



TESINA DE LICENCIATURA

Título: Usabilidad Comunitaria. Un sistema colaborativo para la mejora de la experiencia de usuario

Autores: Gabriel Ricardo Zanetti - Juan José Pertino

Director: Dra. Alejandra Garrido

Codirector: Dr. Sergio Firmenich

Asesor profesional: -

Carrera: Licenciatura en Sistemas

Resumen

Las aplicaciones web tienen un papel cada vez más importante en nuestras vidas. La usabilidad y la experiencia de usuario se han convertido en un importante campo de investigación. En este trabajo nos proponemos desarrollar un sistema que permita la mejora de la experiencia de navegación web haciendo uso de un mecanismo de inyección de scripts de usuario basado en una extensión al navegador web, permitiendo de esta manera afectar a cualquier sitio web, sin necesidad de colaboración de los dueños de dicho sitio en todo el proceso.

Considerando el gran volumen de trabajo que implicaría la generación de estas soluciones proponemos construir una comunidad donde los usuarios reporten los problemas detectados, manifiesten su interés en obtener una solución a problemas previamente reportados, propongan las soluciones que resolverían dichos problemas y las evalúen dando feedback sobre las mismas para que, luego de concluido el período de evaluación, pueda determinarse la mejor solución.

Dado que a lo largo de este trabajo se detectaron múltiples posibilidades de profundización decidimos promover la extensibilidad del sistema exponiendo sus funcionalidades a través de una API, facilitando de este modo su interacción con otras aplicaciones.

Palabras Claves

Usabilidad, Crowdsourcing, A/B testing, scripts de usuario, extensiones de navegador

Conclusiones

Dar soporte a problemas de usabilidad es una tarea compleja debido a la enorme variedad de tipos de usuario en términos de conocimientos informáticos, género, edad o de sus expectativas y necesidades.

Este trabajo permite facilitar la resolución de estos problemas construyendo una comunidad que contribuya con soluciones que son implementadas como scripts de usuario y distribuidas mediante una extensión de navegador, haciendo innecesaria la intervención del propietario del sitio afectado.

Trabajos Realizados

- *Definición de un método para mejorar la experiencia de navegación web basado en la colaboración entre usuarios en un esquema de crowdsourcing*
- *Definición de un mecanismo de evaluación de contribuciones inspirado en procesos de A/B testing, empleando el feedback de los usuarios como métrica de efectividad*
- *Implementación del sitio web, API y extensión del navegador que le dan soporte a la comunidad*

Trabajos Futuros

- *Herramientas de incentivo a la participación: medallas, distintivos recompensas a usuarios*
- *Herramientas de moderación: edición colaborativa de publicaciones, reportar contenido inapropiado*
- *Mejoras a la comunidad: votos en problemas*
- *Mejoras a la extensión del navegador: recordatorio de usuarios, inicio de sesión automático*
- *Herramientas para el desarrollo de soluciones: detección de soluciones inválidas*
- *Mejoras de seguridad*

Usabilidad Comunitaria.

Un sistema colaborativo para la mejora de la experiencia de usuario.

Gabriel Ricardo Zanetti

Juan José Pertino

Tesis presentada para alcanzar el grado de

Licenciado en Sistemas

Facultad de Informática

Universidad Nacional de La Plata

Director: Dra. Alejandra Garrido

Co-Director: Dr. Sergio Firmenich

Índice

Índice	1
Agradecimientos	7
1. Introducción	8
1.1 Motivación	8
1.2 Objetivo	11
1.3 Organización de la tesina	11
Capítulo 2	11
Capítulo 3	11
Capítulo 4	11
Capítulo 5	12
Capítulo 6	12
Capítulo 7	12
Capítulo 8	12
2. Marco teórico y trabajos relacionados	13
2.1 Background	13
Usabilidad	13
Utilidad	14
Accesibilidad	14
Refactoring	15
Web refactoring	16
Scripts de usuario, manejador de scripts y extensiones al navegador web	17
Crowdsourcing	18
A/B testing	18
2.2 Otros trabajos relacionados	20

2.2.1 Mejoras a aplicaciones web a través de la comunidad de usuarios	20
Social Accessibility: Achieving Accessibility through Collaborative Metadata Authoring	21
Social4all: Definition of specific adaptations in Web applications to improve accessibility	21
2.2.2 Manejo de comunidades de crowdsourcing	23
Reputation Management in Crowdsourcing Systems	23
Incentives and Rewarding in Social Computing	24
2.2.3 Crowdsourcing para el testeo de aplicaciones web	25
A novel approach to collaborative testing in a crowdsourcing environment	25
3. Diseño arquitectural	28
3.1 Arquitectura básica de la solución	29
3.2 Definiciones tecnológicas	30
3.3 Entidades y funcionalidades	30
3.3.1 Usuario	30
3.3.2 Problema	31
3.3.3 Proceso de evaluación	32
3.3.4 Solución	34
3.4 Detalle de las funcionalidades principales	36
3.4.1 Asociación y desasociación de problemas	36
3.4.2 Inyección de soluciones	37
3.4.3 Votar solución	40
3.4.4 Votar para iniciar un nuevo proceso de evaluación	41
4. A/B testing	43
4.1 Arquitectura básica de un sistema de experimentación de A/B Test	43
4.1.1 El algoritmo de aleatorización	44
Cached pseudorandom	44
Hash and partition	45
4.1.2 El método de asignación	46

División de tráfico	46
Reescritura de página	47
Asignación del lado del cliente	48
Asignación del lado del servidor	49
4.1.3 El camino de los datos	50
Captura	50
Usar un mecanismo de captura de métricas existente:	50
Registro en archivos de logs locales:	50
Empleo de un servicio especializado:	51
Análisis	51
4.2 Nuestro proceso de evaluación.	51
4.2.1 Nuestro proceso de evaluación como un proceso continuo de A/B Test	52
Objetivos, problemas y oportunidades de mejora	52
Experimentos y variantes	53
Población y muestra	53
4.2.2 Componentes de arquitectura básica de A/B Test	53
Nuestro algoritmo de aleatorización	53
Nuestro mecanismo de asignación	54
Nuestro sistema de recolección de métricas	54
4.2.3 Evaluación de resultados.	55
5. Administración de la comunidad	56
5.1 Evolución de los miembros	56
5.1.1 Privilegios	57
5.1.2 Privilegios democráticos o autocráticos	59
5.1.3 Reputación	61
6. API	66
6.1 Detalle de la API desarrollada	67
6.1.1 Autenticación	67

6.1.2 Llamadas a la API	68
Ver perfil de usuario	68
Ver etiquetas de interés para el usuario	69
Ver problemas asociados	70
Ver soluciones asociadas	71
Ver problema	71
Ver etiquetas de problema	72
Crear problema	73
Actualizar problema	74
Sugerencias de problemas	74
Asociación a problemas	75
Desasociación de problemas	76
Ver proceso de evaluación	76
Ver solución	76
Ver script solución	77
Crear solución	77
Actualizar solución	78
Votar para crear un nuevo proceso de evaluación	78
Votar solución	79
Ver voto	81
Iniciar proceso de evaluación	81
Finalizar proceso de evaluación	82
Navegar	83
6.2 Potenciales usos de la API	87
7. Uso de la aplicación	88
7.1 Aplicación web	88
7.1.1 Registración	88
7.1.2 Inicio de sesión	89

7.1.3 Reportar un problema	89
7.1.4 Listado de problemas	90
7.1.5 Ver problema	91
7.1.6 Ver proceso de evaluación	92
Aceptando soluciones	93
Evaluando soluciones	94
Finalizado	95
7.1.7 Agregar nueva solución	97
7.1.8 Privilegios	98
7.1.9 Perfil de usuario	99
Creación de tokens de acceso	100
7.2 Extensión de Google Chrome	102
7.2.1 Opciones de la extensión	102
7.2.2 Extensión en ejecución	103
8. Conclusiones y trabajo a futuro	105
8.1 Conclusiones	105
8.2 Contribuciones	106
8.2.1 Contribuciones relacionadas a la definición de procesos de trabajo	106
8.2.2 Contribuciones relacionadas a las herramientas desarrolladas	107
8.3 Limitaciones	107
8.4 Trabajos futuros	108
8.4.1 Herramientas de incentivo a la participación	108
8.4.2 Herramientas de moderación	109
8.4.3 Otras mejoras a la comunidad	110
8.4.4 Mejoras a la extensión del navegador	110
8.4.5 Herramientas para el desarrollo de soluciones	111
8.4.6 Propuestas de mejora de seguridad	112
Referencias bibliográficas	114

Anexos	115
1. Extensiones de Google Chrome	116
1.1 Conceptos generales	116
1.2 Interacción con el usuario	117
1.3 Arquitectura de las extensiones	120
Background pages	120
Content scripts	121

Agradecimientos

Queremos darle las gracias a todos los profesores y ayudantes de nuestra facultad quienes, en el transcurso de la carrera, nos compartieron sus conocimientos y experiencia, lo cual, hoy en día, nos permite haber concluido esta tesina.

En especial, queremos darle las gracias a Alejandra Garrido y Sergio Firmenich, quienes nos dieron la oportunidad de llevar a cabo este trabajo. Ellos nos acompañaron a lo largo del mismo y nos propusieron una gran cantidad de sugerencias, siempre con la mejor predisposición y, principalmente, paciencia.

1. Introducción

1.1 Motivación

Desde hace ya varios años, las aplicaciones web tienen un papel cada vez más importante en nuestras vidas. Las mismas se integran en nuestro día a día de diversas maneras, como pueden ser la forma en la cual accedemos al correo electrónico, escuchamos música, leemos noticias, compartimos opiniones, etc. Con el tiempo, las tecnologías web han evolucionado para facilitar cada vez más todas estas actividades, abarcando un público que es día a día más variado, significativo y demandante.

Hoy en día ya no es suficiente con presentar al usuario final la información que necesita sino que la misma debe ser presentada de una manera clara y concisa, debe ser fácil de navegar, no debe generar tiempos de espera, debe poder accederse desde distintos dispositivos, etc. De estos y más aspectos se encarga la usabilidad.

La usabilidad y la experiencia de usuario se han convertido en un importante campo de investigación. Tal es el caso, que recientemente se ha llegado a acuñar el término "usability smell" [Garrido 2011] para referenciar los problemas de usabilidad que se presentan en una aplicación el cual se inspira en el término "bad smell" empleado para denominar síntomas que apuntan a un problema de código. En otro estudio, Jakob Nielsen [Nielsen 2006] realizó una priorización de los problemas de usabilidad en base a su frecuencia, impacto y persistencia para luego agruparlos en categorías. Del mismo concluyó que los principales problemas con los que se topan los usuarios son: la capacidad para navegar la información a fin de alcanzar específicamente la que desean acceder, la facilidad para buscar la misma y la forma en la cual se presenta.

Sin embargo, por más que podamos detectar, priorizar e incluso corregir los problemas de usabilidad más importantes, aún así es imposible lograr la satisfacción de todos los distintos tipos de usuarios debido a su enorme variedad en términos de conocimientos informáticos, edad, género o, incluso, del uso específico que esperan de una aplicación web. Es por ello que, recientemente, ha cobrado cada vez más peso la navegación aumentada, que es básicamente una forma de mejorar la experiencia de navegación de los usuarios a

través modificaciones en las aplicaciones web que resultan en el enriquecimiento de la información que contienen o de la forma en la cual se presenta.

Una característica importante de la navegación aumentada es que la misma se realiza del lado del cliente web y no desde el servidor. Es decir, se logra mediante extensiones al navegador web que permiten la ejecución de scripts de usuario que realizan la manipulación del contenido original. Esto da lugar a que las mejoras buscadas no necesariamente dependan de los propietarios de los sitios web sino que puedan quedar en manos de los usuarios finales.

La creciente popularidad de las herramientas de aumentación de la navegación web ha dado pie a la creación de comunidades en base a distintas tecnologías específicas, por ejemplo, en torno a la extensión de navegador usada para insertar scripts de usuario en el navegador Firefox, denominado GreaseMonkey (o TamperMonkey en el caso de Chrome) [<https://greasyfork.org/en>], así como la publicación de libros tipo cookbooks [Pilgrim 2005] con soluciones en las cuales cualquier usuario con mínimos conocimientos de programación podría basarse para mejorar su experiencia. La popularidad de estas comunidades permite darle un mínimo grado de confiabilidad a las soluciones allí expuestas, gracias a la permanente auditoría del resto de los participantes mediante sus mecanismos de calificación de las contribuciones.

En trabajos anteriores [Garrido 2013], se han definido los refactorings Web como modificaciones menores que mejoran los aspectos observables de las aplicaciones Web, manteniendo toda su funcionalidad. Este tipo de refactorings puede incluso abordar problemáticas relacionadas a la accesibilidad, las cuales pueden diferir considerablemente de acuerdo a las limitaciones que tenga cada uno de los usuarios.

Estos refactorings pueden resultar en una significativa mejoría para cierto grupo de usuarios, que no necesariamente representa la mayoría dado que no todos los usuarios tienen las mismas necesidades o preferencias. Para lograr una experiencia más personalizada, sería deseable que cada usuario pueda aplicar los refactorings que considere más apropiados para su caso. También a tener en cuenta es que por definición los refactorings se limitan a modificaciones que mantengan la funcionalidad original. Para poder atender a una base más amplia de usuarios, sería necesaria la extensión o adición de

funcionalidades, por lo que los conceptos de web augmentation o script de usuario son más abarcativos y por ende más apropiados para nuestros objetivos.

Contemplando, por un lado, la gran variedad de usuarios existentes y, por otro, la existencia de comunidades de desarrolladores y usuarios de scripts, se considera que aplicar la técnica de crowdsourcing [Doan 2011] podría resultar en un beneficio mutuo. Podemos definir en pocas palabras el crowdsourcing como una técnica que nos permite obtener resultados o soluciones a problemas a través de pequeñas colaboraciones de grandes grupos de personas.

La aplicación de crowdsourcing dará lugar a que los usuarios puedan obtener soluciones a los problemas de usabilidad que se les presenten recurriendo a la ayuda de las comunidades de desarrolladores existentes. Dichas comunidades verían facilitado el acceso de los usuarios a sus soluciones, con lo que tendrían un mayor crecimiento y desarrollo, el cual es uno de sus principales objetivos.

En este contexto, se considera esencial proveer un mecanismo que organice las contribuciones en torno a necesidades o problemas específicos conocidos, los cuales, a su vez, puedan ser reportados por la comunidad de usuarios que desean mejorar su experiencia de navegación web.

Esta asociación entre problema y solución fue encarado por otros trabajos [Firmenich 2015] mediante un enfoque negociado entre el dueño o encargado de un sitio web y el desarrollador de la solución. Sería deseable encontrar un mecanismo que nos permita prescindir de este requisito que limita el conjunto de sitios web factibles de ser mejorados. Para esto, se considera viable involucrar nuevamente a la comunidad en la verificación de las distintas soluciones que se presenten para un mismo problema. La implementación de un mecanismo de A/B Testing [Siroker 2013] permitirá justamente lograr este objetivo empleando como métrica determinante el feedback de los usuarios que participen, los cuales tendrán la posibilidad de indicar la eficacia de la solución que fue aplicada durante su navegación en el sitio web.

1.2 Objetivo

Desarrollar un sistema que permita la mejora de la experiencia de usuario de aplicaciones web, incluyendo mejoras a problemas de usabilidad y accesibilidad, apoyándonos en los siguientes conceptos fundamentales:

- Hacer uso de un mecanismo de inyección de scripts de usuario basado en una extensión al navegador web, permitiendo de esta manera afectar a cualquier sitio web, sin necesidad de colaboración de los dueños de dicho sitio en todo el proceso.
- Construir una comunidad donde los usuarios reporten los problemas detectados, propongan los scripts de usuario que resolverían dichos problemas, y evalúen las soluciones dando feedback sobre las mismas.
- Evaluar las soluciones propuestas mediante procesos de A/B Testing donde el feedback de los usuarios es el factor clave para determinar la solución aceptada.
- Promover la extensibilidad del sistema exponiendo sus funcionalidades a través de una API, facilitando de este modo su interacción con otras aplicaciones.

1.3 Organización de la tesina

Capítulo 2

Exponemos el marco teórico de esta tesina: definimos los conceptos esenciales y presentamos trabajos relacionados que fueron analizados para la realización de esta tesina.

Capítulo 3

Presentación de nuestra propuesta. Definimos la arquitectura del sistema, entidades y funcionalidades desarrolladas.

Capítulo 4

Profundizamos en los aspectos de aplicación de A/B Testing en nuestro trabajo.

Capítulo 5

Profundizamos en los aspectos de Crowdsourcing y administración de comunidades en nuestro trabajo.

Capítulo 6

Detallamos la API REST desarrollada para interactuar con nuestro sistema, elemento clave para asegurar la extensibilidad del mismo.

Capítulo 7

Se presenta la aplicación desarrollada a modo de manual de usuario. Explicamos su funcionamiento y mostraremos cómo interactuar con la misma para acceder a cada una de sus funcionalidades, con capturas de pantalla para facilitar su reconocimiento.

Capítulo 8

Se describen las conclusiones de este trabajo y futuras direcciones.

2. Marco teórico y trabajos relacionados

Este capítulo describe, en primera instancia, varios conceptos que fueron estudiados y aplicados para la realización de este trabajo. Como se describió en la Introducción, este trabajo se focaliza en la creación de una herramienta que facilite la colaboración entre usuarios para el desarrollo de mejoras de usabilidad, utilidad y accesibilidad de las aplicaciones web, de modo que primeramente se describen estos conceptos. Luego se describen los conceptos de refactoring y Web refactoring, como mecanismos para realizar estas mejoras de calidad de una aplicación. Se continúa describiendo a los scripts de usuario como otros mecanismos para hacer modificaciones a una página web del lado del navegador, que pueden ser creados y compartidos por la misma comunidad de usuarios. Dado que este trabajo también apunta a que la comunidad de usuarios pueda crear y compartir mejoras de usabilidad, se describe el concepto de Crowdsourcing. Finalmente el desarrollo propuesto está inspirado en los procesos de A/B testing cuyo uso se ha popularizado en los sistemas web, por lo que también se explica dicho concepto.

La segunda parte del capítulo describe otros trabajos que por distintos motivos se relacionan con lo desarrollado en esta tesis.

2.1 Background

Usabilidad

La usabilidad hace referencia a la calidad de la experiencia del usuario al interactuar con productos o sistemas, incluyendo sitios web, software y aplicaciones. Es importante destacar que la usabilidad no se mide sólo en una dimensión sino que es una combinación de los siguientes factores [Nielsen 2012]:

- **Facilidad para recordar:** con qué facilidad puede, un usuario que ya haya usado el sistema anteriormente, recordar lo suficiente como para poder usar el sistema efectivamente en el futuro.

- Severidad y frecuencia de errores: con qué frecuencia el usuario comete errores mientras usa el sistema, si esos errores le impiden realizar la operación deseada o no y cuánto tiempo tarda en recuperarse de la situación inesperada.
- Facilidad de aprendizaje: qué tan rápido puede aprender a operar el sistema un usuario que nunca haya usado el mismo.
- Eficiencia de uso: qué tan rápido puede realizar tareas satisfactoriamente un usuario que ya conoce el sistema.
- Grado de satisfacción subjetiva: qué tanto le ha gustado al usuario el sistema.

Como vimos en la introducción, la usabilidad, o mejor dicho, la falta de la misma, es uno de los principales motivadores de este trabajo.

Utilidad

La utilidad es un atributo de calidad, al igual que la usabilidad. Sin embargo, la utilidad se reduce a que un determinado producto cumpla con la función que se necesita [Nielsen 2012]. Se diferencia de la usabilidad en que esta última abarca aspectos más blandos y subjetivos, es decir, la usabilidad se centra más en el cómo se opera el sistema, mientras que la utilidad se enfoca estrictamente en qué se puede hacer con el sistema.

Cabe destacar que tanto la utilidad como la usabilidad son atributos clave al momento de evaluar la calidad de un sistema y permiten determinar qué tan útil es.

Accesibilidad

La accesibilidad es una característica que indica que todas las personas, incluyendo personas con alguna discapacidad puedan acceder de manera directa o asistida a algún lugar, un servicio o al uso de un objeto. En particular, la accesibilidad web apunta a que personas con discapacidades puedan percibir, interpretar, navegar e interactuar con la web, accediendo a todo el contenido y funcionalidades provistas por un sitio.

En particular, las necesidades especiales que se deben considerar para garantizar la accesibilidad web son del tipo:

- Visuales: en distintos grados, desde escasa visión a ceguera total, incluye problemas para distinguir colores.

- Motrices: dificultad o inhabilidad para usar las manos; temblores, problemas de motricidad fina.
- Auditivas: sordera o deficiencias auditivas.
- Cognitivas: Problemas de aprendizaje, atención y enfoque; capacidad de memoria y habilidades lógicas.

En 1997 la World Wide Web Consortium (W3C) formó un proyecto para fomentar prácticas que mejoren la accesibilidad web llamado Web Accessibility Initiative (WAI). El mismo publicó por primera vez en 1999 una guía de buenas prácticas de accesibilidad web denominado Web Content Accessibility Guidelines (WCAG 1.0) la cual fué sucesivamente revisada y la última revisión aprobada en 2008 (WCAG 2.0).

Esta guía está organizada en base a 4 principios cuyas siglas en inglés forman el acrónimo POUR (Perceivable, Operable, Understandable, Robust)

1. Perceptible: La información o contenido y los elementos de la interfaz de usuario deben ser presentados a los usuarios de manera de que los mismos puedan percibirlos.
2. Operable: El usuario debe poder interactuar con los componentes de la interfaz de usuario y de navegación del sitio, sea de manera directa o mediante elementos de asistencia.
3. Entendible: El contenido y la interfaz de usuario deben ser claros y evitar confusiones y ambigüedades
4. Robusto: El contenido y la interfaz de usuario debe poder ser interpretado y accedido de manera confiable por una gran variedad de agentes de usuario (browsers), incluyendo tecnologías de apoyo (lectores de pantalla, terminales braille, reconocimiento de voz, etc)

De acuerdo al nivel de conformidad con estos guidelines, el documento de WCAG 2.0 clasifica a los sitios en los niveles (de menor a mayor conformidad) A, AA y AAA.

Refactoring

Se puede definir refactoring como el proceso de cambiar un sistema de software de manera tal que no se altera el comportamiento externo del código pero efectivamente se

mejora su estructura interna [Fowler 1999]. Los refactorings conllevan grandes beneficios en términos de mantenibilidad así como de extensibilidad, ya que es más sencillo el proceso de corrección de errores y de desarrollo de nuevas características si el código es fácil de entender.

El proceso de refactoring suele estar motivado por la detección de bad smells o también llamados code smells. Un bad smell es una indicación superficial que suele corresponderse a un problema más profundo en un sistema [Fowler 2006].

Algunos ejemplos frecuentes de bad smells suelen ser métodos o funciones muy extensos o métodos distintos pero que llevan a cabo prácticamente la misma función. Estos problemas comunes suelen tener sus correspondientes refactorings. En el caso de los métodos extensos se puede intentar explotar esos métodos en varios más cortos que, en su conjunto, mantengan la misma funcionalidad que el original. En el caso de los métodos similares, se puede intentar extraer la lógica que tengan en común a otros métodos que puedan ser reusados por los originales, evitando así la duplicidad del código.

Web refactoring

En trabajos anteriores [Garrido 2011] se apuntó a que los refactorings deben no sólo relacionarse a los bad smells que apuntan a resolver sino también a los atributos de calidad que intentan mejorar. De esta manera, podemos considerar que un refactoring podría apuntar a incrementar algún atributo que no sea interno, como podrían ser aquellos relacionados a la estructura del código, y enfocarse en los externos, como podría ser la usabilidad.

Del mismo modo que los bad smells referencian a puntos de mejora en el código, se definen los usability smells, como indicadores de posibles problemas ligados a la usabilidad de una aplicación web [Garrido 2011].

Se puede entender fácilmente el concepto de web refactoring si se considera, por ejemplo, el grado de accesibilidad de los sitios web. Hacer que una interfaz web sea más accesible, no necesariamente implica una reestructuración interna del código ni modifica su

contenido o funcionalidad. Aún así es posible apreciar e, incluso, medir el grado de mejora alcanzado.

Podemos entonces definir a los web refactorings como modificaciones menores que mejoran los aspectos observables de las aplicaciones Web, manteniendo toda su funcionalidad [Garrido 2013].

Scripts de usuario, manejador de scripts y extensiones al navegador web

Un script de usuario es en pocas palabras un programa para modificar páginas web. El mismo es ejecutado en el contexto de una extensión del navegador. Usualmente se los emplea para agregar funcionalidad a una página web, cambiar su apariencia, hacerla más fácil de usar o incluso eliminar elementos que distraigan o dificulten su usabilidad.

Un manejador de scripts de usuario es una extensión del navegador que provee una interfaz para administrar scripts y cuya principal utilidad es la ejecución de los scripts (instalados en dicha extensión) en las páginas web a las que el usuario accede, al momento en el que estas son accedidas. Ejemplos de manejadores de scripts de usuarios son GreaseMonkey, TamperMonkey o ViolentMonkey.

La popularización de los scripts de usuario dio origen a la aparición de repositorios donde los usuarios comparten sus creaciones. Estos repositorios permiten que los usuarios suban sus scripts, de manera que estén inmediatamente disponible para ser descargados por otros usuarios. Estos repositorios normalmente tienen algún mecanismo de evaluación de los mismos, de manera que los usuarios puedan expresar su opinión respecto a su utilidad o seguridad y la misma pueda ser tomada en cuenta por otros usuarios para decidir la conveniencia de su descarga y utilización. El ejemplo mas activo en la actualidad es [<https://greasyfork.org/en>], otros ejemplos que han caido en desuso [<https://openuserjs.org/>] [<http://userscripts-mirror.org/>]

El empleo de scripts de usuario permite la navegación aumentada (web augmentations). A su vez, son una pieza clave para la implementación de client side web refactorings [Garrido 2013], los cuales se definen como web refactorings que corren en el entorno del navegador web, una vez que la página original fue accedida. Los client side web refactorings

le dan al usuario la posibilidad de aplicar los cambios de un refactoring web customizado a sus necesidades.

Mencionamos que los scripts de usuario corren en el contexto del navegador web. Esto es posible gracias al empleo de browser extensions (extensiones del navegador). Una browser extension es un plugin que se instala en el navegador y extiende su funcionalidad. Cada navegador tiene un mecanismo particular para el desarrollo e instalación de extensiones, y proveen una interfaz con la cual el código de la extensión interactúa para cumplir su objetivo, sea este realizar algún cambio en la interfaz de usuario, o modificar el contenido de las páginas accedidas.

Crowdsourcing

La definición más aceptada de crowdsourcing fue dada por Howe y Robinson en 2006 [Howe2006a] [Howe2006b], y lo define como el acto de externalizar trabajo que sería comúnmente realizado por empleados y delegarlo a una comunidad de personas, las cuales pueden participar colaborativamente en una solución, o aportar soluciones individualmente. El aspecto clave es el formato “open call” de la solicitud, que permite la participación abierta a todos los que así lo desean, y la disponibilidad de una gran comunidad de potenciales colaboradores.

En el caso de crowdsourcing de ingeniería de software, el formato open call permite reclutar ingenieros de software a escala global a través de internet, y se delegan tareas de ingeniería de software tal como recolección de requerimientos, arquitectura de sistema, diseño e implementación de código, así como el testeado del resultado.

Si bien más adelante profundizaremos en el tema, hay tres instancias en las que este trabajo se apoyaría en el mecanismo de crowdsourcing: por un lado, la invitación a que cualquier usuario de aplicaciones web pueda reportar problemas de usabilidad que experimenta con las mismas; en segundo lugar, una vez registrado el problema en el sistema, los usuarios con conocimiento de programación de scripts pueden proponer soluciones a los mismos también en forma cooperativa; y en tercer lugar, una vez propuestas las soluciones, evaluar las mismas también a través de la comunidad.

A/B testing

El concepto de A/B Testing predata por mucho la revolución que generó su utilización en internet [Kohavi 2009]. En pocas palabras, es un experimento binario donde se desea tomar alguna decisión entre el candidato A y el candidato B. Según su área de aplicación estos candidatos pueden ser llamados tratamiento, placebo, control, variante, o simplemente A y B. Estos tests binarios también pueden ser generalizados en tests multivariantes, y esta generalización se la conoce como split tests.

En particular a nosotros nos interesa su aplicación en sistemas web. Específicamente en este caso, A / B Testing es un proceso por el cual se someten a experimentación una o mas variantes de un sistema o pagina web. Estas variantes son testeadas en tiempo real por usuarios reales bajo condiciones controladas con el objetivo de determinar su impacto en cierto criterio de evaluación.

La realización de estos experimentos posibilitan a la empresa a tomar decisiones en base a datos reales y verificables, y no tanto en base a un diseño teórico y muy especialmente, evitar tomar lo que se conoce como HiPPO (highest paid person's opinion - opinion de la persona de mayor sueldo) como irrefutable. Múltiples autores han expuesto sus experiencias en los últimos años, en compañías del tamaño de Amazon [Linden 2006], NASA, o Microsoft.

A/B Testing no es necesariamente la solución para cualquiera, y deben darse ciertas condiciones para maximizar su utilidad [Kohavi 2017] entre ellas:

- La organización desea tomar decisiones en base a datos, y tiene un criterio de evaluación formal
- Los experimentos deseados pueden realizarse de manera sencilla, rápida y sus resultados confiables
- Las personas no son perfectas en el cálculo o predicción de valor de una idea.

La estructura general de un experimento es sencilla: se propone realizar el experimento con una variante A en comparación con el estado actual del sistema, el llamado control o simplemente B.

- Se divide durante un tiempo determinado el tráfico de usuarios del sitio web hacia cada una de las variantes (por ejemplo, 50% y 50%)
- Durante el periodo del experimento, se capturan métricas de interacción de los usuarios con cada una de las variantes.
- Estas métricas son claves para calcular el impacto de la variante y determinar su utilidad y puede ser, por ejemplo, el porcentaje de compras sobre visitas
- Al finalizar el experimento, se debe determinar si la diferencia en las métricas tomadas son estadísticamente significantes o el impacto medido es una variación esperable en el tráfico normal del sitio.

En este trabajo se propone utilizar un mecanismo inspirado en procesos de A/B testing para evaluar las distintas soluciones contribuidas por los usuarios para resolver un problema de usabilidad particular y seleccionar de entre ellas la mejor tomando como referencia el feedback de los usuarios que participaron en el proceso de evaluación de las mismas.

2.2 Otros trabajos relacionados

A continuación se describen trabajos de investigación que están relacionados en 3 áreas diferentes a la propuesta presentada en este trabajo: (i) trabajos que proponen mejoras a la experiencia del usuario en páginas web a través de una comunidad de usuarios; (ii) trabajos que se enfocan en el manejo de comunidades de crowdsourcing, y (iii) trabajos que proponen crowdsourcing de pruebas de usuario.

En cada trabajo se describen las diferencias principales con nuestra propuesta.

2.2.1 Mejoras a aplicaciones web a través de la comunidad de usuarios

Como desarrolladores de un sitio web, es imposible lograr la satisfacción de todos los distintos tipos de usuarios debido a su enorme variedad en términos de conocimientos informáticos, edad, género o, incluso, del uso específico que esperan de una aplicación web.

Nuestra intención en este trabajo es presentar un mecanismo por el cual la comunidad de usuarios de un sitio web pueda contribuir a la mejora del mismo para lo cual analizamos artículos relacionados al tema. A continuación describimos dos de los más relevantes.

Social Accessibility: Achieving Accessibility through Collaborative Metadata

Authoring

El trabajo de Asakawa et al., "Social Accessibility: Achieving Accessibility through Collaborative Metadata Authoring" [Asakawa 2008], es una aproximación basada en la colaboración de usuarios en la resolución de problemas de accesibilidad. Como diferencias principales a nuestra propuesta se encontraron, en primer lugar el hecho de que efectivamente la misma no abarca problemas de usabilidad sino sólo de accesibilidad. Más específicamente, se enfoca en el agregado de metadatos a secciones de páginas Web que puedan ser procesados por lectores de pantalla. Estos metadatos permiten identificar un nodo del DOM y contienen el texto que debe ser leído por el lector de pantalla.

Por otro lugar, la arquitectura que se propuso requiere de una extensión del navegador así como una extensión en el lector de pantalla. Esto tiene la complejidad de que los usuarios no sólo deberán usar un determinado navegador sino que también deberán usar un determinado lector de pantalla (o aquellos los que se les haya dado soporte).

Nuestra propuesta es más abarcativa ya que va más allá de las limitaciones de accesibilidad y permite realizar cualquier tipo de modificación que se pueda efectuar del lado del cliente en una página Web. Esto puede incluir problemas de accesibilidad, cosméticos, de usabilidad o incluso agregar, modificar o eliminar características enteras que están faltando en una página Web. Lo único necesario en nuestro caso es la extensión del navegador.

Social4all: Definition of specific adaptations in Web applications to improve accessibility

De manera similar a la propuesta anterior, el artículo "Social4all: Definition of specific adaptations in Web applications to improve accessibility" [Crespo 2016], propone la solución de problemas de accesibilidad a través de una comunidad de usuarios. Hasta donde conocemos, no existen plataformas como las que se proponen en esta tesina para realizar comunitariamente mejoras de usabilidad. Sin embargo, la propuesta de Crespo et al. un poco más genérica que el trabajo de Asakawa et al. Es decir, no se enfoca estrictamente en la inserción de metadatos sino que también permite una manipulación completa del DOM.

Esta aproximación tiene una arquitectura que requiere de un servidor Web local. El usuario hace la navegación directamente desde su navegador a través de este servidor local. Este servidor funciona, a su vez, como un cliente Web que se encarga de obtener la página solicitada por el usuario y una vez hecho esto, le aplica las transformaciones pertinentes. Una vez adaptada la página, se la presenta al usuario.

Las adaptaciones son estrictamente de accesibilidad. El sistema conoce diversos problemas de accesibilidad y tiene la capacidad de identificarlos. El usuario puede sugerir propuestas de solución a esos problemas (adaptaciones). Al conjunto de adaptaciones para un determinado sitio se lo denominó perfil de adaptación. Los perfiles de adaptación son públicos y cualquier usuario puede hacer uso de los mismos. Sin embargo, sólo el usuario que lo creó lo puede modificar. De esta manera, todos los usuarios se ven sujetos a un esquema de "todo o nada" donde deben elegir estrictamente un conjunto de adaptaciones o deben crear su propio perfil. Esto es una diferencia con nuestra propuesta donde el enfoque es por adaptación y no sobre un conjunto de adaptaciones.

Otra diferencia significativa que tiene esta propuesta con la nuestra es la existencia del servidor Web. Dejando de lado cualquier complejidad asociada al uso de un servidor Web por usuarios que no son avanzados, el servidor debe realizar un procesamiento significativo de la página Web que obtiene del navegador. Esto involucra, por ejemplo, modificar los encabezados de los requerimientos HTTP o incluso todas las URLs de la página obtenida a fin de que la navegación pase en su totalidad por el servidor Web local. Estas modificaciones pueden resultar en problemas en la ejecución de código JavaScript del lado del cliente, como podría ser el caso del armado de URLs dinámicamente. De esa manera, incluso se podrían llegar a armar direcciones que eludan al servidor Web local y terminen accediendo de manera directa al sitio Web que se desea adaptar.

Una consideración adicional a tener en cuenta de esta última propuesta es que el usuario puede navegar y obtener las páginas Web adaptadas desde cualquier navegador y sin la necesidad de que exista una extensión desarrollada para el mismo. Este es un punto que nuestra solución no contempla ya que requiere que se use específicamente Chrome, que es el navegador para el cual se está programando la extensión. Sin embargo, debido a la API implementada del lado del servidor, la extensión del soporte a otros navegadores sólo

debería implicar el desarrollo de la extensión en sí misma y no debería ser necesario ninguna modificación en el servidor.

2.2.2 Manejo de comunidades de crowdsourcing

Uno de los objetivos de nuestro trabajo es explotar la capacidad de la comunidad para que pueda autorregularse, es decir, reducir la necesidad de intervenciones de tipo burocráticas del sistema, permitiendo concentrar los esfuerzos en la generación de soluciones de usabilidad. Por este objetivo de autorregulación, y para conocer la mejor forma de motivar la participación de la comunidad, estudiamos trabajos relacionados con manejo de reputaciones e incentivos en una comunidad de crowdsourcing. Se describen a continuación dos artículos que nos han servido de inspiración para desarrollar nuestra plataforma.

Reputation Management in Crowdsourcing Systems

Este paper propone un modelo de manejo de reputación que facilita al solicitante la selección de un candidato de un pool de crowd workers.

El empleo de un modelo de reputación como indicador confiabilidad de los trabajadores es un método popular en sistemas de crowdsourcing. Para calcular la reputación se tienen en cuenta diversas fuentes, como feedback de los miembros de la comunidad, contribuciones particulares o propiedades asociadas a la tarea realizada, como el rédito obtenido o el tiempo en el que se realizó.

La credibilidad del modelo de reputación es una cuestión clave en un sistema de crowdsourcing, y carece de una solución trivial. Evaluadores deshonestos pueden manipular la reputación de trabajadores de diversas maneras, generando evaluaciones que no estén acordes al trabajo realizado. Para evitar manipulaciones, propone incorporar una métrica que defina qué tan justo fue un evaluador al realizar la evaluación. Analiza el comportamiento del evaluador para detectar incoherencias en sus evaluaciones, analiza también relaciones individuales evaluador-evaluado para detectar evaluadores que solo perjudican a un grupo reducido de evaluados, también considera los consensos generados para detectar evaluaciones atípicas y minimizar su impacto en la reputación del trabajador.

Incentives and Rewarding in Social Computing

Este paper analiza mecanismos de incentivos y recompensas como método para promover actitudes deseables y mejorar la performance de trabajadores participantes de un sistema de crowdsourcing.

Para ello, realiza una clasificación de los mismos según literatura previa y su aplicación en empresas tradicionales y expone un relevamiento de los mecanismos aplicados en sistemas de existentes de social computing.

Mucho se ha estudiado sobre este tema desde que surgió junto con los principios de división de trabajo. Su utilidad para alinear los objetivos de la empresa con el de los trabajadores hace que prácticamente toda empresa tradicional haga uso de uno o más de ellos.

Siendo que pretendemos construir una comunidad con capacidad de autorregulación, nos interesa encontrar mecanismos apropiados como elemento de motivación para participar y contribuir, así como para desalentar prácticas no deseadas.

Entre los mecanismos de incentivo más frecuentes se encuentran:

- Pago según performance: compensación proporcional a la contribución. Aplicable cuando existe un mecanismo de evaluación cuantitativo.
- Cupos y bonuses: compensación según el cumplimiento de ciertas metas según métricas de performance en un intervalo de tiempo
- Evaluación relativa: compensación según performance en comparación relativa a sus pares.
- Promoción: propone un set limitado de compensaciones y efectúa una competencia entre pares para conseguirlas.
- Compensación por equipos: empleado cuando es muy compleja la evaluación de la contribución de un individuo en un contexto de trabajo grupal. Generalmente combinada internamente en el equipo con otro mecanismo.
- Incentivos psicológicos: se apoyan en características inherentemente humanas. La manera en que el agente percibe el incentivo es determinante para su efectividad, por ejemplo, un incentivo que se apoye en la percepción de contribución al bien

común pierde efectividad en una cultura altamente individualista. En dicha sociedad, en contraposición, podría ser efectivo la promoción de la competencia.

Los mecanismos de incentivo precedentes pueden ser considerados compuestos de tres elementos: un método de evaluación, una condición de disparo de incentivo, una recompensa o acción compensatoria.

Métodos de evaluación individual:

- Cuantitativo: se basa en atributos medibles de la contribución realizada.
- Subjetivo: cuando no se puede evaluar la contribución sin la intervención humana.

Métodos de evaluación grupal:

- Evaluación entre pares: los miembros de un grupo evalúan las contribuciones de otros miembros del grupo.
- Evaluación indirecta: se evalúa la persona a través de sus contribuciones y las relaciones que generan con otros miembros.

Como describiremos en el capítulo de manejo de comunidad de usuarios y crowdsourcing, este trabajo utiliza un método cuantitativo como evaluación individual, sumado a una evaluación entre pares.

2.2.3 Crowdsourcing para el testeo de aplicaciones web

[A novel approach to collaborative testing in a crowdsourcing environment](#)

Este artículo propone un modelo de testeo colaborativo en un entorno de crowdsourcing. Una aplicación web actual es un sistema complejo con muchas piezas que interactúan para proveer la funcionalidad deseada. El testeo de dichas aplicaciones debe consistir de un conjunto de casos de prueba que provean la mayor cobertura posible, teniendo en cuenta además la gran heterogeneidad de clientes. El modelo de crowdsourcing se presenta como una oportunidad de acceder a los recursos humanos necesarios para ejecutar los casos de pruebas.

Lo interesante además de este artículo es que propone un modelo de asignación de casos de prueba a testers. Es importante notar que el proceso de asignación de pruebas a testers

es complejo ya que debe considerar la diversidad de funcionalidades a testear y su importancia, los casos de prueba existentes y su capacidad de cobertura, la disponibilidad de tiempo de cada tester individual así como también su capacidad y compromiso. Por ello, este proceso puede verse como un problema de asignación de trabajo; un problema considerado NP completo formulable como un problema ILP (integer linear programming). Dada la complejidad del problema, el cálculo de la resolución óptima no es factible de realizar en tiempo aceptable cuando se trata de proyectos de gran escala, por lo que se propone un enfoque basado en cuatro heurísticas para su resolución.

El modelo propuesto consiste en dos fases; el entrenamiento y el testeo. En la fase de entrenamiento se emplean cinco matrices, que representan al sistema, los casos de prueba y los testers y se transforman en una fórmula ILP por medio de un algoritmo de transformación ILP. Las matrices son: cobertura de funcionalidad de cada caso de prueba, tiempo de ejecución de los mismos, objetivo de cobertura de cada funcionalidad, grado de confiabilidad del tester, y disponibilidad del tester. En la fase de testeo se resuelve la fórmula ILP de asignación de casos de prueba a testers mediante heurísticas de selección de casos de prueba.

El modelo fue evaluado mediante el desarrollo de una aplicación web prototipo 'online bookstore' con un set conocido de bugs y puesto a prueba empleando las plataformas de amazon mechanical turk y topcoder. Los resultados obtenidos indican que los tiempos obtenidos con la solución basada en heurísticas se encuentra en el rango de 90% de la solución óptima obtenida mediante la resolución algorítmica del problema NP completo, y reduce en casi un 50% los tiempos comparado con una solución aleatoria.

En este trabajo se consideró este artículo ante la posibilidad de aplicar un algoritmo de asignación similar en distintas etapas del proceso que aquí se propone. En particular, una de estas etapas es en el momento realizar la asignación de problemas introducidos por miembros de la comunidad, a otros miembros que sean capaces de darle solución. Para esto también podría haberse considerado la experiencia anterior, capacidad y compromiso como propone el artículo. En cambio decidimos, respetando los principios de las comunidades open-source, permitir que cada miembro de la comunidad pueda decidir a qué problema darle solución. Otra de estas etapas es la de asignación de soluciones a usuarios que van a

testearlas. En vez de aplicar un algoritmo complejo de asignación decidimos usar el mecanismo de A/B testing que asigna aleatoriamente, porque además es la asignación más razonable dado que no podemos automáticamente inferir la estructura de la solución para asignarla.

3. Diseño arquitectural

Como se explicó en la introducción, este trabajo intenta proveer un mecanismo para la mejora de aplicaciones web, incluyendo mejoras a problemas de usabilidad y accesibilidad, mediante el uso de scripts client-side creados a tal efecto por la comunidad de usuarios y ejecutados por nuestra extensión al navegador, que implementen los cambios necesarios para resolver el problema.

Para comenzar a entender cómo funcionaría esta propuesta, conviene hacer un ligero repaso de cómo funciona la internet en sí misma, y cómo es el flujo de información desde el momento que un usuario intenta acceder a una página web hasta que la misma se muestra en la pantalla de su navegador.

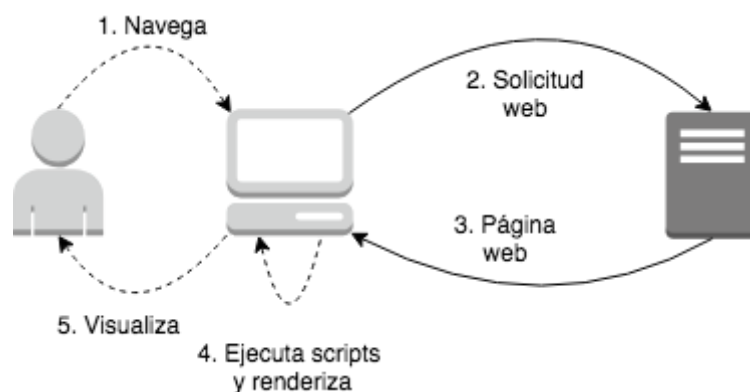


Figura 3.1: Flujo HTTP clásico.

1. El usuario realiza un requerimiento HTTP a un servidor.
2. El servidor (idealmente) responde con el HTML, y enlaces a recursos necesarios para la visualización de la página.
3. El navegador descarga estos recursos y renderiza la página web, ejecutando los scripts necesarios.

Mediante el uso de una extensión del navegador, es posible adicionar a este flujo un paso extra en el cual se realiza una solicitud a nuestro servidor en busca de scripts de usuario que hayan sido desarrollados para mejorar dicha página.

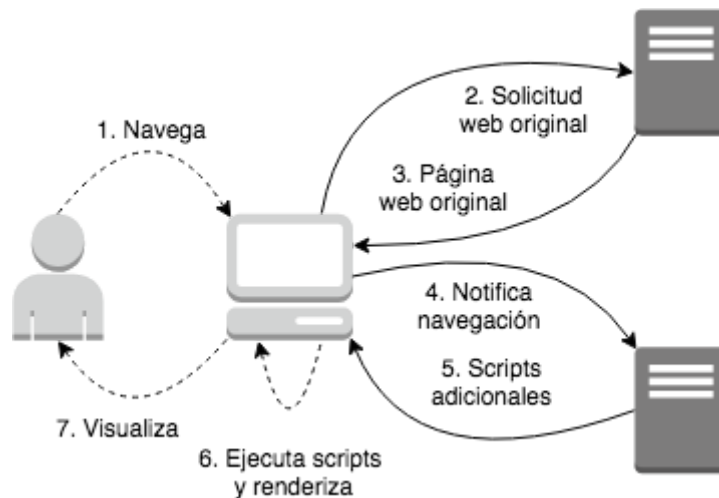


Figura 3.2: Flujo con nuestra extensión activada.

1. El usuario realiza un requerimiento HTTP a un servidor.
2. El servidor (idealmente) responde con el HTML, y enlaces a recursos necesarios para la visualización de la página.
3. El navegador informa a nuestro servidor con la dirección de la página que el usuario está visitando en búsqueda de scripts aplicables.
4. El navegador descarga tanto los recursos originales como los scripts adicionados y renderiza la página web, ejecutando los scripts necesarios, tanto originales como adicionados.

3.1 Arquitectura básica de la solución

Habiendo descrito el flujo en el cual apoyaremos nuestro sistema para inyectar las mejoras necesarias en las páginas que el usuario visita, determinamos que los componentes esenciales son:

- Un front-end para el usuario final, donde se pueda registrar y participar de la comunidad, solicitando o proveyendo soluciones a problemas.
- Una API con la que se puedan realizar las operaciones esenciales que se mencionan debajo de manera sistémica para facilitar la integración con otras aplicaciones.
- Una extensión de Google Chrome que permita la adaptación de páginas visitadas acorde a las soluciones correspondientes.

3.2 Definiciones tecnológicas

Hemos realizado la selección de tecnologías tomando como base aquellas de código abierto, que tengan suficiente documentación y que sean populares, a fin de facilitar el proceso de desarrollo en las mismas así como incrementar la capacidad para que éstas sean extendidas en el futuro.

El lenguaje seleccionado para realizar el front-end web y la API fue PHP, usando el framework Laravel, que ha demostrado ser uno de los frameworks más populares, acorde a las últimas tendencias. Posee un alto nivel de abstracción y facilidad para integración con varios servicios, así como soporte de librerías y componentes de terceros.

Como DBMS seleccionamos a MySQL por la popularidad que tiene pero dejamos abierta la posibilidad de cambiar a otro motor de código abierto como PostgreSQL en caso de que se requiera alguna funcionalidad que MySQL no soporte.

Actualmente estamos haciendo uso del servicio gratuito que provee Heroku para mantener una versión pública y online del sistema. Esta selección está sustentada en la facilidad de uso, las funcionalidades gratuitas y la integración con PHP que provee.

El código lo tenemos administrado en Git, usando como concretamente el servicio provisto por BitBucket, que ha demostrado ser un sistema de versionamiento confiable, flexible y popular. Si bien el código está documentado, en el capítulo 6 se puede apreciar también la documentación esencial para el consumo de una API que permitirá la gestión de las principales entidades del sistema.

3.3 Entidades y funcionalidades

A continuación se describen brevemente las entidades principales en las que se trabajó, junto con las capacidades que tiene cada una.

3.3.1 Usuario

Esta entidad es la abstracción de un miembro de la comunidad. Tiene la capacidad de publicar problemas y soluciones, así como la de controlar los cambios de estado de los

procesos de evaluación. También le es posible usar y evaluar las soluciones, esto último, a través de la emisión de votos. Algunas de estas capacidades son limitadas o provistas por privilegios, como se explicará más adelante.

Notar que no existe un rol publicador de problemas y otro publicador de soluciones. De hecho, no existen roles. Todos los usuarios cuentan con las mismas capacidades desde su registración en la comunidad.

Un usuario también tiene la capacidad de asociarse a un problema. Esto le permite, además de manifestar su interés en la resolución del mismo, registrarse para obtener una solución al mismo.

Los usuarios están representados por piezas de información básicas, como el nombre, email o avatar y tiene la capacidad de indicar sus intereses. Estos intereses se representan a través de **etiquetas** que son también usadas por los problemas. Se entiende entonces que si a un usuario le interesa una determinada etiqueta, entonces le interesarán aquellos problemas que tengan esa misma etiqueta.

Otra característica que tienen las etiquetas es que son extensibles. Es decir, las mismas no están predefinidas en el sistema sino que los usuarios pueden crear nuevas etiquetas, en caso de tener los privilegios para hacerlo. De esta manera, se debe invertir menos tiempo en realizar tareas de administración. Notar que las etiquetas no son más que breves textos los cuales, irán incrementando o decrementando su popularidad en base a las tendencias de internet.

3.3.2 Problema

Representa un problema de usabilidad que fue reportado por un usuario. El problema cuenta con un título que explica brevemente el problema y una descripción que lo detalla. No necesariamente tiene que ser una descripción técnica, ya que cualquier usuario debería poder reportar problemas. Adicionalmente, tiene etiquetas que permiten relacionar tanto problemas entre sí como problemas y usuarios, siendo esta última la representación de intereses mencionada anteriormente.

Un problema no se encuentra relacionado de manera directa con sus soluciones, sino con *procesos de evaluación*. Es a través de éstos que el problema determina su solución actual, si es que la hay.

Dado que los problemas pueden referirse a una o varias páginas web es necesario que los mismos provean alguna forma simple que permita identificar a qué páginas corresponden. Notar que con simplemente pensar en un dominio, como google.com, puede no ser suficiente ya que un problema puede estar presente en google.com/calendar pero no en google.com/gmail. Es por esto que los problemas permiten el ingreso de una ruta, como podría ser google.com/gmail, que permite relacionar el problema tanto al dominio como a los demás componentes de la URL en cuestión.

3.3.3 Proceso de evaluación

Los procesos de evaluación son los nexos entre los problemas y las soluciones. Dada la naturaleza dinámica de la web, ninguna solución durará indefinidamente. Es decir, la adaptación que puede efectuar una solución a una página dada puede dejar de tener efecto tras realizar cambios mínimos en la página original, los cuales están fuera del control del desarrollador de la solución. Sin embargo, el problema podría seguir vigente.

Por este motivo, es difícil pensar que hay una única solución válida para un problema a lo largo del tiempo. Además pueden existir soluciones alternativas para un problema dado al mismo tiempo. Por esto, es más probable que, durante la vigencia de un problema, se propongan distintas soluciones válidas que lo solucionan de acuerdo a la estructura de la página web en ese momento dado. Es esta, justamente, la motivación de los procesos de evaluación.

Los mismos no son más que breves iteraciones de pruebas de soluciones que tienen como resultado la selección de la que la comunidad considere como la mejor. Estos procesos pueden tener uno de tres estados:

- **Aceptando soluciones:** este es el estado inicial. Cuando un proceso de evaluación para un problema específico se encuentra en este estado, entonces es posible agregar soluciones al mismo. Es una etapa de preparación.

- **Evaluando soluciones:** una vez que se han propuesto una o varias soluciones a un problema, este estado es aquel por el cual estas soluciones se ponen a competir entre ellas y con la versión original en un proceso similar al de A/B testing. Es decir, las soluciones se asignan en forma aleatoria entre los usuarios y los mismos hacen uso de la solución que les fue asignada y la evalúan.
- **Finalizado:** cuando un proceso se encuentra en este estado, se debe a que ya fue seleccionada una mejor solución y la misma será asignada a todos los usuarios.

Las transiciones posibles entre los estados del proceso de evaluación son las siguientes:

- **Iniciar evaluación de soluciones:** esta transición desde el estado “Aceptando soluciones” al estado “Evaluando soluciones” dispara la asignación aleatoria de soluciones a todos aquellos usuarios registrados al problema a fin de poder comenzar con el proceso de A/B testing. La misma debe cumplir una condición para poder concretarse: debe existir un número mínimo de soluciones asociadas al proceso de evaluación. De esta manera se asegura que el proceso de A/B testing sea relevante y no simplemente la selección de una única solución.
- **Finalizar evaluación de soluciones:** esta transición desde el estado “Evaluando soluciones” al estado “Finalizado” es la que se encarga de concluir el proceso de evaluación, resultando en la selección de la mejor solución. A fin de poder llevar a cabo esta transición, el proceso de evaluación debe tener al menos una solución con feedback positivo y que haya sobresalido por sobre las demás o, en su defecto, haber pasado un período de tiempo mínimo en el estado “Evaluando soluciones”.

Como se verá en detalle en el Capítulo 5, estas dos posibles transiciones entre estados del proceso de evaluación podrán ser disparadas por un usuario de la comunidad que tenga los privilegios necesarios para ello. Sin embargo, el usuario que inicia la evaluación de soluciones no hace falta que sea el mismo que la finaliza.

El proceso a alto nivel se puede apreciar en el siguiente gráfico:

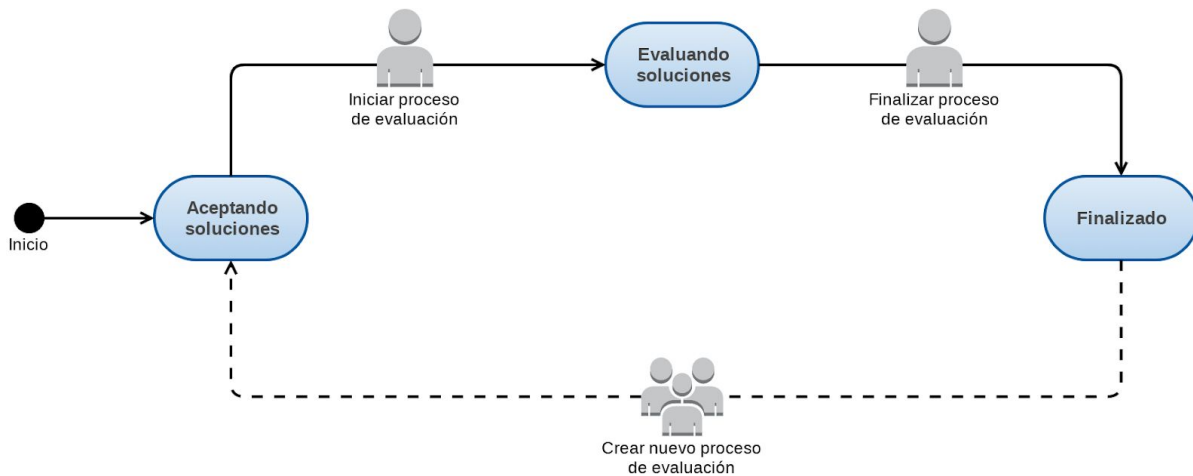


Figura 3.3.3.1: Diagrama de estados y transiciones del proceso de evaluación.

Notar que en el gráfico se puede apreciar una línea punteada con una transición no mencionada anteriormente. La misma no forma parte del proceso de evaluación en sí mismo sino que representa la creación de un nuevo proceso de evaluación. Es decir, la solución de un problema se toma del proceso de evaluación más reciente asociado al mismo problema. Sin embargo, las soluciones pueden, de un momento a otro, dejar de ser eficaces. Es por este motivo que los usuarios deben tener la capacidad de crear un nuevo proceso de evaluación para el mismo problema, lo cual les permita publicar nuevas soluciones y seleccionar una que efectivamente solucione el problema para el nuevo contexto en el que se encuentra. Esto resulta en que los procesos de evaluación anteriores pierdan vigencia ya que sus soluciones muy probablemente hayan dejado de resolver el problema.

En otras palabras, en un momento dado existe un único proceso de evaluación para cada problema específico, y ese proceso permite la publicación de soluciones para ese problema, su evaluación y la selección de la mejor solución. Estos pasos se efectúan hasta que la comunidad considera que un problema está solucionado. Cuando la comunidad cambia de parecer se comienza el ciclo nuevamente, creando un nuevo proceso de evaluación.

3.3.4 Solución

Las soluciones son, en breve, las entidades que se encargan de efectuar las modificaciones pertinentes en las páginas web que navega el usuario a fin de corregir un problema dado. Las mismas se asocian a un proceso de evaluación. El componente esencial

de las mismas es un script que debe cargar el usuario que las publica. El mismo es el corazón de la solución ya que contiene la lógica que permite corregir el problema. Actualmente, dado que las páginas web operan con JavaScript, se espera que los scripts sean en este mismo lenguaje. Este código es inyectado en la página web navegada por el usuario.

Dado que las soluciones posiblemente sean muy específicas y permitan mejorar sólo unas pocas páginas en particular, es importante que cada solución pueda limitar su alcance, es decir, que pueda especificar a qué páginas aplicará. Es por ello que las mismas constan de un campo que se usa para filtrar su alcance. Técnicamente, es una expresión regular que se aplica utilizando como entrada la dirección que el navegador está navegando y, en caso de que coincida con dicha expresión, entonces la solución se ejecuta y, en caso de que no coincida, entonces no se ejecuta.

Durante el ciclo de vida de un proceso de evaluación, las soluciones pueden ser votadas. Los votos permiten determinar el puntaje que va a tener una solución. El puntaje se calcula a través de la diferencia de votos positivos y negativos que tiene la misma. Cuando un proceso de evaluación selecciona una mejor solución lo hace en base al puntaje de las mismas.

Existe un tipo de solución denominada "placebo". La misma no es una solución que haya sido publicada por un usuario sino que se crea de manera automática tras iniciar un proceso de evaluación. Es lo que en experimentos controlados se conoce como "solución de control". Esta solución es igual a cualquier otra con la diferencia de que tiene un script vacío. Esto significa que, al aplicar esa solución, realmente no se solucionará el problema y la página navegada no será modificada.

La solución placebo, además de su rol de solución de control, tiene como motivación adicional determinar, hasta cierto punto, qué tan perceptible es el problema. Es decir, si una solución placebo es seleccionada como ganadora, existe la posibilidad de que realmente el problema no haya sido notado por varios de los usuarios que se les haya asignado esa solución. Esto debería, al menos, despertar dudas en la comunidad acerca de la existencia misma del problema o tal vez de la redacción del mismo.

Otra utilidad importante de la solución placebo es que cumple un rol clave al momento de finalizar un proceso de evaluación. La finalización de uno de ellos debería, por lo general, resultar en una solución que efectivamente resuelva el problema. Sin embargo, si se presenta la situación en la cual ninguna solución resuelve el problema y, como es de esperarse, la solución placebo tampoco lo hace, entonces es muy probable que todas las soluciones se encuentren evaluadas negativamente. Esto lleva a que, como se explicó anteriormente, el proceso de evaluación sólo pueda ser finalizado una vez transcurrido un determinado período de tiempo. Al hacerlo, se termina eligiendo la solución placebo, independientemente de los votos. Esto se debe a que la misma no realiza ninguna acción, resultado en la no solución del problema. De esta manera se permite a los usuarios volver crear un proceso de evaluación para ese problema y proponer nuevas soluciones.

3.4 Detalle de las funcionalidades principales

Habiendo presentado las entidades esenciales del sistema, procederemos a explicar un poco más en detalle las funcionalidades principales.

3.4.1 Asociación y desasociación de problemas

Como se explicó anteriormente, los usuarios tienen la capacidad de manifestar su interés en obtener una solución a un determinado problema. Al hacerlo, se pueden presentar distintas situaciones dependiendo del estado en el cual se encuentran el proceso de evaluación más reciente asignado al problema.

Si al momento en el cual el usuario se asocia al problema, el proceso de evaluación vigente se encuentra *aceptando soluciones* entonces a ese usuario no se le aplicará ninguna solución al navegar. No hace ninguna diferencia la cantidad de soluciones asignadas el proceso de evaluación. Se considera que el usuario está en espera del inicio de la evaluación. Cabe destacar que, al momento de iniciar la evaluación de soluciones es cuando se realiza la asignación de soluciones a los usuarios.

En caso de que el proceso de evaluación vigente para el problema se encuentre *evaluando soluciones*, entonces al usuario se le asigna en ese mismo momento una solución al problema, aplicando el mismo algoritmo de asignación que se usó al iniciar el proceso de

evaluación. De esta manera el usuario podrá comenzar a probar la solución de manera inmediata. Cabe destacar que en esta instancia el usuario puede votar la solución que se le asignó. Notar que esta solución se le mantendrá asignada al menos hasta el momento en el cual el proceso de evaluación haya finalizado, ya que la finalización del mismo puede resultar en la reasignación de una nueva solución (aquella que haya sido mejor evaluada por la comunidad).

El último estado restante de los procesos de evaluación es el estado *finalizado*. Cuando un usuario se asocia a un problema cuyo proceso de evaluación se encuentra en ese estado, el mismo ya tiene seleccionada una solución. De esta manera, no es necesario pasar por una instancia de prueba sino que se le asigna al usuario la solución definitiva. Es justamente en este momento en el cual el usuario puede solicitar que se genere un nuevo proceso de evaluación.

La desasociación a un problema se da cuando un usuario manifiesta no tener más interés en seguir aplicando soluciones a un problema. Es decir, tras desasociarse, no se le aplicará ninguna solución a ese problema durante la navegación. Sin embargo, el sistema lleva cuenta de a qué usuario se le asignó cada solución. De esta manera, en caso de que un usuario que se haya desasociado a un problema vuelva a asociarse al mismo, se le asignará nuevamente la misma solución, a fin de evitar brindarle la capacidad al usuario de cambiar la solución que se le asignó.

Como se puede ver, la asociación a problemas hace posible la obtención de soluciones durante el proceso de navegación. Es una precondition esencial para el mismo, como se explicará a continuación.

3.4.2 Inyección de soluciones

La navegación es quizás la piedra angular en este proceso.

El proceso comienza cuando el usuario navega a un sitio web. Cuando esto sucede, como es de esperarse, se realizan requerimientos web propios del sitio que se está navegando. Sin embargo, nuestra extensión del navegador realiza un requerimiento adicional el cual se

dirige a nuestro servidor. Este requerimiento es el que nos permitirá obtener los scripts que se deben ejecutar para llevar a cabo las modificaciones en la página navegada.

Para ello, la extensión debe enviarle a nuestro servidor información que permita identificar al usuario que está navegando y la página web que se solicitó. Con estas piezas de información, nuestro servidor se encarga de buscar problemas cuya ruta se ajuste a la ruta que el usuario está navegando y que, al mismo tiempo, el usuario se encuentre asociado.

La respuesta del servidor incluye distintos tipos de información. Retorna información de los problemas que se están solucionando, información de la solución a los mismos, sugerencias de problemas para esa misma página que se ajustan a los intereses del usuario y también información de los votos que haya realizado el usuario a las soluciones retornadas. El servidor responde con toda esa información a fin de minimizar la cantidad de requerimientos web necesarios.

Cabe destacar que una parte esencial de las soluciones es el script, ya que contiene la lógica de la solución en sí misma. La extensión del navegador procesará la respuesta en búsqueda de las mismas, y las inyectará en la página web actual, modificando así la página original e, idealmente, dando solución a los problemas correspondientes.

El usuario tendrá disponible un breve reporte de los resultados de la navegación realizada. Este reporte le mostrará varias piezas claves de información que le permitirán entender el estado actual de la página en la que está navegando y las opciones que tiene disponibles. Es decir, el reporte indicará, por ejemplo, si existe algún script en ejecución e incluirá un listado de aquellos problemas que cumplan dos condiciones: por un lado, que sean problemas a los cuales el usuario se haya asociado y, por otro, que esos problemas correspondan a la página que se está visitando.

Adicionalmente, el usuario podrá tener un cierto grado de interacción con los problemas y soluciones que se aplican en la página que navega. Más precisamente, para cada una de las soluciones que se apliquen al navegar, se le presentará al usuario las operaciones que éste puede realizar sobre ellas así como sobre el problema mismo. Cabe destacar, que esas operaciones son un subconjunto de las operaciones que el usuario puede realizar desde la

aplicación web. Es decir, el usuario puede, por ejemplo, votar una solución o solicitar que se cree un nuevo proceso de evaluación para un problema.

Todas las operaciones que un usuario puede realizar están en función del estado en el cual se encuentra el proceso de evaluación actual de cada uno de los problemas. Por ejemplo, de estar el problema aceptando soluciones, no habrá ningún script para ejecutar ya que no habrá ninguna solución asignada y el usuario sólo podrá desasociarse del problema.

Por otro lado, si el proceso de evaluación actual del problema se encuentra probando soluciones, el usuario tendrá asignada una solución de entre las que están siendo evaluadas, la cual debería haberse inyectado en la página actual. De esta manera, el usuario podrá votar por la misma, indicando si el problema fue efectivamente solucionado o si, por el contrario, no tuvo el efecto esperado. Si el usuario ya hubiera votado una solución con anterioridad, su elección estaría visible y tendría la posibilidad de modificar su voto en caso que lo considere necesario.

En caso en que el proceso de evaluación haya finalizado, el usuario tendrá asignada la solución aceptada, la cual ya debería haber sido inyectada en la página actual. De esta manera, el usuario podrá votar para crear un nuevo proceso de evaluación, en caso de que considere que la solución actual no haya tenido el efecto esperado.

Además de estas operaciones dependientes del estado del proceso de evaluación, el usuario siempre tendrá la posibilidad de desasociarse del problema.

Además de los problemas y soluciones a los cuales el usuario se asoció y las operaciones que el usuario puede realizar sobre los mismos, el reporte mostrará otros problemas a los cuales el usuario no se asoció pero corresponden a la misma página web en la que el usuario se encuentra navegando. Cabe destacar que estos problemas deben tener al menos una etiqueta que se ajuste a las etiquetas que el usuario haya indicado que son de su interés

Al margen de las preferencias de etiquetas, el usuario tiene otra característica que puede modificar desde su perfil que tiene un impacto significativo en la navegación. La misma le permite al usuario indicar si desea que el sistema lo asocie de manera automática a problemas cuyas etiquetas coincidan con las que el usuario marcó como que eran de su

interés. Por ejemplo, consideremos un usuario interesado en la etiqueta *daltonismo* que indicó que desea que se apliquen soluciones a problemas que se ajusten a sus intereses. Cuando ese usuario navegue a una página web para la cual existan problemas que tengan la etiqueta *daltonismo*, entonces el sistema lo registrará de manera automática a los mismos y le proveerá las soluciones correspondientes, ejecutando así los scripts asociados a las mismas.

Esta opción reduce las interacciones necesarias con el sistema para obtener soluciones potencialmente útiles para el usuario, ya que elimina la necesidad de consultar el listado de sugerencias y asociarse a los problemas de manera manual.

3.4.3 Votar solución

Como comentamos en la sección de navegación, cuando el usuario navega una página web para la cual corresponde aplicar alguna solución, el script asociado a la misma es inyectado en el código de la página para, *idealmente*, solucionar el problema al cual se asoció el usuario. Decimos *idealmente* porque es una posibilidad que el mismo no tenga el efecto esperado. Sea por error de interpretación de requerimientos, por error de programación o simplemente porque el script quedó desactualizado desde el momento de su creación, el usuario debe poder reportar dicha situación al sistema. El mecanismo por el cual el usuario indica la efectividad de la solución que le fue aplicada es, justamente, el voto.

Estos votos aplican sólo a soluciones. Más específicamente, aplican a soluciones que los usuarios hayan tenido la oportunidad de probar. Es decir, un usuario no podrá evaluar una solución que no haya probado. Adicionalmente, sólo se pueden votar soluciones mientras los procesos de evaluación a los cuales se asocian las mismas se encuentran evaluando soluciones. Notar que los votos pueden ser tanto positivos como negativos, representado de esta manera la opinión que el usuario tiene respecto de si la solución fue efectiva o, por el contrario, no cumple con lo esperado.

Los votos emitidos por todos los usuarios para una solución que está siendo evaluada en un proceso de evaluación afectarán positiva o negativamente a que la misma sea aceptada al momento de finalización de dicho proceso de evaluación. Recordemos que cuando describimos el proceso de evaluación mencionamos que al momento de finalizar se tomaría

en cuenta el feedback de los usuarios de las soluciones que participan para seleccionar una única solución aceptada.

El usuario tendrá la posibilidad de emitir el voto tanto desde la extensión del navegador (en el reporte que tiene disponible al navegar la página a la cual hace referencia al problema) como desde el sistema (accediendo a la página web correspondiente al proceso de evaluación al que pertenece la solución).

A su vez, cuando el usuario haya votado, sea tanto de manera positiva como negativa, dicha elección quedará registrada y estará visible para el usuario de manera que el mismo tendrá la posibilidad de anular o cambiar completamente su voto en caso de que así lo considere necesario.

3.4.4 Votar para iniciar un nuevo proceso de evaluación

Del mismo modo en que un usuario puede emitir su voto para evaluar una solución que le haya sido asignada mientras el proceso se encontraba evaluado soluciones, el usuario también podrá votar una vez que el proceso haya finalizado.

Este voto, a diferencia del mencionado en el punto anterior, no aplicará de manera directa a una solución sino que el mismo se asociará al problema en sí mismo. El mismo le permite al usuario manifestar su intención de iniciar un nuevo proceso de evaluación para el problema en cuestión. Esto podría deberse, por ejemplo, a que la solución seleccionada haya perdido vigencia.

Cabe destacar que este voto no es exactamente lo mismo que un voto negativo a una solución. Los votos que reciban las soluciones mediante el período de prueba siguen siendo válidos y ayudan determinar una mejor solución en un momento dado. Este tipo de votos, se realizan en una instancia posterior, en la cual las páginas web a las que aplican las soluciones pueden haber cambiado sustancialmente. Por lo tanto, no sería apropiado evaluar negativamente a una solución si la misma deja de tener efecto por causas externas.

Los usuarios pueden acceder a esta funcionalidad de voto tanto desde la página web del sistema como desde la extensión del navegador. La funcionalidad es en sí, la misma. La única diferencia es su facilidad de acceso.

Al igual que para votar soluciones, si el usuario ya votó para generar un nuevo proceso de evaluación para un problema dado, éste podrá visualizar dicho voto y tendrá la posibilidad de anularlo.

4. A/B testing

En el Capítulo 2 dimos una pequeña introducción al concepto de A/B Testing y mencionamos que nuestra intención era diseñar un mecanismo inspirado en el mismo para la evaluación de contribuciones.

En este capítulo profundizaremos un poco más en el tema de A/B Testing, revisaremos el proceso de evaluación que definimos para nuestra necesidad particular, y trazaremos paralelos y diferencias con procesos de A/B testing propiamente dichos.

4.1 Arquitectura básica de un sistema de experimentación de A/B Test

Nos concentraremos en este caso en el microcosmo de un sistema básico, su estructura y componentes esenciales.

Recordemos que en su mínima expresión, un sistema de experimentación A/B Test propone la evaluación el desempeño de una variante del sistema B en comparación al sistema base A mediante la realización de un experimento donde se desplegarán durante un periodo de tiempo tanto la variante como el sistema base, se distribuirán los accesos al sistema de manera equitativa entre las mismas, y se tomarán métricas de performance de cada una de ellas de manera de poder analizar el impacto que tuvo la variante para poder tomar decisiones respecto a su conveniencia.

Esta expresión mínima del A/B test con 1 sola variante puede extenderse a múltiples variantes al sistema base, lo cual también suele denominarse split-testing [Kohavi 2009]. En este trabajo nosotros llamaremos A/B testing en general a este tipo de experimentación randomizada, tenga esta una o varias variantes.

En el artículo de Kohavi et al. [Kohavi 2009] se descompone un sistema de experimentación en tres componentes esenciales:

- El algoritmo de aleatorización.
- El método de asignación.

- El camino de los datos.

4.1.1 El algoritmo de aleatorización

En pocas palabras, el algoritmo de aleatorización determina qué variante aplicar de entre las disponibles.

La importancia de este algoritmo se debe a que el modelo estadístico que se aplica para la interpretación de los resultados de un experimento asume que cada variante participante del experimento tuvo un flujo de usuarios aleatoriamente seleccionado.

Segun el trabajo de investigacion de Kohavi, el algoritmo debiera cumplir con tres características claves, y opcionalmente dos características deseables.

1. Debe permitir una distribución equitativa de las variantes.
2. Debe ser consistente en las asignaciones. Esto es, un usuario debiera ser asignado a la misma variante cada vez que reingresa al sitio.
3. La asignación de una variante en un experimento debe ser completamente independiente de la asignación realizada en otro experimento.
4. Sería deseable que el algoritmo permita que el porcentaje de asignación de usuarios a variantes pueda ser lentamente creciente a medida que se lo desea.
5. Sería deseable también que pueda ser controlado de manera externa. Esto es, permitir la asignación o desasignación manual de variantes.

Dadas estas condiciones, se sugieren dos algoritmos que cumplen al menos con las primeras tres:

Cached pseudorandom

Un algoritmo de generacion de numeros pseudoaleatorios puede ser usado como algoritmo de aleatorización siempre y cuando se lo acompañe con alguna clase de memoria que permita que una vez asignado un usuario a una variante, la misma quede registrada de alguna manera (sea en el servidor en alguna base de datos o en el cliente mediante una cookie del navegador) para permitir que un usuario reciba siempre la misma variante.

El algoritmo de generación de números aleatorios por sí mismo cubre los requisitos esenciales 1 y 3, mientras que el almacenamiento de la asignación permite cumplir con el requisito 2 y potencialmente, de permitir la modificación del valor almacenado mediante una acción externa, podría cubrir con la característica deseable n. 5.

Hash and partition

Este método permite determinar la variante a ser asignada mediante el uso de una función de hash, la cual se aplicaría sobre algún elemento identificador del usuario y del experimento (sea una clave de base de datos o una cookie en el navegador) y tendría como resultado un número conocido como hash code, el cual se espera esté uniformemente distribuido en un rango de valores. El rango de valores posibles es particionado asignando una partición a cada variante del experimento.

A diferencia de un generador de números aleatorios, este método permite la asignación de un usuario a una variante de manera consistente sin depender de un mecanismo de almacenamiento de asignaciones. Esto se debe a que una función de hash es una función matemática que opera sobre una entrada produciendo para cada entrada siempre la misma salida.

La elección de la función de hash es clave para garantizar la correcta distribución de usuarios a las variantes.

Es necesario que la función empleada genere hashes distribuidos uniformemente en el rango de valores, y el resultado de una función debiera ser impredecibles, esto es, por más pequeño que sea el cambio en el valor de entrada, el resultado debiera poder ser radicalmente distinto, evitando especialmente mantener secuencialidades.

No ahondaremos en el tema de funciones de hash ya que la profundidad del mismo excede por mucho nuestras necesidades, siendo el mismo un gran foco de investigación fuertemente asociado a cuestiones de seguridad.

Este algoritmo cumple con los requisitos esenciales 1, 2 y 3 de manera automática. Para permitir además el soporte para la quinta propiedad y debido a que este mecanismo no hace uso de almacenamiento de manera nativa, podría usarse un esquema híbrido, donde

sea almacenado y consultado algún valor exclusivamente para casos que diverjan del estándar.

4.1.2 El método de asignación

En pocas palabras, el método de asignación determina cómo se aplica la variante seleccionada al usuario al interactuar este con el sistema.

Un buen método de asignación debiera permitir manipular cualquier variable en el sistema, sea esta algún elemento visible como la presentación de la página web, no visible como algún algoritmo de back-end, o una combinación de los mismos.

Veamos algunas alternativas de métodos de asignación relevadas en el artículo de Kohavi et al. [Kohavi 2009].

División de tráfico

La familia de métodos conocidos por el nombre de “división de tráfico” (traffic split) implica que cada variante de un experimento a ser testeada sea un servicio diferente, los cuales pueden estar desplegados en distintos servidores, sean estos físicos o virtuales, o incluso en un mismo servidor corriendo en distintos puertos.

Estos sistemas hacen uso de un balanceador de carga o proxy que realiza la división del tráfico hacia una u otra variante en base a la lógica del algoritmo de aleatorización.

Estos sistemas tienen como principal ventaja el hecho de que son no intrusivos respecto al código de aplicación, es decir, la aplicación no requiere de lógica propia del experimento y el único cambio de código necesario son los cambios propios de cada variante.

La principal desventaja radica en el costo en infraestructura de los mismos, que podría contrarrestar las ganancias obtenidas al minimizar costos de desarrollo.

Cabe aclarar que esta desventaja está perdiendo relevancia con los avances tecnológicos y nuevas prácticas que facilitan el despliegue de sistemas (virtualización / containers, microservicios, infraestructura como servicio, infraestructura como código, automatización)

al permitir reducir tanto los tiempos necesarios para inicializar y dar de baja servidores, como los costos y esfuerzo humano.

Aun así, veamos los puntos que históricamente influyeron en este incremento de costo.

- Experimentos pequeños tienen un costo desproporcionado al tener que replicar la aplicación entera independientemente del tamaño del cambio.
- La variante de control debiera mantener el 100% de la capacidad en caso que las variantes presenten algún problema y deban ser canceladas, las otras variantes pueden tener menor capacidad pero esto limitaría la cantidad de usuarios que pueden ser asignados a cada una lo que podría limitar la potencia del experimento.
- La ejecución de múltiples experimentos en paralelo podrían requerir de una cantidad exponencial de servidores que permita cubrir cada combinación de variantes.
- Cualquier diferencia en la infraestructura de las variantes podría influir en el resultado del experimento por lo que idealmente debiera emplearse hardware y topología de red idénticos.

Reescritura de página

El mecanismo de reescritura de página incorpora un tipo de proxy especial para la reescritura de contenido html que modifica el contenido de la página antes de que el mismo sea presentado al usuario. Este servidor proxy es quien implementa el algoritmo de aleatorización para determinar la variante de uno o más experimentos que corresponde al usuario.

Al igual que el método de división de tráfico, este método es esencialmente no intrusivo, sin embargo, presenta varias desventajas que limitan su aplicabilidad a ciertos casos particulares, esencialmente excluyendo cambios de back-end.

- El tiempo de carga de una página se ve incrementado al adicionarse el tiempo de procesamiento de la reescritura del servidor proxy y latencia de conexión extra.
- La experimentación en sitios grandes tiene altos requerimientos de hardware. Como el servidor proxy debe manejar todo el tráfico, puede volverse un cuello de botella.
- El desarrollo de las variantes es más costoso y limitado, ya que los cambios deben ser expresados en un lenguaje de reglas de transformación de html. Es posible que una

variante no pueda ser expresada con dichas transformaciones, o sea muy difícil de implementar y testear

- La posibilidad de ejecutar experimentos para testear cambios algorítmicos de back end es escasa o nula, ya que en la mayoría de los casos las decisiones serían tomadas antes del renderizado del html.

Asignación del lado del cliente

Este método de asignación permite ejecutar experimentos del lado del cliente sin necesidad de modificar el código en el servidor.

Se inserta en la página retornada al cliente un código javascript a ser ejecutado por el cliente previo al renderizado de la página. Este código realiza una llamada a un servidor de asignación, el cual implementa la lógica del algoritmo de randomización seleccionando la variante a aplicar en el usuario. El servidor de asignación informa la variante asignada, de manera que el script pueda ahora realizar las modificaciones necesarias al DOM. Los recursos necesarios para realizar estas modificaciones podrían estar presentes en la respuesta original o ser incluidas en la respuesta del servidor de asignación.

Este método, si bien puede parecer intrusivo, es bastante sencillo de implementar y esta simpleza es su principal ventaja.

Las desventajas por otro lado son:

- Aplicabilidad limitada. Al igual que la reescritura de página, está principalmente orientado a cambios de front-end.
- Las páginas complejas con mucho contenido dinámico pueden dificultar la implementación de variantes, necesitando quizás profundas modificaciones al código de front end inicial para facilitar la implementación de variantes.
- La ejecución de código extra del lado del cliente, así como la llamada al servidor de asignación puede introducir retrasos e impactar negativamente la experiencia de usuario.

Asignación del lado del servidor

Esta familia agrupa los métodos que integran los experimentos en el código de la aplicación para que sea la misma la que genera resultados para cada una de las variantes.

Esto puede tomar forma mediante el uso de simples hooks de código, llamadas a alguna API particular que determine la variante a aplicar seguidos de la ejecución de código asociado, o abstracciones que inyecten variantes en puntos claves permitiendo al código depender de interfaces locales abstractas.

Este mecanismo es extremadamente intrusivo y requiere de cambios profundos en el código de la aplicación, especialmente en los primeros experimentos, mientras el equipo de desarrollo no haya aún implementado un framework que facilite dichos experimentos.

Las principales ventajas son:

- Es un método extremadamente genérico, permitiendo experimentar virtualmente con cualquier aspecto del sistema.
- Inserta la experimentación en el lugar más preciso posible, sea esto en decisiones algorítmicas de back end o de renderizado en front end.
- Esencialmente transparente al usuario. Empleando la misma infraestructura (mismos servidores) para todas las variantes se minimiza el potencial tiempo de retraso de aplicar una variante.

Las desventajas de este método provienen de la inherente intrusividad.

- El costo en cuanto a tiempo de desarrollo es superior, especialmente los primeros experimentos.
- Debido a que cada experimento requiere de cambios profundos de código, se introduce un riesgo adicional aumentando las probabilidades de errores, especialmente para experimentos complejos que modifican múltiples variables.
- Dependiendo de qué tan bien desarrollado se encuentre el proceso de experimentación, el mantenimiento del código puede volverse complejo y propenso a errores. El hecho que coexistan múltiples versiones de código que deberán ser eliminadas al finalizar cada experimento requiere de especial cuidado y disciplina.

4.1.3 El camino de los datos

Captura

Para poder capturar métricas de los distintos experimentos, es necesario primero poder asociar los eventos capturados (vistas a la página, clicks, ventas, tiempo de carga, etc) al experimento y variante que corresponde.

Los mecanismos para capturar los eventos en bruto para un experimento controlado son iguales a los clásicos usados por cualquier sitio web. Al momento de realizar un experimento, contamos entonces con las siguientes alternativas:

Usar un mecanismo de captura de métricas existente:

En caso que el sitio web ya posea algún mecanismo que capture métricas del sitio, sea este un desarrollo propio interno o empleando un servicio externo como Google Analytics, es posible hacer uso del mismo con la sola adición de un registro que indique experimento y variante aplicada.

Esta alternativa en un principio es muy sencilla de implementar, al menos en la fase de captura de datos. El análisis por otro lado puede resultar muy complejo dependiendo de las capacidades de dicho sistema para procesar o extraer datos, ya que generalmente no se especializan en el tipo de análisis estadístico necesario para un experimento controlado. Frecuentemente es necesario extraer y procesar los datos mediante algún otro sistema.

Registro en archivos de logs locales:

Este mecanismo también se apoya en el uso de infraestructura existente, en este caso, la que típicamente es generada por un sitio web para información del desarrollador. A medida que el sistema se hace más complejo, es necesario recolectarlos de los servidores y procesarlos en servidores centralizados de logs. Al igual que el caso anterior, será necesario extraer los datos en bruto y procesarlos para obtener resultados.

Empleo de un servicio especializado:

Con este modelo se hace uso de algún servicio especializado en recolectar eventos y datos observados. Este servicio podría ser invocado tanto desde los servicios de back end, los servidores web o desde el navegador del usuario.

Este método es el que provee la mayor flexibilidad

Análisis

Para poder comparar el impacto de cada variante y obtener un resultado del experimento, el sistema debiera poder sumarizar estos valores, agrupándolos por experimento y variante. Estas métricas pueden ser tan sencillas como un total de visitas o más complejas como deducciones respecto a la satisfacción del cliente.

Los valores de estas métricas se comparan buscando determinar variaciones estadísticamente significativas y generar un reporte que nos permita aprender algo sobre cada variante puesta en experimentación y nos permita tomar decisiones para futuros experimentos.

Además de analizar las métricas del experimento en su conjunto, es común además realizar un análisis en profundidad, individualizando distintos segmentos de usuarios para determinar si alguna subpoblación en particular generó resultados con diferencias estadísticamente significativas entre variantes, de manera de poder comprender mejor el impacto generado por la variante correspondiente, con el objetivo de determinar el camino a seguir con la idea implementada en dicha variante.

Este análisis puede realizarse de manera manual o haciendo usos de técnicas de aprendizaje automático o minería de datos.

4.2 Nuestro proceso de evaluación.

Nuestro sistema pretende proveer a los usuarios un mecanismo para la mejora de aplicaciones web mediante el uso de scripts client-side. Estos scripts son creados por la misma comunidad de usuarios como respuesta a la presentación de un problema específico. Al existir la posibilidad de que múltiples soluciones sean propuestas para un mismo

problema, se realiza una evaluación de las mismas para determinar la más apropiada. De considerarse en algún momento que la solución seleccionada ya no cumple con su propósito, se reabre la posibilidad de presentar nuevas soluciones y se realiza una nueva evaluación.

Con este breve repaso de lo presentado en el capítulo 3 y habiendo hecho una reseña del marco teórico de los procesos A/B testing en la primera parte de este capítulo podemos ver que el proceso de evaluación que realizamos no es otra cosa que un experimento A/B test.

En general los experimentos randomizados o A/B test son utilizados para determinar qué variante genera mayores ganancias en términos lucrativos dentro de una empresa. De esta manera, la variante más lucrativa suele medirse por cantidad de compras, o cantidad de clicks que indican intención de compra, o cantidad de agregados al carrito, etc. [King 2017]. En el caso de nuestra plataforma, hay varias consideraciones que debemos analizar porque hay 2 diferencias fundamentales con la manera en que en general se aplica el A/B testing: una diferencia es que en nuestro caso, el objetivo del experimento no es determinar la variante más lucrativa, sino la que sea más usable; la segunda diferencia es que no hacemos los experimentos sobre un sitio propio sino ajeno, justamente porque la plataforma fue desarrollada para dejar el poder de cambiar cualquier sitio web con problemas de usabilidad en manos de los mismos usuarios, y que no tengan que depender de una respuesta de los desarrolladores.

4.2.1 Nuestro proceso de evaluación como un proceso continuo de A/B Test

Objetivos, problemas y oportunidades de mejora

El objetivo de nuestro sistema es único: Mejorar la experiencia de usuario. La métrica contra la que evaluaremos siempre las variantes es la misma: El feedback del usuario.

Cada problema reportado por el usuario se convierte inmediatamente en un problema u oportunidad de mejora. Es difícil saber a priori las características de los problemas que podrían presentarse y, más difícil aún, crear un lenguaje único que permita representar estas características, así como un mecanismo de evaluación de la efectividad de las soluciones que sea además, comprensible y manejable por un usuario común, quienes son

nuestra principal audiencia. Este problema por ende tendrá una descripción textual y su interpretación quedará a cargo de cada contribuyente.

Experimentos y variantes

Como mencionamos en el Capítulo 3, un problema a lo largo del tiempo podrá disparar procesos de evaluación los cuales seguirán el ciclo común de aceptar variantes, realizar el experimento, y finalizar experimento y determinar la variante aceptada.

Así como cada proceso de evaluación que realizamos es un experimento que se realiza para verificar una hipótesis, cada solución que participa en dicho proceso de evaluación es una variante del sistema que intentamos poner a prueba. La solución que llamamos “placebo” creada al iniciar el proceso de evaluación es la variante de control. Esta variante de control permite que haya usuarios que accedan al sistema original sin cambios, y de esta manera tener un marco de referencia con el cual comparar cada variante.

Población y muestra

Los usuarios a los cuales se les inyecta alguna solución son una muestra de la población total de los usuarios del sistema. Cada usuario será asignado a una variante (incluyendo la de control) y tendrá la capacidad de votar indicando si considera la solución como válida o si por el contrario, no cumple con lo esperado de ella. Este altamente subjetivo voto es la métrica que empleamos para tomar alguna decisión respecto a la variante ganadora.

4.2.2 Componentes de arquitectura básica de A/B Test

Según la clasificación de los componentes de arquitectura básica de un experimento propuesta por Kohavi:

Nuestro algoritmo de aleatorización

Hacemos uso de un algoritmo de aleatorización de tipo cached pseudorandom

Cuando un usuario accede por primera vez a un sitio al cual aplica una solución para un problema al cual está asociado y que se encuentra en estado de evaluando soluciones, se determina de manera aleatoria a qué variante pertenece. Esto queda registrado y se utilizara para todo acceso futuro mientras dure el proceso de evaluación. Cuando un

proceso se cierra esta asociación es eliminada del sistema ya que el experimento terminó, y se desea ahora hacer uso de la variante ganadora por lo que de reabrirse, el usuario quedará abierto a ser asignado a las cualquiera de las variantes que en ese momento sean testeadas.

Esto es lo más acotado que nos es posible realizar, ya que no conocemos ni las características del experimento ni las de la variante (problema y solución). Podría aplicar a una sección del sitio web en particular que el usuario nunca accede y por ello no verse impactado. Lo mejor que podemos hacer es asegurarnos que el usuario al menos entro al sitio y por lo tanto, tuvo oportunidad de estar expuesto.

Nuestro mecanismo de asignación

Hacemos uso de un mecanismo de asignación de tipo client side:

Más allá de ser el único realmente factible para nosotros al no tener acceso ni a los servidores ni al código fuente del sitio web original, es además particularmente apropiado para nuestro caso de uso ya que nuestras variantes son, justamente, scripts client-side.

Nuestro servidor funciona como servidor de asignación, y nuestra extensión se encarga automáticamente de ajustar el código Javascript del sitio para que realice un requerimiento web a dicho servidor. De esta manera, para cada navegación que el usuario realiza, el servidor de asignación retornará el código necesario para aplicar la variante asignada.

Nuestro sistema de recolección de métricas

Hacemos uso de un sistema de recolección de métricas especializado

Nuestro servidor recibe la información de la navegación de usuario y registra los votos que los mismos emiten y se los asocia a la variante que le corresponde al usuario para el experimento en el cual participó. Si bien el usuario no tiene conocimiento exacto de los cambios realizados, tiene acceso a un listado de los problemas a los cuales el mismo se asoció y para los cuales debiera haberse aplicado algún script que le de solución al mismo. La simple verificación del usuario respecto a la persistencia del problema o su resolución satisfactoria por medio del voto es la métrica que registramos y asociamos a la variante.

4.2.3 Evaluación de resultados.

Cuando el proceso de evaluación se cierra, se selecciona la variante ganadora por un método de mayoría simple. Este método de votos y de mayorías está lejos de ser el ideal, pero considerando la escasa información que poseemos respecto tanto del problema como de la solución, es lo suficientemente aproximado.

Pensemos un momento que es lo que sabemos respecto a lo que estamos evaluando. Sabemos en abstracto que existe un problema y múltiples soluciones las cuales intentamos evaluar. Cual es objetivo, la naturaleza exacta del problema no tenemos conocimiento. Cual es la hipótesis, de qué manera la solución espera resolverlo tampoco. Tampoco sabemos qué clase de impacto tiene tanto el problema como la solución en la experiencia de usuario. No sabemos qué métricas se ven impactadas por la presencia de dicho problema y que debieran mejorar una vez aplicada la solución, ni tampoco tendríamos acceso a las mismas ya que no tenemos relación alguna con los sitios web para los cuales se están aplicando los scripts de soluciones.

Todo esto es lo que determina que dependamos de un simple mecanismo de voto, y dependemos fuertemente del feedback consciente de los usuarios que participan en los experimentos.

Como hemos visto, el principal objetivo de un experimento A/B testing es poder sacar conclusiones respecto a las características de las variantes puestas a prueba, permitiendo un aprendizaje basado en hechos y para esto es importante poder saber que tan estadísticamente significativo fueron los resultados. La decisión termina siendo en definitiva, humana.

En nuestro caso, el usuario que tiene la capacidad de cerrar el proceso de evaluación deberá tener en cuenta al momento de cerrarlo la cantidad de feedback existente y contrastarlo con el tiempo durante el cual el proceso de evaluación estuvo corriendo. Es posible que la cantidad de votos recolectados al momento sea insuficiente como para poder obtener un resultado estadísticamente significativo, pero esto debe también contrastarse con las expectativas de los usuarios de ver solucionados los problemas para los cuales espera resolución.

5. Administración de la comunidad

Como se mencionó en la Introducción, esta tesina propone la mejora de la experiencia de navegación web haciendo uso de un enfoque *colaborativo*, es decir, que la misma comunidad de usuarios pueda realizar reclamos y proveer soluciones a la experiencia de navegación, dándole además el poder a la comunidad de autorregularse. Este capítulo describe cómo llevamos a cabo este objetivo. Las diferentes secciones describen: roles en una comunidad, evolución de los miembros a través de privilegios y definición de privilegios para este trabajo, concepto de reputación, características, y por último, cambios de reputación para la asignación automática de privilegios.

5.1 Evolución de los miembros

Amy Jo Kim sugiere que el ciclo de vida de las membresías de los usuarios de una comunidad tiene 5 etapas [Kim 2000]:

1. Visitantes: personas que no tienen una entidad persistente en la comunidad.
2. Principiantes: miembros nuevos que aún deben aprender cómo opera la comunidad y cómo relacionarse con los demás miembros. Para llegar a esta instancia, los visitantes deben pasar por un ritual de membresía.
3. Regulares: miembros establecidos que pueden cómodamente participar del día a día de la comunidad.
4. Líderes: voluntarios, empleados o cualquier tipo de personal que mantiene operativa a la comunidad. Para llegar a esta instancia, los regulares deben pasar por un ritual de liderazgo.
5. Ancianos: miembros regulares o líderes que llevan un tiempo significativo en el rol y se encargan de compartir su conocimiento y fomentar la cultura a de la comunidad a los demás miembros.

En este trabajo, tomamos los conceptos básicos de estas etapas mencionadas. Sin embargo, no los estructuramos estrictamente de esta manera. Como se verá a continuación, optamos por seguir un esquema similar al que implementa **StackOverflow.com**, donde los usuarios tienen la posibilidad de evolucionar su membresía pero no tanto en un esquema

"en cascada" como el que sugiere [Kim 2000], sino dando pequeños pasos de distintas etapas en paralelo, en la medida que sea posible.

5.1.1 Privilegios

Cuando hablamos de privilegios, en el contexto de este trabajo, hacemos referencia a la capacidad que se le puede dar a un usuario de realizar una determinada acción. Esta capacidad es pública, es decir, cualquier usuario puede determinar si otro usuario la tiene o no. Esto en sí mismo puede llegar a ser considerado como un incentivo para los usuarios.

En nuestra propuesta no existen roles. Es decir, no existe un usuario que estrictamente se dedique a probar soluciones, a desarrollarlas o a moderar la comunidad. Todos los usuarios son iguales y pueden acceder a los mismos privilegios. Por este motivo, se pueden ver entrelazados privilegios relacionados a moderación con otros relacionados a la gestión de procesos de evaluación.

A continuación se presentan los privilegios propuestos:

- **Controlar el flujo de los procesos de evaluación:** este privilegio es crítico en esta comunidad. El mismo permite controlar el inicio y fin de procesos de evaluación, siempre y cuando se respeten las restricciones correspondientes.
- **Acceder a herramientas de moderación:** este privilegio suele ser esencial en cualquier comunidad, ya que permite controlar la calidad de la información que se provee así como la penalización correspondiente a usuarios conflictivos que no se ajustan a las normas de la comunidad. Este privilegio incluiría la capacidad de aceptar o rechazar publicaciones marcadas como SPAM o que pueden requerir moderación, revisión de problemas duplicados, revisión de votos de edición de problemas, etc.
- **Rechazar soluciones:** permite remover una solución de un proceso de evaluación. Esto se puede deber a que una solución puede llegar a ser inapropiada o incluso peligrosa para los usuarios. El rechazo de la solución implica una penalización al autor de la misma.

- **Editar problema:** da la posibilidad de editar el título, descripción, ruta de un problema así como los etiquetas que se le hayan asignado, a usuarios que no sean los autores del problema.
- **Marcar como duplicado:** los usuarios pueden indicar que un nuevo problema ya se había cargado anteriormente y así desestimar el duplicado, poniendo foco en el original.
- **Enviar a la cola de moderación:** este privilegio de moderación involucra sugerir una publicación para ser moderada. La misma es luego procesada por aquellos usuarios que tengan el privilegio que les brinda acceso a las herramientas de moderación.
- **Agregar recompensa:** permite definir una recompensa para un problema. La misma se asignaría una vez encontrada una solución al problema.
- **Crear etiquetas:** las etiquetas a los problemas siempre pueden elegirse de un conjunto determinado. Este privilegio permite crear nuevas etiquetas a fin de que los usuarios novatos no generen cualquier tipo de etiquetas que no sean relevantes y dificulten el relacionamiento de los problemas.

Se analizó también la posibilidad de agregar un privilegio que permita (o no) agregar soluciones a problemas. Sin embargo, se optó por no avanzar con ello debido a que no es la intención limitar la publicación de soluciones sino fomentarla. De esa manera, nuevos usuarios de la plataforma, que tengan habilidad para desarrollar soluciones, podrían hacerlo sin restricciones.

Otro privilegio que se decidió descartar fue el que permitiría crear un nuevo proceso de evaluación para un problema. El mismo sería necesario cuando una solución, por el motivo que sea, dejase de funcionar. De esta manera, los usuarios que contaran con el privilegio podrían solicitar que se cree un nuevo proceso de evaluación a fin de ir aceptando soluciones cuanto antes. El motivo por el cual este privilegio perdió prioridad es que cuando una solución deja de funcionar se espera que aparezca una nueva en el menor tiempo posible. De esta manera, restringirle a los usuarios (que no tengan este privilegio) la capacidad de manifestar que el problema pasó a estar vigente nuevamente, sólo retrasaría la llegada de una nueva solución.

Cabe destacar que no todos los privilegios que se mencionaron han sido implementados en la herramienta, pero quedan previstos para ser agregados en el trabajo futuro.

5.1.2 Privilegios democráticos o autocráticos

A lo largo del desarrollo de este proyecto, hemos detectado distintas acciones que deberían poder realizar los usuarios para mantener operativo el sistema, como reportar problemas, proponer soluciones, votar, etc. Por defecto, consideramos que un usuario debería poder realizar una acción, a menos que se justifique restringirla o limitarla de alguna manera, que es justamente donde aparecen los privilegios.

Sin embargo, los privilegios, que son incentivos naturales en la comunidad, ya que dan una suerte de "status" al usuario, pueden llegar a tener un impacto negativo en la comunidad en caso de que se usen de manera inapropiada. Consideremos la siguiente situación. Un usuario A propone una solución a un determinado problema. Poco tiempo después, un usuario B propone una solución más eficiente, breve, flexible y abarcativa que la que propuso el usuario A para el mismo problema. El usuario A, con la única motivación de llevarse él mismo el crédito por encontrar una solución al problema en cuestión, hace uso de su privilegio para rechazar soluciones y, de esa manera, la solución del usuario B se remueve del proceso de evaluación.

Claro está que, si un usuario tiene privilegios de impacto significativo en la comunidad, necesariamente tuvo que haberse ganado su reputación y es altamente probable que no caiga en este tipo de prácticas. Sin embargo, consideramos que de todos modos nuestra solución tiene que proveer una forma de prevenir este tipo de situaciones que no solamente tienen orígenes en los factores emocionales de los usuarios sino que también pueden deber simplemente a errores que cometan a medida que hagan uso del sistema.

Este tipo de privilegios que le permiten a un usuario realizar por su cuenta una determinada acción los denominamos **privilegios autocráticos**.

Volviendo a nuestra comparativa con StackOverflow, éste toma un enfoque distinto para la ejecución de acciones limitadas por privilegios. Su aproximación a este problema apunta a la distribución de la responsabilidad de la ejecución de acciones destructivas. Es decir, aquellas acciones que impliquen una eliminación, modificación u ocultamiento de

contenido, entre otras, no pueden ser llevadas a cabo sólo por un usuario que tenga un determinado privilegio.

Para StackOverflow, la implementación de esta idea se resolvió permitiendo a los usuarios votar para realizar una acción. Es decir, en su esquema de preguntas y respuestas, un usuario puede votar para que una respuesta de baja calidad sea eliminada pero no puede por sí mismo eliminarla. Al llegar a una determinada cantidad de votos, la misma es eliminada (o se oculta). De esto se deduce que en este sitio no se maneja un privilegio que permita eliminar respuestas sino uno que permita votar para eliminarlas.

A continuación se muestra una comparativa de cómo ven dos distintos usuarios una misma respuesta. El primero es un usuario que no tiene el privilegio para eliminar respuestas:

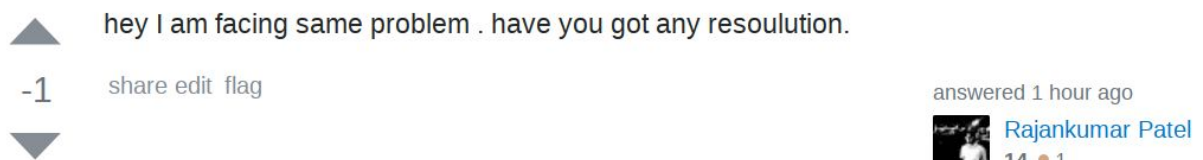


Figura 5.1.2.1: Contenido visto por usuario sin privilegios.

A continuación se puede ver cómo un usuario que tiene el privilegio en cuestión ve el mismo contenido pero además tiene la funcionalidad de votación:

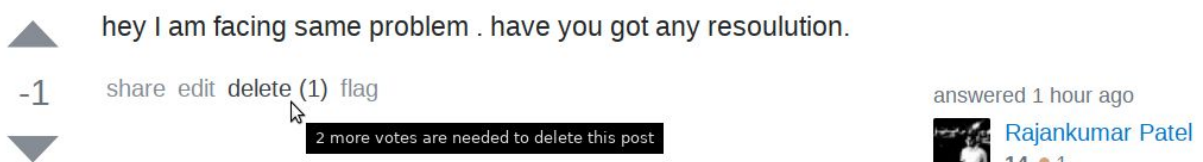


Figura 5.1.2.2: Contenido visto por usuario con privilegios de votación para borrar.

En este caso de eliminación de respuestas, StackOverflow consideró que con 3 votos debería ser suficiente para la eliminación de contenido.

A fin de completar el proceso, así es como se ve exactamente esta misma respuesta una vez eliminada:

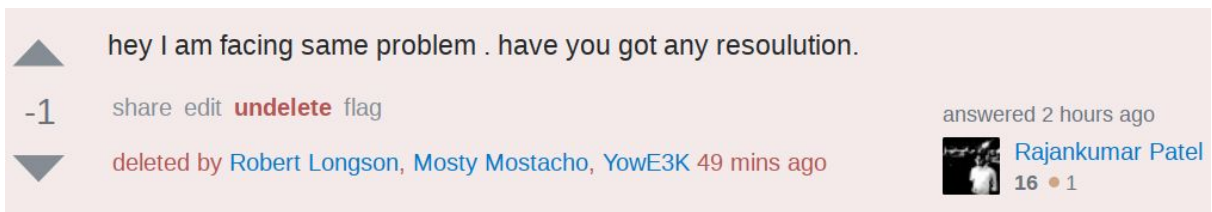


Figura 5.1.2.3: Contenido eliminado visto por usuario con privilegios de votación para borrar.

Las respuestas eliminadas muestran cuándo fueron eliminadas y qué usuarios votaron para eliminarla. Otro punto interesante es que las mismas sólo pueden ser vistas por usuarios que tengan el privilegio de eliminación: aquellos usuarios que no lo tienen, simplemente no verán esta respuesta.

Este tipo de privilegios que requieren la colaboración de varios usuarios para realizar una acción los denominamos **privilegios democráticos**.

Nosotros hemos optado hacer uso de un esquema de privilegios híbrido, donde tenemos tanto privilegios democráticos como autocráticos. Es probable que la determinación de un mejor esquema esté ligada a la cantidad de usuarios que haya en el sistema: a menos usuarios, los privilegios autocráticos tienen más sentido mientras que a más usuarios los privilegios democráticos tienden a ser más necesarios.

5.1.3 Reputación

A lo largo de este capítulo se ha hecho mención a los privilegios y a las capacidades adicionales que proveen a los usuarios. Sin embargo, no se ha hablado de cómo es que se obtienen.

En una primera instancia consideramos la posibilidad de definir privilegios manualmente. Es decir, que un determinado usuario tenga la capacidad de realizar una determinada acción. Esto, claramente, permite un fino control de a qué usuario se le asigna qué capacidad. Sin embargo, no escala.

Con una cantidad limitada de usuarios, definir privilegios por cada uno no es una tarea ardua, pero conforme la cantidad de usuarios se incrementa, la necesidad de más usuarios con privilegios adicionales se incrementa. Si bien asignar privilegios parece una tarea

sencilla, sigue siendo una tarea adicional que otro usuario con más privilegios debería realizar y en la cual invertir su valioso tiempo.

Por ello, y en vista de que la solución provista debe ser escalable, apuntamos a un esquema de privilegios obtenidos de manera automática, determinados en base a la reputación que hayan tenido los usuarios en la comunidad. Por este motivo es importante responder la siguiente pregunta: ¿cómo se mide la reputación?.

Antes de responder la pregunta, intentaremos explicar cuáles son las características que consideramos que debe tener la reputación:

- **Debe ser objetiva y estar claramente definida:** no puede estar ligada a impresiones u opiniones personales de otros usuarios. La reputación no debe estar sujeta a discusión. Adicionalmente, todos los usuarios en la comunidad deben entender exactamente qué acciones incrementan su reputación en la comunidad y cuál es el impacto de esas acciones en la misma.
- **Debe ser fácil de entender y comparar:** no debería ser necesario tener que realizar operaciones complejas para entender la reputación que tiene un usuario en la comunidad. Cualquier usuario (incluso uno visitante) debería tener la capacidad de entender la reputación que un miembro de la comunidad con el que está interactuando tiene. Esto puede brindar seguridad adicional al usuario.
- **Debe ser pública:** nada se gana con mantener la reputación secreta. Por el contrario, los usuarios menos frecuentes no podrían discernir entre usuarios altamente activos en la comunidad y aquellos que no. El hecho de que un usuario tenga una gran cantidad de reputación en la comunidad no asegura que la calidad del contenido que genera el mismo sea alto. Sin embargo, es muy probable que el contenido sea útil. El entender esto le brinda, a un usuario ajeno a la comunidad, la capacidad de valorar más las interacciones con algunos usuarios que las que tenga con otros. Para ello, esta información debe ser pública y accesible.
- **Debe estar relacionada al grado de participación en la comunidad y también a la calidad de la interacción:** la idea es fomentar un incremento de las interacciones pero que esto no suceda en detrimento de la calidad de las mismas. Por ello, se

espera que la reputación permita medir ambos aspectos, pero no estrictamente uno o el otro ya que sería irrelevante.

La idea de poder medir la reputación, da a entender la necesidad de que la misma tenga algún valor ordenable, idealmente numérico. Esto cumpliría automáticamente con la característica de que la reputación sea fácil de entender y de comparar.

A fin de que la reputación esté claramente definida, no puede ser asignada por ningún usuario arbitrariamente. Por ello, es necesario enumerar las acciones concretas que cada usuario debe realizar en la comunidad para incrementar su reputación. A continuación se listan las acciones que un usuario puede realizar y eventos que se pueden generar, que tienen un impacto en la reputación. Cabe destacar que algunas de las mismas no están implementadas en el proyecto pero se considerarán como trabajos a futuro.

- **Recibir un voto positivo en una solución:** el incremento de la reputación se da cuando un usuario que haya publicado una solución a un problema recibe un voto positivo emitido por otro usuario que haya probado esa misma solución. El incremento de la reputación se basa en que, dado que el usuario que probó la solución está votándola positivamente, se supone que la solución provista es buena o cumple su cometido. La idea es motivar al usuario que escribió una buena solución a seguir haciéndolo.
- **Recibir un voto negativo en una solución:** este caso es igual al anterior, pero implica un voto negativo, por lo que se produce un decremento en la reputación del usuario receptor del voto. El mismo se basa en que se entiende que la solución provista realmente no está resolviendo el problema como se espera. Hay una infinidad de motivos por los cuales eso puede pasar (completitud de la solución, qué tan general es, eficiencia, omisiones, etc.) pero lo importante es que el usuario receptor del voto entienda que es importante resolver el problema apropiadamente y que lo debe tener en cuenta al momento de publicar una solución.
- **Recibir un voto en un problema:** cuando un usuario vota un problema, el usuario que lo publicó recibe un incremento en su reputación. La intención es fomentar la publicación de problemas, premiando más a los problemas más populares, ya que recibirían más votos.

- **Votar en una solución:** este incremento de la reputación se da en el usuario que emite el voto en una solución. La motivación del mismo es justamente fomentar que los usuarios voten soluciones. Los votos no solo permiten incrementar la reputación y la motivación de los usuarios que publican soluciones, sino que también permiten realizar un ranking de las mismas, a fin de poder seleccionar aquella más votada.
- **Determinar resultado de un proceso de evaluación:** la selección de una solución se calcula automáticamente en base a los votos que tienen todas las soluciones de un mismo problema. Este proceso se dispara al finalizar un proceso de evaluación. Aquella solución que tenga la mayor cantidad de votos será la seleccionada. El autor de la misma, además de haber recibido el incremento en su reputación por los votos recibidos también recibe un incremento en su reputación fijo. Este incremento en la reputación tiene dos motivaciones. Por un lado, está ligado no sólo a premiar e incentivar a aquellos autores de buenas soluciones a mantener un buen nivel de calidad en las mismas, como pasa con los votos de soluciones, sino también a incentivarlos a que se superen aún más a fin de incrementar lo más que se pueda la calidad de las soluciones. Por otro lado, el incremento de reputación fijo también es un incentivo para que se publiquen soluciones a problemas poco populares. Esto se debe a que los problemas poco populares recibirán pocas soluciones y pocos votos, lo cual podría desmotivar a los usuarios a invertir su tiempo en proponer soluciones. El hecho de que se obtenga un incremento de reputación fijo, independientemente de que tantos votos tengan las soluciones debería compensar esa situación.
- **Recibir una recompensa:** si al momento de finalizar un proceso de evaluación, éste tiene una recompensa asignada, entonces se le incrementa la reputación al usuario autor de la solución seleccionada tanto como indique la recompensa. Notar que esto es un adicional al valor fijo incrementado por la selección de la mejor respuesta. La motivación es incentivar a los usuarios a proveer soluciones de alta calidad que resuelvan un problema específico.
- **Crear una recompensa:** este caso es el complemento del anterior, por el cual los usuarios que necesitan que se resuelva un problema con urgencia pueden asignar parte de su reputación al mismo. Este evento decrementa la reputación del usuario que asigna la recompensa. La motivación es la misma que se explicó en el punto anterior.

- **Borrado de una solución:** el borrado de una solución se debe a que la misma se considera perjudicial para la comunidad por algún motivo. Independientemente de si el perjuicio es intencional o no, el usuario debe recibir una penalización en su reputación. La motivación del decremento en la reputación es incentivar a los usuarios a proveer soluciones de buena calidad y desmotivar a los usuarios malintencionados de publicar soluciones perjudiciales para la comunidad.
- **Indicar que una llamada de atención para un administrador fue útil:** se da cuando un usuario con privilegios de administrador marca la llamada de atención que generó otro usuario sobre el contenido de una publicación como útil, generando así un incremento en la reputación del usuario que realizó la llamada de atención. La motivación es fomentar que los usuarios que detectan contenido inapropiado publicado en la comunidad tomen la iniciativa de realizar un llamado de atención por el mismo.
- **Indicar que una llamada de atención para un administrador no fue útil:** este caso es el opuesto al anterior. La situación que se da es similar a la anterior, salvo que la llamada de atención se realiza sobre una publicación cuyo contenido es apropiado. En este caso se procede a decrementarle la reputación al usuario que realizó la llamada de atención. La motivación es desmotivar a los usuarios a realizar llamadas de atención innecesarias.

Un aspecto relevante en los cambios de reputación puede ser aplicar un tope o límite en el impacto de la misma para operaciones que se pueden realizar más de una vez, como por ejemplo, recibir varias veces votos en una solución. Esto implicaría que, dada una determinada cantidad de votos recibidos en una solución, la reputación se incrementaría hasta esa cantidad de votos recibida. Cualquier voto recibido luego, se registraría como tal, pero no incrementaría o decrementaría la reputación. De esta manera, evitaríamos que los usuarios puedan recibir un impacto muy significativo en su reputación por realizar una única publicación que puede no ser representativa de publicaciones pasadas o incluso futuras. Al mismo tiempo, esto evitaría que los usuarios dejen de publicar soluciones por haber hecho un par de excelentes soluciones que hayan incrementado sustancialmente su reputación. Es decir, es bueno que se publiquen un par de soluciones excelentes pero es mucho mejor que se publiquen varias soluciones buenas.

Una cuestión que se evaluó a lo largo de este trabajo fue si el hecho de hacer uso de una solución debía incrementar la reputación del usuario que la usa o no. Esto implicaría, en términos prácticos, llevar cuenta de cuantas veces cada usuario ejecutó cada solución e incrementar la reputación de manera acorde. Un argumento a favor de formalizar el incremento en la reputación es incentivar a los usuarios a probar soluciones. Sin embargo, no es exactamente probar soluciones lo que hace andar al motor de procesos de evaluación sino los votos en las soluciones, los cuales ya tienen un incentivo asociado.

6. API

Una API (Application Programming Interface) no es más que un mecanismo que permite a aplicaciones comunicarse entre sí. Define los canales y las reglas que gobiernan el intercambio de mensajes entre ellas. Aunque en el contexto de esta tesina no parezca evidente a simple vista que alguna aplicación de terceros pueda llegar a tener que interactuar con nuestro sistema, comentaremos en este capítulo posibles usos para la API desarrollada, al tiempo que la describiremos en detalle.

6.1 Detalle de la API desarrollada

La API que se construyó a lo largo de esta tesina está basada en una arquitectura cliente-servidor que hace uso de servicios web para llevar a cabo la comunicación. Si bien no es puramente REST, mantiene, en mayor o en menor medida, algunas características de esta arquitectura, como el hecho de que es stateless (el servidor no mantiene estado entre requerimientos HTTP), es cacheable, está basada en un sistema de capas y, con algunas excepciones, mantiene una interfaz uniforme.

6.1.1 Autenticación

Algunas llamadas a nuestra API requieren autenticación. La misma se basa en un token que se debe agregar como header a las llamadas que lo requieran. Este token se obtiene desde el sitio web. El usuario, una vez registrado, puede acceder desde el perfil del usuario a un administrador de tokens. Desde allí debe generar uno y tomar nota del mismo a fin de usarlo al momento de consumir la API.

Una llamada a la API que requiera autenticación tendrá la siguiente forma:

```
GET /api/<endpoint> HTTP/1.1
Accept: application/json
Host: <host>
Authorization: Bearer <token>
Content-Type: application/x-www-form-urlencoded
```

Las referencias son las siguientes:

- <endpoint>: corresponde a la llamada específica que se desea hacer de la API. Por ejemplo, `profile`.
- <host>: es el host en el cual se esté ejecutando el servidor web
- <token>: es el token de acceso necesario para realizar las llamadas a la API que requieren autenticación. Puede contener más de mil caracteres alfanuméricos, guiones y guiones bajos.

La API realiza respuestas en formato JSON. Por eso es conveniente definir el header `Accept` correspondiente.

Una llamada a la API que requiera autenticación y que no contenga las credenciales apropiadas resulta en una respuesta de la siguiente forma:

```
{
  "error": "Unauthenticated."
}
```

6.1.2 Llamadas a la API

En esta sección se van a listar todas las llamadas que se pueden realizar a nuestra API. Cabe destacar que estos mismos servicios son las que consume la extensión del navegador.

A fin de que no se extienda demasiado esta sección se suprimieron los campos de fecha de creación (`created_at`) y de fecha de actualización (`updated_at`) de los objetos presentes en las respuestas de las llamadas.

[Ver perfil de usuario](#)

```
GET /api/profile
```

Retorna información del perfil de usuario.

Esta llamada requiere autenticación.

Ejemplo de respuesta:

```
{
```

```
"id": 1,
"name": "Martín Perez",
"email": "martin@server.com",
"points": 123456,
"apply_solutions_automatically": true
}
```

[Ver etiquetas de interés para el usuario](#)

GET `/api/profile/tags`

Retorna un arreglo de objetos que representan etiquetas en las que el usuario manifestó su interés. Además, se incluye un pivote que representa la relación con el objeto usuario. Esta información permite determinar, por ejemplo, la fecha en la cual el usuario indicó su interés en una determinada etiqueta.

Esta llamada requiere autenticación.

Ejemplo de respuesta:

```
[
  {
    "id": 1,
    "name": "usabilidad",
    "pivot": {
      "user_id": 1,
      "tag_id": 1
    }
  },
  {
    "id": 11,
    "name": "daltonismo",
    "pivot": {
      "user_id": 1,
      "tag_id": 11
    }
  }
]
```

[Ver problemas asociados](#)

GET `/api/profile/associated-problems`

Retorna un arreglo de objetos que representan problemas. Para cada uno se incluye información de la asociación que tienen los mismos al usuario y a la solución que se escogió para ese usuario y problema. Notar que la solución que se retorna no representa la solución aceptada para ese problema sino la solución que se le asignó al usuario autenticado. En caso de que aún no se le haya asignado una solución al usuario en autenticado, entonces la misma se retorna como nula.

Esta llamada requiere autenticación.

Ejemplo de respuesta:

```
[
  {
    "id": 1,
    "title": "Generic problem",
    "description": "I want to print something in the console",
    "matching_route": "*",
    "user_id": 1,
    "problem_solution_user": [
      {
        "problem_id": 1,
        "user_id": 1,
        "solution_id": 41,
        "solution": {
          "id": 41,
          "description": "Fixed using a flexible XPath",
          "url_pattern": ".*",
          "script": "console.log('Solution running');",
          "user_id": 1,
          "evaluation_process_id": 21
        }
      }
    ]
  }
]
```


[Ver soluciones asociadas](#)

GET `/api/profile/associated-solutions`

Retorna un arreglo de objetos que representan soluciones. Las mismas son soluciones que corresponden a problemas que el usuario se haya asociado.

Esta llamada requiere autenticación.

Ejemplo de respuesta:

```
[
  {
    "id": 5,
    "description": "Fixed using an extremely flexible XPath",
    "url_pattern": ".*",
    "script": "console.log('A simple solution running');",
    "user_id": 1,
    "evaluation_process_id": 3,
    "problem_id": 1,
    "solution_id": 5
  },
  {
    "id": 7,
    "description": "Fixed using decent programming practices",
    "url_pattern": ".*youtube.com.*",
    "script": "console.log('Running solution for problem');",
    "user_id": 1,
    "evaluation_process_id": 4,
    "problem_id": 2,
    "solution_id": 7
  }
]
```

[Ver problema](#)

GET `/api/problems/{problem}`

Retorna un objeto que representa un problema. Se incluye también como resultado el arreglo de procesos de evaluación asociados al mismo.

El estado de los procesos de evaluación se retornan en base a la siguiente referencia:

- "A": aceptando soluciones
- "T": evaluando soluciones
- "F": finalizado

Esta llamada no requiere autenticación.

Ejemplo de respuesta:

```
{
  "id": 1,
  "title": "Generic problem",
  "description": "I want to print something in the console",
  "matching_route": "*",
  "user_id": 1,
  "evaluationProcesses": [
    {
      "id": 3,
      "started_at": "2017-09-08 20:19:27",
      "finished_at": null,
      "status": "T",
      "problem_id": 1,
      "selected_solution_id": null
    },
    {
      "id": 2,
      "started_at": "2017-05-03 14:33:25",
      "finished_at": "2017-06-11 23:18:57",
      "status": "F",
      "problem_id": 1,
      "created_at": "2017-04-30 12:38:47",
      "selected_solution_id": 3
    }
  ]
}
```

[Ver etiquetas de problema](#)

```
GET /api/problems/{problem}/tags
```

Retorna un arreglo de objetos que representan etiquetas que se encuentran asociadas al problema. Además, se incluye un pivote que representa la relación en sí misma con el problema. Esta información permite determinar, por ejemplo, la fecha en la cual se asoció la etiqueta al problema.

Esta llamada no requiere autenticación.

Ejemplo de respuesta:

```
[
  {
    "id": 1,
    "name": "color-blind",
    "pivot": {
      "problem_id": 1,
      "tag_id": 1
    }
  },
  {
    "id": 2,
    "name": "bitcoin-mining",
    "pivot": {
      "problem_id": 1,
      "tag_id": 2
    }
  }
]
```

[Crear problema](#)

POST `/api/problems`

Crea un problema. Se deben enviar los datos básicos del problema, etiquetas y luego el sistema lo asocia al usuario autenticado. Retorna la información básica del problema creado.

Parámetros:

- title: string (requerido). Título del problema.
- description: string (requerido). Descripción del problema.

- `tag_list[]`: array[int] (requerido). Arreglo de identificadores de etiquetas asociadas al problema. Debe haber al menos una.
- `matching_route`: string (opcional). Ruta a la que aplica el problema.

Esta llamada requiere autenticación.

Ejemplo de respuesta:

```
{
  "title": "Download YouTube videos",
  "description": "I would like to download videos from YouTube",
  "matching_route": "youtube.com",
  "user_id": 1,
  "id": 7
}
```

En caso de que haya información faltante se retorna el error correspondiente. Por ejemplo:

```
{
  "title": [
    "The Title field is required."
  ]
}
```

Actualizar problema

```
PUT /api/problems/{problem}
```

Actualiza un problema. Se deben enviar los datos básicos del problema y etiquetas. Retorna la información básica del problema actualizado. La documentación es similar a la llamada "Crear problema".

Sugerencias de problemas

```
GET /api/problems/suggestions
```

Retorna problemas sugeridos al usuario autenticado en base a una determinada URL. La sugerencia se basa en problemas a los que el usuario no se encuentre asociado, que tengan

rutas que coincidan con la URL de entrada y que tengan al menos una etiqueta en la que el usuario haya indicado interés.

Parámetros:

- url: string (requerido). URL para la cual se filtrarán los problemas relacionados..

Esta llamada requiere autenticación.

Ejemplo de respuesta:

```
[
  {
    "id": 7,
    "title": "Download YouTube videos",
    "description": "I want to download videos from YouTube",
    "matching_route": "youtube.com",
    "user_id": 1,
    "solution": null,
    "evaluation_process": {
      "id": 9,
      "started_at": "2018-02-17 18:10:58",
      "finished_at": null,
      "status": "T",
      "problem_id": 7,
      "created_at": "2018-02-17 17:22:04",
      "selected_solution_id": null
    }
  }
]
```

Asociación a problemas

POST /api/problems/{problem}/associate

Asocia el usuario autenticado al problema ingresado. Si el problema ingresado no existe se retorna un código de error HTTP 404. Si el problema ingresado existe se realiza la asociación y se retorna un código de error HTTP 200 con un cuerpo como se muestra debajo.

Esta llamada requiere autenticación.

Ejemplo de respuesta:

```
{
  "result": {
    "code": "success"
  }
}
```

Desasociación de problemas

POST `/api/problems/{problem}/dissociate`

Desasocia el usuario autenticado al problema ingresado. La documentación es similar a la llamada "Asociación de problemas".

Ver proceso de evaluación

GET `/api/evaluations/{evaluation}`

Retorna información del proceso de evaluación indicado, incluyendo las soluciones que tiene asociadas.

Esta llamada no requiere autenticación.

Ejemplo de respuesta:

```
{
  "id": 3,
  "started_at": "2017-09-08 20:19:27",
  "finished_at": null,
  "status": "T",
  "problem_id": 1,
  "created_at": "2017-08-30 09:40:56",
  "selected_solution_id": null,
  "solutions": [
    {
      "id": 5,
      "description": "Fixed using a flexible XPath",
      "url_pattern": ".*",
      "script": "console.log('Running solution #5');",
    }
  ]
}
```

```
        "user_id": 1,  
        "evaluation_process_id": 3  
    }  
]  
}
```

[Ver solución](#)

```
GET /api/solutions/{solution}
```

Retorna información de la solución indicada.

Esta llamada no requiere autenticación.

Ejemplo de respuesta:

```
{  
  "id": 5,  
  "description": "Fixed using an extremely flexible XPath",  
  "url_pattern": ".*",  
  "script": "console.log('Running solution #5');",  
  "user_id": 1,  
  "evaluation_process_id": 3  
}
```

[Ver script solución](#)

```
GET /api/solutions/{solution}
```

Retorna el script de la solución indicada. La respuesta HTTP tiene el Content-Type **application/javascript**.

Esta llamada no requiere autenticación.

[Crear solución](#)

```
POST /api/solutions/{problem}
```

Crea una solución. Se deben enviar los datos básicos de la solución y luego el sistema la asocia al usuario autenticado y al problema indicado. Retorna la información básica de la solución creada.

Parámetros:

- description: string (requerido). Descripción de la solución.
- url_pattern: string (requerido). Expresión regular que determina si la solución se debe ejecutar para una determinada URL o no.
- script: string (requerido). El script a ejecutar de la solución.

Esta llamada requiere autenticación.

Ejemplo de respuesta:

```
{
  "description": "This solution solves the issue using DOM listeners, that are more performant than polling for DOM changes",
  "url_pattern": ".*youtube.*",
  "script": "console.log(\"Testing script\")",
  "user_id": 1,
  "evaluation_process_id": 7,
  "id": 14
}
```

[Actualizar solución](#)

```
PUT /api/solutions/{solution}
```

Actualiza una solución. Se deben enviar los datos básicos de la solución. Retorna la información básica de la solución actualizada. La documentación es similar a la llamada "Crear solución".

[Votar para crear un nuevo proceso de evaluación](#)

```
POST /api/problems/{problem}/vote-reopen
```

Permite al usuario autenticado votar para crear un nuevo proceso de evaluación en el problema indicado.

Parámetros:

- type: string (requerido). Puede ser el valor "R" o "N". El primero indica que se debe registrar el voto mientras que el segundo indica que se debe remover el mismo.

Esta llamada requiere autenticación.

Ejemplo de respuesta:

```
{
  "result": {
    "code": "success",
    "wasReopened": false,
    "reopenCount": 1
  }
}
```

En caso de error, la respuesta tiene la siguiente forma:

```
{
  "result": {
    "type": "problem-not-votable",
    "message": "The current evaluation process for the problem is not finished",
    "code": "error"
  }
}
```

Los posibles resultados son:

- success: el voto se actualizó con éxito.
- error: el voto no se actualizó con éxito y se especifica en la respuesta el motivo, el cual puede tomar los siguientes valores:
 - problem-not-votable: el proceso de evaluación no se encuentra finalizado.

[Votar solución](#)

POST `/api/solutions/{solution}/vote`

Permite al usuario autenticado votar la solución que tiene asignada. Se retorna información del voto y de la solución votada.

Parámetros:

- type: string (requerido). Puede ser el valor "U", "D" o "N". El primero indica que se debe registrar un voto positivo. El segundo indica un voto negativo. Mientras que el tercero indica que se debe remover el mismo.

Esta llamada requiere autenticación.

Ejemplo de respuesta:

```
{
  "result": {
    "code": "success",
    "vote": {
      "id": 2,
      "user_id": 1,
      "voteable_id": 1,
      "voteable_type": "TS",
      "type": "U"
    },
    "solution": {
      "id": 5,
      "description": "Want to download videos from YouTube",
      "url_pattern": ".*youtube.*",
      "script": "console.log(\"Testing script\")",
      "user_id": 1,
      "evaluation_process_id": 3
    }
  }
}
```

En caso de error, la respuesta tiene la siguiente forma:

```
{
  "result": {
    "code": "error",
    "type": "evaluation-process-not-tested-by-user",
    "message": "You have not tested any solution for the
evaluation process linked to the input solution"
  }
}
```

Los posibles resultados son:

- success: el voto se actualizó con éxito.
- error: el voto no se actualizó con éxito y se especifica en la respuesta el motivo, el cual puede tomar los siguientes valores:
 - solution-not-tested-by-user: el usuario autenticado no tiene asignada la solución votada.
 - evaluation-process-not-tested-by-user: el usuario autenticado no tiene asignada la solución votada y tampoco ninguna otra solución del proceso de evaluación al que corresponde la misma.
 - evaluation-process-not-testing-solutions: el proceso de evaluación no se encuentra en estado de prueba de soluciones.

Ver voto

GET `/api/solutions/{solution}/vote`

Permite al usuario autenticado ver la información del voto que haya realizado en la solución indicada.

Esta llamada requiere autenticación.

Ejemplo de respuesta:

```
{
  "result": {
    "vote": {
      "id": 2,
      "user_id": 1,
      "voteable_id": 1,
      "voteable_type": "TS",
      "type": "U"
    }
  }
}
```

Iniciar proceso de evaluación

POST `/api/evaluations/{evaluation}/start`

Permite al usuario autenticado iniciar el proceso de evaluación indicado.

Esta llamada requiere autenticación.

Ejemplo de respuesta:

```
{
  "result": {
    "code": "success"
  }
}
```

En caso de error, la respuesta tiene esta forma:

```
{
  "result": {
    "code": "error",
    "type": "evaluation-process-invalid-status",
    "message": "The evaluation process should be accepting
solutions"
  }
}
```

Los posibles resultados son:

- success: el proceso de evaluación se inició con éxito.
- error: el proceso de evaluación no se inició con éxito y se especifica en la respuesta el motivo, el cual puede tomar los siguientes valores:
 - evaluation-process-invalid-status: el proceso de evaluación no se encuentra aceptando soluciones
 - evaluation-process-has-not-enough-solutions: el proceso de evaluación no tiene la mínima cantidad de soluciones necesarias para ser iniciado.

Finalizar proceso de evaluación

```
POST /api/evaluations/{evaluation}/finish
```

Permite al usuario autenticado finalizar el proceso de evaluación indicado.

Esta llamada requiere autenticación.

Ejemplo de respuesta:

```
{
  "result": {
    "code": "success"
  }
}
```

En caso de error, la respuesta tiene esta forma:

```
{
  "result": {
    "code": "error",
    "type":
    "evaluation-process-has-no-solution-with-positive-score",
    "message": "The evaluation process must have at least one
    solution with positive score"
  }
}
```

Los posibles resultados son:

- success: el proceso de evaluación se finalizó con éxito.
- error: el proceso de evaluación no se finalizó con éxito y se especifica en la respuesta el motivo, el cual puede tomar los siguientes valores:
 - evaluation-process-invalid-status: el proceso de evaluación no se encuentra probando soluciones.
 - evaluation-process-has-no-solution-with-positive-score: el proceso de evaluación no tiene ninguna solución con un puntaje positivo.
 - evaluation-process-without-outstanding-solution-or-not-old-enough: el proceso de evaluación no tiene ninguna solución que tenga una diferencia significativa de puntaje con la segunda solución o no se llegó a cumplir el plazo mínimo de tiempo desde que se inició el proceso de evaluación.

Navegar

GET /api/navigate

Está llamada está en estrecha relación con la extensión de Google Chrome. La misma retorna información de problemas a los que el usuario se encuentra asociado y sugerencias de otros problemas a los que el usuario no se encuentre asociado.

Más específicamente, los problemas dentro del arreglo problemas asociados también deben tener una coincidencia entre su ruta y la URL provista. Asimismo, el patrón de URL de la solución asignada debe coincidir con la URL provista. Si el usuario tiene seleccionada la opción de aplicar soluciones automáticamente en su perfil, entonces se lo asocia a los problemas que se ajusten a las condiciones anteriores, pero que adicionalmente coincidan con sus intereses.

El arreglo de problemas sugeridos tiene las sugerencias de problemas para el usuario. Estos son problemas a los que no se encuentre asociados pero que tengan al menos alguna etiqueta que coincida con sus intereses.

Parámetros:

- url: string (requerido). URL que se está navegando. Se usa para filtrar los problemas utilizando la ruta de los mismos.

Esta llamada requiere autenticación.

Ejemplo de respuesta:

```
{
  "response": {
    "problems": [
      {
        "id": 1,
        "title": "Generic problem",
        "description": "I want to print in the console",
        "matching_route": "*",
        "user_id": 1,
        "solution": {
```

```

        "id": 5,
        "description": "Download YouTube videos",
        "url_pattern": ".*youtube.*",
        "script": "console.log(\"Testing script\")",
        "user_id": 1,
        "evaluation_process_id": 3,
        "vote": null,
        "evaluation_process": {
            "id": 3,
            "started_at": "2017-09-08 20:19:27",
            "finished_at": "2018-03-11 19:18:24",
            "status": "F",
            "problem_id": 1,
            "created_at": "2017-08-30 09:40:56",
            "selected_solution_id": 5
        }
    },
    "evaluation_process": {
        "id": 3,
        "started_at": "2017-09-08 20:19:27",
        "finished_at": "2018-03-11 19:18:24",
        "status": "F",
        "problem_id": 1,
        "created_at": "2017-08-30 09:40:56",
        "selected_solution_id": 5
    },
    "user_voted_reopen": false
}
],
"suggestions": [
    {
        "id": 2,
        "title": "YouTube problem",
        "description": "Want to change YT videos titles",
        "matching_route": "www.youtube.com",
        "user_id": 1,
        "solution": null,
        "evaluation_process": {
            "id": 4,
            "started_at": "2017-07-16 14:02:31",
            "finished_at": "2017-06-22 14:21:36",

```

```
    "status": "F",  
    "problem_id": 2,  
    "created_at": "2017-05-20 06:54:00",  
    "selected_solution_id": 7  
  }  
]  
}
```


6.2 Potenciales usos de la API

Hemos visto que a través de la API es posible acceder a la totalidad de las funcionalidades del sistema. Esto mejora la extensibilidad del mismo, permitiendo el desarrollo de aplicaciones que provean nuevas funcionalidades sin necesidad de tener acceso al código de nuestro sistema.

Por ejemplo, una extensión a nuestro sistema podría estar alojada en su propio servidor, consumir información de los problemas y soluciones existentes y mostrárselos a sus usuarios con el formato y filtros que considere más apropiado para sus fines. Podría también reportar problemas, crear soluciones y disparar procesos de A/B Testing sobre las mismas, obteniendo a posteriori los resultados de dicho proceso.

Teniendo la posibilidad de crear problemas, una extensión podrá especializarse en el análisis del comportamiento de usuario para la detección de problemas de usabilidad y la automatización del reporte de los mismos.

Al tener la posibilidad de crear soluciones, una extensión a nuestro sistema podría llegar a simplificar el desarrollo de las mismas, haciendo uso de librerías propias o incluso un catálogo de refactorings. Podría también generar soluciones que estén preparadas para interactuar con la extensión y reportar sobre su comportamiento para tener feedback inmediato sobre su utilización.

No ahondaremos más sobre este tema en esta sección, pero más ejemplos podrán encontrarse en la sección de trabajos a futuro del Capítulo 8.

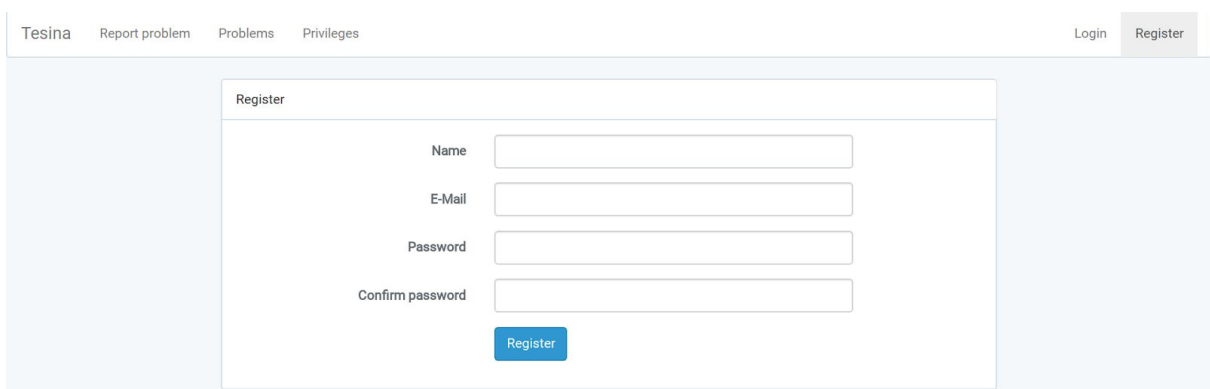
7. Uso de la aplicación

En este capítulo se presentarán las interfaces de usuario de la aplicación web y de la extensión de Google Chrome. También se explicará brevemente cómo se accede a las funcionalidades principales.

7.1 Aplicación web

La aplicación web es el principal punto de interacción con el usuario y permite realizar todas las operaciones necesarias, a excepción de la inyección de scripts de soluciones. En las siguientes secciones se explicarán las funcionalidades más relevantes.

7.1.1 Registración

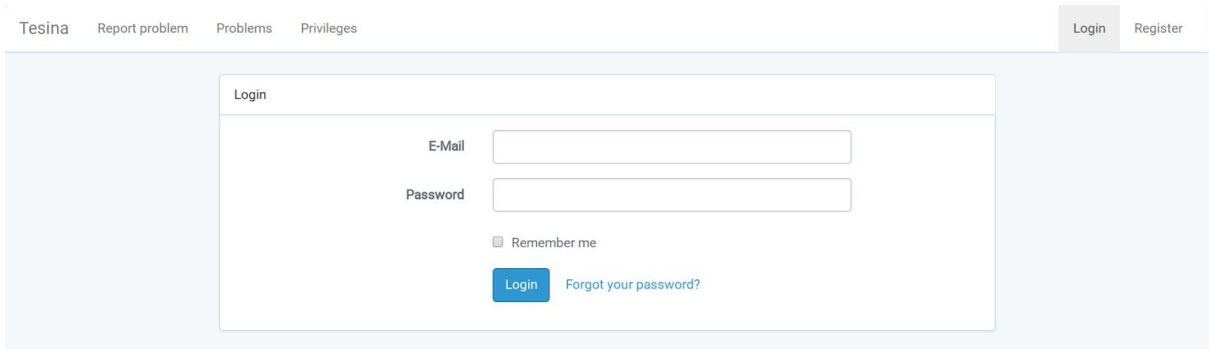


The image shows a web registration form. At the top, there is a navigation bar with links: 'Tesina', 'Report problem', 'Problems', and 'Privileges'. On the right side of the navigation bar, there are 'Login' and 'Register' buttons. The main content area is titled 'Register' and contains four input fields: 'Name', 'E-Mail', 'Password', and 'Confirm password'. Below these fields is a blue 'Register' button.

Figura 7.1.1.1: Formulario web de registraci3n.

Desde esta secci3n, los usuarios pueden darse de alta en el sistema. Se le solicitan al usuario los datos m3nimos y 3ste puede extenderlos desde su perfil de usuario. Tras ingresar los datos correspondientes, el usuario hace click en el bot3n de registraci3n, se realiza una validaci3n de los datos ingresados y, finalmente, se crea el usuario.

7.1.2 Inicio de sesión



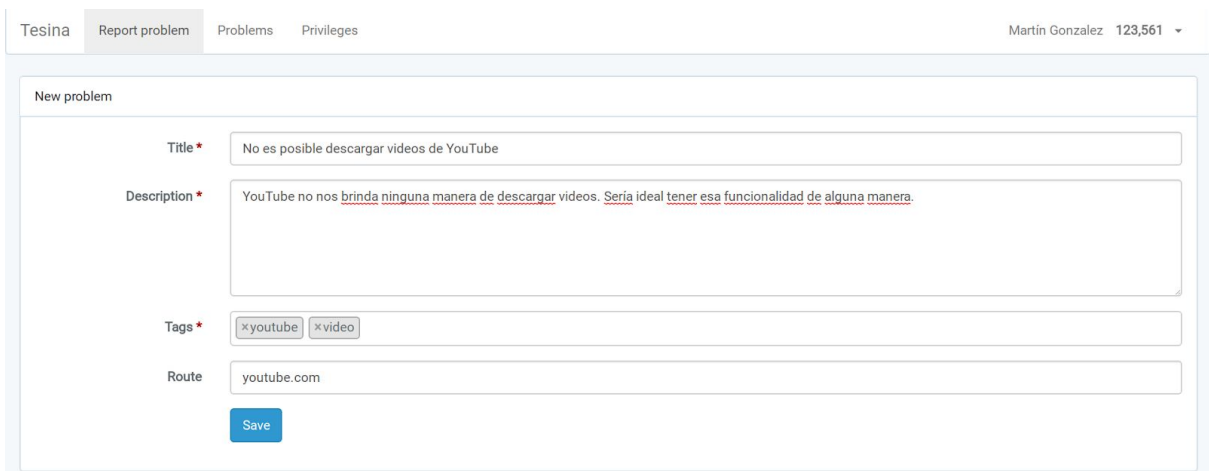
The screenshot shows a web interface with a navigation bar at the top containing 'Tesina', 'Report problem', 'Problems', and 'Privileges'. On the right side of the navigation bar are 'Login' and 'Register' buttons. The main content area features a 'Login' form with the following elements: an 'E-Mail' input field, a 'Password' input field, a 'Remember me' checkbox, a blue 'Login' button, and a link for 'Forgot your password?'.

Figura 7.1.2.1: Formulario web de inicio de sesión.

El inicio de sesión mantiene una estructura y funcionalidad clásicas. Solicita el ingreso de un email y una contraseña. Tras presionar el botón de inicio de sesión se intenta crear la sesión para el usuario en cuestión o se presentan los errores de validaciones correspondiente. La casilla de verificación permite extender la longitud de la sesión.

Se incluye un link al olvido de contraseña, el cual inicia el flujo correspondiente, pero se deja fuera de esta sección ya que tiene poca relevancia en el contexto de esta tesina.

7.1.3 Reportar un problema



The screenshot shows a web interface with a navigation bar at the top containing 'Tesina', 'Report problem', 'Problems', and 'Privileges'. On the right side of the navigation bar is a user profile 'Martin Gonzalez 123,561'. The main content area features a 'New problem' form with the following elements: a 'Title*' input field containing 'No es posible descargar videos de YouTube', a 'Description*' text area containing 'YouTube no nos brinda ninguna manera de descargar videos. Sería ideal tener esa funcionalidad de alguna manera.', a 'Tags*' input field containing 'xyoutube' and 'xvideo', a 'Route' input field containing 'youtube.com', and a blue 'Save' button.

Figura 7.1.3.1: Formulario web de reporte de problema.

El formulario de reporte de problema es simple. Solicita el ingreso de su título, descripción, etiquetas asociadas y la ruta a la que aplica el mismo. Tanto el título como la

descripción son de índole informativo. Las etiquetas permiten relacionar al problema con los intereses de los usuarios, como se explicó en capítulos anteriores.

La ruta se usa para identificar problemas relevantes en base a la navegación del usuario. Es decir, a medida que el usuario navega por internet, la extensión de Google Chrome consulta frecuentemente al servidor web (a través de la API) solicitando soluciones a aplicar para la URL que se esté navegando. Es justamente la ruta de los problemas el valor con el cual se compara la URL.

7.1.4 Listado de problemas

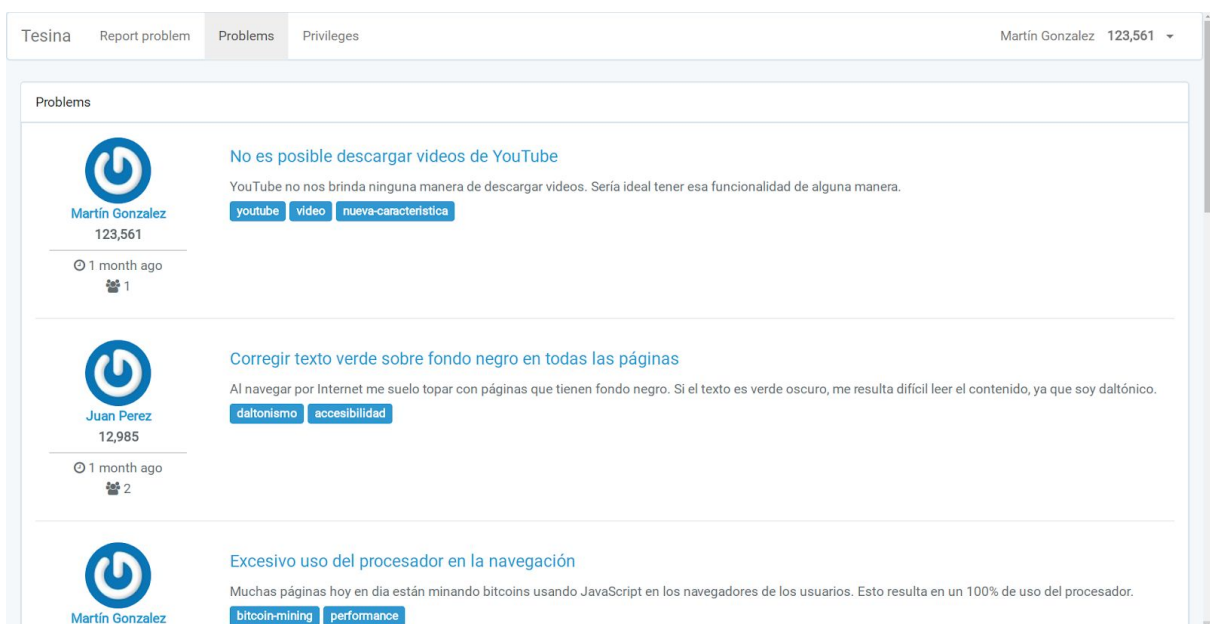


Figura 7.1.4.1: Listado de problemas.

El listado de problemas presenta todos los problemas que fueron creados. Muestra la información básica del problema (título, descripción, etiquetas y fecha de creación) así como información del usuario que lo reportó (nombre, avatar, puntos). Adicionalmente, debajo de la fecha de creación, se presenta la cantidad de usuarios que se asociaron al problema. Esta sección es accesible tanto por usuarios registrados como anónimos.

7.1.5 Ver problema

Excesivo uso del procesador en la navegación

Muchas páginas hoy en día están minando bitcoins usando JavaScript en los navegadores de los usuarios. Esto resulta en un 100% de uso del procesador.

Route: *

bitcoin-mining performance

1 month ago 2

Edit Dissociate from problem Add solution

Evaluation processes

Status	Created at	Started at	Finished at	Solutions	Solution author	Actions
Accepting solutions	2018-03-25 20:58:19			0		
Finished	2018-03-25 20:54:53	2018-03-25 20:55:32	2018-03-25 20:58:00	2	Juan Perez	
Finished	2018-02-17 17:22:04	2018-02-17 18:10:58	2018-03-25 20:54:19	2	System	

Figura 7.1.5.1: Sección de visualización de problema e historial de procesos de evaluación.

Al visualizar un problema, se puede apreciar la información básica del mismo (autor, descripción, ruta, etc.) así como información referente a los procesos de evaluación que tenga asociados. De estos últimos, cabe destacar que se muestran en orden cronológico descendente, siendo el primero el proceso de evaluación actual.

Cada proceso de evaluación se muestra con su estado, fechas y horas en las que cambió de estado, la cantidad total de soluciones que se crearon para el mismo y el autor de la solución seleccionada como la mejor, en caso de existir. A la derecha de cada proceso de evaluación hay dos botones: uno que permite visualizar el proceso de evaluación y otro que permite al usuario visualizar la mejor solución.

Respecto del problema en sí mismo, el usuario puede ver varios botones que le permiten interactuar con éste. Tanto el autor, como cualquier usuario con el privilegio de edición de problemas ve un botón que le permite editarlo. Los usuarios también ven un botón que les permite asociarse o desasociarse del problema.

Cuando el proceso de evaluación actual del problema se encuentra aceptando soluciones, los usuarios también ven un botón que les permite agregar soluciones a ese proceso de evaluación. Por otro lado, cuando el proceso de evaluación se encuentra finalizado, los

usuarios pueden ver un botón con un contador de votos, el cual se usa para votar para la creación de un nuevo proceso de evaluación y, cuando el contador llega a cero, efectivamente se concreta este paso.

7.1.6 Ver proceso de evaluación

Los procesos de evaluación pueden estar en tres posibles estados. La información que se presenta para cada estado es similar: datos propios del proceso de evaluación y del problema. Sin embargo, para cada estado, difieren las acciones que puede realizar el usuario sobre ese proceso de evaluación. Se muestran a continuación cómo se ven los procesos de evaluación para cada uno de esos estados.

Aceptando soluciones

The screenshot displays a web interface for an evaluation process. It is divided into three main sections: 'Evaluation process', 'Problem summary', and 'Solutions'.

- Evaluation process:** Shows metadata such as 'Creation date' (2018-03-25 20:58:19), 'Start date' (Not started), and 'Finish date' (Not finished). It indicates 'Amount of solutions' as 2 and 'Current process status' as 'Accepting solutions'. There are buttons for 'Add solution' and 'Start process'.
- Problem summary:** Features a blue circular icon with a power symbol. The title is 'Excesivo uso del procesador en la navegación'. The description states: 'Muchas páginas hoy en día están minando bitcoins usando JavaScript en los navegadores de los usuarios. Esto resulta en un 100% de uso del procesador.' There are tags for 'bitcoin-mining' and 'performance'. The user 'Martín Gonzalez' (123,571) is associated with the problem, and it was posted '1 month ago' with 2 votes.
- Solutions:** Lists two solutions. The first solution, by 'Juan Perez' (13,105), is titled 'Una solución eficiente que hace uso de un XPath más genérico'. Its script is `console.log("Solución genérica")` and its URL pattern is '*'. The second solution, by 'Martín Gonzalez' (123,571), is titled 'Solución que hace uso de DOM listeners para detectar cambios en el DOM que agreguen miners'. Its script is `console.log("Solución con DOM listeners")` and its URL pattern is '*'. Both solutions have 0 votes and were posted recently.

Figura 7.1.6.1: Proceso de evaluación aceptando soluciones

Éste es el estado inicial. En este estado cualquier usuario puede agregar soluciones al proceso de evaluación. En la imagen se puede ver un proceso de evaluación al cual se le agregaron dos soluciones.

Adicionalmente, aquellos usuarios que tienen el privilegio de controlar el flujo de los procesos de evaluación pueden ver un botón que les permite iniciar el proceso de evaluación, en caso de que el mismo esté en condiciones de ser iniciado.

Evaluando soluciones


Tesina Report problem Problems Privileges Martín Gonzalez 123,576

Evaluation process

Creation date: 2018-03-25 20:58:19
Start date: 2018-03-25 21:10:09
Finish date: Not finished
Amount of solutions: 3
Current process status: Testing solutions

[▶ Finish process](#)

Problem summary



Martín Gonzalez
123,576

1 month ago

2

Excesivo uso del procesador en la navegación

Muchas páginas hoy en día están minando bitcoins usando JavaScript en los navegadores de los usuarios. Esto resulta en un 100% de uso del procesador.


bitcoin-mining performance

Solutions

1

↑ 1

↓ 0



Martín Gonzalez
123,576

2 minutes ago

Solución que hace uso de DOM listeners para detectar cambios en el DOM que agreguen miners.

Script

```
console.log("Solución con DOM listeners")
```


URL pattern: *

[Show](#)

0

↑ 0

↓ 0



Juan Perez
13,110

5 minutes ago

Una solución eficiente que hace uso de un XPath más genérico

Script

```
console.log("Solución genérica")
```


URL pattern: *

[Show](#)

-1

↑ 0

↓ 1



System
0

45 seconds ago

Placebo solution

Script

URL pattern: *

[Show](#)

Figura 7.1.6.2: Proceso de evaluación evaluando soluciones.

94

Cuando un proceso de evaluación se encuentra evaluando soluciones ya no es posible agregar soluciones. Sin embargo, en esta instancia, el usuario ya puede ver la solución que se le asignó y también votarla. La solución asignada se muestra primero y se remarca con un recuadro verde. A la misma, y sólo a la misma, se le habilitan los botones de voto positivo y negativo.


Si el usuario cuenta con el privilegio de control de flujos de proceso de evaluación, entonces también va a ver un botón que le permite finalizar el proceso en cuestión, en caso de cumplir con las condiciones previamente establecidas.

Finalizado

Evaluation process

Creation date	2018-03-25 20:54:53
Start date	2018-03-25 20:55:32
Finish date	2018-03-25 20:58:00
Amount of solutions	2
Current process status	Finished

Problem summary


Martín Gonzalez
123,571
1 month ago
2



Excesivo uso del procesador en la navegación

Muchas páginas hoy en día están minando bitcoins usando JavaScript en los navegadores de los usuarios. Esto resulta en un 100% de uso del procesador.

[bitcoin-mining](#) [performance](#)


Solutions

2
↑ 2
↓ 0



Juan Perez
13,105
8 minutes ago

Placeholder for solution details (Script, URL pattern, etc.)

0
↑ 0
↓ 0


System
105
7 minutes ago

Placeholder for solution details (Script, URL pattern, etc.)

Figura 7.1.6.3: Proceso de evaluación finalizado

Éste es el último estado en el que puede estar un proceso de evaluación. Ya no es posible interactuar con el mismo; sólo se puede ver la información propia del proceso y de sus soluciones. A la mejor solución se la presenta primera en el listado y se la marca con una estrella. Notar que ya no se puede votar ninguna solución.

7.1.7 Agregar nueva solución

New solution

Description * Esta solución fue implementada usando listeners de DOM, que son más eficientes que usar la técnica de polling.

URL Pattern * youtube.com/videos

Script * console.log("Script en ejecución")

[Go to problem](#)

[Go to evaluation process](#)

Figura 7.1.7.1: Formulario web de alta de solución.

La solución en sí misma es una entidad simple. Al agregar una solución a un proceso de evaluación se le presenta al usuario un formulario que permite el ingreso de una breve descripción de la solución, una expresión regular para filtrar las URLs en las que se debe aplicar esta solución y el script correspondiente.

Debajo hay un par de enlaces que facilitan la navegación al problema y al proceso de evaluación correspondiente a esta solución. Finalmente, sólo falta mencionar el botón de guardado.

7.1.8 Privilegios

Tesina Report problem Problems **Privileges** Juan Perez 12,985 ▾

Privileges

Privileges control what you can do on on the site. Gain more privileges by increasing your reputation (points you receive from other users or performing some actions on the site).

Granted	Points	Privilege	Description
	20,000	Manage evaluation processes workflow	Fire the start and finish steps of evaluation processes
	18,000	Access moderation tools	Access to the moderation tools to review flags, problem edit votes, duplicated problem votes, etc.
	15,000	Reject solutions	Ability to vote to remove solutions
✓	10,000	Edit problem	Edit title, description, route and assigned tags in a not authored problem
✓	5,000	Mark as duplicate	Ability to vote to mark a given problem as a duplicate
✓	2,000	Flag posts	Vote for a post to be sent to the moderation queue
✓	1,000	Add bounty	Give personal points as a reward for a problem to be solved
✓	500	Create tags	Create new tags while creating a problem, rather than selecting them from a list

Figura 7.1.8.1: Sección de listado de privilegios.

Los usuarios del sistema van adquiriendo reputación a medida que realizan contribuciones. Desde esta sección los usuarios pueden ver el listado completo de privilegios incluyendo, para cada uno de ellos, los puntos necesarios para adquirirlos y una breve descripción de los mismos. A la izquierda, también se puede ver con un tilde verde aquellos privilegios que el usuario autenticado ha adquirido.

7.1.9 Perfil de usuario

The screenshot shows a user profile page for 'Juan Perez' with a score of 12,985. The page has a navigation bar with 'Tesina', 'Report problem', 'Problems', and 'Privileges'. The profile section includes fields for Name, Email, Gravatar, and Interests tags. A checkbox is checked for 'Apply solutions automatically based on my interests'. There are two password fields and a 'Save' button. Below the profile is a section for 'Personal access tokens' with a 'Create new token' link and a table showing one token: 'Juan Perez Access Token' with a 'Delete' link.

Name	Points
Juan Perez	12,985

Solutions contributed	Associated problems
4	3

Name	Action
Juan Perez Access Token	Delete

Figura 7.1.9.1: Formulario web con datos de perfil de usuario.

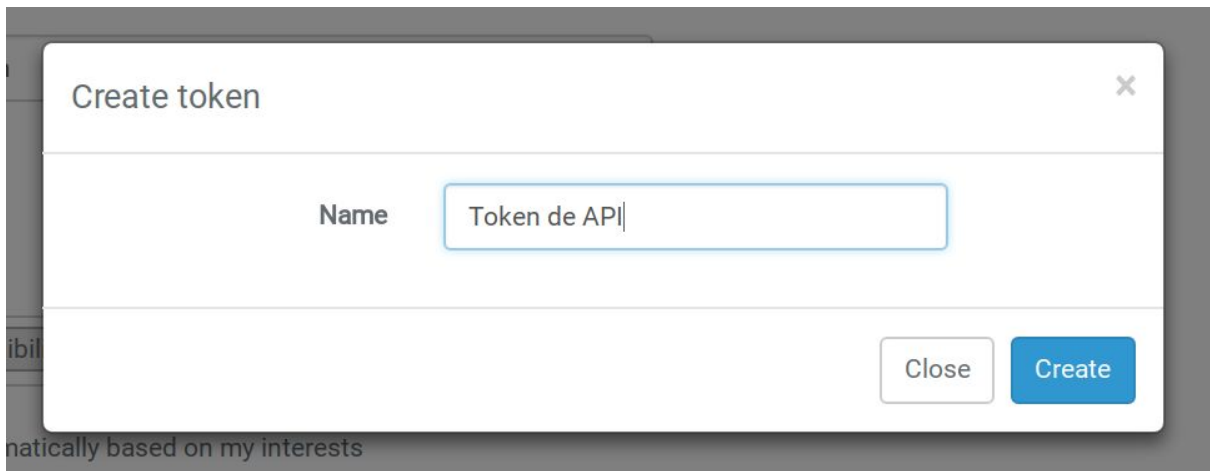
La información que se almacena del usuario es escasa, ya que, si bien nuestra herramienta modela una comunidad, nos interesa enfocarnos en los aportes que se generan por sobre los usuarios en sí mismos. En la parte superior se puede ver que el usuario puede modificar su información de contacto y las etiquetas de interés, usadas para la asociación automática a problemas y para mostrar sugerencias. Debajo, se encuentra la casilla de verificación que determina si efectivamente este usuario desea o no activar la asociación automática de problemas en la navegación.

Desde esta sección, el usuario puede cambiar su contraseña y ver algunos datos referentes a su grado de colaboración en la comunidad como los puntos obtenidos, la cantidad de soluciones propuestas y la cantidad de problemas a los que se haya asociado. El botón de guardado permite salvar los cambios realizados.

En la parte inferior del perfil de usuario se puede ver la sección de gestión de tokens de acceso de la API. Estos son los tokens que se pueden generar para efectivamente consumir

las llamadas de la API de manera autenticada. Esta sección presenta un listado, que sólo muestra el nombre del token, un botón para crearlos y otro para eliminarlos o revocarlos.

Creación de tokens de acceso



The image shows a modal window titled "Create token" with a close button (X) in the top right corner. The main content area contains a label "Name" followed by a text input field containing the text "Token de API". At the bottom right of the modal, there are two buttons: "Close" and "Create".

Figura 7.1.9.2: Formulario web de ingreso de nombre del nuevo token.

Cuando desde el listado de tokens se presiona el botón de creación de token se presenta esta ventana modal. La misma solicita el ingreso del nombre del token, el cual se guarda sólo para facilitar la identificación del mismo.

Una vez que el usuario confirma la creación del token se presenta el siguiente resultado.

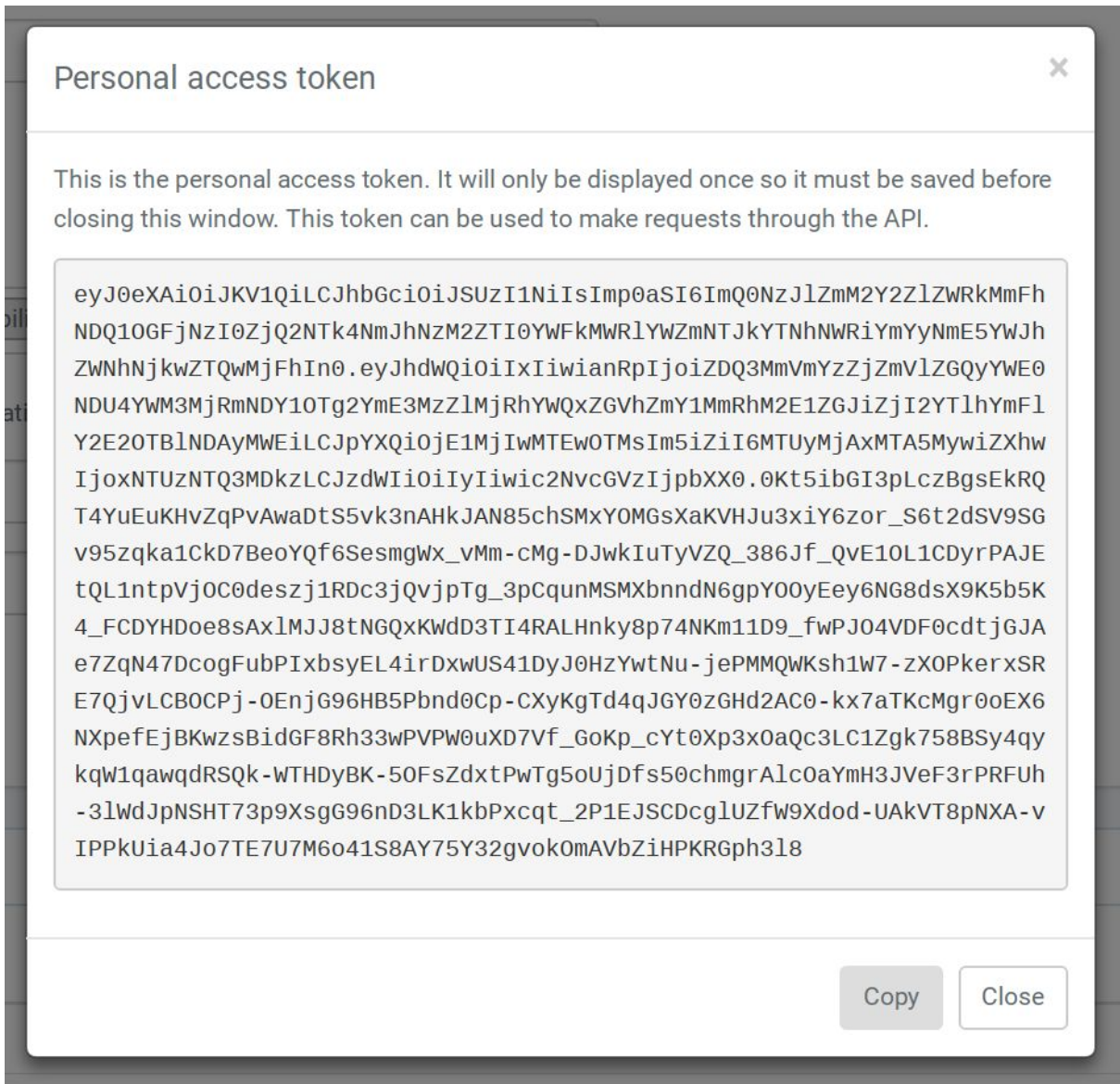


Figura 7.1.9.3: Visualización del token generado.

Esta ventana modal muestra el token generado y facilita su copia al portapapeles. Por motivos de seguridad este token no puede ser visualizado a posteriori. En caso de que el mismo sea comprometido, el usuario siempre puede eliminarlo del listado.

Notar que un mismo usuario puede tener varios tokens para flexibilizar el acceso a la API y organizarlo en base a sus necesidades.

7.2 Extensión de Google Chrome

La extensión de Google Chrome tiene un subconjunto de las operaciones que se pueden realizar en la aplicación web y, adicionalmente, permite la inyección de scripts de soluciones. En las siguientes secciones se explicarán las funcionalidades más relevantes.

7.2.1 Opciones de la extensión

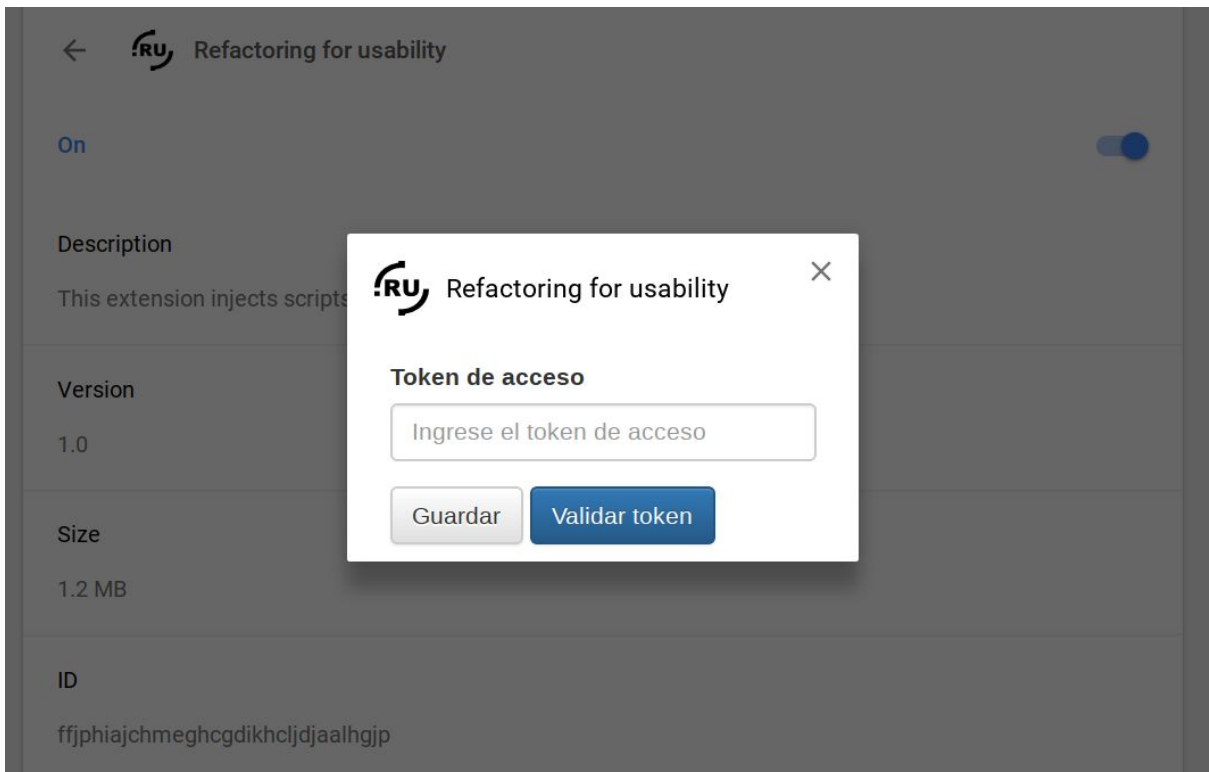


Figura 7.2.1.1: Formulario de la extensión de Chrome para el ingreso del token.

Esta sección presenta las configuraciones propias de la extensión de Chrome. Como puede verse, se apuntó a una interfaz mínima nuevamente. La misma permite el ingreso de un token, el cual se debería haber generado desde la página web. Lo único que necesita la extensión es el token por lo que, tras ingresar uno válido, no debería ser necesario realizar operaciones adicionales.

A fin de que el usuario pueda discernir entre un token válido y uno inválido, se incluyó un botón de validación de token. Al presionarlo, se obtiene una notificación que le informa al usuario si el token es válido o no. El caso de éxito se muestra a continuación:

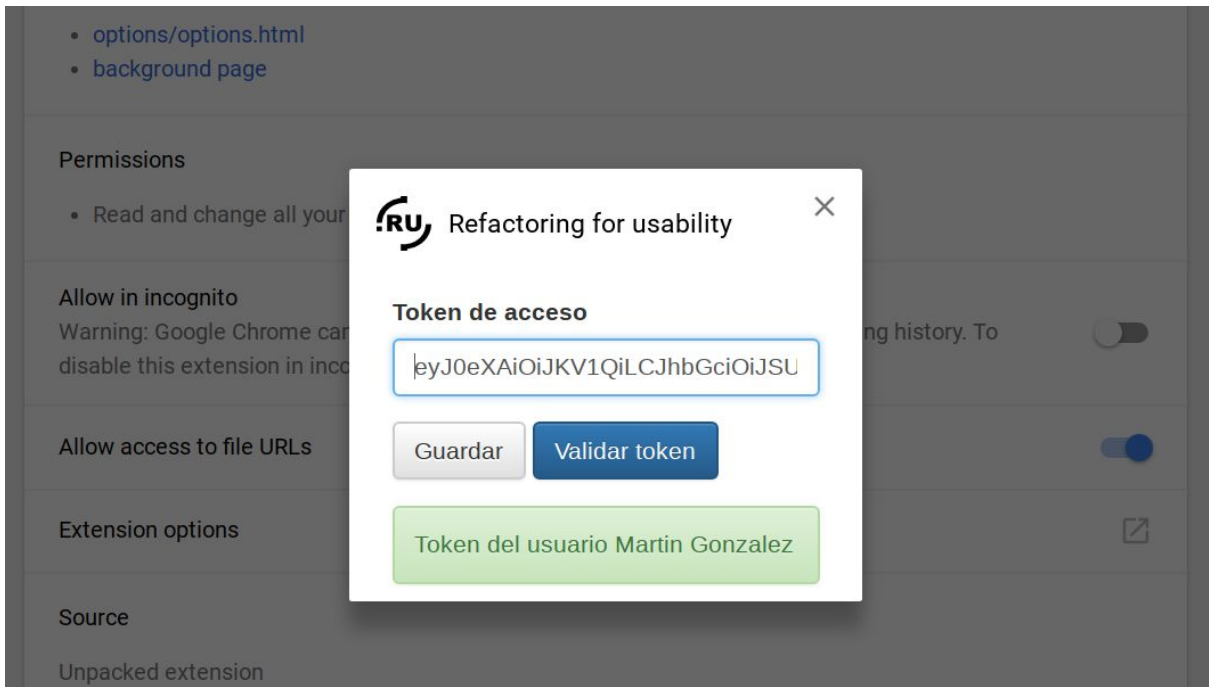


Figura 7.2.1.2: Formulario de la extensión de Chrome con un usuario válido.

7.2.2 Extensión en ejecución

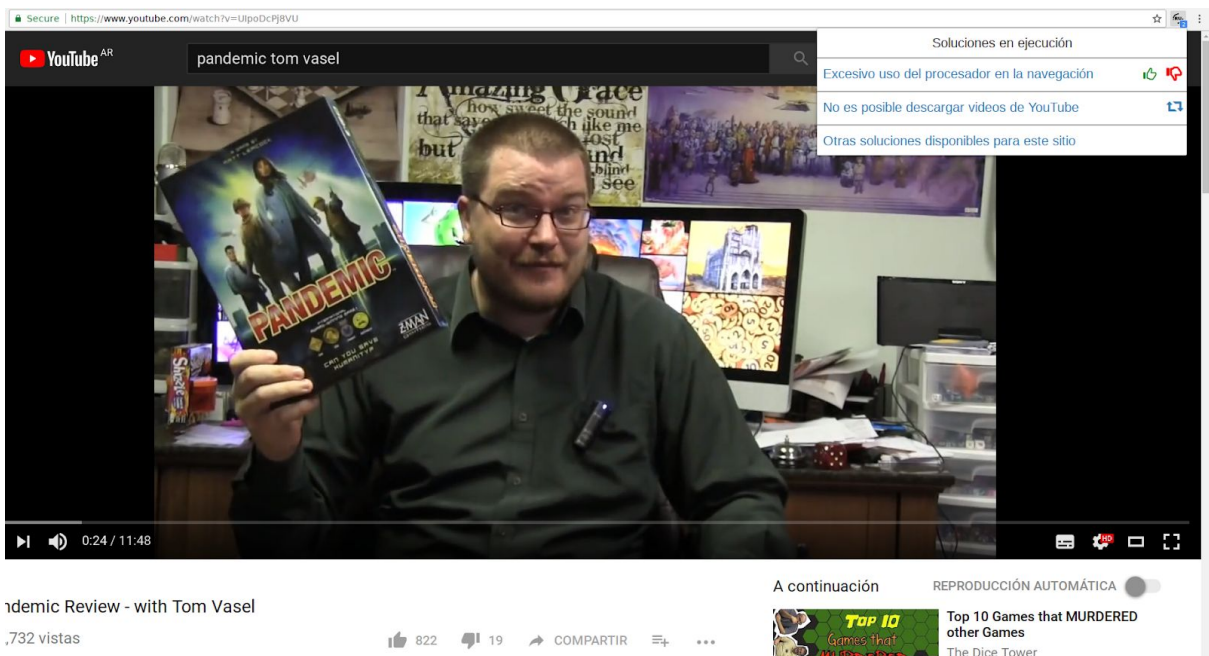


Figura 7.2.2.1: Ejemplo de ejecución de la extensión habiendo navegado el sitio YouTube.

En esta imagen se puede apreciar las funcionalidades que provee la extensión. La misma presenta un botón que puede verse en la parte superior derecha. Ese botón tiene un

número blanco en un recuadro azul. El mismo se usa para informar la cantidad de soluciones que se ejecutaron al navegar ese sitio.

Cuando se presiona el botón de la extensión, se abre un pop-up que presenta información un poco más detallada. La misma incluye un listado de las soluciones que se están ejecutando, las acciones que se pueden realizar sobre los problemas que tienen asociados y acceder a un listado de recomendaciones.

En este caso se puede ver que el usuario se asoció a dos problemas que tienen soluciones en ejecución. El primero de ellos, se puede inferir que se encuentra en el estado de prueba de soluciones. Esto se debe a que le permite al usuario votar la solución que está ejecutando. El segundo de ellos, se encuentra en estado finalizado. Esto se debe a que le permite al usuario votar para crear un nuevo proceso de evaluación.

La última opción en el pop-up no es un problema sino un hipervínculo al sitio web. El mismo le permite al usuario acceder al listado de problema sugeridos para el sitio al que navegó. A continuación se muestra un ejemplo de esta página:

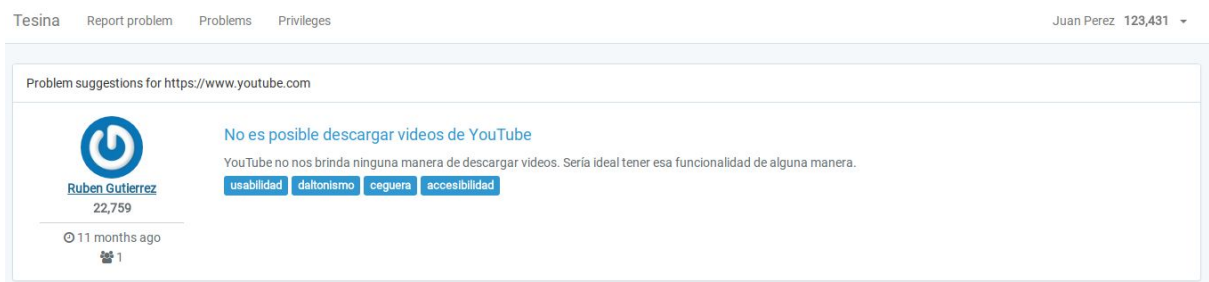


Figura 7.2.2.2: Sección del sitio web con el listado de problemas sugeridos.

8. Conclusiones y trabajo a futuro

8.1 Conclusiones

A medida que nos sumergimos más en la era digital generamos una mayor dependencia de la tecnología. Si comparamos nuestro día a día con el de 20 años atrás seguramente podremos apreciar una gran cantidad de diferencias producto de la evolución de las comunicaciones. Si bien, en líneas generales, estos cambios tienden a mejorar nuestra calidad de vida, también traen consigo nuevos desafíos y dificultades. Es justamente en estos últimos donde, como informáticos, debemos enfocar nuestros esfuerzos, a fin de disminuir la brecha entre la creciente complejidad del mundo digital y la sociedad.

Siguiendo este lineamiento, este trabajo se enfoca en facilitar la navegación web a través de la aplicación de mejoras de usabilidad y accesibilidad. La web tiene la complejidad de que los desarrolladores o propietarios de las páginas web son quienes tienen el control total sobre las mismas. Es decir, ellos controlan el contenido y el formato en el que se presenta la información a los usuarios. Sin embargo, hoy en día, esto no es algo bloqueante, ya que las aumentaciones web, a través de la ejecución de scripts en el contexto del navegador, nos permiten derribar esta barrera. Las mismas nos permiten, no sólo realizar cambios en cómo se presenta la información al usuario sino que también nos permiten enriquecerla.

Indiscutiblemente, las aumentaciones web cumplen un rol esencial en nuestro trabajo. Sin embargo, una gran parte de los objetivos del mismo se asocian a la construcción de una comunidad. Existen algunos sitios y foros que se enfocan en la publicación de scripts de usuario, como puede ser Greasemonkey, pero a diferencia de nuestra propuesta no tienen un foco en la comunidad en sí sino en los scripts de usuario. Es decir, en esas comunidades es posible para los usuarios de las mismas publicar sus scripts pero no es posible relacionarlos con problemas de navegación, determinar si una solución es mejor que otra o no y, posiblemente lo más importante, motivar a los usuarios a través del reconocimiento de sus logros.

Desde el inicio de este trabajo, nos enfocamos siempre en el usuario final y en facilitar su acceso a soluciones a sus problemas. Para lograr esto, priorizamos los scripts de usuario, la

comunidad y los procesos de prueba. La comunidad se autorregula, pública problemas, sugiere soluciones y las califica a través de los procesos de evaluación, brindando incentivos a los usuarios que colaboraron. Este flujo de trabajo es simple, fácil de comprender y sólo requiere de la registración en el sitio y de la instalación de la extensión de Chrome. Con este aporte, basado en las herramientas con las que contamos hoy en día, creemos que efectivamente estamos acercando un gran y variado abanico de soluciones a los usuarios a cambio de un esfuerzo muy bajo.

8.2 Contribuciones

8.2.1 Contribuciones relacionadas a la definición de procesos de trabajo

- Definición de un método para mejorar la experiencia de navegación web basado en la colaboración entre usuarios y un esquema de crowdsourcing. En el capítulo 3 realizamos una descripción de alto nivel del sistema diseñado, así como la descripción de las entidades involucradas y funcionalidades claves del mismo.
- Definición de un mecanismo de evaluación de contribuciones inspirado en procesos de A/B Testing, empleando el feedback de los usuarios como métrica de efectividad. En el capítulo 4 realizamos un resumen teórico del concepto de A/B Testing, y presentamos el mecanismo de evaluación de soluciones desarrollado como un proceso de A/B Testing.
- Definición de reglas e incentivos que regirían la comunidad de usuarios, brindándoles la capacidad de autorregularse mediante la asignación de puntajes y privilegios a sus integrantes basados en la calidad de sus contribuciones. En el capítulo 5 detallamos los mecanismos de regulación propuestos, así como los incentivos que tendrían los usuarios para participar.

8.2.2 Contribuciones relacionadas a las herramientas desarrolladas

- Desarrollo de una extensión del navegador Google Chrome que posibilita la inyección de scripts que implementan las soluciones a los problemas que los usuarios desean resolver. Más información en el capítulo 7.2.

- Desarrollo de un sitio web que implementa las funcionalidades básicas que le dan soporte a la comunidad, como la recepción de problemas, publicación y evaluación de soluciones y selección de una mejor solución aplicando A/B testing. Más información en el capítulo 7.1.
- Desarrollo de una API que posibilita la interacción con otros sistemas y favorece la extensibilidad del sitio web. Más información en el capítulo 6.

8.3 Limitaciones

El objetivo que nos propusimos para este trabajo fue armar un sistema que pueda solucionar problemas de navegación web. El mismo fue concretado e, incluso, nos vas a permitir solucionar problemas que hoy en día no sabemos que tenemos.

A pesar de esta flexibilidad en lo funcional, el mecanismo de inyección de soluciones tiene una limitación fuerte a tener en cuenta, relacionada a la seguridad.

A fin de que la extensión de Google Chrome pueda inyectar soluciones en cualquier página web, es necesario que la misma tenga permisos para leer y escribir la información (la página web en sí misma) que circula entre el navegador y el sitio web de cualquier página. Esto no implica más que aceptar la solicitud de otorgamiento de permisos al momento de instalar la extensión de Google Chrome. Sin embargo, las implicancias de esto no son despreciables.

Nuestra extensión tiene acceso de lectura y escritura sobre cualquier página lo cual implicaría que la misma podría, por ejemplo, monitorear el ingreso de un usuario y contraseña y enviar esos datos a alguna otra parte. De más está decir que la extensión en sí misma no hace eso. Sin embargo, hay que tener en cuenta que la misma ejecuta scripts creados por usuarios. Esto resulta en que usuarios malintencionados puedan crear soluciones que comprometen información sensible y privada de quienes hacen uso de su solución.

Este problema se mitiga a través de la intervención y autorregulación de la propia comunidad, en la cual usuarios con los privilegios necesarios pueden borrar una solución que consideren maliciosa. Sin embargo, no puede asegurarse la integridad de la información

completamente porque desde que se publica una solución hasta que la misma se borra puede llegar a pasar un cierto tiempo, el cual puede resultar en varias ejecuciones de esa solución.

En el contexto de este trabajo dejamos por fuera esta cuestión de seguridad y nos enfocamos en brindar las funcionalidades más relevantes. Hay que tener en cuenta que existen otras aproximaciones más seguras para esta cuestión de seguridad, como por ejemplo, forzosamente validar todas las contribuciones de todos los usuarios antes de que sean efectivamente publicadas. De esta manera, estaríamos seguros de que un usuario privilegiado efectivamente aprobó la solución. Nuevamente, esto no sería una solución absoluta ya que la gente puede cometer errores al evaluar, pero debería mitigar considerablemente el problema.

8.4 Trabajos futuros

Durante el desarrollo de esta tesina nos encontramos en muchos momentos con temas en los cuales nos hubiera interesado profundizar pero quedaron excluidos para poder concentrarnos en los objetivos principales planteados originalmente. Hemos anotado dichos temas para ser revisados en el futuro e incorporados al trabajo. Los presentaremos ahora como sugerencias de trabajo futuro. Algunas de estas ideas consideramos que serían de gran utilidad para los usuarios y contribuirían mucho a la solidez de la comunidad.

8.4.1 Herramientas de incentivo a la participación

Corresponden a funcionalidades que motivan a los usuarios a participar de la comunidad, haciendo una suerte de sana competencia (conocida como *ludificación*) entre los miembros de la misma, donde lo que se premia es la colaboración.

- **Medallas y distintivos:** son reconocimientos públicos que se asignan a los usuarios cuando cumplen alguna condición que suma valor a la comunidad. Por ejemplo, podría asignarse una medalla a un usuario que publica su primer solución o a un usuario que, por primera vez, se le asigne una solución como la mejor para un problema.

- **Recompensas:** se encuentran asociadas al intercambio de reputación en la comunidad. Por ejemplo, un usuario podría tener una gran necesidad de que un problema sea solucionado. Esta funcionalidad permitiría a ese usuario poder ceder parte de su reputación como premio adicional al usuario que tenga la mejor solución en el siguiente proceso de evaluación.

8.4.2 Herramientas de moderación

Involucran funcionalidades que facilitan, o hacen más colaborativa y distribuida, la administración del sitio.

- **Edición colaborativa de publicaciones:** permitiría que los textos de las publicaciones puedan ser editados por varios usuarios que tengan la reputación y privilegios correspondientes. De esta manera la calidad y claridad de las publicaciones mejoraría.
- **Reportar publicación:** representa marcar una publicación para que sea revisada por un administrador.
- **Herramientas de moderación:** son las funcionalidades que permitirían efectuar la moderación propiamente dicha. Es decir, le permitirían a los usuarios con el privilegio de administración ver listados de publicaciones reportadas, revisarlas y tomar las acciones correspondientes, por ejemplo, eliminarlas, en caso de que sean inapropiadas.
- **Cerrar problemas:** esta funcionalidad se asocia a problemas reportados que son inapropiados no por uso de lenguaje o claridad, sino por no estar dentro del contexto de problemas esperados o incluso simplemente por estar duplicados. Esta funcionalidad no permitiría a ningún usuario proponer soluciones para esos problemas.

8.4.3 Otras mejoras a la comunidad

Corresponden a mejoras genéricas, pero no por eso menos importantes, que se podrían implementar para mejorar usabilidad o darles más flexibilidad a los usuarios.

- **Votos en problemas:** permitirían ordenar los problemas en base a la importancia que los miembros de la comunidad le dan. De esa manera, los usuarios desarrolladores

de soluciones pueden ver qué problemas despiertan más interés para así enfocarse en los mismos.

- Búsqueda y ordenación de problemas: implicaría el desarrollo de búsquedas por distintos criterios, como por ejemplo, títulos, descripciones, etiquetas. La ordenación de problemas podría implicar distintas subsecciones con listados ordenados por criterios como fecha de publicación, problemas más votados, problemas que hayan recibido varias soluciones recientemente (problemas más activos), etc.
- Modificaciones a etiquetas de interés: se podría permitir a los usuarios definir dos tipos de etiquetas de interés: aquellas cuyos problemas se espera que se inyecten de manera automática, como se presentó en este trabajo, y etiquetas cuyos problemas se espera recibir como recomendación pero que no se inyecten de manera automática.
- Agregar comentarios a problemas y soluciones: éstos permitirían a los usuarios poder interactuar con los autores de los problemas y soluciones a fin de proponer recomendaciones de mejora, solicitar clarificaciones, etc.

8.4.4 Mejoras a la extensión del navegador

Estas funcionalidades son aquellas que deberían implementarse estrictamente en la extensión de Google Chrome, ya sea por limitaciones técnicas que requieren de esto o porque es más oportuno interactuar con el usuario en un determinado momento donde no necesariamente estaría navegando nuestro sitio web.

- Recordatorios de usuario: a fin de que el usuario no se olvide de emitir su voto a una solución de la que está haciendo uso, la extensión podría recordarle al usuario que aún tiene pendiente votar la misma.
- Inicio de sesión automático: esto implicaría que la extensión pueda identificar que el usuario haya iniciado sesión en el sitio web y autenticarse de manera automática, es decir, sin la necesidad de que el usuario copie y pegue el token de acceso. De manera similar, la extensión podría iniciar sesión de manera automática en el sitio web en caso de que el usuario haya finalizado su sesión.
- Habilitación y deshabilitación de soluciones a demanda: la extensión le permitiría al usuario deshabilitar una solución para que la misma no se aplique durante la

navegación. De esta manera, el usuario podría comparar la experiencia de navegación con la solución aplicada y sin la solución aplicada a fin de comprender y evaluar mejor el impacto de la misma.

- Implementación de caché: apunta a disminuir la carga del servidor web utilizando una caché en el navegador. La misma podría almacenar soluciones completas y aplicarlas sin la necesidad de sincronizarse en tiempo real con el servidor. Esta sincronización podría realizarse en intervalos frecuentes, así como cuando se detecte que el usuario esté navegando nuestro sitio web. De esta manera, al navegar a un sitio web, si la caché de soluciones aún es válida se tomarían desde la misma, evitando un requerimiento web a nuestro servidor.

8.4.5 Herramientas para el desarrollo de soluciones

Corresponden a funcionalidades que facilitarían el desarrollo de soluciones.

- Detección de invalidez de soluciones: esto se encuentra ligado a detectar cuando una solución ya deja de ser válida debido a cambios en el sitio web al que aplica la solución. Por ejemplo, si una solución depende de un elemento del DOM con un determinado identificador y luego ese elemento es removido o cambia su identificador, entonces la solución dejaría de ser válida. Esto podría detectarse fácilmente haciendo uso de JavaScript y se le podría dar una librería a los desarrolladores que incorpore funcionalidades de detección y de notificación de estas situaciones. Estas notificaciones podrían desencadenar una creación de un nuevo proceso de evaluación en el servidor sin la interacción de ningún usuario.
- Importación de scripts de soluciones: la idea de este punto es darle a los usuarios desarrolladores que estén cargando una solución, la opción de, en vez de cargar un script como texto, cargarlo como un enlace a un snapshot de un repositorio público, como [GitHub](#), [BitBucket](#), etc.

8.4.6 Propuestas de mejora de seguridad

Esta sección apunta a mitigar las limitaciones de seguridad explicadas en la sección 8.3 de este trabajo.

- Solicitar revisión de rutas antes de aplicar soluciones: la idea es permitirle al usuario poder identificar rutas (dominios o patrones de URLs) a las cuales apliquen soluciones que requieran confirmación del usuario para inyectar los scripts. Por ejemplo, esto implicaría definir que una ruta de soluciones requiere algún tipo de seguridad adicional, por lo que todas las soluciones que apunten a esa ruta no se aplicarían sin consentimiento expreso del usuario.
- Definir autores confiables: podría considerarse que los usuarios que superen un determinado nivel de reputación son *confiables*, en base a los aportes que realizaron a la comunidad. La propuesta en este caso sería que los usuarios tengan la opción de ejecutar automáticamente soluciones de usuarios confiables y que reciban una solicitud de confirmación antes de ejecutar soluciones de usuarios no confiables.
- Extender el proceso de evaluación: sería posible modificar el proceso de evaluación ligeramente para que el mismo sea un poco más seguro. Por ejemplo, entre el estado de aceptación de soluciones y el de evaluación de soluciones podría agregarse un paso de verificación de soluciones. Este permitiría a usuarios con determinado privilegio y reputación suficiente poder marcar qué soluciones continúan a la instancia de evaluación y qué soluciones no. La aceptación o rechazo de soluciones se concretaría con votos de varios usuarios.
- Eliminación de soluciones: este sería similar al anterior pero no necesariamente implica una modificación en el proceso de evaluación sino simplemente brindarle a los usuarios que cuenten con el privilegio correspondiente, la capacidad de eliminar soluciones específicas, independientemente de la instancia del proceso en la que se encuentren. La eliminación se concretaría con votos de varios usuarios.

Referencias bibliográficas

- [Asakawa 2008] Asakawa, C. et al. "Social Accessibility: Achieving Accessibility through Collaborative Metadata Authoring"
- [Crespo 2016] Crespo R. et al. Social4all: Definition of specific adaptations in Web applications to improve accessibility
- [Nielsen 2006] Nielsen, J., Loranger, H.: "Prioritizing Web Usability". Pearson Education (2006)
- [Nielsen 2012] Nielsen, J. 2012. Introduction to Usability. Enlace al artículo web: <https://www.nngroup.com/articles/usability-101-introduction-to-usability>
- [Fowler 1999] Fowler, M. 1999. Refactoring: Improving the Design of Existing Code, Addison Wesley.
- [Fowler 2006] Fowler, M. 2006. CodeSmell. Enlace al artículo web: <https://martinfowler.com/bliki/CodeSmell.html>
- [Garrido 2011] Garrido, A. et al. 2011. Refactoring for Usability in Web Applications
- [Garrido 2013] Garrido, A., Firmenich, S. Rossi, G., Grigera, J., Medina Medina, N., Harari, I. "Personalized Web Accessibility using ClientSide Refactoring". IEEE Internet Computing Vol. 17 No.4. 5866. IEEE Computer Soc. (2013)
- [Firmenich 2015] Firmenich, D., Firmenich, S., Rossi, G., Winckler, M., Distanto, D. "User Interface Adaptation Using Web Augmentation Techniques: Towards a Negotiated Approach". In proceedings of the International Conference on Web Engineering, ICWE2015 (Rotterdam, Países Bajos): 147-164 (2015)
- [Howe2006a] <https://www.wired.com/2006/06/crowds>
- [Howe2006b] http://crowdsourcing.typepad.com/cs/2006/06/crowdsourcing_a.html
- [Mao 2015] Ke Mao, Licia Capra, Mark Harman and Yue Jia. A Survey of the Use of Crowdsourcing in Software Engineering. Technical Report RN/15/01, Department of Computer Science, University College London, 2015
- [Kim 2000] Amy Jo Kim, 2000, "Community Building on the Web: Secret Strategies for Successful Online Communities"
- [Doan 2011] Doan, A., Ramakrishnan, R., and Halevy, A. "Crowdsourcing systems on the World-Wide Web". Communications ACM 54, 4 (April 2011), 86-96.

- [Kohavi 2009] Kohavi, R., Longbotham, R., Sommerfield, D. et al. Data Min Knowl Disc (2009) 18: 140. <https://doi.org/10.1007/s10618-008-0114-1>
- [Kohavi 2017] Kohavi R., Longbotham R. (2017) Online Controlled Experiments and A/B Testing. In: Sammut C., Webb G.I. (eds) Encyclopedia of Machine Learning and Data Mining. Springer, Boston, MA
- [Linden 2006] Linden G (2006a) Early Amazon: shopping cart recommendations. Geeking with Greg. [Online] April 25, 2006. <http://glinden.blogspot.com/2006/04/early-amazon-shopping-cart.html>
- [Siroker 2013] Siroker, D., Koomen, P. "A/B Testing: The Most Powerful Way to Turn Clicks Into Customers". Wiley, (2013)
- [Pilgrim 2005] Pilgrim, Mark. "Greasemonkey Hacks: Tips & Tools for Remixing the Web with Firefox". O'Reilly (2005)
- [<https://www.w3.org/standards/webdesign/accessibility>]
- [<https://www.w3.org/TR/WCAG20/>]
- [<https://webaim.org/intro/>]
- [<https://developers.google.com/web/fundamentals/accessibility/>]

Anexos

1. Extensiones de Google Chrome

1.1 Conceptos generales

Las extensiones de Google Chrome no son más que aplicaciones que se ejecutan dentro de este navegador extendiendo su funcionalidad básica. Estas funcionalidades adicionales pueden incluir integración con sitios o servicios de terceros o proveer una experiencia de navegación personalizada.

Cabe destacar que las extensiones no se ejecutan como componentes independientes del navegador sino que se ejecutan en el contexto del mismo y, por consiguiente, se encuentran regidas por un conjunto de limitaciones propias de Chrome. Algunos ejemplos de estas limitaciones son la cantidad de datos que pueden almacenar localmente (en el cliente), las páginas (o dominios, más específicamente) con los que pueden interactuar o la información de usuario a la que se tiene acceso (historial de navegación, marcadores, portapapeles, entre otros).

A diferencia de la mayoría de las aplicaciones web, las extensiones de Chrome no dependen de contenido web para su ejecución. Las mismas son tan sólo un archivo que una vez descargado se ejecuta localmente en el navegador. Sin embargo, se debe considerar que es posible que una determinada extensión, por su naturaleza, realice solicitudes web de contenido remoto a fin de interactuar con algún servicio o página web.

Desde un enfoque orientado al desarrollo, las extensiones se pueden crear usando las clásicas tecnologías que se usan para el desarrollo de sitios web. Es decir, es posible usar HTML como un lenguaje de maquetado, CSS para dar estilo y JavaScript para agregar la lógica necesaria. De hecho, dado que Chrome soporta HTML5 y CSS3, es posible utilizar tecnologías de última generación para el desarrollo de extensiones, como pueden ser canvas y animaciones CSS. Adicionalmente, las extensiones tienen acceso a una gran

variedad de APIs de JavaScript que provee Chrome mismo que brindan una considerable cantidad de herramientas para el desarrollo y la interacción con el navegador.

1.2 Interacción con el usuario

A fin de entender mejor las características y limitaciones que tienen las extensiones es esencial presentar las distintas formas de interacción que pueden tener las extensiones.

Anteriormente se mencionó que las extensiones no son más que un archivo que contiene archivos HTML, CSS, JavaScript, imágenes, etc. De esto se puede deducir que una extensión de Chrome es, en esencia, algo similar a una página web, que tiene su lógica de negocio, presentación y maquetado. Sin embargo, también se deben adaptar y hacer uso de la API que les provee el navegador.

La API de Chrome permite a las extensiones presentar botones a la derecha de la barra de navegación. Estos botones disparan eventos propios de la extensión a la que pertenecen. A continuación, se puede ver un ejemplo de una extensión de GMail que agrega un botón al navegador que muestra la cantidad de correos no leídos y que, al hacer click sobre el mismo, indica a Chrome que navegue al sitio web de GMail.



Figura Anexo 1.2.1: Icono de la extensión en la barra de navegación.

Estos botones se denominan browser actions. Se caracterizan por mantener su funcionalidad a lo largo del tiempo e independientemente de la página o dominio que el usuario esté visitando en un momento dado.

Sin embargo, existe otro tipo de interacción similar que permite agregar un botón el cual sólo es útil en determinadas circunstancias. Por ejemplo, es posible que al navegar una

página web, la misma provea metadatos que indiquen al navegador la existencia de un RSS y que una extensión brinde la capacidad de suscribirse al mismo tan sólo con un click. Este es un claro ejemplo de cómo la interacción con las extensiones no siempre es necesaria en todo momento sino que puede depender del contexto.

A continuación, se muestra cómo se vería esta extensión de ejemplo en caso de que la página actual tenga metadatos de RSS:



Figura Anexo 1.2.2: Icono de la extensión activado.

En la siguiente imagen se muestra el caso opuesto:



Figura Anexo 1.2.3: Icono de la extensión desactivado.

Este tipo de interacciones que proveen las extensiones, dependientes de la relevancia que tienen en la página actual, se logran a través de las denominadas page actions y son, en sí mismas, muy similares a las browser actions mencionadas anteriormente.

Las extensiones también permiten integrarse en los menús contextuales. Estos menús tienen acciones que permiten a la extensión interactuar con la página actual. A continuación, se muestra un ejemplo de un menú contextual al cual se le adicionaron 4 menús de distintas extensiones:

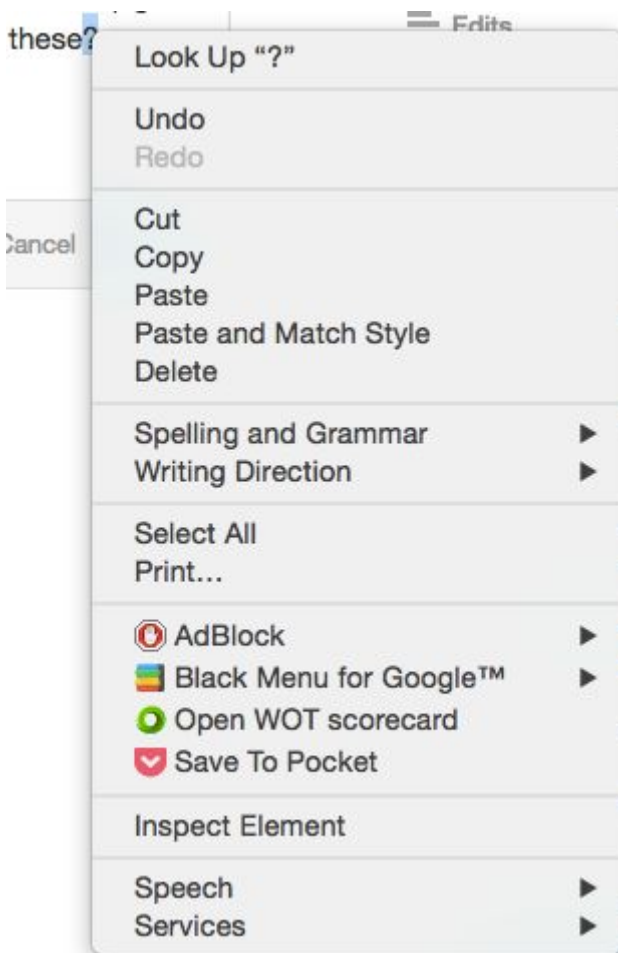


Figura Anexo 1.2.4: Barra de menú contextual.

Otro mecanismo de interacción con el usuario que provee la API de Chrome son las options pages. Éstas no son más que páginas web que permiten la configuración de la extensión y se puede acceder a la misma desde la administración de complementos de Chrome, haciendo click en el botón de opciones de la extensión en cuestión.

Las páginas de opciones se presentan en un popup modal, como se muestra a continuación:

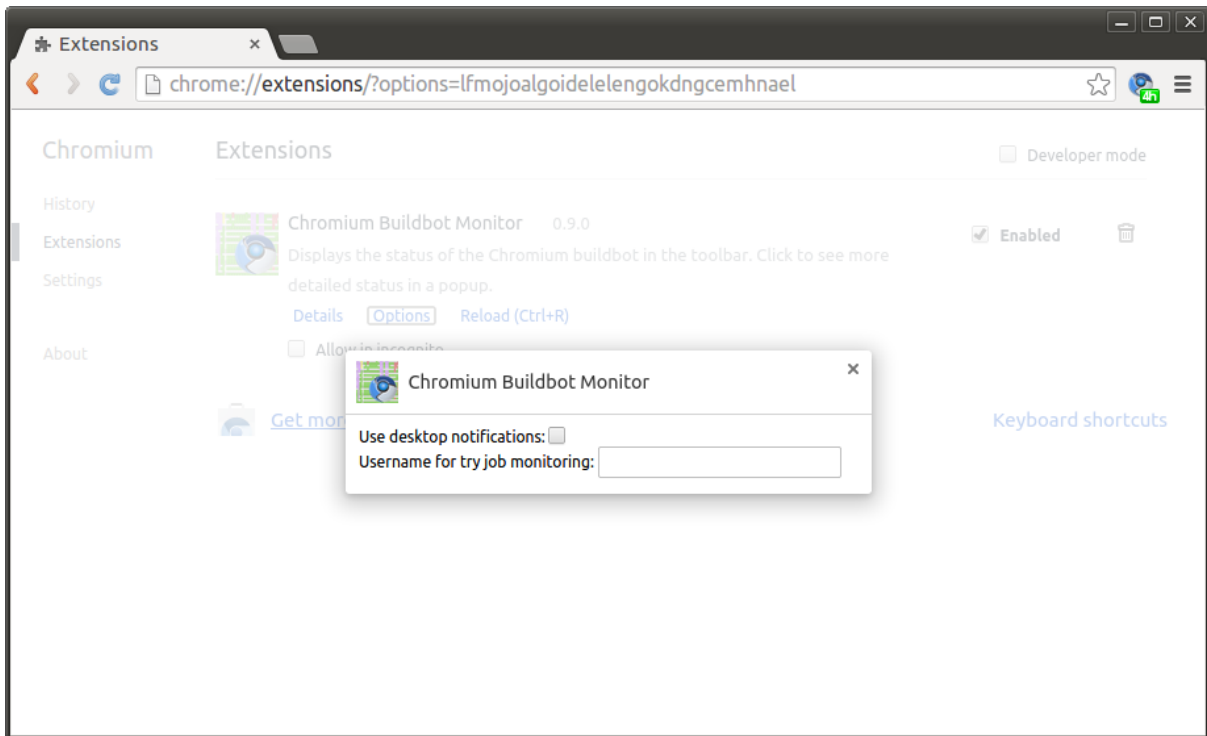


Figura Anexo 1.2.5: Popup de configuración.

Finalmente, Google Chrome provee una forma adicional de interactuar con el usuario. Ésta es a través de los denominados content scripts. Los mismos se describirán más adelante, pero en esta instancia sólo es relevante mencionar que los mismos permiten modificar el contenido de la página sobre la que estén actuando. Estas modificaciones pueden ser realmente cualquier tipo de modificación, como traducirlas, cambiar colores o formatos, agregar o remover botones, etc.

1.3 Arquitectura de las extensiones

A alto nivel, Chrome hace una distinción significativa entre los componentes que operan sobre páginas web (por ejemplo, modificando su contenido) y aquellos componentes que se encargan de la lógica de la extensión (por ejemplo, que consumen un web service y realizan alguna operación con su resultado).

Background pages

Las denominadas background pages cumplen la función de incluir código JavaScript que controla el comportamiento de la extensión, realizando así tareas y modificando estado. Las background pages son páginas HTML que se ejecutan en el contexto de la extensión. Por

ende, las mismas existen mientras la extensión esté en ejecución y sólo puede haber una instancia de ellas activa en un momento dado.

Las extensiones que operan con background pages suelen implementar los componentes de interacción con el usuario a través de interfaces livianas, es decir, que no son esos componentes los que realizan la lógica de la aplicación sino que cuando los mismos requieren obtener algún estado interno, se lo solicitan a la background page. Adicionalmente, cuando la background page detecta cambios de estado, ésta notifica a las vistas que corresponda de las actualizaciones.

Content scripts

Los content scripts, como se mencionó anteriormente, permiten a las extensiones interactuar con páginas web. Más específicamente son componentes que permiten la ejecución de JavaScript en el contexto de la página que se haya cargado en Chrome. De hecho, son la forma de interacción con páginas web navegadas por Chrome que más se acerca a las mismas.

La característica más significativa de los content scripts, en el contexto de este trabajo, es que los mismos pueden acceder a la información de las páginas web que Chrome visita y también pueden hacer cambios a las mismas. Por ejemplo, pueden acceder y manipular el DOM de estas páginas. Sin embargo, no tienen acceso a las variables o funciones de JavaScript definidas por la página web. Del mismo modo, las páginas web no pueden acceder a variables o funciones de JavaScript definidas por el content script.

Esta división entre el contexto de la página web y el del content script evita conflictos de colisiones de nombres de identificadores o conflictos de versionado de librerías, entre otros. Por ejemplo, una página web podría incluir jQuery v1 y el content script podría hacer uso de jQuery v2 y ninguno entraría en conflicto con el otro.

Una limitación importante de los content scripts es que no tienen acceso directo al DOM de las páginas propias de la extensión, por ejemplo, la background page o la página de opciones. Sin embargo, esto no significa que los mismos estén aislados de la extensión, ya que un content script puede intercambiar mensajes con la extensión de la que forma parte. Continuando con el ejemplo de RSS mencionado anteriormente, un content script podría

monitorear el DOM de las páginas que el usuario visita y, en caso de que el mismo detecte una fuente de contenido RSS, enviaría un mensaje a una background page para su procesamiento. Inversamente, una background page puede también comunicarse con un content script enviándole mensajes para que el mismo actualice el contenido de la página web que se esté visitando.