Nota: Algunos de los puntos están en respuesta de investigación mas no de experiencia ya que no he tenido la oportunidad de estar en esas áreas o proyectos.

1. **Proceso de desarrollo de software utilizando metodologías tradicionales y ágiles:**
   1.1. **Metodologías Tradicionales:** Enfoque secuencial, conocido como cascada. Implica una planificación detallada al principio del proyecto, seguida por diseño, implementación, pruebas y mantenimiento en fases sucesivas y lineales.
   1.2. **Metodologías Ágiles:** Enfoque iterativo e incremental, donde se prioriza la entrega continua de software funcional. Se adapta a cambios y retroalimentación constante del cliente. Métodos como Scrum, Kanban, entre otros.
2. **Rol del analista de calidad de software en proyectos ágiles:**

El analista de calidad en proyectos ágiles actúa como parte del equipo, colaborando en la definición de historias de usuario, criterios de aceptación y pruebas. Se enfoca en asegurar la calidad del producto mediante pruebas continuas, automatización, y facilitando la comunicación entre desarrolladores y clientes.

3. **Diferencias principales:**
   3.1. **Verificar y Validar:** Verificar se refiere a revisar si se construyó correctamente el producto, mientras que validar implica asegurar que se construyó el producto correcto.
   3.2. **Issue y Bug:** Un issue es cualquier problema que afecta el progreso del proyecto, mientras que un bug es un defecto en el software que produce un comportamiento no deseado.
   3.3. **Novedad y Hallazgo:** Una novedad es cualquier cambio nuevo o adición al sistema, mientras que un hallazgo es cualquier observación significativa identificada durante las pruebas.
4. **Entregables de etapas:**
   4.1. **Planeación:** Plan de pruebas, estrategia de pruebas.
   4.2. **Diseño de Escenarios y Casos de Prueba:** Casos de prueba, matrices de trazabilidad.
   4.3. **Ejecución:** Resultados de pruebas, registros de ejecución.
   4.4. **Reporte de Hallazgos:** Informe de defectos, seguimiento de acciones correctivas.
   4.5. **Feedback:** Retroalimentación del cliente, lecciones aprendidas.
5. **Herramientas de gestión de pruebas:**

Las herramientas que he utilizado hasta ahora son Jira y Jodiz, para el seguimiento de las incidencias encontradas.

6. **Eventos Scrum:**

Ejemplos de eventos en Scrum incluyen Sprint Planning, Daily Standup, Sprint Review, Sprint Retrospective. Cada evento tiene un propósito específico para la planificación, seguimiento y mejora continua del trabajo del equipo.

7. **Automatización y Programación Orientada a Objetos (POO):**
   - 7.1. **Pilares de POO:** Encapsulamiento, herencia, polimorfismo y abstracción. Al automatizar pruebas, estos pilares son fundamentales para crear un diseño de automatización flexible, mantenible y reutilizable.
   - 7.2. **Principios SOLID:** Los principios SOLID (Single Responsibility, Open/Closed, Liskov Substitution, Interface Segregation, Dependency Inversion) son fundamentales para escribir código de calidad en POO, lo que también aplica a la automatización de pruebas.
   - 7.3. **Escenarios críticos:** Se definen basándose en la importancia del flujo de trabajo o función que se está probando, su frecuencia de uso y el impacto en los usuarios finales.
   - 7.4. **Pirámide de Automatización de Cohn:** Propuesta por Mike Cohn, la pirámide sugiere que las pruebas unitarias deben ser la base de la pirámide, seguidas de pruebas de integración y luego pruebas de interfaz de usuario, con menor cantidad de pruebas de extremo a extremo.
8. **Servicios:**
   - 8.1. **¿Qué es un servicio?:** Un servicio es una unidad de funcionalidad lógica que se expone a través de una interfaz y se accede de forma remota.
   - 8.2. **Diferencia entre SOAP y REST:** SOAP es un protocolo de comunicación basado en XML, mientras que REST es un estilo arquitectónico que utiliza URIs, métodos HTTP y representaciones de recursos para la comunicación.
   - 8.3. **Métodos HTTP:** GET se utiliza para recuperar datos, POST para enviar datos, PUT para actualizar datos y DELETE para eliminar datos.
   - 8.4. **Herramientas:** Postman.
   - 8.5. **Automatización de servicios:** Se puede automatizar utilizando frameworks como RestAssured o librerías de pruebas como Requests en Python.
   - 8.6. **Validaciones:** Validaciones comunes incluyen comprobaciones de estado HTTP, comparaciones de valores de respuesta, validación de estructura JSON/XML, entre otras.
9. **Pruebas No Funcionales:**

9.1. **Explique las más utilizadas:** Seguridad y Performance son dos de las más utilizadas. La seguridad se enfoca en proteger el sistema contra amenazas, mientras que la performance se enfoca en la capacidad de respuesta y eficiencia del sistema bajo carga.

9.2. **Herramientas:** Para seguridad, pueden incluir OWASP ZAP, Burp Suite "No he tenido la oportunidad de hacer este tipo de pruebas de seguridad". Para performance, JMeter.

9.3. **Pruebas de carga:** Se realizan cargas simultaneas esperadas o incluso superiores a las esperadas para evaluar el rendimiento del sistema bajo estrés.

10. Automatización de Móviles:

10.1. **Explique los siguientes conceptos:** apps nativas, apps híbridas, tipos de sistemas operativos.

10.1.1. **App nativas:** son aplicaciones desarrolladas específicamente para una plataforma.

10.1.2. **App híbridas:** son aplicaciones que combinan elementos nativos y web.

10.2. **Diferencias con Selenium:** Selenium está diseñado para aplicaciones web, mientras que Appium es específico para aplicaciones móviles, proporcionando acceso a elementos de la interfaz de usuario de las aplicaciones móviles.

10.3. **Granjas de dispositivos móviles:** Son entornos que alojan múltiples dispositivos móviles para realizar pruebas en diferentes configuraciones y sistemas operativos, por ahora no he trabajado en ninguna o escuchado.

11. Seguridad:

11.1. **Conceptos de seguridad:** Incluyen autenticación, autorización, cifrado, protección contra ataques (como XSS, SQL injection), entre otros.

11.2. **Herramientas:** OWASP ZAP, Burp Suite, AppScan, "No he utilizado ninguna de estas herramientas ya que no he trabajado en esta área".

12. Tecnología Cloud:

12.1. **Proyectos Cloud:** Proyectos que utilizan servicios y recursos de computación en la nube para almacenamiento, procesamiento de datos y ejecución de aplicaciones, "Por ahora no he trabajado en proyectos cloud".

12.2. **Herramientas:** AWS (Amazon Web Services), Microsoft Azure, Google Cloud Platform (GCP), entre otras, "Actualmente estoy haciendo un curso de AWS".

13. Integración Continua:

13.1. **Tareas de Integración y Despliegue Continuo:** Integración continua implica la fusión regular de cambios de código en un repositorio compartido, mientras que el despliegue continuo implica la entrega automatizada de cambios a un entorno de producción.

13.2. **Git:** Sistema de control de versiones distribuido. Comandos comunes incluyen git add, git commit, git push, git pull. GitHub y GitLab son plataformas populares de alojamiento de repositorios Git.

13.3. **Proceso con Jenkins:** Jenkins es una herramienta de integración continua que automatiza el proceso de compilación, pruebas y despliegue de aplicaciones. Se configuran pipelines que especifican las acciones a realizar en cada etapa del proceso.

13.4. **Proceso con Azure DevOps:** Azure DevOps ofrece herramientas para la planificación, desarrollo, pruebas y entrega de software. El proceso de integración continua con Azure DevOps incluye la configuración de pipelines de compilación y despliegue utilizando Azure Pipelines.