme(I), familias (II) y profesionales (III) del clasificador categórico. Iteración 21

Haciendo tres tablas comparativas (Tabla 3.26, 3.27 y 3.28), se puede observar que en el modelo de las familias, ha habido en general una bajada de rendimiento en los clasificadores multinomial y categórico. que ha habido una ligera mejora en el rendimiento del modelo de las familias y, especialmente, de los profesionales, tanto en el modelo gausiano como el multinomial. Por lo tanto, el entrenamiento de la iteración 21 será la seleccionada para implementarlo en la aplicación y usarlo para los procesos de predicción.

	Autoinforme		Familia		Profesionales	
Iteración	13	21	18	21	20	21
Accuracy	0.854	0.854	0.848	0.848	0.439	0.712
Precision	0.834	0.834	0.864	0.864	0.611	0.741
Recall	0.854	0.854	0.848	0.848	0.439	0.712
F1	0.838	0.838	0.852	0.852	0.447	0.711

Tabla 3.26: Tabla comparativa de las métricas del modelo de Gauss

	Autoinforme		Familia		Profesionales	
Iteración	13	21	18	21	20	21
Accuracy	0.820	0.798	0.727	0.758	0.530	0.530
Precision	0.803	0.720	0.668	0.753	0.396	0.396
Recall	0.820	0.798	0.727	0.758	0.530	0.530
F1	0.796	0.751	0.678	0.750	0.412	0.412

Tabla 3.27: Tabla comparativa de las métricas del modelo multinomial

	Autoinforme		Familia		Profesionales	
Iteración	13	21	18	21	20	21
Accuracy	0.831	0.798	0.788	0.758	0.621	0.621
Precision	0.816	0.766	0.792	0.753	0.609	0.617
Recall	0.831	0.798	0.788	0.758	0.621	0.621
F1	0.815	0.753	0.773	0.749	0.605	0.610

Tabla 3.28: Tabla comparativa de las métricas del modelo categórico

3.5.5. Conclusiones del entrenamiento

Finalmente, después de realizar el entrenamiento, se ha realizado una tabla comparativa de las puntuaciones obtenidas en los mejores entrenamientos de cada uno de los 3 modelos de acuerdo a la siguiente Tabla 3.29.

En general, ha habido una mejora considerable en el rendimiento con respecto a la primera versión. Sin embargo, entre los tres clasificadores, el clasificador gausiano es el que ha obtenido mejores resultados que el multinomial y el categórico, en todos los modelos (Autoinforme, Familia y Profesionales). Es por esta razón por la que decidí utilizar el primer clasificador para realizar las predicciones de la aplicación. Posteriormente, tuve que decidir qué métrica usar para el cálculo de la fiabilidad del Sistema Experto. Por otra parte, entre las 4 métricas gausianas, se tuvo en cuenta finalmente la métrica de la precisión, debido que es la métrica más favorable para 2 de los 3 modelos: el de las familias y el de los

Clasificador	Autoinforme			Familia			Profesionales		
	C1	C2	C3	C1	C2	C3	C1	C2	C3
Accuracy	0.854	0.798	0.798	0.848	0.758	0.758	0.712	0.530	0.621
Precision	0.834	0.720	0.766	0.864	0.753	0.753	0.741	0.396	0.617
Recall	0.854	0.798	0.798	0.848	0.758	0.758	0.712	0.530	0.621
F1	0.838	0.751	0.753	0.852	0.750	0.749	0.711	0.412	0.610
Error medio	0.057	0.078	0.045	0.057	0.074	0.077	0.023	0.044	0.044

C1=Gauss, C2=Multinomial, C3=Categórico

Tabla 3.29: Tabla comparativa de las métricas. Versión final.

profesionales. Esta elección, aunque es perjudicial para el modelo del Autoinforme, porque es la métrica con el valor más bajo, en líneas generales, es la más alta para los otros 2, y por lo tanto, es más beneficioso para la fiabilidad del Sistema Experto.

Tomando en consideración los resultados obtenidos y el análisis posterior, se puede concluir que se ha obtenido una fiabilidad del 0.834 (83.4 %) en las predicciones en el modelo del autoinforme, 0.864 (86.4 %) en el modelo de las familias, y un 0.741 (74.1 %) en el modelo de los profesionales.

Realizamos una media aritmética de las tres mediciones de fiabilidad F :

$$F_{SistemaExperto} = \frac{F_{autoinforme} + F_{familias} + F_{profesionales}}{NúmeroModelos} = \frac{0,834 + 0,864 + 0,741}{3} \approx 0,813$$

Por ende, podemos afirmar que hemos logrado crear un Sistema Experto con una fiabilidad general del 0.813 (81.3 %), lo que supone una aumento del 64 % con respecto a la primera versión (17.3 %). Esta fiabilidad final se ha considerado lo suficientemente alta como para detener el entrenamiento en este punto y comenzar con el desarrollo de la aplicación multiplataforma.

3.6. Aplicación

En este apartado se va a explicar todos los procesos que se han realizado para la elaboración de la aplicación.

Tras la realización de los entrenamientos de los modelos. Se procedió al desarrollo de una aplicación móvil y web que pudiese conectarse al Sistema Experto y pudiese realizar predicciones en tiempo real. Para poder desarrollar dicha aplicación, era necesario realizar un análisis en profundidad sobre las tecnologías y herramientas disponibles para su implementación. Finalmente, se llegó a la conclusión de que se iban a crear dos proyectos independientes que iban a ejercer como servidores locales. Ambos proyectos se encuentran dentro de la carpeta *app* del repositorio de Github.

3.6.1. Backend

El primer proyecto es un servidor local API REST que funcionará como backend de la aplicación para procesar las peticiones de la aplicación.

3.6.1.1. API REST

Una API es una interfaz de programación de aplicaciones que permite la comunicación entre sistemas de información a través del protocolo HTTP (*HyperText Transfer Protocol*). Se basa en un modelo cliente-servidor, en el que el cliente, que puede ser una aplicación o un sistema de software, se comunica con el servidor API mediante la utilización de peticiones HTTP para obtener datos como respuesta o realizar operaciones sobre estos datos en diferentes formatos, como XML o JSON.

Por otro lado, la transferencia de estado representacional o *REST* es una arquitectura de software que establece una serie de condiciones sobre las que debe trabajar una API. Uno de los principios fundamentales para que un API se considere REST es que debe ser una interfaz *stateless*, es decir, que debe ser capaz de procesar todas las peticiones del cliente sin tener en cuenta el estado de las solicitudes anteriores. De esta manera, se puede garantizar que el servidor pueda cumplir con las solicitudes en todo momento. Además, tiene que tener una interfaz uniforme y devolver un recurso con el mismo formato para todos los clientes.

Dentro de la API, se distinguen principalmente 4 tipos de peticiones HTTP:

- POST: crea un dato nuevo en base de datos o realiza una operación.
- GET: devuelve datos.
- PUT: modifica un dato en concreto.
- DELETE: borra un dato.

3.6.1.2. Django y Django Rest Framework

Para poder implementar una API REST en local, tomé la decisión de utilizar Django (Django (2024)), que es un framework de código abierto de Python. El motivo principal de su selección fue porque así será más fácil conectarlo con el Sistema Experto, que también fue desarrollado y entrenado en Python. Habían otras alternativas, como NodeJS, que es un servidor backend de Javascript, pero consideré que sería problemática su implementación y comunicación con el Sistema Experto.

Django sigue una arquitectura MVT (*Model View Template*), que es una variante del conocido MVC (*Model View Controller*), por lo que se pueden separar de la siguiente forma:

- **Model**: la interfaz para los datos en la aplicación. Un Model de Django es la representación de una tabla de la base de datos conectada.
- **View**: es un *endpoint* o URL que recibirá las peticiones HTTP, se conecta a los modelos (Model) de Django para realizar las operaciones correspondientes a la base de datos, y posteriormente devolverá una respuesta HTTP al navegador.
- **Template**: es el código HTML que devuelve Django para que se renderice en el frontend. En este caso, no es importante ya que va a funcionar como una API REST, así que su funcionamiento se va a centrar en la gestión de peticiones del cliente.

Sin embargo, Django no está configurado para que funcione como un API REST por defecto, por lo que fue necesario implementar un módulo de apoyo llamado Django Rest

Framework (DRF), que nos proporcionará una serie de herramientas para acceder, modificar o eliminar datos desde una aplicación.

3.6.1.3. Base de datos

Se utilizó una base de datos (BBDD) relacional, por mi familiaridad con este tipo de bases de datos, aunque se podría haber optado por una base de datos NoSQL, como por ejemplo MongoDB, orientado a documentos, o Neo4j, orientado a grafos. En este caso, me decanté por utilizar la base de datos local de SQLite que genera Django por defecto al momento de crear la aplicación. A diferencia del resto de sistemas gestores de bases de datos, que siguen un modelo cliente-servidor, SQLite se incrusta directamente en el proyecto de forma local. Por lo tanto, los registros se leen y se escriben directamente en disco, en lugar de un servidor externo. Se escogió de esta manera porque requería de una solución rápida y sencilla de poder almacenar los datos de la aplicación, y el crear una base de datos externa y que sean gestionados por un servidor suponía emplear tiempo en su creación, configuración y testeo dentro de DRF.

Dentro de esta BBDD, Django crea una serie de tablas por defecto para la aplicación, aunque también se han creado tablas específicas para la aplicación. Estos son los siguientes:

- sivaria_appuser (Model: AppUser): tabla de usuarios de la aplicación. Es una tabla que deriva de la tabla que crea por defecto Django.

Listing 3.3: Tabla sivaria_appuser

```
CREATE TABLE IF NOT EXISTS "sivaria_appuser" ("id" integer NOT NULL
    PRIMARY KEY AUTOINCREMENT, "last_login" datetime NULL,
    "username" varchar(50) NOT NULL, "date_joined" datetime NOT
    NULL, "email" varchar(254) NOT NULL UNIQUE, "password"
    varchar(255) NULL, "phone" varchar(9) NULL, "first_name"
    varchar(255) NOT NULL, "last_name" varchar(255) NOT NULL,
    "is_staff" bool NOT NULL, "is_superuser" bool NOT NULL,
    "is_active" bool NOT NULL, "rol_id" bigint NULL REFERENCES
    "sivaria_rol" ("id") DEFERRABLE INITIALLY DEFERRED,
    "expo_token" varchar(200) NULL, "birth_date" date NULL, "code"
    varchar(30) NULL UNIQUE);
CREATE INDEX "sivaria_appuser_rol_id_a6bbee19" ON "sivaria_appuser"
    ("rol_id");
```

- sivaria_userhasparent (Model: UserHasParent): tabla con la asignación de los padres y profesional que le corresponde al joven.

Listing 3.4: Tabla sivaria_userhasparent

```
CREATE TABLE IF NOT EXISTS "sivaria_userhasparent" ("id" integer
    NOT NULL PRIMARY KEY AUTOINCREMENT, "email_parent_1"
    varchar(254) NULL, "email_parent_2" varchar(254) NULL,
    "responsible_id" bigint NULL REFERENCES "sivaria_appuser"
    ("id") DEFERRABLE INITIALLY DEFERRED, "child_id" bigint NOT
    NULL REFERENCES "sivaria_appuser" ("id") DEFERRABLE INITIALLY
    DEFERRED);
CREATE INDEX "sivaria_userhasparent_responsible_id_3730d6cf" ON
    "sivaria_userhasparent" ("responsible_id");
CREATE INDEX "sivaria_userhasparent_child_id_6ce05bf2" ON
    "sivaria_userhasparent" ("child_id");
```

- sivaria_rol (Model: Rol): tabla con los roles existentes en la plataforma. Estos roles son *joven*, *padre*, *madre* y *profesional*.

Listing 3.5: Tabla sivaria_rol

```
|| CREATE TABLE IF NOT EXISTS "sivaria_rol" ("id" integer NOT NULL
|| PRIMARY KEY AUTOINCREMENT, "slug" varchar(200) NOT NULL UNIQUE,
|| "description" varchar(200) NOT NULL, "code" varchar(2) NOT NULL
|| UNIQUE);
```

- sivaria_pushnotificationtype (Model: PushNotificationType): tabla con los templates para las notificaciones push que se enviarán al dispositivo móvil. Dependiendo del rol del usuario y del nivel de riesgo predicho, el contenido de la notificación será distinto

Listing 3.6: Tabla sivaria_pushnotificationtype

```
|| CREATE TABLE IF NOT EXISTS "sivaria_pushnotificationtype" ("id"
|| integer NOT NULL PRIMARY KEY AUTOINCREMENT, "slug" varchar(50)
|| NOT NULL UNIQUE, "body" text NULL, "data" text NULL, "title"
|| varchar(80) NOT NULL);
```

- sivaria_emailtemplate (Model: EmailTemplate): tabla con los templates para los mensajes de email que se enviarán a los usuarios. Dependiendo del rol del usuario y del nivel de riesgo predicho, el contenido del email será distinto.

Listing 3.7: Tabla sivaria_emailtemplate

```
|| CREATE TABLE IF NOT EXISTS "sivaria_emailtemplate" ("id" integer
|| NOT NULL PRIMARY KEY AUTOINCREMENT, "code" varchar(50) NOT
|| NULL, "subject" varchar(80) NOT NULL, "message" text NULL);
```

- sivaria_senaform (Model: SenaForm): contiene las respuestas al cuestionario SENA del cuestionario de los jóvenes.

Listing 3.8: Tabla sivaria_senaform

```
|| CREATE TABLE IF NOT EXISTS "sivaria_senaform" ("id" integer NOT
|| NULL PRIMARY KEY AUTOINCREMENT, "sena19" integer NULL, "sena23"
|| integer NULL, "sena69" integer NULL, "sena99" integer NULL,
|| "sena103" integer NULL, "sena111" integer NULL, "sena112"
|| integer NULL, "sena115" integer NULL, "sena117" integer NULL,
|| "sena129" integer NULL, "sena137" integer NULL, "sena139"
|| integer NULL, "sena141" integer NULL, "sena146" integer NULL,
|| "sena150" integer NULL, "sena188" integer NULL, "code"
|| varchar(30) NULL UNIQUE);
```

- sivaria_inqform (Model: InqForm): contiene las respuestas al cuestionario INQ.

Listing 3.9: Tabla sivaria_inqform

```
|| CREATE TABLE IF NOT EXISTS "sivaria_inqform" ("id" integer NOT NULL
|| PRIMARY KEY AUTOINCREMENT, "inq1" integer NULL, "inq2" integer
|| NULL, "inq3" integer NULL, "inq4" integer NULL, "inq5" integer
|| NULL, "inq6" integer NULL, "inq7" integer NULL, "inq8" integer
|| NULL, "inq9" integer NULL, "inq10" integer NULL, "inq11"
|| integer NULL, "inq12" integer NULL, "inq13" integer NULL,
|| "inq14" integer NULL, "inq15" integer NULL, "code" varchar(30)
|| NULL UNIQUE);
```

- sivaria_rrssform (Model: RrssForm): contiene las respuestas al cuestionario de redes sociales (RRSS).

Listing 3.10: Tabla sivaria_rrssform

```
CREATE TABLE IF NOT EXISTS "sivaria_rrssform" ("id" integer NOT
NULL PRIMARY KEY AUTOINCREMENT, "rrss1" varchar(2) NULL,
"rrss2" varchar(2) NULL, "rrss3" varchar(2) NULL, "rrss4"
varchar(2) NULL, "rrss5" varchar(2) NULL, "rrss6" varchar(2)
NULL, "rrss7" varchar(2) NULL, "code" varchar(30) NULL UNIQUE);
```

- sivaria_multicagecad4form (Model: MulticageCad4Form): contiene las respuestas al cuestionario MULTICAGE-CAD-4.

Listing 3.11: Tabla sivaria_multicagecad4form

```
CREATE TABLE IF NOT EXISTS "sivaria_multicagecad4form" ("id"
integer NOT NULL PRIMARY KEY AUTOINCREMENT, "mcad1" varchar(2)
NULL, "mcad2" varchar(2) NULL, "mcad3" varchar(2) NULL, "mcad4"
varchar(2) NULL, "mcad5" varchar(2) NULL, "mcad6" varchar(2)
NULL, "mcad7" varchar(2) NULL, "mcad8" varchar(2) NULL, "mcad9"
varchar(2) NULL, "mcad10" varchar(2) NULL, "mcad11" varchar(2)
NULL, "mcad12" varchar(2) NULL, "code" varchar(30) NULL UNIQUE);
```

- sivaria_ebipqecipqform (Model: EbipqEcipqForm): contiene las respuestas al cuestionario EBIPQ y ECIPQ.

Listing 3.12: Tabla sivaria_ebipqecipqform

```
CREATE TABLE IF NOT EXISTS "sivaria_ebipqecipqform" ("id" integer
NOT NULL PRIMARY KEY AUTOINCREMENT, "vb1" integer NULL, "vb2"
integer NULL, "vb4" integer NULL, "ab1" integer NULL, "ab2"
integer NULL, "ab4" integer NULL, "cybv1" integer NULL, "cybv2"
integer NULL, "cybv3" integer NULL, "cybb1" integer NULL,
"cybb2" integer NULL, "cybb3" integer NULL, "code" varchar(30)
NULL UNIQUE);
```

- sivaria_cerqsform (Model: CerqsForm): contiene las respuestas al cuestionario CERQS.

Listing 3.13: Tabla sivaria_cerqsform

```
CREATE TABLE IF NOT EXISTS "sivaria_cerqsform" ("id" integer NOT
NULL PRIMARY KEY AUTOINCREMENT, "cerqs1" integer NULL, "cerqs2"
integer NULL, "cerqs3" integer NULL, "cerqs4" integer NULL,
"cerqs5" integer NULL, "cerqs6" integer NULL, "cerqs7" integer
NULL, "cerqs8" integer NULL, "cerqs9" integer NULL, "cerqs10"
integer NULL, "cerqs11" integer NULL, "cerqs12" integer NULL,
"cerqs13" integer NULL, "cerqs14" integer NULL, "cerqs15"
integer NULL, "cerqs16" integer NULL, "cerqs17" integer NULL,
"cerqs18" integer NULL, "code" varchar(30) NULL UNIQUE);
```

- sivaria_atiform (Model: AtiForm): contiene las respuestas al cuestionario ATI.

Listing 3.14: Tabla sivaria_atiform

```
CREATE TABLE IF NOT EXISTS "sivaria_atiform" ("id" integer NOT NULL
PRIMARY KEY AUTOINCREMENT, "ati1" integer NULL, "ati2" integer
NULL, "ati3" integer NULL, "ati4" integer NULL, "ati5" integer
NULL, "ati6" integer NULL, "code" varchar(30) NULL UNIQUE);
```

- sivaria_ateform (Model: AteForm): contiene las respuestas al cuestionario ATE.

Listing 3.15: Tabla sivaria_ateform

```
CREATE TABLE IF NOT EXISTS "sivaria_ateform" ("id" integer NOT NULL
    PRIMARY KEY AUTOINCREMENT, "ate1" integer NULL, "ate10" integer
    NULL, "ate2" integer NULL, "ate3" integer NULL, "ate4" integer
    NULL, "ate5" integer NULL, "ate6" integer NULL, "ate7" integer
    NULL, "ate8" integer NULL, "ate9" integer NULL, "code"
    varchar(30) NULL UNIQUE);
```

- sivaria_edform (Model: EdForm): contiene las respuestas al cuestionario ED.

Listing 3.16: Tabla sivaria_edform

```
CREATE TABLE IF NOT EXISTS "sivaria_edform" ("id" integer NOT NULL
    PRIMARY KEY AUTOINCREMENT, "ed1" integer NULL, "ed2" integer
    NULL, "ed3" integer NULL, "ed4" integer NULL, "ed5" integer
    NULL, "ed6" integer NULL, "ed7" integer NULL, "ed8" integer
    NULL, "ed9" integer NULL, "ed10" integer NULL, "ed11" integer
    NULL, "ed12" integer NULL, "ed13" integer NULL, "ed14" integer
    NULL, "ed15" integer NULL, "ed16" integer NULL, "code"
    varchar(30) NULL UNIQUE);
```

- sivaria_erform (Model: ErForm): contiene las respuestas al cuestionario ER.

Listing 3.17: Tabla sivaria_erform

```
CREATE TABLE IF NOT EXISTS "sivaria_erform" ("id" integer NOT NULL
    PRIMARY KEY AUTOINCREMENT, "er1" integer NULL, "er10" integer
    NULL, "er2" integer NULL, "er3" integer NULL, "er4" integer
    NULL, "er5" integer NULL, "er6" integer NULL, "er7" integer
    NULL, "er8" integer NULL, "er9" integer NULL, "code"
    varchar(30) NULL UNIQUE);
```

- sivaria_familysubform (Model: FamilySubForm): contiene las respuestas a las preguntas relacionadas con las familias, tanto en el cuestionario de jóvenes, como en el de los padres y profesionales.

Listing 3.18: Tabla sivaria_familysubform

```
CREATE TABLE IF NOT EXISTS "sivaria_familysubform" ("id" integer
    NOT NULL PRIMARY KEY AUTOINCREMENT, "adiccion_padre_madre"
    varchar(2) NULL, "madre_adolescente" varchar(2) NULL,
    "maltrato_a_la_pareja" varchar(2) NULL,
    "maltrato_al_adolescente" varchar(2) NULL, "padre_adolescente"
    varchar(2) NULL, "padres_divorciados" varchar(2) NULL,
    "relaciones_conflictivas_hijo_padre_madre" varchar(2) NULL,
    "situacion_economica_precaria" varchar(2) NULL,
    "supervision_parental_insuficiente" varchar(2) NULL, "duelo"
    varchar(2) NULL, "familia_monoparental" varchar(2) NULL,
    "ingreso_familiar_mensual" real NULL, "familia_reconstruida"
    varchar(2) NULL, "code" varchar(30) NULL UNIQUE,
    "tratamiento_psicologico_padre_madre" varchar(2) NULL);
```

- sivaria_injuryform (Model: InjuryForm): contiene las respuestas a la pregunta de autolesión.

Listing 3.19: Tabla sivaria_injuryform

```
CREATE TABLE IF NOT EXISTS "sivaria_injuryform" ("id" integer NOT
NULL PRIMARY KEY AUTOINCREMENT, "injury1" varchar(2) NULL,
"code" varchar(30) NULL UNIQUE);
```

- sivaria_parqform (Model: ParqForm): contiene las respuestas al cuestionario PARQ para el cuestionario de las familias.

Listing 3.20: Tabla sivaria_parqform

```
CREATE TABLE IF NOT EXISTS "sivaria_parqform" ("id" integer NOT
NULL PRIMARY KEY AUTOINCREMENT, "parq1" integer NULL, "parq2"
integer NULL, "parq3" integer NULL, "parq4" integer NULL,
"parq5" integer NULL, "parq6" integer NULL, "parq7" integer
NULL, "parq8" integer NULL, "parq9" integer NULL, "parq10"
integer NULL, "parq11" integer NULL, "parq12" integer NULL,
"parq13" integer NULL, "parq14" integer NULL, "parq15" integer
NULL, "parq16" integer NULL, "parq17" integer NULL, "parq18"
integer NULL, "parq19" integer NULL, "parq20" integer NULL,
"parq21" integer NULL, "parq22" integer NULL, "parq23" integer
NULL, "parq24" integer NULL, "parq25" integer NULL, "parq26"
integer NULL, "parq27" integer NULL, "parq28" integer NULL,
"parq29" integer NULL, "code" varchar(30) NULL UNIQUE);
```

- sivaria_socialdataform (Model: SocialDataForm): contiene las respuestas a las preguntas relacionadas con las variables de identificación del usuario.

Listing 3.21: Tabla sivaria_socialdataform

```
CREATE TABLE IF NOT EXISTS "sivaria_socialdataform" ("id" integer
NOT NULL PRIMARY KEY AUTOINCREMENT, "course" varchar(30) NULL,
"age" integer NULL, "gender" varchar(30) NULL, "trans"
varchar(10) NULL, "job_situation_father" varchar(30) NULL,
"job_situation_mother" varchar(30) NULL,
"academic_level_father" varchar(30) NULL,
"academic_level_mother" varchar(30) NULL,
"academic_performance" varchar(30) NULL,
"previous_psychiatric_treatment" varchar(2) NULL,
"chronic_disease" varchar(2) NULL, "female_self_perception"
integer NULL, "male_self_perception" integer NULL,
"female_others_perception" integer NULL,
"male_others_perception" integer NULL, "weight" real NULL,
"height" real NULL, "discrimination_type" varchar(30) NULL,
"code" varchar(30) NULL UNIQUE);
```

- sivaria_senafamilyform (Model: SenaFamilyForm): contiene las respuestas al cuestionario SENA para el cuestionario de las familias.

Listing 3.22: Tabla sivaria_senafamilyform

```
CREATE TABLE IF NOT EXISTS "sivaria_senafamilyform" ("id" integer
NOT NULL PRIMARY KEY AUTOINCREMENT, "sena104" integer NULL,
"sena117" integer NULL, "sena118" integer NULL, "sena121"
integer NULL, "sena123" integer NULL, "sena124" integer NULL,
"sena125" integer NULL, "sena135" integer NULL, "sena137"
integer NULL, "sena138" integer NULL, "sena139" integer NULL,
```

```

    "sena140" integer NULL, "sena145" integer NULL, "sena146"
    integer NULL, "sena148" integer NULL, "sena154" integer NULL,
    "code" varchar(30) NULL UNIQUE);

```

- sivaria_youngform (Model: YoungForm): contiene las respuestas a todos los cuestionarios al cuestionario de los jóvenes.

Listing 3.23: Tabla sivaria_youngform

```

CREATE TABLE IF NOT EXISTS "sivaria_youngform" ("id" integer NOT
NULL PRIMARY KEY AUTOINCREMENT, "date" datetime NULL,
"prediction" varchar(30) NULL, "ate_id" bigint NULL REFERENCES
"sivaria_ateform" ("id") DEFERRABLE INITIALLY DEFERRED,
"ati_id" bigint NULL REFERENCES "sivaria_atiform" ("id")
DEFERRABLE INITIALLY DEFERRED, "cerqs_id" bigint NULL
REFERENCES "sivaria_cerqsform" ("id") DEFERRABLE INITIALLY
DEFERRED, "ebipq_ecipq_id" bigint NULL REFERENCES
"sivaria_ebipqecipqform" ("id") DEFERRABLE INITIALLY DEFERRED,
"ed_id" bigint NULL REFERENCES "sivaria_edform" ("id")
DEFERRABLE INITIALLY DEFERRED, "er_id" bigint NULL REFERENCES
"sivaria_erform" ("id") DEFERRABLE INITIALLY DEFERRED,
"family_id" bigint NULL REFERENCES "sivaria_familysubform"
("id") DEFERRABLE INITIALLY DEFERRED, "injury_id" bigint NULL
REFERENCES "sivaria_injuryform" ("id") DEFERRABLE INITIALLY
DEFERRED, "inq_id" bigint NULL REFERENCES "sivaria_inqform"
("id") DEFERRABLE INITIALLY DEFERRED, "mcad_id" bigint NULL
REFERENCES "sivaria_multicagecad4form" ("id") DEFERRABLE
INITIALLY DEFERRED, "participant_youth_form_id" bigint NOT NULL
REFERENCES "sivaria_appuser" ("id") DEFERRABLE INITIALLY
DEFERRED, "rrss_id" bigint NULL REFERENCES "sivaria_rrssform"
("id") DEFERRABLE INITIALLY DEFERRED, "sena_id" bigint NULL
REFERENCES "sivaria_senaform" ("id") DEFERRABLE INITIALLY
DEFERRED, "social_data_id" bigint NULL REFERENCES
"sivaria_socialdataform" ("id") DEFERRABLE INITIALLY DEFERRED,
"code" varchar(30) NULL UNIQUE);
CREATE INDEX "sivaria_youngform_ate_id_92257b73" ON
"sivaria_youngform" ("ate_id");
CREATE INDEX "sivaria_youngform_ati_id_6911f091" ON
"sivaria_youngform" ("ati_id");
CREATE INDEX "sivaria_youngform_cerqs_id_3c51b4fc" ON
"sivaria_youngform" ("cerqs_id");
CREATE INDEX "sivaria_youngform_ebipq_ecipq_id_1e6873f5" ON
"sivaria_youngform" ("ebipq_ecipq_id");
CREATE INDEX "sivaria_youngform_ed_id_6d290a3f" ON
"sivaria_youngform" ("ed_id");
CREATE INDEX "sivaria_youngform_er_id_19a34630" ON
"sivaria_youngform" ("er_id");
CREATE INDEX "sivaria_youngform_family_id_ac21ca94" ON
"sivaria_youngform" ("family_id");
CREATE INDEX "sivaria_youngform_injury_id_ea2164cd" ON
"sivaria_youngform" ("injury_id");
CREATE INDEX "sivaria_youngform_inq_id_eba3551d" ON
"sivaria_youngform" ("inq_id");
CREATE INDEX "sivaria_youngform_mcad_id_c5db45cd" ON
"sivaria_youngform" ("mcad_id");
CREATE INDEX "sivaria_youngform_participant_youth_form_id_4b327261"

```

```

    ON "sivaria_youngform" ("participant_youth_form_id");
CREATE INDEX "sivaria_youngform_rrss_id_31a2aad5" ON
    "sivaria_youngform" ("rrss_id");
CREATE INDEX "sivaria_youngform_sena_id_de64fa53" ON
    "sivaria_youngform" ("sena_id");
CREATE INDEX "sivaria_youngform_social_data_id_1756971b" ON
    "sivaria_youngform" ("social_data_id");

```

- sivaria_familyform (Model: FamilyForm): contiene las respuestas a todos los cuestionarios del cuestionario de las familias.

Listing 3.24: Tabla sivaria_familyform

```

CREATE TABLE IF NOT EXISTS "sivaria_familyform" ("id" integer NOT
    NULL PRIMARY KEY AUTOINCREMENT, "date" datetime NULL,
    "prediction" varchar(30) NULL, "family_id" bigint NULL
    REFERENCES "sivaria_familysubform" ("id") DEFERRABLE INITIALLY
    DEFERRED, "parq_id" bigint NULL REFERENCES "sivaria_parqform"
    ("id") DEFERRABLE INITIALLY DEFERRED,
    "participant_family_form_id" bigint NOT NULL REFERENCES
    "sivaria_appuser" ("id") DEFERRABLE INITIALLY DEFERRED,
    "sena_family_id" bigint NULL REFERENCES
    "sivaria_senafamilyform" ("id") DEFERRABLE INITIALLY DEFERRED,
    "social_data_id" bigint NULL REFERENCES
    "sivaria_socialdataform" ("id") DEFERRABLE INITIALLY DEFERRED,
    "to_user_family_form_id" bigint NOT NULL REFERENCES
    "sivaria_appuser" ("id") DEFERRABLE INITIALLY DEFERRED, "code"
    varchar(30) NULL UNIQUE);
CREATE INDEX "sivaria_familyform_family_id_1ac77682" ON
    "sivaria_familyform" ("family_id");
CREATE INDEX "sivaria_familyform_parq_id_46a8d7d1" ON
    "sivaria_familyform" ("parq_id");
CREATE INDEX
    "sivaria_familyform_participant_family_form_id_85953a61" ON
    "sivaria_familyform" ("participant_family_form_id");
CREATE INDEX "sivaria_familyform_sena_family_id_a1ac873e" ON
    "sivaria_familyform" ("sena_family_id");
CREATE INDEX "sivaria_familyform_social_data_id_14f26bdc" ON
    "sivaria_familyform" ("social_data_id");
CREATE INDEX "sivaria_familyform_to_user_family_form_id_33166d78"
    ON "sivaria_familyform" ("to_user_family_form_id");

```

- sivaria_professionalform (Model: ProfessionalForm): contiene las respuestas a todos los cuestionarios del cuestionario de los profesionales.

Listing 3.25: Tabla sivaria_professionalform

```

CREATE TABLE IF NOT EXISTS "sivaria_professionalform" ("id" integer
    NOT NULL PRIMARY KEY AUTOINCREMENT, "date" datetime NULL,
    "prediction" varchar(30) NULL, "family_id" bigint NULL
    REFERENCES "sivaria_familysubform" ("id") DEFERRABLE INITIALLY
    DEFERRED, "participant_professional_form_id" bigint NOT NULL
    REFERENCES "sivaria_appuser" ("id") DEFERRABLE INITIALLY
    DEFERRED, "to_user_professional_form_id" bigint NOT NULL
    REFERENCES "sivaria_appuser" ("id") DEFERRABLE INITIALLY
    DEFERRED, "code" varchar(30) NULL UNIQUE, "social_data_id"

```

```
bigint NULL REFERENCES "sivaria_socialdataform" ("id")
DEFERRABLE INITIALLY DEFERRED);
CREATE INDEX "sivaria_professionalform_family_id_f6f935a2" ON
"sivaria_professionalform" ("family_id");
CREATE INDEX
"sivaria_professionalform_participant_professional_form_id_1323f0bf"
ON "sivaria_professionalform"
("participant_professional_form_id");
CREATE INDEX
"sivaria_professionalform_to_user_professional_form_id_0d1b5909"
ON "sivaria_professionalform" ("to_user_professional_form_id");
CREATE INDEX "sivaria_professionalform_social_data_id_1cbd379f" ON
"sivaria_professionalform" ("social_data_id");
```

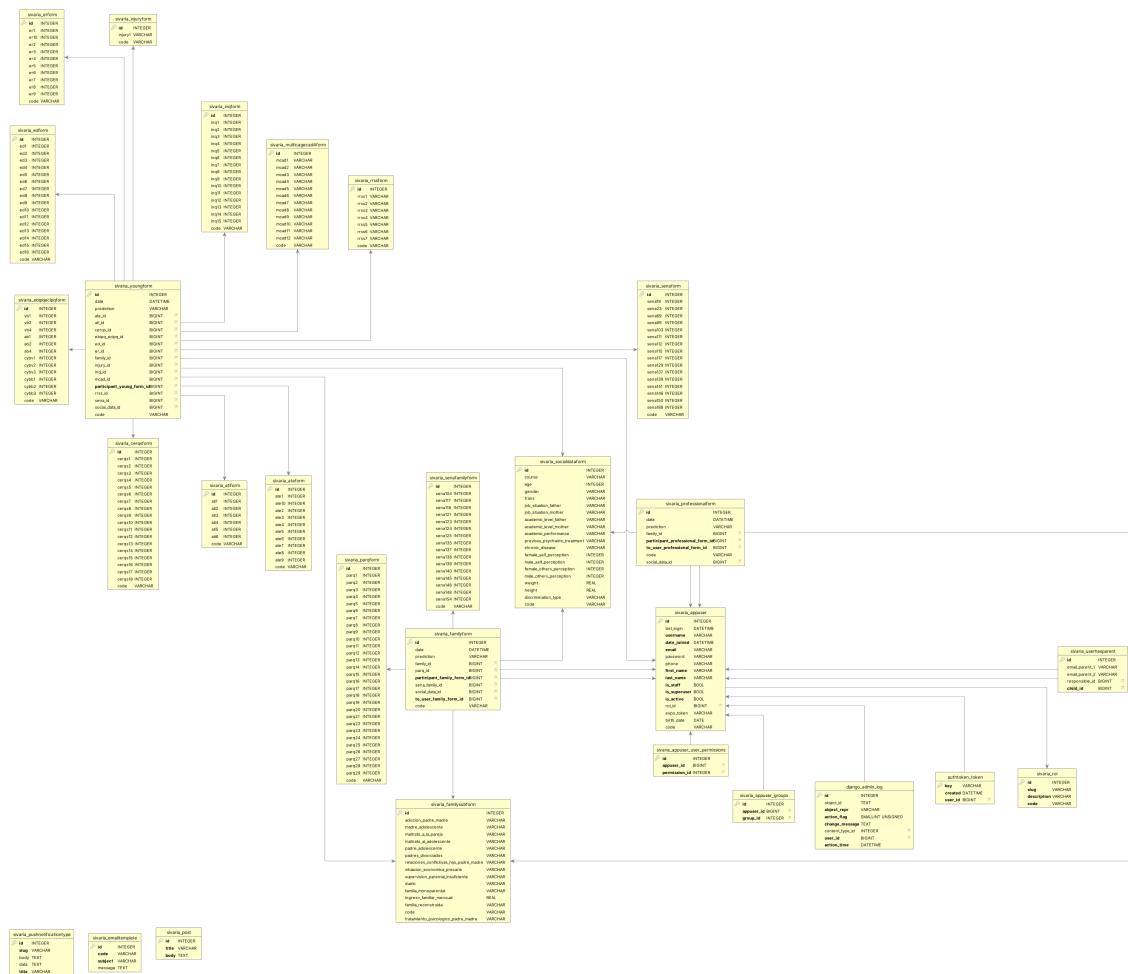


Figura 3.37: Diagrama de las relaciones de las entidades

Todas estas tablas se generan a partir de los modelos de forma local mediante migraciones SQL. Por otro lado, Django también genera automáticamente otras tablas adicionales para implementar las funcionalidades de seguridad, permisos, grupos, autenticación, autorización, logs, entre otros. Cuando se crean y se ejecutan las migraciones por primera vez, se crea la base de datos SQLite en local. Aparecerá un nuevo archivo llamado db.sqlite3 que es la base de datos, a la que se podrá acceder con el comando.

```
python manage.py dbshell
sqlite> .tables
auth_group                                sivaria_erform
auth_group_permissions                      sivaria_familyform
auth_permission                            sivaria_familysubform
auth_token_token                           sivaria_injuryform
django_admin_log                           sivaria_inqform
django_content_type                        sivaria_multicagecad4form
django_migrations                          sivaria_parqform
django_session                            sivaria_professionalform
sivaria_appuser                            sivaria_pushnotificationtype
sivaria_appuser_groups                     sivaria_rol
sivaria_appuser_user_permissions           sivaria_rrssform
sivaria_ateform                            sivaria_senafamilyform
sivaria_atiform                            sivaria_senaform
sivaria_cerqsform                          sivaria_socialdataform
sivaria_ebipqecipqform                    sivaria_userhasparent
sivaria_edform                             sivaria_youngform
sivaria_emailtemplate
```

Todos los modelos creados se encuentran disponibles dentro del panel de administración de Django, a la que sólo tienen acceso los superusuarios de la aplicación. Por lo que, para ingresar en el panel, es necesario crearse previamente una cuenta de superusuario. Su URL es la siguiente DJANGO_BASE_URL/admin

3.6.2. Frontend

El segundo proyecto es un servidor frontend desde donde se desarrollará la interfaz de la aplicación. Para ello, se ha utilizado Expo, un framework de React Native.

3.6.2.1. React Native

React Native (React (2024)) es un framework de Javascript de código abierto utilizado para el desarrollo de aplicaciones móviles nativas. La gran ventaja y la principal razón por la que seleccione esta herramienta es que permite crear plataformas compatibles con los sistemas operativos iOS y Android, por lo que no es necesario tener que crear dos tipos de proyectos para cada uno.

3.6.2.2. Expo framework

Por otro lado, Expo (Expo (2024)) es un framework específico para aplicaciones de React Native. Proporciona una serie de herramientas adicionales para desarrollar, construir y desplegar aplicaciones para iOS, Android, e incluso Web, característica que no está presente en los proyectos que no usan Expo. Utiliza en todos los casos componentes basados en código Javascript o Typescript.

Otra de las grandes ventajas es que dispone de multitud de funcionalidades nativas del móvil a Javascript, como el uso de las cámaras, notificaciones, síntesis de voz, entre otros. Además, dispone de una serie de plataformas para poder desarrollar, desplegar y publicar nuestro proyecto mientras al mismo tiempo estamos trabajando en ellos. Como

dato adicional, dicho framework se encuentra activo y recibe constantes actualizaciones, por lo tanto no hay que preocuparse de desarrollar código desfasado.

En este caso, la aplicación se ha estado probando desde un móvil con un sistema operativo Android 14, y para hacerlo es necesario descargar una aplicación en Google Play llamado Expo Go. Aunque también se puede ejecutar el código en un simulador de un móvil Android creado a través de Android Studio.

3.6.3. Conexión entre el frontend y el backend

Una vez explicados ambos proyectos, a continuación se procederá a explicar la conexión entre el frontend y el backend. Para poder realizar peticiones al servidor desde el frontend, se ha requerido el uso de una librería externa llamada *axios*.

Axios (Axios (2024)), es una de Javascript basada en promesas. Es isomórfico, es decir, que se puede emplear tanto en el lado del cliente, a través del navegador con peticiones XMLHttpRequests, como en el lado del servidor, a través del módulo nativo *http* de Node.js.

En este caso, en la aplicación se crea una instancia de axios con una configuración por defecto, añadiendo la URL base del servidor para poder realizar las peticiones. Esta URL va a variar dependiendo de si la conexión se hace desde el navegador web o desde el dispositivo móvil. Esto es debido a que el backend se ejecuta en local.

Si la conexión se realiza desde la web, la URL es el que genera el servidor cuando se inicia.

Sin embargo, si se hace desde un dispositivo móvil, se requiere de un servicio extra para crear un puente entre el móvil y el servidor local.

Para resolver este problema, decidí utilizar Ngrok. Es un servicio que permite hostear un servidor local en un subdominio suyo. De esta manera, cualquier dispositivo externo puede acceder al servidor introduciendo la URL del dominio. El proceso de acceso se realiza mediante *Tunnelling*, en donde se crea un túnel para recibir peticiones del exterior. Esta petición se redirige a la URL local donde se aloja el servidor.

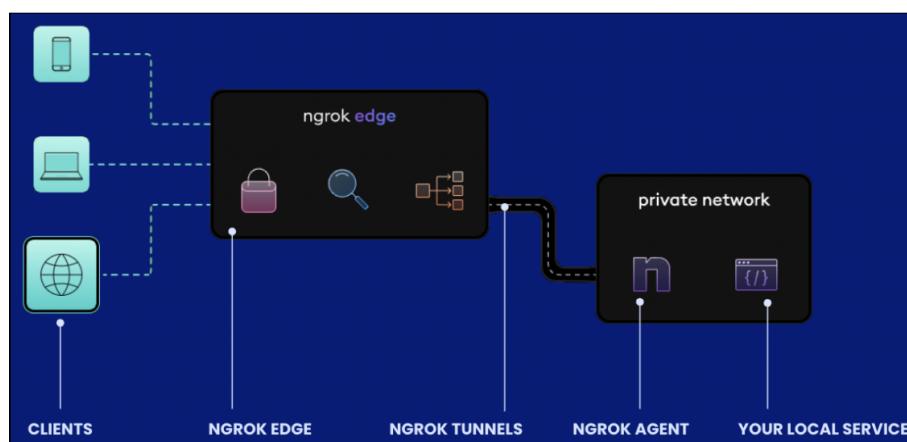


Figura 3.38: Arquitectura de Ngrok. Fuente: Página oficial de Ngrok

3.6.4. Casos de uso

En esta sección se van a mostrar los casos de uso diseñados para la aplicación.

Para empezar, especificué cuáles eran los actores presentes en la aplicación.

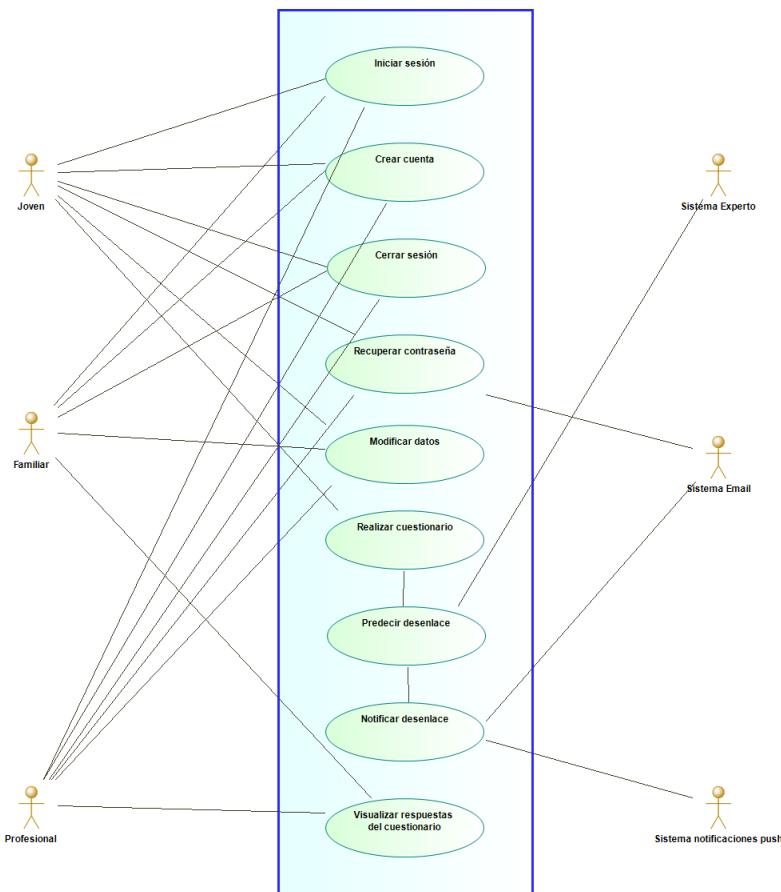


Figura 3.39: Diagrama de casos de uso.

- Usuario de la aplicación, que puede ser:
 - Joven.
 - Familiar.
 - Profesional.
- Sistema Experto.
- Sistema email
- Sistema de notificaciones push de Expo

Después, realicé el siguiente diagrama de casos de uso, para visualizar y representar la relación entre los actores y los casos de uso.

A continuación, desarrollé los correspondientes casos de uso.

UC-01	Iniciar sesión	
Descripción	El usuario inicia sesión introduciendo el correo electrónico y la contraseña como credenciales.	
Actores	Usuario (joven, familiar, profesional)	
Precondición	Ninguna	
Secuencia Principal	Paso	Acción
	1	El usuario introduce el email y la contraseña.
	2	La aplicación valida el email y la contraseña (AS-01).
	3	La aplicación revisa si un usuario existe en la plataforma que coincide con las credenciales introducidas (AS-02).
	4	El usuario es validado.
	5	La aplicación muestra al usuario un mensaje de éxito.
Postcondición	Las credenciales son validadas y el usuario accede a la plataforma.	
Secuencias Alternativas	AS-01	El email y/o la contraseña no son válidos.
	1	Se muestra un mensaje de error al usuario indicando que el email y/o la contraseña no son válidos.
	AS-02	El usuario no existe en la plataforma.
	1	La aplicación muestra un mensaje de error indicando que el usuario no ha sido encontrado en la plataforma.

Tabla 3.30: Caso de uso 1: Iniciar sesión

UC-02	Crear cuenta	
Descripción	El usuario se registra en la plataforma introduciendo una serie de datos iniciales.	
Actores	Usuario (joven, familiar, profesional).	
Precondición	Ninguna	
Secuencia Principal	Paso	Acción
	1	El usuario introduce su nombre, apellidos, correo electrónico, contraseña, número de teléfono y fecha de nacimiento (AS-01, AS-02).
	2	El usuario envía los datos introducidos.
	3	La aplicación valida los datos (AS-05).
	4	El usuario es validado
	5	La aplicación crea una cuenta nueva al usuario (AS-06).
	6	La aplicación muestra un mensaje indicando que la cuenta se ha creado exitosamente.
Postcondición	Se crea la cuenta del usuario en la plataforma.	
Secuencias Alternativas	AS-01	El usuario ha puesto una fecha de nacimiento menor de 21 años.
	1	El usuario debe introducir el email del padre o figura parental 1, o madre o figura parental 2, y email del profesional asignado.
	2	Se vuelve al paso 2 de la secuencia principal.
	AS-02	El usuario ha puesto una fecha de nacimiento mayor de 21 años
	1	El usuario debe especificar si es padre, madre o profesional (AS-03, AS-04).
	AS-03	El usuario indica que es padre o madre.
	1	El usuario introduce el email del hijo.
	2	Se vuelve al paso 2 de la secuencia principal.
	AS-04	El usuario especifica que es un profesional
	1	Se vuelve al paso 2 de la secuencia principal.
AS-05	Los datos introducidos no son válidos.	
	1	Se muestra mensaje de error al usuario de que los datos no son válidos.
	AS-06	Ya existe un usuario con el mismo correo electrónico.
	1	La aplicación muestra un mensaje de error indicando que el usuario ya existe en la plataforma.

Tabla 3.31: Caso de uso 2: Crear cuenta

UC-03	Cerrar sesión	
Descripción	El usuario cierra sesión	
Actores	Usuario (joven, familiar, profesional).	
Precondición	El usuario debe de ser autenticado previamente.	
Secuencia Principal	Paso	Acción
	1	El usuario selecciona la opción de cerrar sesión
	2	La aplicación revisa los datos del usuario (AS-01).
	3	La aplicación elimina los datos almacenados en el dispositivo.
	4	La aplicación redirige al usuario a la pantalla de inicio de sesión
	5	La aplicación muestra un mensaje indicando que se ha cerrado la sesión exitosamente.
Postcondición	Se cierra la sesión del usuario de la plataforma	
Secuencias Alternativas	AS-01	Ha habido un error revisando los datos del usuario.
	1	La aplicación muestra un mensaje de error indicando que ha habido un fallo en el cierre de sesión.

Tabla 3.32: Caso de uso 3: Cerrar sesión

UC-04	Recuperar contraseña	
Descripción	El usuario recupera la contraseña de su cuenta.	
Actores	Usuario (joven, familiar, profesional), sistema de emails	
Precondición	Ninguna	
Secuencia Principal	Paso	Acción
	1	El usuario introduce un correo electrónico.
	2	El sistema de email envía un correo electrónico al usuario.
	3	El usuario accede al enlace.
	4	El usuario establece una nueva contraseña.
	5	La aplicación valida la nueva contraseña (AS-01).
	6	La aplicación actualiza la contraseña.
	7	Se muestra un mensaje de éxito indicando que se ha modificado la contraseña del usuario.
Postcondición	El usuario crea una nueva contraseña para su cuenta.	
Secuencias Alternativas	AS-01	La contraseña no es válida
	1	La aplicación muestra un mensaje de error indicando que la contraseña no es válida.

Tabla 3.33: Caso de uso 4: Recuperar contraseña

UC-05	Modificar datos	
Descripción	El usuario actualiza sus datos	
Actores	Usuario (joven, familiar, profesional)	
Precondición	El usuario debe de estar autenticado	
Secuencia Principal	Paso	Acción
	1	El usuario introduce los datos nuevos (contraseña y número de teléfono).
	2	El usuario envía los datos nuevos.
	3	La aplicación valida la contraseña y número de teléfono (AS-01, AS-02).
	4	La aplicación actualiza los datos del usuario.
	5	La aplicación muestra un mensaje de éxito indicando que se han modificado correctamente los datos del usuario.
Postcondición	La contraseña y/o el número de teléfono se han actualizado.	
Secuencias Alternativas	AS-01	La contraseña no es válida
	1	La aplicación muestra un mensaje de error indicando que la contraseña no es válida.
	AS-02	El número de teléfono no es válido
	1	La aplicación muestra un mensaje de error indicando que el número de teléfono no es válido.

Tabla 3.34: Caso de uso 5: Modificar datos

UC-06	Realizar cuestionario	
Descripción	El usuario realiza el cuestionario y obtiene un predicción.	
Actores	Usuario (joven, familiar, profesional), Sistema Experto, Sistema de Emails, Sistema de notificaciones push	
Precondición	El usuario debe de estar autenticado	
Secuencia Principal	Paso	Acción
	1	El usuario responde al cuestionario.
	2	El usuario envía el cuestionario.
	3	La aplicación revisa que todas las preguntas no están vacías (AS-01).
	4	La aplicación guarda las respuestas.
	5	La aplicación conecta con el Sistema Experto.
	6	La aplicación envía las respuestas del cuestionario.
	7	El Sistema Experto realiza una predicción (UC-07).
	8	La aplicación notifica del desenlace al usuario (UC-08).
	9	La aplicación muestra un mensaje indicando que el mensaje se ha enviado correctamente.
Postcondición	El usuario ha completado el cuestionario y se le ha notificado el resultado de la predicción.	
Secuencias Alternativas	AS-01	Al menos una pregunta del cuestionario está vacía.
	1	La aplicación muestra un mensaje de error indicando las preguntas que están vacías.

Tabla 3.35: Caso de uso 6: Hacer cuestionario

UC-07	Predecir desenlace	
Descripción	El Sistema Experto realiza una predicción	
Actores	Sistema Experto	
Precondición	El usuario debe de estar autenticado y el usuario debe haber respondido a un cuestionario.	
Secuencia Principal	Paso	Acción
	1	El Sistema Experto recibe las respuestas del cuestionario
	2	El sistema se conecta con el modelo dependiendo del rol del usuario.
	3	El Sistema Experto envía las respuestas al modelo.
	4	El modelo realiza la predicción (AS-01)
	5	El Sistema Experto devuelve la predicción
Postcondición	El Sistema Experto devuelve una predicción.	
Secuencias Alternativas	AS-01	Error durante la predicción del Sistema Experto
	1	La aplicación muestra un mensaje de error indicando el fallo en la predicción del Sistema Experto.

Tabla 3.36: Caso de uso 7: Predecir desenlace

UC-08	Notificar desenlace	
Descripción	La aplicación notifica a los usuarios involucrados a través del sistema de emails y el sistema de notificaciones push	
Actores	Sistema de Emails, Sistema de notificaciones push	
Precondición	El usuario debe de estar autenticado y debe haber una predicción realizada previamente.	
Secuencia Principal	Paso	Acción
	1	La aplicación selecciona el contenido del email y push en base a la predicción recibida.
	2	La aplicación busca a los familiares y profesionales relacionados con el que ha llenado el cuestionario.
	3	La aplicación se conecta al Sistema de Emails (AS-01).
	4	La aplicación se conecta al Sistema de notificaciones push (AS-02).
	5	La aplicación envía un correo electrónico a los usuarios relacionados con el que ha realizado el cuestionario (familiares y profesionales) (AS-03).
	6	La aplicación envía una notificación push a los usuarios relacionados con el que ha realizado el cuestionario (familiares y profesionales) (AS-04).
Postcondición	El Sistema Experto devuelve una predicción.	
Secuencias Alternativas	AS-01	Error de conexión con el sistema de emails.
	1	Muestra mensaje de error indicando el error de conexión con el sistema.
	AS-02	Error de conexión con el sistema de notificaciones push.
	1	Muestra mensaje de error indicando el error de conexión con el sistema.
	AS-03	Error en el envío del correo electrónico.
	1	Muestra un mensaje error indicando el fallo en el envío del correo.
	AS-04	Error en el envío del push.
	1	Muestra un mensaje error indicando el fallo en el envío de la notificación.

Tabla 3.37: Caso de uso 8: Notificar desenlace

UC-09	Visualizar respuestas del cuestionario	
Descripción	El usuario puede visualizar las respuestas de los cuestionarios	
Actores	Familiar, profesional	
Precondición	El usuario debe de estar autenticado	
Secuencia Principal	Paso	Acción
	1	La aplicación identifica las personas relacionadas con el familiar o profesional que solicita visualizar las respuestas.
	2	La aplicación selecciona los cuestionarios respondidos por las personas relacionadas y el propio usuario.
	3	La aplicación muestra estos resultados.
	4	El usuario selecciona uno de los cuestionarios.
	5	La aplicación muestra las respuestas de ese cuestionario.
Postcondición	El familiar y profesional obtienen todos los cuestionarios de las personas relacionadas con ellos, y obtienen las respuestas de todos los cuestionarios.	

Tabla 3.38: Caso de uso 9: Ver respuestas del cuestionario

3.6.5. Funcionamiento de la aplicación

En primer lugar, se va a mostrar la pantalla de inicio y se le va solicitar al usuario si le da permiso a la aplicación de recibir notificaciones. Una vez haya dado permiso a la aplicación, el usuario puede realizar diversas acciones. Puede iniciar sesión introduciendo email y contraseña. Si no tiene una cuenta creada, puede darle al botón de registro para crearse una cuenta nueva. Si por el contrario, tiene una cuenta, pero no recuerda su contraseña, puede darle al enlace de Recuperar contraseña".

A continuación, explicaré cada funcionalidad de la aplicación.

3.6.5.1. Registrarse

Para registrarse, el usuario debe dar al botón de *Registrarse*. Después, el usuario es redirigido a la pantalla de registro, donde tiene que introducir sus datos, como nombre, apellidos, número de teléfono, email. Además, hay un campo para introducir su fecha de nacimiento. Dependiendo de la fecha que introduzca, la aplicación mostrará más campos adicionales o no.

- Si el usuario es menor de 21 años, se mostraran 3 campos nuevos para introducir el email del padre o figura parental 1, madre o figura parental 2, y el profesional asignado.
- Si el usuario es mayor de 21 años, aparecerá un dropdown donde puede ser identificado como padre, madre o profesional. Si el usuario es un padre o madre, entonces se aparecerá un campo nuevo donde debe especificar el correo de su hijo o hija. Si selecciona la opción de *profesional*, no aparecerá nuevos campos, así que no se le pedirá más información.

Finalmente, después de una serie de validaciones en el frontend, los datos son enviados al backend, donde se obtienen los datos del JSON, se genera el hash de la contraseña, se realizan las mismas validaciones que en el frontend y se crea un nuevo registro en base de datos.

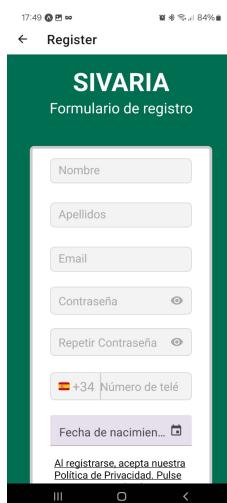


Figura 3.40: Pantalla de registro

3.6.5.2. Iniciar sesión

El usuario introduce el email y la contraseña. Posteriormente, cuando el usuario dé al botón de *Iniciar sesión*, se envía una petición al backend. Cabe resaltar que en este punto, el usuario todavía no se ha autenticado, por lo que el endpoint de login es de los pocos que es de acceso libre, por razones lógicas.

Dentro del backend, se obtienen los datos del usuario, se validan el email y la contraseña y se comprueba si existe un usuario con las credenciales. Si no existe, devuelve un mensaje de error. Sino, se genera un nuevo token de autorización para el usuario y se devuelve en la respuesta, junto con los datos del usuario. Finalmente, estos datos los recibe el cliente, los almacena en el almacenamiento local del móvil o navegador, y es redirigido a la página *Home* de la aplicación.



Figura 3.41: Pantalla de inicio de sesión

3.6.5.3. Recuperar contraseña

Si el usuario pincha en el enlace de recuperación de contraseña, se redirigirá a la pantalla de *Recuperación de Contraseña*. Ahí hay un campo para introducir el correo electrónico que utilizó para crearse la cuenta. Posteriormente, da al botón de *Enviar* y se enviará automáticamente un correo usando el sistema de emails de Django. Si no lo recibe es probable que esté en *Correo no deseado*. En dicho email, se le proporciona un enlace **web** con el correo del usuario y el token de autorización. Si se da al enlace, se abrirá la página de *Nueva contraseña*. En ella, introduce la nueva contraseña, la envía al servidor, éste la valida y actualiza la contraseña del usuario.

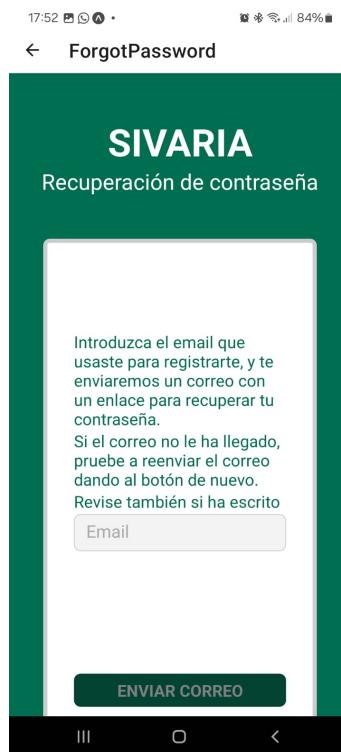


Figura 3.42: Pantalla de recuperación de contraseña

3.6.5.4. Hacer cuestionarios

Una vez el usuario consigue iniciar sesión, es redirigido a la pantalla de "Home", donde habrá un botón visible para realizar el cuestionario correspondiente. Dependiente del rol del usuario, las preguntas del cuestionario serán distintos. Esto se hizo con el objetivo de replicar en la medida de lo posible los cuestionarios del sitio web de SIVARIA.

Para poder enviar el cuestionario, todos los campos deben estar llenados, sino devolverá un error. Posteriormente, cuando el usuario envíe el cuestionario al servidor, se hará una petición POST al endpoint de predicción, que realizará las siguientes tareas:

1. Guarda las respuestas del cuestionario en base de datos. Cuando estos cuestionarios son almacenados, se genera un código para cada tipo de sub-cuestionario, y para el cuestionario principal, que también se guardan en base de datos. Este código consta de una secuencia de caracteres alfanuméricos formados por el ID del usuario que ha llenado el cuestionario en base de datos, su rol (*J*, joven; *P*, padre; *M*, madre; *PR*, profesional), la fecha (AAAA/MM/DD) y hora (HH:MM:SS) de realización del cuestionario.

<ROL><ID><FECHA><HORA>

Por ejemplo, si un profesional cuya ID es 12 y rellena el formulario el día 16/08/2024 a las 16:04:34, entonces el código sería PR 15 20240816 160434. O si fuera de un parent en lugar de un profesional, sería P1520240816160434.

Además, el código de cada sub-cuestionario sigue la misma estructura, pero añadiendo el nombre del sub-cuestionario dentro del propio código.

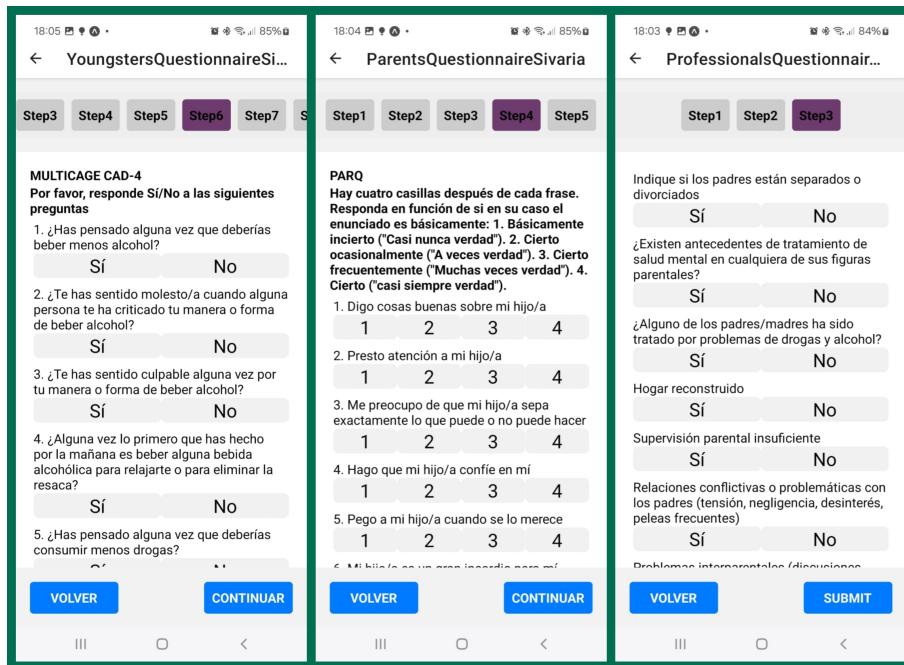


Figura 3.43: Pantallas de cuestionarios de jóvenes (I), familiares (II) y profesionales (III)

<ROL><CODIGO_CUESTIONARIO><ID><FECHA><HORA>

Siguiendo con el ejemplo anterior, el sub-cuestionario SENA del cuestionario de los padres, su código sería PSENAFAMILY1520240816160434. Mediante esta técnica, podemos obtener un código único.

2. Mapea los datos a un formato JSON que sea válido para enviar al Sistema Experto.
3. Se envía el nuevo JSON al Sistema Experto, ejecutando internamente el comando de predicción. Dicha predicción devolverá un desenlace probable del caso.
4. Procede a enviar un correo y notificación push a los padres y profesionales involucrados en el caso. Dependiendo del tipo de desenlace predicho, el contenido del correo y push será distinto.
5. Finalmente devuelve una respuesta al frontend, indicando que la operación se ha realizado correctamente.

3.6.5.5. Ver respuestas de cuestionarios

Dentro de la aplicación, hay un apartado exclusivo para los padres y profesionales donde pueden ver y realizar un seguimiento de los cuestionarios realizados en una Datatable.

En el caso de los padres y madres, pueden observar los cuestionarios que se realizan en todo momento. En el caso de los padres, podrán observar los cuestionarios realizados por sus hijos o hijas, pareja y el profesional asignado a su hijo o hija. Mientras que el profesional puede visualizar los cuestionarios de los familiares y pacientes asignados a él o ella. De esta manera, se garantiza que las personas deseadas puedan ver toda la actividad de las personas involucradas en el seguimiento y monitorización del paciente.

En la tabla aparecerá el código del cuestionario realizado, el nombre de la persona que lo ha realizado, qué rol tiene, la fecha y hora en la que lo hizo y el desenlace predicho. Esta tabla se puede recargar deslizando hacia arriba en el móvil, o dándole al botón de recarga de la página desde el navegador. Por otra parte, si el usuario pincha sobre una de las filas, aparecerá un pop up donde se mostrarán todas las respuestas de ese cuestionario en concreto.

Mediante este método, podemos asegurarnos de que tanto los padres como los profesionales lleven a cabo una monitorización de las respuestas del joven.

Dashboard					
Código	Hecho por	Paciente	Rol	Fecha y Hora	Resultado
PR15202408...	Ald...	Tomás Ramírez profesional		05/08/202...	NINGUNO
P262024072...	Ald...	Tomás Ramírez padre		29/07/202...	AUTOLESION
P262024072...	Ald...	Tomás Ramírez padre		28/07/202...	IDEACION
PR15202407...	Ald...	Tomás Ramírez profesional		27/07/202...	NINGUNO
PR15202407...	Ald...	Tomás Ramírez profesional		27/07/202...	NINGUNO

 Home
 Historial
 Perfil

Figura 3.44: Pantalla de registro de cuestionarios

3.6.5.6. Acciones de perfil

Dentro del apartado del perfil, se le mostrará al usuario su información, como los nombres y apellidos, email, número de teléfono y el rol que tiene. En el caso de que sea un joven, aparte de la información previa, también se va a mostrar la información específica de cada rol. A los jóvenes, aparecerán los emails de sus padres, así como el email de su profesional asignado. Con los padres, aparecerá el nombre y apellidos de sus hijos o hijas registrados en la plataforma. Finalmente, a los profesionales se les muestra el nombre de los pacientes que están a su cargo.

A parte, en la pantalla de perfil están las funcionalidades adicionales para que el usuario cierre sesión, edite los datos de su usuario, e incluso la posibilidad de poder eliminar su cuenta.

3.6.6. Envío de notificaciones push

Dentro de la aplicación, las respuestas a los cuestionarios es informado a los padres y profesionales a través de un correo electrónico y una notificación push que les llega al dispositivo móvil. Con respecto a los pushs, internamente tienen una lógica diferente dependiendo de si se debe enviar a un dispositivo Android o iOS.

- Si el receptor es Android, se debe enviar la notificación a través de FCM (*Firebase Cloud Messaging*), que es una API nativa de Google específica para la recepción y envío de notificaciones push. Una vez la petición llega a Firebase, lo reenvia al destinatario.
- Si por el contrario, el sistema operativo es iOS, el servicio al que se envía es APNS (*Apple Push Notification Service*), que realiza las mismas funciones que el FCM.

Esto implica que el desarrollador tenga que crear dos códigos distintos para ambas plataformas, lo cuál es poco eficiente y supone un trabajo tedioso. Para facilitar este trabajo, Expo ha implementado un servicio de API que agrega una capa de lógica por encima de ambas, permitiendo al programador enviar notificaciones a cualquiera de los dispositivos sin necesidad de duplicar funcionalidades.

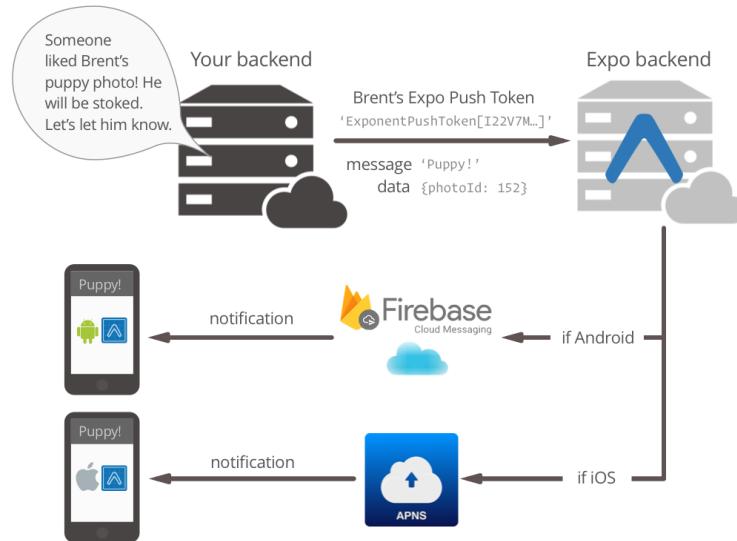


Figura 3.45: Arquitectura del envío de notificaciones de Expo. Fuente: Página oficial de Expo

Para ello, el usuario, cuando entre por primera vez a la aplicación, se le solicitará si puede recibir notificaciones. Si lo acepta, automáticamente se genera un token de Expo denominado *Expo token*. Este token se almacena en base de datos junto con la información del usuario. Cuando se envíe la notificación, el backend local, llamará al endpoint del API de Expo, añadiendo el token del dispositivo, junto el mensaje y datos adicionales, en caso de que los haya. Posteriormente, Expo evalúa el token y determina a qué servidor debe conectarse para enviar el push.

3.6.7. Conexión con el Sistema Experto

La conexión con el Sistema Experto se hará desde el backend a través de una línea de comandos interno. Se desarrolló un prototipo de Sistema Experto en la carpeta *scripts*. Dicha versión del prototipo se replicó dentro del proyecto de backend, en la carpeta *app/-backend/sivaria/expert_system*. Por lo tanto, cuando se necesita ejecutar alguna función del Sistema Experto, el backend inicia un servicio llamado *ExpertService*, que ejecutará el comando especificado por nosotros a través del código. Por ejemplo, si queremos recibir una predicción en base a un JSON recibido de la aplicación de SIVARIA, habría que ejecutar el siguiente comando internamente en el servicio.

```
python controller.py -p --json <base64\_encoded\_json>
```

Para utilizar los modelos, se copiaban los ficheros SAV que se generaban en los cuadernos de Jupyter y se colocaban en una carpeta *configFiles* dentro del backend de la aplicación. Así, el Sistema Experto tendría acceso a él para realizar la predicción.

El prototipo consta de los siguientes componentes.

- **Controller:** recibe el comando introducido por el usuario y ejecuta la correspondiente opción y devuelve el resultado realizado por el Sistema Experto.
- **Expert System:** Sistema Experto de la aplicación. Se encarga internamente de entrenar un modelo seleccionado por el usuario, testear, crear la matriz de confusión y realizar predicciones dado un CSV o un JSON.
- **Checker:** realiza revisiones en el código sobre el formato de los valores recibidos. Si el formato no se cumple, se lanza una excepción.
- **File Manager:** se encarga de administrar los archivos de guardado de las versiones de los modelos. Su trabajo es crear, mostrar y eliminar los archivos de guardado de un modelo. De esta manera, podemos garantizar el control de versiones para todos los modelos y la persistencia de los entrenamientos realizados.
- **Configurator:** se encarga de administrar el archivo de configuración del Sistema Experto. Es capaz de añadir y sustituir valores de la configuración, como el tipo de modelo o puntuación seleccionado.
- **Decoder:** se encarga de recibir el dataframe del CSV, y lo prepara para el entrenamiento del modelo, sustituyendo las variables continuas del CSV por intervalos de valores, y de esta manera, discretizándolos.

Para poder desarrollar todos los componentes, se ha seguido la programación orientada a objetos en Python (*Object Oriented Programming*). Gracias a este paradigma, se pueden dividir los objetos en clases u objetos, lo que permite crear scripts de Python específicos para cada objeto.

Por otro lado, es de importancia mencionar que las peticiones del usuario no se realizan de manera directa al Sistema Experto, sino que al principio van dirigidos al Controlador, quien es el que recibe las órdenes del usuario y llama al Sistema Experto para realizar las correspondientes acciones.

Entonces, la estructura sería similar a lo que se puede ver en la Figura 3.46.

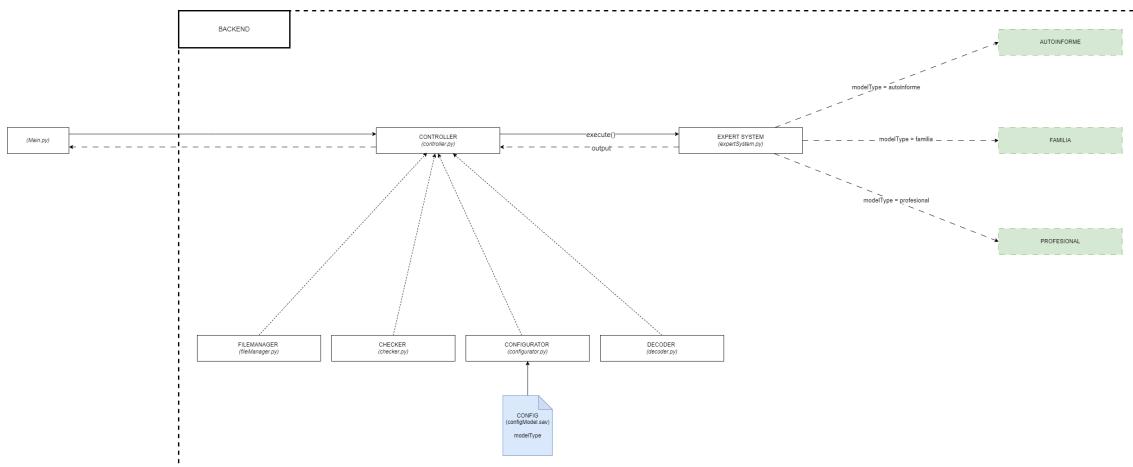


Figura 3.46: Estructura del Controlador con el Sistema Experto

3.6.7.1. Funcionalidades del código

En esta sección se va mostrar las funcionalidades que hay en el código.

- **Establecimiento del tipo de modelo:** esta funcionalidad permite mostrar y establecer el tipo de modelo que se va a utilizar para el entrenamiento del modelo. Los valores permitidos son: *autoinforme*, *familia* y *profesional*.

Posteriormente, el valor establecido se guarda en el archivo de configuración en la carpeta *configFiles* del repositorio.

En la Figura B.5 se puede ver un diagrama de secuencia del comando en cuestión.

- **Listado de archivos de guardado:** devuelve una lista de los archivos de guardado que hay dentro de la carpeta *configFiles/*<*tipo del modelo*>. Se puede especificar el modelo para ver los archivos de guardado de ese modelo, o no se introduce ningún parámetro extra, y como resultado, se devuelven los archivos de guardado de todos los modelos.

En la Figura B.6 se puede observar el diagrama de secuencia del comando que muestra la lista de archivos de guardado de los modelos.

- **Información del archivo de guardado:** el Controlador recibe el nombre de un archivo de guardado y devuelve la información básica de ese archivo. Esta información consta de:

- El nombre del archivo.
- La fecha y hora del último acceso al archivo.
- La fecha y hora de la última modificación.
- La fecha hora de la última vez que ha habido un cambio en los metadatos en Unix y la hora de creación en Windows.
- El tamaño del archivo
- La unidad del tamaño del archivo (KiB, MiB, GiB, TiB, etc.).

En la Figura B.7 se puede observar el diagrama de secuencia del comando.

- **Eliminación de un archivo de guardado:** elimina una archivo de guardado dado el nombre del archivo. En la Figura B.8 se puede observar el diagrama de secuencia del comando del borrado del archivo de guardado.
- **Entrenamiento del modelo:** dado un dataset por parte del usuario (se debe especificar el path del archivo), divide el dataset en base al balance seleccionado en la Sección 3.5.1.7.

Posteriormente, se entrena al modelo con la parte correspondiente del dataset, y la parte restante se usará para crear la matriz de confusión y testear el modelo.

Finalmente, el modelo entrenado se guarda en un archivo SAV dentro de la carpeta *configFiles/*<*tipo de modelo*>, y se devuelve un diccionario de Python, que contiene el número de verdaderos/falsos positivos y negativos, tanto totales como de cada clase de *Desenlace*. Es necesario que previamente a la ejecución de esta opción, es necesario que se haya especificado previamente el tipo de modelo. Para visualizarlo mejor, en la Figura B.9 se puede observar el diagrama de secuencia donde se muestra los pasos internos del entrenamiento.

- **Predicción:** dado un dataset enviado a través de los parámetros del comando, se realiza una predicción de las filas almacenadas dentro de dicho dataset. Se carga el modelo de un fichero SAV dentro de la carpeta *configFiles*. El modelo seleccionado dependerá del tipo especificado en el archivo de configuración. Si no hay un modelo especificado, se devuelve un mensaje de error. Si hay un modelo especificado, entonces se conecta al Sistema Experto, carga la versión del modelo más reciente y devuelve un dataframe con los resultados de esa predicción. En la Figura B.10 se puede observar el diagrama de secuencia del comando.

Todas las funcionalidades previamente explicadas se podrán acceder a ella y ejecutarlas a través del Controlador mediante una serie de comandos creados específicamente para la aplicación. Estos comandos están explicados en el Apéndice A.

3.6.8. Seguridad de la plataforma

Se han implementado medidas de seguridad para la comunicación entre los servidores frontend y backend, ya que se encuentran en dominios distintos, y pueden generar ciertos problemas de seguridad.

En primer lugar, se han optado medidas en relación a la conexión y comunicación entre el dispositivo móvil y el servidor, que se realiza mediante *Tunnelling* con Ngrok. Este proceso genera un agujero en el firewall, lo cuál puede generar un problema de vulnerabilidad. Además, expone nuestro servidor local a Internet al alojarlo en un subdominio de Ngrok. Ante este problema, el comando ngrok genera las URLs usando el protocolo HTTPS, garantizando que las comunicaciones entre ambos componentes vayan cifradas. Por otro lado, la URL se regenera automáticamente cada vez que se ejecuta el comando ngrok. Por último, para añadir más seguridad, la URL generada no se comparte públicamente, sino que se guarda en un fichero .env, que se usa para la configuración de variables de entorno. Este archivo no se sube al repositorio, sino que el desarrollador debe generarlo de forma local.

Con respecto al servidor backend, por lo general un servidor está expuesto a ataques de inyecciones SQL, en donde el atacante inserta en un formulario consultas o parte de una consulta SQL para que se ejecute en el servidor, y devuelva datos comprometidos de la base de datos. Para solucionar esta vulnerabilidad, Django se encarga internamente de limpiar y escapar caracteres especiales. Esto está implementado a través de los querysets de Django, que están creados usando la parametrización de la query. Este proceso consiste en definir el código SQL aparte de los parámetros recibidos del exterior, ya que pueden ser inseguros.

Por otro lado, el servidor de Django, por defecto, está configurado para recibir peticiones solamente de su propio dominio. Esto se hace para garantizar la protección frente a ataques de script de otros sitios (*XSS, Cross-Site Scripting*), ataques CSRF (*Cross-site Request Forgery*) o ataques de envenenamiento de caché o envenenamiento de los links de los correos electrónicos. No obstante, como el frontend se aloja en un servidor distinto, se tuvo que modificar dicha configuración para añadir una *whitelist*, de tal forma que sólo pueda recibir peticiones externas del dominio del frontend. Cualquier otra petición hecha desde otro servidor, tanto externo como interno, será rechazada.

A su vez, la aplicación sigue un método de autenticación y autorización por token. De tal modo que cuando el usuario inicia sesión o crea una cuenta en la plataforma, se genera un token para poder ejecutar los endpoints del backend, ya que la gran mayoría de los endpoints se han creado para que sólo los usuarios autorizados puedan acceder y ejecutar el endpoint. Por último, las contraseñas de los usuarios no se almacenan en texto en claro,

puesto que esto representa una vulnerabilidad. Para evitar esto, antes de almacenar el registro, se genera el hash de la contraseña utilizando el algoritmo criptográfico SHA-256, creando una secuencia de números y caracteres de 256 bits. Este hash es el se guarda finalmente en base de datos.

Como medida adicional, durante la generación de un hash o de un token, Django los firma mediante el uso de una clave secreta. Esta clave secreta se crea automáticamente cuando se inicia un proyecto nuevo. Pero esta clave secreta no se puede compartir en el repositorio de Github, ya que quedaría pública. Es por ello que se ha cambiado la configuración del servidor para que dicha clave la genere cada uno de forma local, y se debe especificar en el archivo de configuración. De este modo, evitamos que la clave secreta se comparta a través del repositorio, garantizando su seguridad.

Capítulo 4

Herramientas de desarrollo y tecnologías

En esta sección se realizará un resumen de las herramientas y tecnologías utilizadas para la realización del TFM

4.1. Herramientas

4.1.1. Visual Studio Code

Visual Studio Code es un editor de código fuente, gratuito, de código abierto y multiplataforma, ya que fue creado por Microsoft para Windows, Linux, MacOS y Web. Cuenta con herramientas internas de depuración, resaltado de sintaxis, control de versiones de Git, autocompletado de código, entre otros.

Con VSCode, se ha desarrollado los scripts de Python, así como la aplicación multiplataforma.

Cabe resaltar, que, en un principio, se iba a emplear Android Studio para el desarrollo de la aplicación. Esto era debido a mi familiaridad con la herramienta, además de su popularidad dentro de la creación de aplicaciones móviles. Sin embargo, lo descarté porque no permitía que la aplicación pudiera servir también para web o iOS, lo que limitaba su alcance sólo a móviles con sistema operativo Android.

4.1.2. Github

Todas las novedades y cambios en el código del Sistema Experto, aplicación o en la documentación se añaden a un repositorio de Github. Github es una plataforma en línea para alojar una gran diversidad de proyectos. El enlace al repositorio de SIVARIA es el siguiente:

<https://github.com/NILGroup/TFM-2324-SIVARIA>

Entre las características más notables de Github, destaca su sistema de control de versiones basado en Git. Git facilita el seguimiento de todos los cambios realizados en el código fuente, abarcando todas las ramas del repositorio. Esta capacidad resulta sumamente útil para administrar el desarrollo del software de manera eficiente y efectiva. Con respecto al flujo de trabajo, se tenía pensado seguir el flujo de trabajo *GitFlow* que ya está implementado en Github, en donde se parte de una rama *main*, que simulará ser una rama de

producción. De esta rama se deriva otra nueva llamada *develop*, y a partir de esta se van a crear subramas llamadas *feature*, una por cada tarea o corrección que haya que realizar.

De esta manera, cada tarea que se vaya a desarrollar se hará en una rama nueva *feature*, y cuando dicha tarea se termine, se mergea a la rama *develop*, y posteriormente a la rama *main* mediante una *release*, liberando las tareas realizadas para que los directores o cualquier usuario pueda visualizar las nuevas implementaciones. Siguiendo esta metodología, podemos tener el trabajo bien diferenciado por cada tarea.

Por otra parte, se utilizará la herramienta Github Desktop en local para subir todos los cambios al repositorio remoto de Github.

4.1.3. Latex

LATEX(Urban, 1986), creado y lanzado inicialmente por Leslie Lamport en los años 1980, es un sistema open-source de composición de texto ampliamente utilizado en la redacción de documentos, textos y artículos académicos y científicos. Una de las características más destacadas de Latex es que se basan en comandos de texto, que a su vez provienen del lenguaje de comandos original TEX, a diferencia de un procesador de textos convencional, que utilizan un enfoque visual. Esto nos permite escribir caracteres especiales y funciones matemáticas y, además de enumerar más fácilmente imágenes y tablas.

Esta herramienta es la que se ha utilizado para la redacción de la memoria.

4.1.4. Programas de creación de diagramas

En esta sección, se han empleado diversas herramientas para la creación de los diagramas del proyecto. Cada una de ellas se especializan en un tema en concreto.

- **Draw.io:** Draw.io es una herramienta online de dibujo gráfico que es utilizado para crear diagramas de flujo, organigramas, diagramas UML o de red de forma rápida y sencilla. Es gratuita y se puede vincular con Google Drive, de tal forma que podemos guardar el archivo draw.io en la nube. Se ha utilizado para diseñar y exportar los diagramas de Bayes.
- **Modelio:** Modelio es una herramienta de código abierto que se utiliza para la creación de diagramas UML. En mi caso, se empleó para el diseño de los diagramas de secuencia y los diagramas de casos de uso de la aplicación.
- **DBVisualizer:** es un programa que permite representar diagramas de relaciones en una base de datos. Se utilizó para crear el diagrama de relaciones de entidades de la base de datos de SQLite de la aplicación.

4.1.5. Anaconda

Anaconda es una plataforma de distribución libre y abierta para los lenguajes de Python y R. Se utiliza en la ciencia de datos y en el aprendizaje automático de modelos y algoritmos.

Desde esta herramienta, se pueden crear entornos virtuales de Python para el desarrollo de scripts y la administración de paquetes mediante el sistema de gestión de paquetes *conda*.

Además, también desde su interfaz se puede acceder a herramientas internas de Anaconda, como Spyder, JupyterLab, PyCharm Professional, Jupyter Notebook, entre otros, como se puede observar en la Figura 4.1. Es precisamente esta última herramienta la que se

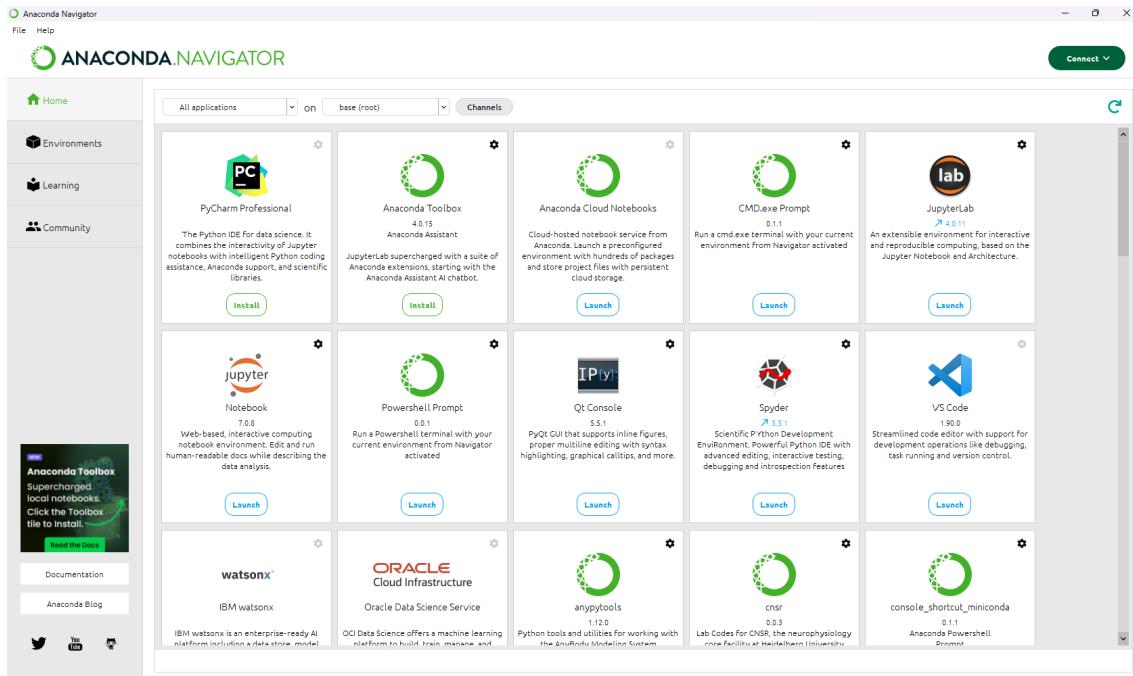


Figura 4.1: Programas de Anaconda

ha utilizado para la creación de cuadernos donde se ejemplifica el entrenamiento, predicción y testeо de los modelos del Sistema Experto.

4.1.6. Postman

Postman es una plataforma utilizada para probar APIs, permitiendo a los desarrolladores enviar peticiones al servidor web y ver las respuestas que devuelve. En este caso, se ha empleado para realizar pruebas sobre todos los endpoints de mi servidor local API.

Había otras alternativas, como usar el comando *curl* o la interfaz web que ofrece Django Rest Framework. No obstante, me decanté finalmente por Postman, ya que ofrece una interfaz más intuitiva y sencilla de utilizar, además de la posibilidad de crear variables de entorno para los endpoints del API, exportar los endpoints. Aparte, permite ordenar los endpoints en carpetas y documentarlos.

4.1.7. Expo Go

Expo Go es una plataforma móvil de código abierto que permite compilar y ejecutar con facilidad aplicaciones nativas de Android e iOS sin necesidad de instalar o configurar ninguna dependencia nativa, como Android Studio.

Se ha seleccionado por su facilidad para probar aplicaciones móviles, y se uso para poder visualizar la aplicación en un dispositivo Android físico, en lugar de un emulador.

4.1.8. Gmail

Google Gmail es un servicio de Google que gestiona el envío y recibimiento de correos electrónicos. Se ha utilizado Google para crear una cuenta de Gmail desde donde se enviarán los correos de notificación automatizados de las predicciones de la aplicación.

4.2. Tecnologías y lenguajes

4.2.1. Python

Se decidió utilizar el lenguaje de programación Python para desarrollar el Sistema Experto de la aplicación, concretamente, la versión de Python utilizada es la 3.9 (Python (2020)). Los motivos de su elección se deben a la gran variedad de bibliotecas de código abierto que ofrece Python para el aprendizaje automático, además de la manipulación y visualización de datos.

Por otro lado, Python es fácilmente escalable, es decir, se puede utilizar tanto en proyectos pequeños como grandes. Esto se debe a que da soporte a tecnologías que permiten el procesamiento distribuido, como por ejemplo, Apache Spark.

Dentro de Python, se debe resaltar el uso de la librería *scikit-learn*, que se ha empleado para la creación de los modelos bayesianos, así como su entrenamiento, tratamiento de los datasets, testeo del rendimiento y realización de predicciones.

4.2.1.1. Módulos descartados

No obstante, antes de seleccionar a scikit-learn como módulo principal, se estudió la posibilidad de utilizar otros módulos alternativos para la creación de la red de Bayes. Estos son *pgmpy* (Ankan y Panda (2015)), *pybnesian* (Atienza et al. (2022)) y *pomegranate* (Schreiber (2014)). Pero finalmente se descartaron.

En el caso de los dos últimos, *pybnesian* y *pomegranate*, se descartaron por la complejidad de su sintaxis.

Con respecto a *pgmpy*, era la opción principal para aplicarlo al proyecto. De hecho, se intentó desarrollar una primera versión del Sistema Experto utilizando este módulo para la creación de las redes bayesianas. Sin embargo, durante las pruebas realizadas, la implementación presentaba severos problemas con la capacidad de memoria local. Esto era debido a la elevada cantidad de variables y valores que poseían los modelos. Esto provocaba que los procesos de creación y entrenamiento tardasen demasiado tiempo, llegando incluso a no completarse y entrar en un bucle. Por consiguiente, se tuvo que descartar dicho módulo.

4.2.2. Django framework

Se empleó Django, un conocido framework desarrollado en Python, para crear el proyecto de backend de la aplicación, y se configuró para que realizase las funciones de un servidor API REST. De esta forma, el proyecto de Django podría recibir peticiones del cliente y devolver los datos correspondientes en la respuesta.

4.2.3. React Native

React es una biblioteca de Javascript de código abierto que permite crear interfaces de usuario, tanto para aplicaciones de escritorio como para teléfonos móviles. Fue desarrollada y lanzada por Facebook en el año 2013, aunque también es usado por otras aplicaciones, como Instagram, Whatsapp, X, entre otros.

Se decidió este lenguaje por delante de otros lenguajes de desarrollo de aplicaciones, como Kotlin, por la falta de familiaridad con dicho lenguaje, o Java, por su incompatibilidad con los distintos sistemas operativos. Esto último, es una de las características más

destacadas de React para nuestro proyecto. Permite que se puedan desarrollar aplicaciones móviles para Android, iOS y web (Wikipedia (2024a)).

Para facilitar y agilizar el desarrollo de la aplicación, se ha propuesto usar un framework de React, ya que traen varios componentes básicos implementados, como botones, enlaces, formularios, etc. para ahorrar tiempo y esfuerzo. Existen diversos framework de React, cada una enfocada en un propósito específico.

- **React Native:** ampliamente utilizado en aplicaciones móviles, ya que permite a los desarrolladores construir aplicaciones nativas para iOS y Android utilizando React.js y JavaScript.
- **React Bootstrap:** es un framework que implementa a su vez el framework CSS de Bootstrap, que trae de por sí un abanico de estilos previamente diseñados y que el desarrollador sólo necesita declararlos dentro del código.
- **Next.js:** se utiliza para construir aplicaciones web de gran tamaño. Una de sus características destacables es que ofrece funcionalidades de enrutamiento y renderización del lado del servidor (SSR) y la generación de sitios estáticos.
- **Gatsby:** es un framework que usa GraphQL para gestionar los datos de la aplicación, además de que cuenta con una gran variedad de plugins para extender sus funciones. Es especialmente popular para la creación de blogs, sitios web de comercio electrónico y portafolios.

En este caso, se seleccionó React Native React (2024) como framework de desarrollo. El motivo principal de esta decisión es porque, de todas las variantes de React, esta es la más enfocada a crear aplicación móviles nativas, tanto para Android como para iOS.

4.2.4. Expo framework

Expo es un framework creado específicamente para crear proyectos de React Native. En este caso, se empleó para crear el proyecto de frontend de la aplicación. Cumple la función de cliente, por lo que ofrece una interfaz web y móvil responsive con la que el usuario puede interactuar, y realiza las peticiones correspondientes al servidor para obtener una serie de datos en concreto, o para realizar operaciones de inserción, actualización o eliminación de un dato.

Capítulo 5

Conclusiones y Trabajo Futuro

5.1. Conclusiones

En conclusión, el objetivo primordial del proyecto es reducir y prevenir el suicidio en adolescentes, una problemática creciente en estos últimos años. Con esa finalidad, se busca poder identificar y predecir con antelación las conductas autolesivas en los jóvenes, para avisar a los profesionales de la salud y que puedan atender a la potencial víctima lo más pronto posible.

Con el desarrollo de este Sistema Experto, se busca cumplir con dicho objetivo mediante la aplicación de la inteligencia artificial, más concretamente, de las redes bayesianas, al campo de la salud, método que se está comenzando a utilizar más en este ámbito.

Finalmente, el incremento de este tipo de proyecto que implementan la técnica de las redes bayesianas ayudarán a contribuir al campo de la salud mental, ayudando a los profesionales realizar diagnósticos con mayor precisión, y permitiendo a los jóvenes y familiares pueden recibir una predicción del modelo, que se hará con gran exactitud ya que estará basado en datos de otros pacientes anteriores, que se habrán usado durante el entrenamiento y el testeo.

5.2. Trabajo a futuro

Con respecto al trabajo a futuro, se ha planteado la posibilidad de llevar el trabajo del Sistema Experto a la nube. La velocidad y el rendimiento de los entrenamientos de los modelos han sido de los principales inconvenientes a la hora de escoger una librería de Python para desarrollar el Sistema Experto, ya que la gran mayoría necesitaban demasiado tiempo para realizar los procesos de creación, entrenamiento, testeo y predicción del modelo en una máquina local. En algunos casos, ha provocado fallos en la ejecución. Con esta solución, se podría distribuir la carga de trabajo entre varios servidores que tengan mayor capacidad que una máquina local, así como el almacenamiento, agilizando el trabajo y obteniendo respuestas más rápidas de los modelos.

Por otro lado, se podrían aplicar ciertas mejoras al Sistema Experto. A pesar de que se han podido obtener unos resultados buenos con los entrenamientos de los modelos, se han tenido que recurrir a datasets generados a través de un LLM, en lugar de emplear los datasets de los psicólogos que colaboraban en este proyecto. Se espera que en el futuro, se puedan obtener los datasets originales de los psicólogos, lo que permitiría diseñar un

modelo o modelos más preciso, y acorde a sus intereses y necesidades. También evitarían posibles incoherencias en los datos, lo que mejoraría el entrenamiento. Además, con esta mejora se podría añadir más variables y valores al modelo, calibrandólos más y teniendo en cuenta nuevos factores que afecten a la salud mental de los jóvenes.

Finalmente, en el tema de la aplicación, una de las principales mejoras es la migración de la base de datos. Actualmente, la base de datos que se está utilizando es una base de datos relacional, que es SQLite. Este tipo de base de datos puede almacenar datos en disco, y es de utilidad para una aplicación que vaya a manejar pocos datos. Sin embargo, a largo plazo, puede ser contraproducente el seguir usándolo como forma de almacenamiento principal. Es por eso, que se está planeando en un futuro, migrar la base de datos orientada a grafos, que es una base de datos no relacional. Con esta modalidad, se pueden establecer relaciones más fácilmente entre los padres, jóvenes y profesionales. También se puede alojar esta base de datos en un servidor de datos externo, en lugar de crearlo y guardar la información en disco. Aparte de esto, también nos permitiría poder emplear medidas adicionales de seguridad y protección de los datos, como el enmascaramiento de datos.

Capítulo 6

Introduction

6.1. Motivation

In recent years, suicide has become a global health problem, being one of the leading causes of unnatural death among young people and adults.

This is why its prevention and mitigation has presented a challenge for health professionals, since this problem affects both the individuals themselves, as well as their family members and close friends.

For this reason, in recent years, efforts and economic resources have been focused on the creation of suicide prevention programs. In these plans, it is essential to make diagnoses as early as possible in order to detect mental symptoms in the person in time, and therefore, be able to offer treatment. The problem is that many times, people at risk of adopting these suicidal and self-injurious behaviors are not aware that they suffer from them or that they are at risk of suicide or self-injury, so they do not attend prevention plans. It leads to the aggravation of the problem, until it ends in a fatal outcome. It is important to be able to inform the person in time about his or her mental health condition automatically, quickly and effectively, so that he or she is aware of the risk and therefore be able to ask for help in advance.

Against this backdrop, the aim is to develop tools to prevent this type of behavior in young people. This is where artificial intelligence begins to play a fundamental role. The artificial intelligence is being used in the field of mental health, being of great use in suicide prevention and to support professionals in achieving an accurate assessment of suicide risk. In this way, potential patients with these self-injurious or non-self-injurious behaviors can be treated at very early stages in the development of these behaviors.

The main reason for the choice of this topic is because of the relevance that this subject has been acquiring over the years. In addition, it is a project that covered the section of web and mobile application development, which I am more familiar with.

6.2. Goals

The main objective of the project is to develop a platform that, through a series of questionnaires, can make predictions by querying an Expert System. To achieve this main objective, the following project objectives were established throughout the development of

the thesis.

- To design and validate an Expert System for the assessment of the risk of suicidal and non-suicidal self-injurious behaviors in adolescent population (12-21 years old).
- The Expert System should be able to predict those adolescents and young people at high risk.
- Implement a way to prevent future self-injurious behaviors through professional monitoring and supervision.
- Design and develop a multi platform application that can host the predictions of the Expert System and implement the above objectives.

6.3. Work Plan

Based on the objectives set out in the previous section, the following work plan was established.

- Selection of the type of Expert System. A search of the different types of Expert Systems available will be carried out and the most suitable one will be selected for each type.
- Understanding the SIVARIA project. Access the SIVARIA project website and investigate and understand the questionnaires on the page, which will be useful to design the Expert System models.
- Selection of tools and libraries to implement the models for the selected Expert System.
- Design and implementation of the models using the selected tools and libraries.
- Train and validate the models created through several cycles or iterations. The results of the iterations will be interpreted and the parameters and input data will be improved until an acceptable performance of the Expert System models is achieved.
- Study of tools and technologies for the creation of the mobile application.
- Creation of the use cases for the development of the application.
- Development of the application is carried out based on the use cases specified by using the tools studied.

All the code made during the development of the project can be found in the following Github repository.

<https://github.com/NILGroup/TFM-2324-SIVARIA>

It has MIT license, and the repository is public. Inside the repository there are 4 main folders:

- **DiagramBayesSketches:** there are the sketches of the Bayes diagrams made.

- **Memory**: folder with the project memory.
- **app**: the two multiplatform application projects are located here, both the frontend and backend projects.
- **scripts**: all the scripts used for the training of the model are located here. It has a series of Jupyter notebooks with the steps of the trainings performed, divided into versions as they appear throughout the memory. In addition, it has a prototype of the Expert System that is located inside the backend server in the *app* folder. You can also use those Python scripts to perform the same tests as in the notebooks, although in this case the operations in the Expert System prototype is command-based.

Capítulo 7

Conclusions and Future Work

7.1. Conclusions

In conclusion, the main objective of the project is to reduce and prevent suicide in adolescents, a growing problem in recent years. To this end, the aim is to be able to identify and predict self-harming behaviour in young people in advance, in order to warn health professionals so that they can attend to the potential victim as soon as possible.

With the development of this Expert System, the aim is to fulfil this objective by applying artificial intelligence, more specifically Bayesian networks, to the field of health, a method that is beginning to be used more in this field.

Finally, the increase of these type of projects that implement the Bayesian Networks technique will contribute in the mental health field, helping professionals in their diagnoses, and allowing young people and relatives receive a highly accurate prediction from the model, based on training data coming from previous patients that had similar pathologies.

7.2. Future work

With respect to future work, the possibility of taking the work of the Expert System to the cloud has been raised. The speed and performance of the training of the models have been the main drawbacks when choosing a Python library to develop the Expert System, since most of them required too much time to perform the processes of creating, training, testing and predicting the model on a local machine. In some cases, it has led to execution failures. With this solution, the workload could be distributed among several servers that have more capacity than a local machine, as well as the storage, speeding up the work and obtaining faster responses from the models.

On the other hand, certain improvements could be applied to the Expert System. Although we have been able to obtain good training results, I have had to resort to datasets generated by an LLM, instead of using the datasets of the psychologists that were supposed to be collaborating in this project. It is hoped that in the future, it will be possible to obtain the original datasets from the psychologists, which would allow the design of a more accurate models, and according to their interests and needs. This solution would avoid possible inconsistencies in the data, improving model training. In addition, with this improvement, more variables and values could be introduced to the model, calibrating them further and taking into account new factors affecting the mental health of young people.

Finally, regarding the application, one the main improvements is the migration of the database. Currently, the database that is used is SQLite. This type can store data on disk, and is useful to handle small data. Nevertheless, as a long-term solution, could be counterproductive to continue using it as the primary form of storage. For this reason, in the future, it is planned to move to a graph-oriented database, which is non-relational. With this approach, relationships between parents, young people and professionals can be established more easily. Besides, it is possible to host on an external data server, instead of creating it and saving the information on disk. Apart from this, moving to an external data server will allow employing additional security and protection measures, such as data masking.

Bibliografía

- ANKAN, A. y PANDA, A. pgmpy: Probabilistic graphical models using python. En *SciPy*, páginas 6–11. Citeseer, 2015.
- ATIENZA, D., BIELZA, C. y LARRAÑAGA, P. Pybnesian: An extensible python package for bayesian networks. *Neurocomputing*, vol. 504, páginas 204–209, 2022.
- AXIOS. *Axios Documentation*, 2024. Último acceso: 3 de agosto de 2024.
- DEL BARRIO, V., RAMÍREZ-UCLÉS, I., ROMERO, C. y CARRASCO, M. Á. Adaptación del child-parq/control: versiones para el padre y la madre en población infantil y adolescente española. *Acción psicológica*, vol. 11(2), páginas 27–46, 2014.
- DJANGO. *Django Documentation*, 2024. Último acceso: 3 de agosto de 2024.
- EXPO. *Expo Documentation*, 2024. Último acceso: 3 de agosto de 2024.
- FERNÁNDEZ-PINTO, I., SANTAMARÍA, P., SÁNCHEZ-SÁNCHEZ, F., CARRASCO, M. y DEL BARRIO, V. Sistema de evaluación de niños y adolescentes. *SENA. Madrid: TEA Ediciones*, 2015.
- FONSEKA, T. M., BHAT, V. y KENNEDY, S. H. The utility of artificial intelligence in suicide risk prediction and the management of suicidal behaviors. *Australian & New Zealand Journal of Psychiatry*, vol. 53(10), páginas 954–964, 2019.
- GARNEFSKI, N. y KRAAIJ, V. Cognitive emotion regulation questionnaire—development of a short 18-item version (cerq-short). *Personality and individual differences*, vol. 41(6), páginas 1045–1053, 2006.
- GILBERT, P. y ALLAN, S. The role of defeat and entrapment (arrested flight) in depression: an exploration of an evolutionary view. *Psychological medicine*, vol. 28(3), páginas 585–598, 1998.
- GROSAN, C. y ABRAHAM, A. *Rule-Based Expert Systems*, páginas 149–185. Springer Berlin Heidelberg, Berlin, Heidelberg, 2011. ISBN 978-3-642-21004-4.
- HILL, R. M., REY, Y., MARIN, C. E., SHARP, C., GREEN, K. L. y PETTIT, J. W. Evaluating the interpersonal needs questionnaire: Comparison of the reliability, factor structure, and predictive validity across five versions. *Suicide and Life-Threatening Behavior*, vol. 45(3), páginas 302–314, 2015.

- KOLODNER, J. L. An introduction to case-based reasoning. *Artificial intelligence review*, vol. 6(1), páginas 3–34, 1992.
- ORTEGA-RUIZ, R., DEL REY, R. y CASAS, J. A. Evaluar el bullying y el cyberbullying validación española del ebip-q y del ecip-q. *Psicología educativa*, vol. 22(1), páginas 71–79, 2016.
- PEDREGOSA, F., VAROQUAUX, G., GRAMFORT, A., MICHEL, V., THIRION, B., GRISEL, O., BLONDEL, M., PRETTENHOFER, P., WEISS, R., DUBOURG, V., VANDERPLAS, J., PASSOS, A., COURNAPEAU, D., BRUCHER, M., PERROT, M. y DUCHESNAY, E. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, vol. 12, páginas 2825–2830, 2011.
- PÉREZ, E. P., MONJE, M. R., ALONSO, F. G., GIRÓN, M. F., LÓPEZ, M. P. y ROMERO, J. C. Validación de un instrumento para la detección de trastornos de control de impulsos y adicciones: el multicage cad-4. *Trastornos adictivos*, vol. 9(4), páginas 269–278, 2007.
- PYTHON. *Documentación de Python*. Versión electrónica, 2020. Disponible en <https://docs.python.org/3.9/> (último acceso, Febrero, 2024).
- REACT. *React Native Documentation*, 2024. Último acceso: 3 de agosto de 2024.
- SALSMAN, N. y LINEHAN, M. M. Dialectical-behavioral therapy for borderline personality disorder. *Primary Psychiatry*, vol. 13(5), página 51, 2006.
- SÁNCHEZ, M. I. S., DE PEDRO, M. M. y IZQUIERDO, M. G. Propiedades psicométricas de la versión española de la escala de resiliencia de 10 ítems de connor-davidson (cd-risc 10) en una muestra multiocupacional. *Revista latinoamericana de psicología*, vol. 48(3), páginas 159–166, 2016.
- SCHREIBER, J. pomegranate. *Software download*. URL <https://github.com/jmschrei/pomegranate>, 2014.
- SUCAR, L. E. y TONANTZINTLA, M. Redes bayesianas. *Aprendizaje Automático: conceptos básicos y avanzados*, vol. 77, página 100, 2006.
- URBAN, M. *An introduction to LATEX*. TEX users group, 1986.
- WIKIPEDIA. React — wikipedia, la enciclopedia libre. 2024a. [Internet; descargado 11-abril-2024].
- WIKIPEDIA. Sistema experto — wikipedia, la enciclopedia libre. 2024b. [Internet; descargado 17-mayo-2024].

Apéndice A

Guía de instalación

Todo el desarrollo del TFM, tanto código como bocetos de los diagramas, se encuentran en el repositorio online de Github.

A.1. Contenido del repositorio

Al descargarse el repositorio, se pueden observar tres carpetas principales:

- **DiagramBayesSketches**: esta carpeta contiene los bocetos de los diagramas de Bayes de todos los modelos: autoinforme, familia y profesional. Además, aparte hay una subcarpeta con el boceto de cada grupo de variables, para poder ver con mayor claridad cómo está compuesto cada grupo.
- **Memoria**: en ella se encuentra la presente memoria.
- **scripts**: en esta carpeta se encuentran los cuadernos de Jupyter de los entrenamientos, junto con un prototipo del Sistema Experto. Ambos componentes realizan la misma función de entrenamiento y testeo.
- **app**: en esta carpeta se encuentran los dos proyectos de la aplicación, tanto el frontend como el backend.

A.1.1. Scripts

Constan de los siguientes archivos:

- **exceptions**: carpeta que contiene una serie de excepciones personalizadas creadas para el Sistema Experto.
- **configFiles**: carpeta donde se almacenará los archivos de guardado de los modelos del Autoinforme, Familia y Profesional.
- **datasets**: carpeta donde se encuentran los datasets de prueba creados de forma manual, aleatoria y artificial para entrenar a los modelos, y así, probar el rendimiento del Sistema Experto.
- **csvV1Creator**: en esta carpeta se encuentran los scripts que se utilizaron en la primera versión para crear los datasets aleatorios.

- **jupyterNotebooks**: contiene los cuadernos de Jupyter con el código de los entrenamientos de las 4 versiones.
- *expertSystem.py*: contiene la clase Sistema Experto, que utiliza el módulo scikit learn para realizar las funciones de predicción, entrenamiento y testeo. Se encarga de construir el modelo inicial a partir del tipo de modelo especificado por el usuario, que puede ser el modelo del autoinforme, el de la familia o el de los profesionales. Además, también posee una función para guardar el modelo entrenado en un archivo .sav y que lo puede colocar en las carpetas *scripts/configFiles/autoinforme*, *scripts/configFiles/familia* o *scripts/configFiles/profesional*, también dependiendo del tipo de modelo. Por último, habrá una carpeta **configFilesJupyter**, donde se almacenarán los modelos entrenados, dependiendo del tipo de modelo.
- *controller.py*: ejerce de Controlador que es capaz de utilizar métodos del Sistema Experto a través de una serie de comandos:
 - -h: muestra un mensaje de ayuda que explica los comandos disponibles del controller.py. Tiene un formato similar a las páginas *man* de Linux.
 - -mt: muestra y establece el tipo de modelo seleccionado por el usuario (autoinforme, familia, profesional).
 - -lst: devuelve una lista de archivos de guardado que hay según el tipo de modelo.
 - -fs: devuelve la información del archivo dado su nombre.
 - -rm: elimina un archivo de guardado dado su nombre.
 - -t: entrena al modelo. Es necesario que previamente se establezca el tipo de modelo (-mt). Además, debemos añadir en los parámetros el dataset que se va a usar para el entrenamiento. Devuelve una lista de verdaderos/falsos positivos y verdaderos/falsos negativos de cada tipo de desenlace.
 - -p: realiza la predicción de un dataset especificado a través del path por los parámetros del comando.
- *constants.py*: contiene las constantes que se usa en los dos scripts principales.
- *checker.py*: objeto estático Checker que contiene métodos para revisar el formato del dato de una variable.
- *decoder.py*: objeto estático Decoder con métodos que decodifica un dataframe recibido.
- *fileManager.py*: objeto FileManager con métodos para administrar la creación y eliminación de archivos de guardado de los modelos.
- *main.py*: script principal que crea el objeto Controlador y ejecuta el comando.

A.2. Instrucciones de ejecución

Una vez descargado el repositorio, se necesita crear un entorno virtual para ejecutar los scripts de Python. Se recomienda instalar Anaconda, ya que permite crear entornos virtuales que incluyen una serie de paquetes que son utilizados en el código desarrollado, por ejemplo, Numpy, que ya se encuentra instalado en el entorno de Anaconda.

A.2.1. Entornos de Python con Anaconda

Por defecto, Anaconda tiene un entorno general llamado *base* (*root*). Sin embargo, también existe la opción de crear un entorno nuevo aparte del principal. Para ello, dentro de Anaconda hay que ir a la sección *Environments*, y una vez ahí, dar al botón *Create*. Aparecerá una ventana donde se especifica la versión de Python. En mi caso, se ha utilizado la versión 3.9.19.

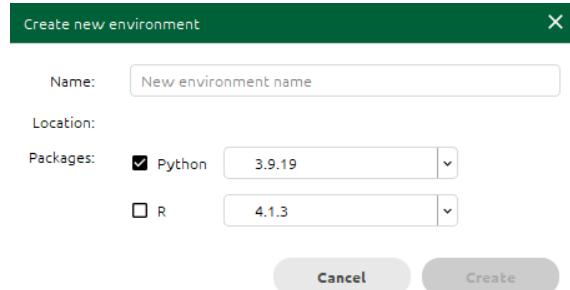


Figura A.1: Ventana de creación de nuevo entorno de Python en Anaconda

Finalmente, se termina el proceso dando botón *Create*.

A.2.2. Conexión del entorno virtual con Visual Studio Code

Cuando se haya creado el entorno, el siguiente paso es conectar el nuevo entorno, o el *base* si no se ha creado ninguno, a Visual Studio Code.

Para ello, se debe abrir el panel de comandos (CTRL+Shift+P) y seleccionar la opción

```
>Python:Select Interpreter
```

Una vez ahí, seleccionamos el entorno. Finalmente, se abre un terminal que se ejecuta dentro del entorno virtual y desde dónde se podrán descargar los paquetes del proyecto y ejecutar los scripts.

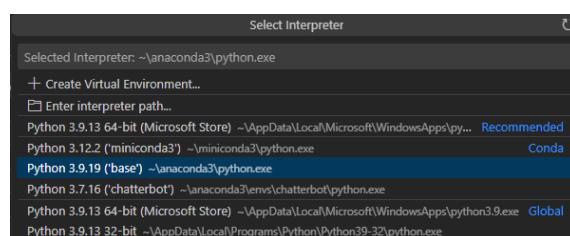


Figura A.2: Panel de selección de entorno de Python en Visual Studio Code

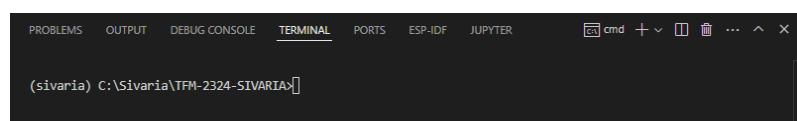


Figura A.3: Consola del nuevo entorno de Python

A.2.3. Instalación de los paquetes

A.2.3.1. Frontend

En esta sección, se explicarán los pasos necesarios para poder ejecutar el servidor frontend del repositorio en local. Todos los comandos que se mostrarán a continuación se deben ejecutar desde la consola del VSCode.

1. Instalar NodeJS, que habilita el uso del comando npm, lo que permite instalar las dependencias del proyecto.
2. Instalar Expo en el móvil, para probar la aplicación en el móvil.
3. Instalar Ngrok, que permite crear un túnel en un dispositivo externo y el servidor backend local.
4. Ir al proyecto *app/sivaria/*
5. Instalar las dependencias de React Native y Expo mediante el siguiente comando:

```
(sivaria) C:\Sivaria\TFM-2324-SIVARIA\app\sivaria>npm install expo
```

6. Crear el archivo de variables de entorno: se debe crear un archivo .env en el directorio raíz de este proyecto. Este .env debe contener los siguientes valores.
 - API_SERVER_MOBILE : enlace https que genera Ngrok al ejecutarse. Este enlace se usa cuando se quiere probar la aplicación desde el dispositivo móvil.
 - API_SERVER_WEB : servidor local para probar la aplicación desde un navegador web.

De esta manera, el frontend ya estaría listo para iniciarse en cualquier momento. Para arrancar el servidor, se ejecuta el siguiente comando dentro del directorio anteriormente especificado.

```
(sivaria) C:\Sivaria\TFM-2324-SIVARIA\app\sivaria>npm start
```

Este comando inicia el servidor del proyecto, a la cual se puede acceder de dos maneras: a través del navegador o a través de un dispositivo móvil.

Para ver la app desde un navegador, se debe pulsar la tecla *w* en la consola del servidor, y automáticamente se abre el navegador predeterminado y se realiza el build de la aplicación.

Para ver la app desde el móvil, es necesario utilizar e iniciar la aplicación de Expo. El servidor genera dos formas de ejecutar la aplicación: con un código QR o un enlace expo, como se puede observar en las Figuras A.4 y A.5.

Es importante destacar que la máquina local donde se aloja el servidor API y la aplicación móvil deben estar conectados a la misma conexión WiFi. De lo contrario, la aplicación no se podrá conectar.

```
C:\Users\aldai\Downloads\Universidad\Curso23_24\TFM\EjemploReact\MyReactNative>npm run web
npm WARN config global `--global`, `--local` are deprecated. Use `--location=global` instead.

> myreactnative@1.0.0 web
> expo start --web

Starting project at C:\Users\aldai\Downloads\Universidad\Curso23_24\TFM\EjemploReact\MyReactNative
Starting Metro Bundler
The following packages should be updated for best compatibility with the installed expo version:
  react-native@0.73.2 - expected version: 0.73.4
Your project may not work correctly until you install the correct versions of the packages.


> Metro waiting on exp://192.168.1.40:8081
> Scan the QR code above with Expo Go (Android) or the Camera app (iOS)
> Web is waiting on http://localhost:8081

> Using Expo Go
> Press s | switch to development build

> Press a | open Android
> Press w | open web

> Press j | open debugger
> Press r | reload app
> Press m | toggle menu
> Press o | open project code in your editor

> Press ? | show all commands

Logs for your project will appear below. Press Ctrl+C to exit.
Web Bundled 3534ms (C:\Users\aldai\Downloads\Universidad\Curso23_24\TFM\EjemploReact\MyReactNative\node_modules\expo\appEntry.js)
```

Figura A.4: Consola del servidor frontend

A.2.3.2. Scripts de Python y servidor backend

En esta sección, se explicarán los pasos necesarios para poder ejecutar el servidor backend del repositorio en local, además de los scripts de Python.

1. Navegar al directorio *app/backend*
2. Instalar los paquetes necesarios de Python.

```
(sivarria) C:\Sivarria\TFM-2324-SIVARIA\app\backend> pip install -r requirements.txt
```

A partir de aquí se pueden ejecutar los scripts de Python. Sin embargo, para poder ejecutar el servidor backend, es necesario realizar más pasos.

3. Crear archivo de variables de entorno. Al igual que en el frontend, el backend también tiene sus variables de entorno, a través del archivo .env. Este fichero se debe crear desde el directorio raíz del servidor de backend (*app/backend*). Sus variables deben ser las siguientes:

- DEFAULT_FROM_EMAIL : dirección del correo electrónico desde donde se enviarán los correos de la aplicación.
- EMAIL_HOST_PASSWORD : contraseña de la cuenta de correo electrónico.
- FRONTEND_SIVARIA_URL : URL del frontend.
- SECRET_KEY : clave secreta usada para firmar el hash de las contraseñas y los tokens. Se debe generar de forma local. Para eso, se debe abrir la consola de python e introducir las siguientes instrucciones:

```
(sivarria) C:\Sivarria\TFM-2324-SIVARIA>python
>>> from django.core.management.utils import get_random_secret_key
>>> print(get_random_secret_key())
```

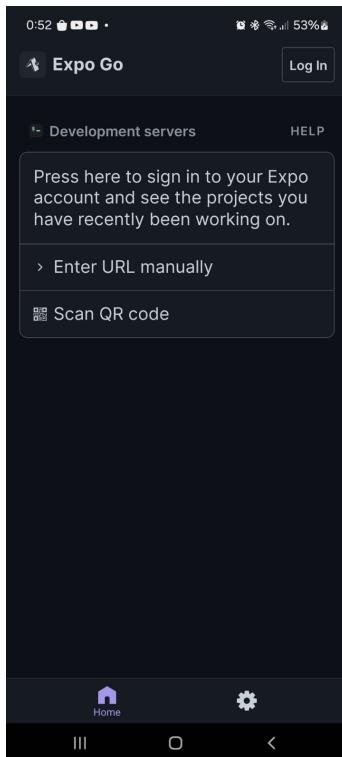


Figura A.5: Interfaz de la aplicación de Expo

Esto devolverá la clave secreta generada. Este valor se copia y pega en la variable SECRET_KEY.

4. Crear migraciones (opcional).

```
(sivarria) C:\Sivarria\TFM-2324-SIVARIA\app\backend>python manage.py makemigrations
```

5. Ejecutar las migraciones.

```
(sivarria) C:\Sivarria\TFM-2324-SIVARIA\app\backend>python manage.py migrate
```

6. Crear un superusuario, para poder acceder al panel de administración de Django.

```
(sivarria) C:\Sivarria\TFM-2324-SIVARIA\app\backend>python manage.py createsuperuser
```

En cuanto se haya creado el superusuario se puede probar su acceso introduciendo el email y la contraseña en el panel de administración, que se puede acceder a través de la siguiente URL:

SERVER_URL/admin

Donde *SERVER_URL* es la URL del servidor cuando se ejecuta.

Por ejemplo, *http://127.0.0.1:8000/admin*.

De esta manera, ya se podría ejecutar el servidor backend, ejecutando el siguiente comando.

(sivarria) C:\Sivarria\TFM-2324-SIVARIA\app\backend>python manage.py runserver

Además, ya se podrían ejecutar los endpoints del API, a través del comando *curl*, desde el navegador accediendo a la URL exacta del endpoint, o usando la herramienta Postman, que es la usé para acceder y probar los endpoints.

Dentro del directorio *app/backend* se encuentra una colección JSON y las variables de entorno que se puede importar al Postman, y que contiene todos los endpoints del API. No obstante, para ejecutarlos, la mayoría de ellos requieren de un token de autorización, y para ello, se debe ejecutar previamente el endpoint del login: *BASE_URL/sivarria/v1/user/login*. Este endpoint devuelve en la respuesta el token de autorización dentro del campo *token* (Figura A.6).

Por lo tanto, cada vez que se quiera ejecutar un endpoint, el token debe estar incluido en la cabecera de la petición, como se puede ver en la Figura A.7.

```

POST [BASE_URL]/sivarria/v1/user/login
{
  "email": "ranking123@hotmail.es",
  "password": "test"
}
{
  "status": [
    "ok"
  ],
  "message": [
    "Usuario logeado correctamente"
  ],
  "data": {
    "id": 15,
    "first_name": "Aldeair",
    "last_name": "Maldonado Profesional",
    "email": "ranking123@hotmail.es",
    "code": "AMP24861998",
    "phone": "646532196",
    "rol": {
      "id": 4,
      "slug": "profesional",
      "description": "profesional",
      "code": "pr"
    },
    "birth_date": "1998-06-24",
    "token": "7a5e1a9e2937bec22c5a5411e1a4fa8ca2882008"
  }
}

```

Figura A.6: Login endpoint en Postman

Key	Value	Description
Authorization	Token 7a5e1a9e2937bec22c5a5411e1a4fa8ca2882008	

Figura A.7: Cabecera de la petición para los endpoints con autorización

Apéndice **B**

Imágenes adicionales

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 48 entries, 0 to 39
Data columns (total 55 columns):
 #   Column          Non-Null Count  Dtype  
 --- 
 0   Index           48 non-null     int64  
 1   Edad            48 non-null     object 
 2   Curso           48 non-null     object 
 3   Peso            48 non-null     int64  
 4   Altura          48 non-null     int64  
 5   Sexo asignado  48 non-null     object 
 6   Transexuales   48 non-null     object 
 7   Tratamiento del rendimiento academico
 8   Situacion laboral madre
 9   Situacion laboral padre
 10  Nivel profesional madre
 11  Nivel profesional padre
 12  Nivel de autopercpcion masculina
 13  Nivel de autopercpcion femenina
 14  Nivel de percepcion masculina externa
 15  Nivel de percepcion femenina externa
 16  Tratamiento psiquiatrico previo
 17  Presenta enfermedad cronica
 18  Bullying victim
 19  Bullying perpetrador
 20  Cyberbullying victim
 21  Cyberbullying perpetrador
 22  Adiccion/abuso alcohol
 23  Adiccion/abuso sustancias
 24  Adiccion/internet
 25  Problemas interiorizados
 26  Problemas exteriorizados
 27  Problemas de contexto
 28  Problemas_recurso_psicologicos
 29  Fuente de discriminacion
 30  Nivel de resistencia/resiliencia
 31  Nivel de regulacion positiva
 32  Nivel de regulacion negativa
 33  Atrapamiento interno
 34  Atrapamiento externo
 35  Nivel perciplido de derrota o fracaso
 36  Sentido_de_pertenencia_frustrada
 37  Relacion_conflictiva_hijo_padre_madre
 38  Autoeficacia para el suicidio
 39  Madre adolescente
 40  Padre adolescente
 41  padres divorciados
 42  Familia monoparental
 43  Tratamiento psicologico padre/madre
 44  Adiccion padre/madre
 45  relaciones conflictivas hijo-padre/madre
 46  Familia reestructurada
 47  Busqueda informacion autolesion
 48  Compartir_en_rss5_pensamiento_autolesion
 49  peticion_de_ayuda_en_internet
 50  Realizar_autolesion_despues_de_ver_contenido
 51  Conocidos que comparten autolesion en Internet
 52  Contacito informacion autolesion
 53  Denuncia_autolesion_internet
 54  denuncia_internet
dtype: int64(11), object(44)
memory usage: 17.3+ KB
```

Figura B.1: Estructura del dataframe en la segunda versión de entrenamiento

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 123 entries, 0 to 122
Data columns (total 55 columns):
 #   Column          Non-Null Count  Dtype  
 --- 
 0   id              123 non-null     int64  
 1   edad            123 non-null     object 
 2   curso           123 non-null     object 
 3   peso            123 non-null     object 
 4   altura          123 non-null     object 
 5   sexo_asignado  123 non-null     object 
 6   transexuales   123 non-null     object 
 7   nivel_promedio_academico
 8   situacion_laboral_madre
 9   situacion_laboral_padre
 10  nivel_promocional_madre
 11  nivel_promocional_padre
 12  nivel_autopercepcion_masculina
 13  nivel_autopercepcion_femenina
 14  nivel_percepcion_masculina_externa
 15  nivel_percepcion_femenina_externa
 16  tratamiento_psiquiatrico_previo
 17  enfermedad_cronica
 18  bullying_victim
 19  bullying_perpetrador
 20  cyberbullying_victim
 21  cyberbullying_perpetrador
 22  adiccion_sustancias
 23  adiccion_internet
 24  problemas_interiorizados
 25  problemas_exteriorizados
 26  problemas_recursos_psicologicos
 27  problemas_contexto
 28  problemas_recurso_psicologicos
 29  fuente_discriminacion
 30  sentimiento_frustracion
 31  nivel_regulacion_positiva
 32  nivel_regulacion_negativa
 33  atrapamiento_interno
 34  atrapamiento_externo
 35  nivel_perciplido_fracaso
 36  sentido_pertenencia_frustrada
 37  percepcion_de_ser_una_carga
 38  relaciones_conflictivas_hijo_madre
 39  madre_adolescente
 40  padre_adolescente
 41  padres_divorciados
 42  tratamiento_psicologico
 43  tratamiento_psicologico_padre_madre
 44  adiccion_padre_madre
 45  relaciones_conflictivas_hijo_padre_madre
 46  adiccion_internet
 47  busqueda_informacion_autolesion
 48  compartir_en_rss5_pensamiento_autolesion
 49  peticion_de_ayuda_en_internet
 50  Realizar_autolesion_despues_de_ver_contenido
 51  tener_comunicacion_abierta_con_autolesion_internet
 52  contacto_informacion_autolesion
 53  denuncia_autolesion_internet
 54  denuncia_internet
dtype: int64(8), object(47)
memory usage: 53.6+ KB
```

Figura B.2: Estructura del dataframe del modelo del autoinforme en la tercera versión

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 107 entries, 0 to 106
Data columns (total 23 columns):
 #   Column           Non-Null Count Dtype  
 --- 
 0   id               107 non-null    int64   
 1   curso            107 non-null    object  
 2   situacion_laboral_madre 107 non-null    object  
 3   situacion_laboral_padre 107 non-null    object  
 4   nivel_profesional_madre 107 non-null    object  
 5   nivel_profesional_padre 107 non-null    object  
 6   problemas_interiorizados 107 non-null    object  
 7   problemas_exteriorizados 107 non-null    object  
 8   problemas_contexto      107 non-null    object  
 9   problemas_recursos_psicologicos 107 non-null    object  
 10  madre_adolescente     107 non-null    object  
 11  padre_adolescente     107 non-null    object  
 12  padres_divorciados    107 non-null    object  
 13  familia_monoparental 107 non-null    object  
 14  tratamiento_psicologico_padre_madre 107 non-null    object  
 15  adiccion_padre_madre  107 non-null    object  
 16  relaciones_conflictivas_hijo_padre_madre 107 non-null    object  
 17  familia_reconstruida  107 non-null    object  
 18  aceptacion_rechazo_parental 107 non-null    object  
 19  control_parental      107 non-null    object  
 20  situacion_economica_precaria 107 non-null    object  
 21  ingreso_familiar_mensual 107 non-null    object  
 22  desenlace           107 non-null    object  
dtypes: int64(1), object(22)
memory usage: 19.4+ KB
```

Figura B.3: Estructura del dataframe del modelo de las familias en la tercera versión

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 220 entries, 0 to 219
Data columns (total 22 columns):
 #   Column           Non-Null Count Dtype  
 --- 
 0   id               220 non-null    int64   
 1   curso            220 non-null    object  
 2   situacion_laboral_madre 220 non-null    object  
 3   situacion_laboral_padre 220 non-null    object  
 4   nivel_profesional_madre 220 non-null    object  
 5   nivel_profesional_padre 220 non-null    object  
 6   situacion_economica_precaria 220 non-null    object  
 7   ingreso_familiar_mensual 220 non-null    object  
 8   tratamiento_psiquiatrico_previo 220 non-null    object  
 9   madre_adolescente     220 non-null    object  
 10  padre_adolescente     220 non-null    object  
 11  padres_divorciados    220 non-null    object  
 12  familia_monoparental 220 non-null    object  
 13  tratamiento_psicologico_padre_madre 220 non-null    object  
 14  adiccion_padre_madre  220 non-null    object  
 15  relaciones_conflictivas_hijo_padre_madre 220 non-null    object  
 16  familia_reconstruida  220 non-null    object  
 17  supervision_parental_insuficiente 220 non-null    object  
 18  maltrato_al_adolescente 220 non-null    object  
 19  maltrato_a_la_pareja  220 non-null    object  
 20  duelo              220 non-null    object  
 21  desenlace           220 non-null    object  
dtypes: int64(1), object(21)
memory usage: 37.9+ KB
```

Figura B.4: Estructura del dataframe del modelo de los profesionales en la tercera versión

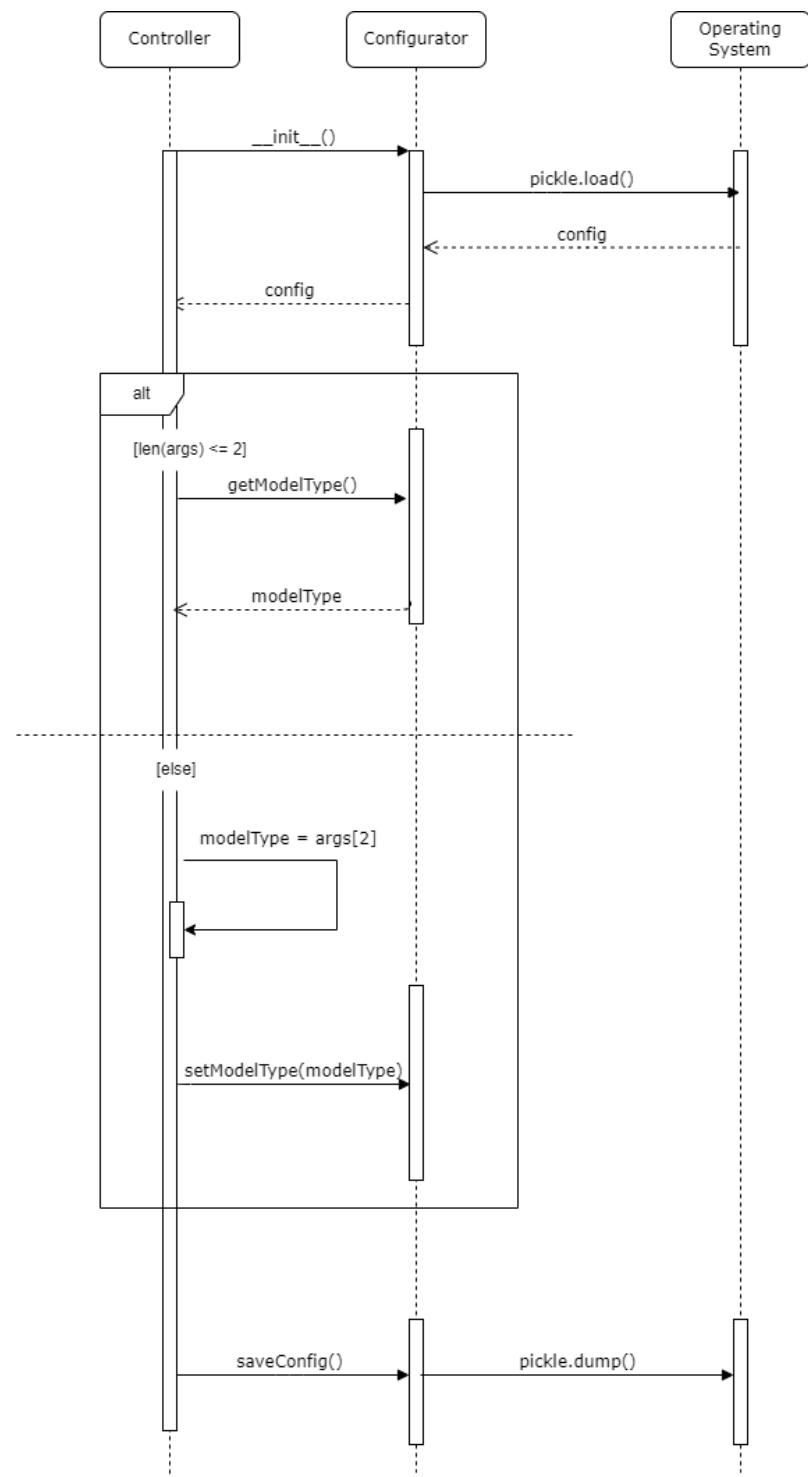


Figura B.5: Diagrama de secuencia del comando para establecer el tipo de modelo

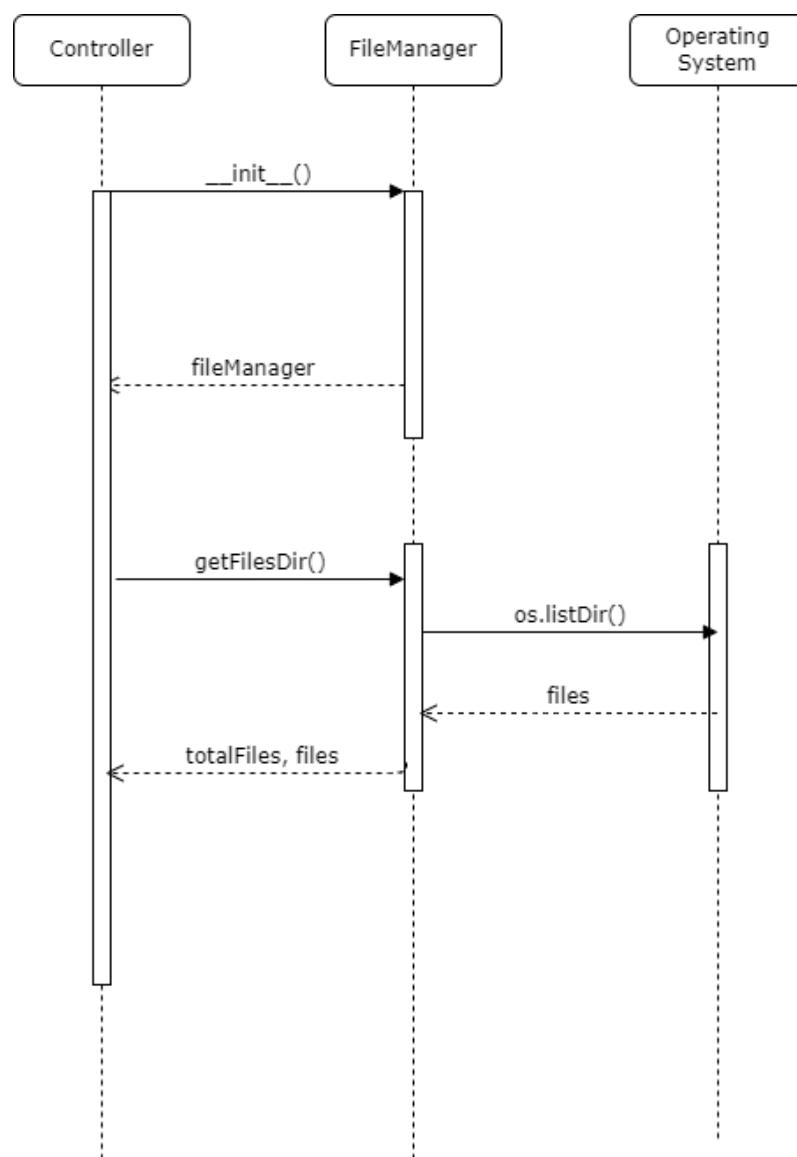


Figura B.6: Diagrama de secuencia del comando para listar los archivos de guardado

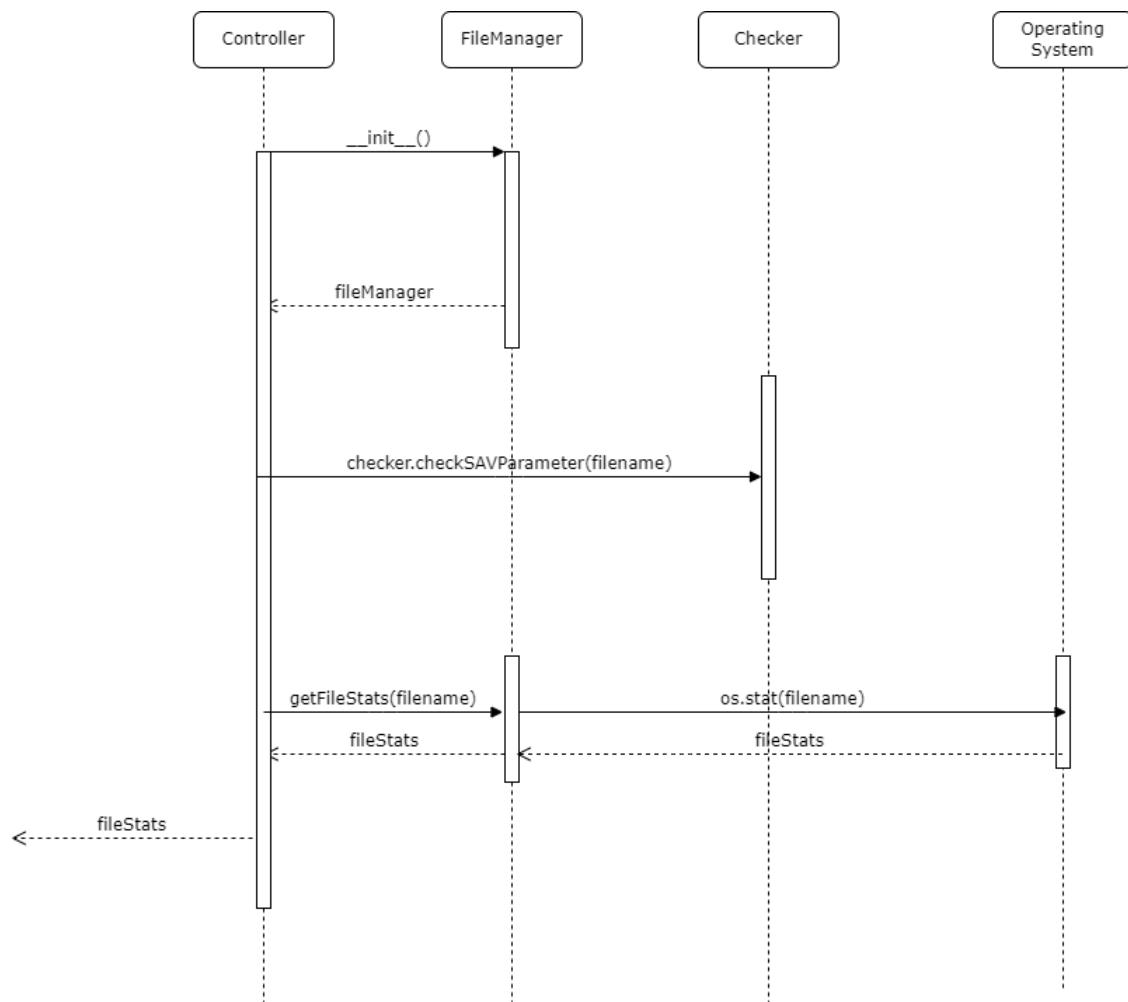


Figura B.7: Diagrama de secuencia del comando para mostrar la información del archivo de guardado

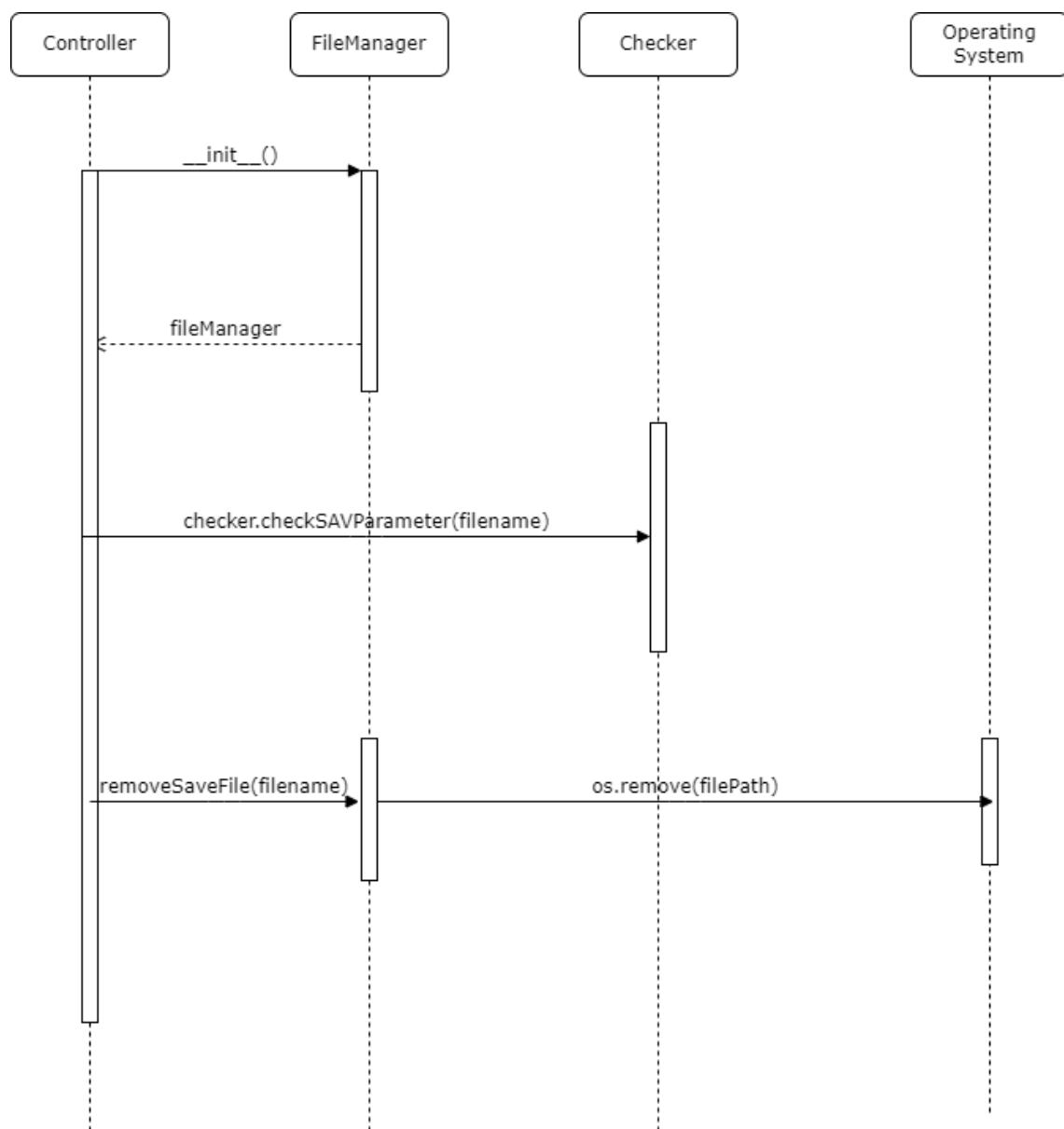


Figura B.8: Diagrama de secuencia del comando para eliminar un archivo de guardado

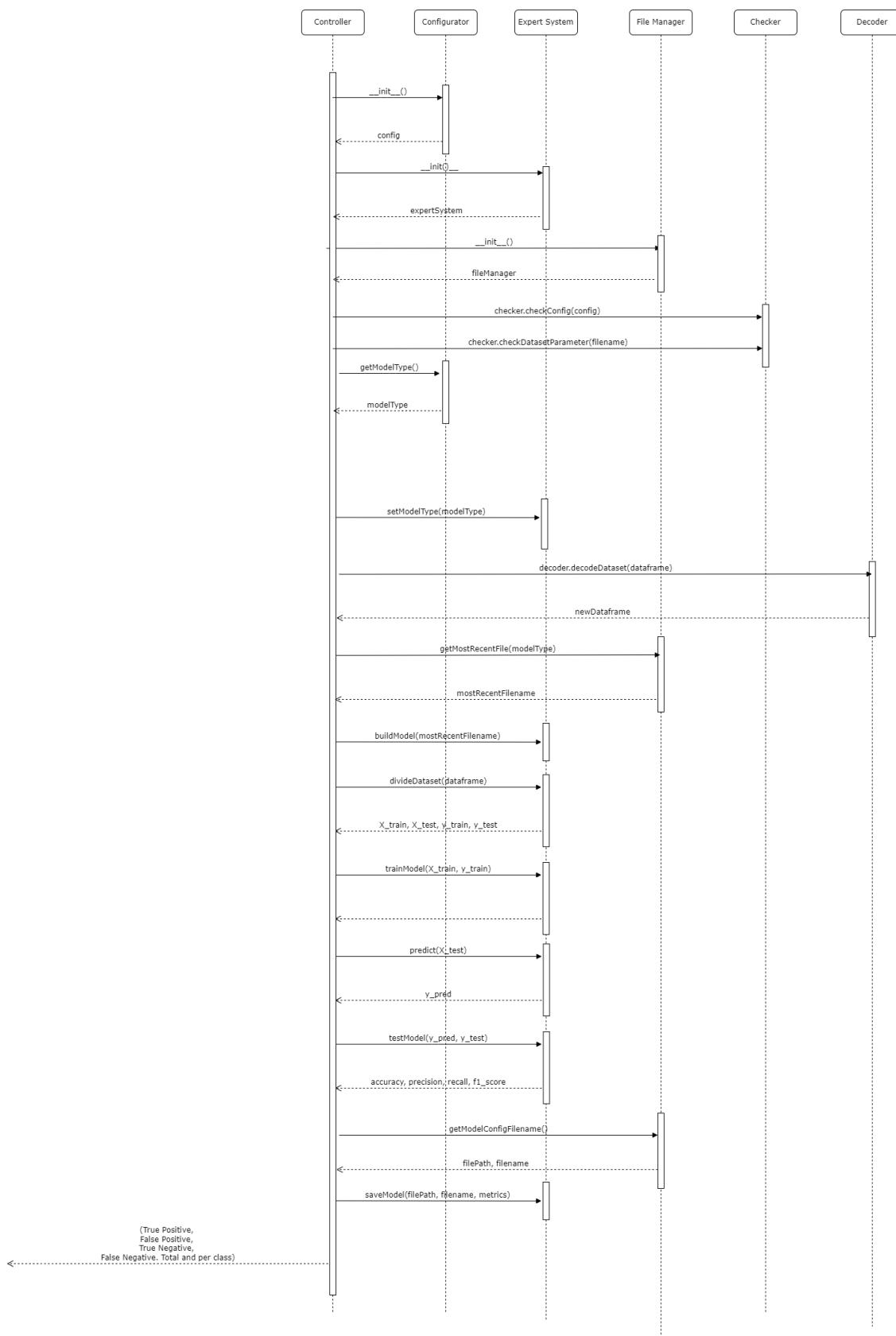


Figura B.9: Diagrama de secuencia del comando para entrenar al modelo

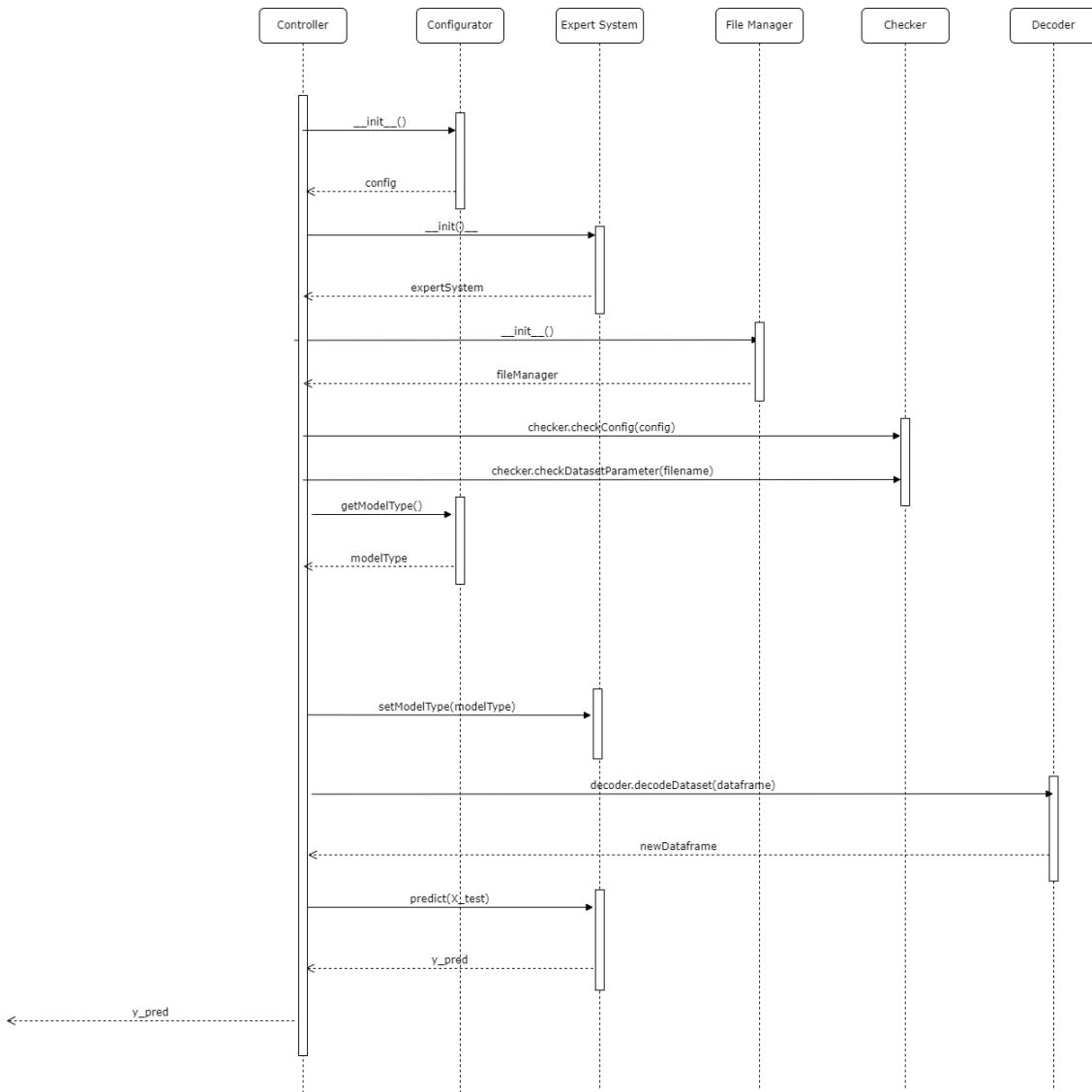


Figura B.10: Diagrama de secuencia del comando para realizar una predicción

