Aplicación móvil para la realización y evaluación de test neuropsicológicos

Mobile application for the realization and evaluation of neuropsychological tests

Adonai Suárez González
Departamento de Ingeniería Informática
Escuela Técnica Superior de Ingeniería Informática
Trabajo de Fin de Grado

La Laguna, 10 de junio de 2014

D. **Patricio García Báez**, con N.I.F. 43.356.987-D profesor Contratado Doctor adscrito al Departamento de Ingeniería Informática de la Universidad de La Laguna

C E R T I F I C A

Que la presente memoria titulada:

"Aplicación móvil para la realización y evaluación de test neuropsicológicos"

ha sido realizada bajo su dirección por D. Adonai Suárez González, con N.I.F. 54.054.288-X.

Y para que así conste, en cumplimiento de la legislación vigente y a los efectos oportunos firman la presente en La Laguna a 10 de junio de 2014

Agradecimientos

En primer lugar quiero mostrar mi profundo agradecimiento al director de este trabajo, D. Patricio García Báez, quien ha sabido guiarme satisfactoriamente durante estos meses para la consecución de los objetivos. Su ayuda tiene un valor incalculable y experiencia de trabajar junto a él ha sido muy agradable y enriquecedora.

En segundo lugar, me gustaría agradecer al Doctor Norberto Rodríguez Espinosa, del Departamento de Neurología del Hospital Universitario Nuestra Señora de Candelaria, toda la información que nos ha facilitado sobre el campo en el que es especialista, así como las opiniones y sugerencias que nos ha comunicado.

Y por último, y no por ello menos importante, me gustaría dedicar unas líneas a mi familia, es especial a mis padres, Pedro y Josefa, y a mi novia, Emma. La finalización de este trabajo supone la conclusión de una etapa en muy importante en mi vida, que sin sus ánimos, apoyo y afecto es muy probable que no la hubiera completado.

Resumen

El objetivo de este trabajo ha sido el de desarrollar una aplicación móvil que proporcione las funcionalidades de realizar testa neuropsicológicos y evaluarlos. Además, se ha diseñado un protocolo de configuración que sirve para añadir nuevos tests configurados por el usuario.

El trabajo previo realizado para llevar a cabo estos objetivos se ha dividido en tres frentes. El primero de ello ha sido seleccionar un conjunto de tests neuropsicológicos sobre los que realizar un análisis. Hemos extraído las características estructurales de los tests, así como tipos de preguntas y respuestas peculiares. A partir de los resultados de este análisis hemos diseñado un protocolo para configurar nuevos tests utilizando el lenguaje XML, que se adapta a todos los tests que hemos estudiado, y según el experto consultado, a la mayoría de los tests que se suelen utilizar en neuropsicología. El segundo frente ha sido el de analizar las plataformas móviles existentes, así como las tecnologías que podrían ayudarnos a desarrollar la aplicación, para posteriormente tomar decisiones a este respecto. Y finalmente, el último frente, ha sido el de analizar la oferta de aplicaciones similares que está disponible para extraer ideas útiles que incluir en nuestra aplicación.

Una vez completado este trabajo previo, hemos completado una fase de diseño y desarrollo de la aplicación, de la cual hay que destacar la definición de requisitos y especificaciones, el diseño del protocolo de configuración y la implementación de la aplicación.

Finalmente, el resultado que hemos obtenido ha sido aplicación móvil capaz de realizar, de forma satisfactoria, la tarea para la que fue diseñada. Además, hemos desarrollado un protocolo de configuración válido para los tests estudiados en este trabajo y que también se adapta a otros tipos de tests. Por lo tanto, tenemos una aplicación funcional que puede realizar y evaluar tests, almacenar y mostrar los resultados, gestionar información relativa al paciente, y que puede ser ampliada añadiendo nuevos tests utilizando los métodos diseñados para ello.

Palabras clave

Neuropsicología, neurología, test, aplicación, móvil, XML, PhoneGap, Alzheimer, deterioro cognitivo, Android, JQueryMobile, JQuery, HTML5, Javascript.

Abstract

The objective of this work was to develop a mobile application that allows performing neuropsychological tests and evaluating them. In addition, we have designed a configuration protocol used to add new tests configured by the user.

The previous work done to accomplish these objectives has been divided into three fronts. The first of them is to select a set of neuropsychological tests on which to perform an analysis. We have extracted the structural characteristics of the tests and peculiar kinds of questions and answers. From the results of this analysis, we have designed a protocol to configure new tests using the XML language that adapts to all the tests we have studied, and according to the experts consulted, most of the tests that are commonly used in neuropsychology. The second front was to analyze existing mobile platforms and technologies that could help us to develop the application. And finally, the last front has been to analyze the offer of similar available applications to extract useful ideas to be included in our application.

Once finished this previous work, we have completed a phase of design and development of the application, which should be noted the definition of requirements and specifications, the design configuration protocol and the application deployment. Finally, the result we have obtained is a mobile application able to perform satisfactorily the task for which it was designed. Furthermore, we developed a valid configuration protocol for studied tests, which also adapts to other types of tests. Therefore, we have a functional application that can perform tests and evaluate them, store and display the results, manage patient information, and which can be expanded by adding new tests using methods designed for it.

Keywords

Neuropsychology, neurology, test, app, mobile, XML, PhoneGap, Alzheimer, cognitive impairment, Android, jQueryMobile, JQuery, HTML5, Javascript.

Índice general

Índice de Figuras

Índice de Tablas

CAPÍTULO 1: INTRODUCCIÓN

En el campo de la neuropsicología es común utilizar tests diseñados por expertos para detectar y evaluar a personas con Deterioro Cognitivo Leve, Alzheimer u otras demencias.

En la actualidad, la mayoría de estos tests están disponibles en formato electrónico y para utilizarlos deben ser impresos y rellenados como cualquier tipo de formulario o encuesta. De este proceder se pueden extraer tanto aspectos negativos como positivos. En el primer grupo tenemos el gasto importante de papel y tinta que supone tener que utilizar una copia por paciente y evaluación; la dificultad o molestia de realizar el cálculo del resultado del test; y el espacio físico utilizado para el almacenamiento de los resultados, así como su organización y consulta en el futuro. Por otro lado, como aspecto positivo tenemos la comodidad que ofrece este formato para la introducción de las respuestas del paciente; la movilidad que proporciona el hecho de no necesitar, en la mayoría de los casos, más que unos papeles y un lápiz o bolígrafo para evaluar al paciente; y el hecho de no tener que superar una fase de aprendizaje en el caso de utilizar otro método ajeno al encargado de realizar el test.

En este sentido, se propone para este proyecto desarrollar una nueva herramienta que sirva para la realización y evaluación de test, y proporcione un método sencillo para almacenar y hacer consultas posteriores sobre los resultados obtenidos por los pacientes. La nueva herramienta que se diseñe y se desarrolle debe intentar eliminar las desventajas que supone utilizar el método actual y, a la vez, intentar conservar, en la medida de lo posible, las ventajas que proporciona.

La herramienta que se desarrolle tiene como fin principal servir como apoyo a los neurólogos, o cualquier otro especialista que la utilice, para hacer más fáciles estos procesos de evaluación bastante comunes y repetitivos en el campo de la neuropsicología.

1.1. PROPUESTA

La herramienta propuesta para sustituir al método actual que se utiliza para realizar los test neuropsicológicos es una aplicación informática. Esta aplicación deberá soportar la realización y evaluación de tests neuropsicológicos, así como el almacenamiento y consulta de los resultados generados. La aplicación tiene como finalidad ser utilizada como una herramienta de apoyo por parte del encargado de realizar el test, el paciente no tendrá que interactuar con ella. También es cierto que en determinados tests, el paciente tendrá que mirar la pantalla para leer un texto o observar una imagen, pero nada más allá.

Teniendo en cuenta la movilidad que proporciona el método que actualmente se utiliza para la realización de estos tests, la aplicación deberá ser desarrollada para ser ejecutada en plataformas móviles. De esta manera se iguala esta característica y se evita, en la medida de lo posible, que personas con movilidad reducida tengan que desplazarse para ser evaluados. En este punto, cabe señalar que la aplicación deberá ser totalmente operativa en la fase de realización, evaluación y almacenamiento de resultados en todo momento, sin depender de una conexión a internet o cualquier tipo de sensor del dispositivo móvil.

La aplicación ofrecerá al usuario algunos de los tests más usuales en el campo de la neuropsicología. Sin embargo, también deberá estar diseñada de tal manera que se adapte a la aparición de nuevos tests, proporcionando algún método para que el usuario pueda crearlos, añadirlos a la aplicación y utilizarlos con sus pacientes. Se propone como método un fichero de configuración XML [1] que siga un protocolo de configuración que habrá que diseñar. Este protocolo deberá inspirarse en la estructura de los tests neuropsicológicos más comunes y ser lo suficientemente flexible para soportar nuevos tests.

1.2. OBJETIVOS

Una vez presentada la propuesta que conforma la base de este proyecto toca definir los objetivos que deben cumplirse durante el desarrollo de este proyecto.

Se definen inicialmente unos objetivos básicos que han de completarse en la fase inicial del proyecto, y serán la base donde se apoyará el resto. En este punto hablamos de objetivos que nos ayuden a tener una visión más amplia sobre los tests que se utilizan en la neuropsicología, sobre las tecnologías que se utilizarán y sobre el estado actual del tema abordado.

El primero de ellos es el de analizar los test más utilizados en el campo de la neuropsicología. Se extraerán las partes comunes entre los diferentes tests, así como las características diferenciadoras de cada uno, que también se deberán contemplar. Esto permitirá abordar el diseño del protocolo de configuración necesario para el desarrollo de la aplicación.

El siguiente objetivo es el de analizar las plataformas móviles existentes en la actualidad sobre las que se podría desarrollar la aplicación. A su vez, se analizará los diferentes entornos de desarrollo y tecnologías que se podrían utilizar. Una vez finalizado el análisis habrá que tomar decisiones justificadas en este apartado.

También nos marcamos como objetivo el de analizar la oferta actual de aplicaciones similares, que, junto a opiniones de expertos en la materia de la neuropsicología, nos servirá para definir unos requisitos y especificaciones que nuestra aplicación debería cumplir.

Y finalmente, los objetivos que se marcan como principales y que se apoyaran en los definidos anteriormente son: Completar el desarrollo de la aplicación móvil, que deberá poder sustituir plenamente a las herramientas utilizadas actualmente, sin que ello suponga un suponga una pérdida de calidad en el proceso; y definir el protocolo de configuración necesario para diseñar nuevos tests, y que éste sea lo más flexible posible para adaptarse a la aparición de nuevos tests.

1.3. ANÁLISIS DE LA ESTRUCTURA DE TESTS NEUROPSICOLÓGICOS

El fin de este apartado es identificar los tests que más se utilizan en el campo de la neuropsicología y definir un conjunto sobre el que se trabajara, tanto en la fase de diseño del protocolo de configuración, como en la fase de implementación del mismo. También se estudiará la estructura de estos tests, identificando aspectos comunes y diferenciadores para posteriormente hacer una propuesta de formato básico del protocolo de configuración, que debe definir las bases del protocolo final diseñado.

1.3.1. IDENTIFICACIÓN DE TESTS UTILIZADOS EN LA ACTUALIDAD

Con la inestimable ayuda del Doctor Norberto Rodríguez Espinosa, del Departamento de Neurología del Hospital Universitario Nuestra Señora de Candelaria, en Tenerife, hemos identificado los test más comunes y que más se utilizan actualmente en neuropsicología. Este conjunto de test será utilizado para el diseño del protocolo de configuración así como para la definición de requisitos y especificaciones de la aplicación.

Entre ellos, por sus peculiaridades, hemos querido destacar los siguientes:

Clock Drawing Test (CDT)

El Clock Drawing Test o Test del Dibujo del Reloj [2] es una prueba de detección sencilla, rápida y de fácil aplicación empleada tanto en la práctica clínica como en investigación para valorar el estado cognitivo del sujeto. Evalúa diferentes mecanismos implicados en la ejecución de la tarea, fundamentalmente funciones visoperceptivas, visomotoras y visoconstructivas, planificación y ejecución motoras.

Es una prueba elaborada originariamente por Battersby, Bender, Pollack y Kahn en 1956 para detectar la negligencia contralateral en pacientes con lesión en el lóbulo parietal. Su aplicación se ha extendido al ser una prueba que proporciona valiosa información acerca de diversas áreas cognitivas activadas en la ejecución de la misma, que corresponden a funciones cognitivas semejantes a las que valora el Mini-Mental State Examination de Folstein que veremos posteriormente, entre ellas, lenguaje, memoria a corto plazo, funciones ejecutivas, práxicas y visoespaciales.

Cuestionario de Actividad Funcional de Pfeffer

El Cuestionario de Actividad Funcional de Pfeffer [3] es un cuestionario breve de diez items diseñado por Pfeffer E. y que ha sido testeado, estandarizado y validado. Este test está diseñado para medir las capacidades funcionales de pacientes de avanzada edad. En las pruebas de validación del mismo se observó que las puntuaciones obtenidas en el test se correspondían con los diagnósticos clínicos existentes.

Escala de Depresión Geriátrica. Test de Yesavage

La Escala de Depresión Geriátrica se trata [4] de un cuestionario heteroadministrado breve diseñado para el cribado de la depresión en personas mayores de 65 años.

Fue creado por Jerome A. Yesavage y existen dos versiones: una de quince items y otra de cinco. En las dos versiones se define una escala que criba a al paciente entre un estado normal y un estado depresivo.

Eurotest

El Eurotest [5] es un test cognitivo breve que evalúa el conocimiento y habilidad en el manejo de las monedas de curso legal (euros), así como, la capacidad de recordar las monedas que previamente se han manipulado. Está especialmente indicado para la detección de sujetos con deterioro cognitivo y demencia, tanto en Atención Primaria como en consultas especializadas.

Ha sido desarrollado por el Dr. C. Carnero Pardo, y ha sido sometido a un riguroso proceso de validación en el que han participado activamente múltiples y prestigiosos profesionales de la Neurología y otras disciplinas.

Fototest

El Fototest [6] es un test cognitivo breve que evalúa la capacidad de recordar seis elementos que previamente se le han mostrado al sujeto y se le ha pedido que nombre; entre denominación y recuerdo se inserta una tarea de fluidez verbal en la que el sujeto debe evocar nombres de personas agrupadas por sexo.

Al igual que otros test cognitivos breves, está especialmente indicado para la detección de sujetos con deterioro cognitivo y demencia, tanto en Atención Primaria como en consultas especializadas, pero también es un instrumento adecuado para el seguimiento de pacientes o la evaluación de la respuesta al tratamiento.

Ha sido desarrollado por el Dr. C. Carnero Pardo, y se ha sometido a un riguroso proceso de validación demostrando unas adecuadas propiedades psicométricas y de aplicabilidad.

Inventario Neuropsiquiátrico de Cummings

El Inventario Neuropsiquiátrico (NPI) [7] es una escala que valora la presencia de alteraciones psicopatológicas en pacientes con enfermedades neurológicas, principalmente demencias.

La prueba consiste en valorar la frecuencia y la gravedad de diez trastornos neuropsiquiátricos. El valor para un trastorno será el resultado de multiplicar su frecuencia por su gravedad según las escalas que proporciona el test.

Mini-Mental State Examination

El test Mini-Mental State Examination (MMSE) [8] o Mini Prueba del Estado Mental es un método muy utilizado para detectar el deterioro cognitivo y vigilar su evolución en pacientes con alteraciones neurológicas, especialmente en ancianos.

Fue desarrollado por Marshal F. Folstein, Susan Folstein, y Paul R. McHugh en 1975 como un método para establecer el estado cognoscitivo del paciente y poder detectar demencia o delirios.

Es una herramienta de evaluación rápida que permite sospechar déficit cognitivo pero que, sin embargo, no permite detallar el dominio alterado ni conocer la causa del padecimiento.

Test de Alteración de la Memoria

El Test de Alteración de Memoria (T@M) [9] es un test cognitivo de cribado, con un alto valor discriminatorio para el deterioro cognitivo leve de tipo amnésico y para la enfermedad de Alzheimer leve, entre la población general.

Es un test breve, y fácil de administrar y puntuar. Es un test de cribado completo que evalúa varios subtipos de memoria, y que está basado en la teoría de la consolidación de la memoria, que dice que la memoria episódica requiere del lóbulo temporal medio para su consolidación, alojándose después en los circuitos neocorticales como parte del sistema de memoria semántica.

1.3.2. ANÁLISIS DE SU ESTRUCTURA

Este pretende ser un primer análisis no muy profundo para acercarnos a la estructura básica de los test resultantes del apartado anterior, y que se utilizarán como base en el desarrollo del proyecto.

La primera conclusión de este análisis ha sido conocer la necesidad de diferenciar entre dos aspectos importantes de los test: el diseño y la evaluación.

Cuando hablamos de diseño, nos referimos a todos aquellos elementos que tienen que ver con la interfaz, con la estructura del documento, y que son necesarios para presentar el test al usuario.

El elemento de mayor rango en esta estructura es el propio test y sus metadatos, como pueden ser el título, la descripción, breves instrucciones, etc. En un escalón por abajo, se encuentran las secciones. Estas secciones agrupan preguntas, con sus metadatos asociados, y en algunos casos incluyen nuevas instrucciones. Es necesaria la diferenciación de estas secciones, ya que algunas necesitan del algún tipo de preparación previa a la batería de preguntas. Seguidamente, encontramos uno de los elementos más importantes del test, las preguntas. Cada una de ellas con un texto diferente, y a veces con unas instrucciones específicas. Y por último, otro elemento de los más importantes, la respuesta. En este caso, aunque finalmente se reduzca dar como entrada un número o un texto, existe mucha diversidad en cuanto a la manera de introducirlo: Seleccionar una lista, seleccionar varias opciones o introducir un número son algunas de las que aparecen en los tests estudiados.

Por otra parte, tenemos que hablar del fin de estos tests, que no es sólo recoger las respuestas del paciente, sino evaluarlos cuando se haya completado el test. En este aspecto las opciones son más escasas. Las combinaciones detectadas en el análisis son las de suma, multiplicación o no combinar. Estas combinaciones, aparte de presentarse a nivel de sección y test como era de esperar, también se presentan a nivel de pregunta. Esto plantea la dificultad de que una pregunta puede tener más de una entrada y para obtener un resultado global hay que combinarlas todas.

1.4. ANÁLISIS DE PLATAFORMAS Y ENTORNOS DE DESARROLLO

Actualmente la cuota de mercado de los smartphones se la reparten tres plataformas pertenecientes a tres grandes compañías de software: Android, de Google, con casi un 70%; iOS, de Apple, con algo más de un 20%; y por último Windows Phone, de Microsoft, con casi un 4% [10].

Por lo tanto, una de las primeras decisiones que debemos tomar es si queremos que nuestra aplicación esté disponible para estas tres plataformas o si, por el contrario, nos decantaremos por publicarla exclusivamente en una de ellas.

Después de tomar esta decisión, inmediatamente debemos decidir si optar por un desarrollo nativo o un desarrollo multiplataforma. Si nos hemos decantado por sólo una plataforma, la decisión está más o menos clara, no tendríamos ningún problema en desarrollar de manera nativa, aprovechando al máximo las funcionalidades del SO elegido. Por el contrario, si decidimos que nuestra aplicación esté presente en las tres plataformas, esta decisión se antoja un poco más determinante.

Aunque luego haremos un análisis más exhaustivo sobre este tema, podemos anticipar que desarrollar de manera nativa tiene una desventaja evidente frente a el desarrollo multiplataforma: mayor tiempo de desarrollo. Necesitaremos programar nuestra aplicación tres veces, una por cada lenguaje de programación de cada plataforma, adaptándonos a las características diferenciadoras de cada SO. Y en esta dirección, pero en sentido contrario, con el desarrollo multiplataforma nos alejamos de las peculiaridades de cada SO, consiguiendo aplicaciones menos adaptadas a su look&feel, pero ganando una disminución del tiempo de desarrollo.

Éstas, y algunas otras, son las ventajas y desventajas que podremos ver a continuación, cuando hagamos la comparativa entre las diferentes opciones existentes para llevar a cabo el desarrollo de la aplicación. Una vez terminado este análisis, estaremos en disposición de tomar una decisión respecto de este tema, para seguir adelante con el desarrollo del proyecto.

1.4.1. DESARROLLO NATIVO

Cada una de las plataformas principales tiene su lenguaje de programación nativo: Java en Android, Objetive-C en iOS, y C# en Windows Phone. Una de las ventajas de desarrollar en cada uno de estos lenguajes es que proporciona a nuestra aplicación un mejor rendimiento, ya que cada plataforma está optimizada para sacarle el máximo partido a su lenguaje. Esto también tiene su parte mala, y es que nos obliga a especializarnos en cada uno de los lenguajes.

Otra clara ventaja del desarrollo nativo es la posibilidad de exprimir al máximo las funcionalidades de cada sistema operativo, y conseguir una aplicación adaptada a la cultura y look&feel de cada plataforma. También tiene su parte negativa, y es que debemos cambiar las formas de hacer las cosas en cada una de las plataformas, así como la interfaz de usuario.

En cualquier caso, la principal desventaja por decantarse por sólo una plataforma, que sería la opción viable si se elige el desarrollo nativo, es la de ignorar a los potenciales usuarios que podríamos tener en las otras plataformas.

Android

Una de las principales ventajas de desarrollar para Android [11] es la gran cuota de mercado que ocupa, lo cual nos haría llegar a un número de usuarios bastante atractivo. Por otra parte, existen muchos terminales que llevan este SO que tienen un precio bastante asequible, lo cual también puede ser un factor importante de acuerdo al target de nuestra aplicación.

En cuanto al ecosistema proporcionado por Google para los desarrolladores podemos decir que es bastante completo. El IDE que normalmente se utiliza es Eclipse, compatible con Windows, Linux y MacOS, permitiéndonos utilizar cualquiera de estos tres sistemas operativos para desarrollar nuestra aplicación. Además del IDE, necesitaremos instalar el Android SDK y el plugin ADT para Eclipse [12], y con ello tendremos un entorno totalmente productivo para desarrollar nuestras aplicaciones.

Por último, como ya hemos dicho anteriormente, es necesario conocer el lenguaje de programación Java, y para las interfaces, XML. Además, será necesario conocer la estructura de los proyectos que genera Eclipse y las directrices que se deben seguir para desarrollar la aplicación en esta plataforma.

iOS

iOS es la segunda plataforma con más cuota de mercado actualmente, y esto es uno de sus principales atractivos. Además, sus usuarios tienen más asimilada la cultura del pago por aplicación que los usuarios del resto de plataformas, lo cual no deja de ser un factor interesante si queremos obtener un beneficio económico por nuestra aplicación.

El ecosistema proporcionado por Apple es mucho más cerrado que el de Google, pues tenemos como requisito fundamental el poseer un Mac. Esto es así, porque para desarrollar aplicaciones para dispositivos iOS es necesario utilizar el IDE X-Code, una herramienta exclusiva del Sistema Operativo MacOS. Este entorno nos proporciona todo lo necesario para desarrollar una aplicación de principio a fin sin necesidad de instalar software adicional.

Además, como ya habíamos comentado, tendremos que conocer el lenguaje de programación Objetive-C, el único con el que podremos desarrollar en esta plataforma. Y al igual que en las otras dos plataformas, debemos conocer la estructura de la aplicación y las directrices que debemos cumplir para llevar a cabo su desarrollo.

Windows Phone

Windows Phone es la plataforma con menor cuota de mercado de las tres. También es cierto que es un Sistema Operativo móvil más joven que sus dos competidores. Tampoco hay que ignorar, que en determinados mercados, está obteniendo cuotas de mercado cercanas a iOS. En cualquier caso, Windows Phone se considera una plataforma en crecimiento, lo cual nos proporciona nichos de mercado que están casi sin explotar o incluso sin explotar donde podemos posicionarnos con nuestra aplicación de una manera fácil.

Por su parte, Microsoft, proporciona un ecosistema de desarrollo parecido al de Apple. Es necesario desarrollar aplicaciones desde un Sistema Operativo Microsoft Windows. El IDE que nos proporciona es Visual Studio, con versiones Express gratuitas específicas para el desarrollo de aplicaciones para su plataforma móvil, que se instalan de manera muy fácil y no debemos configurar nada. También podemos adquirir versiones de pago más completas, aunque con las gratuitas podremos hacer todo lo necesario para desarrollar una aplicación desde cero y publicarla en su tienda.

En cuanto a los conocimientos necesarios, debemos conocer uno de los dos lenguajes con los que podremos programar: C# ó VB. Además, el diseño de las interfaces se lleva a cabo con otro lenguaje de marcas: XAML. También es cierto, que el IDE nos proporciona un editor de interfaces bastante completo, haciendo que no sea tan necesario un conocimiento avanzado de XAML. Por último, como en el resto de plataformas, debemos conocer la estructura de la aplicación así como las directrices que debemos seguir para el desarrollo de la aplicación.

1.4.2. DESARROLLO MULTIPLATAFORMA

El desarrollo multiplataforma busca apoyarse en otras tecnologías comunes a todas las plataformas para desarrollar la aplicación. Algunas de estas herramientas utilizan HTML5 [13], CSS [14] y Javascript [15] para generar aplicaciones web que serán ejecutadas en un componente WebKit (navegador) del sistema operativo.

Otras, utilizan un lenguaje común como Javascript, para el desarrollo, como si de un lenguaje nativo se tratara, que después es convenientemente tratado para ejecutarlo en cada una de las aplicaciones.

Independientemente de la herramienta utilizada, la principal ventaja que tiene este tipo de desarrollo es el bajo tiempo de desarrollo frente al desarrollo nativo. Programaremos una vez nuestra aplicación, y nos servirá para muchas plataformas. En ocasiones, hay que hacer pequeños ajustes, pero la ganancia en tiempo de desarrollo sigue siendo notable.

La principal desventaja de este desarrollo es el rendimiento. Todas las herramientas que se han valorado basan su funcionamiento en Javascript, lenguaje que debe ser interpretado por Javascript Engine o por un navegador, dependiendo del caso, lo cual se conoce que afecta negativamente al rendimiento de la aplicación. Además, en las herramientas que utilizan el navegador para ejecutar la aplicación, se añade una dificultad más: el renderizado HTML y CSS que hace cada navegador. Aunque esto es parcialmente salvable, desde el punto de vista del programador, utilizando alguno de los frameworks disponibles para implementar las UI's.

Appcelerator Titanium

Appcelerator Titanium [16] nos permite crear aplicaciones para Android, Iphone y, además, de escritorio. Para programar proporciona Titanium Studio, un IDE basado en Eclipse con el que crear los proyectos y editar los ficheros Javascript y el resto de recursos y lanzar los scripts de creación.

Las aplicaciones se programan con Javascript, y toda la interfaz hay que programarla utilizando este lenguaje, usando para ello una serie de librerías "puente" que hacen de intermediario entre tu aplicación Javascript y los controles del sistema. Por lo tanto, los controles utilizados serán nativos, lo cual hace más rápido su renderizado y la respuesta del usuario.

Una vez se tiene la aplicación desarrollada se procede a su empaquetamiento. En este proceso, el Javascript es transformado y compilado. Y posteriormente, cuando se arranca la aplicación en el móvil, el código se ejecuta dentro de un Javascript engine, que depende de cada plataforma.

El hecho de utilizar un Javascript compilado y que los controles sean nativos, hace que consiga el máximo rendimiento de este tipo de desarrollo, que viene limitado por el propio lenguaje Javascript.

PhoneGap

PhoneGap [17] es un framework para el desarrollo de aplicaciones para iOS, Android, Windows Phone 7/8 y Blackberry 10, entre otros. Principalmente, permite a los programadores desarrollar aplicaciones para dispositivos móviles utilizando herramientas genéricas tales como JavaScript, HTML5 y CSS3. Las aplicaciones resultantes son híbridas, es decir que no son realmente aplicaciones nativas al dispositivo (ya que el renderizado es realizado mediante vistas web y no con interfaces gráficas específicas a cada sistema), pero no se trata tampoco de aplicaciones web (teniendo en cuenta que son aplicaciones que son empaquetadas para poder ser desplegadas en el dispositivo incluso trabajando con el API del sistema nativo).

PhoneGap maneja APIs que permiten tener acceso a elementos como el acelerómetro, cámara, contactos en el dispositivo, red, almacenamiento, notificaciones, etc. Estas API's se conectan al sistema operativo usando el código nativo del sistema huésped a través de una Interfaz de funciones foráneas en Javascript.

Además nos permite el desarrollo ya sea ejecutando las aplicaciones en nuestro navegador web, sin tener que utilizar un simulador dedicado a esta tarea, nos da la posibilidad de soportar funciones sobre frameworks como Sencha Touch o JQuery Mobile [18].

1.4.3. TABLAS COMPARATIVAS

En las siguientes tablas, hemos intentado hacer una comparación entre las diferentes opciones que hemos visto hasta ahora para desarrollar la aplicación. El objetivo es resumir de una manera más gráfica lo que hemos comentado hasta el momento y que esto nos ayude a tomar la decisión final.

Se han comparado las características principales de cada una de ellas y se han asignado valores {Bajo, Medio, Alto}, según la información que se ha podido sacar de la documentación oficial de cada plataforma y los diferentes artículos que se han consultado.

	Android	iOS	Windows Phone
Cuota de mercado	Alto	Medio	Bajo
Remuneración	Bajo	Alto	Medio
Flexibilidad Desarrollo	Alto	Medio	Medio

Tabla 0.1: Comparativa desarrollo nativo

	Titanium	PhoneGap
Rendimiento	Alto	Medio
Renderización interfaz	Alto	Medio
Uniformidad desarrollo	Bajo	Alto
Plataformas compatibles	Medio	Alto

Tabla 0.2: Comparativa desarrollo multiplataforma

	Nativo	Multiplataforma
Rendimiento	Alto	Medio
Ahorro en tiempo de desarrollo	Bajo	Alto
Adaptabilidad al SO	Alto	Bajo
Uniformidad desarrollo	Bajo	Alto

Tabla 0.3: Comparativa desarrollo nativo y multiplataforma

1.5. ANÁLISIS DE APLICACIONES SIMILARES

En este punto ya se sabe que el objetivo del proyecto es desarrollar una aplicación que tenga dos características esenciales: evaluación de tests y posibilidad de diseñar nuevos tests.

Tras analizar la oferta disponible en las diferentes tiendas de aplicaciones móviles, ésta puede ser dividida en dos grupos de interés: las que permiten el diseño y las que permiten evaluación. Además, existe un tercer grupo de aplicaciones, fuera de las plataformas móviles, que también se analizará y que permite tanto una cosa como la otra.

1.5.1. APLICACIONES QUE PERMITEN EL DISEÑO DE TESTS

En el primer grupo, las aplicaciones están enfocadas principalmente a ámbitos de marketing y consumo. En este grupo se encuentra Rotator Survey [19] [20], una aplicación que "Permite el diseño de cuestionarios con todo tipo de preguntas, el manejo sencillo del flujo de la entrevistas mediantes saltos entre preguntas y formulas de flujo", según sus creadores. Se apoya en una estructura en la nube, permitiendo que el usuario pueda consultar los resultados de las encuestas vía web. Además, ofrece la posibilidad de trabajar de forma offline, guardando los datos de manera local para no depender de una conexión a internet, pudiendo enviarlos al servidor cuando sea posible.

Esta aplicación sólo permite el diseño de cuestionarios para sacar una conclusión sobre un grupo de encuestados, no para evaluar el resultado individual de cada uno. Por lo tanto es una aplicación que nos sirve para fijarnos en la forma en la que se pueden diseñar los cuestionarios, y sacar alguna idea que nos ayude en nuestro proyecto.

1.5.2. APLICACIONES QUE PERMITEN LA EVALUACIÓN DE TESTS

En el segundo grupo grupo de aplicaciones, se encuentran aquellas que sólo sirven para evaluar al usuario en sobre un determinado tema, y no permiten crear test propios.

Hemos encontrado algunas aplicaciones de este tipo y además enfocadas al tema que nos ocupa. La primera de ellas es Mini Mental Test (portuguese) [20], y consiste, como se puede adivinar, en una aplicación para evaluar a un paciente siguiendo las preguntas del MMSE que se explicó en apartados anteriores. La segunda es Geriatric Depresion J Yesavage [21],y como ocurre en el caso anterior evalúa sólo sobre la Escala de Depresión Geriátrica de de Yesavage. En estos dos casos, no se permite almacenar un historial de paciente ni exportar los resultados, por lo que sólo se trata de una versión electrónica de los tests sin ninguna funcionalidad extra.

También hemos encontrado una tercera aplicación, algo más avanzada, no sólo por la cantidad de tests que ofrece, sino por algunas de las funcionalidades que coinciden con las propuestas para este proyecto. Hablamos de Nurse Test [22], una aplicación que ofrece más de cuarenta tests, todos relacionados con la medicina, pero sólo algunos con la neuropsicologías, entre los que se encuentran la Escala de Depresión Geriátrica de de Yesavage y el Cuestionario de Actividad Funcional de

Pfeffer. Las funcionalidades que destacamos son la posibilidad de llevar un historial de cada paciente y poderlo exportar en cualquier momento.

1.5.3. APLICACIONES QUE PERMITEN EL DISEÑO Y LA EVALUACIÓN DE TESTS

No hemos encontrado una aplicación móvil que cumpla estos dos requisitos, pero sí varias herramientas, muchas de ellas online, orientadas a un entorno de escritorio que si los cumplen. En concreto, queremos destacar la herramienta de Cuestionarios de la plataforma Moodle [23].

Esta herramienta permite diseñar y aplicar cuestionarios dirigidos principalmente al ámbito académico. Ofrece una amplia variedad de tipos de pregunta, como "opción múltiple", "verdadero/falso" o "respuestas cortas". La parte negativa es que no es una herramienta adaptada a dispositivos móviles y su uso en ellos es muy incómodo.

De las que hemos encontrado, quizá ésta sea la herramienta más parecida a la que se quiere desarrollar en este proyecto. Además de poder diseñar un cuestionario con la ayuda de un asistente, también permite la carga de cuestionarios mediante ficheros de configuración en varios formatos. En este punto, se asemeja mucho a lo que queremos conseguir en el proyecto.

Por otra parte, tiene soporte para la parte más importante del cuestionario: la evaluación. Se puede configurar la manera en la que se puntúa cada una de las preguntas, secciones o todo el cuestionario. Y esto es algo esencial para nuestro proyecto.

CAPÍTULO 2: DISEÑO Y DESARROLLO DE LA APLICACIÓN

En este capítulo describiremos la fase de diseño y desarrollo de la aplicación móvil y el protocolo de configuración que se han planteado como objetivos principales de este proyecto en el capítulo anterior.

En la primera mitad se tomarán algunas decisiones necesarias para el desarrollo del proyecto, como la elección de la plataforma móvil sobre la que trabajar y la definición exacta de requisitos y especificaciones de la aplicación a desarrollar.

En la segunda mitad, el contenido está más orientado a mostrar los resultados, centrándonos en explicar el protocolo de configuración y la estructura de la aplicación, así como describir las interfaces gráficas utilizadas para realizar las funcionalidades más importantes de la aplicación.

2.1. ELECCIÓN DE LA PLATAFORMA Y ENTORNO DE DESARROLLO

Tras valorar el análisis expuesto en el punto 1.4, es el momento de elegir una plataforma y un entorno de desarrollo como propuesta para la ejecución del proyecto.

En primer lugar, quedemos descartar a iOS como candidata porque no contamos con el hardware y software necesarios para desarrollar en esta plataforma.

En cuanto, a las otras dos plataformas presentadas en el apartado1.4.1, sí que disponemos de los medios necesarios para llevar a cabo en ellas el desarrollo de la aplicación del proyecto, aunque nuestra decisión no va por esta dirección. En cualquier caso, de haber elegido el desarrollo nativo, nos hubiéramos decantado por Android, debido a su cuota de mercado y a que el target al que va dirigido la aplicación se beneficiaría más de la diversidad de dispositivos y precios que proporciona esta plataforma.

Como se puede adivinar en el párrafo anterior, nos hemos decantado por un desarrollo multiplataforma. Las razones principales son que este tipo de desarrollo está enfocado a aplicaciones sencillas que no precisan utilizar las características más avanzadas del Sistema Operativo. Y que además, se pueda reutilizar la mayor parte del código para lanzar la misma aplicación en las plataformas principales del panorama actual. El mayor problema que hemos observado en el análisis es el del rendimiento, pero la aplicación no tendrá un

coste computacional muy elevado, así que no es algo que deba preocupar, por lo menos a priori.

En cuanto a la herramienta elegida dentro de las presentadas en el apartado de desarrollo multiplataforma elegimos PhoneGap. Aquí, mis conocimientos sobre tecnologías web hacen que se decante la balanza. El hecho de poder utilizar HTML y CSS, así como frameworks que conozco como JQuery Mobile o Sencha, para el diseño de las interfaces es un punto a favor que me empuja hacia PhoneGap y me aleja de Titanium. Además, la uniformidad entre plataformas que proporciona esta herramienta nos asegura que, quizá estando limitados en algunos aspectos, todo lo que se haga para una plataforma funcionará para las demás. Tampoco está demás añadir que soy partidario de utilizar tecnologías y lenguajes abiertos y estandarizados como son las tecnologías web que se mencionan más arriba. Creo que sería beneficioso para todos los desarrolladores de software, y centrándonos de momento sólo en aplicaciones móviles, que todas las plataformas existentes en la actualidad adoptaran estos estándares y dieran la posibilidad de desarrollar aplicaciones que aprovecharan al máximo las posibilidades de sus dispositivos.

2.2. DEFINICIÓN DE REQUISITOS Y ESPECIFICACIONES

De manera unilateral hemos definido unos requisitos que consideramos esenciales para considerar como éxito el desarrollo del proyecto. Estor requisitos se basan en la funcionalidades mínimas exigibles a una aplicación con las características que se han propuesto en el capítulo uno. Además, estos requisitos han sido ampliados en algunos casos teniendo en cuenta la opinión de un experto en la materia, el Doctor Norberto Rodríguez Espinosa.

El primero de ellos y más importante es que se pueda llevar a cabo cómodamente la ejecución de tests neuropsicológicos. Se deberá diseñar una interfaz gráfica clara y que no confunda al usuario, mostrando en cada pantalla sólo la información necesaria para el paso que esté completando. Hay que recordar que la aplicación no está dirigida al usuario al que se evalúa, sino al evaluador, por lo tanto se permitirá cierta libertad de desplazamiento entre las preguntas que conforman un test sin ningún tipo de limitación. Lo cierto es que en algunos tests, el orden para realizar las preguntas es esencial, pero entendemos que el evaluador es consciente de ello y le ofrecemos esta "libertad de movimiento" para su comodidad, corrección de errores o cualquier cosa que estime oportuna. Será necesario también controlar el

estado del test en cada momento, de manera que si se detiene la evaluación del mismo por demanda del evaluador, porque se cierre la aplicación o por cualquier otro motivo, se pueda retomar el test desde donde se dejó sin perder la información introducida hasta el momento anterior a la detención.

El segundo de los requisitos esenciales es el de calcular y mostrar los resultados. Una vez finalizado el tests, se calcularán los resultados, haciendo las combinaciones oportunas a nivel de pregunta, sección y test, y se presentarán al usuario. La opinión del experto consultado ha sido esencial para definir este requisito. Nos ha instado a presentar más que un resultado final del test, un resumen completo donde aparezca la puntuación en cada pregunta, sección y finalmente el valor final del test. Nos comenta que él como neurólogo no sólo analiza la puntuación final del test, sino las partes del mismo que han influido con mayor peso en el resultado. Además, nos ha recomendado que el resultado sea presentado de manera íntegra y que no sea la aplicación la que interprete, sino el propio experto el que examine los resultados y emite una valoración. Y por último nos indica que en tests como el MMSE o el CDT, en los que se pide al paciente que dibuje figuras, sería interesante adjuntar al resultado una copia de ese dibujo, la cual proporcionaría información más valiosa que el propio resultado numérico de la pregunta.

Otro de los requisitos esenciales es el del almacenamiento y gestión de los resultados obtenidos en las diferentes evaluaciones que se realicen. Es necesario llevar un control de cada uno de los resultados, guardando información relativa al paciente, fecha en la que se realizó, tiempo que se tardó en ejecutarse y alguna otra información que se considere útil. También se considera necesario la posibilidad de visualizarlos en todo momento y poder exportarlos en un formato estándar que pueda ser portado a otro dispositivo y examinarlo sin dificultades.

Del requisito anterior se desprende un requisito también necesario, que es la gestión de pacientes y la información relativa a ellos. En este punto, hemos tenido en cuenta la LOPD y hemos considerado evitar ofrecer la posibilidad de guardar datos de carácter personal y ofrecer solamente la posibilidad de almacenar un identificador que proporcionará el evaluador y que será él quién se ocupe de que sea lo suficientemente seguro para que el paciente no sea identificado.

El último requisito esencial que hemos definidos es el de la gestión de los tests. Esto incluye la posibilidad de añadir, visualizar y eliminar los tests que tengamos almacenados en el dispositivo. Para este requisito es necesario el diseño de un protocolo de configuración, que el usuario deberá respetar para poder introducir en

su aplicación cualquier test que no esté disponible y que desee utilizar con sus pacientes. Este protocolo deberá ser los suficientemente genérico y flexible para adaptarse a la aparición de nuevos tests. En la reunión con el experto consultado, nos indicó, como ya sabíamos, que existen tests como el Fototest o el MMSE que se apoyan en imágenes en algunas de sus preguntas, por lo tanto, el protocolo diseñado debe aceptar este tipo de preguntas si el usuario considera que se deben visualizar desde el dispositivo móvil.

Una vez definidos los requisitos de la aplicación, pasaremos a definir algunas especificaciones técnicas que han de combinarse con los anteriores para llevar a cabo el desarrollo de la aplicación. Todas las especificaciones se han decidido por acuerdo entre el tutor y el alumno tras discutirlas en las tutorías que se han llevado a cabo a los largo de estos meses.

Una de las especificaciones principales, que se definió desde el principio del proyecto, y que se ha ratificado en el punto 1.3.3. es la de utilizar el lenguaje de marcado XML para el diseño del protocolo de configuración.

Estos ficheros XML serán cargados desde el sistema de ficheros del propio dispositivo. Habrá que proporcionar al usuario los métodos necesarios para seleccionar el fichero que desee añadir y posteriormente transformarlo al modelo de datos interno de la aplicación.

En el lado opuesto, en la exportación del historial de resultados se utilizará el formato de fichero estándar CSV, por su amplia compatibilidad con aplicaciones de ofimática tanto de código abierto como privativo. Esta vez, el resultado de la exportación se almacenará en el sistema de archivos del propio dispositivo, y estarán disponibles para que el usuario pueda extraerlos del mismo sin limitaciones.

En cuanto al uso de funcionalidades del dispositivo móvil necesarias para cumplir los requisitos de la aplicación, habrá que proporcionar acceso a la cámara del mismo, a la galería de imágenes y al sistema de archivos. Además, también será necesario habilitar la conexión a internet, sólo para el momento de la importación de nuevos tests.

Para el almacenamiento de los datos en el dispositivo no se especifica ninguna herramienta o tecnología en particular. Sin embargo si se requerirá que exista integridad y coherencia en los datos almacenados.

2.3. PLAN DE TRABAJO

En este apartado pasamos a explicar las actividades e hitos que se definieron en la fase inicial del proyecto así como su distribución temporal para su ejecución. Estas actividades incluyen el diseño e implementación de la aplicación y el protocolo de configuración, así como los pasos previos necesarios, explicados en su mayoría en el capítulo 1, para llevar a cabo los primeros.

2.3.1. DEFINICIÓN DE ACTIVIDADES E HITOS

Preparación

Para conocer el estado actual del tema a tratar de definieron una serie de actividades o hitos que se deberían completar antes de empezar con la fase de diseño y desarrollo. El resultado de estas actividades se expone en los puntos 1.3 1.4 y 1.5 y son: Analizar la estructura de los tests más comunes en neuropsicología, analizar las plataformas y entornos de desarrollo en los que se podría apoyar el proyecto y analizar las aplicaciones similares existentes en la actualidad.

Una vez completadas estas actividades, entramos de lleno en la fase de diseño tanto de la aplicación como del protocolo de configuración. En esta fase se definen previas a la fase de desarrollo.

Diseño

La primera de estas actividades es la elección, teniendo en cuenta el análisis previamente realizado, de la plataforma y entorno de desarrollo en los que se realizará la aplicación. Posteriormente se definirán los requisitos y especificaciones de la misma.

Llegados a este punto estamos en disposición de diseñar la estructura de la aplicación, incluyendo todas las entidades que la conforma y sus relaciones, y definir los pasos que deberá realizar el usuario para conseguir realizar cada una de las tareas disponibles.

Otra de las actividades consideradas importantes que se programó es la de diseñar la interfaz gráfica que se utiliza para representar cada uno de los tipos de respuesta que se esperan utilizar y se contemplan en el protocolo de configuración.

Por último y pensada para ser desarrollada de forma paralela al resto actividades anteriormente definidas, se encuentra la del diseño del protocolo de configuración.

Desarrollo

Las actividades que se pueden considerar actividades de la fase de desarrollo son aquellas que han de completarse después de la fase de diseño y que deberán completar para pasar la fase de pruebas.

Entre estas actividades se encuentra la de desarrollar el modelo de datos necesario para representar los tests dentro de la aplicación, así como el resto de entidades que necesiten ser modeladas. Habrá que implementar los métodos necesarios para el almacenamiento de estos modelos, recuperación y modificación. Por otra parte, habrá que implementar la lógica de comunicación entre estos modelos y otras entidades encargadas del flujo de información.

Otra de las actividades programadas es la de implementar la interfaz de usuario y los flujos de navegación apoyándonos en los resultados de las actividades de la fase de diseño. También en este punto se contempla el control del estado de la aplicación en todo momento.

Por último, y no menos importante, se implementará la parte que quizá sea más crítica, el intérprete del protocolo de configuración. Como se dijo en la fase anterior, el diseño del protocolo de configuración se hacía de manera paralela a otras actividades, y en concreto nos referíamos a esta. Deben ser de alguna manera paralela para la detección de errores de diseño y actuar en consecuencia. Además, será necesario para esta actividad configurar algunos de los tests mencionados en el punto 1.3.1 que sirvan como apoyo a las pruebas que se vayan realizando sobre la validez del protocolo.

Pruebas

En esta fase de definió un hito que era el de someter la aplicación a un grupo de usuario externos y utilizar algún método de testeo y mejora incremental, enfocado sobre todo a las interfaz gráfica, que finalmente, por falta de tiempo, no se ha podido llevar a cabo.

En sustitución de estas pruebas, nos hemos reunido con el experto consultado, Norberto Espinosa, le hemos dejado probar la aplicación y nos ha sugerido algunos cambios y sugerencias que fueron añadidos a la aplicación final.

2.3.2. DISTRIBUCIÓN TEMPORAL

A continuación se expone la tabla de tareas con su distribución temporal:

Aplicación móvil para la realización y evaluación de tests neuropsicológicos

Nombre	Fecha de inicio	Fecha de fin	Recursos
Preparación	27/01/14	7/03/14	
Análisis tests más comunes	27/01/14	7/02/14	
Análisis plataformas y entornos	10/02/14	21/02/14	
Análisis aplicaciones similares	24/02/14	7/03/14	
Diseño	10/03/14	4/04/14	
Elección plataforma y entorno	10/03/14	14/03/14	
Definición requisitos y especificaciones	17/03/14	21/03/14	
Diseño estructura aplicación	24/03/14	28/03/14	
Diseñar interfaz gráfica	31/03/14	4/04/14	
Diseño protocolo configuración	10/03/14	4/04/14	
Desarrollo	24/03/14	30/05/14	
Implementar intérprete de los ficheros de configuración	24/03/14	18/04/14	
Desarrollar modelo de datos	7/04/14	18/04/14	
Desarollar gestor almacenamiento	21/04/14	2/05/14	
Desarrollar lógica comunicación	5/05/14	9/05/14	
Implementar interfaz de usuario	12/05/14	23/05/14	
Implementar control del estado	26/05/14	30/05/14	
Pruebas	2/06/14	3/06/14	
Testeo con experto	2/06/14	3/06/14	

Tabla 0.1: Distribución temporal de actividades

Y a continuación un esquema con línea temporal para observar la distribución de manera más clara:

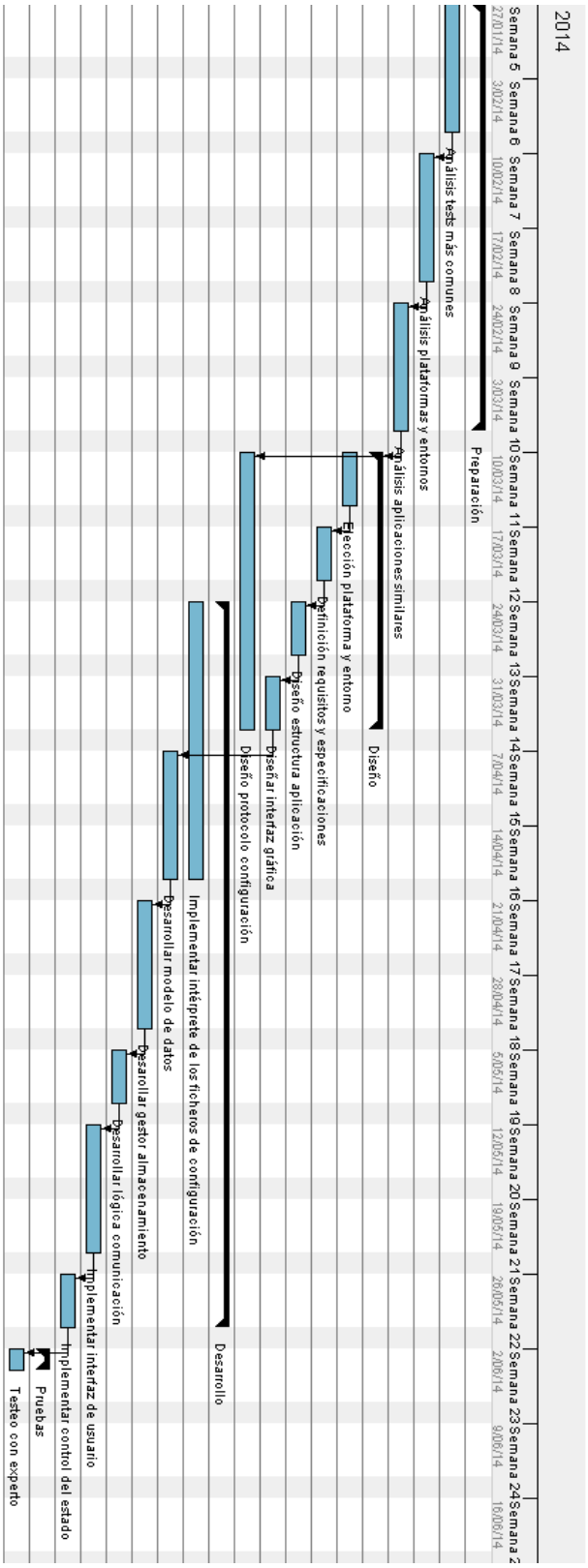

Figura 0.1: Distribución actividades con línea temporal

2.4. PROTOCOLO DE CONFIGURACIÓN

En este apartado procedemos a proponer una estructura base del protocolo de configuración, extenderemos la misma y explicaremos todos los elementos que la componen y por último expondremos las opciones especiales diseñadas para dar cabida a las características especiales encontradas en los tests estudiados.

Pero antes nos detendremos para indicar los conocimientos que debería tener una persona que quiera incorporar nuevos tests. En primer lugar es esencial que conozca el lenguaje XML para poder escribir un fichero que sea válido según las especificaciones de este lenguaje. En segundo lugar, deberá conocer las características del protocolo que hemos diseñado para poder escribir un fichero que, aparte de ser válido como XML, sea válido para nuestra aplicación. Esta estructura, que veremos a continuación destacando los detalles más importantes, está definida también en un documento XML Schema [24] que se puede consultar en el apéndice. En cualquier caso, a la hora de importar un nuevo test, la aplicación notifica al usuario de los errores que encuentre en su estructura.

Por último, y no por ello menos importante, debería tener conocimientos en neuropsicología, apoyándose, si fuera necesario, en un neurólogo que controlara la calidad del test diseñado. En otro caso, se podrían diseñar tests totalmente compatibles con nuestra aplicación, pero no válidos para la disciplina médica que se ha tenido en cuenta en este proyecto.

2.4.1. PROPUESTA DE ESTRUCTURA BASE DEL PROTOCOLO DE CONFIGURACIÓN

Tras completar el análisis del punto 1.3, referente a los tests utilizados en neuropsicología y su estructura, estamos en disposición de proponer una estructura básica del protocolo de configuración para empezar a trabajar sobre ella.

Lo primero de todo es señalar que se utilizará el lenguaje de marcas XML para los ficheros de configuración, como se había propuesto en el punto 1.1. Gracias a que es un estándar es compatible con muchas herramientas y tecnologías que podrían ser útiles para el desarrollo de este proyecto.

La estructura básica contará con cuatro nodos principales, los que fueron nombrados como elementos en el punto anterior: *test*, *section*, *question*, y *answer*. Además, contará con nodos secundarios para la definición de metadatos y otro tipo de información, como: *title*, *description*, *result* e *instructions*. Y por último se añaden otros nodos que en la mayoría de los casos no añaden información

adicional, sino que sirven para hacer la estructura del protocolo más coherente, como: *questions*, *answers* y *display*.

En la siguiente ilustración podemos ver la estructura básica que hemos definido y sobre la que se trabajará para el desarrollo del protocolo de configuración de test que se incluirá en el proyecto.

```
1
2
3  <?xml version="1.0" encoding="utf-8" ?>
4  <Test>
5      <Title></Title>
6      <Description></Description>
7      <Result />
8      <Section>
9          <Title></Title>
10         <Description></Description>
11         <Result />
12         <Questions>
13             <Question>
14                 <Display>
15                     <Title></Title>
16                     <Instructions></Instructions>
17                 </Display>
18                 <Answers>
19                     <Answer>
20                     </Answer>
21                 </Answers>
22             </Question>
23         </Questions>
24     </Section>
25 </Test>
```

Figura 0.2: Estructura base protocolo de configuración

2.4.2. EXTENSIÓN DE LA ESTRUCTURA

A partir de la estructura base propuesta en el apartado anterior se ha ampliado la misma para poder dar soporte a los tests estudiados en este proyecto. Para describir la estructura nombraremos los nodos desde el nivel más alto al nivel más bajo de la jerarquía e iremos señalando a su vez los nodos que acepta cada uno de ellos.

Empezamos por el nodo *Test*, que será el que englobe a todos los demás. Este nodo no acepta ningún parámetro adicional y se debe añadir de manera obligatoria un nodo *Title*, *ShortTitle* y *Description* que dependan de él. La finalidad de cada uno de ellos queda descrita por su propio nombre. Además, con carácter opcional, se podrá añadir un cuarto nodo *Result*, que será explicado más adelante, y que servirá para definir el tipo de evaluación a nivel de test que se desea. Por último, el nodo *Test* acepta un tipo de nodo llamado *Section*, y se podrán añadir tantos como sean necesarios.

```
<?xml version="1.0" encoding="utf-8" ?>
<Test>
    <Title>FotoTest</Title>
    <ShortTitle>FT</ShortTitle>
    <Description>
        El Fototest es un test cognitivo breve que evalúa la capacidad de recordar seis elementos que
        previamente se le han mostrado al sujeto y se le ha pedido que nombre; entre denominación y recuerdo se
        inserta una tarea de fluidez verbal en la que el sujeto debe evocar nombres de personas agrupadas por sexo.
        Al igual que otros test cognitivos breves, está especialmente indicado para la detección de sujetos con
        deterioro cognitivo y demencia, tanto en Atención Primaria como en consultas especializadas, pero también
        es un instrumento adecuado para el seguimiento de pacientes o la evaluación de la respuesta al tratamiento.
    </Description>
    <Result />
    <Section></Section>
</Test>
```

Figura 0.3: Detalles del nodo Test

Con el nodo *Section* entramos en el que sería el segundo nivel de la jerarquía de nodos, y con el podremos dividir el test en secciones.

Una sección de un test es una serie de preguntas agrupadas, que normalmente tienen alguna característica similar. Este tipo de elemento también sirve para diferenciar fases del test, que tratan procesos cognitivos diferentes, o que deben de ejecutarse de una manera diferente al resto. Sirva como ejemplo el test T@M expuesto en el punto 1.3.1, en el que se divide el contenido en cinco secciones: Memoria inmediata, memoria de orientación temporal, memoria remota semántica, memoria de evocación libre y memoria de evocación con pistas. Cada una de estas secciones contiene preguntas que hacen trabajar la memoria del paciente de diferente manera, y su estudio por separado y en un determinado orden hacen que el test tenga validez.

Como en el caso anterior, primero se deben añadir unos nodos necesarios para describir la sección. Sin embargo, el único de carácter obligatorio es el nodo *Title*. También acepta de manera opcional un nodo *Description* y un nodo *Result*. Por último acepta un nodo *Questions* que agrupará todas las preguntas de la sección.

```
    <Section>
        <Title>Fluidez Verbal</Title>
        <Description></Description>
        <Result />
        <Questions></Questions>
    </Section>
```

Figura 0.4: Detalles del nodo Section

Del nodo *Questions* colgará el penúltimo nodo más significativo de la estructura, *Question*. Se podrán añadir todos los nodos necesarios para describir cada una de las preguntas que conforman la sección. En este nodo se definirá todo lo que tenga que ver con la pregunta. El primero de ellos es el nodo *Display*, que sirve para agrupar a todos aquellos nodos y parámetros que tienen que ver únicamente con la pregunta y su presentación. Dentro de este nodo es obligatorio añadir de nuevo un

nodo *Title*, que será la pregunta que tendremos que plantear al paciente. Y también se puede añadir opcionalmente un nodo *Instructions*, que servirá como ayuda al evaluador y le indicará cómo debe proceder para llevar a cabo esta pregunta. Al mismo nivel que el nodo *Display*, se podrá añadir un nodo *Result* igual a los utilizados anteriormente. Y por último se debe añadir, en la mayoría de los casos obligatoriamente, un nodo *Answers* en el que se describirá cada una de las respuestas disponibles.

```
29
30  <Question>
31      <Display>
32          <Title>¿Cuántos gatos había?</Title>
33          <Instructions>Si falla, decirle la respuesta correcta.</Instructions>
34      </Display>
35      <Result />
36      <Answers></Answers>
37  </Question>
38
```

Figura 0.5: Detalles del nodo Question

Dentro de este nodo *Answers*, que agrupa todas las respuestas, nos encontramos con el nodo *Answer*. Este nodo representa la entrada que debe proporcionar el evaluador. Es obligatorio añadir un nodo de este tipo y se pueden añadir hasta tres. Dependiendo del tipo de respuesta que se indique, este nodo aceptará nodos hijos o no.

2.4.3. CARACTERÍSTICAS ESPECIALES

Una vez descrita de manera general la estructura, debemos centrarnos en algunos nodos que ofrecen opciones de configuración especial mediante parámetros, en algunos casos, opcional y en otros obligatoria.

Empezaremos por el nodo *Result*, que en el apartado anterior lo habíamos dejado pendiente. Este nodo se puede encontrar a nivel de test, de sección y de pregunta. Como se debe esperar, el resultado general del test es la combinación de los resultados de sus secciones, que a su vez son la combinación de los resultados de sus preguntas, y que a su vez son la combinación de sus respuestas. Se aceptan dos tipos de resultados, *integer* y *text* y se definen mediante el atributo *type*. En el caso de definir un tipo de resultado *integer*, será necesario definir un tipo de combinación necesario para calcular el resultado en la fase de evaluación del test. Esto se hace mediante el parámetro *combination*, que puede ser *sum* o *multiplication*. No ha sido necesario añadir ningún tipo de combinación más compleja porque no se han presentado en ninguno de los tests estudiados.

```
<Result type="text" />
```

Figura 0.6: Detalles tipo de resultado 'text'

```
<Result type="integer" combination="multiplication" />
```

Figura 0.7: Detalles tipo de resultado 'integer'

Además, este nodo acepta un tercer parámetro *attachment-required*, sólo a nivel de test, que sirve para indicar, que al finalizar el mismo es necesario adjuntar algún tipo de información complementaria. En esta versión de la aplicación el único tipo que acepta es *image* que será una foto capturada con la cámara del dispositivo móvil, útil para tests como el MMSE.

```
<Result type="integer" combination="sum" attachment-required="image" />
```

Figura 0.8: Detalles de resultado con adjunto requerido

El nodo *Question* acepta un parámetro *type* no obligatorio para especificar un tipo de pregunta especial. En esta versión se contempla el tipo especial *not-evaluate* que indica que una pregunta no va a formar parte de la evaluación del test. Normalmente se trata de preguntas de distracción o preguntas en las que se debe realizar una acción por la que se preguntará más adelante. Si no se indica un tipo determinado, la pregunta por defecto formará parte de la evaluación del test. Como es de esperar, en este tipo de preguntas no es necesario añadir los nodos *Result* y *Answers*.

```
<Question type="not-evaluate">
    <Display>
        <Title>Repita: cereza (R), hacha(R), elefante(R), piano(R), verde(R)</Title>
        <Instructions>Pídale que intente memorizar estas palabras y que esté atento/a.</Instructions>
    </Display>
</Question>
```

Figura 0.9: Detalles tipo de pregunta sin evaluación

La pregunta también puede tener un límite de tiempo recomendado para su ejecución, es por eso por lo que el nodo *Question* acepta otro parámetro *time-limit* el cual indica en segundos el tiempo indicado para una pregunta en concreto. Cuando tengamos que ejecutar esta pregunta, se mostrará un

temporizador que ayudará al evaluador a realizar la evaluación dentro de los parámetros definidos por el diseñador del test.

```
92
93  <Question time-limit="30">
94      <Display>
95          <Title>Quiero que me diga todos los nombres de hombres/mujeres que recuerde (sexo contrario)</Titl
96          <Instructions>Empezar siempre por el sexo contrario. Conceda 30 segundo y empiece a contar en cuan
97      </Display>
98  </Question>
99
```

Figura 0.10: Detalles tipo de pregunta con límite de tiempo

El nodo *Display* de una pregunta acepta un parámetro *media-required* para indicar que en la fase de configuración del test es necesario añadir algún tipo de multimedia. Ésta será utilizada para ejecutar una pregunta como las que aparecen en el Fototest, en las cuales es necesario mostrar una imagen al paciente y que identifique objetos que aparecen en ella. En esta versión de la aplicación sólo se acepta el tipo de multimedia *image*, que podrá ser añadida en la fase de configuración seleccionándola desde la galería de imágenes del dispositivo o utilizando la cámara.

```
71
72  <Question>
73      <Display media-required="image">
74          <Title>Enséñele la lámina con las fotos y pídale que nombre lo que aparece.</Title>
75          <Instructions>En caso de error, indíquele el nombre correcto y no marque la casilla corres
76      </Display>
77  <Question>
78
```

Figura 0.11: Detalla tipo de pregunta con imagen

Por último hay que mencionar el nodo *Answer*. Todas las combinaciones en su configuración definen los tipos de respuestas que acepta el sistema. El primero de todos los parámetros que hay que tener en cuenta es *type*, que indica el tipo de entrada que se espera y puede ser *integer*, *text*, *option* y *checkbox*.

El tipo *integer* espera un número entero que normalmente es el resultado de contar respuestas válidas o acciones que realiza el paciente. Por este motivo, si utilizamos este tipo de respuesta, también podremos añadir otro parámetro al nodo *Answer*, *input-helper*, que ayuda al evaluador a introducir la respuesta. En esta versión se ha implementado únicamente el ayudante *counter* que presentará en la interfaz gráfica dos botones para incrementar y disminuir el valor de la respuesta.

```
102
103   <Answer type="integer" input-helper="counter"></Answer>
104
```

Figura 0.12: Detalles respuesta de tipo 'integer'

El tipo *text* espera como entrada un texto. En tests o secciones con este tipo de preguntas no se produce una evaluación como tal, sino una recopilación de las diferentes respuestas que serán presentadas en el resultado como mismo se introduce.

```
102
103  <Answer type="text"></Answer>
104
```

Figura 0.13: Detalles respuesta de tipo 'text'

El tipo *checkbox* y *option* son prácticamente lo mismo, salvo que en el primero se permite seleccionar más de una de las opciones disponibles y en el segundo sólo una. En ambos casos hay que añadir una lista de nodos *Option* con un parámetro obligatorio *value* en cada uno, que será el valor que otorgará al resultado de la pregunta la selección de cada opción. Además, entre sus etiquetas se añade el texto que se presentará como respuesta.

```
108
109  <Answer type="checkbox">
110      <Option value="1">Cereza</Option>
111      <Option value="1">Hacha</Option>
112      <Option value="1">Elefante</Option>
113      <Option value="1">Piano</Option>
114      <Option value="1">Verde</Option>
115  </Answer>
116
```

Figura 0.14: Detalles respuesta tipo 'checkbox' y 'option'

Por último queda nombrar un parámetro más que acepta el nodo *Answer* que es *label*. Está especialmente diseñada para respuestas con más de una entrada necesarias y sirve para añadir una etiqueta a cada una de ellas y que el evaluador sepa en todo momento la información que se le está solicitando.

```
119
120  <Answers>
121      <Answer type="option" label="Frecuencia">
122          <Option value="0">Ausente</Option>
123          <Option value="1">Ocasionalmente</Option>
124          <Option value="2">A menudo</Option>
125          <Option value="3">Frecuentemente</Option>
126          <Option value="4">Muy frecuentemente</Option>
127      </Answer>
128      <Answer type="option" label="Gravedad">
129          <Option value="1">Leve</Option>
130          <Option value="2">Moderada</Option>
131          <Option value="3">Grave</Option>
132      </Answer>
133  </Answers>
134
```

Figura 0.15: Parámetro label en pregunta multirespuesta

2.5. APLICACIÓN

2.5.1. ESTRUCTURA

La lógica de la aplicación la hemos divido en tres tipos de clases: Aquellas que sirven para definir las estructuras de datos, las utilizadas para controlar el flujo de información entre otras clases y las clases auxiliares que recogen algún tipo de lógica específica.

En el primer grupo nos encontramos las clases *Test*, *Section*, *Question*, *SingleInputAnswer*, *MultiInputAnser* y *Option*, que juegan un papel crucial en la fase de parsing de la configuración de un nuevo test. Estas clases son utilizadas agrupar la información relativa a cada uno de los elementos que conforman el test. En este grupo también se incluye una clase importantísima para la ejecución, almacenamiento y recuperación de un test. Su nombre es *TestInstance*, y en ella se agrupan toda la información relativa a una instancia de test. Una instancia de test está formada principalmente por una estructura de test y un estado en el que se guardan las respuestas introducidas durante una evaluación.

En el segundo grupo encontramos dos clases: *AppController* y *TestController*. La primera de ellas se puede considerar la clase principal, de la cual dependen todas las demás. Esta clase se encarga de controlar todo lo que ocurre en la aplicación y el flujo de información que existe entre las otras clases. Entre sus acciones está, por ejemplo, la de recuperar una estructura de test de la memoria interna del dispositivo, entregársela al *TestController* con un estado vacío y delegar éste la presentación y control sobre su ejecución.

Por otro lado tenemos la clase ya mencionada *TestController*, que se encarga, entre otras cosas, de controlar el estado de ejecución de un test, así como generar su interfaz, ocuparse de la navegación entre preguntas y evaluarlo al final de su ejecución.

En el último grupo nos encontramos con clases que agrupan funcionalidades necesarias para la aplicación, que son *StorageManager* y *TestParser*. La primera de ellas recoge toda la lógica que tiene que ver con el almacenamiento, modificación y recuperación de los datos en la memoria interna del dispositivo móvil. En ella se define el tipo de almacenamiento que se utiliza, *LocalStorage* del estándar HTML5, y será explicada en detalle en un apartado posterior.

La segunda clase, *TestParser*, es la encargada de leer el fichero de configuración y transformarlo para que la aplicación pueda trabajar con la información que contiene. *TestParser* actúa en el momento en el que se proporciona un URL o una URI que ubique el fichero de configuración en la red o en la memoria del dispositivo. Los métodos utilizados para ello han sido jQuery.ajax() [25] que pasándole unos parámetros concretos obtiene el fichero XML utilizando una llamada HTTP asíncrona le realiza un parsing y lo devuelve un XML Document [1], que se puede explorar con Javascript. Este método, además de obtener el documento, nos da información acerca de la validez del mismo como XML en el momento del parsing, es decir, genera un error si no se considera un documento XML bien construido. En este caso, el proceso de agregar un nuevo test se detiene y se muestra al usuario un mensaje de error indicándole que el documento facilitado no respeta el estándar XML. En caso contrario, el documento pasa a una segunda fase en la que comprobamos si su estructura respeta nuestro protocolo de configuración. En esta fase se utilizan los métodos getElementsByTagName() [26] y getAttribute() [27] propios de Javascript, para acceder a los elementos y atributos del documento, respectivamente. Durante este proceso, se comprueba la validez de la estructura, teniendo en cuenta el XML Schema mostrado en el punto 2.4. Si se encuentra algún error se notifica al usuario, deteniendo el proceso si fuera necesario. A partir de aquí, si todo ha ido bien, se le pasa el modelo de test creado al *StorageManager* para que se encargue de guardarlo en la memoria del dipositivo.

Para terminar este apartado, presentamos un diagrama de clases para que de manera gráfica pueda comprenderse mejor la estructura cuyos componentes acabamos de exponer.

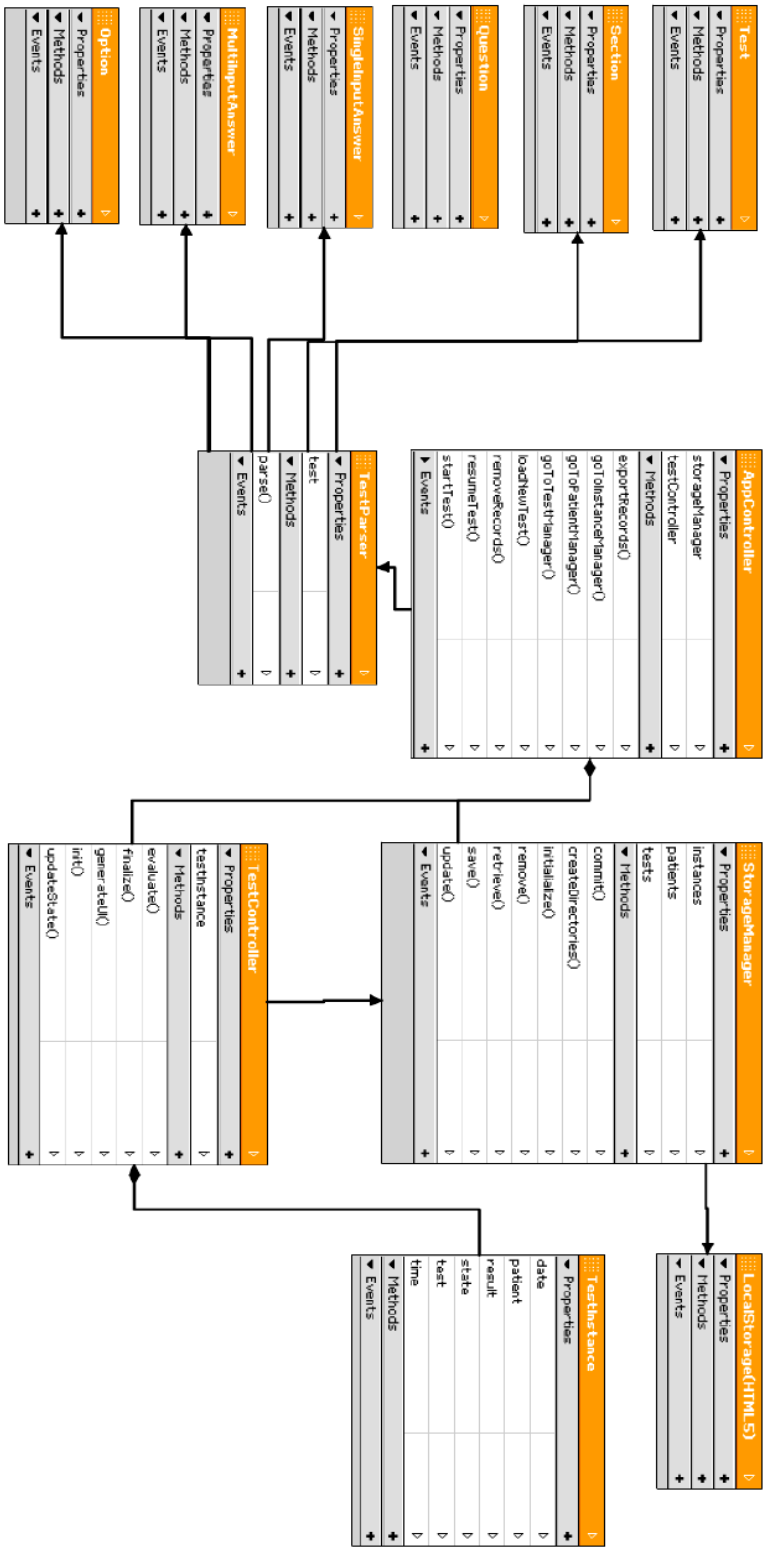

Figura 0.16: Diagrama de clases

2.5.2. INTERFACES GRÁFICAS

Este apartado servirá para mostrar las interfaces gráficas diseñadas, prestando especial atención a la correspondencia entre la configuración de un test y la interfaz que se genera para él.

Pantallas principales

Empezamos mostrando la pantalla principal, la pantalla para iniciar un test, y la pantalla del historial, de izquierda a derecha, figura 2.17.

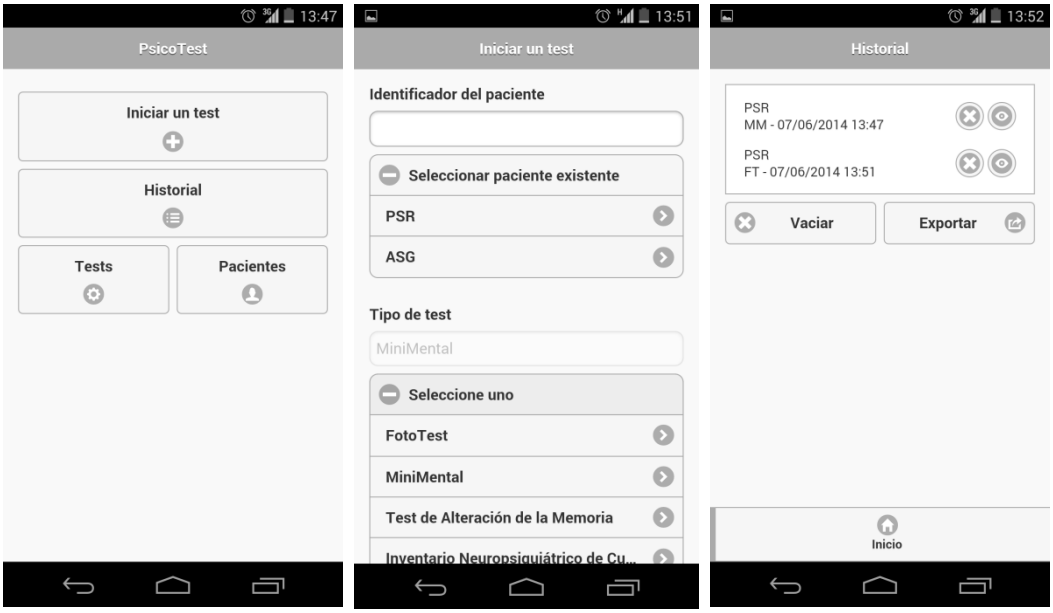

Figura 0.17: Pantallas principales

En la pantalla de inicio se muestran los accesos a las funcionalidades principales de la aplicación. En la pantalla para iniciar un test, el usuario deberá seleccionar un paciente existente o crear uno nuevo, y seleccionar el tipo de test que desea empezar. Por último, en la pantalla del historial, habrá una lista con los tests finalizados, con la posibilidad de visualizar el resultado, eliminar individual o masivamente y exportar a un fichero CSV.

Preguntas con imagen

Queremos mostrar también el tipo de pregunta que permite la inclusión de una imagen para mostrársela al paciente.

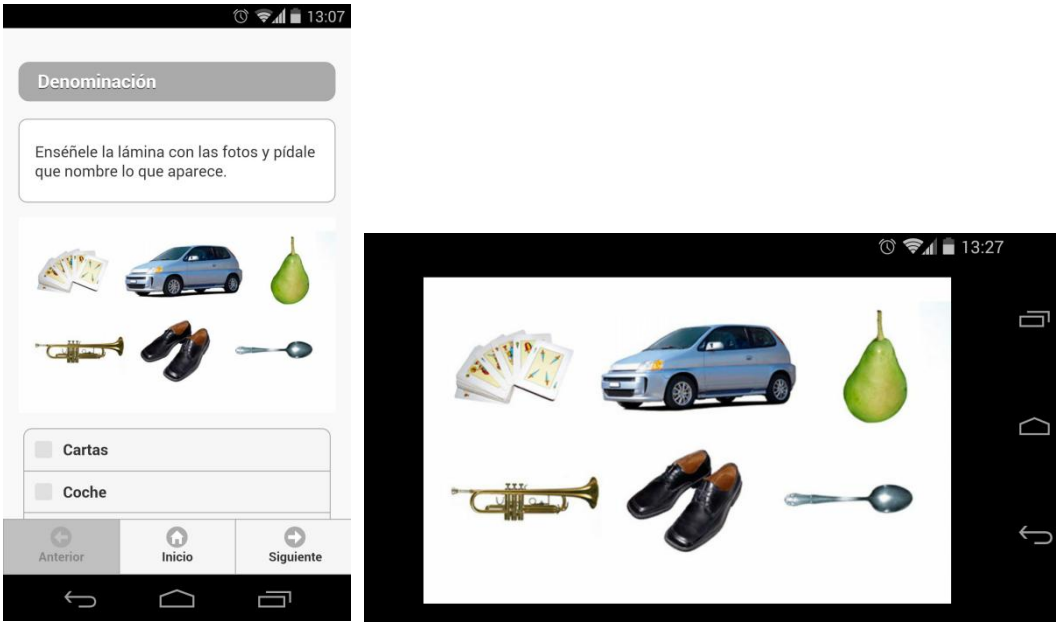

Figura 0.18: Pantalla tipo de pregunta con imagen

En la captura de la izquierda, figura 2.18, podemos ver la pregunta tal y como se mostraría, con una miniatura de la imagen. En el momento que debemos mostrarle al paciente la misma, con tocarla y girar el dispositivo la podremos ver a mayor resolución facilitando la interacción con el paciente, como se muestra en la captura de la derecha, figura 2.18.

Tipos de respuesta

A continuación veremos unas capturas con las diferentes interfaces gráficas para los tipos de respuestas y configuraciones especiales más interesantes que acepta el protocolo de configuración.

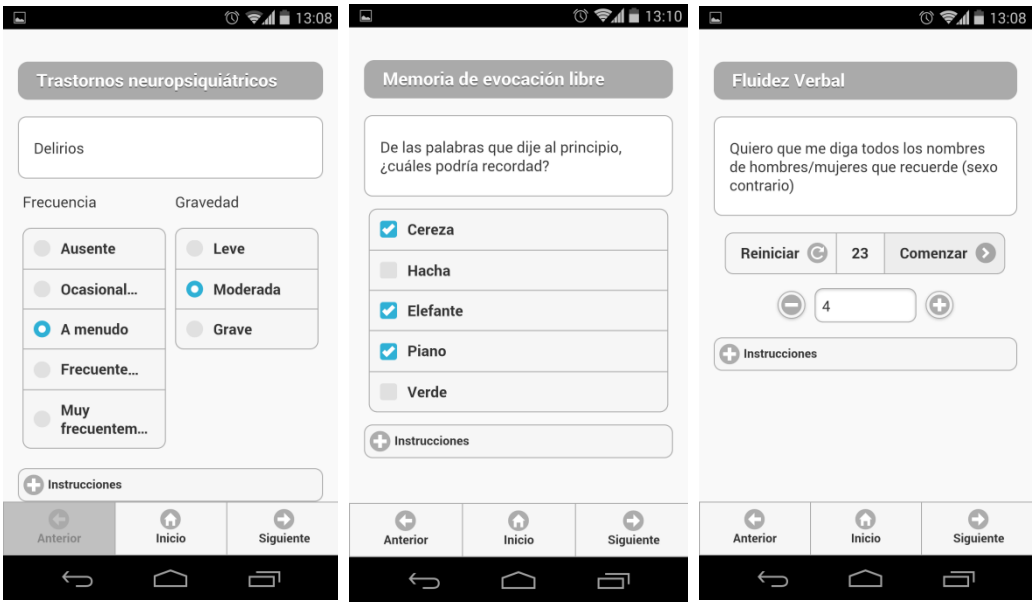

Figura 0.19: Pantallas tipos de respuesta

Empezando de izquierda a derecha, figura 2.19, en primer lugar tenemos un tipo de respuesta option, que en este caso tiene más de una entrada. En concreto en este test, el resultado de la pregunta se obtiene multiplicando los valores de cada una de las entradas. En la siguiente captura podemos ver el tipo de respuesta *checkbox* que nos permitirá elegir tantas opciones como sean precisas. Y en la última pantalla tenemos el tipo de respuesta *integer*, que utiliza el asistente *counter* para ayudar a contar, en este caso, los nombres de hombres/mujeres que diga el paciente. También podemos ver que hace uso del temporizador que se añade a la interfaz si la pregunta tiene límite de tiempo. Este temporizador inicialmente marca los segundos que se han definido como límite y una vez toquemos el botón *comenzar* la cuenta atrás empieza. Cuando ésta llegue a 0 se detiene y el dispositivo vibra para alertar al evaluador que el tiempo para esta pregunta ya ha terminado. Si por cualquier motivo es necesario reiniciar el temporizador, se podrá toca el botón *reiniciar*, la cuenta volverá a su tiempo original y se esperará a que el usuario de nuevo dé la orden de comenzar.

2.5.3. IMPORTACIÓN DE NUEVOS TESTS

Cuando se definieron los requisitos, indirectamente se hacía obligatorio proporcionar al usuario los métodos necesarios para la importación de ficheros XML y poder así cumplir con uno de los requisitos fundamentales, el de ampliar los tests disponibles mediante el protocolo de configuración que se diseñe.

El requisito básico planteado en este sentido fue el de poder importar ficheros del sistema de archivos del dispositivo. Pero en la fase de diseño, también se añadió la posibilidad de proporcionar al usuario un sistema para cargar ficheros utilizando una conexión a internet. En este apartado, explicaremos como funciona cada uno de ellos.

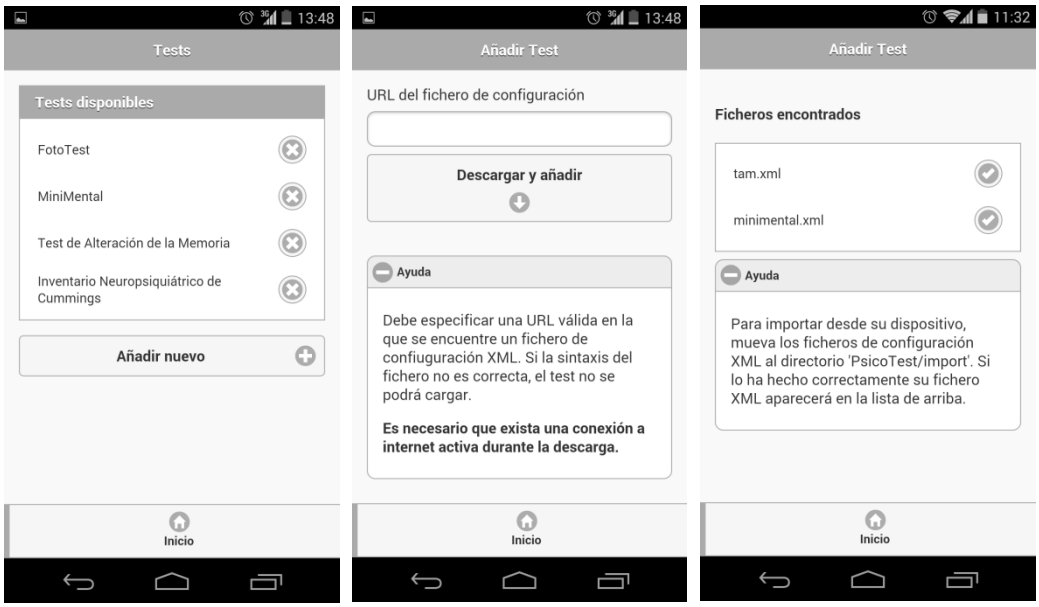

Figura 0.20: Pantallas importación de nuevos tests

La carga desde el sistema de archivos del dispositivo necesariamente hubo que limitarla un poco. Esto quiere decir, que para poder importar un test, es necesario ubicar su fichero de configuración en un directorio específico. Este directorio es creado, a la vez que otros, por la aplicación en su primera ejecución y es totalmente accesible al usuario siempre que disponga de una aplicación para explorar el sistema de archivos. Decidimos hacerlo así porque la API de PhoneGap no proporciona ningún explorador de archivos, sino órdenes de bajo nivel para listar, crear o eliminar, entre otras muchas, elementos del sistema de archivos. Por lo tanto, si se decidía que el usuario pudiera cargar el fichero desde cualquier ubicación habría que utilizar algún plugin de terceros o diseñar nuestro propio explorador. Las dos opciones fueron descartadas, la primera por los riesgos que supone en cuanto a soporte y seguridad, y la segunda porque se desvía bastante de los objetivos planteados para este proyecto.

La carga desde una URL fue añadida para ofrecer más opciones al usuario. En esta opción sólo es necesario proporcionar una URL válida que localice al fichero de configuración en la red, se descargará y se transformará la estructura de datos necesaria para poder utilizarlo.

2.5.4. ALMACENAMIENTO

El sistema elegido para almacenar los datos en la memoria del teléfono ha sido LocalStorage proporcionado por la API de HTML5. Hemos elegido este dada la poca complejidad de los datos a almacenar y las ventajas que aporta este sistema en cuanto a facilidad de implementación y compatibilidad en todas las plataformas soportadas por PhoneGap.

LocalStorage no es más que un sistema de almacenamiento clave-valor, por lo tanto, es necesario guardar cadenas de texto que mantengan intactas las propiedades que definen a los datos y que puedan ser recuperadas y convertidas a objetos manejables desde Javascript. Para esto utilizamos dos métodos, uno es *stringify* de JSON, que convierte un objeto Javascript en una cadena de texto JSON. Y el segundo, utilizado para recuperar la información, es el método *parseJSON* de JQuery, que hace justo lo contrario que el anterior, convierte una cadena JSON en un objeto Javascript.

Con estos dos métodos y con los métodos propios de LocalStorage hemos construido nuestro sistema de almacenamiento que nos permite la serialización y deserialización de los objetos que utilizamos en nuestra aplicación.

2.5.5. EXPORTACIÓN DE RESULTADOS

Uno de los requisitos que se plantearon para nuestra aplicación es la posibilidad de exportar el historial de tests que hayamos realizado utilizando la misma. Desde el principio se propuso hacerlo en algún formato, siempre pensando en un fichero de texto plano, independientemente de cuál se eligiera al final. Surgió un problema después de la reunión con el experto en Neurología consultado, y es que al añadir la posibilidad de adjuntar imágenes al resultado, el fichero de salida no podía ser texto plano.

En busca de posibles soluciones valoramos la posibilidad de utilizar el formato PDF, utilizando algunas de las librerías en Javascript que encontramos para crear fichero de este tipo. Sin embargo, no conseguimos integrarlas en nuestra aplicación para que funcionaran de manera correcta por lo que se descartó.

La solución que finalmente adoptamos fue la que se propuso de inicio, y elegimos el formato de fichero CSV para la tarea. La manera de abordar el problema de la imagen, fue proporcionar, en una de las columnas la ruta en la que se encuentran las imágenes dentro del sistema de archivos del dispositivo.

El fichero que se genera en la exportación se almacena en la estructura de directorios de la aplicación que se crea al iniciarla por primera vez. No damos la posibilidad de que el usuario seleccione una ruta cualquiera, ya que la API de PhoneGap no incluye un explorador de archivos. Habría que utilizar algún plugin de terceros, con el riesgo que ello supone, o crear un explorador desde cero, lo cual añadiría mucha carga de trabajo al proyecto y consideramos que se aleja del objetivo del mismo.

2.5.6. TESTS CONFIGURADOS

Una de las actividades marcadas en el plan de trabajo fue la de configurar con nuestro protocolo algunos tests que pudieran ser interesantes por sus características diferenciadoras.

Para esta actividad hemos seleccionado cuatro de los test explicados en el apartado 1.3.1. El primero de ellos, el T@M, lo hemos seleccionado por su estructura claramente dividida en secciones. El segundo fue el MMSE, en este caso se seleccionó por la característica especial de añadir una foto al resultado. El tercero, el Fototest, fue seleccionado porque incluye una pregunta que necesita que se le añada una imagen. Y por último, seleccionamos el Inventario Neuropsiquiátrico de Cummings, en este caso por presentar un tipo de combinación diferente para calcular el resultado.

Todos los tests mencionados son compatibles con nuestra aplicación y están incluidos en la versión actual de la misma. Hay que destacar, que algunos de estos tests están protegidos por derechos de autor, y que han sido incluidos en el prototipo de la aplicación únicamente para comprobar el funcionamiento de la misma y la utilidad del protocolo de configuración diseñado.

CAPITULO 3: CONCLUSIONES

3.1. CONCLUSIONES

En las fases iniciales de este proyecto nos habíamos planteado unos objetivos básicos y otros complementarios que debíamos completar para dar por satisfactorio la ejecución de este proyecto. Llegados a este punto, es momento de repasar esos objetivos y apuntar si los hemos llevado a cabo con éxito o no.

El primero de ellos era el de analizar los tests utilizados en neuropsicología y estudiar su estructura para poder diseñar un protocolo de configuración adaptable a la gran mayoría de los tests estudiados. Hemos consultado a un experto en este campo, quién nos ha proporcionado alguno de los tests más usuales, y alguno que otro más inusual, y hemos conformado un amplio conjunto de tests sobre los que trabajar. De este conjunto, hemos extraído los tests con características un poco más peculiares y que presentaban algún tipo de dificultad de implementación, para tenerlos en cuenta a la hora de diseñar el protocolo de configuración. Por lo tanto consideramos que este objetivo ha sido cumplido satisfactoriamente y que gracias a ello se han creado las bases sobre las que se apoya el diseño del protocolo de configuración.

Otro de los objetivos planteados era el de analizar las plataformas móviles existentes en la actualizar para elegir aquella que se adaptara mejor a nuestras necesidades. También consideramos que este objetivo ha sido cumplido y nos ha ayudado a tener una visión general sobre las opciones existentes y elegir la que finalmente creímos que era mejor para nosotros. Elegir utilizar PhoneGap dentro de la plataforma Android ha sido satisfactorio, ya que no nos hemos encontrado con grandes problemas a la hora de implementar las funcionalidades que se requerían en la aplicación.

El último de los objetivos enfocado a estudiar el estado actual del tema tratado en este proyecto, era el de analizar la oferta existente de aplicaciones similares a la propuesta. Nos ha sido útil este análisis para observar cómo se resolvían en estas aplicaciones algunos de los problemas que se nos han presentado en la fase de diseño y desarrollo.

En cuanto a los objetivos principales que nos habíamos planteado, el primero de ellos era el de desarrollar una aplicación móvil que sirviera para la realización y evaluación de tests que pudiera sustituir al método que se utiliza en la actualidad. Estamos satisfechos con los resultados obtenidos respecto de

este objetivo, ya que hemos conseguido implementar una aplicación capaz de realizar tests, evaluarlos, almacenar un historial y demás requisitos marcados. Es una aplicación actualmente funcional y se podría sacar al mercado en cuestión de semanas, no sin antes revisar algunas cuestiones legales respecto del copyright de los tests implementados. Lo único que no nos deja satisfechos del todo es la exportación del historial. Nos hubiera gustado poder ofrecer la posibilidad de exportar, como se barajó en su momento, en formato PDF. Así el usuario podría visualizar de manera más cómoda en un PC o imprimirlos si quisiera.

El otro objetivo principal planteado, el de diseñar un protocolo de configuración para la importación de nuevos tests, también lo consideramos cumplido. Hemos conseguido extraer los elementos más significativos del conjunto de tests con los que hemos trabajado y hemos sabido diseñar un protocolo de configuración bastante flexible. El protocolo se adapta perfectamente a los tests del campo estudiado y también es válido para muchos otros tests de ramas similares, como podría ser la psicología.

En cuanto a lo que me ha aportado la participación en este proyecto a nivel personal he de destacar dos cosas. La primera de ella es la experiencia tan enriquecedora que supone desarrollar un proyecto informático desde sus fases iniciales hasta las finales. He aprendido la importancia de establecer unas bases sólidas sobre la que poder apoyarme para ofrecer soluciones a los problemas planteados en la propuesta del proyecto. Y la segunda ha sido la adquisición de conocimientos más profundos sobre el desarrollo de aplicaciones móviles y sobre las tecnologías utilizadas en el proyecto: PhoneGap, Javascript y JQuery.

CONCLUSIONS

In the initial stages of this project we had proposed some basic objectives and others complementary that we must complete to take for satisfactory the project. At this point, it is time to review those objectives and check if we had implemented successfully or not.

The first was to analyze the tests used in neuropsychology and study its structure in order to design a customizable configuration protocol to the most of tests studied. We have consulted an expert in this area, who has provided some of the most common tests, and some other more unusual, and we have assembled a comprehensive set of tests. From this set, we extracted the peculiar tests and which showed some kind of difficulty in the implementation,

to take them into account in the design of the configuration protocol. Therefore we consider that this objective has been achieved successfully and as a result it have been created the bases on which rests the design of the configuration protocol.

Another objective was to analyze existing mobile platforms to choose one that is best adapted to our needs. We also consider that this goal has been accomplished and helped us to have an overview of the different options and choose the one that was best for us. Using PhoneGap in the Android platform has been satisfactory, because we have not find big problems when implementing the functionalities that are required in the application.

The last of the objectives, which we considered completed, was to analyze the existing offer of similar applications to the proposal. This analysis have been useful to notice how these applications solved some problems presented to us in the design and development phase.

As for to the main objectives we had proposed, the first of them was to develop a mobile application that would serve for the realization and evaluation of tests that could replace the method used today. We are pleased with the results obtained about this objective, as we have managed to implement an application able to perform tests, evaluate, store a record and other requirements. It is a functional application and we could publish in a few weeks, but not before reviewing some legal issues concerning the copyright of the implemented tests. The only thing we are not satisfied at all is the export of records. We would like to offer the ability to export in PDF format. So the user could display more comfortably on a PC or print them if he wanted.

The other main objective, to design a configuration protocol for the importation of new tests, we also consider reached. We have managed to extract the most significant elements of the set of tests with which we have worked and we have learned to design a flexible configuration protocol. The protocol is perfectly adapted to the studied tests and it is also valid for many other tests of similar areas, like psychology.

The participation in this project has given me an enriching experience that involves developing a software project from its initial stages to the final. I have learned the importance of establishing a solid basis on which support me to provide solutions to the problems raised in the project proposal. Also, I have

acquired deeper knowledge about the development of mobile applications and technologies used in the project: PhoneGap, JQuery and Javascript.

3.2. TRABAJOS FUTUROS

También me gustaría hablar en este capítulo de temas que no han quedado del todo resueltos en este proyecto, así como ideas que han ido surgiendo y que no se han podido presentar.

Como comentamos en el apartado anterior, nos hubiera gustado que la exportación del historial fuera en un formato del tipo PDF. Dado el fracaso ocurrido al intentar utilizar librerías Javascript para tal fin, se abre la posibilidad de trabajar con algún lenguaje de los llamados "lenguajes del lado del servidor", como Java o PHP para suplir las carencias del primero. En este sentido, para seguir manteniendo la característica multiplataforma de la aplicación, se apostaría por un sistema en la nube, que sirviera como apoyo y ofreciera esta funcionalidad. Aprovechando ese sistema en la nube, se podría implementar una especie de red social para compartir ficheros de configuración de tests, pudiéndolos descargar o cargar desde la propia aplicación. Así, de forma colaborativa, se generaría una gran base de tests disponibles para todos los usuarios de la aplicación.

Por otra parte, nos gustaría mejorar el sistema de almacenamiento utilizado, abandonar el uso del LocalStorage de HTML5 y haciendo uso de un sistema gestor de bases de datos relacionales. De esta manera podremos estar seguros que el crecimiento del volumen de datos almacenados en la aplicación no supondrá un problema.

Otro de los trabajos que se podrían realizar en el futuro es trabajar en la seguridad de los datos que maneja la aplicación. En la versión actual, sólo se puede guardar una cadena alfanumérica que identifique internamente al paciente y se recomienda que ésta no sea un dato de carácter personal. En cualquier caso no es la mejor solución para este problema, y se puede plantear la posibilidad de añadir seguridad a los datos tratados, añadiendo los sistemas necesarios para ello, como claves de acceso, encriptación, etc. De esta manera, podríamos guardar con total seguridad una ficha de paciente con más información, que nos ayudaría a utilizar la aplicación de una manera más intuitiva y sencilla.

Por último, se podría plantear la realización de estudios más profundos sobre las cuestiones legales referentes a la utilización de los tests que en este proyecto se mencionan, de cara a una posible comercialización de la herramienta.

FUTURE WORKS

I would also like to speak in this chapter about issues that have not been entirely resolved in this project, as well as ideas that have emerged and could not be present. As discussed in the previous section, we wished that the export of records was in a format of PDF type. Given the failure occurred when attempting to use Javascript libraries, it considered the possibility of working with a language called "server side languages" such as Java or PHP. In this sense, to keep application cross-platform feature, will bet on a system in the cloud, to serve as support and offer this functionality. Taking advantage of this system in the cloud, it could implement a kind of social network to share configuration files tests, being able to download or upload from the application. So, in a collaborative way, it would generate a large base of available tests to all users of the application.

Moreover, we would like to improve the storage system used, stop using LocalStorage HTML5 and using a database management system. This way we can be sure that the volume growth of data stored in the application will not be a problem. In the future we could work on the security of the data handled by the application. In the current version, you can only store one alphanumeric string that identifies the patient internally and it is recommended that this is not a personal data. In any case it is not the best solution for this problem, and we can consider the possibility of adding security to processed data by adding the necessary systems, as passwords, encryption, etc.. In this way, we could safely save a patient file with more information that would help us to use the application in a more intuitive and easy way.

Finally, it could propose the development of deeper studies about legal issues concerning to the use of the tests mentioned in this project, facing a possible publication.

Referencias

[1] «W3C - XML,» [En línea]. Available: http://www.w3.org/TR/REC-xml/. [Último acceso: Marzo 2014].

[2] «Portal de Salud Mental - Test del Reloj,» [En línea]. Available: http://www.portaldesaludmental.com.ar/index.php/neurocognitivo/41-testsdeterioro/54-reloj.html. [Último acceso: Marzo 2014].

[3] «Neuropsicol - Cuestionario de ACTIVIDAD FUNCIONAL de PFEFFER,» [En línea]. Available: http://www.neuropsicol.org/Protocol/faqofe.pdf. [Último acceso: Marzo 2014].

[4] «Hvn - Test de Yesavage,» [En línea]. Available: http://www.hvn.es/enfermeria/ficheros/test_de_yesavage.pdf. [Último acceso: Marzo 2014].

[5] «Eurotest,» [En línea]. Available: http://www.eurotest.es. [Último acceso: Marzo 2014].

[6] «Fototest,» [En línea]. Available: http://www.fototest.es/. [Último acceso: Marzo 2014].

[7] «Elsevier - Invetario Neuropsiquiátrico,» [En línea]. Available: http://zl.elsevier.es/es/revista/revista-psiquiatria-salud-mental--286/perfil-psicopatologico-pacientes-traumatismo-craneoencefalico-evaluados-mediante-90141474-originales-2012. [Último acceso: Marzo 2014].

[8] «Wikipedia - MMSE,» [En línea]. Available: http://es.wikipedia.org/wiki/Mini-mental_state_examination. [Último acceso: Marzo 2014].

[9] «Recorda - Test de alteración de la memoria,» [En línea]. Available: http://www.recorda.info/u/uploads/File/TM.pdf.

[10] «Celularis - Sistemas Operativos Móviles 2013,» [En línea]. Available: http://www.celularis.com/mercado/sistemas-operativos-moviles-2013/.. [Último acceso: Diciembre 2013].

[11] «Android,» [En línea]. Available: http://www.android.com/.

[12] «Eclipse,» [En línea]. Available: http://www.eclipse.org/.

[13] «W3C - Html5,» [En línea]. Available: http://www.w3.org/TR/html5/.

[14] «W3C - CSS,» [En línea]. Available: http://www.w3.org/TR/CSS/.

[15] «W3C - Javascript,» [En línea]. Available: http://www.w3schools.com/js/DEFAULT.asp.

[16] «AppCelerator - Titanium,» [En línea]. Available:

http://www.appcelerator.com/titanium/. [Último acceso: Diciembre 2013].

[17] «Phonegap,» [En línea]. Available: http://phonegap.com/. [Último acceso: Diciembre 2013].

[18] «JQueryMobile,» [En línea]. Available: http://jquerymobile.com/. [Último acceso: Diciembre 2013].

[19] «Rotator Survey,» [En línea]. Available: http://www.rotatorsurvey.com/. [Último acceso: Diciembre 2013].

[20] «Play Google - Mini Mental Test,» [En línea]. Available: https://play.google.com/store/apps/details?id=appinventor.ai_rafaelandrademd.MiniMentalOtimizado&hl=es. [Último acceso: Mayo 2014].

[21] «Play Google - Geriatric Depresion J Yesavage,» [En línea]. Available: https://play.google.com/store/apps/details?id=elias.sys.indices.yesabage&hl=es. [Último acceso: Mayo 2014].

[22] «Play Google - Nurse Test,» [En línea]. Available: https://play.google.com/store/apps/details?id=com.redebersalud.nursetest&hl=es. [Último acceso: Mayo 2014].

[23] "Moodle - Questions," [Online]. Available: http://docs.moodle.org/27/en/Questions.

[24] «W3C - XML Schema,» [En línea]. Available: http://www.w3schools.com/Schema/. [Último acceso: Marzo 2014].

[25] «JQuery - ajax,» [En línea]. Available: http://api.jquery.com/jquery.ajax/. [Último acceso: Marzo 2014].

[26] «W3C Schools - getElementsByTagName,» [En línea]. Available: http://www.w3schools.com/dom/met_document_getelementsbytagname.asp. [Último acceso: Marzo 2014].

[27] «W3C Schools - getAttribute,» [En línea]. Available: http://www.w3schools.com/dom/met_element_getattribute.asp. [Último acceso: Marzo 2014].

[28] "PublMed," [Online]. Available: http://www.ncbi.nlm.nih.gov/pubmed/1159263. [Accessed Marzo 2014].

Apéndice: XML Schema protocolo de configuración

```
<?xml version="1.0"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema">
 <xs:element name="Test">
  <xs:complexType>
   <xs:sequence>
    <xs:element name="Title" type="xs:string" maxOccurs="1" minOccurs="1" />
    <xs:element name="ShortTitle" type="xs:string" maxOccurs="1"
minOccurs="1" />
    <xs:element name="Description" type="xs:string" maxOccurs="1"
minOccurs="1" />
    <xs:element name="Result" maxOccurs="1" minOccurs="0">
     <xs:complexType>
      <xs:attribute name="type" type="xs:string" use="required" />
      <xs:attribute name="combination" type="xs:string" use="required" />
      <xs:attribute name="attachment-required" type="xs:string" />
     </xs:complexType>
    </xs:element>
    <xs:element name="Section" minOccurs="1" maxOccurs="unbounded">
    <xs:complexType>
     <xs:sequence>
       <xs:element name="Title" type="xs:string" maxOccurs="1" minOccurs="1"
/>
       <xs:element name="Description" type="xs:string" maxOccurs="1"
minOccurs="0" />
       <xs:element name="Result" maxOccurs="1" minOccurs="0">
        <xs:complexType>
         <xs:attribute name="type" type="xs:string" use="required" />
         <xs:attribute name="combination" type="xs:string" use="required" />
        </xs:complexType>
       </xs:element>
       <xs:element name="Questions" maxOccurs="1" minOccurs="1">
        <xs:complexType>
         <xs:sequence>
          <xs:element name="Question" maxOccurs="unbounded">
              <xs:complexType>
               <xs:sequence>
            <xs:element name="Display" maxOccurs="1" minOccurs="1">
            <xs:complexType>
```

```
            <xs:sequence>
             <xs:element name="Title" type="xs:string" maxOccurs="1"
minOccurs="1" />
             <xs:element name="Instructions" type="xs:string" maxOccurs="1"
minOccurs="0" />
            </xs:sequence>
            <xs:attribute name="media-required" type="xs:string" />
           </xs:complexType>
          </xs:element>
          <xs:element name="Result" maxOccurs="1" minOccurs="0">
           <xs:complexType>
            <xs:attribute name="type" type="xs:string" use="required" />
            <xs:attribute name="combination" type="xs:string" use="required"
/>

           </xs:complexType>
          </xs:element>
          <xs:element name="Answers" minOccurs="0">
           <xs:complexType>
            <xs:sequence>
             <xs:element name="Answer" maxOccurs="3" minOccurs="1">
              <xs:complexType>
               <xs:sequence>
                <xs:element name="Option" maxOccurs="unbounded"
minOccurs="0">
                 <xs:complexType>
                  <xs:simpleContent>
                   <xs:extension base="xs:string">
                    <xs:attribute name="value" type="xs:string"
use="required" />
                    <xs:attribute name="input-helper" type="xs:string" />
                   </xs:extension>
                  </xs:simpleContent>
                 </xs:complexType>
                </xs:element>
               </xs:sequence>
               <xs:attribute name="type" type="xs:string" use="required" />
               <xs:attribute name="label" type="xs:string" />
               <xs:attribute name="input-helper" type="xs:string" />
              </xs:complexType>
             </xs:element>
```

```
            </xs:sequence>
           </xs:complexType>

                </xs:element>
              </xs:sequence>
              <xs:attribute name="time-limit" type="xs:integer" />
              <xs:attribute name="type" type="xs:string" />
             </xs:complexType>
            </xs:element>
           </xs:sequence>
          </xs:complexType>
         </xs:element>
        </xs:sequence>
       </xs:complexType>
      </xs:element>
     </xs:sequence>
    </xs:complexType>
   </xs:element>
</xs:schema>
```