

**Version 1.0**

DESCARGARPORTADAS.COM



**Plan de Pruebas**

2020

**Aplicación Móvil Tacna Food and Drinks**

Revisión Histórica

|  |  |  |  |
| --- | --- | --- | --- |
| **Fecha** | **Versión** | **Descripción** | **Autor** |
| 28/11/2020 | 1.0 | Primera realización del Plan de Pruebas | Marko Rivas Rios  Ronald Ordoñez Quilli |

1. Introducción
   1. **Propósito**

Este documento tiene como propósito establecer las técnicas, herramientas y actividades relacionadas con la ejecución y validación del plan de pruebas; incluye responsabilidades de cada una de las tareas, los recursos y los requisitos que deben se considerados en el esfuerzo de las pruebas, en la búsqueda de garantizar el cumplimiento de los requerimientos planteados en el marco del desarrollo del proyecto “Tacna Food and Drinks”.

Este documento, específicamente:

* Identifica información del proyecto y componentes de software que pueden ser probados
* Los tipos de prueba a ser implementados
* Recomendaciones y descripciones de estrategias de prueba a ser empleadas.
  1. **Alcance**

Este documento de Plan de Pruebas se convierte en una guia para desarrollar de forma organizada las diferentes actividades que se realizarán en el proceso de las pruebas durante del desarrollo del proyecto “Tacna Food and Drinks”.

* 1. **Definciones, acrónimo y abreviaturas**
* Plan de pruebas: Describe todos los métodos que se utilizaran para verificar que el aplicativo satisface la especificación del producto y las necesidades del cliente. Incluye los objetivos de calidad, necesidades de recursos, cronograma, métodos, etc.
* Casos de prueba: Lista los ítems específicos que serán probados y describe los pasos detallados que serán seguidos para verificar el aplicativo.
* Reporte de pruebas: Describen los problemas o errores encontrados al ejecutar los casos de prueba.
  1. **Referencias**
* Alcance del proyecto
* Documento de especificación de requisitos
* Casos de uso
* Descripción de casos de uso
* Diagrama de clases

1. **Misión de Evaluación**

**2.1. Contexto y antecedentes del proyecto**

Debido a la coyuntura actual del país y la realidad de la ciudad de Tacna en cuanto a los establecimientos de comida y bebidas, las ventas han bajado y el gobierno ha estipulado protocolos sanitarios que el establecimiento tiene que cumplir para poder seguir con su funcionamiento por medio de Delivery.

Existe la ausencia de una plataforma en Tacna que centre la información de los restaurantes en un solo sitio y que sea capaz de realizar pedidos con pagos integrados y seguimiento del mismo. Es por esto que Tacna Food and Drinks permitirá a la población tener una perspectiva de la información de los distintos establecimientos de comida, realizar compras, realizar los pagos y ver el seguimiento de los pedidos, lo cual supondrá una mejora para que tanto los administradores de los establecimientos de comida y los clientes puedan realizar sus intereses de forma "rápida y eficaz"; como también visualizar el cumplimiento de los protocolos sanitarios establecidos por el gobierno ante situaciones como la situación actúa. La mejora será a base de interfaces graficas sencilla y amigables.

El sistema permite a los usuarios acceder a la plataforma a través de una aplicación móvil, de forma rápida y sencilla y sin necesidad de intermediarios.

**2.2. Misiones de evaluación aplicables al proyecto**

Las misiones para el esfuerzo de prueba y evaluación sobre el alcance de este plan son:

* Encontrar tantos errores como sea posible
* Verificar las especificaciones de software
* Asegurar la calidad del producto, satisfacer a las partes interesadas

1. **Necesidades ambientales**

En esta sección se establecen los recursos del sistema necesarios para el plan de prueba.

**3.1. Hardware base**

|  |  |
| --- | --- |
| **Aplicación** | **Recurso** |
| Tacna F&D | GPS |
| Tacna F&D Delivery | GPS |

**3.2. Software base en el entorno de prueba**

|  |  |
| --- | --- |
| **Nombre del elemento de software** | **Versión** |
| Android | 6.0 |

**3.3. Herramientas**

|  |  |
| --- | --- |
| **Herramienta** | **Versión** |
| jUnit | 4.12 |

1. **Responsabilidades y personal**

|  |  |  |
| --- | --- | --- |
| **Recursos Humanos** | | |
| **Rol** | **Responsable** | **Responsabilidad** |
| Gerente de pruebas | Ronald Ordoñez | Proporciona supervisión de la administración |
| Analista de pruebas | Ronald Ordoñez | Identifica y define las pruebas específicas que se van a realizar |
| Diseñador de prueba | Marko Rivas  Ronald Ordoñez | Define el enfoque técnico para la implementación del esfuerzo de prueba. |
| Tester | Marko Rivas | Implementa y ejecuta las pruebas. |
| Administrador de base de datos | Marko Rivas  Ronald Ordoñez | Garantiza que el entorno y los activos de datos de prueba (base de datos) se administran y mantienen. |
| Implementador | Marko Rivas  Ronald Ordoñez | Implementa y prueba unitaria las clases de prueba y los paquetes de prueba. |

1. **Enfoque de prueba**

Como herramienta principal se utilizará jUnit, el cual es un marco simple para escribir pruebas unitarias.

**5.1. Medición y evaluación de la extensión de pruebas**

Se evaluará mediante 3 criterios de medición:

* **Errores graves:** Información crítica presentada erróneamente, información mal registrada en la base de datos, caídas de programa, incumplimiento de principales objetivos funcionales.
* **Errores Medios:** Errores en presentación de datos primarios, incumplimiento de objetivos funcionales secundarios, caídas de programas secundarios.
* **Errores Leves:** Errores en presentación de datos secundarios, no adecuación a estándares, comportamiento correcto pero diferentes en situación similares, dificultades de operación.

**5.2. Identificación de pruebas**

* Inventario de pruebas priorizado.
* Estimación de esfuerzo de cada funcionalidad.
* Plazos previstos para la solución
* Criterios de terminación.

**5.3. Iteraciones**

Se aplicarán las pruebas en 3 iteraciones:

* Primera iteración: Tacna F&D Business
* Segunda iteración
* Tercera iteración

**5.4. Realización de pruebas**

**Junit:**

JUnit es un framework de pruebas unitarias para el lenguaje de programación Java. Es decir, JUnit se sitúa en la fase de pruebas dentro del ciclo de Ingeniería del Software. Desempeña un papel crucial en el desarrollo impulsado por pruebas y es una familia de marcos de pruebas unitarias conocidos colectivamente como xUnit.

JUnit promueve la idea de "primero probar y luego codificar", que enfatiza la configuración de los datos de prueba para un fragmento de código que se puede probar primero y luego implementar. Este enfoque es como "probar un poco, codificar un poco, probar un poco, codificar un poco". Aumenta la productividad del programador y la estabilidad del código del programa, lo que a su vez reduce el estrés del programador y el tiempo dedicado a depurar.

* Proporciona afirmaciones para probar los resultados esperados.
* Las pruebas JUnit le permiten escribir códigos más rápido, lo que aumenta la calidad.
* Las pruebas JUnit se pueden ejecutar automáticamente y verifican sus propios resultados y brindan retroalimentación inmediata. No es necesario revisar manualmente un informe de resultados de prueba.
* Las pruebas de JUnit se pueden organizar en conjuntos de pruebas que contienen casos de prueba e incluso otros conjuntos de pruebas.

JUnit muestra el progreso de la prueba en una barra que es verde si la prueba se está ejecutando sin problemas y se vuelve roja cuando falla una prueba.

**Pruebas de caja negra**

Las pruebas de caja negra se definen como una técnica de testing en la que se prueba la funcionalidad de una aplicación ignorando la parte interna de dicha aplicación. Esto quiere decir que se obvia la estructura del código, la arquitectura, los detalles relacionados con la implementación de los diferentes módulos, paquetes o rutas en la que se compone el código. Este tipo de prueba se basan por completo en en los requisitos de la aplicación y en sus especificaciones técnicas.

En las pruebas de caja negra el Tester solo se centra en las entradas y salidas de la aplicación, sin preocuparse por el contenido interno. Lo que pase por dentro es indiferente, solo importa que, si se realiza cierta acción, la salida sea la indicada según los requerimientos.

Este tipo de prueba se aplican a cualquier tecnología que se desee. Por ejemplo, se puede realizar pruebas de caja negra en Software de escritorio, en aplicaciones web o incluso en aplicaciones móviles. Cualquier software se podría probar simplemente enfocándose en las entradas y salidas sin conocer las implementaciones del código interno. Lo que lo hace unas de las pruebas más realizadas a nivel profesional.

**Pruebas unitarias**

Las pruebas unitarias consisten en aislar una parte del código y comprobar que funciona a la perfección. Son pequeños tests que validan el comportamiento de un objeto y la lógica.

El unit testing suele realizarse durante la fase de desarrollo de aplicaciones de software o móviles. Normalmente las llevan a cabo los desarrolladores, aunque en la práctica, también pueden realizarlas los responsables de QA. Con ellas se detectan antes errores que, sin las pruebas unitarias, no se podrían detectar hasta fases más avanzadas como las pruebas de sistema, de integración e incluso en la beta. Realizar pruebas unitarias con regularidad supone, al final, un ahorro de tiempo y dinero.

* Las pruebas unitarias demuestran que la lógica del código está en buen estado y que funcionará en todos los casos
* Aumentan la legibilidad del código y ayudan a los desarrolladores a entender el código base, lo que facilita hacer cambios más rápidamente.
* Los test unitarios bien realizados sirven como documentación del proyecto.
* Se realizan en pocos milisegundos, por lo que podrás realizar cientos de ellas en muy poco tiempo.

**Formato de caso de prueba**

|  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **INFORMACIÓN GLOBAL DEL CASO DE PRUEBA** | | | | | | | | | | | |
| **CASO DE PRUEBA No.** | | *<Número del caso de prueba constituido [número del caso de uso]-[Numero del caso de prueba]>* | | | | **VERSIÓN DE EJECUCIÓN** | | | | | *<Versión diligenciado por el analista de pruebas en el momento de ejecutarla. Este número se incrementa de 1 en 1>* |
| **FECHA EJECUCIÓN** | | | | | *<Fecha de ejecución diligenciado por el analista de pruebas>* |
| **CASO DE USO:** | | *<Identificación del caso de uso objeto de la prueba>* | | | | **MODULO DEL SISTEMA** | | | | | *<Nombre del modulo al que corresponde el caso de uso objeto de la prueba>* |
| **Descripción del caso de prueba:** | | *<Descripción de lo que se pretende probar en el caso de prueba>* | | | | | | | | | |
| 1. **CASO DE PRUEBA** | | | | | | | | | | | |
| 1. **Precondiciones** | | | | | | | | | | | |
| *<Lista de precondiciones que deben cumplirse para realizar la prueba>* | | | | | | | | | | | |
| 1. **Pasos de la prueba** | | | | | | | | | | | |
| *<Pasos secuenciales que deben ser ejecutados por el analista de pruebas o usuario, ante el sistema para ejecutar la prueba>* | | | | | | | | | | | |
| **DATOS DE ENTRADA** | | | | **RESPUESTA ESPERADA DE LA APLICACIÓN** | | | **COINCIDE** | | **RESPUESTA DEL SISTEMA** | | |
| **CAMPO** | **VALOR** | | **TIPO ESCENARIO** | **SI** | **NO** |
| *<Descripción del dato de entrada>* | *<Valor que debe ser suministrado en la prueba para el dato de entrada>* | | *<Tipo de escenario que pretende probarse: Correcto/Incorrecto>* | *<Respuesta que se espera de la aplicación>* | | |  |  | *<Respuesta que se obtuvo de la aplicación en el momento de la ejecución de la prueba>* | | |
|  |  | |  |  | | |  |  |  | | |
|  |  | |  |  | | |  |  |  | | |
|  |  | |  |  | | |  |  |  | | |
|  |  | |  |  | | |  |  |  | | |
|  |  | |  |  | | |  |  |  | | |
| 1. **Post condiciones** | | | | | | | | | | | |
| *<Lista de pos condiciones que deben cumplirse después de realizar la prueba>* | | | | | | | | | | | |
| 1. **RESULTADOS DE LA PRUEBA** | | | | | | | | | | | |
| **Defectos y desviaciones** | | | | | | | | | | **Veredicto** | |
| *<Lista de defectos o desviaciones encontrados por el analista o usuario al ejecutar la prueba>* | | | | | | | | | | * **Pasó** * **Falló** | |
| **Observaciones** | | | | | **Probador** | | | | | | |
| *<Observaciones generales del analista o usuario sobre la ejecución de la prueba>* | | | | |  | | | | | | |
| **Firma:**  **Nombre:**  **Fecha:** | | | | | | |

1. **Riesgos del plan**

|  |  |  |
| --- | --- | --- |
| **Riesgo** | **Estrategia de mitigación** | **Contingencia** |
| Tiempo reducido asignado a la fase de pruebas. | Preparación de las pruebas que se realizaran y asignar un orden por prioridades. | Contratar nuevo personal de test o realizar pruebas de los requerimientos de mayor prioridad. |
| Falta de los recursos necesarios para ejecutar las pruebas | Disponer de recursos necesarios para realizar la prueba y disponer de una cartera de contactos para alquiler de equipos. | Llamar y alquilar equipo necesario de la cartera de contacto. |
| Falta de coordinación entre los equipos de desarrollo y testing | Incentivar al trabajo colaborativo y asignar responsabilidades a los miembros del proyecto. | Revisar el cronograma de actividades y verificar la responsabilidad de la actividad. |

1. **Procesos y procedimientos de Gestión**

**7.1. Evaluación de los resultados de este plan de pruebas**

|  |  |
| --- | --- |
| Aplicación Móvil Tacna Food and Drinks | Se aprobará el proyecto con un 100% de las pruebas ejecutadas, pero con un 90% de aceptación. Esto quiere decir el 90% de las pruebas deben ser exitosas y sin errores. El restante 10% pueden existir errores medios o bajos, pero no graves. En caso de ocurrir que el proyecto no cumpla con el nivel exigido, el proyecto se rechaza completo en su etapa de certificación. |
| PROCESO Ciclo de Desarrollo de Software |
| Documento Plan Maestro de pruebas (PMP) |