

UNIVERSIDAD POLITÉCNICA DE MADRID

E.T.S. DE INGENIERÍA DE SISTEMAS INFORMÁTICOS

PROYECTO FIN DE GRADO

GRADO EN INGENIERÍA DE COMPUTADORES

PANOT: Plataforma Móvil para la Gestión de la Inteligencia Relacional mediante Captura Asistida de Interacciones

Desarrollado por: Ángel Rodríguez Morán

Dirigido por: Elvira Amador Domínguez

Madrid, 3 de noviembre de 2025



PANOT: Plataforma Móvil para la Gestión de la Inteligencia Relacional mediante Captura Asistida de Interacciones

Desarrollado por: Ángel Rodríguez Morán

Dirigido por: Elvira Amador Domínguez

Proyecto Fin de Grado, 3 de noviembre de 2025

E.T.S. de Ingeniería de Sistemas Informáticos

Campus Sur UPM, Carretera de Valencia (A-3), km. 7

28031, Madrid, España

Si deseas citar este trabajo, la entrada completa en B_BT_EX es la siguiente:

```
@mastersthesis{2025ngelRodrguezMorn,  
  title = {PANOT: Plataforma Móvil para la Gestión de la Inteligencia Relacional  
mediante Captura Asistida de Interacciones},  
  type = {Bachelor's Thesis},  
  author = {},  
  school = {E.T.S. de Ingeniería de Sistemas Informáticos},  
  year = {2025},  
  month = {11},  
}
```

Esta obra está bajo una licencia [Creative Commons «Atribución-NoComercial-CompartirIgual 4.0 Internacional»](https://creativecommons.org/licenses/by-nc-sa/4.0/). Obra derivada de <https://github.com/blazaid/UPM-Report-Template>.



Todo cambio respecto a la obra original es responsabilidad exclusiva del presente autor.

[Cita opcional para el proyecto]

— [Autor de la cita]

Agradecimientos

[Escribir aquí los agradecimientos del proyecto]

Resumen

El presente Proyecto Fin de Grado (PFG) tiene como objetivo diseñar, desarrollar y desplegar PANOT, una plataforma móvil para la gestión de Inteligencia Relacional a través de la captura asistida de interacciones y operando bajo el paradigma de Experiencia Agéntica (AX).

El objetivo de PANOT es facilitar el enriquecimiento progresivo de la cartera de contactos del usuario, mediante la captura de interacciones a través de una interfaz de entrada multimodal (voz y/o texto). Este proceso transforma los contactos, tradicionalmente estáticos, en entidades dinámicas que evolucionan para reflejar con mayor fidelidad el estado contextual de la relación.

[continuar mas adelante después de terminar el proyecto...]

Palabras clave:

Abstract

[Write here the project summary in English]

Keywords:

Índice general

1	Introducción	1
1.1	Motivación	1
1.2	Descripción y Alcance del Proyecto	1
1.3	Objetivos	1
1.4	Estructura de la memoria	1
2	Estado de la Técnica y Fundamentos Tecnológicos	2
2.1	Gestión de la Inteligencia Relacional	2
2.2	Cambio de Paradigma en el Diseño de Producto	5
2.3	Tecnologías para el Desarrollo de Aplicaciones para iOS	7
2.4	Contexto tecnológico	10
3	Desarrollo del Proyecto	11
3.1	Metodología y Entorno de Desarrollo	11
3.2	Especificación de Requisitos de Software	11
3.3	Diseño de la Arquitectura del Sistema	11
3.4	Fases de la Implementación	12
3.5	Despliegue y Lanzamiento de PANOT	12
4	Verificación y Resultados	13

5	Conclusiones y Trabajo Futuro	14
5.1	Conclusiones Generales	14
5.2	Aplicación de Conocimientos Adquiridos en el Grado	14
5.3	Líneas de Trabajo Futuro	14

1.

Introducción

1.1. Motivación

[Escribir aquí la motivación del proyecto]

1.2. Descripción y Alcance del Proyecto

[Escribir aquí la descripción y alcance del proyecto]

1.3. Objetivos

[Escribir aquí los objetivos del proyecto]

1.4. Estructura de la memoria

[Escribir aquí la estructura de la memoria]

2. Estado de la Técnica y Fundamentos Tecnológicos

2.1. Gestión de la Inteligencia Relacional

Los sistemas CRM convencionales, si bien útiles en entornos corporativos para la gestión masiva de clientes, presentan limitaciones significativas cuando se trata de capturar la complejidad inherente a las relaciones humanas. Estos sistemas operan principalmente con información descontextualizada, almacenando datos de contacto de manera estática y registrando interacciones sin considerar su evolución temporal ni el contexto en el que ocurren.

La Inteligencia Relacional presenta un nuevo paradigma evolutivo en la gestión de información personal y profesional, introduciendo el concepto de dinamismo contextual, donde cada relación evoluciona continuamente reflejando cambios en intereses, preferencias y circunstancias vitales. Este enfoque reconoce que las relaciones que tenemos no son entidades fijas, sino procesos complejos que cambian según un contexto temporal y situacional.

2.1.1. Inteligencia Relacional en el Contexto de la Inteligencia Artificial

Para comprender el alcance de la Inteligencia Relacional, es necesario enmarcar el concepto dentro del ecosistema más amplio de la Inteligencia Artificial y analizar cómo se diferencia con los paradigmas tradicionales.

La diferencia fundamental entre la Inteligencia Relacional y los paradigmas tradicionales de IA radica en que, mientras estos últimos operan mediante asociaciones estadísticas entre patrones de entrada y salida —optimizando funciones de pérdida sobre grandes volúmenes de datos descontextualizados—, la Inteligencia Relacional se fundamenta en la construcción y manipulación de representaciones estructurales de

relaciones que permiten la generalización cruzada y la inferencia analógica¹. La Inteligencia Relacional captura la estructura relacional subyacente que puede transferirse entre dominios aparentemente no relacionados, tal como ocurre en el razonamiento humano.

La investigación en Inteligencia Relacional demuestra capacidades que van más allá del aprendizaje estadístico tradicional. Doumas et al. *doumas2022theory* muestran cómo un modelo computacional puede aprender representaciones relacionales estructuradas y realizar generalización de cero disparos² entre dominios completamente diferentes, como la transferencia de conocimiento entre videojuegos. Esta capacidad de generalización permite que el sistema aprenda a reconocer y comprender relaciones entre entidades en contextos completamente diferentes.

Traspassando la analogía de los videojuegos al contexto de PANOT, la Inteligencia Relacional como se menciona en *doumas2022theory* permite que el sistema aprenda a reconocer y comprender patrones relacionales estructurados entre personas, eventos y contextos, más allá de las asociaciones estadísticas superficiales. En lugar de simplemente almacenar datos estáticos de contactos, PANOT puede construir representaciones relacionales dinámicas que capturan la estructura subyacente de las relaciones humanas —como la evolución temporal de intereses comunes, la frecuencia contextual de interacciones, o los cambios en preferencias y circunstancias vitales—.

Por ejemplo, el sistema puede reconocer que ciertos patrones de comunicación efectivos en relaciones profesionales pueden generalizarse a nuevas relaciones profesionales, o que cambios detectados en el contexto de una relación personal pueden aplicarse para comprender dinámicas similares en otras relaciones. Esta generalización relacional es lo que permite que PANOT evolucione continuamente cada contacto, reflejando no solo quién es esa persona en términos estáticos, sino cómo ha evolucionado y continúa evolucionando la relación según el contexto temporal y situacional. Para ilustrar este proceso, consideremos un ejemplo práctico del flujo de procesamiento relacional en PANOT:

Input: El usuario captura una interacción mediante nota de voz: “Acabo de almorzar con María. Está muy emocionada porque ha conseguido un nuevo trabajo como diseñadora en una startup tecnológica. Le interesa especialmente el trabajo remoto y mencionó que está buscando un piso más cerca de su nueva oficina. Hablamos de pro-

¹Transmisión de conocimientos de una situación a otra

²Escenario de aprendizaje automático en el que se entrena un modelo de IA para reconocer y categorizar objetos o conceptos sin haber visto ningún ejemplo de esas categorías o conceptos de antemano

yectos de diseño colaborativo y se mostró muy receptiva a la idea de futuros proyectos juntos.”

Procesamiento: PANOT procesa esta entrada multimodal extrayendo múltiples capas de información relacional estructurada:

- *Evento:* almuerzo social de contexto informal
- *Cambio de estado:* transición profesional — nuevo trabajo como diseñadora
- *Cambio de preferencias:* prioridad hacia trabajo remoto
- *Necesidad emergente:* búsqueda de vivienda
- *Relaciones:* interés común en proyectos de diseño colaborativo, receptividad a futura colaboración
- *Contexto temporal:* estado emocional positivo, momento de transición vital

El sistema construye una representación relacional estructurada que conecta estas entidades (usuario-contacto) mediante relaciones semánticas representadas en formato JSON:

```
{
  "ha-cambiado-preferencia": {
    "preferencia": "trabajo-remoto"
  },
  "interés-común": {
    "usuarios": ["Usuario", "María"],
    "tema": ["diseño", "startup-tecnológica", "diseño-colaborativo"]
  },
  "contexto-temporal-situacional": {
    "evento": "almuerzo-informal",
    "estado": "transición-profesional"
  }
}
```

Output: PANOT actualiza dinámicamente el contacto de María, generando múltiples outputs contextuales:

- *Actualización automática del perfil:* se añade “Diseñadora en startup tecnológica” como situación laboral actual y se marca “Trabajo remoto” como preferencia. Se registra también el cambio de estado como una nueva etapa profesional.

- *Recordatorio contextual*: en la ficha de María se crea un recordatorio automático para preguntar sobre la búsqueda de piso en futuras interacciones.
- *Recomendaciones de conversación*: el sistema sugiere abordar temas de “diseño colaborativo” y “startups tecnológicas” en próximos contactos, reforzando el interés común detectado en el JSON.
- *Inferencia relacional*: mediante patrones previos, el sistema detecta que los cambios laborales suelen ir acompañados de mayor apertura a colaboraciones y recomienda estrategias de seguimiento específicas para situaciones de transición profesional.
- *Seguimiento temporal*: clasifica la interacción dentro de una “fase de transición profesional positiva”, vinculándola en el timeline relacional y ajustando las siguientes recomendaciones conforme evolucione el contexto.

2.2. Cambio de Paradigma en el Diseño de Producto

Un paradigma de diseño de producto constituye un conjunto de principios fundamentales, patrones de interacción y enfoques conceptuales que guían la creación de experiencias de usuario en productos y servicios digitales o analógicos. Representa más que una simple metodología de diseño; es una filosofía que establece cómo los usuarios perciben, interactúan y se relacionan con una aplicación. Fijese que este apartado no está orientado en principios de diseño de la arquitectura del software, sino en principios de diseño de producto que van más allá del desarrollo de la aplicación y están orientados en la experiencia del usuario.

En el contexto de las aplicaciones móviles, la relevancia de los paradigmas de diseño de producto adquiere una dimensión crítica debido a las características inherentes de estos dispositivos: limitaciones de espacio en pantalla, interacciones predominantemente táctiles y expectativas de inmediatez y estímulo por parte de los usuarios. Un paradigma de diseño adecuado no solo determina la usabilidad de una aplicación, sino que establece la base sobre la cual se construyen las expectativas del usuario, su curva de aprendizaje y, fundamentalmente, su conexión emocional con el producto.

La irrupción de la Inteligencia Artificial como tecnología dominante ha transformado radicalmente el panorama del diseño de productos digitales. La democratización de las capacidades de IA —donde funcionalidades que antes requerían desarrollo especializado ahora están disponibles mediante APIs u otros servicios— ha generado un desplazamiento del valor diferencial de los productos: ya no es suficiente ofrecer una

funcionalidad única o una interfaz atractiva, pues estas características pueden replicarse rápidamente. En este nuevo contexto, la diferenciación competitiva se desplaza hacia dimensiones más profundas y fundamentales de la experiencia humana.

2.2.1. Valores Diferenciadores en la Era de la IA

En busca de la diferenciación y lealtad a largo plazo de estos productos, se deben incorporar valores fundamentales más allá de la funcionalidad técnica. Para construir un producto que se diferencie, es esencial y crítico en la era en la vivimos generar valor desde flancos más profundos y fundamentales de la experiencia humana. Como punto de partida, PANOT se ha centrado en los siguientes dos principios:

- *Conexión y resonancia emocional*: El producto debe generar una conexión emocional genuina con los usuarios, comprendiendo su contexto y acompañando la evolución de sus necesidades, para establecer vínculos sostenibles que trasciendan la interacción funcional.
- *Personalización adaptativa y comprensión contextual*: El sistema debe de tener la capacidad de aprender activamente de las interacciones con el usuario, infiriendo patrones, intereses y necesidades implícitas sin requerir configuraciones explícitas, y adaptando la experiencia de manera proactiva y sin fricción, asegurando una experiencia personalizada continua.

Los usuarios no solo buscan que una aplicación funcione bien; buscan que se *adapte* a ellos, que *comprenda* su contexto, que *evolucione* con sus necesidades y que establezca una conexión que trascienda la mera transacción funcional.

2.2.2. Arquitectura de PANOT

Para materializar estos valores, el sistema se ha estructurado en dos componentes arquitectónicos fundamentales que operan de manera complementaria:

- PANOT (cliente): Constituye la capa de contacto entre el usuario y el sistema, proporcionando una interfaz intuitiva y adaptativa que facilita la captura de interacciones, la visualización de sus relaciones, y la presentación de recomendaciones contextuales entre otras características. El diseño de esta aplicación se centra en maximizar la conexión emocional con el usuario y en proporcionar una experiencia orgánica y sin fricción.

- PANOT OS (servidor): Representa la capa de inteligencia o el cerebro del sistema, encargada de procesar, analizar y aprender continuamente de las interacciones capturadas por la aplicación móvil. PANOT OS es el responsable de construir y mantener las representaciones relacionales estructuradas, de inferir patrones y preferencias del usuario, de generar recomendaciones contextuales y de evolucionar dinámicamente el conocimiento sobre las relaciones.

En este marco, PANOT se encarga de crear la conexión emocional y facilitar la interacción inmediata, mientras que PANOT OS proporciona la personalización adaptativa y la comprensión contextual que permiten que esa conexión se profundice con el tiempo. Ambos componentes trabajan de manera sinérgica para materializar los valores diferenciadores de conexión emocional y personalización adaptativa, garantizando que el sistema no solo responda a las necesidades actuales del usuario, sino que evolucione continuamente para acompañar la dinámica natural de sus relaciones.

2.3. Tecnologías para el Desarrollo de Aplicaciones para iOS

2.3.1. Desarrollo Nativo

El desarrollo nativo de aplicaciones para iOS se basa en un ecosistema integrado de lenguajes, herramientas y frameworks diseñados y mantenidos por Apple para garantizar la calidad, seguridad y consistencia de las aplicaciones en sus dispositivos.

Dos lenguajes de programación principales conforman el núcleo del desarrollo nativo: *Swift* y *Objective-C*. *Swift*, introducido en 2014, es el lenguaje moderno y recomendado para nuevos proyectos, ofreciendo tipado estático, gestión automática de memoria mediante referencia contadora (ARC) y una sintaxis expresiva que mejora la seguridad y la productividad del desarrollador. *Objective-C* es el lenguaje tradicional usado por Apple, sigue siendo compatible y se utiliza principalmente en proyectos legados o para integración con frameworks de bajo nivel.

El entorno de desarrollo integrado por excelencia es *Xcode*, el IDE oficial y exclusivo para iOS. *Xcode* proporciona todas las herramientas necesarias para el flujo completo de desarrollo: editor de código, depurador, simulador de dispositivos y utilidades como *Interface Builder* para el diseño visual de interfaces, *Instruments* para análisis de rendi-

miento, y gestión de certificados y perfiles de aprovisionamiento imprescindibles en el proceso de firma y distribución de aplicaciones.

En cuanto a la construcción de la interfaz de usuario, existen dos frameworks principales: *SwiftUI* y *UIKit*. *SwiftUI*, presentado en 2019, representa el enfoque declarativo y moderno donde la interfaz se define desde código Swift utilizando un paradigma funcional y reactivo. Por su parte, *UIKit* es el framework tradicional fundamentado en un enfoque imperativo y el patrón Modelo-Vista-Controlador (MVC), manteniendo hoy en día una presencia relevante tanto en desarrollos existentes como en escenarios que requieren un control más directo del sistema.

Apple incorpora también diversas herramientas adicionales usadas frecuentemente en el desarrollo de aplicaciones para iOS. Por un lado, *CocoaPods* y *Swift Package Manager* (SPM) que permiten la gestión de dependencias externas. Por otro lado, *Core Data* y *CloudKit* que facilitan la persistencia de datos local y sincronización en la nube respectivamente. O, por otro lado, *Combine*, que es un framework introducido junto con *SwiftUI* que permite la gestión de flujos de datos asíncronos.

La combinación de estos lenguajes, herramientas y frameworks permite la creación de aplicaciones iOS de alto rendimiento y experiencia de usuario optimizada, garantizando compatibilidad, seguridad y aprovechamiento completo de las capacidades nativas del sistema operativo y el hardware de Apple.

2.3.2. Frameworks Multiplataforma

Además del desarrollo nativo, existen varios frameworks multiplataforma que permiten desarrollar aplicaciones para iOS junto con otras plataformas (principalmente Android) desde una base de código compartida.

- *React Native*, desarrollado por Meta, permite crear aplicaciones móviles utilizando JavaScript y React. El framework utiliza un puente nativo que comunica el código JavaScript con componentes nativos de cada plataforma, permitiendo acceso a APIs nativas mientras se comparte la mayor parte de la lógica de negocio.
- *Expo*, construido sobre React Native, proporciona un conjunto de herramientas y servicios que simplifican el desarrollo, incluyendo un runtime unificado, APIs listas para usar y un sistema de compilación en la nube. Expo reduce significativamente la complejidad de configuración del proyecto y facilita el despliegue, aunque con algunas limitaciones en el acceso a funcionalidades nativas avanzadas.

- *Flutter*, desarrollado por Google, utiliza el lenguaje *Dart* y un motor de renderizado propio que compila a código nativo. Flutter construye la interfaz de usuario desde cero en cada plataforma, evitando la necesidad de componentes nativos del sistema operativo y proporcionando mayor consistencia visual entre plataformas.

Otras alternativas multiplataforma incluyen *Xamarin* (ahora *.NET MAUI*) que utiliza C# y .NET, *Ionic* que combina tecnologías web (HTML, CSS, JavaScript) con capacidades nativas mediante *Cordova*, y *Unity* para aplicaciones que requieren capacidades gráficas avanzadas, esencialmente, videojuegos.

La elección entre desarrollo nativo y multiplataforma depende de factores como requisitos de rendimiento, necesidad de acceso a funcionalidades nativas avanzadas, tiempo de desarrollo, mantenimiento a largo plazo o recursos del equipo.

2.3.3. Proceso de Desarrollo y Distribución

El ciclo de vida de una aplicación iOS desde el desarrollo hasta su distribución sigue un proceso estructurado:

1. *Desarrollo*: Durante esta fase, el código se compila y ejecuta en simuladores iOS o dispositivos físicos mediante perfiles de desarrollo.
2. *Pruebas internas*: Para pruebas internas, las aplicaciones, una vez compiladas, es posible su distribución mediante *TestFlight*, plataforma que permite a desarrolladores invitar hasta 10.000 beta testers externos sin necesidad de certificados adicionales.
3. *Archive*: El proceso de *Archive* genera un paquete de distribución de la aplicación (*.ipa*) optimizado y firmado con los certificados de distribución ³. Esta versión archivada puede subirse a *App Store Connect*, portal web de Apple para gestión de aplicaciones, donde se configura información de la aplicación, capturas de pantalla, descripciones y metadatos requeridos para la publicación.
4. *App Review*: Una vez hecha la petición de subida, este proceso de revisión de Apple verifica que el producto cumple con las directrices de la App Store, incluyendo segu-

³Los certificados de distribución son credenciales digitales emitidas por Apple que permiten identificar y autenticar al desarrollador, garantizando que la aplicación proviene de una fuente confiable y verificada. Son esenciales para firmar digitalmente las aplicaciones antes de su distribución en la App Store.

ridad, privacidad, calidad técnica y contenido. Una vez aprobada, la aplicación está disponible para distribución pública o privada según la configuración establecida.

5. *Distribución*: La distribución puede realizarse mediante tres canales principales:

- App Store: Para usuarios finales a través de la tienda oficial de Apple.
- Distribución empresarial (*Enterprise*): Permite a las organizaciones internas distribuir aplicaciones privadas a sus empleados, fuera de la App Store.
- Distribución ad-hoc: Permite instalar la aplicación en un número limitado de dispositivos específicos, identificados mediante perfiles de aprovisionamiento ⁴.

2.4. Contexto tecnológico

[Escribir aquí el análisis comparativo]

⁴Los perfiles de aprovisionamiento son archivos que vinculan a un desarrollador y su aplicación con una cuenta de desarrollador, dispositivos y servicios autorizados

3. Desarrollo del Proyecto

3.1. Metodología y Entorno de Desarrollo

[Escribir aquí la metodología y entorno de desarrollo del proyecto]

3.1.1. Gestión Ágil del Proyecto con GitHub Projects (Git Flow + Kanban)

[Escribir aquí la gestión ágil del proyecto con GitHub Projects (Git Flow + Kanban)]

3.1.2. Estrategia de Ramificación y Control de Versiones

[Escribir aquí la estrategia de ramificación y control de versiones]

3.1.3. Estructura de Repositorios en la Organización

[Escribir aquí la estructura de repositorios en la organización]

3.2. Especificación de Requisitos de Software

[Escribir aquí la especificación de requisitos de software]

3.3. Diseño de la Arquitectura del Sistema

[Escribir aquí el diseño de la arquitectura del sistema]

3.4. Fases de la Implementación

[Escribir aquí las fases de la implementación]

3.5. Despliegue y Lanzamiento de PANOT

[Escribir aquí el despliegue y lanzamiento de PANOT]

4. Verificación y Resultados

[Escribir aquí los resultados y evaluación del proyecto]

5. Conclusiones y Trabajo Futuro

5.1. Conclusiones Generales

[Escribir aquí las conclusiones generales]

5.2. Aplicación de Conocimientos Adquiridos en el Grado

[Escribir aquí la aplicación de conocimientos adquiridos en el grado]

5.3. Líneas de Trabajo Futuro

[Escribir aquí las líneas de trabajo futuro]

Índice de términos

