AŇO 2020

3300 - 001853 / 19 - 003

Creado: 22-10-2020

Iniciador: MESA DE ENTRADAS - INFORMATICA
DA COSTA FARO JULIAN MATHIAS

Extracto: E/TRABAJO FINAL DE TESINA DE GRADO TITULADA: "MODERACION REMOTA DE PRUEBAS DE USUARIO CON SOPORTE PARA A/B TESTING Y FOCO EN USABILIDAD WEB", BAJO LA DIRECCION DEL DR. ALEJANDRO FERNANDEZ Y LA CODIRECCION DEL DR. JULIAN GRIGERA.-
--

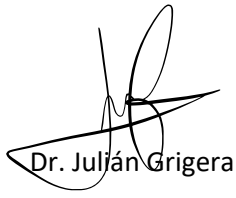
La Plata, 21 de Octubre de 2020

A la Sra. Decana, Lic. Patricia Pesado

Avalamos, por considerarla concluida, la presentación de la tesina de grado del señor *Julián da Costa Faro* titulada: "*Moderación remota de pruebas de usuario con soporte para A/B Testing y foco en usabilidad web*".

Saludo a usted atentamente.

Dr. Alejandro Fernandez



Dr. Julián Grigera

UNIVERSIDAD NACIONAL DE LA PLATA

FACULTAD DE INFORMÁTICA

TESINA DE LICENCIATURA

TÍTULO: Moderación remota de pruebas de usuario con soporte para A/B Testing y foco en usabilidad web

AUTORES: Julián Mathías da Costa Faro

DIRECTOR: Dr. Alejandro Fernández

CODIRECTOR: Dr. Julián Grigera

ASESOR PROFESIONAL: -

CARRERA: Licenciatura en Sistemas

Resumen

Una de las técnicas más utilizadas para hallar problemas de usabilidad son las pruebas de usuario, cuyo objetivo es observar a un grupo de voluntarios mientras realizan tareas típicas en una aplicación con el fin de detectar problemas que dificultan la interacción con el sistema. Al mismo tiempo, existen los tipos de prueba A/B Testing que es una técnica para realizar experimentos en la que se cuenta con dos versiones ("A" y "B") de una misma solución a un problema y, en base a pruebas, se evalúan los resultados para poder comparar ambas versiones. Las pruebas de usuario remotas fueron ganando terreno con el correr de los años. En la presente tesina se busca atacar el desafío de aplicar A/B Testing sobre pruebas de usuario remotas, capturando métricas durante las prueba, aplicando algoritmos de distribución y permitiendo la comparación de los datos obtenidos.

Palabras Clave

- Pruebas de usuario
- Moderación de pruebas remotas
- A/B Testing
- Split test

Conclusiones

En base a los objetivos propuestos se alcanzó la premisa fundamental de extender la herramienta Tycho para aplicar A/B Testing sobre pruebas de usuario. Para ello fue necesario realizar distintas extensiones a lo largo de toda la herramienta que permita comparar dos o más pruebas de usuario que se encuentren bajo la misma sesión de experimento, capturar nuevas métricas durante las ejecuciones y llevar a cabo distintos métodos para la visualización de resultados. Además se crearon estrategias para balancear la distribución de las pruebas que se encuentren bajo una misma sesión de experimentos. Se demuestra la herramienta en funcionamiento con las extensiones mediante casos de prueba.

Trabajos Realizados

Se extendió la herramienta Tycho con las siguientes características:

- *Habilitación de soporte de A/B Testing*
- *Estrategias de distribución de experimentos*
- *Captura de nuevas métricas durante las ejecuciones*
- *Comparación de resultados a través de distintos métodos de visualización*

Trabajos Futuros

- *Introducir nuevas métricas.*
- *Mejorar la presentación de las tareas a los voluntarios que ejecutarán las pruebas.*
- *Brindarle al experto alternativas de distribución de las pruebas.*
- *Habilitar la captura de la pantalla tanto en imágenes como en video para un posterior análisis.*

Fecha de la presentación: Octubre 2020

Índice general

Agradecimientos

En primer lugar, quiero agradecerles a Alejandro Fernández y Julián Grigera por permitirme participar en uno de sus proyectos, por la predisposición y el tiempo dedicado durante el desarrollo de este trabajo.

A mis padres, por brindarme el apoyo constante e incondicional durante toda mi vida, sin ellos nada de esto sería posible.

A mi hermano Dario, por tener siempre la palabra justa ante cada decisión que tenga que tomar.

A mi hermano Andres, por abrirme los ojos al momento de elegir esta carrera y por ser un gran compañero en todos los aspectos.

A mi novia Gisella, por aparecer en mi vida y bancarme en cada paso que doy.

Y por último, a todos aquellos compañeros y amigos que me dió esta carrera a lo largo de los años, quienes, de una u otra forma, son parte de mi desarrollo personal y profesional.

¡A todos ellos, Muchas gracias!

Julián da Costa Faro

Capítulo 1
Introducción

Es de público conocimiento el incremento en el uso de aplicaciones web en las últimas décadas para realizar todo tipo de tareas. Al mismo tiempo es posible observar la diversidad de usuarios de los diferentes sitios que facilitan desde tareas cotidianas hasta interacciones relacionadas con el ocio posibilitando que se alcance a un universo de usuarios cada vez mayor y más diverso. El éxito de una aplicación está determinado en su mayoría por el grado de facilidad que experimenten los usuarios al utilizarla. La usabilidad además se encuentra entre las características más importantes que deben contener las aplicaciones web, como también lo son la seguridad y fiabilidad (Offutt, 2002).

Existen una gran cantidad de métodos para estimar el grado de usabilidad de las aplicaciones y estos pueden ser utilizados en distintas etapas del proceso de desarrollo. En el estudio de Bass y John (2003) se propone resolver los problemas de usabilidad desde la primera etapa del proceso de desarrollo a través de un buen diseño arquitectónico, sin embargo la mayoría de los problemas de usabilidad suelen detectarse recién al final de dicho proceso (Firmenich et al., 2018).

Una de las técnicas más utilizadas para hallar problemas de usabilidad son las pruebas de usabilidad (Krug 2009). Una prueba de usabilidad consiste en observar a un grupo de usuarios mientras realizan tareas típicas en una aplicación con el fin de detectar cualquier problema que dificulte la interacción con el sistema. Para guiar y observar a los usuarios mientras realizan las pruebas, en un principio era necesario que el diseñador del experimento y los voluntarios se encontraran bajo un mismo laboratorio al mismo tiempo, de esta forma el evaluador puede guiar y detectar problemas mientras los voluntarios ejecutan las pruebas. Una de las alternativas a la ejecución presencial de los experimento surgió con la popularidad de la conexión a través de redes de internet, gracias a esto es posible realizar pruebas de usabilidad de forma remota evitando tener que reunir a todos los evaluadores bajo un mismo laboratorio (Hartson et al., 1996) y reduciendo costos obteniendo resultados similares a los que se logran con las pruebas presenciales (Martin et al., 2014).

En distintas fuentes sobre la temática de las pruebas de usuario se destaca que no hace falta un gran número de usuarios o recursos para realizar pruebas que sean consideradas exitosas (Nielsen, 2000) (Krug, 2005), pero está comprobado que cuanto más crezca el número de

usuarios que ejecutan las pruebas de usabilidad mayor va a ser la efectividad a la hora de detectar problemas (Faulkner, 2003, p. 381).

Steve Krug afirma que "*A la gente le gusta pensar que puede utilizar las pruebas de usabilidad para comprobar si un sistema de navegación "a" es mejor que uno "b", pero esto no es posible. Nadie tiene los recursos para establecer el tipo de experimento controlado que se necesita. Dicha prueba puede proporcionarle un aporte de gran valor que, junto a su experiencia, decisión profesional y sentido común, le facilitará la elección adecuada, y con mayor confianza, entre"a" y "b*" " (Krug 2005). Con esto, Krug afirmaba que no es una tarea sencilla demostrar con rigor estadístico que una alternativa sea mejor que otra, y que es necesaria una evaluación minuciosa de los resultados arrojados para acercarnos a una certeza. Krug no está refiriéndose a realizar pruebas comparativas como el A/B testing, pero hoy en día es posible aplicar este procedimiento en las pruebas de usabilidad para facilitar la realización de las mismas y reducir los costos que conllevan.

Cuando se menciona "A/B Testing" se hace referencia a una técnica de creación de pruebas que es principalmente utilizada para evaluar aquellos casos en los que se cuenta con dos o más versiones de una interfaz que buscan solucionar un problema concreto. Para esto se distribuyen las pruebas a los usuarios en tantas partes como versiones disponibles existan. A raíz de los datos que entreguen las ejecuciones de las pruebas sobre las distintas versiones, el equipo experto en usabilidad podrá tomar decisiones definiendo qué versión brinda mejores resultados con respecto a lo que se quería evaluar con las pruebas. Por lo general, las diferentes versiones suelen ser similares, pero presentan diferencias puntuales que son significativas a la hora de evaluar la usabilidad. No existe una limitación con respecto a que las pruebas A/B Testing se realicen de forma presencial o con evaluaciones remotas.

Las pruebas de usuario de forma remota fueron ganando terreno a la hora de evaluar la usabilidad en situaciones en las que simplemente no es posible realizar las evaluaciones de forma presencial o debido a limitaciones presupuestarias o temporales (Moran & Pernice, 2020). Asociado a las pruebas remotas se encuentra el concepto de pruebas de usuario remotas *moderadas* y *no moderadas* (Schade, 2013). En las pruebas de usuario moderadas los voluntarios y los expertos en usabilidad se encuentran en constante comunicación durante las pruebas, de esta manera el experto puede observar la prueba mientras se comunica con el participante mediante algún medio como puede ser una conferencia, chat, mail, teléfono, etc. En las pruebas no moderadas el voluntario realiza el estudio cuando puede y sin contar con la presencia del experto de ninguna manera. En estas pruebas el usuario simplemente debe grabar la sesión de pruebas para que luego sea enviada a los evaluadores y, ocasionalmente, responder algunas preguntas genéricas sobre la prueba.

Tycho[1] es una herramienta que busca ofrecer soluciones a la hora de realizar pruebas de usabilidad moderadas de forma remota sobre aplicaciones web. La herramienta cuenta por un lado con un portal web, en el que el diseñador puede crear distintas secuencias de tareas con sus respectivas indicaciones con el objetivo de guiar al usuario a lo largo de la prueba. Cada tipo de tarea que el experto puede utilizar tiene características particulares que permitirán guiar al usuario, capturar métricas de lo que este vaya realizando durante la ejecución y, si se desea, obtener la opinión subjetiva del voluntario. Con respecto a los tipos de tareas, por un lado se encuentran aquellas que están orientadas a guiar al usuario, como pueden ser alertas de texto o videos explicativos que se presentarán en el orden diagramado. Por otro lado, se encuentran las tareas que permiten capturar información de lo que realice el usuario durante la prueba, estas no interfieren en la interacción del voluntario con la aplicación a probar porque son ejecutadas en segundo plano, de este tipo de tareas el voluntario sólo verá en un panel emergente la indicación que el experto le brindó para realizarla. Por último, se encuentran los tipos de tareas que permiten obtener opiniones de los usuarios con respecto a la prueba mediante distintos cuestionarios personalizables y otros tipos de cuestionarios que cuentan con un formato particular como, por ejemplo, los cuestionarios SUS de los cuales está comprobado que arrojan buenos resultados para evaluar la usabilidad de un sistema (Brooke, 2013). Cuando el experto se comunique con los voluntarios para que este ejecute las pruebas deberá brindarle los pasos a seguir para que instale una extensión para el navegador y decirle cual es el identificador de la prueba a realizar. Con estos elementos el usuario cuando desee realizar la prueba solo debe encender la extensión instalada e ingresar el identificador indicado. A partir de este punto se le presentarán al usuario las tareas diagramadas en el orden que propuso el experto. Al finalizar las ejecuciones el experto podrá ver las respuestas de los cuestionarios y las métricas capturadas en una tabla la cual podrá exportar a csv.

Con respecto a las pruebas de usuarios que se pueden realizar a través de Tycho, es importante aclarar que no es necesario modificar la aplicación que se vaya a testear ni agregar ningún tipo de script. Al tratarse de una extensión del navegador el motor que permite guiar al usuario y capturar métricas se ejecuta “por encima” de cualquier aplicación web.

Si bien Tycho cuenta con características interesantes para la ejecución de pruebas remotas, la herramienta carece de la posibilidad de comparar resultados de las ejecuciones de diferentes pruebas, como si se tratase de A/B Testing. En primer lugar, hasta el momento del desarrollo de la presente tesina no se cuenta con la posibilidad de distribuir de manera balanceada dos pruebas de usuarios asociadas entre sí. En segundo lugar, la herramienta captura pocos datos durante la ejecución de los experimentos, lo que hace que sea difícil la evaluación de información relevante desde el portal web. En tercer lugar, y más allá de que hasta el momento sean capturados pocos

[1] http://tycho.lifia.info.unlp.edu.ar

datos, Tycho carece de un apartado de visualización que le permita al experto tener información clasificada de manera tal que se obtenga una perspectiva general de las ejecuciones realizadas. Al mismo tiempo, un apartado de visualización más sofisticado permitirá comparar los valores de las ejecuciones de dos o más pruebas de usuario en la medida de lo posible.

A través del desarrollo de la presente tesina se intenta agregar el valor mencionado en el párrafo anterior haciendo hincapié en mantener la flexibilización y facilidad de extensión con la que cuenta la herramienta hasta el momento. En el apartado 1.1 se encuentran detallados los objetivos del desarrollo realizado así como también distintas consideraciones de la herramienta extendida.

1.1 Objetivos

La tesina tiene como objetivo primordial extender la herramienta Tycho con la finalidad de realizar pruebas moderadas de usuario de forma remota y tener contar con la opción de aplicar A/B Testing en ellas. Vale aclarar que a lo largo del documento el lector se encontrará con distintas explicaciones de la herramienta Tycho, tanto en la versión heredada (sin soporte de A/B Testing) como en la versión posterior a la realización del presente trabajo. Esto se debe a que no existe una documentación previa de la herramienta que pueda ser referenciada. Por lo tanto, poner a disposición una documentación completa y detallada de Tycho que pueda ser utilizada para referenciar a la herramienta en investigaciones futuras se puede identificar como un objetivo secundario de este trabajo.

Tycho permite crear experimentos con una serie de tareas que deben ejecutar los voluntarios, para luego poder observar y ver los resultados arrojados. Cada ejecución de un experimento en Tycho se transforma en una muestra que contiene los resultados de dicha ejecución. Actualmente la herramienta no cuenta con un método de visualización de resultados sofisticado, sino que solo se muestran los datos de cada ejecución en una tabla. Este es uno de los puntos importantes a la hora de extender la herramienta, porque para realizar A/B testing es necesario que los datos arrojados por las ejecuciones de experimentos sean fácilmente comparables con la finalidad de determinar eficientemente qué alternativa de interfaz de usuario brinda mejores resultados de usabilidad.

Las pruebas del tipo A/B Testing consisten básicamente en tener dos o más ejecuciones que sean similares pero que cuenten con algunas pequeñas diferencias entre sí. Si bien Tycho cuenta con la definición de dos o más secuencias de tareas diferentes dentro de la misma sesión de experimento, no tiene implementada la funcionalidad de copiar o clonar un tipo de ejecución

definido previamente en esa sesión. Esto será de utilidad a la hora de realizar las pequeñas modificaciones mencionadas entre las ejecuciones.

Una vez definidos los tipos de experimentos que se encuentran dentro de una sesión es necesario tener una estrategia de distribución. Cuando se habla de estrategia de distribución se está haciendo referencia a cómo se le debe indicar a los distintos voluntarios que experimento debe ejecutar. Esto se puede hacer de forma manual o se puede automatizar entregándoles a todos los voluntarios el mismo identificador de experimento y distribuyendolo en base a una estrategia. Para que las pruebas del tipo A/B Testing sean fructíferas se requiere un número considerable de voluntarios, por lo que automatizar la distribución es un aporte fundamental, por lo tanto forma parte de los objetivos de la tesina contar con esta funcionalidad en la herramienta.

Por otro lado, la comparación de resultados es vital a la hora de realizar A/B Testing y para esto es necesario tener datos y métricas que sean comparables. Tycho captura algunos datos interesantes como el tiempo transcurrido de ejecución o si las tareas fueron finalizadas o abandonadas. Es necesario adicionar otras capturas de métricas en la ejecución de las tareas que lo permitan para darle un valor agregado a la comparación de ejecuciones mediante un módulo de visualización.

De lo mencionado hasta el momento nacen algunos objetivos específicos:

- Identificar los desafíos que presenta el actual soporte de diseño de protocolos de Tycho para la comparación de estos en el contexto de pruebas en A/B testing. Se incluye el desafío de comparar protocolos y/o tareas equivalentes.

- Implementar la administración de participantes, y la asignación automatizada para las pruebas de usabilidad, acorde a las prácticas usuales de A/B testing.

- Generar un apartado en el que sea posible realizar comparaciones de los resultados arrojados por las ejecuciones de las pruebas. Es necesario capturar nuevas métricas durante las ejecuciones y disponer de distintos métodos de visualización de los datos capturados.

- Presentar casos de prueba donde se demuestre el funcionamiento de la herramienta y el valor agregado de las extensiones realizadas.

1.2 Estructura de la tesina

Este documento se encuentra organizado en 8 capítulos.

En el presente capítulo se brinda una introducción a la tesina y se mencionan los objetivos que fueron definidos en la propuesta, además de la estructura descripta.

En el capítulo dos se explican conceptos que son fundamentales conocer para tener una comprensión completa del desarrollo realizado. Posteriormente se mencionan los distintos trabajos relacionados que fueron investigados.

En el capítulo tres se describe con mayor detalle la propuesta de extensión a la herramienta Tycho con la finalidad de aplicar A/B Testing. Se exponen cuáles fueron las funcionalidades heredadas y donde debió ser modificada la herramienta. Por último, se presentan los resultados esperados.

En el capítulo cuatro es presentada la arquitectura de la herramienta Tycho tanto en la aplicación web como en la extensión utilizada para realizar las pruebas.

En el capítulo cinco se detalla el funcionamiento completo de Tycho a modo de documentación. Se explica cómo se diseñan los experimentos, cuales son los pasos que debe seguir cada voluntario para ejecutarlos y de qué manera deben visualizar los resultados los evaluadores.

En el capítulo seis son presentados dos casos de prueba sobre escenarios diferentes con la participación de usuarios reales.

En el capítulo siete se detallan las conclusiones obtenidas, los alcances y las limitaciones de la tesina.

En el capítulo ocho se mencionan posibles ampliaciones a los realizado en este trabajo.

Capítulo 2
Trabajo Relacionado

El presente capítulo tiene la finalidad de presentar los distintos temas y trabajos que debieron evaluarse para el desarrollo del presente trabajo. El mismo se encuentra dividido en dos secciones.

En la sección 2.1 se presentan conceptos generales que son necesarios comprender para tener una mejor alcance al momento de leer el trabajo.

En la sección 2.2 se describen aquellos proyectos, investigaciones, artículos y propuestas que se encuentran relacionados con las pruebas de usabilidad y distintas formas de aplicarlas que motivaron la realización del presente trabajo.

2.1 Conceptos generales

2.1.1 Usabilidad

La usabilidad es uno de los conceptos principales que se deben comprender previo a la lectura de la presente tesina. A través de la herramienta Tycho es posible realizar distintos tipos de pruebas para evaluar cual es el grado de usabilidad de los sistemas que desean ser valorados para brindar mejoras en este aspecto.

Una de las definiciones de usabilidad que más impacto tiene es la de Nielsen (1993), que fue uno de los principales actores de la década del 80 con respecto a la usabilidad. Nielsen define a la usabilidad como un atributo de calidad que no es unidimensional, sino que está compuesto por varios componentes de los cuales destaca los siguientes:

- ➔ **Aprendizaje**: El sistema debe ser fácil de aprender para que el usuario pueda comenzar a trabajar con él rápidamente.

- ➔ **Eficiencia**: El sistema debe permitir un nivel alto de productividad una vez que el usuario lo conoce, para esto debe cumplir con este atributo.

- ➔ **Recordable**: El sistema debe ser intuitivo y fácil de recordar, de modo que el usuario pueda volver a utilizar el sistema después de un tiempo sin haberlo usado y no deba aprender todo denuevo.

- ➔ **Errores**: El sistema debe tener una baja tasa de errores y si estos ocurren que sea fácil recuperarse de ellos. La aplicación no debe tener errores catastróficos.

- ➔ **Satisfacción**: El sistema debe ser agradable de usar para que el usuario se encuentre subjetivamente satisfecho al utilizarlo.

Otra de las definiciones que es utilizada ampliamente para definir la usabilidad es la de la Organización Internacional de Normalización, en la que la definen como la medida en que un producto puede ser utilizado por usuarios específicos para alcanzar objetivos concretos con eficacia, eficiencia y satisfacción en un contexto de utilización determinado (ISO, 1994).

2.1.2 Pruebas de Usabilidad

Una vez conocido el atributo de usabilidad es necesario poder evaluar cual es el grado de dicho atributo con el que cuentan las diferentes aplicaciones. Para esto existe una metodología popular del diseño de aplicaciones que son las pruebas de usabilidad [2].

En el libro de Jacko y Sears (2003) titulado "*The Human–Computer Interaction Handbook"* se marcan seis puntos principales con los cuales deben cumplir todas las pruebas de usabilidad:

1. Están centradas en la usabilidad.
2. Los participantes son usuarios finales o usuarios finales potenciales.
3. Hay un sistema a evaluar.
4. Los participantes realizan tareas.
5. Se registran datos capturados de las tareas realizadas por los participantes y se analizan.
6. Los resultados de las pruebas se comunican a quienes corresponda.

Desde otro ángulo, Nielsen (2012) explica que una sesión de pruebas de usabilidad, consiste en que un investigador le solicite a un participante que realice determinadas tareas, generalmente utilizando una o más interfaces de usuario específicas. Mientras el participante completa cada tarea el investigador observa el comportamiento del mismo y presta atención a sus comentarios sobre la aplicación.

[2] También son conocidas como *"Pruebas de usuario"* por la traducción de "*User test*" al español.

Si bien el objetivo de las pruebas de usuario varían con respecto al estudio que se está realizando, en su mayoría son de utilidad para identificar problemas de diseño, descubrir oportunidades para mejorar la aplicación y aprender del comportamiento y/o preferencias del usuario al cual se encuentra dirigida la aplicación.

Si bien hay diferentes tipos de pruebas de usuario y con diferentes objetivos, Nielsen (2012) menciona tres elementos claves que debe tener la mayoría de las pruebas de usuario:

1. **Investigador**: el encargado de guiar al participante durante el proceso de la prueba y diseñar el experimento. Además es quien analiza los resultados que arrojen las pruebas ejecutadas.

2. **Tareas**: Actividades diseñadas por el investigador y ejecutadas por el participante. Estas tareas pueden ser muy específicas o muy abiertas dependiendo de lo que el diseñador desee analizar.

3. **Participante**: Usuario final que ejecutará las tareas diagramadas por el investigador. En base al modo en el que cada participante ejecute las tareas se recopila información que puede ser de utilidad para los diseñadores de la aplicación.

En sus orígenes, las pruebas de usuario eran solo presenciales, es decir reuniendo a los evaluadores y los voluntarios en un mismo laboratorio. Con el paso del tiempo se abrió la posibilidad de realizar pruebas de forma remota, reduciendo el costo que estas conllevan y siendo tan eficaces como las pruebas presenciales (Thompson et al., 2004). Cuando hablamos de pruebas de forma remota también pueden clasificarse en dos grandes grupos:

- ➔ **Pruebas remotas moderadas**: Son pruebas que se realizan de la misma manera que si estuvieran presencialmente pero a distancia. Ya sea a través de una conferencia, correo, o incluso una herramienta desarrollada particularmente para esto, el investigador interactúa con el voluntario de forma constante tanto para brindar las tareas como para capturar y evaluar lo que el usuario realiza.

- ➔ **Pruebas remotas no moderadas**: Las pruebas que no son moderadas son aquellas en la que los diseñadores utilizan alguna herramienta para llevarlas a cabo y el participante ejecuta las pruebas por sus propios medios y no tiene comunicación alguna con el invesetigador.

Tycho implementa una variante híbrida que combina características de pruebas remotas moderadas y de no moderadas. El voluntario no tiene un contacto directo con el evaluador, pero cuando se utiliza Tycho se cuenta con un seguimiento en cada tarea, en la que el evaluador puede describir que se espera que haga o realizar las preguntas deseadas.

2.1.3 A/B Testing

La técnica de "A/B Testing", también conocida como *pruebas de división*, se utiliza para realizar experimentos en los que se cuenta con dos versiones ("A" y "B") de una misma solución a un problema determinado, y en base a distintas pruebas se evalúan los resultados para decidir cuál es la mejor. Ambas versiones son similares y se aplican generalmente para una misma tarea pero su variación puede cambiar significativamente el comportamiento del usuario.

A veces, la versión "A" y la "B" son diseños diferentes que compiten directamente entre sí, otras veces la versión "A" es el diseño con el que se contaba de antemano y la versión "B" puede ser una versión alternativa o experimental para determinadas soluciones.

Según la definición del libro "*A/B Testing: The Most Powerful Way to Turn Clicks into Customers"* (Siroker 2012) "Las pruebas A/B son la simple idea de mostrar varias versiones diferentes de una misma página web a los usuarios que se conectan a ella. Usando este tipo de pruebas las empresas pueden mejorar la efectividad de su marketing y la experiencia del usuario".

Las pruebas A/B son utilizadas principalmente por diferentes empresas de renombre como lo son Microsoft, Linkedin, Google con el objetivo de atraer el flujo de navegación hacia los sitios que son de su interés. Estas empresas realizan miles de experimentos cada año, contando frecuentemente con la participación de millones de usuarios, probando distintos escenarios como cambios en la interfaz de usuario, algoritmos de búsqueda, latencia o rendimiento de distintas funcionalidades, sistemas de gestión, etc. Los experimentos también son ejecutados en diferentes canales como sitios web, aplicaciones de escritorio, aplicaciones móviles o correos electrónicos (Kohavi et al., 2020).

Con el paso del tiempo se efectuaron diferentes estudios en los que se focalizó la utilización de A/B Testing para tratar problemáticas exclusivamente de usabilidad de la web. Tal es así que se elaboró un proceso teórico que permitiría mejorar la usabilidad a través del A/B Testing y la refactorización (Firmenich et al., 2018). En dicha investigación se propone un proceso iterativo que es compatible con las metodologías ágiles para mejorar la usabilidad a través de pruebas de usuario y su correspondiente *feedback*. Este proceso permite evaluar soluciones alternativas

cuando los desarrolladores se encuentran con problemas de usabilidad, y para evaluar las distintas soluciones se propone utilizar A/B Testing.

En esta tesina se agrega en Tycho la posibilidad de ejecutar los experimentos en formato de pruebas A/B Testing, cambiando las tareas que se van a ejecutar o parte de las indicaciones que estas tienen. Vale aclarar que en el presente trabajo se habla de A/B Testing como un término formal pero las versiones pueden ser multivaluadas, es decir que pueden existir más de dos versiones del experimento.

2.2 Trabajos Relacionados

Uno de los temas importantes que busca tratar en su tesis Luzik (2014) es cómo se pueden combinar las pruebas de usabilidad y las pruebas A/B testing para utilizar lo mejor de ambos tipos en los escenarios correspondientes. Luzik menciona que con las pruebas de usabilidad los expertos pueden probar un área más amplia de un sistema o producto, en cambio las pruebas A/B Testing se focalizan en una parte específica del sistema que se está probando, por lo tanto es posible concluir que cuando se identifican los problemas y se propone una solución las pruebas A/B testing pueden ayudar a reafirmar la propuesta con escenarios reales y con mayor rigor estadístico. Tanto las pruebas puramente de usabilidad como las pruebas A/B testing dan respuestas a diferentes tipos de preguntas y arrojan resultados distintos, utilizar la combinación de ambos tipos de prueba puede ser una excelente alternativa para identificar todo tipo de problemas. Cuando existen dos o más diseños con enfonces diferentes, las pruebas A/B pueden ser una herramienta muy útil. Por otro lado, las pruebas de usabilidad van a ser una mejor herramienta para entender el pensamiento de los usuarios a la hora de interactuar con el sistema. Las pruebas de usabilidad responden a la pregunta de "*¿Por qué?"* arrojando datos cualitativos, mientras que las pruebas A/B testing responden a la preguntas "*¿Cual?*" "*¿Cuantos?*" entregando como respuesta datos cuantitativos.

Existen diversos modelos que buscan incorporar las pruebas de usabilidad en las distintas etapas de los procesos de desarrollo y no solo en el último paso como se realiza en la mayoría de los casos (Fernández et. al., 2011) (Firmenich et al., 2018) (Speicher et al., 2014).

En el artículo "*A Web Usability Evaluation Process for Model-Driven Web Development*" Fernández et al. (2011) propone un proceso de evaluación de usabilidad basado en el proceso de evaluación de calidad que es propuesto en la norma ISO 25000[3], para evaluar los sistemas web que se obtienen como resultado de los procesos de desarrollo dirigido por modelos. En este trabajo se proponen cinco etapas a la hora de definir las pruebas de usabilidad:

[3] Normas ISO 25000: https://www.iso.org/standard/64764.html

1. Establecer los requisitos de evaluación para delimitar el alcance de la prueba.
2. Especificación de la evaluación para determinar qué métricas se pretenden aplicar y cómo los valores obtenidos por estas métricas permiten detectar problemas de usabilidad.
3. Diseñar cómo se realizará la evaluación y qué información se recogerá durante la misma.
4. Ejecutar la evaluación de acuerdo con el plan diagramado.
5. Evaluar los resultados obtenidos.

Se puede intuir sobre la necesidad de una herramienta que facilite el diseño, la captura de métricas que se decide tomar, la ejecución de las pruebas y la posterior evaluación de resultados en el proceso propuesto por Fernández et al. (2011).

Por otro lado, la investigación realizada por Firmenich et al. (2018) "*Usability improvement through A/B testing and refactoring"* está orientada a atacar el problema de la utilización de las pruebas de usuario en desarrollos ágiles permitiendo la utilización de pruebas A/B testing. Se menciona la dificultad en el acceso a este tipo de pruebas por parte de las pequeñas y medianas empresas a raíz del costo que conlleva aplicarlas, tendiendo a que confíen en la primera solución propuesta. Se propone atacar dicha problemática utilizando un proceso de mejora continua de la usabilidad basada en el feedback del usuario y que además sea compatible con procesos de desarrollo ágiles.
En este artículo se propone utilizar un proceso iterativo que es compatible con las metodologías ágiles para mejorar la usabilidad a través de pruebas de usuario y su correspondiente feedback. A partir de diferentes herramientas este proceso permite diseñar pruebas de usuarios, ejecutarlas de forma remota y evaluar soluciones alternativas cuando se encuentran problemas de usabilidad como lo hacen las pruebas A/B Testing.
El método de pruebas y mejoras de la usabilidad con pruebas A/B propuesto por Firmenich et al. (2018) consiste en cinco etapas bien definidas:

- ➢ **Etapa Nº 1**: El experto diseña las pruebas de usuario teniendo en cuenta las tareas que se deben ejecutar y las métricas que se deben capturar y calcular durante la ejecución de las pruebas
- ➢ **Etapa Nº 2**: Se realiza un análisis de los resultados de las pruebas en busca de detectar problemas de usabilidad. Los resultados deben ser presentados en gráficos claros que permitan al diseñador identificar un problema de usabilidad y diseñar posibles soluciones. Es posible aplicar diferentes soluciones a partir de una herramienta del estilo *Client-Side Web Refactoring (CSWR)*[4]. Las posibles soluciones pueden diseñarse de manera tal que

[4] Un CSWR es una herramienta que permite realizar modificaciones a las aplicaciones web desde el lado del cliente y en tiempo de ejecución.

sean experimentos similares con pequeñas diferencias y que posteriormente se puedan comparar para decidir qué solución arrojó los resultados más apropiados.

- ➢ **Etapa Nº 3**: Los voluntarios realizan las pruebas de las distintas versiones de las tareas, dividiendo las pruebas en tantos grupos como versiones de la aplicación se deseen evaluar (cómo se hace en las pruebas A/B Testing).
- ➢ **Etapa Nº 4**: El experto en UX compara los resultados de las pruebas ejecutadas para cada versión con los resultados de la primera etapa, de esta manera se determina la mejor solución.
- ➢ **Etapa Nº 5**: Una vez detectada la mejor solución se indica a los desarrolladores que se implemente dicha solución de manera efectiva.

Otro de los trabajos que proponen pruebas A/B Testing basadas en la usabilidad es "*Ensuring Web Interface Quality through Usability-Based Split Testing*" (Speicher et al., 2014). En el paper se propone un método que asegure la calidad de la interfaz de las aplicaciones web basadas en mediciones cuantitativas e interacciones con los usuarios. El método consta de dos pasos generales:

1. Capturar el comportamiento del usuario en la aplicación al momento de ejecutar las pruebas A/B testing y aplicar métricas de interacción resultantes a las reglas heurísticas para la evaluación de la usabilidad. Por último comprobar si la diferencia entre ambas interfaces es significativa.
2. Si el resultado no es significativo es necesaria información más específica que se puede obtener a través de un cuestionario de usabilidad asociado a las interfaces presentadas.

Disponer de una herramienta para distribuir, ejecutar y comparar resultados de distintas pruebas de usuario, hace que el proceso de evaluación sea más efectivo y eficiente. Tycho, en conjunto con la extensión realizada para que sea posible aplicar A/B Testing, podría ser utilizada en las etapas de proceso de desarrollo propuesto en el diseño de los experimentos, la ejecución de experimentos simples, la generación de experimentos de tipo A/B Testing, la distribución de este tipo de experimentos y la comparación de resultados.

En cuanto a la arquitectura utilizada para realizar pruebas remotas existen distintas alternativas que fueron estudiadas e implementadas a lo largo del tiempo. Una de esas alternativas es realizar el registro de actividades del usuario mediante un proxy como presenta en su estudio Hong (et al., 2001), donde propone un sistema para llevar a cabo el registro y las visualizaciones de lo que se realiza en la web con el objetivo de ayudar al equipo de diseño a ejecutar pruebas de usabilidad (tanto locales como remotas) para posteriormente analizar los datos recogidos. Para esto propone una herramienta llamada *WebQuilt* realizando la captura de datos a través del proxy mencionado anteriormente. Hong (et al., 2001) describe que el análisis de registros de servidor es

útil para comprender cuantitativamente lo que hace un gran número de personas en un sitio web. Utilizar estos registros es simple, ya que la mayoría de los servidores web registran esta información automáticamente, pero desde el punto de vista del equipo de diseño este tipo de registros no permite interpretar las acciones de un usuario individual de una manera sencilla. En contrapartida con la captura de datos en el servidor existe la captura de datos en el cliente que, para la época en la que Hong hizo su propuesta (Hong et al., 2001), sólo era posible a través de programas que registren las acciones del usuario sobre el navegador web, o bien creando un navegador personalizado específicamente para capturar información sobre el uso. Sin embargo, para que esta solución sea posible se debe instalar un software especial en el cliente, lo que podría limitar a los participantes en las pruebas de usabilidad a usuarios que son experimentados, y por consiguiente pueden no ser representativos del público al cual está dirigida la prueba. A raíz de las desventajas por los métodos de captura mencionados, Hong et al. (2001) pensó en una solución intermedia que denomina *WebQuilt*, a partir de esta herramienta se va a poder capturar, analizar y visualizar el uso que le dan los usuarios finales a la aplicación. Para esto propone la utilización de un proxy que intercepta todas las acciones que el cliente realiza con el servidor y agrega valor persistiendo datos que sean de utilidad para la toma de decisiones de diseño. La utilización del proxy no requiere que los usuarios finales realicen ninguna descarga.

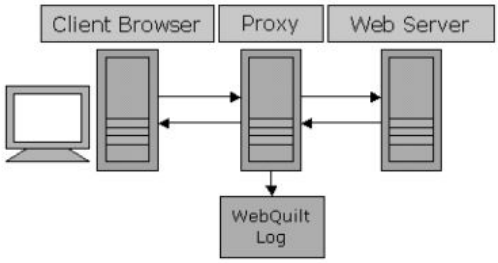


Figura 2.1. Solución propuesta por Hong et al. (2001) utilizando un proxy

Una alternativa similar a la de Hong es planteada por Atterer et al. (2006), en la que se propone utilizar un *proxy* que permita interceptar y modificar las respuestas del servidor al cliente incluyendo distintos scripts que serán de utilidad a la hora de realizar un seguimiento del usuario en la aplicación. En el trabajo de Atterer se estudian las distintas alternativas y métodos para obtener más información sobre cómo interactúan los usuarios con las aplicaciones web, incluyendo un seguimiento en detalle de toda la interacción del usuario, como por ejemplo el desplazamiento realizado con el mouse o la utilización del scroll en las páginas. Para llevar a cabo el seguimiento se agrega comportamiento a los widgets y elementos html para que cuando el usuario interactúe con ellos se envíe la información al proxy de la acción realizada y la ubicación de la misma, generando de esta manera un log con más información.

Si bien una tecnología como el proxy planteado tanto por Hong et al. (2001) como por Atterer et al. (2006) podría servir para la captura de métricas en pruebas remotas, en principio no serían de utilidad para guiar al usuario a lo largo de la ejecución de una prueba ya que ambas alternativas están orientadas a habilitar la captura de datos por un periodo de tiempo y procesar la información de las acciones realizadas.

Es evidente que las desventajas planteadas para la captura de datos desde el lado del cliente se fueron disminuyendo con el paso de los años y los avances tecnológicos. La utilización del proxy puede ser simple pero al mismo tiempo puede fallar en aquellas aplicaciones que son renderizadas solo del lado del cliente con javascript. Otra de las ventajas planteadas por la utilización de un proxy es la portabilidad de la captura de datos para distintos navegadores. Para esas épocas si se quería capturar información del lado del cliente existían alternativas que limitaban el navegador y sistema operativo a utilizar para realizar las pruebas de usuario (Goecks & Shavlik, 2000) (Reeder et. al., 2001, p19). Hoy es posible realizar una extensión del navegador que es ampliamente utilizada por los usuarios finales para la realización de las pruebas de usuario y la captura de métricas.

Una herramienta diferente para la captura de datos y la creación de pruebas remotas A/B testing orientadas a la usabilidad es WaPPU (Speicher et al., 2014). La herramienta se ocupa de todo el proceso de creación de pruebas de tipo A/B testing como un servicio centralizado, en el cual las pruebas divididas se pueden crear desde un tablero de control para el experto modificando las interfaces que se quieran probar, generando una versión alternativa. Además para el seguimiento del usuario se requiere la instalación de un plugin jquery en el cliente.

Con respecto a las métricas, hay evidencia científica de que los datos que son generados por los usuarios en la interacción con las diferentes aplicaciones pueden ser de utilidad a la hora de analizar las mejoras que se pueden realizar con respecto a la usabilidad. Uno de los trabajos que fue de mayor utilidad para el desarrollo de la tesina fue "*TellMyRelevance! Predicting the Relevance of Web Search Results from Cursor Interactions*" (Speicher et al. 2013).

El trabajo realizado por Speicher et al. (2013) está enfocado a aquellas aplicaciones web basadas en la búsqueda. Para este tipo de aplicaciones es importante saber cuando los usuarios encuentran la información que buscaban, e indica que los modelos basados solamente en clicks no son tan relevantes, ya que estos son imprecisos y dejan información valiosa de lado de las interacciones de los usuarios sin realizar clicks. La imprecisión en los modelos basados en clicks se puede dar en diferentes aspectos como, por ejemplo, cuando un usuario realiza un click en un resultado que no era lo que el usuario buscaba. Se plantea como alternativa hacer un seguimiento del mouse más allá de los clicks, ya que los movimientos del mouse son una aproximación razonablemente buena con la mirada del usuario al utilizar el sitio. Para realizar el seguimiento

Speicher et al. (2014) propone utilizar un plugin jquery que no sea muy invasivo y que tome distintas métricas de lo que ocurre cuando un usuario realiza una búsqueda de la misma manera que lo hizo en su posterior trabajo. Rastrea las interacciones como por ejemplo cuando ingresa a determinado elemento que sea de interés y cuando sale, cuando ingresa a un link, la inactividad del usuario cuando el cursor permanece quieto durante un determinado tiempo, entre otras métricas que se detallarán a continuación.

Adentrándose en la información más relevante para la tesina podemos encontrar un listado detallado de cuáles son los eventos que se capturan y que brindan información relevante de lo que el usuario realizó:

- ➔ El tiempo en el que el mouse estuvo por encima del resultado.
- ➔ Tiempo de llegada, el tiempo que transcurrió entre que el usuario realizó la búsqueda hasta que llegó al resultado por primera vez
- ➔ La cantidad de clicks que el usuario realizó sobre los links que llevan a las páginas que arrojó la búsqueda
- ➔ El número de veces que el mouse pasó por encima del resultado
- ➔ El número de veces que el mouse pasó por encima del resultado sin realizar ningún click
- ➔ El tiempo máximo que de permanencia sobre el resultado
- ➔ La cantidad de píxeles que el cursor recorre hasta llegar al resultado
- ➔ Velocidad del cursor: La trayectoria del cursor dividida el tiempo de movimiento del mismo

Además de la captura de estos eventos, en esta tesis se consideran dos eventos adicionales

- ➔ Cantidad de clics realizados
- ➔ Tiempo en el que el cursor se encontró en movimiento

Estos datos son procesados por un módulo que indica a través de la realización de diferentes cálculos un valor que permitirá brindar información al experto con respecto a la efectividad del buscador del sitio.

Por último existen diferentes mecanismos para la distribución de las pruebas a un número considerable de voluntarios. En el paper *CrowdStudy: General Toolkit for Crowdsourced Evaluation of Web Interfaces* de Nebeling, et al. (2013) se pone en conocimiento de la comunidad científica de un conjunto de herramientas genéricas que combina el apoyo a las pruebas automatizadas de usabilidad utilizando crowdsourcing[5] para que sea posible facilitar la

[5] Crowdsourcing: Método de trabajo para externalizar o delegar tareas a una gran cantidad de personas buscando reducir los tiempos de realización de las mismas y la carga de trabajo del personal de una institución.

generación de pruebas a gran escala, a este conjunto de herramientas se lo denominó en este paper CrowdStudy.

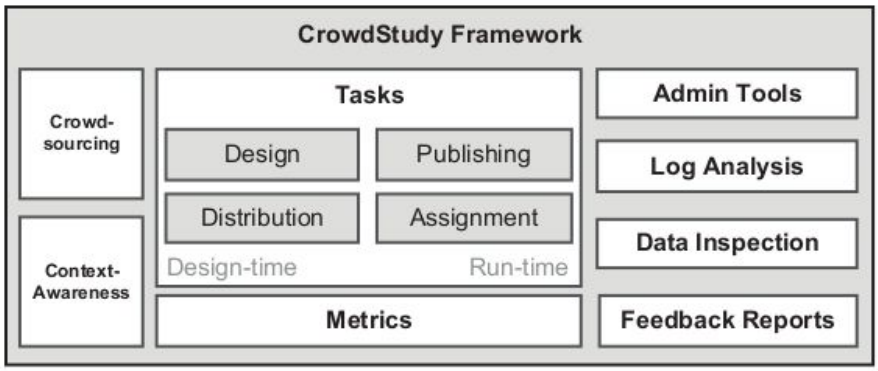


Figura 2.2. Componentes del framework CrowdStudy (Nebeling, et al., 2013)

CrowStudy está orientado específicamente para estudios de usabilidad, permitiendo a los interesados en realizar experimentos tener control del muestreo de usuarios y realizar evaluaciones para diferentes contextos particulares. Se puede decir que esta herramienta proporciona apoyo para la recopilación y el análisis de datos contextuales basados en un conjunto de métricas que son extensibles, así como herramientas para gestionar, revisar y analizar cualquiera de los datos que fueron recopilados durante la ejecución de las pruebas.

La herramienta es presentada como un framework que cuenta con distintos componentes que pueden visualizarse en la figura 2.2. Entre estos componentes se destacan:

Diseño de tareas, distribución y asignación: Permite definir las tareas que formarán parte del experimento. Estas tareas pueden incluir más de un componente web y pueden diseñarse con distinto nivel detalle según lo requiera el diseñador. Además cuenta también con un método de distribución automática de la tareas con servicios como por ejemplo con *Amazon Web Turk*[6]. Por último, los evaluadores pueden decidir si la distribución de las tareas se realiza de forma aleatoria o la asignación de tareas es de manera controlada realizando un balanceo automático en tiempo de ejecución basado en las tareas que se hayan ejecutado hasta el momento.

Modelo abstracto para diferentes experimentos: Si bien la herramienta está orientada a la realización de experimentos diferentes, a veces es necesario contar con ciertos componentes básicos y genéricos para realizar pruebas tradicionales. Crowstudy cuenta con la posibilidad para que los expertos generen casos de estudios genéricos.

[6] Amazon Web Turk: https://www.mturk.com/

Conjunto de métricas extensibles: El framework permite la captura de distintas métricas durante la ejecución de los experimentos:

- ❏ Métricas relacionadas a los dispositivos ejecutados, como sistema operativo, navegador, resolución de la pantalla, etc.
- ❏ Métricas de los usuarios, en la cual el voluntario responde información sobre él mismo como el nivel de estudios, edad, género, conocimientos o preferencias que permiten capturar información demográfica del estudio realizado.
- ❏ Métricas de tiempo, tiempo que llevo la ejecución de cada tarea como así también el tiempo total que se demoró en ejecutar el experimento.
- ❏ Métricas basadas en contadores, como puede ser la relación de tareas ejecutadas satisfactoriamente.
- ❏ Métricas de la actividad del usuario
- ❏ Comentarios y sensaciones del usuario

El objetivo de esta herramienta es integrar el apoyo al crowdsourcing en las herramientas de pruebas de usabilidad para apoyar una amplia gama de métodos y escenarios de evaluación.

Administración y herramientas para el análisis de datos: Por último el framework cuenta con herramientas de administración que permiten gestionar los experimentos, sus tareas y la forma en la que se van a distribuir. Además cuenta con un apartado de visualización de resultados de las ejecuciones realizadas para que los expertos puedan ver tanto los datos más genéricos como el detalle de las ejecuciones realizadas.

Se puede observar como existen ciertas similitudes en la arquitectura de Crowstudy y Tycho. Si bien Tycho no cuenta con la integración con un servicio de crowdsourcing, ambas herramientas permiten que el diseñador decida el método de distribución de los experimentos, ya sea aleatoria o de manera controlada. También es posible observar que ambas herramientas permiten la creación de experimentos genéricos que puedan ser reutilizados. La captura de métricas y su posibilidad de extensión es otra de las características compartidas como también permitir la visualización de resultados de manera tal que el experto tenga una visión global de lo que sucedió en las ejecuciones. Pese a que los dos proyectos están focalizados en solucionar problemas diferentes, en ambos existe una estructura común de las herramientas implementadas.

Capítulo 3
Propuesta: A/B Testing Moderado

En este capítulo se realiza la explicación de la propuesta de extensión a la herramienta Tycho para que sea posible aplicar A/B Testing. En primer lugar se realiza una introducción a la herramienta Tycho. Posteriormente se describen las funcionalidades provistas por la versión original (a extender) de la herramienta en cuanto al modelo, los distintos conceptos utilizados y sus componentes. Por último, se presentan cuales son los resultados esperados de esta propuesta de extensión.

3.1 Introducción a Tycho

Tycho es una herramienta que permite automatizar pruebas de usuario guiadas en aplicaciones web. A través de las pruebas el diseñador de los experimentos puede observar cómo es que los usuarios interactúan con las aplicaciones que están siendo probadas, capturando distintos tipos de métricas a lo largo de la ejecución.

Años atrás era necesario contar con un espacio físico para realizar pruebas de este estilo, lo que las hacía muy costosas y difíciles de realizar por la dificultad de atraer una cantidad considerada de voluntarios y guiarlos en la ejecución de las pruebas. Con el paso del tiempo y el avance tecnológico se fueron empleando nuevas técnicas para la realización de pruebas de usuario, una de ellas son las pruebas de usuario guiadas de forma remota. Tycho cuenta con esta característica, la herramienta está diseñada para realizar las ejecuciones de experimentos de manera remota, y para esto el voluntario simplemente debe descargar un addOn para el navegador e ingresar el identificador del experimento a realizar que le envió el diseñador de las pruebas.

El diseño de las pruebas de usuario consiste en que el diseñador diagrame una secuencia de tareas que el voluntario debe ejecutar con el objetivo de guiarlo a lo largo de la ejecución del experimento y obtener distintas métricas que permitan tomar algún tipo de decisión con respecto a la funcionalidad que se está testeando. En este aspecto se encuentra otra de las particularidades de Tycho, es decisión del diseñador del experimento el nivel de detalle que tendrá cada prueba de usuario. A la hora de diagramar los experimentos el diseñador puede decidir cuánta información adicional le brinda al voluntario al momento de ejecutar las pruebas. De esta manera

puede guiar al usuario a través de múltiples tareas para asegurarse de que se prueben determinadas funciones de la aplicación, o bien puede dar indicaciones mínimas y dejar que el voluntario ejecute la prueba con mayor libertad.

Otro de los puntos a destacar es que con esta herramienta se permite realizar pruebas sobre cualquier aplicación web sin la necesidad de modificar el código del sistema que se quiere probar. Esto es posible gracias a que Tycho trabaja con una extensión del navegador que se ejecuta por encima de la aplicación web.

Con las cualidades mencionadas en los párrafos anteriores la amplitud de las pruebas de usuario que se vayan a realizar quedan en manos del diseñador de los experimentos, permitiendo ejecutar pruebas sobre aplicaciones propias o externas y guiando al usuario con distintos niveles de detalle.

3.1.1 Conceptos de Tycho

En Tycho existen algunos conceptos que necesitan ser presentados para tener una comprensión más abarcativa del trabajo afrontado en la presente tesina.

Tareas

El diseño de las pruebas de usuario consiste en que el experto diagrame una secuencia de tareas que el usuario debe ejecutar. Existen distintos tipos de tareas cuyo objetivo es interactuar de formas diferentes cuando se ejecutan ante el voluntario. Algunos de las tareas que se encuentran a disposición son las siguientes

- *Basic task instructions*: Cuando el voluntario debe ejecutar una tarea de este tipo visualizará en su navegador un cartel que muestra un texto que fue escrito por el diseñador del experimento a la hora de generar la tarea, la idea es que ese texto contenga información relevante para realizar la prueba. Una vez que el voluntario lo lea tendrá la opción de apretar un botón para iniciarla. Cuando el voluntario aprieta ese botón deberá comenzar a realizar la prueba que el diseñador le indicó. Para el voluntario existe la posibilidad de pausar la ejecución, finalizar la ejecución o abandonarla.

- *Message screen*: Esta tarea simplemente bloquea la pantalla y permite visualizar un cartel. Es el tipo de tarea ideal para los casos en los que se desea brindar indicaciones o alertas. La diferencia con el tipo de tarea *"Basic task Instructions"* es que no es una tarea ejecutable, simplemente muestra el mensaje en pantalla, una vez que el voluntario indica que ya fue leído se pasa a la siguiente tarea, si es que existe.

- *Simple questionaire*: Es un tipo de tarea cuya función es presentarle una o más preguntas al voluntario para que él mismo las responda. Si bien la forma de que el voluntario las responda es un campo de texto libre, el objetivo es que las respuestas sean sintéticas para que sea posible realizar un posterior análisis de las mismas.

- *SUS questionaire*: Con esta tarea lo que se busca es que el voluntario responda un cuestionario SUS. Los cuestionario de tipo SUS fueron desarrollados en el año 1986 en el área de la ingeniería de la usabilidad de los sistemas (Brooke, 2013). El objetivo de este tipo de cuestionarios es proporcionar una serie de preguntas que sean simples de completar (rango de 1 a 5 donde 1 es "*muy en desacuerdo*" y 5 es "*muy de acuerdo*") y que en base a las respuestas obtenidas se pueda realizar un cálculo que permita identificar el grado de satisfacción de los usuarios. El resultado de cada pregunta por sí solo no tiene un valor significativo, pero realizando el cálculo correspondiente se logra obtener un valor el cual puede ser comparado con diferentes escalas de resultados. Por otro lado es importante tener en cuenta que la disposición de las preguntas está diagramada para evitar que el usuario responda sin leer, y si lo hace que no arroje un resultado que no sea interesante.
  Por lo tanto este tipo de cuestionarios permite calcular una escala de usabilidad de un sistema en base a las respuestas del usuario con respecto a preguntas específicas.[7]

- *Youtube Video*: Con esta tarea se le presentará al usuario algún video de YouTube con la finalidad de que sea visualizado en algún punto de la ejecución del experimento.

Protocolos

El diseñador del experimento se encarga de organizar una secuencia de tareas que el voluntario debe ejecutar, esta secuencia de tareas se realizan dentro de lo que en Tycho se llaman protocolos. Cada protocolo tiene un nombre, una descripción, un identificador (que será entregado a los voluntarios cuando se desee que sea ejecutado) y las tareas con el orden en el que deben ser ejecutadas. Las tareas dentro de cada protocolo pueden ser modificadas dependiendo de cada una, por ejemplo en las tareas del tipo *"Basic task instructions"* es posible modificar el mensaje que se le va a mostrar al voluntario, lo mismo sucede con las tareas del tipo *"Message Screen"*. En cambio, una tarea del tipo *"SUS questionnaire"* no es posible modificarla ya que este tipo de cuestionarios cuentan con una estructura específica para que posteriormente las respuestas sean analizables.

[7] Puntuación de cuestionarios SUS:
https://www.usability.gov/how-to-and-tools/methods/system-usability-scale.html

En cuanto a los protocolos también es posible definir un protocolo genérico si es que el diseñador tiene en mente crear varios experimentos con protocolos similares. Para esto basta con crear una vez el protocolo que se utilizará como base y luego cuando vaya a ser utilizado modificar lo que sea necesario, pero se ahorra el tiempo de definir la secuencia de tareas una vez más.

Sesiones de experimentos

Por último se encuentra el concepto de sesiones de experimento, cada sesión tiene dentro uno o más protocolos con la finalidad de englobar todos los que sean parte del mismo experimento. Las sesiones ayudarán al diseñador a organizarse.

3.1.2 Modelo

En la siguiente figura (Figura 3.1) se puede observar un diagrama de las principales clases del proyecto Tycho, entre lo que se encuentran los conceptos mencionados en esta sección. En cada clase se puede observar los métodos más importantes.

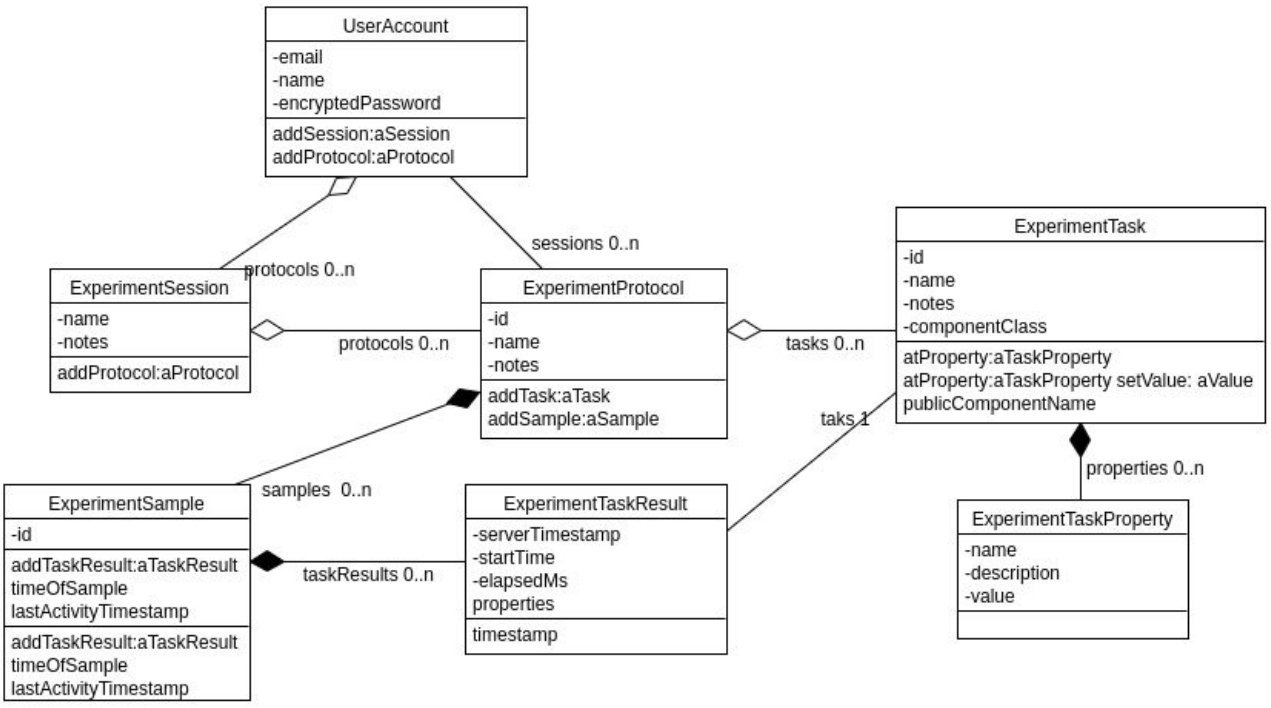


Figura 3.1. Modelo de clases abreviado de la aplicación Tycho

A continuación se detalla que representa cada una de las clases del modelo:

UserAccount

Cuando un usuario se registra en la aplicación se crea una instancia de UserAccount. Además de los datos datos de acceso del usuario esta clase cuenta con dos relaciones:

- *ExperimentSession*: En esta relación se representan las distintas sesiones de experimento que el usuario va a ir generando en Tycho.

- *ExperimentProtocol*: El usuario puede crear distintos protocolos que luego usará en las sesiones de experimentos. La relación de *UserAccount* con *ExperimentProtocol* representa esos protocolos que el usuario genera sin que estén asociados a una sesión.

ExperimentSession

Cada *ExperimentSession* representa una sesion de experimento. Cada uno de las sesiones tienen un nombre y notas que son una pequeña descripción. A su vez cada sesión puede tener uno o más protocolos asociados.

ExperimentProtocol

Cada instancia de esta clase representa un protocolo. Los protocolos tienen un identificador (*id*) que es brindado a los voluntarios para que realicen la ejecución del mismo. Además, y de la misma manera que las sesiones, los protocolos pueden tener un nombre y notas.
En cuanto a relaciones, cada protocolo tiene una o más instancias de *ExperimentTask* que representan a las tareas y también tiene una relación con *ExperimentSample*.

ExperimentSample

Los protocolos pueden ser ejecutados por distintos voluntarios. En Tycho las pruebas son anónimas, es decir no se solicita información del voluntario a la hora de realizar la ejecución de un experimento. Sin embargo es necesario tener identificada cada muestra de los protocolos.
Las instancias de *ExperimentSample* representan a una muestra del protocolo ejecutado y sirven para identificarlas a la hora de ver los resultados en detalle.

ExperimentTask

Las instancias de esta clase representan una tarea dentro de cada protocolo. Cada una tiene un identificador, nombres y notas al igual que los protocolos. Además se caracterizan por tener un atributo denominado *componentClassName*, en este atributo se define el tipo de tarea que se está

creando, el cual es de utilidad para identificar cuales es la particularidad de esa tarea. En conjunto con el *componentClassName* las *ExperimentTask* tienen una relación de una a muchas con la clase *ExperimentTaskProperties*.

ExperimentTaskProperties

A través de esta clase es posible darle distintas propiedades a las tareas. Cada tarea tiene asociada una o varias *ExperimentTaskProperties*, y es en base a estas propiedades se detallan los elementos de la tarea. Las propiedades son del tipo "*clave: valor*", es por eso que entre sus atributos tienen un nombre, un valor y una descripción.
Gracias a estas propiedades es posible realizar nuevos tipos de tareas sin necesidad de modificar el modelo de datos.

ExperimentTaskResults

Esta clase simboliza los resultados de las ejecuciones. Solo es posible llegar a ellas a través de los *ExperimentSample*. Todos los resultados están asociados a una tarea. Entre sus atributos se encuentran *serverTimestamp*, que representa el horario de inicio de la ejecución de esa tarea por parte del voluntario, *startTime* el tiempo en el que comenzó desde que se ejecuta el experimento y el *elapsedMs*, que es el tiempo total que llevo la ejecución en milisegundos. Otro atributo importante es llamado *properties*, es un diccionario en el que se permite guardar información extra relacionada a la ejecución de la tarea en particular, la inclusión de esta propiedad da la libertad al desarrollador de la extensión de enviar distintos tipos de datos dependiendo el tipo de tarea sin modificar el modelo de Tycho.

3.1.3 Componentes

Para lograr una mayor comprensión del diagrama de clases se realiza una explicación de cómo se relacionan las instancias entre sí.

Cuando un diseñador de experimentos realiza el registro de un nuevo usuario en la aplicación web de Tycho, se crea una instancia de *UserAccount.* Cada *UserAccount* puede contener sesiones de experimentos (*ExperimentSession*) y protocolos genéricos (*ExperimentProtocol*), es decir que no estén asociados a una sesión en particular. En el figura 3.2 es posible observar como en Tycho cada nueva cuenta tiene estos elementos.

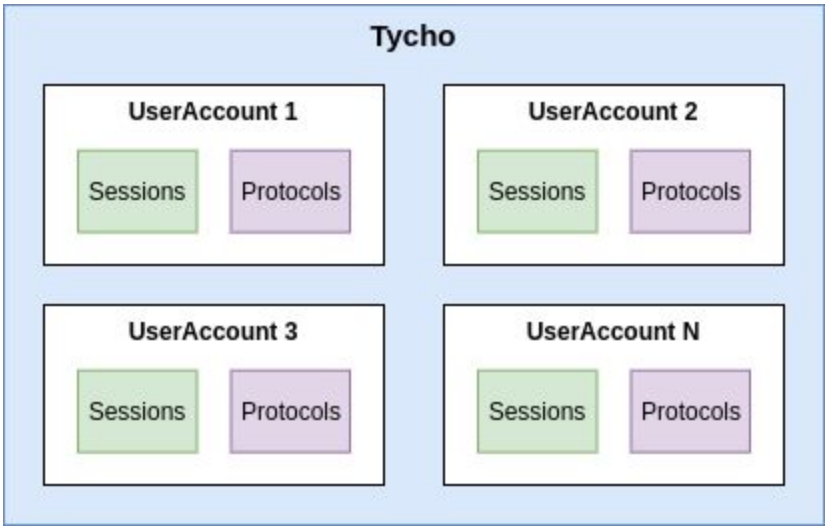


Figura 3.2. Componentes de Tycho a nivel de cuentas de usuario

Si nos centramos en los componentes del tipo *UserAccount* es posible observar cómo se componen las sesiones y los protocolos. Un *UserAccount* puede tener varias sesiones (*Session*). A su vez cada sesión tiene asociada uno o más protocolos que están compuestos por varias tareas (*Tasks*). Al mismo tiempo cada *UserAccount* puede tener varios protocolos genéricos (*Protocols*), que no van a ser ejecutables hasta que se los ponga en una sesión de experimento. Estos últimos son de utilidad a la hora de querer definir un protocolo básico que se vaya a utilizar en distintos experimentos con mínimas modificaciones.

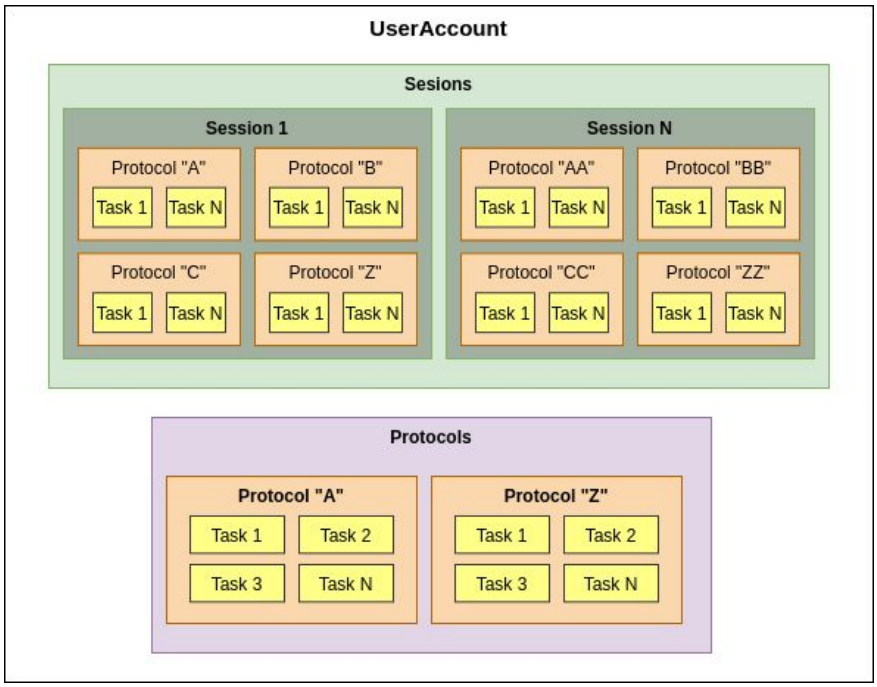


Figura 3.3. Componentes de cada cuenta de usuario (*UserAccount*)

Con las figuras anteriores, es posible detectar que predominan las instancias de protocolos. En la siguiente figura (Figura 3.4) se puede observar en detalle cómo están compuestos los protocolos que están dentro de una sesión de experimentos.

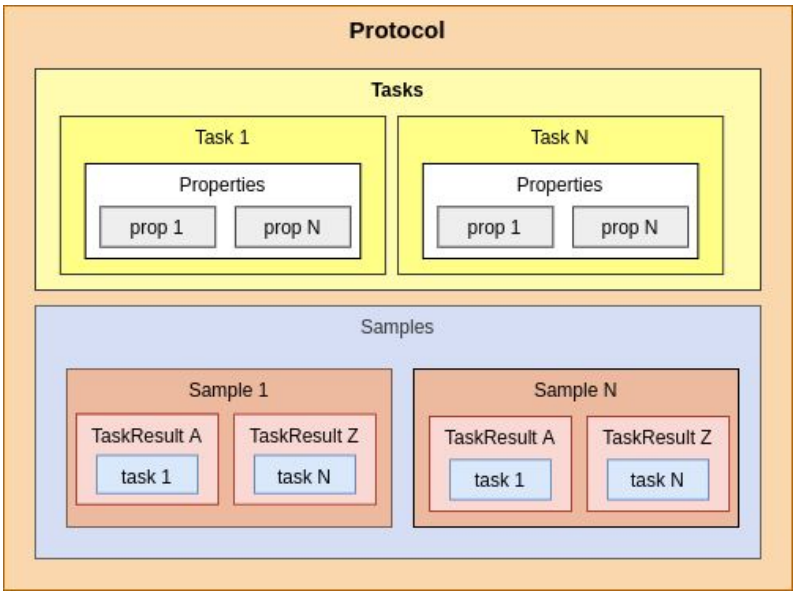


Figura 3.4. Componentes de cada protocolo asociado a una sesión

Los protocolos que ya están asociados a una sesión cuentan con las tareas (*Tasks*) que fueron incluidas para el protocolo y cada tarea tiene asociada las propiedades (*Properties*).

Además de las tareas también se encuentran las muestras de las ejecuciones del protocolo (*Sample*). Cada muestra es una ejecución por parte de un voluntario. Está compuesta por varios resultados de ejecución de tareas (*TaskResults*). Por cada muestra debe haber la misma cantidad de resultados de ejecución que tareas definidas en el protocolo, ya que cada una de los resultados están asociados a una tarea en particular (*Task*).

3.1.4 Ejecución de un experimento

Los experimentos se ejecutan desde el lado del cliente, es decir desde la extensión. Al ejecutarse estos experimentos también se recolectan datos obtenidos de la ejecución de las tareas y métricas que están asociadas a cada una de ellas. Si bien se puede deducir que no es necesaria una gran cantidad de interacción entre el *addOn* del navegador y el backend de tycho, son necesarias dos interacciones importantes:

1. **Obtener las tareas a ejecutar**: Es necesario poder consultar al backend de Tycho cuáles son las tareas que el voluntario debe ejecutar cuando ingresa el identificador del protocolo en la extensión del navegador. Una vez identificadas las tareas la ejecución de las mismas se realiza en la extensión.

2. **Persistir los resultados obtenidos**: Una vez que el voluntario logra ejecutar todas las tareas y finaliza la ejecución del experimento, se deben persistir los datos obtenidos y las métricas capturadas.

Para esta interacción entre la extensión del navegador y el backend, Tycho cuenta con una API Rest, implementada con el framework Seaside.

En la figura 3.5 se detalla cuáles son los endpoints expuestos y qué acción se realiza al interactuar con ellos:

Metodo	Endpoint	Descripción
GET	/protocols/{id}	Retorna las tareas que se deben ejecutar para el protocolo con el id pasado por parámetro. Es utilizado cuando un voluntario ingresa el identificador del protocolo en la extensión para ejecutar una prueba de usuario.
POST	/task-results	Recibe un json en el cuerpo de la petición en el que se encuentran los resultados de la ejecución de un protocolo. Es utilizado cuando un voluntario culmina la ejecución de una prueba.

Figura 3.5. API Rest de tycho

En la siguiente figura (Figura 3.6) se puede visualizar a través de un diagrama de secuencia cómo es la interacción entre la extensión y el backend. Se utiliza a modo de ejemplo el momento en el que un voluntario desea ejecutar un protocolo.

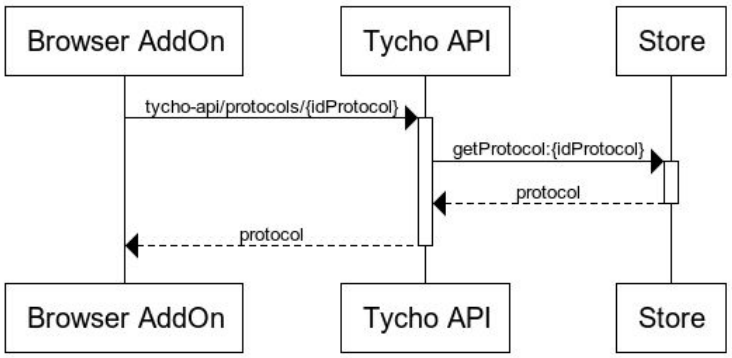


Figura 3.6. Diagrama de secuencia de la interacción de Tycho entre la extensión y el backend al introducir un id de protocolo.

Para ejecutar el protocolo el voluntario ingresa el identificador del mismo en la extensión (*Browser AddOn*). Una vez ingresado el identificador del protocolo desde la extensión se realiza un llamado a la API Rest de Tycho (*Tycho API*) solicitando las tareas que se deben ejecutar para el protocolo ingresado. En el servicio se busca en la base de datos las tareas para ese protocolo, y se retornan para que sean entregadas a la extensión. Cuando la extensión obtiene las tareas que debe ejecutar, estas serán presentadas al voluntario en pantalla.

Para explicar el proceso de ejecución de un experimento se pone a disposición el diagrama de flujos de la figura 3.7.

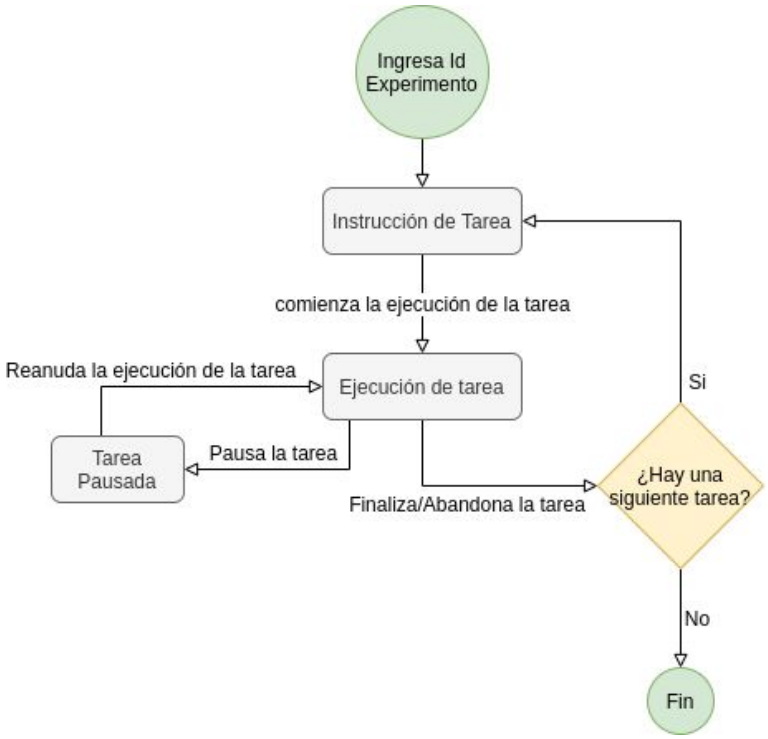


Figura 3.7. Diagrama de flujo de la ejecución de un experimento

Cuando el voluntario ingresa en el *AddOn* el identificador que le brindó el experto se presenta la la primera tarea a ejecutar. Poniendo como ejemplo el caso de una tarea del tipo *BasicTaskInstructions,* una vez que el voluntario lee las instrucciones tiene la opción de presionar un botón que da inicio a la ejecución de la tarea. Cuando esto sucede el voluntario comienza a realizar la tarea encomendada y en una de las esquinas del navegador queda una solapa que permitirá que el voluntario abandone, finalice o pause la tarea. Cuando se pausa la tarea simplemente no se contabiliza el tiempo ni las métricas que se están capturando y cuando la reanuda se sigue el curso normal de la tarea que estaba ejecutando. El voluntario también tiene la opción de finalizar o abandonar la tarea, ante cualquiera de estas opciones se finaliza la ejecución del experimento y se envían los resultados al servicio si no hay próximas tareas a ejecutar. En el caso de que haya más tareas para este experimento se vuelve a repetir el ciclo de ejecución.

Vale aclarar que las tareas respetan el orden que le dió el diseñador del experimento, es por eso que la ejecución comienza por la primera tarea y no avanza a la siguiente hasta que no haya sido finalizada o abandonada.

3.2 Resultados esperados

Del desarrollo de la presente tesina se espera poder tener a disposición mecanismos sencillos que posibiliten afrontar el costo elevado que conlleva realizar pruebas de usuario en grandes cantidades. Además, el objetivo principal que se busca es poder automatizar pruebas que permitan aplicar A/B Testing a través del desarrollo propuesto sin afectar el normal funcionamiento de Tycho.

Por otro lado es importante que la herramienta Tycho no vea afectada su facilidad de extensión para agregar nuevos comportamientos y funcionalidades a la aplicación. Para esto es necesario aplicar buenas prácticas y patrones de programación orientada a objetos.

Por lo tanto, a continuación se detallan los resultados que se desean obtener a partir del desarrollo propuesto.

Generación de pruebas A/B Testing

Tycho cuenta con un método simple de creación de pruebas de usuario. Consiste en definir la sesión de experimento con un nombre y una descripción. Luego se comienzan a agregar protocolos que en su interior tienen las tareas ordenadas según el orden secuencial que se vayan a ejecutar.

Es deseable que con la misma facilidad con la que se crea una sesión de experimento se pueda configurar la misma para que realice pruebas del tipo A/B Testing.

Distribución de las pruebas

Si bien la aplicación Tycho ya permite ejecutar pruebas de forma remota, la misma no tiene implementado un método de distribución de protocolos. Hasta el momento el diseñador debe entregarle al voluntario el identificador del protocolo que desea ejecutar. Cuando existe un solo protocolo es un trabajo que no conlleva ningún tipo de problema. El inconveniente surge cuando la sesión de experimentos tiene más de un protocolo y se desea distribuir los protocolos de manera equitativa o con un ratio en particular.

A través del presente desarrollo se desea brindar la posibilidad de definir distintas estrategias de distribución para que el diseñador entregue un identificador global de la sesión de experimento y que el protocolo que se le asigne al voluntario surja a partir de un algoritmo seleccionado en la configuración. Además se desea que sea sencilla la creación de nuevas estrategias de distribución y la modificación de la misma en tiempo de ejecución.

Clonación de protocolos para A/B Testing

Se espera poder desarrollar una opción de clonación de un protocolo que pertenezca a una sesión de experimentos. Esta necesidad surge a partir de que al realizar pruebas del tipo A/B Testing es probable que los protocolos contengan mínimos cambios entre el protocolo "A" y el protocolo "B". La implementación de esta funcionalidad conlleva una mayor velocidad a la hora de crear las pruebas.

Si bien la funcionalidad presentada está pensada para realizar A/B Testing, también se encontrará disponible para las sesiones de experimento que no lo sean.

Seguimiento del usuario

Tycho ya captura algunas métricas que fueron mencionadas anteriormente, como el tiempo transcurrido en la ejecución de una tarea, de la sesión en general o si las tareas fueron abandonadas o finalizadas correctamente.

Estas métricas sirven, pero es necesario agregar capturas adicionales de datos que permitan saber qué acciones realizó el usuario a la hora de ejecutar el experimento. Estas métricas deben ser capturadas en las tareas que lo permitan y configurables a la hora de definir el protocolo. Cuando se menciona configurables se hace referencia a que sea decisión del diseñador del experimento que estas métricas se capturen o no.

Comparación de resultados

Por último, para que sea posible realizar A/B Testing se requiere que los resultados de las ejecuciones realizadas por parte de los voluntarios se puedan visualizar de forma centralizada para que el diseñador pueda verificar la diferencia entre los protocolos que fueron ejecutados.

Gracias al ítem anterior se tienen métricas capturadas que son comparables y permiten realizar esta visualización de resultados.

El objetivo es tener un dashboard de visualización para que sea posible tener un pantallazo general de las ejecuciones realizadas.

Capítulo 4
Arquitectura

La arquitectura de Tycho está compuesta por cuatro componentes que se relacionan entre sí. En la figura 4.1 se puede ver un diagrama de la arquitectura utilizada.

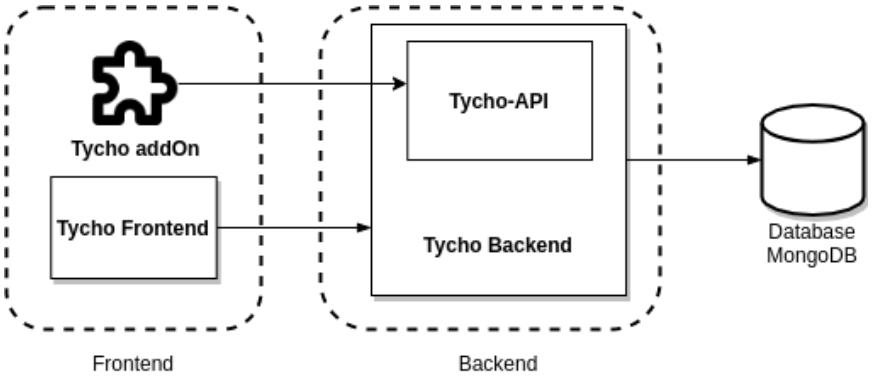


Figura 4.1. Arquitectura básica de Tycho

En primer lugar se debe mencionar a la aplicación web Tycho, la misma se encuentra desarrollada en Smalltalk y el frontend se realizó con el framework Seaside. La aplicación web Tycho está dividida en dos, por un lado se encuentra el backend (*Tycho Backend*) que es donde se encuentra implementado el diseño del modelo de datos de Tycho. Por otro lado se encuentra el frontend (*Tycho Frontend*) que es el medio que tiene el experto para definir las sesiones y poder visualizar los resultados de los experimentos ejecutados.

Para poder ejecutar los experimentos fue necesario desarrollar una extensión del navegador que le permita a los voluntarios seleccionar una sesión de experimento a ejecutar y los guíe a lo largo de la ejecución del mismo.

El tercer componente es un servicio REST (*Tycho-API*), que permite la vinculación de la extensión del navegador con el backend.

Por último, el cuarto componente es una base de datos *MongoDB*[8], en la cual se registran las sesiones de experimento, resultados, usuarios de Tycho y toda la información necesaria de los usuarios que interactúan con el sistema.

[8] https://www.mongodb.com/

4.2 Soporte de A/B testing

Para realizar A/B testing en Tycho toman un papel preponderante las sesiones y los protocolos. Como fue indicado en capítulos anteriores cada sesión de experimento puede tener uno o más protocolos. La versión anterior de Tycho cuenta con los identificadores solamente en los protocolos, es decir que cuando el diseñador de los experimentos desea que se ejecute una prueba de usuario es necesario identificar el protocolo específico que quiere que se ejecute, y luego brindarle ese identificador al voluntario.

Como fue explicado en la sección 2.1.3, a la hora de realizar A/B Testing es necesario tener dos versiones de pruebas de usuario diferentes y que sean comparables. En Tycho cada una de estas versiones están asociadas a un protocolo. Por lo tanto, se puede visualizar como ejemplo una sesión de experimento que cuente con dos protocolos: el protocolo "A" y el protocolo "B". Cuando se realiza A/B Testing es necesario hacer una distribución equitativa de las pruebas a realizar, en este caso se debería repartir equitativamente el protocolo "A" y el protocolo "B" a los diferentes voluntarios. Cómo está desarrollado Tycho hasta el momento, esto solo se podría realizar de forma manual, es decir, elegir los voluntarios que recibirán el protocolo "A" y lo que recibirán el protocolo "B". Para evitar que la distribución se realice de manera manual surge la idea de colocar un identificador de experimento para los casos en los que se desee aplicar A/B testing. Al disponer de un identificador de experimento es posible brindarle a todos los voluntarios el mismo identificador, delegando de esta manera la distribución a Tycho.

A raíz de lo comentado en el párrafo anterior surge la idea de disponer de distintos tipos de distribución de protocolos para una sesión y que no siempre sea de manera equitativa. Este es otro cambio que se realiza al modelo de Tycho, ya que no existía esta necesidad previo a la aplicación de A/B testing.

Conjuntamente para poder realizar la comparación entre la ejecución de dos o más pruebas de usuarios es necesario tener métricas que sean de utilidad para el diseñador del experimento y además sean comparables. Todas las tareas contabilizan el tiempo transcurrido en la ejecución de cada una de ellas, esa es una de las métricas comparables con la que se contaba. Por otro lado, es posible definir cuando el voluntario finaliza la tarea correctamente o la abandona seleccionando la opción disponible en cada caso. Si bien el tiempo transcurrido y el modo de finalización en la ejecución de una tarea son métricas útiles, estas no alcanzan para realizar un análisis sobre las acciones llevadas a cabo por los voluntarios en cada uno de los test. Por lo tanto, fue necesario realizar modificaciones en la extensión del navegador agregando métricas en algunas de las tareas que posibilitan esta práctica.

Al tener en cuenta los tipos de tareas de los que se disponen en Tycho es posible detectar que el tipo de tarea que sería de mayor utilidad para capturar eventos es la de "*Basic Task Instructions*", ya que su objetivo es brindar indicaciones sobre la prueba que deba realizar el voluntario y darle la libertad para que la realice. Cuando se ejecuta este tipo de tarea es posible capturar datos para hacer un seguimiento de lo realizado y posteriormente evaluar los resultados.

Existe una gran cantidad de métricas que pueden obtenerse en la ejecución de las tareas. Si bien la contabilización de clicks es una captura interesante de datos por parte del usuario, no es un modelo que por sí solo brinde información relevante (Speicher et al., 2013), sino que es posible complementarlo con otros datos relevantes que pueden ser capturados de los distintos tipos de interacciones que el usuario realiza. Por ejemplo, un tipo de interacción interesante a capturar son los movimientos que realice el voluntario con el mouse, ya que con ellos es posible detectar ciertas tendencias a la hora de la usabilidad, hasta incluso se desarrollaron técnicas para identificar procesos psicológicos a la hora de la toma de decisiones de un usuario a través de los movimientos que realiza con el mouse (Hehman et al., 2014). Por otro lado está comprobado que se encuentran altamente relacionados los movimientos del mouse con la posición de la mirada (Huang et al., 2001) y que realizar un seguimiento del mouse es una buena alternativa a realizar *eye-tracking*[9] (Chen et al., 2001).

Por otro lado, en la sección 2.2 fue mencionada una herramienta llamada "*TellMyRelevance!*" (Speicher et al., 2013) que permite capturar métricas relacionadas con los movimientos del mouse en diferentes sitios orientados a búsqueda y, en base a los datos capturados, permite realizar un análisis que brinda información relevante en cuanto a la usabilidad. A lo largo del paper se mencionan diferentes métricas que son interesantes capturar, algunas enfocadas a la dificultad por parte de los usuarios para encontrar determinado elemento, y otras métricas más generales. Entre las métricas generales mencionadas se encuentran:

- La **cantidad de clicks que realiza el usuario** en una búsqueda: Permite tener una medida básica de la interacción que requiere del usuario realizar una tarea determinada.

- El **tiempo de inactividad del usuario**: Si bien existe la posibilidad de pausar la tarea por si ocurre algún inconveniente durante la ejecución del experimento, puede pasar que el voluntario no la utilice y simplemente deje de interactuar con el experimento. Capturar el tiempo de inactividad va a permitir conocer cuánto tiempo estuvo realmente activo el voluntario.

[9] *Eye-tracking: Tecnología basada en el seguimiento ocular para identificar que secciones observa un usuario al interactuar con una aplicación.*

- La **velocidad en la que el usuario mueve el mouse** durante una búsqueda: Esta métrica puede dar indicios de lo que realiza el usuario, por ejemplo una velocidad de cursor lenta en un sitio de búsqueda puede dar indicios de que el voluntario se encuentra observando los resultados lentamente mientras que una mayor velocidad se puede entender como que encontró rápidamente el resultado que buscaba y está dispuesto a abrirlo (Guo & Agichtein, 2012).

El trabajo de Speicher está orientado a páginas SERP[10] y además de las métricas generales se capturan métricas más específicas de los resultados deseados. Estos datos capturados brindan información relevante sobre cuánto le costó al usuario encontrar el resultado que buscaba. Si bien está orientado a páginas SERP, este concepto puede llevarse a cualquier ámbito en el cual se puedan encontrar elementos en la página sobre los cuales puedan ser de interés para el experto saber cómo el usuario interactúa con ellos. Entre esas métricas más específicas sobre los elementos se encuentran:

- El tiempo que tardó el usuario en pasar por encima del elemento.
- La cantidad de veces que el usuario pasa por encima del elemento.
- La cantidad de veces que el usuario clickea sobre el elemento.

Cada una de estas métricas cobran sentido en base a lo que se esté probando y al escenario que plantee el diseñador del experimento para la tarea. Es por esto que se ofrece el poder de elección sobre qué métricas genéricas se van a capturar cuando la tarea se ejecute, siendo todas o algunas en particular. De la misma forma que con las métricas genéricas se ofrece la posibilidad de capturar las métricas sobre elementos html de las páginas que el experto indique, cada una de las métricas que se permite capturar pueden ser habilitadas bajo demanda.

Al contar con los protocolos y las métricas capturadas falta una forma de visualizar los resultados para que la comparación de las ejecuciones sea lo más sencilla dentro de las posibilidades. Tycho cuenta con un método de visualización de resultados de las ejecuciones en la que se dispone de una tabla donde cada fila es una tarea ejecutada. Si bien con este método es posible visualizar toda la información, esta no es una manera adecuada para poder comparar las ejecuciones a simple vista. Fue necesario realizar cambios en este punto, agregando un módulo donde se procesan los datos obtenidos y se presentan, a través de gráficos y valores promediados de las ejecuciones, los resultados de cada protocolo.

En los próximos ítems se detalla cuáles fueron los cambios realizados en cada una de las áreas del sistema para posibilitar la realización de A/B testing. Los cambios se separan en tres puntos:

[10] SERP: Search Engine Results Page (Página de resultados del motor de búsqueda).

- ➔ Los cambios realizados en el backend de Tycho, entre los que se encuentran cambios en el modelo y la implementación de estrategias de distribución para experimentos del tipo A/B Testing.

- ➔ Los cambios que fueron necesarios en la extensión del navegador, desde las modificaciones para iniciar la ejecución con un identificador de experimento hasta la captura de nuevas métricas.

- ➔ Las modificaciones en el frontend para visualizar el nuevo tipo de experimento y poder presentar resultados útiles para los diseñadores de experimentos.

4.2.1 Cambios en el backend

En cuanto al modelo fue necesario realizar algunas modificaciones con respecto a lo que ya se encontraba implementado en la herramienta. En la figura 4.2 se podrán ver con color celeste las clases que fueron agregadas en Tycho y en color rosado las clases en las que fueron agregados atributos que figuran en negrita. Los métodos que fueron heredados de la primera implementación de Tycho se excluyeron del diagrama para simplificarlo, por lo tanto los métodos que figuran son los que se incorporaron para aplicar A/B Testing.

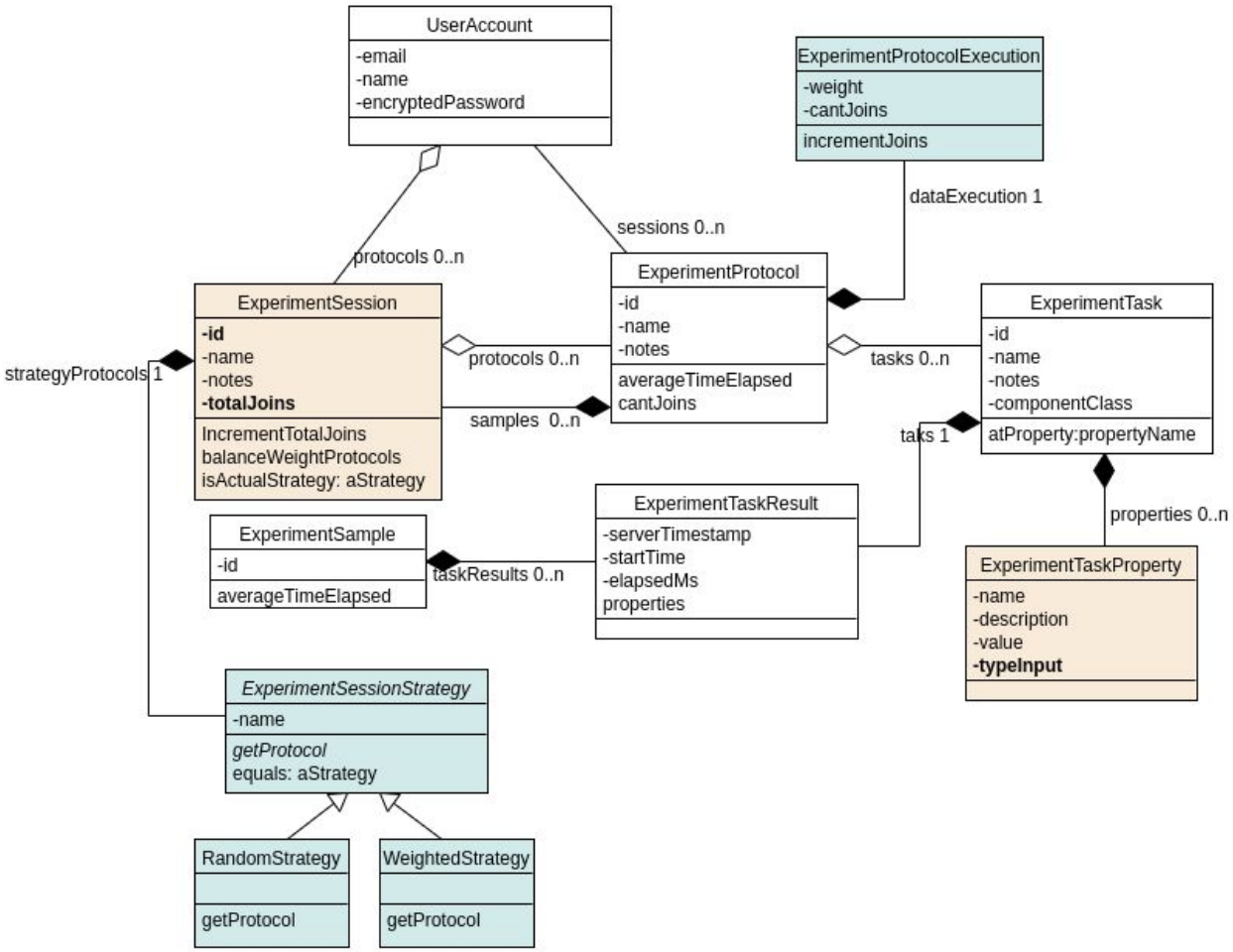


Figura 4.2. Modelo de clases de Tycho con las modificaciones introducidas para aplicar A/B Testing.

Almacenar información de cada ejecución

Durante el desarrollo de la aplicación de A/B Testing fue necesario recolectar información de la cantidad de ejecuciones que se van realizando tanto por protocolo como por sesión de experimento. Para esto fue creada la clase *ExperimentProtocolExecution* asociada a *ExperimentProtocol* bajo el nombre de *dataExecution*. En esta clase se incrementa un contador cada vez que un voluntario ejecuta alguno de los protocolos. Además de ser una información interesante para el diseñador del experimento el contador es utilizado para las estrategias de distribución de protocolos.

También se agregó un contador en la clase *ExperimentSession*, de esta manera es posible saber cuántas ejecuciones fueron realizadas para esa sesión ya sea realizando A/B Testing o a través del id del protocolo.

Estrategia de distribución de protocolos

Como parte del desarrollo de la tesina se agregó el identificador del experimento en la clase *ExperimentSession*.

Además se desarrollaron las estrategias de selección de protocolos. Para esta funcionalidad fue creada una clase abstracta llamada *ExperimentSessionStrategy* la cual tiene un método abstracto denominado *getProtocol*. Las clases concretas que extiendan de esta deben implementar el método *getProtocol*, cuya función es retornar el próximo protocolo a ejecutar dependiendo de la estrategia implementada. De esta manera se puede observar como se hace uso del patrón strategy para esta solución.

Al crear una sesión de experimento con más de un protocolo es posible seleccionar la estrategia a utilizar. Se desarrollaron dos estrategias, pero es posible crear otras y adaptarlas al sistema gracias a la utilización del patrón. Las estrategias implementadas son las siguientes:

- **RandomStrategy**: Como su nombre en inglés lo indica, al ejecutar un experimento con el identificador del mismo, se retornará aleatoriamente uno de los protocolos asignados al experimento.

- **WeightedStrategy**: Con esta estrategia se busca asignar una ponderación a cada protocolo para que a la hora de seleccionar alguno para la ejecución de la sesión se respete el peso asignado.

  Para la segunda estrategia el diseñador del experimento puede definir, a través de porcentajes, cuál es la ponderación que desea asignarle a cada protocolo a la hora de ejecutarse. La suma de los porcentajes asignados a cada protocolo debe ser igual a cien, permitiendo de esta manera que siempre se asigne la totalidad de las ejecuciones. Si el peso se reparte en partes iguales entre los protocolos la selección se realizará al estilo *Round Robin*[11], por el contrario si alguno de los protocolos tiene una ponderación mayor que otros, este protocolo va a ser seleccionado con la frecuencia indicada. Es necesario aclarar que no es un método probabilístico sino determinístico.

  Para implementar la estrategia *WeightedStrategy* se utilizó la clase *ExperimentProtocolExecution* para guardar el peso que se le asigna a cada protocolo. El peso es guardado en el atributo *weight.*

[11] Round Robin: Método para seleccionar todos los elementos equitativamente.

4.2.1.1 API de Tycho

Otro de los cambios que fueron realizados es la disponibilización de un nuevo endpoint a la hora de solicitar un protocolo a ejecutar. En la implementación base de Tycho se cuenta con un endpoint que permite solicitar las tareas de un protocolo en particular. Se extendió la API para que sea posible solicitar las tareas a ejecutar para una sesión de experimento, es decir enviando el identificador del mismo.

A partir de este cambio, a la tabla de la figura 3.5 se le debe agregar el endpoint que de la siguiente figura (4.3).

Metodo	Endpoint	Descripción
GET	session/{id}/protocols	Retorna las tareas del próximo protocolo que se debe ejecutar para el experimento con el id pasado por parámetro. El protocolo a ejecutar se selecciona en base a la estrategia seleccionada para la sesión de experimentos.

Figura 4.3. Endpoint agregado a Tycho para brindar soporte de A/B Testing

4.2.2 Extensión del navegador

Las extensiones del navegador, también conocidas como "*AddOn*", no son más que un pequeño módulo que generalmente está escrito con código HTML, CSS y Javascript, y que permiten extender al navegador agregando un comportamiento adicional a los sitios web que son visitados cuando la extensión se encuentra habilitada.

Los navegadores brindan acceso a una API que permite a los desarrolladores interactuar con distintos componentes del browser para realizar diferentes acciones sobre el mismo, consultar el estado del navegador que está ejecutando la extensión, enviar mensajes a las pestañas abiertas, entre otras cientos de funcionalidades.

Tycho cuenta con una extensión del navegador que es utilizada a la hora de realizar las pruebas de aplicaciones web. El actor que utiliza el AddOn es el voluntario que ejecutará las pruebas. El principal objetivo de realizar la ejecución a través de esta herramienta es el de poder guiar al voluntario a través de las tareas que el experto diseñó en la aplicación de Tycho y capturar métricas durante la ejecución del experimento. A continuación se describen las ventajas de contar con un AddOn para realizar las pruebas de usuario:

- **Realizar pruebas moderadas de forma remota**: El voluntario ingresa el identificador del experimento en cuestión y, a partir de una interacción entre el *addOn* y el servidor de Tycho, se presentan las tareas que deben guiar al voluntario a lo largo de la prueba. La tarea que debe hacer el usuario de la extensión es simplemente instalarla e ingresar el identificador de la prueba que le envió el diseñador de experimentos.

- **Captura de datos en ejecución**: Cuando se agrega una extensión al navegador es posible realizar un seguimiento de lo que el usuario está realizando en los distintos sitios web que visita, gracias a que es posible adicionar código javascript a estos sitios mientras la extensión esté habilitada. A partir de esta herramienta se pueden capturar métricas que brinden datos precisos de las acciones que el voluntario realizó sobre la sesión de experimento ejecutada.

- **Alta flexibilidad de pruebas de usuario en aplicaciones web**: Para realizar pruebas de usuario no es necesario realizar modificaciones en la aplicación web que se está probando, ya que tanto la moderación de la prueba para el voluntario como la captura de información se realiza a través del *addOn*. Otro punto a destacar es que es factible realizar pruebas de usuario sobre cualquier aplicación web, no es de suma necesidad que sea una aplicación propia la que se encuentra siendo probada.

A partir de las características generales mencionadas es posible deducir que realizar la ejecución de experimentos a partir de una extensión del navegador es un método que aporta:

➔ **Simplicidad** a la hora de ejecutar una prueba.

➔ **Flexibilidad** para diseñar pruebas de usuario sobre distintas aplicaciones web.

➔ **Amplitud** sobre el seguimiento que se puede realizar al voluntario que ejecuta las pruebas.

➔ **Facilidad** a la hora de agregar nuevas capturas de métricas y tipos de tareas ya que las extensiones se pueden escribir con código javascript.

4.2.2.1 Arquitectura de la extensión

La extensión del navegador está implementada en base a la API brindada por Firefox denominada "*WebExtension*". Si bien cada navegador maneja sus API de manera diferente, con

el paso del tiempo *WebExtension* brinda un alto grado de compatibilidad de sus APIs con la de los otros navegadores altamente utilizados como pueden ser Chrome, Opera y Edge[12].

Estructura general

No existe una extensión del navegador si la misma no cuenta con el archivo llamado "*manifest.json*". En este archivo están definidos todos los elementos que se van a utilizar, desde el nombre de la extensión, pasando por los permisos que le solicita la extensión al browser, hasta los scripts utilizados.

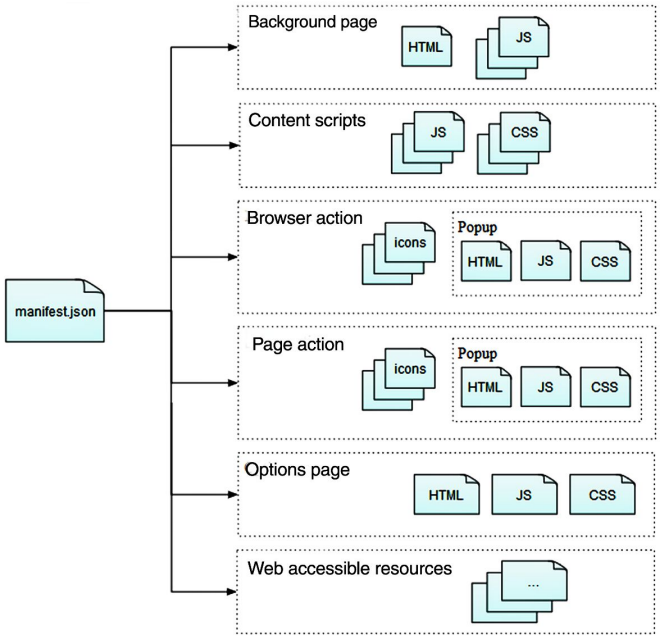


Figura 4.4. Componentes del *manifest.json*. Fuente: https://developer.mozilla.org/es/docs/Mozilla/Add-ons/WebExtensions/Anatomia_de_una_WebExtension

Cómo es posible observar en la figura 4.4, el archivo *manifest.json* permite configurar y definir datos de la extensión a partir de definir los enlaces a distintos tipos archivos que la componen. A continuación se realiza una breve explicación de que debe contener cada uno de los elementos más importantes del *manifest.json*:

- *Background Scripts*: Son scripts que corren en segundo plano y generalmente cumplen el rol de "controlador" de la extensión. Son de utilidad para ejecutar tareas que van más

[12] Descripción de la compatibilidad de "*WebExtensions*" con otros navegadores https://developer.mozilla.org/es/docs/Mozilla/Add-ons/WebExtensions/Chrome_incompatibilities

allá del tiempo de ejecución que tuvo una página web en particular o las distintas ventanas del navegador, o simplemente para mantener estados en la extensión. Para este tipo de problemáticas están destinados los scripts del tipo background. Son cargados cuando la extensión está habilitada y se mantienen hasta que la misma se desinstala o deshabilita. A través de estos scripts es posible utilizar cualquier API que brinda *WebExtension* siempre y cuando se haya solicitado el permiso correspondiente en el *manifest.json*.

- *Content Scripts*: Los "*Content scripts*" son un conjunto de archivos javascript que se ejecutarán en el contexto de la página web que está siendo renderizada por el navegador, permitiendo realizar modificaciones en el DOM y capturar eventos que sucedan en la misma.

- *Browser Action*: Sirve para definir cómo se verá la extensión en la barra de tareas, como por ejemplo el icono y el nombre que tiene la extensión.

- *Web accessible resources*: Brinda la posibilidad de que desde el contenido empaquetado se acceda a distintos recursos, como pueden ser librerías, imágenes o páginas web.

Uno de los elementos que no es obligatorio y por esa condición no se encuentra en la figura anterior es el de "*permissions*". Para utilizar algunas de las APIs que brindan los navegadores es necesario solicitar permisos sobre las mismas. Este elemento en el manifest permite realizar esta acción. Además en "*permissions*" también es posible indicar sobre qué dominios será ejecutable el AddOn, escribiendo las direcciones de las páginas correspondientes.

APIs utilizadas

Dentro de la implementación del AddOn son utilizadas diferentes APIs para implementar la funcionalidad deseada. En su mayoría el uso de estas APIs se da por los scripts de Background, aunque existen algunas que están disponibles para que se usen en *content scripts* [13].
En el AddOn de Tycho son utilizadas principalmente las APIs que se enumeran a continuación. Las definiciones fueron obtenidas de la documentación oficial de Mozilla.

Runtime

Este módulo provee información de la extensión y el ambiente en el que se está ejecutando. El módulo es utilizado principalmente para la comunicación entre los content scripts y los

[13] APIs utilizables en *content scripts*:
https://developer.mozilla.org/en-US/docs/Mozilla/Add-ons/WebExtensions/Content_scripts#WebExtension_APIs

background scripts. Tanto los scripts de "background" como los scripts de "content" se suscriben a distintos listeners del navegador que procesarán todos los mensajes enviados desde el otro extremo.

Para la suscripción a un listener se debe ejecutar la siguiente sentencia:

```
browser.runtime.onmessage.addlistener(function (message){..});
```

El código escrito dentro de la función anónima será lo que se ejecute cuando un mensaje sea enviado. Para enviar un mensaje que sea procesado por el listener anterior se debería ejecutar la siguiente sentencia:

```
var message =  {"aProp": aValue};
browser.runtime.sendMessage(message);
```

Luego de ejecutarse la sentencia anterior se estará enviando el objeto message a los diferentes listeners que estarán escuchando para procesarlo de alguna manera.

Tabs

Este módulo es utilizado para interactuar con las distintas pestañas del navegador. Si bien no se permite acceder al contenido que la pestaña está mostrando, si es posible agregar código javascript o css en ella.

En el desarrollo de Tycho se utiliza principalmente para asegurarse de que la ejecución de un experimento continúe normalmente si el voluntario abre otra pestaña.

En su mayoría son utilizados dos métodos. Uno es *tabs.query* que permite consultar todas las pestañas abiertas y además filtrar en base a distintas condiciones. La sentencia para ejecutar las consultas es la siguiente:

```
// Retorna todas las tabs abiertas
var tabs = browser.tabs.query({})
```

Como existe la posibilidad de enviar mensajes entre los *content scripts* y los *background scripts* es posible enviar mensajes a tabs en particular. Para enviar mensajes se debe escribir la misma sentencia que en *runtime* ("*sendMessage*") pero invocando a la api "*tabs*". La principal

diferencia está en el objeto que se debe mandar, ya que es necesario indicar el identificador de la pestaña, el mensaje que se desea enviar y es posible enviar un objeto opcional. Con lo mencionado la sentencia que se debe utilizar para enviar mensajes a los tabs es la siguiente:

```
var msg = browser.tabs.sendMessage(
  tabId,     // identificador de la pestaña
  message,   // un mensaje
  options    // objeto opcional
)
```

localStorage

El módulo permite acceder al almacenamiento del navegador a nivel de extensiones. Permite guardar objetos en el browser a partir de una clave y recuperarlos. Los métodos utilizados en Tycho son asincrónicos y el objetivo de ambos es guardar y recuperar información.
Para guardar un elemento, se utiliza el método *set(object)*. El objeto que se va a guardar tiene uno o más elementos del tipo clave valor, si alguna clave ya existe la misma va a ser reemplazada por el elemento enviado.

```
let settingItem = browser.storage.<storageType>.set(object)
```

Para recuperar información es utilizado el método *get(keys)*, al cual se le envía una clave "*keys*" y en base a la clave enviada se retornan los objetos que estén guardados en el local storage bajo esa clave. Si se envía *null* o un objeto "*undefined*" se retornará la lista completa de objetos que están en el local storage.

```
let results = browser.storage.local.get(
keys    // null, string, object o un arreglo de strings
)
```

I18n

Para permitir tener la aplicación disponible en base al idioma que el usuario elija o tenga configurado en el navegador, es necesario utilizar este módulo. Es posible utilizar esta API para seleccionar los mensajes que están configurados en base al idioma que se está utilizando.

Para hacer uso de i18n tiene que ejecutar el mensaje *getMessage(messageName)*, donde *messageName* es la clave del mensaje definido.

```
browser.i18n.getMessage(
  messageName,  // string
)
```

browserAction

Este módulo permite realizar modificaciones sobre el botón que figura en la barra de navegación. Es posible definir funciones para cuando se hace click sobre el botón, setear un icono en particular, setear un título, un popUp, habilitar o deshabilitar el botón entre otras acciones.

Elementos principales de Tycho

En el *manifest.json* se referencia a distintos componentes del *addOn* que son los encargados de ejecutar su comportamiento. En la figura 4.5 se encuentra un diagrama en el cual se puede observar la distribución e interacción entre los elementos del *addOn*. Para dar a conocer con mayor profundidad la implementación, posteriormente se realiza una explicación de cuál es la función de cada uno de ellos.

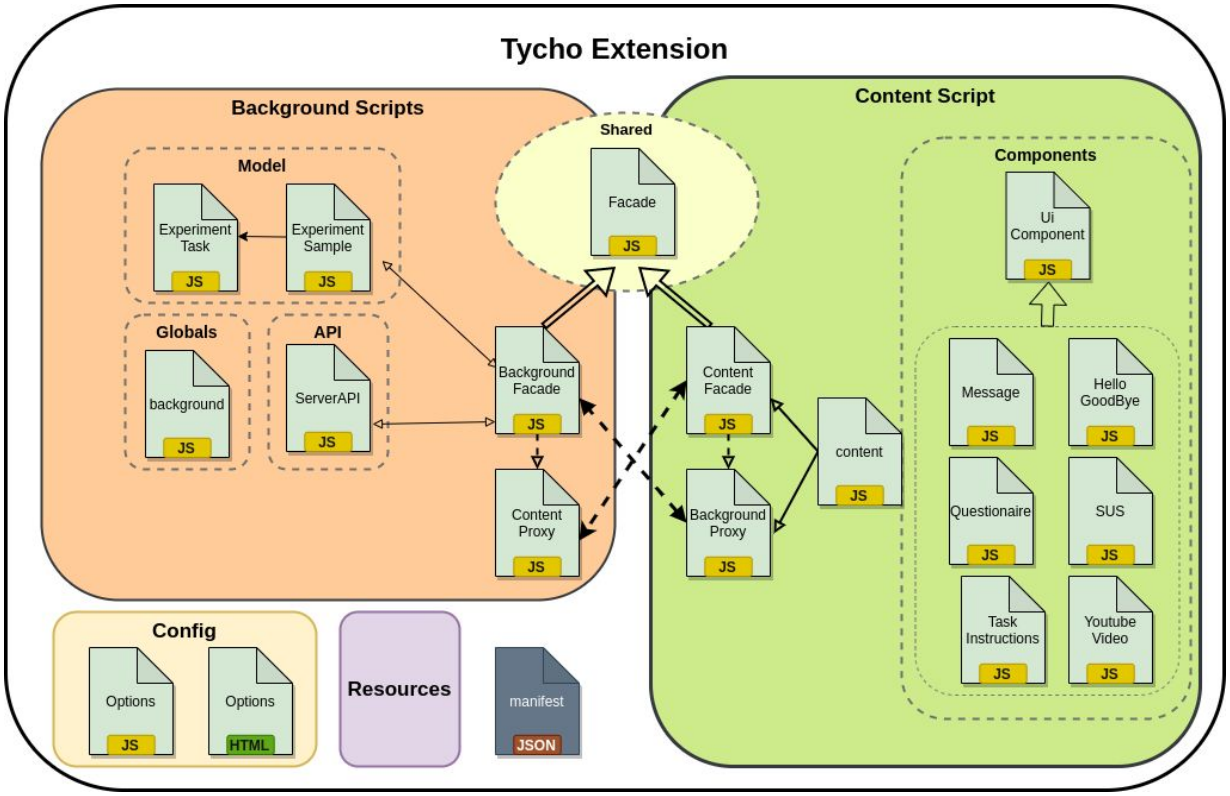


Figura 4.5. Componentes de la extensión de Tycho

Como fue mencionado en la sección anterior, las extensiones se componen de dos tipos de scripts, los "*Background Scripts"* y los "*Content scripts*". Si bien cada uno tiene objetivos diferentes es necesario mantener una comunicación entre ellos. Esta comunicación se realiza a través de la API del navegador "*runtime*" que permite enviar y recibir mensajes por medio de métodos particulares.

Al mismo tiempo es posible observar en la figura que en Tycho la comunicación entre los scripts de *Background* y los de *Content* se realiza mediante los objetos de tipo *Facade* y los objetos de tipo *Proxy*, es decir, *BackgroundFacade* intercambia mensajes con *BackgroundProxy* y *ContentProxy* hace lo propio con *ContentFacade.*

Si bien *ContentFacade* y *BackgroundFacade* son scripts de tipo "*content*" y "*background*" respectivamente, ambas clases son subclase *Facade*, la cual se encuentra en el módulo *shared*.

Facade tiene un método implementado llamado "*handle*" cuya función es interpretar los mensajes que llegan. Estos mensajes tiene objetos con un formato "*rmcRequest*".

Los objetos del tipo "*rmcRequest*" que se transmiten entre ambos contextos son invocaciones a métodos de forma remota. Estos objetos tienen una estructura un tanto particular ya que cuentan

con dos propiedades que sí o sí deben contener: una es el nombre del método que se desea invocar (*methodName*) y la otra los argumentos que se envían (*arguments*).

```
rmcRequest: {
methodName: 'messageX',
arguments: {arg1: 'value', arg2:'value'}
}
```

El método handle que se implementa en la clase abstracta *Facade* retorna una promesa en base a las siguientes condiciones:

1. El *rmcRequest* recibido es un objeto que tiene definidas las propiedades *methodName* y *arguments*.
2. El valor de la propiedad *methodName* es un método de la clase que está invocando a *handle*

Si ambas condiciones se cumplen se invoca el método con los argumentos del *rmcRequest* y se resuelve la promesa satisfactoriamente con el resultado de la ejecución de dicho método invocado. En caso de que alguna de las condiciones falle se retorna con error la promesa creada.

A continuación se pasa a explicar los distintos componentes que se encuentran en la figura anterior.

Background scripts

Model (ExperimentSample.js | ExperimentTask.js)

Las clases que se encuentran dentro del módulo "*Model*" representan a los experimentos y sus tareas. Los métodos que componen al *ExperimentSample* son invocados a través del *BackgroundFacade*. *ExperimentSample* tiene métodos para iniciar un experimento, solicitar cual es la próxima tarea a ejecutar, solicitar el identificador del experimento, entre otros. Por la lógica que tiene Tycho *ExperimentSample* tiene una colección de *ExperimentTask*.
Ambas clases se utilizan tanto para comenzar la ejecución de un experimento como para enviar los resultados finales.

ServerAPI.js

Es un servicio cuyos métodos tienen el objetivo de realizar los llamados a la API Rest de Tycho.

Es instanciada e invocada solamente por *BackgroundFacade*.

background.js

Cuando la extensión se inicia se ejecuta el script que se encuentra en Background scripts llamado "***background.js***", en este script la primera acción que se realiza es verificar si los datos de configuración están cargados. Los datos de configuración son almacenados en el "*local storage*" del navegador mediante la API *browser*, si los datos no están cargados se inicializan. Una vez asegurados los datos de configuración general se procede a inicializar el Background. Para ello es creada una instancia de *BackgroundFacade* y además son agregados algunos listeners entre los que se destacan aquellos que se ejecutarán ante mensajes enviados a través del método *browser.runtime.sendMessage()* de la API del navegador. Estos mensajes serán procesados por medio de una instancia de *BackgroundFacade* que fue creada previamente y contiene manejador para procesar los mensajes enviados que se explicó anteriormente.

BackgroundFacade.js

BackgroundFacade es una clase que implementa el patrón de diseño estructural *Facade*[14]. Esta clase tiene implementados métodos simples para interactuar tanto con la clase *ServerAPI* como con *ExperimentSample*.
Al ser principalmente utilizada por *BackgroundProxy* desde los subsistemas de "*content script*" se la puede ver como un punto de entrada a los "scripts de background".

ContentProxy.js

La función principal de esta clase es la de enviar mensajes que serán capturados "*content scripts*". Es llamado por *BackgroundFacade* y el método más utilizado es el de solicitar que se rendericen las tareas que se deben ejecutar.
Una de las particularidades es que hace uso de la API del navegador "*tabs*", ya que cuando se debe mostrar información nueva en la extensión es necesario que la acción se realice en todas las tabs que se encuentren abiertas.

Content scripts

Content.js

A nivel *content script*, *content.js* es el que da inicio a este módulo. Para ello tiene una instancia

[14] Facade: Patrón de uso común cuyo objetivo es simplificar la comunicación entre una clase y otra clase compleja u otro subsistema de clases.

de *ContentFacade,* la cual es utilizada en uno de los listeners definidos para responder a los mensajes del background.

ContentFacade.js

Hereda de Facade, y cumple la misma función que el *BackgroundFacade* en los *background scripts*: manejar los mensajes enviados mediante la API del navegador "*runtime*" y estructurar los mensajes que se quieran enviar hacia los *background scripts*.

BackgroundProxy.js

Clase utilizada por el *ContentFacade* que se encarga de realizar el envío de mensajes hacia el background. A diferencia del *ContentProxy* no hace uso del módulo *tabs*, ya que los mensajes enviados son para realizar consultas/modificaciones sobre el modelo o consumir la API rest.

Components (módulo)

Cada componente es un tipo de tarea que se puede ejecutar durante un experimento. Teniendo una clase por cada tipo de tarea es posible definir las particularidades de cada una ya sean visuales, en la captura de métricas, o en las acciones que deben realizar al comenzar y/o al finalizar.
Todos los componentes heredan de la clase abstracta *UIComponent*, la misma cuenta con el comportamiento común que tienen las diferentes tareas. Los métodos que se implementan pueden ser sobrescritos por las subclases. Además todas las subclases de *UIComponent* deben implementar el método *buildComponent()* cuya función es retornar el código html que tendrá la tarea al ser presentada al voluntario que la ejecuta.

Otros componentes

Config

Además de los módulos principales existe un módulo de configuración. El mismo es definido en el manifest.json y tiene como funcionalidad permitir realizar modificaciones sobre la extensión.

Resources

Son diferentes recursos a los que se pueden acceder desde los diferentes módulos que componen al addOn, como por ejemplo imágenes o archivos .css, etc. Por ejemplo, bajo el directorio "*resources"* de la extensión de Tycho se encuentran los iconos de la aplicación.

4.2.2.2 Soporte de A/B Testing en el AddOn

Para que Tycho pueda brindar la posibilidad de realizar A/B Testing fue necesario realizar cambios en todos los elementos de la herramienta y la extensión desarrollada no fue la excepción.

Comienzo de ejecución de un experimento en base a su ID

En la primera versión de Tycho solo era posible comenzar la ejecución de un experimento a partir del identificador del protocolo que pertenezca a un experimento. Para que sea posible realizar A/B testing fue necesario realizar cambios en la extensión, ya que es necesario poder ingresar el identificador del experimento además del del protocolo.

Cuando el voluntario inicia la ejecución del experimento se le presenta una pantalla como la de la siguiente figura (Figura 4.6) para que el mismo ingrese el identificador enviado por el diseñador.

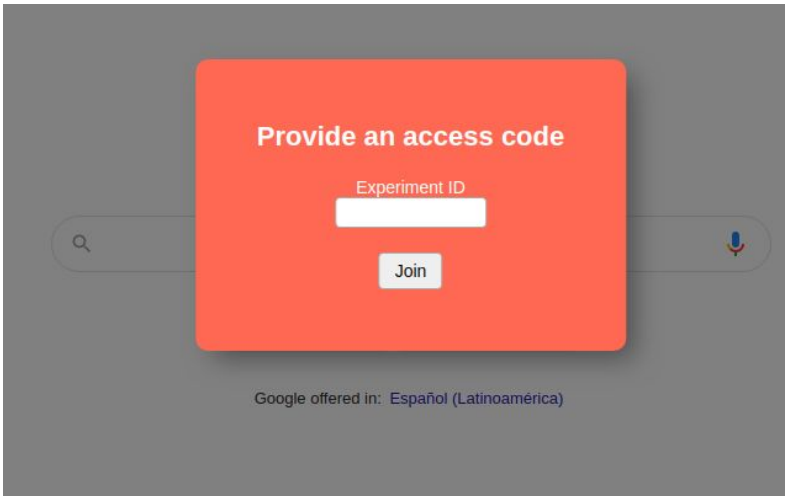


Figura 4.6. Componentes de la extensión de Tycho

Hasta el momento los identificadores de de los protocolos eran un número único dentro de la aplicación. Para permitir ingresar dos tipos de identificadores fueron evaluados dos métodos en este punto:

1. Disponer de dos campos para ingresar el identificador según el tipo en uno u otro campo.

2. Que los identificadores tengan un patrón en su conformación que permitan diferenciarlos.

Se optó por la segunda opción generando identificadores diferentes para cada tipo de elemento. De esta manera al ver el identificador a simple vista es posible saber de qué tipo se trata: los identificadores de experimento comienzan con la letra "E" y los de protocolos con la letra "P".

En base a lo mencionado al ingresar un identificador lo primero que se hace es verificar de qué tipo es el mismo y en base a esto solicitar las tareas ejecutar al endpoint de los experimentos o al de los protocolos.

Captura de métricas

Cada tarea está orientada a guiar al voluntario de manera diferente a lo largo del experimento, es por eso que en Tycho se capturaban algunas métricas a modo general:

- ➔ El tiempo que el voluntario demoró en la ejecución de cada tarea
- ➔ El tiempo transcurrido en la ejecución total del experimento
- ➔ El estado en el que fue finalizada la tarea en el caso de que esto sea posible de evaluar (abandonada o finalizada satisfactoriamente).

Para realizar A/B testing es necesaria la comparación de datos para verificar que experimento arroja mejores resultados con respecto a lo que el diseñador desea evaluar. Al evaluar los distintos tipos de tareas que se mencionaron en la sección 3.1.1 se puede observar que el tipo de tarea que permite realizar en cierta medida un seguimiento del usuario es el de "*Basic Task Instructions*". Este tipo de tarea consiste en brindarle al voluntario indicaciones básicas de lo que se espera que haga. Cuando el voluntario comienza la ejecución la interfaz de Tycho es minimizada en el navegador permitiéndole realizar las acciones en pos de resolver la petición. Al tratarse de una tarea en el que el voluntario tiene libertad es posible llevar un cierto seguimiento del usuario a lo largo de la ejecución de la misma.

Las métricas capturadas en el tipo de tarea *Basic Task Instructions* se pueden separar en dos grupos: Aquellas métricas que se capturan de forma general, y las que están relacionadas con elementos html de una página determinada.

Las métricas generales están pensadas para brindar una visión global de cómo se realizaron las ejecuciones de los experimentos. Además del tiempo que tardó la ejecución en general las métricas generales capturadas son:

- Cantidad de Clicks que realiza el voluntario

- Velocidad en la que el usuario realizó los movimientos del cursor
- Tiempo de inactividad

Por otro lado se encuentran las métricas capturadas sobre elementos html. Al crear una tarea del tipo "*Basic Task Instructions*" existe la opción de ingresar una página web y un elemento html a través de su id o clase. Cuando el voluntario ingrese a esa página web se cargarán los listeners que permitirán capturar eventos sobre los elementos html también indicados por el diseñador del experimento. Entre las métricas que pueden ser capturadas sobre elementos particulares se encuentran las siguientes:

- Cantidad de veces que el usuario pasa por encima del elemento
- Tiempo que tarda el usuario hasta llegar por primera vez al elemento
- Cantidad de clicks sobre el elemento

Para capturar estas métricas en las tareas del tipo "*Basic Task Instructions*" se realizaron modificaciones en los *content scripts*, más precisamente en el componente *TaskInstructions* que hereda de *UiComponent*. Este tipo de tarea cuenta con el método *startTask()* que se ejecuta cuando la misma comienza a ejecutarse. Dentro de este método se agregan los listeners javascript en base a lo que fue seteado por el diseñador. Estos listeners irán capturando y contabilizando los eventos de clicks, los movimientos que realice con el mouse el usuario y el tiempo en el que el mismo permanezca inactivo. El método utilizado para capturar los diferentes eventos consiste en inyectar scripts en las páginas web que el voluntario está observando sin realizar modificaciones interactivas ni visuales. De esta manera cada movimiento del mouse es detectado por uno de estos listeners para calcular las distancias recorridas, cada click realizado es capturado por otro listeners para contabilizarlo. Por otro lado, se agregó un timeout que se ejecuta cuando pasa un determinado periodo de tiempo sin ninguna interacción con la aplicación por parte del usuario, ya sea con teclado o mouse.

Como fue mencionado, la tarea puede ser pausada por el voluntario, en ese momento los eventos deben ignorarse y volver a reactivarse cuando la prueba vuelva a ejecutarse. Dicha implementación fue realizada en los métodos *pauseTask()* y *resumeTask()* que se ejecutan cuando la tarea es pausada y reactivada respectivamente, manejando un estado de la tarea.

Los listeners son capturados en la página visitada, pero es interesante aclarar que si el voluntario cambia de pestaña o pantalla durante la ejecución la misma seguirá contabilizando los eventos. Esto es posible porque cada vez que se tiene que representar la tarea en el navegador es ejecutado un método llamado *renderTask()* el cual permite realizar una verificación de si la tarea está activa o pausada. Si se encuentra activa los listener se cargan en la página que se está

visualizando y en el caso de que este pausada se cargan pero no se capturan hasta que la tarea cambie su estado y continúe su ejecución.

Por último, al abandonar o finalizar la tarea es necesario recolectar los datos capturados para que cuando finalice el experimento puedan ser enviados al servidor esto se realiza en los métodos *abandonTask()* y *finishTask()*.

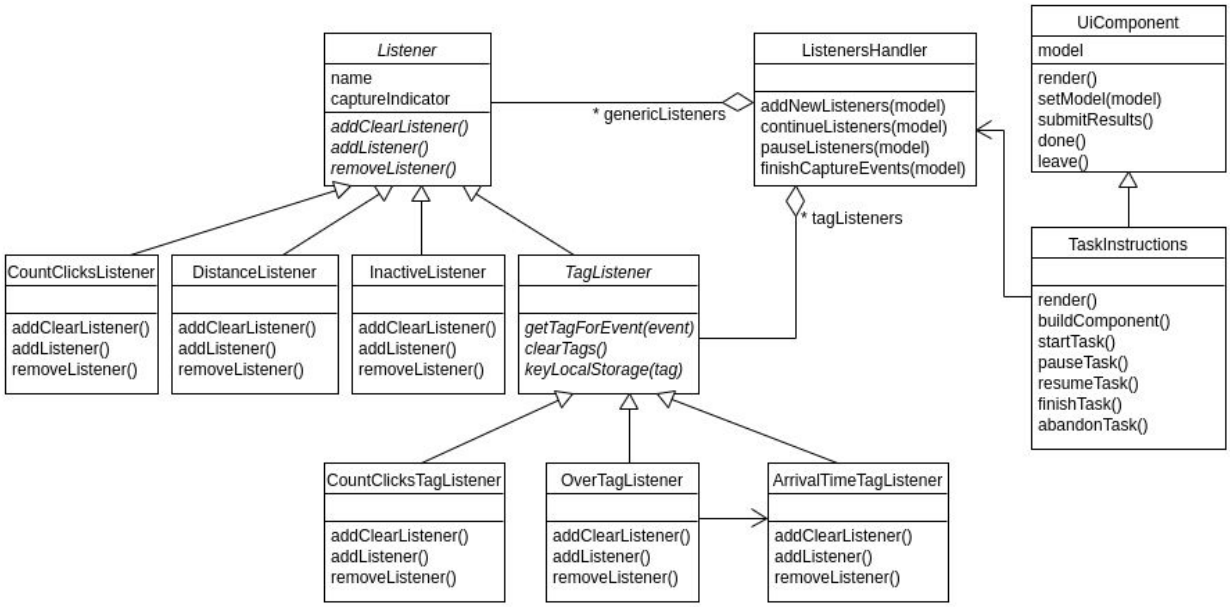


Figura 4.7. Modelo de clases de los listeners agregados para la captura de métricas. *TaskInstructions* no es la única tarea que hereda de *UiComponent*. El modelo fue simplificado a los efectos de explicar la utilización de los listeners

El manejo de los listeners se realiza a través de un manejador. En la figura 4.7 se puede observar el modelo generado para utilizar los listeners. La clase *TaskInstructions* conoce a un *ListenerHandler* mediante el cual va a tener acceso a los distintos listeners en particular ya sea agregando un nuevo listener (*addNewListeners*), pausandolos (*pauseListeners*), reanudandolos (continueListeners) o finalizandolos retornando los resultados capturados (*finishCaputreEvents*).

Cuando hablamos de tipos de listeners podemos encontrar dos grupos. Aquellos que extienden de la clase "*Listener*" ("*CountClicksListener*", "*DistanceListener*" e "*InactiveListener*") están orientados a la captura de métricas genéricas durante la ejecución del experimento sin importar el sitio que se encuentre visitando. Por otro lado están los que extienden de "*TagListener*" ("*CountClicksTagListener"*, "*OverTagListener*" y "*ArrivalTimeTagListener*"), estos se cargan en los elementos html identificados a través de selectores CSS (id o clase) en las páginas indicadas por el experto a la hora de diseñar el experimento.

Es necesario que la clase *ListenerHandler* conozca todos los posibles listeners que pueden ser incluidos en una tarea, de esta manera solo basta saber que listeners fueron los que habilitó el diseñador del experimento. Vale aclarar que los métodos que permiten agregar, pausar y finalizar los capturadores de eventos reciben el objeto "*model*". Este objeto contiene datos relacionados con la ejecución de la tarea, entre los que se incluyen los listeners que se van a utilizar, por lo tanto al llamar al manejador y pasarle el objeto "*model*" serán agregados únicamente los listeners que fueron habilitados por el diseñador y en el caso de los listeners para elementos particulares de un sitio serán habilitados cuando el voluntario visite la página en cuestión.

El mismo objeto "*model*" que tiene los datos necesarios para la ejecución de la tarea es utilizado para enviar los resultados una vez que la ejecución es finalizada. Es por esto que al finalizar la captura de eventos se llama al método "*finishCaptureEvents*" pasándole también el "*model*". De esta manera se recolectan los datos de los eventos capturados, se guardan en el "*model*" y es retornado a la tarea *TaskInstructions*.

Si bien las métricas son agregadas con el objetivo de dar una mejor utilidad a las pruebas de A/B testing los datos recolectados durante la ejecución también pueden ser habilitados durante ejecuciones que no pertenezcan a este tipo de prueba.

Manejador de almacenamiento local

Los eventos pueden capturarse para una misma tarea desde distintas ventanas o pestañas si el voluntario va alternando sobre ellas durante la ejecución de las tareas. Para que desde todas las instancias de ejecución de la tarea en simultáneo puedan centralizar los datos capturados se requiere acceder a esta información de manera centralizada. Para atacar esta problemática es utilizada la API de javascript "*localStorage*".

Como fue explicado previamente, mediante el "*localStorage*" es posible almacenar información accesible desde distintos componentes del *addOn*, pudiendo de esta manera guardar los datos recolectados de forma centralizada. Por otro lado, el módulo "*localStorage*" es uno de los que es accesible desde los *background scripts* y también desde los *content scripts*.

Para utilizar el *localStorage* de manera unificada y centralizada fue creada una clase llamada *BrowserStorageLocalHandler*, la misma cumple la función de simplificar y ser un punto de acceso único al módulo. Esta clase cumple con una interfaz simple con un método *set(key,value)* que permite guardar el objeto al estilo "clave: valor" en el *localStorage* y un método asincrónico *get(key)* que retorna el elemento con se encuentre en el almacenamiento local con la clave pasada por parámetro.

Al poder ser utilizada tanto por los background scripts como desde los content scripts, la clase *BrowserStorageLocalHandler* fue agregada como un componente más del modulo *shared* descripto en la sección 4.2.2.1.

4.2.3 Frontend

Cuando se hace referencia al frontend de Tycho se habla del portal web que permite realizar la administración de los experimentos y visualizar los resultados que arrojan las ejecuciones de los mismos.

El frontend se encuentra desarrollado en Smalltalk a través del framework Seaside[15]. Para el diseño y desarrollo de la web fue utilizado Bootstrap en su versión 3.1.1[16]. Como el soporte para realizar A/B testing es una extensión a Tycho se decidió mantener la misma versión de Bootstrap para mantener la compatibilidad con lo que ya se encontraba desarrollado.

Para la creación de gráficos en los resultados fue utilizada la librería *HighCharts*[17] mediante una adaptación para el framework Seaside llamada *HighChartsSt*[18] que permite utilizar dicha librería solo utilizando código Smalltalk.

[15] Seaside: http://www.seaside.st/
[16] Bootstrap 3.1.1: https://bootstrapdocs.com/v3.1.1/docs/getting-started/
[17] HighCharts: http://highcharts.com/
[18] *HighChartsSt:* https://github.com/ba-st/HighchartsSt

Capítulo 5
Aplicación Tycho

Con el objetivo de que el lector tenga un pantallazo general de Tycho y dado que no existe una documentación previa de la herramienta (incluso sin soporte para A/B Testing), en la presente sección se explica cómo funcionan los diferentes elementos que componen la herramienta, tanto desde la posición del experto en usabilidad para gestionar pruebas de usuario y visualizar resultados de las ejecuciones, como desde el lugar de los voluntarios que ejecutan las tareas. Vale aclarar que todo lo que es presentado en esta sección relacionado con la aplicación incluye las modificaciones implementadas para la presente tesina, como la habilitación de A/B Testing, la captura de métricas y la visualización de resultados, entre otras.

5.1 Diseño y administración de experimentos

Tycho cuenta con un portal que le permite a los interesados en realizar pruebas de usabilidad registrarse e iniciar sesión como cualquier aplicación web.

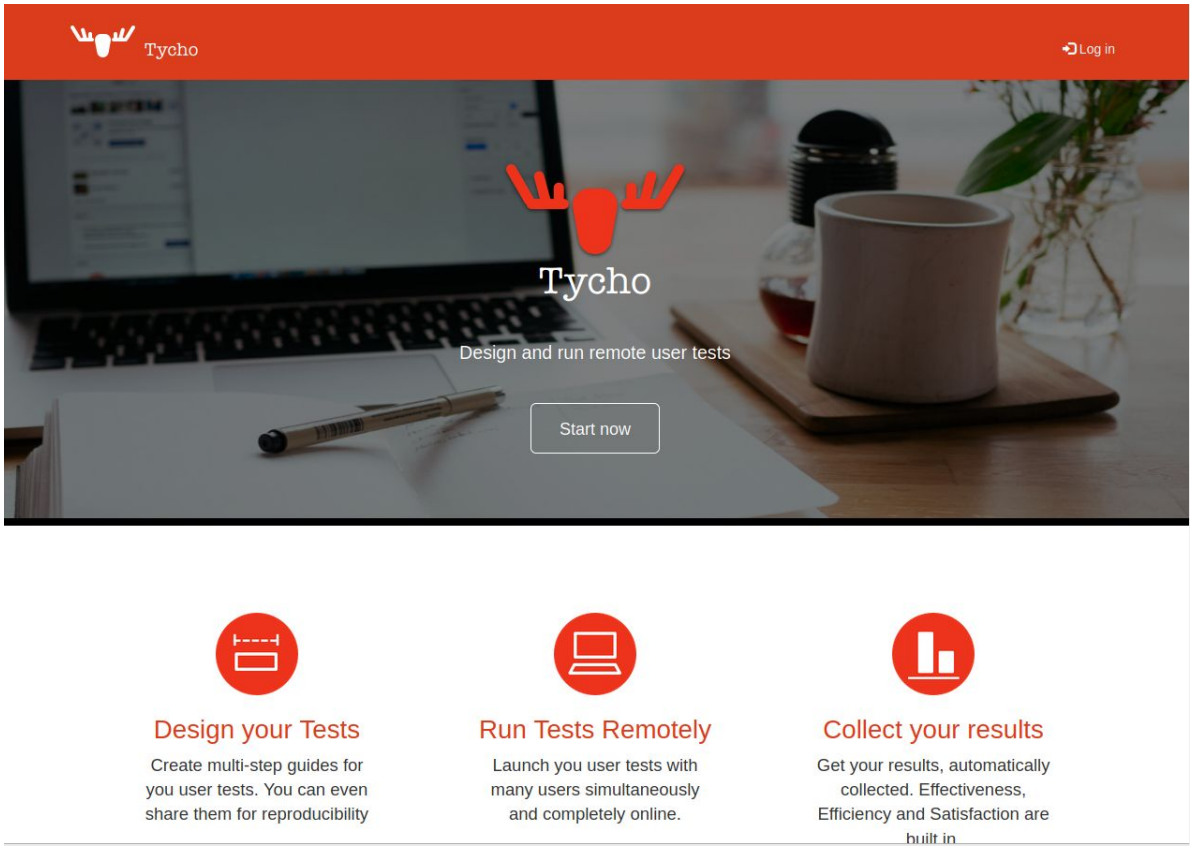


Figura 5.1. Pantalla principal que ve el experto previo a iniciar sesión en Tycho

Al registrarse por primera vez el experto se encuentra con dos secciones: La primera se esta relacionada con los experimentos, en ella se van a ver todas las sesiones de experimentos que el experto crea, la segunda sección es para los protocolos genéricos donde también se visualizarán todos aquellos que se vayan creando a medida que se haga uso del portal. En la figura 5.2 se puede observar una captura de dicha pantalla con las secciones mencionadas marcadas con rectángulos.

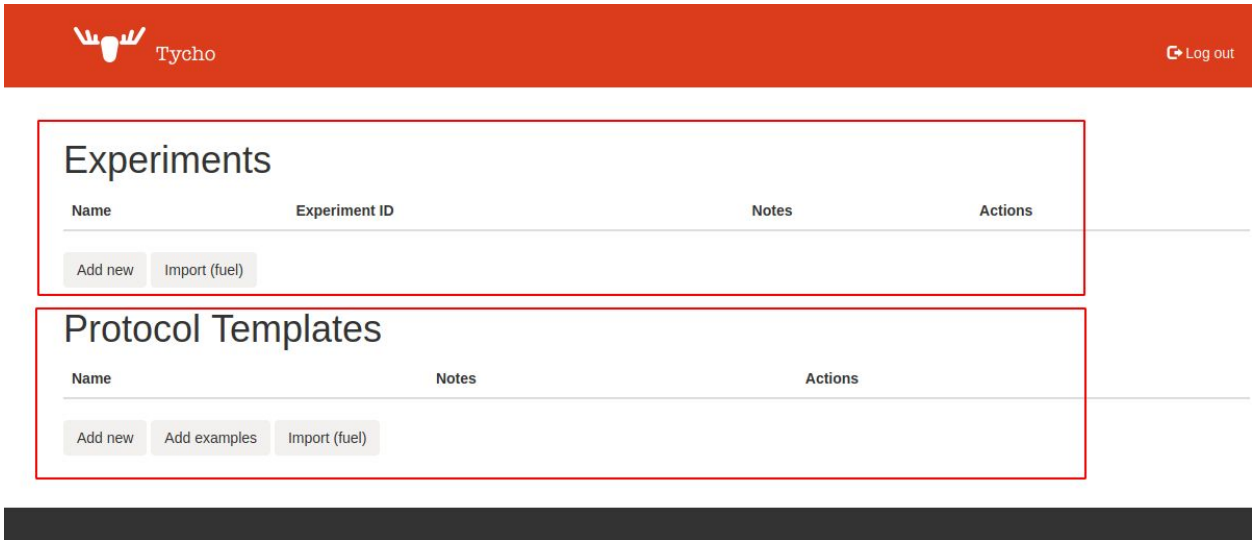


Figura 5.2. Primera pantalla que ve el experto al iniciar sesión por primera vez cuando aún no se creó ni experimentos ni protocolos.

A simple vista se identifican los botones para agregar un nuevo experimento o un nuevo protocolo a través de los botones con la leyenda "*Add new*" en cada sección.

Si se elige la opción para la creación de un protocolo genérico se abrirá una pantalla como la de la figura 5.3. Como se indicó en el capítulo 3.1.3 los protocolos genéricos sirven para realizar una secuencia de tareas que puedan utilizarse como base en otras sesiones de experimentos donde van a ser ejecutados. A la hora de definir un protocolo es posible definir el nombre del mismo, asociar algunas notas o descripciones y agregar distintos tipos de tareas permitiendo utilizar el método "*drag and drop*"[19] para definir el orden en el que se ejecutarán las tareas.

[19] Drag & Drop: Arrastrar y soltar. Es una técnica que permite llevar elementos en una aplicación de un lugar a otro intuitivamente para facilitar la interacción del usuario con el sistema.

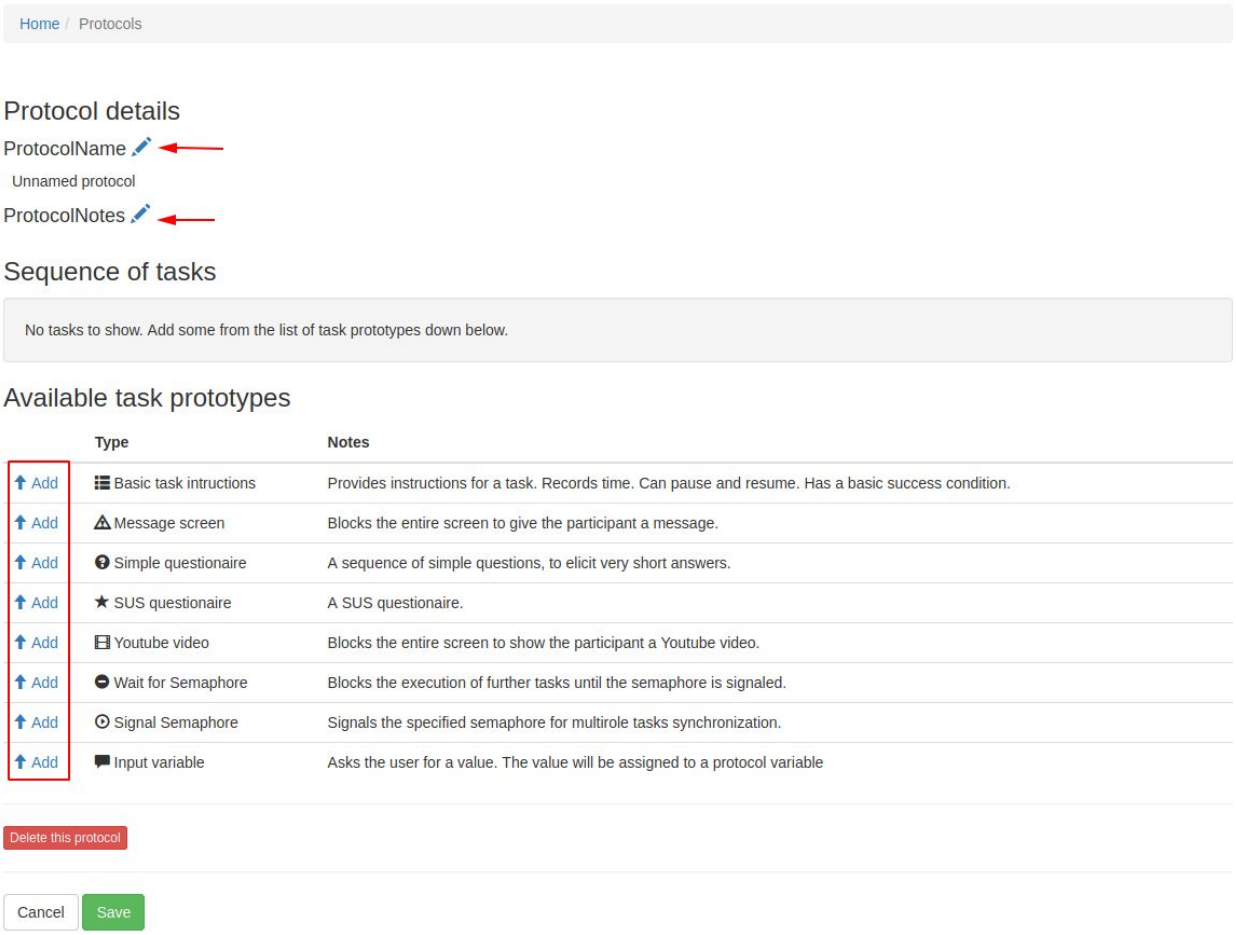


Figura 5.3. Pantalla básica de la creación de un protocolo sin realizar ninguna modificación hasta el momento.

En la figura 5.3 se puede ver los distintos tipos de tareas que se pueden agregar y que fueron descritas en la sección 3.1.1 donde se presentaron los conceptos de Tycho. No hay un límite ni en la cantidad de tareas ni en las veces que estas se repitan, ya que a través de ellas el experto intentará guiar al usuario y capturar métricas en el caso de que lo requiera. Como escenario de ejemplo de un protocolo base se creará un protocolo de tres tareas en el que se le presentará al voluntario un mensaje con información relacionado a la prueba ("*Message Screen*"), posteriormente se agrega una tarea del tipo "*Basic task instructions*" para que el voluntario realice las acciones indicadas y finalmente se realiza un cuestionario SUS ("*SUS questionaire*") con el objetivo de que el voluntario conteste preguntas sobre la usabilidad de la página sobre la cual debió realizar las acciones.

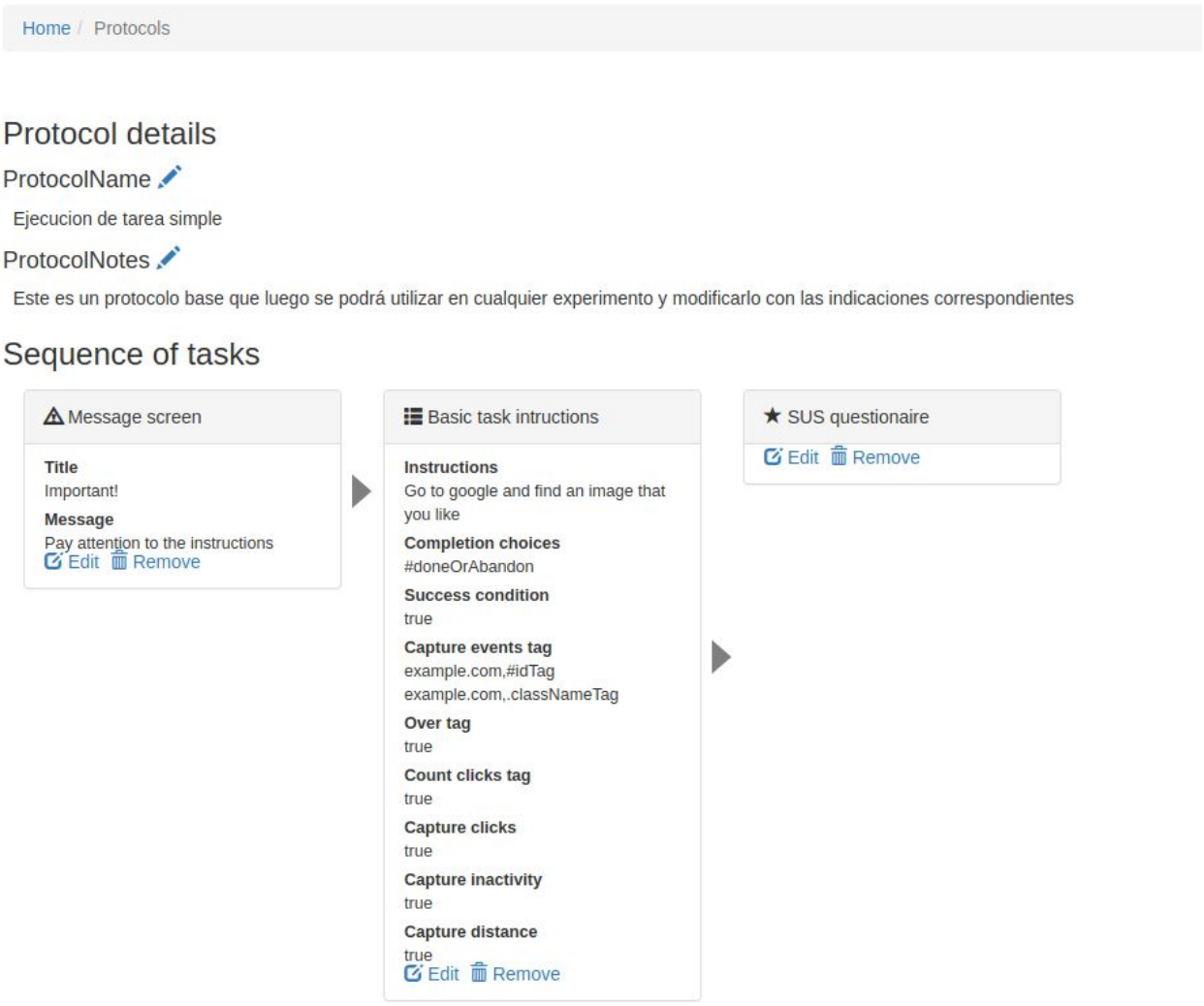


Figura 5.4. Protocolo genérico con tres tareas

En la figura 5.4 se puede ver cómo queda generada la secuencia de tareas. El objetivo de este protocolo genérico es poder seleccionarlo y editar lo que sea necesario para una sesión de experimento en particular, es por eso que las indicaciones dentro de cada tarea son generadas automáticamente por Tycho a modo de ejemplo cuando se agrega la tarea al protocolo.

Una vez que el protocolo genérico se guarda, el mismo se puede ver en la lista de "*Protocol Templates*" como se puede observar en la figura 5.5.

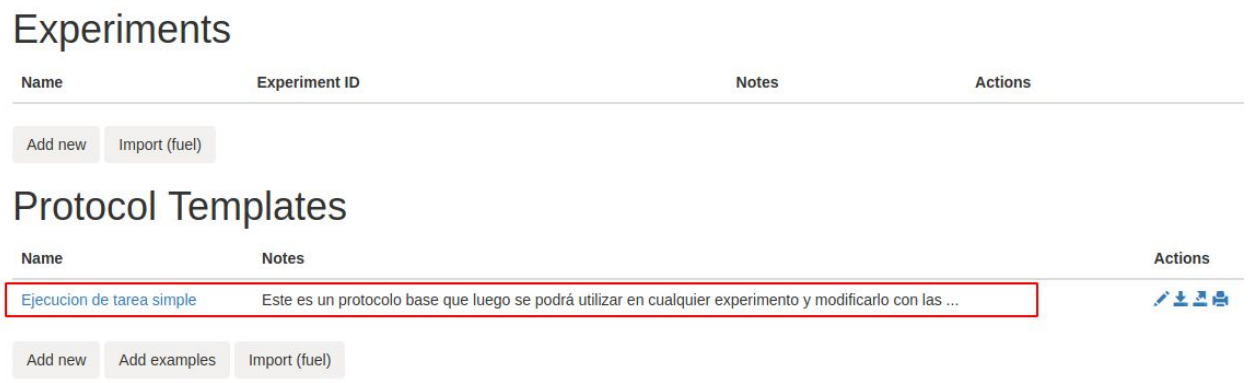


Figura 5.5. Pantalla principal con el protocolo genérico ya creado

El próximo paso es crear una sesión de experimentos. Para crearlo debemos presionar el botón "*Add new*" en la sección de experimentos. Al ingresar se puede observar que de la misma manera que con los protocolos, un experimento puede tener un nombre y notas asociadas, pero además cada sesión de experimento podrá tener asociado uno o más protocolos. En la figura 5.6 se puede ver cómo es posible agregar nuevos protocolos a través de dos botones:

- ➔ "*Add Empty*" agrega un protocolo totalmente vacío sin ninguna tarea ni nombre.
- ➔ "*Add selected*" agrega el protocolo genérico que seleccionamos del *select* que se encuentra desplegado a su izquierda. Se puede ver que figura el protocolo que creamos llamado "Ejecución de tarea simple" en el paso anterior.

Por último se marca una sección en donde se podrá ver un resumen de las ejecuciones que se van realizando de cada protocolo en una sesión de experimento.

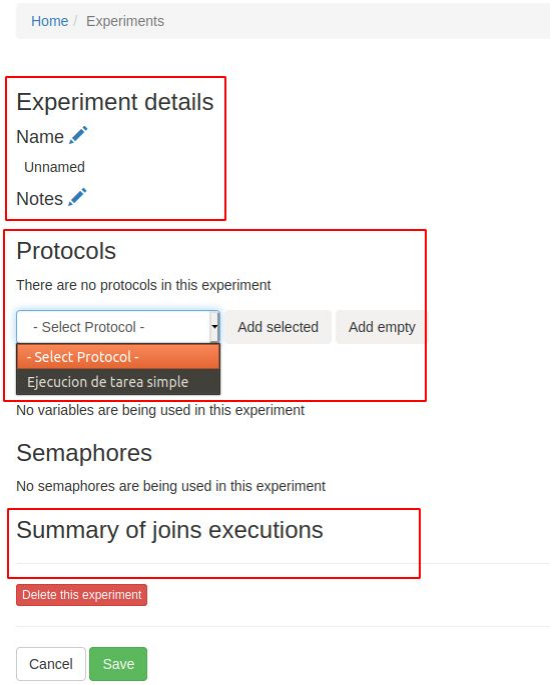


Figura 5.6. Creación de una sesión de experimentos

A modo de ejemplo se selecciona el protocolo genérico que fue creado (Figura 5.7). Al agregar el protocolo podemos observar que, como su nombre lo indica, se trata de una copia del protocolo genérico creado anteriormente. También se muestra el "*Access code*" asociado. El experto puede entregar este código a los voluntarios para que puedan ejecutar el protocolo. Por último, a la derecha nos encontramos con una serie de acciones que se pueden realizar sobre el protocolo, estas son (de izquierda a derecha): editarlo, clonarlo, eliminarlo y generar un reporte de sus tareas.

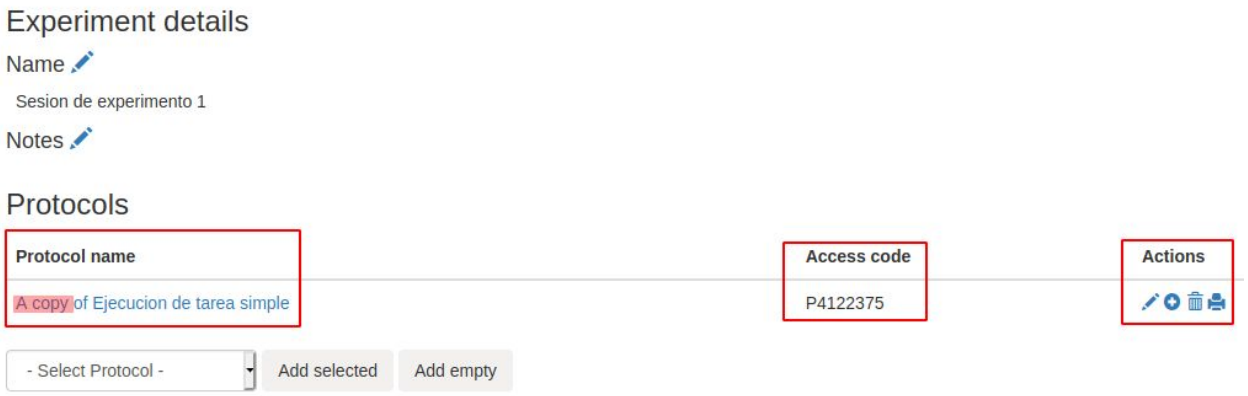


Figura 5.7. Creación de una sesión de experimentos agregando un protocolo genérico

Para que el usuario sea guiado correctamente, y con la granularidad que el experto desee, es necesario editar y cambiar las indicaciones de las tareas. Por ejemplo, en la tarea del estilo "Message Screen" se debe poner un mensaje de bienvenida, y en la tarea "Basic Task Instructions" indicar que es lo que debe hacer el usuario y qué métricas serán capturadas durante la ejecución. Para editar alguna de las tareas solo es necesario presionar sobre el botón "*Edit*" que se puede observar en la figura 5.4, ya que la pantalla que se visualizará será la misma que en esa figura. Las particularidades de cada tarea a la hora de su edición se encuentran en el apartado 5.2.

Soporte para A/B Testing

Mientras el experimento tenga un solo protocolo para ejecutar el mismo deberá ejecutarse a través del identificador del protocolo, pero al agregar un nuevo protocolo se habilita la opción de "*A/B Testing*", si esta es encendida es posible utilizar una distribución ponderada de los protocolos a través de un identificador del experimento. En la figura 5.8 se clona el protocolo anterior y se puede ver el botón para habilitar la opción mencionada.

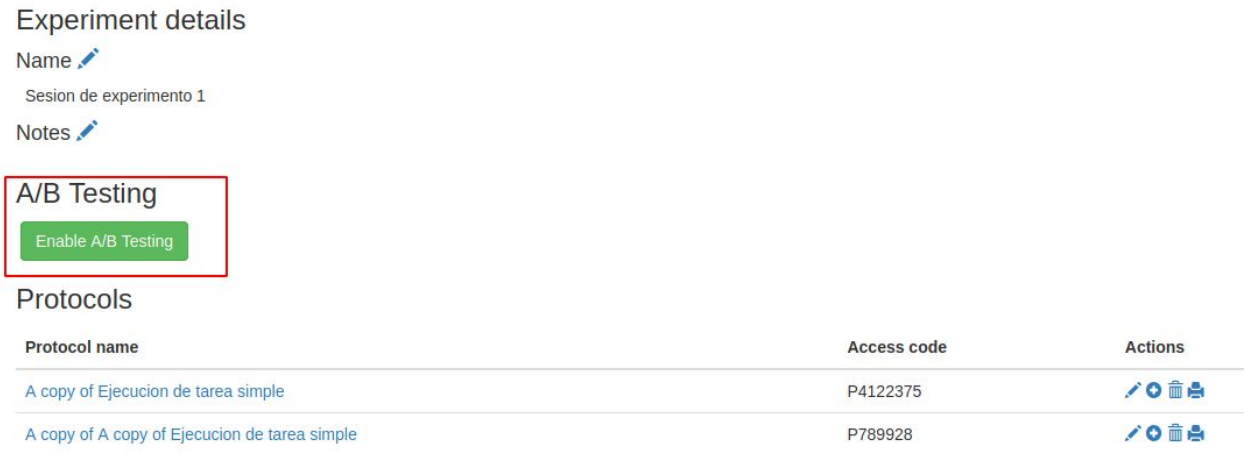


Figura 5.8. Sesión de experimento con dos protocolos y la posibilidad de habilitar A/B Testing

Al presionar el botón que habilita A/B testing, el experto se encontrará con otras opciones. En la figura 5.9 se visualiza (mediante recuadros) como se agrega nuevos datos y formularios en la sección de "*Experiment details*".

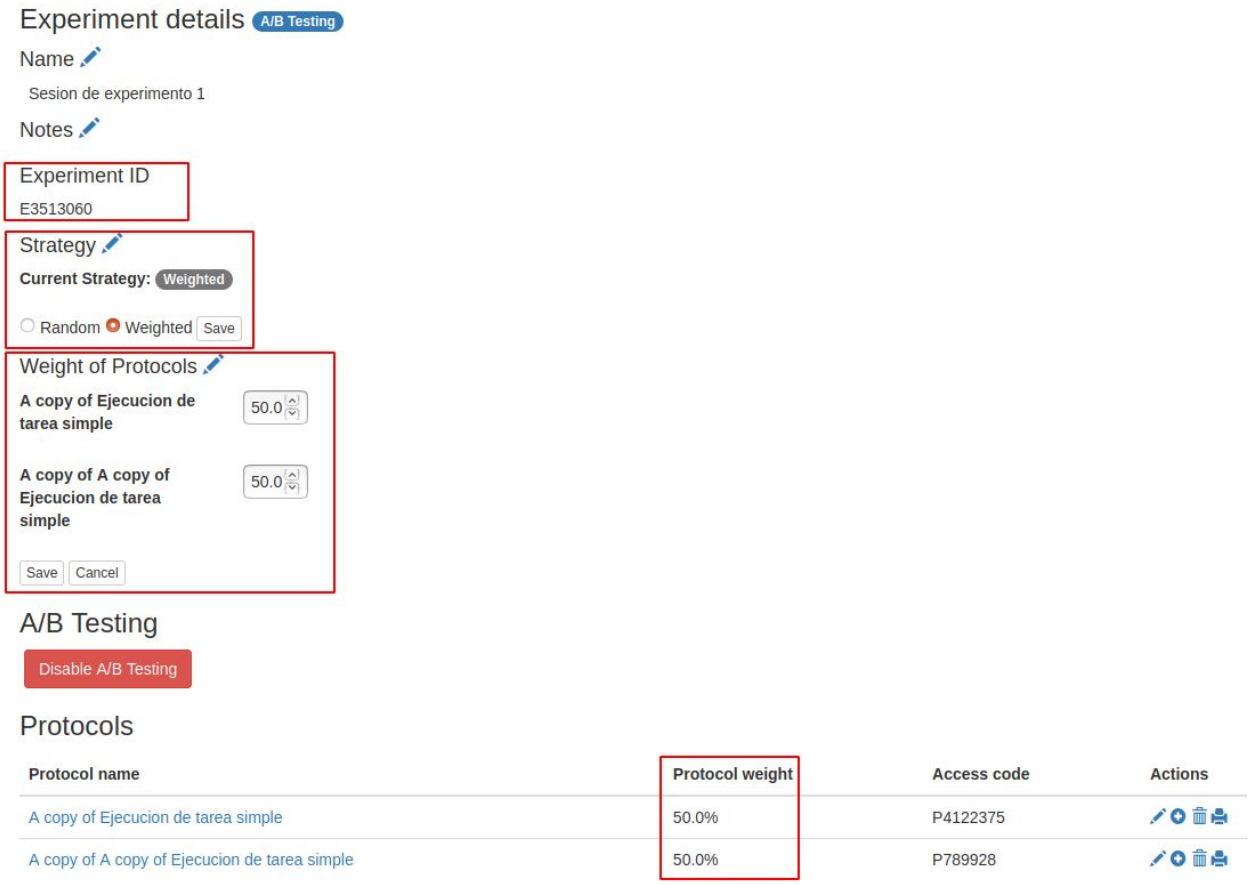


Figura 5.9. Sesión de experimento con dos protocolos y A/B testing habilitado

La primera particularidad que se puede observar es que se agrega una etiqueta al lado de "*Experiment details*" que indica que se habilitó A/B Testing. Además bajo la misma sección se agregó el identificador del experimento, el mismo podrá ser entregado a los voluntarios si el experto quiere que la distribución de cada protocolo sea realizada por alguna estrategia de distribución. Por debajo del id de experimento se encuentran las estrategias de distribución que se pueden seleccionar: una estrategia de distribución aleatoria "*Random*" y la otra estrategia de distribución ponderada "*Weighted*" en base a un valor asignado a cada protocolo.

La ponderación que se le asigna a cada protocolo para la segunda estrategia mencionada es posible realizarla en el último formulario de *Experiment details* llamado "*Weight of Protocols*", en el cual se podrá ver una entrada por cada protocolo donde el experto debe asignar un valor a cada uno del 0 al 100 debiéndose distribuir el 100% entre todos los protocolos disponibles. Cada vez que se habilita A/B Testing se distribuye el peso en partes iguales entre los diferentes protocolos, posteriormente el diseñador podrá modificarlos como desee. Todo lo mencionado anteriormente se encuentra detallado en la sección 4.2.1.

5.2 Edición de tareas

Cada tarea tiene sus particularidades y por lo tanto es posible editar diferentes elementos de cada una de ellas. Ya que el único campo que se repite en todas las tareas es el nombre de la misma, en esta sección se explica cómo se pueden personalizar las tarea dependiendo de su tipo.

Message screen

Este tipo de tarea es de uso múltiple y tiene como objetivo informar de una situación al usuario durante la ejecución del experimento. Puede utilizarse tanto al principio, por ejemplo un saludo de bienvenida, como al final para despedirse e indicar que finalizó la prueba o incluso en el medio de la ejecución para explicar algo en particular. Cuando el voluntario ejecuta este tipo de tarea se bloquea la pantalla del navegador y el usuario lo único que ve es el mensaje redactado por el experto.

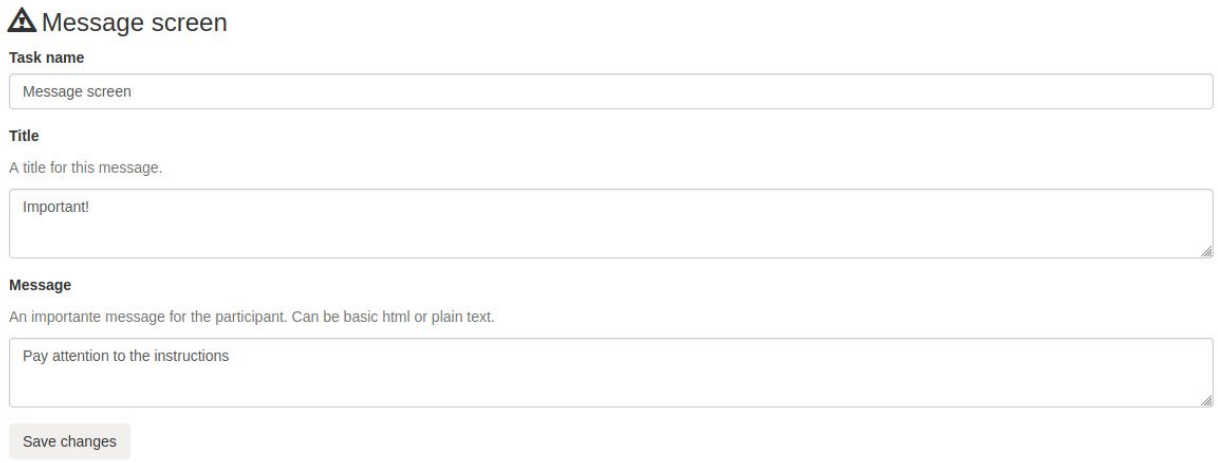


Figura 5.10. Pantalla de edición de una tarea del tipo "*Message screen*"

Entre los datos que se pueden editar es el título del mensaje que se querrá mostrar y el mensaje en cuestión.

Simple questionnaire

A través de este tipo de tarea se busca que el voluntario tenga un espacio de texto libre para responder a preguntas que el experto quiera realizar durante cualquier etapa de la ejecución del experimento.

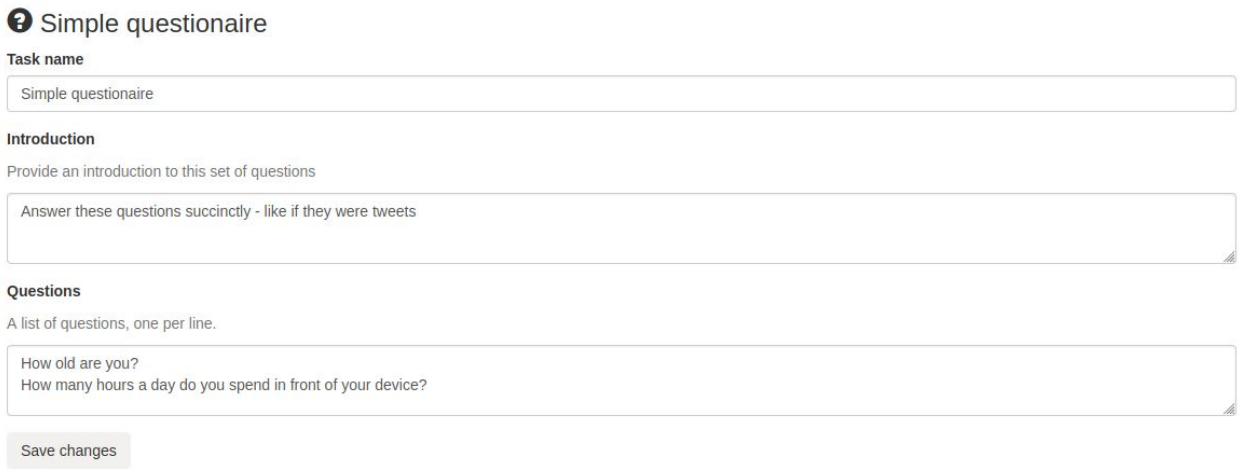


Figura 5.11. Pantalla de edición de una tarea del tipo "*Simple questionaire*"

Como se puede observar en la figura 5.11, la tarea cuenta con dos inputs además del nombre de la misma. Uno de ellos es "*Introduction*" que agrega un texto previo a las preguntas que deberá completar el voluntario. El segundo campo, "*Questions*", es donde el experto ingresará las preguntas y deben ingresar una por cada línea. La recomendación es que se realicen preguntas simples que orienten al voluntario a responder de forma concisa para que el procesamiento de las respuestas sea una tarea no tan compleja para el experto, de todas formas es solo una recomendación, no hay una limitación de este estilo en la aplicación.

SUS questionaire

Son cuestionarios orientados a la usabilidad y ya cuentan con una estructura y preguntas predefinidas, por lo cual no son editables. A partir de las respuesta que el voluntario ingresa se realiza un cálculo que indica el grado de satisfacción del voluntario con el sistema.

Youtube video

Con este tipo de tareas se bloquea la pantalla de la misma manera que con las tareas "*Message screen*" pero en lugar de mostrar solo un mensaje se despliega un video de youtube. Puede ser de utilidad para mostrar indicaciones de lo que quiera realizar previamente a la ejecución de una tarea o un video de finalización, entre otros.

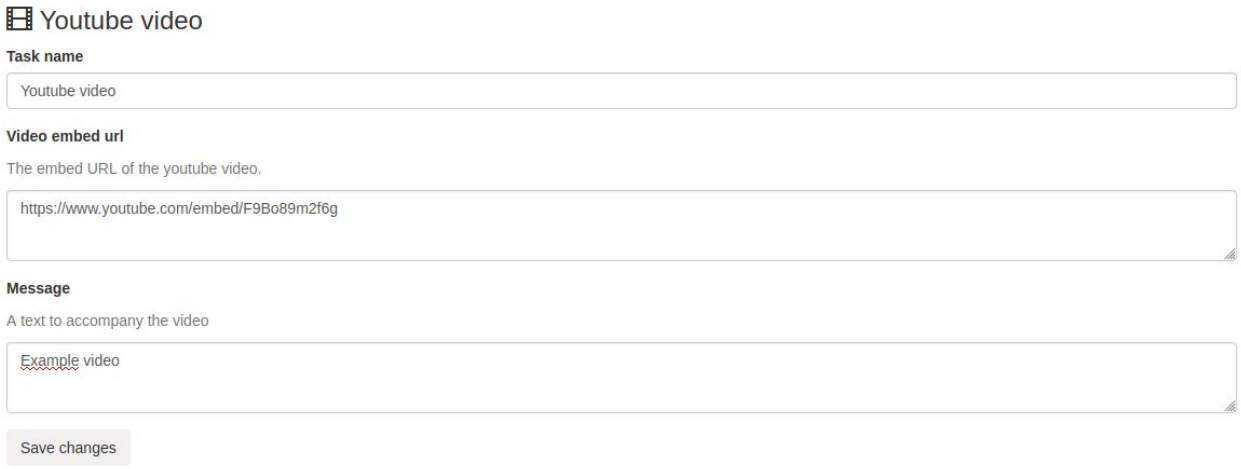


Figura 5.12. Pantalla de edición de una tarea del tipo "*Youtube video*"

Cuenta con dos inputs además del nombre de la tarea, el input "*Video embed url*" es para agregar la url al video de youtube que se desea mostrar. El input "*Message*" es para agregar algún mensaje adicional al video en cuestión.

Basic task instructions

Este tipo de tarea está orientada a que el usuario realice interacciones con alguna aplicación web en particular. Cuando llega la ejecución de esta tarea se bloquea la pantalla presentando las indicaciones que el experto realizó y finalmente cuando el voluntario indique que terminó de leer las indicaciones se libera la pantalla para que el mismo ejecute la tarea mencionada. En ese instante se comienzan a capturar las distintas métricas que se seleccionaron tanto las generales como aquellas que se agregaron sobre elementos html (si es que el usuario se encuentra en la web indicada).

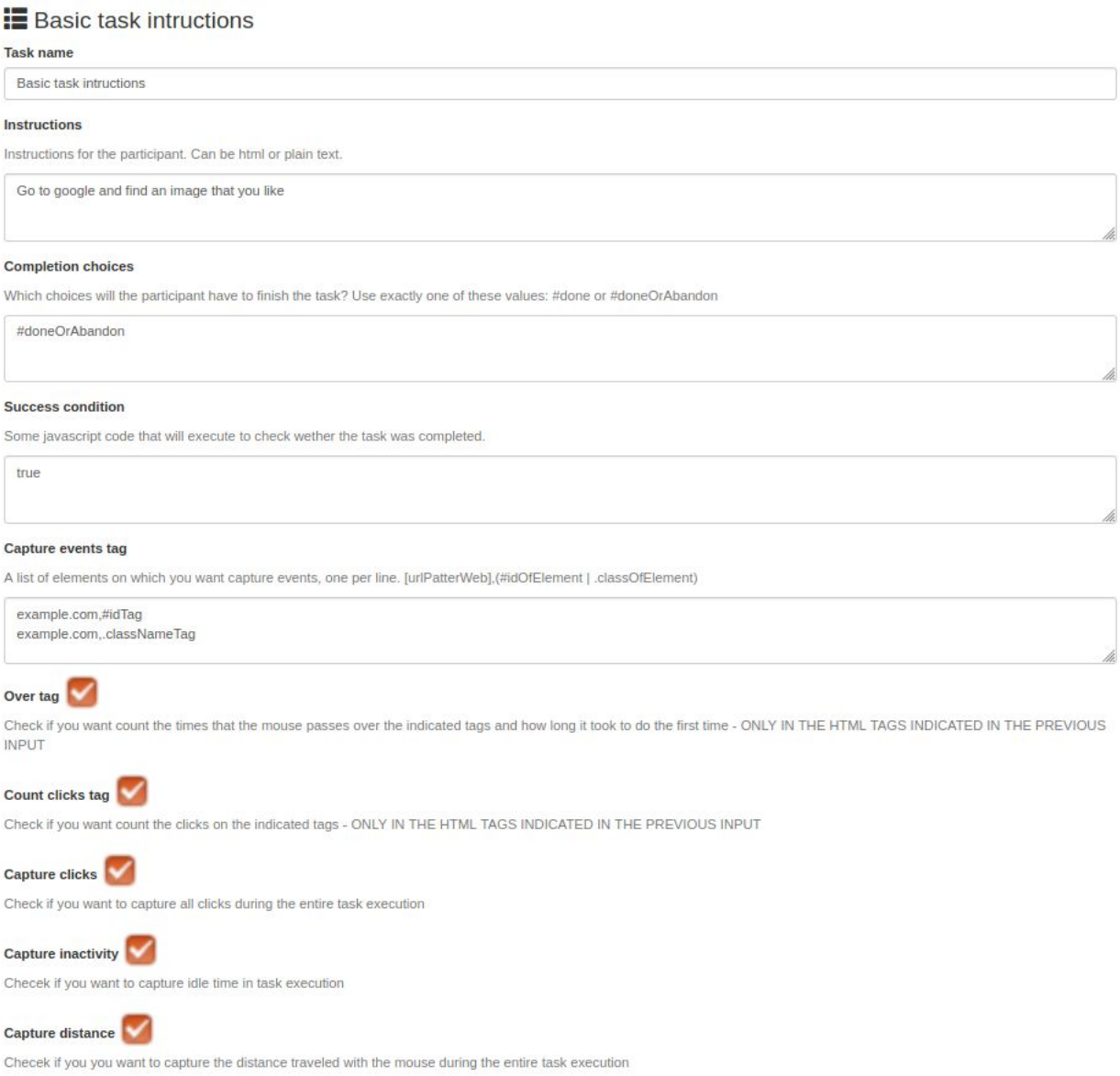


Figura 5.13. Pantalla de edición de una tarea del tipo "*Basic task instructions*"

Al editar una tarea de este estilo nos encontramos con varios inputs y algunos checkboxes, que se pueden visualizar en la figura 5.13. A continuación se repasan uno a uno.

- *Task name*: Permite definir un nombre a la tarea

- *Instructions*: Es el texto que se le presentará al usuario a modo de instrucción previa a la ejecución e incluso durante la ejecución de la tarea

- *Completions Choices*: Es el modo de concluir la tarea que tendrá el usuario. Si se ingresa *#doneOrAbandon* el voluntario podrá finalizar la tarea correctamente o abandonarla, si se

ingresa solo *#done* el usuario solo podrá completarla pero no tendrá la opción de abandonar.

- *Success condition*: es posible ingresar un código javascript que permita evaluar alguna condición en el cliente una vez que el usuario haya finalizado la ejecución de la tarea.

- *Capture events tag*: Es una lista de componentes html sobre los cuales se desean capturar métricas. Los componentes html deben definirse uno por línea y cada uno tendrá parte de la url donde se desea capturar eventos y el id o class del elemento html a través del selector css ("#" para los id o "." para las clases).
  El experto debe indicar parte de la url donde se deben capturar los eventos, una coma y el identificador del elemento en cuestión a través del css selector.
  Por ejemplo, si se quisiera capturar los eventos sobre el componente con *id="container"* en el sitio www.example.com se deberia ingresar "*example.com,#container*" (sin comillas). Si se quisiera capturar eventos sobre el elemento con el atributo *class="search-button"* en example.com se debe ingresar "*example.com,.search-button*" (sin comillas). En el caso de que se ingrese información errónea tanto en la url, como el tag o incluso en el formato en el que se escribe, el sistema descarta la entrada y no capturará datos.

- Los primeros dos checkboxes sirven para habilitar captura de datos sobre estos elementos:
  - *Over tag*: captura y contabiliza las veces que el voluntario pasa por encima del elemento html indicado
  - *Count click tags*: Contabiliza la cantidad de clicks que el voluntario realiza sobre el elemento seleccionado

- Los últimos tres checkboxes habilitan capturas genéricas que se realizarán a lo largo de la ejecución de la tarea:
  - *Capture clicks*: Captura todos los clicks que el voluntario realiza durante la tarea
  - *Capture inactivity*: Captura el tiempo que el voluntario se encuentra inactivo durante la ejecución de la tarea.
  - *Capture distance*: Captura la distancia recorrida con el mouse por parte del voluntario.

5.3 Ejecución de los experimentos

Una vez que el experto finaliza el diseño del experimento, debe entregarle a los voluntarios el identificador con el cual debe unirse al experimento. El mismo puede ser del protocolo que se quiera ejecutar o del experimento si es que la sesión cuenta con más de un protocolo y el diseñador habilitó la opción de A/B Testing.

El voluntario deberá descargarse e instalar un complemento en el navegador donde realizará la prueba. Una vez instalado, cuando el voluntario quiera ejecutar las pruebas deberá encender la extensión e ingresar el identificador que le envió el experto para la ejecución de la misma como se ve en la siguiente figura (5.14).

Figura 5.14. Ejecución de un experimento por parte del voluntario

Una vez que el voluntario ingresa el identificador y presiona el botón con la leyenda "*Unirse*", comenzará la prueba diagramada. Cada tarea tiene una forma de visualizarse, a continuación se presentan como se visualizan para el voluntario cada tipo de tarea al ser ejecutada.

En la figura 5.15 se puede observar que el experimento utilizado a modo de prueba tiene una primera tarea del tipo "*Message screen*" con un mensaje simple.

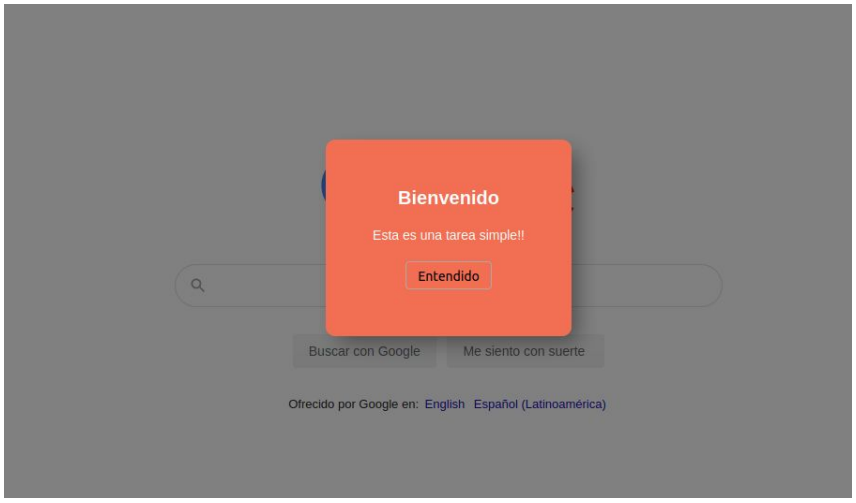


Figura 5.15. Ejecución de la tarea del tipo "*Message screen*" con un mensaje de bienvenida

Las tareas del tipo "*Basic Task Instructions*" cuentan con dos pasos: El primero es la presentación de las indicaciones como se puede observar en la figura 5.16.

Figura 5.16. Primer paso de la ejecución de una tarea "*Basic Task Instructions*"

El segundo paso es cuando el voluntario ya leyó las instrucciones y presiona en el botón "*Comenzar*". Al realizar esta acción el voluntario verá las indicaciones en una solapa inferior del navegador y permitirá realizar las acciones que le indicó el experto mientras se capturan las métricas cargadas como se ve en la siguiente figura (5.17).

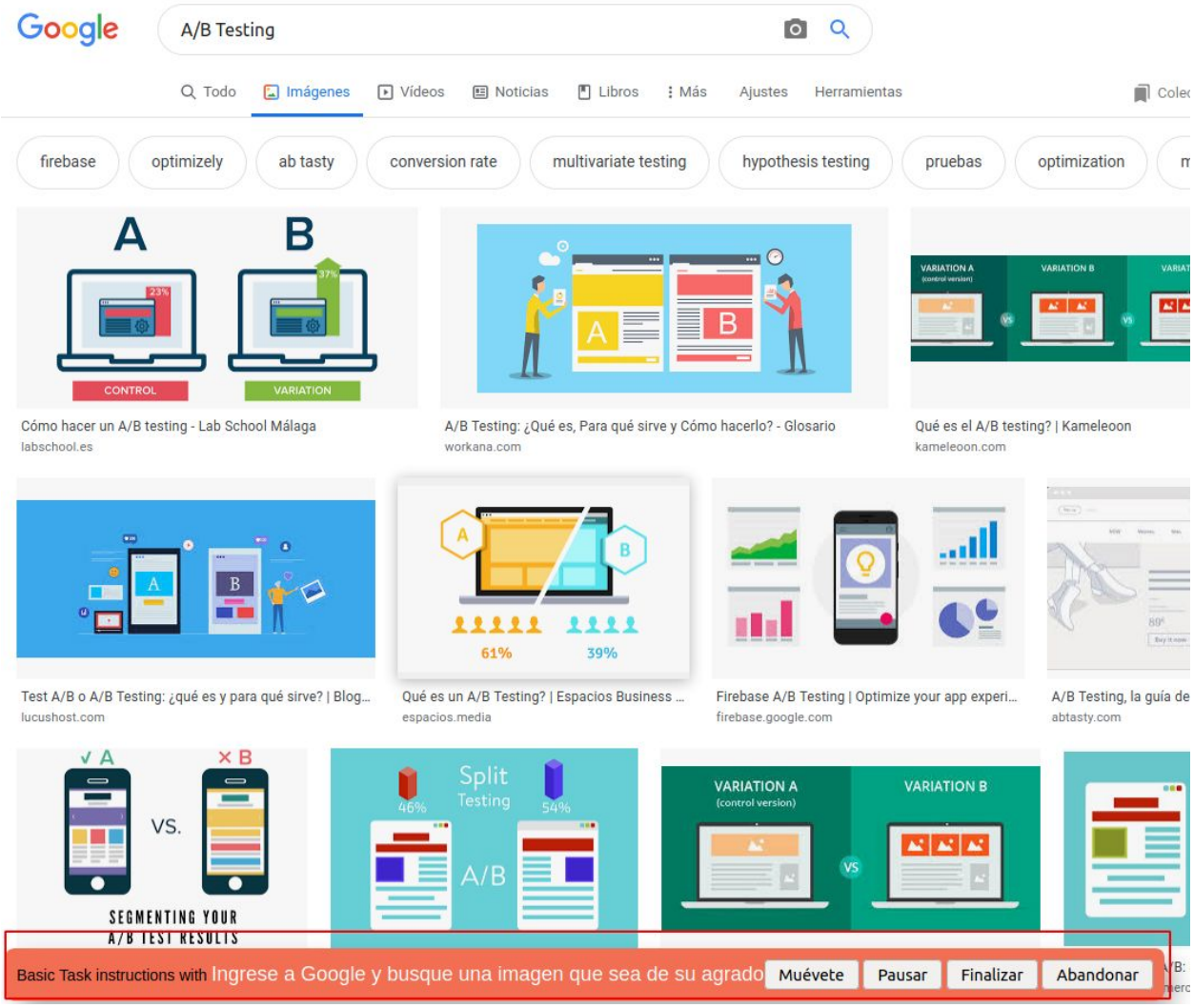


Figura 5.17. Segundo paso de la ejecución de una tarea "*Basic Task Instructions*"

En la figura 5.17 se puede observar que en la solapa inferior aparecen los botones que le permiten pausar, finalizar o abandonar la tarea que está ejecutando. Además se le permite al usuario mover el panel a otro sector de la pantalla a través del botón "*Muévete*".

Para las tareas del estilo *SUS Questionnaire* (Figura 5.18) se despliega un formulario con diez afirmaciones genéricas relacionadas a la usabilidad de un sistema. Generalmente estos formularios son utilizados al final de la ejecución de las pruebas para saber la opinión del voluntario con respecto a la aplicación que probó. El voluntario debe completar estas afirmaciones en un rango de 1 a 5 donde 1 es "*muy en desacuerdo*" y 5 es "*muy de acuerdo*".

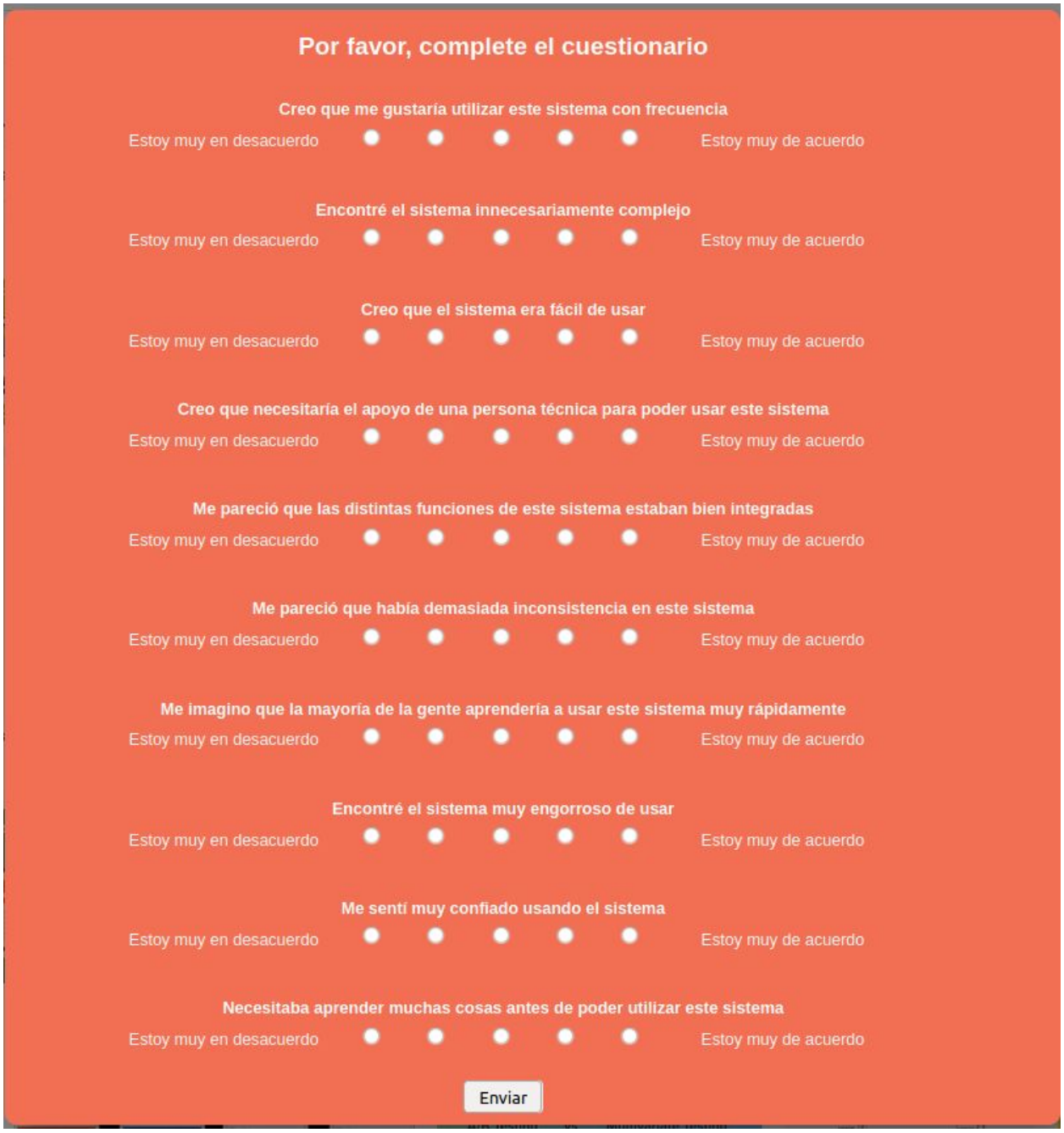


Figura 5.18. Tarea de tipo "*Sus Questionaire*" en ejecución.

Cuando el voluntario ejecute tareas del tipo "*Simple Questionaire*" (Figura 5.19) se desplegará una pantalla con las preguntas que el experto realizó y un campo de texto para que pueda responder a las mismas.

Figura 5.19. Tarea de tipo "*Simple Questionaire*" en ejecución.

Por último al ejecutarse una tarea del tipo "*Youtube video*" el voluntario se encontrará con una pantalla como la de la figura 5.20, en la cual se encuentra el video embebido y un botón para que pueda indicar que finalizó de ver el video.

Figura 5.20. Tarea de tipo "*Youtube video*" en ejecución.

5.4 Visualización de resultados

A medida que las ejecuciones de los experimentos se van realizando, el experto puede ver en la pantalla de edición del mismo un resumen de cuantas ejecuciones se realizaron de los protocolos que componen una sesión de experimento. Además se habilita una opción para ver en detalle el resultado de las ejecuciones "*View Results*" (Figura 5.21).

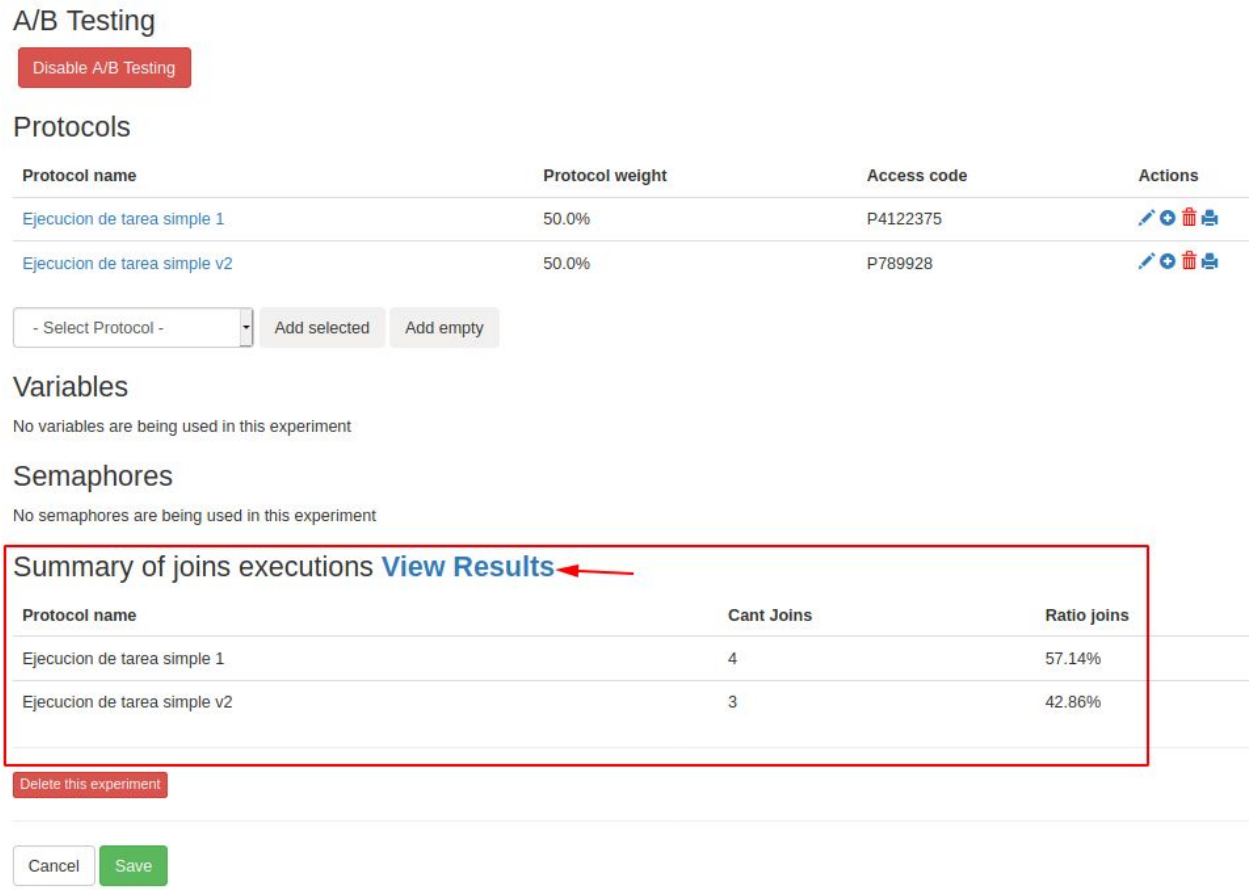


Figura 5.21. Pantalla de edición de un experimento donde ya se realizaron ejecuciones. Aparece un nuevo link para ver los resultados en detalle llamado "*View Results*"

Cuando se ingresa a la sección donde se encuentran los resultados podemos observar un resumen genérico de las distintas ejecuciones realizadas en las que se verá un gráfico de torta con la cantidad de ejecuciones por cada protocolo asociado y el tiempo promedio que estas le llevaron a los voluntarios (Figura 5.22).

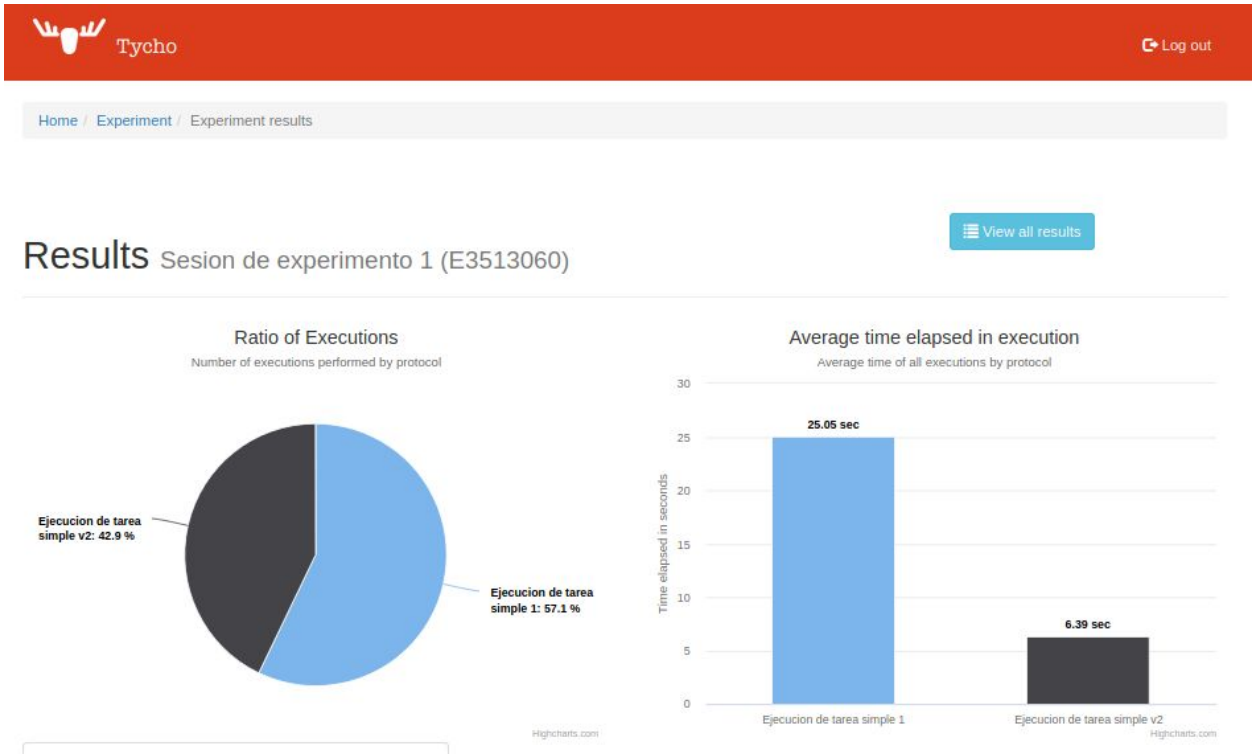


Figura 5.22. Resultado genérico de las ejecuciones en las que se muestra la cantidad de estas por cada protocolo y el tiempo promedio que llevó cada una.

Un poco más abajo se verán los resultados de cada tarea en particular de los protocolos en distintas solapas o pestañas como se muestra marcadas con flechas en la figura 5.23. Cada solapa es un protocolo y dentro de cada una se encontrarán los resultados de las tareas relacionadas a cada uno de ellos. El experto podrá ir cambiando de solapa para comparar los resultados de cada uno de los protocolos.

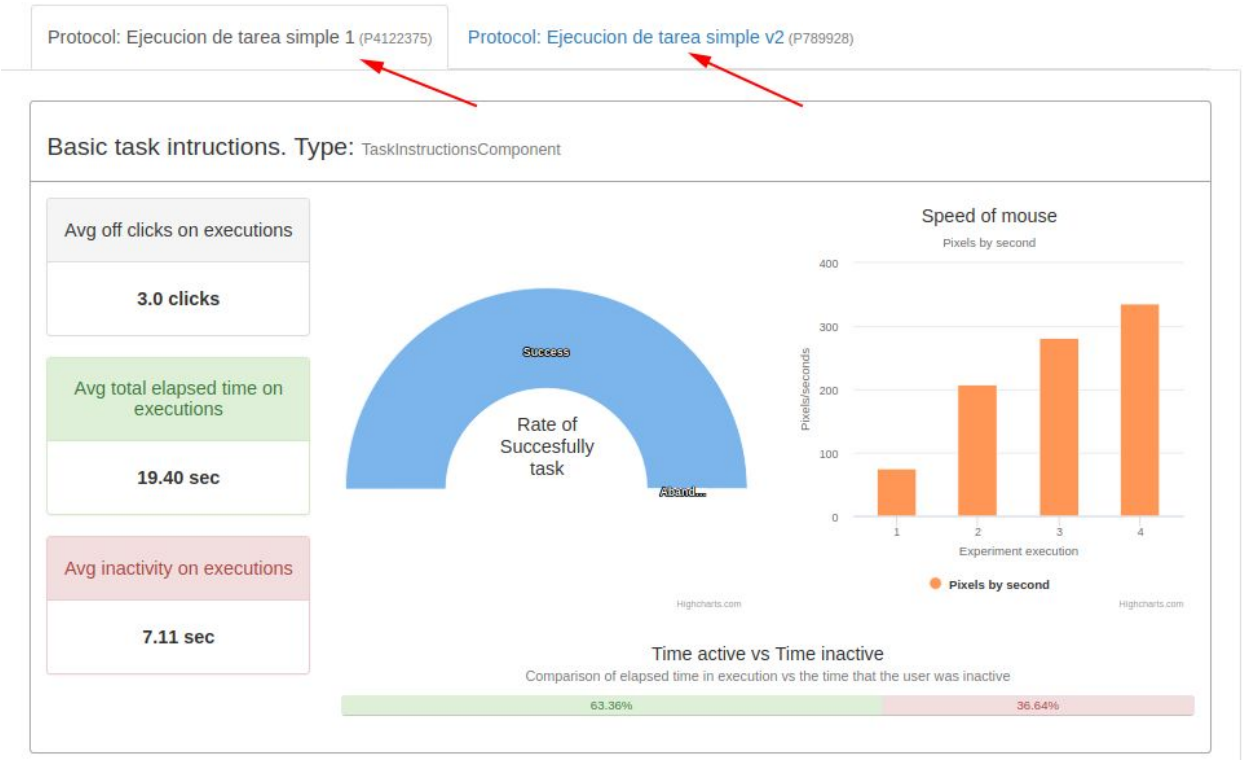


Figura 5.23. Resultados de las tareas de cada protocolo donde se pueden ver las pestañas. La tarea visualizada es del tipo "*Basic Task instructions*"

Cada tarea captura métricas diferentes y tienen formas distintas de mostrar la información relacionada a ellas. En la figura 5.23, además de ver las pestañas por cada protocolo, se pueden ver los resultados de las ejecuciones de las tareas del tipo "*Basic task instructions*". En la izquierda se ubican resultados promedios de cantidad de clicks realizados, del tiempo que se tardó en ejecutar estas tareas y del el tiempo promedio que los usuarios estuvieron inactivos. Con respecto a la inactividad, por debajo se puede ver una comparación del tiempo de actividad con el tiempo de inactividad en una barra. Los valores mencionados son promedios de las ejecuciones realizadas de esta tarea en particular del protocolo visualizado.

En el centro se ve un gráfico en el que se verá la cantidad de finalizaciones "correctas" comparadas con el porcentaje de usuarios que abandonaron la ejecución de estas tareas.

Por último, a la derecha se ve un histograma de la velocidad del mouse que emplearon los usuarios al completar la tarea.

Cuando a las tareas "*Basic task instructions*" se le agrega la captura de métricas sobre elementos html los mismos se visualizan en paneles desplegables por debajo de la información mencionada. Los mismos se verán en los casos de prueba del capítulo 6.

Las tareas *SUS Questionnair*e muestran otros datos. En la figura 5.24 se puede ver un gráfico de torta en el que se agrupan en base a la clasificación de los cuestionarios SUS. Más abajo se pueden ver las preguntas y unos gráficos de barra asociadas a cada una de ellas en las que se agrupan los tipos de respuesta que se dieron. Por último a la derecha se despliega un histograma con la información del tiempo que tardaron los voluntarios en completar el cuestionario.

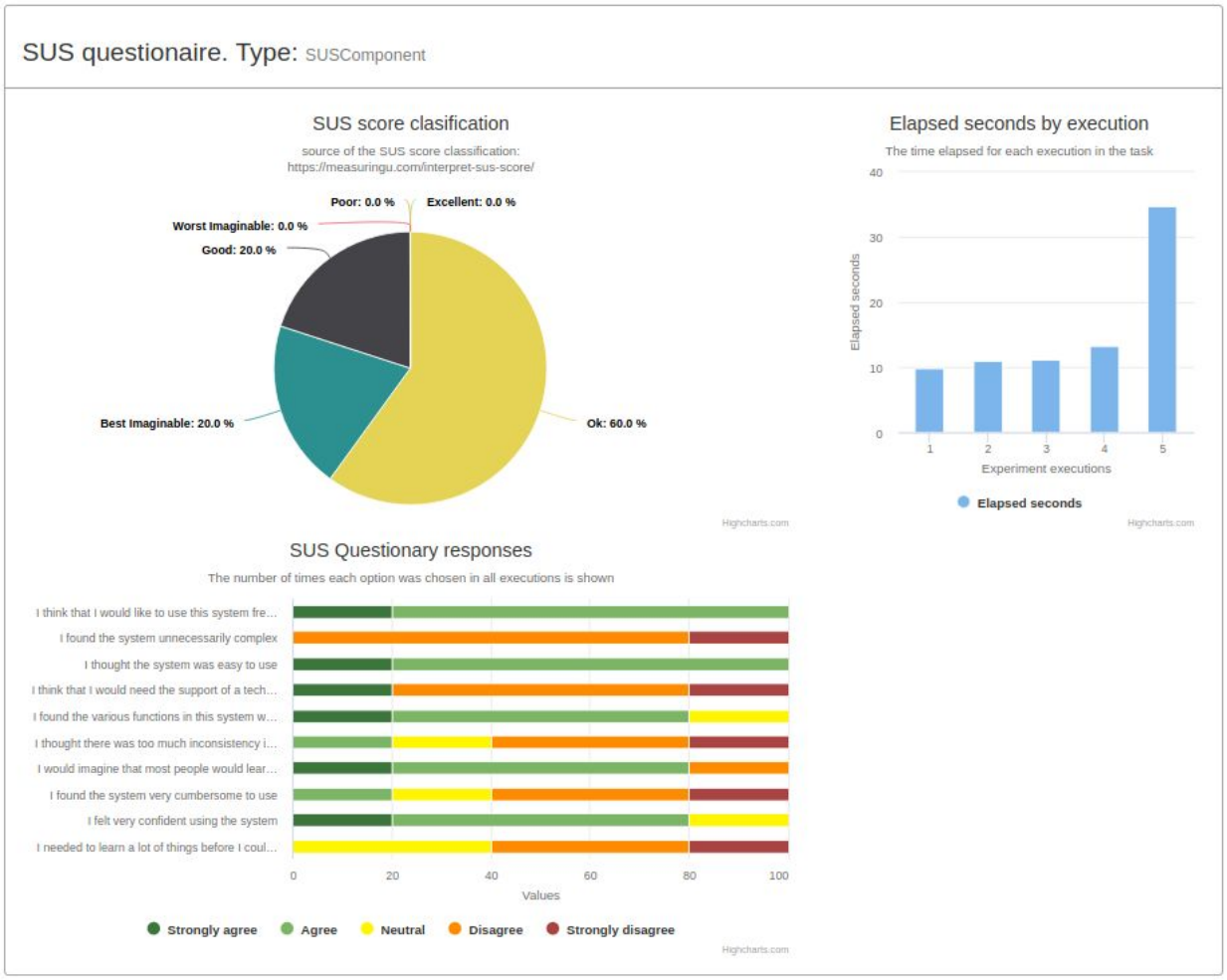


Figura 5.24. Resultados de ejecuciones de una tarea *SUS questionnaire*

Cuando se trata de tareas *Simple questionnaire* lo que se puede observar a modo de resultados son las respuestas a cada una de las preguntas y el tiempo que demoraron los usuarios en responderlas (Figura 5.25).

Figura 5.25. Resultados de ejecuciones de una tarea "*Simple questionaire*"

Las demás tareas como *Youtube Video* o *Message Screen* no se presentan en esta sección porque no capturan información adicional al tiempo que tardó cada voluntario en ejecutar las tareas.

Capítulo 6
Caso de Prueba

En este capítulo se presentan dos casos de prueba ejecutados por distintos voluntarios con el objetivo de demostrar cómo es posible realizar la comparación de los datos capturados en las ejecuciones y como el experto puede evaluarlas para sacar distintas conclusiones. Además se muestran los pasos que debe realizar el experto para diseñar y distribuir las pruebas de usuario así como también las evaluaciones previas que el mismo debe realizar.

6.1 Caso de prueba Nº 1: Simulación de sistema de registro

En este primer caso de prueba se simula una sección de una aplicación web donde el usuario debe registrarse para pagar un servicio o realizar una compra ingresando sus datos personales, datos de la cuenta que va a crear y datos de la tarjeta de crédito. Para realizar esta acción se dispone de dos diseños diferentes para completar el registro. El primer diseño se presenta en la figura 6.1 en la cual se puede observar que es un simple formulario en el que se solicitan los datos uno después del otro. En el segundo diseño, que se visualiza en la figura 6.2, el usuario debe cargar los mismos datos pero a través de un wizard: en primer lugar se solicita la información de la cuenta, en segundo lugar se pide que el usuario ingrese sus datos personales y en el tercer paso los datos de la tarjeta.

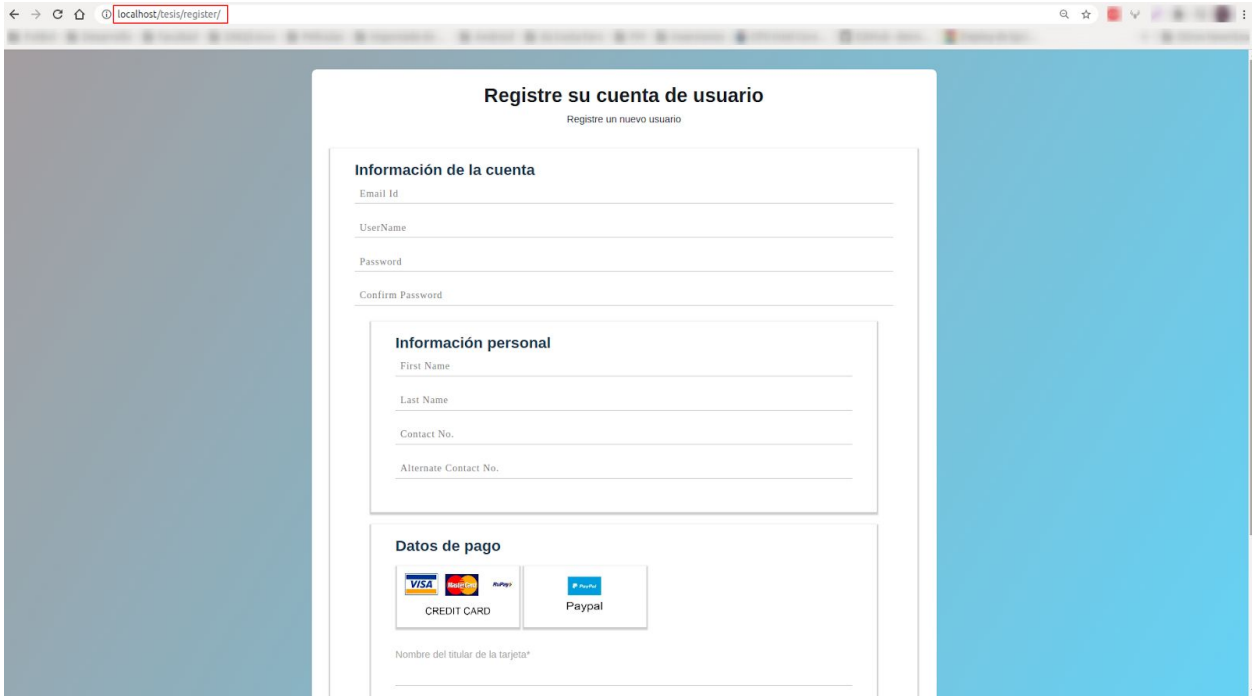


Figura 6.1. Formulario de registro en el que se solicitan todos los campos. La url es *localhost/tesis/register*

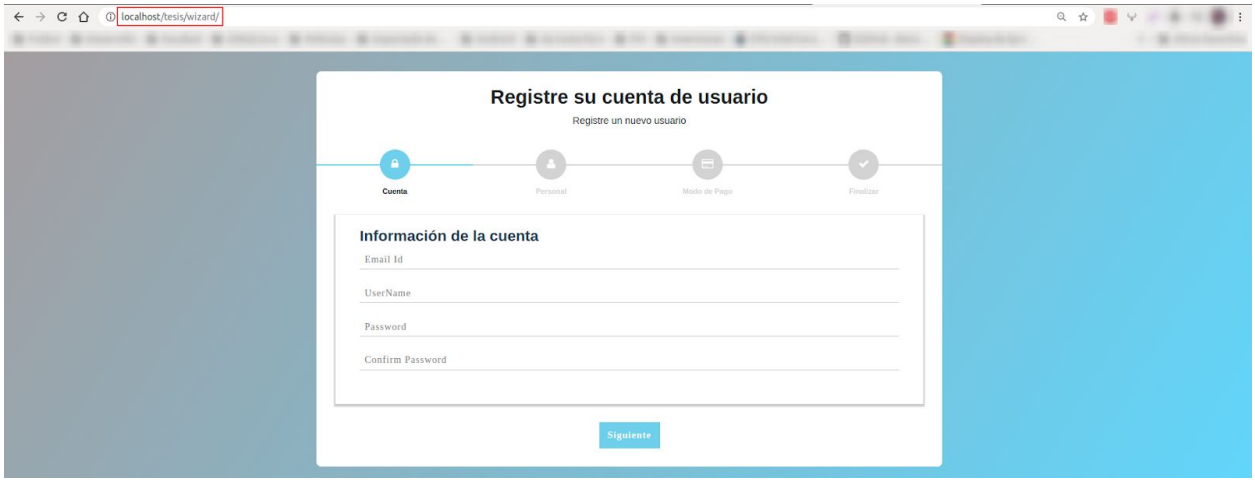


Figura 6.2. Formulario de registro en el que se van solicitando los campos a través de un wizard. La url es *localhost/tesis/wizard*

Tener el mismo formulario con modificaciones en su presentación es un caso típico de pruebas A/B testing en las que se quiere detectar qué alternativa puede ser más amigable con el usuario final.

En las figuras 6.1 y 6.2 queda en evidencia que la url de cada uno de los formularios es diferente, por lo tanto, es posible crear dos protocolos dentro de una sesión de experimento. Los protocolos

diseñados cuentan con dos tareas, la primera es un mensaje de bienvenida a través de una tarea "*Message Screen*" y la segunda es una tarea del tipo "*Basic task instructions*" en la cual se indicará al usuario que es lo que debe hacer. En la figura 6.3 se muestra como es la definición de la segunda tarea. En esta tarea se capturarán todas las métricas genéricas (Clicks realizados, tiempo de inactividad y distancia recorrida por el mouse en pixeles) y además se capturan eventos sobre el botón de confirmación cuando el usuario finalice de completar el formulario. El botón html asociado a la acción de confirmación tiene asociado un *id="confirm"*.

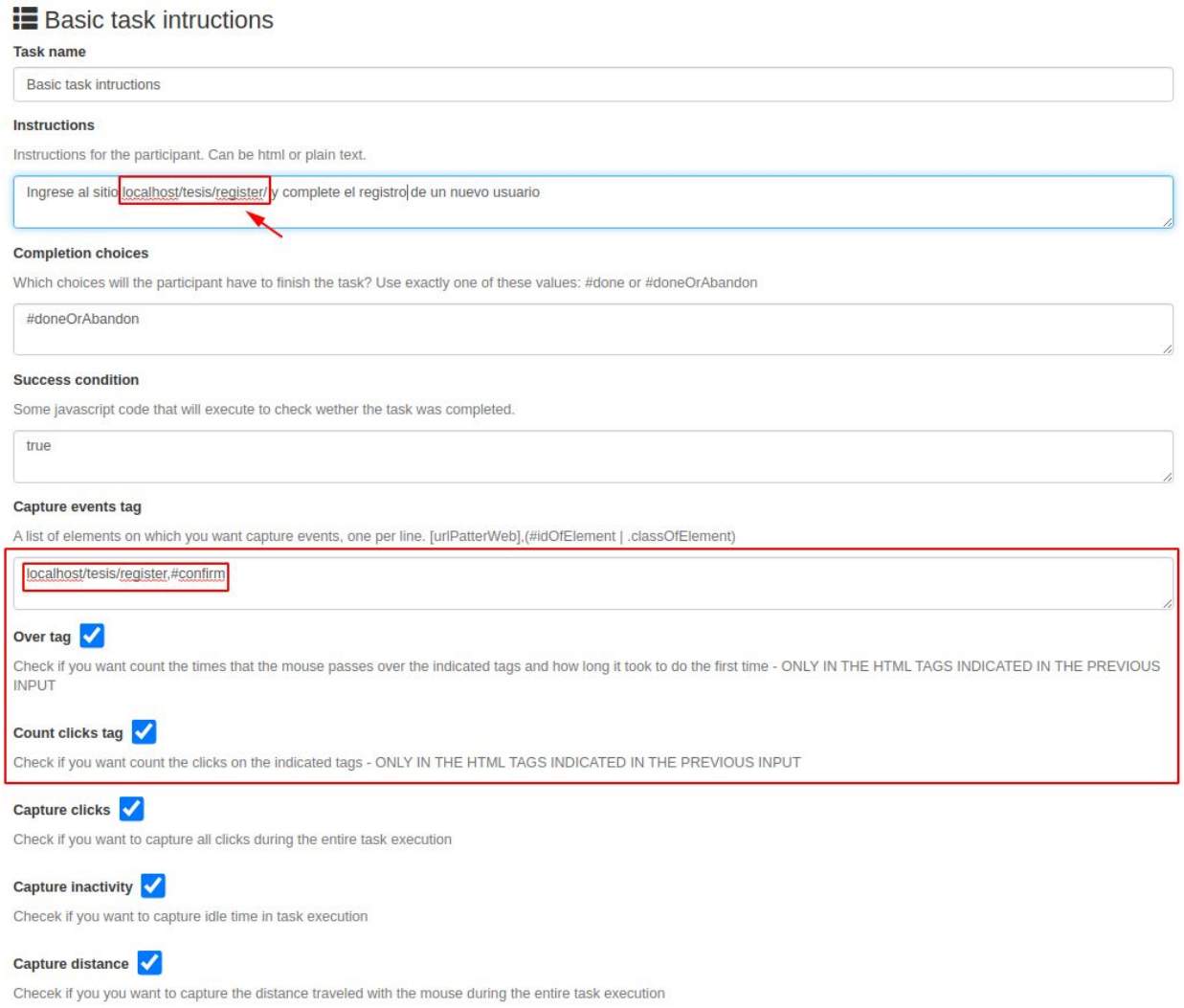


Figura 6.3. Definición de la tarea que captura eventos sobre el sitio *localhost/tesis/register*

En la figura 6.3 solo se está mostrando la definición del protocolo que guiará al voluntario hacia el formulario de registro que se visualiza en la figura 6.1. El otro protocolo es idéntico con la salvedad de que dirigirá al voluntario al formulario mostrando en la figura 6.2.

Luego de definir ambos protocolos se habilita la opción de realizar A/B testing con la estrategia "*Weighted"* y con los protocolos ponderados en partes iguales. En la figura 6.4 se puede observar como se visualiza el detalle de la sesión de experimentos.

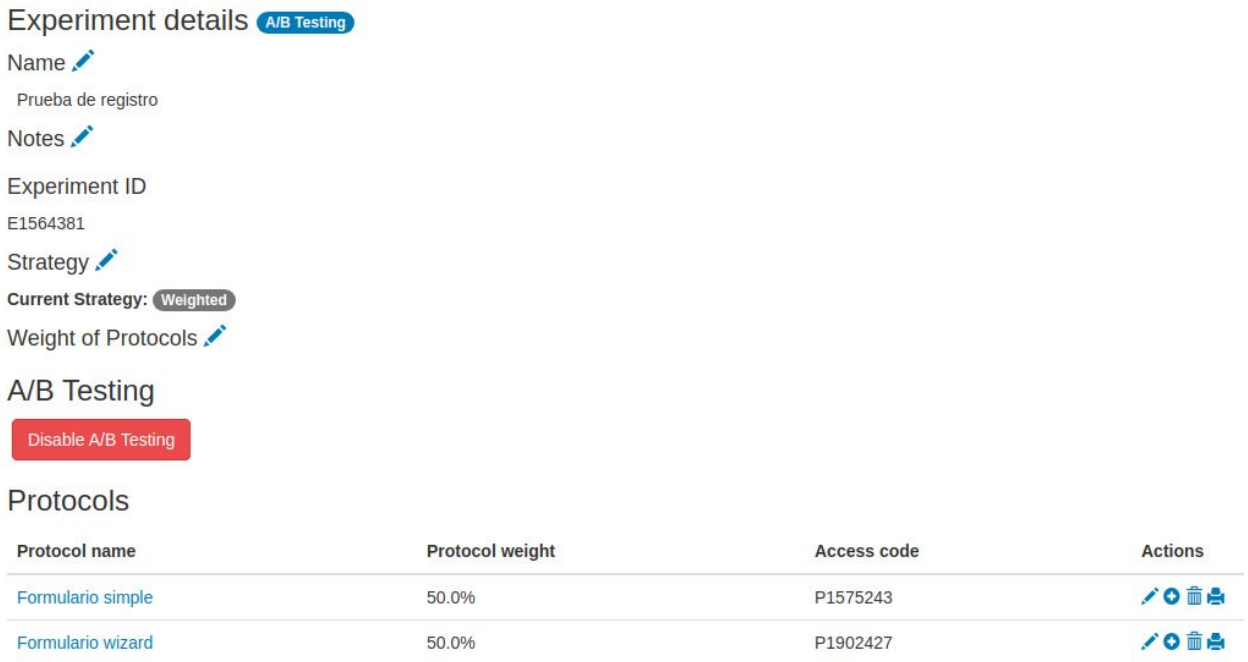


Figura 6.4. Sesión de experimento de los formularios de registros

Al haber distribuido y tener algunas ejecuciones de experimentos realizadas el experto puede visualizar los resultados de las ejecuciones de ambos protocolos. A los efectos de demostrar la utilización de la herramienta desarrollada y siguiendo la lógica de Nielsen en la que indica que con solo 5 pruebas de usuarios es posible detectar al menos el 85% de las problemas de usabilidad (Nielsen, 2000) se realizaron 10 pruebas de usuario con el objetivo de que 5 personas realicen las pruebas sobre el formulario simple y las demás hagan lo propio con el formulario que tiene el wizard.

Lo primero que se puede analizar en la pantalla de visualización de resultados es que se distribuyeron en partes iguales las ejecuciones como se deseaba. En segundo lugar se puede ver una pequeña inclinación en que el protocolo del formulario con wizard llevó un poco menos de tiempo de ejecución que los protocolos con el formulario simple. Esto se puede visualizar en la figura 6.5.

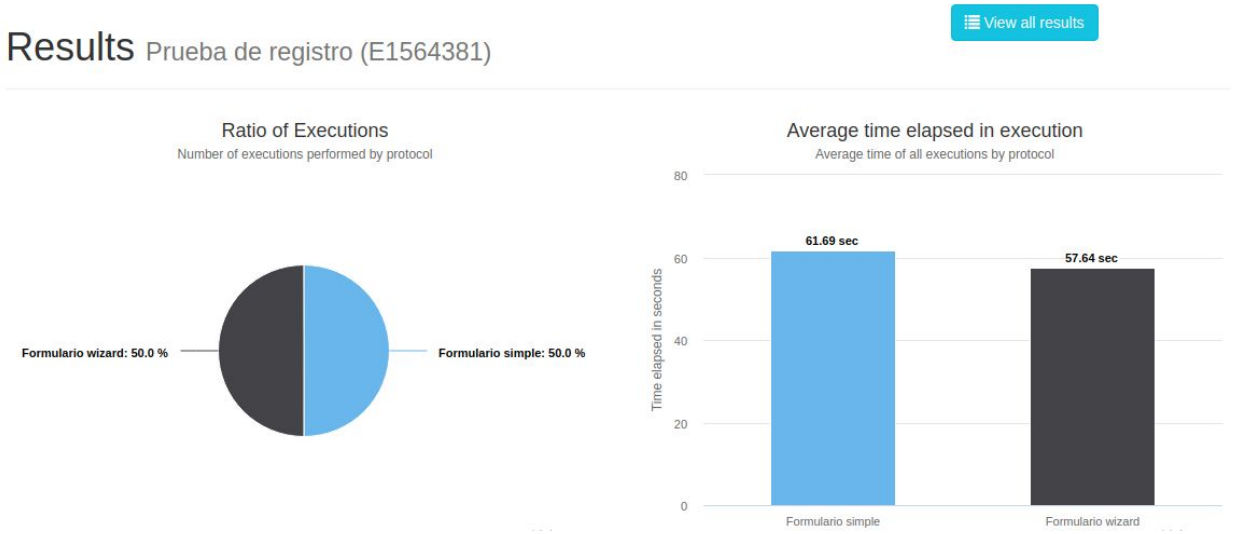


Figura 6.5. Resultados genéricos de las ejecuciones realizadas

Si nos adentramos en la tarea fundamental del experimento, se puede observar que mientras el formulario simple (Figura 6.6) tuvo un tiempo promedio de ejecución en un valor de *60.46 segundos*, el formulario con el wizard (Figura 6.7) tuvo un tiempo promedio de ejecución de *56.65 segundos*. Con respecto a los movimientos y utilización del mouse, el que menos requirió de su intervención fue el formulario con el wizard, ya que tanto en cantidad de clicks realizados en promedio como en la velocidad con la que movieron el mouse los voluntarios los valores son menores. Otra característica que se puede marcar es que en las ejecuciones del formulario simple los voluntarios estuvieron menos tiempo inactivos que los del formulario con el wizard.

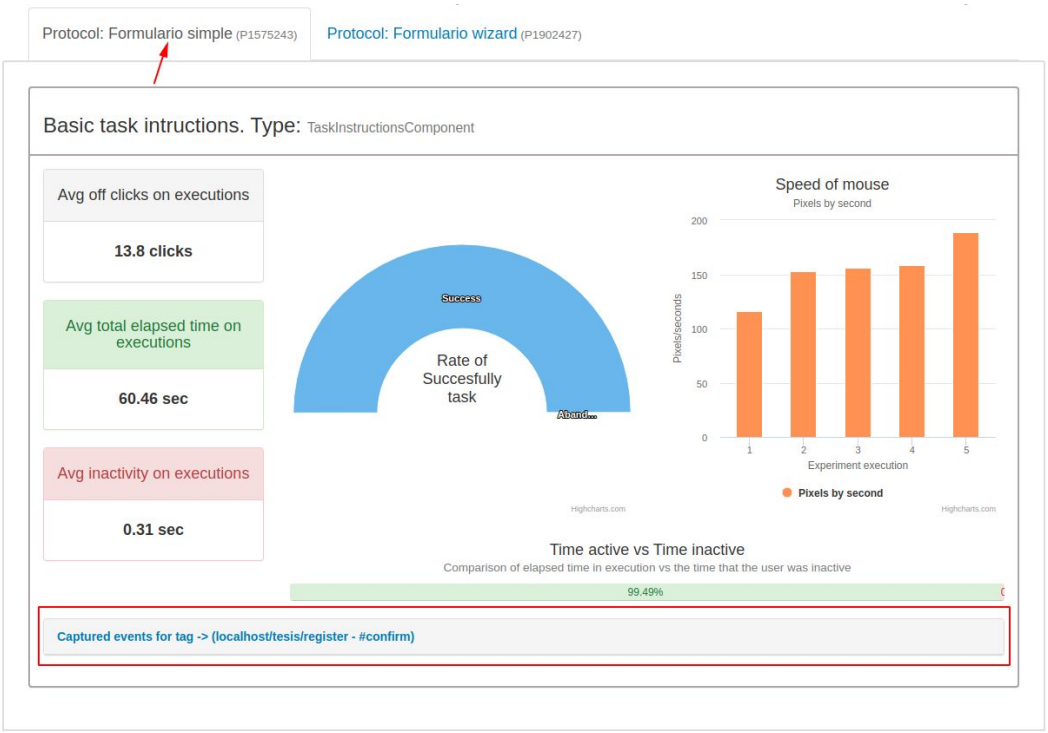


Figura 6.6. Resultados de ejecuciones del formulario simple

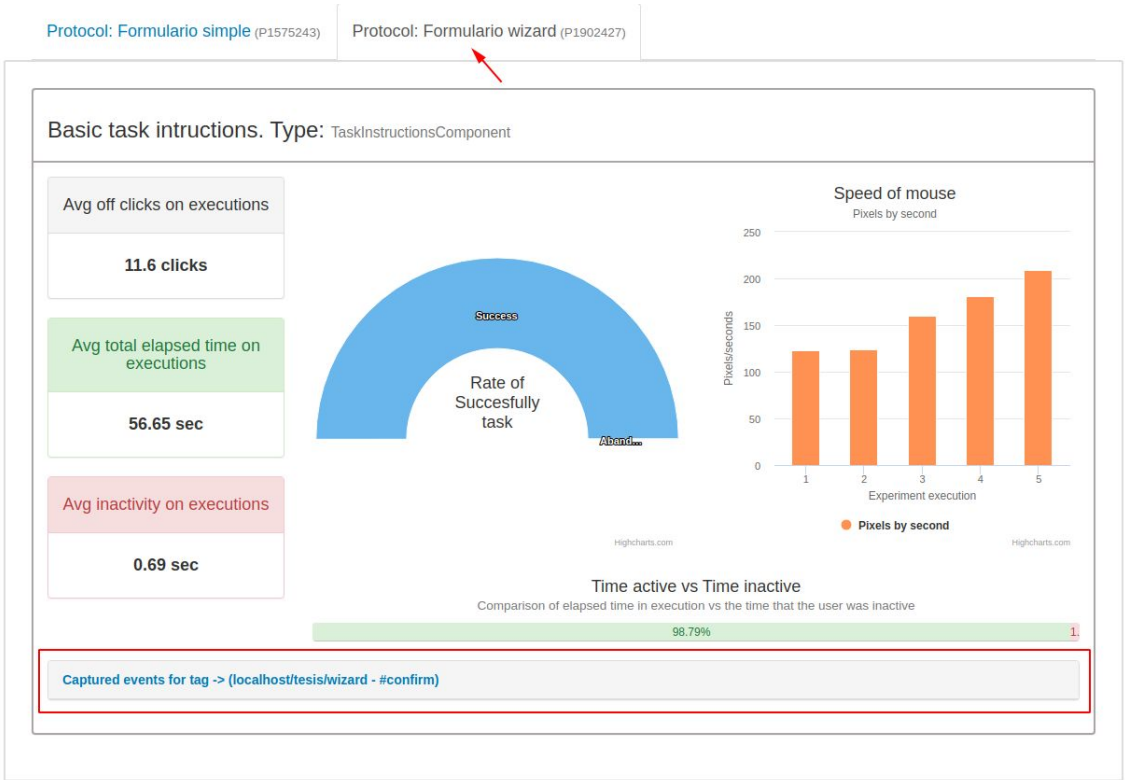


Figura 6.7. Resultados de ejecuciones del formulario con el wizard

Luego de ver la información capturada de manera genérica en las ejecuciones se puede observar que debajo (enmarcado en ambas figuras) hay un panel que se puede expandir y en el que se encontrará la información capturada con respecto al elemento html con *id="confirm"* (#confirm), que en este caso es el botón de confirmación del formulario. Si se expande el panel podemos ver en las figuras 6.8 los tiempos de llegada al botón y los clicks realizados sobre el elemento en el caso del formulario simple y en la figura 6.9 lo mismo con el formulario wizard.

Figura 6.8. Resultados sobre el botón de confirmación del formulario simple

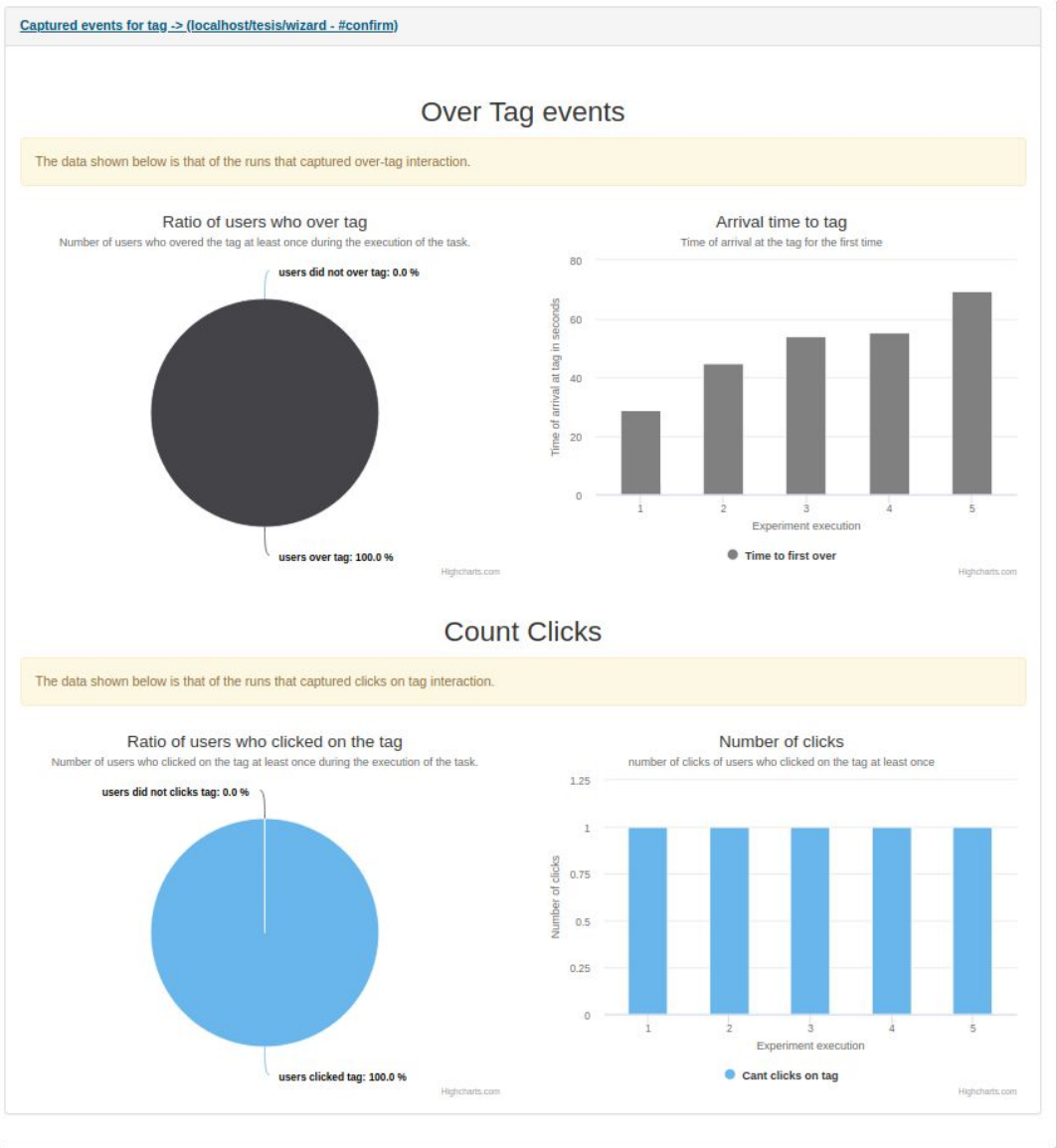


Figura 6.9. Resultados sobre el botón de confirmación del formulario con el wizard

En primer lugar vemos los eventos en que los usuarios pasaron por encima del elemento. En el gráfico de tortas se muestra que el 100% de los usuarios pasaron por encima del botón "confirmar" en ambos formularios. A la derecha se despliega un histograma en el que se muestra el tiempo que tardaron los usuarios en llegar al botón por primera vez. A simple vista se puede observar que aquellos voluntarios que ejecutaron la prueba del formulario con el wizard llegaron un poco antes al botón de confirmación que los que lo hicieron con el formulario simple.

Más abajo en la misma figura 6.9 se pueden observar unos gráficos similares para la cantidad de clicks. En el que se muestra que el 100% de los usuarios hicieron click sobre el botón. A la derecha se despliega un histograma con la cantidad de clicks realizados en cada ejecución, en

este caso los resultados son idénticos ya que todos los voluntarios lograron completar el formulario por lo tanto tuvieron que hacer click sobre el botón de confirmación para que esto sea correcto. De la misma manera se determina que todos los usuarios clickearon sobre dicho botón una sola vez.

Las conclusiones que se puedan llegar a tomar con los datos brindados por las pruebas ejecutadas están vinculadas con los valores que previamente el experto crea que puedan tener importancia en el momento de tomar la decisión de cuál de las alternativas es la mejor. En este ejemplo se pueden ver algunas diferencias entre ambas versiones:

- ➔ El tiempo que se demoró en realizar el registro fue algo menor en el caso del formulario con wizard.

- ➔ En el formulario simple hubo una mayor utilización del mouse, esto se puede ver en la cantidad de clicks promedio, en la velocidad empleada con el puntero y en el tiempo de inactividad promedio.

- ➔ A través del formulario con wizard los voluntarios llegaron antes al botón de confirmación que con el formulario simple más allá de que con este último el botón se encuentre presente todo el tiempo y en el formulario con el wizard solo se presente al final.

A modo de ejemplo se podría deducir que a la hora de realizar el registro el ir solicitando los datos de a poco (cómo en el formulario con wizard) puede reducir la carga de datos al usuario haciendo que este tenga una mejor predisposición al llenado y no que haga una lectura general del formulario antes de completarlo. De todas formas, mayor va a ser la efectividad de estas pruebas a medida que crezca la cantidad de usuarios (Faulkner, 2003).

6.2 Caso de prueba Nº 2: Gmail

Para este segundo caso de prueba se utilizó la aplicación web de gmail (*gmail.com*). La elección de esta web tiene tres motivos:

1. Gmail es el cliente de correo electrónico más utilizado en todo el mundo. En este caso de prueba, se busca demostrar que es posible realizar pruebas de usuario en aplicaciones externas al experto a través de Tycho. Esto es de utilidad para diferentes tipos de

experimentos a nivel global.

2. Gmail cuenta con dos interfaces diferentes, ya que tiene una versión llamada "clásica" o "en HTML" en la que se puede utilizar la aplicación como se utilizaba hace años atrás y está pensada para navegadores antiguos o conexiones más lentas (Figure 6.10) y tiene otra versión, identificada como "moderna", que es la más conocida y utilizada actualmente (Figure 6.11)

3. Es fácilmente configurable la utilización de gmail con una versión u otra. A través del link "*https://mail.google.com/mail/u/0/h/1pq68r75kzvdr/?v%3Dlui*" es posible ingresar a la sección en la cual se le pregunta al usuario si desea utilizar la versión más moderna de Gmail o la versión en HTML, como se puede observar en la Figura 6.12.

Figura 6.10. Aplicación de Gmail clàsica

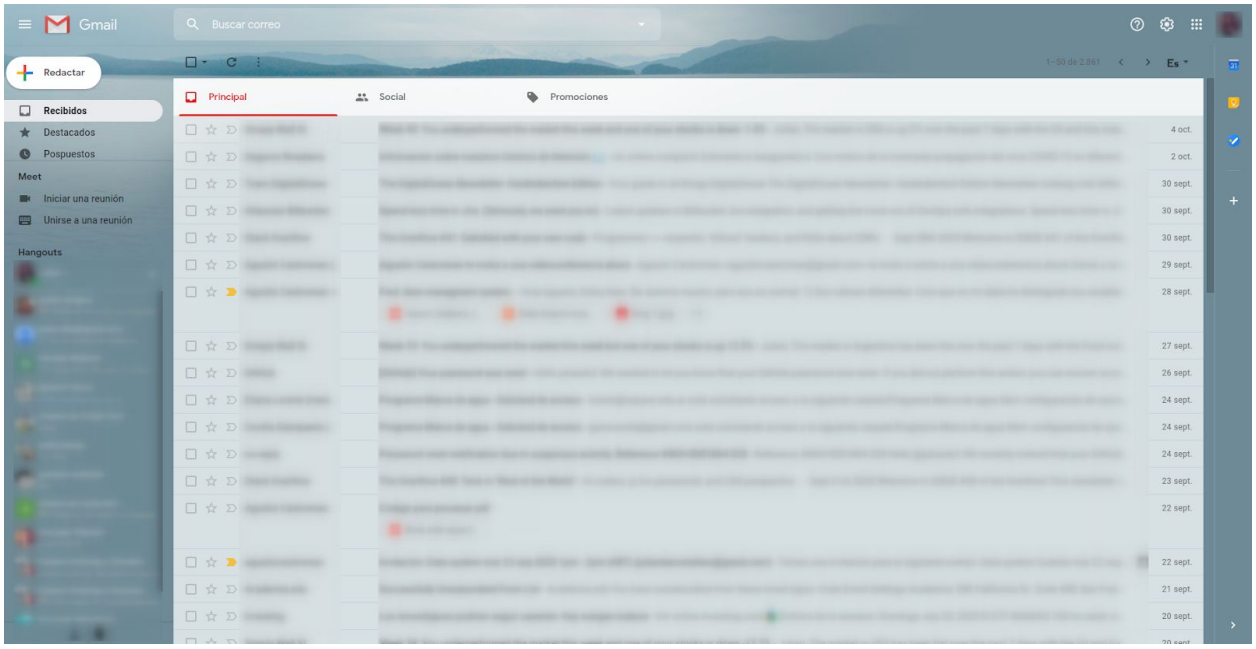


Figura 6.11. Aplicación de Gmail moderna

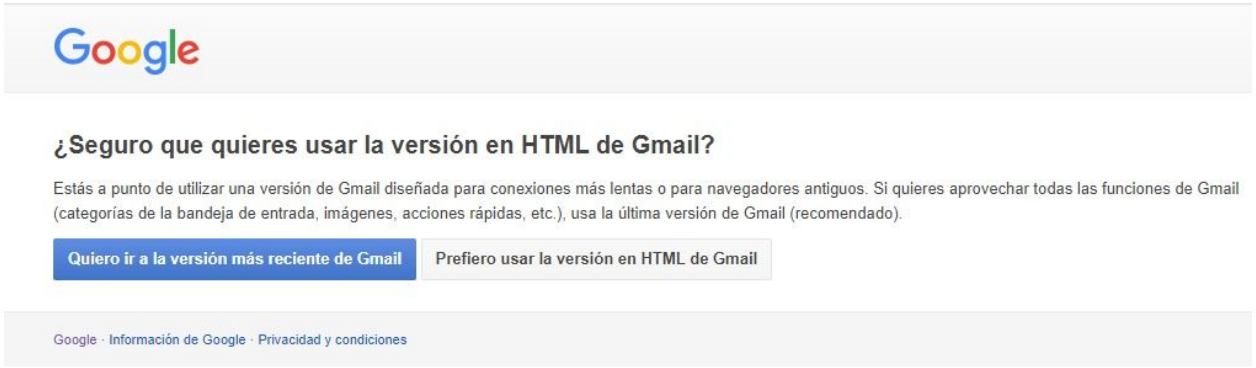


Figura 6.12. Al acceder al link mencionado se presentan las opciones de ingresar a la versión moderna o a la versión con HTML

En base a estas interfaces se crearon tres tareas para la prueba de usuario a realizar:

1. En la primera se indicará al usuario que ingrese al link que lo dirige a la pantalla de la figura 6.12 para pueda seleccionar que versión de gmail quiere utilizar dependiendo del protocolo.

2. En la segunda tarea se le solicita al voluntario que se envíe un correo a sí mismo con el asunto "*Prueba*" y con el cuerpo "*Prueba de usuario*", la tarea finaliza cuando el usuario reciba el correo.

3. En la tercer tarea se le indica al voluntario que marque como destacado los primeros 4 mails que se encuentren en su casilla.

4. Para finalizar en la última tarea se le indica al voluntario que desmarque los mails que destacó y finalice la ejecución del experimento con el objetivo de dejar la casilla de la misma manera que como estaba previamente.

En todas las tareas se capturan las métricas generales al igual que en el caso de prueba anterior. Además en cada una de las tareas se capturan distintos tipos de métricas de elementos html dependiendo de las facilidades que nos brindan las interfaces. Se busca realizar una captura de métricas sobre elementos html que tengan cierta relación, por ejemplo para la versión de gmail con HTML para marcar como destacado un mensaje es necesario marcar con el checkbox cada mail que se desea destacar y, una vez marcados, el usuario debe seleccionar la opción "*Destacar*" a través de un select en el encabezado de la tabla donde se encuentran todos los mails, como se puede ver en la figura 6.13.

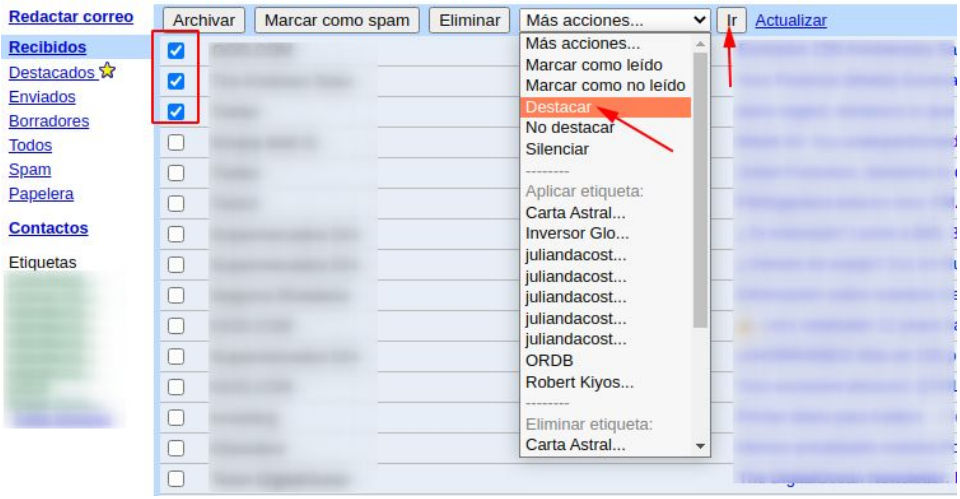


Figura 6.13. Acciones que debe hacer el usuario para destacar mails en la versión HTML

En contrapartida una de las mejoras en la usabilidad que se realizaron con la modernización del sitio es el poder destacar mails presionando en una estrella asociada a cada mail (figura 6.14), lo que agiliza la tarea de destacarlos.

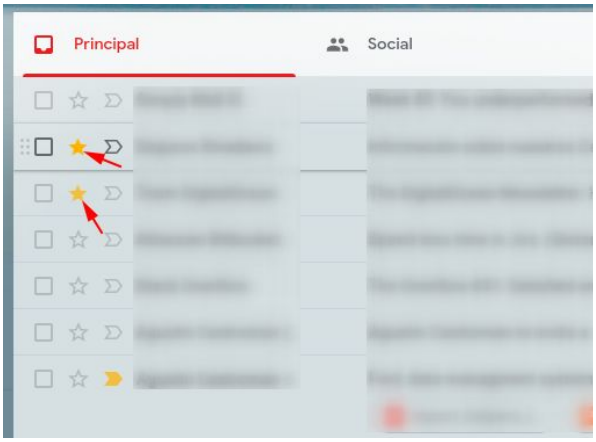


Figura 6.14. Se muestran las estrellas que permiten destacar un mail con un solo click

Para capturar datos relacionados con estas acciones se podrían capturar eventos en elementos html asociados a ellas y ver cómo interactúan los voluntarios para completar la tarea.

Por otro lado, si bien la versión moderna de la aplicación cuenta con un botón "*Actualizar*" para que se carguen los mails encolados en el caso de que la interfaz no se haya actualizado, los mails se cargan de forma automática en la aplicación a medida que van llegando, no es necesario recargar. En la versión HTML muchas veces es necesario recargar o presionar el botón "*Actualizar*" la página para que llegue el mail. Esta es otra de las mejoras en cuanto a la usabilidad que se hicieron en la modernización, y se pueden capturar métricas con respecto a estos datos.

Teniendo en cuenta lo mencionado fue creada una sesión de experimento que cuenta con dos protocolos, uno para la versión HTML clásica, y el otro para la versión moderna. Se habilitó A/B Testing al igual que en el caso de prueba Nº 1 (Figura 6.15).

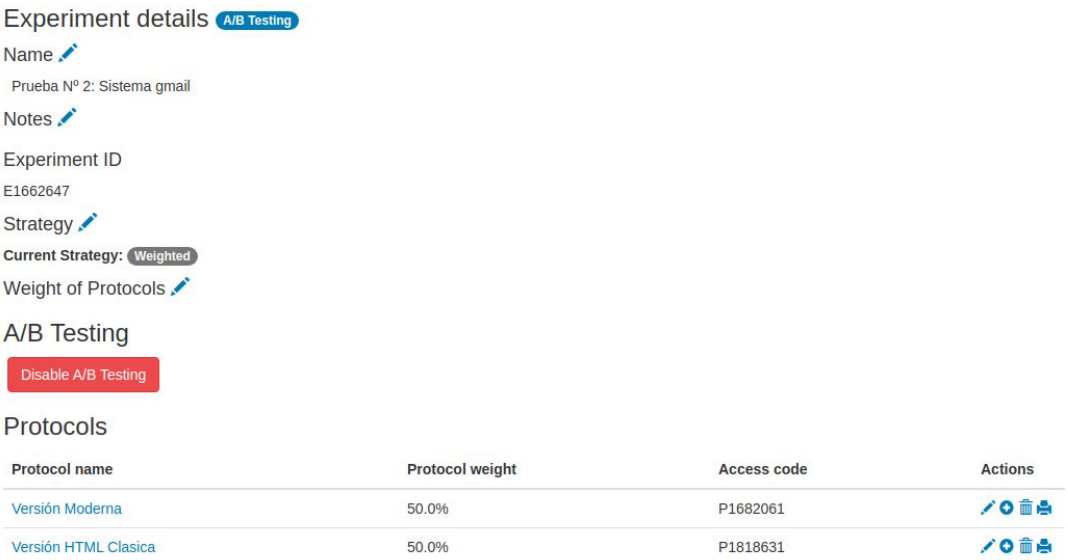


Figura 6.15. Sesión de experimento para las pruebas de Gmail con ambos protocolos definidos

Siguiendo la misma lógica empleada para el caso de prueba Nº 1 con respecto a la regla de los 5 usuarios, se distribuyeron las pruebas a 10 usuarios para que cada protocolo cuente con al menos 5 ejecuciones. Al visualizar los resultados genéricos se puede ver como la interfaz moderna tiene un tiempo menor de ejecución que la versión con html (Figura 6.16).

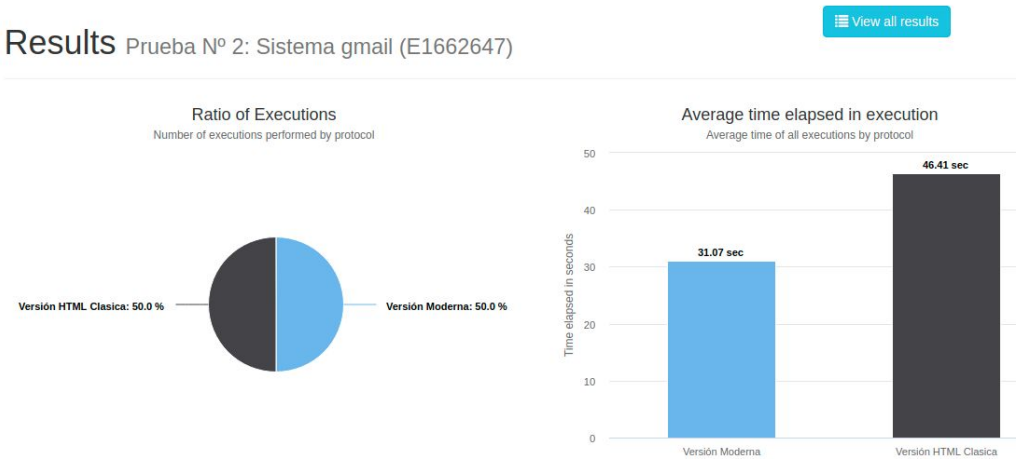


Figura 6.16. Resultados genéricos de la ejecución de ambos protocolos

Adentrándose en la primer tarea de peso del experimento, que consiste en el envío de un mail a sí mismo y posteriormente leerlo, se puede observar que en cuanto a tiempo e interacción con el mouse el protocolo con la interfaz clásica (Figura 6.18) fue más costoso que el protocolo con la interfaz moderna (Figura 6.17), aunque en esta última los usuarios tuvieron un mayor tiempo de inactividad.

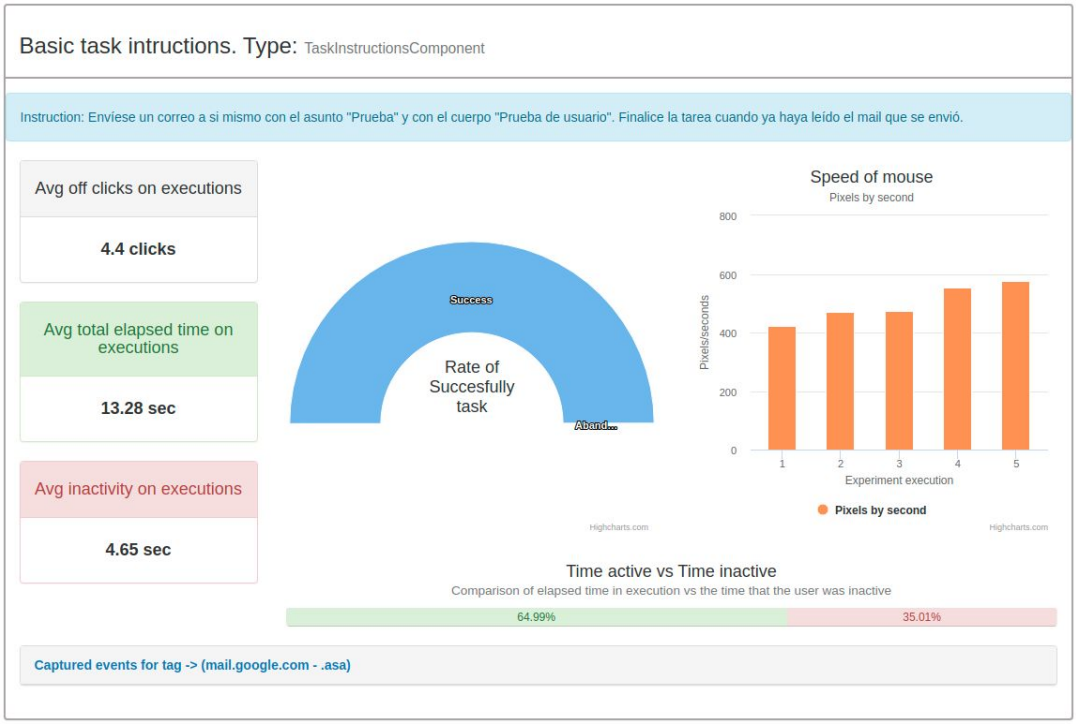


Figura 6.17. Resultados envío de mail y lectura del mismo en versión moderna

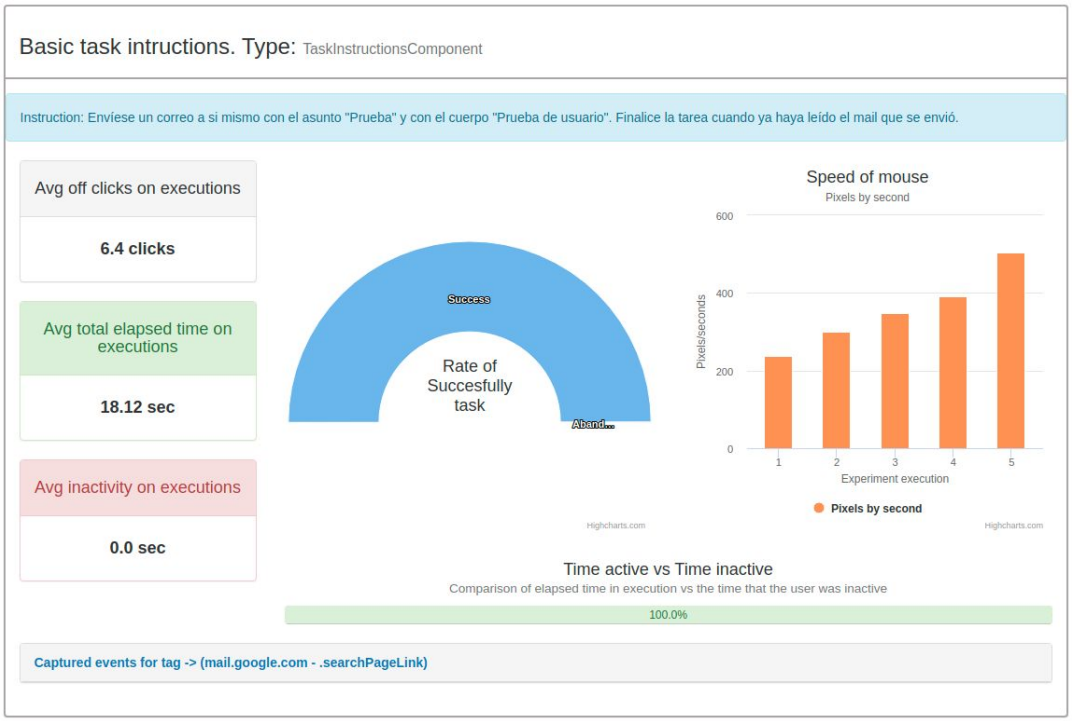


Figura 6.18. Resultados envío de mail y lectura del mismo en versión HTML clásica

En esta tarea además se capturaron datos sobre los elementos html que permiten recargar la bandeja de entrada, la interacción con estos elementos podrían ser un buen indicio de fallas en las cargas de nuevos mails. A través de la inspección del código html en ambos sitios se detectó que en la versión clásica este elemento html tiene asociada la clase "*searchPageLink*" (Figura 6.19) y en la versión moderna tiene asociada la clase "*asa*" (Figura 6.20). En la primera se nota cierta interacción: el 60% de los voluntarios hicieron click sobre el botón de recarga. Con respecto a la versión moderna ninguno de los voluntarios debió recargar la página para obtener el mail que se habían enviado anteriormente.

Figura 6.19. Indicios de interacciones con el elemento para recargar la bandeja de entrada en la versión HTML

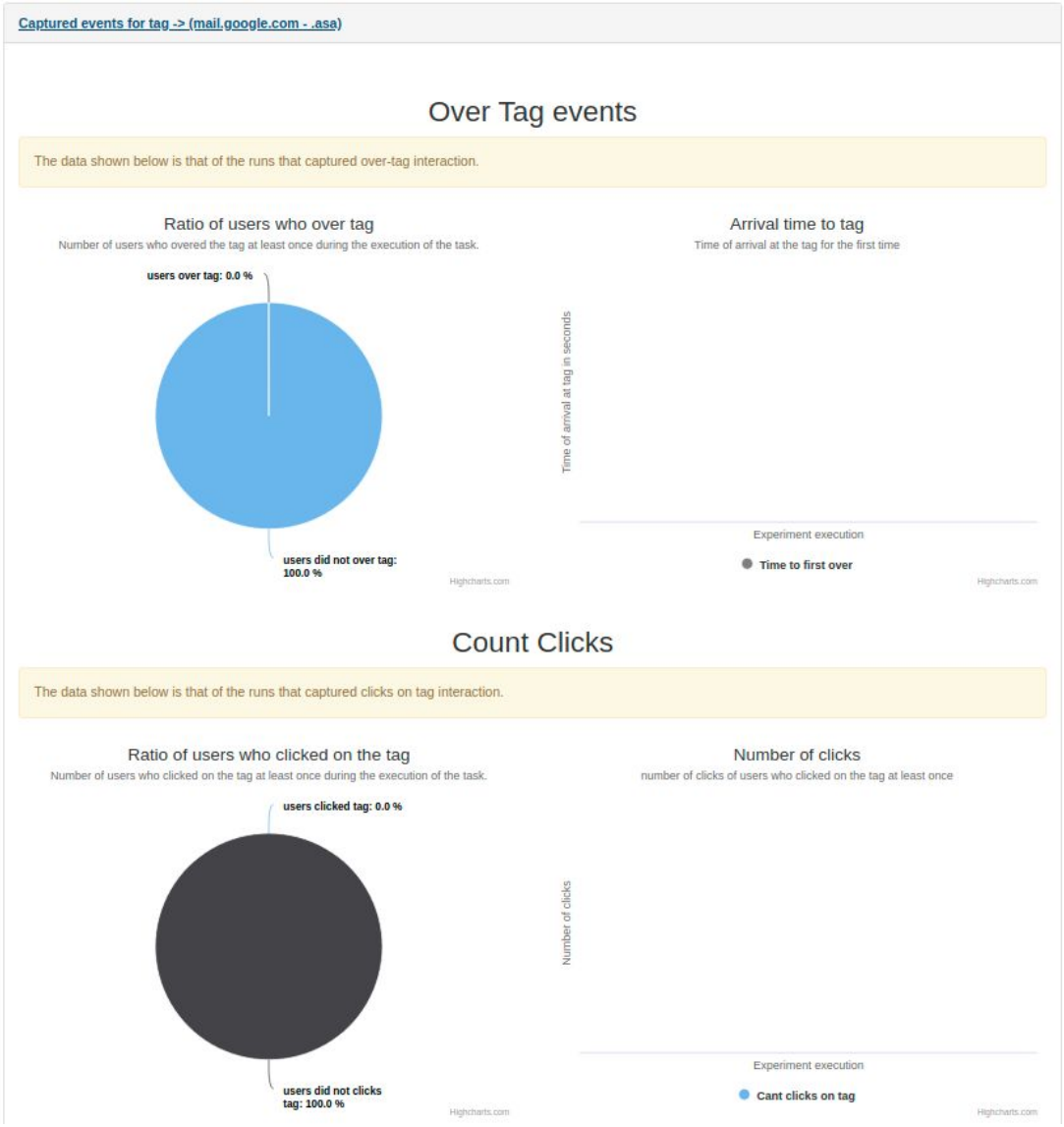


Figura 6.20. Indicios de interacciones con el elemento para recargar la bandeja de entrada en la versión moderna

Por último con respecto a la tarea en la que el voluntario debe marcar un determinado número de mails como destacados, se detecta que la versión moderna arroja resultados considerablemente mejores que la versión HTML. En cuanto a tiempo, la versión moderna cuenta con un promedio de 8,22 segundos para realizar esta acción (Figura 6.21), en cambio la versión HTML tiene un tiempo promedio de 15,04 segundos (Figura 6.22), casi el doble que la versión moderna. Por último, los clicks realizados durante las ejecuciones son menos en la versión moderna que en la versión con HTML.

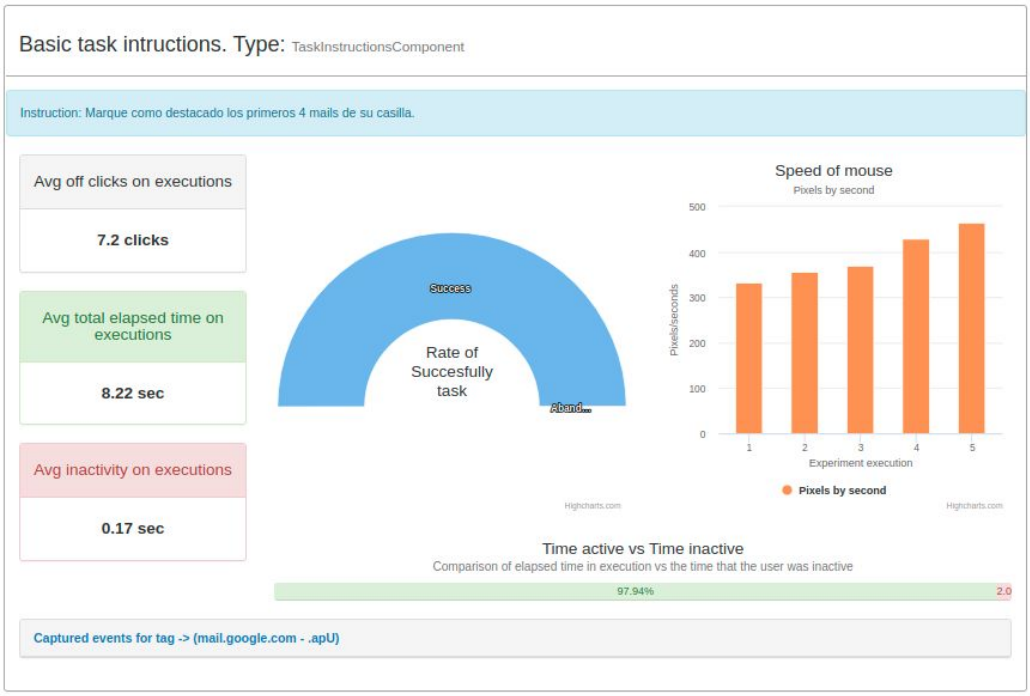


Figura 6.21. Resultados de las ejecuciones de la tarea en la que el voluntario debe marcar como destacados algunos mails en la versión moderna

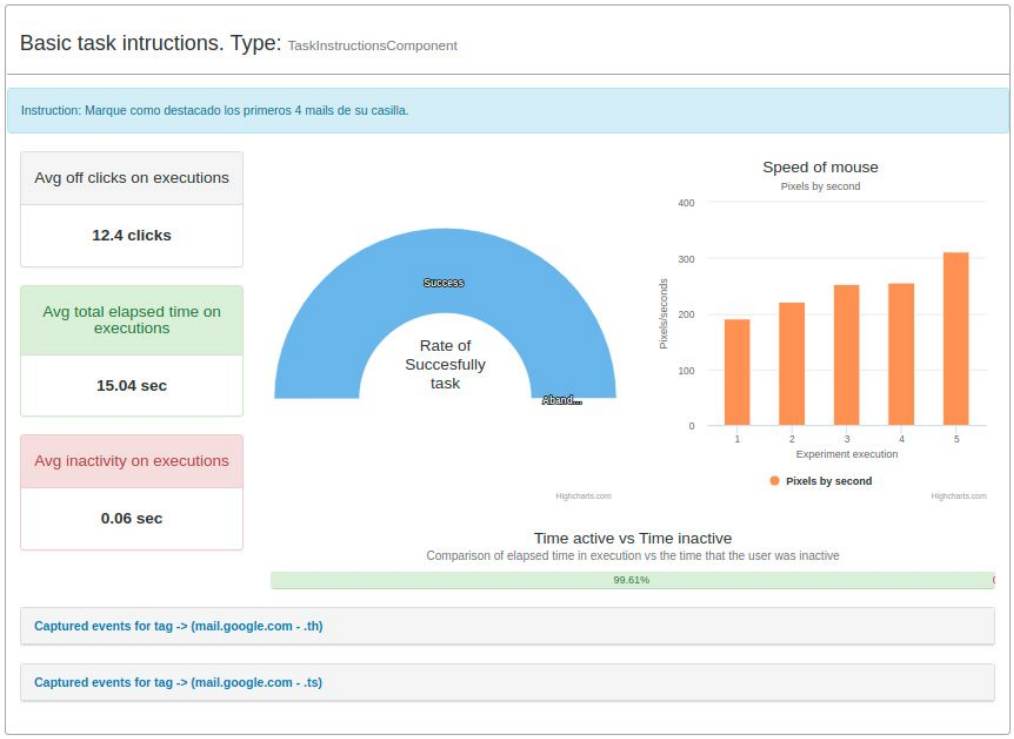


Figura 6.22. Resultados de las ejecuciones de la tarea en la que el voluntario debe marcar como destacados algunos mails en la versión HTML

También fueron capturados eventos sobre elementos html puntuales para esta tarea: en la versión HTML se capturaron eventos sobre los links que redireccionan a los mails y sobre la filas de la tabla donde se muestran los mismos (Figura 6.23), con el objetivo de detectar confusión en los voluntarios con respecto a dónde deben dirigirse ya que la interfaz no es muy amigable. En la versión moderna se capturaron eventos sobre las estrellas que se muestran en la figura 6.14 y que sirven para marcar como destacados los mails. Efectivamente las estrellas fueron ampliamente utilizadas por todos los usuarios (Figura 6.24).

Figura 6.23. Interacciones del usuario que indican confusión a la hora de marcar como destacados los mails en la versión HTML.

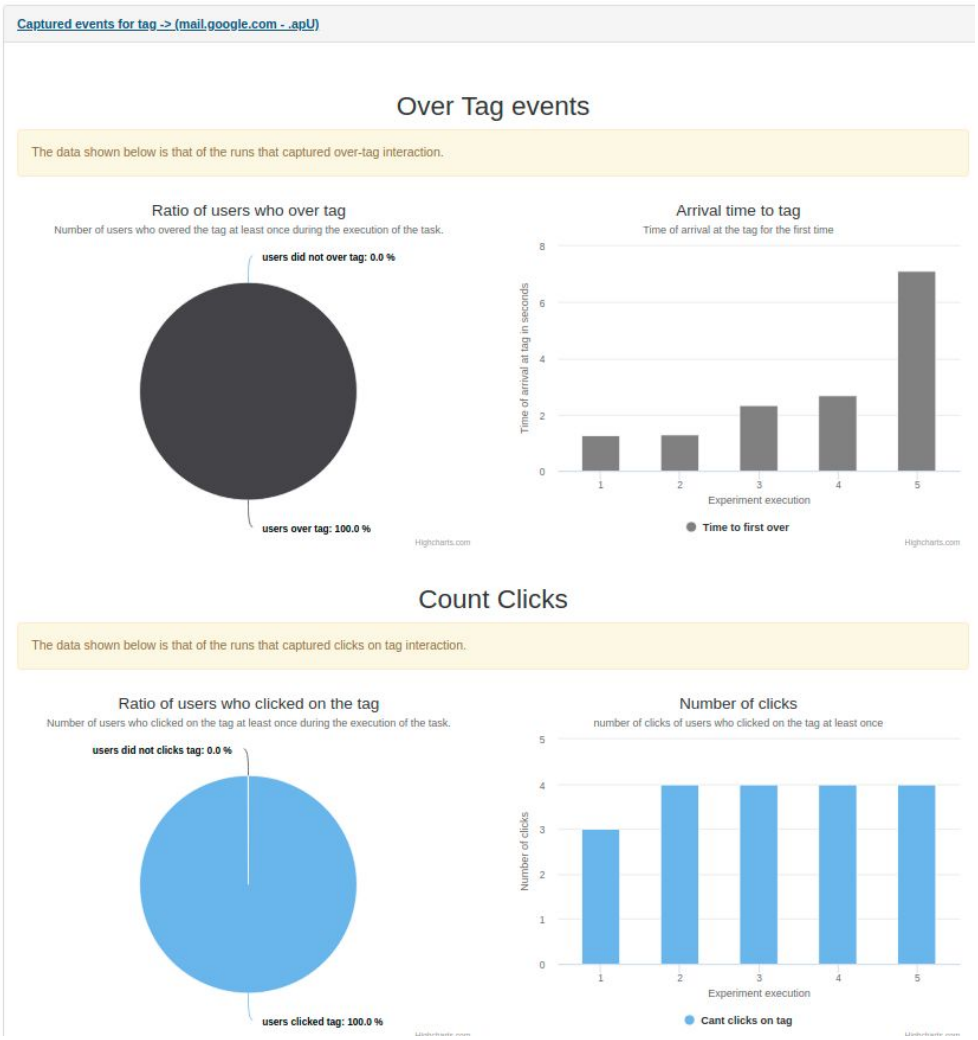


Figura 6.24. Interacciones del usuario que indican confusión a la hora de marcar como destacados los mails en la versión HTML.

Si bien los resultados favorables a la versión moderna eran esperables por una cuestión de diseño y un mejor tratamiento de la usabilidad, lo que se quiere probar con este caso es la posibilidad que brinda Tycho de poder realizar pruebas de usabilidad en sistemas externos y sin realizar ningún tipo de modificación sobre ellos. Por último se comprueba que los resultados que arroja la herramienta se condicen con una mejor usabilidad en la aplicación y queda demostrado que el valor agregado que se le pueden dar a los datos capturados quedan en manos de la profundidad del análisis realizado por el evaluador.

Capítulo 7
Conclusiones

A través de Tycho se pueden realizar pruebas de usuarios moderadas de forma remota sobre cualquier aplicación web y comparar los resultados de sus ejecuciones. En base a los objetivos propuestos se alcanzó la premisa fundamental de extender la herramienta para aplicar A/B Testing sobre las pruebas de usuario que Tycho permite diseñar. Para ello fue necesario realizar distintas extensiones a lo largo de toda la herramienta para que sea posible asociar dos o más pruebas de usuario que se encuentren bajo la misma sesión de experimento, capturar nuevas métricas durante las ejecuciones y llevar a cabo distintos métodos para la visualizaciones de resultados que permitan comparar las ejecuciones de protocolos equivalentes.

Vale aclarar, que no fue implementada una solución que permita vincular tareas específicas de cada protocolo y que la herramienta realice un análisis o comparación previo facilitando el estudio que deba realizar el experto. Esto se debe a que no existe una restricción a que todos los protocolos deban tener la misma cantidad de tarea ni el mismo nivel de detalle en las instrucciones, lo que complejiza una vinculación directa entre tareas de distintos protocolos cuando se define la sesión de experimentos.

Por otro lado, se crearon estrategias para balancear la distribución a los distintos participantes de las pruebas que se encuentren bajo una misma sesión de experimentos. El desarrollo realizado tuvo el foco en que la herramienta sea extensible utilizando distintos patrones de diseño en las modificaciones que se debieron realizar.

En cuanto a la comparación de resultados obtenidos de las pruebas ejecutadas, el experto contará con los datos capturados durante las ejecuciones de cada protocolo puesta a disposición a través de diferentes métodos de visualización. La granularidad con la que se define el nivel de detalle de cada uno de los experimentos depende de lo que el evaluador desea probar. A raíz de esta decisión, desemboca cuánta libertad se le brinda al usuario para realizar las pruebas y sobre que sitios en particular se capturarán datos. Tycho brinda la libertad de que cada experto pueda elegir la captura de métricas que crea relevante para la prueba que se desea evaluar. Por ejemplo, como se mencionó en la sección 2.2, en la investigación de Speicher et al. (2014) se relacionan ciertos elementos de la usabilidad con métricas que son capturadas para páginas de búsqueda de tipo SERP. En ese trabajo Both llegó a conclusiones como por ejemplo: si el cursor se mueve más lento en un input de búsqueda hay mayor densidad de información que si lo hace lentamente, una

velocidad mayor de cursor en la navegación indica más confusión, etc. La herramienta extendida en la presente tesina tiene como objetivo brindar amplitud al experto a la hora de definir qué métricas se quieren capturar durante las ejecuciones para que el mismo pueda detectar que utilidad le brindan en base al tipo de prueba realizada y el formato de la aplicación probada.

Por último, en el capítulo 6 se llevaron a cabo dos casos de prueba en los que se demuestra la herramienta en funcionamiento con todas sus extensiones. A través de estos casos se puede observar todo el proceso que conlleva la definición de un experimento y el posterior análisis de los resultados por parte del experto en usabilidad. El primero de los casos de prueba (sección 6.1) fue sobre un sistema que simula la registración de un usuario a través de dos formularios que solicitan la misma información pero son presentados de manera diferente. El segundo de los casos de prueba (sección 6.2) fue realizado sobre dos versiones del sitio de Gmail, la versión actual y una versión anterior del sitio a la que aún hay acceso. Mediante la naturaleza de las pruebas queda en evidencia la flexibilidad de Tycho a la hora de realizar sesiones de experimentos sobre cualquier sitio web, ya sea un desarrollo propio (sección 6.1) o una aplicación externa (sección 6.2).

Capítulo 8
Trabajos futuros

Tycho es una herramienta extensible y se trabaja con el objetivo de potenciar esta característica y que la misma se mantenga a lo largo de las actualizaciones. A través del desarrollo del presente trabajo se abrió la posibilidad de trabajar sobre dos elementos fundamentales para las pruebas de usuario: la captura de métricas y la visualización de resultados. Más allá de cualquier otra mejora que se realice en el futuro, es posible seguir potenciando esta área para hacer de Tycho una herramienta más robusta y completa que lo que es hasta el momento.

Considerando lo anteriormente dicho, algunos trabajos futuros posibles son:

- ➔ Introducir nuevas métricas, como por ejemplo información del dispositivo donde se está realizando la tarea o aumentar el nivel de detalle con respecto a las interacciones que el usuario realiza sobre la aplicación. Además sería interesante tener información genérica sobre los voluntarios que ejecutan las tareas para poder realizar un relevamiento demográfico y relacionarlo con los resultados obtenidos.

- ➔ Mejorar la usabilidad de la herramienta por parte de los voluntarios, por ejemplo podría ser de utilidad que el voluntario sepa el progreso que lleva realizado del experimento o un tiempo estimado que tardará en finalizar. Además en algunas circunstancias se podrían agregar imágenes a la hora de definir las instrucciones con el objetivo de ser más explícito en las instrucciones de la tarea a realizar y el voluntario las entienda con mayor facilidad.

- ➔ Considerando la mejora realizada del balanceo de cargas en base a una estrategia cuando se habilita A/B Testing, podría ser interesante brindar el servicio de distribución del experimento a un listado de usuarios o a través de herramientas de crowdsourcing como en (Nebeling et al., 2013).

- ➔ Que el experto pueda habilitar la grabación de la pantalla durante las ejecuciones realizadas por los voluntarios sería un valor agregado interesante a Tycho. Posterior a la grabación se podría procesar y analizar la captura para sacar nuevas métricas o incluso para que el experto pueda ver cómo se realizaron los experimentos.

Bibliografía

Atterer, R., Wnuk, M., & Schmidt, A. (2006). Knowing the user's every move: User Activity Tracking for Website Usability Evaluation and Implicit Interaction. Proceedings of the 15th international conference on World Wide Web - WWW '06. https://doi.org/10.1145/1135777.1135811

Bass, L., & John, B. E. (2003). Linking usability to software architecture patterns through general scenarios. Journal of Systems and Software, 66(3), 187-197. https://doi.org/10.1016/s0164-1212(02)00076-6

Brooke, John. (2013). SUS: a retrospective. Journal of Usability Studies. 8. 29-40.

Chen, M. C., Anderson, J. R., & Sohn, M. H. (2001). What can a mouse cursor tell us more? Correlation of eye/mouse movements on web browsing. CHI '01 extended abstracts on Human factors in computing systems - CHI '01, 281-282. https://doi.org/10.1145/634067.634234

Faulkner, L. (2003). Beyond the five-user assumption: Benefits of increased sample sizes in usability testing. Behavior Research Methods, Instruments, & Computers, 35(3), 379-383. https://doi.org/10.3758/bf03195514

Fernandez, A., Abrahão, S., & Insfran, E. (2011). A Web Usability Evaluation Process for Model-Driven Web Development. Advanced Information Systems Engineering, 108-122. https://doi.org/10.1007/978-3-642-21640-4_10

Firmenich, S., Garrido, A., Grigera, J., Rivero, J. M., & Rossi, G. (2018). Usability improvement through A/B testing and refactoring. Software Quality Journal, 27(1), 203-240. https://doi.org/10.1007/s11219-018-9413-y

Goecks, J., & Shavlik, J. (2000). Learning users' interests by unobtrusively observing their normal behavior. Proceedings of the 5th international conference on Intelligent user interfaces - IUI '00, 129-132. https://doi.org/10.1145/325737.325806

Guo, Q., & Agichtein, E. (2012). Beyond dwell time. Proceedings of the 21st international conference on World Wide Web - WWW '12, 569-578. https://doi.org/10.1145/2187836.2187914

Hartson, H. R., Castillo, J. C., Kelso, J., & Neale, W. C. (1996). Remote evaluation. Proceedings of the SIGCHI conference on Human factors in computing systems common ground - CHI '96, 228-235. https://doi.org/10.1145/238386.238511

Hehman, E., Stolier, R. M., & Freeman, J. B. (2014). Advanced mouse-tracking analytic techniques for enhancing psychological science. Group Processes & Intergroup Relations, 18(3), 384-401. https://doi.org/10.1177/1368430214538325

Hong, J., Heer, J., Waterson, S., & Landay, J. A. (2001). WebQuilt: A Proxy-based Approach to Remote Web Usability Testing. ACM Transactions on Information Systems, 19(3), 263-285. https://doi.org/10.1145/502115.50211

Huang, J., White, R. W., & Dumais, S. (2011). No clicks, no problem: : Using Cursor Movements to Understand and Improve Search. Proceedings of the 2011 annual conference on Human factors in computing systems - CHI '11. https://doi.org/10.1145/1978942.1979125

International Organization for Standardization. (1994). Ergonomic requirements for office work with visual display terminals.

Jacko, J. A., & Sears, A. (2003). The Human-Computer Interaction Handbook: Fundamentals, Evolving Technologies and Emerging Applications;Human Factors and Ergonomics. Lawrence Erlbaum Assoc Inc.

Kohavi, R., Tang, D., & Xu, Y. (2020). Trustworthy Online Controlled Experiments: A Practical Guide to A/B Testing. Cambridge University Press.

Krug, S. (2005). Don't Make Me Think: A Common Sense Approach To The Web Usability (2.a ed.). New Riders Pub.

Krug, S. (2009). Rocket Surgery Made Easy. Pearson Education.

Luzik, M. (2014), A/B testing and usability assessment methods in small companies. Aalto University School of Science.

Martin, R. & Shamari, M. & Seliaman, Mohamed & Mayhew, P.J.. (2014). Remote Asynchronous Testing: A Cost-Effective Alternative for Website Usability Evaluation. International Journal of Computer and Information Technology. 03.

Moran, K., & Pernice, K. (2020, 26 abril). Remote Moderated Usability Tests: How to Do Them. Nielsen Norman Group. https://www.nngroup.com/articles/moderated-remote-usability-test/

Nebeling, M. & Speicher, M. & Norrie, M. (2013). CrowdStudy: General Toolkit for Crowdsourced Evaluation of Web Interfaces. EICS 2013 - Proceedings of the ACM SIGCHI Symposium on Engineering Interactive Computing Systems. 10.1145/2480296.2480303.

Nielsen, J. (1993). Usability Engineering. Morgan Kaufmann.

Nielsen, J. (2000). Why You Only Need to Test with 5 Users. Nielsen Norman Group. https://www.nngroup.com/articles/why-you-only-need-to-test-with-5-users/

Nielsen, J. (2012). Usability 101: Introduction to Usability
https://www.nngroup.com/articles/usability-101-introduction-to-usability/

Offutt, J.. (2002). Quality attributes of Web software applications. Software, IEEE. 19. 25-32. 10.1109/52.991329.

Reeder, R. W., Pirolli, P., & Card, S. K. (2001). WebEyeMapper and WebLogger. CHI '01 extended abstracts on Human factors in computing systems - CHI '01, 19-20. https://doi.org/10.1145/634067.634082

Schade, A. (2013, 12 octubre). Remote Usability Tests: Moderated and Unmoderated. Nielsen Norman Group. https://www.nngroup.com/articles/remote-usability-tests/

Siroker, D., & Koomen, P. (2013). A / B Testing: The Most Powerful Way to Turn Clicks into Customers. Wiley.

Speicher, M., Both, A., & Gaedke, M. (2013). TellMyRelevance! Predicting the Relevance of Web Search Results from Cursor Interactions. Proceedings of the 22nd ACM international conference on Conference on information & knowledge management - CIKM '13. https://doi.org/10.1145/2505515.2505703

Speicher, M., Both, A., & Gaedke, M. (2014). Ensuring Web Interface Quality through Usability-Based Split Testing. Lecture Notes in Computer Science, 93-110. https://doi.org/10.1007/978-3-319-08245-5_6

Thompson, K. & Rozanski, E. & Haake, A. (2004). Here, there, anywhere: Remote usability testing that works. 132-137. 10.1145/1029533.1029567.

UNIVERSIDAD NACIONAL DE LA PLATA
FACULTAD DE INFORMÁTICA

La Plata, 22 de Octubre de 2020

Sr.Decano
Facultad de Informática
S/D

De mi consideración:

En referencia a lo solicitado por expediente 3300-1853/19, cumplo en informar que el trabajo de grado “Moderación remota de pruebas de usuario con soporta para A/B testing y foco en usabilidad Web ", realizada por Julian Da Costa Faro y dirigida por Alejandro Fernandez y Julia Grigera

- Propuesta presentada en diciembre 2019.
- Informe de avance presentado en agosto de 2020

La Comisión Evaluadora del trabajo está integrada por los profesores:

- Garrido
- Rodriguez Ch.
- Pasini
- Schiavoni (suplente)

El trabajo no cuenta con respuestas no favorables por parte de los miembros de la Comisión Evaluadora.

Dra. Patricia Bazán
Coordinación de Trabajos de Grado

3300-1853/19-002. Julián Mathias Da Costa Faro. “Moderación remota de pruebas de usuario con soporte para A/B Testing y foco en usabilidad web”. Director: Dr. Alejandro Fernández. Codirector: Dr. Julián Grigera.

27/10/2020

El 27/10/20 por unanimidad la Comisión de Enseñanza aconseja ratificar la Comisión Evaluadora formada por los Prof Alejandra Garrido, Ariel Pasini y Christian Rodríguez (titulares) y Alejandra Schiavoni (suplente).

Dr. Marcelo Naiouf
Secretario Académico
Facultad de Informática

Calle 50 y 120 | C.P.1900 | La Plata | Buenos Aires | República Argentina
Tel.: (54) 221-427-7270/71| www.info.unlp.edu.ar

Corresponde a Expediente N° 3300-001853/19-003

La Plata, 29 de Octubre de 2020

VISTO

las Resoluciones UNLP 667/20, 805/20, 807/20, 812/20, 819/20, 1568/20, 1635/20, 1788/20, 1881/20, 1885/20, 1943/20, 2177/20, 2470/20, 3052/20 y 3238/20 que facultan a los decanos a tomar medidas extraordinarias.

El informe final de tesina de Licenciatura presentado por Julián Mathias Da Costa Faro a través de expediente N° 3300-001853/19-003

la Ord. 307 de la Facultad de Informática de la UNLP

CONSIDERANDO

la opinión de la Comisión Asesora de Enseñanza del 27/10/2020

----el Honorable Consejo Directivo en su sesión de fecha 29/10/2020, por unanimidad (16 votos)

RESOLVIÓ

ARTICULO 1°.- RATIFICAR la comisión evaluadora de la tesina de Licenciatura titulada: “Moderación remota de pruebas de usuario con soporte para A/B Testing y foco en usabilidad web”, presentada por Julián Mathias Da Costa Faro. Director: Dr. Alejandro Fernández. Codirector: Dr. Julián Grigera, con los siguientes miembros:

Titulares:
Prof. Alejandra Garrido
Prof. Ariel Pasini
Prof. Christian Rodríguez
Suplentes:
Prof. Alejandra Schiavoni

ARTICULO 2°.- NOTIFÍQUESE de la presente a los interesados. GIRESE a la Dirección de Enseñanza a sus efectos. REGÍSTRESE por Mesa de Entradas y Archivo.

Prof. Dr. Marcelo Naiouf
Secretario Académico

Prof. Lic. Patricia Pesado
Decana

Resolución HCD Nº 247/20

Calle 50 y 120 | C.P.1900 | La Plata | Buenos Aires | República Argentina
Tel.: (54) 221-427-7270/71| www.info.unlp.edu.ar

3300-1853/19-003. Julián Mathias Da Costa Faro. “Moderación remota de pruebas de usuario con soporte para A/B Testing y foco en usabilidad web”. Director: Dr. Alejandro Fernández. Codirector: Dr. Julián Grigera.

29/10/2020

Pase a la Oficina de Alumnos hasta la defensa oral y pública de la tesina.

Cumplido, pase el expediente a Archivo.

Dr. Marcelo Naiouf
Secretario Académico
Facultad de Informática