**Plan de Pruebas**

1. **INFORMACIÓN DEL PROYECTO**

**Nombre: Marketplace Financiero (VIT)**

1. **OBJETIVO**

Evaluar el rendimiento, estabilidad y capacidad de respuesta de los diferentes componentes críticos de la aplicación móvil bajo condiciones de carga extrema y picos de usuarios concurrentes, asegurando que el sistema:

* Mantenga la integridad de las transacciones financieras.
* Responda adecuadamente a solicitudes masivas de autenticación y consulta.
* Entregue notificaciones push sin retrasos significativos.
* Soporte actualizaciones de versión sin afectar la experiencia del usuario ni causar interrupciones.

1. **ALCANCE DE LAS PRUEBAS**

**Dentro del Alcance**

**Fuera del Alcance**

1. **COMPONENTES A EVALUAR**
2. **Autenticación (Inicio de Sesión):**
   * Mecanismo de login y validación de credenciales.
   * Protección contra ataques de fuerza bruta.
   * Bloqueo de cuentas o IP tras intentos fallidos.
   * Gestión de sesiones y tokens.
3. **Transacciones Financieras:**
   * Flujos de transferencias y pagos.
   * Validación de integridad de datos (sin duplicidad ni pérdida de transacciones).
   * Tiempo de procesamiento de transacciones.
   * Resiliencia de la base de datos durante picos de carga.
4. **Consulta de Saldo (API):**
   * Respuesta de la API bajo múltiples solicitudes de consulta de saldo.
   * Rendimiento de la base de datos durante operaciones de solo lectura.
5. **Notificaciones Push:**
   * Entrega de notificaciones masivas a usuarios concurrentes.
   * Tiempos de latencia en la entrega de notificaciones.
   * Impacto en el rendimiento del servidor durante la entrega.
6. **Actualización de la Aplicación (Deployment):**
   * Capacidad del sistema para manejar solicitudes mientras se realiza una actualización de versión.
   * Impacto de los despliegues en el tiempo de respuesta de la aplicación.
   * Minimización de tiempos de inactividad durante las actualizaciones.
7. **ESTRATEGIA DE PRUEBAS**
8. **Herramientas recomendadas**:
   * JMeter: Autenticación masiva y transacciones financieras.
   * LoadRunner: Pruebas distribuidas para consultas de saldo.
   * BlazeMeter: Notificaciones push.
   * NeoLoad: Pruebas durante actualizaciones.
9. **Entornos de prueba**:
   * **Servidor**: Réplica del entorno de producción.
   * **Base de datos**: Conjunto de datos anonimizados.
10. **Escenarios de carga**:
    * **Carga esperada**: Simulación de 10% de usuarios diarios activos simultáneamente (~10,000 usuarios).
    * **Carga pico**: Simulación de 20%-30% de usuarios (~20,000-30,000 usuarios).
    * **Sobrecarga**: Simulación de 200% de la carga prevista (~100,000 usuarios).
11. **Escenarios Cubiertos:**

**1. Autenticación (Inicio de Sesión):**

* Inicio de sesión exitoso con carga alta.
* Intentos fallidos repetidos (fuerza bruta).
* Saturación del servidor (200% de carga).

**2. Transacciones Financieras:**

* Pagos masivos simultáneos.
* Transferencias entre cuentas con usuarios concurrentes.
* Simulación de inconsistencias en transacciones bajo carga extrema.

**3. API de Consulta de Saldo:**

* Consulta simultánea de saldo por miles de usuarios.
* Saturación de la base de datos (lectura masiva).
* Tiempos de respuesta bajo carga esperada y extrema.

**4. Notificaciones Push:**

* Envío masivo de notificaciones durante eventos críticos.
* Carga de notificaciones durante promociones o mantenimientos.
* Evaluación del retraso y fallos en la entrega.

**5. Actualización de la Aplicación:**

* Despliegue de una nueva versión durante carga pico.
* Evaluación de tiempo de inactividad durante la actualización.
* Simulación de pérdida de conexión durante el despliegue.

1. **Recursos y Roles**

* **QA Team**: Configuración y ejecución de pruebas.
* **DBA**: Monitoreo del rendimiento del servidor durante las pruebas.
* **Architecture Team**: Resolver cuellos de botella detectados.

1. **Cronograma**

|  |  |  |
| --- | --- | --- |
| Actividad | Responsable | Tiempo |
| *Configuración del entorno* | QA/BE | TPD |
| *Configuración de herramientas* | QA | TPD |
| *Creación de los scripts de prueba para cada escenario* | QA | TPD |
| *Ejecución de pruebas de autenticación* | QA | TPD |
| *Ejecución de pruebas de transacciones* | QA | TPD |
| *Ejecución de consultas de saldo* | QA | TPD |
| *Ejecución de notificaciones push* | QA | TPD |
| *Ejecución de pruebas de actualización* | QA | TPD |
| *Análisis y reporte de resultados* | QA/BE | TPD |

1. **CRITERIOS DE ACEPTACIÓN**

**Prueba de Sobrecarga de Solicitudes de Inicio de Sesión (Autenticación)**

1. El sistema debe manejar al menos 10,000 usuarios concurrentes con un tiempo de respuesta promedio inferior a 2 segundos.
2. La tasa de error en la autenticación no debe superar el 5%.
3. Los mecanismos de seguridad (como bloqueo temporal de cuentas) deben activarse correctamente tras múltiples intentos fallidos.

**Prueba de Sobrecarga en Transacciones Financieras**

1. El sistema debe procesar al menos 5,000 transacciones concurrentes con un tiempo promedio inferior a 3 segundos por transacción.
2. No deben generarse inconsistencias en los datos, como duplicaciones o pérdidas de transacciones.
3. La tasa de éxito en las transacciones debe ser igual o superior al 95%.

**Prueba de Sobrecarga de Solicitudes a la API de Consulta de Saldo**

1. El sistema debe responder al 95% de las solicitudes en menos de 2 segundos.
2. El uso de CPU y memoria en el servidor debe mantenerse por debajo del 80% de la capacidad total.
3. No debe haber errores en las consultas de saldo superiores al 3%.

**Prueba de Sobrecarga de Notificaciones Push**

1. El sistema debe enviar notificaciones a al menos 95% de los usuarios en menos de 5 minutos.
2. La latencia promedio en la entrega no debe superar los 2 segundos por notificación.
3. No debe haber un impacto significativo en el rendimiento general del servidor (CPU y RAM no deben superar el 85%).

**Prueba de Sobrecarga Durante una Actualización de la Aplicación**

1. El tiempo de inactividad del servicio no debe superar los 2 minutos (si aplica).
2. El sistema debe permitir que al menos el 90% de las interacciones con la versión actual sean exitosas durante la actualización.
3. No debe haber pérdida de datos ni interrupciones graves en el servicio mientras se implementa la nueva versión.

**7. DEPENDENCIAS**

**Dependencias Técnicas**

**Infraestructura**

* **Endpoint funcional**:
  + El backend debe garantizar que el endpoint de autenticación esté configurado y accesible desde el entorno de pruebas.
  + Especificar el método HTTP (POST, GET, etc.), parámetros requeridos, y cualquier cabecera necesaria (como Content-Type).
* **Ambiente de pruebas (Testing Environment)**:
  + Debe existir un entorno de pruebas (staging o QA) que replique la configuración de producción para garantizar resultados representativos.
  + Asegurar que el backend esté aislado para no impactar a los usuarios reales.
* **Acceso a Logs del Servidor**: Para monitorear errores, tasas de respuesta y posibles bloqueos durante las pruebas.

**Configuración de Herramientas**

* **JMeter**:
  + Instalación de Apache JMeter en un sistema con recursos adecuados para simular la cantidad de usuarios planificada.
  + Configuración de plugins (como gráficos avanzados o integración con monitores).
* **Herramientas de monitoreo**:
  + Asegurar que haya herramientas en el backend para monitorear el uso de recursos (CPU, RAM, disco, red) durante las pruebas.
  + Ejemplos: Grafana, Prometheus, New Relic.

**Dependencias de Datos**

**Usuarios**

* **Listado de usuarios válidos**: Un archivo (CSV o JSON) con credenciales válidas (username, password) proporcionado por backend.
* **Reglas de autenticación**: Documentación que explique el comportamiento esperado (e.g., bloqueo tras X intentos fallidos). Tokens o claves necesarias si hay autenticación adicional (como OAuth).

**Datos adicionales**

* **Cargas esperadas y límites**:
  + Backend debe proporcionar métricas como:
    - Número esperado de usuarios concurrentes en producción.
    - Número máximo de usuarios que el sistema puede manejar antes de degradarse.
* **Datos sensibles**: Si se usan datos reales, garantizar que estén anonimizados para cumplir con regulaciones de privacidad.

**Dependencias Organizacionales**

**Equipo de Desarrollo y Backend**

* **Soporte en tiempo real**: El equipo de backend debe estar disponible para atender incidentes durante las pruebas y garantizar que las configuraciones sean las correctas.
* **Sincronización**: Confirmar con el equipo de backend que no habrá despliegues u otros procesos que puedan interferir durante las pruebas.

**8. REPORTE DE RESULTADOS**

Los resultados deben incluir:

* Gráficas de tiempo de respuesta y uso de recursos.
* Identificación de cuellos de botella.
* Validación de mecanismos de seguridad (bloqueos temporales).