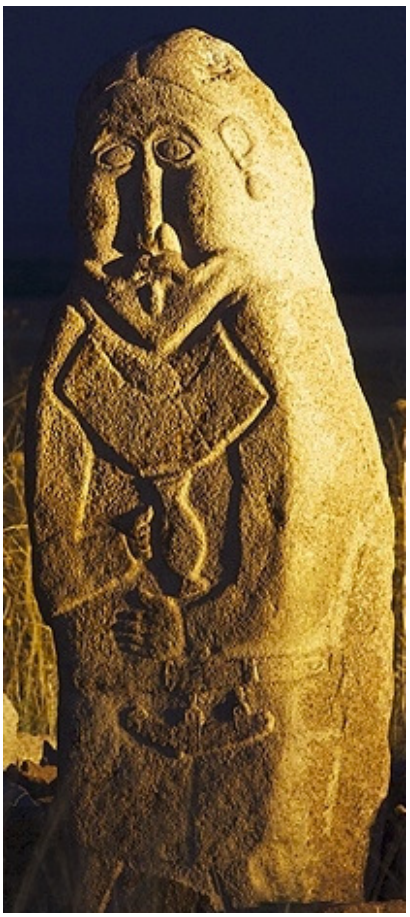
UOC
www.uoc.edu

PINTORESCO

EVALUACION DE LOS PROCESOS DE APRENDIZAJE CONCEPTUAL



AREA: DESARROLLO DE APLICACIONES EN DISPOSITIVOS MOVILES

AUTOR: JAVIER SAINZ
ENERO, 2014

CONSULTOR: ALBERT GRAU PERISÉ

PINTORESCO: EVALUACION DE LOS PROCESOS DE APRENDIZAJE CATEGORIAL

Resumen

La memoria recoge los hitos del desarrollo de **Pintoresco**, una aplicación orientada a la evaluación de los procesos de aprendizaje conceptual que se examinan a partir de su uso por usuario ordinario. La aplicación permite obtener datos del proceso de aprendizaje conceptual en un contexto en que la estructura interna de las clases en que se produce una partición no depende de las propiedades específicas de los patrones de estímulo que son distintos para cada usuario sino de la forma lógica de la propia partición. **Pintoresco** permite obtener datos del proceso de aprendizaje y conservar de forma permanente los datos adquiridos para su tratamiento posterior.

PICTURESQUE: AN ASSESSMENT OF CONCEPTUAL LEARNING PROCESSES

Summary

This final report presents the milestones for the development of **Picturesque**, an application aimed at analyzing conceptual learning processes as can be assessed from an ordinary user's behavior when s/he uses the application with that purpose. **Picturesque** allows us to collect data from this learning process in a context which the internal structure of classes is manipulated in since stimulus patterns vary across users treated by the application as experimental subjects. **Picturesque** allows us to examine the way people distribute a set of exemplars in two classes according to a partition defined by the logical form of the underlying structure of the displayed patterns. **Picturesque** provide data about this learning process and keep them permanently for subsequent analyses thereafter.

Estructura de la memoria

El proyecto que se presenta en esta memoria tiene por objeto el desarrollo de una aplicación orientada a la evaluación de los procesos de aprendizaje conceptual, los procesos que permiten hacer una partición de un conjunto de patrones de estímulo ejemplares de acuerdo con una clasificación binaria. La memoria se estructura de acuerdo con este objetivo en distintos capítulos.

En el primer capítulo se presenta la motivación general del proyecto, las razones que se encuentran detrás de la decisión de desarrollar **Pintoresco**, el proceso de generación y concepción de la idea, la justificación del enfoque empleado y su expresión en el programa que se desarrolla.

En el segundo capítulo, muy brevemente, se menciona la metodología del proyecto, específicamente por lo que se refiere a qué criterios se aplican al desarrollo de la aplicación en relación con la estructura lógica interna de una partición en clases a partir de diferentes reglas de clasificación.

En el tercer capítulo se desarrolla la arquitectura e implementación del proyecto, el proceso que se sigue para sentar las bases del proyecto en el plano teórico, la estructura de la aplicación con las decisiones de diseño adoptadas, la implementación y las pruebas que deben desarrollarse.

En el cuarto capítulo se describen las funcionalidades de la aplicación describiendo los resultados de una aplicación piloto, los perfiles de usuario, las condiciones de aplicabilidad de la aplicación y las características que aseguran su usabilidad. El capítulo concluye refiriendo el estado actual del desarrollo de aplicaciones de este carácter y las conclusiones de este estudio preliminar.

En el quinto capítulo se presentan las interficies de usuario de la aplicación y el prototipo desarrollado. Se describen las interficies de usuario y la secuencia de ejecución de la aplicación.

En el capítulo sexto se incluyen referencias explícitas al código fuente que expresan las funcioanalidades que permiten la ejecución de la aplicación a partir del Modelo de Controlador de Vistas y los protocolos de diseño habitualmente empleados en aplicaciones móviles bajo iOS. Se presta atención a la gestión de la persistencia y las clases que se asocian con la persistencia de datos, y se presenta la implementación de las clases o métodos de clase que constituyen la arquitectura interna de la aplicación desde la perspectiva del usuario.

En el capítulo séptimo se evalúa la aplicación, la verificación de su funcionamiento, sus deficiencias y/o ineficiencias, y su impacto en la mejora de la aplicación en sucesivas versiones. Se mencionan a este respecto los problemas detectados y las características destacables. Se concluye con el estudio de las perspectivas de la aplicación, la utilidad de desarrollar la aplicación incorporando tecnologías más elaboradas y complejas a partir de las funcionalidades implementadas.

El último capítulo de conclusiones reevalúa los objetivos que se han logrado en el desarrollo, los retos que se presentan en la evolución de la herramienta y el valor objetivo y subjectivo de la aplicación desarrollada.

Palabras clave:

iPhone, iPad, Objective-C, iOS, Cocoa-Toucher, X-Code, Interface Builder, Model View Controller, Core Graphics, Data Bases, Core Data, SQLite, Human Interface Guidelines, User Centered Design

Indice de contenidos del proyecto

Indice de figuras

1. Motivación general del proyecto

Las imágenes de portada permiten ilustrar las ideas que estuvieron en el origen del proyecto. Estas imágenes presentan dos esculturas que, aunque próximas en el tiempo, pertenecen a distintos periodos de la cultura kazaja, del sur de Kazakhstan, de la región que toma el nombre de Zhambyl, en honor del poeta kazajo Zhambyl Dzhabayev (1846-1945). En la escultura menos elaborada, y también la más antigua, ambas de un periodo preislámico, pueden reconocerse rasgos fundamentales de un rostro humano, extraordinariamente simplificado. En la escultura más elaborada se presenta a un sacerdote oferente en el que se reconocen rasgos muy elaborados del rostro, e incluso de la actitud; de su brazo izquierdo cuelga un amuleto del que a su vez cuelga una barquichuela en la que es posible reconocer hasta tres figuras humanas minúsculas. La mano derecha sostiene una vasija que apenas oculta un ancho torque probablemente fijado a la prenda que entalla la figura. El reconocimiento de estos objetos no sería posible de no identificarse ciertas formas geométricas canónicas, algunas más complejas que otras, por cuya composición se infiere un objeto en particular. Las cuencas de los ojos presentan una forma almendrada, el torque es un rectángulo deformado por las curvas que identifican su contorno, la vasija, la composición de cilindros que se estrechan y elongan dibujando un recipiente, tiene un asa; la forma que se ha identificado como una barquichuela apenas se distingue, salvo por su tamaño y curvatura, de lo que se identifica como el ancho torque que parece integrado en la vestimenta. El bigote, la barba característica, los pendientes que cuelgan de la única oreja visible también se conforman de acuerdo con una serie de formas canónicas, sometidas a variaciones de posición, tamaño, y forma geométrica.

Desde un punto de vista topológico, no todas las formas empleadas en la composición son idénticas: la conectividad, continuidad y bordes de algunas de las formas geométricas que permiten identificar los objetos representados varían; p. ej. el recipiente que sostiene en su mano derecha el sacerdote tiene un asa que identifica esta forma geométrica con un toro, un objeto topológico de una dimensión, una superficie de revolución que se genera por el producto cartesiano de dos circunferencias. El objeto en cuestión es una transformación de esta forma topológica aparentemente elemental. Los otros objetos son todavía topológicamente más simples, son transformaciones de formas primitivas elementales, que a veces se inscriben en otras. En cualquiera de estos casos, la escultura resulta de la composición de formas geométricas que definen ciertas invariantes cognitivas en términos de las cuales puede identificarse cualquier objeto de nuestro entorno. La distribución de estas formas contribuye a los procesos de reconocimiento de cada

objeto en cuestión. De la composición de estas formas geométricas derivan restricciones de alto nivel, es decir, ciertas propiedades no accidentales que tienen un papel decisivo en la identificación de un objeto en cuestión. De esta composición derivan ciertas propiedades configurales de modo que una alteración de la distribución de las mismas formas no permitirían evocar ningún tipo de representación. Es fácil apercibirse que éste es particularmente el caso del reconocimiento del rostro humano: una distribución aleatoria en un contorno local cerrado de los elementos que identifican los ojos, la nariz, la boca, el bigote, la barba y la oreja no evocarían ningún rostro humano, menos aún la imagen de un sacerdote oferente que en una representación sacrificial ofrenda a la divinidad una libación para propiciar el viaje de los humanos que se presentan como figuras en miniatura de una barquichuela que cuelga de un amuleto cilíndrico bajo su brazo izquierdo.

La elemental apariencia de estos motivos escultóricos no puede esconder la complejidad que subyace a los procesos de reconocimiento de objetos, y de cómo se distribuyen estos objetos en sendas categorías o conceptos; un problema fundamental computacionalmente insoluble es precisamente de cuántos modos puede distribuirse un conjunto finito de objetos en un conjunto finito de conceptos o categorías. A fortiori deben haber ciertas restricciones que a priori determinen que particiones del conjunto pueden ser relevantes en términos informacionales, qué propiedades o rasgos y en qué combinaciones permitirían a los humanos reducir incertidumbre y detectar regularidades que permiten a su vez anticipar ciertos estados de la naturaleza.

Es tal nuestra familiaridad con los objetos de nuestra experiencia que apenas nos es posible reconocer de forma espontánea qué propiedades debería tener un objeto para ser un objeto imposible. Parece más fácil encontrar objetos biestables, objetos que pueden eventualmente interpretarse de, al menos, dos formas distintas; el más famoso y mejor conocido ejemplo es el cubo de Necker. Sin embargo, existe un número indefinido de objetos cuya interpretación puede cambiar con un cambio de orientación, una propiedad configural que afecta a la distribución espacial de aquellas formas geométricas que Biederman (1987) denomina geones, invariantes de forma por cuya transformación se obtienen todas las formas visuales. El tamaño, orientación y distribución espacial de estas formas geométricas son propiedades que afectan al valor funcional que adquieren. Aunque pudiera parecer sorprendente, éste es el caso cuando se emplean las letras b-q, p-d. Existen, también, objetos para los que es imposible encontrar una interpretación unitaria: se reconoce a Escher por haber transformado nuestra percepción espacial al crear complejas escenas de coherencia imposible. El grado de aceptabilidad de un objeto como miembro de una cierta categoría es expresión de alteraciones cognitivas severas. La facilidad con la que el sujeto cambia de criterios en el reconocimiento de una serie de patrones cuando trata de identificar una cierta regularidad se asocia con flexibilidad cognitiva, y la flexibilidad cognitiva es un indicio inequívoco

de que están preservados los procesos que afectan a la evaluación de la evidencia y a la adopción de decisiones racionales; su ausencia un indicio de alteraciones funcionales. No en vano se ha identificado la ausencia de conexiones de vías largas y la ausencia de flexibilidad cognitiva con el autismo, ausencias que se identifican con ecolalia, perseverancia observacional y rigidez de la acción.

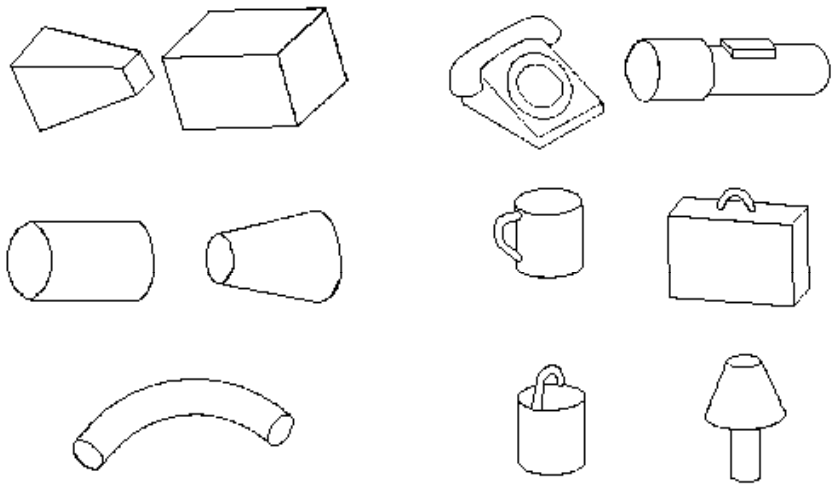


Figura 1: Geones elementales y composición de objetos según la Teoría de Reconocimiento por Componentes de Biederman (1987)

Los objetos de nuestra experiencia se reconocen como tales en tanto en cuanto resultan de la composición de formas geométricas elementales. La Figura 1 presenta una muestra de geones y algunos de los objetos que pueden reconocerse por la composición de aquellos. Es tal la diversidad de objetos que pueden construirse que un conjunto finito de geones pueden dar lugar a un número infinito de objetos sin más que combinar tales formas geométricas elementales bajo ciertas relaciones geométricas. Estas composiciones resultan habitualmente muy redundantes en términos informacionales. La imagen de un objeto contiene información redundante; un número indefinido de rasgos o propiedades permiten identificar exactamente el mismo objeto. Un dibujo, en cambio, es una versión simplificada del objeto que representa, y contiene, en este sentido, menos información que la que puede extraerse de un patrón visual mínimamente complejo. Un dibujo contiene tanta información cómo se requiere para distinguir entre representaciones de distintos objetos. Frente a la complejidad aparente de una imagen, una palabra, por contra, es una combinación de un conjunto finito de letras/fonemas y su identidad proviene de cómo se distribuyen las letras que la conforman. Que las palabras resulten de la composición de un conjunto finito de símbolos no es ni más ni menos cierto que los objetos resulten de la composición de un conjunto finito de formas geométricas elementales, al menos por lo que se refiere a la forma en que tales objetos pueden reconocerse como tales. Es decir, un objeto no es ni más complejo ni más simple de lo que lo es una palabra excepto por el hecho de que sea más fácil determinar la economía o la redundancia de una palabra en

términos de las relaciones que pueden contraer los símbolos que se combinan. ¿Qué "letras" o "símbolos" conforman un patrón visual y bajo qué relacones pueden combinarse?. Es fácil darse cuenta de que en PATA/TAPA están implicados los mismos fonemas y las mismas letras, pero es más difícil reconocer que GENIO es una recombinación de NIEGO, que eventualmente dá lugar a cambios fonológicos y/o fonéticos. En cualquier caso, estas palabras resultan ser, por su parte, una combinación de los mismos símbolos. ¿Qué símbolos contribuyen a conformar la identidad de un objeto?. El modelo de Reconocimiento por Componentes de Biederman (1987) es un intento de dar respuesta a esta pregunta.

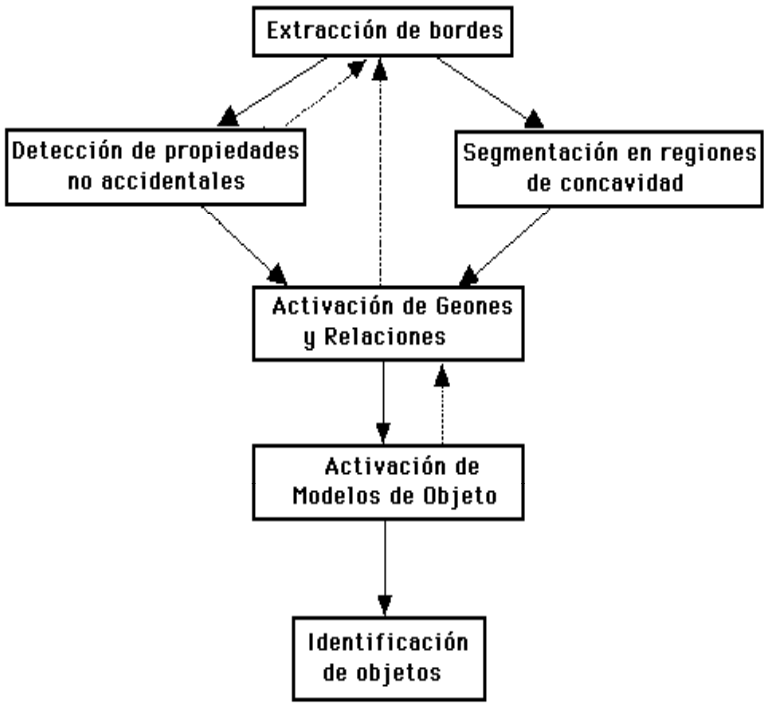


Figura 2: El procesamiento visual de alto nivel según la Teoría de Reconocimiento por Componentes de Biederman (1987)

Max Wertheimer, en 1923, se preguntaba: ¿cómo es posible que las personas perciban un mundo organizado en el espacio mas que una yuxtaposición incoherente de distintos colores recayendo sobre los receptores individuales de la retina? Esta cuestión más que ninguna otra estuvo en el origen del movimiento de la Gestalt y le condujo a sus mas importantes y reconocidas contribuciones. Wertheimer intentó responder a esta pregunta analizando los factores de estímulo que afectan al agrupamiento perceptivo; los principios guestálticos de proximidad, de semejanza, de continuidad y cerramiento permiten, sin embargo, aportar buenos ejemplos del modo en que el conocimiento previo puede estar jugando un papel importante en la percepción, por ejemplo cuando el principio de cerramiento prevalece sobre el de continuidad. Corrigiendo a Marr (1982)

Biederman ha señalado y aportado evidencia de que no es posible un proceso por completo ascendente de la extracción de bordes, una hipótesis que se adoptó sobre la base de que la familiaridad con el objeto no tiene ningún efecto en este proceso. De acuerdo con la Teoría de Reconocimiento por Componentes (RPC) la forma integrada de un objeto realimenta el proceso de extracción de bordes. La Figura 2 presenta un diagrama de procesos en el reocnocimiento de objetos. A partir de una analogía que traza entre la percepción de objeto y la percepción del habla, Biederman (1983) aporta evidencia de la descomposición de los objetos en componentes mas elementales o primitivos que denomina geones. La mayor parte de los objetos comprenderían de dos a siete geones. La organización de estos geones en objetos se justificaría por las relaciones que estos adquieren y por la experiencia previa, es decir, por el conocimiento previamente adquirido a través del aprendizaje. El reconocimiento del objeto resultaría de la identificación de estos geones predefinidos y de sus relaciones de un modo tal que los resultados se puedan corresponder con un modelo de objeto en la memoria semántica. Estos geones pueden distinguirse entre sí por las propiedades no accidentales de sus respectivas descripciones estructurales, es decir, por aquellas propiedades que resultan de su composición que determinan la emergencia de ciertas invariantes.

Biederman (1987) señala que es posible observar efectos del conocimiento previo en procesos responsables de la segmentación, definición de componentes y comparación de escenas y objetos, especialmente cuando como estímulos se emplean imágenes perceptivas degradadas (c.f. Biederman, 1981; Biederman, Mezzanotte y Rabinowitz, 1982). La multiestabilidad perceptiva de representaciones perceptivas ambiguas, los cambios de interpretación de objetos idénticos en orientaciones distintas (cf. Hinton, 1981) o el hecho de que transformaciones ópticas continuas no produzcan descripciones perceptivas únicas (Epstein y Park, 1986) se explican porque en el procesamiento perceptivo, la interpretación de una descripción como un objeto depende de la satisfacción de un conjunto múltiple de restricciones. Este es el fundamento teórico de un sistema de reconocimiento de patrones que se corresponde con el tipo de sistema que conforma la mente humana.

La motivación teórica de este proyecto puede resultar ahora clara. Si los geones pueden tratarse como las letras del reconocimiento de objetos, es razonable suponer que existen regularidades detectables que afectan a la comisión de errores de migración en el reconocimiento de un objeto. Cuando las palabras BATO-RATA se presentan visualmente por un tiempo breve y se le pide a los sujetos que informen de una palabra, a menudo el sujeto ofrece como respuesta la palabra BATA ó RATO. Dado qué este tipo de errores de respuesta se dán menos a menudo con la presentación de BATO-RISA parece que una letra ha migrado de una palabra a la otra en la presentación BATO-RATA como para que las respuestas sean BATA o RATO. La Figura 3

ilustra el mismo caso para dibujos de objetos; las etiquetas verbales que replican el fenómeno se emplean para identificar cada uno de los objetos que permiten replicar el fenómeno que se observa con palabras.

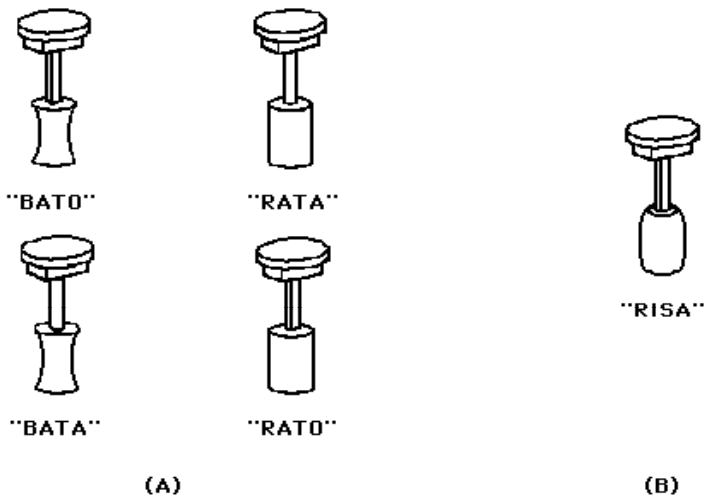


Figura 3: Patrones de objeto codificados bajo un patrón verbal análogo

Estas regularidades de nivel de símbolo -letras y geones- y de nivel de conocimiento -regularidades detectables en su composición- son análogas, respectivamente, a los efectos de regularidad léxica y estatuto semántico del patrón en el reconocimiento de palabras. ¿Que objetos pueden ser tan semejantes como las palabras BATO y RATA en un contexto en el que también existen BATA y RATO? La respuesta dista de ser clara a menos que identifiquemos los objetos en los términos propuestos por Biederman (1983, 1987).

Siguiendo con la analogía de Biederman (1987), el efecto de superioridad de objeto justificaría que se obtuvieran más errores de migración entre diferentes presentaciones del mismo objeto que entre presentaciones de objetos diferentes en un contexto en el que se controlara cualquier otra variable. En este proyecto se trata de examinar la habilidad de un sujeto para aprender a clasificar una serie de objetos que responden a distintas reglas de clasificación. Se trata, pues, de diseñar un instrumento de evaluación de la habilidad de un sujeto para aprender reglas lógicas complejas que responden a las propiedades de los patrones de estímulo que se presentan. El interés del proyecto es diseñar una tarea experimental en una aplicación móvil que servirá luego para identificar qué tipos de errores de migración comete el sujeto cuando deba reconocer un objeto entre un conjunto de distractores. La evaluación de estos errores es una extensión de la aplicación que aquí se presenta y por tanto no se desarrolla. Estos errores de migración mostrarían el papel que el conocimiento previo tiene en el reconocimiento de un objeto. En este proyecto se propone una tarea de clasificación en la que el sujeto debe aprender a clasificar una serie de objetos en una clasificación binaria, cuando se neutraliza el papel de los patrones de estímulo en la clasificación. El

sujeto aprende a clasificar tales objetos según responden a la regla que aprende, y no según sus características.

En efecto, la tarea de clasificación emplea el mismo tipo de objetos, excepto por el tipo de regla que permite hacer una clasificación por completo correcta. Así, por ejemplo, puede aparentemente ser más fácil identificar una clasificación unidimensional en la que todos los ejemplares miembros de un concepto pertenecen a una categoría atendiendo a un único rasgo, que identificar una clasificación multidimensional en la que los miembros de un concepto implica reconocer más de un rasgo. Sin embargo, cuanto más "abstracta" es la representación mayor generalización se produce, menor coste tiene su representación, y más errores pueden cometerse; cuanto menos "abstracta" es la representación menor generalización se produce, mayor coste tiene la representación del conjunto y menos errores pueden cometerse.

Desde el punto de vista de su implementación, la tarea es extraordinariamente simple; se trata de una tarea de aprendizaje de clasificación, en la que se le presentan al sujeto sucesivamente patrones de estímulo extraidos de una serie de patrones de estímulo que satisfacen la misma regla. El sujeto debe decidir si el patrón en cuestión es de tipo A, o de tipo B. Se trata de una clasificación binaria, donde una categoría se asigna por defecto de aquella que permite identificar la categoría alternativa. La idea es testar cuatro tipos distintos de reglas de clasificación, una regla de clasificación unidimensional -que denominamos, regla de rasgo-, una regla de clasificación multidimensional en la que el conjunto de formas geométricas que permite la clasificación puede reducirse a una relación abstracta única bajo una única etiqueta -que denominamos, regla idéntica-, una regla multidimensional en la que el conjunto de formas geométricas que permite la clasificación no puede etiquetarse bajo una relación más abstracta -que denominamos, regla desigual-, y una regla de clasificación que fuerza al sujeto a memorizar todos los ejemplares miembros de una cierta categoría. Cada tipo de regla se asocia con una serie de patrones de entre los que la aplicación escoge al azar la serie a presentar. Como en cualquier caso, la serie que se presenta se escoge aleatoriamente de un conjunto de patrones que satisface la misma regla, y el conjunto de patrones de estímulo es técnicamente el mismo, la clasificación que realiza el sujeto sólo puede relacionarse con la forma o estructura lógica del conjunto de clasificación, y no propiamente hablando con los patrones específicos que se presentan. Desde una perspectiva racional, el sistema cognitivo humano debe tratar de distinguir entre dos entidades si representan diferentes consecuencias para la conducta.

En suma, en este proyecto se examinan y evalúan algunas de las restricciones que operan en el aprendizaje conceptual, es decir, cómo un sujeto humano clasifica, reconoce y expresa la estructura interna de una clase, cómo forma criterios que determinan la partición de una serie de objetos en clases. El problema de la partición de un conjunto de objetos en clases representa un tipo

de problema de complejidad exponencial. El desarrollo de una aplicación de este carácter tiene por objeto identificar y evaluar los procesos que subyacen a ese proceso de aprendizaje de clasificación.

Los objetos deben diferenciarse en clases mutuamente diferentes si se asocian con su aparición eventos distintos (Anderson, 1990). La proposición inversa de esta proposición es que dos objetos deben ser diferentes si se asocian con distintas clases. En otros términos, el modo en que los objetos se agrupan afecta a su discriminación. Si la semejanza determina qué categorías pueden formarse, las categorías disponibles deben afectar a la discriminación de los objetos. En términos operacionales, en un contexto donde se controla la producción de errores, más errores deben obtenerse entre miembros del mismo concepto que entre miembros de conceptos distintos. El efecto debe ser, sin embargo, distinto dependiendo de la partición del conjunto de objetos posibles cuando su clasificación depende únicamente de la estructura lógica que identifica las categorías en cuestión. Esta es la lógica que se encuentra detrás de nuestra propuesta en el plano teórico: una herramienta que permite evaluar el aprendizaje conceptual de un sujeto cuando aprende a discriminar una serie de objetos por su relación con las clases en que se agrupan.

1.1. Justificación y oportunidad del proyecto

Aunque el objetivo primario de este proyecto es satisfacer los requisitos del desarrollo de un Trabajo Final de Carrera en Ingeniería Informática de Sistemas, su motivación y proyección no se agota en este propósito. El área temática en que se inscribe es el Desarrollo de Aplicaciones Móviles, y su objeto es el desarrollo de una aplicación en una tableta gráfica, el IPad de Apple con Sistema Operativo iOS. El trabajo tiene por objeto el desarrollo de una aplicación móvil que permite al usuario aprender a clasificar una serie de objetos artificiales de acuerdo con un tipo de regla de clasificación que requiere del sujeto que cree, invente o descubra una hipótesis que finalmente le permita clasificar con éxito una serie de ejemplares de los que sólo conoce a priori que se distribuyen en dos clases. La aplicación registra todos los eventos conductuales que tienen lugar, registra el tiempo de exposición del patrón, la categoría que el sujeto le asigna, y el número de ensayos que le requiere aprender la clasificación. La aplicación genera un documento que luego puede analizarse, al objeto de identificar qué tipo de regla resulta eventualmente más eficiente en términos de eficiencia y velocidad de aprendizaje. En una extensión de esta aplicación será posible también determinar cómo este entrenamiento previo afecta a su habilidad para discriminar el objeto que clasifica entre un conjunto de distractores. Por hipótesis, el tipo de clasificación que aprende afecta a su habilidad para reconocer un cierto patrón en cuestión.

El potencial de esta aplicación es sencillamente enorme; su proyección excede las condiciones de tiempo que usualmente se asignan a un Trabajo Final de Carrera. Es propósito del autor extender esta aplicación para que constituya una herramienta de evaluación de aprendizaje conceptual con relevancia diagnóstica. La aplicación que se presenta permite identificar qué regla de clasificación puede ser más eficiente en términos de tiempo de aprendizaje, y otros parámetros conductuales. La aplicación que se presenta es la de una aplicación móvil que permite simultáneamente evaluar la competencia de un usuario en una tarea de aprendizaje de clasificación, e identificar las bases lógicas, el tipo de restricciones que operan sobre un sujeto para aprender con mayor o menor facilidad un tipo de estructura o forma lógica, cuando la serie de estímulos es formalmente idéntica para cualquier tipo de regla. De este modo, la evaluación se realiza sobre la estructura lógica de la clasificación que resuelve una partición binaria del conjunto, y no sobre una serie específica de patrones de estímulo.

A pesar de la extraordinaria difusión de las tecnologías móviles en numerosos ámbitos de la vida cotidiana, e incluso del enorme número de aplicaciones que apelan a habilidades humanas básicas, no se ha producido una implantación paralela de tecnologías móviles orientadas a la evaluación conductual, en particular, de aquellas que se orientan a la evaluación de aquellas habilidades que se suponen en un usuario ordinario. Esta aplicación es un paso en esa dirección al emplear fenómenos bien conocidos en los procesos de reconocimiento de objetos, a saber, aquellos procesos que operan en la formación de un concepto, y muy específicamente en el caso en que esos procesos afectan a una serie idéntica de ejemplares cuya clasificación sólo es posible por relación a la estructura lógica del propio concepto. Parece, pues, razonable trabajar en este nicho de mercado aún cuando no haya recibido todavía la atención suficiente, cuando precisamente viene produciéndose la implantación de tecnologías móviles orientadas a la adquisición de destrezas y conocimientos en el ambito educativo.

1.2. Objetivos generales y específicos

Con la realización de este proyecto se han perseguido los objetivos siguientes:

1. Implementar una aplicación que conforma un experimento de aprendizaje conceptual.
2. Obtener medidas dependientes de latencia y error del proceso de aprendizaje conceptual y tasa de aprendizaje dependiendo de la estructura interna de las clases.
3. Identificar restricciones que operan en el aprendizaje conceptual según la partición de una serie de objetos en clases se realiza en base a la estructura lógica del concepto y no a las propiedades

específicas de los estímulos. La estructura lógica de un concepto se encuentra representada por las propiedades no accidentales que contraen los geones de una serie de objetos artificiales.

4. Identificar fenómenos experimentales en el reconocimiento de objetos, en particular los que afectan a la dificultad relativa de clasificar una serie de patrones de estímulo según reglas de clasificación de diferente complejidad.

5. Establecer las bases experimentales para evaluar el aprendizaje conceptual en una tarea de reconocimiento y discriminación visual de objetos.

Basicamente el objetivo consiste en establecer una relación entre la descripción perceptiva de un objeto en cuanto se reconocen por la clase a la que pertenece, y el tipo de representación que el sujeto construye para interpretar este objeto, en tanto una representación es siempre una clase. La aplicación permitirá responder a la pregunta: ¿Qué correspondencia existe entre la representación de una clase y el objeto que se reconoce en virtud de cómo se clasifica?.

1.3. Descripción de la aplicación

La aplicación que se presenta imita el desarrollo de un experimento clásico de aprendizaje conceptual en el que se le presentan al sujeto una serie de patrones visuales que representan objetos de diferentes características y se le pide que descubra la regla que permite su clasificación en dos clases de objetos. Las instrucciones presentan la tarea informando al sujeto que va a ver dibujos de dos tipos de objetos muy útiles, dos tipos de encendedores por émbolo de presión, a menudo denominados encendedores de pistón, que se emplean para hacer fuego. Este tipo de encendedores por émbolo de presión existen, en efecto, aunque no son bien conocidos por el público en general, y son instrumentos antiguos y clásicos en la cultura japonesa de hace siglos y otras culturas del oceáno pacífico. Los dibujos que se presentan, en cuestión, son variaciones de este tipo de objetos. El hecho de que la tarea de aprendizaje haga uso de este tipo de encendedores es el motivo de portada de la aplicación: los quemadores de un globo calientan el aire bajo el globo que permiten su ascensión. La aplicación presenta simplemente el entorno de trabajo, una vez que ha recogido toda la información del usuario. El entorno de trabajo es una vista en la que el sujeto escoge el patrón de estímulo que quiere clasificar en una de las dos categorías propuesta hasta completar la tarea. Debe clasificar tantos ejemplares como número de ejemplares existen en la serie. La tarea concluye cuando el sujeto ha clasificado todos y cada uno de los patrones de estímulo presentados y supera un cierto nivel de eficiencia. La aplicación registra todos los eventos conductuales que tienen lugar, registra el tiempo de exposición del patrón, la categoría que el sujeto le asigna, y el número de ensayos que le requiere al sujeto aprender la clasificación. La aplicación guarda los datos obtenidos de forma permanente y

genera un documento de texto que puede tratarse en un análisis posterior. Los datos obtenidos permiten identificar qué tipo de regla resulta eventualmente más fácil de adquirir en términos de eficiencia y velocidad de aprendizaje, y secundariamente permitiría en una extensión de la aplicación examinar el impacto del aprendizaje previo en el reconocimiento de un objeto en condiciones de búsqueda visual.

1.4. Elección del nombre

Pintoresco es el término que dá nombre a la aplicación que se presenta. Este término responde a que evoca el concepto de una representación "pictórica", un tipo de dibujo de singulares características que soporta o al que viene a trasladarse una representación mental. Esta es la idea que es origen del proyecto, cómo los dibujos que realizan los seres humanos sobre un soporte, sea sobre piedra o sobre papel, tienen por objeto externalizar una representación mental de los objetos representados, lo que se encuentra en el origen del concepto de símbolo, y lo que se encuentra en el origen del lenguaje: externalizar intenciones recurriendo a un sistema de símbolos que permiten el intercambio y la distribución técnica del trabajo. **Pintoresco** tiene la misma raiz que el término inglés picture, o la misma raiz que el término español pintura, un término que identifica el producto de un proceso de ejecución por el que esa pintura se realiza. **Pintoresco** contiene un sufijo "sc" presente en otras muchas palabras, a veces de un modo transparente como en "arabesco", o de un modo opaco como en "grotesco", y siempre con el mismo sentido. Este sufijo es un operador conceptual que modifica la base semántica, la raiz, la base sobre la que se construye por composición la palabra: expresa un modo de ejecución o de presencia. Las esculturas de la cultura kazaja son esculturas pintorescas en la medida en que conforman una representación peculiar singularmente concreta y abstracta a la par. Los dibujos que sirven como patrones de estímulo de la tarea de clasificación en esta aplicación son representaciones que emulan o representan ciertos objetos dotados de características o propiedades singulares. Ninguno de estas propiedades hacen a la función que se supone en estos objetos artificiales, a saber, la de servir como encendedores por émbolo de presión de modo que la cabeza del encendedor se desplaza a través de un eje hacia un recipiente inferior en el que se encuentra el material ígneo, habitualmente yesca, hilo o tela que es lo que produce luego la brasa. En este sentido, los rasgos que identifican los objetos de estímulo son superfluos respecto de la función, son variaciones estéticas de un mecanismo que satisface una función única. Pues bien, de entre los distintos recursos que un ser humano ha empleado para producir fuego, este dispositivo es ciertamente singular, pintoresco, en tanto consiste en un principio físico por el que se produce una compresión rápida del aire en un un cilindro sobre el que se mueve un émbolo o pistón y se desarrolla en un objeto de rasgos prescindibles o meramente artísticos. El calor que se produce

como resultado de la compresión origina el incendio del material previamente introducido. El término evoca a la vez la tarea de clasificación y los dibujos que sirven como patrones de estímulo evocan un modo peculiar de producir fuego. La Figura 4 presenta los patrones de estímulo que se emplean en la tarea de aprendizaje de clasificación en esta aplicación.

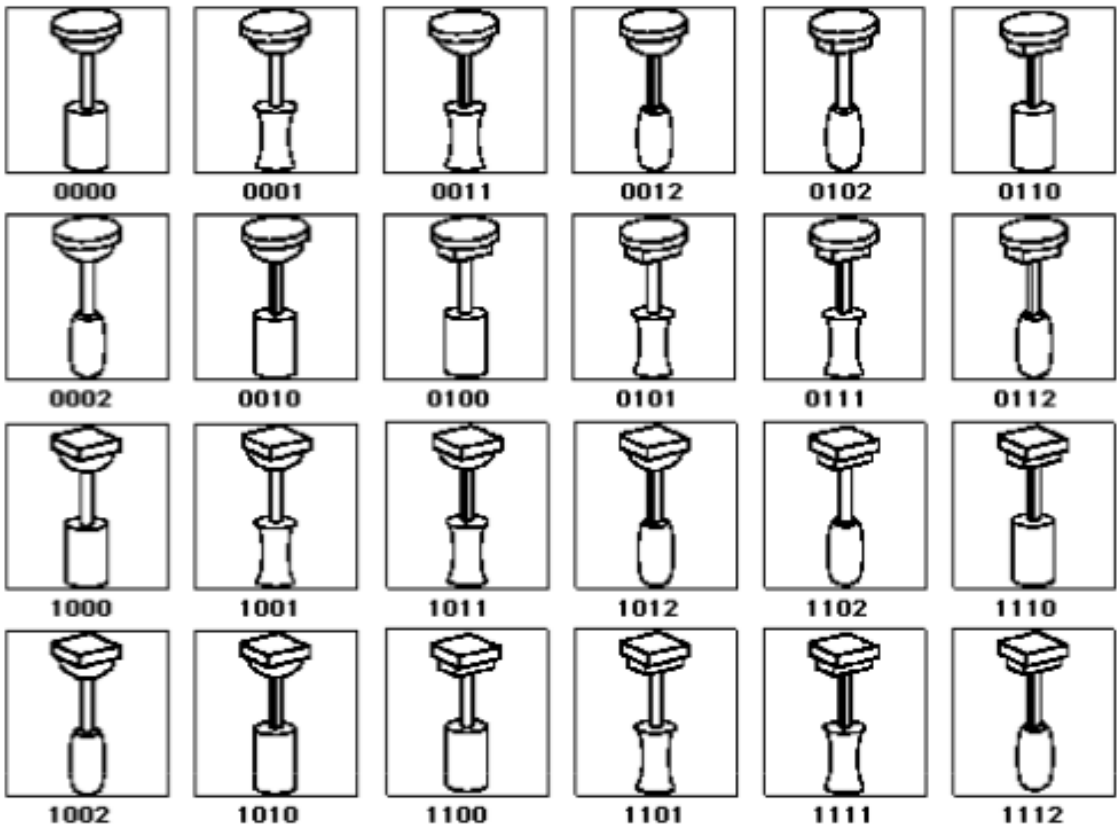


Figura 4: Patrones de estímulo empleados en la tarea de aprendizaje

2. Metodología. Análisis y Diseño de la aplicación.

¿Qué correspondencia existe entre la representación de una clase y el objeto que se reconoce como miembro de esa clase en cuestión?. El repertorio de objetos que se emplean en esta herramienta se presentan en la Figura 4. De este conjunto se extrae la muestra de estímulos que define cuatro tipos de reglas de clasificación. El sujeto es expuesto a una serie de patrones cuya clasificación varía según la estructura lógica de estos cuatro tipos de reglas:

- Regla de Rasgo: Un único rasgo permite identificar los miembros de una categoría.
- Regla de Rasgo Supraordinado Igual: Tres rasgos de los patrones se emplean como criterios de clasificación. Estos tres rasgos pueden a su vez identificarse con un único rasgo abstracto.

- Regla de Rasgo Supraordinado Desigual: Tres rasgos de los patrones se emplean como criterios de clasificación. Estos tres rasgos no pueden reducirse a un único rasgo abstracto.
- Regla de Ejemplar: La tarea de clasificación requiere memorizar todos y cada uno de los ejemplares que se clasifican como miembros de un concepto.

3. Arquitectura e implementación del proyecto.

Desarrollar una aplicación exige situarse en la perspectiva del usuario final. Para el caso, en el sujeto experimental que va a usar la aplicación autoadministrándose la tarea de aprendizaje en que esta aplicación se expresa. Con ese fin, se procede bajo lo que se conoce como diseño centrado en el usuario (DCU). El desarrollo de la aplicación exigirá entonces un análisis de los requisitos, la elaboración de un diseño de la aplicación, su implementación en un prototipo y en una evaluación iterativa de sus prestaciones. Este proceso de desarrollo en cuatro pasos identifica el ciclo operativo de una aplicación bajo las normas DCU. La aplicación que se presenta trata de ser una reproducción suficientemente fiel de este proceso de desarrollo, aunque la evaluación y modificación de la herramienta no se agote en su expresión actual. En efecto, parte del proceso está condicionado al uso efectivo de la aplicación por usuarios humanos que en interacción con la aplicación proveen información acerca de sus prestaciones y cómo la aplicación se ajusta tanto a las necesidades de los usuarios como al objetivo final. Con esta perspectiva se desarrollan los siguientes puntos de este epígrafe de la memoria.

3.1. Definición y planificación del proyecto.

La aplicación se desarrolla a partir de un área de investigación de amplísima tradición en psicología. A saber, cómo el sujeto humano conceptualiza e interpreta su experiencia a partir de la información que provee el sistema perceptivo en su distinta complejidad. Forma parte de la tradición de la psicología experimental que se origina con la teoría del aprendizaje y en particular de los modelos de condicionamiento clásico y operante u instrumental, tratar de construir un modelo que explique el proceso de formación de conceptos, o, en otros términos, el proceso por el que el ser humano establece una partición de un conjunto de eventos en clases. En ese proceso constructivo, el sistema emplea tanto información guiada por los patrones de estímulo o datos de entrada, como información proporcionada por un modelo previo de los eventos de la naturaleza y la regularidad con que se suceden. Técnicamente, la mente humana explota a través de un examen probabilístico de la concurrencia de rasgos que eventos pueden predecirse o no predecirse sobre la base de la frecuencia con que aquellos eventos ocurren. La teoría de la información proporciona un modelo

para este tipo de análisis. La aplicación desarrolla esta idea empleando una tarea de aprendizaje de clasificación que ha sido siempre la expresión empírica de un proceso de aprendizaje conceptual. Con este propósito se precisaron al inicio una serie de funcionalidades que debía tener la aplicación, y una secuencia de tareas que ahora ven cumplida respuesta en esta memoria.

3.2. Definición y análisis de requerimientos.

El objetivo principal de este proyecto era desarrollar una aplicación que permite identificar fenómenos que afectan a los procesos de aprendizaje conceptual en condiciones donde el criterio que permite hacer una partición de un conjunto de patrones reside no en los rasgos característicos de los estímulos mismos sino en la estructura lógica subyacente. La aplicación se diseño inicialmente para estudiantes de secundaria, bachiller y universitarios con el objetivo secundario de mostrar a éstos cómo se producen estos procesos de aprendizaje, ilustrando ciertos fenómenos empíricos para mostrar en un desarrollo o una ampliación posterior de la aplicación los efectos que es posible identificar en los procesos de reconocimiento de objetos, en particular, el proceso que parte del estímulo y se resuelve en la representación, y el proceso que parte de la representación y permite clasificar un patrón de estímulo.

Dado que la aplicación implica la intervención de un usuario, se concibió distinguir entre usuarios ordinarios y supervisores, de modo que la aplicación satisfaciera los criterios de control del usuario y de seguridad. Estos criterios identificaban los requisitos de funcionalidades relativas al registro del usuario que afectan a su creación e identificación, y funcionalidades relativas a la seguridad del acceso a la aplicación que afectan al control de acceso a la aplicación, y a la ejecución. Se ha resuelto conforme al plan previsto la identificación del usuario, pero no se ha creado una base de datos contra la que verificar si el usuario está autorizado o no a acceder a la aplicación. En este estadio se adoptó la decisión de dejar la aplicación abierta. Una versión posterior de la aplicación introducirá esta funcionalidad. El motivo de orillar esta funcionalidad en la versión actual fué, precisamente, para centrarse en aquellas funcionalidades que afectan a la tarea misma de aprendizaje conceptual.

Centrándose en los objetivos propiamente dichos del desarrollo de la aplicación se han desarrollado las funcionalidades específicas del proceso de aprendizaje conceptual que se pretendía imitar. A saber, funcionalidades relativas a la identificación de patrones de estímulo y funcionalidades relativas a la creación de particiones que fueran congruentes, de una parte con restricciones combinatorias y propiedades lógicas de la estructura interna de las clases en que se resuelve una partición, y de otra con las propiedades o rasgos de los patrones de estímulo en tanto

son los indicios sobre los que se soporta una partición de una serie de patrones en clases. Funcionalidades complementarias de estas son una serie de utilidades, en un caso utilidades para la construcción de una serie que se ha de presentar al sujeto de forma aleatoria, y en otros casos utilidades para la gestión de la aplicación. Estas funcionalidades constituyen un primer bloque de desarrollo de la aplicación y tienen una cierta complejidad.

Un segundo bloque de funcionalidades está representado por el desarrollo de las interficies de usuario, las vistas que permiten obtener información identificativa del usuario, la vista que permite establecer los parámetros de ejecución de la aplicación, la vista que permite la ejecución de la tarea de aprendizaje y la vista que permite guardar y verificar la ejecución de un usuario. Un tercer bloque de funcionalidades integrados en las clases que soportan estas interficies de usuario, está representado por la gestión de eventos iniciados por el usuario en interacción con la aplicación, la gestión de ensayos de la tarea de aprendizaje de clasificación, la gestión de imágenes de patrones de estímulo, la gestión de vistas integradas de tablas, la gestión de recursos que soportan el registro permanente y la edición de datos en archivos independientemente generados.

La aplicación se inicia obteniendo información del sujeto, los datos que le identifican. A partir de aquí se le presenta la opción de proceder a un proceso de aprendizaje por el que viene a ensayar sus respuestas en alguna de cuatro reglas de clasificación posibles. Se han identificado cuatro tipos distintos de reglas para la misma serie de estímulos. En esta fase del proyecto, la aplicación se aplica a distintos grupos de sujetos para distintos tipos de reglas.

El programa crea una serie de patrones que satisfacen la regla escogida, presenta las instrucciones al sujeto para que proceda, y presenta secuencial y aleatoriamente los ejemplares de la serie escogida, sometiendo cada ejemplar a clasificación, y devolviendo información sobre el carácter correcto o incorrecto de la clasificación realizada por el sujeto. Al término de una serie, el sistema vuelve a presentar la misma serie de ejemplares en orden aleatorio conservando la misma regla de clasificación. El sistema presenta, por decisión del sujeto, tantas veces como sea preciso para su completo aprendizaje, la misma serie de ensayos aleatoriamente hasta que se alcanza una cierta tasa de aprendizaje preestablecida. Al término del proceso la aplicación genera una base de datos que contiene información sobre la identidad del sujeto, el tipo de regla que ha escogido, la serie de ejemplares que se le han presentado y el tiempo de exposición y errores cometidos en la identificación de cada ejemplar en cuestión como miembro de alguna de las dos categorías alternativas, convencionalmente A o B. Simultáneamente a la presentación de la vista de Resultados, la aplicación crea sendos archivos que contienen los datos de la ejecución. Estos datos se guardan de forma permanente y se registran en archivos de texto accesible al análisis.

3.3. Diseño y arquitectura de la aplicación.

El desarrollo de la aplicación requiere dar respuesta a todas las funcionalidades especificadas. Del planteamiento inicial de clases se ha pasado a un diseño notoriamente más complejo (Figura 5).

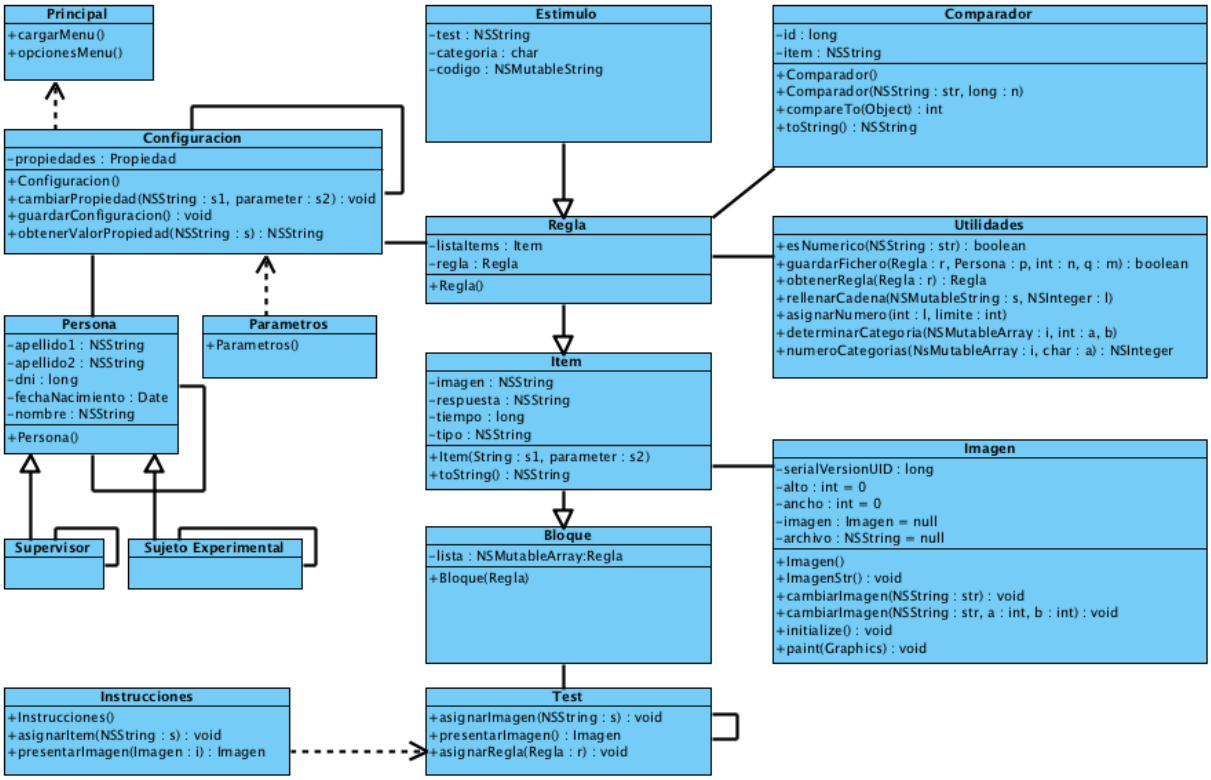


Figura 5: Diagrama de clases en la propuesta inicial

La aplicación demanda del sujeto el aprendizaje de cómo clasificar una serie de ejemplares. El sujeto procede generando hipótesis, y a medida que mejora su experiencia, comprueba la validez de sus hipótesis contrastando sus criterios de decisión con los rasgos que el estímulo presenta y la realimentación que recibe. La aplicación devuelve en cada ensayo, un mensaje que confirma o desmiente su elección. En una versión posterior de la aplicación se emplearán iconos informando al sujeto de los criterios e hitos de ejecución de la tarea tal como se ilustra en la Figura 6.

Figura 6: Iconos informativos de la ejecución

3.4. Implementación.

Para el desarrollo del proyecto se ha adoptado el sistema operativo IOS para IPad. La aplicación se ha desarrollado por completo en Objective-C empleando la aplicación Xcode que cuenta con un constructor de interficies de usuario potente y sofisticado. La aplicación es una aplicación autocontenida que se ejecuta con independencia de cualquier otra aplicación en el dispositivo móvil. No obstante, para comprobar y verificar la viabilidad del proyecto, se desarrolló con carácter previo, una aplicación en Java cuyo código fuente no se incluye en esta memoria. El objeto de desarrollar la misma aplicación en Java fué tener un ensayo piloto de la aplicación que al tiempo que permitía diseñar la aplicación en un nuevo lenguaje sirviera de guía en el desarrollo de una aplicación en Xcode cuando quien suscribe no tenía suficiente confianza en su competencia técnica para abordar una aplicación móvil desde el principio en un lenguaje de programación que aunque emparentado con el lenguaje c++, del que es un superconjunto, presentaba características propias, como el lenguaje Objective-c, complejo, en sí mismo, por la misma complejidad y extensión de las librerías que es posible emplear.

La aplicación Java piloto se sirve de una serie de documentos de texto, obtenidos, independientemente a través de una aplicación en Think Pascal, que contenía todos los casos posibles de cada uno de los cuatro tipos de reglas. En el momento de la segunda entrega de este proyecto, se contaba con una aplicación Java completa, excepto por lo que se refiere al empleo de estos ficheros de texto de los cuatro tipos distintos de reglas, previamente obtenidos. En la tercera entrega del proyecto, se completó una versión preliminar de la aplicación en Objective-C que permitió substituir estos archivos de texto por una serie de rutinas aleatorias que permiten construir ex novo sendas series aleatorias de los cuatro tipos distintos de reglas, lo que facilitaba una aplicación autocontenida en la que venían a integrarse las reglas que definen las condiciones experimentales de la tarea que se le propone al usuario. La tercera entrega de este proyecto, incluía la programación básica de la interfaz gráfica con la facilidad de haber desarrollado un prototipo de la aplicación en Java para un ordenador de sobremesa. La aplicación experimental de esta aplicación en Java nos permitió someter a prueba la herramienta que simultáneamente estaba desarrollando en XCode. La redacción de esta memoria incluye previos desarrollos, con nuevas rutinas e interficies, y notorias mejoras respecto de la última entrega del proyecto.

3.5. Integración y pruebas.

En esta fase del proyecto se examina el funcionamiento de la aplicación implementada según las especificaciones definidas previamente. La puesta en servicio de la aplicación a una muestra de

usuarios permite hacer correcciones y mejorar su funcionamiento al tiempo que permite obtener los primeros resultados. En este proyecto este proceso de puesta en funcionamiento se realiza en dos pasos, una primera directa en una prueba piloto que se presenta más abajo, la segunda indirecta en el momento de evaluación de la aplicación que finalmente se ha desarrollado.

3.6. Documentación.

En la última fase del proyecto se proporciona documentación de soporte de la aplicación. Esta documentación se expresa en un breve manual de usuario representado por un archivo de texto ReadMe que contiene instrucciones de instalación y uso de la aplicación junto con instrucciones de usuario. Además de esta documentación se adjuntan documentos de resultados de una prueba piloto final. Al objeto de explicar el proyecto se ha editado en formato de video una presentación de la aplicación por el propio autor.

4. Funcionalidades de la aplicación

El reto a superar de cualquier aplicacion con vocación de satisfacer ciertas necesidades reales de los usuarios consiste en poner a disposición la aplicación al público destinatario. Esencialmente, estos potenciales usuarios son de dos tipos, el usuario ordinario que ejecuta la tarea en el dispositivo y el usuario que hemos denominado supervisor que emplea la tarea como un modo de obtener datos del proceso de aprendizaje conceptual bien con un propósito estrictamente experimental, como un instrumento educativo en el aula universitaria, o como un instrumento diagnóstico de ciertas habilidades cognitivas humanas básicas. Dado el carácter experimental de esta aplicación, su desarrollo se ha centrado en un grupo reducido de usuarios, sujetos universitarios que emplean la aplicación como un recurso de aprendizaje, y como un modo de participación en una prueba de carácter experimental sobre los procesos de aprendizaje conceptual. Este grupo de usuarios tiene caraterísticas comunes y bien definidas y especifican un perfil de usuario concreto. Sin embargo, el mismo perfil puede básicamente aplicarse a otros usuarios ampliando los destinatarios objetivos de la aplicación.

4.1. Análisis de requerimientos.

En el plan inicial ya se contemplaba la dificultad de examinar el uso de la aplicación en un entorno real en un contexto en que el propio desarrollador debía probar la viabilidad de la propuesta y estudiar el modo de implementarla en un nuevo entorno de trabajo y en un nuevo lenguaje de programación. Aunque la aplicación se diseñaba como un instrumento de evaluación conductual de

una amplia muestra de usuarios, el propósito inicial sólo podía cumplirse para una muestra homogenea y bien definida de usuarios, la misma muestra que pudo experimental la herramienta en una prueba piloto. Concluida la aplicación, ésta podría ponerse a disposición de un grupo más amplio de usuarios hasta cumplir todos los objetivos propuestos de adaptación al usuario y de adaptación a los propósitos de la propia herramienta. El desarrollo de una aplicación piloto en Java tuvo el propósito que se expresa habitualmente en el análisis de requisitos previo al desarrollo de una aplicación del tipo que ahora se presenta.

4.2. Ejecución experimental de una prueba piloto

El desarrollo de la aplicación implica dos tipos de usuarios, un usuario que hemos denominado supervisor, que es quien provee el equipo para el desarrollo de la prueba, es decir, una tableta gráfica iPad, y que usa la herramienta con un propósito investigador, y un usuario que actúa como sujeto experimental. El usuario supervisor, en este caso, no tiene más implicación en la ejecución que poner a disposición del sujeto experimental el equipo, asegurándose eventualmente de que el sujeto ha comprendido las instrucciones, guiando al usuario ordinario en el caso en que el sujeto ignore cómo proceder y anotando cuántas observaciones puedan ser útiles para mejorar la herramienta, adicionalmente a lo que la ejecución del sujeto puede ofrecer por sí mismo a través de su propia ejecución. También puede decidir por él, en qué regla ha de entrenarse el sujeto experimental en lugar de permitirle a éste que escoja una regla al azar. Las instrucciones de ejecución de la tarea son idénticas para cada regla, y la aplicación es autocontenida, no requiere técnicamente ninguna instrucción que no sea proporcionada por la propia aplicación. El supervisor puede acompañar la ejecución, pero no se requiere en principio salvo por el tipo de usuario que puede pasar el experimento. El rango de edad abarca desde la temprana infancia hasta la vida adulta y la ancianidad. Sin embargo, en el contexto de este desarrollo se resolvió la aplicación con una muestra de 10 jóvenes universitarios asignados de manera forzosa aleatoriamente a una de las cuatro reglas de clasificación. Estas pruebas se ejecutaron sobre la aplicación piloto en Java. La actual aplicación que se presenta en esta memoria incluye una nueva implementación desarrollada para un dispositivo móvil con mejoras funcionales y nuevas funcionalidades en un entorno de programación y en un lenguaje distinto. Todos los sujetos de la muestra tenían visión normal o corregida a normal y estaban familiarizados con el uso de dispositivos móviles. Su identidad se oculta bajo sus iniciales. Por simplicidad no se incluyen latencias de respuesta y tasas de error, ya que el objetivo era unicamente someter a prueba una prueba piloto y examinar la viabilidad de la propuesta y la dificultad relativa de las distintas condiciones experimentales. En la Tabla 1 se presentan los resultados de esta prueba piloto en una versión en Java de la aplicación que se pretendía desarrollar.

Iniciales	Edad	Curso	Lateralidad	Regla	Aprendizaje
cfm	21	4	diestra	1	4 Series
cdb	22	4	diestra	1	3 Series
cbv	22	4	diestra	2	2 Series
cat	21	4	diestra	2	5 Series
cap	21	2	diestra	3	8 Series
asg	25	4	diestra	3	11 Series
abl	21	3	diestra	4	24 Series
rmm	20	2	diestra	4	> Criterio

Tabla 1: Resultados de una aplicación piloto en Java

De la muestra representada, unicamente un sujeto no alcanzó el criterio de aprendizaje en la presentación de 25 series de patrones de estímulo que es el criterio límite para determinar que se ha superado la etapa de aprendizaje. Obviamente no quiere decir que no ha aprendido a clasificar sino que fracasa para clasificar correctamente la serie de ejemplares propuesto. Recuérdese que la muestra efectiva de estímulos se determina aleatoriamente, y a pesar de que se construyen a partir de la muestra de estímulos presentada en la Figura 4, las reglas difieren en dificultad, siendo la más dificil la que requiere aprender la totalidad de los ejemplares, al no existir una regularidad detectable. Ningún sujeto tuvo dificultad en la comprensión de la tarea, ni en su ejecución, con independencia del tiempo que les requirió aprender, o fracasar en el proceso de aprendizaje.

El resultado de esta prueba piloto fue obtener evidencia razonable de la comprensión de la tarea y de la posibilidad de una correcta ejecución. La dificultad que experimentó cada sujeto no derivaba del manejo de la propia aplicación, sino de la misma dificultad de la tarea. Con independencia del lenguaje de programación empleado y del nuevo entorno de programación, esta experiencia permitió mejorar las instrucciones, incluir nuevas rutinas, automatizar el proceso de ejecución con alertas de tiempo controlado, rediseñar la aplicación por completo para soportar y dar funcionalidad a la posibilidad de emplear una pantalla táctil, frente al empleo previo de respuestas en el teclado, y rediseñar las interficies de usuario, además de incorporar rutinas para guardar de forma permanente los datos y grabar archivos de datos de la aplicación.

En suma, el resultado de esta prueba piloto permitió esencialmente confirmar la validez del formato y diseño de la aplicación en un equipo de sobremesa, lo que luego fue extraordinariamente útil para diseñar la aplicación para un dispositivo móvil.

4.3. Perfiles de usuario. Contextos y casos de uso

En su diseño actual esta aplicación tiene un uso técnicamente de herramienta de laboratorio, es decir, es una aplicación que a modo de juego permite evaluar ciertas habilidades cognitivas. Su validez última depende de la obtención de una muestra amplia y representativa del comportamiento de la población en esta prueba, en el modo en que ha sido usual en el desarrollo de un típico test. No está diseñada con el propósito de que el sujeto, mas allá de aceptar el reto, se vea recompensado por una actividad lúdica típica. No se dán condiciones para que la aplicación sea especialmente atractiva. Sin embargo, la prueba misma resulta ser entretenida, y en la prueba piloto los estudiantes se sintieron implicados en la ejecución con interés. En ello cuenta que fueran voluntarios y que contara como mérito. Mas allá de este objetivo de evaluación de una serie de habilidades humanas básicas como es la categorización y la conceptualización y el reconocimiento de objetos no presenta la aplicación ningún interés especial, aunque existen multitud de aplicaciones que sólo se difunden por la propia curiosidad, y el puro desarrollo de destrezas complejas. No importa cuán grande sea la libertad del sujeto en el proceso de ejecución de la tarea, es fácil comprobar que el incumplimiento de los requisitos por parte del usuario sólo afecta a la validez de los resultados obtenidos, y no tanto al propio usuario que puede siempre puede alegar desidi o ignorancia para no responder a lo que se le solicita, por ejemplo, eludiendo responder en ciertos ensayos o tratando de interrumpir la tarea. Esta posibilidad que no está abierta en el caso de la aplicación piloto, sí es, en cambio, posible en el caso de una aplicación móvil que puede verse interrumpida en cualquier momento.

4.4. Requisitos de usuario. Usabilidad de la aplicación.

Fuera de la necesidad de comprender las instrucciones y seguir de forma autónoma el proceso de autoaplicación de la herramienta, no exigen requisitos especiales por parte del usuario. El uso intensivo de la aplicación puede proporcionar información extra acerca de qué formato resulta más conveniente para evitar pérdida de tiempo e información. Una versión de la aplicación que ofrece menos grados de libertad al sujeto puede diseñarse sobre la misma base sin más que incorporar premios y/o castigos más eficaces que puedan responder mejorr a un modelo canónico de condicionamiento operante. En cualquier caso, y cómo se ha podido demostrar a través de la prueba piloto, funcionalmente mucho más hostil para un usuario comprometido, el uso de la aplicación en un contexto real permite (a) comprobar la usabilidad de la aplicación, especialmente por lo que se refiere a su visualización y a los controles táctiles que son sencillos y efectivos; (b) asegurar el acceso a la información y administrar los tiempos y ritmo de ejecución; (c) asegurar la correcta ejecución de la prueba; (d) garantizar la fidelidad de los datos obtenidos y confirmar su precisión; y (e) ofrecer

una aplicación visualmente atractiva aunque el verdadero valor de la prueba central de la aplicación, la tarea de aprendizaje conceptual, sea la funcionalidad principal y no existan concesiones al usuario para convertirla en un juego que compensa la ejecución por sí misma.

En suma, la aplicación satisface bajo mi criterio las pautas recomendadas por las Human Interface Guidelines: (a) ofrece un uso sencillo y sumamente obvio. La entrada de información no representa ningún problema. El usuario puede autoadministrarse la tarea sin más que escoger el estímulo sobre el que desea dar su respuesta, sabiendo que no puede ignorar ningún patrón, y con el incentivo de acertar en la clasificación; (b) minimiza las acciones del sujeto; las alertas emergen y desaparecen después de que el sujeto haya tomado sus decisiones y sin su intervención; (c) existe un equilibrio entre las necesidades del usuario y el objetivo de la aplicación de obtener datos de ejecución en una tarea de aprendizaje conceptual sin sacrificar su libertad de acción, requiriéndole unicamente su cooperación. La aplicación se centra en la tarea principal que el usuario ordinario debe ejecutar, las demandas de la tarea se adecúan a lo que un usuario puede necesitar al requerir respuestas táctiles de fácil ejecución, la información es accesible y simple desde las instrucciones que se ofrecen de la tarea hasta la forma en que el sujeto debe resolverla, la terminología empleada es accesible e intuitiva, y la visualización de los patrones de estímulo y los botones de ejecución son claros y sus significados comprensibles. La aplicación aprovecha las posibilidades que ofrece la implementación de controles táctiles, y puede visualizarse en cualquier orientación sin deformación de la imagen. El proceso de ejecución de la tarea principal facilita la rápida activación de un patrón de estímulo, y las respuestas de la aplicación son contingentes al ritmo de ejecución del sujeto, que conserva toda su libertad para continuar usando la aplicación o para interrumpirla.

4.5. Aplicaciones relacionadas.

Al igual que se propuso en el desarrollo de este proyecto, existió hace tiempo en la comunidad científica en Psicología un amplio consenso sobre la utilidad de implementar ciertos experimentos clásicos de papel y lápiz en ordenador. A pesar de ello, la mayor parte de las aplicaciones en Psicología no han salido del ámbito del laboratorio; ese interés no se ha traducido en un uso extensivo y sistemático del ordenador como herramienta de evaluación e intervención. Las aplicaciones existentes en hospitales y centros de rehabilitación y neuropsicología son especialmente pobres técnicamente además de inactuales en relación con el desarrollo de la psicología actual. Para colmo de males la mayor parte de las aplicaciones en uso exigen pago de derechos a terceros, y forman parte de un desequilibrio de capital entre importaciones y exportaciones. Este estado de cosas no ha impedido que se manifieste el mismo consenso en torno a la necesidad de desarrollar aplicaciones móviles que implementen pruebas de evaluación psicológica de naturaleza

experimental, aplicaciones de evaluación neuropsicológica y herramientas de intervención conductual y rehabilitación neurocognitiva. Una muestra de ese interés estuvo en el origen de este proyecto. Los objetivos del proyecto se relacionan con el propósito de construir una herramienta de evaluación de las habilidades cognitivas del sujeto, las habilidades que se expresan en su capacidad para discriminar entre clases de objetos por referencia al proceso de aprendizaje que precede a la formación de conceptos y categorías. No existe en nuestro ámbito, sin embargo, ninguna aplicación conocida de este carácter, una aplicación orientada al desarrollo de una herramienta de evaluación implementada con Sistema Operativo iOS en una tableta gráfica como el iPad de Apple. Sin embargo, existen aplicaciones, en otros continentes especialmente estadounidenses y japonesas, y en Europa alemanas, y británicas, que se ejecutan en un equipo estacionario con objetivos y propósitos análogos al que se presenta en este proyecto. Recientemente, se han desarrollado algunos experimentos piloto para tableta gráfica en Francia, sin la pretensión de constituir una herramienta de diagnóstico, pero, excepto su propósito general, sus prestaciones no son accesibles, celosos sus creadores de guardar las aplicaciones para su uso en sus laboratorios. Por otra parte, la consideración y estudio de unas u otras aplicaciones no es pertinente en este contexto, por cuanto o bien no son herramientas desarrolladas para dispositivos móviles, o cuando es el caso excepcional, no son productos comerciales y han tenido un carácter puramente instrumental, y sus autores no son proclives a compartir sus estrategias técnicas, y aplicar la herramienta más allá del entorno de laboratorio. En suma, la oferta que el completo desarrollo de este proyecto puede representar en el futuro, no se encuentra en la actualidad disponible en el mercado, definiendo su desarrollo un nicho de mercado de muy alto valor añadido. En lugar de las usuales pruebas de papel y lápiz que a menudo sólo examinan errores o en lugar de las usuales pruebas psicotécnicas e informales de un servicio de neurología típico, o la implementación de ciertas pruebas clásicas en un ordenador, no existen aplicaciones representativas del propósito que inspiró este proyecto. Las herramientas móviles exigen un cambio conceptual al permitirle al sujeto abandonar el teclado por una pantalla interactiva, y al permitir la incorporación de sensores y actuadores que se adecúan mejor a la dinámica del usuario. Este proyecto inicia una andadura que pretende que, en el futuro, este tipo de herramientas se conviertan en modelos usuales de diagnóstico y rehabilitación, lo que será consecuencia tanto del conocimiento que estas herramientas integren, como de su diseño para ofrecer datos interpretados procesados automáticamente, al modo en que se emplean en otros ámbitos pruebas radiológicas invasivas y no invasivas, y en correspondencia con el precio y versatilidad que ha de procurar el desarrollo de la tecnología móvil. Precisamente, se ha abierto un espacio extraordinario con la integración de sensores para sistemas de diagnóstico automático ejecutables en dispositivos móviles.

Este proyecto se centra en el desarrollo de una aplicación móvil que permite al usuario aprender a clasificar una serie de objetos artificiales de acuerdo con un tipo de regla de clasificación que requiere del sujeto la formación de una hipótesis que finalmente resuelve cómo se clasifica una serie de ejemplares. La aplicación registra todos los eventos conductuales que tienen lugar, registrando el tiempo de exposición del patrón, la categoría que el sujeto le asigna, y el número de ensayos que le requiere aprender la clasificación. La aplicación genera un documento cuyos datos pueden analizarse posteriormente, al objeto de identificar qué tipo de regla resulta eventualmente más eficiente en términos de eficiencia y velocidad de aprendizaje. Una tarea secundaria permitirá examinar cómo contribuye una etapa previa de aprendizaje a una etapa en que el conocimiento adquirido se transfiere a la ejecución de una nueva tarea, una tarea de búsqueda visual de un objeto entre un conjunto de objetos distractores.

El potencial de desarrollo de la aplicación que se presenta es sencillamente enorme; sin embargo, su proyección excede las condiciones de tiempo que usualmente se asignan a un Trabajo Final de Carrera. Es nuestro propósito extender la aplicación para que constituya una herramienta de evaluación de aprendizaje conceptual con relevancia diagnóstica. El desarrollo que ahora ha venido a completarse en el curso de un cuatrimestre es la de una aplicación móvil que permite simultáneamente evaluar la competencia de un usuario en una tarea de aprendizaje de clasificación, e identificar las bases lógicas, el tipo de restricciones que operan en un sujeto para aprender con mayor o menor facilidad un tipo de estructura, cuando la serie de estímulo es formalmente idéntica a cualquiera del mismo tipo y los patrones de estímulo son análogos para cualquier regla. El usuario es un usuario de cualquier edad, capaz de discriminar objetos distintos por referencia a la clase que pertenecen, según los rasgos, y la combinación de rasgos que presenta cada ejemplar de una cierta categoría. En cualquier caso, la clasificación es un tipo de clasificación binaria, que implica el aprendizaje de clasificación de un máximo de 16 ejemplares y un mínimo de 10 ejemplares. La serie específica que se le presenta al sujeto es resultado de un proceso de construcción aleatorio, cuya estructura se evalúa según ciertas condiciones lógicas, de modo que los criterios de clasificación son por completo independientes de los rasgos específicos de los ejemplares. De este modo, la evaluación se realiza sobre la estructura lógica de la clasificación que resuelve una partición binaria del conjunto, y no sobre una serie específica de patrones de estímulo.

4.6. Conclusiones del estudio.

La aplicación piloto del experimento en la versión desarrollada en Java nos permite confirmar la validez y viabilidad de la propuesta. El modo en que se ha desarrollado la aplicación, las interficies desarrolladas en la interacción de la aplicación y el usuario satisfacen las recomendaciones y criterios

de las Human Interface Guidelines y el diseño satisface las condiciones de usabilidad que determinan la mínima complejidad de la aplicación para el usuario. No contamos, de otra parte, con aplicaciones comparables a la que se ha desarrollado. Sin embargo, esta es una cuestión de oportunidad que pronto ocupará buena parte de la oferta para los destinatarios profesionales de esta aplicación. Es previsible, incluso, que se desarrollen de forma extensiva aplicaciones abiertas que permitan desarrollar experimentos en dispositivos móviles. Sin embargo, la aplicación que presentamos tiene por objetivo convertirse en un modelo de evaluación del aprendizaje conceptual. Esta necesidad no está por ahora cubierta.

5. Diseño de la aplicación e interficies de usuario.

Según los objetivos propuestos de desarrollar una aplicación orientada a la evaluación del aprendizaje mediante el recurso a una tarea de aprendizaje de clasificación, se han definido dos tipos de usuarios, un usuario supervisor que facilita la ejecución de la tarea, y un usuario ordinario o no profesional que hemos denominado sujeto experimental tal como Figura 7 que presenta el Diagrama de Casos. Apoyándonos en el diseño conceptual del proyecto hemos diseñado un prototipo de la interfaz de usuario.

5.1. Interacción de casos de uso.

La Figura 7 presenta el Diagrama de Casos de Uso de la aplicación. Pueden reconocerse en la figura como usuarios humanos, el supervisor y el sujeto experimental con los roles que se han definido previamente.

Tanto el supervisor o ayudante, como el usuario ordinario se identifican frente a la aplicación. Sobre cada tipo de usuario operan restricciones distintas que puede evaluar el propio sistema. Sin embargo, la versión actual de la aplicación no incorpora bases de datos contra las que verificar la identidad del usuario autorizado.

Los usuarios especiales, el sistema y el reloj controlan la interacción y hacen posible la ejecución de la tarea por parte del usuario humano. El sujeto experimental es el eje en torno al que gira la aplicación. Procede al inicio a autenticarse como usuario registrado, o a identificarse frente al sistema. En tal caso, el sistema verificaría la autenticidad y aplicaría criterios selectivos para determinar la posibilidad de que el sujeto en cuestión fuera usuario de la aplicación. En la actual implementación el sistema procede, incondicionalmente, a la creación del usuario y su registro. El sujeto, una vez se inicia la aplicación, y se autentica, procede a escoger entre las opciones de tarea y las opciones de reglas. En el estado actual de la aplicación sólo una tarea es posible: el aprendizaje

de la clasificación. Procede entonces a elegir una regla de clasificación, y leer las instrucciones de la tarea. El supervisor puede haber configurado los parámetros de la aplicación, o pueden aplicarse los parámetros de la aplicación por defecto. En este contexto, sólo se ha implementado un conjunto único de parámetros, aunque la aplicación puede ser fácilmente adaptada a cambios muy sustantivos, que afectan al número de ejemplares de cada categoría según la regla de clasificación escogida. El supervisor puede también seleccionar tarea y regla y someterle al sujeto experimental a las condiciones que ha determinado. La herramienta se ha diseñado para que el usuario ordinario pueda autoadministrarse sin más que seguir ciertas instrucciones previas.

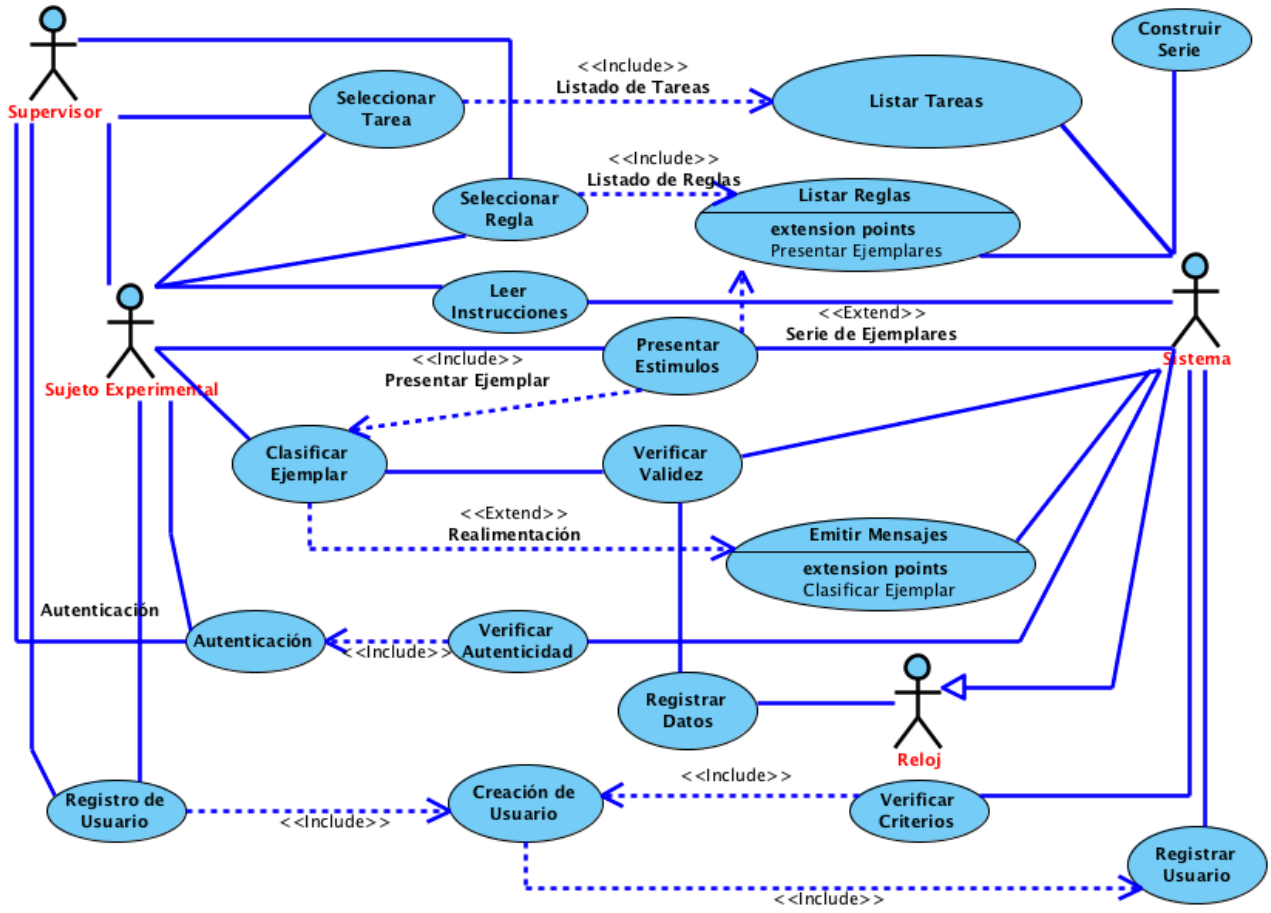


Figura 7: Diagrama de casos de uso de la aplicación

5.2. Diagrama de estados. Flujos de la interacción

Las Figuras 8 y 9 presentan los flujos de la interacción de los componentes básicos de la aplicación. En la Figura 8 se presenta el flujo que identifica el proceso por el que se identifican los usuarios. Los usuarios se identifican frente al sistema, pero el acceso es incondicional. No se ha implementado el proceso de autenticación de usuarios en la actual versión.

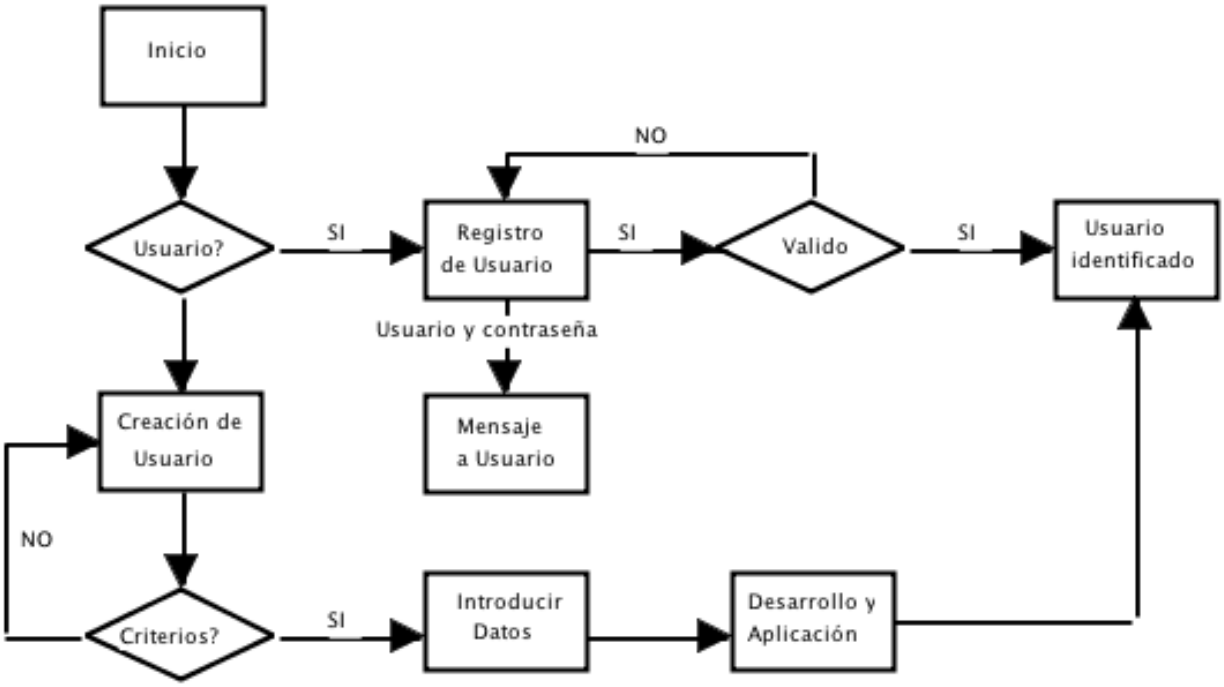


Figura 8: Diagrama de flujo de interacción del proceso de autenticación

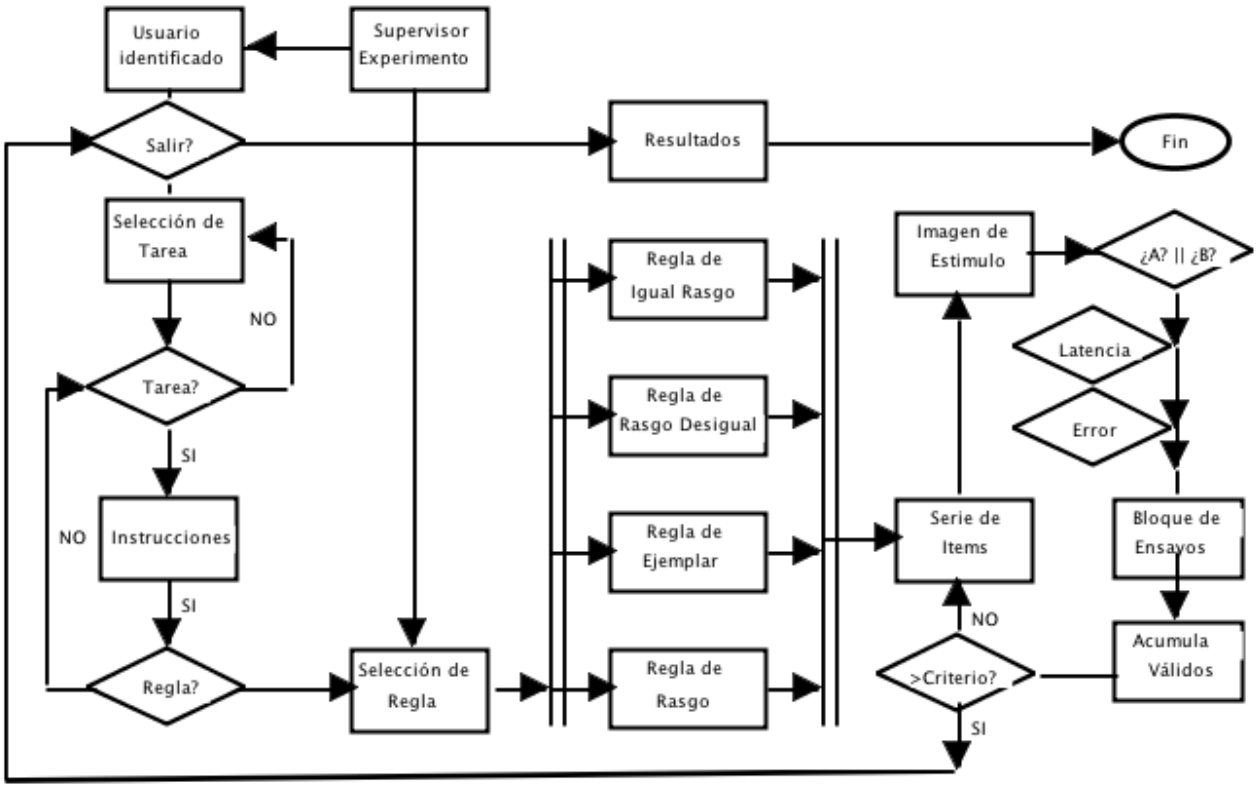
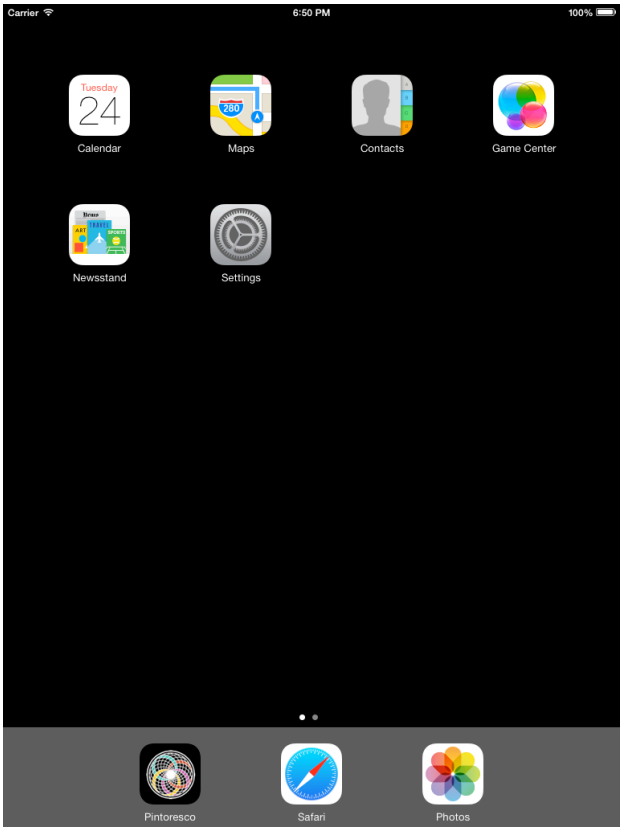


Figura 9: Diagrama de flujo de interacción del proceso de ejecución de tarea

En la Figura 9 se presenta el Diagrama de Flujo de la Interacción del proceso de ejecución de tarea por parte del usuario de la aplicación, y la intervención de otros tipos de usuario.

5. 3. Desarrollo del prototipo y diseño de interficies.

La vista inicial se abre tras la activación de la aplicación en una ventana propia de acceso. La Figura 10 presenta los iconos de las aplicaciones disponibles en la ventana del iPad. El icono de la aplicación **Pintoresco** se encuentra en la barra inferior de aplicaciones. Aunque puede moverse a la ventana principal se ha singularizado de este modo para esta presentación.



En su versión actual la aplicación no distingue entre tipos de usuario, supervisor u ordinario o experimental; es éste último el que adquiere todo el protagonismo en la versión actual. En cualquier caso, se identifica al usuario según si se encuentra ya registrado, o si se registra como usuario por primera vez permitiendo un acceso más inmediato al usuario ya registrado. Esta funcionalidad no se encuentra habilitada en la versión actual. Sobre ambos tipos de usuario, supervisor u ordinario operarían distintas restricciones de acceso o restricciones singulares, pudiendo un usuario figurar en ambos registros, o figurar únicamente en el registro de usuarios ordinarios.

Figura 10: Icono de la Aplicación

El formulario de registro inicial recoge información que permite identificar al usuario ordinario. En la carátula de la aplicación se requiere la identificación del usuario, la contraseña de acceso, y el correo electrónico del usuario, información de cuenta de correo que debe repetirse. La información puede verificarse mediante el envío de un mensaje de correo electrónico a la dirección de correo indicada. La Figura 11 presenta el contenido de la página de acceso inicial. La funcionalidad de recuperación de contraseña se ha suplido con el envío de un mensaje electrónico que incorpora información del nombre del usuario y la contraseña si este usuario constara en la base de datos que no se ha implementado en la versión actual. El botón "**Crear una cuenta de usuario**" dá acceso a la ventana en la que se requiere al sujeto que supla la información que se le solicita.

En el caso de encontrarse ya registrado, el usuario puede pulsar en el botón **Acceso** de la barra inferior de navegación. Pulsando este botón se presenta la vista que permite definir los parámetros que afectan a la ejecución de la tarea experimental. Una versión futura de la aplicación impedirá este acceso directo a usuarios no registrados previamente.

La pantalla inicial ilustra el motivo de la aplicación; la imagen de fondo presenta los quemadores que se emplean para la elevación de un globo aerostático. La imagen de los quemadores evoca el tipo de estímulos que se emplean.

Figura 11: Vista de Acceso

Una funcionalidad pendiente en la versión actual es no permitir que un usuario sea únicamente supervisor. En la práctica, la distinción entre supervisor y ordinario es artificial si se cumplen los objetivos de la aplicación, de captar datos de un usuario que ejecuta la aplicación de forma autónoma, o por sí mismo. La Figura 12 presenta el formulario de datos personales que permiten identificar a los usuarios. La ventana de creación de un nuevo usuario se divide en dos secciones, la primera sección recoge datos personales, y la segunda recoge el domicilio. Rellenados los campos, y verificado su contenido, se presenta una ventana

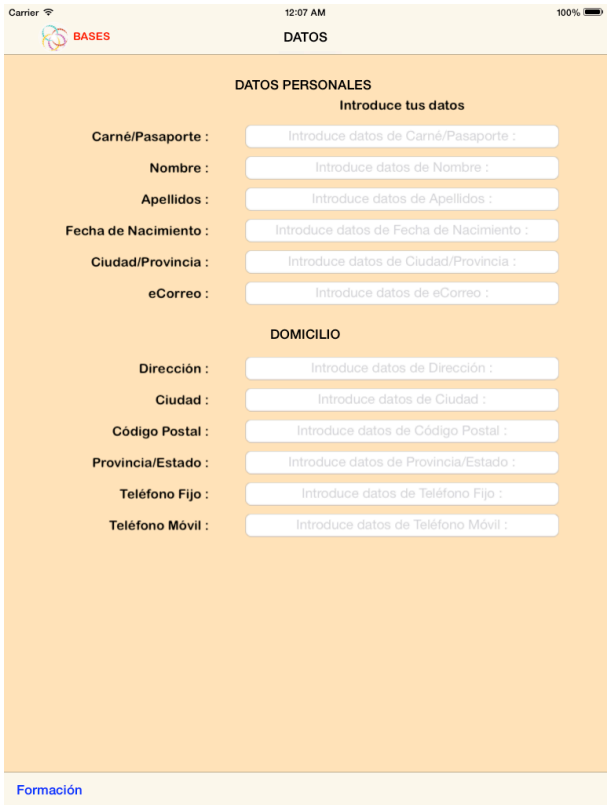


Figura 12: Vista de Datos

de datos complementarios que recoge información sobre la formación académica del usuario y su experiencia laboral. Eventualmente estos datos pueden ampliarse con información complementaria previamente obtenida que puede afectar al rendimiento de la aplicación, y a sus objetivos de medida.

La Figura 12 solicita la información relativa a la identidad y domicilio del usuario. La Figura 13 extiende ese formulario recogiendo información académica, y profesional o laboral. El formulario permite la inclusión de una fotografía que se toma en el momento de la ejecución de la aplicación, y se incorpora a la base de datos de la aplicación. (En esta oportunidad se ha incluido la fotografía de quien suscribe el proyecto). En el diseño de los campos de texto se ha seguido una doble metodología, de una parte se han definido manualmente campos de texto que se incluyen como posibles objetos de la interficie de usuario en el entorno Xcode; de otra parte, se han programado campos de texto de forma explícita, configurando el campo en la forma que resulta más apropiada. Este método permite una definición métrica estricta de los campos, y un control de su contenido.

La información que puede recuperarse de este formulario puede parametrizarse de modo que pueda tratarse posteriormente estadísticamente para establecer relaciones entre la ejecución del sujeto en la tarea, sus habilidades para identificar los ejemplares como miembros de distintas categorías en una tarea de clasificación, y detectar regularidades de distinta complejidad. Esta parametrización permite la validación automática de la aplicación cuando sea ejecutada por una muestra suficiente representativa de sujetos experimentales. Eventualmente, esta información puede completarse con distintos

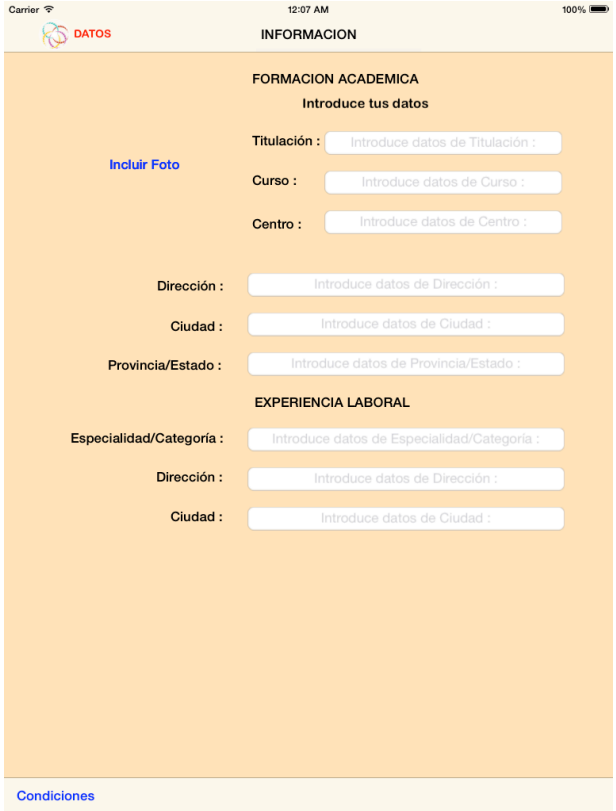


Figura 13: Vista de Datos

índices potencialmente diagnósticos independientemente evaluados. En esta oportunidad se ha adoptado un criterio de datos mínimo. En los campos de texto de los formularios de recogida de datos se restringe el tipo y rango de datos que pueden introducirse de modo que pueda filtrarse información indeseada; de este modo cada campo prohibe la introducción de ciertos datos, numéricos u otros, o datos alfanuméricos. En el campo de correo electrónico se permiten caracteres

alfanuméricos y especiales, aquellos que sirven para validar si la dirección de correo en cuestión puede ser efectivamente real. El desarrollo de una funcionalidad específica en versiones futuras permitirá comprobar en tiempo real si la dirección de correo existe efectivamente. En esta oportunidad la aplicación sólo verifica si el formato de la dirección de correo es plausible. El usuario cuenta con una ayuda en tiempo real acerca del campo que está rellenando; asimismo, puede borrar el campo automáticamente y proceder a rellenarlo de nuevo desde el principio. Cada campo de texto activa el teclado apropiado al campo, y desaparece tras pulsar la tecla de retorno de carro.

Una vez identificado el usuario, o tras su registro en el caso en que no se hubiera registrado antes, es posible activar la ventana de parámetros de la aplicación a través del botón de la barra inferior de navegación en la ventana activa, sea la ventana inicial o la ventana que recoge información del usuario.

Los parámetros por defecto se presentan en la Figura 14. De los 16 ejemplares posibles, se ha adoptado el criterio de que 7 pertenezcan a la categoría A (ó B), y 5 pertenezcan a la categoría B (ó A). Esta distribución permite testar con el mismo conjunto de ejemplares todas las condiciones experimentales. Aunque la aplicación soporta otras opciones, que tambien pueden satisfacer las cuatro condiciones experimentales, por simplicidad se ha adoptado este criterio frente a otras opciones posibles.

La selección de los ejemplares se efectúa por la propia aplicación, de modo que existen según el tipo de condición experimental o regla un número indefinido de posibles combinaciones de ejemplares que satisfacen la misma condición experimental en cuestión. Una serie de ejemplares define un bloque de ejecución. Cada bloque está compuesto por parámetro por 5 series consecutivas. El usuario puede modificar a voluntad el número de series consecutivas que desea que se le presenten. La presentación de cada ejemplar en una serie es aleatoria por lo que el sujeto no detecta ruptura entre una serie y otra. Simplemente, se repiten las series hasta completar un bloque.

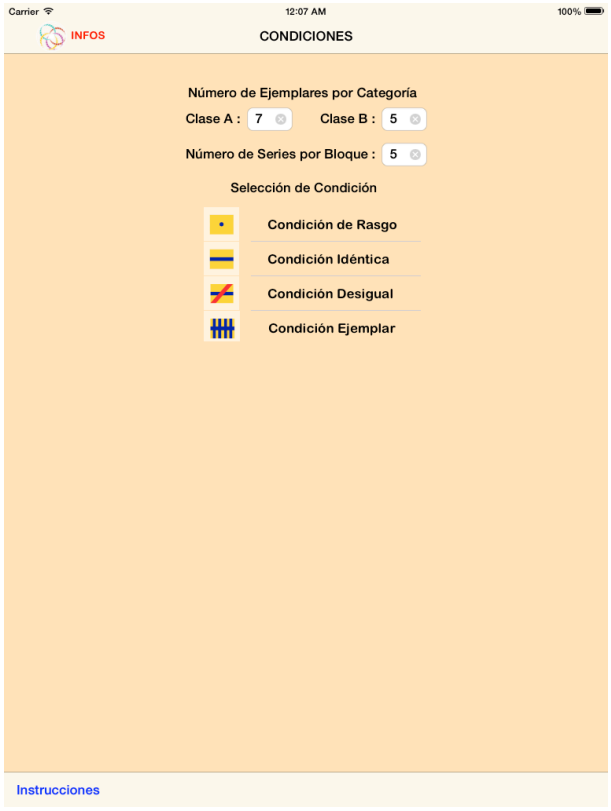


Figura 14: Vista de Parámetros

Si el sujeto no ha aprendido, tras la ejecución de un bloque, el proceso se repite a voluntad. El supervisor, o en su caso el usuario por sí mismo, y de acuerdo con las instrucciones recibidas, elige de entre las posibles opciones una determinada condición experimental. Todas las opciones son alternativas. La pulsación de una opción anula todas las otras opciones potencialmente posibles.

Al pulsar en una de las opciones emerge una alerta que le indica la condición elegida. Esta alerta desaparece automáticamente sin que el sujeto se vea obligado a pulsar en el botón de aceptación. Una vez, el sujeto ha pulsado el botón de retorno de carro en cada campo de texto, y ha escogido la condición experimental, puede pulsar el botón de **Instrucciones** de la barra de navegación inferior. Este botón dá acceso a la ventana de Tarea en la que el usuario ejecuta la tarea de clasificación. La ventana de parámetros es accesible al usuario ordinario. Sin embargo, habitualmente estos parámetros no se cambian de un sujeto a otro. Esta es una decisión que corresponde al supervisor, en el caso en que exista, quien habitualmente se limita a comprobar que estos parámetros son los correctos.

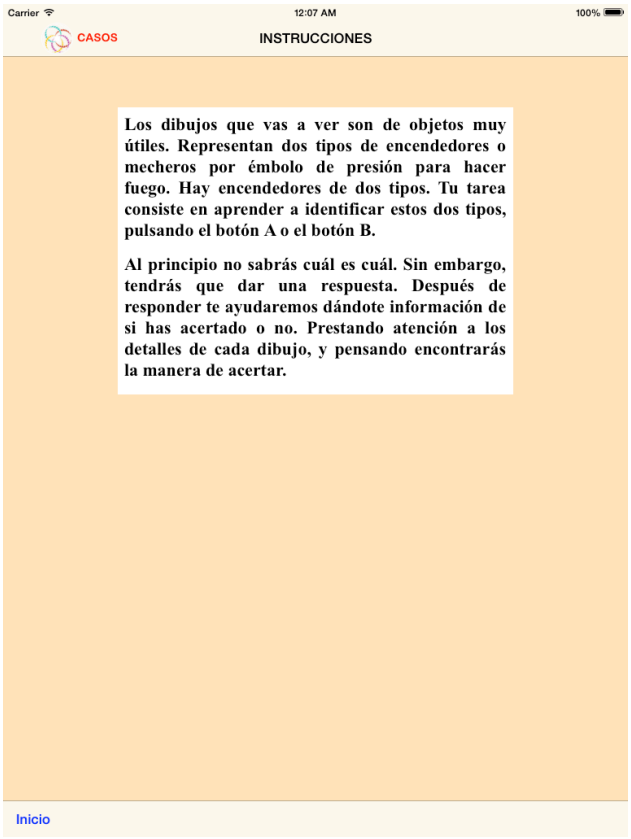


Figura 15: Vista de Instrucciones

La propia aplicación no permite introducir cualquier valor en la ventana de parámetros.

La Figura 15 presenta la ventana de **Instrucciones** en la que se presenta la caja de texto desplegable que contiene las instrucciones. Cuando concluye la lectura de las instrucciones, el sujeto experimental puede pulsar el botón **Inicio** que inicia la ejecución; el botón que se encuentra a la izquierda en la barra inferior de navegación le dá acceso a la ventana de Tarea. La vista de **Instrucciones** presenta un texto html con una barra lateral deslizante, permitiendo al usuario leer todo el texto.

Tan pronto como se han fijado los parámetros de la aplicación, el sistema procede a construir la serie experimental que se ha de presentar al sujeto para su aprendizaje. Generada la serie de patrones de estímulo que se corresponde con la condición en cuestión, y se cierra la vista de Instrucciones se activa la ventana de Tarea una vez se pulsa el botón de Inicio.

La aplicación construye un listado de patrones de estímulo que satisface la condición escogida, y aleatoriza el orden en que se han de presentar al sujeto. La ventana de tarea le permitirá visualizar los distintos ejemplares de la muestra sobre los que el sujeto elige clasificar en las categorías A/B. Entretanto no se ha procedido a la ejecución de la tarea, el sujeto puede retornar a las ventanas anteriores que recogen los datos ingresados. La información previamente introducida se borra en tal caso, debiendo procederse a rellenar de nuevo los distintos campos de texto. Para proceder hacia atrás el usuario debe pulsar el botón de regreso en la ventana superior de navegación que presenta el icono de la aplicación y el título de la ventana anterior.

Cada condición experimental impone costes cognitivos crecientes, de acuerdo con un orden de complejidad predecible: Condición de Rasgo </>/= Condición Idéntica <= Condición Desigual <= Condición Ejemplar. Existe una diferencia crítica entre la habilidad del sujeto para aprender en la Condición de Rasgo mejor que en la Condición Idéntica cuando el sujeto atiende a un único rasgo de los ejemplares específicamente recuperado de la información de estímulo frente a cuando el sujeto atiende a un rasgo abstracto de los ejemplares que elabora a un nivel de mayor complejidad, como sería el caso de la Condición Idéntica. La condición de Ejemplar requiere, habitualmente, la completa memorización de los ejemplares. Estas condiciones experimentales identifican habilidades básicas del sujeto experimental. Grosso modo, aquellos sujetos menos proclives a establecer relaciones entre rasgos, clasificarán mejor los ejemplares de la Condición de Rasgo que los ejemplares de cualquier otra condición. Sin embargo, aquellos más proclives a establecer relaciones abstractas entre los rasgos que identifican los ejemplares, clasificarán los ejemplares de la condición Idéntica mejor o igual que la Condición de Rasgo y mejor que en cualquier otra condición experimental. El valor diagnóstico de este desarrollo se expresa en la disposición del sujeto a reconocer en una tarea de búsqueda visual un cierto objetivo entre un conjunto de patrones distractores. La tarea de búsqueda visual no se ha implementado en la versión actual. La habilidad del sujeto para reconocer un ejemplar depende del tipo de regla en la que ha sido entrenado, y el proceso de aprendizaje a que ha sido sometido afecta a su habilidad para discriminar entre distintos ejemplares. Tan pronto como el sujeto alcanza el criterio de tres series consecutivas sin cometer ningún error con cada uno de los ejemplares, la aplicación concluye el proceso de ejecución de la tarea.

En la ventana de Tarea de Aprendizaje (Figuras 16 y 17) se visualiza una pantalla en fondo negro que contiene cuatro botones virtuales, programados de manera que la pulsación táctil o de ratón dá lugar a la ejecución de una cierta acción. El botón de **Inicio** despliega dinámicamente los cuatro primeros ejemplares de la serie experimental que representan la condición escogida. Este botón debe pulsarse unicamente una vez, al principio.

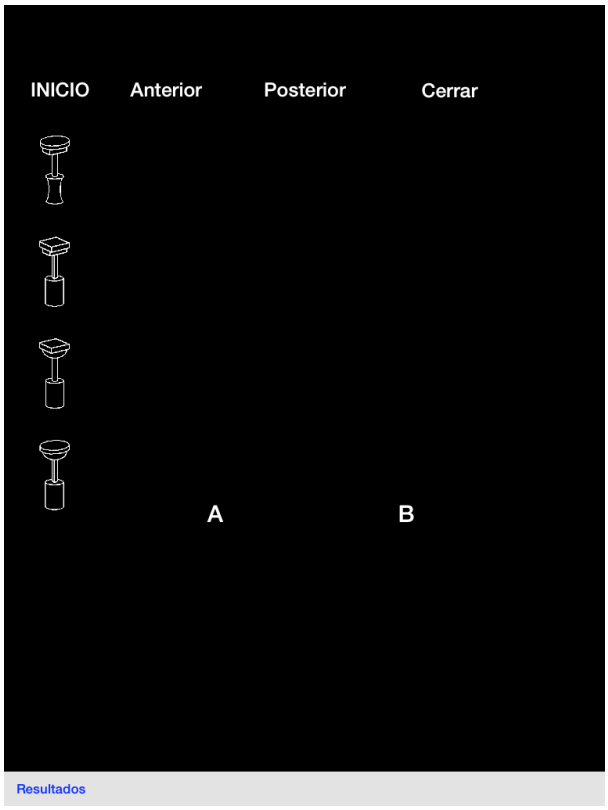


Figura 16: Vista de Tarea

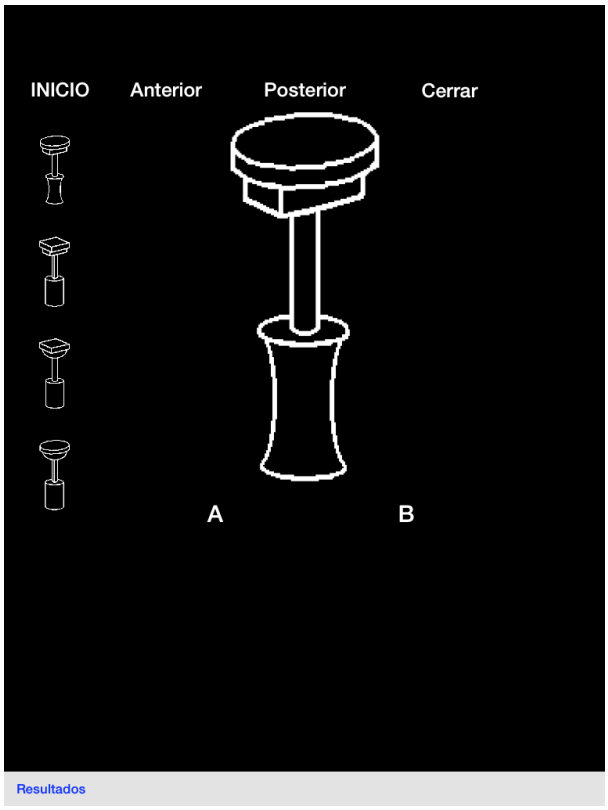


Figura 17: Vista de Ejecución

Tantas veces como se pulse el botón **Inicio** aparecerá una nueva serie de ejemplares, pero toda la información hasta entonces obtenida en el proceso de ejecución se pierde. Así pues este botón reinicia siempre la tarea, de tal modo que sólo está previsto que se pulse una única vez al principio de la ejecución. Aparecen entonces en el margen izquierdo 4 ejemplares en miniatura. Pulsando en cualquier de ellos se presenta el patrón de estímulo en cuestión en el centro de la pantalla. Una vez se presenta en su tamaño normal el estímulo, el sujeto puede responder pulsando A o B, en las etiquetas que identifican cada tipo de respuesta. El botón **Anterior** le permite volver hacia atrás en la muestra. Obviamente este botón sólo actúa si existe una serie anterior de 4 ejemplares, lo que no siempre es el caso. El botón **Posterior** le permite seguir hacia adelante en la serie, mostrándole un nuevo conjunto de ejemplares, hasta la última pantalla en que ya no se presentan más ejemplares por haberse visualizado y respondido a todos los ejemplares de la serie experimental. Cuando se han visualizado y se ha respondido a cada uno de los ejemplares de la serie experimental, concluye la ejecución de la serie, y cuando se pulsa el botón **Posterior** ninguna nueva serie se presenta. El sujeto puede entonces pulsar el botón **Cerrar** que hace desaparecer el último ejemplar a que se ha respondido, o pulsar en tal caso una de las opciones de respuesta A/B, o pulsar directamente en el botón **Resultados** de la barra inferior de navegación. Es posible cerrar la ventana y pasar a la vista final de resultados sin haber concluido por completo la tarea. En tal caso, la tarea no se ha completado y los resultados obtenidos no son técnicamente válidos.

Durante el proceso de ejecución, el sistema y el reloj del sistema identifican la velocidad de respuesta en la emisión del sujeto, y determina la validez de la respuesta según corresponda. El sistema devuelve una alerta que le permite al sujeto determinar si su respuesta es o no correcta. Para que la tarea concluya debe obtener una tasa de acierto del 100% durante al menos tres series consecutivas en el último bloque de ejecución. Cuando se produce este evento, el sistema devuelve información de que la tarea experimental ha finalizado.

El sistema presenta al sujeto en cada ensayo un patrón de estímulo. Junto con este patrón de estímulo se presentan dos botones, A o B. Según corresponda al patrón en cuestión el sujeto recibe información de si su elección es o no correcta. Con esta información, el sujeto experimental evalúa la validez de su hipótesis de clasificación, y determina como proceder en el siguiente ensayo. A medida que experimenta con la muestra de patrones de estímulo, va refinando sus hipótesis para clasificar correctamente todos y cada uno de los ejemplares.

Cada patrón de estímulo se extrae aleatoriamente en cada serie. La presentación de un patrón de estímulo permite sumar una línea al documento de resultados, que incluye información sobre qué patrón se ha presentado, de qué categoría, y qué medidas se han obtenido de latencia o tiempo de reacción y error. Al término de un bloque, cuando concluye la ejecución de la serie se genera un documento de resultados que incluye información de cada patrón de estímulo, la latencia de respuesta entre el momento de la presentación y la emisión de una respuesta, y el resultado de la ejecución si se trata o no de un error. Este archivo de resultados puede posteriormente guardarse de forma permanente en la base de datos de la aplicación en la vista de Resultados.

Las Figuras 18 y 19 presentan la vista de Resultados. La aplicación no permite regresar en la secuencia de vistas. Presenta una tabla de datos en que cada archivo de datos se identifica por fecha y hora, y nombre y apellido del usuario. La vista cuenta con dos botones en la barra de navegación superior. El botón Borrar Datos le advierte de cómo proceder para borrar datos de la Base de Datos. Para borrar datos debe seleccionarse una fila de la tabla. Esta selección dá lugar a que se active una alerta que solicita al sujeto que confirme si desea o no borrar una línea de datos. El botón Guardar Datos guarda los datos obtenidos en la última ejecución. Si no se guardan los datos del último sujeto que ha ejecutado la tarea, los datos que se han producido durante la ejecución no se guardan. El efecto de pulsar el botón Guardar Datos no se hace visible hasta que no se reejecuta la aplicación, momento en que la aplicación genera los documentos de texto que contiene información de la ejecución de los sujetos cuyos datos se han guardado con anterioridad. En la ventana de Resultados se visualizará entonces una fila de texto que identifica por fecha, hora, nombre y apellidos el documento que se ha generado tras la ejecución del usuario, si los datos se han guardado.

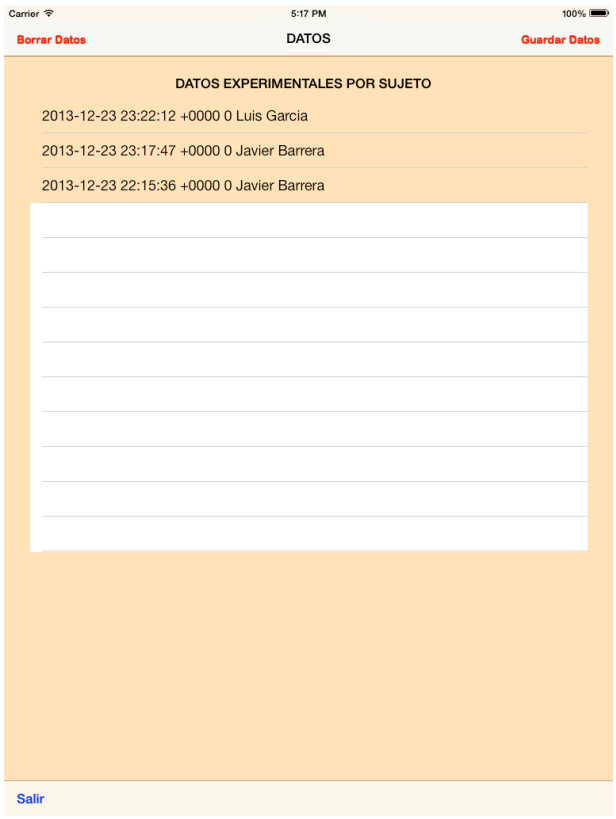


Figura 18: Resultados de tarea

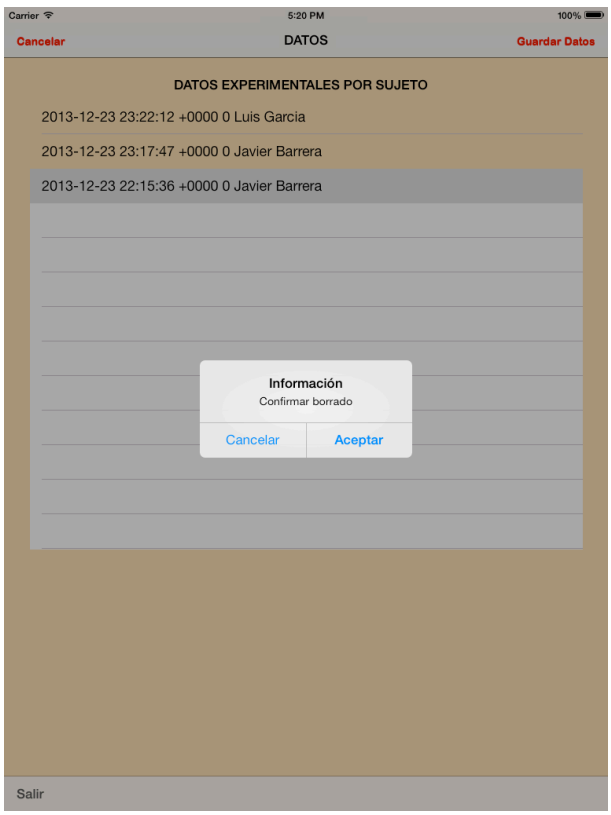


Figura 19: Borrado de resultados

6. Implementación

La implementación de la aplicación requiere la programación de una serie de funcionalidades, en primer término, aquella que se refiere a la conformación e identificación de los patrones de estímulo y aquella que se refiere a las condiciones que deben cumplir tales patrones de estímulo para satisfacer una regla de clasificación. Estas funcionalidades, a su vez, exigen la programación de aquellas funcionalidades que se asocian con la presentación visual de los patrones de estímulo, la selección por parte del usuario de un patrón para la emisión de una respuesta, el registro de respuestas y el registro permanente de datos conductuales y edición de los resultados. Funcionalidades concurrentes que deben programarse con éstas son las que recogen la información que identifica al sujeto y los parámetros de ejecución de la aplicación.

6.1. Consideraciones sobre el diseño. Patrones de diseño.

La fase de implementación responde a la generación del código de programación de la aplicación en Objective-C satisfaciendo una serie de criterios de buenas prácticas con la finalidad de permitir la reutilización de código, la encapsulación, la herencia y el polimorfismo, en suma, aquellos criterios que idealmente representan una aplicación elegante y bien estructurada.

6.1.1. Modelo de Controlador de Vistas.

Una propiedad singular del entorno de programación X-code, tradicional en los desarrollos de la compañía Apple desde sus inicios bajo distintas formas, es el empleo de un patrón de diseño conocido como Modelo de Controlador de Vistas (MVC por sus siglas en inglés), un tipo de diseño optimizado que garantiza la cooperación de los objetos o clases de un modelo con las vistas que permiten al usuario interactuar con la aplicación. Este patrón de diseño se realiza a través de tres subsistemas:

- Modelo: Representa el subsistema de clases que hacen posible la interacción de datos y métodos que sirven al desarrollo de la aplicación.
- Vista: Representa el subsistema de clases que permiten presentar la información que se asocia con las clases y datos del subsistema Modelo, y que garantiza la interacción entre usuario e información.
- Controlador: Representa el subsistema que garantiza el acoplamiento de datos e interficies de usuario, es decir, entre el Modelo y sus Vistas. Interactúa con los otros subsistemas, procesando la información para que sea visualizable y manejable según el flujo de trabajo entre ambos.

Este es el modelo sobre el que se organiza Cocoa y otros entornos de utilidad en el desarrollo de aplicaciones con iOS. La aplicación que se presenta se implementa sobre este diseño.

Existen dos alternativas para implementar el diseño MVC. La alternativa que recomienda Apple es emplear Storyboard, un sistema de vistas preconfiguradas relacionadas que facilitan extraordinariamente la implementación de la aplicación al facilitar la transferencia de control de una vista a otra bajo distintas modalidades. La idea central de este modelo de aplicación es distinguir entre navegación y vista, y enlazar de forma natural vistas y subvistas de una forma integrada sin limitaciones de complejidad funcional. Los objetos o clases que se asocian con estas vistas pueden ser plenamente independientes. La segunda alternativa consiste en programar de manera unitaria clases y vistas de modo que cada vista se asocia con una clase o unas clases en particular. En este segundo caso, el programador debe generar vistas xib y programar explicitamente a través del Interface Builder los contenidos que se presentan en esas vistas. El grado de control que el programador puede tener sobre la aplicación en ambos casos puede ser muy diferente ya que en el primero las interficies están preconstruidas, y en el segundo el programador debe desarrollarlas por su cuenta. En ocasiones, como es el caso de la aplicación que se presenta, las vistas preconstruidas no permiten anidar vistas, lo que ocurre cuando se manejan simultáneamente vistas de tabla (UITableViewController) como subclase de la vista general (UIViewController). En razón de este

control sobre la aplicación, y la versatilidad que le permite al programador, programar una aplicación a medida de sus intereses, se ha optado por esta segunda alternativa.

6.1.2. Delegado.

El Delegado es uno de los patrones de programación que se utilizan para acoplar el subsistema Modelo y los otros subsistemas. De gran sencillez y versatilidad, el Delegado tiene por objeto servir de intermediatrio entre las diversas clases de la aplicación con las que interactúa a través de mensajes que conciernen al flujo de control y la ejecución de tareas. El Delegado se sustenta en un protocolo que establece el inicio de un proceso del que conocen tanto el delegado como la clase que delega. El Delegado es un protocolo especial que facilita la coordinación entre objetos de forma transparente.

6.1.3. Gestión de memoria en dispositivos móviles con iOS.

Coexisten dos modelos de gestión de memoria en aplicaciones con dispositivos móviles bajo iOS. Un modelo empleado hasta las versiones más recientes consiste en una gestión transparente de lo que en otros lenguajes como Java se conoce bajo el término garbage collection, es decir, recolección de recursos de memoria que han dejado de emplearse en la ejecución de una rutina. El programador en este caso recluta recursos de memoria cuando los necesita y ordena su liberación cuando tales recursos ya no son funcionales en el proceso. Una gestión ineficiente de la memoria puede dar lugar a problemas que deriven en un consumo de recursos inasequible para las características del dispositivo lo que origina fallos de funcionamiento de la aplicación. Las versiones más recientes de Xcode incluyen otro modelo de gestión de memoria que se conoce como ARC (Automatic Reference Counting). Este modelo de gestión de memoria representa un avance considerable sobre el modelo anterior. En efecto, en aplicaciones Objective-C tradicionales la gestión de memoria exige retener referencias y liberar objetos para no agotar los recursos de la pila de memoria. Frente a la "recolección de basura" -garbage collection-, el compilador bajo ARC gestiona la memoria de modo que retiene objetos y envía mensajes de liberación de memoria en el código compilado examinando el código fuente. Se trata de una extensión de un tipo de gestión de memoria ya implícita en el modelo anterior bajo un protocolo especial denominado Autorelease Pool, que afecta a todas las clases y métodos de la aplicación. En esta aplicación se ha empleado este sistema de gestión de memoria ofrecido por la versión Xcode 5, la última de que se dispone, difundida mientras se estaba programando su desarrollo.

6.1.4. Gestión de la persistencia.

La aplicación no requiere en esta versión, en la fase de diseño, datos que deban ser persistentes a través de los usuarios de la aplicación. Estos datos iniciales que deberían persistir son los que afectaran a los parámetros de ejecución de la aplicación. Sin embargo, en este estadio no se ha considerado necesario fijar parámetros que requieran mantenimiento en una memoria asociada con el uso de la aplicación. No obstante, la aplicación si requiere el mantenimiento de datos de usuarios, muy en particular, aquellos datos que afectan simultáneamente a distintas clases u objetos de la aplicación durante el proceso de ejecución, y aquellos datos que deben conservarse una vez se cierra la aplicación y que no deben perderse al objeto de permitirle al usuario volver a ejecutar la tarea de aprendizaje de clasificación. Existen distintos mecanismos capaces de satisfacer esta propiedad de persistencia de datos. Se ha hecho uso de las siguientes opciones:

- Parámetros: Los parámetros de ejecución de la aplicación no son permanentes. Se obtienen en el mismo proceso de ejecución por defecto.
- Archivos de información. Se basan en el archivo de textos integrados en la propia aplicación para instruir al sujeto sobre la tarea a realizar. La aplicación también emplea este modelo de conservación de información generando documentos de texto que incluyen información del usuario así como el resultado del uso de la aplicación por parte del usuario.
- NSUserDefaults. Diseñada para conservar información simple de los parámetros de uso de una aplicación, NSUserDefaults permite hacer persistente o conservar información con otros propósitos. Se emplea para conservar información del usuario a través de la aplicación que se usa con distintos fines en distintas partes del programa.
- Core Data. Core Data es el recurso que Apple recomienda para la persistencia de datos. Permite realizar de manera sencillay eficiente todas las funciones relacionadas con la interacción con una estructura de base de datos. Esta base de datos no es una base relacional así que unicamente se permite interactuar con estos datos. En esta aplicación se emplea para conservar el resultado de la ejecución del usuario, de modo que su información es recuperable cada vez que se reinicie la aplicación en el mismo dispositivo. Para que pudiera trasladarse entre dispositivos esta base persistente de datos se debe integrar en el propio código. En esta aplicación se hace uso de esta propiedad pero no se ha integrado en el propio código, por lo que la instalación de la aplicación en otros dispositivos parte de cero. Sólo persisten los datos, salvo borrado, en el dispositivo en donde se han generado estos datos.

- SQLite3. Representa la base datos integrada en iOS. Permite realizar las operaciones propias de cualquier base de datos. Este recurso está implícito en el empleo de Core Data. Eventualmente puede integrarse en la aplicación. Esta opción se encuentra mínimamente desarrollada en la aplicación que se presenta.

En suma, en esta aplicación se emplean para permitir la persistencia de los datos NSUserDefaults, Core Data y SQLite, aunque no se explotan las posibilidades de esta última permitiendo todas las funciones de una base de datos no relacional.

6.2. Implementación del modelo de persistencia de datos.

Buena parte de las decisiones que se han tomado en el diseño de la aplicación resultan de esa compleja relación entre el tiempo disponible para el desarrollo y el manejo técnico de los recursos que permiten las librerías del entorno de programación Xcode, todo ello en concurrencia con el hecho de que buena parte de las necesidades se alumbran en el momento en que se pretende llevar a la práctica el diseño. Aun contando con estas limitaciones en el desarrollo de esta aplicación, se han implementado razonablemente aquellas funcionalidades que satisfacen buena parte de los objetivos inicialmente propuestos.

El objeto principal de la aplicación es la tarea de aprendizaje conceptual. Los resultados generados en el uso de la aplicación son el objetivo crítico ya que cualquier análisis de datos sólo puede basarse en estos resultados. Para hacer posible el análisis se ha prestado especial atención a la conservación de datos. Los datos del sujeto se guardan en un archivo de texto. Los datos de ejecución se guardan en un archivo de texto y en la propia aplicación empleando NSUserDefaults y Core Data. En Core Data se emplea con este fin, un modelo de datos que incluye la información que debe preservarse en el tiempo. La Figura 20 presenta este modelo de datos.

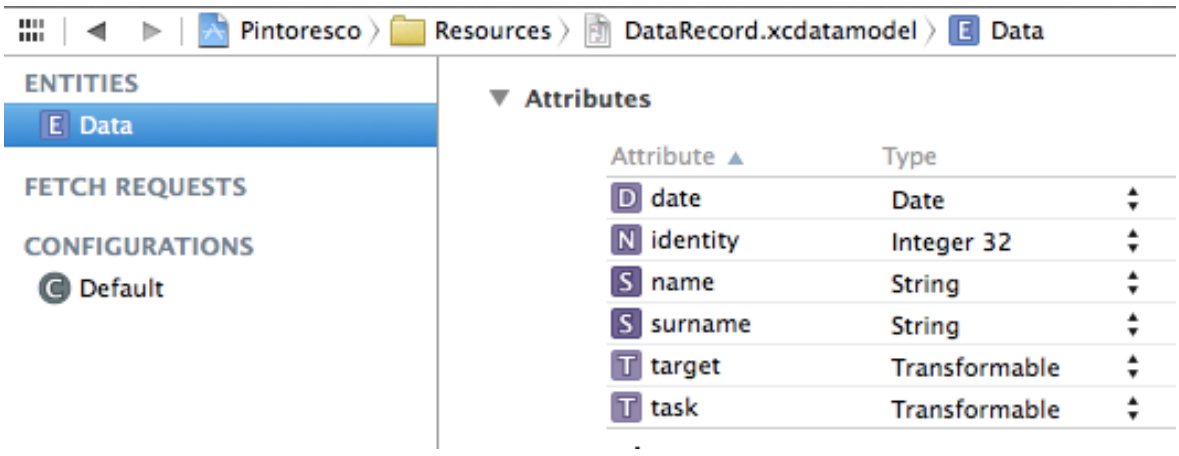


Figura 20: Modelo de datos de registro permanente

El modelo contiene información de la fecha y hora en que se emplea la aplicación, un identificador numérico del usuario, nombre y apellidos, y dos tipos de datos transformables que se expresan en una matriz de datos que contiene toda la información recopilada del usuario -target- y todos los resultados generados por el usuario cuando ejecuta la tarea de aprendizaje de clasificación -task. La aplicación no explota en todos sus extremos las posibilidades que permite este modelo de persistencia de datos en CoreData. Específicamente sólo se conservan bajo CoreData en la base de datos SQLite la fecha y hora de uso de la aplicación, el nombre, y apellidos del usuario, y los resultados del proceso de ejecución de la tarea de aprendizaje de clasificacion en el atributo target que contiene todos los resultados de ejecución. Paralelamente, la misma información puede obtenerse de los archivos de texto que se crean en el proceso de ejecución. Este modelo de datos será objeto de refinamiento en una versión posterior de la aplicación.

6.3. Gestión de clases de persistencia.

La persistencia de datos está relacionada con las clases que se presentan; la clase patrón de estímulo y la clase regla que establece las condiciones que se aplican a una serie aleatoria de patrones para que responda a las restricciones combinatorias de la regla de clasificación en cuestión.

Clase JSSStimulus. Es la clase que permite identificar un patrón de estímulo.

```
@interface JSSStimulus : NSObject {
    NSInteger atIndex;
    char category;
    NSMutableString *pictureCode;
    NSString *test;
    NSInteger nItem;
    double latency;
    NSInteger error;
}
/* */
@property (nonatomic) NSInteger atIndex;
@property (nonatomic)char category;
@property (nonatomic, strong) NSMutableString *pictureCode;
@property (nonatomic) NSString *test;
@property (nonatomic) NSInteger nItem;
@property (nonatomic) double latency;
@property (nonatomic) NSInteger error;

+ (JSSStimulus*) aStimulus;
- (id) initStimulus: (char)i picture: (NSMutableString*)j;
- (NSMutableArray*)arrayOfItems;
- (void)itemToString:(NSMutableArray*)item;
- (void)writeStringToFile:(NSString*)aString;
- (NSString*)readStringFromFile;
@end
```

Clase JSSRules. Es la clase que permite conformar una serie de patrones de estímulo en términos de una regla de clasificación.

```
@interface JSSRules : NSObject {
    NSMutableArray *params;
}

@property (nonatomic, strong) NSMutableArray *params;

+ (JSSRules*)rule;
- (NSMutableArray*)featureRule:(NSMutableArray*)item;
- (NSMutableArray*)equalRule:(NSMutableArray*)item;
- (NSMutableArray*)unEqualRule:(NSMutableArray*)item;
- (NSMutableArray*)exemplarRule:(NSMutableArray*)item;
- (NSMutableArray*)checkFile:(NSMutableArray*)item;
@end
```

Clase JSSLearnView. Es la clase que gestiona una base permanente de datos con CoreData.

```
@interface JSSLearnView : UIViewController<UIAlertViewDelegate> {
    NSMutableArray *itemFile;
    NSMutableArray *learnUser;
    NSMutableArray *learnInfoUser;
    NSMutableArray *identifier;
    NSMutableArray *summary;
    NSUserDefaults *defaults;
}

@property (nonatomic, strong) NSMutableArray *itemFile;
@property (nonatomic, strong) NSMutableArray *learnUser;
@property (nonatomic, strong) NSMutableArray *learnInfoUser;
@property (nonatomic, strong) NSMutableArray *identifier;
@property (nonatomic, strong) NSMutableArray *summary;

@property (strong, nonatomic) NSFetchedResultsController *fetchedResultsController;
@property (strong, nonatomic) NSManagedObjectContext *managedObjectContext;
@property (nonatomic, strong) UITableView *tableView;

@end
```

6.4. Implementación de la clase Delegado.

La clase Delegado implementa los recursos que permiten la gestion de vistas y la permanencia de datos. La clase Delegado implementa UIApplicationDelegate.

Clase JSSAppDelegate. La clase Delegado permite la coordinación de vistas y crear los métodos que permiten la creación de una base de datos permanente con CoreData. A la clase Delegado apelan las clases que permiten la gestión de la base de datos.

```
@interface JSSAppDelegate : NSObject <UIApplicationDelegate>

@property (strong, nonatomic) UIWindow *window;
@property (readonly, strong, nonatomic) NSManagedObjectContext *managedObjectContext;
@property (readonly, strong, nonatomic) NSManagedObjectModel *managedObjectModel;
@property (readonly, strong, nonatomic) NSPersistentStoreCoordinator
        *persistentStoreCoordinator;

- (void)saveContext;
- (NSURL *)applicationDocumentsDirectory;
@end
```

La clase Delegado implementa los métodos de creación de la base de datos.

```
// Returns the managed object model for the application.
// If the model doesn't already exist, it is created from the application's model.
- (NSManagedObjectModel *)managedObjectModel {
    if (managedObjectModel != nil) {
        return managedObjectModel;
    }
    NSURL *modelURL = [[NSBundle mainBundle] URLForResource:@"DataRecord"
        withExtension:@"mom"];
    managedObjectModel = [[NSManagedObjectModel alloc]
initWithContentsOfURL:modelURL];
    return managedObjectModel;
}
// Returns the persistent store coordinator for the application.
// It is created and the application's store added to it.
- (NSPersistentStoreCoordinator *)persistentStoreCoordinator {
    if (persistentStoreCoordinator != nil)
        return persistentStoreCoordinator;

    NSURL *storeURL = [[self applicationDocumentsDirectory]
        URLByAppendingPathComponent:@"DataRecord.sqlite"];

    NSError *error = nil;
    persistentStoreCoordinator = [[NSPersistentStoreCoordinator alloc]
        initWithManagedObjectModel:[self managedObjectModel]];
    if (![persistentStoreCoordinator addPersistentStoreWithType:NSSQLiteStoreType
        configuration:nil URL:storeURL options:nil error:&error]) {

        NSLog(@"Error no identificado %@, %@", error, [error userInfo]);
        abort();
    }
    return persistentStoreCoordinator;
}
```

6.5. Implementación de la gestión de respuestas.

En la vista de tarea se presenta la información en que consiste la tarea de aprendizaje de clasificación. La clase asociada con esta vista permite al usuario autoadministrarse la tarea de

aprendizaje de clasificación y controlar el proceso de ejecución pulsando en los botones virtuales que le quedan a la vista. La clase ItemView gestiona la presentación de los patrones de estímulo y detecta las pulsaciones de pantalla y por su localización asigna los métodos con que cada botón virtual se asocia. Los métodos básicos que permiten gestionar los patrones de la vista y las respuestas táctiles en la pantalla se presentan a continuación:

```
- (void)touchesBegan:(NSSet *)touches withEvent:(UIEvent *)event {
    UITouch *touch = [[event allTouches]anyObject];

    float height = ([[UIScreen mainScreen] bounds].size.height);
    float width = ([[UIScreen mainScreen] bounds].size.width);

    CGRect aRect = CGRectMake(width/3.7, height/1.64, 100.0f, 100.0f);
    CGRect bRect = CGRectMake(width/1.7, height/1.64, 100.0f, 100.0f);
    CGRect sRect = CGRectMake(width/10.6, height/9.48, 105.0f, 40.0f); // Start Button
    CGRect pRect = CGRectMake(width/3.7, height/9.48, 105.0f, 40.0f); // Previous Button
    CGRect nRect = CGRectMake(width/2.0, height/9.48, 105.0f, 40.0f); // Next Button
    CGRect cRect = CGRectMake(width/1.4, height/9.48, 105.0f, 40.0f); // Close Button

    if (imageSelected) {
        if ([self isTouch:touch WithinBoundsOf:&aRect]) {
            finish = [NSDate date];
            NSTimeInterval latency = [finish timeIntervalSinceDate:start];
            [self button:1 time:&latency];
        }
        else if([self isTouch:touch WithinBoundsOf:&bRect]) {
            finish = [NSDate date];
            NSTimeInterval latency = [finish timeIntervalSinceDate:start];
            [self button:2 time:&latency];
        }
    }
    if ([self isTouch:touch WithinBoundsOf:&sRect])
        [self startLearning];
    if ([self isTouch:touch WithinBoundsOf:&pRect])
        [self onPrevious];
    if ([self isTouch:touch WithinBoundsOf:&nRect])
        [self onNext];
    if ([self isTouch:touch WithinBoundsOf:&cRect])
        [self onClose];
}
```

La rutina consulta el método isTouch para identificar la localización de la pulsación táctil. Aunque la rutina es potencialmente útil para cualquier pulsación, sólo ejecuta una acción cuando la pulsación recae en un entorno definido por la etiqueta de un botón virtual. Esta implementación podría substituirse por la actuación de una serie de botones preconfigurados, en lugar de hacerse a través de una pantalla sensible a cualquier pulsación. Sin embargo, la programación de estos métodos es más transparente para el programador y facilita el control de la aplicación.

```
- (BOOL)isTouch:(UITouch *)touch WithinBoundsOf:(CGRect *)rect {
    CGRect _frameRect = *rect;
    CGFloat _imageTop = _frameRect.origin.y;
    CGFloat _imageLeft = _frameRect.origin.x;
    CGFloat _imageRight = _frameRect.size.width+_imageLeft;
    CGFloat _imageBottom = _frameRect.size.height+_imageTop;

    CGPoint _touchPoint = [touch locationInView:self.view];
    // NSLog(@"click in %f %f", _touchPoint.x, _touchPoint.y);

    if ((_touchPoint.x >= _imageLeft) && (_touchPoint.x <= _imageRight) &&
        (_touchPoint.y >= _imageTop) && (_touchPoint.y <= _imageBottom))
        return YES;
    else
        return NO;
}
```

6.6. Implementación de la vista de tarea.

La presentación de estímulos se gestiona en la clase ItemView. El usuario controla la presentación a partir del inicio de la tarea decidiendo sobre qué estímulo de los presentados al margen dar su respuesta de clasificación. Los métodos básicos de la clase ItemView se presentan a continuación:

```
#pragma mark - Methods for select and move images

- (void)showSelectedImage:(UITapGestureRecognizer *)gestureRecognizer {

    start = [NSDate date];
    UIImageView *tappedImageView = (UIImageView *)[gestureRecognizer view];
    itemImage.image = tappedImageView.image;

    imageSelected = YES;
    JSSStimulus *oArray = [[JSSStimulus alloc]init];
    buffer = [[NSMutableArray alloc]init];
    for (NSInteger i=0; i<images.count; i++)
        if ([[images objectAtIndex:i] isEqual:itemImage.image]) {
            oArray = [inItem objectAtIndex:i];
            oArray.atIndex = i;
            // NSLog(@"Item: %02d %02d %02d %c %@", i, i/4, i%4, oArray.category,
         oArray.pictureCode);
            if ((i/4) == block) {
                [buffer addObject:oArray];
            }
        }
    oArray = [buffer objectAtIndex:0];
    /* NSLog(@"Item: %02d Block: %02d Position: %02d Cat: %c Picture: %@",
        oArray.atIndex, oArray.atIndex/4, oArray.atIndex%4, oArray.category, oArray.pictureCode);
    */
}
```

```
- (void)setupThumbnailImageViews {
    for (int i = 0; i < MAXTHUMBNAILS; i++) {
        UIImageView *imageView = [[UIImageView alloc] initWithFrame:
                                  CGRectMake(20.0,166.0+(130.0*i),90.0,90.0)];

        UITapGestureRecognizer *tapGestureRecognizer =
        [[UITapGestureRecognizer alloc] initWithTarget:self
                                        action:@selector(showSelectedImage:)];
        tapGestureRecognizer.numberOfTapsRequired = 1;
        tapGestureRecognizer.numberOfTouchesRequired = 1;
        [imageView addGestureRecognizer:tapGestureRecognizer];

        imageView.userInteractionEnabled = YES;

        itemImage.image = [UIImage imageNamed:[NSString stringWithFormat:@"black.png"]];

        [imageViews addObject:imageView];
        [self.view addSubview:imageView];
    }
}
```

```
- (void)setThumbnailsForPage:(int)aPage {
    for (int i = 0; i < MAXTHUMBNAILS; i++) {
        UIImageView *imageView = (UIImageView *)[imageViews objectAtIndex:i];
        UIImage *image = [self imageForIndex:aPage * MAXTHUMBNAILS + i];
        if (image) {
            imageView.image = image;
            imageView.hidden = NO;
            imageView.tag = i;
            block = aPage;
        }
        else
            imageView.hidden = YES;
    }
}
```

Para permitir la presentación sucesiva de la muestra de patrones de estímulo que constituyen una serie, el usuario dispone de los botones virtuales **Anterior** y **Posterior** permitiéndole avanzar o regresar en estas vistas. Cuando la tarea concluye el sujeto puede **Cerrar** la vista o pasar directamente a la vista de resultados.

6.7. Implementación de la vista de resultados.

Concluida la tarea de aprendizaje de clasificación es posible guardar los datos adquiridos durante la ejecución en la vista de resultados. La clase LearnView se asocia con esta vista. Incluye métodos privados que permiten guardar datos o borrar datos previamente registrados de forma permanente y recuperar en el proceso de ejecución de la aplicación la información disponible en la base de datos.

```
#pragma mark - Private Methods

- (void)insertNewObject:(id)sender {
    JSSAppDelegate *appDelegate = (JSSAppDelegate*)[[UIApplication
sharedApplication]delegate];
    // Entity for table name: (Experimental) Data
    NSEntityDescription *entity = [NSEntityDescription
                                   insertNewObjectForEntityForName:@"Data"
                                   inManagedObjectContext:appDelegate.managedObjectContext];

    defaults = [NSUserDefaults standardUserDefaults];
    NSString *userName = [defaults stringForKey:@"KEYUSERNAME"];
    NSString *surname = [defaults stringForKey:@"KEYSURNAME"];
    if (surname.length > 0) {
        NSString *name = [defaults stringForKey:@"KEYNAME"];
        NSString *surname = [defaults stringForKey:@"KEYSURNAME"];
        [entity setValue:name forKey:@"name"];
        [entity setValue:surname forKey:@"surname"];
    }
    else if (userName.length > 0) {
        [entity setValue:userName forKey:@"name"];
        [entity setValue:@"BBDD" forKey:@"surname"];
    }
NSDate *timeStamp = [NSDate date];
    [entity setValue:timeStamp forKey:@"date"];
    NSMutableArray *taskArray = [NSMutableArray array];
    NSMutableString *oString = [[NSMutableString alloc]init];
    NSString *iString = [[NSMutableString alloc]init];
    JSSStimulus *oArray = [[JSSStimulus alloc]init];
    NSLog(@"Items #: %d", itemFile.count);
    for (NSInteger i = 0; i < itemFile.count; i++) {
        oArray = [itemFile objectAtIndex:i];
        iString = [NSString stringWithFormat:@"\r"];
        [oString appendFormat:@"%@ ", iString];
        iString = [NSString stringWithFormat:@"%02d", oArray.atIndex];
        [oString appendFormat:@"Item: %@ ", iString];
        iString = [NSString stringWithFormat:@"%hhd", oArray.category];
        [oString appendFormat:@"Cat: %@ ", iString];
        [oString appendFormat:@"Estimulo: %@ ", oArray.pictureCode];
        iString = [NSString stringWithFormat:@"%.6f", oArray.latency];
        [oString appendFormat:@"Latencia: %@ ", iString];
        iString = [NSString stringWithFormat:@"%d", oArray.error];
        [oString appendFormat:@"Error: %@", iString];

        [taskArray addObject:oString];
    }

    if ((surname > 0) || (userName > 0)) {
        [entity setValue:taskArray forKey:@"task"];
        NSError *error;
        BOOL isSaved = [appDelegate.managedObjectContext save:&error];
        NSLog(@"Datos guardados: %hhd", isSaved);
    }
}
```

```
-#pragma mark - Properties

- (NSFetchedResultsController *)fetchedResultsController {
    JSSAppDelegate *appDelegate = (JSSAppDelegate*)[[UIApplication
sharedApplication]delegate];
    self.managedObjectContext = appDelegate.managedObjectContext;

    if (_fetchedResultsController != nil)
        return _fetchedResultsController;
    // Create entity object for table: Data
    NSEntityDescription *entity = [NSEntityDescription entityForName:@"Data"
                                   inManagedObjectContext: appDelegate.managedObjectContext];
    NSFetchRequest *fetchRequest = [[NSFetchRequest alloc]init];
    [fetchRequest setEntity:entity];
    // Get all rows in a mutable array
    NSMutableArray *array = [[appDelegate.managedObjectContext
                          executeFetchRequest:fetchRequest error:nil]mutableCopy];

    NSMutableArray *data = [NSMutableArray array];
    NSDateFormatter *dateFormatter = [[NSDateFormatter alloc] init];
    [dateFormatter setDateFormat:@"HH:mm:ss Z"];
    NSString *fileName = [[NSString alloc]init];

for(NSManagedObject *object in array) {
        NSLog(@"Fecha: %@", [object valueForKey:@"date"]);
        NSLog(@"Nombre: %@", [object valueForKey:@"name"]);
        NSLog(@"Apellidos %@", [object valueForKey:@"surname"]);
        NSLog(@"%@", [object valueForKey:@"task"]);

        fileName = [[object valueForKey:@"date"] description];
        [data addObject:[object valueForKey:@"date"]];
        [data addObject:[object valueForKey:@"surname"]];
        [data addObject:[object valueForKey:@"task"]];
        [data addObject:@"\n------------------------------\n"];
    }
    [self writeFile:data title:fileName];

    [fetchRequest setFetchBatchSize:20];
    NSSortDescriptor *sortDescriptor = [[NSSortDescriptor alloc] initWithKey:@"date"
        ascending:NO];
    NSArray *sortDescriptors = @[sortDescriptor];

    [fetchRequest setSortDescriptors:sortDescriptors];

    _fetchedResultsController = [[NSFetchedResultsController alloc]
        initWithFetchRequest:fetchRequest managedObjectContext:[self managedObjectContext]
        sectionNameKeyPath:nil cacheName:nil];

        NSError *error = nil;
        if (![self.fetchedResultsController performFetch:&error]) {
            NSLog(@"Error no identificado %@, %@", error, [error userInfo]);
            abort();
        }
    return _fetchedResultsController;
}
```

6.8. Implementación de la tabla de resultados.

La tabla de resultados es una vista anidada de la vista LearnView. Los métodos que implementan la tabla de resultados heredan sus propiedades del controlador de vista LearnView, pero exigen una gestión independiente. Los métodos básicos se presentan a continuación:

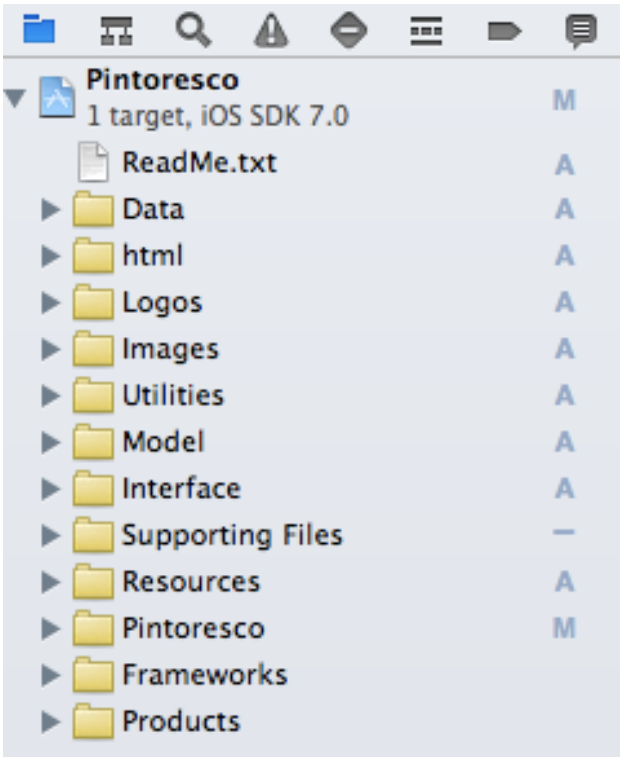
```
#pragma mark - Table view data source

- (void)tableView:(UITableView *)tableView willDisplayCell:(UITableViewCell *)cell
                                forRowAtIndexPath:(NSIndexPath *)indexPath {
    cell.backgroundColor = [UIColor colorWithRed:1 green:0.871 blue:0.678 alpha:1];
}

// Determines the number of sections within this table view
- (NSInteger)numberOfSectionsInTableView:(UITableView *)tableView {
    return [[self.fetchedResultsController sections] count];
}

- (BOOL)tableView:(UITableView *)tableView canEditRowAtIndexPath:(NSIndexPath
*)indexPath {
    return YES;
}

- (BOOL)tableView:(UITableView *)tableView canMoveRowAtIndexPath:(NSIndexPath
*)indexPath {
    return NO;
}

// Determines the number of rows for the argument section number
- (NSInteger)tableView:(UITableView *)tableView numberOfRowsInSection:(NSInteger)section {
    id <NSFetchedResultsSectionInfo> sectionInfo = [self.fetchedResultsController sections]
[section];
    return [sectionInfo numberOfObjects];
}

- (UITableViewCell *)tableView:(UITableView *)tView cellForRowAtIndexPath:(NSIndexPath
*)indexPath {
    UITableViewCell *cell = [tableView dequeueReusableCellWithIdentifier:CellIdentifier];
    if (cell == nil)
        cell = [[UITableViewCell alloc] initWithStyle:UITableViewCellStyleDefault
         reuseIdentifier:CellIdentifier];
    [self configureCell:cell atIndexPath:indexPath];
    return cell;
}
```

Con este conjunto de métodos se asocian los métodos que permiten el borrado permanente de la base de datos. Las acciones que se asocian con los botones Guardar Datos, o Borrar Datos no se hacen efectivas hasta que no se usa de nuevo la aplicación, momento en que se actualiza la información cuya referencia se presenta en la vista de Resultados.

```
- (void)tableView:(UITableView *)tableView commitEditingStyle:
(UITableViewCellEditingStyle)editingStyle
                                forRowAtIndexPath:(NSIndexPath *)indexPath {

    if (editingStyle == UITableViewCellEditingStyleDelete) {

        NSManagedObjectContext *context = [self.fetchedResultsController
managedObjectContext];
        [context deleteObject:[self.fetchedResultsController objectAtIndexPath:indexPath]];

        NSError *error = nil;
        if (![context save:&error]) {
            NSLog(@"Error no identificado %@, %@", error, [error userInfo]);
            abort();
        }
    }
}

- (void)tableView:(UITableView *)tableView moveRowAtIndexPath:(NSIndexPath
*)fromIndexPath
                                toIndexPath:(NSIndexPath *)toIndexPath {
}

// Call the cell touch handler and pass in the cell associated
- (void)tableView:(UITableView *)tView didSelectRowAtIndexPath:(NSIndexPath *)indexPath {
    UITableViewCell *cell = [tableView dequeueReusableCellWithIdentifier:CellIdentifier
forIndexPath:indexPath];

    if (cell == nil)
        cell = [[UITableViewCell alloc]initWithStyle:UITableViewCellStyleDefault
         reuseIdentifier:CellIdentifier];

    indexPathToBeDeleted = indexPath;
    NSLog(@"Borrar fila %d", indexPath.row + 1);
    [self buttonAlertPressed];
}
```

6.9. Organización del código de la aplicación.

Aunque las clases que conforman la aplicación serían, en princpio, accesibles cualquiera sea la ubicación en el programa, se han organizado de un modo coherente organizándolas según las funcionalidades a que sirven. Internamente a la clase se han organizado los métodos según su coherencia con las funciones que permiten ejecutar. La Figura 20 presenta las carpetas en que se organiza el código fuente de la aplicación. La carpeta Data contiene la clase que determina el registro permanente de datos. La carpeta html, el documento que contiene las instrucciones. La carpeta Logos, los iconos que identifican la aplicación y las condiciones experimentales de la tarea de aprendizaje conceptual. En la carpeta Images se han incluido las imágenes de los patrones de estímulo empleados en la tarea de aprendizaje conceptual.



La carpeta Utilities contiene la clase de métodos que representan funciones de utilidad transversales a distintas clases u objetos. La carpeta Model incluye las clases de estímulos que se emplean en el desarrollo de la tarea experimental. La carpeta Interface incluye las clases que se asocian a las interficies o vistas de usuario que se han programado explicitamente y asociado mediante el Interface Builder. La carpeta **Pintoresco** incluye la clase Delegate. Frameworks incluye las librerías que se emplean en la aplicación. Finalmente, Resources, Supporting Files y Products son creados por el compilador XCode excepto por

Figura 21: Carpetas de la aplicación

lo que se refiere al modelo de datos en CoreData, modelo que se requiere para la gestión de permanencia de datos.

7. Evaluación de la aplicación. Pruebas de usuario

Concluida la aplicación y cuando ya se cuenta con un producto final es el momento de verificar el funcionamiento correcto de la aplicación, su adecuación a los objetivos y su ajuste a las necesidades y expectativas de los usuarios finales. Con este propósito pueden identificarse tres contextos en la evaluación de la aplicación, (1) verificación de su funcionamiento, (2) evaluación de la aplicación, y (3) modificación y mejora de funcionalidades. Este proceso debe ser iterativo, en el sentido de que comprobar la validez de la aplicación, ajustarla a las necesidades de los usuarios o mejorar sus prestaciones no puede hacerse sin un testeado permanente a través de los usuarios de la aplicación. Resultaría quimérico pensar que un proceso de este carácter pudiera desarrollarse cuando es justamente ahora cuando se cuenta con una aplicación utilizable. De este modo, parte de la evaluación es especulativa, una inferencia de los usos que realiza el propio desarrollador con la aplicación, o resulta de un análisis racional que compara los objetivos propuestos y los objetivos efectivamente desarrollados.

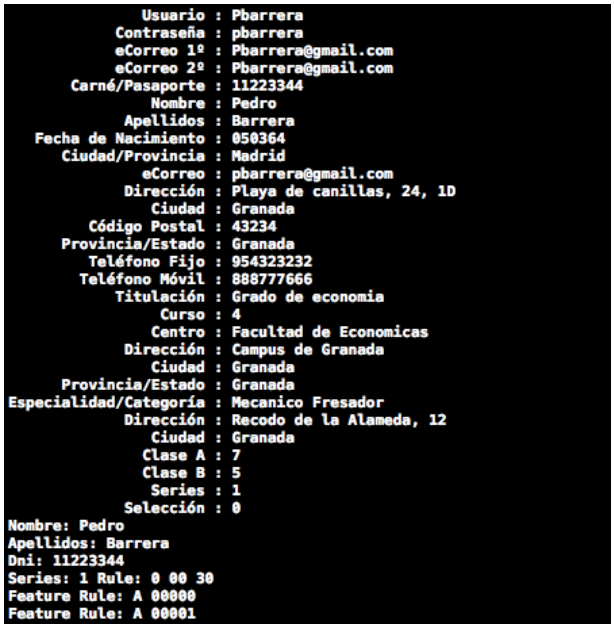


Figura 22: Salida de datos en consola

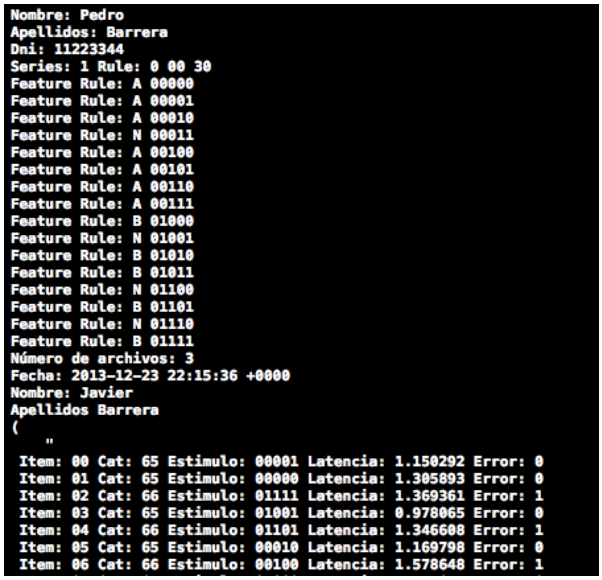


Figura 23: Salida de datos en consola

Las pruebas del prototipo en Java desarrolladas sobre sujetos reales nos permitieron anticipar algunas de las dificultades que luego se presentarían en el desarrollo. Sin ese estudio piloto hubiera sido técnicamente muy dificil, en el plazo propuesto, desarrollar una aplicación como la que se presenta. Además, la aplicación Java permite comprobar el funcionamiento de los usuarios y calibrar la ejecución de los usuarios de una aplicación móvil en un entorno más amigable. Las respuestas del sujeto en la aplicación Java se ejecutaban a través del teclado, en la aplicación que ahora se presenta, las respuestas se efectúan de forma táctil, mucho más eficiente y económica. La aplicación que se presenta en este proyecto, no ha sido, cómo tal, objeto de evaluación aunque el prototipo en Java nos permitiera comprobar que la aplicación era accesible, comprensible, y útil para los propósitos propuestos y a los usuarios finales. La aplicación piloto permitió corregir algunas decisiones de diseño técnico, también como resultado de la distinta versatilidad que ofrecen los distintos lenguajes, el lenguaje Java y el lenguaje Objective-C y en relación con el dispositivo, singularmente, el hecho de que el dispositivo permita respuestas táctiles, frente a las respuestas en teclado de la aplicación Java al no contar con una pantalla táctil.

8.1. Verificación del funcionamiento.

La primera fase de evaluación de la aplicación es la verificación de su funcionamiento. Consiste en que la aplicación pueda garantizar que satisface las funcionalidades previstas. Se ha revisado la ejecución y asegurado que funciona correctamente. Como documentos adjuntos a este proyecto se incluye una serie de archivos creados a partir de la ejecución de la aplicación en tres oportunidades. Esta comprobación se ha hecho sobre el simulador que incorpora Xcode, prestando atención al

funcionamiento anómalo de la aplicación que ha determinado ciertas correcciones en la aplicación final. Por fortuna la gestión de memoria que implementan los compiladores mas actuales Xcode, Automatic Reference Counting, facilita extraordinariamente el funcionamiento y evita los problemas que se asocian a lo que en el pasado era habitual bajo la tecnología garbage collection. El simulador es suficientemente eficiente; los problemas que presenta la aplicación final derivan más bien de que no se haya diseñado por completo el registro permanente de datos bajo los recursos que proporciona Core Data, y se haya empleado una estrategia mixta de salvado de datos en formato de sólo texto, simultáneamente a un salvado permanente de datos con NSUsersDefault y Core Data.

8.2. Evaluación de la aplicación.

Para poder determinar si la aplicación satisface los estándares mínimos de usabilidad y si satisface las necesidades y objetivos para las que se ha diseñado, deberíamos contar con experiencia suficiente de su uso por usuarios reales en este estadio, una vez se ha concluido su diseño e implementación. Obviamente, el tiempo es un factor determinante y compromete ese propósito. Específicamente sería importante contar con una muestra de usuarios que hubieran completado con éxito la tarea, y el análisis de datos mostraría si la aplicación satisface los objetivos y se adecúa a las condiciones y necesidades de los usuarios que se prestaran a utilizarla. Sin embargo, esta limitación no implica que no pueda evaluarse la aplicación en su estado actual. En distintas oportunidades en esta memoria se ha indicado que algunas de las funcionalidades que sería útil desarrollar se postergan a una versión posterior. La autenticación de los usuarios, la creación de una base de datos diferenciada según perfiles de usuario, la incorporación de una tarea de transferencia de aprendizaje que se ha indicado en esta memoria contemplará una tarea de búsqueda visual consecutiva al aprendizaje conceptual, son algunas de las funcionalidades previstas en un desarrollo posterior de la aplicación que ahora no están más que incoadas en alguna rutina que no se emplea efectivamente. Tomando un criterio más estricto, el éxito de la aplicación debe evaluarse en el actual contexto identificando ciertos problemas o anomalías que deberán subsanarse, problemas que se han detectado en un uso relativamente intensivo por parte de quien suscribe, el propio desarrollador.

8.4. Detección de problemas.

El primer conjunto de deficiencias o ineficiencias de la aplicación se relacionan con la propia estructura de datos de usuario. Sería, por ejemplo, útil que la información recogida no se perdiera cuando el usuario regresa en la aplicación a otras vistas. No sería particularmente útil desarrollar código para que la información recogida en cada campo de texto reapareciera, tanto como que esa

información fuera permanente desde el mismo momento en que se rellena el campo. La base de datos del usuario debe crearse desde el primer momento con ese propósito.

Un segundo conjunto de ineficiencias de la aplicación se refieren a la propia versatilidad de los parámetros de la aplicación. En la actual versión es posible modificar a voluntad el número de series pero no es posible cambiar el número de ejemplares por categoría, que son siempre 7/5 indistintamente para las categorías A/B. Es posible encontrar otras series de ejemplares capaces de satisfacer las condiciones de las cuatro reglas de clasificación que se emplean. Sin embargo, el hecho de que se presentan 12 de los 16 posibles patrones de estímulo asegura que la mayor parte de los ejemplares se repiten entre reglas, lo que contribuye al objetivo propuesto de que el aprendizaje se evalúa sobre las propiedades lógicas de la partición, en lugar de que se evalúe sobre características específicas de los estímulos, lo que se ha llamado en la presentación del proyecto, las propiedades no accidentales de los objetos formados a partir de los cuatro geones utiizados en este caso.

Un tercer conjunto de ineficiencias se refieren al proceso de navegación de la aplicación. En particular, debemos revisar los botones configurados en cada vista, y específicamente aquellos que permiten la navegación entre vistas. Pueden también mejorarse la presencia de los elementos de las vistas para que resulten más atractivas. La tarea de aprendizaje de clasificación puede incluir iconos durante el desarrollo, lo que puede hacer más atractivo su seguimiento.

Un cuarto conjunto de ineficiencias o deficiencias técnicas existen en la aplicación que son más relevantes para el propósito que se quiere satisfacer. La mas importante de estas deficiencias se refiere a la edición de la información de la base de datos. En tanto los resultados de la ejecución de la tarea de aprendizaje conceptual se han guardado una única vez, se imprimen multiples veces según el número de líneas de datos. Este resultado es sumamente incómodo y disfuncional y es resultado del modo en que se procede a la edición de datos. La solución a este problema que dificulta y entorpece el análisis pasa por separar el modelo de datos bajo distintas entidades, o pasa por filtrar los resultados para que no se repitan en la edición si ya han sido previamente editados. Si no se han hecho progresos en esta dirección es por las limitaciones impuestas por el plazo de entrega. La primera solución parece más simple y obvia.

Por ultimo, sin duda, la aplicación puede beneficiarse del propio uso, con una versión más sólida e integrada de los recursos técnicos disponibles. Queden estos retos para una versión más y mejor desarrollada. Si puede decirse claramente cuando el proyecto comenzó a cobrar forma, no puede decirse lo propio de cuando podrá darse por concluido. Como sucede con todas las tareas que se hacen por propio interés y pasión, esta tarea ha de prolongarse hasta el punto de que pueda ser efectivamente una herramienta útil mas allá de su validez académica en este contexto.

8.5. Características destacables.

La aplicación que se presenta es una muestra representativa de las posibilidades que ofrece el empleo de las tecnologías móviles en el diseño de aplicaciones orientadas a la evaluación conductual, al análisis de procesos cognitivos y a la rehabilitación neurocognitiva. Entre los rasgos positivos del producto final pueden destacarse,

- La sencillez de uso de la aplicación.
- La claridad de la información que se presenta.
- La naturaleza intuitiva y precisa de la tarea que debe ejecutar el sujeto.
- Las facilidades de navegación y uso, y el entorno gráfico contribuyen a un entorno amigable.
- La organización de los contenidos bien estructurada y de fácil acceso.

8.6. Resultado de la aplicación.

El producto final puede juzgarse de la presentación que se hace de la herramienta en la grabación que acompaña esta memoria y en el código fuente que se adjunta preparado para poner en marcha Pintoresco y verificar lo que se ha explicado a lo largo de esta memoria. Puede asegurarse a partir de esta información que esta aplicación ha respondido a las expectativas iniciales del proyecto si se toma en cuenta el plazo de tiempo para llevarla a cabo y los procesos involucrados en el proceso de programación.

8.7. Perspectivas.

El análisis de eficiencia de la aplicación ya se pueden apreciar algunos de los objetivos a lograr en una próxima versión de lo que hace muy poco tiempo era sólo un proyecto. Existen otros retos que junto con los expuestos podrían destacarse, entre ellos, el uso mas extenso e intenso de las herramientas de permanencia de datos, la extensión de la aplicación para incorporar otras tareas con el mismo propósito, el desarrollo de una base de datos que permita recoger información del usuario susceptible de relacionarse con su ejecución en un típico desarrollo de minería de datos, la implementación de los estímulo empleando Open GL ES, y la corrección de algunas ineficiencias de uso ya comentadas. La implementación de un sistema de construcción gráfica de los estímulos, es uno de los retos que no pudieron lograrse dado la complejidad de la herramienta y el tiempo disponible para su aprendizaje y desarrollo. Es este el precio a pagar para concluir con éxito a tiempo una aplicación que se presentó tan ambiciosa al principio. Nada de esa ambición se ha perdido en el trayecto, pero la propia complejidad del lenguaje y de los recursos que se pueden

emplear en el desarrollo han templado el carácter, y constituyen un buen pretexto para seguir en este camino.

8. Conclusiones

El dominio de las destrezas que se requerían para llevar a cabo con éxito el proyecto se han ido obteniendo a medida que se desarrollaba la aplicación. Al inicio no tenía quien suscribe ningún conocimiento específico del compilador Xcode ni del lenguaje de programación Objective-c. Si se descuenta la formación previa en programación orientada a objetos, en Java y en c++, se partía de cero. A pesar de ello quien suscribe se propuso desarrollar una aplicación que iniciara un largo camino: desarrollar aplicaciones móviles en iPad orientadas a la evaluación conductual y orientadas al desarrollo de instrumentos que permitieran replicar fenómenos psicológicos básicos, susceptibles de emplearse como criterios e indicios del procesamiento cognitivo. Se proyectaba entonces cubrir ciertos objetivos, esencialmente, el de contar con una aplicación que nos permitiera estudiar los procesos de aprendizaje conceptual. Llegados a este punto, es posible confirmar que buena parte de las expectativas se han cumplido, y el desarrollo que se presenta puede, desde la perspectiva de quien suscribe considerarse un éxito, con independencia de que se hayan abierto más oportunidades y se requiera un desarrollo mas sofisticado para mejorar aquí y allá rasgos de la aplicación, extender la aplicación integrando una tarea de búsqueda visual, o facilitar el proceso de análisis de datos.

El objetivo principal era diseñar e implementar una aplicación iOS sobre un dispositivo móvil iPad que replicara una tarea de aprendizaje conceptual con una novedad inédita: emplear reglas de clasificación distintas, con distintas propiedades formales, sobre un conjunto de objetos bien definido de modo que la serie de objetos a clasificar tuviera el menor impacto sobre la tarea misma, al variar aleatoriamente entre sujetos la serie de patrones sometidos a evaluación. En efecto, lo que es común a todos los sujetos que pasan una cierta condición experimental es el hecho de que a todas las series de patrones de una cierta condición les corresponde la misma forma o estructura lógica. Lo que se está evaluando es el impacto de la forma lógica en el aprendizaje, no las características específicas que presentan los patrones de estímulo, cuyo impacto se neutraliza de este modo. **Pintoresco** ha supuesto un salto adelante en esta dirección, un paso adelante en reconocer el impacto de cómo se combinan bajo ciertas reglas de composición una serie de rasgos o propiedades de objeto. La experiencia ha valido la pena. Además de haberme permitido manejar un nuevo entorno de programación, de experimentar el sufrimiento y la satisfacción de resolver cada día un problema sobrevenido o no anticipado por completo, el proyecto me ha permitido desarrollar un producto informático más allá de que requiera mejoras y verificación en un entorno real. El proyecto me ha proporcionado una visión mas precisa de las dificultades que entraña el desarrollo de una

aplicación de este carácter y la necesidad de ajustar el producto a las necesidades de un usuario humano.

Comenzamos nuestro trabajo planificando las tareas a desarrollar de acuerdo con una distribución temporal que tenía como horizonte un cuatrimestre. Como se caminaba en este horizonte casi a ciegas algunos de los objetivos previstos se han visto postergados para alumbrar otros no previamente contemplados. Parte del éxito de este desarrollo ha nacido tanto de la necesidad de seguir el plan, como de la necesidad de resolver problemas con una representación de que forma debería tener la aplicación final. La aplicación piloto que se desarrolló en Java fué util para tener una representación más fidedigna de la posibilidad de ajustarse a las necesidades de un usuario en cuanto sujeto del proceso de testado de la aplicación. El usuario es el eje en torno al que gira el desarrollo de una aplicación. En la aplicación se ha preservado por completo su libertad, de tal modo que en la práctica ninguna de sus acciones puede afectar al proceso de ejecución de la aplicación, sino más bien a la utilidad final de los datos obtenidos.

El desarrollo de la aplicación nos ha permitido conocer la potencia, elegancia y eficiencia del lenguaje Objetive-c. Esta potencia, esta elegancia y esta eficiencia son deudoras de las posibilidades que representa para el programador las librerías del entorno Xcode. En el transcurso del desarrollo se ha puesto en circulación la versión Xcode 5. Se comenzó con iOS 6, y sobre la marcha se trasladó la aplicación a iOS7. Cualquier aplicación exige una permanente puesta al día, al mismo tiempo que se acometen nuevos desarrollos y se incorporan nuevos recursos.

Rapidamente se puso de manifiesto que no sería posible implementar la tarea de búsqueda visual para estudiar los fenómenos de superioridad de objeto y de migración que resultan tras un proceso de aprendizaje previo de una partición conceptual. Nunca prometimos por prudencia este resultado. En este sentido, la aplicación final queda a la vez cerca y lejos de ese propósito. La sencillez del diseño funcional oculta la complejidad de la tarea que se acometió, y sin embargo, esa sencillez no debe ocultar que el estado actual de la aplicación representa un éxito. Está muy próximo el futuro, ese momento que comienza precisamente con la entrega de este trabajo final, el objetivo de desarrollar una aplicación que finalmente merezca encontrarse entre los recursos disponibles en la Apple Store. Permítase a quien suscribe estar a la vez orgulloso del resultado e insatisfecho por todo lo que queda por hacer.

Glosario de términos

Apple	Multinacional norteamericana que ha desarrollado el sistema operativo y aplicaciones propietarias para la gama de equipos Macintosh
Tablet/tableta	Ordenador portátil integrado en una pantalla gráfica de control táctil.
iPad	Dispositivo multimedia táctil diseñado, producido y comercializado por Apple
iOS	Sistema operativo propietario de Apple utilizado en sus dispositivos móviles.
Objective-C	Lenguaje de programación orientado a objetos que integra como subclase el lenguaje c++ orientado a objetos.
Cocoa-Touch	Interficie de programación de aplicaciones (API) que proporciona una capa de abstracción de alto nivel para la creación de aplicaciones móviles con iOS.
X-Code	Entorno de desarrollo gráfico (IDE) creado por Apple, destinado al diseño de aplicaciones para dispositivos móviles de su creación.
Interface Builder	Componente de X-Code para diseñar las interficies de una aplicación Apple.
MVC	Arquitectura de controladores basados en vistas de usuario. Patrón de arquitectura de programación que distingue entre los datos de la interficie de usuario y la lógica de la aplicación, tratando cada componente como piezas diferentes de un programa.
Delegate/Delegado	Patrón de programación que permite crear objetos que realizan transaciones entre clases minimizando los problemas de sincronización y control.
Modal View Controller	Controlador que permite interumpir el flujo de control normal para realizar una tarea concreta y bien definida que presenta una nueva ventana.
Push View Controller	Controlador que permite pasar el flujo de control normal a una ventana de ejecución.
ARC	(Automatic Reference Counting). Sistema de gestión de liberación automática de los recursos de memoria gestionada por el compilador X-Code. En aplicaciones Objective-C tradicionales la gestión de memoria exige retener referencias y liberar objetos para no agotar los recursos de la pila de memoria. Frente a la "recolección de basura" -garbage collection-, el compilador bajo ARC gestiona la memoria de modo que retiene objetos y envía mensajes de liberación de memoria en el código compilado examinando el código fuente.
RBC	Teoría de reconocimiento visual de objetos propuesto por I. Biederman (1987).

Open Gl ES	Variante de la API gráfica Open GL para dispositivos móviles que permite generar patrones gráficos en dos y tres dimensiones.
Core Graphics	API de Quarz 2D que ofrece funciones para la creación y renderización de gráficos en dos dimensiones.
DCU	Diseño centrado en el usuario. Modelo de programación de aplicaciones basada en la creación de productos útiles a las personas garantizando la satisfacción de los usuarios.
Guías para el desarrollo de interficies hombre-máquina	(Human Interface Guidelines). Documentación destinada a desarrolladores que ofrece recomendaciones específicas para mejorar la experiencia del usuario.
Garbage collector	Mecanismo incluido en distintos lenguajes de programación destinados a facilitar la gestión de memoria y liberar automáticamente los recursos caducados.

Bibliografía

Allan, A. (2010). Learning iPhone Programming. O'Reilly Media.

Apple Inc. (2013). iOS Developer Library (en línea). https://developer.apple.com/

Apple Inc. (2013). iOS Human Interface Guidelines (en línea). https://developer.apple.com/library/ios/documentation/userexperience/conceptual/mobilehig/

ARC (2007-2013) The Clang Team. Created using Sphinx 1.1.3. http://clang.llvm.org/docs/AutomaticReferenceCounting.html

Biederman, I. (1987). Recognition-by-components: A theory of human image understanding. Psychological Review, 94, 2, 115-147.

Biederman, I. Mezzanotte, R.J. y Rabinowitz, J.C. (1982). Scene perception: Detecting and judging objects undergoing relational violations. Cognitive Psychology, 14, 143-177.

Borraz Morón, D. (2012). Copy2Cloud. Trabajo Final de Carrera. FUOC: Fundació per a la Universitat Oberta de Catalunya.

Buck, E. M., Yacktman, D.A. (2010). Cocoa Design Patterns. Pearson Education.

Daniel, S. F. (2011). XCode 4. iOS Development. Beginner's Guide. Packt Publishing: Birmingham - Mubai.

Goodman, D. (2010). Learning the iOS 4 SDK for JavaScript Programmers. O'Reilly: Sebastopol: California.

Kochan, S.G. (2011). Programming in objective-C (3th edition). Pearson Education.

López Hernández, F. (2013). Objective-C. Curso práctico para Programadores Mac OS X, iPhone y iPad. Madrid: RC Libros.

Mark, D, Nutting, J, La Marche, J. (2011) Beginning iPhone 4 Development. Apress.

Muriel Garreta, D., y Mor Pera, E. (2006). Diseño centrado en el usuario. FUOC: Fundació per a la Universitat Oberta de Catalunya.

Rodriguez-Díez, M.G. (2013). Trabajo Final de Carrera. FUOC: Fundació per a la Universitat Oberta de Catalunya.

Treisman, A. & Souther, J. (1986). Illusory words: The roles of attention and of top-down constraints in conjoining letters to form words. Journal of Experimental Psychology: Human Percepcion and Performance, 12, 1, 3-17.

Wenk, R. (2011). XCode 4. Developer Reference. Wiley Publishing: Indianapaplis, Indiana.

Licencia de uso

Anexos

Pintoresco v1.0

Esta aplicación desarrolla una tarea experimental que evalúa la dificultad relativa de un usuario para aprender a clasificar un conjunto de patrones de estímulo en dos categorías. Se trata, pues, de una tarea de aprendizaje de clasificación binaria, en la que un conjunto de patrones se distribuye entre dos clases. Se han definido cuatro reglas posibles, la condición de rasgo, la condición idéntica, la condición desigual y la condición ejemplar.

Según la regla que aprende, el sujeto aplicará implicitamente el conocimiento adquirido para una tarea de búsqueda visual en un paradigma de migración que ilustra el efecto de superioridad de objeto. Esta tarea de búsqueda visual no se ha integrado en esta aplicación.

La interfaz de usuario se ha programado explícitamente a traves de nib/xib's. Estas interficies de usuario obtienen, en primer término, los datos que identifican al usuario de la aplicación, que se registra mediante nombre de usuario y contraseña. Esta información se remite al usuario a una dirección de correo. La información obtenida se complementa con la que se obtiene del usuario relativa a su identidad, domicilio, formación y profesión. Estos datos se obtienen consecutivamente a travésde las interficies MainView, DataView, e InfoView.

Para usuarios ya registrados la aplicación puede ejecutarse desde la ventana principal -MainView- a la ventana de parámetros donde se fijan los parámetros de la ejecución. Aunque sería posible con los mismos métodos variar el número de ejemplares que satisface cada categoría de la clasificación binaria, se ha definido por defecto que las categorías A/B tengan indistintamente 7/5 ejemplares por categoría. La aplicación obtiene aleatoriamente una muestra de 7+5 patrones de estímulo que satisface las condiciones de la regla escogida.

La selección de parámetros se realiza en la interficie TableView. El sujeto debe determinar el número de series que se le presentarán en el proceso de aprendizaje. Al sujeto se le presentará el número de series x el número de patrones de las clases en que se distribuyen en la clasificación binaria de acuerdo con una cierta condición.

La interficie ScrollView con una barra desplegable presenta en código html las instrucciones que se aplican a la tarea en cuestión.

La interficie ItemView presenta una serie de botones virtuales que permiten obtener una serie de patrones de estímulo a razón de 4 patrones por ventana de ejecución. La interficie ItemView ejecuta la tarea de aprendizaje. Para iniciar la presentación de una serie, el sujeto debe pulsar con una pulsación táctil o con clic de ratón en el botón **Inicio** de la tarea. Tras la pulsación, puede observar a la izquierda de la ventana, un conjunto de 4 patrones de estímulo. Con pulsación táctil o de ratón, el sujeto escoge el estímulo que desea clasificar como miembro de la clase A o miembro de la clase B, botones virtuales que aparecen al pie del patrón de estímulo. Los botones virtuales **Anterior** y **Posterior** le permiten ir hacia atrás en la serie, o ir hacia adelante. No importa la secuencia de ejecución; en cualquier caso deberá responder a todos y cada uno de los patrones de estímulo que se presentan en la serie. Se reinicia la serie cada vez que el sujeto pulse en el botón virtual **Inicio**, borrándose datos previos. Concluida la serie experimental, el sujeto pulsa el botón **Cerrar**, botón que se puede pulsar en cualquier momento.

A la interficie LearnView se accede a partir de la interficie ItemView. En esta interficie se presentan una referencia al archivo de resultados de los sujetos que han ejecutado la tarea hasta el momento.

Especificaciones técnicas

Esta aplicación ilustra el empleo de una serie de herramientas técnicas que permiten el registro permanente de los datos obtenidos en la ejecución de la tarea experimental por parte del usuario. Se emplean herramientas NSUserDefaults y CoreData en la gestión de la información obtenida del usuario, a través de un modelo de datos.

* Funciones de utilidad de múltiple usabilidad.
* Gestión de vistas de usuario. Campos de textos programados.
* Gestión de campos de texto con reinicialización y ayuda en tiempo real.
* Gestión de correo alerta de envío al usuario.
* Gestión de imágenes PickerView.

* Gestión de clases+categorías.
* Funciones aleatorias de construcción de series de patrones de estímulo.
* Función de aleatorización de presentacioes de series de patrones.
* Verificación y validación de condiciones experimentales.

* Gestión de vistas ViewController anidadas.
* Gestión de vistas de documentos WebView con barra de visualización.

* Gestión de imágenes de album.
* Ejecución táctil y gestión por reconocimiento gestual.
* Gestión de alertas automáticas y opcionales.

* Registro de datos permanentes con NSUserDefaults y Core Data.
* Gestión de instanciación de objetos NSFetchedResultsController.
* Gestión de vistas anidadas y actualización de cambios de vistas.
* Gestión de edición, borrado y actualización de la base de datos.
* Inicialización de la Base de Datos.
* Gestión de impresión de datos.
* Generación de archivos sólo texto de datos de usuario.
* Generación de archivos sólo texto de datos de ejecución, latencia y error.

LOS ARCHIVOS DE DATOS SE GUARDAN POR DEFECTO EN LA DIRECCION
NSString *filePath = @"/Users/Load/";
DEBE CREAR LA CARPETA "LOAD" EN LA DIRECCION /Users;
LA APLICACIÓN NO INCLUYE NINGUNA BASE DE DATOS PREVIA

Requisitos de compilación

iOS 7 SDK o posterior

Requisitos de ejecución

iOS 7 SDK o posterior
iPad, iPadRetina

Proceso de ejecución

La tarea incluye 7 interficies de usuario. Cada ventana permite ir hacia atrás pulsando el icono que identifica la aplicación en la barra de navegación, o hacia adelante pulsando en el botón en la barra al pie de la vista que permite pasar a la siguiente vista. A partir de la ejecución de la tarea experimental no es posible volver atrás.

La tarea experimental está representada por la vista ItemView. El sujeto selecciona un patrón de estímulo que clasifica según su convicción en la clase A o B. El objetivo de esta tarea es que el sujeto llegue a deducir la regla de clasificación que le permita una ejecución perfecta en tres series consecutivas. En teoría la tarea no concluye hasta que el sujeto domina por entero la clasificación de modo que, en caso contrario, el sujeto debe repetir la serie tantas veces como sea preciso para controlar los criterios que permiten hacer la clasificación.

El registro permanente de datos está representado internamente por el uso de parámetros NSUserDefaults, y por el uso de CoreData. La vista que cierra la aplicación, la vista LearnView guarda los datos de los usuarios de forma permanente, datos que se añaden a la vista a medida que se ejecuta la aplicación. El usuario no tiene libertad para la incorporación de nuevos datos, pero puede eliminar datos registrados sin más que pulsar en la celdilla de la tabla que contiene los datos de un sujeto identificado por su DNI/Pasaporte y por su nombre. Pulsando en una celdilla de la tabla que contiene los datos del usuario se activa una alerta que le permite eliminar de forma permanente los datos registrados. Esta opción requiere prudencia para no eliminar el trabajo realizado por el usuario.

Los datos obtenidos en la aplicación se imprimen en formato de sólo texto para su tratamiento posterior. Eventualmente, estos datos podrían enviarse a un servidor, mediante un correo electrónico o apelando a una aplicación Cloud.

Instrucciones de ejecución (Documento html)

<body>

Los dibujos que vas a ver son de objetos muy útiles. Representan dos tipos de encendedores o mecheros por émbolo de presión para hacer fuego. Hay encendedores de dos tipos. Tu tarea consiste en aprender a identificar estos dos tipos, pulsando el botón A o el botón B.

Al principio no sabrás cuál es cuál. Sin embargo, tendrás que dar una respuesta. Después de responder te ayudaremos dándote información de si has

acertado o no. Prestando atención a los detalles de cada dibujo, y pensando encontrarás la manera de acertar.

Con suficiente tiempo aprenderás a clasificar todos los ejemplos. No importa lo que tardes. Lo que importa es aprender. La tarea concluye cuando has aprendido a clasificar todos los ejemplos bien. Te repetiremos la serie de ejemplos para que descubras cómo se clasifican.

</body>
</html>

Listado de clases

Clases de Utilidades

JSSUtils {h,m}. Define al clase que contiene las rutinas que se emplean en la activación de las clases que permiten el desarrollo de la aplicación.

JSSArrayShuffling {h,m}. Identifica la clase que permiten la aleatorización de una serie de patrones de estímulo.

Clases del Modelo

JSSStimulus {h,m}. Define la clase de estímulo que permiten identificar un patrón por relación a sus propiedades.

JSSRules {h,m}. Define la clase de condiciones experimentales en que puede entrar una serie de patrones de estímulo.

Clases de Interficies de Usuario

JSSMainView {h,m,xib}. Presenta la vista de inicio de la aplicación a partir de la cual se navega en la aplicación. Recoge la información del usuario registrado identificado por el nombre de usuario y

su contraseña. Esta información no se trata en una Base de Datos independiente que verifique si el usuario está o no autorizado. Esta funcionalidad carece de sentido en una aplicación que permite una ejecución sin restricciones.

JSSDataView {h,m,xib}. Presenta la vista de recogida de datos que permiten identificar al usuario de la aplicación.

JSSInfoView {h,m,xib}. Presenta la vista de recogida de datos característicos del usuario de la aplicación, incluida la fotografía del usuario.

JSSTableView {h,m,xib}. Presenta la vista de recogida de opciones de usuario que identifican los parámetros de la aplicación que afectan a la ejecución de la tarea experimental.

JSSScrollView {h,m,xib}. Presenta la vista de las instrucciones para la ejecución de la tarea experimental que se propone.

JSSItemView {h,m,xib}. Presenta la vista que permite activar la serie de patrones de estímulo que son la fuente del proceso de aprendizaje de clasificación. En la ejecución se obtienen datos de latencia de respuesta en milisegundos y errores, cuyo cálculo permite resolver si el usuario ha aprendido la clasificación. Los datos se registran y se guardan de forma permanente.

JSSLearnView {h,m,xib}. Presenta la vista de los datos que se han registrado de forma permanente en la base de datos, que pueden conservarse o eliminarse a decisión del usuario.

Recursos

Carpeta Logos. La carpeta que contiene los iconos de la aplicación, incluidos el que representa la carátula de la aplicación y las imágenes que identifican las condiciones experimentales.

Carpeta Images. La carpeta que contiene las imágenes de los patrones de estímulo que se emplean en la aplicación.

text.htm. Contiene las instrucciones de la tarea para el usuario.

DataRecord.sqlite. Un archivo de base de datos que se copia en el proceso de ejecución de la aplicación cuando se inicia la aplicación. (No implementada)

DataRecord.xcdatamodel. El modelo de Base de Datos Core Data cuya gestión se identifica en la aplicación.