Universidad de Concepción
Departamento de Ingeniería Informática y Ciencias de la Computación

DESARROLLO DE TESTS DE APRENDIZAJE, MEMORIA E INDUCTORES DE ESTRÉS PARA APLICACIÓN MÓVIL STRESSMAPP

POR

Jorge Santos Vallejos

Memoria de Título presentada a la Facultad de Ingeniería de la Universidad de Concepción para optar al título de Ingeniero(a) Civil Informático.

Profesor Patrocinante
Gonzalo Rojas Durán

Profesoras Co-patrocinantes
Pamela Guevara Alvez
Kristin Schmidt

Agosto de 2023
Concepción (Chile)

Resumen

El equipo multidisciplinario liderado por la Dra. Kristin Schmidt, del Departamento de Psiquiatría y Salud Mental, de la Facultad de Medicina de la Universidad de Concepción busca desarrollar la aplicación móvil "StressMApp" (Stress Map Application). Esta aplicación recopila datos sobre el estrés en estudiantes a través de autoinformes y seguimiento de comportamiento. La app consiste en 3 test en formato de juego que evalúan el rendimiento cognitivo y provocan estrés en los participantes. Estas pruebas van acompañadas de formularios antes y después de los juegos. Esto, junto al comportamiento capturado de los juegos, es comunicado a un servidor back-end que permite almacenar esta información en una base de datos para posterior análisis.

Lo que concierne este documento es específicamente hacia el desarrollo de los 3 test, donde dichas pruebas incluyen un test de memoria llamado "N-Back", un test de "Imágenes y Decisiones" que induce estrés usando imágenes aversivas, y un test de "Exploración y Predicción" donde el usuario debe ir descubriendo y reafirmando las reglas que dominan el juego. El desarrollo se realiza de manera iterativa e incremental, con el uso del framework "React Native" para implementar el primer prototipo funcional con un alcance inicialmente dirigido hacia dispositivos Android.

El equipo evaluó el funcionamiento de los test mediante pruebas de funcionalidad y se obtuvo una evaluación favorable en lo que respecta a las funcionalidades y expectativas. Se cumple con los requisitos y especificaciones analizadas de cada test y solo se considera evaluar posibles cambios en duraciones de tiempo y cantidades de ensayos. Sin embargo, las pruebas con estudiantes aún no se han realizado debido a trámites pendientes con el Comité de Ética de la Facultad de Medicina. El trabajo futuro incluye ajustar la duración de las pruebas y realizar pruebas piloto con participantes reales para continuar con la investigación y posiblemente lanzar la aplicación tanto para sistemas operativos Android y iOS en sus respectivas tiendas virtuales.

Índice

1. Introducción

El estrés (en contextos de mala adaptación) está relacionado con un mayor riesgo, aparición, mantenimiento y recaída de una variedad de enfermedades físicas y mentales. Por otro lado, ciertos patrones de respuesta al estrés definen una mejor resiliencia, lo que permite a las personas navegar con éxito las circunstancias negativas y mantener el bienestar. Basado en un enfoque psiquiátrico transdiagnóstico de la salud y de enfermedades mentales, el estrés puede verse como un impulsor clave en materias de salud y enfermedad, y su estudio es la base del trabajo propuesto por uno de los 25 proyectos ganadores del último concurso de financiamiento interno VRID Investigación Multidisciplinaria administrado por la Vicerrectoría de Investigación y Desarrollo (VRID) de la Universidad de Concepción [1], titulado "StressMApp: Desarrollo de una aplicación móvil para identificar marcadores cognitivos relacionados con el estrés, y sus relaciones con medidas clínicamente relevantes de autorreporte en una muestra de estudiantes chilenos".

En el presente documento, se detalla la Memoria de Título (MT) que consiste en colaborar en este equipo multidisciplinario mediante el desarrollo desde una perspectiva front-end de la aplicación móvil "StressMApp", la cual consiste en una serie de cuestionarios y test tipo juego de exploración y aprendizaje, y de memoria, incluyendo el reporte a una base de datos de los respectivos resultados y comportamientos. Esto con el fin de que el equipo detrás de este proyecto cuente con un prototipo funcional de una herramienta clave en su investigación del estudio del estrés y su relación con el rendimiento cognitivo en estudiantes chilenos.

1.1. Antecedentes generales del problema

El equipo multidisciplinario liderado por la Dra. Kristin Schmidt, del Departamento de Psiquiatría y Salud Mental, de la Facultad de Medicina, requiere comenzar el desarrollo de su aplicación móvil "StressMApp" (stress map application). Este software permitirá recabar información en forma de auto-reportes y captura de comportamiento en un contexto de investigación médica sobre el estrés en estudiantes.

La aplicación móvil consiste de 3 test en formato juego, con factores que evalúan el rendimiento cognitivo e inducen al estrés del participante. Previo y posterior a los juegos, se deben completar una serie de formularios para conocer el estado del participante. Esto, junto al comportamiento capturado del juego, deben ser enviados a un back-end que permita almacenar esta información en una base de datos para posterior análisis.

Los test en cuestión son basados en pruebas que se realizan en espacios controlados (test de laboratorio). Estos fueron previamente estudiados y evaluados por los investigadores de este proyecto, considerándolos factibles de realizar remotamente. Es decir, que mediante un teléfono inteligente el sujeto en estudio, y fuera de condiciones de laboratorio, pueda ser inducido a estrés y reportar efectivamente su estado.

A continuación, se muestran las tareas desde un diseño preliminar de la interfaz. Cabe mencionar que cuando se habla de estado “controlable” o “incontrolable”, corresponde a cuando las acciones del usuario tienen influencia en los resultados del juego o no, es decir, consecuencias determinísticas o aleatorias, respectivamente.

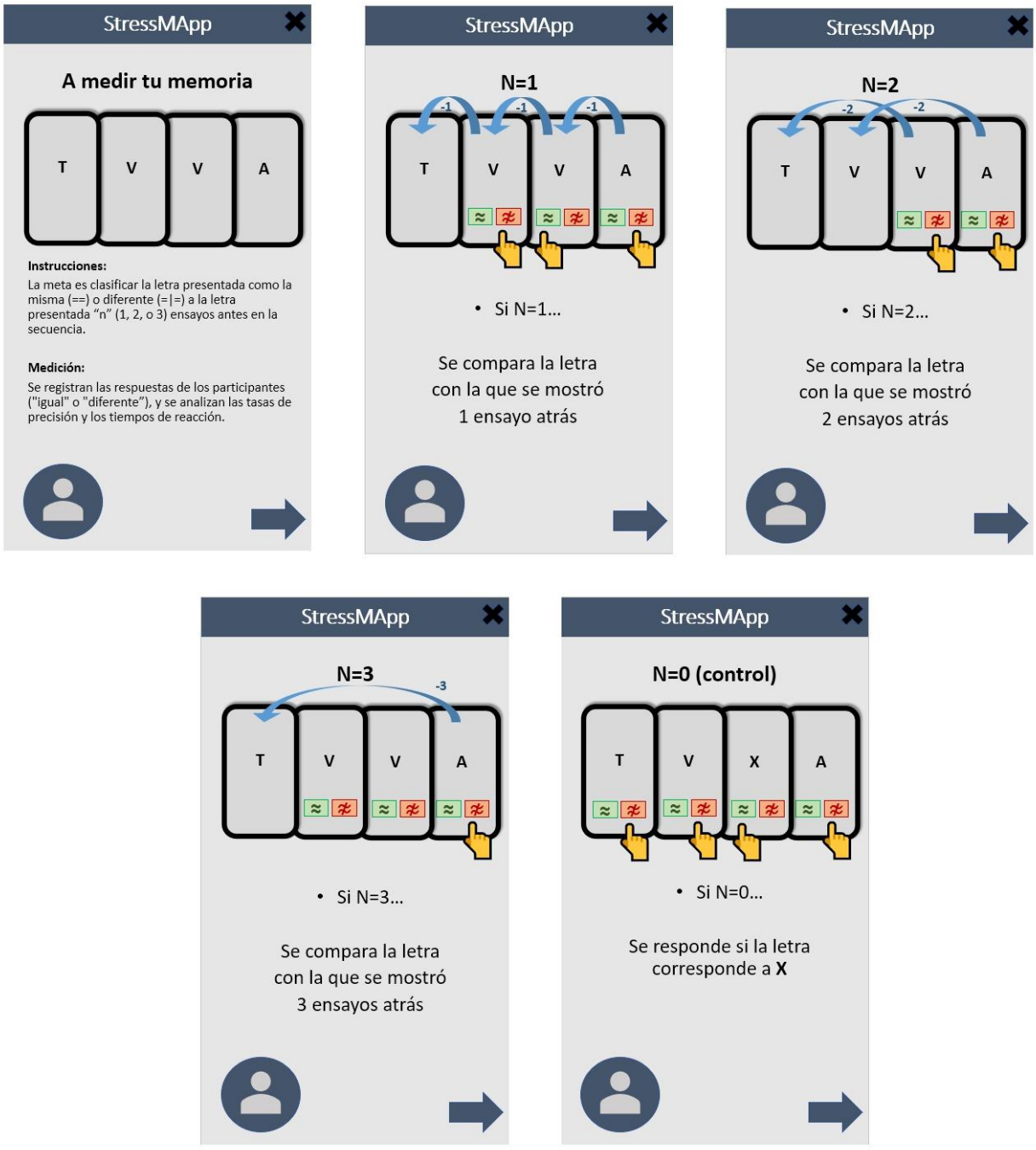

Figura 1. Test 1: Tarea N-Back, correspondiente a una prueba de memoria, donde el usuario deberá recordar la letra mostrada en “N” ensayos atrás.

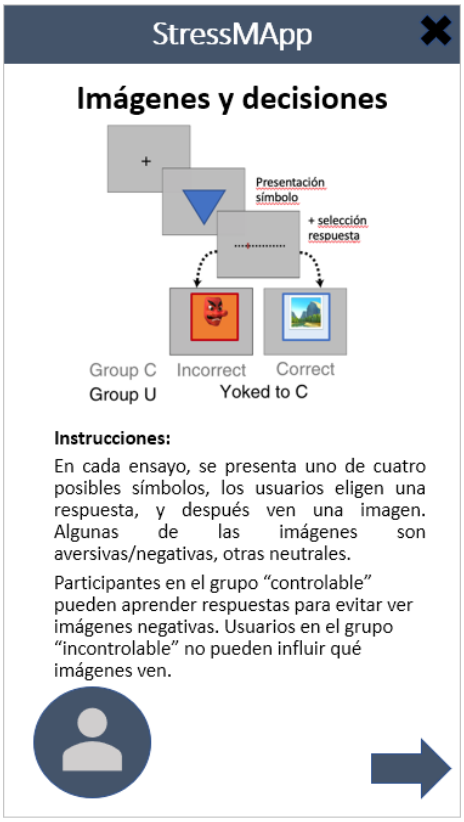

Figura 2. Test 2: Prueba de imágenes y decisiones, la cual es más sugerente a la inducción de estrés al incluir imágenes aversivas/negativas del IAPS (International Affective Picture System) en base a decisiones de símbolos que escojan los usuarios.

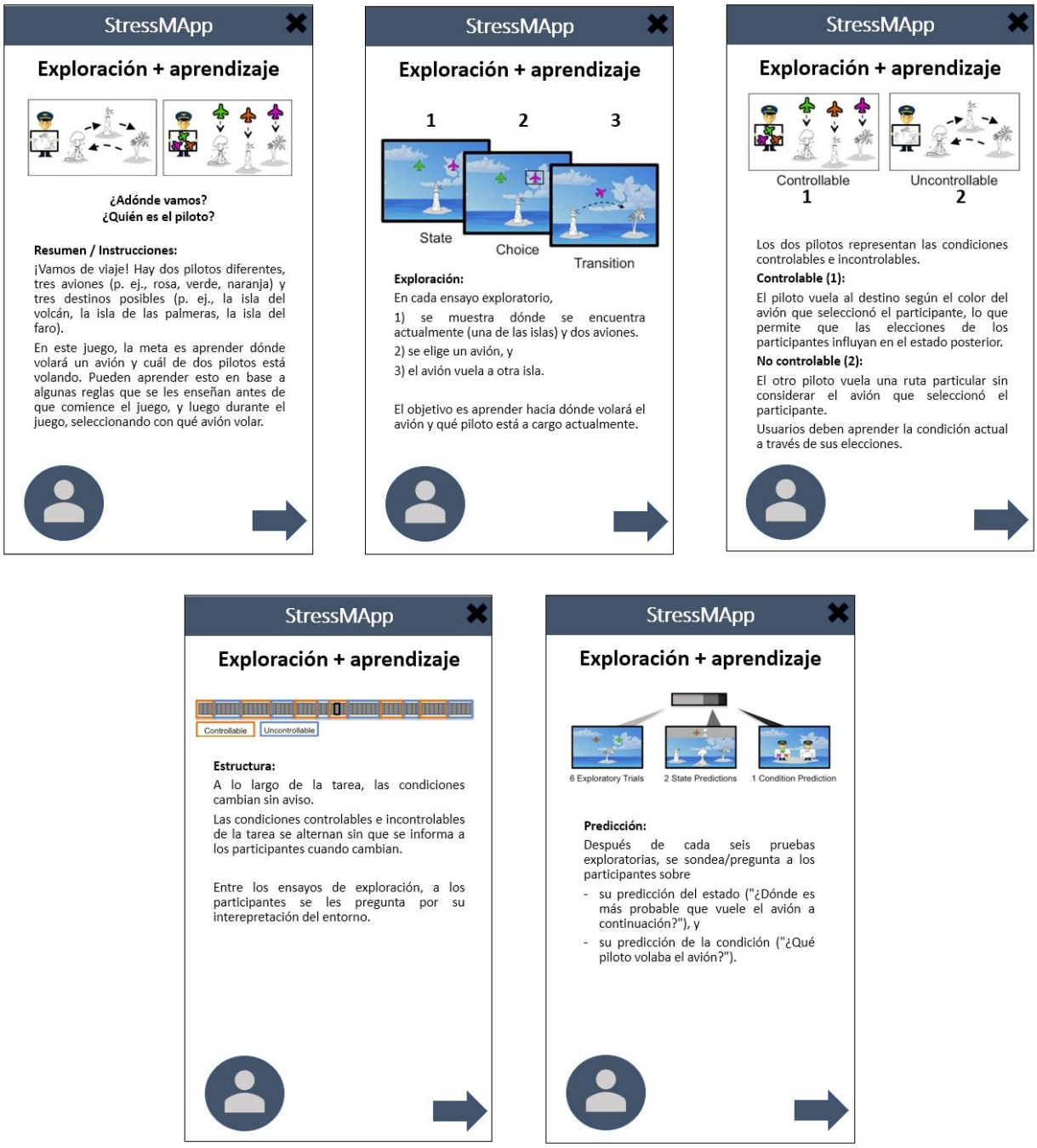

Figura 3. Test 3: Prueba de exploración y aprendizaje, en el cual el usuario debe aprender hacia dónde volará el avión y con cuál piloto, basándose en la exploración de algunas reglas que se mostrarán antes del juego. Este test también induce al estrés relacionado a las condiciones “controlables” e “incontrolables”.

1.2. Solución propuesta y alcances

Se propone colaborar en este equipo multidisciplinario mediante la inclusión del desarrollo front-end de la aplicación móvil en cuestión y obtener un primer prototipo funcional de esta herramienta. Específicamente, se plantea desarrollar los 3 test tipo juego requeridos para poder realizar las primeras pruebas en una cantidad reducida de usuarios.

Para construir la aplicación, se propone utilizar el framework[2] de desarrollo móvil “React Native”[3] con el lenguaje de programación “JavaScript”[4], ya que se cuenta con experiencia previa en estas tecnologías, además de permitir abordar los requerimientos mencionados anteriormente y el despliegue de la aplicación tanto para sistemas operativos iOS y Android con un solo desarrollo de código, aunque de primera instancia, el alcance solo irá dirigido a dispositivos con sistema operativo Android.

De igual forma, como esta aplicación cuenta con múltiples formularios que pueden hacer extenso el tiempo de uso de la aplicación, se propone la utilización de almacenamiento local para los avances tanto en juegos, como en los formularios, ante un posible uso permitido de sesiones para completar el contenido de la aplicación y enviar los resultados al back-end.

1.3. Objetivo general

Desarrollar el primer prototipo funcional de la aplicación móvil StressMApp, con la cual el equipo multidisciplinario detrás de este proyecto pueda realizar las primeras pruebas en usuarios y así obtener datos que les permita avanzar en su investigación sobre el estrés y relación en el rendimiento cognitivo de estudiantes chilenos.

1.4. Objetivos específicos

1. Diseñar e implementar la aplicación base
2. Desarrollar el test de inducción al estrés “N-back”
3. Desarrollar el test de inducción al estrés de prueba de imágenes y decisiones
4. Desarrollar el test de inducción al estrés de prueba de exploración y aprendizaje
5. Obtener los datos de comportamiento en los test y comunicarlos al back-end de manera segura
6. Desplegar un prototipo integral de la aplicación móvil
7. Diseñar y realizar pruebas con usuarios

1.5. Metodología

Como se ha mencionado, este es un problema de desarrollo de software, donde el foco de esta propuesta específicamente son 3 test tipo juego en una aplicación móvil desarrollada desde cero. La forma de trabajo se realizó aplicando métodos ágiles, de modo de abordar el trabajo de manera iterativa e incremental, incluyendo también reuniones periódicas de manera semanal (cada jueves a mediodía) con el equipo multidisciplinario con el que se discutieron los distintos aspectos, requerimientos y retroalimentación del desarrollo en cuestión.

Se comenzó el desarrollo con los cimientos de la aplicación móvil, desde donde iniciar el desarrollo de cada uno de los 3 test tipo juego mostrados anteriormente. El desarrollo de estos test se realizó de manera incremental, uno a la vez, para obtener prototipos funcionales.

Es importante aclarar que el trabajo hacia el producto software integral de esta aplicación incluye colaboración hecha de manera simultánea y con objetivos unipersonales de otros compañeros memoristas, por lo que la metodología de trabajo también es orientada hacia la modularidad. Lo que respecta al software back-end y bases de datos corresponde a David Torres. El desarrollo de formularios y aspectos de diseño de interfaz, incluyendo de los test, fue trabajado por Francisca Hillerns.

Ya que el trabajo de David Torres es el relacionado con el software back-end, incluye el diseño de las bases de datos (BD) que almacenan los resultados capturados de los test en la aplicación móvil (front-end). Lo que respecta a esta MT es la correcta elección y captura de los datos por cada test en el front-end. Por ello aquí no se verá un MER, sino mas bien el detalle de los datos a capturar y el output (JSON) resultante. También cabe mencionar que el diseño final de las tablas en la BD es realizado una vez finaliza el diseño y selección de datos desde el front-end.

Tampoco es objetivo de esta MT el diseño de interfaces de los test. La metodología de trabajo que se utiliza en este caso y que corresponde a este trabajo, es diseñar e implementar enfocando hacia el funcionamiento de los test, entregando una interfaz básica, etiquetada y 100% funcional, pero sin los estilos gráficos finales. Por ello en los apartados donde se muestran los resultaos de implementación, se hace hincapié en prestar especial atención a los aspectos funcionales que se detallan y no al diseño de la interfaz.

1.6. Estructura del informe

A continuación, se explican brevemente las diferentes secciones incluidas en el informe.

En el capítulo 2 se presenta el marco teórico, donde se mencionan los conceptos y temas previos requerido para comprender a cabalidad el contenido del informe dentro de un contexto de ingeniería informática. Se presenta una visión integral del funcionamiento de la aplicación y

conceptos relacionados al ambiente de desarrollo móvil. También se comenta sobre otros trabajos relacionados y se concluye con una discusión sobre los mismos, en contraste con esta aplicación.

El capítulo 3 es el de descripción de la propuesta. Se profundiza la solución mencionada para esta problemática, se detalla la metodología de abordaje y las prácticas a utilizar. También se menciona sobre la arquitectura del sistema y se justifican las herramientas a utilizar.

El cuarto capítulo corresponde a los detalles de la propuesta, que es donde se profundiza en el desarrollo del software, todo mediante un enfoque de análisis, diseño e implementación para cada test. Correspondiendo a la etapa de análisis, la descripción profunda de cada test, su comportamiento y requisitos a cumplir. El diseño abarca la solución del problema, se planifican los algoritmos a un alto nivel y se establece la estructura fundamental de cada test. Se finaliza con la implementación, presentando los resultados del funcionamiento y comportamiento obtenidos de materializar las etapas anteriores.

Luego, el capítulo 5 comprende a la evaluación de la propuesta. Aquí se describe la metodología de evaluación para el producto obtenido del capítulo anterior, se presentan y analizan los resultados.

Finalmente, en los capítulos 6 y 7 se incluyen las conclusiones del trabajo realizado y las referencias bibliográficas.

2. Marco teórico

A continuación, se abordan los conceptos y temas previos necesarios para una comprensión completa del informe bajo un contexto de la Ingeniería Informática. Junto con incluir información sobre otros trabajos relacionados y concluir con una discusión sobre los mismos, en contraste con esta aplicación.

2.1. Visión integral de la aplicación StressMApp

Resulta fundamental entender la completa visión de software que abarca esta aplicación móvil. Primeramente, que se utiliza un esquema clásico de arquitectura de software “cliente-servidor”. Es objetivamente un instrumento de investigación, buscando obtener información de los sujetos de estudio, usuarios objetivos de esta aplicación, que corresponden a estudiantes universitarios. La aplicación móvil cumple el rol de cliente en esta arquitectura. Desde el lado del servidor se reciben los datos capturados en la app y se almacenan en la base de datos, también se cuenta con plataforma web que permite a los investigadores revisarlos y extraerlos para su posterior análisis.

Volviendo a la aplicación, esta cuenta con autentificación para poder ingresar al contenido e identificar a los usuarios. Luego el contenido consta básicamente de cuestionarios y 3 test tipo juego que se van intercalando (estos test que ya han sido introducidos, también se profundizan en el capítulo de Detalles de la propuesta). El avance dentro de la aplicación es progresivo y siguiendo una única ruta. Es decir, se accede a un contenido y se completa una única vez, dando paso al siguiente contenido y así sucesivamente hasta completar todos los cuestionarios y test, lo que da por finalizado el uso de la aplicación.

2.2. Framework de desarrollo de software

El framework [2] es un término que proviene del inglés y significa “marco de trabajo” o “estructura”. En el ámbito de la programación, un framework es un conjunto de herramientas y librerías que se utilizan para desarrollar aplicaciones más fácilmente y de manera más eficiente.

Un framework es un conjunto de reglas y convenciones que se usan para desarrollar software de manera más eficiente y rápida. Estos marcos de trabajo se emplean para ahorrar tiempo y esfuerzo en el desarrollo de aplicaciones, ya que proporcionan una estructura básica que se puede utilizar como punto de partida. Además, los frameworks también ofrecen soluciones a problemas comunes en el desarrollo de software, lo que significa que los desarrolladores pueden

centrarse en las funcionalidades específicas de su aplicación en lugar de perder tiempo resolviendo problemas técnicos.

2.3. React

React[5] es un framework para construir aplicaciones web altamente eficientes y escalables. Proporciona una forma única de generar interfaces de usuario a través de componentes modulares, los cuales pueden ser reutilizados en diferentes secciones de la aplicación para mejorar la eficiencia del proceso de desarrollo. En términos generales, React utiliza JavaScript y HTML para construir aplicaciones web, y una arquitectura basada en componentes para mejorar la capacidad de reutilización de código de la aplicación.

2.3.1. Componente

Un componente de React es una pieza modular de código que se utiliza para representar elementos de interfaz de usuario (UI) reutilizables dentro de una aplicación React. Pueden ser definidos como funciones o como clases y están diseñados para aceptar datos (props) como entrada e incluir lógica del funcionamiento. Los componentes de React también pueden mantener un estado interno que se puede actualizar a medida que cambia la entrada.

2.3.2. Variables de estado

Las variables de estado en React se refiere a un objeto que contiene datos sobre el estado actual del componente. Cuando se modifica el estado, React vuelve a renderizar el componente para reflejar el estado actualizado. El estado es local a un componente, lo que significa que cada componente puede tener su propio estado.

Es importante destacar que cuando se actualiza un estado, ocurre lo siguiente:

1. React llama de nuevo a la función que retorna el componente con el valor actualizado del estado
2. Dicha función retorna el componente (elemento de interfaz)
3. React actualiza la pantalla con este último retorno

2.4. React Native

React Native[3] (también conocido como RN) es un popular framework de aplicaciones móviles basado en React y JavaScript que permite crear aplicaciones móviles con renderizado nativo para iOS y Android. El framework permite crear una aplicación para ambas plataformas utilizando el mismo código base, desplegando componentes nativos para crear una experiencia de usuario auténtica y de alta calidad en los dispositivos móviles.

React Native fue lanzado por primera vez por Facebook como un proyecto de código abierto en 2015. En solo un par de años, se convirtió en una de las principales soluciones utilizadas para el desarrollo móvil.

2.5. Trabajos relacionados

En esta subsección se comentará sobre la existencia de algunos enfoques software relacionados con estos tipos de test. Recordemos, que estas pruebas son basadas en experiencias realizadas originalmente en condiciones de laboratorio, donde los participantes son sujetos de estudio. También, que la aplicación integral incluye cuestionarios y el objetivo se vincula al estudio del estrés y su relación con el rendimiento cognitivo en estudiantes chilenos.

Dadas estas características, se encontró una única aplicación de similar enfoque, detallada a continuación.

Neureka[6] propone en su página web: “Tomar la investigación fuera del laboratorio utilizando teléfonos inteligentes”. Es una aplicación de ciencia ciudadana, de origen irlandés que permite a los usuarios participar en investigaciones o desafíos científicos sobre la salud cerebral, jugando y respondiendo a cuestionarios que miden distintos aspectos de la función cerebral y la salud mental, como los factores de riesgo de demencia y los trastornos del estado de ánimo. Cuenta con un enfoque Big-Data al contar ya con miles de usuarios. Entre sus objetivos destaca saber por qué algunas personas son más resistentes que otras, cómo las emociones, el estado de ánimo, irritabilidad y motivación interactúan entre sí y con el ambiente, y cómo pueden desencadenar círculos viciosos de los que es difícil salir.

Respecto a los test, no se encontraron aplicaciones que mencionen el uso de estas pruebas de origen científico con inductores de estrés explícitos como imágenes aversivas, ni el uso de condiciones de controlabilidad. Solo del test de memoria “N-back” es posible encontrar resultados, como el descrito a continuación.

Nombrado “N-Back Challenge”[7] es una aplicación que promete elevar el IQ con pocos minutos en sesiones diarias durante un desafío de 20 días. Utiliza una versión llamada “dual”, ya que comprende estímulos visuales y sonoros. Su objetivo es incrementar la inteligencia del participante mediante la constancia de este exigente juego de memoria a modo de desafío de 20 días.

Por otro lado, el último test a implementar para esta MT, el juego de Exploración y Predicción, es basado en la adaptación realizada por los investigadores Hillary A. Raab, Careen Foord, Romain Ligneul y Catherine A. Hartley. Contextualizándolo hacia una narrativa de pilotos, aviones e islas que permite que el test pueda ser entendido incluso por niños. Dichos investigadores tienen un prototipo programado en Matlab que prueban con usuarios en condiciones de laboratorio [8]. Por parte de equipo de StressMApp, se han puesto en contacto con estos investigadores y se ha

conseguido tener acceso a este prototipo (aún no público) para utilizar dicha adaptación como base para este desarrollo.

2.6. Discusión

En este apartado se discute sobre los trabajos expuestos en el punto anterior y en cuál escenario tecnológico se encuentra la presente propuesta.

Se comentó sobre 3 enfoques software, los 2 primeros de alcance público y el último un prototipo privado. Donde destaca la aplicación Neureka por tener un enfoque científico y utilizar las mecánicas de varios test tipo juego en conjunto con cuestionarios para realizar estudios. Es de hecho esta aplicación una referencia importante para los investigadores de StressMApp, que respalda la factibilidad de este tipo de herramientas.

También existen esfuerzos por generar versiones más amigables y factibles de ejecutar remotamente de estas pruebas, junto con disposición de colaborar a la comunidad.

Pero, tal como se mencionó en el punto anterior, es difícil encontrar aplicaciones que refieran al uso de estas pruebas de origen científico, específicamente con inductores de estrés explícitos como imágenes aversivas, así como con uso de condiciones de controlabilidad. Es un universo acotado de estas herramientas, más aún con un enfoque hacia la investigación. En cambio, es posible encontrar versiones de estos juegos de memoria dirigidos al participante como entrenamiento mental.

Esto posiciona al pilotaje de StressMApp como un interesante y novedoso instrumento de investigación en este acotado y reciente contexto de estudio psiquiátrico de manera remota a través de aplicaciones móviles, con un enfoque más específico hacia la inducción de estrés y la repercusión sobre la cognición de estudiantes universitarios.

3. Descripción de la propuesta

Como se ha señalado anteriormente, existe un grupo multidisciplinario de investigadores con un proyecto que ha obtenido recientemente apoyo y financiamiento por parte de la VRID de la Universidad de Concepción. Dichos investigadores requieren del desarrollo de un primer prototipo funcional de una aplicación móvil basada en cuestionarios sobre información del participante y 3 test tipo juego relacionados con la memoria, exploración, aprendizaje e inducción al estrés. Desde la aplicación instalada en el dispositivo móvil debe ser capaz de recolectar estos datos de respuesta e interacción y comunicarlos a un servidor con un base de datos, desde el cual los investigadores puedan acceder para extraer estos datos.

Este primer prototipo funcional, en conjunto con resultados obtenidos de realizar pruebas con una pequeña cantidad de usuarios, también es requerido para ser mostrado como avance sólido a distintos fondos concursables de financiamiento.

Lo propuesto a realizar en esta memoria de título es el desarrollo de los 3 test tipo juego para la aplicación móvil. Dichos test que fueron mencionados en la introducción y son individualmente profundizados en el capítulo siguiente, en los apartados de “análisis” respectivos, corresponden a adaptaciones experimentalmente viables de ejecutar de manera remota, ya que estos son originalmente basados en ejercicios que se llevan a cabo en condiciones controladas (test de laboratorio) y con la instrucción de un especialista.

Al ser el primer prototipo funcional, se tiene presente una perspectiva a largo plazo, donde continúe el desarrollo y las investigaciones por parte del equipo multidisciplinario. Por ende, cobra aún más importancia atender esta situación con altos estándares, como abordar el desarrollo mediante los procesos ingenieriles de desarrollo de software: análisis, diseño e implementación, que entrega un orden de etapas para comprender el problema, sus especificaciones, diseñar la solución con fuertes bases lógicas y algoritmos sólidos que cumplan los requerimientos y expectativas, para finalmente implementar con buenas prácticas de programación y documentación.

Como se mencionó anteriormente, la aplicación móvil desempeña el papel de cliente en un modelo clásico de arquitectura de software "cliente-servidor". Los investigadores obtienen la información de los participantes del test directamente desde la base de datos del servidor. Por lo tanto, es crucial manejar adecuadamente la recopilación de datos y el comportamiento de los participantes en los test, ya que esta información es de gran importancia para los investigadores. Por ende, también es necesario asegurarse de una correcta captura de datos y una comunicación efectiva con el servidor.

La idea, además, es lograr un desarrollo iterativo e incremental, donde al terminar uno a uno los test, estos puedan ser evaluados por el equipo y obtener así retroalimentación respecto a sus

funcionalidades y especificaciones. Para finalmente desplegar versiones integrales de la aplicación y poder realizar pruebas en dispositivos y diferentes usuarios.

Para llevar a cabo esta propuesta, se optó por utilizar las siguientes herramientas

- React Native: Framework de desarrollo móvil
- JavaScript: Lenguaje de programación
- JSON: Formato de comunicación de datos

La justificación de escoger estas herramientas recae primeramente por la experiencia previa que se tiene utilizándolas, lo que agiliza el desarrollo, pero además no se deja de lado que cumplen perfectamente lo requerido por el alcance y además con ciertas ventajas como poder utilizar el mismo código para desplegar en sistemas operativos iOS. Sumado a esto, también se considera lo anterior mencionado, de tener una perspectiva a largo plazo, ya que el lenguaje de programación JavaScript es por lo menos hace 5 años el lenguaje más utilizado del mundo [9], teniendo gran cantidad de documentación, lo que reduce esfuerzos de poder comprender la sintaxis o incluso poder encontrar fácilmente nuevos desarrolladores que colaboren en un futuro.

En resumidas cuentas, se prioriza el lenguaje de programación JavaScript. Con él, el framework para construir aplicaciones móviles es React Native, que a su vez cumple perfectamente los alcances y cuenta con amplia documentación. Y por otro lado utilizar JSON [10] (JavaScript Object Notation) para comunicación de datos, dado el soporte nativo con JavaScript, sumado a su amplia adopción y estándar de facto en la comunicación cliente-servidor, junto con ser ligero, eficiente y fácil de estructurar.

4. Detalle de la propuesta

En este capítulo se profundiza en el desarrollo de los test conforme a lo anteriormente descrito sobre la propuesta, destacando el enfoque de análisis, diseño e implementación como etapas para el proceso exitoso del desarrollo de software. Mediante el análisis, se logra una comprensión profunda de las necesidades de cada test y los requisitos que deben cumplir. El diseño abarca la solución del problema, se planifican los algoritmos a un alto nivel y se establece la estructura fundamental de cada test [11]. Finalmente, la implementación materializa los aspectos previos, derivando en la exposición de los resultados respecto al funcionamiento y comportamiento obtenido.

Es importante hacer hincapié en que a la hora de presentar resultados de implementación, se debe prestar atención a las funcionalidades que se mencionan y no en aspectos de diseño de la interfaz, producto de que el foco de esta propuesta no abarca dichos aspectos y tampoco los de back-end que serán necesarios de mencionar para profundizar en las características de los test, y que estos 2 aspectos (diseño de interfaces y back-end) fueron desarrollados por otros compañeros memoristas.

4.1. Test N-Back

4.1.1. Análisis

El test N-Back [12] es una tarea de memoria de trabajo utilizada en la investigación psicológica y cognitiva para evaluar la memoria de corto plazo y la capacidad de atención de una persona.

En el test N-Back, se presenta una secuencia de estímulos, como letras, números o imágenes, en este caso letras. El participante debe indicar si el estímulo actual es el mismo que el presentado N pasos atrás en la secuencia. Por ejemplo, en un test 1-Back, el participante debe indicar si el estímulo actual es igual al presentado inmediatamente antes. En un test 2-Back, el participante debe comparar el estímulo actual con el presentado dos pasos antes, y así sucesivamente. Tal como se muestra en la figura 4.

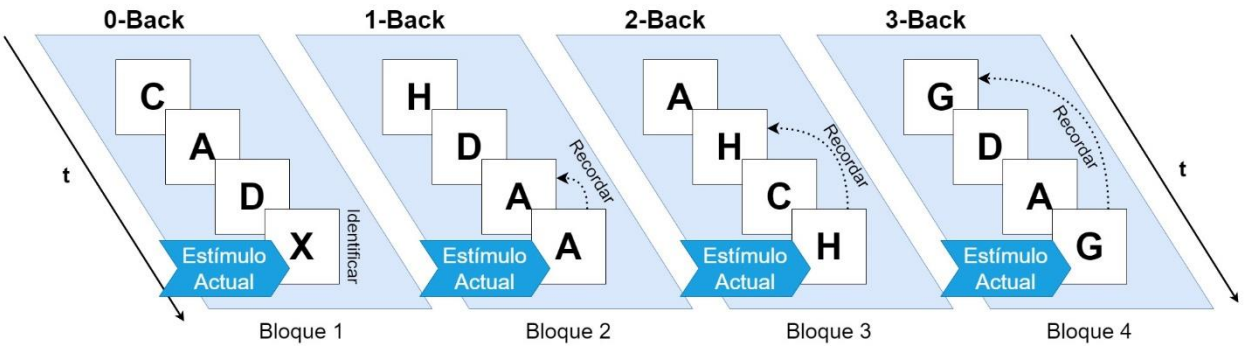

Figura 4. Objetivos de las diferentes versiones de N-Back.

También se utiliza una versión 0-Back, llamada “de control”, donde el objetivo es indicar cuándo el estímulo señalado, en este caso la letra “X”, es mostrada. Solo mide capacidad de atención, pero sirve para introducir al usuario a la mecánica e interfaz del test.

Para esta implementación, la tarea se estructura en bloques donde se desarrolla una versión “N” del test. De igual forma, como se muestra en la figura 4, comenzando desde el 0-Back hasta llegar a la versión 3-Back. Es decir, finalizar el test requiere de completar 4 bloques. Cada uno se compone, en este caso, de 25 ensayos, donde primeramente se muestra una letra al azar durante 1 segundo, luego se oculta por un tiempo de 2 segundos y una vez cumplido este tiempo, comenzar un nuevo ensayo.

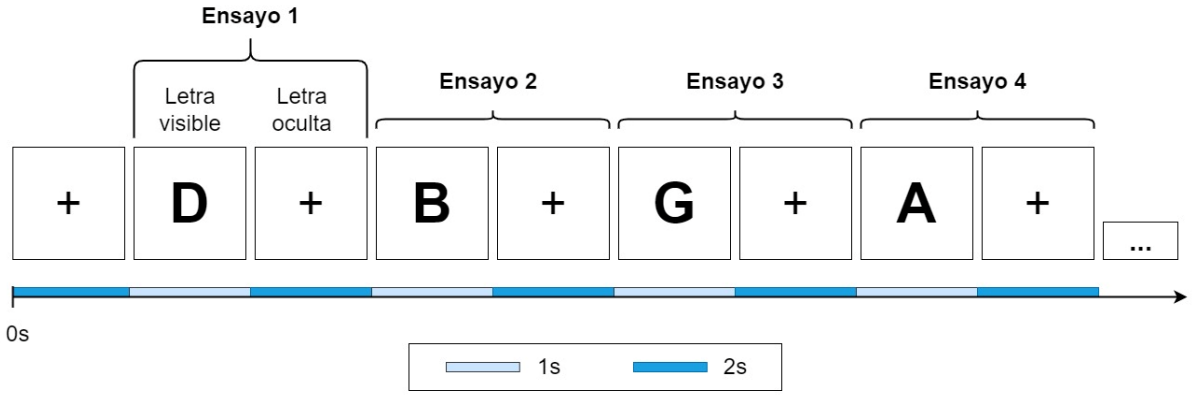

Figura 5. Estructura de ensayos dentro de un bloque.

Antes del primer ensayo y como se muestra en la figura 5, se despliega un signo “más (+)” por 2 segundos, lo que llamaremos “Landing view”. El cual sirve para no comenzar el test de golpe y atraer la vista del usuario a ese punto, en el que se desplegará la letra del ensayo. Así mismo, se expondrá este símbolo en la etapa de los ensayos donde la letra se oculta.

El participante estará habilitado para interactuar trascurridos N ensayos. Debe responder mediante botones si la letra mostrada N ensayos atrás es igual o distinta a la última letra

desplegada, teniendo para responder los 3 segundos que comprende el ensayo, antes de que se renderice una nueva letra. Solo se permite una respuesta por ensayo habilitado.

Es importante registrar los datos característicos de los bloques y ensayos, junto con capturar las respuestas de los participantes, para comunicarlas de forma estructurada al servidor y a la base de datos.

4.1.2. Diseño

A continuación, se presentan los principales desafíos obtenidos al momento de diseñar la lógica de programación de este test.

En primer lugar, se debe poder reutilizar el programa para los distintos valores de N admisibles, que recordemos, son enteros positivos incluyendo el cero, en este caso hasta el 3. Para ello se diseñó que el valor de N fuera una variable global del test, donde su valor se extrajera desde un array que contenga la secuencia de niveles a cursar para completar el test y cuyo índice aumente al finalizar cada bloque. Pudiendo obtener así, algoritmos genéricos del funcionamiento completo del test, en función a los distintos valores que puede tomar N.

También, se debe generar las letras de forma aleatoria pero controlada. A los casos en que la letra (estimulo) de los ensayos coincida con la de N ensayos atrás, los llamaremos “match”. La generación es controlada en el sentido de que el caso de match no sea una situación aislada dentro de los 25 ensayos que dura un bloque. Con este propósito, se diseñó que los ensayos tuvieran un identificador de caso (match o no match), el cual se nombró como “tipo de ensayo” y acorde a esta variable de condición, asignar un tipo de letra correspondiente. El tipo de ensayo se define conforme a la elección de un número entero positivo que considere la cantidad inversa de opciones (1/opciones) de obtener un caso de match por ensayo. A continuación, se muestran los algoritmos N°1 y N°2 con el seudocódigo de solución propuesto.

Algoritmo N°1: Obtención del tipo de ensayo.

```
OPCIONES = 3 //constante

FUNCTION obtenerTipoDeEnsayo(){
      aleatorio = RandomIntBetween(1, OPCIONES)
      IF aleatorio == 1 THEN
            RETURN "match"
      ELSE
            RETURN "no match"
      ENDIF
END

*RandomIntBetween(A,B) retorna un número entero aleatorio entre el rango de
A y B, ambos incluidos.
```

Algoritmo N°2: Obtención de letra estímulo acorde al tipo de ensayo y valor de N.

```
LETRAS = ['A', 'B', 'C', 'D', 'E', 'F', 'G', 'H'] //no incluye X
N_NIVELES = [0, 1, 2, 3]
id_bloque = ObtenerIdBloqueActual() //retorna 1, 2, 3 o 4
N = N_NIVELES[id_bloque -1]
historial = [] //stack

FUNCTION obtenerNuevaLetra()
      hLen = historial.length
      tipoDeEnsayo = obtenerTipoDeEnsayo()
      IF tipoDeEnsayo == "match" and hLen >= N THEN
            IF N == 0 THEN
                  nuevaLetra = 'X'
            ELSE
                  nuevaLetra = historial[hLen - N]
      // en otro caso, se asigna una letra aleatoria pero diferente a la
      // del caso de match
      ELSE
            nuevaLetra = LETRAS.randomItem()
            IF N != 0 and hLen >= N THEN
                  WHILE (nuevaLetra == historial[hLen - N])
                        nuevaLetra = LETRAS.randomItem()
                  ENDWHILE
      ENDIF
      historial.push(nuevaLetra)
      RETURN nuevaLetra
END
```

Globalmente, se diseñó la lógica del test en base al desarrollo e iteraciones de bloques de una versión N del test y en cada ensayo se logran diferenciar 3 fases: landing view, letra visible y letra oculta. En el cual los ensayos comienzan después del landing view, iteran 25 veces por ciclos conformados por la fase de letra visible y de letra oculta, tal como se describió en el análisis del test. El cambio entre estas fases se gatilla por un temporizador. Desde el ensayo N+1 en adelante se habilitan los 2 botones (igual y distinto) para que el usuario pueda responder al test. Con la respuesta ejecutada, se deshabilitan los botones hasta el comienzo del siguiente ensayo, independiente de si la respuesta fue durante la fase de letra visible u oculta. Todo este comportamiento se plasma en el Algoritmo N°3.

Algoritmo N°3: Estructura general de fases y manejo de estados del test N-Back.

```
ENSAYOS_POR_BLOQUE = 25
contadorEnsayos = 0

FUNCTION NBack()
      fase = estadoInicial('landing')
      botonesDisponibles = estadoInicial(false)
      autorizacion = referenciaInicial(true)

      IF autorizacion == true THEN
            IF fase == 'landing' THEN
                  AFTER 2000ms DO
                        fase = cambioDeEstado('letra visible')
                  END
            ELSE IF fase == 'letra visible' THEN
                  contadorEnsayos++
                  letraAMostrar = obtenerNuevaLetra()
                  AFTER 1000ms DO
                        autorizacion = nuevaReferencia(true)
                        fase = cambioDeEstado('letra oculta')
                  END
                  IF contadorEnsayos > N THEN
                        autorizacion = nuevaReferencia(false)
                        botonesDisponibles = cambioDeEstado(true)
                  ENDIF
            ELSE IF fase == 'letra oculta' THEN
                  AFTER 2000ms DO
                        IF contadorEnsayos < ENSAYOS_POR_BLOQUE THEN
                              autorizacion = nuevaReferencia(true)
                              fase = cambioDeEstado('letra visible')
                        ELSE
                              finalizarBloque()
                        ENDIF
                  END
            ENDIF
      ENDIF

      // función local
      FUNCTION alPresionarBoton(id_boton)
            chekearRespuesta(id_boton)
            autorizacion = nuevaReferencia(false)
            botonesDisponibles = cambioDeEstado(false)
      END
END
```

De lo anterior surge un importante aspecto de diseño ligado al framework de desarrollo. Para ello se debe recordar que para ejecutar un nuevo renderizado que permita actualizar lo que ve el

usuario, se requiere cambiar el valor de alguna variable de estado, lo cual a su vez ejecuta una nueva lectura del código, previo a este nuevo renderizado. De esta forma se origina el caso de que si dentro de alguna de estas 3 fases se requiera actualizar un estado antes del estado que maneja el cambio de fase, se ejecute paralelamente un nuevo arranque del temporizador de dicha fase, producto de que un nuevo renderizado no termina con la ejecución de los procesos ya iniciados, generando un comportamiento no deseado de cambios de estados. Un ejemplo es habilitar los botones luego del ensayo N+1, o deshabilitarlos luego de una respuesta del usuario.

El mecanismo que se ideó para prevenir este comportamiento no deseado de renderizados, y que se muestra en el Algoritmo N°3, fue una condición de ingreso a las fases, también llamada condición de renderizado, que permite autorizar o bloquear el acceso a las fases y su temporizador. Por ejemplo, se debe abrir antes de gatillar un cambio de fase y bloquearlo antes de cualquier otro cambio de estado. Se utiliza una variante de variable de estado ofrecida por el framework, llamada “referencia” que de igual forma es una variable mutable a un valor durante el ciclo de vida del componente, pero sin provocar un nuevo renderizado tras su actualización.

Finalmente, los datos que se capturan para ser enviados al servidor son los siguientes:

Etiqueta	**Tipo de dato**	**Descripción**
num_bloque	{1, 2, 3, 4}	Bloque actual
n_type	{0, 1, 2, 3}	Valor de N, tipo de N-Back
num_prueba	{1, 2, ..., 25}	Número del ensayo dentro del bloque
tipo_prueba	{0, 1}	1: El ensayo muestra un estímulo de match; 0: El ensayo no corresponde a un caso de match
letra_mostrada	{1, 2, 3, …}	Número asociado a la posición de letra en el alfabeto inglés, mostrada durante el ensayo. A=1, B=2, C=3, …
fecha_inicio	Timestamp	Fecha y hora (DD/MM/AAAA HH:MM:SS.SSS) del inicio del ensayo
fecha_respuesta	Ttimestamp	Fecha y hora de la respuesta del usuario. Si el usuario no responde, se registra el timestamp del término del ensayo
tiempo_respuesta	[ms]	Tiempo en milisegundos de la diferencia entre el instante de comienzo del ensayo y el de respuesta
score	{-1, 0, 1}	1: Respuesta correcta; 0: Respuesta incorrecta; -1: No responde
match	{0, 1}	1: Respuesta correcta ante un estímulo de match; 0: En otro caso

no_match	{0, 1}	1: Respuesta correcta de identificación de un tipo de ensayo “no match”; 0: En otro caso
miss	{0, 1}	1: Cuando no se respondió al estímulo: 0: Se respondió al estímulo

4.1.3. Resultados de implementación

A continuación, se exponen los resultados respecto al funcionamiento y comportamiento de la implementación.

Primeramente, se presenta una abstracción general del test para los 4 bloques (figura 6), donde se pueden apreciar los distintos instantes en que se habilitan los botones para responder (se vuelven azules), junto la pantalla que indica el final de cada etapa y permite enlazar a la siguiente.

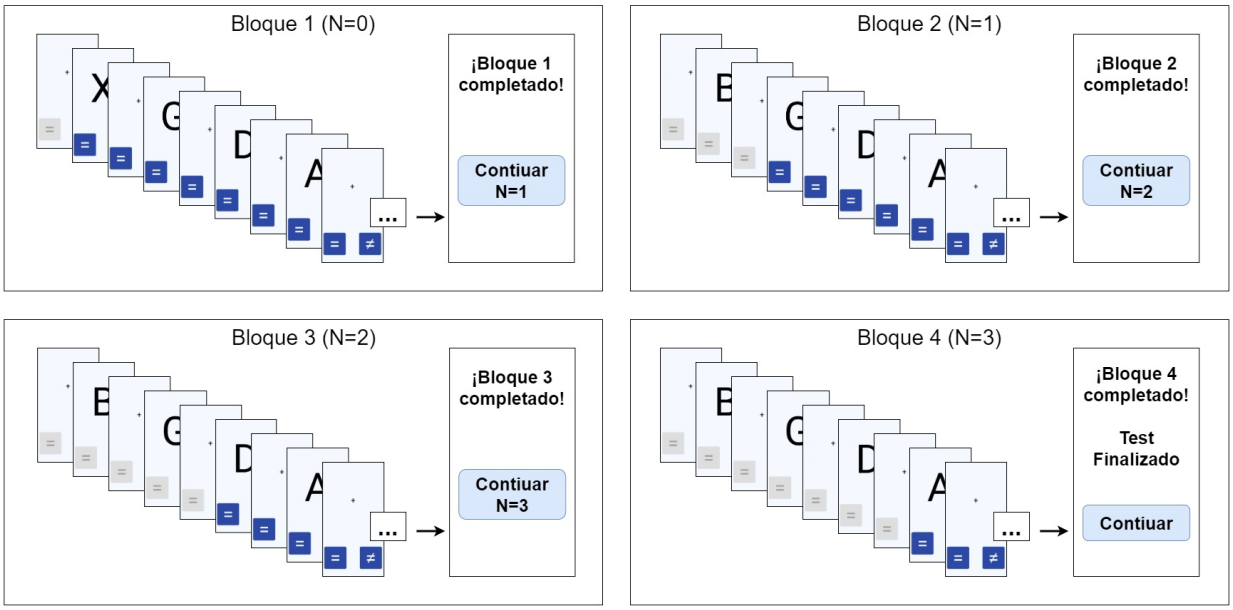

Figura 6. Resumen de 4 bloques del test N-Back.

Por otro lado, el comportamiento o cambios de estado ante la respuesta del usuario tanto para la fase visible del estímulo, como para la fase de letra oculta en el transcurso de un ensayo se muestran en las figuras 7 y 8 respectivamente.

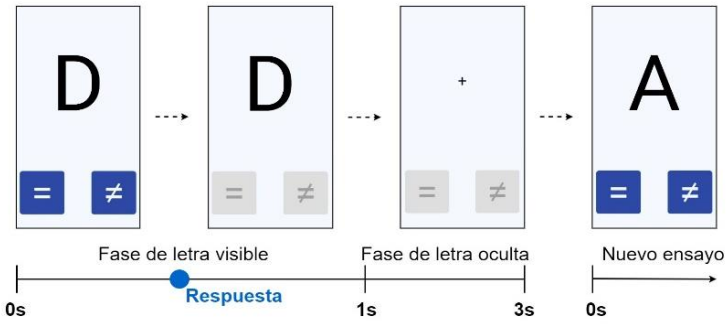

Figura 7. Comportamiento ante respuesta de usuario en fase de letra visible.

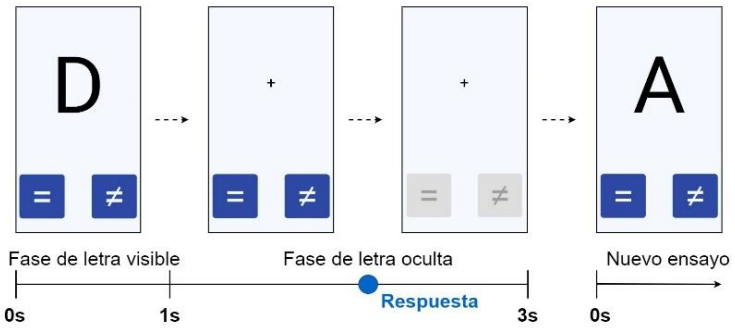

Figura 8. Comportamiento ante respuesta de usuario en fase de letra oculta.

Finalmente, un ejemplo del JSON resultante (graficado en jsongrid.com), donde para fines demostrativos, sólo se utilizaron 10 ensayos por bloque. En la figura 9 se muestran expandidos los resultados del bloque 3 (N=2), donde se puede notar en la columna “tipo_prueba” que 3 de los 10 ensayos resultan en caso de match. Es decir que para esta versión 2-Back, se presenta la misma letra mostrada hace 2 ensayos atrás.

[-] bloques [4]

	num_bloque	n_type	pruebas
1	1	0	[+] pruebas [10]
2	2	1	[+] pruebas [10]
3	3	2	[-] pruebas [10]
4	4	3	[+] pruebas [10]

[-] pruebas [10]

	num_prueba	tipo_prueba	score	match	no_match	miss	tiempo_respuesta	letra_mostrada	fecha_respuesta	fecha_inicio
1	1	0	-1	0	0	1	3039	6	2023-07-04T00:39:52.681Z	2023-07-04T00:39:49.642Z
2	2	0	-1	0	0	1	3041	1	2023-07-04T00:39:55.731Z	2023-07-04T00:39:52.690Z
3	3	0	1	0	1	0	878	5	2023-07-04T00:39:56.618Z	2023-07-04T00:39:55.740Z
4	4	0	1	0	1	0	880	8	2023-07-04T00:39:59.684Z	2023-07-04T00:39:58.804Z
5	5	0	1	0	1	0	792	2	2023-07-04T00:40:02.668Z	2023-07-04T00:40:01.876Z
6	6	1	0	0	0	0	846	8	2023-07-04T00:40:05.768Z	2023-07-04T00:40:04.922Z
7	7	0	1	0	1	0	649	6	2023-07-04T00:40:08.618Z	2023-07-04T00:40:07.969Z
8	8	1	0	0	0	0	764	8	2023-07-04T00:40:11.766Z	2023-07-04T00:40:11.002Z
9	9	1	1	1	0	0	1075	6	2023-07-04T00:40:15.110Z	2023-07-04T00:40:14.035Z
10	10	0	1	0	1	0	1104	2	2023-07-04T00:40:18.192Z	2023-07-04T00:40:17.088Z

Figura 9. Ejemplos de datos capturados del test N-Back.

4.2. Test de inducción al estrés

4.2.1. Análisis

Este test es el más experimental de los 3 a desarrollar, busca inducir estrés en el participante con estímulos aversivos/negativos, además de presentar versiones controlables e incontrolables producidas a través de emparejamiento de usuarios [13]. Los resultados obtenidos pueden proporcionar a los investigadores información sobre efectos del estrés en las capacidades cognitivas de los usuarios y sus habilidades para enfrentar el estrés.

Aunque se tiene una versión controlable y otra incontrolable del test, los participantes no están al tanto de aquello y la descripción del test que ellos ven es la misma. En cada ensayo se presenta una figura, denominada “señal”, junto con 4 botones. En este caso son 3 señales: un polígono en forma de “L”, un círculo y un triángulo, que van apareciendo aleatoriamente hasta cumplir 6 apariciones cada uno. Independiente de cuál de estas 3 señales que presenta esta versión, siempre son los mismos 4 botones por ensayo que acompañan a la figura. Internamente lo que ocurre es que a cada una de estas 3 señales se le asigna aleatoriamente uno de los 4 botones, generando un par “señal-botón”, tal como se muestra en la figura 10.

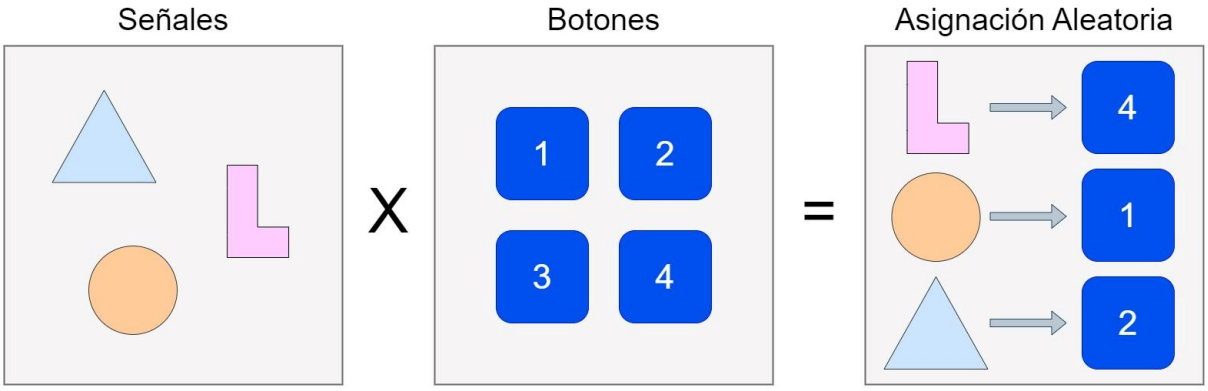

Figura 10. Ejemplo de proceso de asignación aleatoria entre señales y botones.

La idea es que el participante, a través de los ensayos, vaya identificando dichos pares “señal-botón” a través de prueba y error, ya que cada respuesta del usuario, consistente en presionar uno de los botones, arroja una retroalimentación. En esta ocasión, se muestra un estímulo visual de una imagen neutral en caso de respuesta correcta o una imagen negativa/aversiva en caso contrario. Esta es la mecánica de cada ensayo, lo que se grafica en la figura 11.

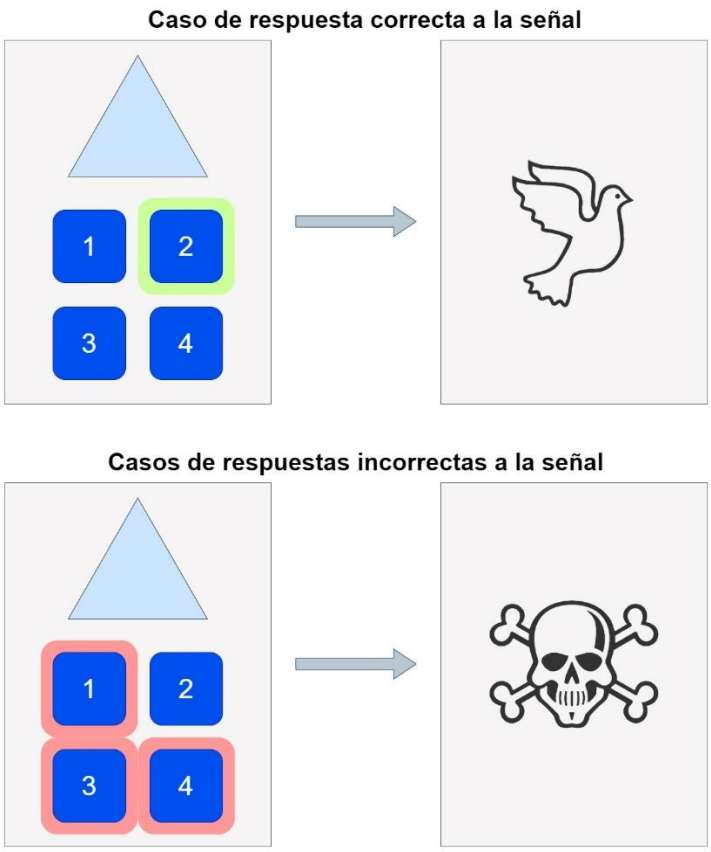

Figura 11. Ejemplo de mecánica de ensayo y casos de respuesta en test de inducción al estrés, utilizando asignación mostrada en figura 10.

Los estímulos de imágenes neutras son del estilo de fotografías de paisajes y objetos cotidianos, y en general no llaman la atención del usuario. Por otro lado, los estímulos de imágenes aversivas/negativas se utilizan para inducir estrés en los participantes y son obtenidas del banco de imágenes del International Affective Picture System (IAPS) [14], donde se pueden encontrar ejemplos muy desagradables y crudos, como mutilaciones de guerra, procedimientos médicos, crueldad animal, entre otros.

Otro aspecto importante y que se menciona al inicio de esta sección, es la presencia de versiones controlables e incontrolables, junto con el emparejamiento. Esto es, que sin previo aviso se tendrá un comportamiento diferente para algunos usuarios. Donde el comportamiento controlable, es precisamente lo expuesto hasta ahora sobre el test, donde las respuestas correctas tienen un estímulo neutro y el estímulo negativo se despliega ante las respuestas erróneas del usuario. En este caso, el usuario puede evitar que se le presenten los estímulos desagradables aprendiendo por prueba y error las relaciones señal-botón.

Para el caso incontrolable de este test, un usuario es asignado a esta condición por un método de emparejamiento llamado por los investigadores como “yugo” o “yoke” en inglés. Utilizando

cuestionarios previos a esta prueba, le asigna (en back-end) a un usuario, por ejemplo acorde a su edad y/o género, a otro usuario de estas mismas características y que haya completado el test de inducción al estrés en condición controlable. Cabe mencionar que siempre se asigna la condición controlable a un usuario antes de asignar la incontrolable a otro. En cuanto al método del "yugo", este tiene como objetivo lograr que la experiencia de los usuarios emparejados sea la misma. Así, al usuario incontrolable se le mostrará la secuencia idéntica de estímulos de retroalimentación que tuvo su pareja controlable, independiente de la secuencia de botones con la que el usuario decida experimentar. Es decir, el participante incontrolable, no percibirá un comportamiento determinístico de sus acciones, que le permita evitar los estímulos negativos.

Completar el test no toma mucho tiempo y su estructuración es simple. Un único bloque de 18 ensayos, consistentes en la aparición aleatoria de una de las 3 señales, en 6 ocasiones cada una.

Para los investigadores es fundamental conocer el comportamiento de los participantes, sus respuestas y el tiempo que utilizan para hacerlo, junto con las condiciones de cada ensayo, como la señal e imagen de feedback mostradas, independiente de la condición controlable o incontrolable.

4.2.2. Diseño

Los aspectos sustanciales de diseño lógico y algorítmico que se requirieron para implementar el test y sus componentes son los presentados a continuación.

Como se detalló recientemente, cada ensayo consta esencialmente de 2 fases: la vista donde se presenta una señal, junto a los 4 botones y la presentación del estímulo de retroalimentación. Esta última etapa, se planeó desplegar mediante un componente “modal”, que es una pantalla “flotante” que se superpone a la interfaz actual incluyendo una sutil animación de transición. En este caso, haciendo un desplazamiento vertical desde abajo hacia arriba al aparecer, para luego de un segundo desaparecer hacia abajo.

En adición a estas 2 fases, se agrega una tercera, igual a la “landing view” del test N-Back, que es una vista que solo presenta un símbolo “más (+)” durante un segundo, previo de comenzar cada ensayo. Este sirve de amortiguación al estímulo de retroalimentación y para volver a dirigir la mirada del participante a dicho punto, donde aparecerá la siguiente señal. Cabe mencionar que el registro del tiempo que utiliza el usuario en responder solo considera desde el momento que se despliega la vista que presenta la señal.

Así, estas 3 fases del ensayo e iteración entre las mismas, fueron diseñadas en base a la representación de 2 variables de estado: una que maneje la aparición del modal que muestra la imagen de retroalimentación y otra que indica el despliegue de la señal en la interfaz base, tal como se muestra en la figura 12.

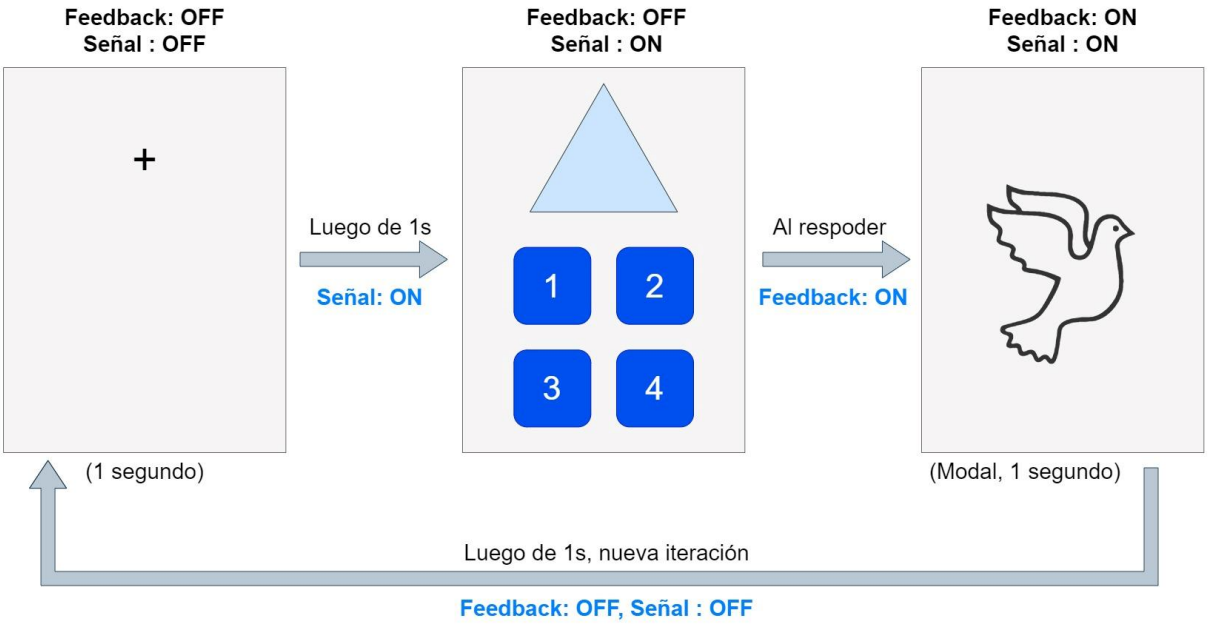

Figura 12. Estructura de ensayos del test de inducción al estrés y los estados que intervienen.

A diferencia del diseño del test anterior, donde la respuesta del participante ocurría paralela al transcurso de alguna fase delimitada por tiempo fijo (temporizador), aquí la respuesta marca el fin de la etapa de señal y el comienzo del despliegue del modal. Es decir, no hay caso de pérdida de respuesta (caso de “miss”), el usuario tiene tiempo libre para responder. Por lo tanto, no existe el problema de entrar en bucle o comportamiento no deseado por actualizaciones de estado gatilladas por el usuario en etapas gestionadas por un temporizador, de manera que no fue necesario incluir mecanismos de prevención de dicho caso. Aun así, como se ve en la figura 12, al final del ensayo se actualizan 2 estados en una etapa controlada por temporizador, lo que también ocasiona doble renderizado y arranque repetido del temporizador de la fase. Pero, en este caso al ser una actualización múltiple, fija (no aleatoria en el tiempo como la respuesta de un usuario) y se requiere de ambas al mismo tiempo, se diseñó un manejo de estados que no ocasiona este problema, de forma que se incluyesen las 2 variables de estado en una sola, utilizando el tipo de dato diccionario (pares clave-valor), como se muestra en el Algoritmo N°4, que también refleja el diseño del test mencionado hasta ahora.

Algoritmo N°4: Estructura general de fases y manejo de estados del test de Inducción al Estrés.

```
CANT_DE_ENSAYOS = N_SEÑALES * N_APARICIONES // 3 * 6 = 18
contadorEnsayos = 0

FUNCTION induccionAlEstres()
      estados = estadoInicial({señalVisible: false, modalVisible: false})

      IF estados.modalVisible == false THEN
            IF estados.señalVisible == false THEN
                  contadorEnsayos++
                  señalAMostrar = obtenerSeñal()
                  AFTER 1000ms DO
                        aux = estados
                        aux.señalVisible = true
                        estados = cambioDeEstado(aux)
                  END
            ELSE IF estados.señalVisible == true THEN
                  señalAMostrar = obtenerSeñal()
                  // luego, esperar respuesta del usuario
            ENDIF
      ELSE IF estados.modalVisible == true THEN
            IF contadorEnsayos < CANT_DE_ENSAYOS THEN
                  AFTER 1000ms DO
                        estados = cambioDeEstado({señalVisible: false,
                                                  modalVisible: false})
                  END
            ELSE
                  finDelTest()
            ENDIF
      ENDIF
      // función local
      FUNCTION alPresionarBoton(id_boton)
            cheackearRespuesta(id_boton)
            aux = estados
            aux.modalVisible = true
            estados = cambioDeEstado(aux)
      END
END
```

Este diseño es la base de las dos versiones del test: controlable e incontrolable. Se identificarán estas condiciones como C y U (de uncontrollable), respectivamente.

Para la versión controlable, se tienen las imágenes agrupadas en agradables y desagradables. Estas serán mostradas al azar y sin repetir, a menos que ya se hayan mostrado todas las del grupo correspondiente según el tipo de respuesta. Para ello se diseñó que el recorrido de los arreglos de imágenes fuera circular e inicialmente reordenarlos al azar, como se muestra en Algoritmo N°5. Cabe mencionar que todo está diseñado para que funcione con al menos una

imagen por grupo, ya que las imágenes deben ser aprobadas por el comité de ética, por ello no se tiene una selección y cantidad fija. Así mismo, estas no forman parte de los archivos en la aplicación (assets), sino que son descargadas previo al test desde el servidor, de manera temporal.

Algoritmo N°5: Manejo de imágenes a mostrar en versión C del test de Inducción al Estrés.

```
imgsAgradables = [0, 1, 2, 3, 4, 5] // id de imágenes
imgsDesagradables = [6, 7, 8, 9, 10, 11]

imgsAgradables = ordenAleatorio(imgsAgradables)
imgsDesagradables = ordenAleatorio(imgsDesagradables)

indAgradables = -1 //indice global del arreglo
indDesagradable = -1

FUNCTION recorridoCircular(arr, *indice)
      n = arr.length
      *indice = (*indice + 1) modulo n
      mostrarImagen(arr[indice])
END
```

En la versión incontrolable, no es necesario separar las imágenes por su etiqueta de agradable o desagradable, ya que de los datos capturados y enviados al servidor de la versión C emparejada, se obtiene el historial de los estímulos mostrados y solo basta recorrer dicho arreglo de imágenes de forma lineal conforme avanzan los ensayos, sin importar qué botón presione el participante para responder.

Con lo que respecta a la bifurcación en el transcurso de la aplicación hacia la versión C o U de este test, se planificó obtener dicha información en la pantalla de instrucciones del test. Allí se realiza una petición al servidor de los datos requeridos y así cuando el usuario decida comenzar el test luego de leer las instrucciones, se le redirija a la versión que corresponda.

Finalmente, los datos que se capturan para ser enviados al servidor son los siguientes

Etiqueta	**Tipo de dato**	**Descripción**
condición	{‘C’, ‘U’}	Condición de controlabilidad o incontrolabilidad del usuario
*asignacion_senal_boton	Diccionario (id_señal, id_boton)	Arreglo de pares clave-valor resultantes de la asignación aleatoria de botones a señales
*secuencia_estimulos	Arreglo de enteros	Identificadores de imágenes en orden de aparición

num_prueba	{1, 2, 3, .., 18}	Número del ensayo
senal_mostrada	{0, 1, 2}	Identificador de señal mostrada durante el ensayo. 0: polígono L; 1: circulo; 2: triangulo
imagen_mostrada	{1, 2, 3, ..}	Identificador de imagen mostrada durante el ensayo.
boton_respuesta	{1, 2, 3, 4}	Identificador del botón presionado por el usuario en el ensayo
tipo_respuesta	Booleano	True: respuesta correcta; False: respuesta incorrecta
fecha_inicio	Timestamp	Fecha y hora (DD/MM/AAAA HH:MM:SS.SSS) del inicio del ensayo
fecha_respuesta	Timestamp	Fecha y hora de la respuesta del usuario.
tiempo_respuesta	[ms]	Tiempo en milisegundos de la diferencia entre el instante de comienzo del ensayo y el de respuesta

(*) datos capturados solo en versión controlable. “asignacion_senal_boton” en condición U, se envía el id de dicha asignación entregada desde el servidor.

4. 2. 3 Resultados de implementación

Cabe mencionar que en las secciones anteriores se grafica a los botones con números para hacer más entendibles las situaciones explicadas, pero en realidad la interfaz final de los botones es con símbolos fijos y distintos a las señales.

A continuación, en la figura 13, se exponen los resultados respecto al funcionamiento y comportamiento de la implementación, contrastando la versión controlable, como con su pareja con incontrolabilidad.

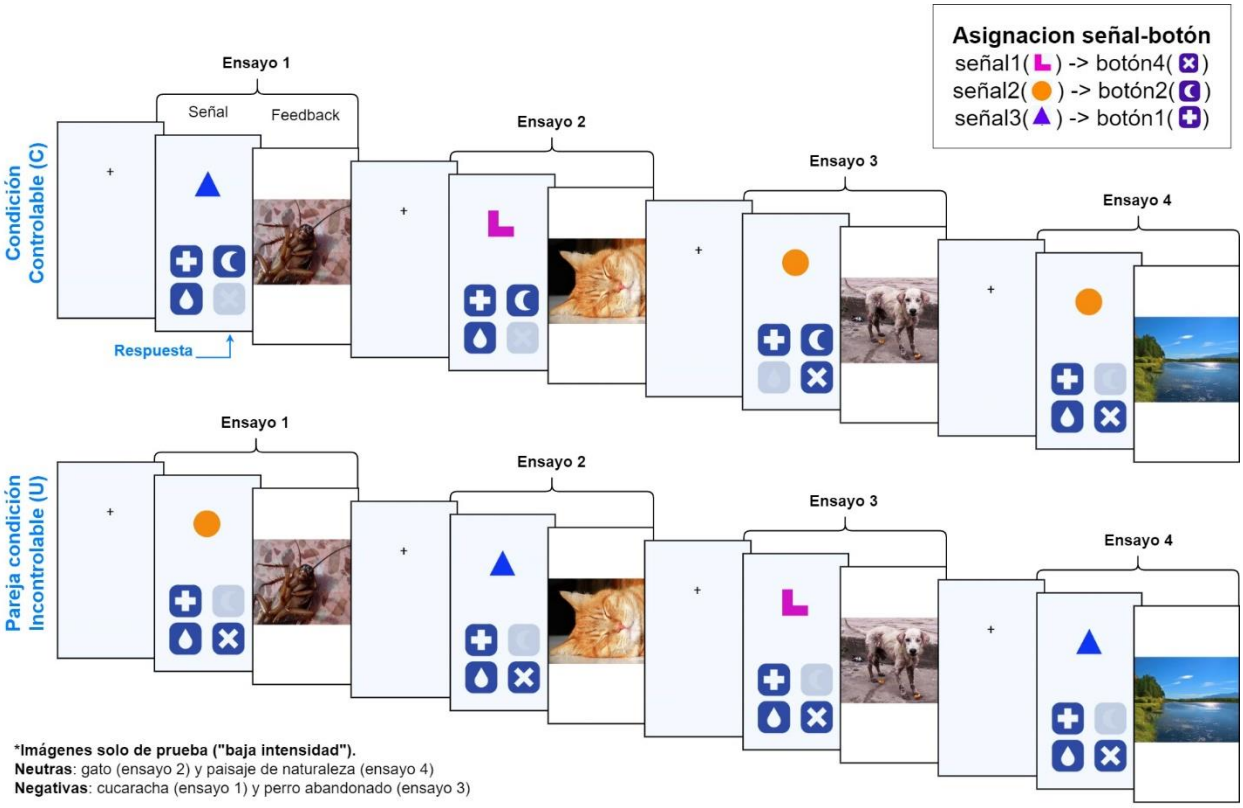

Figura 13. Resumen de funcionamiento de implementación en condición controlable y en pareja incontrolable del test de Inducción al Estrés.

Finalmente, se presenta un ejemplo de los datos capturado durante un ensayo (JSON resultante y graficado en jsongrid.com). En la figura 14 se muestran los resultados para un participante de condición C y en la figura 15 los resultados capturados del participante U emparejado. Notar que la columna de “imagen_mostrada” es la misma para las 2 versiones.

En la figura 14, para dato de “asignacion_senal_boton”, la herramienta utilizada para graficar los datos no entrega un resultado muy claro al incluir su propio índice de posición, pero lo que se debe interpretar es que a la señal 0, le corresponde el botón 4; a la señal 1, el botón 2 y a la señal 2 el botón con identificador 1. Este mismo dato para la versión U mostrada en la figura 15, solo muestra el identificador del participante C emparejado para referirse a dicha asignación.

id_participante	7
condicion	C
asignacion_senal_boton	[-] **asignacion_senal_boton [3]**

	0
1	4

	1
2	2

	2
3	1

secuencia_estimulos	[10, 9, 1, 12, 11, 4, 6, 2, 3, 5, 1, 8, 4, 7, 6, 2, 3, 5]

	num_prueba	senal_mostrada	imagen_mostrada	boton_respuesta	tipo_respuesta	fecha_inicio	fecha_respuesta	tiempo_respuesta
1	1	2	10	4	false	2023-07-10T07:24:27.390Z	2023-07-10T07:24:31.419Z	4029
2	2	2	9	4	false	2023-07-10T07:24:33.454Z	2023-07-10T07:24:34.368Z	914
3	3	0	1	4	true	2023-07-10T07:24:36.403Z	2023-07-10T07:24:37.400Z	997
4	4	1	12	3	false	2023-07-10T07:24:39.438Z	2023-07-10T07:24:41.351Z	1913
5	5	2	11	4	false	2023-07-10T07:24:43.386Z	2023-07-10T07:24:44.968Z	1582
6	6	1	4	2	true	2023-07-10T07:24:47.003Z	2023-07-10T07:24:48.267Z	1264
7	7	1	6	2	true	2023-07-10T07:24:50.303Z	2023-07-10T07:24:51.350Z	1047
8	8	0	2	4	true	2023-07-10T07:24:53.386Z	2023-07-10T07:24:54.351Z	965
9	9	0	3	4	true	2023-07-10T07:24:56.385Z	2023-07-10T07:24:57.401Z	1016
10	10	1	5	2	true	2023-07-10T07:24:59.437Z	2023-07-10T07:25:00.783Z	1346
11	11	0	1	4	true	2023-07-10T07:25:02.820Z	2023-07-10T07:25:04.016Z	1196
12	12	1	8	3	false	2023-07-10T07:25:06.054Z	2023-07-10T07:25:07.250Z	1196
13	13	0	4	4	true	2023-07-10T07:25:09.287Z	2023-07-10T07:25:11.067Z	1780
14	14	2	7	2	false	2023-07-10T07:25:13.102Z	2023-07-10T07:25:15.468Z	2366
15	15	1	6	2	true	2023-07-10T07:25:17.501Z	2023-07-10T07:25:18.467Z	966
16	16	2	2	1	true	2023-07-10T07:25:20.520Z	2023-07-10T07:25:21.950Z	1430
17	17	2	3	1	true	2023-07-10T07:25:23.987Z	2023-07-10T07:25:25.384Z	1397
18	18	0	5	4	true	2023-07-10T07:25:27.420Z	2023-07-10T07:25:32.167Z	4747

Figura 14. Resultados de un ensayo del test Inducción al Estrés, en condición controlable (C).

id_participante	8
condicion	U
asignacion_senal_boton	7

	num_prueba	senal_mostrada	imagen_mostrada	boton_respuesta	tipo_respuesta	fecha_inicio	fecha_respuesta	tiempo_respuesta
1	1	1	10	2	true	2023-07-10T07:25:56.356Z	2023-07-10T07:26:03.083Z	6727
2	2	2	9	2	false	2023-07-10T07:26:05.131Z	2023-07-10T07:26:07.583Z	2452
3	3	0	1	2	false	2023-07-10T07:26:09.619Z	2023-07-10T07:26:11.635Z	2016
4	4	2	12	2	false	2023-07-10T07:26:13.671Z	2023-07-10T07:26:14.618Z	947
5	5	0	11	2	false	2023-07-10T07:26:16.652Z	2023-07-10T07:26:17.184Z	532
6	6	2	4	2	false	2023-07-10T07:26:19.219Z	2023-07-10T07:26:20.050Z	831
7	7	0	6	2	false	2023-07-10T07:26:22.085Z	2023-07-10T07:26:22.635Z	550
8	8	2	2	2	false	2023-07-10T07:26:24.669Z	2023-07-10T07:26:25.117Z	448
9	9	1	3	2	true	2023-07-10T07:26:27.152Z	2023-07-10T07:26:27.633Z	481
10	10	1	5	2	true	2023-07-10T07:26:29.687Z	2023-07-10T07:26:30.201Z	514
11	11	2	1	2	false	2023-07-10T07:26:32.236Z	2023-07-10T07:26:32.867Z	631
12	12	0	8	2	false	2023-07-10T07:26:34.903Z	2023-07-10T07:26:35.367Z	464
13	13	0	4	2	false	2023-07-10T07:26:37.401Z	2023-07-10T07:26:37.850Z	449
14	14	2	7	2	false	2023-07-10T07:26:39.886Z	2023-07-10T07:26:40.400Z	514
15	15	1	6	2	true	2023-07-10T07:26:42.435Z	2023-07-10T07:26:42.834Z	399
16	16	1	2	2	true	2023-07-10T07:26:44.870Z	2023-07-10T07:26:45.367Z	497
17	17	0	3	2	false	2023-07-10T07:26:47.403Z	2023-07-10T07:26:47.901Z	498
18	18	1	5	2	true	2023-07-10T07:26:49.935Z	2023-07-10T07:26:50.667Z	732

Figura 15. Resultados de un ensayo del test Inducción al Estrés, en condición incontrolable (U) emparejada con la versión C mostrada en la figura 14.

4.3. Test de exploración y predicción

4.3.1. Análisis

Este es el último de los 3 test a ejecutar en la aplicación. Es la adaptación de una versión anterior del paradigma creada para ser utilizada con adultos [8]. Ahora incluye una narrativa amistosa de aviones, pilotos, colores e islas, que permite que el test pueda ser entendido incluso por niños.

Aquí, los participantes actúan como guías de viaje para otros pasajeros proporcionándoles información sobre vuelos a tres destinos: la isla del volcán, la isla de la palmera y la isla del faro. Hay tres aviones y son de colores diferentes: rosado, verde y naranjo. Entran en juego 2 pilotos, uno de ellos vuela al destino siguiendo una ruta específica, mientras que el otro lo hace en función del color del avión. En la figura 16 se ilustra esta situación. La idea es que los participantes, tal como indica el nombre del test, vayan explorando y puedan identificar cuál de estos 2 pilotos es el que está al mando del avión y así predecir un siguiente destino.

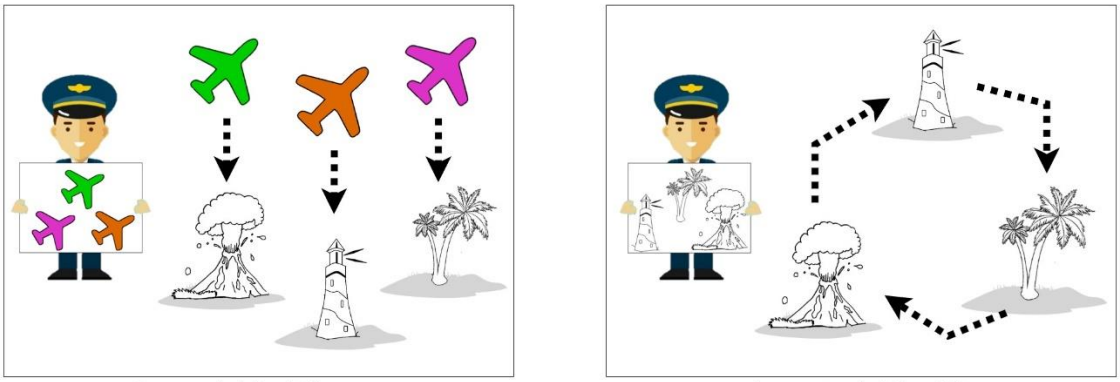

Figura 16. Reglas de que utilizan los pilotos C y U.

Para todos los participantes los pilotos siguen las mismas reglas. El piloto que llamaremos “de colores” solo depende del color del avión para trazar su destino. Por otro lado, el piloto “de ruta” solo considera la isla de despegue para dirigirse a su destino.

- Piloto de colores: Con avión verde vuela a isla del volcán; con avión naranjo vuela a isla del faro; con avión rosado vuela a isla de la palmera

- Piloto de rutas: Desde isla del faro vuela a isla de palmera; desde isla de palmera vuela a isla del volcán; desde isla del volcán vuela a isla del faro.

Como se habrá notado en la figura 16, estos pilotos están asociados a condiciones de controlabilidad: Controlable (C) para el piloto de colores e incontrolable (U) para el de rutas. Pero en esta ocasión la incontrolabilidad no es del modo introducido en el test anterior, de no poder controlar el resultado de sus respuestas, lo que gatillaba en no lograr comprender las relaciones o lo que en este caso sería no lograr predecir el piloto. Aquí más bien es referente a que la ruta ya está trazada, independiente del color de avión que se escoja. Y aunque de igual forma hay aspectos estocásticos que dificultan un poco la percepción y que serán mencionados más adelante, es importante mencionar que estos aspectos están presentes para los 2 pilotos. También cabe destacar que estos conceptos de controlabilidad no son mencionados a los participantes para explicarles sobre los pilotos y sus reglas.

Figura 17. Etapas que conforman un ensayo en test de Exploración y Predicción.

Para este test, cada ensayo se conforma esencialmente de 3 partes: exploración, predicción de estado y predicción de condición (ver figura 17). Primero, es importante mencionar que en cada ensayo hay presente una condición o piloto ya asignado internamente. En la etapa de exploración, son 6 iteraciones donde al usuario se le mostrará una isla, lo que también es llamado estado, junto con 2 opciones de avión y acorde a cuál de estos escoja y según la condición del ensayo, se le mostrará el destino al que viajaría el piloto incognito con dicho avión, lo que es llamado transición. La idea es que durante estas 6 iteraciones el participante vaya rescatando información sobre cuál piloto está al mando. La siguiente etapa es la de predicción de estado, aquí son 2 iteraciones y se le pide al participante indicar hacia donde volará a continuación un avión determinado. Se le muestra una isla de despegue (estado) y un avión, junto con las 3 islas para que el usuario responda. Finalmente, la última etapa, de predicción de condición o también llamada predicción de piloto, consta de un intento y precisamente se le pide al usuario responder cuál de los 2 pilotos cree que está dirigiendo los vuelos vistos en la etapa de exploración.

En la etapa de exploración descrita se presenta una particularidad, sólo uno de los dos aviones mostrados en cada prueba exploratoria podrá servir para diagnosticar la condición actual. Elegir el otro avión no es informativo, ya que volará a la misma isla tanto en la condición controlable como en la incontrolable. En la figura 18 se ilustra un ejemplo, donde escoger el avión rosado en la exploración de la parte 18.a llevará a la palmera, y como se indica en 18.b, esta elección no

es informativa ya que los 2 pilotos volarían hacia allá. Mientras que escoger el avión verde, sí permite discriminar la condición C o U.

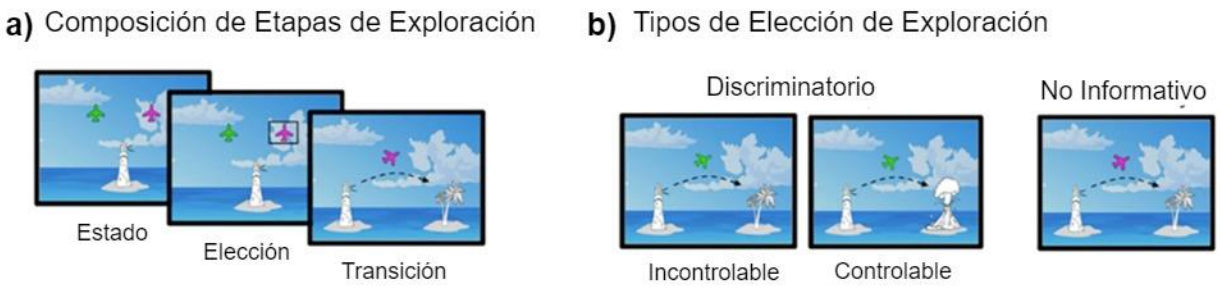

Figura 18: Composición de etapas de exploración y tipos de elección.

Por otro lado, la estructura completa del test consta de varias partes. Partiendo de que se van alternando las condiciones de C y U. El cambio entre condiciones se llama “reversión” y antes de cada reversión deben ocurrir entre 3 y 7 ensayos (ambos incluidos). Mediante 11 reversiones es posible formar grupos de 3 condiciones cada uno, llamados “Run”, como se ilustra en la figura 19. Así se dividen en total los 60 ensayos que dura el test.

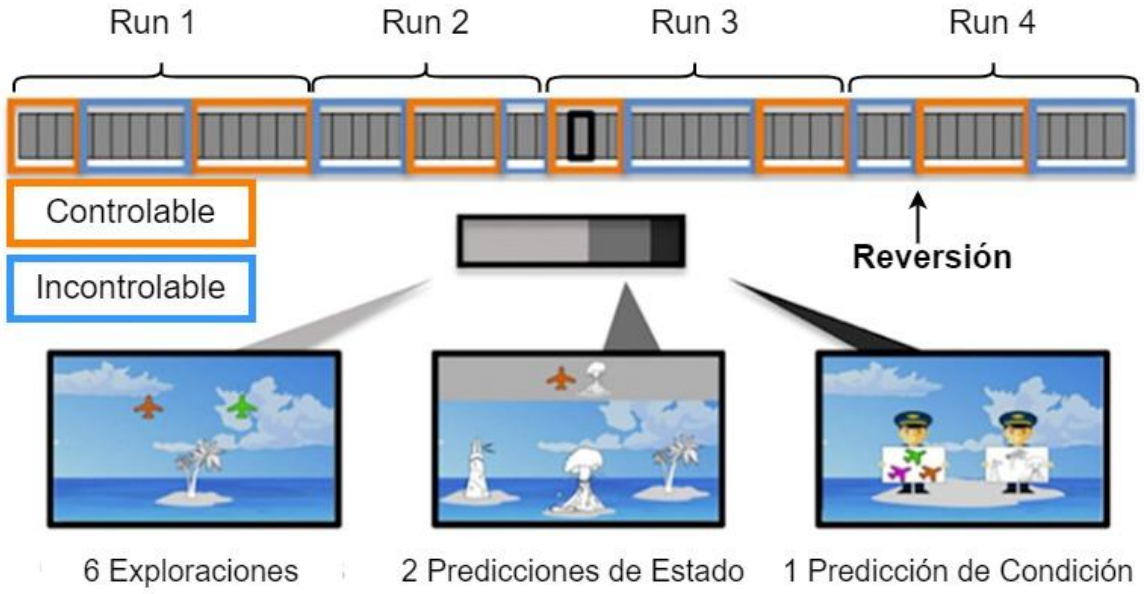

Figura 19. Estructura completa del test de Exploración y Predicción.

Sobre los aspectos estocásticos mencionados anteriormente, para que el aprendizaje resulte más desafiante en la etapa de exploración, las transiciones serán probabilísticas a lo largo de los 4 run de la tarea. Se explicó a los participantes que a veces hay turbulencias y el control del tráfico aéreo indica al piloto dónde volar, pero que la mayoría de las veces los pilotos siguen la pauta habitual. Durante el primer y tercer run, los aviones hacen la transición esperada el 90% de las veces, y se dirigen a cada una de las otras dos islas (la actual y la restante) el 5% de las veces (10% de ruido). En el segundo y cuarto run, cambian las probabilidades de transición para dificultar la tarea. El 80% de las veces, ejecutan la transición esperada, y el 10% de las veces, por cada una de las otras dos islas (20% de ruido).

Luego, también hay este tipo de situaciones en la etapa de predicción de estado. Fase en que los participantes son evaluados respecto a la información recabada en la exploración. En esta ocasión se agrega feedback respecto a la respuesta del participante. Para incentivar el aprendizaje, pero sin revelar muy evidentemente la condición subyacente, sólo se proporciona retroalimentación sobre una de las dos predicciones de estado, aleatoriamente. Las predicciones correctas pueden mostrar la imagen de un cofre lleno de tesoros y las respuestas incorrectas, la imagen de un cofre vacío. Se les dice a los participantes que cuando logran ayudar a otros pasajeros con sus predicciones, ellos pueden agradecer con el cofre lleno, pero si no predicen bien, ellos le entregan el cofre vacío, y que también en ocasiones los pasajeros no hacen nada independiente de su predicción.

4.3.2. Diseño

Para esta ocasión, cabe recordar que esta adaptación del test, con la narrativa de los pilotos, es el proyecto de un grupo de investigadores diferente al de StressMApp, los cuales tienen un prototipo programado en Matlab que prueban con usuarios en condiciones de laboratorio. Desde el grupo de investigadores de StressMApp, han estado en contacto con el grupo de investigadores de esta adaptación del test, facilitando este prototipo en Matlab [8], el cual aún no es público, para utilizar en este proyecto.

Con esta referencia no fue necesario diseñar algoritmos para

- Generar el cronograma de iteraciones de ensayos que alterna 11 veces la condiciones C y U y divide los 60 ensayos en 4 Run.
- Mapeo para la obtención de las 2 opciones de aviones a mostrar en la fase de exploración. Uno con resultado discriminatorio y el otro no informativo.
- Transiciones probabilísticas en la etapa de exploración.

Siendo así, los aspectos de diseño lógico y algorítmico que se verán en esta sección están más ligados a la estructuración y restante funcionamiento del test en consideración al framework de desarrollo.

Como los ensayos son conformados por las 3 fases mencionadas, y dichas fases a su vez incluyen sub-fases, se diseñaron 3 componentes principales (ver figura 20) que conforman los ensayos con sus respectivas iteraciones y agrupaciones en run. Lo que a su vez también es posible ya que el framework de desarrollo dispone de un canal de comunicación al momento de navegar entre vistas, lo que permite compartir datos y señales entre estos componentes.

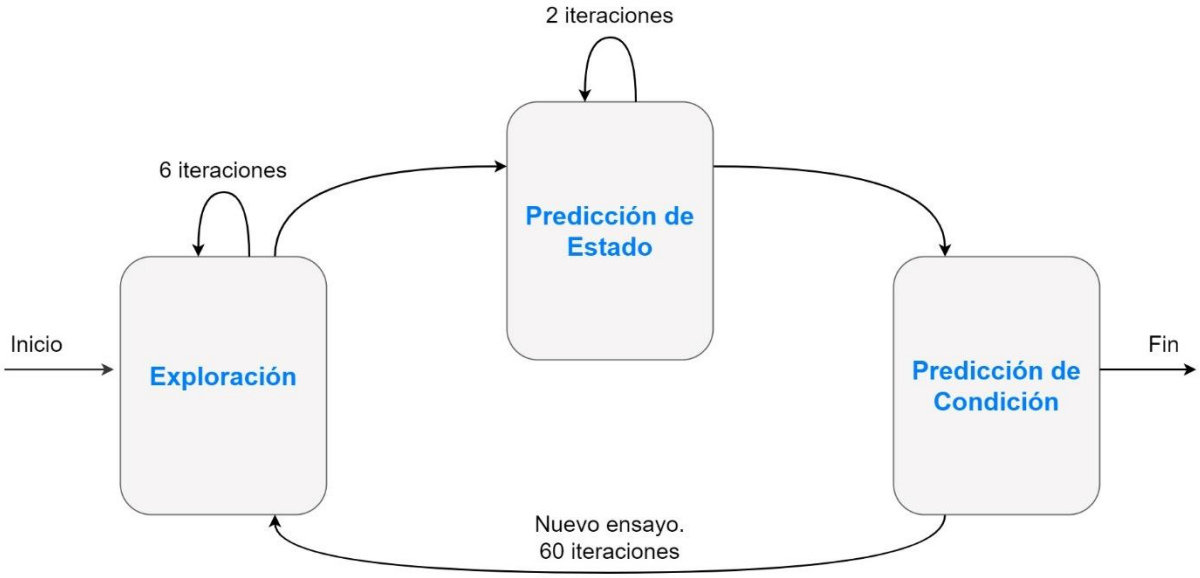

Figura 20. Diagrama de componentes y comunicación de ensayos en test de Exploración y Predicción.

Para los componentes de la fase de exploración y de predicción de estado, se diseñó utilizar el recurso del modal para mostrar la transición y el feedback respectivamente. Recurso que es gatillado al momento de responder mediante algún botón en la correspondiente etapa y cuyo contenido permanece visible durante un segundo.

Al ser la exploración el componente de entrada al test, desde aquí se inicializan y gestionan las variables globales que dirigen el test completo, como lo son: el contador de ensayos, de run, de reversiones y la condición de controlabilidad (C o U). Información que es requerida y desde aquí es comunicada a las etapas restantes.

Por otro lado, la representación interna de la fase de exploración solo requiere del manejo de una variable de estado y que gestiona la aparición del modal de transición (ver Algoritmo N°6).

Algoritmo N°6: Estructura de sub-fases y manejo de estado en etapa de exploración.

```
ITERACIONES_POR_ENSAYO = 6
contadorIteraciones = 0
id_islas = [0, 1, 2] //faro, palmera, volcán
isla = eleccionAleatoria(id_islas) //aleatoria en primera iteración

FUNCTION exploracion()
      modalVisible = estadoInicial(false)

      IF modalVisible == false THEN
            contadorIteraciones++
            islaAMostrar = isla
            avion1AMostrar = mapeoOpcion1(isla)
            avion2AMostrar = mapeoOpcion2(isla)
      ELSE
            AFTER 1000ms DO
                  IF contadorIteraciones+1 > ITERACIONES_POR_ENSAYO THEN
                        finEtapaExploracion()
                        //navega a sig etapa, de predicción de estado
                  ELSE
                        modalVisible = cambioDeEstado(false)
                  ENDIF
            END
      ENDIF

      // función local
      FUNCTION alPresionarBoton(id_boton)
            // retorna isla a la que se llegaría
            transicionAMostrar = obtenerTransicion(contadorRun, condicion,
                                                   isla, id_boton)
            isla = transicionAMostrar //isla a mostrar en la sig iteración
            modalVisible = cambioDeEstado(true)
      END
END
```

De manera similar se desarrolla la etapa de predicción de estado, aunque aquí el modal actúa como retroalimentador de la respuesta en solo una de las 2 iteraciones. Es decir, en una iteración puede mostrar feedback referente a si la respuesta fue correcta o no (cofre lleno o vacío, respectivamente) y en la siguiente iteración, solo mostrar el modal vacío, como se muestra en el Algoritmo N°7.

Algoritmo N°7: Estructura de sub-fases y manejo de estado en etapa de predicción de estado

```
//datos globales del ensayo
condicion = cond_del_ensayo //C o U
id_islas = [0, 1, 2] //faro, palmera, volcán
id_aviones = [0, 1, 2] //verde, rosado, naranjo

ITERACIONES_POR_ENSAYO = 2
contadorIteraciones = 0
islaEstado = eleccionAleatoria(id_islas) //aleatoria en primera iteración
avionEstado = eleccionAleatoria(id_aviones)
ordenDeOpciones = ordenAleatorio(id_islas)
iteracionConFeedback = eleccionAleatoria([1, 2])

FUNCTION prediccionDeEstado()
      modalVisible = estadoInicial(false)
      IF modalVisible == false THEN
            contadorIteraciones++
            islaAMostrar = islaEstado
            avionAmostrar = avionEstado
            opcionesAMostrar = ordenDeOpciones
      ELSE
            //nuevos estados diferentes para sig iteracion
            islaEstado = islaEstado + eleccionAleatoria([1,2]) modulo 3
            avionEstado = avionEstado + eleccionAleatoria([1,2])modulo 3
            ordenDeOpciones = ordenAleatorio(id_islas)
            AFTER 1000ms DO
                  IF contadorIteraciones+1 > ITERACIONES_POR_ENSAYO THEN
                        finEtapaPrediccionDeEstado()
                        //navega a sig etapa, de predicción de condición
                  ELSE
                        modalVisible = cambioDeEstado(false)
                  ENDIF
            END
      ENDIF
      FUNCTION alPresionarBoton(id_isla_respuesta)
            respCorrecta = obtenerRespuestaCorrecta()
            IF iteracionConFeedback == contadorIteraciones THEN
                  IF id_isla_respuesta == respCorrecta THEN
                        feedbackAMostrar = COFRE_LLENO
                  ELSE
                        feedbackAMostrar = COFRE_VACIO
                  ENDIF
            ELSE
                  feedbackAMostrar = NADA
            ENDIF
            modalVisible = cambioDeEstado(true)
      END
END
```

La función de “obtener respuesta correcta” del Algoritmo N°7, se detalla en el Algoritmo N°8. Se basa en las reglas de los pilotos, tomando como referencia el adecuado mapeo entre los aviones e islas con sus respectivos identificadores (id). Básicamente y como se ha mencionado antes, en la condición C, solo importa el avión de estado y para U, solo importa la isla de estado para llegar a una nueva isla de destino.

Algoritmo N°8: Función para obtener respuesta correcta en etapa de predicción de estado, basada en reglas de los pilotos (condición)

```
//datos globales del ensayo
condicion = cond_del_ensayo //C o U
id_islas = [0, 1, 2] //faro, palmera, volcán
id_aviones = [0, 1, 2] //verde, rosado, naranjo

islaEstado = eleccionAleatoria(id_islas)
avionEstado = eleccionAleatoria(id_aviones)

//retorna el id de la isla correcta acorde a la condición y estado
FUNCTION obtenerRespuestaCorrecta()
      IF condicion == C THEN
            RETURN respuestaCorrecta = 2 – avionEstado
      ELSE
            //condicion == U
            RETURN respuestaCorrecta = (islaEstado + 1) modulo 3
      ENDIF
END
```

Finalmente, la última etapa para completar un ensayo es la fase de predicción de condición. Se diseñó que aquí se fueran acumulando todos los datos del ensayo, hasta completar un run. Es decir, en exploración, donde inicia un ensayo, se establece también la condición, el número del ensayo, cuántas reversiones se llevan y a qué run corresponde. Estos datos, junto con los capturados del comportamiento del participante, son comunicados a la fase de predicción de estado, donde también se añaden las respuestas del usuario. Finalmente son enviados a la etapa final, donde luego de la respuesta del usuario, se completa toda la información de un ensayo. Conforme en la exploración se identifica que es el último ensayo para completar el run, se envía esta señal a la fase de predicción de piloto para que luego de la respuesta del usuario, se envíe el conjunto de datos de los ensayos hacia el servidor.

Se optó por hacer el envío de datos en cada run, en lugar de al finalizar el test, como se ha hecho en los 2 test anteriores, ya que es un test largo (60 ensayos) y hay riesgo de abandono. Así, en el caso que esto ocurra, los investigadores cuenten con los datos de estos run completados, que no dejan de ser importantes.

El funcionamiento de la etapa de predicción de condición es simple, no hay manejo de estado ni visualización de alguna pantalla más. Así que todo se gatilla desde la función que utilizan los botones de respuesta (ver Algoritmo N°9).

Algoritmo N°9: Funcionamiento de fase de predicción de condición

```
//datos globales del ensayo
RUN_TOTALES = 4
contadorRun = runActual
condicion = cond_del_ensayo //C o U
esUltimoEnsayoDelRun = señalDeUltimoEnsayo

FUNCTION alPresionarBoton(id_boton)
      checkearRespuesta(id_boton, condicion)
      IF esUltimoEnsayoDelRun == true THEN
            enviarDatosDeEnsayos()
            IF contadorRun == RUN_TOTALES THEN
                  finalizarTest()
            ELSE
                  comenzarNuevoEnsayo() //navega a exploración, nuevo run
            ENDIF
      ELSE
            comenzarNuevoEnsayo() //navega a exploración
      ENDIF
END
```

Para finalizar, los datos que se capturan para ser enviados al servidor son los siguientes

Etiqueta	**Tipo de dato**	**Descripción**
id_run	{1, 2, 3, 4}	Número del run
num_prueba	{1, 2, 3, .., 60}	Número del ensayo
condicion	{1, 2}	*Identificador de la controlabilidad del ensayo
frec_reversion_cond	Arreglo de 3 enteros	Número de ensayos a realizar por condición en un run
num_prueba_condicion	{1, 2, 3, …, 7}	Número del ensayo antes de la reversión
reversiones	{0, 1, 2}	Número de reversiones en el run
num_prueba_explore	{1, 2, 3, …, 6}	Número de iteración en la fase de exploración
noise	{0.1, 0.2}	(Exploración) Porcentaje de ruido en la transición probabilística
isla_llegada_afectada_noise	Booleano	(Exploración) Indica si la isla resultante de la transición probabilística fue afectada
isla_mostrada	{0, 1, 2}	Identificador de la isla de estado

isla_llegada	{0, 1, 2}	(Exploración) *Identificador de la isla mostrada en la transición
avion_1	{0, 1, 2}	(Exploración) *Identificador del avión mostrado como opción 1 o izquierda
avion_2	{0, 1, 2}	(Exploración) *Identificador del avión mostrado como opción 2 o derecha
avion_elegido	{0, 1, 2}	(Exploración) *Identificador del avión seleccionado por el participante
num_prueba_predict	{1, 2}	Número de iteración en fase de predicción de estado
isla_respuesta	{0, 1, 2}	(Pred. de estado) *Identificador de la isla seleccionada por el participante
avion	{0, 1, 2}	(Pred. de estado) *Identificador del avión mostrado
recompensa	Booleano	(Pred. de estado) Indica si se mostró feedback en el ensayo
piloto_respuesta	{1, 2}	(Pred. de condición) *Identificador del piloto seleccionado por el usuario.
tipo_respuesta	Booleano	Indicador de respuesta correcta
fecha_inicio	Timestamp	Fecha y hora (DD/MM/AAAA HH:MM:SS.SSS) del inicio del ensayo
fecha_respuesta	Timestamp	Fecha y hora de la respuesta del usuario
tiempo_respuesta	[ms]	Tiempo en milisegundos de la diferencia entre el instante de comienzo del ensayo y el de respuesta

(*) Los identificadores señalados de islas, aviones y de las condiciones o pilotos, corresponden a las siguientes:

ID	Isla
0	Faro
1	Palmera
2	Volcán

ID	Avión
0	Verde
1	Rosado
2	Naranjo

ID	Condición/Piloto
1	Incontrolable (U)
2	Controlable (C)

4.1.3. Resultados de implementación

A continuación, se exponen los resultados respecto al funcionamiento y comportamiento de la implementación.

En la figura 21 se ilustra una abstracción del funcionamiento completo del test, basándose en la estructura de un ensayo. Pasando por 6 iteraciones de exploración, presentadas al usuario como “Juego de Exploración”, luego 2 iteraciones de predicción de estado y una predicción de condición o “Predicción de Piloto” como se le muestra al participante”. Se ejecutan 60 ensayos, divididos en 4 run y no hay cambios en la interfaz al ocurrir una reversión de condición.

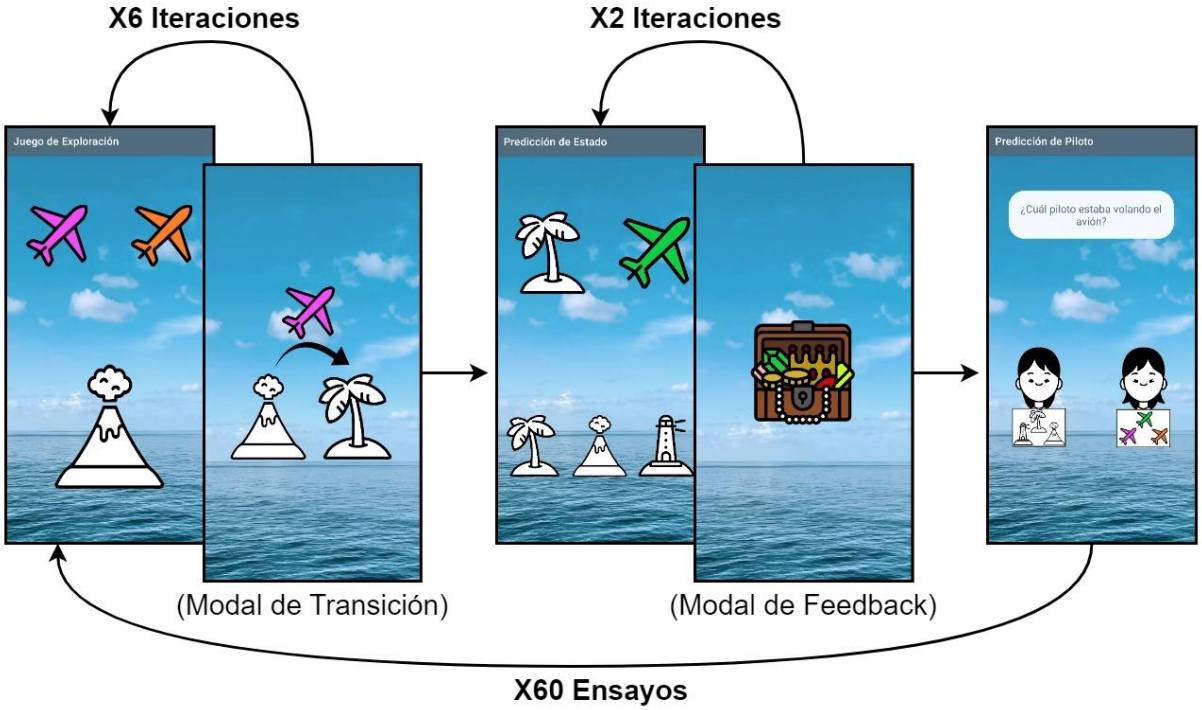

Figura 21. Estructura de un ensayo y abstracción de funcionamiento resultante del test de Exploración y Predicción.

Durante la exploración, es posible que el ruido predefinido para cada run afecte a la transición mostrada al usuario, lo que en el contexto entregado al participante sería una “turbulencia” y otorga un resultado de transición errónea para la condición presente. En el ejemplo ilustrado en la figura 22, se muestra esta situación considerando que el piloto que dirige es el de condición controlable (C).

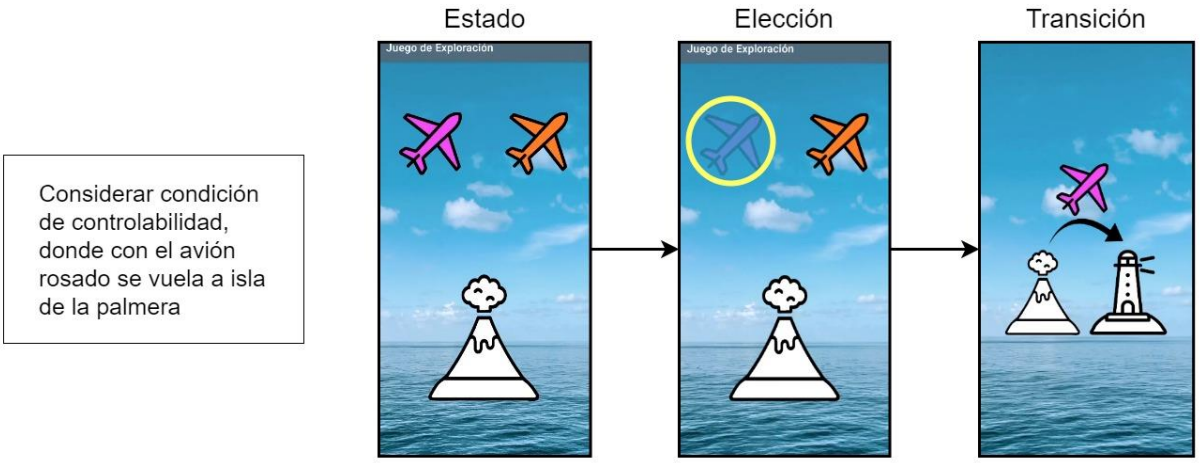

Figura 22. Ejemplo de “turbulencia”. Transición afectada por el ruido en etapa de exploración.

Por otro lado, en la predicción de estado, y como se ejemplifica en la figura 23, ocurre la situación donde se despliega aleatoriamente en una de las 2 iteraciones, feedback referente a si la respuesta fue correcta o errónea mediante un cofre lleno como el de la figura 21 o un cofre vacío como es este caso, respectivamente. Para este ejemplo el feedback se muestra en la primera iteración, por la tanto, ante la segunda respuesta no se presentará retroalimentación.

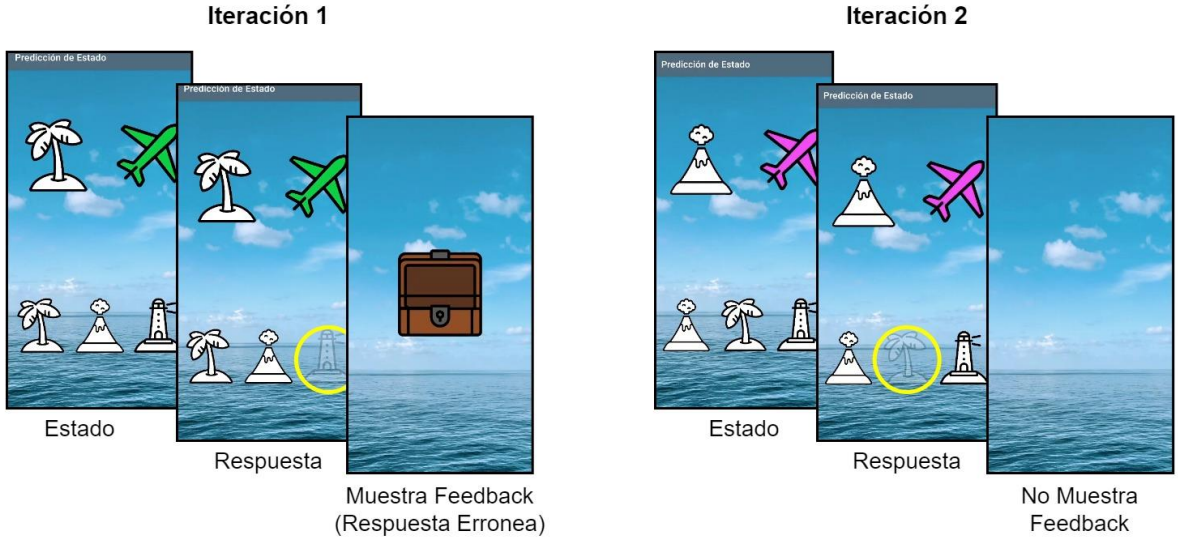

Figura 23. Ejemplo de despliegue de feedback en etapa de predicción de estado.

Finalmente, se presenta un ejemplo de los datos capturado durante un run (JSON resultante y graficado en jsongrid.com). En la figura 22 se muestra la estructuración macro de los datos para posteriormente en las figuras 23, 24 y 25 expandir los resultados de las etapas de exploración, predicción de estado y predicción de condición, respectivamente del ensayo 1.

id_participante	16
id_run	1
frec_reversion_cond	[-] **frec_reversion_cond [3]** 1: 5 2: 6 3: 4
noise	0.1
run	[-] **run [15]**

	num_prueba	reversiones	exploratory	island_prediction	pilot_prediction
1	1	0	[+] exploratory [6]	[+] island_prediction [2]	[+] pilot_prediction [1]
2	2	0	[+] exploratory [6]	[+] island_prediction [2]	[+] pilot_prediction [1]
3	3	0	[+] exploratory [6]	[+] island_prediction [2]	[+] pilot_prediction [1]
4	4	0	[+] exploratory [6]	[+] island_prediction [2]	[+] pilot_prediction [1]
5	5	0	[+] exploratory [6]	[+] island_prediction [2]	[+] pilot_prediction [1]
6	6	1	[+] exploratory [6]	[+] island_prediction [2]	[+] pilot_prediction [1]
7	7	1	[+] exploratory [6]	[+] island_prediction [2]	[+] pilot_prediction [1]
8	8	1	[+] exploratory [6]	[+] island_prediction [2]	[+] pilot_prediction [1]
9	9	1	[+] exploratory [6]	[+] island_prediction [2]	[+] pilot_prediction [1]
10	10	1	[+] exploratory [6]	[+] island_prediction [2]	[+] pilot_prediction [1]
11	11	1	[+] exploratory [6]	[+] island_prediction [2]	[+] pilot_prediction [1]
12	12	2	[+] exploratory [6]	[+] island_prediction [2]	[+] pilot_prediction [1]
13	13	2	[+] exploratory [6]	[+] island_prediction [2]	[+] pilot_prediction [1]
14	14	2	[+] exploratory [6]	[+] island_prediction [2]	[+] pilot_prediction [1]
15	15	2	[+] exploratory [6]	[+] island_prediction [2]	[+] pilot_prediction [1]

Figura 22. Ejemplo de estructuración macro de los datos capturados de un run del test de Exploración y Predicción.

En la figura 22 se puede notar que la “frecuencia de reversión de condición” indica que ese run estará conformado por 15 ensayos, donde los primeros 5 serán en condición C, luego 6 en U y finaliza revirtiendo a condición C por 4 ensayos. También que el ruido o “noise” presente es del 10% y que como se refleja en la figura 23, este afecta a la transición mostrada en la interacción 2 (num_prueba_explore 2), ya que se escogió el avión con id 1 (rosado), el cual en condición controlable vuela a la isla con id 1 (palmera) pero se le muestra en la transición, la isla de id 0 (faro).

[-] exploratory [6]

	num_prueba_condicion	num_prueba_explore	condicion	isla_llegada_afectada_noise	isla_mostrada	isla_llegada	avion_1	avion_2	avion_elegido	fecha_inicio	fecha_respuesta	tiempo_respuesta
1	1	1	2	false	0	2	0	1	0	2023-08-10T01:53:18.041Z	2023-08-10T01:53:28.248Z	10207
2	1	2	2	true	2	0	1	2	1	2023-08-10T01:53:29.266Z	2023-08-10T01:53:40.049Z	10783
3	1	3	2	false	0	2	0	1	0	2023-08-10T01:53:41.066Z	2023-08-10T01:53:49.265Z	8199
4	1	4	2	false	2	1	1	2	1	2023-08-10T01:53:50.284Z	2023-08-10T01:53:57.048Z	6764
5	1	5	2	false	1	0	0	2	2	2023-08-10T01:53:58.069Z	2023-08-10T01:54:02.833Z	4764
6	1	6	2	false	0	1	1	0	1	2023-08-10T01:54:03.851Z	2023-08-10T01:54:12.549Z	8698

Figura 23. Ejemplo de datos obtenidos en la etapa de exploración, capturados del primer ensayo mostrado en la figura 22.

[-] island_prediction [2]

	num_prueba_predict	isla_mostrada	isla_respuesta	avion	tipo_respuesta	recompensa	fecha_inicio	fecha_respuesta	tiempo_respuesta
1	1	1	2	0	1	1	2023-08-10T01:54:13.612Z	2023-08-10T01:54:20.183Z	6571
2	2	2	1	1	1	0	2023-08-10T01:54:21.204Z	2023-08-10T01:54:31.931Z	10727

Figura 24. Ejemplo de datos obtenidos en la etapa de predicción de estado, capturados del primer ensayo mostrado en la figura 22.

Para este ejemplo, del primer ensayo del run, se refleja en la figura 25 que la respuesta fue dirigida hacia el piloto 2 o de condición controlable, arrojando una respuesta correcta, reflejado en la columna “tipo_respuesta”.

[-] pilot_prediction [1]

	piloto_respuesta	tipo_respuesta	fecha_inicio	fecha_respuesta	tiempo_respuesta
1	2	1	2023-08-10T01:54:33.013Z	2023-08-10T01:54:38.957Z	5944

Figura 25. Ejemplo de datos obtenidos en la etapa de predicción de condición, capturados del primer ensayo mostrado en la figura 22.

5. Evaluación de la propuesta

La evaluación de la propuesta fue llevada a cabo de manera iterativa y de forma particular por cada test, así como con los 3 test en conjunto. Primeramente, se fueron presentando durante algunas de las reuniones semanales los resultados finales de cada desarrollo, de forma telemática, es decir, se proyectaba la ejecución del test frente al equipo. Luego, finalizados los 3 test y otros aspectos de la aplicación e infraestructura de despliegue, se procede a generar archivos de instalación de la aplicación en formato APK (Android Package Kit) para que todo el equipo pudiera probar el prototipo integral de la aplicación en algún dispositivo móvil con sistema operativo Android. Esta última etapa de pruebas con el prototipo integral tuvo 3 iteraciones.

El mecanismo de evaluación de la propuesta fue utilizando el método de pruebas de funcionalidad. Estas se centran en verificar si el software cumple con los requisitos y especificaciones analizadas. En este caso, las pruebas son de forma manual llevadas a cabo tanto en las proyecciones telemáticas como en el prototipo desplegado, donde el equipo completo, especialmente los investigadores proceden a la inspección, verificando que todas las funciones y características funcionan correctamente, según lo diseñado y que no haya errores o comportamientos inesperados.

Cabe destacar que a pesar de que la evaluación mediante la proyección telemática de la ejecución puede parecer menos eficaz frente a la ejecución del prototipo de manera personal, no resulta así, porque de la primera forma se obtiene una visualización “interna” del comportamiento de los test, ya que se proyecta el ambiente de desarrollo completo, que incluye la consola donde se van imprimiendo en tiempo real los registros (logs) que indican los estados, repuestas, resultados y demás información que refleja lo que va ocurriendo durante la ejecución. Esto no es posible de percibir desde el prototipo en el dispositivo móvil. Además, en los test hay situaciones y comportamientos diseñados para que el usuario no se percate, como el tipo de condición de controlabilidad presente o resultados estocásticos, por lo que evaluar el test de esta manera, con la lectura paralela de los registros identificando dichas particularidades, también fue favorable.

De estas pruebas se han obtenido algunas modificaciones a los test y también se ha puesto el foco en algunos aspectos, que se mencionan a continuación:

- En N-Back se tenía la percepción de que el tiempo para responder en los ensayos del último bloque (3-Back) era menor a los bloques anteriores. Finalmente, esto fue solo percepción al ser la versión más difícil del test y requerir más concentración. También se verifica que el tiempo por ensayo siempre es de 3 segundos mediante los timestamps de cada ensayo. De igual forma, está en análisis por parte de los investigadores, modificar el tiempo para responder a los ensayos.

- En Inducción al Estrés, que recordemos es el test más experimental de los 3, ocurrieron modificaciones, ya que originalmente el test funcionaba con 4 señales y 6 botones, pero

resultaba ser muy complejo identificar las relaciones para evitar un estímulo negativo, así que se redujo finalmente a 3 señales y 4 botones.

- En Exploración y Predicción, se tiene en vista reducir la duración del test, que recordemos son 60 ensayos. Además de la cantidad de ensayos, se propone reducir las iteraciones de exploración y la cantidad de bloques run, lo que está en análisis por parte de los investigadores.

Otro punto importante al probar los test, en particular mediante el prototipo de la aplicación, es verificar que la comunicación con el servidor se ejecute correctamente. En este caso, se envían las respuestas correspondientes de cada test y desde la interfaz del servidor se verificó que se ejecuta correctamente.

Durante las ejecuciones de los test por parte del equipo de StressMApp no se han encontrado comportamientos no deseados. Se verificó la correcta captura de datos en los ensayos y comportamiento de los participantes, enviada al servidor, reflejando la correcta ejecución de las particularidades de cada test.

Fuera de los aspectos en discusión mencionados, de parte de los investigadores, y del equipo de desarrollo, se ha obtenido una evaluación favorable en lo que respecta a las funcionalidades y expectativas. Se cumple con los requisitos y especificaciones analizadas de cada test.

De momento no se han podido extender las pruebas hacia los usuarios objetivos (estudiantes), producto de tramitaciones no concluidas por parte del Comité de Ética con respecto al proyecto en general. Probar la aplicación sin autorización del Comité de Ética no resulta en datos válidos para los investigadores.

6. Conclusiones

Al finalizar esta memoria de título, se consideran cumplidos los objetivos exceptuando las pruebas con los usuarios finales (estudiantes), ya que parte de la idea era que el grupo multidisciplinario detrás de este proyecto tuviera sus primeras capturas de datos para sus investigaciones y postulaciones a fondos concursables. Esto es por motivos externos al grupo del proyecto, debido a la tardanza de aprobación por parte del Comité de Ética del Departamento de Psiquiatría para el pilotaje.

Fuera de esta situación, el prototipo integral y funcional de la aplicación se ha podido probar internamente con los integrantes del grupo multidisciplinario del proyecto. Particularmente, el desarrollo de los 3 test tipo juego ha sido bien valorado, pasando las pruebas de funcionalidad, lo que resulta de mucha importancia respecto a los objetivos de esta memoria de título, de que los test cumplan con todas las funcionalidades y expectativas.

Sumado a lo anterior, y considerando que es el primer prototipo funcional, también se considera valioso poder dejar un producto abordado con la seriedad y estándar que amerita este instrumento de investigación. Se logró un resultado con diseño sólido e implementación ordenada basada en las buenas prácticas de programación adquiridas durante la carrera, que de seguro servirá de base para iteraciones futuras.

Por otro lado, en lo que respecta a los principales obstáculos y dificultades que se tuvieron durante el proceso, se detallan a continuación.

Técnicamente, a pesar de que se tenía algo de experiencia con las tecnologías utilizadas, desde el comienzo del desarrollo del primer test (N-Back), encontrarse con los conflictos de actualización de múltiples estados y utilizar temporizadores que además gatillaran dichas actualizaciones, fue bastante complejo de controlar y mitigar. Pero luego de haber comprendido profundamente el funcionamiento de todos los aspectos que entraban en juego, sirvió para abordar de mejor forma las estructuras y manejo de estados de los siguientes test.

Luego, conforme transcurrió tiempo y también los avances de lo otros aspectos del proyecto, la incorporación de los 3 test en la aplicación integral, dificultó las pruebas particulares de los mismos en etapa de implementación, producto de que el funcionamiento de la aplicación es progresivo, se requiere ir completando etapas para poder acceder a las otras y también se había incluido la autentificación de usuarios, necesaria para identificar a los participantes y sus respuestas. Por ello, las pruebas para verificar los aspectos de funcionamiento de los test y de su comunicación hacia el servidor en esta integración, requerían ingresar clave y avanzar dentro de otros apartados en la aplicación. Por ende, para agilizar el debugging y la comprobación de los aspectos señalados, se tuvo que invertir tiempo en automatizar estos procesos previos y así poder ingresar directamente a los test en particular dentro de esta aplicación integral.

Ya en etapas de pruebas del prototipo mediante el despliegue de la apk, una condición que dificultó un poco esta fase, fue que no todos los integrantes del equipo del proyecto contaran con un dispositivo móvil con sistema operativo Android y debían conseguirlo, haciendo así que obtener la retroalimentación completa del equipo tomara más tiempo del esperado.

Un aspecto crítico desde un punto de vista personal, que guarda cierta relación con lo mencionado anteriormente y que surgió tras el descubrimiento de un sitio web llamado “Snack Expo”, que proporciona un entorno de desarrollo en línea (incluyendo consola, editor y emulador) para el framework React Native, es la siguiente reflexión: A pesar de que la evaluación de la propuesta demostró satisfactoriamente las funcionalidades de las pruebas y se cumplieron las expectativas, hubiera sido sumamente enriquecedor brindar al equipo un enlace que les permitiera ejecutar individualmente las pruebas en línea, sin requerir ninguna instalación. Recordando que las pruebas de funcionalidad se llevaron a cabo de manera remota, proyectando una ejecución, así como posteriormente a través de la apk. La utilización de este recurso habría combinado los beneficios de ambas versiones de las pruebas de funcionalidad: por un lado, la disponibilidad de una consola que muestra en tiempo real los registros de lo que ocurre en la aplicación, y por otro lado, la posibilidad de interactuar individualmente con las pruebas.

Así y pese a que no se ha solicitado, se considera conveniente y práctico entregar una versión de cada test adecuada a la plataforma mencionada, para que ante cualquier inquietud o experimentación sobre las funcionalidades, estructura o parámetros, puedan ejecutar y probar directamente desde allí de forma sencilla.

Para finalizar, y más allá de lo recién mencionado a implementar, se considera que el trabajo futuro a realizar particularmente sobre los test, es principalmente converger hacia las duraciones adecuada de cada uno de estos 3 test y también poder realizar un pilotaje con participante reales y así obtener información adecuada que permita continuar con las investigaciones, posibles nuevas versiones de la aplicación con despliegues para sistema operativo Android y iOS y presencia en las tiendas virtuales de las mismas.

7. Bibliografía

[1] VRID Universidad de Concepción. (27 de Abril de 2022). UdeC financiará 25 nuevos proyectos multidisciplinarios. Recuperado de https://noticias.udec.cl/udec-financiara-25-nuevosproyectos-multidisciplinarios/

[2] Universidad CESUMA (sf) ¿Qué es el framework? Recuperado de https://www.cesuma.mx/blog/que-es-el-framework.html

[3] Budziński, M. (2023) What is react native?. Recuperado de https://www.netguru.com/glossary/react-native

[4] MDN. (1 de mayo de 2023). JavaScript. Recuperado de https://developer.mozilla.org/en-US/docs/Web/JavaScript

[5] Meta Open Source. (2023) React. Recuperado de https://react.dev/

[6] Neureka. (2023) Taking research out of the lab and into your smartphone. Recuperado de https://www.neureka.ie/

[7] N-Back Challenge. (2023) Raise your IQ, in a few minutes per day, by taking our 20-day N-Back Challenge. Recuperado de https://nbackchallenge.com/

[8] Raab HA, Foord C, Ligneul R, Hartley CA (2022) Developmental shifts in computations used to detect environmental controllability. PLoS Comput Biol 18(6): e1010120. https://doi.org/10.1371/journal.pcbi.1010120

[9] GitHub. (Diciembre de 2022). The top programming languages. Recuperado de https://octoverse.github.com/2022/top-programming-languages

[10] JSON.org. (s.f.) Introducing JSON. Recuperado de https://www.json.org/

[11] Geeks for Geeks. (27 de enero de 2023). Software Engineering - Software Design Process. Recuperado de https://en.wikipedia.org/wiki/Software_design

[12] Kane, M.J. & Conway, A. (2007). Working memory, attention control, and the N-back task: A question of construct validity. Journal of Experimental Psychology: Learning, Memory, and Cognition, 33(3), 615-622.

[13] Ligneul, Romain & Mainen, Zachary & Ly, Verena & Cools, Roshan. (2022). Stress-sensitive inference of task controllability. Nature Human Behaviour. 6. 1-11. 10.1038/s41562-022-01306-w.

[14] International Affective Picture System (IAPS). (s.f.) Center for the Study of Emotion and Attention. Recuperdado de https://csea.phhp.ufl.edu/media/iapsmessage.html