Neiva, 10 de marzo del 2022

Señores

CENTRO DE INFORMACIÓN Y DOCUMENTACIÓN

UNIVERSIDAD SURCOLOMBIANA

NEIVA - HUILA

El (Los) suscrito(s):

William Andrés Otálora Torres, con C.C. No. 1.075.316.785 de Neiva (Huila),

Cristian Camilo Bravo Vargas, con C.C. No. 1.019.138.514 de Bogotá D.C (Cundinamarca),

Autor(es) de la tesis y/o trabajo de grado o proyecto de grado

titulado DESPLIEGUE DEL SOFTWARE GTAXI MEDIANTE LA METODOLOGÍA DE SERVICIOS MÚLTIPLES PARA LA SOLICITUD DEL SERVICIO DE TAXI EN EL MUNICIPIO DE GARZÓN HUILA presentado y aprobado en el año 2022 como requisito para optar al título de Ingeniero de Software;

Autorizo (amos) al CENTRO DE INFORMACIÓN Y DOCUMENTACIÓN de la Universidad Surcolombiana para que, con fines académicos, muestre al país y el exterior la producción intelectual de la Universidad Surcolombiana, a través de la visibilidad de su contenido de la siguiente manera:

- Los usuarios puedan consultar el contenido de este trabajo de grado en los sitios web que administra la Universidad, en bases de datos, repositorio digital, catálogos y en otros sitios web, redes y sistemas de información nacionales e internacionales “open access” y en las redes de información con las cuales tenga convenio la Institución.

- Permita la consulta, la reproducción y préstamo a los usuarios interesados en el contenido de este trabajo, para todos los usos que tengan finalidad académica, ya sea en formato Cd-Rom o digital desde internet, intranet, etc., y en general para cualquier formato conocido o por conocer, dentro de los términos establecidos en la Ley 23 de 1982, Ley 44 de 1993, Decisión Andina 351 de 1993, Decreto 460 de 1995 y demás normas generales sobre la materia.

- Continúo conservando los correspondientes derechos sin modificación o restricción alguna; puesto que, de acuerdo con la legislación colombiana aplicable, el presente es un acuerdo jurídico que en ningún caso conlleva la enajenación del derecho de autor y sus conexos.

De conformidad con lo establecido en el artículo 30 de la Ley 23 de 1982 y el artículo 11 de la Decisión Andina 351 de 1993, “Los derechos morales sobre el trabajo son propiedad de los autores” , los cuales son irrenunciables, imprescriptibles, inembargables e inalienables.

William Andrés Otálora Torres

Firma: _____

Cristian Camilo Bravo Vargas:

Firma: _____

TÍTULO COMPLETO DEL TRABAJO: DESPLIEGUE DEL SOFTWARE GTAXI MEDIANTE LA METODOLOGÍA DE SERVICIOS MÚLTIPLES PARA LA SOLICITUD DEL SERVICIO DE TAXI EN EL MUNICIPIO DE GARZÓN HUILA.

AUTOR O AUTORES:

Primero y Segundo Apellido	Primero y Segundo Nombre
Otálora Torres	William Andrés
Bravo Vargas	Cristian Camilo

DIRECTOR Y CODIRECTOR TESIS:

Primero y Segundo Apellido	Primero y Segundo Nombre
Medina Rojas	Ferley

ASESOR (ES):

Primero y Segundo Apellido	Primero y Segundo Nombre
Rodríguez Rojas	Fabian Ernesto

PARA OPTAR AL TÍTULO DE: Ingeniero de Software

FACULTAD: Ingeniería

PROGRAMA O POSGRADO: Ingeniería de Software

CIUDAD: Neiva **AÑO DE PRESENTACIÓN:** 2022 **NÚMERO DE PÁGINAS:** 3

TIPO DE ILUSTRACIONES (Marcar con una **X**):

Diagramas___ Fotografías___ Grabaciones en discos___ Ilustraciones en general _X_ Grabados___ Láminas___ Litografías___ Mapas___ Música impresa___ Planos___ Retratos___ Sin ilustraciones___ Tablas o Cuadros__

SOFTWARE requerido y/o especializado para la lectura del documento:

MATERIAL ANEXO:

- Carta de aprobación por parte del gerente en COOTRANSGAR LTDA.
- Constancia informe de proyecto del gerente de COOTRANSGAR LTDA.
- Constancia por parte de la empresa COOTRANSGAR LTDA.

PREMIO O DISTINCIÓN *(En caso de ser LAUREADAS o Meritoria)*: No aplica laureado ni mentoría

PALABRAS CLAVES EN ESPAÑOL E INGLÉS:

	Español	Inglés		Español	Inglés
1.	Taxistas	Taxi drivers	6.	Implementación	Implementation
2.	Usuarios	Users			
3.	Aplicación móvil	Mobile application			
4.	Despliegue	Deployment			
5.	Servicio	Service			

RESUMEN DEL CONTENIDO: (Máximo 250 palabras)

Los inconvenientes de movilidad interna vehicular, inseguridad, demoras e inconformidad, son problemáticas que han venido levantando preocupación entre los ciudadanos de Garzón. Temas como la informalidad han surgido en respuesta a esa intranquilidad, sin embargo, el uso de estos transportes volubles no cuentan con la autorización de los entes regulatorios y ponen en una situación de riesgo tanto al usuario como al prestador del servicio.

De esta manera, la cooperativa COOTRANSGAR LTDA, ha venido desarrollando un software enfocado a la solicitud del servicio de taxi, para que los usuarios que cuentan con un dispositivo móvil Android puedan acceder y pedir un taxi de una manera segura y acortando los tiempos de espera. Es por ello, que el siguiente paso que quiere dar la cooperativa y en el que se basa el proyecto es testear, actualizar, implementar y desplegar el desarrollo de dicha herramienta, que esté disponible en los medios digitales para así llegar en primer lugar, a los ciudadanos del municipio y que estos comiencen a utilizarla de manera pública, en segundo lugar, a sus taxistas vinculados, ofreciéndoles una alternativa fácil de usar y finalmente a la cooperativa quien está a la vanguardia de las tecnologías en pro de sus asociados.

Con el desarrollo de este proyecto se busca desplegar GTAXI, garantizando la calidad en la herramienta, determinar sus versiones, ajustes pertinentes y actualizaciones satisfactorias en el uso, beneficiando tanto a ciudadanos de Garzón como al gremio de taxistas asociado a COOTRANSGAR LTDA.

ABSTRACT: (Máximo 250 palabras)

The inconveniences of internal vehicular mobility, insecurity, delays, and nonconformity are problems that have been raising concern among the citizens of Garzón. Issues such as informality have arisen in response to this concern, however, the use of these fickle transports does not have the authorization of regulatory entities and puts both the user and the service provider at risk.

In this way, the COOTRANSGAR LTDA cooperative has been developing software focused on requesting taxi service, so that users who have an Android mobile device can access and order a taxi safely and shorten waiting times. That is why the next step that the cooperative wants to take and on which the project is based is to test, update, implement and deploy the development of said tool, which is available in digital media in order to reach, in the first place, the citizens of the municipality and that they begin to use it publicly, secondly, to their associated taxi drivers, offering them an easy-to-use alternative and finally to the cooperative, which is at the forefront of technologies for its associates.

With the development of this project, it is sought to deploy GTAXI, guaranteeing the quality of the tool, determining its versions, pertinent adjustments, and satisfactory updates in use, benefiting both the citizens of Garzón and the taxi drivers' union associated with COOTRANSGAR LTDA.

APROBACION DE LA TESIS

Nombre Presidente Jurado: Fernando Rojas.

Firma: Frojas.

Nombre Jurado: Luis Gregorio Ramón Carvajal

Firma: Luis Gregorio Ramón C.

Nombre Jurado: JUAN A. CASTRO S.

Firma: JUAN A. CASTRO S.

DESPLIEGUE DEL SOFTWARE GTAXI MEDIANTE LA METODOLOGÍA DE SERVICIOS MÚLTIPLES PARA LA SOLICITUD DEL SERVICIO DE TAXI EN EL MUNICIPIO DE GARZÓN HUILA.

TRABAJO DE TITULACIÓN

PREVIO A LA OBTENCIÓN DEL TÍTULO DE

INGENIERO DE SOFTWARE

AUTORES

WILLIAM ANDRÉS OTÁLORA TORRES

CRISTIAN CAMILO BRAVO

UNIVERSIDAD SURCOLOMBIANA

FACULTAD DE INGENIERÍA

PROGRAMA DE INGENIERÍA DE SOFTWARE

DIRECTOR DEL TRABAJO

ING. DOC. FERLEY MEDINA

2022

NEIVA – HUILA

Tabla de contenido

Lista de tablas

Lista de figuras

Resumen

Los inconvenientes de movilidad interna vehicular, inseguridad, demoras e inconformidad, son problemáticas que han venido levantando preocupación entre los ciudadanos de Garzón - Huila. Temas como la informalidad han surgido como respuesta inmediata a esa intranquilidad, sin embargo, el uso de estos transportes volubles no cuenta con la autorización de los entes regulatorios designados por el gobierno y que, sin duda, ponen en una situación de riesgo tanto al usuario como al prestador del servicio.

De esta manera, la cooperativa de transportadores COOTRANSGAR LTDA, ha venido desarrollando un software enfocado a la solicitud del servicio de taxi, para que los usuarios que cuentan con un dispositivo móvil Android puedan acceder a este servicio de pedir taxi de una manera segura y acortando los tiempos donde los clientes tratan de encontrar un taxi que los traslade a su destino. Es por ello, que el siguiente paso que quiere dar la cooperativa y en el que se basa el proyecto, es testear, actualizar, implementar y desplegar el desarrollo de dicha herramienta para así llegar en primer lugar, a los ciudadanos del municipio y que estos comiencen a utilizarla de manera pública, en segundo lugar a sus taxistas vinculados, ofreciéndoles una alternativa fácil de usar y que les genere un trabajo más seguro y finalmente a la cooperativa quien está a la vanguardia de las tecnologías en pro de sus asociados.

Con el desarrollo de este proyecto se busca desplegar el software GTAXI y que esté disponible en los medios digitales. Garantizando la calidad en la herramienta, determinar sus versiones, ajustes pertinentes, actualizaciones y realizar un estudio de satisfacción en los taxistas vinculados a la cooperativa. De esta manera se puede considerar si el software cumple con los requerimientos que necesita la cooperativa beneficiando así tanto a ciudadanos de Garzón como al gremio de taxistas de COOTRANSGAR LTDA.

Palabras clave: Taxistas, usuarios, aplicación móvil, despliegue, servicio, implementación.

Abstract

The inconveniences of internal vehicular mobility, insecurity, delays, and non-conformity are problems that have been raising concern among the citizens of Garzón - Huila. Issues such as informality have emerged as an immediate response to this concern, however, the use of these informal transports doesn't have the authorization of the regulatory entities designated by the government and that, without a doubt, put both the user at risk as well as the service provider.

In this way, the cooperative of transporters COOTRANSGAR LTDA, has been developing a software focused on the request of the taxi service, so that users who have an Android mobile device can access this service of requesting a taxi in a safe way and shortening the costs times where customers try to find a taxi to take them to their destination. That is why the next step that the cooperative wants to take and on which the project is based, is to test, update, implement and deploy the development of said tool in order to reach the citizens of the municipality in the first place and that they begin to use it publicly, secondly to their associated taxi drivers, offering them an easy-to-use alternative that generates a safer job for them and finally to the cooperative, which is at the forefront of technologies of its associates.

With the development of this project, it's sought to deploy the GTAXI software and that it is available in digital media. Ensuring the quality of the tool, determining its versions, appropriate adjustments, updates and carrying out a satisfaction study of taxi drivers linked to the cooperative. In this way, it can be considered if the software meets the requirements needed by the cooperative, thus benefiting both the citizens of Garzon and taxi drivers' union of COOTRANSGAR LTDA.

Keywords: Taxi drivers, users, mobile application, deployment, service, implementation.

Introducción

La Cooperativa de Transportadores de Garzón COOTRANSGAR LTDA, con sede en el municipio de Garzón-Huila, presta los servicios de transportista a nivel municipal y departamental, siendo una de las cooperativas más reconocidas en la región de Garzón por sus años de trayecto, conocimiento de los servicios con calidad que garantiza seguridad y eficiencia a sus asociados. Hoy por hoy, la cooperativa se encuentra en la búsqueda de la implementación de herramientas para mejorar la movilidad interna de la ciudad, específicamente en el servicio de taxi y posicionar a la Cooperativa como una empresa líder en el ramo del centro y sur colombiano.

Actualmente, en la ciudad conocida como la Capital Diocesana del Huila, se han venido presentando inconvenientes en los traslados y servicio de taxi debido a la inseguridad en el transporte y la congestión vehicular. Dichos aspectos han generado inconformidad en los usuarios, poca fiabilidad y vulnerabilidad a la espera de un taxi en las avenidas, lo que ha conllevado al poco uso del servicio y/o que se opte por servicios ilegales como el mototaxismo.

Es preciso a esta serie de problemáticas existentes en nuestra realidad que surge GTAXI, un software de solicitud de asistencia de taxi mediante la tecnología de GPS y enfocado en los dispositivos móviles, para el servicio de movilidad en el municipio de Garzón – Huila, desarrollado por la cooperativa con el objeto de dar solución a las cuestiones y dificultades vehiculares, que, aunque son mayores, buscan dar el primer paso y mitigar, poco a poco, lo que tanto se cuestiona en la región.

Hoy en día, dicho software no se encuentra a la mano, ni de los taxistas ni de los usuarios, debido a que no se ha realizado un proceso de implementación que tanto se requiere y es por ello, que el presente trabajo pretende desarrollar el despliegue para que esta herramienta sea una

realidad, pasando por un proceso de requerimientos, pruebas donde se evalúa la calidad, modificación y/o mejora de la aplicación, corrección de errores o bugs, re-pruebas y publicación en tienda digital PlayStore.

Problema

En la actualidad, en el municipio de Garzón Huila, la cooperativa de transportadores COOTRANSGAR LTDA en su servicio de taxistas, cuenta con un software desarrollado de nombre GTAXI, que permite a los usuarios generar peticiones para la asistencia de taxi desde su localización hasta su destino, ubicación del vehículo más cercano, conocimiento de quién lo traslada y calificación del beneficio prestado.

Sin embargo, el software para facilitar el desplazamiento de las personas aún no se ha desplegado o puesto en marcha, ni tampoco detalla pruebas que generen la satisfacción deseada por los usuarios finales y los taxistas, ni se ha dado a conocer al público general, como también, hay desconocimiento del aplicativo por parte del gremio de conductores.

El no despliegue de esta herramienta sigue contribuyendo a la inseguridad en el transporte público a nivel de los taxistas y usuarios. De la misma forma, la búsqueda de un servicio, por parte del taxista, ocasiona un aumento de la circulación vehicular que congestiona el tráfico.

Es por ello, que la cooperativa requiere y necesita de un proceso donde se evalúe y mejore su software para llevarlo a disposición del público general.

Preguntas

- ¿Cómo desplegar la herramienta de solicitud de asistencia de taxi GTAXI de la cooperativa Cootransgar mediante la metodología de despliegue de servicios múltiples para el municipio de Garzón?
- ¿Cómo testear el software de servicio de taxi GTAXI y su plataforma administrativa frente a procesos sincrónicos de varios usuarios a través pruebas funcionales y no funcionales en el municipio de Garzón?
- ¿Cómo evaluar la calidad de la herramienta de servicio de taxi GTAXI del municipio de Garzón mediante el modelo ISO 25000?
- ¿Cómo implementar el software de la cooperativa Cootransgar para la mejora del servicio público de los usuarios de taxis mediante actividades de lanzamiento, actualización y adaptación en el municipio de Garzón?

Objetivos

Objetivo general

Desplegar el software GTAXI mediante la metodología de servicios múltiples para la solicitud de servicio de taxi en el municipio de Garzón - Huila.

Objetivos específicos

- Testear el software de servicio de taxi GTAXI y su plataforma administrativa a través de pruebas funcionales y no funcionales frente a procesos sincrónicos de varios usuarios en el aplicativo.
- Evaluar la calidad de la herramienta de servicio de taxi GTAXI de COOTRANSGAR LTDA mediante el modelo ISO/IEC 25000.
- Implementar el software de la cooperativa para la mejora del servicio público de los usuarios de taxis mediante actividades de lanzamiento, actualización y adaptación en el municipio de Garzón.

Alcance y limitaciones

Durante el desarrollo del proyecto se busca cumplir con los requerimientos de implementación y despliegue que necesita la aplicación existente de la cooperativa para que pueda estar a disposición del público. El software pensado en mejorar la movilidad, los procesos de carreras vehiculares y la seguridad tanto para taxistas como para usuarios que solicitan el servicio, será testeado y desplegado de acuerdo con el cumplimiento de parámetros de calidad de software.

En el proyecto enfocado hacia el sistema de GTAXI se trabajó, en primer lugar, con la herramienta ya existente. Posteriormente se le realizaron actualizaciones y segmentaciones del versionamiento para el cumplimiento y garantías que requiere la cooperativa apuntando hacia un software óptimo.

Con las pruebas y el despliegue de la aplicación se espera llegar a la mayoría de los residentes y visitantes de Garzón que tengan un dispositivo móvil limitado con sistema operativo Android con versión igual o superior a la 6.0. El acceso a esta aplicación de solicitud de taxi por el usuario podrá ser desde descarga vía PlayStore, mientras que la aplicación para taxista será distribuida por la cooperativa para única y exclusivamente sus conductores vinculados y estará disponible en la página web administrativa.

Alcance de necesidades

Para el proceso de solicitud de taxis se realizan algunos procesos de peticiones que necesitan ser testeados para el funcionamiento correcto. Aquí un breve detalle tabulado donde se muestra el código de la necesidad y la prueba de dicha.

Tabla 1
Alcance de las necesidades

Código	Prueba necesaria
A-01	Pruebas en registro y sesión del usuario Se debe inscribir satisfactoriamente la información del usuario y su solicitud de acceso, conservando los datos de manera segura. Para la confidencialidad, se realiza el motor de cifrado de Firebase Authentication con medidas de seguridad exhaustiva y transito HTTPS. Para inicio de administrador se debe validar su inicio en Firebase y validar mientras se mantenga en toda la plataforma hasta finalizar su sesión.
A-02	Pruebas de la solicitud de taxi Al realizar la solicitud del servicio de taxi, esta puede ser cancelada o iniciada por usuario o taxista, cumpliendo las limitaciones para solo conductores vinculados y en el municipio de Garzón. Las pruebas deben cumplir con la cancelación exitosa del trayecto cuando se lo requiere y que estas puedan volver a ser pedidas.
A-02	Pruebas en el CRUD administrativo El software del administrador necesita testear la creación, lectura, actualización y eliminación tanto de taxis como de taxistas para llevar un control de estos, en caso de eliminación se cambiará su estado para llevar un registro pasado o historial. La información debe ser manejada de forma precisa para la operación que se requiera y así también garantizar agilidad en el proceso.
A-03	Pruebas en la asignación de taxi y taxista La información tanto de taxi como de taxista debe de estar disponible para conocer quién está detrás del volante y le ofrece el servicio al usuario, con el fin de brindar seguridad al momento de que llegue dicho vehículo y sea el correspondiente. La placa, nombre del taxista, distancias y ubicación serán la clave

	para la aceptación de la carrera y su verificación a través de pruebas, dará como resultado su correcta asignación.
A-04	Prueba en el historial de usuario Se genera un historial de las asistencias de taxi que ha tomado ese usuario, para llevar un reporte personal del servicio y en caso de no haber calificado el servicio anteriormente poder realizarlo en un lapso disponible. Se debe probar que dicho historial se guarde, al igual que la calificación del taxista.
A-05	Pruebas del sistema Se realizan las respectivas pruebas del sistema, pruebas en un entorno real, donde todos los componentes que interactúan funcionen correctamente en conjunto, cumpliendo en su mayoría funcionalidades y no funcionalidades, para finalmente realizar el despliegue en las plataformas digitales y estar a disposición del público.
A-06	Pruebas de despliegue Se llevará a cabo el despliegue del software con la finalidad de poner a disposición la aplicación de manera efectiva, tanto en la tienda para la app como en la web para la página administrativa.

Fuente: elaboración propia

Limitaciones

- Se realizarán pruebas de campo en celulares con sistema operativo Android 6.0 o superior, a través de los usuarios y taxistas para que, en tiempo real, se corrobore la calidad de la aplicación y cumpla las funciones a fin de lo que fue desarrollado.
- No se van a adquirir nuevos dispositivos móviles o celulares para la realización de pruebas de desarrollo y demostración de la aplicación.

- El software no estará disponible para otras ciudades, solo para el municipio de Garzón y sus veredas aledañas.
- Los taxistas podrán gozar de los beneficios del software GTAXI, única y exclusivamente, a través de la cooperativa COOTRANSGAR quienes realizarán la vinculación y registro del vehículo y del conductor por medio del portal web.
- El software no genera tarifas de costos por trayectos o carreras ni aceptación de medios de pago digitales, pagos electrónicos, de tarjeta, o también conocido como pasarela de pagos.
- En este proyecto no se van a implementar nuevas funcionalidades de la aplicación ni se van a crear nuevas operatividades que amplíen la robustes del software, se limitará a llevar a cabo correcciones de las funciones ya creadas y su correcto despliegue.
- En el proyecto va a tomar en base un modelo de calidad sin embargo no va a estar certificado por el modelo.
- El despliegue final del software GTAXI para usuarios se realizará en la Google PlayStore y estará disponible solo para celulares Android que soporten este sistema operativo.

Cronograma

En el proyecto se estableció un cronograma de actividades el cual se fue cumpliendo progresivamente hasta llegar a su finalización. Si bien es cierto es una medición de tiempos para la ejecución a cabalidad de cada tarea, se fueron realizando modificaciones las cuales corrieron, en poca medida, los días programados, no obstante, esto no afectó el resultado final que se entregó con la realización de cada objetivo.

La siguiente tabla es el resultado del itinerario de tareas de manera global en un lapso definido en meses.

Tabla 2
Cronograma de actividades

#	Actividad	Meses											
		1	2	3	4	5	6	7	8	9	10	11	12
1	Comunicación con Cootransgar	X	X	X	X	X	X	X	X	X	X	X	X
2	Pruebas no funcionales		X	X	X								
3	Pruebas funcionales			X	X	X	X						
4	Pruebas aplicadas a la norma ISO/IEC 25000						X	X	X				
5	Implementación en entorno real							X	X	X	X		
6	Corrección de riesgos										X	X	
7	Escritura de informe y elaboración del artículo									X	X	X	X

Fuente: elaboración propia

Justificación

Teniendo en cuenta el proceso que ha llevado la cooperativa con el software de servicio de taxi, este proyecto surge por la necesidad de implementar y poner en marcha la aplicación tecnológica ya desarrollada, que se encuentre disponible al público y cubra los aspectos de seguridad y descongestión de la circulación vehicular.

Así mismo, el proyecto tiene la finalidad de evaluar la condición del software antes de su despliegue, a través del testeo con diferentes tipos de pruebas, que ayuden a mitigar errores que se encuentren: de usabilidad, funcionalidad, sistema, componentes, entre otros, para que al fin y al cabo se efectúe su realización.

Para cubrir este propósito se va a trabajar con los modelos de calidad y finalmente apreciar la aceptación de los actores que intervienen en el servicio, tanto usuarios, como taxistas y la asociación.

El software que está requiriendo del despliegue, se pondrá a disposición cumpliendo métricas que garanticen calidad, llegando así a beneficiar a los involucrados dando la certeza de un buen aplicativo a sus asociados y miembros vinculados.

Continuo a esto, el software al final de proceso será productivo y eficiente, para los usuarios que requieren de un servicio de transporte atento, seguro, ágil y eficaz, ya sean visitantes, residentes del municipio o veredas aledañas. Los taxistas que se verán beneficiados al contar con la herramienta puesta en marcha y la cooperativa que, en busca de actualizar procesos, dar al público general una aplicación tecnológica estructurada en el despliegue y ver esos beneficios reflejados y materializados.

Estado del arte

Según la empresa OWIUS enfocada en el desarrollo de apps en Barcelona, existen 10 típicos motivos por los cuales los proyectos de software no se terminan a cabalidad y fracasan. (OWIUS, 2019) entre los 5 primeros que dan a conocer son: “Requerimientos incompletos. Pobre inclusión de los usuarios en el proyecto. Falta de planificación y estrategias. Expectativas no realistas. Falta de soporte desde la gerencia”.

Muchos de los proyectos pensados o desarrollados no se finiquitan hasta la fase de despliegue y eso lo corrobora la columnista Nelli Aca del diario Merca2.0. quien afirma 3 razones por las que falla la implementación de los proyectos. 1. La deficiencia de supervisión objetiva, donde no se indican calendarios, responsables, ni documentación. 2. Falta de profesionalismo, al no involucrar un equipo comprometido. 3. Latosa planeación del proyecto, donde no hay roles definidos, metodologías claras ni políticas de trabajo (Aca, 2018).

Son por estos y más inconvenientes, que las empresas y las organizaciones trabajan bajo unas metodologías, su implementación guiada por modelos y la calidad respondiendo a unas métricas. En este aspecto, existen compañías tanto a nivel nacional como internacional que tienen el conocimiento de metodologías de implementación que llevan a cabo, trabajan bajo unos modelos que garantizan calidad en el producto y a la vez son certificadas en calidad de proceso o producto.

En el mercado encontramos AWS CodePipeline, herramienta de Amazon Web Services, la cual permite una canalización de implementación que, automáticamente labora con una de las estrategias de implementación como lo es la arquitectura Blue-Green Deployment. Según su descripción trabaja en un entorno original donde se aplican las pruebas y el mantenimiento, mientras que en un entorno clonado se dirige el tráfico y que este no se vea afectado. Una vez sea

exitoso el proceso interno de revisión, se intercambian las URL de los entornos para volver a encargarse del tráfico en vivo y el entorno restante queda libre para futuros testeos. Y así constantemente se va permutando de un entorno a otro (Amazon Web Services [AWS], s.f.).

Sumado a lo anterior, Amazon también ofrece una alternativa similar llamada AWS Elastic Beanstalk. Según (Amazon Web Services [AWS], s.f.) el funcionamiento de este servicio cuenta con dos entornos idénticos, azul y verde, que se encuentran separados el uno del otro y por esto se disminuye la exposición y se incrementa la disponibilidad. Mientras el entorno de producción trabaja con el tráfico en vivo (azul), el otro entorno se encarga de la arquitectura de canalización (verde). Al terminar e implementar se cambian las rutas o apuntadores y se invierten los roles entre los entornos.

En contraste a estas herramientas mencionadas, la empresa de servicios en la nube más adoptada en el mundo brinda la solución de AWS Route 53, definido por ellos como un "servicio de DNS web de alta disponibilidad en la nube y escalable, con el fin de ofrecer un método para redirigir y dividir el tráfico" (Amazon Web Services [AWS], s.f.). Tal como se realiza en la estrategia de implementación Canary Deployment.

Pero no solo encontramos herramientas existentes disponibles, sino también empresas de software que dan estos servicios de calidad y despliegue, como lo es el caso de SoftExpert, una empresa que brinda soluciones empresariales en herramientas digitales a las diferentes demandas. Según su propia página web "implementan software a medida de las necesidades y la incorporación de nuevas funciones" (SoftExpert, s.f.). El despliegue se realiza de acuerdo con el tamaño del software en cuestión y a través de su herramienta SoftExpert EQM garantizan la "gestión de calidad completa por medio de la certificación ISO 9001 gracias a procesos

automatizados y altamente interactivos, adaptados a las prácticas de negocio, productos y operaciones específicas de cada organización" (SoftExpert, s.f.).

Otra de las organizaciones con el mismo modelo de brindar soluciones tecnológicas a las organizaciones es F5, una empresa que dispone de recursos de equilibrio de cargas enfocadas en la nube y visto desde la implementación Blue-Green Deployment, "ayudan a reducir el riesgo a las aplicaciones que cada vez son más complejas, utilizando proveedores de canalización CI/CD para implementar simultáneamente el equilibrio de carga, la seguridad y consolide los servicios nativos duplicados en una solución de gestión de tráfico unificada" (F5, s.f.).

Continuo a la anterior idea, se encuentra en la web de la compañía Sumo Logic, una firma que labora en facilitar la captura y el análisis de eventos para monitorear errores introducidos, problemas de rendimiento y en general errores en las nuevas actualizaciones de software para evaluar el éxito de las implementaciones Blue-Green Deployment (Sumo Logic, s.f.).

Como vemos en los anteriores ejemplos, se ha venido trabajando en el tema de implementación de calidad. También encontramos empresas que brindan alternativas, libres de metodologías de implementación, es decir, que no están atadas a un modelo, sino que con sus herramientas trabajan para que el desarrollador sea independiente de elegir cómo laborar, un ejemplo de ello es el caso de Desktop Central de la corporación ManageEngine quieres otorgan una solución automatizada a los endpoint y que estos cuenten con la última versión de software instalada, por medio de la gestión de parches y de archivos de TI para así preservar un software actualizado, controlado y visible en su totalidad con la finalidad de facilitar en todo momento al desarrollador, permitiendo la distribución de paquetes de software a todos los equipos de su red sin importar su sistema operativo (ManageEngine, s.f.).

Como lo vimos en este capítulo de estado del arte, Blue-Green Deployment es de las estrategias de implementación de software más usadas en el mercado por compañías que llevan tiempo y conocen a cabalidad esta área, sin embargo, esto no quiere decir que no existan más metodologías de implementación, sino que, al brindar ciertas ventajas sobre el resto, es de las más optadas. Esta comparativa de las cualidades de cada estrategia de implementación se aborda con más claridad en el apartado de marco teórico del documento.

Marco teórico

Dispositivo móvil

"Un dispositivo móvil se define como un aparato de menor tamaño, con capacidades de procesamiento, con conexión intermitente o permanente a una red, diseñado específicamente para una función que puede a la vez llevar a cabo muchas más funciones" (Baz Alonso, Ferreira Artime, Álvarez Rodríguez, & García Baniello, pág. 1).

Sistemas operativos móviles

En primer lugar, encontramos la definición de sistemas operativos la cual nos menciona que un sistema operativo es el "software fundamental y principal, que permite a los usuarios interactuar con los dispositivos, gestionando datos, las aplicaciones y los componentes (…) permite una eficaz ejecución de los programas sin que haya conflicto entre estos" (Ranieri, Villar, & Rodríguez, pág. 3).

Es por ello, que todos los sistemas se diseñan para administrar eficientemente el manejo del equipo, permitiendo una eficaz ejecución de las aplicaciones y ajustes propios del dispositivo, sin que exista un conflicto entre ellos. El sistema de mano como también se le conoce a este grupo, están en constante evolución para mejorar sus prestaciones, adecuarse a las nuevas tecnologías que surgen y cubrir las necesidades de los usuarios. Hoy en día vemos como muchos de ellos se han ido adaptando a lo largo de la historia para llegar a diseños con entornos gráficos que faciliten su manejo y resulte sencillo e intuitivo para el usuario.

Un sistema operativo móvil en síntesis es la definición de un sistema operativo general como el de los computadores, que se controla en el dispositivo móvil, mucho más simple y orientados a estos equipos, como lo es la conectividad inalámbrica, la información rápida y los formatos multimedia.

Actualmente en el mercado encontramos 2 dominantes de sistemas operativos los cuales son: Android por parte de Google y iOS por parte de Apple. Algunos otros sistemas operativos ya no se encuentran fuertes en el diario vivir o están descontinuados como lo son Symbian, BlackBerry y Windows Mobile.

Android. Es un sistema operativo móvil "basado en Linux y Java que ha sido liberado bajo la licencia Apache versión 2 y hoy en día el sistema operativo para móviles más consumido en el mundo" (Baz et al., pág. 6). Este sistema busca un modelado estandarizado de programación en el campo de la telefonía móvil desarrollado por Google que, por su aspecto, promete ser una plataforma flexible, económica en el desarrollo de apps para los programadores y simple de interactuar (Android, s.f.).

iOS. iPhone OS o simplemente iOS, es una versión pequeña de Mac OS X perfeccionada para los procesadores de dispositivos móviles; su firma de Apple hace que no se instale en cualquier dispositivo si no se consta de la autorización de la empresa ya sea de la forma iPhone Developer Program o en sus dispositivos, su gran cantidad de herramientas hacen de iPhone una interfaz realmente interesante para el usuario (Baz et al., pág. 7).

Aplicaciones móviles

Una aplicación móvil es un software pensado y desarrollado para que sea ejecutado por medio de los dispositivos móviles, esto se refiere a poder acceder a varias funciones o datos, desde cualquier lugar debido a la portabilidad que hay en estos dispositivos de menor tamaño físico (Enriquez & Casas, 2014). En palabras concisas, "las aplicaciones son para los móviles lo que los programas son para los ordenadores" (Cuello & Vittone, 2013, pág. 14).

Tipos de aplicaciones móviles según su desarrollo. Dentro del desarrollo de aplicaciones móviles encontramos 3 tipos de apps en el mercado comúnmente: aplicación nativa, aplicación web y aplicación híbrida.

Aplicación nativa. "Las aplicaciones nativas son aquellas que se conciben para ejecutarse en una plataforma específica, es decir, se debe considerar el tipo de dispositivo, el sistema operativo y su versión" (Delía, Galdamez, Thomas, & Pesado, 2013, pág. 768). Cuando hablamos de este tipo de apps, su código fuente varía para cada plataforma por lo que algunas organizaciones colocan en una balanza los pros y los contras para este arquetipo de inversiones en desarrollo.

Dentro de las ventajas que encontramos en este tipo de aplicaciones se encuentra la posibilidad de interactuar, acceder y trabajar con los gadgets del dispositivo, como lo es la cámara u otras características del móvil. Además, su tiempo de ejecución de manera rápida o incluso en modo background notificando al usuario de cada suceso de la app es otro de los beneficios que entran dentro de lo nativo (Artica Navarro & Pita Astengo, 2014).

Por otra parte, uno de los no atractivos para este tipo de aplicaciones es el alto costo de desarrollo, debido al uso de un lenguaje de programación diferente según sea la plataforma. Por consiguiente, si se quiere cubrir varias plataformas, se deberá generar un desarrollo para cada una de ellas lo que genera a mayor plazo incrementos en los costos de actualización y distribución de nuevas versiones (Artica Navarro & Pita Astengo, 2014).

Con respecto a la deliberación de la app en la plataforma de distribución digital, en el libro Un Análisis Experimental de Tipo de Aplicaciones para Dispositivos Móviles nos menciona: "Estas tienen un proceso de auditoría para evaluar si la aplicación se adecúa a los requerimientos de la plataforma a operar. Cumplido este paso, la aplicación se pone a disposición de los usuarios." (Delía, Galdamez, Thomas, & Pesado, 2013, pág. 768).

Aplicación web. Hablamos de aplicaciones web para móviles aquellas apps que se adaptan al sistema operativo debido a que corren o se ejecutan en el navegador del dispositivo móvil, por lo que no tienen que programarse en uno u otro sistema. En su gran mayoría son desarrolladas utilizando un lenguaje JavaScript, es decir, lenguajes similares utilizados para crear sitios web (SOLBYTE, 2021).

Una de sus ventajas es que no necesitan de gran elaboración específica, es decir, su sencillez de desarrollo. No requieren de instalación de componentes particulares ni aprobaciones de fabricante, única y exclusivamente se requiere el acceso a internet. Otra de las ventajas que conocemos son sus actualizaciones que al ejecutarse se visualizan directamente en los dispositivos, además de no depender del sistema operativo que se posee. Sin embargo, dentro de sus desventajas se encuentra la disminución de la velocidad de ejecución además de ser más limitadas, este tipo de aplicaciones no pueden acceder a los elementos del dispositivo, como GPS, cámara, flash y demás componentes (Cadenas, 2019).

Aplicación híbrida o multiplataforma. Se conocen como híbridas debido a que combinan aspectos de las aplicaciones nativas, pero a la vez un solo desarrollo sin consumir altos recursos. A favor de esta tecnología se encuentra la facilidad del tipo del desarrollo, el cual en su mayoría de herramientas son gratuitas, la reutilización del código para las diferentes plataformas y pudiendo integrarse con todos los sistemas operativos de fácil manera sumado a utilizar las características del hardware del dispositivo (Angulo, 2013).

Aunque en otro sentido, al utilizar la misma interfaz para un sistema operativo y para otro, no será la misma experiencia que se busca brindar y cómo se ve en una aplicación nativa y finalmente la ejecución de la aplicación tardará más que el resto. Todo está en lo que se requiere para optar por una web, híbrida o nativa (Puetate, Ibarra, Villamagua Portilla, Garcés, & Ibarra).

Firebase

Es una plataforma digital creada por Google pensada en desarrollar y facilitar la creación de aplicaciones de calidad con rapidez, efectividad y sencillez. Ubicada en la nube, está disponible para plataformas iOS, Android y web, además de contar con diversidad de funciones para combinar en el desarrollo y adaptar a las necesidades a su medida (Giraldo, 2019).

Algunas de las características que cuenta Firebase son:

- Compatibilidad con extensiones de paquetes de código abierto para automatizar tareas de desarrollo comunes.
- Integración fácil con las herramientas favoritas de su equipo o dispositivo.
- Alto grado de crecimiento y escalabilidad.
- Es multiplataforma soportando Android, iOS y web, además de brindar con una excelente documentación y contar con APIs integradas a SDK individuales, de tal manera que se pueda gestionar sin tener que salir de la propia Firebase. (Muradas, 2021).

Dentro de Firebase encontramos los servicios que esta gran herramienta nos ofrece, todos diferentes y con distintas utilidades, los más destacados en desarrollo son:

- Realtime Database: nos proporciona una base de datos en tiempo real.
- Authentication: se ejecuta para identificar a los usuarios y llevar su registro.
- Storage: almacenamiento y envío de archivos a la escala de Google.
- Hosting: Nos ofrece un espacio para la publicación de nuestro software.
- Remote Config: se emplea para modificar ciertos aspectos de nuestra app sin la necesidad de actualizar la misma.
- Test Lab: Pruebas de la aplicación antes de publicarla.
- Crash Reporting: Reporte de errores de la aplicación. (López Mora, 2020)

Firebase Hosting. Es un servicio de alojamiento de hosting enfocado en microservicios y contenido tanto dinámico como estático. Cuenta con una CDN (red de entrega de contenido), además de incluir SSL sin necesidad de configuración para que, de forma segura, se realice la publicación de contenido. El hosting de Firebase nos permite cobijar APIs en el marco de trabajo de Express.js con la finalidad de compilar aplicaciones y actualizar datos en tiempo real.

Sumado a lo anterior, este servicio nos posibilita configurar secuencias de trabajo de producción, realizando consultas y pruebas de cambios, en forma local e interacción con recursos en backend emulado, para luego implementar los cambios realizados (Firebase, 2021).

A continuación, se grafica el flujo que maneja Firebase Hosting con la integración de Cloud Functions y Storage.

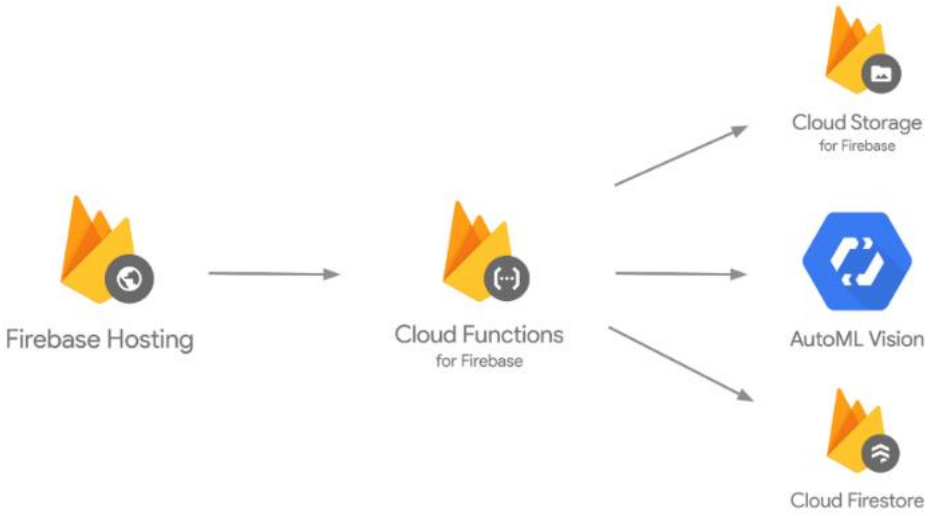


Figura 1. Modelo de Trabajo con Cloud Functions y Firebase.

Fuente: *(Robinson, 2018)*: Query a custom AutoML model with Cloud Functions and Firebase. Recuperado de https://hackernoon.com/

Calidad de software

Según la IEEE, Std. 610-1990, “la calidad del software es el grado con el que un sistema, componente o proceso cumple los requerimientos especificados y las necesidades o expectativas del cliente o usuario” (IEEE STD 610-1990).

Muy similar encontramos la definición por parte de ISO/IEC 25010, la cual nos menciona que la calidad es el: “grado en que dicho producto satisface los requisitos de sus usuarios aportando de esta manera un valor. Son precisamente estos requisitos, (...) los que se encuentran representados en el modelo de calidad, el cual categoriza la calidad del producto en características” (ISO/IEC 25010, 2011).

Por otro lado, tenemos la definición de (Pesado, y otros, 2006) que aseguran: “Las normas de aseguramiento de calidad definen un conjunto de reglas y patrones que deben ser cumplidos por las empresas desarrolladoras de software, para generar productos que alcancen umbrales mínimos de confiabilidad y satisfacción de usuario” (pág. 501).

En los temas de calidad de software, existen unos modelos que al tener diversos factores se componen de criterios evaluados por unas métricas propias de cada uno, con la finalidad de valorar desde lo más amplio hasta lo más complejo para finalmente asignar un valor cualitativo o cuantitativo.

Modelos de calidad. Según (Cuervo, Alarcón Aldana, & Álvarez Carreño, 2017) afirman que: “Los modelos de calidad de software se clasifican de acuerdo con el enfoque de evaluación, ya sea a nivel de proceso, producto o calidad de uso” (pág. 238). Cada uno de ellos con unas presentaciones de nivel que especifican características claves.

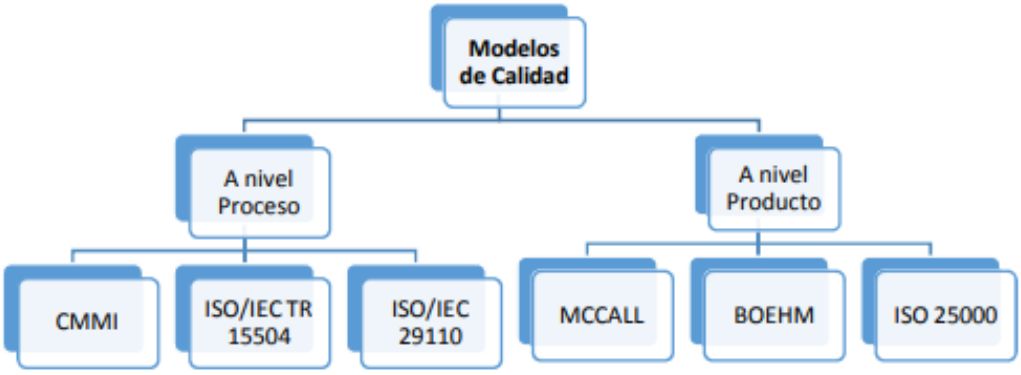


Figura 2. Modelos de Calidad de Software

Fuente: (Sánchez Jaya, Robayo Jácome, & López Sevilla, 2018). Apuntes teóricos sobre modelos de evaluación de calidad en procesos de desarrollo de software para personas no videntes. Recuperado de https://repositorio.pucesa.edu.ec/

Calidad a nivel de proceso. La calidad de proceso se refiere a una evaluación que se realiza de manera continua y seguida del proyecto, con el fin de ir evaluando etapa por etapa si se cumple o no con la calidad esperada, con la finalidad de verificar el cumplimiento en cada fase y mejorar continuamente los procesos que se llevan a cabo mientras se está desarrollando el software (Alarcos Quality Center).

Existen varias métricas que se agrupan en este nivel de proceso, dentro de las más conocidas encontramos:

- ISO 9000: Este grupo de normas tienen como objetivo instaurar fundamentos que conforman un sistema de calidad para que las empresas garanticen el control de las actividades que incurren en la calidad de los productos y servicios. Es por ello que este grupo de normas 9000n garantizan que las organizaciones cuentan con la capacidad de desarrollar software de confianza y cumplen con requisitos durante todo el ciclo de vida del servicio o producto (Hernández García, 2007).

- CMMI: Son un conjunto de buenas prácticas referidos en el desarrollo y mantenimiento, lo cual busca verificar el cumplimiento de la calidad cubriendo todo el transcurso del ciclo de vida del proyecto. CMMI trabaja bajo unos niveles de madurez los cuales son etapas que se establecen para evaluar el proyecto, desde su concepción hasta su cierre y optimización (Huayta García, 2006).
- ISO/IEC 20000: Esta agrupación de normas es pensada en generar beneficios en la gestión de los servicios de TI, por ello procura que se funden unos requerimientos para realizar entregas de productos o servicios que cumplan con la necesidad, objetivos y una producción de calidad. Dentro del concepto de la ISO 20000 está el potenciar tanto la calidad como el valor añadido y por ende la satisfacción del cliente, reduciendo los riesgos, costos y minimizar tiempos en el ciclo de vida (Oltra Badenes, 2017).

Calidad a nivel de producto. Tiene la finalidad de que, con unos criterios de producto definidos, se vayan evaluando la totalidad de los atributos clasificados en niveles jerárquico y sus respectivos cumplimientos, para así garantizar la satisfacción del cliente (Universidad Abierta y a Distancia de México, 2016). Dentro de la calidad a nivel de producto encontramos los siguientes modelos más destacados:

- GILB: es una pauta de calidad pensado en una valoración centrada en el usuario y que lo satisfaga. Su estructura se dividide en 4 atributos: capacidad de trabajo, adaptabilidad, disponibilidad y utilizabilidad, con tal de servir de apoyo a la gestión de proyectos (Modelo de Metricas de GILB, 2018).
- ISO 9126: es un estándar para la evaluación del software, el cual distingue entre los fallos y la no conformidad del usuario. Su foco se divide en 4 secciones las cuales son: modelo de calidad, métricas externas, métricas internas y calidad en las métricas de uso;

el primero de estas divisiones se encuentra fraccionado en 6 atributos clave de la calidad los cuales son: funcionalidad, fiabilidad, usabilidad, eficiencia, mantenibilidad y portabilidad. Con esto, cada empresa define un modelo de calidad con objetivos y métricas para su producto (Sicilia, 2007).

- ISO 25000: es un modelo que se compone de 8 características: adecuación funcional, eficiencia de desempeño, compatibilidad, capacidad de uso, fiabilidad, seguridad, mantenibilidad y portabilidad. Estas se enfatizan al producto software con el propósito de crear modelos, métricas, procesos y herramientas que guían el desarrollo de requisitos para la evaluación de propiedades de calidad y estas enfocadas en el software como un producto (Paola, Morales, & Gutiérrez, 2015).

Calidad en uso. Es definida como un conjunto de atributos centrados en la consecución de objetivos específicos relacionados con la efectividad, productividad, satisfacción y seguridad, todos acorde y basados en la aceptación por parte del usuario final (Estayno, Dapozo, Cuenca Pletsch, & Greiner, 2009).

Pruebas de software

Según la definición de prueba del (SWEBOK, 2004) afirma que es: “Una actividad realizada con el objetivo de evaluar la calidad del producto y mejorarlo, identificando defectos y problemas”. Como complemento a lo anterior, el International Software Testing Qualifications Board ISTQB (como se citó en Mera Paz, 2016), define la prueba de software como: “La verificación dinámica del comportamiento de un programa contra el comportamiento esperado, usando un conjunto finito de casos de prueba, seleccionados de manera adecuada” (pág. 168).

A partir de estas y muchos conceptos, existen diferentes tipos de pruebas de software que buscan el objetivo en común de la calidad de software.

Tipos de pruebas de software. Referente a los tipos de pruebas existentes para el software, existen varias clasificaciones. De acuerdo con la clasificación por verificación, el portal educativo Platzi nos da a conocer 4 tipos de pruebas las cuales son: pruebas funcionales, pruebas no funcionales, pruebas estructurales y pruebas de manejo de cambios (Morales, 2019).

Pruebas funcionales. Según la organización de soluciones tecnológicas Visual Engineering, se definen las pruebas funcionales como ese "Tipo de pruebas que revisan tanto el comportamiento del sistema, subsistema o componente de software. Son unas pruebas documentadas en especificaciones de requisitos o en casos de uso" (Ares, 2017).

Además de testear la funcionalidad del software, en dichas pruebas funcionales también nos cuestionamos el qué se debe estar haciendo y el cómo están interactuando con los usuarios.

En algunos sitios se precisa también como Pruebas de caja negra o "black-box testing" en su defecto en inglés.

Dentro de este tipo de pruebas encontramos: pruebas unitarias, pruebas de sistema, pruebas de aceptación, pruebas de sanidad, pruebas de integración, pruebas de componentes, entre otros.

Como complemento adicional, la empresa Trycode en su página web brinda un resumen en imágenes de las pruebas funcionales y las pruebas internas que la conforman:

Figura 3. Tipos de pruebas funcionales I

Fuente: (Vargas, Tipos de pruebas funcionales para el aseguramiento de la calidad, s.f.). Recuperado de https://trycore.co/transformacion-digital/tipos-de-pruebas-funcionales/

Figura 4. Tipos de pruebas funcionales II

Fuente: (Vargas, Tipos de pruebas funcionales para el aseguramiento de la calidad, s.f.). Recuperado de https://trycore.co/transformacion-digital/tipos-de-pruebas-funcionales/

Pruebas no funcionales. Según (Araya Solís, Méndez Marín, & Jiménez Segura, 2014), se definen las pruebas no funcionales como esas que evalúan al software en aspectos de características de funcionamiento, mas no el funcionamiento como tal, y si se cumple o no con los requerimientos no funcionales del cliente que se dialogaron inicialmente. Dentro de este tipo encontramos testing enfocado en de usuario, seguridad, diseño y, en general de calidad de sistema. En el seno de estas pruebas encontramos algunas como lo son: pruebas de rendimiento, pruebas de estrés, pruebas de carga, pruebas de portabilidad, pruebas de usabilidad, entre otros.

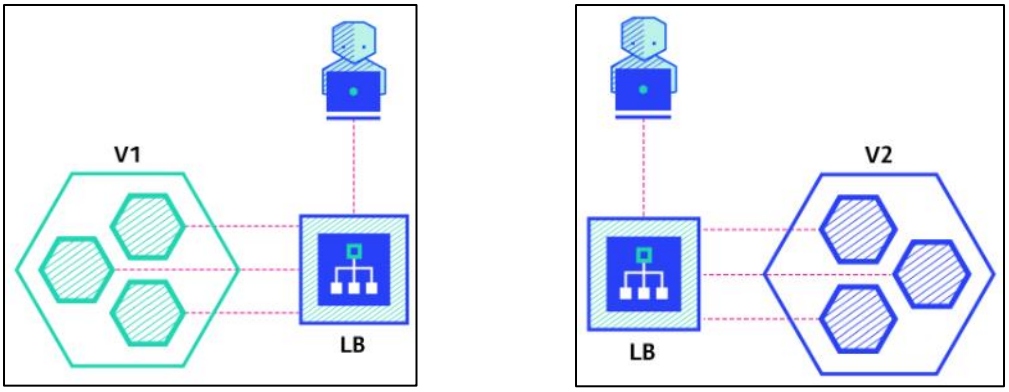
Pruebas estructurales. También conocidas como pruebas de caja blanca, son aquellas pruebas que, con la perspectiva del desarrollador, se evalúa el cómo funciona nuestro sistema internamente, es decir, el comportamiento interno donde se revisan los componentes y la integración de estos. Por lo tanto, el tester en este tipo de pruebas debe de tener a la mano, conocer y comprender el programa y el código fuente (Castro Alvites, 2019).

Pruebas de manejo de cambios o re-pruebas. Estas pruebas se realizan para confirmar que un error fue eliminado cuando se corrigió. Además de cubrir solamente el error o componente modificado, es importante repasar, chequear, examinar y verificar todo para descubrir cualquier defecto como consecuencia de la variación que se hizo anteriormente (PMO Informática, 2014).

Implementación de software

Estrategias de implementación de software. Se describe como estrategia de implementación aquella "forma de cambiar o actualizar una aplicación con el objetivo de realizar el cambio sin tiempo de inactividad de manera que el usuario apenas note las mejoras" (RedHat OpenShift). Con estas estrategias se verifica la preparación para determinar si la nueva actualización está lista para usarse. Dentro de ellas encontramos las siguientes.

Basic Deployment. Es la estrategia de implementación más sencilla ya que consiste en cerrar la versión A y luego implementar la versión B del software, después de que se finalice o se de cierre a la versión A. Esta técnica es poco recomendada en desarrollos complejos debido a que implica un tiempo donde el software va a estar inactivo o el propio servicio, además de no contar con reversión en caso de fallos. Dentro de sus pros encontramos su fácil configuración y económica (Harness, 2021).



(A) (B)

Figura 5. Basic Deployment.

A) Interacción con la versión 1. B) Interacción con la versión 2 y su anterior desaparece.

Fuente: (Tremel, Six Strategies for Application Deployment, 2017) Recuperado de https://thenewstack.io/deployment-strategies/

Multi-Service Deployment. En esta implementación se encuentran todos los nodos en un entorno que se actualizan en conjunto con múltiples servicios nuevos. Es usado comúnmente en productos que tienen dependencias de versión o cuando se hacen implementaciones fuera del horario de atención de empresas para recursos que no estén en uso. Por supuesto, destaca que es una implementación sencilla, rápida, económica y menos propensa a riesgos como lo es la implementación básica (Harness, s.f.).

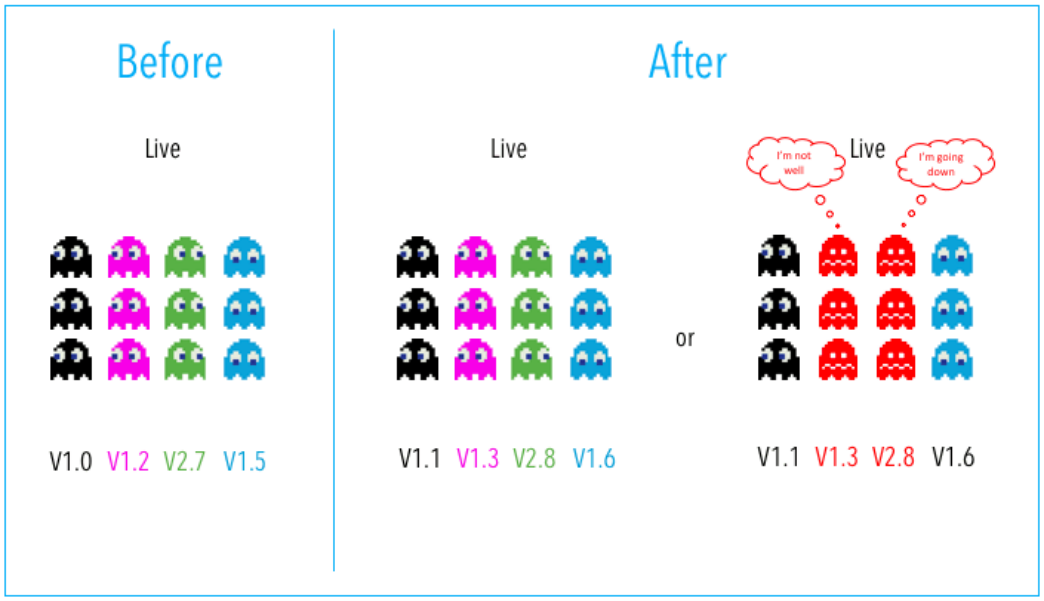


Figura 6. Multi-service deployment

Fuente: (Harness, Intro To Deployment Strategies: Blue-Green, Canary, And More, 2021). Recuperado de https://harness.io/blog/blue-green-canary-deployment-strategies/

Rolling Deployment. En este tipo de implementación, se van actualizando las instancias en ejecución de una aplicación con la nueva versión, bajando el servicio en una y subiendo otra. Como su nombre lo indica es un progreso o una forma incremental donde su destino se va a ver modificado. Dentro de sus ventajas se encuentra que es una implementación sencilla de revertir en caso de errores, es menos riesgoso que una implementación básica, además de ser más ágil que un blue/green deployment. Sin embargo, dentro de sus contras se encuentra que, debido a que los nodos se van actualizando en lotes esta implementación requiere de servicios para admitir tanto nuevas como antiguas versiones de un artefacto, lo que genera que cada verificación de cada cambio incremental se tarde o sea lenta su implementación (Amanda, 2018).

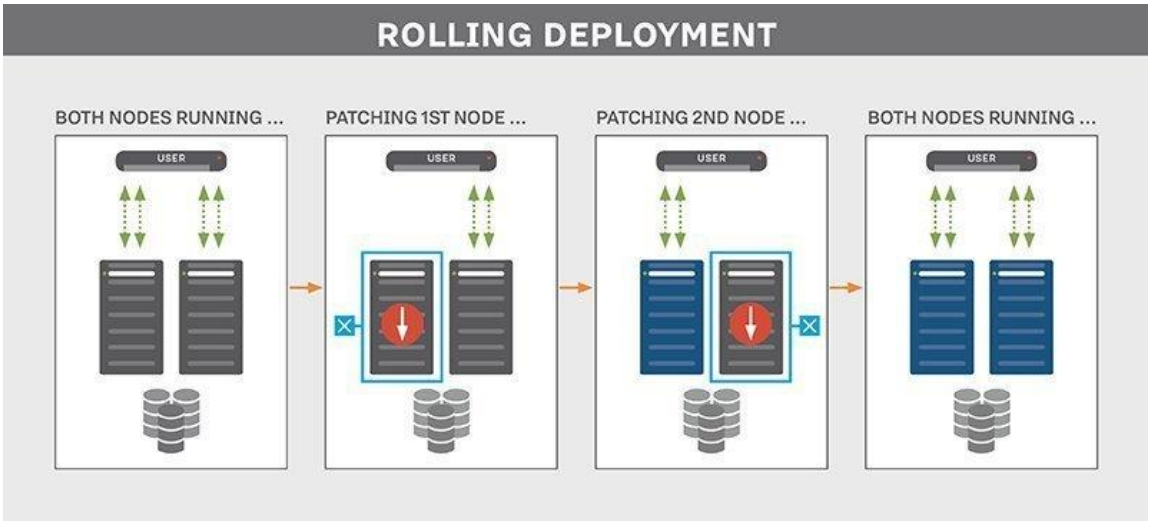


Figura 7. Rolling deployment

Fuente: (Bigelow, Rolling deployment, 2016). Recuperado de https://searchitoperations.techtarget.com/definition/rolling-deployment

Canary Deployments. La implementación canary se basa en ir cambiando gradualmente el tráfico de la versión A, a la versión B. Muchas de las veces cuando se usa canary, el tráfico se va dividiendo en función del peso, así se reparten las solicitudes hasta finalmente transferir todas las peticiones a la nueva versión. Es comúnmente usada cuando hay pocas pruebas, no son confiables, o cuando hay poca confianza en la nueva versión. Por su puesto, dentro de sus virtudes se encuentra que es muy conveniente para la tasa de error y la supervisión del rendimiento, y en un caso negativo, retroceder rápidamente a una versión anterior (Tremel, Six Strategies for Application Deployment, 2017).

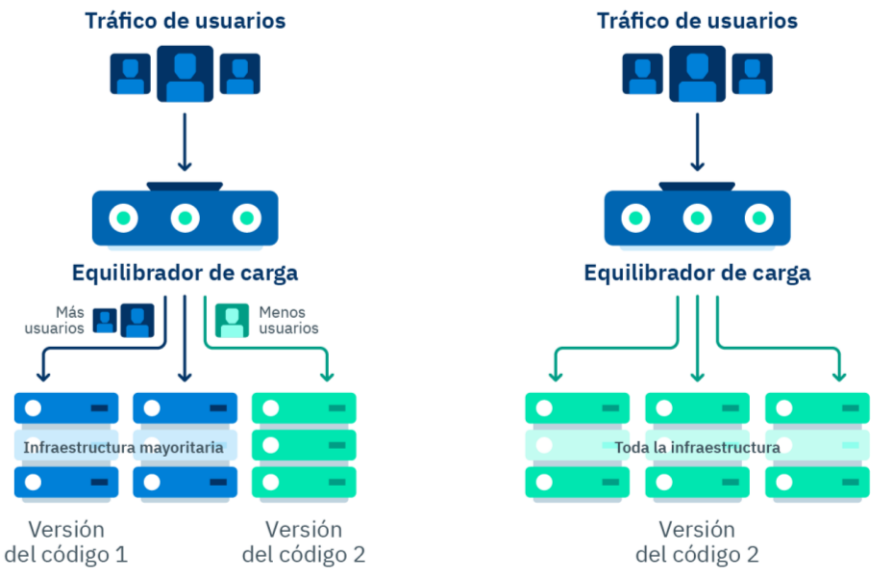


Figura 8. Canary deployment

Fuente: (Moreano, Canary Deployments and how to sleep peacefully at night, 2020). Recuperdado de https://www.kushki.com/en/blog/deployments-tipo-canary-y-como-dormir-tranquilos-por-las-noches

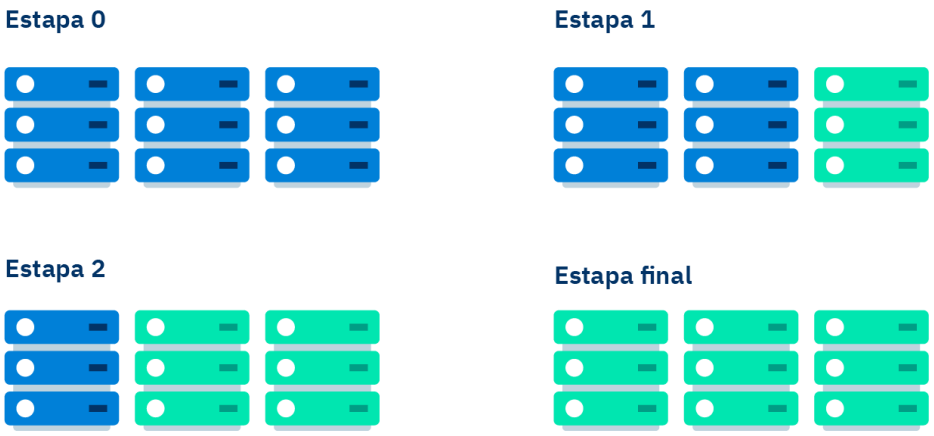
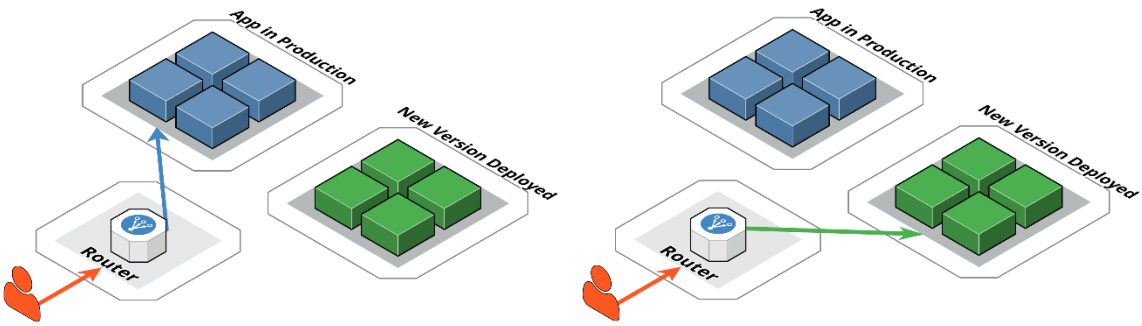


Figura 9. Ejemplo de cargas con Canary deployment

Fuente: (Moreano, Canary Deployments and how to sleep peacefully at night, 2020). Recuperado de https://www.kushki.com/en/blog/deployments-tipo-canary-y-como-dormir-tranquilos-por-las-noches

Nota: en el ejemplo, vemos reflejado que en la Estapa 0 se encuentra implementada la versión de código 1, posteriormente, en la Estapa 1 se observa la división donde se encuentra en su mayoría la versión del código 1 y en menor proporción la versión del código 2. En la Estapa 2 miramos el avance de la versión del código 2 en mayor proporción y finalmente en la Estapa final percibimos la implementación completa de la versión del código 2 en su totalidad.

Blue-Green Deployment. En esta implementación se utilizan 2 entornos idénticos, el primero es un entorno de ensayo, éste es nombrado entorno "azul" y el segundo es un entorno de producción que es nombrado entorno "verde", ambos con versiones diferentes de la aplicación. El tráfico de usuarios se traslada del entorno verde al entorno azul, una vez que se ha testeado y aceptado los cambios en el entorno azul. Luego, se vuelve a cambiar al nuevo entorno una vez que la implementación sea exitosa (Amazon Web Services, 2020, pág. 15). Dentro de sus desventajas se encuentra el alto costo de correr 2 versiones en simultáneo. Su réplica puede ser compleja y más cuando se trabaja con microservicios. Sumado a esto, algunas pruebas de garantía de calidad no llegan a identificar todas las anomalías que pueden presentar riesgos. Sin embargo, dentro de los beneficios se encuentra que es una implementación simple, rápida, bien entendida y su revisión es sencilla porque se puede interactuar con el tráfico y llevarlo a un entorno anterior en caso de algún inconveniente (Split, s.f.).



(A) (B)

Figura 10. Blue-green deployment

A) Tráfico dirigido a la versión azul en implementación y a disposición del público, mientras se trabaja en una nueva actualización en versión verde

B) Tráfico dirigido a la versión verde la cual es una actualización ya mejorada

Fuente: (Candost's Space, The Blue-Green Deployment Strategy, 2021). Recuperado de https://candost.blog/the-blue-green-deployment-strategy/

A/B Deployment. La estrategia de implementación A/B permite testear la aplicación en un entorno de producción donde se esté enfocando. Mientras una versión recibe la mayoría de las peticiones de los usuarios, la otra versión recibe una limitada fracción. En dicho proceso se pueden ir variando las porciones de solicitudes de cada versión de manera incremental hasta tal punto de dejar de usar la versión anterior, enrutar y estar al 100% en la nueva versión, dejando así a la anterior obsoleta pero lista para un nuevo proceso.

En esta implementación también, tanto versión A como versión B deben ser lo suficientemente similares para que ambas se ejecuten en simultaneo y al mismo tiempo. Las versiones pueden variar, sin embargo, el servicio que se prueba es el mismo.

Dentro de los pros de este tipo de prueba, es que es un método estándar y económico para probar funciones de producción. Uno de los inconvenientes es que las pruebas pueden alterar la

naturaleza experimental de su caso de uso, llegando a tal grado de que pueden escalar a ser unas pruebas tan complejas y hasta cierto punto de dañar la experiencia de usuario (Red Hat OpenShift).

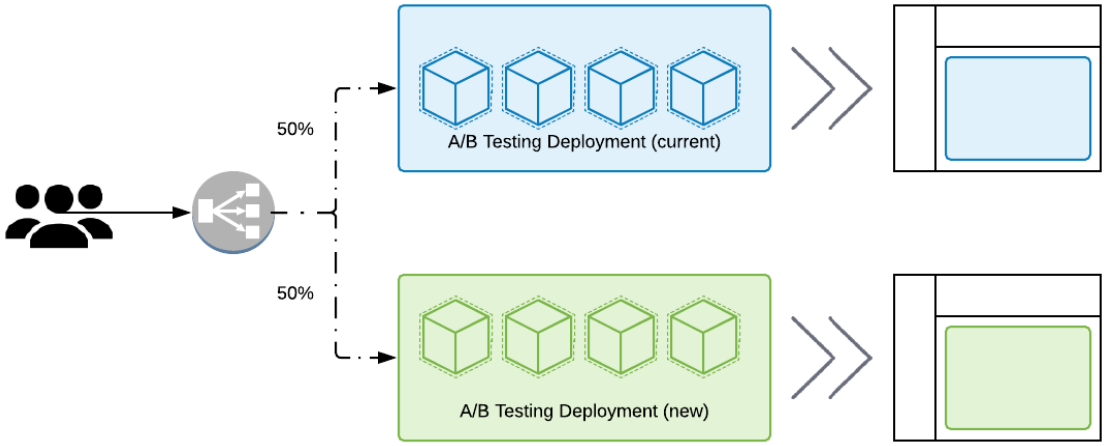


Figura 11. A/B deployment

Fuente: (Fugaro & Vocale, A/B testing deployment, 2019). Recuperado de https://subscription.packtpub.com/book/cloud_and_networking/9781788837866/9/ch09lvl1sec58/a-b-testing-deployment

Shadow. Esta implementación radica en lanzar a la vez tanto la versión B o versión ficticia o shadow, junto con la versión A. Las solicitudes entrantes de la versión A también se envían a la versión B sin afectar el tráfico. Con esto probamos la carga de producción en una nueva función y cuando se cumple con una estabilidad y rendimiento entonces se implementa a la app. Esta técnica es muy compleja de configurar y si bien, podemos ejecutar ambos entornos donde el shadow recibe la copia del tráfico original, necesita de unos requisitos especiales en el tráfico de salida, además de consumir el doble de recursos y servicio de burla para ciertos casos (Sahoo, 2021).

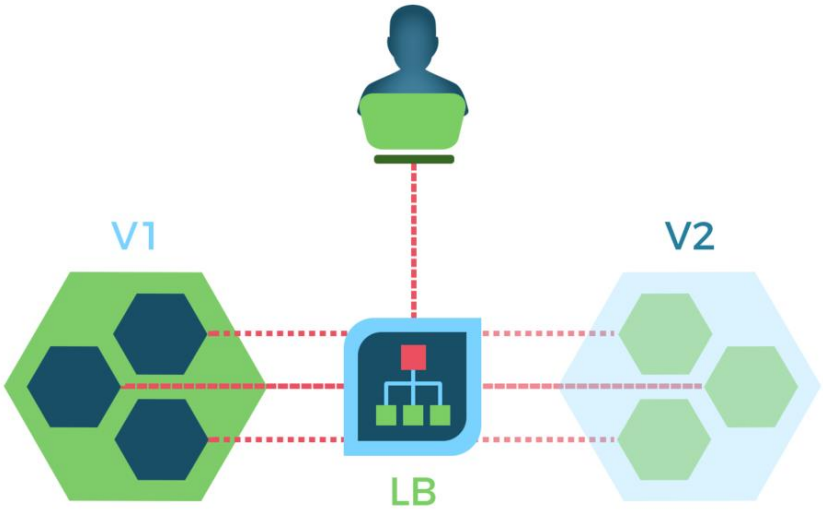


Figura 12. Shadow deployment

Fuente: (ChristopherGS, Deploying Machine Learning Models in Shadow Mode, 2020). Recuperado de https://christophergs.com/machine%20learning/2019/03/30/deploying-machine-learning-applications-in-shadow-mode/

Versionado de software

El versionamiento de software, según en la plataforma educativa EDteam, es la asignación de un rótulo único a un software el cual sirve para identificar el nivel del desarrollo en el que se encuentra, sus actualizaciones y entregas realizadas (Luján, s.f.).

Por supuesto, este proceso trae consigo beneficios, tanto en el desarrollo como en el despliegue, que según en el portal Kyocode son:

- Identificación de bugs a partir de una versión.
- Regresar la aplicación a versiones anteriores en caso de ser necesario.
- Comparación entre versiones.
- Control de versiones en equipos de trabajo (Álvarez, 2019).

Y, si bien es cierto que cada empresa puede establecer sus propias reglas de versionado, también existen unas pautas y unas versiones ya existentes, que muchas de las compañías aplican y son importantes al desplegar el software, entre estas encontramos:

- Versiones por número: manejo de versiones mediante 3 números, X.Y.Z.
- Versiones por estabilidad: sumado a la numeración se deja una clasificación de estable o inestable.
- Versiones por fecha: se anota la fecha de publicación.
- Versiones por parche: adición de un dígito a la numeración. X.Y.Z.P.

Versiones por número. De las más comunes en los desarrollos de software y su estructura basada en 3 números son asignados mediante las siguientes indicaciones.

- El primero (X): es la versión mayor y nos indica la versión principal. Cambia cuando se agregan nuevas funcionalidades importantes como lo son módulos o características para la funcionalidad que cambian drásticamente.
- El segundo (Y): son los cambios a la funcionalidad ya existente, correcciones de fallos en el sistema o incluso se agregan funcionalidades poco cruciales en el proyecto.
- El tercero (Z): también conocido como revisión y nos indica que se hizo la inspección del código por algún fallo. Cambia cada que se entrega el proyecto.

Diseño

Con el software establecido, se aclaró el funcionamiento y la interacción de los actores. A continuación, se detallan los diagramas de caso de uso UML (Lenguaje de Modelado Unificado) realizados con base en la explicación de la cooperativa debido a que anteriormente no se detallaban ni contaban con diagramas que explicaran los roles dentro del software y cómo este implicaba a cada uno de ellos.

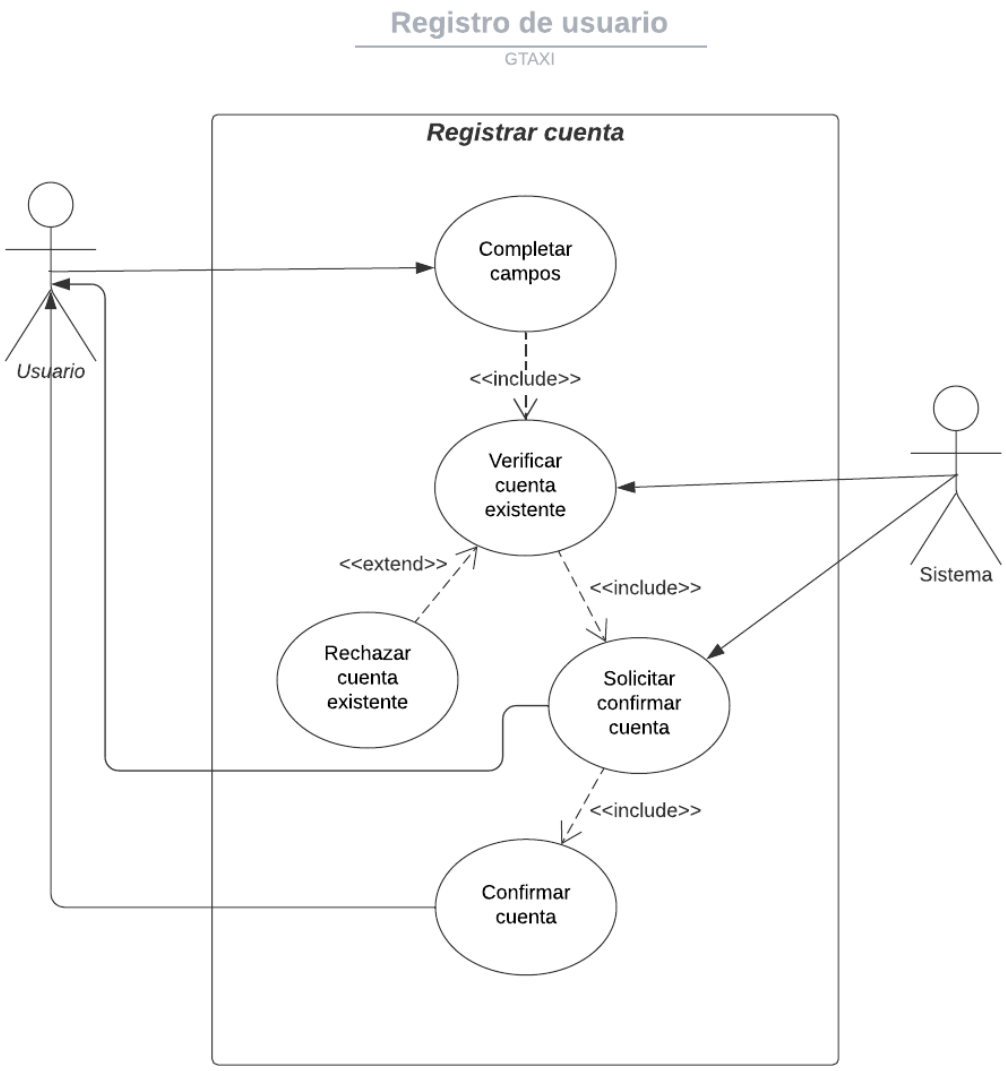


Figura 13. Diagrama de caso de uso para registro de nuevo usuario

Fuente: elaboración propia. Realizada en lucidchart.com

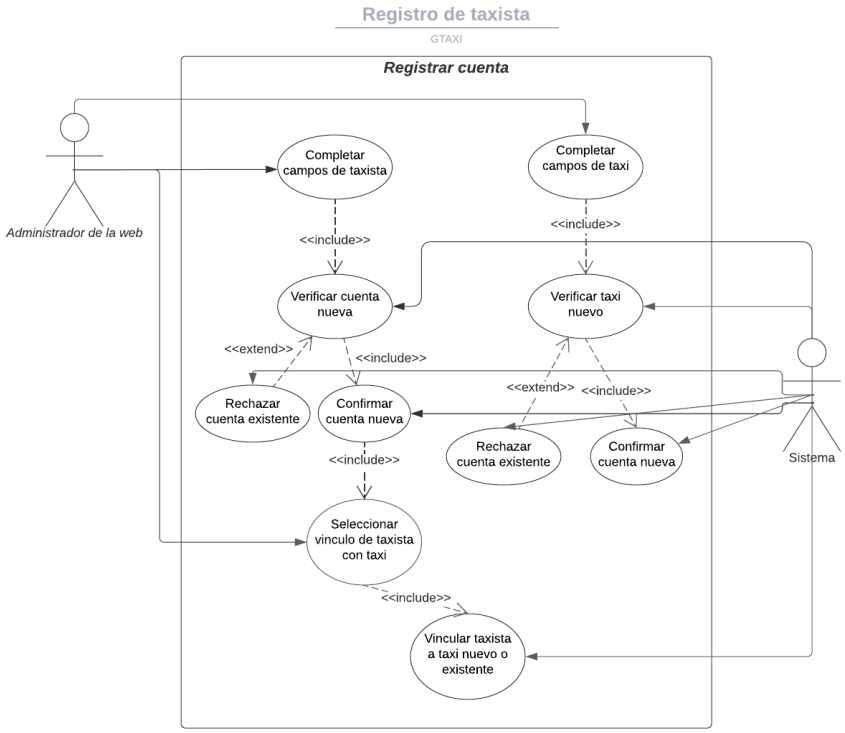


Figura 14. Diagrama de caso de uso para registro de nuevo taxista

Fuente: elaboración propia. Realizada en lucidchart.com

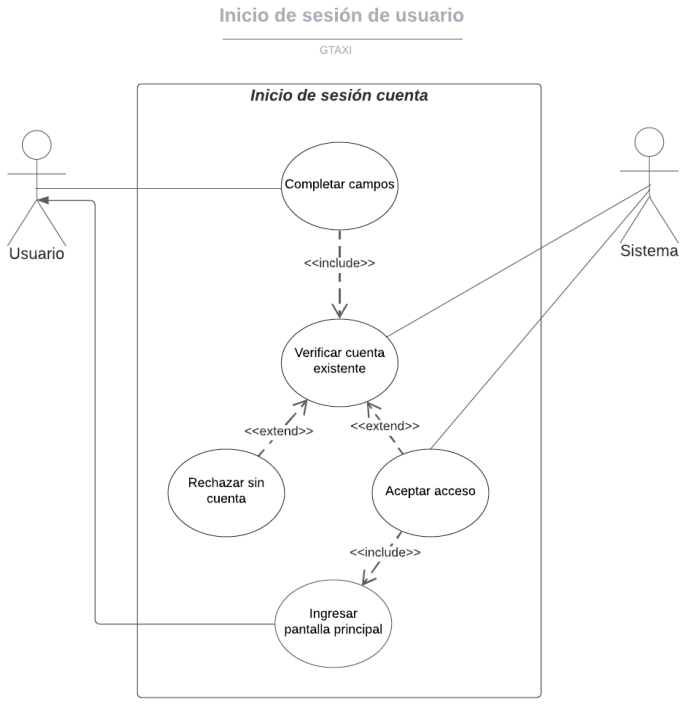


Figura 15. Diagrama de caso de uso para inicio de sesión de usuario

Fuente: elaboración propia. Realizada en lucidchart.com

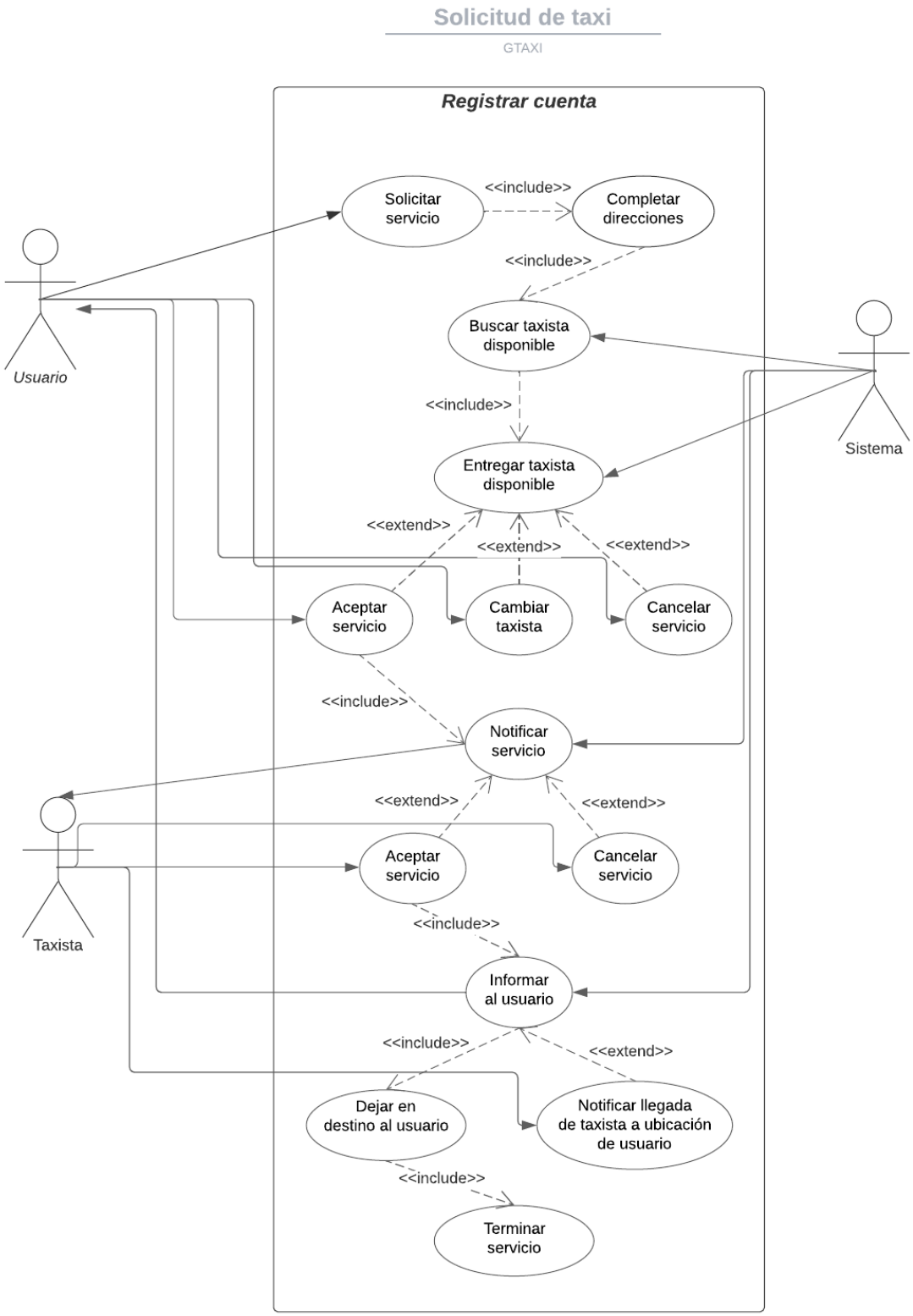


Figura 16. Diagrama de caso de uso para solicitud del servicio de taxi

Fuente: elaboración propia. Realizada en lucidchart.com

Metodología

La metodología desarrollada en este trabajo es guiada bajo varios estándares de calidad con la finalidad de cumplir el objetivo final de la cooperativa, se tendrá en cuenta los 3 últimos procesos del PMBOK que se basan en la ejecución, en el monitoreo y control, y en el cierre, estos enfocados en las pruebas y versiones para llevar a cabo el proyecto.

- Ejecución: con la versión del software de GTAXI hasta el momento, se ejecuta y se trabaja en los entornos de dispositivos móviles de manera general, se deja clara la versión y se pasa a un proceso de monitoreo y control para realizar las pruebas de la ejecución, en caso de existir errores se devuelve a ejecución donde se corregirán los errores.
- Monitoreo y control: en esta fase, se monitorea cómo se involucran los interesados del proyecto, se mitigan riesgos, validamos el alcance, realizamos las pruebas necesarias, versionamiento y finalmente realizar un seguimiento de todo el trabajo y su control integrado de cambios. En caso de encontrar errores se devuelve al proceso de ejecución para realizar las respectivas correcciones y se vuelve a monitorear el software.
- Cierre: Se da por concluido y procede al cierre del proyecto

Metodología de implementación de software a acentuar para la app

Con lo descrito hasta el momento y con el conocimiento de ciertas metodologías o estrategias de implementación de software, la que mejor se acomoda al requerimiento final de la cooperativa COOTRANSGAR y la limitante de equipos es la estrategia Multi-Service Deployment o en su interpretación en español: Implementación de Servicios Múltiples; una metodología que, por su dinamismo en el despliegue es ideal para la evaluación de objetivos, revisiones y correcciones.

Esta metodología es un proceso en el que se aplican, de manera regular, un conjunto de buenas prácticas para trabajar con el despliegue de nuevas versiones previamente testeadas y así obtener el mejor resultado posible en el proyecto.

En la implementación de servicios múltiple se realizan instauraciones parciales y regulares del producto final. Por ello, esta metodología está especialmente indicado para proyectos en entornos versionados, donde se necesita testear constantemente, donde se puede realizar despliegue y retroceso, que brinda economía en su puesta en marcha y donde se evitan riesgos en comparación a una Basic Deployment.

Para finalizar, uno de los agregados que evidenció este proyecto es el trabajo enfocado a la cooperativa, donde el software existente es propio de ellos, para sus requerimientos locales y las pruebas e implementación va a ser acorde a su entorno requerido, no son pruebas fuera de los límites o abruptas. Al analizar los resultados de pruebas en las aplicaciones, tanto de usuarios como de taxistas y en la página web administrativa, se determinó el cumplimiento del proyecto para llevar a la práctica el software.

Al efectuar el despliegue final con la última versión, el software está disponible a todo público, que será el que realmente calificará el software como un todo, desde sesión hasta petición de taxi y por supuesto la cooperativa, quien dará el aprobado y visto bueno de la aplicación y su exhibición, por medio de la tienda virtual PlayStore.

Manejo de la infraestructura de alojamiento web

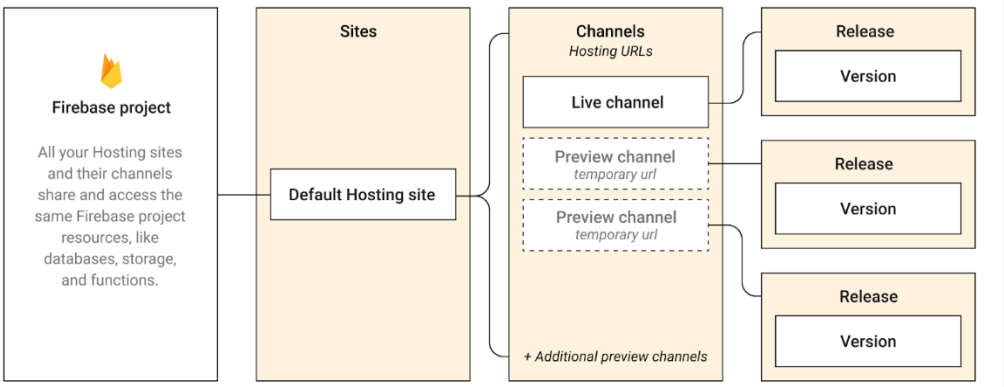


Figura 17. Infraestructura de alojamiento en Firebase con gestión de canales y versiones

Fuente: (Firebase, s.f.) Manage live & preview channels, releases, and versions for your site. Recuperado de: https://firebase.google.com/docs/hosting/manage-hosting-resources?hl=cs

Para el proceso de testeo, versionado y despliegue de la página administrativa donde la cooperativa va a vincular a sus taxistas asociados, se llevó un proceso de implementación básica o también conocida como recreate deployment en conjunto con Firebase, donde se fue actualizando la web del administrador a medida que iban surgiendo nuevos errores encontrados.

En el servicio de alojamiento se utilizó Firebase Hosting, una herramienta de la compañía de Google para la publicación y administración de canales, lanzamientos y versiones del software, que se controla a través de la consola de Firebase y su CLI.

Desarrollo

En primer lugar, se realizó una compilación de sugerencias escuchando a los taxistas sobre qué debía mejorar la aplicación enfocada para su beneficio y qué aspectos visuales se podrían perfeccionar. Luego se escuchó la opinión de la administración de Cootransgar, quienes dieron su punto de vista, tanto de las apps móviles como de la página web administrativa de Gtaxi. Con esta información tomada en cuenta, se revisó el software que se recibió inicialmente, desde su funcionalidad como su entorno y de allí se partió el trabajo a desarrollar.

Versiones de software

Aplicación móvil. Hasta el momento actual del aplicativo de usuario y taxi, se ha llegado a la versión 1.2.9 en los cuales se mejoró la interconexión de petición de taxi, registro de cuentas de usuario de manera más ágil y sin inconvenientes al guardar datos, mejoras en el diseño e interfaz de usuarios taxistas y se aplicaron las respectivas pruebas, funcionales y no funcionales, para la satisfacción exitosa de la app en su versión final.

En la siguiente tabla observamos las distintas versiones y los cambios realizados hasta el momento, llevando a cabo modificaciones de la versión inicial hasta llegar a la versión final actual.

Tabla 3
Versionamiento del software de la app de usuario

Número de versión	Fecha de ejecución	Descripción
1.2.12	20/01/2022	Corrección con el tamaño de los logos de la aplicación y publicación a la Google PlayStore
1.2.11	15/01/2022	Corrección con la forma de los botones, la cooperativa optó por botones en forma cuadrada para el inicio de la aplicación
1.2.10	11/01/2022	Corrección al momento de ingresar, hubo un inconveniente en el código fuente donde no se permitía logear el usuario
1.2.9	10/01/2022	Se agregó la dirección de destino en los campos donde el taxista puede ver la información del usuario que solicita el servicio.
1.2.8	5/01/2022	Se ajustó para que se ingrese la dirección donde se encuentra el usuario y la dirección de destino de la carrera
1.2.7	3/11/2021	Se realizaron pruebas con más de 5 conductores y 5 usuarios simulados. Se corrigió listas de asignación de taxi y se amplió el tiempo para aceptar o rechazar la carrera
1.2.6	6/10/2021	Se corrigió un error con los tiempos de sanción del taxista
1.2.5	29/07/2021	Se cambió y mejoró la claridad de redacción de los textos Se agregó la capacidad configurar el zoom del mapa Se agregó la capacidad de activar y desactivar el zoom del mapa
1.1.4	29/07/2021	Se corrigió un error al entrar en la información del viaje que cancelaba la carrera Se limitó el mapa para solicitudes solo dentro del municipio y veredas aledañas
1.1.3	22/07/2021	Se corrigió un error con el nombre del icono de la aplicación donde este no aparecía
1.1.2	10/03/2021	Se agrego un sistema que determina si un taxista está en activo o sancionado
1.0.0	9/03/2021	Versión inicial Incluye las siguientes características: Logueo, Mapa, Información del viaje, Marcado de rutas, Cancelar viaje, Activar/Desactivar servicio, Cerrar sección

Fuente: elaboración propia

Tabla 4
Versionamiento del software de la app de taxista

Número de versión	Fecha de ejecución	Descripción
1.2.8	5/01/2022	Se ajustó para que se visualice la dirección donde se encuentra digitada por el usuario y la dirección de destino de la carrera. Se cambió la imagen principal de fondo cuando el taxista ingresa a la aplicación
1.2.7	3/11/2021	Se realizaron pruebas con más de 5 conductores y 5 usuarios simulados. Se corrigió el tiempo en que se le asignaba al taxista un usuario y se amplió el tiempo para aceptar o rechazar la carrera
1.2.6	6/10/2021	Se corrigió un error con los tiempos de sanción del taxista
1.2.5	29/07/2021	Se cambió y mejoró la claridad de redacción de los textos Se agregó la capacidad configurar el zoom del mapa Se agregó la capacidad de activar y desactivar el zoom del mapa
1.1.4	29/07/2021	Se corrigió un error al entrar en la información del viaje que cancelaba la carrera Se limitó el mapa para solicitudes solo dentro del municipio y veredas aledañas
1.1.3	22/07/2021	Se corrigió un error con el nombre del icono de la aplicación donde este no aparecía
1.1.2	10/03/2021	Se agrego un sistema que determina si un taxista está en activo o sancionado
1.0.0	9/03/2021	Versión inicial Incluye las siguientes características: Logueo, Mapa, Información del viaje, Marcado de rutas, Cancelar viaje, Activar/Desactivar servicio, Cerrar sección

Fuente: elaboración propia

Aplicación web. En el programa web administrativo de Gtaxi, se ha llegado a la versión 1.5.4 donde se han venido realizando, en el transcurso de todas las versiones, pruebas tanto funcionales como no funcionales para tener el cumplimento de calidad en la herramienta que será clave para el software de solicitud de taxi y en especial para los taxistas, quienes serán administrados mediante la web por parte de la cooperativa, desde procesos de vinculación, consulta de datos, sanciones, eliminación y distribución del aplicativo.

Al igual que con la versión de la app móvil, se llevaron a cabo ciertos cambios los condujeron a las modificaciones hasta la versión final actual.

Tabla 5
Versionamiento del software de la web

Número de versión	Fecha de ejecución	Descripción
1.5.4	06/10/2021	Realización de pruebas funcionales finales.
1.5.3	01/09/2021	Se realizaron pruebas no funcionales finales y se observó mejoras en las cargas de la web.
1.5.2	27/08/2021	Se configuró la actualización de datos de taxi y taxistas dentro del módulo de consulta y no por separado como se tenía anteriormente.
1.5.1	12/08/2021	Se colocó nuevo color en el hover del puntero al pasar por el menú lateral.
1.5.0	29/07/2021	Adaptación a la mayoría de los exploradores web. Se añadieron cuadros emergentes de diálogo que nos comunican los procesos realizados.
1.4.0	03/07/2021	Cambio en la forma de ver los datos del perfil personal del administrador, consulta, modificación y actualización.
1.3.0	19/06/2021	Modificación del proceso en el que se va a sancionar a los taxistas por parte de la empresa cuando se requiera.

1.2.0	04/06/2021	Cambios en el método de consulta de taxis y taxistas, pasando de placa y UID generada automáticamente por Firebase, a una consulta a través de placa y cédula respectivamente.
1.1.2	17/05/2021	Se cambiaron las imágenes recortadas que simulaban íconos para nuevos íconos del menú lateral con librería Font Awesome Icon.
1.1.1	10/05/2021	Se adicionó una descripción de los pasos de cada función del CRUD, con la finalidad de entender, de una manera mucho más detallada, el paso a paso.
1.1.0	03/05/2021	Mejora en la vinculación con la base de datos y se establecieron mejores reglas internas en el almacenamiento de Firebase y se corrigieron errores en la misma. Se ajustó el orden en que se solicitaban y guardaban los datos en el sitio web.
1.0.3	26/04/2021	Se cambió la ubicación de botones clave como Cerrar Sesión y Descargar con el fin de hacer mucha más fácil su ubicación.
1.0.2	19/04/2021	Adaptación a pantallas de menor resolución (adaptabilidad).
1.0.1	12/04/2021	Adquisición del dominio Gtaxi.com.co. Se aumentó el tamaño de los formularios, desde login administrativo hasta campos del CRUD.
1.0.0	05/04/2021	Se recibió el software Gtaxi

Fuente: elaboración propia

En cada una de las actualizaciones del producto software, se buscó mejorar en un aspecto o servicio en específico. Con esto, se realizaban feedbacks con la cooperativa donde daban una aprobación de ese contenido particular, además de tracks de acción para esos errores que surgían o procedimientos que pasaban a trabajarse. Las versiones que se fueron produciendo se implementaron a través de la metodología de servicios múltiples o multi-service deployment.

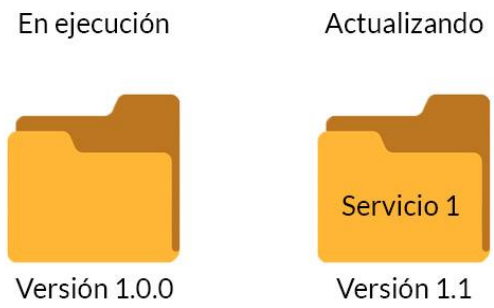


Figura 18. Ejemplo de intervención en un servicio

Fuente: elaboración propia

En cada proceso se llevaron a cabo pruebas internas hasta llegar a una versión mejorada final, donde se le hizo la última inspección de calidad aplicando los parámetros de evaluación a través de la norma ISO/IEC 25000.

Desarrollo del ISO/IEC 25000 en la versión final

Figura 19. Divisiones de la norma ISO/IEC 25000

Fuente: (ISO/IEC, s.f.). La familia de normas ISO/IEC 25000. Recuperado de: https://iso25000.com/index.php/normas-iso-25000

Para garantizar la calidad en el producto software, se desarrolló el siguiente proceso de evaluación que consta de 5 actividades, cada una de ellas desarrolladas, tal como lo explica la norma ISO/IEC 25040, las cuales son:

Figura 20. Proceso de evaluación según ISO/IEC 25040

Fuente: (ISO/IEC, s.f.). ISO/IEC 25040. Recuperado de: https://iso25000.com/index.php/normas-iso-25000/iso-25040

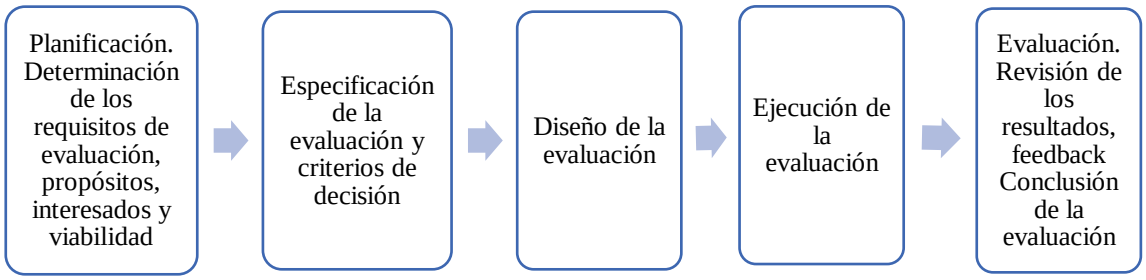


Figura 21. Secuencia del proceso de evaluación

Fuente: elaboración propia. Basada de: ISO/IEC 25040 - https://iso25000.com/index.php/normas-iso-25000/iso-25040.

1. Establecimiento de los requisitos para la evaluación

En este proceso de planificación se establecieron las bases para llevar a cabo la evaluación del producto software, guiados en las instrucciones se realizó una tabulación inicial donde se conoció de primera mano la información dada por la cooperativa.

Tabla 6
Proceso de planificación de la evaluación

Proceso de planificación de evaluación	
Ciudad	Garzón - Huila
Nombre del software	GTAXI
Generalidad del software	
Verificar la solicitud del servicio de asistencia de taxi mediante peticiones desde el teléfono móvil para taxis vinculados a la cooperativa Cootransgar	
Especificación del software	
• Demanda del servicio de taxi desde cualquier ubicación dentro del municipio de Garzón obteniendo su localización en tiempo real • Aceptación o denegación del servicio por parte de usuario o taxista • Historial de carreras realizadas por parte del usuario	
Participantes	
Cargo	Nombre
Estudiante tester app	Cristian Camilo Bravo
Estudiante tester web	William Andrés Otálora
Concejal municipal	José Luis Chaux

Fuente: elaboración propia

Se identificó los elementos necesarios para llevar a cabo las diferentes pruebas en la aplicación móvil y en la página web administrativa, los cuales son: un computador de cualquier sistema operativo, preferiblemente Windows 7 o superior, con acceso a internet.

Dos dispositivos móviles propios con sistema operativo Android 6.0 o superior, que realizan las funciones de taxista y de usuario.

Con estos se llevaron a cabo los testeos en el navegador con respecto al apartado administrativo y en la app con relación al usuario y conductor, pruebas de sistema operativo y compatibilidad de versiones que serán detalladas mucho más adelante.

Entradas: Cronograma de trabajo.

Salidas: Proceso de planificación de evaluación.

2. Especificación de la evaluación

Para especificar la evaluación del software, tomamos como referencia el modelo de calidad del producto definido por la norma ISO/IEC 25010, la cual nos menciona 8 características de calidad y cada una de ellas con unas subcaracterísticas específicas en el área del producto:

Figura 22. Características de calidad según ISO/IEC 25010

Fuente: (ISO/IEC, s.f.). ISO/IEC 25010. Recuperado de: https://iso25000.com/index.php/normas-iso-25000/iso-25010

Llevando a cabo esta especificación de la evaluación y en las características que hay que abordar según la norma, se estableció un nivel de importancia para cada una de las características y subcaracterísticas mencionadas por la ISO, con el fin de dar prioridad a

ciertas peculiaridades que son más necesarias en este proyecto en la evaluación del producto software.

Tabla 7
Simbología del nivel de importancia

Nivel de importancia	**Simbología**	**Porcentaje referencial del nivel de importancia**	**Significado**
Alto	A	68% - 100%	El grado de importancia de la característica o subcaracterística es alto, por ende, se realizará las mediciones
Medio	M	34% - 67%	La característica o subcaracterística no es tan relevante, pero puede o no ser medida dependiendo del criterio del evaluador
Bajo	B	1% - 33%	La característica o subcaracterística no tiene relevancia y no será evaluada o su relevancia es mínima
No aplica	NA	0%	La característica o subcaracterística no puede ser medida por diferentes factores

Fuente: elaboración propia

Con esta definición de cómo será el nivel de importancia, se tomó cada característica y se le asignó su importancia de acuerdo con el proyecto del software, adaptada cada una de ellas y pensadas tanto en la app como en la web.

Tabla 8
Características de calidad en la app

Características	**Descripción**	**Nivel de importancia**
Adecuación funcional	Capacidad para proporcionar funciones que satisfacen las necesidades declaradas en condiciones especificadas	A
Eficiencia de desempeño	Desempeño relativo a la cantidad de recursos utilizados bajo determinadas condiciones	M
Compatibilidad	Capacidad de 2 o más sistemas para intercambiar información y/o llevar funciones cuando comparten el mismo entorno	B
Usabilidad	Capacidad de ser entendido, aprendido, usado y atractivo al usuario bajo determinadas condiciones	A
Fiabilidad	Capacidad para desempeñar funciones, bajo condiciones y periodo de tiempo determinado	M
Seguridad	Protección de información y datos para que sistemas no autorizados no puedan leer o modificar	A
Mantenibilidad	Capacidad de ser modificado eficiente y efectivamente al surgir correcciones o evoluciones	M
Portabilidad	Capacidad de ser transferido de un entorno a otro	M

Fuente: elaboración propia

Tabla 9
Características de calidad en la web

Características	**Descripción**	**Nivel de importancia**
Adecuación funcional	Capacidad para proporcionar funciones que satisfacen las necesidades declaradas en condiciones especificadas	A
Eficiencia de desempeño	Desempeño relativo a la cantidad de recursos utilizados bajo determinadas condiciones	M
Compatibilidad	Capacidad de 2 o más sistemas para intercambiar información y/o llevar funciones cuando comparten el mismo entorno	B
Usabilidad	Capacidad de ser entendido, aprendido, usado y atractivo al usuario bajo determinadas condiciones	M
Fiabilidad	Capacidad para desempeñar funciones, bajo condiciones y periodo de tiempo determinado	M
Seguridad	Protección de información y datos para que sistemas no autorizados no puedan leer o modificar	A
Mantenibilidad	Capacidad de ser modificado eficiente y efectivamente al surgir correcciones o evoluciones	B
Portabilidad	Capacidad de ser transferido de un entorno a otro	B

Fuente: elaboración propia

Así mismo, se apropió cada una de las subcaracterísticas y se les asignó su nivel de importancia para desarrollar cada una de ellas. Las siguientes tablas representan dicha asignación.

Tabla 10
Subcaracterísticas y atributos de calidad en la app

Características	Subcaracterísticas	Nivel de importancia
Adecuación funcional	Completitud funcional	A
	Corrección funcional	A
	Pertinencia funcional	NA
Eficiencia de desempeño	Comportamiento temporal	A
	Utilización de recursos	A
	Capacidad	M
Compatibilidad	Coexistencia	M
	Interoperabilidad	B
Usabilidad	Inteligibilidad	A
	Aprendizaje	A
	Operabilidad	A
	Protección frente a errores de usuario	M
	Estética	A
	Accesibilidad	M
Fiabilidad	Madurez	A
	Disponibilidad	A
	Tolerancia a fallos	M
	Capacidad de recuperación	M
Seguridad	Confidencialidad	A
	Integridad	A
	No repudio	M
	Autenticidad	A
	Responsabilidad	NA
Mantenibilidad	Modularidad	A
	Reusabilidad	NA
	Analizabilidad	M
	Capacidad de ser modificado	M
	Capacidad de ser probado	A
Portabilidad	Adaptabilidad	M
	Facilidad de instalación	A
	Capacidad de ser reemplazado	B

Fuente: elaboración propia

Tabla 11
Subcaracterísticas y atributos de calidad en la web

Características	Subcaracterísticas	Nivel de importancia
Adecuación funcional	Completitud funcional	A
	Corrección funcional	A
	Pertinencia funcional	NA
Eficiencia de desempeño	Comportamiento temporal	A
	Utilización de recursos	M
	Capacidad	B
Compatibilidad	Coexistencia	B
	Interoperabilidad	B
Usabilidad	Inteligibilidad	M
	Aprendizaje	A
	Operabilidad	A
	Protección frente a errores de usuario	M
	Estética	M
	Accesibilidad	B
Fiabilidad	Madurez	M
	Disponibilidad	A
	Tolerancia a fallos	M
	Capacidad de recuperación	M
Seguridad	Confidencialidad	A
	Integridad	A
	No repudio	M
	Autenticidad	M
	Responsabilidad	NA
Mantenibilidad	Modularidad	A
	Reusabilidad	NA
	Analizabilidad	M
	Capacidad de ser modificado	M
	Capacidad de ser probado	M
Portabilidad	Adaptabilidad	M
	Facilidad de instalación	M
	Capacidad de ser reemplazado	NA

Fuente: elaboración propia

3. Diseño de la evaluación

Tomando la referencia de las anteriores tablas, se desarrolló la especificación de las métricas en cada subcaracterísticas las cuales serán la base diseñada para llevar a cabo la posterior ejecución de la evaluación, dando como resultado:

Tabla 12
Métricas para la característica de calidad en la app. Adecuación funcional

Subcaracterística	**Métrica**	**Fase del ciclo de vida de calidad del producto**	**Propósito de la métrica de calidad**	**Método de aplicación**	**Fórmula**	**Valor deseado**	**Tipo de medida**	**Recursos utilizados**
Completitud funcional	Funcionalidades cubiertas	Interna	¿Cuáles funciones se están cumpliendo?	Tomar funciones planeadas y funciones actuales	X = A/B A= función actual B= función planeada	0<=X<=1 Si A>B, error Si A=B, éxito total	Entero por función evaluada	Tester
Corrección funcional	Proveer resultados	Interna	¿Los resultados que arroja son los indicados o esperados?	Resultado obtenido y resultado esperado	¿A=B? A= resultado obtenido B= resultado esperado	Si A=B, caso de éxito Si A!=B, caso de fallo	Booleano 0,1 (No, Si)	Tester

Fuente: elaboración propia

Tabla 13
Métricas para la característica de calidad en la web. Adecuación funcional

Subcaracterística	Métrica	Fase del ciclo de vida de calidad del producto	Propósito de la métrica de calidad	Método de aplicación	Fórmula	Valor deseado	Tipo de medida	Recursos utilizados
Completitud funcional	Funcionalidades cubiertas	Interna	¿Cuáles funciones se están cumpliendo?	Tomar funciones planeadas y funciones actuales	X = A/B A= función actual B= función planeada	0<=X<=1 Si A>B, error Si A=B, éxito total	Entero por función evaluada	Tester
Corrección funcional	Proveer resultados	Interna	¿Los resultados que arroja son los indicados o esperados?	Resultado obtenido y resultado esperado	¿A=B? A= resultado obtenido B= resultado esperado	Si A=B, caso de éxito Si A!=B, caso de fallo	Booleano 0,1 (No, Si)	Tester

Fuente: elaboración propia

Tabla 14
Métricas para la característica de calidad en la app. Eficiencia de desempeño

Subcaracterística	Métrica	Fase del ciclo de vida de calidad del producto	Propósito de la métrica de calidad	Método de aplicación	Fórmula	Valor deseado	Tipo de medida	Recursos utilizados
Comportamiento temporal	Respuestas bajo condiciones dadas	Interna	¿Cuáles funciones se están cumpliendo?	Tomar funciones planeadas y funciones actuales	X = A/B A= función actual B= función planeada	0<=X<=1 Si A>B, error Si A=B, éxito total	Entero por función evaluada	Tester
Utilización de recursos	Cantidad y tipo de recursos usados	Interna	¿Los resultados que arroja son los indicados o esperados?	Resultado obtenido y resultado esperado	¿A=B? A= resultado obtenido B= resultado esperado	Si A=B, caso de éxito Si A!=B, caso de fallo	Booleano 0,1 (No, Si)	Tester
Capacidad	Parámetro cumpliendo con el requisito	Interna	¿El parámetro en su límite cumple con el requisito?	Medir el parámetro en su límite y si el resultado arrojado es igual al parámetro en promedio	A=A’ A= resultado parámetro en límite A’= resultado parámetro en promedio	A=A’, parámetro correcto A!=A’, parámetro incorrecto	Booleano 0,1 (No. Si)	Tester

Fuente: elaboración propia

Tabla 15
Métricas para la característica de calidad en la web. Eficiencia de desempeño

Subcaracterística	Métrica	Fase del ciclo de vida de calidad del producto	Propósito de la métrica de calidad	Método de aplicación	Fórmula	Valor deseado	Tipo de medida	Recursos utilizados
Comportamiento temporal	Respuestas bajo condiciones dadas	Interna	¿Cuáles funciones se están cumpliendo?	Tomar funciones planeadas y funciones actuales	X = A/B A= función actual B= función planeada	0<=X<=1 Si A>B, error Si A=B, éxito total	Entero por función evaluada	Tester
Utilización de recursos	Cantidad y tipo de recursos usados	Interna	¿Los resultados que arroja son los indicados o esperados?	Resultado obtenido y resultado esperado	¿A=B? A= resultado obtenido B= resultado esperado	Si A=B, caso de éxito Si A!=B, caso de fallo	Booleano 0,1 (No, Si)	Tester
Capacidad	Parámetro cumpliendo con el requisito	Interna	¿El parámetro en su límite cumple con el requisito?	Medir el parámetro en su límite y si el resultado arrojado es igual al parámetro en promedio	A=A’ A= resultado parámetro en límite A’= resultado parámetro en promedio	A=A’, parámetro correcto A!=A’, parámetro incorrecto	Booleano 0,1 (No. Si)	Tester

Fuente: elaboración propia

Tabla 16
Métricas para la característica de calidad en la app. Compatibilidad

Subcaracterística	Métrica	Fase del ciclo de vida de calidad del producto	Propósito de la métrica de calidad	Método de aplicación	Fórmula	Valor deseado	Tipo de medida	Recursos utilizados
Coexistencia	Coexiste el software con otro software	Externa	¿Es el software capaz de coexistir con otro, usando mismos recursos?	Prueba del software realizando petición y abrir otro software independiente	X = A/1 A= prueba de coexistencia con otro software 1 = si 0 = no	X=1. Entonces la app si es coexistente X=0. Entonces la app no es coexistente	Booleano 0,1 (No, Si)	Tester
Interoperabilidad	Intercambio de información y uso	Interna	¿Es posible intercambiar información y su uso es disponible?	Lectura y escritura en almacenamiento	X = (∑A/B) + (∑C/D) A=prueba de lectura correcta B=total pruebas de lectura C=prueba de escritura correcta D=total pruebas de escritura	X = 2	Decimal, sumatoria de 2 promedios	Tester

Fuente: elaboración propia

Tabla 17
Métricas para la característica de calidad en la web. Compatibilidad

Subcaracterística	Métrica	Fase del ciclo de vida de calidad del producto	Propósito de la métrica de calidad	Método de aplicación	Fórmula	Valor deseado	Tipo de medida	Recursos utilizados
Coexistencia	Coexiste el software con otro software	Externa	¿Es el software capaz de coexistir con otro, usando mismos recursos?	Prueba del software realizando petición y abrir otro software independiente	X = ∑A/B A= prueba de coexistencia con otro software 1 = si 0 = no B = número total de pruebas	X=1. Entonces la app si es coexistente X=0. Entonces la app no es coexistente	Booleano 0,1 (No, Si)	Tester
Interoperabilidad	Intercambio de información y uso	Interna	¿Es posible intercambiar información y su uso es disponible?	Lectura y escritura en almacenamiento	X = (A==B && A==C) A=prueba de lectura B=lectura correcta C=escritura correcta	X=1. Entonces la app si lee y escribe el intercambio. X!= 1. Entonces no es capaz de intercambiar información	Booleano 0,1 (No, Si)	Tester

Fuente: elaboración propia

Tabla 18
Métricas para la característica de calidad en la app. Usabilidad

Subcaracterística	Métrica	Fase del ciclo de vida de calidad del producto	Propósito de la métrica de calidad	Método de aplicación	Fórmula	Valor deseado	Tipo de medida	Recursos utilizados
Inteligibilidad	Cumplimiento de la necesidad de la cuál fue creado	Externa	¿Cumple con sus requerimientos iniciales?	Calificación de satisfacción del usuario	X = promedio calificación A = sumatoria calificación. B = número de encuestados	X = 5 Cumple con todo. Donde 1 es no cumple y 5 si cumple	Decimal. Promedio	Usuario
Aprendizaje	Entendible su aplicación	Externa	¿Es aprehensible su uso o aplicación?	Calificación de satisfacción del usuario	X = promedio calificación A = sumatoria calificación. B = número de encuestados	X = 5 Cumple con todo. Donde 1 es no cumple y 5 si cumple	Decimal. Promedio	Usuario
Operabilidad	Fácil de operar y controlar	Externa	¿Es fácil su operación?	Calificación de satisfacción del usuario	X = promedio calificación A = sumatoria calificación. B = número de encuestados	X = 5 Cumple con todo. Donde 1 es no cumple y 5 si cumple	Decimal. Promedio	Usuario
Protección frente a errores de usuario	Protege a los usuarios de errores realizables	Externa	¿Evita ser propenso a errores?	Calificación de satisfacción del usuario	X = promedio calificación A = sumatoria calificación. B = número de encuestados	X = 5 Cumple con todo. Donde 1 es no cumple y 5 si cumple	Decimal. Promedio	Tester y usuario
Estética	Satisface y agrada la interacción con el usuario	Externa	¿Es de uso agradable o por el contrario satura?	Calificación de satisfacción del usuario	X = promedio calificación A = sumatoria calificación. B = número de encuestados	X = 5 Cumple con todo. Donde 1 es no cumple y 5 si cumple	Decimal. Promedio	Usuario
Accesibilidad	Incluyente frente a características y discapacidades	Externa	¿Es incluyente a discapacidades?	Calificación de satisfacción del usuario	X = promedio calificación A = sumatoria calificación. B = número de encuestados	X = 5 Cumple con todo. Donde 1 es no cumple y 5 si cumple	Decimal. Promedio	Usuario

Fuente: elaboración propia

Tabla 19
Métricas para la característica de calidad en la web. Usabilidad

Subcaracterística	Métrica	Fase del ciclo de vida de calidad del producto	Propósito de la métrica de calidad	Método de aplicación	Fórmula	Valor deseado	Tipo de medida	Recursos utilizados
Inteligibilidad	Cumplimiento de la necesidad de la cuál fue creado	Externa	¿Cumple con sus requerimientos iniciales?	Calificación de satisfacción del usuario	X = promedio calificación A = sumatoria calificación. B = número de encuestados	X = 5 Cumple con todo. Donde 1 es no cumple y 5 si cumple	Decimal. Promedio	Usuario
Aprendizaje	Entendible su aplicación	Externa	¿Es aprehensible su uso o aplicación?	Calificación de satisfacción del usuario	X = promedio calificación A = sumatoria calificación. B = número de encuestados	X = 5 Cumple con todo. Donde 1 es no cumple y 5 si cumple	Decimal. Promedio	Usuario
Operabilidad	Fácil de operar y controlar	Externa	¿Es fácil su operación?	Calificación de satisfacción del usuario	X = promedio calificación A = sumatoria calificación. B = número de encuestados	X = 5 Cumple con todo. Donde 1 es no cumple y 5 si cumple	Decimal. Promedio	Usuario
Protección frente a errores de usuario	Protege a los usuarios de errores realizables	Externa	¿Evita ser propenso a errores?	Calificación de satisfacción del usuario	X = promedio calificación A = sumatoria calificación. B = número de encuestados	X = 5 Cumple con todo. Donde 1 es no cumple y 5 si cumple	Decimal. Promedio	Tester y usuario
Estética	Satisface y agrada la interacción con el usuario	Externa	¿Es de uso agradable o por el contrario satura?	Calificación de satisfacción del usuario	X = promedio calificación A = sumatoria calificación. B = número de encuestados	X = 5 Cumple con todo. Donde 1 es no cumple y 5 si cumple	Decimal. Promedio	Usuario
Accesibilidad	Incluyente frente a características y discapacidades	Externa	¿Es incluyente a discapacidades?	Calificación de satisfacción del usuario	X = promedio calificación A = sumatoria calificación. B = número de encuestados	X = 5 Cumple con todo. Donde 1 es no cumple y 5 si cumple	Decimal. Promedio	Usuario

Fuente: elaboración propia

Tabla 20
Métricas para la característica de calidad en la app. Fiabilidad

Subcaracterística	Métrica	Fase del ciclo de vida de calidad del producto	Propósito de la métrica de calidad	Método de aplicación	Fórmula	Valor deseado	Tipo de medida	Recursos utilizados
Madurez	Es fiable al desempeñar funciones	Externa	¿Desempeña funciones de manera adecuada?	Tomar funciones ejecutadas y el resultado esperado	X = A/B A= función actual B= función planeada	0<=X<=1 Si A>B, error Si A=B, éxito total	Entero por función evaluada	Tester
Disponibilidad	Operable y accesible cuando se requiere	Externa	¿Se tiene acceso en cualquier momento y es operable?	Calificación de acceso correcto	X = ∑ A / B A=prueba de acceso B=número de pruebas	X = 1 Cumple con todo. Donde 0 es no cumple y 1 si cumple	Decimal. Promedio	Usuario
Tolerancia a fallos	Ejecuta en presencia de fallos	Externa	¿Trabaja según lo previsto en presencia de fallos?	Fallo encontrado y su desarrollo	X = ∑ A / B A=ejecución a pesar del fallo B=fallo encontrado	X = 1 Cumple con todo. Donde 0 es no cumple y 1 si cumple	Decimal. Promedio	Tester y Usuario
Capacidad de recuperación	Recupera datos afectados en caso de interrupción o fallo	Externa	¿Recuperó el dato afectado?	Dato recuperado tras una interrupción o fallo	X = A/1 A= Datos recuperados 1 = si 0 = no	X=1. Entonces la app recupera datos X=0. Entonces la app no recupera datos	Booleano 0,1 (No, Si)	Tester

Fuente: elaboración propia

Tabla 21
Métricas para la característica de calidad en la web. Fiabilidad

Subcaracterística	Métrica	Fase del ciclo de vida de calidad del producto	Propósito de la métrica de calidad	Método de aplicación	Fórmula	Valor deseado	Tipo de medida	Recursos utilizados
Madurez	Es fiable al desempeñar funciones	Externa	¿Desempeña funciones de manera adecuada?	Tomar funciones ejecutadas y el resultado esperado	X = A/B A= función actual B= función planeada	0<=X<=1 Si A>B, error Si A=B, éxito total	Entero por función evaluada	Tester
Disponibilidad	Operable y accesible cuando se requiere	Externa	¿Se tiene acceso en cualquier momento y es operable?	Calificación de acceso correcto	X = ∑ A / B A=prueba de acceso B=número de pruebas	X = 1 Cumple con todo. Donde 0 es no cumple y 1 si cumple	Decimal. Promedio	Usuario
Tolerancia a fallos	Ejecuta en presencia de fallos	Externa	¿Trabaja según lo previsto en presencia de fallos?	Fallo encontrado y su desarrollo	X = ∑ A / B A=ejecución a pesar del fallo B=fallo encontrado	X = 1 Cumple con todo. Donde 0 es no cumple y 1 si cumple	Decimal. Promedio	Tester y Usuario
Capacidad de recuperación	Recupera datos afectados en caso de interrupción o fallo	Externa	¿Recuperó el dato afectado?	Dato recuperado tras una interrupción o fallo	X = A/1 A= Datos recuperados 1 = si 0 = no	X=1. Entonces la app recupera datos X=0. Entonces la app no recupera datos	Booleano 0,1 (No, Si)	Tester

Fuente: elaboración propia

Tabla 22
Métricas para la característica de calidad en la app. Seguridad

Subcaracterística	Métrica	Fase del ciclo de vida de calidad del producto	Propósito de la métrica de calidad	Método de aplicación	Fórmula	Valor deseado	Tipo de medida	Recursos utilizados
Confidencialidad	Protección de datos contra acceso no autorizado	Interna	¿Protege contra accesos no autorizados?	Pruebas de acceso a la cuenta y sus datos	X = ∑ A / B A= acceso denegado B= número de accesos	X=1. Entonces la app protege contra accesos X=0. Entonces la app no protege contra accesos	Decimal. Promedio	Tester y Usuario
Integridad	Protección a modificaciones no autorizadas	Interna	¿Evita modificaciones no autorizadas de datos?	Pruebas de modificación a la cuenta y sus datos	X = ∑ A / B A=prueba de acceso B=número de pruebas	X=1. Entonces la app protege contra cambios X=0. Entonces la app no protege contra cambios	Decimal. Promedio	Tester y Usuario
No repudio	Demostración de acciones que sean repudiadas	Externa	¿Demuestra irrenunciabilidad?	Acción realizada y registro	X = ∑ A/B A=acciones enviadas B=acciones registradas	X=1. Entonces la app registra X=0. Entonces la app no registra	Decimal. Promedio	Tester y Usuario
Autenticidad	Identificación de usuario	Externa	¿Recuperó el dato afectado?	Dato recuperado tras una interrupción o fallo	X = ∑ A/B A=datos recuperados B=datos afectados	X=1. Entonces la app recupera X=0. Entonces la app no recupera	Decimal. Promedio	Tester y Usuario

Fuente: elaboración propia

Tabla 23
Métricas para la característica de calidad en la web. Seguridad

Subcaracterística	Métrica	Fase del ciclo de vida de calidad del producto	Propósito de la métrica de calidad	Método de aplicación	Fórmula	Valor deseado	Tipo de medida	Recursos utilizados
Confidencialidad	Protección de datos contra acceso no autorizado	Interna	¿Protege contra accesos no autorizados?	Pruebas de acceso a la cuenta y sus datos	X = ∑ A / B A= acceso denegado B= número de accesos	X=1. Entonces la app protege contra accesos X=0. Entonces la app no protege contra accesos	Decimal. Promedio	Tester y Usuario
Integridad	Protección a modificaciones no autorizadas	Interna	¿Evita modificaciones no autorizadas de datos?	Pruebas de modificación a la cuenta y sus datos	X = ∑ A / B A=prueba de acceso B=número de pruebas	X=1. Entonces la app protege contra cambios X=0. Entonces la app no protege contra cambios	Decimal. Promedio	Tester y Usuario
No repudio	Demostración de acciones que sean repudiadas	Externa	¿Demuestra irrenunciabilidad?	Acción realizada y registro	X = ∑ A/B A=acciones enviadas B=acciones registradas	X=1. Entonces la app registra X=0. Entonces la app no registra	Decimal. Promedio	Tester y Usuario
Autenticidad	Identificación de usuario	Externa	¿Recuperó el dato afectado?	Dato recuperado tras una interrupción o fallo	X = ∑ A/B A=datos recuperados B=datos afectados	X=1. Entonces la app recupera X=0. Entonces la app no recupera	Decimal. Promedio	Tester y Usuario

Fuente: elaboración propia

Tabla 24
Métricas para la característica de calidad en la app. Mantenibilidad

Subcaracterística	Métrica	Fase del ciclo de vida de calidad del producto	Propósito de la métrica de calidad	Método de aplicación	Fórmula	Valor deseado	Tipo de medida	Recursos utilizados
Modularidad	Mínimo impacto en cambio de un componente	Externa	¿Cuál es el nivel de impacto en el cambio de un componente?	Evaluación de cada cambio y su grado o nivel de impacto 0,1,2	X = ∑ A / B A= nivel de impacto B= número de componentes evaluados	X=2. Entonces los cambios tienen alto impacto X=0. Entonces los cambios tienen poco impacto	Decimal. Promedio	Tester
Analizabilidad	Identificación de partes a modificar	Interna	¿Fácil detección de las partes de la deficiencia a modificar?	Partes por modificar y tiempo de detección	X = ∑ A / B A=tiempo transcurrido B=número de partes a modificar	X tienda a valor mínimo	Decimal. Promedio	Tester
Capacidad de ser modificado	Capacidad de transformación del producto	Interna	¿Es modificable sin degradar el desempeño?	Desempeño anterior y actual del servicio	X = A-B A=Desempeño anterior B=Desempeño actual	X=0. Entonces si es capaz de ser modificada X=1 Entonces no es modificable sin degradar	Entero	Tester y Usuario
Capacidad de ser probado	Facilidad de establecer criterios en el software	Externa	¿Son claros los objetivos de cada servicio para establecer evaluaciones?	Objetivo del servicio y construcción de criterios a evaluar	X = ∑B/A A=número de servicios B=número de criterios a evaluar	Promedio superior a la mediana	Decimal. Promedio	Tester

Fuente: elaboración propia

Tabla 25
Métricas para la característica de calidad en la web. Mantenibilidad

Subcaracterística	Métrica	Fase del ciclo de vida de calidad del producto	Propósito de la métrica de calidad	Método de aplicación	Fórmula	Valor deseado	Tipo de medida	Recursos utilizados
Modularidad	Mínimo impacto en cambio de un componente	Externa	¿Cuál es el nivel de impacto en el cambio de un componente?	Evaluación de cada cambio y su grado o nivel de impacto 0,1,2	X = ∑ A / B A= nivel de impacto B= número de componentes evaluados	X=2. Entonces los cambios tienen alto impacto X=0. Entonces los cambios tienen poco impacto	Decimal. Promedio	Tester
Analizabilidad	Identificación de partes a modificar	Interna	¿Fácil detección de las partes de la deficiencia a modificar?	Partes por modificar y tiempo de detección	X = ∑ A / B A=tiempo transcurrido B=número de partes a modificar	X tienda a valor mínimo	Decimal. Promedio	Tester
Capacidad de ser modificado	Capacidad de transformación del producto	Interna	¿Es modificable sin degradar el desempeño?	Desempeño anterior y actual del servicio	X = A-B A=Desempeño anterior B=Desempeño actual	X=0. Entonces si es capaz de ser modificada X=1 Entonces no es modificable sin degradar	Entero	Tester y Usuario
Capacidad de ser probado	Facilidad de establecer criterios en el software	Externa	¿Son claros los objetivos de cada servicio para establecer evaluaciones?	Objetivo del servicio y construcción de criterios a evaluar	X = ∑B/A A=número de servicios B=número de criterios a evaluar	Promedio superior a la mediana	Decimal. Promedio	Tester

Fuente: elaboración propia

Tabla 26
Métricas para la característica de calidad en la app. Portabilidad

Subcaracterística	Métrica	Fase del ciclo de vida de calidad del producto	Propósito de la métrica de calidad	Método de aplicación	Fórmula	Valor deseado	Tipo de medida	Recursos utilizados
Adaptabilidad	Adaptación a entornos determinados	Externa	¿Es adaptable a entornos determinados de hardware y software?	Prueba en diferentes entornos de hardware y de software	$X = \sum A / B$ A= funcionamiento B= número de pruebas en hardware y software	Alto o total número de entornos aptos para la app	Decimal. Promedio	Tester y Usuario
Facilidad de instalación	Instalación y desinstalación de la app	Externa	¿Es fácil de instalar/ desinstalar?	Pruebas de instalación y desinstalación en diferentes hardware y software	$X = \sum A > \sum B$ $Y = \sum C > \sum D$ A=instalación correcta B=instalación incorrecta C=desinstalación correcta D=desinstalación incorrecta	Mayor número de fácil instalación / desinstalación	Entero	Usuario
Capacidad de ser reemplazado	Realización de tareas similares a otro software	Externa	¿Es potencial para reemplazar otro producto software?	Software con funciones similares y alcance	$X = \sum A > \sum B$ A=Tareas similares B=Tareas diferentes	Mayor número de tareas similares	Entero	Tester y Usuario

Fuente: elaboración propia

Tabla 27
Métricas para la característica de calidad en la web. Portabilidad

Subcaracterística	Métrica	Fase del ciclo de vida de calidad del producto	Propósito de la métrica de calidad	Método de aplicación	Fórmula	Valor deseado	Tipo de medida	Recursos utilizados
Adaptabilidad	Adaptación a entornos determinados	Externa	¿Es adaptable a entornos determinados de hardware y software?	Prueba en diferentes entornos de hardware y de software	X = ∑ A / B A= funcionamiento B= número de pruebas en hardware y software	Alto o total número de entornos aptos para la app	Decimal. Promedio	Tester y Usuario
Facilidad de instalación	Instalación y desinstalación de la app	Externa	¿Es fácil de instalar/ desinstalar?	Pruebas de instalación y desinstalación en diferentes hardware y software	X = ∑ A > ∑ B Y = ∑ C > ∑ D A=instalación correcta B=instalación incorrecta C=desinstalación correcta D=desinstalación incorrecta	Mayor número de fácil instalación / desinstalación	Entero	Usuario
Capacidad de ser reemplazado	Realización de tareas similares a otro software	Externa	¿Es potencial para reemplazar otro producto software?	Software con funciones similares y alcance	X = ∑ A > ∑ B A=Tareas similares B=Tareas diferentes	Mayor número de tareas similares	Entero	Tester y Usuario

Fuente: elaboración propia

4. Ejecución de la evaluación

En este apartado se llevó a cabo las pruebas con el fin de cumplir, en mayor porcentaje posible, las métricas especificadas y diseñadas anteriormente. A continuación, se plasmó la ponderación en porcentajes para la calidad para cada una de las características donde la sumatoria de todo es del 100% y la matriz de la evaluación de calidad con los resultados arrojados en la ejecución.

Tabla 28
Ponderación de la calidad en la app

Características	Nivel de importancia	Ponderación
Adecuación funcional	A	18%
Eficiencia de desempeño	M	10%
Compatibilidad	B	6%
Usabilidad	A	18%
Fiabilidad	M	10%
Seguridad	A	18%
Mantenibilidad	M	10%
Portabilidad	M	10%

Fuente: elaboración propia

Tabla 29
Ponderación de la calidad en la web

Características	Nivel de importancia	Ponderación
Adecuación funcional	A	20%
Eficiencia de desempeño	M	12%
Compatibilidad	B	8%
Usabilidad	M	12%
Fiabilidad	M	12%
Seguridad	A	20%
Mantenibilidad	B	8%
Portabilidad	B	8%

Fuente: elaboración propia

Tabla 30
Matriz para evaluar la calidad en la app del producto software

Característica	Subcaracterística	Métrica	Fórmula	Valor deseado (umbral)	Valor obtenido (x)	Valor parcial total	Nivel de importancia	Valor final	Calidad del sistema
Adecuación funcional	Completitud funcional	Funcionalidades cubiertas	X = A/B A= función actual B= función planeada	1	A=9 B=10	$\frac{9}{10}$	A (50%)	8.1%	16.2% de 18%
	Corrección funcional	Proveer resultados	¿A=B? A= resultado obtenido B= resultado esperado	1=1	Registro 1=1 Inicio sesión 1=1 Cerrar sesión 1=1 Solicitar taxi 1=1 Asignar taxi 1=1 Recibir petición 1=1 Buscar otro taxi 0!=1 Marcar trayecto 1=1 Calificación 1=1 Historial 1=1	$\frac{9}{10}$	A (50%)	8.1%	
Eficiencia de desempeño	Comportamiento temporal	Respuestas bajo condiciones dadas	X = ∑ A/ ∑ B A= función actual B= función planeada	1	A=9 B=10	$\frac{9}{10}$	A (50%)	4.5%	9.12% De 10%
	Utilización de recursos	Cantidad y tipo de recursos usados	¿A=B? A= resultado obtenido B= resultado esperado	1	Solicitar taxi 1=1 Asignar taxi 1=1 Recibir petición 1=1 Buscar otro taxi 1=1 Marcar trayecto 1=1	1=1	M (35%)	3.5%	
	Capacidad	Parámetro cumpliendo con el requisito	X = ∑ A / A' A=A' A= resultado parámetro en límite A'= resultado esperado	1=1	A=3 A'=4	$\frac{3}{4}$	B (15%)	1.12%	
Compatibilidad	Coexistencia	Coexiste el software con otro software	X = A/B A= prueba de coexistencia con otro software 1 = si 0 = no B = número de software probado en coexistencia	1	A=4 B=5	$\frac{4}{5}$	M (66%)	3.168 %	5.2% De 6%
	Interoperabilidad	Intercambio de información y uso	X = (A==B && A==C) A=prueba de lectura B=lectura correcta C=prueba de escritura D=escritura correcta	1==1	A=8 B=8	$\frac{8}{8}$	B (34%)	2.04%	

Usabilidad	Inteligibilidad	Cumplimiento de la necesidad de la cuál fue creado	X = ∑ A / B A = calificación B = número de encuestados	5	A = 72 B = 15	$\frac{72}{15} = 4.8$	A	3.456 %	16.92% de 18%
	Aprendizaje	Entendible su aplicación	X = ∑ A / B A = calificación B = número de encuestados	5	A = 69 B = 15	$\frac{69}{15} = 4.6$	A	3.312 %	
	Operabilidad	Fácil de operar y controlar	X = ∑ A / B A = calificación B = número de encuestados	5	A = 72 B = 15	$\frac{72}{15} = 4.8$	A	3.456 %	
	Protección frente a errores de usuario	Protege a los usuarios de errores realizables	X = ∑ A / B A = calificación. B = número de encuestados	5	A = 69 B = 15	$\frac{69}{15} = 4.6$	M	1.656 %	
	Estética	Satisface y agrada la interacción con el usuario	X = ∑ A / B A = calificación. B = número de encuestados	5	A = 72 B = 15	$\frac{72}{15} = 4.8$	A	3.456 %	
	Accesibilidad	Incluyente frente a características y discapacidades	X = ∑ A / B A = calificación. B = número de encuestados	5	A = 66 B = 15	$\frac{66}{15} = 4.4$	M	1.584 %	
Fiabilidad	Madurez	Es fiable al desempeñar funciones	X =∑ A/B A= función ejecutada B= función esperada	1	A = 28 B = 30	$\frac{28}{30}$	A	3.173 %	8.973% De 10%
	Disponibilidad	Operable y accesible cuando se requiere	X = ∑ A / B A=prueba de acceso B=número de pruebas	1	A = 6 B = 6	$\frac{6}{6}$	A	3.4%	
	Tolerancia a fallos	Ejecuta en presencia de fallos	X = ∑ A / ∑ B A=ejecución a pesar del fallo B=fallo encontrado	1	A = 6 B = 6	$\frac{6}{6}$	M	1.6%	
	Capacidad de recuperación	Recupera datos afectados en caso de interrupción o fallo	X = ∑ A / B A= Datos recuperados B= número de intentos	1	A = 2 B = 4	$\frac{2}{4}$	M	0.8%	
Seguridad	Confidencialidad	Protección de datos contra acceso no autorizado	X = ∑ A / B A= acceso denegado B= número de intentos de accesos	1	A = 12 B = 12	$\frac{12}{12}$	A	5.2%	18% de 18%
	Integridad	Protección a modificaciones no autorizadas	X = ∑ A / B A=prueba de acceso B=número de pruebas	1	A = 12 B = 12	$\frac{12}{12}$	A	5.2%	

	No repudio	Demostración de acciones que sean repudiadas	$X = \sum A/B$ A=acciones enviadas B=acciones registradas	1	A = 5 B = 5	$\frac{5}{5}$	M	2.4%	
	Autenticidad	Identificación de usuario	$X = \sum A/B$ A=usuarios con datos recuperados B=número de usuarios testeados	1	A = 10 B = 10	$\frac{10}{10}$	A	5.2%	
Mantenibilidad	Modularidad	Mínimo impacto en cambio de un componente	$X = \sum A / B$ A= nivel de impacto B= número de componentes evaluados	5	A = 30 B = 10	$\frac{30}{10}$	A	2.04%	9.696% de 10%
	Analizabilidad	Identificación de partes a modificar	$X = \sum A / B$ A=calificación de facilidad para diagnosticar código B=número de partes a modificar	5	A = 20 B = 5	$\frac{20}{5}$	M	2.72%	
	Capacidad de ser modificado	Capacidad de transformación del producto	$X = \sum A / B$ A=Facilidad de modificación de versión nueva B=número de versiones tomadas	5	A = 24 B = 5	$\frac{24}{5}$	M	1.536 %	
	Capacidad de ser probado	Facilidad de establecer criterios en el software	$X = \sum A / B$ A=facilidad de pruebas en el servicio B=número de servicios	5	A = 25 B = 5	$\frac{25}{5}$	A	3.4%	
Portabilidad	Adaptabilidad	Adaptación a entornos determinados	$X = \sum A / B$ A=funcionamiento B= número de pruebas en hardware y software	5	A = 49 B = 10	$\frac{49}{10}$	M	3.43%	
	Facilidad de instalación	Instalación y desinstalación de la app	$X = \sum A > \sum B$ $Y = \sum C > \sum D$ A=instalación correcta B=instalación incorrecta C=desinstalación correcta D=desinstalación incorrecta	10	10>0 10>0	10	A	5%	9.63% de 10%
	Capacidad de ser reemplazado	Realización de tareas similares a otro software	$X = \sum A / B$ A=Tareas similares B= número de tareas evaluadas	1	A = 8 B = 10	$\frac{8}{10}$	B	1.2%	

Fuente: elaboración propia

Tabla 31
Matriz para evaluar la calidad en la web del producto software

Característica	Subcaracterística	Métrica	Fórmula	Valor deseado (umbral)	Valor obtenido (x)	Valor parcial total	Nivel de importancia	Valor final	Calidad del sistema
Adecuación funcional	Completitud funcional	Funcionalidades cubiertas	X = A/B A= función actual B= función planeada	1	A=11 B=11	$\frac{11}{11}$	A (50%)	10% de 10%	20% de 20%
	Corrección funcional	Proveer resultados	¿A=B? A= resultado obtenido B= resultado esperado	1=1	Inicio sesión 1=1 Cerrar sesión 1=1 Datos perfil 1=1 Agregar taxi nuevo 1=1 Agregar taxista nuevo 1=1 Buscar taxi existente 1=1 Buscar taxista existente 1=1 Sancionar taxista 1=1 Eliminar taxista 1=1 Descargar app taxista 1=1 Descargar app usuario 1=1	$\frac{11}{11}$	A (50%)	10% de 10%	
Eficiencia de desempeño	Comportamiento temporal	Respuestas bajo condiciones dadas (Cond: 3 equipos en la web. Cond: en acceso móvil. Cond: diferente navegador)	X = ∑ A/ ∑ B A= función actual B= función planeada	1	A=33 B=33	$\frac{33}{33}$	A (50%)	6% de 6%	11.55% de 12%
	Utilización de recursos	Cantidad y tipo de recursos usados (Cond: solo Wi-Fi. Cond: por cable)	¿A=B? A= resultado obtenido B= resultado esperado	1	Inicio sesión 1=1 Cerrar sesión 1=1 Datos perfil 1=1 Agregar taxi nuevo 1=1 Agregar taxista nuevo 1=1 Buscar taxi existente 1=1 Buscar taxista existente 1=1 Sancionar taxista 1=1 Eliminar taxista 1=1 Descargar app taxista 1=1 Descargar app usuario 1=1	$\frac{22}{22}$	M (35%)	4.2% de 4.2%	
	Capacidad	Parámetro en su límite cumpliendo con el requisito (Cond: conexión)	X = ∑ A / A' A=A' A= resultado parámetro en límite A'= resultado esperado	1=1	A=3 A'=4	$\frac{3}{4}$	B (15%)	1.35% de 1.8%	
Compatibilidad	Coexistencia	Coexiste el software con otro software	X = A/B	1	A=5 B=5	$\frac{5}{5}$	B (50%)	4% de 4%	7.92% de 8%

			A= prueba de coexistencia con otro software 1 = si 0 = no B = número de software probado en coexistencia		X = ∑A/B				
	Interoperabilidad	Intercambio de información y uso	X = ∑A/B A=prueba de lectura B=lectura correcta C=prueba de escritura D=escritura correcta	100	X = (∑A/B) + (∑C/D) A=50 B=50 C=48 D=50	1 + 0.96 = 1.96	B (50%)	3.92% de 4%	
Usabilidad	Inteligibilidad	Cumplimiento de la necesidad de la cuál fue creado	X = ∑ A / B A = calificación B = número de encuestados	5	A = 14 B = 3	$\frac{14}{3} = 4.66$	M	1.4% de 1.5%	11.46% de 12%
	Aprendizaje	Entendible su aplicación	X = ∑ A / B A = calificación B = número de encuestados	5	A = 15 B = 3	$\frac{15}{3} = 5.0$	A	2.5% de 2.5%	
	Operabilidad	Fácil de operar y controlar	X = ∑ A / B A = calificación B = número de encuestados	5	A = 15 B = 3	$\frac{15}{3} = 5.0$	A	2.5% de 2.5%	
	Protección frente a errores de usuario	Protege a los usuarios de errores realizables	X = ∑ A / B A = calificación. B = número de encuestados	5	A = 14 B = 3	$\frac{14}{3} = 4.66$	M	1.4% de 1.5%	
	Estética	Satisface y agrada la interacción con el usuario	X = ∑ A / B A = calificación. B = número de encuestados	5	A = 15 B = 3	$\frac{15}{3} = 5.0$	M	1.5% de 1.5%	
	Accesibilidad	Incluyente frente a características y discapacidades	X = ∑ A / B A = calificación. B = número de encuestados	5	A = 13 B = 3	$\frac{13}{3} = 4.33$	A	2.16% de 2.5%	
Fiabilidad	Madurez	Es fiable al desempeñar funciones	X =∑ A/B A= función ejecutada B= función esperada	1	A = 11 B = 11	$\frac{11}{11}$	M	2.5% de 2.5%	10.63% de 12%
	Disponibilidad	Operable y accesible cuando se requiere	X = ∑ A / B A=prueba de acceso B=número de pruebas	1	A = 11 B = 11	$\frac{11}{11}$	A	4.5% de 4.5%	
	Tolerancia a fallos	Ejecuta en presencia de fallos	X = ∑ A / B A=ejecución a pesar del fallo B=total ejecuciones	1	A = 9 B = 11	$\frac{9}{11}$	M	2.045 % de 2.5%	

	Capacidad de recuperación	Recupera datos afectados en caso de interrupción o fallo	X = ∑ A / B A= Datos recuperados B= número de intentos	1	A = 7 B = 11	$\frac{7}{11}$	M	1.59% de 2.5%	
Seguridad	Confidencialidad	Protección de datos contra acceso no autorizado	X = ∑ A / B A= acceso denegado B= número de intentos de accesos	1	A = 12 B = 12	$\frac{12}{12}$	A	6.7% de 6.7%	20% de 20%
	Integridad	Protección a modificaciones no autorizadas	X = ∑ A / B A=prueba de acceso B=número de pruebas	1	A = 12 B = 12	$\frac{12}{12}$	A	6.7% de 6.7%	
	No repudio	Demostración de acciones que sean repudiadas	X = ∑ A/B A=acciones enviadas B=acciones registradas	1	A = 10 B = 10	$\frac{10}{10}$	M	3.3% de 3.3%	
	Autenticidad	Identificación de usuario	X = ∑ A/B A=equipos con datos recuperados B=número en equipos testeados	1	A = 4 B = 4	$\frac{4}{4}$	M	3.3% de 3.3%	
Mantenibilidad	Modularidad	Mínimo impacto en cambio de un componente	X = ∑ A / B A= nivel de impacto B= número de componentes evaluados	5	A = 53 B = 11	$\frac{53}{11}$	A	3.08% de 3.2%	7.88% de 8%
	Analizabilidad	Identificación de partes a modificar	X = ∑ A / B A=calificación de facilidad para diagnosticar código B=número de partes a modificar	5	A = 40 B = 8	$\frac{40}{8}$	M	1.6% de 1.6%	
	Capacidad de ser modificado	Capacidad de transformación del producto	X = ∑ A / B A=Facilidad de modificación de versión nueva B=número de versiones tomadas	5	A = 25 B = 5	$\frac{25}{5}$	M	1.6% de 1.6%	
	Capacidad de ser probado	Facilidad de establecer criterios en el software	X = ∑ A / B A=facilidad de pruebas en el servicio B=número de servicios	5	A = 55 B = 11	$\frac{55}{11}$	M	1.6% de 1.6%	
Portabilidad	Adaptabilidad	Adaptación a entornos determinados	X = ∑ A / B A=funcionamiento B= número de pruebas en hardware y software	5	A = B = 15	$\frac{73}{15}$	M	3.89% de 4%	7.89% de 8%
	Facilidad de instalación	Instalación y desinstalación de la app (se tomó como facilidad de acceso)	X = ∑ A > ∑ B Y = ∑ C > ∑ D A=instalación correcta B=instalación incorrecta C=desinstalación correcta D=desinstalación incorrecta	10	10>0 10>0	10	M	4% de 4%	

Fuente: elaboración propia

5. Conclusión de la evaluación.

Con la obtención de los resultados, se clasificaron las mediciones para determinar el grado de satisfacción obtenido de la evaluación realizada en cada aspecto de la métrica propuesta.

Tabla 32
Puntuación final para la calidad de evaluación

Escala de medición	Niveles de puntuación	Grado de satisfacción
8.75 - 10	Cumple con los requisitos	Muy satisfactorio
5 – 8.74	Aceptable	Satisfactorio
2.75 – 4.9	Mínimamente aceptable	Insatisfactorio
0 – 2.74	Inaceptable	

Fuente: elaboración propia

Tabla 33
Resultado final de la evaluación en la app

Resultado final de la evaluación en la app			
Calidad	Calidad del sistema	Nivel de puntuación	Grado de satisfacción
Total	9.37 (93%)	Cumple con los requisitos	Muy satisfactorio

Fuente: elaboración propia

Tabla 34
Resultado final de la evaluación en la web

Resultado final de la evaluación en la web			
Calidad	Calidad del sistema	Nivel de puntuación	Grado de satisfacción
Total	9.7 (97.3%)	Cumple con los requisitos	Muy satisfactorio

Fuente: elaboración propia

Con estos resultados obtenidos en la evaluación de las métricas en las características y subcaracterísticas de la calidad del producto software, se observó una mejora considerablemente amplia en comparación al software inicial recibido. Con ello se

consideró que esta versión del software en este estado finalizado está lista para su despliegue final y es un producto viable para el público.

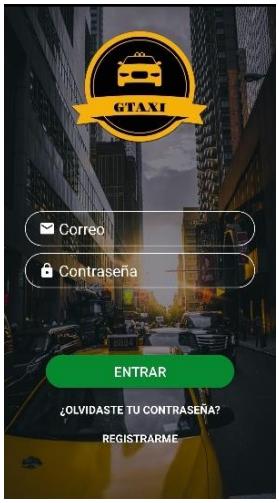
A continuación, se adjuntan algunos soportes de trabajos específicos en el área de la calidad donde se utilizaron herramientas adicionales para el testeo.

Pruebas no funcionales

Pruebas de compatibilidad. Por parte del aplicativo, se utilizaron los dispositivos Android de cada uno y emuladores del propio IDE Android Studio, entorno integrado que fue utilizado para el desarrollo y para las correcciones.

Dentro del diseño de las pruebas se definió la carga inicial de la app, inicio de sesión con autenticación, conexión a la base de datos, registro de datos. Los resultados nos arrojaron y ratificaron el funcionamiento del software en sistemas operativos Android con versiones igual o superior a la 6.0. Sin embargo, se observaron mejorías cuando el S.O.es de las últimas versiones lanzadas por Android.

(23)



(24)

Figura 23. Prueba de compatibilidad Android 10 con Huawei honor 8x

Figura 24. Prueba de compatibilidad en Android 9 con Pixel 3

Fuente: Tomada en dispositivo propio

Se realizó la simulación a través de Firebase y dentro de los dispositivos disponibles se arrojaron los resultados de compatibilidad con versiones Android superiores a la 6.0 Marshmallow, los cuales fueron exitosos y todos cumplieron con la ejecución completa de la aplicación.

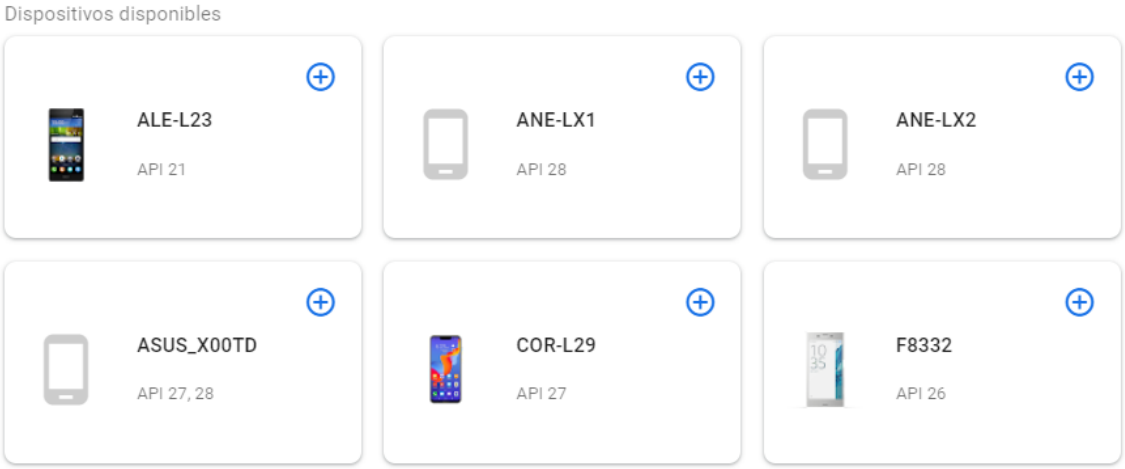


Figura 25. Pruebas de compatibilidad con simulación de dispositivos virtuales

Fuente: prueba elaborada propiamente. Realizada en: https://console.firebase.google.com/

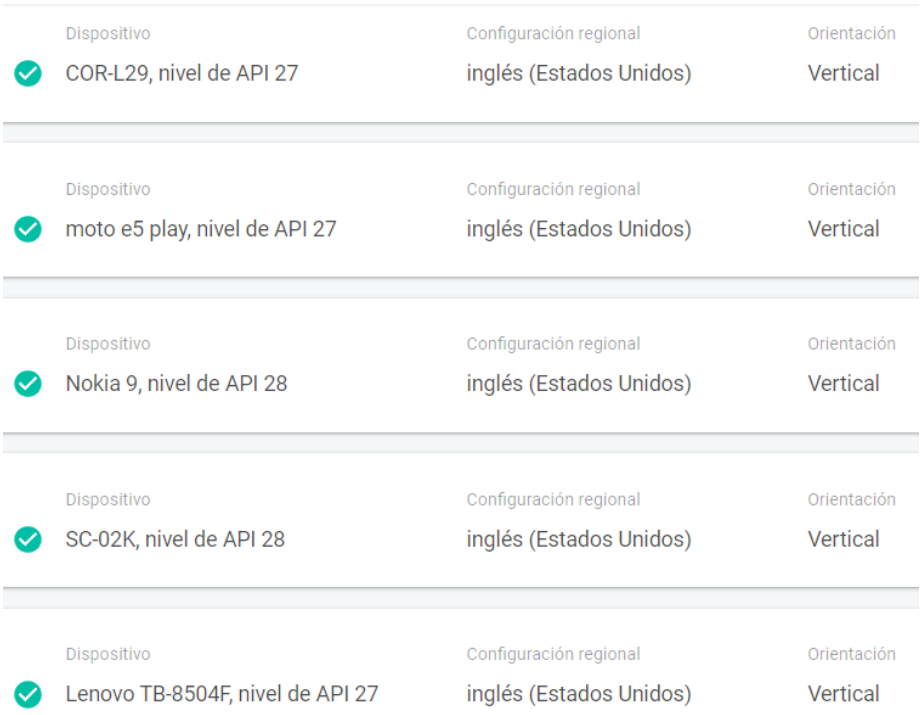


Figura 26. Pruebas de compatibilidad con otros dispositivos simulados

Fuente: elaboración propia. Resultados de: https://console.firebase.google.com/

Figura 27. Resultados de la prueba de compatibilidad

Fuente: elaboración propia. Resultados de: https://console.firebase.google.com/

Por otro lado, para este tipo de pruebas en la página administrativa, se realizó la emulación del entorno de cualquier navegador web con su versión, para testear el funcionamiento del software en los diferentes entornos. Para ello se utilizó la herramienta web Browserling que, de

manera gratuita, nos permite efectuar estos entornos virtuales simulando los buscadores o navegadores web. Seguidamente podemos ver las capturas de la simulación:

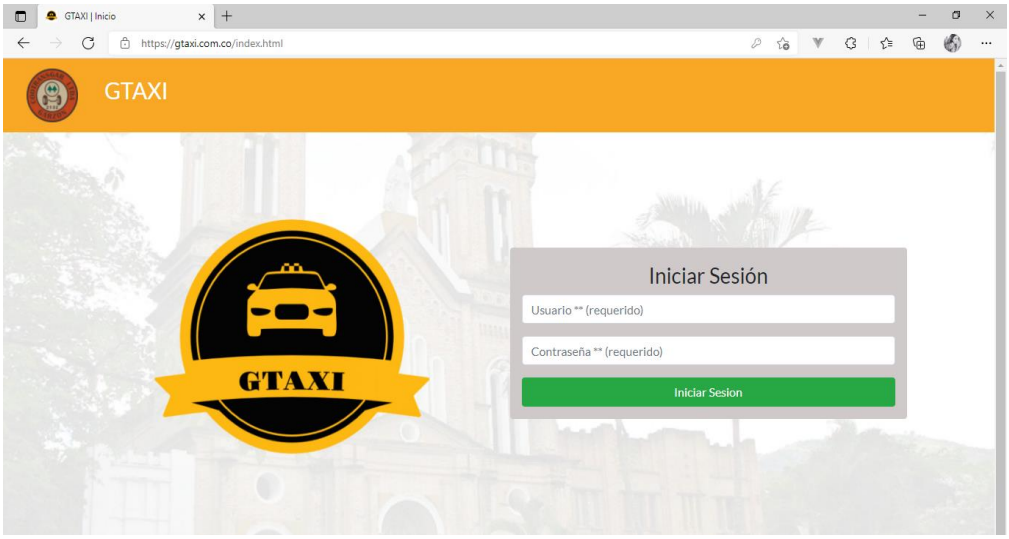


Figura 28. Prueba de compatibilidad web con Microsoft Edge

Fuente: elaboración propia. Realizada en browserling.com

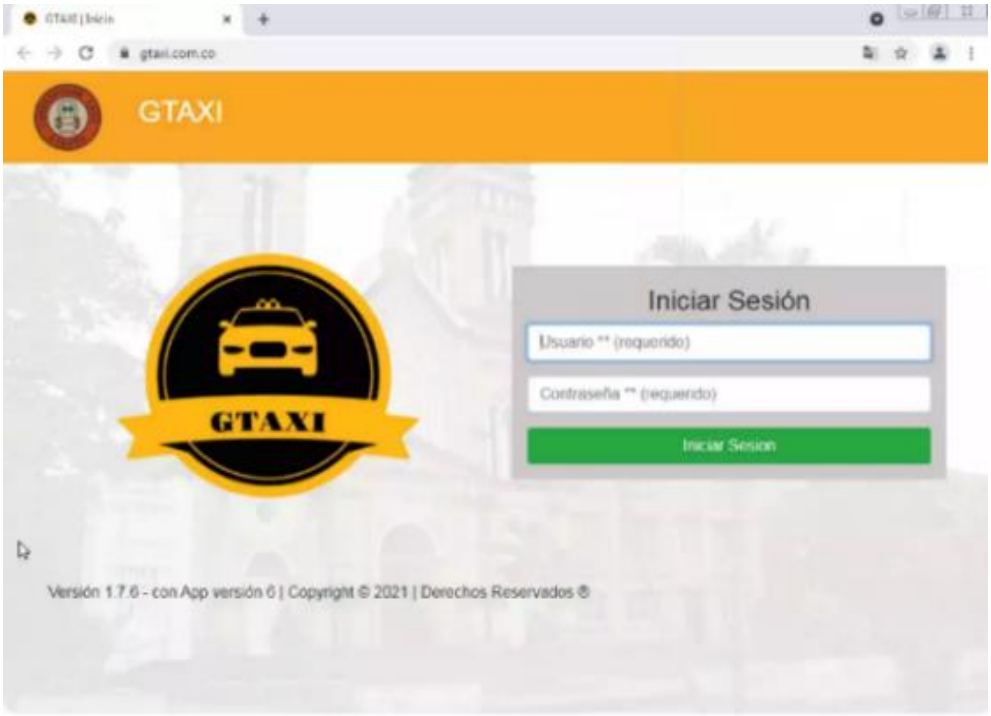


Figura 29. Prueba de compatibilidad web con Google Chrome

Fuente: elaboración propia. Realizada en browserling.com

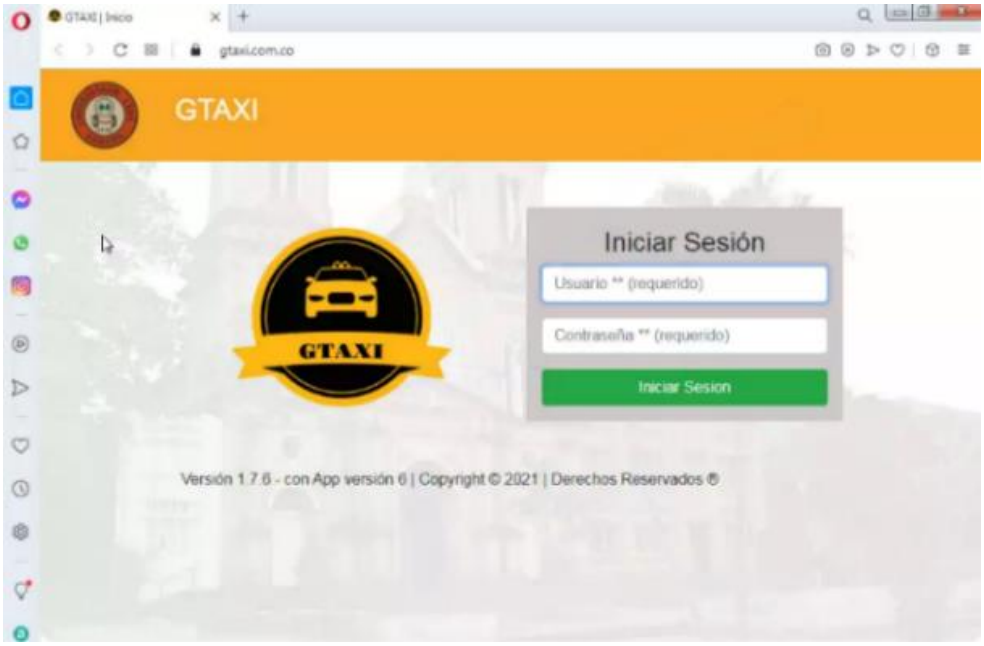


Figura 30. Prueba de compatibilidad web con Opera

Fuente: elaboración propia. Realizada en browserling.com

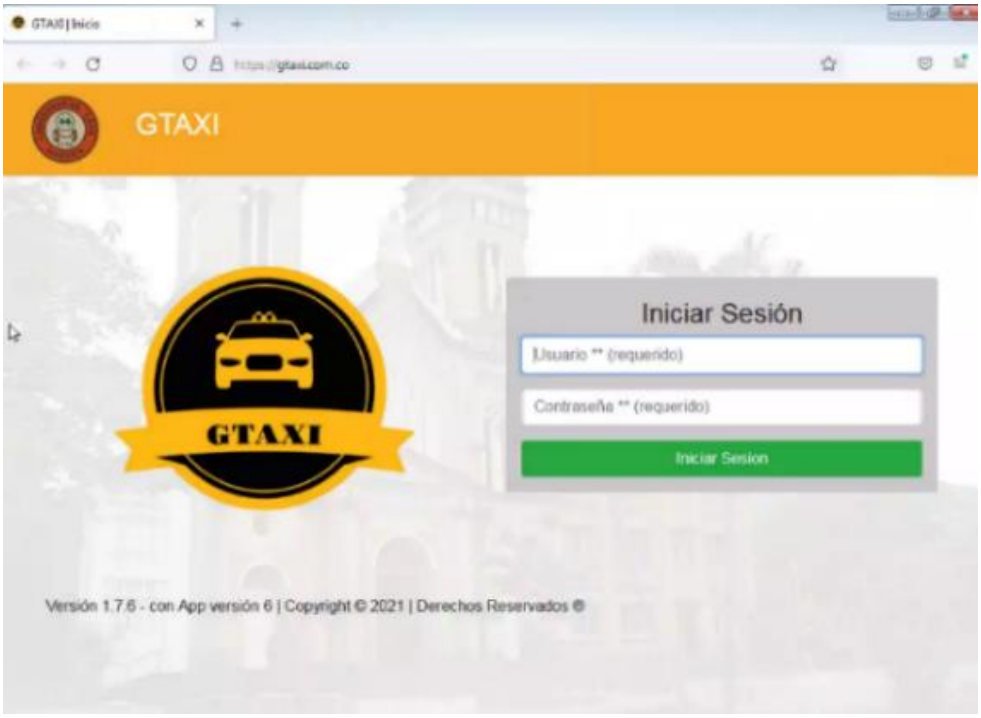


Figura 31. Prueba de compatibilidad web con Mozilla Firefox

Fuente: elaboración propia. Realizada en browserling.com

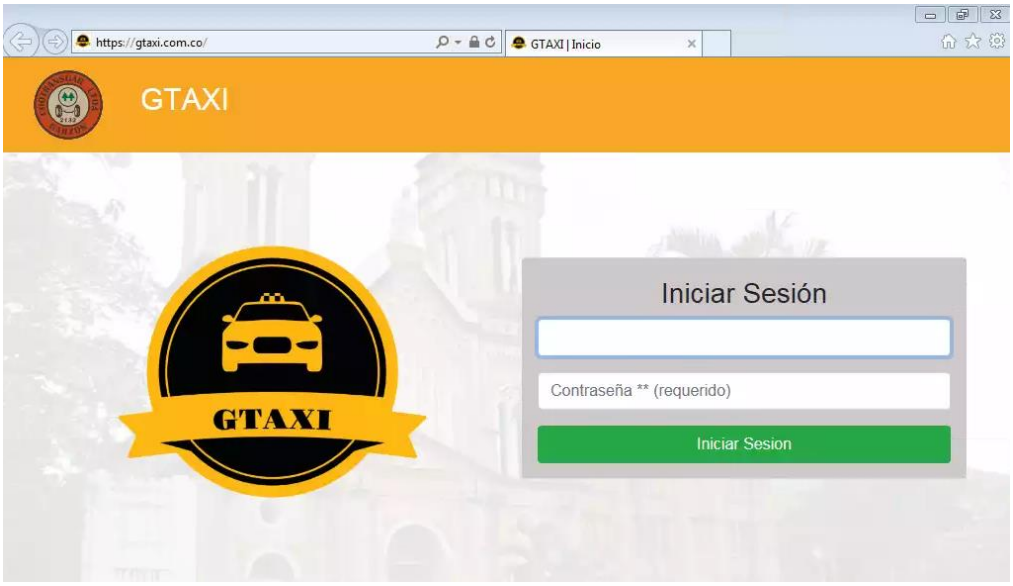


Figura 32. Prueba de compatibilidad web con Internet Explorer

Fuente: elaboración propia. Realizada en browserling.com

En todos los navegadores que se probaron se cargaba correctamente, inicio de sesión con autenticación y una conexión estable.

Finalmente, se realizó una reunión donde se socializaron estos resultados y las no conformidades. En ello se especificó que el software funciona correctamente en los distintos navegadores y los administrativos encargados de estos procesos internos quedaron satisfechos con lo que se tiene montado en la web para sus tareas en GTAXI. En forma de resumen se realizó una tabla de chequeo de la compatibilidad de la aplicación web con los diferentes navegadores probados. También se vio conformidad con el aplicativo móvil que se vio mejorado en el transcurso de las versiones.

Tabla 35
Prueba web de compatibilidad con navegadores

Prueba	Google Chrome	Mozilla Firefox	Opera	Microsoft Edge
1	(v.92) Si	(v.90) Si	(v.77) Si	(v. 93) Si
2	(v.90) Si	(v.88) Si	(v.75) Si	
3	(v.85) Si	(v.85) Si	(v.70) Si	

Fuente: elaboración propia

Pruebas de estrés y de carga. Por parte de la aplicación móvil, tanto usuario como taxista, se presentaron inconvenientes por las sesiones abiertas de varios usuarios donde las acciones afectaban tanto a uno como a otro, es por ello que se debió realizar unos cambios respectivos donde se mejoró la concordancia y concurrencia de usuarios y taxistas dentro del aplicativo, con la finalidad de que varios de estos en simultáneo realicen las peticiones y automáticamente las asignaciones.

Las siguientes gráficas nos muestran los resultados obtenidos con las nuevas actualizaciones mejoradas del consumo de CPU y de memoria en el aplicativo en un lapso de tiempo limitado, simulando el funcionamiento a traves de Firebase.

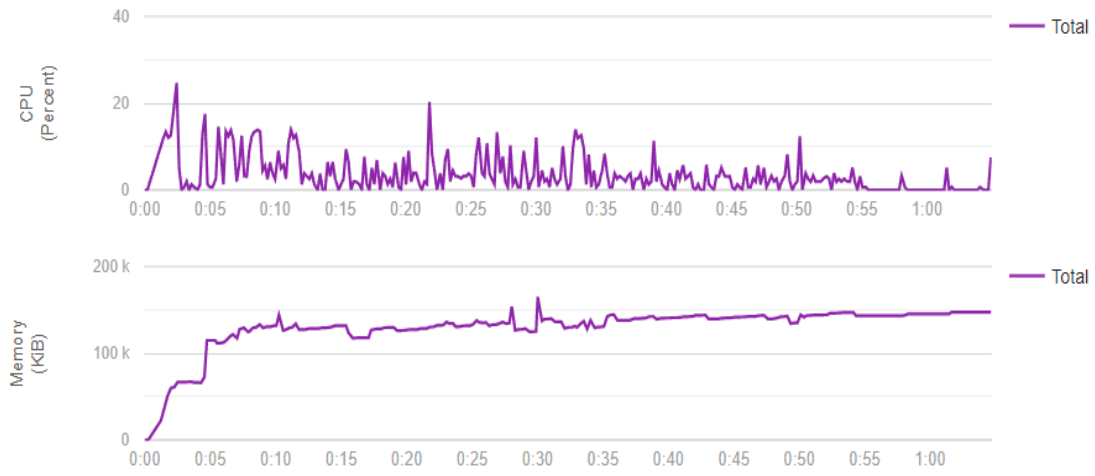


Figura 33. Prueba de estrés en Android 9 con Pixel 3

Fuente: elaboración propia. Realizada en https://console.firebase.google.com/

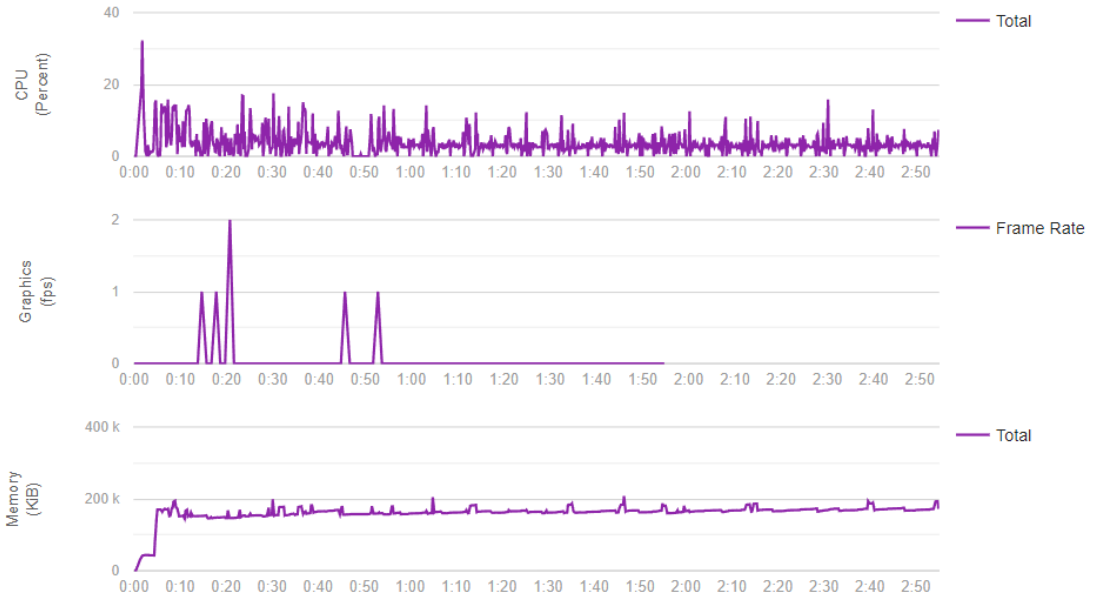


Figura 34. Prueba de estrés en Android 7 con Sony Xperia XZ2 Compact H8314

Fuente: elaboración propia. Realizada en https://console.firebase.google.com/

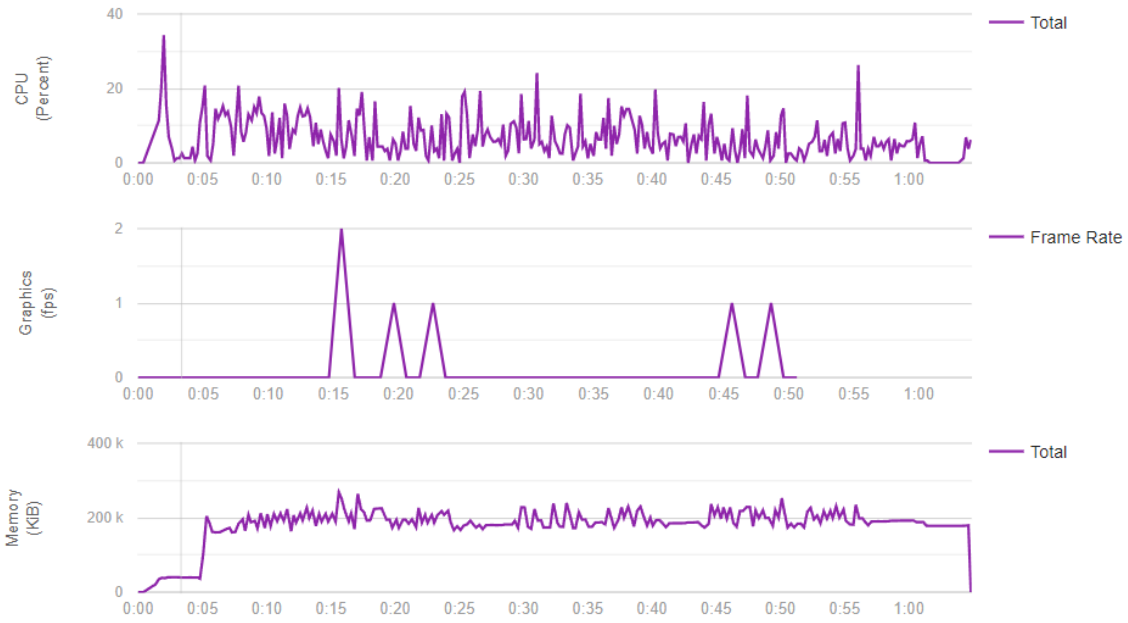


Figura 35. Prueba de estrés en Android 8 con Honor play dual sim COR-L29

Fuente: elaboración propia. Realizada en https://console.firebase.google.com/

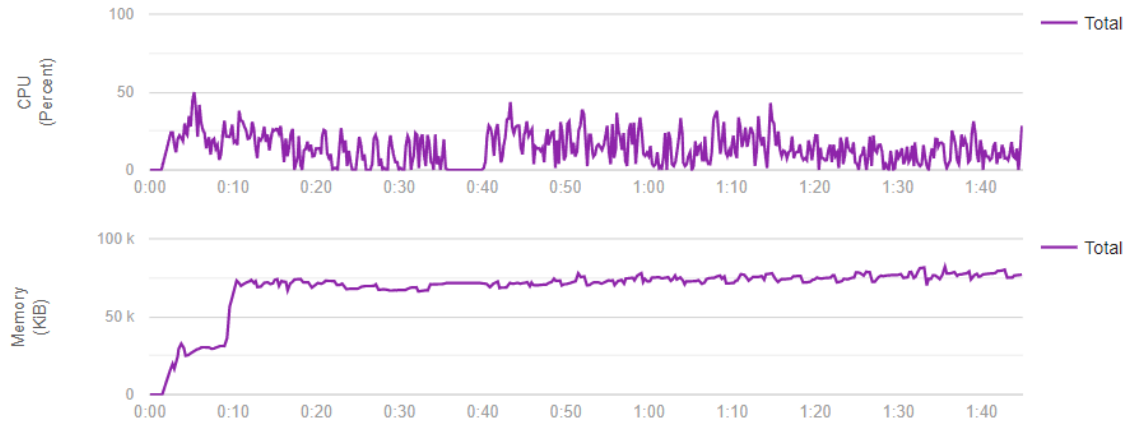


Figura 36. Prueba de estrés en Android 8 con Motorola moto e5 play

Fuente: elaboración propia. Realizada en https://console.firebase.google.com/

Sumado a estas pruebas realizadas, también se llevaron a cabo unas pruebas dentro del IDE de Android Studio, donde fue creada la app. Para esto, se tuvo en cuenta el rendimiento de la

CPU, memoria, network y energía. A continuación, se observan los resultados de la evaluación realizada en un dispositivo Android 11.

Rendimiento de CPU con funcionamiento del 100%: en descanso (aplicación abierta): 30% CPU. En reposo (aplicación reposo): 10% CPU.

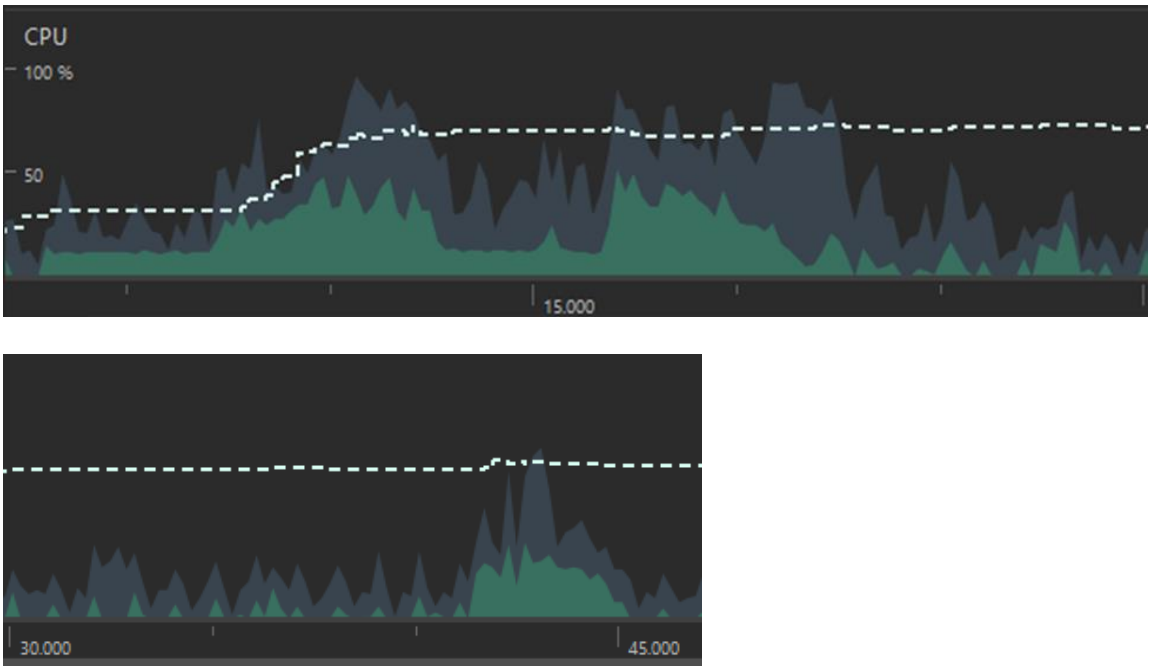


Figura 37. Prueba de estrés, carga y consumo de CPU en porcentajes

Fuente: elaboración propia. Realizada en Android Studio

Rendimiento de la memoria.

Al iniciar la aplicación: 60 – 190 MB. Al paso de un tiempo de iniciar la aplicación (30 segundos): 130MB.

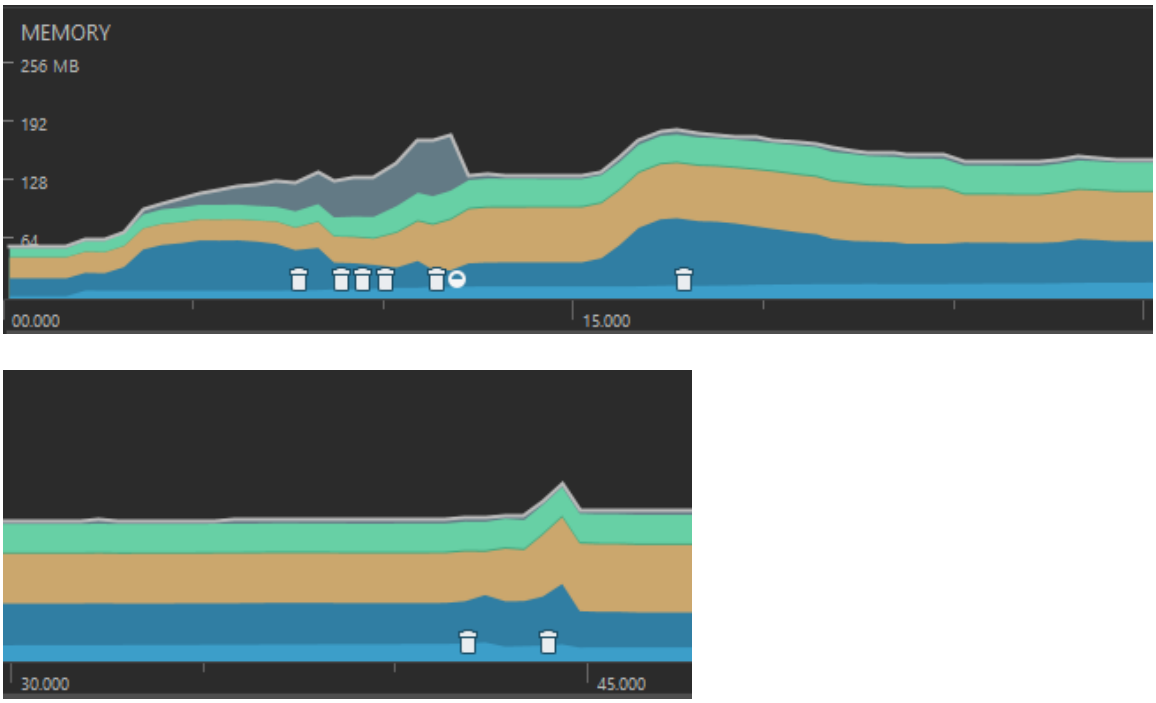


Figura 38. Prueba de estrés, carga y consumo de memoria

Fuente: elaboración propia. Realizada en Android Studio

Rendimiento network en funcionamiento: 20KB/s

En Descanso (aplicación abierta): 8 KB/s. En Reposo (aplicación minimizada): 8 KB/s.

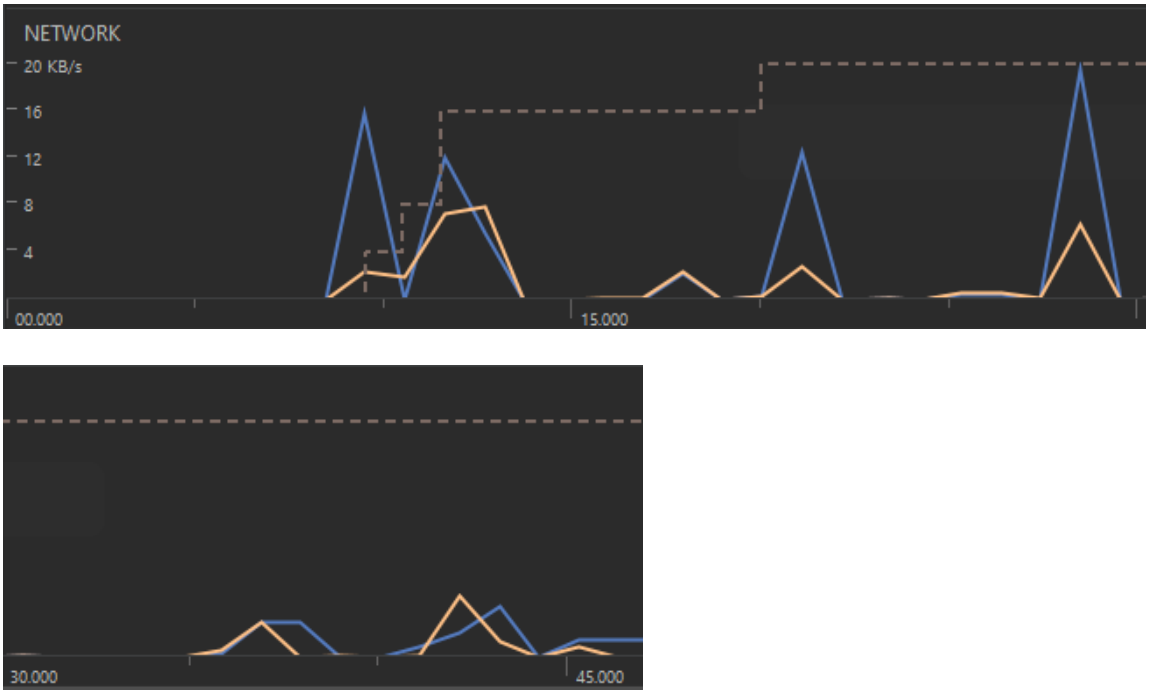


Figura 39. Prueba de estrés, carga y consumo de network

Fuente: elaboración propia. Realizada en Android Studio

Rendimiento de energía: en funcionamiento alto.

En Descanso (aplicación abierta): media. En Reposo (aplicación minimizada): baja.

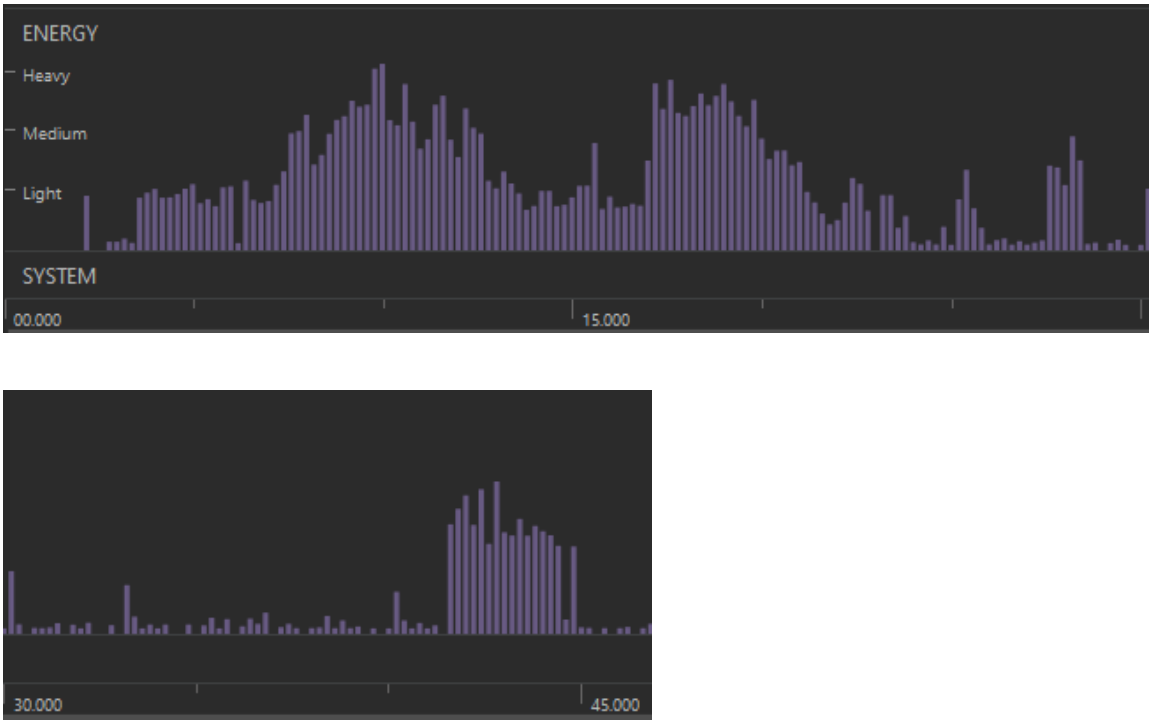


Figura 40. Prueba de estrés, carga y consumo de energía

Fuente: elaboración propia. Realizada en Android Studio

Para la realización de las pruebas de estrés y de carga de la página web, se utilizó la herramienta online Loadview de Dotcom Monitor, un modelo de servicio en la nube, la cual nos permitió simular la carga de usuarios concurrentes y peticiones que se realizan.

Con respecto a la prueba de estrés, donde se probaron los límites que soporta el sistema, se realizaron pruebas las cuales arrojaron un total de 7 usuarios virtuales administrativos disponibles, sin caídas en ninguno de ellos. Una cantidad considerable para el requerimiento que desea la página administrativa de GTAXI ya que en el momento solo se necesita 1 usuario administrativo.

Figura 41. Prueba de estrés en la web

Fuente: prueba propia. Realizada en https://www.loadview-testing.com/es/

Y con respecto a las pruebas de carga, donde se testeo el protocolo https y los usuarios concurrentes que generan solicitudes al servidor web, se obtuvieron los siguientes resultados. Para un total de 100 usuarios virtuales pico, en la duración de 7 minutos se llevaron a cabo 5766 sesiones estimadas; una cifra bastante alta, lo que nos permite realizar trabajos y actualizaciones en tiempo real sin saturar ni afectar la página web administrativa.

Figura 42. Prueba de carga en la web

Fuente: prueba propia. Realizada en https://www.loadview-testing.com/es/

Pruebas de usabilidad. En primer lugar se planificó un rango de trabajo en el cual se hicieron las primeras pruebas de usabilidad. El primer rango lo conformaron: pantalla principal, inicio de sesión, crear cuenta y términos y condiciones,. Estas primeras visualizaciones en los que tienen interacción los usuarios tuvieron las siguiente acotaciones que ellos mismos realizaron (grupo de estudio de taxistas: 10 y grupo de estudio de usuarios):

- Incomodidad con respecto al tamaño del texto y tamaño de los botones, se encontraban muy pequeños lo que perjudicaba visualmente al aplicativo.
- Su aprendizaje de uso es bastante rápido e intuitivo.
- Las imágenes que cumplen la función de íconos se ven pixelados en dispositivos de una pantalla más grande.
- No hay protección al usuario de hacer o cometer errores.
- El color amarillo es muy saturado a la vista y su diseño es muy simple.
- La estética de los botones de registro no era la adecuada por lo que se requería cambiar su apariencia para hacerlos de mayor tamaño.

Frente a estos aspectos de diseño y usabilidad, se realizaron los siguientes cambios en beneficio de los actores que intervenían con la aplicación y se mejoró considerablemente la aprobación de estos.

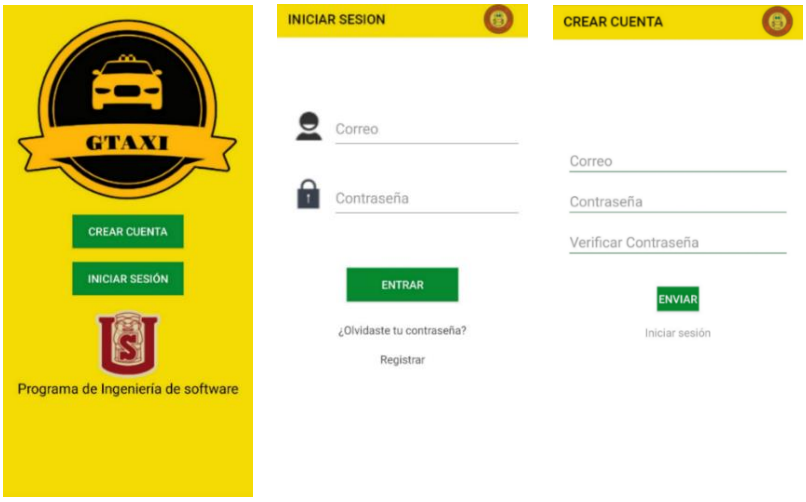


Figura 43. Pantallas iniciales antes de cambios de usabilidad

Fuente: elaboración propia. Tomada de dispositivo de prueba con Android 8

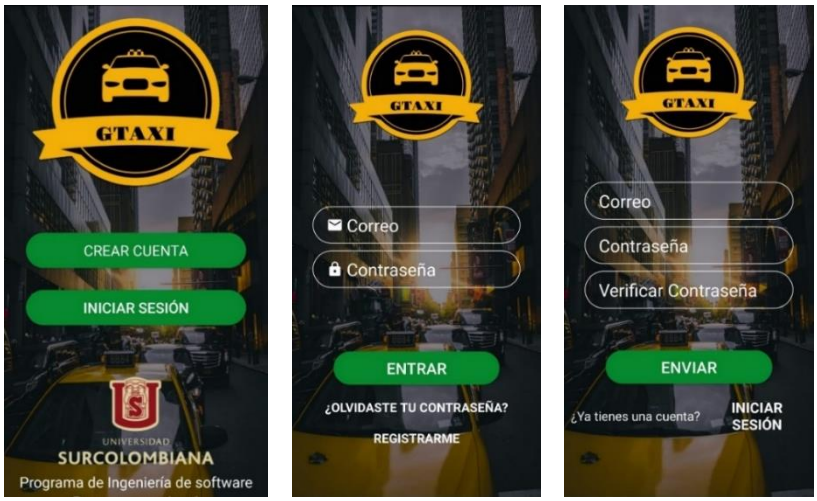
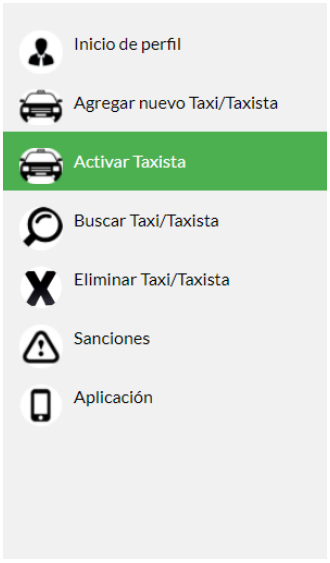


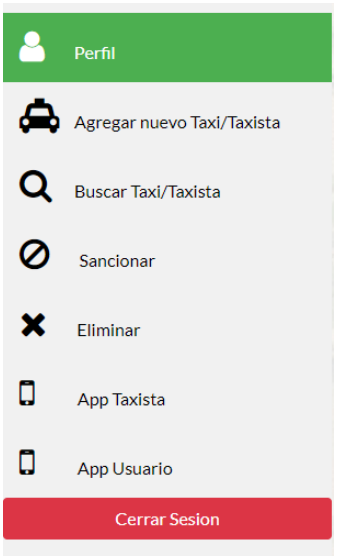
Figura 44. Pantallas iniciales después de cambios de usabilidad

Fuente: elaboración propia. Tomada de dispositivo de prueba con Android 10

Por otro lado, en la página administrativa se modificaron los tamaños de los campos de texto de todos los formularios, se cambiaron los íconos que existían en el menú lateral y se adicionó un hover al pasar el mouse por el menú. Anteriormente eran usadas imágenes en tamaño pequeño lo que ocupaba más tiempo de carga en la página. Ahora se

dispusieron íconos de la librería Font Awesome, sumado a esto, se adicionó el apartado de app de usuario y app de taxistas en caso de requerir futuras pruebas, hacerlas de manera interna antes de poner la app en producción.





(45) (46)

Figura 45. Diseño de menú antes de cambios de usabilidad

Figura 46. Diseño de menú después de cambios de usabilidad

Fuente: elaboración propia. Tomada en computador propio

Con estas modificaciones, los intervinientes observaron mejorías y dieron su visto bueno.

Pruebas de rendimiento. En este tipo de prueba se determinó la rapidez en tiempos de carga, de tareas en el sistema bajo condiciones de trabajo. Para la evaluación en la aplicación móvil, se utilizó la herramienta de productividad de Firebase donde se arrojaron los siguientes resultados de tiempo de visualización inicial, estadísticas gráficas de pruebas y el tiempo de procesamiento donde nos indicó que en los primeros milisegundos se realiza el mayor trabajo de cantidad de marcos (eje Y) medido en el tiempo (eje X) para posteriormente mantenerse estable en un promedio.

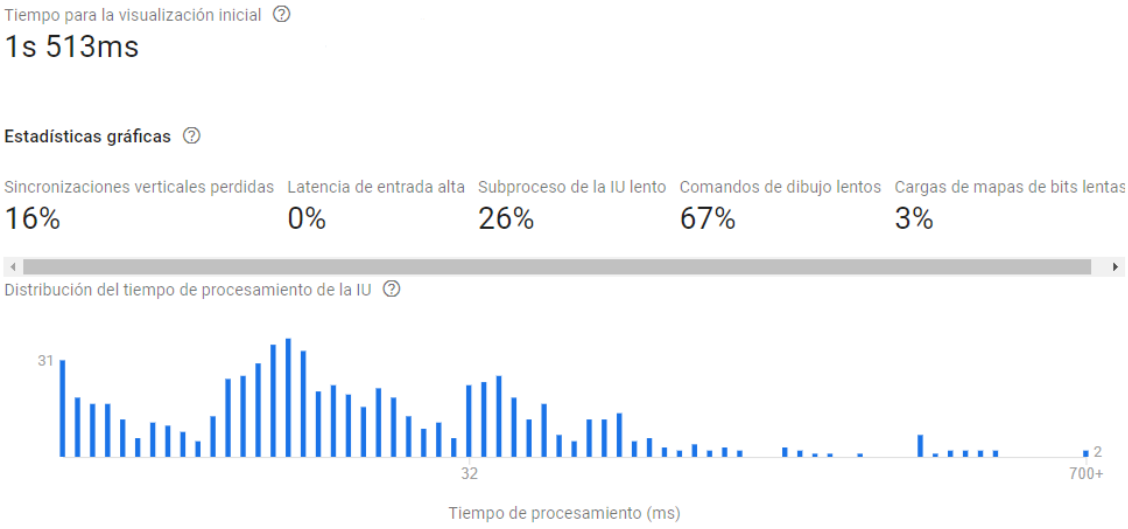


Figura 47. Prueba de rendimiento a través del tiempo de procesamiento

Fuente: elaboración propia. Realizada en https://console.firebase.google.com/

Para el entorno web se utilizó el software Dotcom-monitor, una plataforma online basada en la nube, que en este caso nos permitió medir los tiempos y monitoreo externo global de la web para asegurar la disponibilidad, funcionalidad y desempeño de los servicios / peticiones de internet.

En esta prueba realizada con cargas en varias partes del mundo se obtuvieron los siguientes resultados en las imágenes, donde podemos apreciar datos importantes, por ejemplo, el promedio de tiempo de carga es de 2,2 segundos y en algunas ubicaciones baja hasta los 1,01 segundos, el acceso por el dominio nos permite mayor facilidad al administrador que se encargará de los procesos internos.

Por otro lado, no se encontraron errores de carga y sí un tamaño reducido de 667 KB de consumo de recursos, una mejora considerable entre la primera y la última versión analizada hasta la fecha (v.1.5.4).

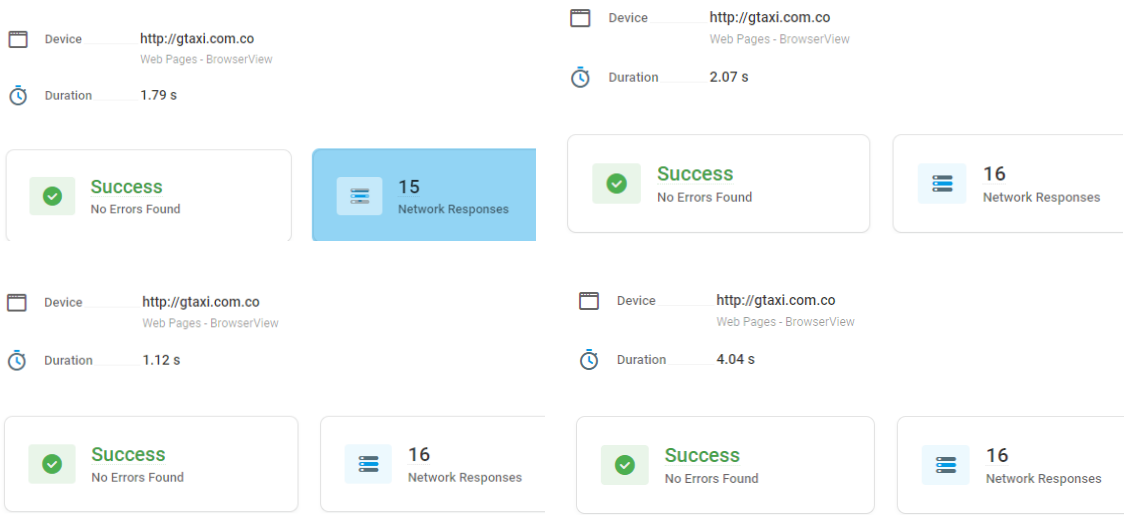


Figura 48. Pruebas de rendimiento de la web

Fuente: elaboración propia. Realizada en https://www.dotcom-monitor.com/es/

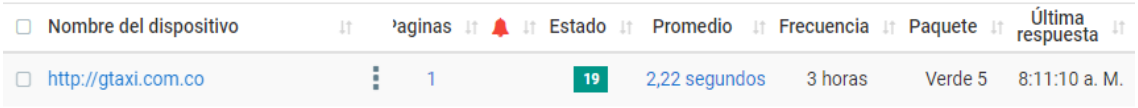


Figura 49. Resultado en conjunto de las pruebas de rendimiento de la web

Fuente: elaboración propia. Realizada en https://www.dotcom-monitor.com/es/

Sumado a esto, se utilizó otro software web para corroborar resultados del rendimiento de la web, para ello se optó por PageSpeed el cuál a través del acceso directo a la url, nos genera un reporte del rendimiento del sitio dando como resultado la siguiente imagen:

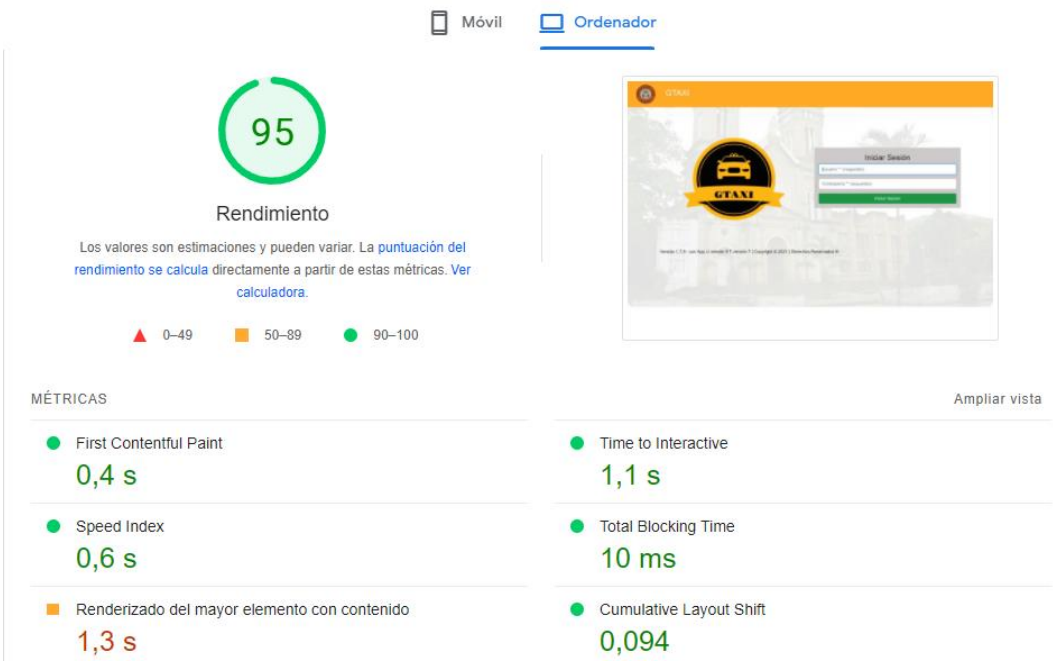


Figura 50. Resultado de prueba de rendimiento en la web con PageSpeed

Fuente: elaboración propia. Realizada con el software https://pagespeed.web.dev/

Pruebas de portabilidad. En este tipo de pruebas encontramos especificaciones que se desglosan en:

- Adaptabilidad: En este apartado, en la aplicación móvil de solicitud de taxi, se corroboro su funcionamiento en dispositivos Android. Aquellos dispositivos con otro sistema operativo como el iOS no podrán contar con el aplicativo, ni de usuario ni de taxista, debido a su creación en lenguaje nativo java. Se corrigió, a la vez que, en la web la adaptación a pantallas de móviles con una resolución menor al promedio, brindando así, soporte a estos equipos de menor tamaño físico.

Figura 51. Logo del sistema operativo Android

Fuente: (WORLDVECTORLOGO, s.f.). Descargar Powered By Android vector logotipo en SVG. Recuperado de: https://worldvectorlogo.com/es/logo/powered-by-android-1

Por otro lado, se determinó la flexibilidad del sistema hacia un cambio de ambiente, hardware, software y compatibilidad de idiomas. Se observó en la página administrativa web, que existe la compatibilidad de idiomas tanto de inglés como español. El software no presenta inconvenientes corriendo en cualquier sistema operativo de equipo. Sin embargo, si existen dificultades para ser usado en algunos hardware, específicamente en dispositivos extremadamente pequeños ya que los botones y textos ocupan la mayor parte del espacio visual lo que imposibilita la fácil lectura. Es por ello, que se hicieron correcciones en este ámbito para ser adaptable a la pantalla.

Figura 52. Adaptabilidad de la web en entorno de escritorio

Fuente: elaboración propia. Realizada en el navegador de Google Chrome

Figura 53. Adaptabilidad de la web en entorno móvil

Fuente: elaboración propia. Realizada en el navegador de Google Chrome

- Instalabilidad: En este aspecto se determinó si el software se instala fácilmente en nuevos entornos o si por el contrario requiere de mucho esfuerzo y cambios, también si es fácil de implementar en los recursos de memoria disponibles.

Con respecto a esto, para dispositivos móviles resulta fácil su instalación al ser descargada a través de la PlayStore para usuarios y distribuida por la cooperativa para conductores taxistas para su posterior publicación por medios digitales.

Figura 54. Logo de la tienda digital Google PlayStore

Fuente: (Google Play, s.f.). Google Play. Recuperado de: https://play.google.com/intl/es-419/badges/

Por otro lado, con la web, se llegó a que el software GTAXI es fácil de instalar ya que se maneja como web y no requiere de mayores conocimientos específicos, solo con

buscar el dominio de Gtaxi.com.co. Sumado a esto, no consume bastantes recursos por lo que sí es posible correr el programa administrativo con la cantidad de memoria disponible en los equipos de la cooperativa.

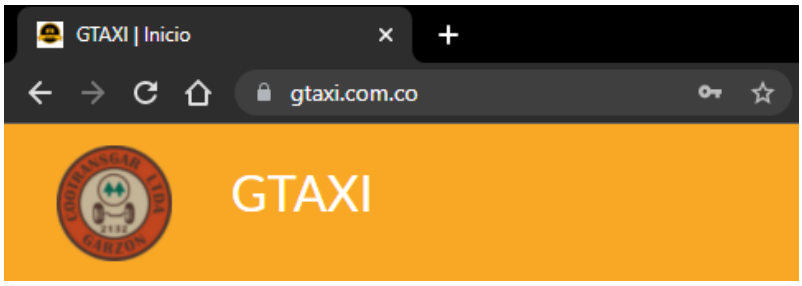


Figura 55. Búsqueda de la web por el navegador

Fuente: Fuente: elaboración propia. Realizada en el navegador de Google Chrome

- Reemplazabilidad: En este aspecto se implicó la capacidad de que el software sea utilizado en lugar de otro producto software, que realiza el mismo conjunto de operaciones. En este aspecto, el producto cumple en cierto aspecto con este término, ya que, si bien está limitado a solo el municipio de Garzón y no tiene todas las funcionalidades de otras aplicaciones, como lo son las tarifas, puede trabajar a la par de herramientas similares como Easytaxi o alternativas como llamadas vía telefónica a la operadora. Con respecto a la página web, está enfocada únicamente a los datos de los taxistas vinculados a la cooperativa, sin embargo, si el software no dependiera de ello, si pudiese ser utilizado, pero en una mínima medida.

Figura 56. Pantalla interna de la app en tema de reemplazabilidad

Fuente: elaboración propia. Tomada en dispositivo de prueba Android 10

- Coexistencia: En este indicativo, se evaluó la presencia de varios sistemas juntos en un mismo entorno sin que se afecten negativamente entre sí. Para el aplicativo móvil si es necesario tenerlo abierto en primer plano para poder realizar la aceptación o cancelación del servicio, ya para otros temas como lo son las alarmas o notificaciones no es necesario mantenerla activa, solo basta con tenerla en segundo plano.

Figura 57. Formulario de login para la coexistencia de software

Fuente: elaboración propia. Tomada en dispositivo propio de prueba con Android 10

En este tema para la web, los administradores que tienen acceso se podrán conectar e ingresar mientras que todo el resto del público no podrá tener el acceso. Es posible correr este programa sin la necesidad de cerrar otro tipo de software existente, por lo que la coexistencia con otro tipo de software se lleva a cabo de manera perfecta.

Figura 58. Formulario de login para la coexistencia de software

Fuente: elaboración propia. Tomada en navegador de prueba Google Chrome

Pruebas funcionales

Para especificar más acerca de las pruebas funcionales, se basó en el siguiente esquema con el objetivo de cumplir y llevara cabo todas las pruebas pertinentes que nos muestra la norma ISO/IEC 25000 pero de una manera mucho más detalla en cada una.

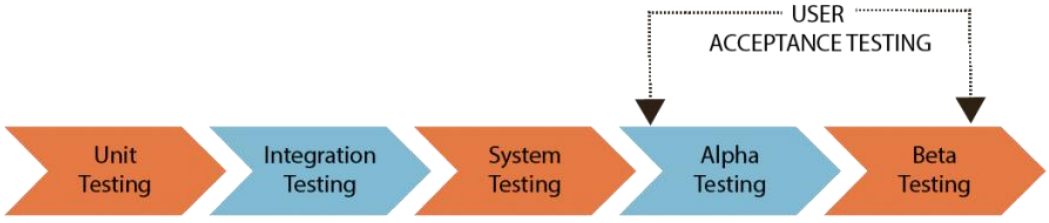


Figura 59. Secuencia de pruebas funcionales

Fuente: (Conversant Technologies, 2017). Pruebas de software y análisis de calidad: los elementos esenciales de cualquier desarrollo de sistema infalible. Recuperado de https://conversantech.com/software-testing-quality-analysis/

Con este esquema ya definido, se realizaron las siguientes pruebas descritas a continuación.

Pruebas unitarias. En el desarrollo de este tipo de pruebas, se tuvo el apoyo en el software de Android Studio y así realizar testeo unitario donde se evaluó el comportamiento correcto de la unidad de código con el objetivo de asegurar un funcionamiento eficiente, que esa sección realiza lo que debe de realizar y entregar un resultado o estado final.

Para ello, se establecieron unos casos de prueba de software donde a cada uno se le estableció la funcionalidad, el resultado esperado, procedimientos y resultado obtenido, **ver anexo**

Pruebas de componentes. Se realizo la prueba en el componente de búsqueda o consulta de taxis y de taxistas. En estos testing se observó la no existencia de errores, como lo fue también en los componentes de adición y modificación. Sin embargo, si lo hay cuando se consulta al taxista, debido a una carga de todos los conductores, mostrar el indicado y volver a eliminar los conductores que no coinciden.

Web administrativo

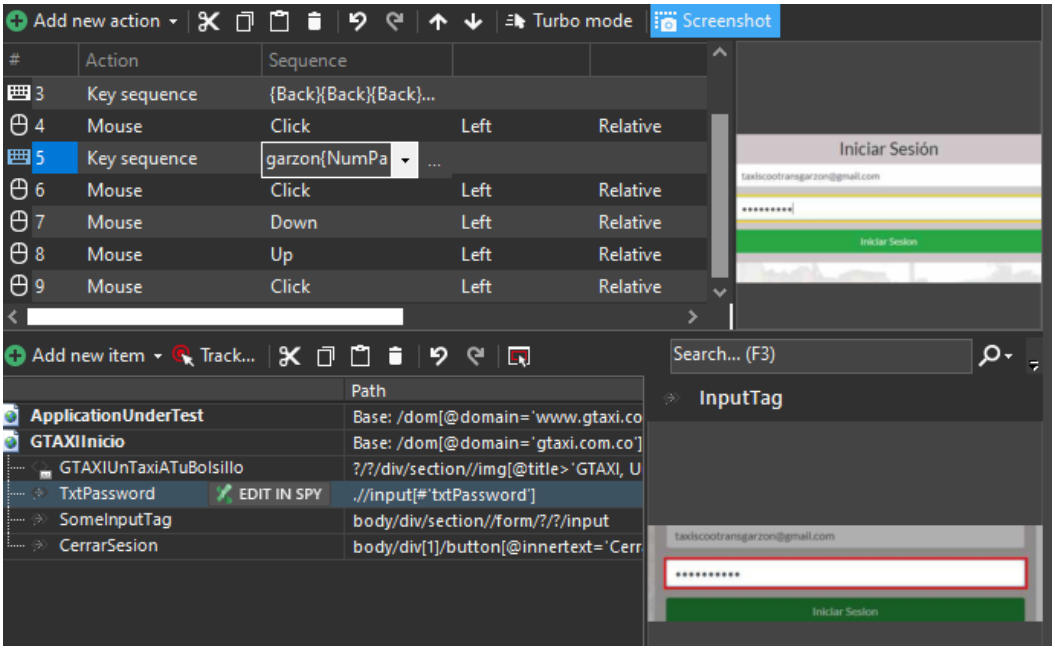


Figura 60. Prueba de componente de sesión

Fuente: elaboración propia. Tomada del software Ranorex

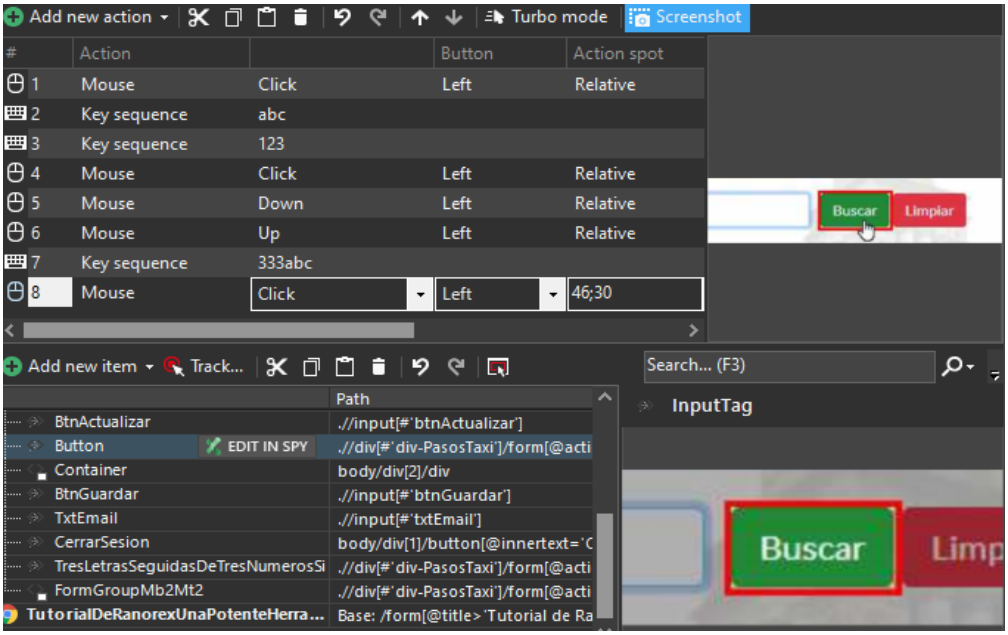


Figura 61. Prueba de componente consultar taxi

Fuente: elaboración propia. Tomada del software Ranorex

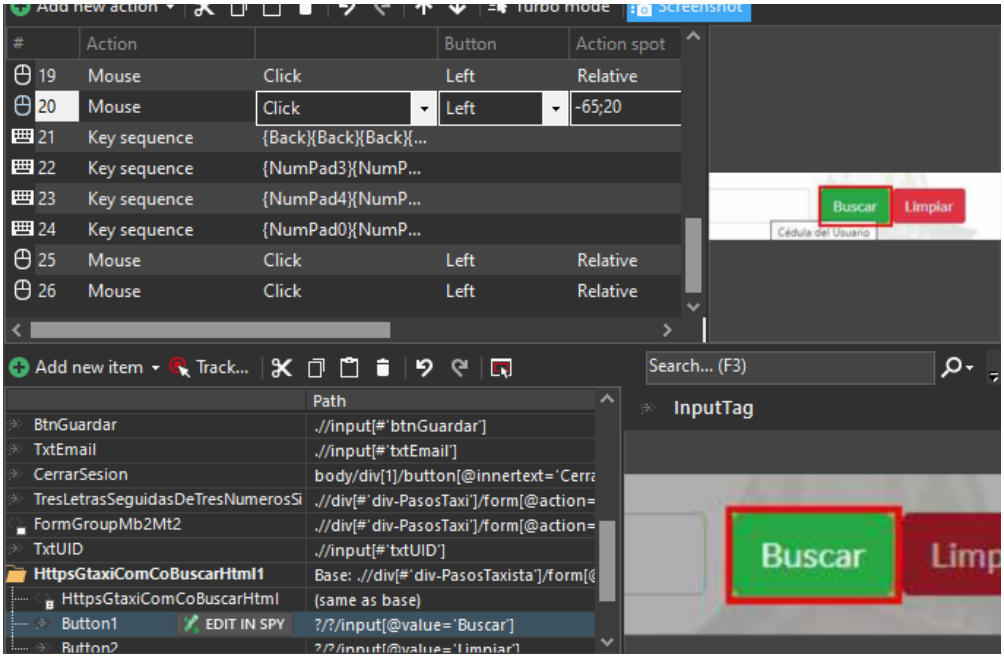


Figura 62. Prueba de componente consultar taxista

Fuente: elaboración propia. Tomada del software Ranorex

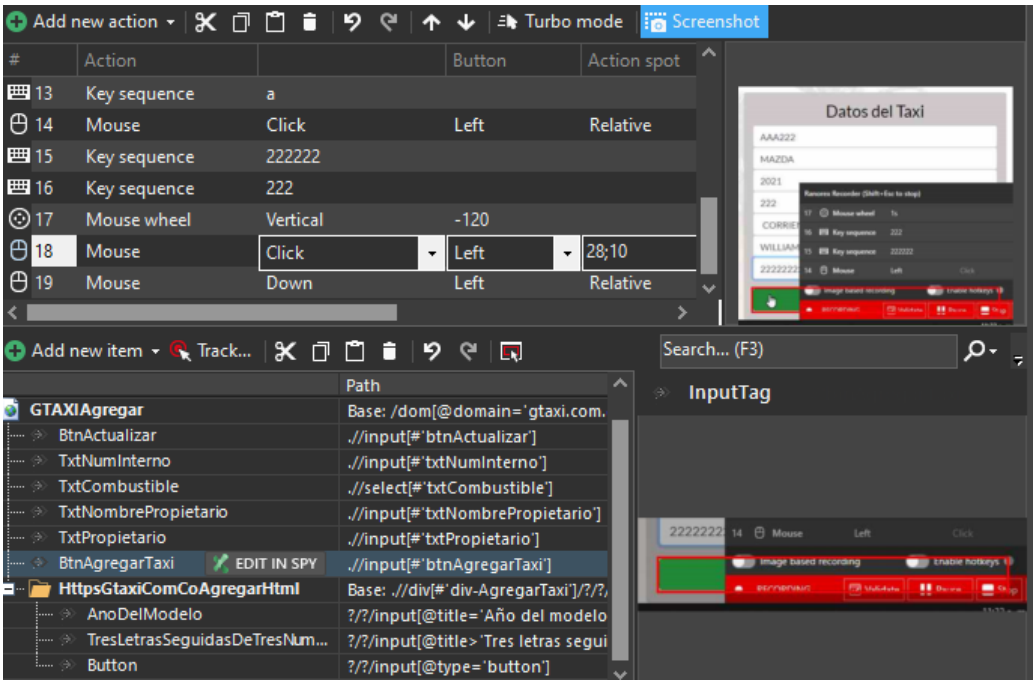


Figura 63. Prueba de componente agregar taxi

Fuente: elaboración propia. Tomada del software Ranorex

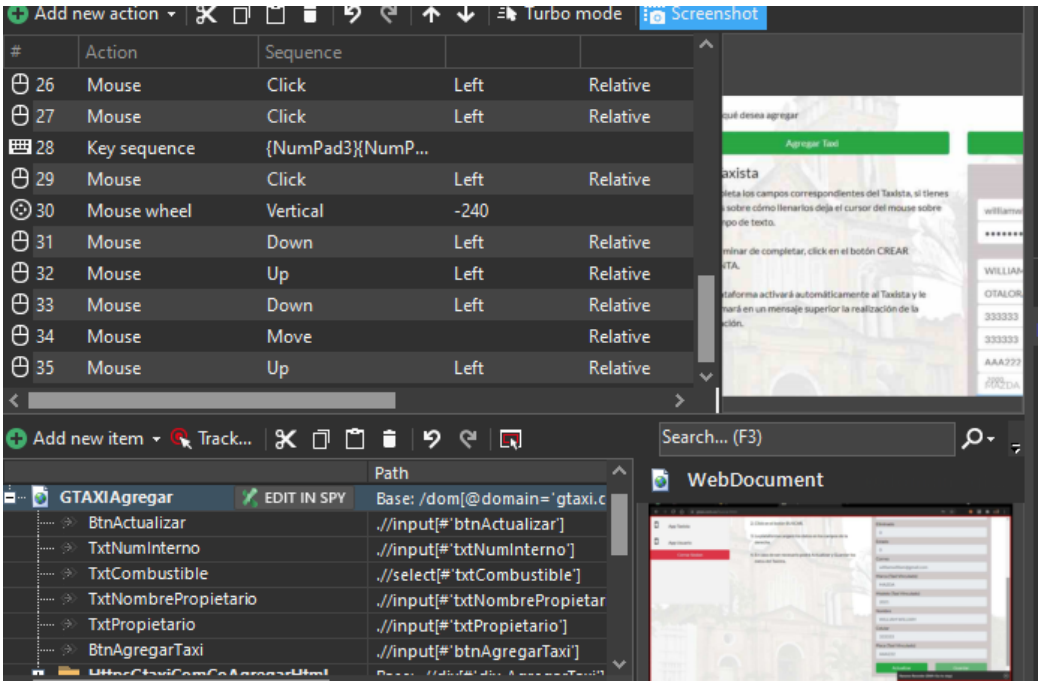


Figura 64. Prueba de componente agregar taxista

Fuente: elaboración propia. Tomada del software Ranorex

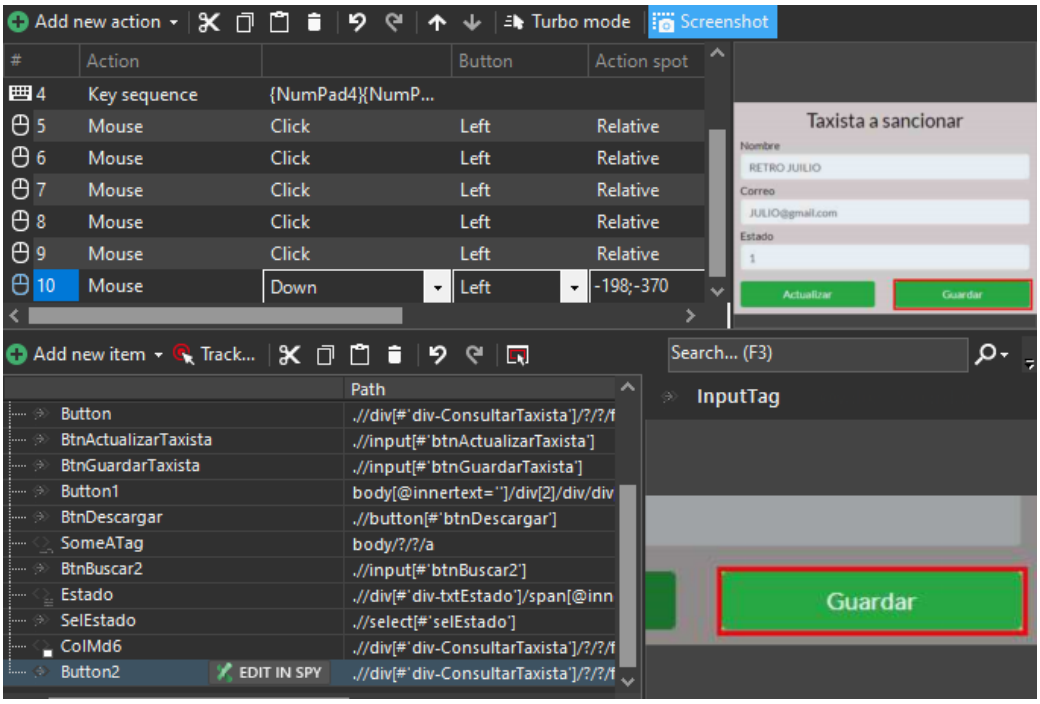


Figura 65. Prueba de componente sancionar taxista

Fuente: elaboración propia. Tomada del software Ranorex

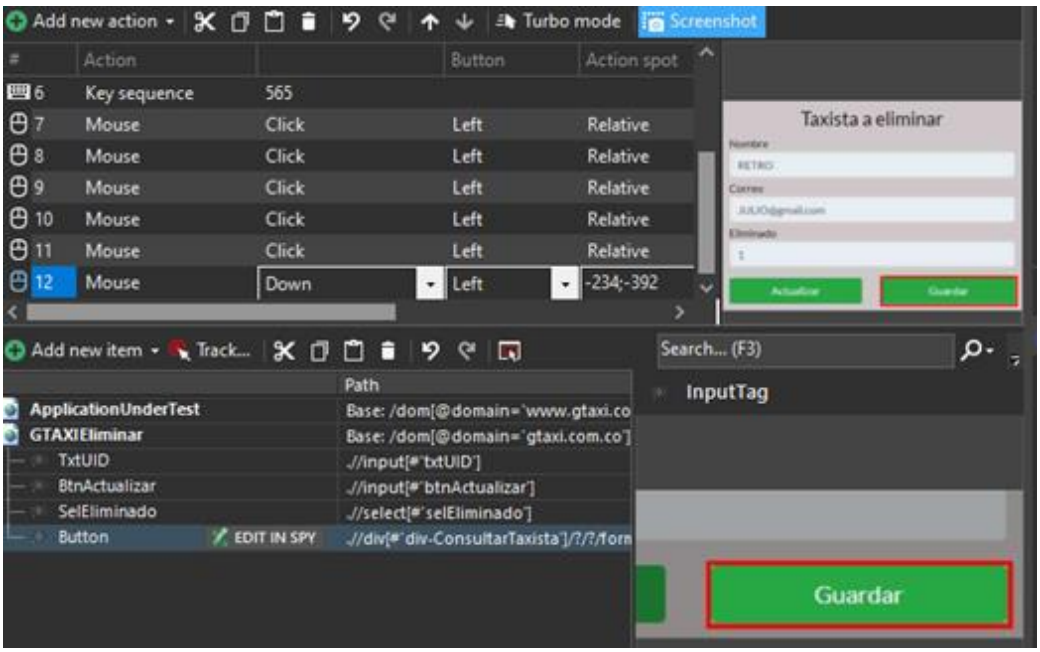


Figura 66. Prueba de componente eliminar taxista

Fuente: elaboración propia. Tomada del software Ranorex

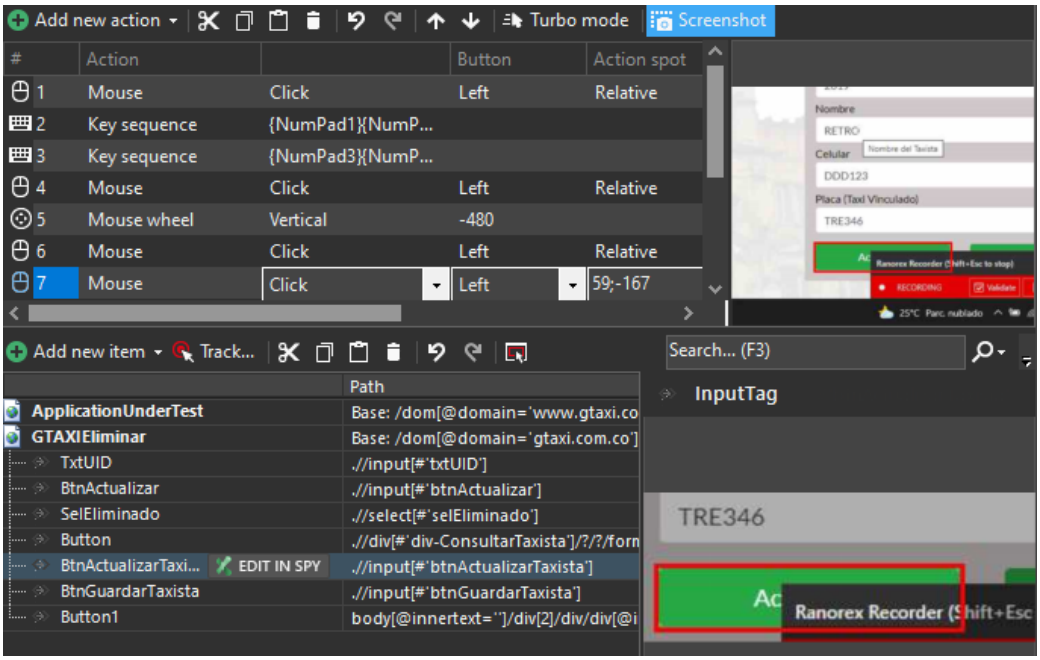


Figura 67. Prueba de componente de actualizar taxista

Fuente: elaboración propia. Tomada del software Ranorex

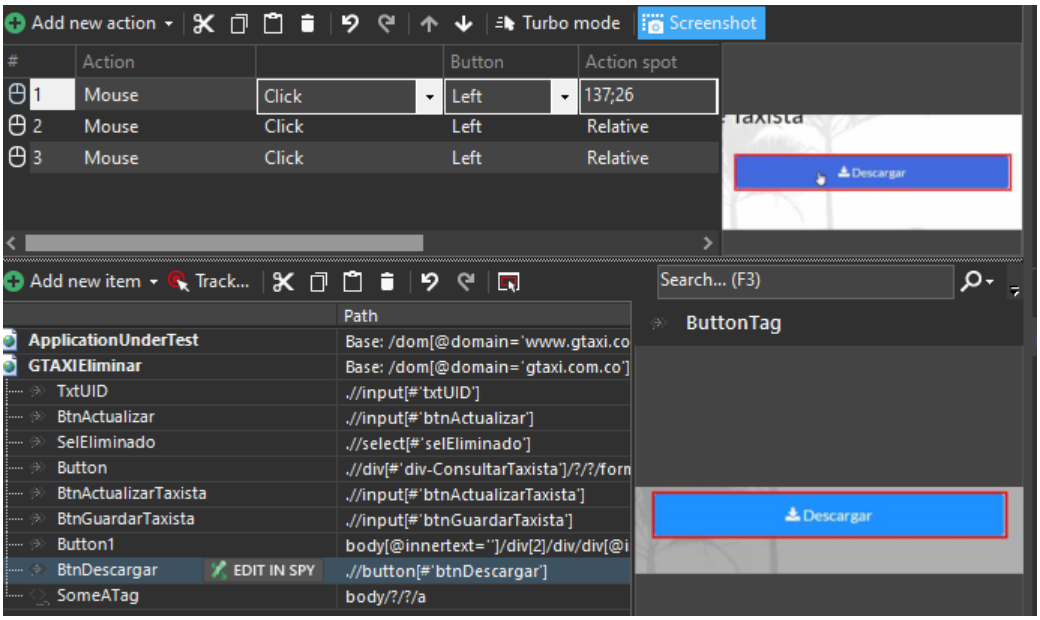


Figura 68. Prueba de componente de descargar app

Fuente: elaboración propia. Tomada del software Ranorex

Pruebas de humo. En este tipo de pruebas se realiza una revisión rápida del producto para comprobar su funcionamiento correcto y la inexistencia de errores que afecten las operaciones básicas del programa.

Para llevar a cabo estas pruebas se maneja de 2 maneras. La primera manera es de forma manual, donde se realizarán pruebas como si el software estuviese en un entorno real. La segunda manera es de forma automática donde se utilizaron 2 herramientas: Pentest tools, una web que nos evalúa el software con pruebas de caja negra y Nibbler, un software web gratuito que nos brinda un informe que califique el sitio web para áreas clave, incluida la accesibilidad, seo, experiencia y tecnología.

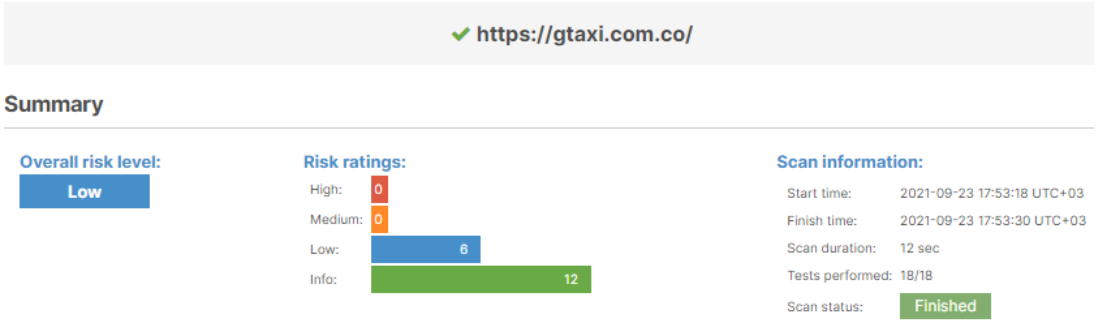
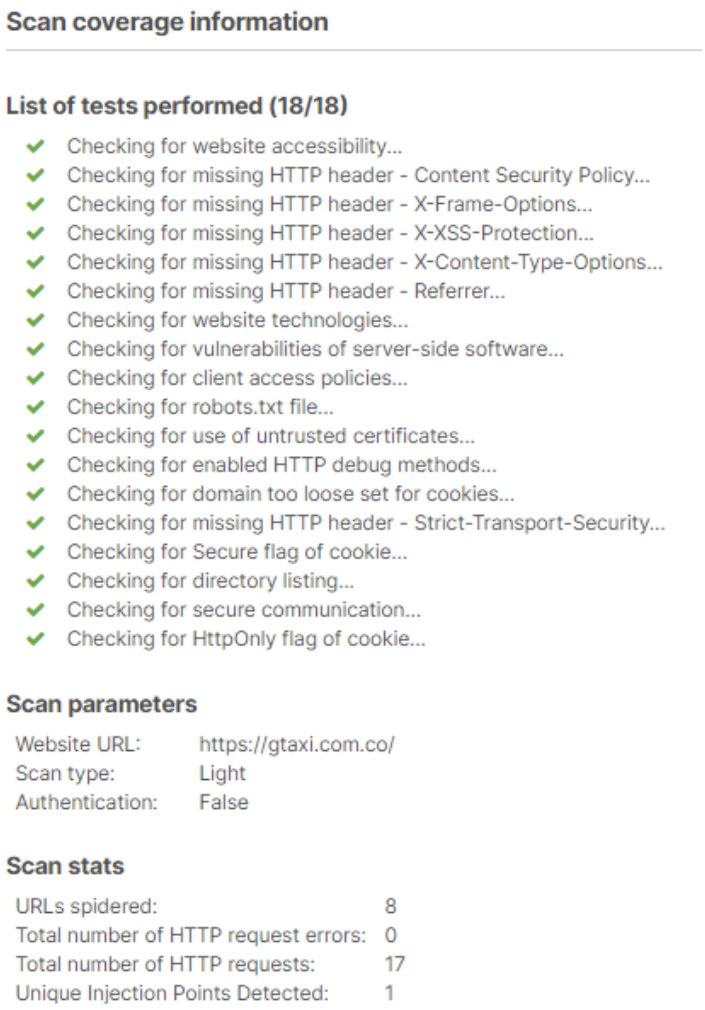


Figura 69. Prueba de humo e información escáner para la web

Fuente: elaboración propia. Realizada en el software Pentest tools



Scan coverage information

List of tests performed (18/18)

- Checking for website accessibility...
- Checking for missing HTTP header - Content Security Policy...
- Checking for missing HTTP header - X-Frame-Options...
- Checking for missing HTTP header - X-XSS-Protection...
- Checking for missing HTTP header - X-Content-Type-Options...
- Checking for missing HTTP header - Referrer...
- Checking for website technologies...
- Checking for vulnerabilities of server-side software...
- Checking for client access policies...
- Checking for robots.txt file...
- Checking for use of untrusted certificates...
- Checking for enabled HTTP debug methods...
- Checking for domain too loose set for cookies...
- Checking for missing HTTP header - Strict-Transport-Security...
- Checking for Secure flag of cookie...
- Checking for directory listing...
- Checking for secure communication...
- Checking for HttpOnly flag of cookie...

Scan parameters

Website URL:	https://gtaxi.com.co/
Scan type:	Light
Authentication:	False

Scan stats

URLs spidered:	8
Total number of HTTP request errors:	0
Total number of HTTP requests:	17
Unique Injection Points Detected:	1

Figura 70. Reporte detallado de la prueba de humo

Fuente: elaboración propia. Tomada del software Pentest tools

En esta prueba automatizada se realizaron pruebas de accesibilidad, protocolos http, comunicación con el servidor y base de datos. Dentro de este reporte de pruebas se observa una seguridad estable, un funcionamiento general correcto, con el cumplimiento de requerimientos y con calidad en los procesos de guardado y petición de datos.

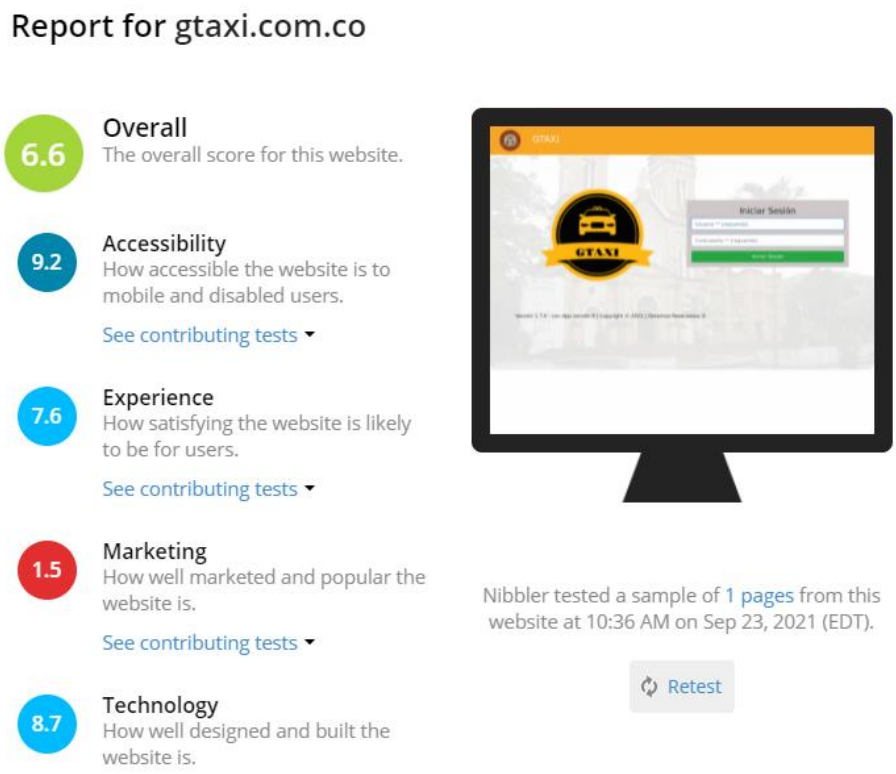


Figura 71. Prueba automatizada de reporte general para la web

Fuente: elaboración propia. Tomada del software Nibbler

En esta herramienta más general, se hace un testing sobre la accesibilidad del sitio administrativo, testing sobre la experiencia que puede tener el usuario y la tecnología implementada sea de fácil entendimiento para quien administre y lleve a cabo el control del software.

Pruebas de integración. En este testing se prueban los elementos unitarios que componen el software y el funcionamiento correcto en conjunto. Se centra en probar la comunicación, entre los componentes, y la comunicación entre software externo.

El principal aspecto por destacar es que el funcionamiento en conjunto se relaciona de manera correcta sin ninguna alteración o error en los procesos principales, sin embargo, existe una baja calificación en el performance, reporte que utilizando una de las herramientas extensiones del

navegador de Google de nombre Lighthouse nos muestra esa estadística de nuestra web, sin afectar el funcionamiento principal, pero si para tener en cuenta.

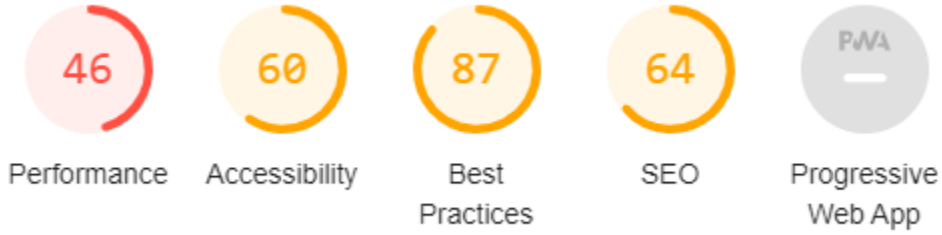


Figura 72. Prueba de integración general en la web

Fuente: elaboración propia. Tomada del navegador Google Chrome

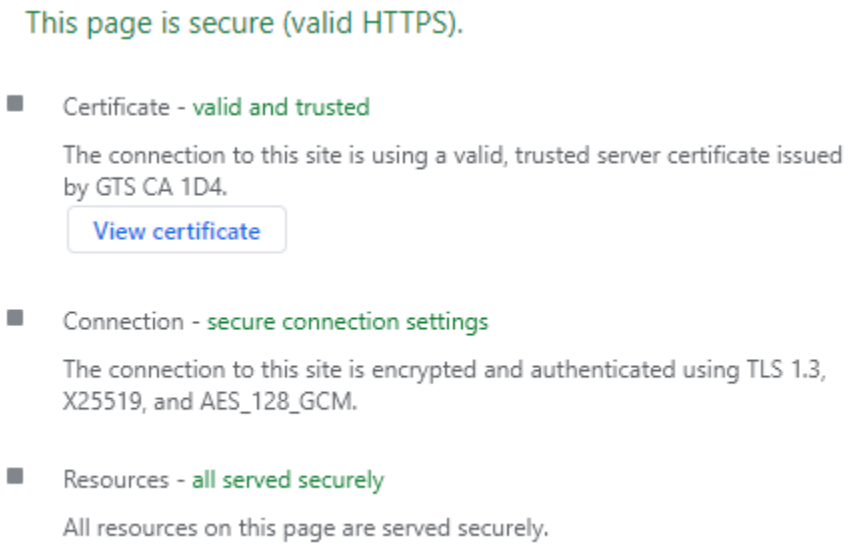


Figura 73. Prueba de seguridad y protocolo HTTPS

Fuente: elaboración propia. Tomada del navegador Google Chrome

Sumando a esto, se hicieron pruebas manuales donde se trabajaba con los módulos de crear usuario y buscar usuario para que trabajaran en simultaneo y su correcto funcionamiento lo más antes posible.

Pruebas del sistema. Se realizaron este tipo de examen de forma manual, donde el personal encargado de la administración pudo testear de manera completa el software web. Gracias a las pruebas individuales que se detallaron anteriormente, la página se probó de forma global donde se distinguieron los aspectos de componentes, rendimiento, carga, facilidad de uso, seguridad y más. Finalmente, la cooperativa validó su visto bueno donde se aprobó y dio por sentado el software.

Pruebas de aceptación. Para realizar este tipo de pruebas se tuvo que haber realizado unas pruebas de software previamente, que son fundamentales para continuar con las aceptaciones que darán los testers de software, usuario cliente, usuario taxista y la cooperativa.

Para llevar a cabo estas pruebas de aceptación, se dividirán en pruebas alfa y pruebas beta.

Pruebas alfa. En este tipo de testing, se usó el software GTAXI de manera natural, donde el usuario administrador iba controlando y realizando las tareas pertinentes como un proceso cotidiano, mientras que uno de los integrantes del proyecto realizaba observación y registrando errores y/o problemas de uso.

Se hizo validación de la aplicación móvil, se aplicó una prueba piloto liberando única y exclusivamente la última versión de la app a un grupo selecto de taxistas y usuarios clientes, quienes probaron el software desde sus propios dispositivos móviles y comentaron los inconvenientes que tenían en el momento y formulaban sugerencias que se tomaron en cuenta para las versiones que se iban actualizando, dentro de las pronunciadas recomendaciones que se realizaron al aplicativo móvil fueron:

- Se necesita un sistema para notificar al usuario en caso de que el taxi ya esté esperándolo, por ejemplo, fuera de la casa.

- Se requiere aparte de la ubicación por GPS, una forma de digitar la dirección de donde estoy ubicado de forma manual.
- Ayudaría bastante que los campos de texto sean de mayor tamaño, así como los botones que permiten interacción.

Al tener en cuenta estas manifestaciones, se corrigió el software y se les presentaron los siguientes formatos que fueron aprobados.

Tabla 36
Prueba de aceptación: notificación

Título	**Descripción**
Nombre:	Notificación al usuario de taxi llegó a su ubicación
Código:	PA-01
Descripción:	La aplicación debe permitir al taxista generarle una alarma tipo notificación al usuario para avisar en caso de que ya llegó a la ubicación.
Condiciones de ejecución:	• Debe estar en marcha la ejecución de una carrera del servicio de taxi. • Debe haber conexión a internet. • Tanto usuario como taxistas deben estar autenticados.
Entradas:	• Una petición de taxi abierta y en proceso.
Pasos de ejecución:	1. Abrir la aplicación. 2. El taxista dará click al botón de notificar usuario cuando se encuentre en su ubicación. 3. El software notificará al receptor (en este caso usuario) cuando su taxi de la petición ya haya llegado.
Resultado esperado:	Se debe recibir una alarma en forma de notificación, donde el usuario sea informado y avisado de su transporte.
Evaluación de la prueba:	La prueba realizada con la interacción de taxista – usuario a través de notificación a este último, concluyó con un caso de éxito del 100% debido a que ya entablando la conexión y existiendo una petición o proceso entre las 2 cuentas autenticadas, es mucho más fácil su interacción.

Fuente: elaboración propia

Tabla 37
Prueba de aceptación: dirección

Título	Descripción
Nombre:	Ubicación escrita manualmente
Código:	PA-02
Descripción:	La aplicación debe brindar al usuario un espacio para diligenciar su dirección de recogida para que el taxista llegue a la ubicación de una manera mucho más concisa y exacta, diferente a la marcada por el mapa de ubicación del dispositivo que en circunstancias desactualizadas de mapeo puede errar.
Condiciones de ejecución:	• Debe haber conexión a internet. • El usuario debe estar autenticado o haber iniciado sesión. • Se debe hacer iniciado la solicitud del servicio en el botón pedir taxi.
Entradas:	• Iniciar una solicitud de taxi (click en botón pedir taxi).
Pasos de ejecución:	1. Abrir la aplicación. 2. Autenticarnos en caso de tener una cuenta existente, de lo contrario hacer el proceso de registro. 3. Click al botón de solicitud “Pedir taxi” y marcar en el mapa nuestro destino. 4. Completar el campo emergente de mi ubicación o donde queremos que nos recoja el taxi.
Resultado esperado:	Se debe registrar en la solicitud de taxi la dirección diligenciada, para que el taxista al ver la petición y tras aceptarla, pueda tener la información y la claridad hacia donde se debe dirigir para recoger al pasajero, además de la información brindada por el puntero del GPS del dispositivo solicitante.
Evaluación de la prueba:	La prueba realizada con la dirección de forma manual concluyó con un caso de éxito del 100% del funcionamiento debido a que se almacenó de forma temporal para que luego sea accedida por el taxista con quien se entable la petición del servicio para así poder llegar a la ubicación descrita de una forma mucho más consistente.

Fuente: elaboración propia

Tabla 38
Prueba de aceptación: tamaño de campos y texto

Título	Descripción
Nombre:	Mayor tamaño de campos de texto y botones
Código:	PA-03
Descripción:	La aplicación requiere de un tamaño mayor en cuanto a los campos de texto y botones ya que hasta el momento se encuentran de una dimensión reducida.
Condiciones de ejecución:	• Debe haber conexión a internet. • El usuario debe ingresar a la aplicación. • Para botones internos como lo son el: marcar ruta, otro taxi o cancelar, es necesario autenticarse y solicitar taxi.
Entradas:	• Iniciar la aplicación.
Pasos de ejecución:	1. Abrir la aplicación. 2. Observar tanto botones como campos iniciales. También captar la dimensión de los campos existentes en la continuación cuando se completa la primera parte del registro. 3. Para demás botones internos, iniciamos nuestra sesión y al pedir y llevar a cabo la petición del servicio serán observados dichos botones.
Resultado esperado:	Con los nuevos cambios en cuanto al tamaño de campos de texto por diligenciar y los botones que llaman a acciones dentro de la aplicación, se busca la satisfacción del cliente quien visualmente hacía un esfuerzo anteriormente pero ahora ese descaste no lo es en gran consideración.
Evaluación de la prueba:	La prueba realizada con los campos y botones se concluyó con un caso de éxito del 100%, tanto usuarios como taxistas notaron los cambios de tamaño tomados y destacaron su mejoría visual.

Fuente: elaboración propia

Con respecto a la web, las primeras pruebas realizadas arrojaron fallas menores al momento de mostrar los formularios de taxi/taxista, ya que estos se sobreponían unos con otros.

Adicional, tanto el módulo de la eliminación como la sanción se realizaban, pero no afectaban internamente a la base de datos, lo que no llevaba a cabo el registro de la nueva información para inhabilitar al conductor sancionado o eliminado.

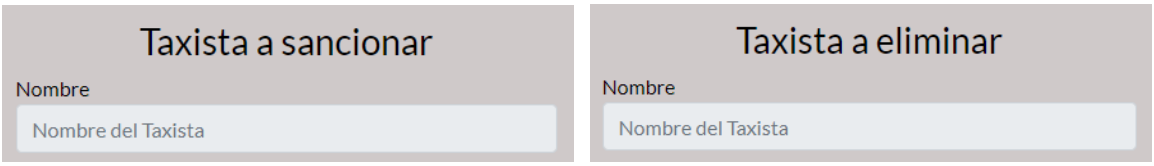


Figura 74. Prueba alfa en sanción y eliminación

Fuente: elaboración propia. Tomada del navegador Google Chrome

Otra acotación vista fue cómo se llenaban los campos de texto, algunos con minúsculas y otros con mayúsculas, lo cual podía generar confusiones en la base de datos o sobreescritura del mismo dato, esto se corrigió otorgando funciones para estandarizar los registros de añadir con solo mayúsculas.

Adicional a esto, la consulta de taxistas se estaba realizando a través del UID de autenticación de Firebase, lo que es un método bastante complicado y engorroso. Para ello, se replanteó la consulta otorgándole así al número de cédula la referencia en el nodo de taxista en la base de datos.

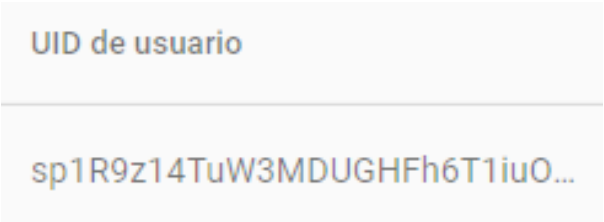


Figura 75. Ejemplo de UID de usuario

Fuente: elaboración propia. Tomada de la consola de Firebase

Pruebas beta. Se corrigieron previamente los errores encontrados en todas las demás pruebas, tanto funcionales como no funcionales. Así mismo, se realizaron unas pruebas beta en las cuales se puso a disposición el software final, del administrador, a la cooperativa y al ente interno

encargado. Se llevaron a la práctica pruebas por usuarios reales en un entorno real, donde se pudieron recopilar comentarios de esta versión del software.

Se validaron la adición de taxis y las creaciones de usuarios taxistas.

Se realizaron pruebas de funcionalidad, usabilidad y confiabilidad del sistema.

El lanzamiento beta se trabajó con la cooperativa Cootransgar y con ellos se evaluó el éxito del producto en comentarios y sugerencias con el fin de valorar la satisfacción del cliente garantizando la preparación de la versión a desplegar.

Finalmente, se concluyó que se efectuó una revisión efectiva de la retroalimentación proporcionada por la cooperativa y los participantes de las pruebas.

Discusión de resultados

Con los resultados obtenidos en el proyecto, se observó una gran mejora en la calidad del software, su objetivo de desplegar la herramienta GTAXI para la solicitud del servicio y que esta se encuentre disponible al público se llevó a cabo de manera efectiva. El proceso partió de un programa que se encontraba en deplorables condiciones, sin un correcto funcionamiento, con un estado inicial donde no se especificaban pruebas, no contaba con evaluación de la calidad sea propia o basado en alguno de los modelos existentes como CMMMI, MCCALL, BOEHM, ISO 9126, ISO 25000, entre otros. Y por supuesto, no contaba con la aprobación de los taxistas quienes iban a ser los directos beneficiarios del software, obteniendo un porcentaje de satisfacción del 0% al 10% de aceptación.

Junto con las nuevas actualizaciones que se realizaron tanto en la app como en la web, los resultados conseguidos evidencian mejoras en la forma de despliegue gracias a la estrategia de implementación optada, dando la facilidad de volver a versiones anteriores en caso de ser necesario y actualizar a una nueva versión. Continuo a esto, se adoptó una norma para la calidad del producto software la cual, con las permutaciones y nuevas versiones corregidas basadas en la calidad a través de la norma ISO/IEC 25000, los resultados reflejan que, a nivel de valoración en la escala de medición establecida, los puntajes arrojados son altos al grado de muy satisfactorio, siendo el coeficiente en porcentaje total del 93%. Esto nos da a entender que el producto software cumple con las características aplicables de la norma y se encuentra dentro de las valoraciones aceptables por la cooperativa y por el gremio de taxistas. En futuros trabajos se piensa trabajar en las características que componen la ISO para llegar a un coeficiente del 96%.

Los resultados de manera específica en las características y subcaracterísticas de ISO 25000, mostraron puntos fuertes y otros que faltan por profundizar en mayor detalle. Dentro de las 8

características con su nivel de importancia clasificado en alto, medio, bajo y no aplica, se destacan las 3 subcaracterísticas con mayor porcentaje de calidad del sistema, las cuales fueron: Disponibilidad (actualmente en 100%) debido a que el software es capaz de estar operativo y accesible para su uso cuando se requiere, sin limitar un tiempo de trabajo. Facilidad de instalación (actualmente en 100%) como causa de su capacidad de ser instalado y desinstalado de forma exitosa en un entorno determinado, las pruebas culminaron de manera satisfactoria. Autenticidad (actualmente en 100%) debido a que el software es capaz de demostrar la identidad de un sujeto o un recurso gracias a la tipificación única trabajada en Firebase. Por otro lado, se visualizaron 3 subcaracterísticas con el menor porcentaje de calidad del sistema, las cuales fueron: Capacidad de recuperación (actualmente en un 50%) debido a que el software no es del todo capaz de recuperar datos directamente afectados, pero si reestablecer el estado deseado del sistema en caso de fallo, que es el programado por defecto. Modularidad (actualmente en 60%) ya que los componentes se encuentran entrelazados y existe afectación de impacto en los demás en caso de intervenir o cambiar los mismos. Capacidad (actualmente en 75%) a causa de presentar inconvenientes en los límites máximos del sistema para llega a cumplir con los requisitos, como lo es en dispositivos móviles con un sistema operativo mínimo al requerido, poca memoria ram y pantalla muy pequeña.

En los trabajos futuros se piensa elevar el porcentaje de cada una de las subcaracterísticas hasta en un mínimo del 80%. La capacidad de recuperación con definir cuándo el sistema automáticamente debe saltar a su estado por defecto sin llegar a crashear la aplicación. La modularidad con métodos mucho más específicos y dividiendo los módulos y paquetes en el código de la app. La capacidad con adaptación a pantallas e implementar diseños para dispositivos aún más antiguos que se pueden llegar a presentar en una ciudad como Garzón.

Con estos resultados podemos analizar que si bien es un producto que cumple con lo propuesto, con lo que planteaba la cooperativa y lo que necesitaban los taxistas, podrían llevarse a cabo más pruebas específicas que abunden en la calidad del producto o incluso con herramientas de pago que sirvan en pro del software de servicio de taxi.

En cuanto a crítica, se puede observar un software que cumple con lo requerido, pero se debe mejorar en aspectos como la experiencia de usuario para la mejor optimización de su interacción, un aspecto gráfico más moderno, nuevas opciones a futuro o incluso llegar a presentar tutoriales de uso o inteligencia artificial en beneficio de la usabilidad.

En pruebas adicionales funcionales y no funcionales, los resultados también dieron a entender que, sin importar el fabricante del dispositivo móvil, sea Huawei, Samsung, LG, etc. Desde que su sistema operativo sea Android, el software se ejecuta sin ninguna anomalía. Por otro lado, en cuanto a rendimiento y consumo de memoria se vio utilizado entre el 128 y 192 MB de dicho recurso por lo que está en óptimas condiciones, pero de ser posible puede mejorarse para acortar y hacer peticiones solo en caso de requerirse.

El software probado, testeado y desplegado se limita y no llega a abarcar la totalidad de usuarios debido a su demarcación única para dispositivos Android, permitiendo su uso para la mayoría de los habitantes, sin embargo, si se deseara llegar a aquel público excluido se optaría bien sea por otro desarrollo nativo iOS o un desarrollo híbrido y su proceso de pruebas también se debe de realizar.

En cuanto a la publicación en la tienda digital Google Play, no se observó ningún inconveniente, ni de despliegue ni de descarga e instalación.

Finalmente, se discute que es un software adaptado a la cooperativa, avalado por la misma y cumple con sus necesidades, sin embargo y en las condiciones actuales de evaluación, no puede

llegar a reemplazar ni competir con herramientas masivas como lo son Uber, Cabify o DiDi, que presentan microservicios para una experiencia de usuario satisfactoria.

Conclusiones

- Si bien es cierto la existencia de este tipo de aplicaciones de taxi ya han venido sonando desde un par de años atrás, el requerimiento actual que tenía la cooperativa de la ciudad de Garzón, de un software propio, con un mínimo entregable y garantizando un buen producto, se cumplió a cabalidad hasta el despliegue del software de asistencia de taxi con unos estándares de calidad.
- En el proyecto se fueron tomando acciones para la corrección de errores surgentes que, mediante la metodología de despliegue de servicios múltiples, permitió un mayor control en las versiones que iban siendo modificadas, actualización tras actualización, mediante cada uno de los servicios hasta concluir con la versión 1.5.4. en la web, versión 1.2.12. en la app usuario y versión 1.2.8. en la app taxista, que hasta el momento son considerados los productos finales y lo que se aprobó con la cooperativa.
- En el trabajo se testearon los softwares que conformaban el servicio de taxi (usuario, taxista y web administrativa) y se probó tanto funcionalidades como no funcionalidades las cuales en su momento contaban con fallas y se llevó a cabo un despliegue eficiente en la tienda digital PlayStore donde finalmente los usuarios pueden descargar sin dificultad el software de GTAXI.
- Así mismo, se evaluó y se aseguró la calidad del software con base en la norma ISO/IEC 25000 en la cual se hizo énfasis en las características y subcaracterísticas del modelo y el cumplimiento de estas, donde se obtuvo un grado de satisfacción del 93% (muy satisfactorio). Además, los miembros de la cooperativa probaron el software

interactuando en un entorno real y dando su aval de satisfacción con el proceso llevado a cabo de actualización, adaptación y finalmente el lanzamiento del producto.

- Todas las bases brindadas por la Universidad Surcolombiana en el transcurso de toda la carrera de Ingeniería de Software, nos permitió tener una formación amplia y general de las habilidades y procesos en el desarrollo de software. Esto destacado como una gran ventaja ya que nos otorgó claridad sobre todos los campos de aplicación que tiene la carrera y las variedades de especializaciones con la que cuenta para desarrollar el potencial humano y profesional.
- Para poder concluir el proyecto, se puso en práctica los conocimientos adquiridos en la carrera, como también los nuevos conocimientos adquiridos en la parte investigativa, como lo fue en metodologías de implementación y métricas de calidad.
- Se generaron aportes importantes de parte de la cooperativa y de parte de los estudiantes hasta llegar a una implementación y despliegue conciso de una herramienta que servirá de mucho al municipio, a los usuarios, taxistas vinculados y a la cooperativa COOTRANSGAR.

Trabajos futuros

Con la implementación y desarrollo de este proyecto, se deja abierta la aplicación que puede ser extendida a futuro para:

- Tecnologías que trabajen sin una conexión continua a internet, donde los usuarios puedan realizar peticiones de taxis o inclusive cargas de mapa y guardarlas en un almacenamiento local.
- Implementar el software de usuario y taxista en dispositivos que cuenten con otro sistema operativo distinto al Android como lo son iOS y Windows Phone o inclusive, adaptar la aplicación existente a tal tamaño que otras versiones del propio Android inferiores al 6.0. lo puedan ejecutar.
- Presentar alternativas en caso de accidentalidades como atracos, brindar al usuario un llamado inmediato a las autoridades, botones de emergencia de contacto a seres cercanos y calificación a la cooperativa.
- Buscar, aún más, mejoras en la velocidad y efectividad del software, tanto móvil como web ya que los tiempos de carga pueden, aún más, ser reducidas.
- Con los datos que se van recolectando, dejar la posibilidad de realizar análisis estadísticos con el fin de informar a la cooperativa sobre el funcionamiento y qué mejorar con esos datos.
- Con respecto a calidad del producto, otro trabajo futuro es optar por la certificación de calidad frente a los entes reguladores disponibles de ISO/IEC 25000 como lo es el laboratorio de evaluación AQC LAB.
- En cuanto a la web, se puede complementar con un servicio de consulta de taxis vinculados totales, reportados y que ameritan una sanción superior.

Referencias bibliográficas

Aca, N. (31 de Julio de 2018). *3 razones por las que falla la implementación de proyectos*. Obtenido de https://www.merca20.com/3-razones-por-las-que-falla-la-implementacion-de-proyectos/

Alarcos Quality Center. (s.f.). *Evaluación y mejora de los procesos de software.* Obtenido de Qué es la evaluación y mejora de procesos software: http://www.alarcosqualitycenter.com/index.php/servicios/calidad-procesos-software

Álvarez, G. (5 de Noviembre de 2019). *¿Cómo versionar un software?* Obtenido de [Kyocode]: https://www.kyocode.com/2019/11/como-versionar-software/

Amanda, C. (15 de Mayo de 2018). *Implementación continua: qué es esto y cómo elimina los riesgos de las implementaciones de software*. Obtenido de [CloudBees]: https://www.cloudbees.com/blog/rolling-deployment

Amazon Web Services [AWS]. (s.f.). *Amazon Route 53*. Obtenido de Una forma confiable y rentable de encaminar a los usuarios finales a las aplicaciones de Internet: https://aws.amazon.com/es/route53/

Amazon Web Services [AWS]. (s.f.). *Implementación azul-verde en AWS*. Obtenido de Mediante AWS CodePipeline para implementar en entornos de AWS Elastic Beanstalk: https://aws.amazon.com/es/quickstart/architecture/blue-green-deployment/

Amazon Web Services. (3 de Junio de 2020). *Overview of Deployment Options on AWS.* Obtenido de Blue/Green Deployments: https://docs.aws.amazon.com/whitepapers/latest/overview-deployment-options/bluegreen-deployments.html

Android. (s.f.). *Qué es Android*. Obtenido de La plataforma que está cambiando lo que pueden hacer los móviles.: https://www.android.com/intl/es_es/what-is-android/

Angulo, R. (Enero-Marzo de 2013). *Tecnología Móvil.* Obtenido de Aplicaciones móviles híbridas: lo mejor de dos mundos: https://cmapspublic2.ihmc.us/rid=1NTQ9NMKD-R1SKBP-24M4/Aplicaciones%20moviles%20hibridas-%20lo%20mejor%20de%20dos%20mundos.pdf

Araya Solís, G., Méndez Marín, G., & Jiménez Segura, R. (Julio de 2014). *Pruebas de software para dispositivos móviles android.* Obtenido de UNAN-Managua, Universidad Nacional Autónoma de Nicaragua: http://repositorio.unan.edu.ni/id/eprint/2371

Ares, L. (26 de Octubre de 2017). *Testing: su importancia y los distintos tipos de pruebas de software*. Obtenido de https://visual-engin.com/2017/10/26/importancia-pruebas-de-software-testing/

Artica Navarro, R. L., & Pita Astengo, L. H. (2014). *Desarrollo de aplicaciones móviles.* Obtenido de http://repositorio.unapiquitos.edu.pe/handle/20.500.12737/4515

Baz Alonso, A., Ferreira Artime, I., Álvarez Rodríguez, M., & García Baniello, R. (s.f.). *Dispositivos móviles.* Obtenido de E.P.S.I.G : Ingeniería de Telecomunicación Universidad de Oviedo: https://isa.uniovi.es/docencia/SIGC/pdf/telefonia_movil.pdf

Bigelow, S. J. (Agosto de 2016). *rolling deployment*. Obtenido de [Figura]: https://searchitoperations.techtarget.com/definition/rolling-deployment

Cadenas, R. (22 de Marzo de 2019). *¿QUE NECESITO? ¿WEB APPS, APP NATIVA O APP HÍBRIDA?* Obtenido de https://www.gsoft.es/articulos/que-necesito-web-apps-app-nativa-o-app-hibrida/

Candost's Space. (30 de Enero de 2021). *The Blue-Green Deployment Strategy*. Obtenido de [Figura]: https://candost.blog/the-blue-green-deployment-strategy/

Castro Alvites, A. G. (2019). *Optimización de pruebas de software estructurales usando algoritmos genéticos: revisión sistemática.* Obtenido de https://repositorio.uss.edu.pe/bitstream/handle/20.500.12802/6216/Castro%20Alvites%20Anthony%20Giuseppe.pdf?sequence=1

ChristopherGS. (12 de Abril de 2020). *Deploying Machine Learning Models in Shadow Mode*. Obtenido de [Figura]: https://christophergs.com/machine%20learning/2019/03/30/deploying-machine-learning-applications-in-shadow-mode/

Conversant Technologies. (15 de Noviembre de 2017). *Pruebas de software y análisis de calidad: los elementos esenciales de cualquier desarrollo de sistema infalible*. Obtenido de [Figura]: https://conversantech.com/software-testing-quality-analysis/

Cuello, J., & Vittone, J. (2013). Las aplicaciones. En J. Cuello, & J. Vittone, *Diseñando apps para móviles* (pág. 14). Edición: Catalina Duque Giraldo.

Cuervo, M. C., Alarcón Aldana, A. C., & Álvarez Carreño, A. M. (Enero - Junio de 2017). *Modelos de calidad del software, un estado del arte.* Obtenido de Entramado, 13(1), 236-250: https://dialnet.unirioja.es/servlet/articulo?codigo=6084938

Delía, L., Galdamez, N., Thomas, P., & Pesado, P. (17 de Diciembre de 2013). *Un Análisis Experimental de Tipo de Aplicaciones para Dispositivos Móviles.* Obtenido de Instituto de Investigación en Informática LIDI. Facultad de Informática. Universidad Nacional de La Plata. Argentina: http://sedici.unlp.edu.ar/handle/10915/32397

Enriquez, J. G., & Casas, S. I. (11 de Junio de 2014). *Usabilidad en aplicaciones móviles.* Obtenido de Informes Científicos Técnicos - UNPA, 5(2), 25–47.: https://doi.org/10.22305/ict-unpa.v5i2.71

Estayno, M., Dapozo, G., Cuenca Pletsch, L. R., & Greiner, C. (Mayo de 2009). *Modelos y Métricas para evaluar Calidad de Software.* Obtenido de In XI Workshop de Investigadores en Ciencias de la Computación: http://sedici.unlp.edu.ar/handle/10915/19762

F5. (s.f.). *Rápida implementación azul-verde en cualquier entorno*. Obtenido de https://www.f5.com/es_es/solutions/rapid-blue-green-deployment-in-any-environment

Firebase. (20 de Octubre de 2021). *¿Qué puedes hacer con Firebase Hosting?* Obtenido de https://firebase.google.com/docs/hosting/use-cases?hl=es

Firebase. (s.f.). *Manage live & preview channels, releases, and versions for your site*. Obtenido de [Figura]: https://firebase.google.com/docs/hosting/manage-hosting-resources?hl=cs

Fugaro, L., & Vocale, M. (Enero de 2019). *Hands-On Cloud-Native Microservices with Jakarta EE.* Packt. Obtenido de [Figura]: https://subscription.packtpub.com/book/cloud_and_networking/9781788837866/9/ch09lvl1sec58/a-b-testing-deployment

Giraldo, V. (16 de Abril de 2019). *¿Ya conoces Firebase? La herramienta de desarrollo y análisis de aplicaciones mobile*. Obtenido de https://rockcontent.com/es/blog/que-es-firebase/

Google Developers. (2019). *Firebase*. Recuperado el 30 de Noviembre de 2019, de https://firebase.google.com/docs/firestore?hl=es-419

Google Play. (15 de Mayo de 2019). Obtenido de https://play.google.com/store/apps/details?id=com.gimnasios&showAllReviews=true

Google Play. (s.f.). *Te damos la bienvenida a la página de insignias de Google Play*. Obtenido de [Figura]: https://play.google.com/intl/es-419/badges/

GoogleDevelopers. (2019). *Firebase*. Obtenido de https://firebase.google.com/products/firestore/

Harness. (20 de Octubre de 2021). *Intro To Deployment Strategies: Blue-Green, Canary, And More*. Obtenido de The Basic Deployment: https://harness.io/blog/blue-green-canary-deployment-strategies/

Harness. (20 de Octubre de 2021). *Intro To Deployment Strategies: Blue-Green, Canary, And More*. Obtenido de [Figura]: https://harness.io/blog/blue-green-canary-deployment-strategies/

Harness. (s.f.). *Deployment Concepts and Strategies*. Obtenido de Multi-Service Deployment: https://docs.harness.io/article/325x7awntc-deployment-concepts-and-strategies

Hernández García, D. (1 de Julio de 2007). *ISO9000: Normas voluntarias que se imponen en el mundo.* Obtenido de Lupa Empresarial: https://revistas.ceipa.edu.co/index.php/lupa/article/view/376

Huayta García, M. (Mayo de 2006). *CMMI: Aseguramiento de la calidad.* Obtenido de Ingeniería Industrial, 27(2), 2.

IEEE STD 610-1990. (s.f.). *EEE Standard Computer Dictionary: A Compilation of IEEE Standard Computer Glossaries.* Obtenido de https://standards.ieee.org/standard/610-1990.html

Implementación continua: qué es esto y cómo elimina los riesgos de las implementaciones de software. (15 de Mayo de 2018). Obtenido de CloudBees: https://www.cloudbees.com/blog/rolling-deployment

ISO/IEC 25010. (2011). *Requisitos de calidad y evaluación de sistemas y software (Square).* Obtenido de Sistema y modelos de calidad del software: https://iso25000.com/index.php/normas-iso-25000/iso-25010

ISO/IEC. (s.f.). *ISO/IEC 25010*. Obtenido de [Figura]: https://iso25000.com/index.php/normas-iso-25000/iso-25010

ISO/IEC. (s.f.). *ISO/IEC 25040*. Obtenido de [Figura]: https://iso25000.com/index.php/normas-iso-25000/iso-25040

ISO/IEC. (s.f.). *La familia de normas ISO/IEC 25000*. Obtenido de [Figura]: https://iso25000.com/index.php/normas-iso-25000

López Mora, S. (17 de Mayo de 2020). *Firebase: qué es, para qué sirve, funcionalidades y ventajas*. Obtenido de Funciones de Firebase: https://www.digital55.com/desarrollo-tecnologia/que-es-firebase-funcionalidades-ventajas-conclusiones/

Luján, J. D. (s.f.). *¿Cómo se deciden las versiones del software?* Obtenido de [EDteam]: https://ed.team/blog/como-se-deciden-las-versiones-del-software

ManageEngine. (s.f.). *Implementación de software con Desktop Central*. Obtenido de https://www.manageengine.com/latam/desktop-central/implementacion-de-software.html

Mera Paz, J. A. (19 de Octubre de 2016). *Análisis del proceso de pruebas de calidad de software.* Obtenido de Ingeniería solidaria, 12(20): http://hdl.handle.net/20.500.12494/962

Modelo de Metricas de GILB. (2018). Obtenido de https://es.scribd.com/document/378851018/Modelo-de-Metricas-de-GILB

Morales, K. (2019). *Tipos de Pruebas de Software*. Obtenido de https://platzi.com/blog/tipos-de-pruebas-de-software/

Moreano, P. (16 de Septiembre de 2020). *Canary Deployments and how to sleep peacefully at night*. Obtenido de [Figura]: https://www.kushki.com/en/blog/deployments-tipo-canary-y-como-dormir-tranquilos-por-las-noches

Muradas, Y. (22 de Junio de 2021). *Qué es Firebase: Conoce la plataforma de Google*. Obtenido de https://openwebinars.net/blog/que-es-firebase-de-google/

Oltra Badenes, R. F. (5 de Julio de 2017). *La norma ISO/IEC 20000. Finalidad y.* Obtenido de https://riunet.upv.es/handle/10251/84477

OWIUS. (8 de Febrero de 2019). *¿Porqué puede fracasar un proyecto software? Y como evitarlo*. Obtenido de https://owius.com/porque-puede-fracasar-un-proyecto-software-y-como-evitarlo/

Paola, R. M., Morales, C., & Gutiérrez, P. (2015). *Norma ISO/IEC 25000.* Obtenido de Tecnología Investigación y Academia, 3(2), 27–33: https://revistas.udistrital.edu.co/index.php/tia/article/view/8373

Pesado, P. M., Bertone, R., Ramón, H., Pasini, A., Esponda, S., Alonso, L., & Martorelli, S. (Junio de 2006). *Calidad en el desarrollo de Sistemas de Software.* Obtenido de In Workshop de Investigadores en Ciencias de la Computación (Vol. 8): https://digital.cic.gba.gob.ar/handle/11746/3757

PMO Informática. (29 de Enero de 2014). *Tipos de pruebas de software definidos por el ISTQB.* Obtenido de http://www.pmoinformatica.com/2014/01/tipos-de-pruebas-de-software-istqb.html

Puetate, G., Ibarra, J., Villamagua Portilla, O., Garcés, G., & Ibarra, P. S. (s.f.). *Aplicaciones Móviles Híbridas.* Obtenido de https://www.pucesi.edu.ec/webs2/wp-content/uploads/2021/02/Aplicaciones-M%C3%B3viles-H%C3%ADbridas-2020.pdf

Ranieri, J., Villar, S., & Rodríguez, Á. (s.f.). *Sistemas Operativos.* Obtenido de https://www.academia.edu/43039119/SistemasOperativos_JoaoRanieri_AlvaroRodriguez_SergioVillar

Red Hat OpenShift. (s.f.). *Advanced Deployment Strategies.* Obtenido de A/B Deployment: https://docs.openshift.com/container-platform/3.7/dev_guide/deployments/advanced_deployment_strategies.html#advanced-deployment-strategies-blue-green-deployments

RedHat OpenShift. (s.f.). *Deployment Strategies.* Obtenido de https://docs.openshift.com/container-platform/3.7/dev_guide/deployments/deployment_strategies.html

Robinson, S. (21 de Agosto de 2018). *Query a custom AutoML model with Cloud Functions and Firebase*. Obtenido de [Figura]: https://hackernoon.com

Sahoo, J. (11 de Mayo de 2021). *Five Advanced Deployment Strategies to Consider under DevOps Methodology*. Obtenido de Shadow Deployment Strategy: https://www.opsmx.com/blog/advanced-deployment-strategies-devops-methodology/#:~:text=A%20Shadow%20deployment%20strategy%20is,successful%20traffic%20and%20load%20testing.

Sánchez Jaya, V. L., Robayo Jácome, D. J., & López Sevilla, G. M. (2018). *APUNTES TEÓRICOS SOBRE MODELOS DE EVALUACIÓN DE CALIDAD EN PROCESOS DE DESARROLLO DE SOFTWARE PARA PERSONAS NO VIDENTES.* Obtenido de [Figura]: https://repositorio.pucesa.edu.ec/handle/123456789/2709.

Sicilia, M. Á. (2007). *Estándar ISO 9126 del IEEE y la mantenibilidad.* Obtenido de http://garciagregorio.webcindario.com/ms/iso9126.pdf

SoftExpert. (s.f.). *SoftExpert EQM Gestión de la Calidad Empresarial*. Obtenido de https://www.softexpert.com/es/solucao/gestion-calidad-empresarial-eqm/

SoftExpert. (s.f.). *SoftExpert ExcellentSuite Simplemente innovadora*. Obtenido de https://www.softexpert.com/es/solucao/softexpert_excellence_suite/

SOLBYTE. (21 de Julio de 2021). *ipos de aplicaciones móviles: nativas, webs, híbridas*. Obtenido de https://www.solbyte.com/blog/tipos-de-aplicaciones-moviles-nativas-webs-hibridas/

Split. (s.f.). *Despliegue Azul/Verde*. Obtenido de [Glosario]: https://www.split.io/glossary/blue-green-deployment/

Sumo Logic. (s.f.). *Términos del glosariode DevOps y seguridad*. Obtenido de Despliegue azul-verde: https://www.sumologic.com/glossary/blue-green-deployment/

SWEBOK. (2004). *Guide to the Software Engineering Body of Knowledge (SWEBOK Guide).* Obtenido de version ieee Computer pp. 10-40: https://www.computer.org/web/swebok/v3

Tremel, E. (21 de Noviembre de 2017). *Six Strategies for Application Deployment*. Obtenido de [Figura]: https://thenewstack.io/deployment-strategies/

Tremel, E. (21 de Noviembre de 2017). *Six Strategies for Application Deployment*. Obtenido de [web]: https://thenewstack.io/deployment-strategies/

Universidad Abierta y a Distancia de México. (Mayo de 2016). *Modelos de calidad de software.* Obtenido de Unidad 3. Modelos de calidad de software. Ingeniería en Desarrollo de Software 6º Semestre: https://rockflood.files.wordpress.com/2016/05/unidad_3_modelos_de_calidad_de_software.pdf

Vargas, C. (s.f.). *Tipos de pruebas funcionales para el aseguramiento de la calidad*. Obtenido de [Figura]: https://trycore.co/transformacion-digital/tipos-de-pruebas-funcionales/

Vargas, C. (s.f.). *Tipos de pruebas funcionales para el aseguramiento de la calidad*. Obtenido de [Figura]: https://trycore.co/transformacion-digital/tipos-de-pruebas-funcionales/

WORLDVECTORLOGO. (s.f.). *Descargar Powered By Android vector logotipo en SVG*. Obtenido de [Figura]: https://worldvectorlogo.com/es/logo/powered-by-android-1/.