SUSTAINABILITY DATA SPACES

E5

Código técnico del modelo

E5 publica la implementación técnica del modelo SDS: ingesta controlada de datos de estándares, validación del registro canónico, generación del modelo semántico, publicación de API, cálculos, conversión de unidades y datos de referencia, gestión de valores, controles de mapeo y ganchos de gobernanza operativa.

ENTREGABLE E5	**VERSIÓN** V2026-06-09	**FECHA** 2026-06-09
PROYECTO Sustainability Data Spaces	**RESPONSABLE** Oficina del Programa SDS	**FORMATO** A4 · OOXML

Tabla de contenido

Contenido del entregable

Resumen y control de validacion de cierre E5 3

Anclajes del codigo tecnico 4

Modelo de mapeo, API y seguridad 5

Prueba funcional 6

Transformacion, exportacion y superficie de verificacion 7

Utilizacion de cobertura y capacidades adicionales 8

Estado de cumplimiento y referencias 9

Financiado por la Unión Europea
NextGenerationEU
MINISTERIO PARA LA TRANSFORMACIÓN DIGITAL Y DE LA FUNCIÓN PÚBLICA
SECRETARÍA DE ESTADO DE DIGITALIZACIÓN E INTELIGENCIA ARTIFICIAL
Plan de Recuperación, Transformación y Resiliencia

Tabla 1 Ficha del entregable E5

CAMPO	VALOR
Proyecto	Sustainability Data Spaces (SDS)
Entregable	E5 - Código técnico del modelo
Paquete de trabajo	PT2 - Modelado de Datos e Interoperabilidad Técnica
Actividad principal	A2.3 - Creación de esquemas y pipelines conceptuales para transformar datos brutos en reportes normativos interoperables
Versión	V1.0
Fecha	2026-06-18
Estado	Documento final
Responsable	Oficina del Programa SDS

Resumen

E5 publica la implementación técnica del modelo SDS: ingesta controlada de datos de estándares, validación del registro canónico, generación del modelo semántico, publicación de API, cálculos, conversión de unidades y datos de referencia, gestión de valores, controles de mapeo y ganchos de gobernanza operativa. La implementación sigue los contratos del modelo SDS utilizados por E1, E2, E3, E4 y E6, y los expone mediante código ejecutable en lugar de evidencia solo documental.[1] [2] [3]

El código técnico se organiza en torno a cuatro funciones de producto:

- Cargar estándares, mapeos, unidades, datos de referencia y valores en almacenes SDS controlados.
- Transformar estándares y entradas operativas en las formas canónicas de registro, semántica, cálculo, conversión y valores de SDS.
- Exportar y servir salidas del modelo mediante superficies de API, semánticas y de evidencia.
- Verificar la implementación mediante controles de validación de entregables, pruebas de integración, flujos de cobertura y comprobaciones de entorno de ejecución DB-first.

[1] European Telecommunications Standards Institute, *ETSI GS CIM 009: NGSI-LD API*, 2024, https://cim.etsi.org/NGSI-LD/official/front-page.html.

[2] Sporny, Manu et al., *JSON-LD 1.1*, W3C Recommendation, 2020, https://www.w3.org/TR/json-ld11/.

[3] JSON Schema, *JSON Schema Draft 2020-12*, n.d., https://json-schema.org/draft/2020-12.

Control de validación de cierre requerido para E5

El control de validación de cierre del proyecto para E5 requiere código funcional de carga, transformación y exportación; evidencia de cobertura de pruebas unitarias; pruebas de integración; y documentación/ejemplos.

Tabla 2 Control de validación de cierre requerido para E5

ELEMENTO REQUERIDO	IMPLEMENTACIÓN E5
Carga funcional	El código de importación y arranque carga indicadores, metadatos de versiones de estándares, mapeos, unidades, datos de referencia, valores y registros de muestra controlados mediante scripts y servicios de API.
Transformación funcional	El código de registro, semántica, mapeo, cálculo, conversión y versionado de valores normaliza las entradas en contratos SDS antes de la publicación.
Exportación funcional	Los routers de API, las utilidades de exportación NGSI-LD/semántica, la serialización firmada de conjuntos de datos y los constructores deterministas de evidencia exponen las salidas del modelo.
Cobertura de pruebas unitarias	El flujo de cobertura de API se ejecuta con api/scripts/dev.ps1 TestCoverage, make service-coverage y el espejo de CI nativo de fuente make api-ci-local. El flujo impone --cov-fail-under=97.01. La ejecución completa de 2026-07-10 pasó con 2.682 pruebas aprobadas y 57 omitidas; cubrió 21.714 de 22.303 statements, dejó 589 sin cubrir y obtuvo una cobertura exacta de 97.35909967268977%, por encima del umbral de aceptación E5 de >=90% y del objetivo de calidad actual de >97%.
Pruebas de integración	Los controles de validación de indicadores, mapeo, semántica, cálculo, conversión, valores, DB-first y entregables ejercitan la implementación a través de los límites entre módulos.
Documentación/ejemplos	La documentación raíz/API, los índices de entregables, la documentación de API y los objetivos ejecutables del Makefile documentan cómo se ejecuta y valida el modelo técnico.

Anclajes del código técnico

Tabla 3 Anclajes del código técnico

ÁREA DEL MODELO	ANCLAJES PRINCIPALES DE IMPLEMENTACIÓN	QUÉ DEMUESTRA EL CÓDIGO
Registro y límite de paquete	scripts/prepare_register_for_import.py, semantics/schema/register_schema.json, semantics/shacl/shapes_register_shacl.ttl	Las filas del paquete de estándares se normalizan en el contrato de registro SDS con validación de esquema y SHACL.
Metadatos de versiones de estándares	api/src/services/standard_versioning_import.py, api/scripts/import_standard_versioning.py, api/src/database/models.py	Las versiones de estándares y el linaje de puntos de datos pueden almacenarse como metadatos versionados antes de la publicación en el entorno de ejecución.

ÁREA DEL MODELO	ANCLAJES PRINCIPALES DE IMPLEMENTACIÓN	QUÉ DEMUESTRA EL CÓDIGO
Catálogo de indicadores	api/src/services/indicator_import.py, api/src/services/indicator_store.py, api/src/api/routers/indicators.py	Los indicadores pueden importarse, consultarse, buscarse, exportarse y trazarse mediante superficies de historial/cambio de conjuntos de datos.
Modelo semántico	api/src/services/concept_service.py, api/src/services/ontology_service.py, api/src/ontology/projection_generator.py, semantics/context/, semantics/shacl/	Los conceptos, relaciones, contexto JSON-LD, restricciones SHACL y la proyección ontológica generada son ejecutables.[4] [5]
Modelo de mapeo	api/src/services/canonical_mapping_package.py, api/src/services/canonical_pairwise_materialization.py, api/src/api/routers/mappings.py, api/src/api/routers/mapping_assertions.py	Las relaciones revisadas pueden validarse, materializarse, consultarse y mantenerse separadas de candidatos no revisados.
Contratos de cálculo	api/src/calculation/contracts.py, api/src/calculation/engine.py, api/src/services/calculation_contract_import.py, api/src/api/routers/calculations.py	Las entradas de fórmula, dependencias de cálculo, política de conversión, salida de traza y contratos de cálculo ejecutables se representan en código.
Unidades y conversiones	api/src/calculation/conversion/, api/src/calculation/unit_converter.py, api/src/services/fx_service.py, api/src/database/bootstrap_units.py, api/src/database/bootstrap_conversion_catalog.py, api/src/api/routers/units.py, api/src/api/routers/fx.py	La conversión de unidades físicas, las expresiones compuestas, la política de FX de series temporales, el arranque del catálogo de unidades y las trazas de conversión son deterministas y comprobables.
Valores y revisiones	api/src/services/value_ingest.py, api/src/services/value_csv_import.py, api/src/services/value_revision_store.py, api/src/services/value_versioning.py, api/src/api/routers/values.py	Las importaciones de valores operativos, los contratos estrictos de valores, los metadatos de revisión, el linaje y las superficies de lectura se implementan sin confluir valores con metadatos estructurales de estándares.
Interoperabilidad en el entorno de ejecución	api/src/services/value_resolution.py, api/src/services/runtime_readiness.py, api/src/api/routers/interoperability.py, api/src/api/routers/values.py, api/src/api/openapi_docs.py	La resolución de valores en el entorno de ejecución puede usar valores directos, contratos de cálculo ejecutables, mapeos exactos/equivalentes, conversión de unidades compatible y semántica de rechazo de fallo cerrado.
API y seguridad en el entorno de ejecución	api/src/api/main.py, api/src/api/routers/auth.py, api/src/api/routers/ontology.py, api/src/api/routers/hierarchies.py, api/src/config/settings.py	El modelo se expone mediante FastAPI con autenticación, RBAC, configuración DB-first, health/preparación y routers modulares.
Exportación de	api/src/services/dataset_serialization.py,	Las salidas de conjuntos de datos

[4] Knublauch, Holger and Kontokostas, Dimitris, *Shapes Constraint Language (SHACL)*, W3C Recommendation, 2017, https://www.w3.org/TR/shacl/.

[5] W3C OWL Working Group, *OWL 2 Web Ontology Language Document Overview*, Second edition, W3C Recommendation, 2012, https://www.w3.org/TR/owl2-overview/.

ÁREA DEL MODELO	ANCLAJES PRINCIPALES DE IMPLEMENTACIÓN	QUÉ DEMUESTRA EL CÓDIGO
conjuntos de datos y procedencia	api/src/services/dataset_manifest.py, api/src/services/dataset_history.py, api/src/services/export_signing.py, packages/sds_core/	pueden incluir metadatos de manifiesto, historial, firma, serialización y alineación NGSI-LD/DCAT.[6]

Prueba funcional

Carga

La base de código SDS carga el modelo técnico mediante almacenes y scripts controlados:

- Los servicios de importación de indicadores y versiones de estándares validan la forma de origen antes de la persistencia.
- Los servicios de mapeo validan la compatibilidad de estándares instalados antes de la materialización.
- Los rutinas de arranque de unidades y conversión siembran datos de referencia deterministas en el almacén del entorno de ejecución.
- Los servicios de importación de valores imponen contratos estrictos de valores y mantienen los valores operativos separados de la evidencia de estándares/catálogos.
- La configuración de entorno de ejecución DB-first evita recurrir silenciosamente a catálogos empaquetados cuando el modo de producción requiere PostgreSQL.

Transformación

La capa de transformación del modelo convierte la evidencia de entrada en estructuras SDS ejecutables:

- Las filas de registro se comprueban contra restricciones de esquema y semánticas.
- Los identificadores se canonicalizan antes de la publicación del catálogo.
- La evidencia de relaciones se tipa y se materializa solo después de revisión.
- Los contratos de cálculo vinculan entradas de fórmula, unidades, política de conversión y metadatos de traza.
- Los flujos de conversión de unidades físicas y FX producen pasos de traza deterministas.
- Las revisiones de valores preservan resúmenes resumen hash de contexto, eventos y punteros current/reported.

Exportación

El modelo se expone mediante superficies de producto y evidencia:

[6] European Commission, SEMIC, *DCAT-AP 3.0.0*, 2024, https://semiceu.github.io/DCAT-AP/releases/3.0.0/.

- Los routers de API publican indicadores, conceptos, mapeos, cálculos, unidades, FX/datos de referencia, valores, jerarquías, auth y health/preparación.
- Los puntos de conexión de interoperabilidad en el entorno de ejecución publican comprobaciones de preparación y resolución de valores para uso manual y programático. El resolver se basa en evidencia: devuelve valores resueltos solo cuando el entorno de ejecución activo tiene un valor almacenado, contrato de cálculo ejecutable, mapeo exacto/equivalente o conversión de unidades compatible. Las rutas no soportadas o con evidencia insuficiente devuelven estados estructurados de rechazo en lugar de valores fabricados.
- Los artefactos alineados con NGSI-LD, JSON-LD, SHACL, JSON Schema y DCAT soportan el intercambio semántico y de conjuntos de datos.[7] [8] [9] [10]
- Los servicios de serialización de conjuntos de datos, manifiestos, historial y firma soportan evidencia de exportación reproducible.

Superficie de verificación

E5 se valida mediante una superficie estratificada de pruebas y controles de validación:

Tabla 4 Superficie de verificación

CAPA DE VERIFICACIÓN	COMANDO O EVIDENCIA
Control de validación de paquete público E5	make e5-gate
Control de validación de registro de entregables, suma de comprobación y privacidad	make deliverables-check
Capa de cierre de repositorio	python scripts/repo_closure_check.py
Control de validación de conversión y datos de referencia	make conversion-gate
Control de validación de contrato semántico	make gate-semantics
Control de validación de producción DB-first	make service-wave5
Suite de pruebas de API	api/scripts/dev.ps1 Test o make service-test
Flujo de cobertura	api/scripts/dev.ps1 TestCoverage, make service-coverage o make api-ci-local

Última evidencia de cobertura:

[7] European Telecommunications Standards Institute, *ETSI GS CIM 009: NGSI-LD API*.

[8] Sporny, Manu et al., *JSON-LD 1.1*.

[9] Knublauch, Holger and Kontokostas, Dimitris, *Shapes Constraint Language (SHACL)*.

[10] European Commission, SEMIC, *DCAT-AP 3.0.0*.

- **make api-ci-local se completó correctamente el 2026-07-10 con 2.682 pruebas aprobadas, 57 pruebas omitidas y una línea de calidad impuesta --cov-fail-under=97.01.**
- **La misma ejecución reportó una cobertura TOTAL del 97%, con una cobertura exacta medida de 97.35909967268977% (21.714 statements cubiertos de 22.303, con 589 statements no cubiertos).**
- El control de validación de aceptación E5 documentado requiere una cobertura de pruebas unitarias >=90%; ese umbral se supera sin cambiar el control de validación formal, y el objetivo de calidad actual de cobertura exacta >97% se cumple mediante el comando ejecutable de cobertura.
- La evidencia no afirma una cobertura del 100%; los statements actualmente no cubiertos no se están persiguiendo con pruebas artificiales ni pragmas no-cover.

Por tanto, este documento registra el paquete de código ejecutable y la superficie de verificación actual como evidencia de que se cumple el umbral de cobertura E5 documentado.

Capacidades SDS adicionales más allá del alcance mínimo de E5

El control de validación E5 original requiere código de carga/transformación/exportación, evidencia de cobertura, pruebas de integración y documentación/ejemplos. SDS incluye capacidades técnicas adicionales más allá de ese mínimo:

Tabla 5 Capacidades SDS adicionales más allá del alcance mínimo de E5

CAPACIDAD ADICIONAL	POR QUÉ SUPERA EL CONTROL DE VALIDACIÓN E5 MÍNIMO
Metadatos versionados de versiones de estándares	El modelo puede preservar la identidad de versión y el linaje de puntos de datos en lugar de tratar todos los estándares como un catálogo actual plano.
Materialización de mapeos revisados	La publicación de relaciones se controla mediante evidencia revisada y comprobaciones de compatibilidad de estándares instalados.
Conversión determinista de unidades físicas y FX de series temporales	La conversión de unidades y divisas se modela como servicios del entorno de ejecución trazables, no como simple normalización de etiquetas.
Contratos de cálculo con política de conversión	La ejecución de fórmulas puede vincular entradas, unidades, comportamiento de conversión y salida de traza.
Resolución de valores en el entorno de ejecución entre estándares	La API puede resolver valores operativos mediante almacenamiento directo, contratos de cálculo, mapeos exactos/equivalentes, conversión de unidades compatible y semántica explícita de rechazo.
Punto de conexión de preparación de interoperabilidad	Los operadores pueden verificar las superficies del entorno de ejecución cargadas antes de ejecutar comprobaciones manuales de interoperabilidad en Swagger.
Superficies de revisión y linaje de valores	Los valores operativos pueden llevar eventos de revisión, resúmenes resumen hash de contexto, punteros current y punteros reported.
Modo de entorno de ejecución DB-first sin fallback	El comportamiento de producción puede fallar de forma cerrada cuando los datos del modelo canónico respaldados por PostgreSQL no están disponibles.
Servicios de firma, manifiesto e	La evidencia de exportación puede incluir metadatos de procedencia y

CAPACIDAD ADICIONAL	POR QUÉ SUPERA EL CONTROL DE VALIDACIÓN E5 MÍNIMO
historial de conjuntos de datos	reproducibilidad más allá de un listado básico de código.

Estas capacidades forman parte del producto técnico SDS actual. Deben describirse como profundidad técnica añadida más allá del control de validación de cierre E5 original, no como obligaciones adicionales dentro del alcance original del proyecto.

Estado de cumplimiento

E5 está implementado y publicado como el paquete de código técnico para el modelo SDS. La base de código implementa las capas de carga, transformación, exportación, semántica, mapeo, cálculo, conversión, valores y API del entorno de ejecución del modelo, y proporciona controles de validación y pruebas ejecutables para revisar la implementación técnica.

La aceptación formal del control de validación E5 está cerrada frente al umbral documentado de cobertura de pruebas unitarias >=90%. La evidencia actual no afirma una cobertura del 100%; registra una cobertura exacta medida de 97.35909967268977% y el cierre del control de validación de aceptación E5 definido.

Referencias

- European Telecommunications Standards Institute. ETSI GS CIM 009, Context Information Management (CIM); NGSI-LD API. https://cim.etsi.org/NGSI-LD/official/front-page.html.[11]
- Sporny, Manu, Dave Longley, Gregg Kellogg, Markus Lanthaler, Pierre-Antoine Champin, and Niklas Lindstrom. JSON-LD 1.1. W3C Recommendation, July 16, 2020. https://www.w3.org/TR/json-ld11/.[12]
- JSON Schema. JSON Schema Draft 2020-12. https://json-schema.org/draft/2020-12.[13]
- Knublauch, Holger, and Dimitris Kontokostas. Shapes Constraint Language (SHACL). W3C Recommendation, July 20, 2017. https://www.w3.org/TR/shacl/.[14]
- W3C OWL Working Group. OWL 2 Web Ontology Language Document Overview, second edition. W3C Recommendation, December 11, 2012. https://www.w3.org/TR/owl2-overview/.[15]
- SEMIC. DCAT Application Profile for data portals in Europe. https://semiceu.github.io/DCAT-AP/.[16]

[11] European Telecommunications Standards Institute, *ETSI GS CIM 009: NGSI-LD API*.

[12] Sporny, Manu et al., *JSON-LD 1.1*.

[13] JSON Schema, *JSON Schema Draft 2020-12*.

[14] Knublauch, Holger and Kontokostas, Dimitris, *Shapes Constraint Language (SHACL)*.

[15] W3C OWL Working Group, *OWL 2 Web Ontology Language Document Overview*.

[16] European Commission, SEMIC, *DCAT-AP 3.0.0*.

European Commission, SEMIC. *DCAT-AP 3.0.0*. 2024. https://semiceu.github.io/DCAT-AP/releases/3.0.0/.

European Telecommunications Standards Institute. *ETSI GS CIM 009: NGSI-LD API*. 2024. https://cim.etsi.org/NGSI-LD/official/front-page.html.

JSON Schema. *JSON Schema Draft 2020-12*. n.d. https://json-schema.org/draft/2020-12.

Knublauch, Holger, and Kontokostas, Dimitris. *Shapes Constraint Language (SHACL)*. W3C Recommendation, 2017. https://www.w3.org/TR/shacl/.

Sporny, Manu, Longley, Dave, Kellogg, Gregg, Lanthaler, Markus, Champin, Pierre-Antoine, and Lindstrom, Niklas. *JSON-LD 1.1*. W3C Recommendation, 2020. https://www.w3.org/TR/json-ld11/.

W3C OWL Working Group. *OWL 2 Web Ontology Language Document Overview*. Second edition. W3C Recommendation, 2012. https://www.w3.org/TR/owl2-overview/.