* Ejecución de pruebas

JUnit sigue un modelo de ejecución de pruebas basado en clases, donde los métodos de prueba se definen dentro de las clases de prueba.

Por otro lado, Pytest sigue un modelo de ejecución de pruebas basado en funciones, donde las pruebas se organizan como funciones independientes.

* Informes de pruebas

Junit proporciona una salida que podría no ofrecer mensajes de error tan simples para los principiantes, lo que puede dificultar el diagnóstico de problemas cuando las pruebas fallan.

Pytest proporciona resultados detallados e informativos durante la ejecución de las pruebas, destacando los resultados individuales y cualquier fallo.

* Descubrimiento de pruebas

JUnit requiere que los desarrolladores definan y especifiquen explícitamente las clases y métodos de prueba.

Junit, puede implicar más configuración que Pytest, ya que, también admite descubrimiento de pruebas, pero requiere que los métodos tengan encima la anotación @Test y definidos dentro de clases públicas. Requiere una estructura clara y una configuración correcta.

Por el contrario, Pytest ofrece descubrimiento automático de pruebas, donde las pruebas se encuentran y ejecutan automáticamente según ciertas convenciones de nomenclatura y estructuras de directorios.

Pytest detecta automáticamente funciones, clases y métodos de prueba en el directorio del proyecto y sus subdirectorios, basándose en convenciones de nomenclatura (por ejemplo, nombres que comienzan con test\_). No requiere configuración explícita, lo que simplifica su uso. Además, permite aplicar filtros para ejecutar subconjuntos de pruebas.

* Ecosistema de plugins

JUnit cuenta con un ecosistema amplio y consolidado de plugins que ofrecen características y funcionalidades adicionales.

Puede extenderse usando bibliotecas externas como AssertJ, Mockito o integrarse con marcos como Spring Test (que permite el uso de bases de datos embebidas y el mejor uso de pruebas).

Sin embargo, su configuración tiende a ser más explícita y menos dinámica que la de Pytest, lo cual puede percibirse como más rígido para ciertos escenarios avanzados o específicos.

Por el contrario, Pytest cuenta con un sistema de plugins más flexible y extensible que permite a los desarrolladores crear plugins personalizados e integrarlos fácilmente con otras herramientas y frameworks.

Pytest es altamente extensible mediante una gran variedad de complementos (plugins) oficiales y de terceros, lo que permite adaptar el framework a necesidades muy específicas.

* Comunidad

JUnit tiene una comunidad madura, consolidada y bien soportada, siendo uno de los frameworks de pruebas más antiguos en el ecosistema Java.

Esto garantiza alta estabilidad, consistencia y compatibilidad, lo que es muy valorado en entornos empresariales grandes y sistemas heredados. Además, el hecho de tener pocas dependencias externas lo hace menos vulnerable a incompatibilidades por terceros.

Pytest cuenta con una comunidad muy activa y en crecimiento, especialmente dentro del ecosistema Python. Es ampliamente adoptado tanto en proyectos empresariales como de código abierto, lo que lo convierte en una herramienta poderosa y adaptable.

Sin embargo, esta amplia dependencia de plugins de terceros también puede generar desafíos de compatibilidad, especialmente cuando hay cambios en versiones mayores o conflictos entre extensiones.

* Documentación

JUnit cuenta con una documentación oficial sólida y completa, pero más formal y detallada, para desarrolladores experimentados. Su integración con los entornos de desarrollo más populares (como Eclipse y NetBeans) permite escribir, ejecutar y depurar pruebas de forma directa, lo que mejora significativamente la productividad y facilita el aprendizaje.

Además, al ser un estándar ampliamente adoptado en el ecosistema Java, existen numerosos recursos adicionales, como libros, tutoriales y comunidades activas.

Pytest cuenta con una documentación oficial extensa muy clara, concisa y orientada a ejemplos prácticos, ideal para aprender rápido y comenzar a usarlo sin mucha configuración. Además, para temas avanzados o casos de uso específicos la comunidad de Pytest genera muchos tutoriales, foros, blogs o ejemplos prácticos en videos que complementan la documentación oficial.

* Casos de uso

Junit se adapta a proyectos grandes que requieren dependencias solo de bibliotecas estándar y con un enfoque conservador donde puede ser preferible seguir el marco estándar Junit para mantener la uniformidad y la coherencia en todo el código base.

Conveniente si tu prioridad es la estabilidad, la precisión y el mantenimiento a largo plazo.

Pytest es una excelente opción si desea un Framework simple y poderoso que fomente el desarrollo rápido de pruebas y proporcione funciones útiles listas para usar.

Es especialmente útil para quienes quieren comenzar a hacer pruebas con una configuración sencilla, donde el volumen de datos es alto, pero la lógica de negocio es más cambiante o experimental.

Entonces, es conveniente, si tu prioridad es escalar pruebas de forma ágil y automatizada.